Unsupervised Losses for Clustering and Segmentation of Images: Theories & Optimization Algorithms

by

Zhongwen Zhang

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2024

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Dr. Dale Schuurmans Professor, Department of Computing Science University of Alberta
Supervisor(s):	Dr. Yuri Boykov Professor, School of Computer Science University of Waterloo
Internal Members:	Dr. Olga Veksler Professor, School of Computer Science University of Waterloo
	Dr. Yaoliang Yu Associate Professor, School of Computer Science University of Waterloo
Internal-External Member:	Dr. Paul Fieguth Professor, Department of Systems Design Engineering University of Waterloo

Author's Declaration

Statement of Contributions

I hereby declare that I am the sole author of Chapter 1 and 6.

Chapter 2 is based on part of the manuscript co-authored with Yuri Boykov. I am the first author and main contributor. Yuri Boykov provided the supervision and contributed to the design and analysis of the methods and experiments. I conducted all the experiments.

Chapter 3 is based on a manuscript co-authored with Yuri Boykov. All authors contributed equally to the paper.

Chapter 4 is based on a manuscript co-authored with Yuri Boykov. I am the first author and main contributor of the paper. Yuri Boykov provided the supervision and contributed to the design and analysis of the methods and experiments. I conducted all the experiments.

Chapter 5 based on published material [233] co-authored with Dmitrii Marin, Maria Drangova, and Yuri Boykov. Maria Drangova provided the data of high-resolution micro-CT images while the other two authors provided supervision and contributed to the design and analysis of the methods and experiments. I am the first author and the main contributor of [233] and conducted all the experiments.

Abstract

Unsupervised losses are common for tasks with limited human annotations. In clustering, they are used to group data without any labels. In semi-supervised or weakly-supervised learning, they are applied to the unannotated part of the training data. In self-supervised settings, they are used for representation learning. They appear in diverse forms enforcing different prior knowledge. However, formulating and optimizing such losses poses challenges. Firstly, translating prior knowledge into mathematical formulations can be non-trivial. Secondly, the properties of standard losses may not be obvious across different tasks. Thirdly, standard optimization algorithms may not work effectively or efficiently, thus requiring the development of customized algorithms.

This thesis addresses several related classification and segmentation problems in computer vision, using unsupervised image- or pixel-level losses under a shortage of labels. First, we focus on the entropy-based *decisiveness* as a standard unsupervised loss for softmax models. While discussing it in the context of clustering, we prove that it leads to margin maximization, typically associated with supervised learning. In the context of weakly-supervised semantic segmentation, we combine decisiveness with the standard pairwise regularizer, the Potts model. We study the conceptual and empirical properties of different relaxations of the latter. For both clustering and segmentation problems, we provide new self-labeling optimization algorithms for the corresponding unsupervised losses. Unlike related prior work, we use *soft* hidden labels that can represent the estimated class uncertainty. Training network models with such soft pseudo-labels motivates a new form of cross-entropy maximizing the probability of “collision” between the predicted and estimated classes. The proposed losses and algorithms achieve the state-of-the-art on standard benchmarks. The thesis also introduces new geometrically motivated unsupervised losses for estimating thin structures, e.g. complex vasculature trees at near-capillary resolution in 3D medical data.

Acknowledgments

I would like to express my deepest gratitude to Yuri Boykov, my supervisor, for his endless support, guidance, and invaluable mentorship throughout the entirety of my doctoral journey. He always squeezed his time to have fruitful discussions with me when I was stuck. He showed me the correct way to do research and his passion for the research. I am thankful to him for helping me shape my good taste in scientific research. His philosophy towards simplicity now become my philosophy of doing research. He also taught me not to give up even in the darkest hours. He encouraged me a lot when the papers got rejected many times and his recognition of the work I have done gave me confidence about myself. I am also grateful for his caring and financial support during this long journey.

I would like to thank my committee members, Prof. Dale Schuurmans, Prof. Olga Veksler, Porf. Yaoliang Yu and Prof. Alexander Wong for their valuable and insightful feedback.

I would like to thank my coauthors Dmitrii Marin and Xingye Fan. They are also my best friends. I enjoyed all the good times and interesting discussions during the hiking in the spectacular Rocky Mountains with Dmitrii. Thanks to him for giving me both advice for the research and the camping skills. I also enjoyed all the good times and interesting discussions on lots of different topics with Xingye.

I also thank my colleagues, especially Dr. Juwei Lu, Dr. XinXin Zuo, and Dr. Chuan Guo, during my internship at Huawei. They helped widen my horizons and helped me better understand lots of new concepts in a different field than my expertise. I am grateful to Dr. Juwei Lu for encouraging me to be more open to different types of research topics. They gave me lots of support and offered me an opportunity to continue working there.

I thank all my friends, especially Xiaoyuan Hou, Xingye Fan, Xiyang Feng, Mengqi Lu, Kunxun Qian, Dmitrii Marin and Zhuowei Zhang, who showed me caring and gave me support and suggestions during the time I felt down and sad. They are the kindest, and the most lovely people I have ever met, and I am lucky to have them in my world.

I thank my parents and grandparents who supported me all along the way. They showed me what love is and are always there for me. They always tell me to follow my own heart and choose for myself. I am very lucky to have them as my family.

I thank Canada for allowing me to study on this beautiful continent.

Dedication

This is dedicated to my parents and grandparents for their endless love and suppport.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction: annotation shortage in vision

Computer vision aims to enable computers to understand visual content like human beings. Data-driven methods have proven to be effective for both understanding [58, 182, 134, 90, 147] and generation tasks [172, 177]. However, the amount of data needed is huge. While the generation tasks do not require human annotations, understanding tasks do. For example, the ImageNet dataset [182] for image classification contains nearly 15 million images with labels, which requires massive manual work. While simply labeling images may still sound manageable to some people, consider more complex forms of annotations, such as precise pixel-wise masks for segmentation tasks [66, 131, 49]. To alleviate the annotation issue, this thesis is focused on the study of the unsupervised losses with corresponding optimization algorithms in three different problems. We will start from image-level classification in Section 1.1 and then move on to the pixel-level classification, i.e. segmentation, for natural images in Section 1.2 and for medical images of vessels in Section 1.3. In Section 1.4, we summarize the issues and introduce our contributions.

1.1 Image classification

Image classification is a fundamental task in computer vision aimed at automatically categorizing images into predefined classes or categories. It is important for various problems such as facial recognition [167, 223], image segmentation [134, 147], medical image analysis [132], autonomous driving [126], etc. The classification task is formulated as a mapping function $h : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ is the input space of images and $\mathcal{Y}$ is the set of all possible class labels. We often decompose it into two parts. One maps the image to the feature space, $f : \mathcal{X} \to \mathcal{F}$. The other is a classifier function, $g : \mathcal{F} \to \mathcal{Y}$, which is usually chosen to be as simple as possible, say a linear classifier.

Hence, the label for a given image x is given by

$$h(x) \equiv g(f(x)) \tag{1.1}$$

1.1.1 Supervised classification

To solve the classification problem, one can either explicitly construct the mapping functions or search for the functions from some hypothesis space. If we learn the oracle behind a specific classification task by carefully looking at samples and summarizing the rules, we can give an explicit formulation for h. However, this does not usually work well in the real world. Just consider the image classification task. It is not easy to understand why we make the decision.

Learning-based methods employ algorithms to explore the hypothesis space in search of a model that fits the data. It's important to note that the hypothesis space is still manually defined, often guided by the principle of Occam's razor to ensure good generalization. Examples of classification methods include KNN [68], decision tree [195], logistic regression [95], support vector machine (SVM) [50], and neural networks [124]. These methods can be applied to the raw input directly or to some fixed features. The methods to extract features usually refer to hand-engineered methods such as HOG [54], Harris Corner Detector [87], SIFT [136] and SURF [11].

Among the classification methods mentioned above, neural networks stand out as a very powerful tool, probably due to their hierarchical structure to better extract the features. Also, they can approximate any functions [94] and are scalable. The progression of neural network architectures has advanced from basic Multi-Layer Perceptrons (MLP) to Convolutional Neural Networks (CNN) such as VGG [192], ResNet [90], MobileNet [96], and more recently, transformers like ViT [62]. Typically it consists of many layers in a sequential order and is highly non-linear. In classification tasks, the final layer is often a linear classifier. The linear model is a special case when $f(X) = X$. Typical "deep" representation $f(X)$ significantly changes the dimensions of the input X. It is convenient to assume that M always represents the dimensions of the linear head input so that (homogeneous) matrix $\mathbf{v} \in \mathcal{R}^{(M+1)\times K}$ where K is the number of classes. To optimize these parameters, loss functions and corresponding optimization algorithms are needed. As follows, we first review some losses that have maximum margin property. In Chapter 3, we will extend such property to a family of *unsupervised* losses.

Margin maximization losses Here we will review two important types of classification losses, i.e. hinge loss and cross-entropy loss. For simplicity, we focus on the binary case while the multi-class extension will be given at the end. We also assume the features are fixed and linearly separable so that the maximum margin can be well-defined.

The minimization problem using **hinge loss** is defined as:

$$\min_{\mathbf{v}} \frac{1}{N} \sum_{i} \max(0, 1 - y_i \cdot \mathbf{v}^\top f(X_i)) \tag{1.2}$$

where i is the index of the sample, N is the total number of samples in the training set, $f(X_i) \in \mathcal{R}^M$ is the feature, and $y_i \in \{-1, 1\}$ is the ground-truth label. For shortness, $\mathbf{v}^\top$ includes the bias assuming the "homogeneous" representation of $f(X_i)$. To minimize (1.2), $\max(0, 1-y_i \cdot \mathbf{v}^\top f(X_i))$ should be zero for all samples due to linear separability assumed above. However, the minimum is not unique, and actually, there is an infinite number of equivalent minima. Squared L_2 norm is a common prior/regularization, and the addition of such regularization induces a unique solution with

$$\min_{\mathbf{v}} \frac{1}{N} \sum_{i} \max(0, 1 - y_i \cdot \mathbf{v}^\top f(X_i)) + \lambda \|\mathbf{v}\|_2^2 \tag{1.3}$$

an intriguing geometric property, i.e. maximum margin [217]. Note that $\frac{2}{\|\mathbf{v}\|_2}$ (excluding the bias in the norm!) is the margin width for the formulation in (1.3). Consistent with the intuition of generalization, such margin maximization makes the decision boundary as far from the *nearest* samples on both sides as possible. In (1.3), a small λ will induce the maximum margin solution. Value of λ larger than a certain threshold will make $\mathbf{v}$ too small so that $1 - y_i \cdot \mathbf{v}^\top f(X_i)$ can be larger than 0 for some X_i. Then the margin is violated. As for the non-linearly separable case, λ controls the level of trade-off between margin violation and its width.

Logistic regression is also known for maximum margin property [179]. It uses cross-entropy loss on top of the sigmoid operator which maps the logits $(-\mathbf{v}^\top f(X_i),\ \mathbf{v}^\top f_{\mathbf{w}}(X_i))$ to 2-class probabilities forming a categorial distribution $(\frac{1}{1+e^{\mathbf{v}^\top f_{\mathbf{w}}(X_i)}},\ \frac{1}{1+e^{-\mathbf{v}^\top f_{\mathbf{w}}(X_i)}})$ often interpreted as a posterior. Note that we should also turn the y_i into a distribution, e.g. $([y_i = -1],\ [y_i = 1])$ where $[\ \cdot\ ]$ is the Iverson bracket. Then the minimization problem using the cross-entropy is defined as:

$$\min_{\mathbf{v}} \frac{1}{N} \sum_{i} -[y_i = -1] \ln \frac{1}{1 + e^{\mathbf{v}^\top f(X_i)}} - [y_i = 1] \ln \frac{1}{1 + e^{-\mathbf{v}^\top f(X_i)}} \tag{1.4}$$

The cross-entropy loss (1.4) is differentiable and thus amenable to standard gradient descent. Also, a nice property of using cross-entropy loss on top of the sigmoid is the convexity. In many places, such loss is also interpreted as negative log-likelihood (NLL) although NLL is usually reserved for the density estimation with generative modeling. Minimization of this loss will not only lead to an infinite number of solutions but also result in an infinitely large norm of $\mathbf{v}$. Therefore, the squared L_2 regularization is also needed[1].

$$\min_{\mathbf{v}} \frac{1}{N} \sum_i -[y_i = -1] \ln \frac{1}{1 + e^{\mathbf{v}^\top f(X_i)}} - [y_i = 1] \ln \frac{1}{1 + e^{-\mathbf{v}^\top f(X_i)}} + \lambda \|\mathbf{v}\|_2^2 \tag{1.5}$$

Different from (1.3), the λ in (1.5) needs to be infinitesimally small to induce the margin maximization phenomenon. Intuitively, when λ approaches the zero, the norm of $\mathbf{v}$ goes to infinity and $-\ln \frac{1}{1+e^{-y_i \cdot \mathbf{v}^\top f(X_i)}} = \ln(1 + e^{-y_i \cdot \mathbf{v}^\top f(X_i)})$ will vanish quickly enough to concentrate on the samples nearest to the decision boundary. More rigid mathematical analysis can be found in [179].

We also give the formulations for both the hinge loss (1.6) and the cross-entropy loss (1.7) under multi-class cases where $y_i \in \{1, 2, ..., K\}$. Interested readers can refer to [179] for a detailed description of the margin maximization in multi-class classification.

$$\min_{\mathbf{v}} \frac{1}{N} \sum_i \sum_{j \neq y_i} \max(0, 1 - (\mathbf{v}_{y_i} - \mathbf{v}_j)^\top f(X_i)) + \lambda \|\mathbf{v}\|_F^2 \tag{1.6}$$

The $\|\cdot\|_F$ is the Frobenius norm. To define the multi-class cross-entropy loss, we need to extend the sigmoid to the softmax operator, i.e. $\sigma : \mathcal{R}^K \to \Delta^K$, mapping K classifier outputs $l^k := (\mathbf{v}^\top f(X_i))^k \in \mathcal{R}$ to K-class probabilities $\sigma^k = \frac{\exp l^k}{\sum_c \exp l^c}$ forming a categorical distribution $\sigma = (\sigma^1, \ldots, \sigma^K) \in \Delta^K$ where

$$\Delta^K := \{(p^1, \ldots, p^K) \in \mathcal{R}^K \,|\, p^k \geq 0, \sum p^k = 1\}.$$

Then the minimization problem with multi-class cross-entropy loss can be defined as:

$$\min_{\mathbf{v}} \frac{1}{N} \sum_i \sum_k -[y_i = k] \ln \frac{e^{\mathbf{v}_{y_i}^\top f(X_i)}}{\sum_c e^{\mathbf{v}_c^\top f(X_i)}} + \lambda \|\mathbf{v}\|_F^2 \tag{1.7}$$

1.1.2 Towards unsupervised classification

Learning-based methods often demand substantial labeled data to perform effectively in the real world. The pursuit of larger datasets aims to improve the generalization. However, the labeling process is costly, let alone datasets surpassing millions of images. Even with parallel labeling efforts from many people, ensuring consistency across annotators remains challenging.

[1]Certain optimization algorithms such as gradient descent can lead to maximum margin [196] without explicit regularization.

Consequently, this motivates unsupervised approaches for the unlabelled data. Methods that employ both labeled and unlabelled data try to combine the best aspects of both worlds.

The rationale behind unsupervised learning lies in its capacity to reduce economic costs while improving generalization by using extensive data. However, unlike supervised learning, which has a clear objective, i.e. minimizing the empirical risk, unsupervised learning does not. Generally, it aims to distill meaningful information from data, but it lacks a clear definition of what is meaningful. Despite its name, unsupervised learning still requires manual determination of what to extract. For example, the class label represents one form of meaningful information agreed upon by us all. Recently, there has been a popular line of work focused on so-called self-supervised learning, using priors from the engineer to learn good compact representations [92, 43, 81, 88]. Note that the priors for unsupervised methods are very different from those for supervised learning. In supervised learning, the need for priors diminishes with an increase in the sample size. However, in unsupervised learning, the results are heavily influenced by the prior regardless of data volume [189].

Depending on the modeling type, unsupervised learning can be grouped into generative methods and discriminative ones. The following section will give a review of them.

1.1.3 K-means, GMM, and entropy clustering

We start from two basic generative methods: K-means and GMM. Generative methods basically mean modeling the data density. Why is density estimation useful for classification? Because the density function could delineate distinct clusters, wherein samples within each cluster are connected through regions of high density and each cluster is considered as a class. Naturally, the choice of the density model is crucial[2]. We will use simple examples in Figure 1.1 to illustrate this and show the potential advantage of the discriminative method, e.g. entropy clustering which will be discussed in more detail in Chapter 3.

We start with the most basic K-means. The loss can formulated as:

$$L_{KM}(S,\mu) = \sum_{k=1}^{K} \sum_{i \in S^k} \|X_i - \mu_k\|^2 \tag{1.8}$$

where $\|\cdot\|^2$ is the squared L_2 norm, $S = (S^1, ..., S^k)$ represents the categorical subsets and $\mu = (\mu_1, ..., \mu_K)$ represents the centers of each cluster. To understand how K-means is related to

[2]While there are non-parametric models such as [100, 190], we are focused on the parametric ones here, because non-parametric methods, although versatile, can encounter scalability issues when applied to large datasets.

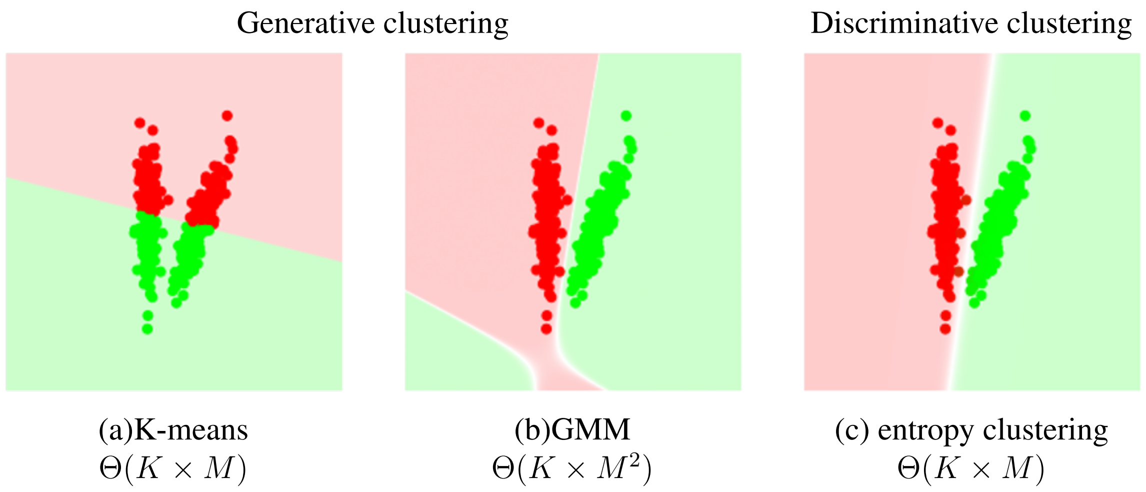


Figure 1.1: K-means vs. GMM vs. entropy (decisiveness and fairness) clustering - binary example ($K = 2$) for 2D ($M = 2$) data $\{X_i\}$ comparing generative and discriminative methods with different parametric complexity: (a) K-means $\mu_k \in \mathcal{R}^2$, (b) GMM $\mu_k \in \mathcal{R}^2$ and $\Sigma_k \in \mathcal{R}^{2\times 2}$ and (c) entropy clustering (see Chapter 3) based on a linear classifier parameterized by K-columns matrix $\mathbf{v}$ where linear discriminants $\mathbf{v_k} \in \mathcal{R}^{2+1}$ include the bias. Red and green colors in (a), (b) and (c) illustrate the decision regions over $X \in \mathcal{R}^2$ produced by the corresponding decision functions for optimal parameters: (a) linear decision boundary (b) quadratic decision boundary and (c) linear classifier. The decision function in (a) is hard, and is soft in (b) and (c), see the white region near the decision boundary. The optimal results in (a) and (b) are analyzed in Section 1.1.3. The optimal result in (c) requires a *margin maximization* term $\|\mathbf{v}\|^2$, see Section 3.2.2.

density estimation, we should reformulate the loss as follows:

$$L_{SNLL}(S, \mu) = -\sum_{k=1}^{K} \sum_{i \in S^k} \ln P(X_i \mid \mu_k) \; + \; const \tag{1.9}$$

where $P(X_i \mid \mu_k) = \mathcal{N}(X_i \mid \mu_k, \sigma^2 I)$, the isotropic Gaussian distribution. From (1.9), K-means loss can be seen as the sum of NLLs. Each component is represented as an isotropic Gaussian distribution, hence the K-means loss aims at finding clusters of blob-like shape. In Figure 1.1 (a), the shape of data clusters is elongated, resulting in the failure of K-means. The optimization algorithm, i.e. Lloyd's algorithm, [133], is block-coordinate descent where S and μ alternate being optimized given the other is fixed. Two steps both have closed-form solutions:

$$S_i = arg \min_k \|X_i - \mu_k\|^2 \tag{1.10}$$

$$\mu_k = \frac{\sum_{i \in S^k} X_i}{|S^k|} \tag{1.11}$$

where $| \cdot |$ is the cardinality. Also note that the decision boundary between each two clusters is linear, as illustrated in Figure 1.1 (a), because the squared data terms will cancel out, leaving the decision boundary in a linear form.

Obviously, to correctly separate the two clusters in Figure 1.1, the density function should not be limited to isotropic Gaussian density. The Gaussian Mixture Model (GMM) offers flexibility by estimating the covariance matrices. The GMM method uses a mixture of Gaussian models to estimate the density. The number of Gaussian components is usually the number of clusters if we could get such information in any way. Then the NLL loss is:

$$L_{GMM}(\theta) = -\sum_{i} \ln \sum_{k} \rho_k P(X_i | \mu_k, \Sigma_k) \tag{1.12}$$

where θ represents all the parameters to estimate, ρ_k represents the proportion of the component k, and Σ_k is the covariance matrix. As shown in Figure 1.1 (b), we can get two clusters correctly separated now. Note that the decision boundary is not linear anymore because the quadratic data term can not cancel out due to different covariance matrices. Clearly, increasing the complexity of the density function can solve the problem but it introduces more parameters to estimate and the algorithm to optimize the loss also becomes more complicated, which limits its scalability. The expectation-maximization (EM) algorithm [16] introduces auxiliary variables to decompose the optimization problem into easier ones.

In Chapter 3, we focus on an alternative discriminative group of clustering methods. Unlike generative methods working with density functions, the clustering losses in Chapter 3 are directly defined over the output of posterior models, as in [30]

$$L_{ec} \quad = \quad \overline{H(\sigma)} \; - \; H(\overline{\sigma}) \tag{1.13}$$

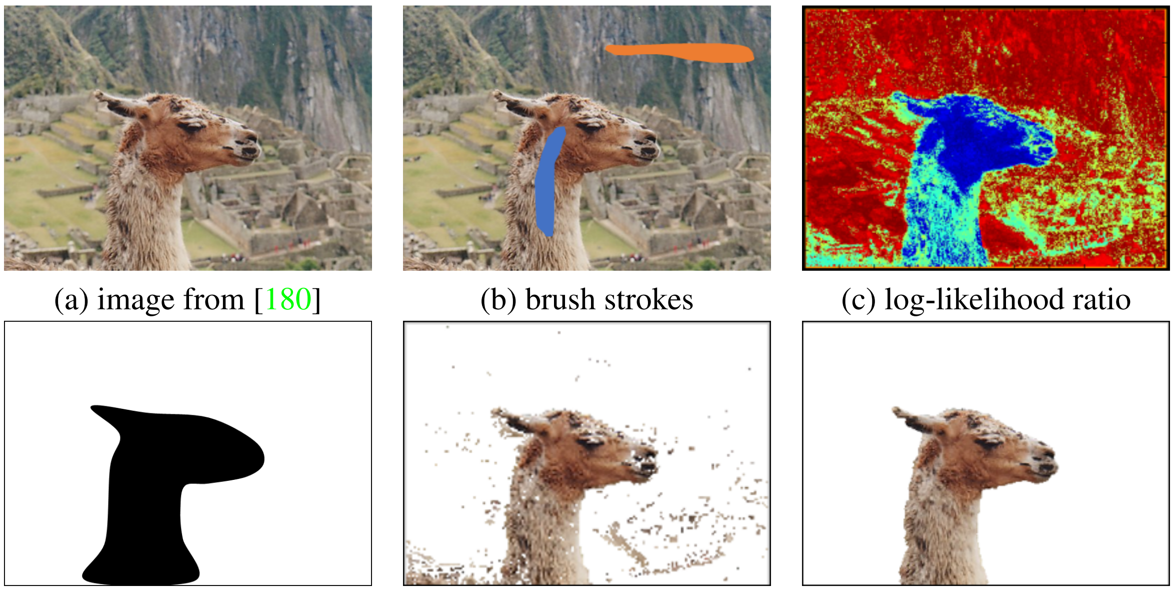
where $\sigma \in \Delta^K$ is *softmax*, which is a typical probabilistic output of neural networks. Such losses commonly use entropy function, e.g. Shannon's entropy H above. The bar operator indicates averaging over data points, see Chapter 3 for more details. We refer to this group of methods as *discriminative entropy clustering*. Figure 1.1 (c) shows the optimal result of unsupervised loss (1.13) for linear classifier model $\sigma(\mathbf{v}^\top X)$. The parameter complexity of this clustering method is the same as K-means, but it finds good linear decision boundaries ignoring complex data density models in GMM.

1.2 Image segmentation

Image classification operates at the image level, providing a single label for the entire image. However, images often contain multiple different objects. To precisely localize and delineate these objects, pixel-level classification is needed. Image segmentation aims to partition image pixels into non-overlapping sub-regions corresponding to some predefined categories. In the next section, we will first introduce the classic interactive image segmentation problem, which relies on low-level cues such as color intensity. We will then move on to fully-supervised semantic segmentation and its weakly-supervised counterpart, where we will see how the low-level segmentation techniques can help.

1.2.1 Low-level segmentation with Potts model

Image segmentation is a classic problem in computer vision. It aims at grouping pixels into different subsets. To begin with, we look at a classic binary segmentation example as shown in Figure 1.2. Assume we are given an image I where each pixel $p \in \Omega$ has intensity I_p. The segmentation task is to partition the pixels into two groups, i.e. foreground and background. In other words, we need to give each pixel a label $S_p \in \{0, 1\}$. There are exponentially large numbers of partitions, so constraints are needed. In Figure 1.2 (b), we use strokes to roughly mark the foreground and background regions. Using the pixels inside the stroke area, we can estimate the so-called appearance model for each class like Figure 1.3. Since this is the density estimation, we can use the GMM discussed in Section 1.1.3. Given the appearance model, the simplest way



(a) image from [180] (b) brush strokes (c) log-likelihood ratio

(d) ground-truth (e) threshold (f) graph cut [26]

Figure 1.2: (Interactive) image segmentation. (a) is the original image to be segmented into the foreground of the llama and background as shown in (d). (b) shows the brush strokes indicating the foreground and background. (c) shows the log-likelihood ratio for all pixels using the model estimated from the stroke region via the maximum likelihood estimation principle. (e) is the result of simple thresholding. Due to the independent treatment of each pixel, the result in (e) is very noisy. Methods, such as [26], exploit relation between pixels to produce a smooth result as in (f).

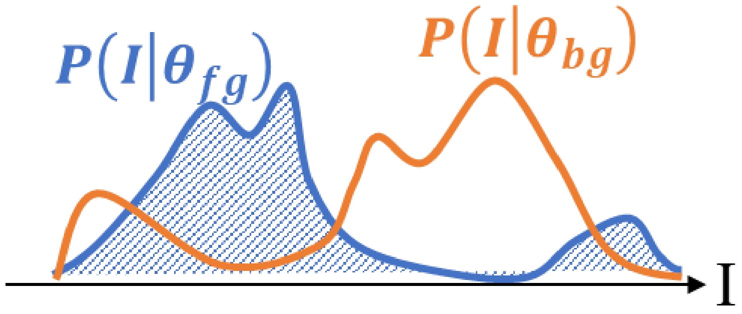


Figure 1.3: Estimated density function for foreground and background based on the pixels inside the brush strokes.

to determine the class of each pixel is thresholding based on the log-likelihood ratio. The formula is given as (1.14) and such ratio is shown for each pixel in Figure 1.2 (c).

$$g_p = -\ln \frac{P(I_p \mid \theta_{fg})}{P(I_p \mid \theta_{bg})} \tag{1.14}$$

If $g_p < 0$, then $S_p = 1$ (we usually designate 1 to the foreground region). Otherwise, $S_p = 0$. Such thresholding can also be formulated into a minimization problem where the objective is given by:

$$E(S) = \sum_{p\in\Omega} g_p \cdot S_p \tag{1.15}$$

Motivated by physics, such objectives are often called *energies*. The result is shown in Figure 1.2 (e), and obviously it is very noisy due to the overlapping of the appearance model. The following energy [73], extending (1.15), accounts for the number of labeling discontinuities:

$$E(S) = \underbrace{\sum_{p\in\Omega} g_p \cdot S_p}_{\text{data/unary term}} + \lambda \underbrace{\sum_{(p,q)\in\mathcal{N}} w_{pq}[S_p \neq S_q]}_{\text{smoothness/pairwise term}} \tag{1.16}$$

where λ is the relative weight between the two terms, $\mathcal{N}$ represents the neighborhood system and each discontinuity is often weighted by $w_{pq} = exp(-\beta\|I_p - I_q\|^2)$ based on color difference and $\beta^{-1} = \frac{\sum_{(p,q)\in\mathcal{N}}\|I_p - I_q\|^2}{|\mathcal{N}|}$. Consequently, the solution with more discontinuities like Figure 1.2 (e) will be assigned a much higher energy value while the cleaner result in (f) gives a lower energy value. Essentially, we introduced a bias towards "smooth" solutions, and the level of the smoothness is based on the value of λ. Since the weights w_{pq} impose a lower penalty if the boundary is highly contrasted, the energy (1.16) also promotes edge alignment. In Bayesian statistics, approaches like (1.16) correspond to a maximum a posteriori probability (MAP) estimate for parameter S [73], where the prior distribution corresponds to the pairwise term. In the literature, they are called Markov Random Field (MRF) regularization [73] or Conditional Random Field (CRF) regularization [121]. Note that (1.16) is formulated for binary case, where the model is called *Ising* model [144], and it can be easily extended to the multi-label case:

$$E(S) = \sum_{p\in\Omega}\sum_{l\in\mathcal{L}} u_p^l S_p^l \;+\; \lambda \sum_{(p,q)\in\mathcal{N}} w_{pq}[S_p \neq S_q] \tag{1.17}$$

where $\mathcal{L} = \{1, 2, ..., K\}$, u_p^l is the unary potential, $S_p^l = [S_p = l]$ is an indicator and $w_{pq} > 0$. This model is called the Potts model, which is a generalization of the Ising model and a special case (second-order) of the general MRF/CRF regularization. As mentioned above, the pairwise term uses a predefined neighborhood system to enforce the pairwise constraints. Why does the neighborhood system need such emphasis? As follows, we will show that different neighborhood system leads to very different properties and optimization algorithms for the Potts model.

(Grid) Potts model The typical Potts model uses the simplest 4-grid (up, left, down, right) neighborhood system. With such a grid neighborhood, the Potts model can be interpreted as an approximation of the length of the boundary between segments:

$$\sum_{(p,q)\in\mathcal{N}} [S_p \neq S_q] \approx \sum_{l\in\mathcal{L}} |\mathcal{S}_l| \tag{1.18}$$

where $\mathcal{S}_l$ is the set of the points on the boundary of segment S_l. Boykov and Kolmogorov [24] established this relation by using the Cauchy-Crofton formula from integral geometry that relates the length of a curve with the number of its intersections by a random line. Consequently, such type of Potts model prefers shorter boundary, which is known as the *shrinking bias*. Intuitively, the bias is towards the blob-like shapes. As for long and thin structures like blood vessels, we need some other regularization, such as curvature [93, 199, 55, 185, 198, 186, 91, 27, 157, 153], which will be discussed in Section 1.3.

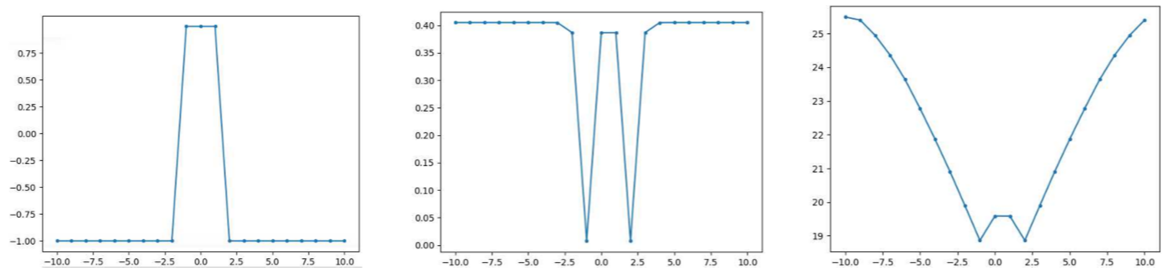
The incorporation of pairwise terms induces better objective function, but the optimization becomes more complex than the simple thresholding. Various algorithms are proposed for energy minimization using the grid Potts model. When $|\mathcal{L}| = 2$, the energy (1.16) can be globally optimized via efficient graph cuts algorithm [26, 114, 85]. In fact, any second-order/pairwise set function that is *submodular* can be globally optimized in polynomial time via graph cuts [86, 168, 19, 114], and the Ising model is just a special case of binary submodular energies. Here the submodularity and set function are the concepts in combinatorial optimization and are also useful in information theory, game theory, graph theory, and related fields [135]. In the multi-label case, i.e. $|\mathcal{L}| > 2$, the problem becomes NP-hard. Different types of approximation algorithms are proposed. For instance, the alpha-expansion [25] algorithm is introduced for a broad range of pairwise energies, leading to various extensions such as label costs [56], auxiliary cuts [12], local submodular approximations [77], and PBO [204]. Several general-purpose approximate pairwise optimization methods have also been proposed, including LBP [164], TRW-S [113], QPBO [18]. For a comprehensive review and comparison, please refer to [202, 105]. Additionally, Geman and Geman [73] employed a (probabilistic) Monte Carlo Markov Chain (MCMC) [17] approach to optimize energy, using the Gibbs sampler with the annealing strategy.

Besides the discrete optimization, different relaxations can be used for continuous optimization. In particular, the Potts model $[S_p \neq S_q]$ can be relaxed to

- total variation (TV) relaxation:
$$\sum_l |S_p^l - S_q^l|$$
- quadratic relaxation:
$$\sum_l \|S_p^l - S_q^l\|^2$$
- bilinear relaxation:
$$\sum_l S_p^l(1 - S_q^l) + S_q^l(1 - S_p^l)$$

where $\sum_l S_p^l = 1$ and $S_p^l > 0$ for all $p \in \Omega$. Here, the TV and quadratic relaxation are convex so the global optimum can be found. The quadratic formulation is smooth and thus supports the standard gradient descent algorithm while a closed-form solution exists and is related to random walker [1, 78] or label propagation [239, 236, 14]. Although the subgradient descent algorithm can be applied to the TV formulation, it usually leads to slow convergence. Chambolle and Pock [32] use the general primal-dual technique to develop a faster algorithm that supports parallel implementation on GPU. The bilinear relaxation is not convex but is tight [174]. Here tightness means that the optimization of the energy using the relaxation formulation leads to the same minimal value as the original discrete problem. Additionally, the probabilistic approach can still be used for continuous optimization, such as mean-field approximation in the variational inference [17].

Dense Potts model Grid Potts model assumes independence from the non-neighboring pixels, i.e. the Markov effect, which makes the evaluation and (approximate) inference in such models tractable [140]. The dense Potts model links the non-neighboring pixels and pushes this to the



(a) 1D image (b) grid Potts (c) dense Potts

Figure 1.4: Synthetic segmentation example for grid and dense Potts models: (a) intensities $I(x)$ on 1D image. The cost of segments $S_t = \{x|x < t\}$ with different discontinuity points t according to (b) nearest-neighbor (grid) Potts and (c) larger-neighborhood (dense) Potts. The latter gives a smoother cost function, but its fatter minimum may complicate discontinuity localization as shown in Figure 1.10. Figure from [142, 140].

limit by connecting to all other pixels. The efficient mean-field approximation can still be used here, but the pairwise term is limited to the Gaussian potential [115]. Despite the "dense" nature, one could control the coordinate bandwidth inside the Gaussian potential to determine the range of the interactions among pixels. Veksler [218] noted that for large neighborhood size, it is equal to volume estimation, suggesting weaker regularization effects. On the algorithmic level, the dense Potts is smoother as shown in Figure 1.4 (c), which makes it amenable to gradient descent. Consequently, it is easier to combine the dense Potts with the neural network training which will be introduced in Section 1.2.3.

1.2.2 Semantic segmentation with neural networks

In previous sections, we are focused on low-level interactive segmentation and introduced the commonly used regularization loss such as the Potts model. Here, we will introduce another type of segmentation where the image will be partitioned into subregions of predefined semantic categories, such as people, birds, trees, etc. Different from interactive segmentation which relies on low-level cues such as color intensity, and pixel coordinates, semantic segmentation needs more complex features. As mentioned in Section 1.1, learning-based methods, particularly neural networks, can be suitable for such tasks. In the following, we introduce the network architecture modified from the classification task and the corresponding pixel-level loss function.

First, let's see how a classification network can be turned into one for segmentation tasks. In Section 1.1, we mainly discussed the losses to train a neural network. Here we give an

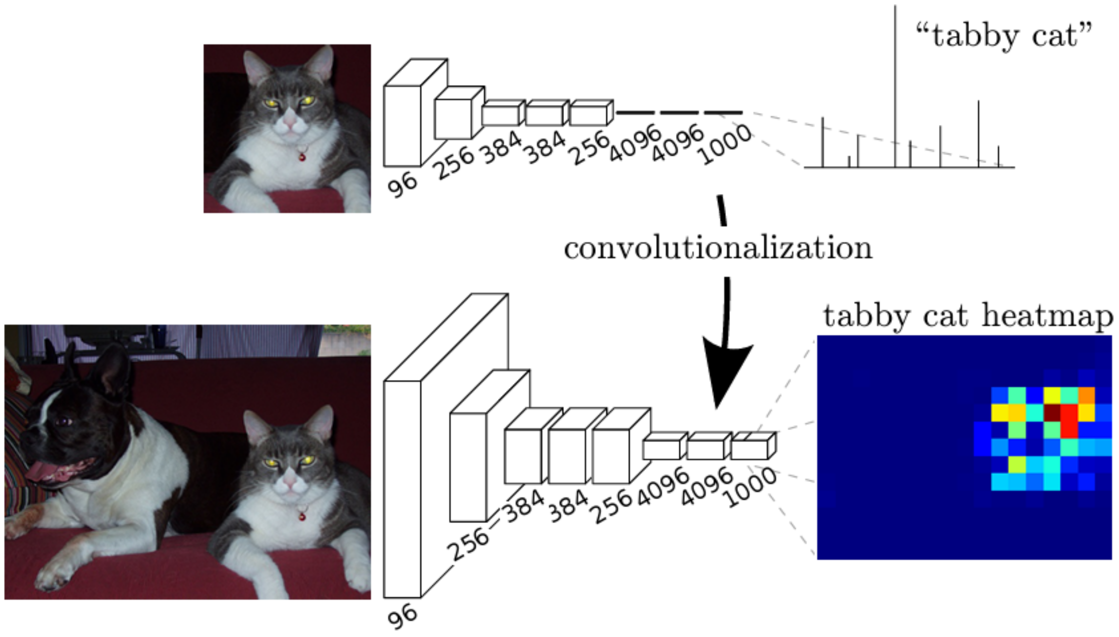


Figure 1.5: The transformation from a classification network to a fully convolutional network (FCN) by changing the last three fully connected layers into convolutional layers. The resulting network (at the bottom) can be applied to images of any size, which makes the architecture suitable for (coarse) semantic segmentation tasks. Each rectangular block represents an intermediate feature map, with two spatial dimensions and one feature dimension. The spatial dimensions are determined by the input and convolution kernel size, while the numbers below the blocks indicate the feature dimensionality. It is implicitly assumed that each consecutive pair of blocks contains a convolution layer, activation layer, and pooling layer if needed. The relative size of the blocks reflects downsampling along the spatial dimensions (due to convolution strides and/or max-pooling) and an increase in feature dimension (due to more convolutions within a single convolution layer). Figure from [134].

introduction to the neural network architectures. The early version of neural networks is the multilayer perceptron (MLP) which is a stack of linear transformations and activations:

$$h(x) = h_L(\sigma(...(h_2(\sigma(h_1(x))))))$$

where L is the number of layers, σ is the element-wise nonlinear activation function and $h_i(x) = \mathbf{W_i}^\top x$ is a linear transformation. Following the philosophy of network architecture design, multiple types of layers are discovered, such as the convolution layer, pooling layer, attention layer, etc [124]. The convolutional neural network (CNN) that usually combines convolution, linear, activation, and pooling layers has recently dominated the field for several years [192, 117, 90]. Compared to MLP, it has fewer parameters and translational equivariance due to the convolution operation. From the Figure 1.5, we can see that the last few layers are linear layers to calculate the score for each class. To support the segmentation task and the different sizes of the input, the fully convolutional neural network (FCN) is proposed [134]. Based on this work, many other variants are proposed such as U-net [178], Segnet [10]. The overall architecture typically consists of an encoder backbone and a decoder, where the encoder is usually pre-trained on ImageNet based on the concept of transfer learning [159]. The decoder is used to recover the original resolution to compute the loss on each pixel. DeeplabV3+ [42], as shown in Figure 1.6, is a classic encoder-decoder framework. In the encoder part, it uses Atrous convolution and pyramid pooling operations to better preserve the spatial information while having a larger receptive field [41]. The decoder part combines the low-level and high-level features to leverage both fine-grained details and semantic context.

The loss function is the pixel-wise cross-entropy loss:

$$L(\theta) = \mathbb{E}_{(I,\bar{y})} \frac{1}{|\Omega|} \sum_{p \in \Omega} - \ln \sigma_p^{\bar{y}_p}(\theta) \tag{1.19}$$

where θ is the trainable network parameters, and $\sigma_p^{\bar{y}_p}$ is the softmax output at pixel p. The ground truth label at any given pixel $p \in \Omega_{\mathcal{L}}$ is an integer

$$\bar{y}_p \in \{1, \ldots, K\} \tag{1.20}$$

where K is the number of classes including the background. Without much ambiguity, it is convenient to use the same notation $\bar{y}_p$ for the equivalent *one-hot* distribution

$$\bar{y}_p \;\; \equiv \;\; (\bar{y}_p^1, \ldots, \bar{y}_p^K) \; \in \Delta_{0,1}^K \qquad \text{for} \quad \bar{y}_p^k := [k = \bar{y}_p] \; \in \{0, 1\} \tag{1.21}$$

where $[\,\cdot\,]$ is the *True* operator for the condition inside the brackets. Set $\Delta_{0,1}^K$ represents K possible one-hot distributions, which are vertices of the K-class *probability simplex* representing all K-categorical distributions. The context of specific expressions should make it obvious if $\bar{y}_i$ is a class index (1.20) or the corresponding one-hot distribution (1.21). The optimization methods introduced in Section 1.1.1 can be applied to minimize (1.19).

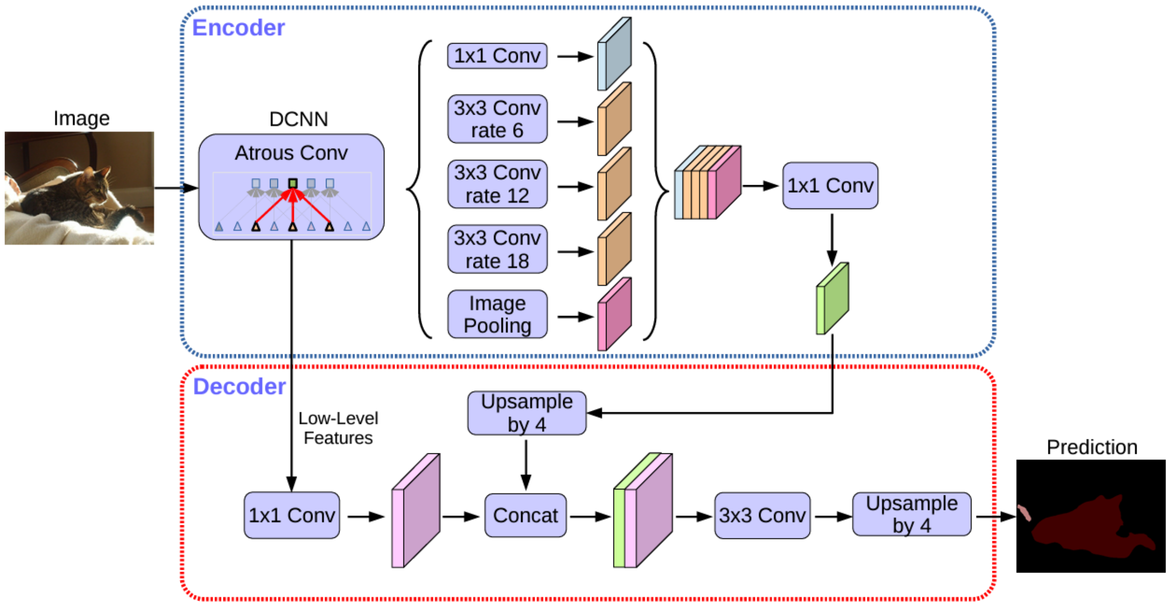


Figure 1.6: The encoder-decoder architecture of deeplabV3+ framework [42]. As shown in Figure 1.5, the output map has a much lower resolution compared to the original image. The decoder part upsamples the feature map to the original resolution to get fine-detailed segmentation output. DCNN represents the deep CNN model and Atrous Conv is a type of convolution supporting a larger receptive field with the same number of parameters as the original convolutional kernel. The rate of atrous convolution determines the spacing between the kernel elements during convolution, controlling how many pixels in the input feature map are skipped when applying the convolution operation. The stack of blocks in the encoder part represents the concatenation of feature maps using the Atrous convolution with different rates. In the decoder part, the low-level features from the first several layers are concatenated with the high-level features from later ones to better capture both the fine details and semantic information.

1.2.3 From pixel-wise ground-truth to weak supervisions

One big issue with pixel-level supervision is the huge amount of labeling cost. According to [6], it takes an average of 19 minutes to annotate an image for the COCO dataset [131] and 1.5 hours to fully annotate a single image from the Cityscapes datasets [49]. This motivates weaker forms of supervision such as image tags, scribbles, or bounding boxes. The most appealing form is the image tag which is the relatively easiest to obtain. How can the pixel label be inferred from just the image tags? To give an intuition on this question, we start from the binary case.

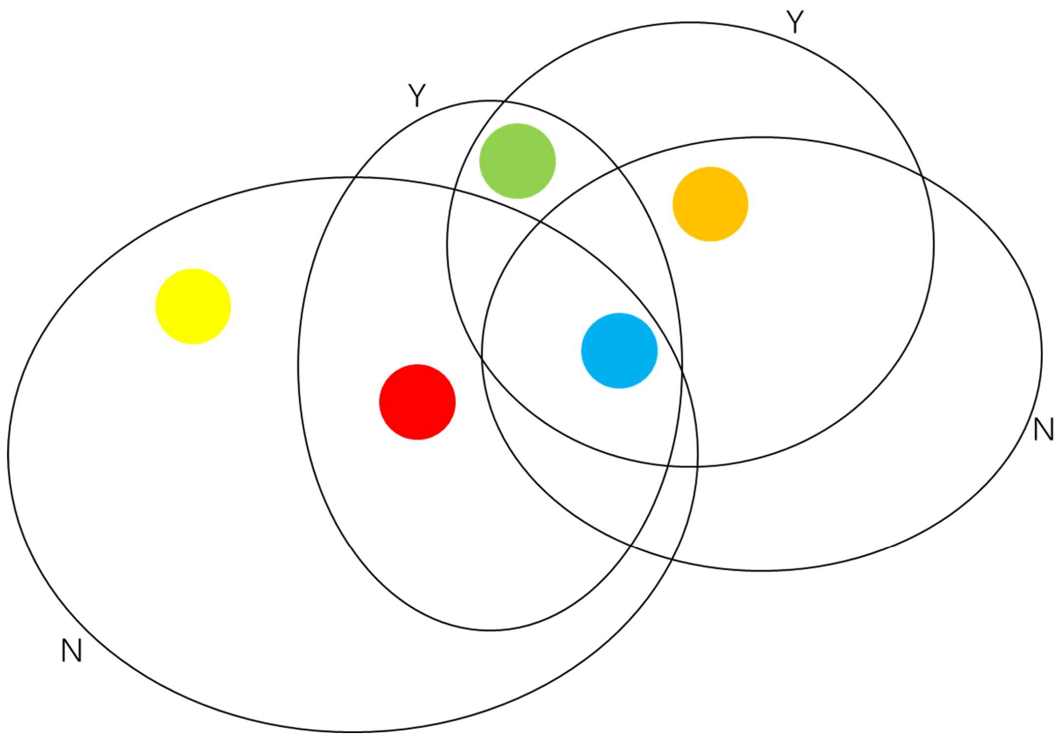


Figure 1.7: The illustration of multiple instance learning (MIL). The colorful solid circles represent different molecules and big black circles represent different drugs. The letters “Y” and “N” are the labels indicating whether a certain drug works or not. The goal is to find the molecules that make the drug work based on the test information.

Imagine we deal with the problem of drug (molecule) discovery and it is only possible to test if a certain combination of molecules works, as shown in Figure 1.7. Given the labels "yes, it works" or "no, it doesn’t" in Figure 1.7, we can infer that the green dot is the one that works. Such binary classification problem is well-known as multiple instance learning (MIL) where the training labels are only available for sets/bags of examples instead of individual ones. This is a combinatorial problem. In Figure 1.8, the task is extended to the multi-label case and is tag-supervised segmentation. We can infer that all the green pixels should be assigned to the grass because the grass is in the first and third images but not the second, which means that the grass can not be blue or yellow. In practice, the segmentation network needs to solve a combinatorial problem to locate the object corresponding to some class and this needs a ton of images. Also, the network needs to figure out good features as well because, not as simple as shown in the figure, color cues are usually not enough. To enable the network to learn the pixel-wise prediction from the image tags, several strategies are proposed to convert the pixel-wise predictions into image-level ones by different types of pooling [162, 169, 112]. Most methods build upon the classification activation map (CAM) [234] and try to refine the map [112, 3, 125, 2, 7, 237]. However, one can argue that the CAM was not originally designed for the segmentation problem, which makes this line of approach sub-optimal.

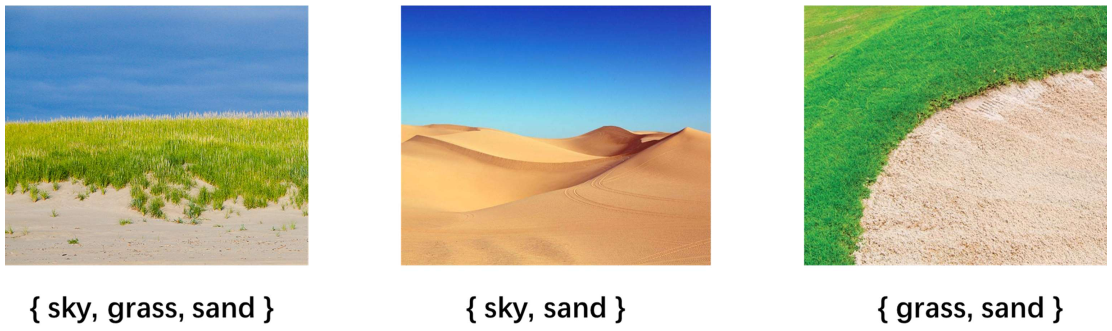


Figure 1.8: A multi-label illustration of MIL. Each image is given a tag containing the objects present in the image. The goal is to find correspondence between labels and objects in the image. For example, it can be inferred that the grass should correspond to the region of green pixels.

Tag supervision typically requires highly specialized systems and complex multi-stage training procedures, which are hard to reproduce. Another form of supervision is the scribble-based approach motivated by its practical simplicity and mathematical clarity. The scribble-supervised semantic segmentation is the focus of the Chapter 4. The corresponding methodologies are focused on the design of unsupervised or self-supervised loss functions and stronger optimization algorithms. The corresponding solutions are often general and can be used in different weakly-supervised applications. Assume a subset of pixels with ground truth labels is $\Omega_{\mathcal{L}} \subset \Omega$, which we call *seeds* or *scribbles* as subset $\Omega_{\mathcal{L}}$ is typically marked by mouse-controlled UI for image annotations, e.g. see seeds over an image in Figure 1.9. Loss functions for weakly supervised segmentation with scribbles typically use *NLL*/cross-entropy loss over scribbles $p \in \Omega_{\mathcal{L}} \subset \Omega$ with ground truth labels $\bar{y}_p$

$$-\sum_{p \in \Omega_{\mathcal{L}}} \ln \sigma_p^{\bar{y}_p} \tag{1.22}$$

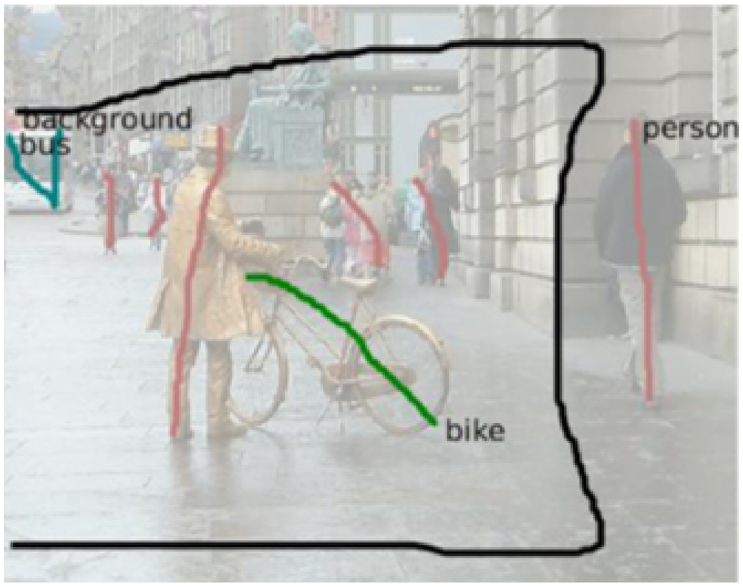


Figure 1.9: Scribbles on the image. Different categories correspond to different colors of the scribbles. Figure from [206].

where $\sigma_p = (\sigma_p^1, \ldots, \sigma_p^K) \in \Delta^K$ is the model prediction at pixel p. Note that here we only show the loss for one image for brevity. This loss is a standard in full supervision where the only difference is that $\Omega_{\mathcal{L}} = \Omega$ and usually, no other losses are needed for training. However, in a weakly supervised setting the majority of pixels are unlabeled, and unsupervised losses are needed for $i \notin \Omega_{\mathcal{L}}$. The most common unsupervised loss in image segmentation is the Potts model and its relaxations as introduced in Section 1.2.1. Tang el.al. [206] proposed combining the bilinear relaxation of the dense Potts model

$$R(S(\theta)) = \sum_k (1 - S^k(\theta))^\top \mathbf{A} S^k(\theta) \tag{1.23}$$

with the NLL loss on the scribbles where θ is the network parameter, vector $S^k(\theta)$ is the flattened

$$S^k = \{\sigma_p^k \mid p \in \Omega\}$$

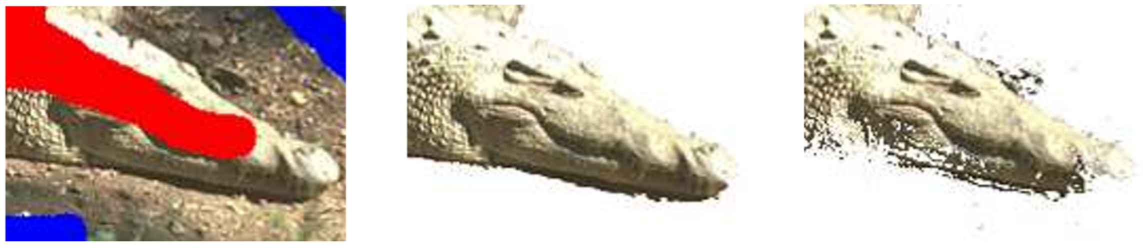
prediction map for class k and $\mathbf{A}$ is the affinity matrix using Gaussian weights. The combined regularized WSSS loss in [206] is

$$-\sum_{p\in\Omega_{\mathcal{L}}} \ln \sigma_p^{\bar{y}_p} \quad + \quad \sum_k (1 - S^k(\theta))^\top \mathbf{A} S^k(\theta) \tag{1.24}$$

During the training backpropagation, the gradient of the Potts model is calculated as

$$\frac{\partial R}{\partial S^k} = -2\mathbf{A}S^k$$

where the bilateral filtering is applied for efficient computation [115].



(a) (b) (c)

Figure 1.10: Low-level segmentation example for grid (b) and dense (c) Potts models for image with scribbles (a). Grid Potts gives smoother segment boundaries with better edge alignment, while dense Potts often gives noisy boundaries. Figure from [142].

As discussed in Section 1.2.1, the grid Potts model enforces stronger regularization but suffers from GD. In [142], they propose using the alternate direction multiplier (ADM) framework by introducing auxiliary variables to split the original optimization problem into easier subproblems, where one can apply strong low-level solvers such as graph cut [25]. The loss is as follows:

$$\begin{aligned} \min_{\theta, Y} \quad & \sum_{p \in \Omega_{\mathcal{L}}} l(S_p(\theta), \bar{y}_p) \; + \gamma \sum_{p \in \Omega_{\mathcal{U}}} D(Y_p | S_p(\theta) \; + \; \lambda \sum_{(p,q) \in \mathcal{N}} w_{pq} [Y_p \neq Y_q] \\ \text{s.t.} \quad & Y_p = \bar{y}_p \qquad\qquad \forall p \in \Omega_{\mathcal{L}} \end{aligned}$$

where $Y_p \in \{0,1\}^K$ is a one-hot distribution and D is some divergence measure, e.g. Kullback-Leibler divergence. R is now a classic *discrete* Potts model as in (1.17). The optimization is done in block-wise descent fashion, where the Y is solved by graph cut solver and the network parameters θ are optimized via SGD.

1.3 Thin structure in medical data

Unlike generic images, medical images present distinct challenges due to their complex and heterogeneous nature, requiring specialized segmentation methods to extract clinically relevant information. In this section, we shift our focus from the broad context of image segmentation to the critical role it plays in medical imaging, particularly 3D vascular computed tomography (CT), where precise delineation and characterization of thin structures are required for accurate diagnosis, treatment planning, and patient care.

1.3.1 Difficulties in medical image labeling

Due to patient confidentiality, both the raw data and the annotations are often harder to acquire. Even with enough raw data, annotating the medical images needs expertise and a large amount of manual work [111]. Moreover, for tasks like segmentation of high-resolution 3D angiography, it is almost impossible to annotate the data at near-capillary level. Although the newer micro-CT is able to provide ultra high resolution (voxel size$<= 20\mu m^3$) and have higher signal-to-noise ratios (SNR), it is still hard to distinguish the thin near-capillary vessels from the background by intensity alone, see Figure 1.11 (b). The sheer volume of data makes it practically impossible to employ supervised methods due to the prohibitive cost of annotations. [140]. Consequently, robust unsupervised methods are needed.

1.3.2 Unsupervised vessel segmentation methods

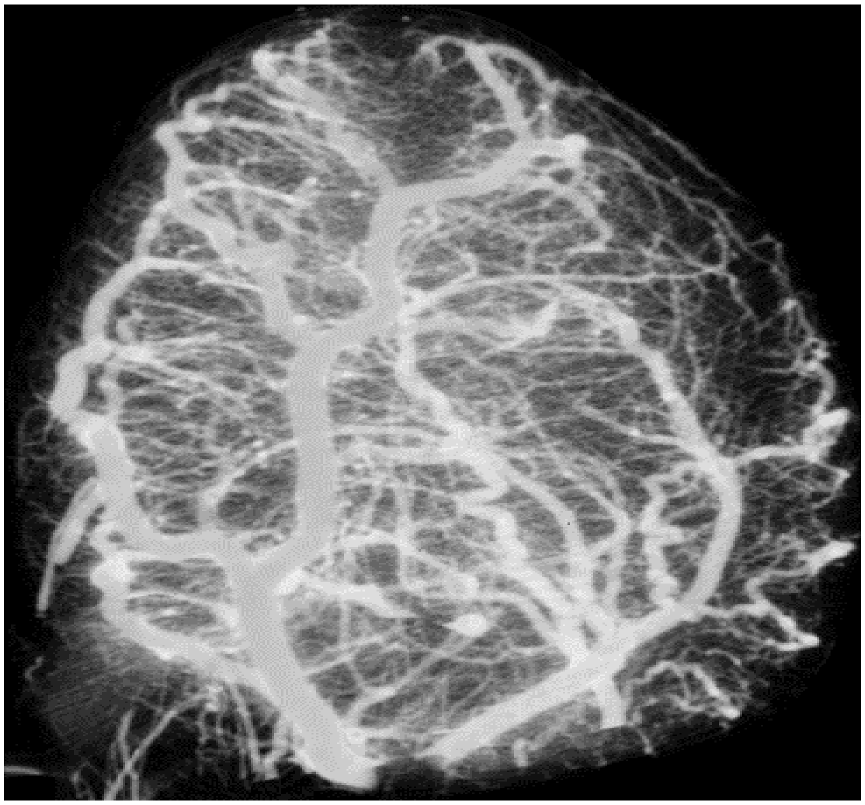
There is plenty of prior work on estimation of vessels in computer vision and biomedical imaging communities [148]. Typically, pixel-level detection of tubular structures is based on multi-scale eigen analysis of intensity Hessians developed by Frangi et.al. [70] and other research groups [64]. At any given point (pixel/voxel) such vessel enhancement filters output a *tubularness/vesselness measure* and estimates of the vessel's scale and orientation, which describes the flow direction upto to a sign. While such local analysis of Hessians is very useful, simple thresholding of points with large-enough vesselness measure is often unreliable as a method for computing the vessel tree structure. While thresholding works well for detecting relatively large vessels, detection of smaller vessels is complicated by noise, partial voluming, and outliers (e.g. ring artifacts caused by an improperly calibrated CT-scan [97]). More importantly, standard tubular filters exhibit signal loss at vessel bifurcations as those do not look like tubes.

Regularization methods can address the vessel continuation problems due to noise, outliers, and signal loss at thinner parts and bifurcations. Curvature is a second-order smoothness term used to regularize the model parameters defining the centerline. In general, curvature has been studied for image segmentation [184, 198, 187, 28, 91, 157, 152, 143], for stereo or multi-view-reconstruction [127, 155, 156, 224], connectivity measures in analysis of diffusion MRI [149], for tubular structures extraction [143], for *inpainting* [5, 33] and edge completion [83, 222, 4]. Marin et.al. [143] and Chesakov [46] employ the curvature regularization for vessel centerline extraction on 3D images resolving near-capillary vessels. My prior work [232] proposes oriented curvature and divergence regularization to improve the orientation estimation around the bifurcations.

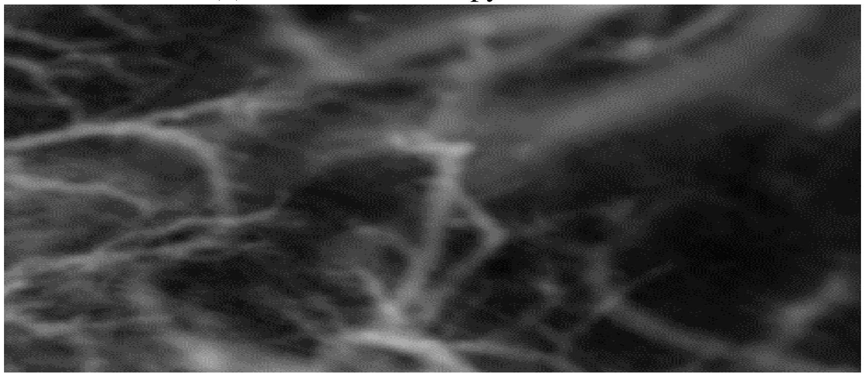
Vessel representation: centerline or segment Two common approaches to representing vessels in reconstruction methods are volumetric binary mask and centerline. A volumetric mask is typical for techniques directly computing vessel segmentation, $i.e.$ binary labeling of pixels/voxels. In contrast, centerline is a 1D abstraction of the vessel. But, if combined with information about vessel radii, it is easy to obtain a volumetric mask or segmentation from the vessel's centerline, e.g. using MAT [191]. Vice versa, centerline could be estimated from the vessel's binary mask using skeletonization algorithms. Bouix et.al. [21] adapted a skeletonization algorithm for centerline extraction on tubular structures. The skeletonization algorithm exploits properties of the average outward flux of the gradient vector field of a Euclidean distance function from the boundary of the structure. However, this method relies on the quality of the segmentation output (or boundary detection output) and it only utilizes local information to estimate the medial curve, which results in very unsmoothed centerline. In the context of regularization-based methods for vessel reconstruction, centerline representation offers significant advantages since powerful higher-order regularizers are easier to apply to 1D structures. For example, centerline's curvature can be regularized [143], while conceptually comparable regularization for vessel segmentation requires optimization of Gaussian or minimum curvature of the vessel's surface, with no known practical algorithms. In general, curvature remains a challenging regularization criteria for surfaces [184, 198, 91, 157, 152]. Alternatively, some vessel segmentation methods use simpler first-order regularizers producing minimal surfaces. While tractable, such regularizers impose a wrong prior for surfaces of thin structures due to their bias to compact blob shapes as discussed around (1.18).

Towards Whole Tree Centerline Many vessel reconstruction methods directly compute centerlines of different types that can be informaly defined as simplified (e.g. regularized) 1D representation of the blood flow *pathlines*. For example, the A/B shortest path methods require a user to specify two end points of a vessel and apply the Dijkstra [61] to find an optimal pathline on a graph with edge weights based on vesselness measure.

Interactive A/B methods are not practical for large vessel tree reconstruction problems. While it is OK to ask a user to identify the tree *root*, manual identification of all the end points (*leaves*) is infeasible. There are *tracing* techniques [9] designed to trace vessel tree from a given root based on vesselness measures and some local continuation heuristics. Chesakov [46] and Zhang et.al. [232] apply the minimum spanning tree (MST) as the last step to extract the topology of the whole tree. As shown in Figure 1.12, the blue dots are the middle points of the estimated



(a) cardiac microscopy CT volume



(b) zoom-in

Figure 1.11: Visualization (Maximum Intensity Projection) of the raw volumetric data obtained from a mouse heart by *micro computer tomography* (micro-CT). The data is provided by Maria Drangova from the Robarts Research Institute in London, Canada.

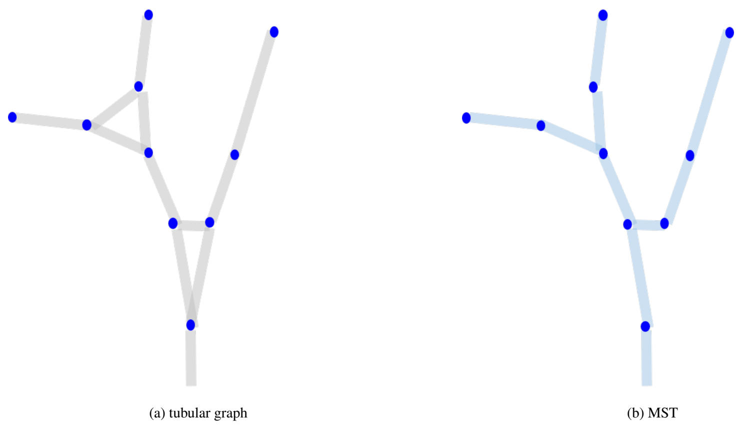


Figure 1.12: *Whole tree reconstruction*: (a) undirected tubular graph is built using K-nearest neighbor (KNN). Graph edges represent distances, geodesics, or other symmetric (undirected) properties. MST reconstruction quality (b) depends on the graph construction (nodes, neighborhoods, edge weights).

(oriented) tangents from the optimization of the objective with oriented curvature and divergence regularization. A so-called tubular graph is built using the K-nearest neighbors (KNN). The edge weight is calculated based on the minimum arc lengths between the tangents on the graph's vertices. Note that the orientation of the tangents is ignored when computing the edge weights. The MST is then applied to extract the tree topology as shown in Figure 1.12 (b). The MST algorithm [118] aims to minimize the following loss:

$$l(T) = \sum_{e \in \mathcal{E}(T)} w_e$$

where $\mathcal{E}(T)$ represents the set of edges of a spanning tree T on the graph, $w_e > 0$ is the weight for the edge.

1.4 Motivation and contributions

This thesis is focused on the design and study of unsupervised losses and corresponding optimization algorithms under the settings of annotation shortage. In Chapter 3, we focus on entropy-based clustering losses with softmax outputs [30]. We formally prove the maximum margin property for the losses using linear models and disprove the equivalence to K-means loss, which is misunderstood in previous literature [101]. The entropy-based losses are considered standard for discriminative clustering [30, 116], but the properties are not fully understood. We also revisit a standard unsupervised loss for weakly-supervised semantic segmentation (WSSS), i.e. the Potts model discussed in Section 1.2.1. Some alternative methods use the so-called foundation model such as the Segment Anything Model (SAM) [201] to address the WSSS problem. However, note that the SAM still needs pixel-wise masks to train the model and massive human interactions to rectify the errors in the masks [109]. Instead, our focus is the general unsupervised losses that can be used to train the segmentation models with much weaker forms of supervision. In Chapter 4, we propose soft self-labeling for WSSS and introduce several new relaxations of the Potts model. Note that the optimization algorithms introduced in Chapters 3 and 4 are based on self-labeling surrogates approximating the corresponding clustering and segmentation losses. In Chapter 2, we review the standard *splitting* techniques [23] that motivate such self-labeling surrogates. We also discuss prior self-labeling losses that lack robustness to errors in estimated pseudo-labels. Given our new losses, corresponding efficient self-labeling algorithms are introduced in Chapters 3, 4.

Chapter 5 is focused on unsupervised segmentation of vessels from 3D high-resolution CT images. Due to the shrinking bias of the Potts model (see Section 1.2.1) and difficulty of optimizing surface curvature regularization (see Section 1.3.2), centerline representation and 1D curvature are adopted in my prior work [232]. The global tree centerline is obtained via MST. However, the MST ignores the orientation resulting in artifacts around bifurcations. Chapter 5 proposes *confluence* as a new principle for computing directed edge weights. In our context of *directed* tubular graphs, we replace MST with the minimum *arborescence* algorithm [63].

Below we provide an overview of the problems addressed in this thesis. The main contributions are divided into four chapters. Chapters 3 and 4 use the general self-labeling methodologies reviewed in Chapter 2 but address different problems: clustering and segmentation. Chapter 5 on medical image analysis is fairly independent of the rest, albeit it uses unsupervised loss and focuses on a segmentation task similar to Chapter 4. We also give the introduction and literature review separately for chapters 3, 4, and 5. Chapters 2 through 4 are based on three papers, with two submitted to NeurIPS 2024, and one on Arxiv. Chapter 5 is based on [233] accepted at CVPR 2021. The following three subsections give a more detailed overview of the contributions in the main Chapters 3, 4, and 5.

1.4.1 Theory and algorithms for entropy-based clustering

In Chapter 3, we formally prove that linear entropy clustering (1.13) has a maximum margin property, distinguishing it from K-means minimizing cluster variances, as illustrated in Figure 1.1. Our theories establish a conceptual relation between discriminative entropy clustering [30] and SVM clustering [227]. The other contributions of Chapter 3 are two efficient optimization algorithms for entropy clustering based on new self-labeling surrogates, see Chapter 2. In particular, new EM-type algorithms proposed for estimating pseudo-labels are significantly faster than standard algorithms such as projected gradient descent or Newton's method. Our experiments show the effectiveness of our approach, improving SOTA for end-to-end deep clustering methods.

1.4.2 Potts relaxation and weakly-supervised semantic segmentation

In Chapter 4, we advocate for soft self-labeling in WSSS using splitting-based surrogates for (1.24), see Chapter 2. Note that soft pseudo-labels enable representation of uncertainty in segmentation estimates while most prior WSSS methods use hard pseudo-labels. As confirmed empirically, soft self-labeling significantly improves the robustness and accuracy of WSSS. We systematically explore different second-order relaxations of the Potts model including the standard quadratic and bilinear formulations. We study their properties both theoretically and empirically. Under the splitting framework for (1.24), our self-labeling surrogates combine these Potts model relaxations over pseudo-labels and different forms of cross-entropy connecting such pseudo-labels to network predictions, as introduced in Chapter 2. We derive an efficient algorithm for estimating soft pseudo-labels from these surrogates. The experimental results show that our method achieves the SOTA on the standard segmentation benchmarks. In some cases, it can outperform the fully-supervised segmentation due to improved generality.

1.4.3 Unsupervised vessel tree estimation with confluence

In Chapter 5, we propose using the minimum arborescence algorithm [63] to extract the tree on the directed graph. To compute the directed edge weights, we propose a new geometric prior, i.e. confluence, to incorporate the orientation estimated from the optimization of the objective with oriented curvature and divergence regularization as discussed in Section 1.3.2. Extensive experiments on synthetic and real data demonstrate better reconstruction around the bifurcations. The results achieve the SOTA and have been published in [233].

Chapter 2

Soft self-labeling for softmax models

This chapter develops new surrogate losses for complex unsupervised losses in clustering and segmentation based on entropy functions and the Potts model. The focus is on the standard *splitting* approach from optimization literature [23] that introduces auxiliary or hidden variables, treated as *pseudo-labels*, breaking the original optimization problem into two simpler sub-problems. Joint optimization of the surrogate losses over pseudo-labels and the network parameters constitutes "self-labeling". One of the key contributions in this Chapter is the new concept of *collision cross-entropy* (CCE) that connects pseudo-labels and soft-max predictions. This is an alternative to the standard Shannon's cross-entropy term used in all prior self-labeling methods. We show numerical and empirical advantages for CCE in the context of **soft** pseudo-labels that can represent estimation uncertainty. The algorithmic contributions in Chapters 3 and 4 depend highly on our general self-labeling approach discussed below.

Self-labeling can be seen as a high-order approximation methodology for optimizing unsupervised losses [57, 74, 8, 101, 142, 141]. Note that this is different from the self-supervised learning which is focused on the representation learning and largely depend on the priors from the engineer. The key components of self-labeling are auxiliary *pseudo-labels*, which have to be estimated similarly to hidden/latent variables in standard EM [17]. This chapter focuses on splitting methodologies [23] for deriving self-labeling surrogates for unsupervised losses in clustering (1.13) and segmentation (1.24). Such surrogates are jointly optimized w.r.t pseudo-labels and network predictions.

Below we discuss self-labeling with *soft* pseudo-labels. The softness during the optimization naturally represents the uncertainty or errors in the estimated pseudo-labels. We propose two losses specific to soft self-labeling: *reverse cross-entropy* (RCE) and *collision cross-entropy* (CCE). Compared to standard cross-entropy, they are more robust to uncertainty in soft pseudo-labels. In particular, CCE is symmetric w.r.t. jointly estimated pseudo-labels and predictions. Statistically, CCE can be seen as a probability that two random variables, class predictions and unknown true labels, are equal. This explains the term “collision”. In contrast, standard cross-entropy enforces softmax predictions to equal the distribution represented by soft pseudo-labels. Unlike our CCE loss, this forces network predictions to mimic the uncertainty of pseudo-labels, resulting in inferior numerical properties.

Our clustering and weakly-supervised segmentation algorithms in Chapters 3 and 4 are based on the soft self-labeling techniques and the cross-entropy functions introduced in this chapter.

2.1 ADM splitting for clustering and segmentation losses

One popular general optimization approach applicable to decomposable losses is ADMM [22] which splits optimization into two efficiently solvable sub-problems. Standard ADMM casts a problem $\min_x f(x) + g(x)$ into $\min_{x,y} \max_\lambda f(x) + g(y) + \lambda^\top(x-y) + \rho\|x-y\|^2$ and alternates updates over x, y and λ. If the L_2 penalty term is replaced by KL divergence, then the method falls into the Bregman-ADMM [23]. In practice, the Lagrangian term is omitted, making the form of splitting similar to a penalty method. Similar splitting techniques to improve the optimization are used in prior work for clustering and regularized losses in segmentation, as discussed in Section 3.1.4 and 1.2.3 respectively.

2.1.1 Our self-labeling loss in clustering

In clustering, the standard MI loss (1.13) can be split into simpler sub-problems separating the “decisiveness” and “fairness” terms. This requires introduction of auxiliary *splitting* variables $y \in \Delta^K$, one for each training example X. The Δ^K represents a general categorical distribution over K-classes. Then, optimization of the loss is equivalent to

$$\min_\theta \quad \overline{H(\sigma_\theta)} \qquad \text{(decisiveness sub-problem)} \tag{2.1}$$

$$\min_y \quad KL(\overline{y}\,\|\,u) \qquad \text{(fairness sub-problem)} \tag{2.2}$$

$$s.t. \quad y = \sigma \qquad \text{(consistency constraint)} \tag{2.3}$$

where θ is the network parameter, u is a uniform distribution and $KL(\overline{y}\,\|\,u) \equiv H(\overline{y}) + const.$ This can be solved approximately using a penalty method enforcing the constraint $y = \sigma$ via the *forward* KL divergence for y

$$L_{SL-clustering} \;:=\; \overline{H(\sigma)} \;+\; \beta\,\overline{KL(\sigma\,\|\,y)} \;+\; \lambda\,KL(\overline{y}\,\|\,u) \tag{2.4}$$

Ideally, the penalty method solves a series of unconstrained problems by changing the β. In practice, we treat it as a fixed hyperparameter and set it to 1. Thus, the equality constraint $y = \sigma$ may not be satisfied exactly. Then, (2.4) is only an approximation of the loss (1.13). When $\beta = 1$, the (2.4) can be written as

$$L_{SL-clustering} \quad \overset{\beta=1}{\equiv} \quad \overline{H(\sigma, y)} \quad + \lambda\, KL(\overline{y} \,\|\, u) \tag{2.5}$$

where $H(\cdot, \cdot)$ is the standard Shannon's cross-entropy. Note that it is not symmetric. In prior work, $H(y, \sigma)$ is commonly used where the pseudo-label y is taken as the target and we refer to it as CE. Asano et.al. [8] treat the pseudo-label y as the one-hot label and the optimization with respect to the network parameters is the same as that for standard supervised learning. Dizaji et.al. [74] and Jabi et.al. [101] use the soft pseudo-labels as the target where the soft pseudo-labels arise naturally from the optimization. Note that the estimated pseudo-labels are noisy. In Section 2.2, we propose different cross-entropy losses that are more robust to the uncertainty/noise in the soft pseudo-labels than the standard CE. The detailed optimization algorithms with respect to the pseudo-labels can be found in Chapter 3.

2.1.2 Our self-labeling loss in segmentation

As for the scribble-supervised segmentation, we apply ADM splitting to weakly supervised loss

$$-\sum_{i\in S} \ln \sigma_i^{\bar{y}_i} \; + \; \eta \sum_{i\notin S} H(\sigma_i) \; + \; \lambda \sum_{ij\in\mathcal{N}} P(\sigma_i, \sigma_j)$$

to formulate the self-labeling loss as below. Here, the scribbles (seeds) contain a subset of pixels $S \subset \Omega$ that have ground truth labels $\bar{y}_i$ and the first term above is the NLL on seeds. The second term above is the entropy term on unlabeled pixels. This term is inspired by the decisiveness term in clustering and is also common in semi-supervised learning [79]. The third term above is the relaxation of the Potts model. A more detailed introduction to this loss can be found in Chapter 4.

To formulate the self-labeling loss, it is convenient to introduce pseudo-labels y_i on all pixels in Ω even though a subset of pixels (seeds) $S \subset \Omega$ have ground truth labels $\bar{y}_i$. We will simply impose a constraint that pseudo-labels and ground truth labels agree on S. Thus, we assume the following set of pseudo-labels

$$Y_\Omega := \{y_i \in \Delta^K \,|\, i \in \Omega, \text{ s.t. } y_i = \bar{y}_i \text{ for } i \in S\}.$$

We split the terms in the loss above into two groups: one includes NLL and entropy H terms keeping the original prediction variables σ_i and the other includes the Potts relaxation P replacing

σ_i with auxiliary variables y_i. This turns into an expression

$$-\sum_{i\in S} \ln \sigma_i^{\bar{y}_i} \;+\; \eta \sum_{i\notin S} H(\sigma_i) \;+\; \lambda \sum_{ij\in\mathcal{N}} P(y_i, y_j)$$

with equality constraints $\sigma_i = y_i$. The standard approximation is to incorporate constraint $\sigma_i \approx y_i$ directly into the loss, e.g. using KL-divergence. For simplicity, we use weight η for $KL(\sigma_i, y_i)$ to combine it with $H(\sigma_i)$ into a single cross-entropy term

$$-\sum_{i\in S} \ln \sigma_i^{\bar{y}_i} \;+\; \underbrace{\eta \sum_{i\notin S} H(\sigma_i) \;+\; \eta \sum_{i\notin S} KL(\sigma_i, y_i)}_{\eta \sum_{i\notin S} H(\sigma_i, y_i)} \;+ \lambda \sum_{ij\in\mathcal{N}} P(y_i, y_j)$$

defining joint *self-labeling loss* for both predictions σ_i and pseudo-labels y_i

$$L_{self}(\sigma, y) \;\;:=\;\; -\sum_{i\in S} \ln \sigma_i^{\bar{y}_i} \;+\; \eta \sum_{i\notin S} H(\sigma_i, y_i) \;+\; \lambda \sum_{ij\in\mathcal{N}} P(y_i, y_j) \tag{2.6}$$

approximating the original weakly supervised loss. Iterative minimization of this loss w.r.t. predictions σ_i (model parameters training) and pseudo-labels y_i effectively breaks the original optimization problem into two simpler sub-problems, assuming there is a good solver for optimal pseudo-labels. The latter seems plausible since the unary term $H(\sigma_i, y_i)$ is convex for y_i and the Potts relaxations were widely studied in image segmentation for decades.

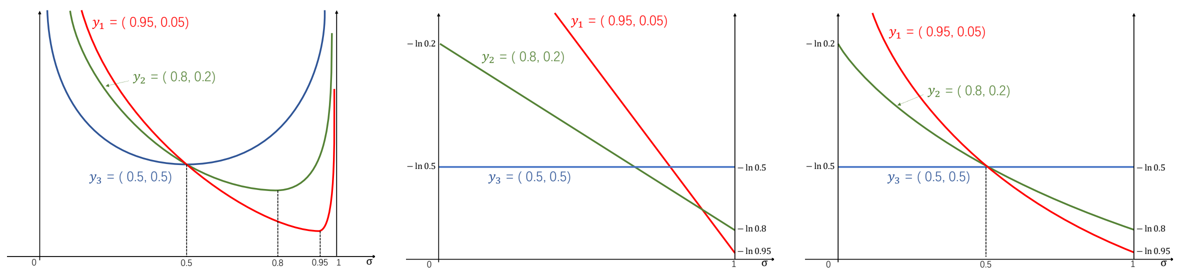
In prior work, Marin et.al. [142, 141] treat the pseudo-labels as hard labels and apply the standard graph cut [25] solver to obtain the pseudo-labels. They also notice that the noise in the estimated pseudo-labels does harm the training of the segmentation networks. Thus, they propose a loss that is robust to the label noise. Instead, we advocate for using soft pseudo-labels where the softness naturally represents the uncertainty/noise. In the following Section, we will introduce new losses that are robust to the noise in the soft pseudo-labels. The corresponding optimization algorithms for pseudo-labels can be found in Chapter 4.

2.2 Cross-entropy functions for soft pseudo-labels

Minimizing divergence enforces proximity between two distributions, which may work as a loss for training model predictions σ with labels y, for example, if y are ground truth one-hot labels. However, if y are pseudo-labels that are estimated jointly with σ, proximity between y and σ is not a good criterion for the loss. For example, highly uncertain model predictions σ in combination with uniformly distributed pseudo-labels y correspond to the optimal zero divergence, but this is not a very useful result for self-labeling. Instead, all existing self-labeling losses for deep clustering minimize Shannon's cross-entropy that reduces the divergence and uncertainty at the same time

$$H(y,\sigma) \;\equiv\; D(y,\sigma) + H(y).$$

The entropy term corresponds to the "decisiveness" constraint in unsupervised discriminative clustering [30, 102, 8, 98, 101]. In general, it is recommended as a regularizer for unsupervised and weakly-supervised network training [79] to encourage decision boundaries away from the data points implicitly increasing the decision margins. Shannon's cross-entropy $H(y,\sigma)$ is the most common loss for training network predictions σ from ground truth labels y in the context of classification, semantic segmentation, etc. However, this loss may not be ideal for applications where the targets y are soft categorical distributions representing various forms of class uncertainty. For example, when the ground truth is not known for most data, and the network training is done jointly with estimating *pseudo-labels* y for the unlabeled data. In this case, soft labels y are



(a) standard $H_{\text{CE}}(y,\sigma)$ (b) reverse $H_{\text{RCE}}(y,\sigma)$ (c) collision $H_{\text{CCE}}(y,\sigma)$

Figure 2.1: Cross-entropy functions: standard (2.7), reverse (2.8), and collision (2.9).

distributions representing class uncertainty. We observe that if such y is used as a target in $H(y,\sigma)$, the network is trained to reproduce the uncertainty, see Figure 2.1(a). This motivates the discussion of alternative "cross-entropy" functions where the quotes indicate an informal interpretation of this information-theoretic concept. Intuitively, such functions should encourage decisiveness, as well as proximity between the predictions and pseudo-labels, but avoid mimicking the uncertainty in both directions: from soft pseudo-labels to predictions and vice-versa. We show that the last property can be achieved in a probabilistically principled manner. The following sections discuss alternative cross-entropy functions that we study in the context of the self-labeling loss.

2.2.1 Standard and reverse cross-entropy

Standard cross-entropy provides the obvious baseline for evaluating two alternative versions that follow. For completeness, we include its mathematical definition

$$H_{\text{CE}}(y_i,\sigma_i) \quad = \quad H(y_i,\sigma_i) \quad \equiv \quad -\sum_k y_i^k \ln \sigma_i^k \tag{2.7}$$

and remind the reader that this loss is primarily used with hard or one-hot labels, in which case it is also equivalent to NLL loss $-\ln \sigma_i^{y_i}$ previously discussed for pixels with ground truth labels (2.6). As mentioned earlier, Figure 2.1(a) shows that for soft pseudo-labels like $y = (0.5, 0.5)$, it forces predictions to mimic or replicate the uncertainty $\sigma \approx y$. In fact, label $y = (0.5, 0.5)$ just tells that the class is unknown and the network should not be supervised by this point. This problem manifests itself in the poor performance of the standard cross-entropy (2.7) in Figure 2.2 and in our experiment discussed in Chapter 3 and 4.

Reverse cross-entropy switches the order of the label and prediction in (2.7)

$$H_{\text{RCE}}(y_i,\sigma_i) \quad = \quad H(\sigma_i,y_i) \quad \equiv \quad -\sum_k \sigma_i^k \ln y_i^k \tag{2.8}$$

which is not too common. Indeed, Shannon's cross-entropy is not symmetric and the first argument is normally the *target* distribution and the second is the *estimated* distribution. However, in our case, both distributions are estimated and there is no reason not to try the reverse order. It is worth noting that our self-labeling formulation (2.6) suggests that reverse cross-entropy naturally appears when the ADM approach splits the decisiveness and fairness into separate sub-problems. Moreover, as Figure 2.1(b) shows, in this case, the network does not mimic uncertain pseudo-labels, e.g. the gradient of the blue line is zero. The results for the reverse cross-entropy in Figure 2.2 and in Chapter 3 and 4 are significantly better than for the standard. Unfortunately, now pseudo-labels y mimic the uncertainty in predictions σ.

2.2.2 Collision cross-entropy

Collision cross-entropy resolves the problem in a principled way. We define it as

$$H_{\text{CCE}}(y_i,\sigma_i) \quad \equiv \quad -\ln \sum_k \sigma_i^k y_i^k \quad \equiv \quad -\ln \sigma^\top y \tag{2.9}$$

which is symmetric w.r.t. pseudo-labels and predictions. The dot product $\sigma^\top y$ can be seen as a probability that random variables represented by the distribution σ, the prediction class C, and the distribution y, the unknown true class T, are equal. Indeed,

$$\Pr(C = T) = \sum_k Pr(C = k) \Pr(T = k) = \sigma^\top y.$$

When training softmax σ with pseudo-label distribution y, the collision event is the exact equality of the predicted class and the pseudo-label, where these are interpreted as specific outcomes for random variables with distributions σ and y. Note that the collision event, i.e. the equality of two random variables, has very little to do with the equality of distributions $\sigma = y$. The collision may happen when $\sigma \neq y$, as long as $\sigma \cdot y > 0$. Vice versa, this event is not guaranteed even when $\sigma = y$. It will happen *almost surely* only if the two distributions are the same one-hot. However, if the distributions are both uniform, the collision probability is only $1/K$. Figure 2.1(c) shows no mimicking of uncertainty (blue line). However, unlike reverse cross-entropy, this is also valid when y is estimated from uncertain predictions σ since (2.9) is symmetric. This leads to the best empirical performance. Our extensive experiments in Chapter 3 and 4 are conclusive that collision cross-entropy is the best option for H in self-labeling loss (2.5) and (2.6).

As easy to check, the collision cross-entropy (2.9) can be equivalently represented as

$$H_{\text{CCE}}(p, q) \;\equiv\; -\ln cos(p, q) \;+\; \frac{H_2(p) + H_2(q)}{2}$$

where $cos(p, q)$ is the cosine of the angle between p and q as vectors in $\mathcal{R}^K$ and $H_2(p) := -\ln \sum_k p_k^2$ is the collision entropy. The first term corresponds to a “distance” between the two distributions: it is non-negative, equals 0 iff $p = q$, and $-\ln cos(\cdot)$ is a convex function of an angle, which can be interpreted as a spherical metric. Thus, analogously to the Shannon’s cross-entropy, H_{CCE} is the sum of divergence and entropy.

The formula (2.9) can be found as a definition of quadratic Rényi cross-entropy [171, 173, 229]. However, we could not identify information-theoretic axioms characterizing a generalized cross-entropy. Rényi himself did not discuss the concept of cross-entropy in his seminal work [176]. Also, two different formulations of “natural” and “shifted” Rényi cross-entropy of arbitrary order could be found in [215, 208]. In particular, the shifted version of order 2 agrees with our formulation of collision cross-entropy (2.9). However, lack of postulates or characterization for the cross-entropy, and the existence of multiple non-equivalent formulations did not give us the confidence to use the name Rényi. Instead, we use “collision” due to its clear intuitive interpretation of the loss (2.9). But, the term “cross-entropy” is used only informally.

The numerical and empirical properties of the collision cross-entropy (2.7) are sufficiently different from the Shannons cross-entropy (2.7). Figure 2.1 illustrates $H_{\text{CCE}}(y, \sigma)$ as a function of σ for different label distributions y. For confident y it behaves the same way as the standard cross

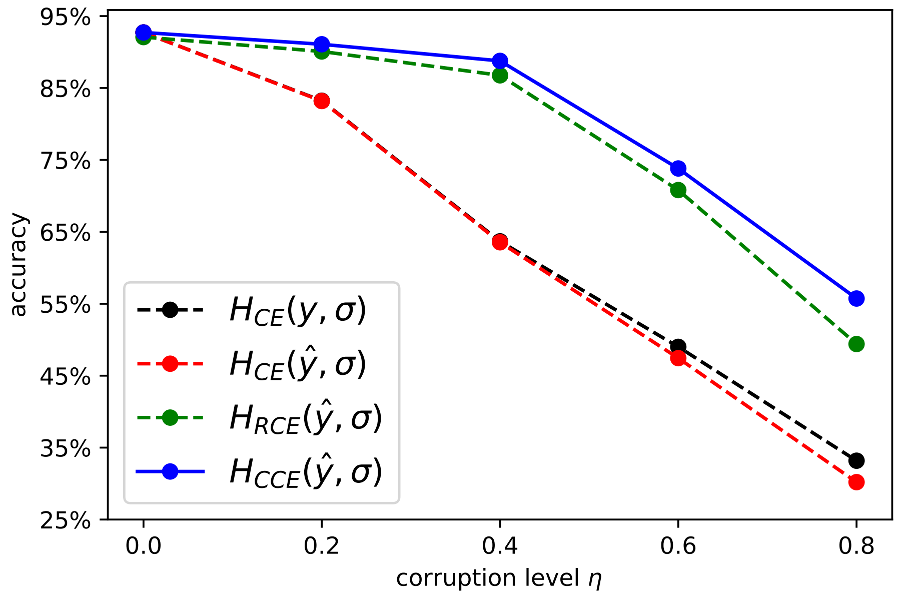


Figure 2.2: Robustness to label uncertainty: forward Shannon cross-entropy (2.7) vs. reverse Shannon cross-entropy (2.8) vs. collision cross-entropy (2.9). The test uses ResNet-18 architecture on fully-supervised *Natural Scene* dataset [154] where we corrupted some labels. The horizontal axis shows the percentage η of training images where the correct ground truth labels were replaced by a random label. All losses trained the model using soft target distributions $\hat{y} = \eta * u + (1-\eta) * y$ representing the mixture of one-hot distribution y for the observed corrupt label and the uniform distribution u, as recommended in [151]. $\hat{y}$ is interpreted as the posterior distribution of the unknown true label given corrupted one-hot y. The vertical axis shows the test accuracy. Training with the reverse and collision cross-entropy is robust to much higher levels of label uncertainty.

entropy $H(y, \sigma)$, but softer low-confident labels y naturally have little influence on the training. In contrast, the standard cross entropy encourages prediction σ to be the exact copy of uncertainty in distribution y. Self-labeling methods based on $H(y, \sigma)$ often "prune out" uncertain pseudo-labels [35]. Collision cross entropy $H_{\text{CCE}}(y, \sigma)$ makes such heuristics redundant. We also demonstrate the "robustness to label uncertainty" on an example where the ground truth labels are corrupted by noise, see Figure2.2. This artificial fully-supervised test is used only to compare the robustness of these cross-entropy terms in complete isolation from other terms in the self-labeling losses.

Soft labels vs noisy labels: Our collision CE for soft labels, represented by distributions y, can be related to loss functions used for supervised classification with *noisy labels* [200, 163, 194], which assume some observed hard target labels l that may not be true due to corruption or "noise".

Instead of our probability of collision

$$\Pr(C=T)=\sum_k \Pr(C=k,T=k)=\sum_k \sigma_k y_k \equiv y^\top \sigma$$

between the predicted class C and unknown true class T, whose distributions are prediction σ and soft target y, they maximize the probability that a random variable L representing a corrupted target equals the observed value l

$$\begin{aligned}\Pr(L=l) &= \sum_k \Pr(L=l|T=k)\Pr(T=k)\\ &\approx \sum_k \Pr(L=l|T=k)\;\sigma^k \;\equiv\; Q_l\,\sigma\end{aligned}$$

where the approximation uses the model predictions σ^k instead of true class probabilities $\Pr(T=k)$, which is a significant assumption. Vector Q_l is the l-th row of the *transition matrix* Q, such that $Q_{lk}=\Pr(L=l|T=k)$, that has to be obtained in addition to hard noisy labels l.

Our approach maximizing the collision probability based on soft labels y is a generalization of the methods for hard noisy labels. Their transitional matrix Q can be interpreted as an operator for converting any hard label l into a soft label $y=Q^\top \mathbf{1}_l=Q_l$. Then, the two methods are numerically equivalent, though our statistical motivation is significantly different. Moreover, our approach is more general since it applies to a wider set of problems where the class target T can be directly specified by a distribution, a soft label y, representing the target uncertainty. For example, in fully supervised classification or segmentation the human annotator can directly indicate uncertainty (odds) for classes present in the image or at a specific pixel. In fact, class ambiguity is common in many data sets, though for efficiency, the annotators are typically forced to provide one hard label.

Chapter 3

Entropy-based clustering

This Chapter is focused on the Mutual Information (MI) principle for clustering. It discusses the relation between generative and discriminative approaches and develops new theories for MI-inspired entropy losses for softmax models. In particular, we prove margin maximization property relating discriminative entropy clustering to unsupervised SVM-based methods [227]. We also develop an efficient EM-based self-labeling algorithm for entropy-based clustering using collision-cross-entropy introduced in the previous Chapter.

3.1 Introduction

Mutual information (MI) was proposed as a criterion for clustering by Bridle et al. [30]. It is motivated as a general information-theoretic measure of the "correlation" between the data X and the class labels C. Starting from MI definition as Kullback–Leibler (KL) divergence between the joint density and the product of the marginals for X and C, Bridle et al.[30] derive a simple clustering loss for softmax models.

The MI criterion also unifies some well-known generative and discriminative approaches to clustering. In particular, consider two equivalent entropy-based MI formulations

$$\begin{aligned} MI(C,X) &= H(X) - H(X\,|\,C) \qquad \text{(gen.)} \qquad (3.1)\\ &\equiv H(C) - H(C\,|\,X) \qquad \text{(disc.)} \qquad (3.2)\end{aligned}$$

where $H(\cdot)$ and $H(\cdot\,|\,\cdot)$ are the *entropy* and the *conditional entropy* functions over distributions corresponding to the random variables X and C. Two terms in (3.2) can directly evaluate classes

C predicted by discriminative posterior models, e.g. softmax models. As detailed in Section 3.1.1, these two terms represent standard clustering criteria, *fairness* and *decisiveness* [30], used for "deep" clustering with neural networks [30, 116, 98, 36, 102, 8, 101]. On the other hand, the equivalent formulation of MI in (3.1) relates to standard generative algorithms for clustering [103, 228, 31] and unsupervised or weakly-supervised image segmentation [238, 34, 180]. Such algorithms minimize the entropy $H(X|C)$ of data X in each cluster C, where fitting density models helps to estimate the entropy. Section 3.2.1 details the relation of criterion (3.1) to the most basic generative clustering algorithm, K-means.

Despite equivalence, criteria (3.1,3.2) can lead to clustering algorithms producing different results depending on their specific generative or discriminative model choices, i.e. *hypothesis spaces*. For example, Figure 3.1 shows optimal solutions for (a) K-means minimizing the *variance* of each cluster, i.e. entropy $H(X|C)$ assuming isotropic Gaussian density, and (b) the linear softmax model minimizing (3.2). While both algorithms make linear decisions, K-means produces compact clusters due to its implicit bias to isotropic Gaussian densities. In contrast, the linear softmax model finds balanced or "fair" clusters with "decisive" decision boundary corresponding to the *maximum margin*, as we later prove in Theorem 3 ($\alpha = 2$). We will revisit Figure 3.1 again.

This chapter's focus is clustering based on softmax models and unsupervised entropy loss functions (3.2) inspired by [30]. We refer to this general group of methods as *discriminative entropy clustering*. The rest of the introduction provides the background and motivation for our work. Sections 3.1.1-3.1.4 present the necessary terminology for the discriminative entropy clustering problem, its regularization, and its *self-labeling* formulations. In particular, Section 3.1.2 discusses the significance of model complexity and data representation. Finally, Section 3.1.5 summarizes our main contributions presented in the main parts of the paper.

3.1.1 Discriminative entropy clustering basics

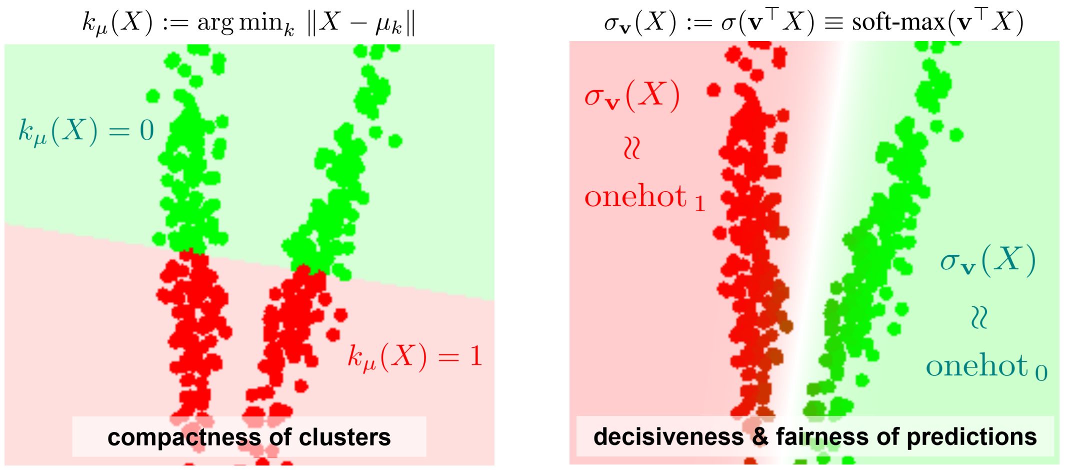
This Section introduces our terminology for entropy-based clustering with softmax models. We consider discriminative classification models that could be linear (single layer) or complex multi-layered, and assume probability-type output often interpreted as a (pseudo-) posterior. Typically, such outputs are produced by the *softmax* function $\sigma : \mathcal{R}^K \to \Delta^K$ mapping K raw classifier outputs $a = (a^1, \dots, a^K)$, e.g. real-valued "logits" or "scores", to K class probabilities

$$\sigma^k(a) \;:=\; \frac{\exp a^k}{\sum_c \exp a^c} \qquad \text{for} \quad 1 \le k \le K$$

forming a categorical distribution $\sigma = (\sigma^1, \dots, \sigma^K) \in \Delta^K$ representing a point on the *probability simplex*

$$\Delta^K \;:=\; \{(p^1, \dots, p^K) \in \mathcal{R}^K \,|\, p^k \ge 0, \sum p^k = 1\}.$$

TWO LINEAR DECISION FUNCTIONS OVER 2D FEATURES $X \in \mathcal{R}^2$


$k_\mu(X) := \arg\min_k \|X - \mu_k\|$
$\sigma_{\mathbf{v}}(X) := \sigma(\mathbf{v}^\top X) \equiv$ soft-max$(\mathbf{v}^\top X)$
$k_\mu(X) = 0$
$k_\mu(X) = 1$
compactness of clusters
$\sigma_{\mathbf{v}}(X) \approx$ onehot $_1$
$\sigma_{\mathbf{v}}(X) \approx$ onehot $_0$
decisiveness & fairness of predictions

(a) variance clustering (K-means) (b) entropy clustering (3.5,3.3)

Figure 3.1: Entropy clustering vs. K-means - binary example ($K = 2$) for 2D data $\{X_i\}$ comparing two linear methods of similar parametric complexity: (a) K-means $\mu_k \in \mathcal{R}^2$ and (b) entropy clustering (3.5) with a linear model (3.3) defined by K-column matrix $\mathbf{v} = [\mathbf{v}_k]$ with linear discriminants $\mathbf{v}_k \in \mathcal{R}^{2+1}$ (incl. bias). Red and green colors in (a) and (b) illustrate decision/prediction functions, $k_\mu(X) := \arg\min_k \|X - \mu_k\|$ and $\sigma_{\mathbf{v}}(X) := \sigma(\mathbf{v}^\top X)$, corresponding to the optimal parameters μ and $\mathbf{v}$ minimizing two losses: (a) compactness or *variance* of clusters $\sum_i \|X_i - \mu_{k_i}\|^2$ where $k_i = k_\mu(X_i)$, and (b) *decisiveness* and *fairness* $\overline{H(\sigma)} - H(\bar{\sigma})$, see (3.5), where $\sigma_i = \sigma(\mathbf{v}^\top X_i)$. Color transparency in (b) visualizes "soft" decisions $\sigma(\mathbf{v}^\top X)$; the linear boundary "blur" is proportional to $\frac{1}{\|v\|}$. Unlike *low-variance* (a), the optimal clusters in (b) have the *maximum margin* among all fair/balanced solutions, assuming "infinitesimal" norm regularization $\|\mathbf{v}\|^2$ discussed in Section 3.2.2.

For shortness, this chapter uses the same symbol for functions and examples of their output, e.g. specific prediction values $\sigma \in \Delta^K$. In particular, σ_X may denote the prediction for any given input X. If i is an integer, σ_i denotes the prediction for the specific example X_i in the training dataset $\{X_i\}_{i=1}^N$.

The simplest *linear* softmax model can be defined as

$$\sigma(\mathbf{v}^\top X) \tag{3.3}$$

where for any input vector X the matrix of parameters $\mathbf{v}$ produces K-logit vector $a = \mathbf{v}^\top X$ mapped to a probability distribution by the softmax function. For shortness, we use a *homogeneous* representation for the linear classifier so that $\mathbf{v}^\top X$ stands for an affine transformation including the *bias*.

More complex non-linear network models compose a linear softmax model (3.3) with some *representation* layers mapping input X to "deep" features $f_\mathbf{w}(X)$

$$\sigma(\mathbf{v}^\top f_\mathbf{w}(X)) \tag{3.4}$$

where the trainable parameters $\mathbf{w}$ of the *embedding function* $f_\mathbf{w}$ are distinguished from the linear classifier parameters $\mathbf{v}$. The linear model (3.3) is a special case of (3.4) for $f(X) = X$. Typical "deep" representation $f_\mathbf{w}(X)$ significantly changes the dimensions of the input X. It is convenient to assume that M always represents the dimensions of the linear head's input, i.e. the (homogeneous) matrix $\mathbf{v}$ has size $(M+1) \times K$.

Assuming softmax models as above, Bridle et al. [30] derive the following clustering loss for data $\{X_i\}_{i=1}^N$

$$L_{ec} \quad := \quad -MI(C,X) \quad \approx \quad \overline{H(\sigma)} \; - \; H(\overline{\sigma}) \tag{3.5}$$

based on the Shannon entropy $H(p) := -\sum_k p^k \ln p^k$ for categorical distributions $p \in \Delta^K$. The average entropy of the model output

$$\overline{H(\sigma)} := \frac{1}{N}\sum_i H(\sigma_i)$$

represents $H(C|X)$ in (3.2). The entropy of the average output

$$H(\overline{\sigma}) \qquad \text{where} \qquad \overline{\sigma} := \frac{1}{N}\sum_i \sigma_i$$

is the entropy of class predictions over the whole dataset corresponding to $H(C)$ in (3.2).

Loss (3.5) is minimized over model parameters $\mathbf{v}$ and $\mathbf{w}$ in (3.3) or (3.4), e.g. by *gradient descent* or *backpropagation* [181]. Larger entropy $H(\overline{\sigma})$ encourages "fair" predictions with a balanced support of all categories across the whole dataset, while smaller $\overline{H(\sigma)}$ encourages confident or "decisive" prediction at each data point suggesting that decision boundaries are away from the training examples [79].

3.1.2 Model complexity and representation learning

Criterion (3.5) provides strong constraints for well-regularized or simple parametric models. For example, in the case of linear softmax models (3.3), we prove the margin-maximizing property, see Fig.3.1(b), relating (3.5) to SVM clustering [227].

Clustering algorithms for softmax models can also be motivated by the powerful representation of data behind the deep network models (3.4). Some criteria related to MI are also studied in the context of *representation learning* [20][1], [92, 158, 210, 8], but this is not our focus. We study (3.2,3.5) as a clustering criterion where decisions C are optimized for fixed data X. Nevertheless, this approach applies to complex networks (3.4) where internal layers can be seen as responsible for representation $f_{\mathbf{w}}(X)$. Our experiments with networks do not evaluate the quality of representation separately from clustering and view the internal layers mainly as an integral part of a complex model. Instead, we are concerned with regularization of complex models in the context of clustering.

3.1.3 Regularized entropy clustering

Bridle & McKay [30] argue that MI maximization may allow arbitrarily complex solutions for under-regularized network models, as illustrated in Figure 3.2(a) for (3.4). They note that

> ”... *[MI] could be maximized by any implausible classification of the input... when we use more complex models. This criterion encourages... objective techniques for regularising classification networks... [138, 137].*”

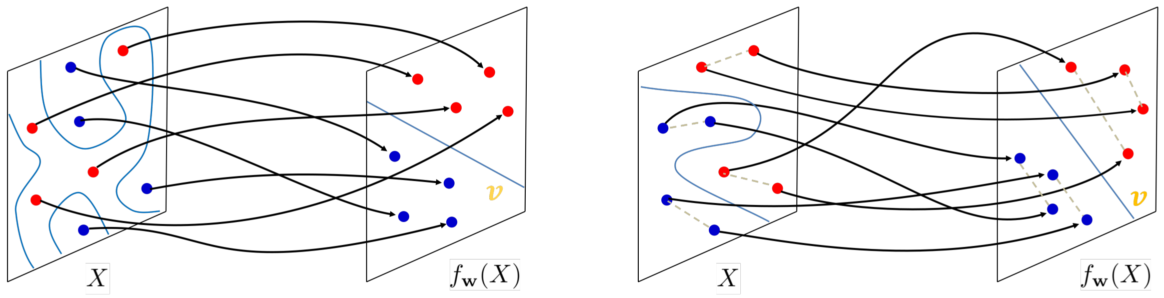
For example, a Bayesian approach to network regularization [138] combines training losses with the squared L_2 norm of all network weights interpreted as a *log-prior* or *weight energy*. Following [138] and [116], regularized version of the entropy clustering loss (3.5) incorporates the norm of network parameters motivated as their isotropic Gaussian prior

$$\begin{aligned} L_{mi+decay} &= \overline{H(\sigma)} \;-\; H(\overline{\sigma}) \;+\; \|[\mathbf{v},\mathbf{w}]\|^2 \\ &\overset{c}{=} \overline{H(\sigma)} \;+\; KL(\overline{\sigma}\,\|\,u) \;+\; \|[\mathbf{v},\mathbf{w}]\|^2 \end{aligned} \tag{3.6}$$

where $\overset{c}{=}$ represents equality up to an additive constant and u is a uniform distribution over K classes. The equivalent loss formulation (3.6) uses KL divergence motivated in [116] by the possibility to generalize the fairness constraint to any target balancing distribution different from the uniform.

Unsupervised representation learning techniques [92, 158, 44, 89] are also relevant as mechanisms for constraining the network. In particular, *self-augmentation* techniques are widely used for both clustering and representation learning [98, 102, 8]. For example, maximization of MI between predictions for input X and its augmentation X' can improve representation [102].

[1]Boudiaf et al. [20] also discuss *generative* and *discriminative* “views” on MI exactly as in (3.1-3.2), but focus on **supervised** representation learning where features X are optimized assuming given class labels C.



(a) under-regularized model ⇒ random clusters (b) self-augmentation ⇒ regular clusters, weak isometry

Figure 3.2: Model regularization is required: (a) arbitrarily complex model (3.4) can create any random clusters of the input $\{X_i\}$ despite the linear partitioning in the space of deep features $f_{\mathbf{w}}(X)$. Self-augmentation loss (3.7) regularizes the clusters (b), as well as the mapping f. Indeed, similar inputs $\{X, X'\}$ are mapped to features equidistant from the decision boundary. Stronger forms of isometry can be enforced by *contrastive losses* [47, 188, 193, 44], particularly if negative pairs are also available. In general, model regularization with domain-specific constraints or augmentation is important for the quality of clustering.

For large network models employed in this work, we use only generic network regularization techniques based on squared L_2 norm of the network weights (3.6) and a standard self-augmentation loss [98, 8, 45] directly enforcing consistent clustering $\sigma_X \approx \sigma_{X'}$ for input pairs $\{X, X'\}$

$$L_{sa} \;=\; \sum_{\{X,X'\}\in\mathcal{N}_a} KL(\sigma_X \| \sigma_{X'}) + KL(\sigma_{X'} \| \sigma_X) \tag{3.7}$$

where $\mathcal{N}_a$ is the set of all pairs of augmented examples. This loss implies that similar inputs are mapped to deep features equidistant from the decision boundary. We refer to this as *weak isometry* between the space of inputs X and the embedding space of "deep" features $f_{\mathbf{w}}(X)$, see Figure 3.2(b).

3.1.4 Self-labeling methods for entropy clustering

Optimization of losses (3.5) or (3.6) during network training could be done with standard gradient descent or backpropagation [30, 116, 98]. However, the difference between the two entropy terms implies non-convexity presenting challenges for the gradient descent. This motivates alternative approaches to optimization. It is common to approximate (3.5) with some *surrogate loss* incorporating auxiliary or hidden variables y representing *pseudo-labels* for unlabeled data points X, which are estimated jointly with optimization of the network parameters [74, 8, 101].

Typically, such *self-labeling* approaches to entropy clustering iteratively optimize the surrogate loss over pseudo-labels and network parameters, similarly to Lloyd's algorithm for K-means or EM algorithm for Gaussian mixtures [16]. While the network parameters are still optimized via gradient descent, the pseudo-labels can be optimized via more powerful algorithms.

For example, [8] formulate self-labeling using the following constrained optimization problem with discrete pseudo-labels y tied to predictions by *cross entropy* $H(y, \sigma)$

$$L_{ce} \quad = \quad \overline{H(y,\sigma)} \qquad s.t. \quad y \in \Delta^K_{0,1} \quad and \quad \bar{y} = u \tag{3.8}$$

where $\Delta^K_{0,1}$ are *one-hot* distributions, i.e. the vertices of the probability simplex Δ^K. Besides proximity between σ and y, the cross-entropy in (3.8) encourages the decisiveness, while the fairness is represented by the hard constraint $\bar{y} = u$. Assuming fixed pseudo-labels y, the network training is done by minimizing the standard cross entropy loss $H(y, \sigma)$ convex w.r.t. σ. Then, model predictions are fixed and (3.8) is minimized over y. Note that $H(y, \sigma)$ is linear with respect to y and its minimum over simplex Δ^K is achieved by one-hot distributions corresponding to $\arg\max_k(\sigma)$ at each data point. However, the "fairness" constraint $\bar{y} = u$ converts minimization of the cross-entropy loss over all pseudo-labels y into a non-trivial integer programming problem that can be approximately solved via *optimal transport* [53].

Self-labeling methods for entropy clustering can also use "soft" pseudo-labels $y \in \Delta^K$ as targets in $H(y, \sigma)$. In general, soft target distributions y are common in the context of noisy labels [203, 194] and network calibration [82, 151]. They can also improve generalization by reducing over-confidence [166]. In the context of entropy clustering, soft pseudo-labels y correspond to a *relaxation* of cross-entropy. Nevertheless, cross-entropy encourages decisive pseudo-labels y since $H(y, \sigma) \equiv H(y) + KL(y\|\sigma)$. The last term also implies proximity $\sigma \approx y$ and, therefore, the decisiveness of predictions σ. Many soft self-labeling methods [74, 101] represent the fairness constraint using $-H(\bar{y})$ or $KL(\bar{y} \,\|\, u)$, as in (3.6). In particular, [101] formulates the following entropy-based soft self-labeling loss

$$L_{ce+kl} \quad = \quad \overline{H(y,\sigma)} \quad - \quad H(\bar{y}) \tag{3.9}$$

representing the decisiveness and the fairness constraints. Similarly to (3.8), the network parameters in (3.9) are trained by the standard cross-entropy loss $H(y, \sigma)$ when y are fixed. Optimization over the relaxed pseudo-labels $y \in \Delta^K$ is relatively easy since negative entropy is convex and cross-entropy is linear w.r.t. y. While there is no closed-form solution, the authors offer an efficient approximate solver.

3.1.5 Summary of our contributions

This chapter provides new theories and algorithms for discriminative entropy clustering. First, we examine its conceptual relations to K-means and SVM. We disprove the theories in [101] on the equivalence between the soft K-means and the linear case of entropy clustering (3.5,3.3). Figures 3.1, 3.3, 3.4, 3.8 study a counterexample and Section 3.2.1 points out specific technical errors in their proof. Despite the equivalence (3.1) $\equiv$ (3.2), two discriminative and generative clustering algorithms can operate within different hypothesis spaces even if both produce linear results. Further contradicting equivalence to K-means, our theories in Section 3.2.2 prove that linear entropy clustering (3.5,3.3) has a *margin-maximizing* property establishing a formal relation to SVM-based clustering [13, 227]. In the general context of entropy clustering (3.5) with deep models (3.4), our results imply margin maximization in the *embedding space*, similar to kernel SVM [52]. Such non-linear methodologies, however, can not use arbitrary embeddings. Indeed, SVM should restrict kernels (implicit embeddings) and networks should regularize (explicit) embedding functions $f_{\mathbf{w}}(X)$.

Second, our Section 3.3 propose a *zero-avoiding* form of KL-divergence as a stronger fairness term that does not tolerate trivial clusters, unlike the standard fairness in (3.9). Our self-labeling losses with reverse cross-entropy and collision cross-entropy as introduced in Chapter 2 are convex w.r.t. y and allow efficient EM solvers for pseudo-labels. The new losses along with the new EM algorithms improve the state-of-the-art on many standard benchmarks for deep clustering, as shown in Section 3.4 empirically confirming our technical insights.

3.2 Linear entropy clustering

This Section analyses the theoretical properties of entropy clustering loss (3.5) in the context of linear discriminative models (3.3). Even such simple models may require some regularization to avoid degenerate clusters. Section 3.2.2 shows a form of regularization implying *margin maximization*. But first, Section 3.2.1 juxtaposes (3.5,3.3) and K-means as representative *linear* cases of discriminative and generative clustering (3.1,3.2).

3.2.1 Relation to K-means

There are many similarities between entropy clustering (3.5) with linear model (3.3) and K-means. Both produce linear boundaries and use (nearly) the same number of parameters, $K \times (M+1)$ vs. $K \times M$. The former corresponds to K linear discriminants $\mathbf{v}_k$ (w. bias) forming the columns of matrix v in (3.3), and the latter corresponds to K means μ_k representing density models at each cluster. Both approaches have good approximation algorithms for their non-convex (3.5) and NP-hard [139] objectives. Two methods also generalize to non-linear clustering using more complex representations, e.g. learned $f_w(X)$ or implicit (kernel K-means).

There is a limited understanding of the differences between linear entropy clustering and K-means as most prior literature, including [30], discusses (3.5) in the context of networks (3.4). One 2D linear example in [116] (Fig.1) helps, but unlike our Figure 3.1, they employ trivial compact clusters typical for the textbook's illustrations of K-means. The two methods are indistinguishable from the example in [116]. Moreover, there is a prior theoretical claim [101] about equivalence between soft K-means and linear entropy clustering, assuming certain regularization. We disprove this claim later in this Section.

Generative and discriminative formulations of MI (3.1,3.2) may also suggest the equivalence of linear entropy clustering and K-means. We already discussed how (3.5) relates to (3.2), and now we show how K-means relates to (3.1). Indeed, the Lloyd's algorithm for K-means minimizes the following self-labeling objective for hard pseudo-labels $y \in \Delta_{0,1}^K$ and parameters μ_k

$$\begin{aligned} L_{k\mu} \quad &:= \quad \sum_k \sum_i y_i^k \|X_i - \mu_k\|^2 \\ &\overset{c}{=} \quad -\sum_k \sum_i y_i^k \ln N_{\mu_k}(X_i) \\ &\approx \quad \sum_k |X^k| \; H(X^k, N_{\mu_k}) \end{aligned} \tag{3.10}$$

where each point X_i contributes a squared distance to the mean μ_k of the assigned cluster such that $y_i^k = 1$. Isotropic Gaussian densities N_μ with covariances $\Sigma_k = I/2$ allow an equivalent formulation on the second line below (3.10). Its Monte Carlo approximation on the third line produces an expression with cross-entropy where X^k represents the "true" density of data in cluster k and $|X^k|$ is its cardinality. The standard relationship between cross-entropy and entropy functions further implies an inequality

$$\sum_k |X^k| \; H(X^k, N_{\mu_k}) \quad \geq \quad \sum_k |X^k| \; H(X^k) \quad \propto \quad H(X|C)$$

concerning the conditional entropy in (3.1). The last $\propto$ relation only ignores a constant factor, the whole dataset cardinality. When recomputing cluster means, Lloyd's algorithm minimizes the cross-entropy above. It achieves its low bound, the entropy in the second expression, only when the clusters are isotropic Gaussian blobs, which is an implicit assumption in K-means. This is when K-means works well and, as we just showed, when it approximates generative MI clustering

(3.1). Thus, K-means and linear entropy clustering are equivalent in the case of isotropic Gaussian clusters. This equivalence is consistent with the toy example in [116].

The arguments above also suggest how the equivalence may fail for more complex clusters. Indeed, our Figure 3.1 demonstrates no equivalence without isotropy. We can also refute the general claim in [101] about the equivalence between the following *soft* variant of K-means loss (3.10)

$$L_{sk\mu} := \overline{\overline{y\|X-\mu\|^2}} \;-\; \gamma\overline{H(y)} \;-\; \overline{\|X\|^2} \tag{3.11}$$

and the linear case of their self-labeling entropy clustering formulation (3.9,3.3) with linear classifier's norm regularization

$$L_{ce+} := \overline{H(y,\sigma)} \;-\; H(\overline{y}) \;+\; \gamma\|\mathbf{v}\|^2 \tag{3.12}$$

where the *single bar* operator averaging over data points i was introduced in Section 3.1.1, and the *double bar* represents averaging over both i and k shortening the expression for loss $L_{k\mu}$ in (3.10). The negative entropy in (3.11) encourages soft labeling, i.e. $y_i \in \Delta^K$ could be any categorical distribution. This term is standard for *soft K-means* formulations. The last term in (3.11) is a constant needed in the equivalence claim [101].

The only difference between the entropy clustering losses (3.9) and (3.12) is the squared norm of the linear classifier parameters $\|\mathbf{v}\|$, excluding the bias. This standard regularization encourages "softness" of the linear classifier's soft-max predictions, similar to the effect of the entropy term in soft K-means (3.11). However, unlike entropy clustering[2], soft K-means fails on elongated clusters in Figures 3.3 and 3.4 in the same way as the basic K-means in Figure 3.1. Indeed, soft formulations of K-means are typically motivated by a different problem - overlapping clusters. The proper generative mechanism to address anisotropic clusters is to drop the constraint on the covariance matrices $\Sigma_k \sim I$ allowing a wider class of Gaussian density models, i.e. extending the *hypothesis space*. For example, GMM can be viewed as an anisotropic extension of soft K-means, and it would work perfectly in Figures 3.1, 3.3, 3.4, 3.8, similar to entropy clustering.

We also have a general argument for why linear entropy clustering is stronger than K-means. As easy to check using Bayes formula, the posterior for isotropic Gaussian densities estimated by K-means is consistent with model (3.3). However, discriminative entropy clustering optimizes (3.3) without any assumptions on densities, implying a larger hypothesis space.

Besides our counterexample in Figure 3.3 and the general argument above, we found a critical error in [101]. They ignore the normalization/denominator in the definition of softmax.

[2]The entropy clustering results in the toy examples in Figs. 3.1, 3.3, 3.4, 3.8 do not depend on the specific formulation. Self-labeling algorithm [101] for (3.12) and our entropy clustering algorithm in Section 3.3 produce the same results. For simplicity, these Figures use basic gradient descent for the regularized version of generic linear entropy clustering (3.5,3.3), as in (3.14).

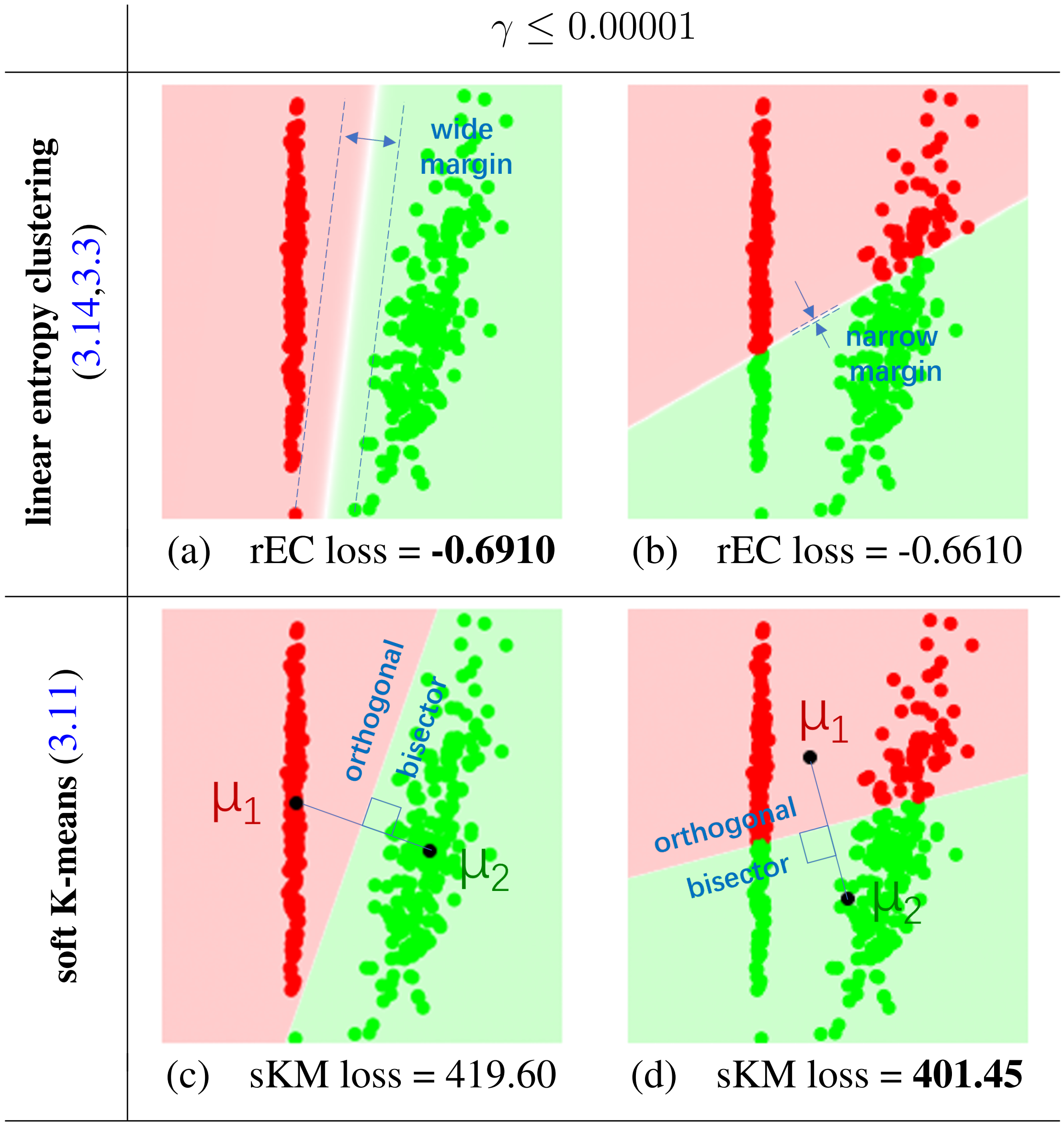


Figure 3.3: Global & local minima: linear *regularized entropy clustering* (rEC) versus *soft K-means* (sKM). For both losses, global optima (a) and (d) are consistent for all $\gamma \in (0, 0.00001]$. Variations in the optimal loss values are negligible. sKM is nearly identical to hard K-means for such γ; it softens only for larger γ, see Figure 3.4. The local minimum for sKM (c) is obtained by Lloyd's algorithm initialized at (a). Vice-versa, gradient descent for rEC converges to (a) from (c). The same "cross-check" works for (b) and (d). Local minima for rEC (a,b) are balanced clusterings with (locally) maximum margins. In contrast, local minima for sKM (c,d) are *orthogonal bisectors* for the cluster centers. K-means ignores the margins.

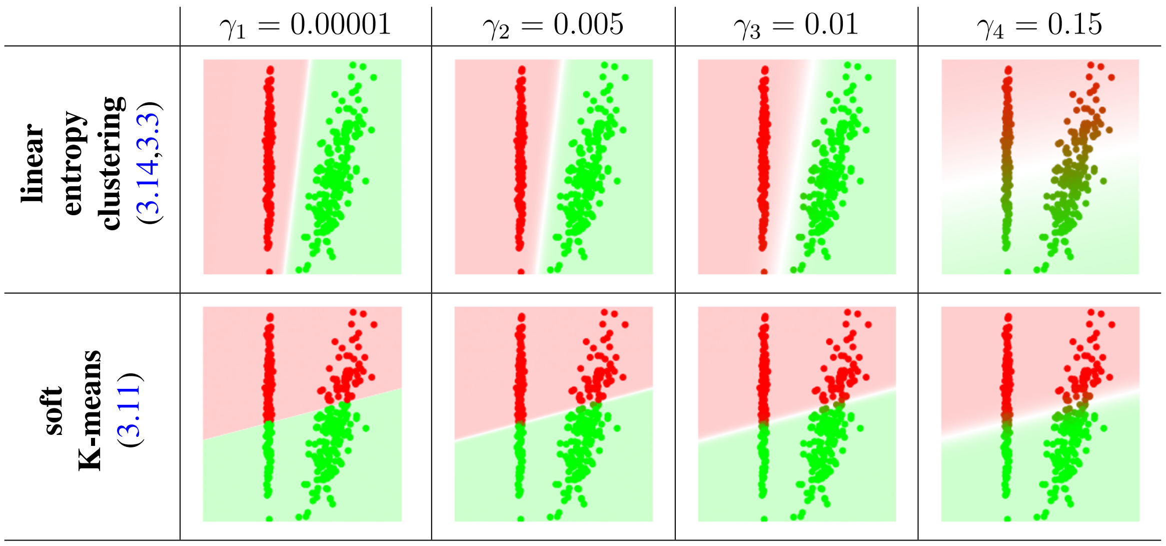


Figure 3.4: Global minima for various γ: linear *regularized entropy clustering* (rEC) versus *soft K-means* (sKM). As $\gamma \to 0$, both approaches converge to hard, but different, clusters. Optimal/low-variance sKM clusters are consistent for all γ. In contrast, rEC produces max-margin clusters for small γ and changes the solution for larger γ_4. The latter reduces norm $\|\mathbf{v}\|$ implying a wider "indecisiveness" zone around the linear decision boundary. Due to the decisiveness term in (3.5), entropy clustering finds the boundary minimizing the overlap between the data and such "softness" zone, explaining the result for γ_4.

Symbol $\propto$ hides it in their equation (5), which is treated as equality in the proof of Proposition 2. Their proof does not work with the normalization, which is important for training softmax models. The regularization term $\|\mathbf{v}\|^2$ and constant $\|X\|^2$ play the following algebraic role in their proof of equivalence between (3.12) and (3.11). Without softmax normalization, $\ln \sigma$ inside cross-entropy $H(y, \sigma)$ in (3.12) turns into a linear term w.r.t. logits $\mathbf{v}^\top X$ and combining it with $\|\mathbf{v}\|^2$ and $\|X\|^2$ creates a quadratic form $\|X - \mathbf{v}\|^2$ resembling squared errors in K-means (3.11). In contrast, Section 3.2.2 shows that regularization $\|\mathbf{v}\|^2$ in (3.12) is needed for the *margin maximization* property of entropy clustering illustrated in Figures 3.3(a) and 3.8(b). Our theories show that instead of K-means, in general, linear entropy clustering is related to discriminative SVM clustering [13, 227].

3.2.2 Margin maximization

This section establishes a *margin-maximizing* property of the regularized decisiveness for linear model (3.3)

$$\gamma\|\mathbf{v}\|^2 \; + \; \overline{H(\sigma)} \tag{3.13}$$

where $\|\mathbf{v}\|$ is L_2 norm of the linear discriminant w/o the bias. Without extra constraints, the decisiveness allows a trivial solution with a single cluster. This can be avoided by focusing on a set of *feasible* solutions, where feasibility can represent balanced clusters or consistency with partial data labels as in *semi-supervised learning*.

Our theory assumes an arbitrary set of feasible solutions but is also relevant for soft feasibility. For example, the regularized entropy clustering loss for linear model (3.3)

$$L_{rec} := \overline{H(\sigma)} \; - \; H(\overline{\sigma}) \; + \; \gamma\|\mathbf{v}\|^2 \tag{3.14}$$

combines the margin-maximizing decisiveness with the fairness term encouraging balanced clusters, see Fig. 3.3(a). The direct relation to MI [30] provides an information-theoretic motivation for this loss, as discussed in Section 3.1.1. Our theories below extend the general conceptual understanding of the decisiveness, entropy clustering, and establish their relation to unsupervised SVM methods [13, 227].

Overview of max-margin clustering

As typical in SVM, our formal theories on the max-margin property of the regularized decisiveness (3.13) in Sections 3.2.2-3.2.2 are mainly focused on the binary linear models, though multi-class extensions are possible, see Sec. 3.2.2. Below we informally preview our max-margin property claims about (3.13) providing some intuition that may be helpful due to differences with standard margin maximization in SVM.

For example, basic SVM assumes *linearly separable* data in a fully supervised setting. *Soft SVM* extension allows a trade-off between margin maximization and separation violations. In contrast, *linear separability* is irrelevant in clustering as there are no ground-truth labels to be violated. Yet, the "gap" can be measured between any pair of clusters, see Fig. 3.3(a) and (b). Thus, the largest margin solution among all feasible (fair) linear clusterings is well-defined, see Fig. 3.3(a).

SVM clustering [227] uses the *hinge loss* instead of decisiveness in (3.13) and also assumes fairness, see Sec.3.2.2. In both cases, however, linear classifier regularization $\gamma\|\mathbf{v}\|^2$ is important for margin maximization. The regularized hinge loss outputs a max-margin solution for all γ

below some threshold. In contrast, our theories show that optimal solutions for the regularized decisiveness (3.13) converge to a max-margin clustering only as $\gamma \to 0$, see Figure 3.4, though this subtle difference may be hard to discern in practice.

Our results are based on the max-margin theories for supervised losses in [179]. We generalize them to unsupervised clustering problems. Also, instead of (3.13), we prove the max-margin property for a larger class of regularized decisiveness measures using *Renyi* entropy R_α [176] of any order $\alpha > 0$ (formally defined in the following Sections)

$$\gamma\|\mathbf{v}\|^2 \; + \; \overline{R_\alpha(\sigma)} \tag{3.15}$$

where Shannon's entropy in (3.13) is a special case $H = R_1$. Focusing on binary clustering, Section 3.2.2 establishes our results for $\alpha = \infty$. In this base case, Renyi entropy R_∞ can be seen as a "self-labeled" *logistic loss* allowing us to prove our result in Theorem 2 by extending the standard max-margin property for logistic regression [179]. Theorem 3 in Section 3.2.2 extends our results for (3.15) to all $\alpha > 0$, including (3.13) as a special case. Then, Section 3.2.2 discusses the multi-class case $K > 2$. Finally, Section 3.2.2 summarizes margin maximization by regularized entropy (3.15) and juxtaposes it with the regularized hinge loss in SVM clustering [227].

Terminology and Renyi decisiveness for $K = 2$

We extend the earlier notation, in part to suit the binary case. For example, besides *softmax* output σ, we also use a scalar *sigmoid* function ς. Below, we summarize our notation. In particular, there is one significant modification specific to this theoretical Section 3.2.2. Here we use only hard class indicators, thus it is convenient to have integer labels y instead of categorical distributions, as in the rest of the paper. This change should not cause ambiguity as labels y appear mainly in the proofs and their type is clear from the context.

Our notation concerning labels y, labelings $\mathbf{y}$, and linear classifiers $\mathbf{v}$ is mostly general. Some parts specific to $K = 2$ are designed to transition smoothly to multi-class problems. Note that for $K > 2$ linear classifiers typically generate K logits $\mathbf{v}^\top X \in R^K$, output a K-categorical distribution using softmax $\sigma(\mathbf{v}^\top X) \in \Delta^K$, and use natural numbers as class labels. In contrast, binary classifiers typically use a single logit, *sigmoid* output, and class labels $\{\pm 1\}$ or $\{0, 1\}$. Our notation for $K = 2$ rectifies the differences using natural class indicators $\{1, 2\}$ and categorical distribution output $\sigma = (\sigma^1, \sigma^2) = (\varsigma, 1 - \varsigma)$. The latter is justified by a simple property that a single-logit sigmoid classifier is equivalent to a softmax for two logits as only their difference matters.

- $\mathbf{v}$ - linear classifier is a vector for $K = 2$. It generates scalar raw output $\mathbf{v}^\top X$. Vector $\mathbf{v}$ includes the *bias* as we assume “homogeneous” data representation X.

- $\|\mathbf{v}\|$ - L_2 norm that excludes the bias.

- $\sigma = (\sigma^1, \sigma^2) := (\varsigma, 1 - \varsigma)$ - soft-max for $K = 2$ can be represented using a scalar *sigmoid* function

$$\varsigma(x) \;:=\; (1 + e^{-x})^{-1} \;\equiv\; \frac{e^{x/2}}{e^{x/2} + e^{-x/2}} \;\in\; [0, 1]$$

  for $x = \mathbf{v}^\top X$. Similarly to σ, we use symbol ς both for specific values in $[0, 1]$ and as a function name.

- $y \in \{1, 2\}$ - binary class indicator/label

- y^σ - hard class label produced by soft prediction σ

$$y^\sigma \;:=\; \arg\max_k \sigma^k \quad \text{(a.k.a. “\textit{hard-max}”)}$$

- y_X - class label at arbitrary data point X

- $y_i = y_{X_i}$ - class label at point X_i

- $\mathbf{y} := \{y_i\}_{i=1}^N$ - labeling of the whole dataset $\{X_i\}_{i=1}^N$

- $y_X^{\mathbf{v}} := y^{\sigma(\mathbf{v}^\top X)}$ - hard class label produced by classifier $\mathbf{v}$ for an arbitrary data point X.

- $y_i^{\mathbf{v}} := y_{X_i}^{\mathbf{v}}$ - hard label produced by $\mathbf{v}$ for point X_i.

- $\mathbf{y}^{\mathbf{v}} := \{y_i^{\mathbf{v}}\}_{i=1}^N$ - dataset labeling by classifier $\mathbf{v}$

- $\mathbf{V}^{\mathbf{y}} := \{\mathbf{v} \,|\, \mathbf{y}^{\mathbf{v}} = \mathbf{y}\}$ - a set of all linear classifiers consistent with given labeling $\mathbf{y}$. By default, this chapter uses “weak consistency” allowing data points on the decision boundary, i.e. zero margins. Set $\mathbf{V}^{\mathbf{y}}$ is non-empty only if labeling $\mathbf{y}$ is (weakly) linearly separable.

- $\mathcal{FL}$ - a set of feasible labelings $\mathbf{y}$ representing allowed clusterings. It could be arbitrary. One example is a set of all “fair” linearly separable clusterings.

- $\mathbf{V}^{\mathcal{FL}} := \cup_{\mathbf{y} \in \mathcal{FL}} \mathbf{V}^{\mathbf{y}}$ - a set of all linear classifiers $\mathbf{v}$ consistent with allowed labelings $\mathbf{y}$ in $\mathcal{FL}$.

- $\mathbf{u}$ - unit norm linear classifier such that $\|\mathbf{u}\| = 1$

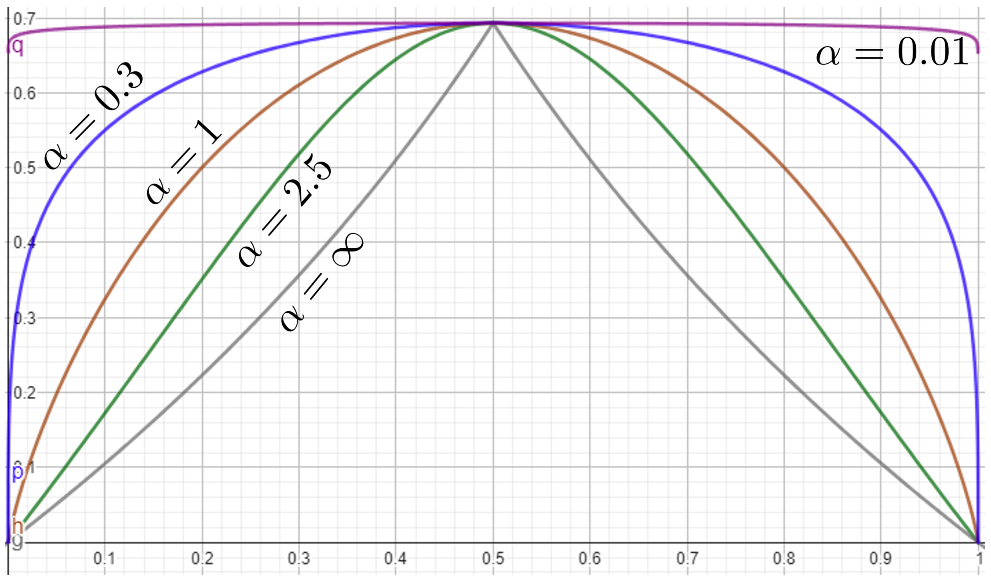


Figure 3.5: Renyi entropy [176] of order α: assuming $K = 2$ the plots above show $R_\alpha(\sigma)$ in (3.16) for binary distributions $\sigma = (\varsigma, 1-\varsigma)$ as functions of scalar $\varsigma \in [0,1]$. Shannon entropy (3.19) is a special case corresponding to $\alpha = 1$.

- $\mathbf{U}^{\mathbf{y}} := \{\mathbf{v} \,|\, \mathbf{y}^{\mathbf{v}} = \mathbf{y},\ \|\mathbf{v}\| = 1\}$ - a set of all unit-norm linear classifiers consistent with given labeling $\mathbf{y}$.
- $\mathbf{U}^{\mathcal{FL}} := \cup_{\mathbf{y}\in\mathcal{FL}} \mathbf{U}^{\mathbf{y}}$ - a set of all unit-norm linear classifiers consistent with labelings in $\mathcal{FL}$.
- $\mathbf{u}^{\mathbf{y}}$ - maximum margin linear classifier of unit norm corresponding to given linearly separable labeling $\mathbf{y}$

Finally, we define the binary version of Renyi entropy R_α [176] of order $\alpha \geq 0$ studied by our max-margin clustering theories for $K = 2$ in Sec. 3.2.2- 3.2.2. One-to-one correspondence between distributions $\sigma = (\sigma^1, \sigma^2) = (\varsigma, 1-\varsigma)$ and scalars ς makes it convenient to define for $K = 2$ two equivalent functions $R_\alpha(\sigma)$ and $R_\alpha(\varsigma)$

$$R_\alpha(\sigma) \;\equiv\; R_\alpha(\varsigma) \;:=\; \frac{\ln(\varsigma^\alpha + (1-\varsigma)^\alpha)}{1-\alpha}. \tag{3.16}$$

The expression in (3.16) is numerically ill-conditioned when $\alpha \in \{0, 1, \infty\}$. In these three cases, Renyi entropy is better defined by the asymptotically consistent formulations

$$
\begin{aligned}
R_0(\sigma) = R_0(\varsigma) &= \ln 2 \cdot [0 < \varsigma < 1] && (3.17)\\
R_1(\sigma) = R_1(\varsigma) &= H(\sigma) \quad \text{(binary Shannon entropy)} &&\\
R_\infty(\sigma) = R_\infty(\varsigma) &= -\ln \max\{\varsigma, 1-\varsigma\} && (3.18)
\end{aligned}
$$

where $[\cdot]$ is *Iverson bracket* indicator returning 1 or 0 depending if the condition inside is true or not. For $\alpha = 1$ one can use the standard formula for the binary Shannon entropy

$$H(\sigma) \;\equiv\; H(\varsigma) \;=\; -\varsigma \ln \varsigma - (1-\varsigma)\ln(1-\varsigma). \qquad (3.19)$$

that has no numerical issues. Figure 3.5 illustrates binary Renyi entropy functions R_α for different orders α.

Max-margin theories for binary R_∞ decisiveness

Our base case is Renyi entropy R_∞ for $K = 2$. As evident from (3.18), it resembles the standard *negative log-likelihood* (NLL) loss for supervised classification, a.k.a. *logistic regression*. To emphasize this relation, we denote supervised NLL loss by $R_\infty(\sigma \,|\, y)$. For $\sigma = (\sigma^1, \sigma^2) = (\varsigma, 1-\varsigma)$ it is

$$R_\infty(\sigma \,|\, y) \;:=\; -\ln \sigma^y \;\equiv\; \begin{cases} -\ln \varsigma & \text{if } y = 1 \\ -\ln(1-\varsigma) & \text{if } y = 2 \end{cases} \qquad (3.20)$$

where y is a given binary label at a data point. The obvious relation between the entropy (3.18) and NLL (3.20), see Fig. 3.7(a),

$$R_\infty(\sigma) \;=\; \min\{\, R_\infty(\sigma \,|\, 1),\; R_\infty(\sigma \,|\, 2)\,\} \qquad (3.21)$$

can be equivalently represented as "self-labeling" identity

$$R_\infty(\sigma) \;\equiv\; R_\infty(\sigma \,|\, y^\sigma) \qquad (3.22)$$

where y^σ is the "hard-max" label for prediction σ.

We exploit the "self-labeling" identity (3.22) to prove the max-margin clustering property for entropy R_∞ by extending the known margin-maximizing property for logistic regression [179] reviewed below.

Max-margin in logistic regression: With some exceptions [13, 227], margin maximization is typically discussed in a fully supervised context assuming a given linearly separable labeling. The max-margin property is easy to prove for the *hinge loss* [217] as it vanishes above some threshold. Some monotone-decreasing losses, e.g. *logistic loss*, also have max-margin property, but the proof requires careful analysis [179]. They prove a sufficient condition, “fast-enough” decay, for monotone non-increasing classification losses that depend only on distances to classification boundary, a.k.a. *margins*.

The standard logistic loss is defined w.r.t. raw classifier output $x = \mathbf{v}^\top X$ as

$$r(x) \quad := \quad \ln(1+e^{-x}) \quad \equiv \quad -\ln(\varsigma(x)) \tag{3.23}$$

which is a composition of NLL (3.20) and sigmoid function. Given true label y, the corresponding supervision loss is

$$R_\infty(\sigma(\mathbf{v}^\top X)\,|\,y) \quad \equiv \quad r(d_X^{\mathbf{v}}(\mathbf{y})\|\mathbf{v}\|) \tag{3.24}$$

where

$$d_X^{\mathbf{v}}(y) \quad := \quad \frac{\mathbf{v}^\top X}{\|\mathbf{v}\|} \cdot [y=1]^{\pm} \tag{3.25}$$

is a signed *margin*, i.e. distance from point X to the classification boundary. The operator $[\cdot]^{\pm}$ returns $+1$ if the argument is true and -1 otherwise[3]. As easy to check, the sign of distance $d_X^{\mathbf{v}}(y)$ in (3.25) is positive for correctly classified points X, and it is negative otherwise, see cyan color in Figure 3.6(a).

Logistic regression is known to satisfy conditions for margin maximizing classification [179].

PROPERTY 1 (**exponential decay for r**). Logistic loss $r(x)$ in (3.23) satisfies the following “fast-enough” decay condition

$$\lim_{x\to\infty} \frac{r(x\cdot(1-\epsilon))}{r(x)} = \infty, \qquad \forall\epsilon$$

which is a sufficient condition for margin-maximizing classification. This is case "$T=\infty$" in Theorem 2.1 from [179].

For completeness, we also state a special case of Theorem 2.1 in [179] for logistic regression. We use the following total loss formulation based on NLL (3.20)

$$R_\infty(\mathbf{v}|\mathbf{y}) := -\overline{\ln\sigma^{y_X}(\mathbf{v}^\top X)} \tag{3.26}$$

where the bar indicates averaging over all data points $\{X_i\}$.

[3]Theories in [179] use binary labels $y \in \{\pm 1\}$ simplifying the signed margin expression in (3.25) to $d_X^{\mathbf{v}}(y) = \frac{y\,\mathbf{v}^\top X}{\|\mathbf{v}\|}$. Instead, our natural labels $y \in \{1,\cdots,K\}$ simplify notation for prediction σ^y at each class y.

Theorem 1 (**max-margin for logistic regression [179]).** Consider any given linearly separable labeling $\mathbf{y} := \{y_i\}_{i=1}^N$ for dataset $\{X_i\}_{i=1}^N$. Assuming linear classifier $\mathbf{v}(\gamma)$ minimizes regularized logistic loss over classifiers $\mathbf{v} \in \mathbf{V}^\mathbf{y}$ consistent with some given (e.g. ground truth) labeling $\mathbf{y}$

$$\mathbf{v}(\gamma) \quad := \quad \arg\min_{\mathbf{v}\in\mathbf{V}^\mathbf{y}} \gamma\|\mathbf{v}\|^2 + R_\infty(\mathbf{v}|\,\mathbf{y})$$

then

$$\frac{\mathbf{v}(\gamma)}{\|\mathbf{v}(\gamma)\|} \quad \xrightarrow{\gamma\to 0} \quad \mathbf{u}^\mathbf{y}$$

where $\mathbf{u}^\mathbf{y}$ is a unit-norm max-margin linear classifier for $\mathbf{y}$.

Max-margins for R_∞-decisiveness: To establish the margin-maximizing property for clustering, we should drop full supervision where true labeling $\mathbf{y}$ is given. Instead, we formulate max-margin clustering for a given set $\mathcal{FL}$ of allowed or *feasible* linearly separable binary labelings $\mathbf{y}$ for dataset $\{X_i\}_{i=1}^N$. In this unsupervised context, labelings $\mathbf{y} \in \mathcal{FL}$ represent possible data clusterings. For example, $\mathcal{FL}$ could represent "fair" clusterings[4]. As a minimum, we normally assume that linearly separable labelings in $\mathcal{FL}$ exclude trivial solutions with empty clusters.

Naturally, labelings $\mathbf{y} \in \mathcal{FL}$ divide the data into clusters differently. Thus, the corresponding inter-cluster gaps vary.

DEFINITION 1. Assume $\mathbf{y} = \{y_i\}_{i=1}^N$ is a binary linearly separable labeling of the dataset $\{X_i\}_{i=1}^N$. Then, the *gap* or *margin size* for labeling (clustering) $\mathbf{y}$ is defined as

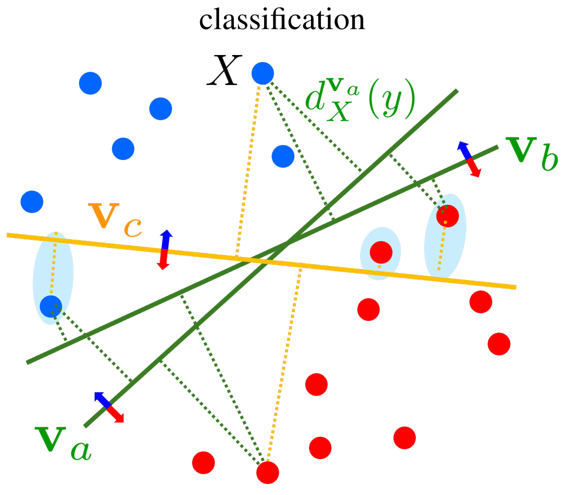
$$|\mathbf{y}| \quad := \quad \max_{\mathbf{u}\in\mathbf{U}^\mathbf{y}} \min_i \; |\mathbf{u}^\top X_i| \quad \equiv \quad \min_i \; |(\mathbf{u}^\mathbf{y})^\top X_i|$$

where $\mathbf{u}^y$ is the max-margin unit-norm linear classifier for $\mathbf{y}$.

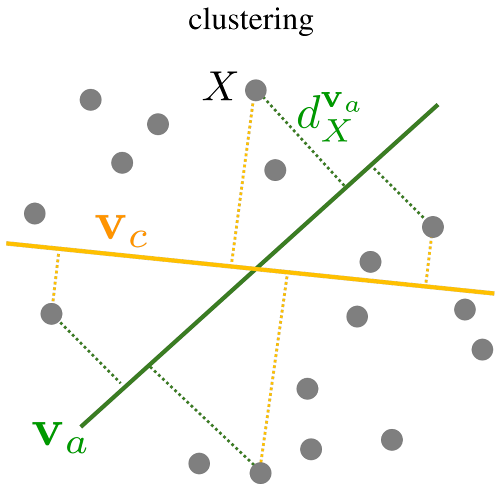
This definition associates the term *margin size* with labelings. Alternatively, *margin size* is often associated with classifiers. Given a separable labeling $\mathbf{y}$, any consistent linear classifier $\mathbf{u} \in \mathbf{U}^\mathbf{y}$ has margin size $min_i|\mathbf{u}^\top X_i|$. The *margin size* for labeling $\mathbf{y}$ in Definition 1 is the margin size of the optimal max-margin classifier $\mathbf{u}^\mathbf{y}$.

Now we define *max-margin clustering* in an unsupervised setting restricted to $\mathcal{FL}$. It extends the standard concept of *max-margin classification* restricted to one (true) labeling $\mathbf{y}$. Our focus is on optimal labelings rather than classifiers.

[4]The exact definition of "fair" clustering is irrelevant here.

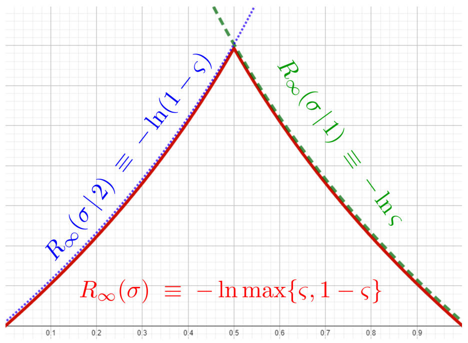


(a) signed margins (3.25)

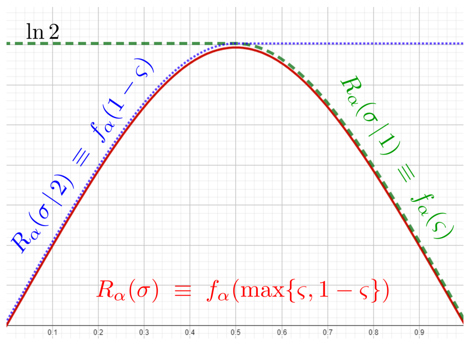


(b) unsigned margins (3.29)

Figure 3.6: Margins: both examples (a) and (b) use distinct colors for classifiers partitioning the data differently. (a) Point colors show given true labeling $\mathbf{y}$. Green classifiers $\mathbf{v} \in \mathbf{V}^{\mathbf{y}}$ are consistent with $\mathbf{y}$ and produce only non-negative margins $d_X^{\mathbf{v}}(y) \geq 0$ in (3.25). Yellow classifier $\mathbf{v}_c \notin \mathbf{V}^{\mathbf{y}}$ has inconsistent labeling $\mathbf{y}^{\mathbf{v}_c} \neq \mathbf{y}$. It has negative (cyan) margins $d_X^{\mathbf{v}_c}(y) < 0$ at incorrectly classified points. In clustering (b), any classifier (green or yellow) produces only non-negative margins $d_X^{\mathbf{v}}(\mathbf{y}^{\mathbf{v}}) \equiv d_X^{\mathbf{v}} \geq 0$ in (3.29) w.r.t. its labeling/clustering $\mathbf{y}^{\mathbf{v}}$.



(a) entropy $R_\infty(\sigma)$ and logistic regression $R_\infty(\sigma \,|\, y)$



(b) entropy $R_\alpha(\sigma)$ and supervised loss $R_\alpha(\sigma \,|\, y)$ for $\alpha = 2$

Figure 3.7: Renyi entropies $R_\alpha(\sigma)$ and their supervised counterparts $R_\alpha(\sigma \,|\, y)$: assuming $K = 2$ the plots above visualize these functions defined in (3.16)-(3.18) and (3.32) for binary distributions $\sigma = (\varsigma, 1-\varsigma) \in \Delta^2$ plotted over interval $\varsigma \in [0, 1]$

DEFINITION 2. Consider any set $\mathcal{FL}$ of allowed linearly separable binary labelings $\mathbf{y}$ for dataset $\{X_i\}_{i=1}^N$. Then, *max-margin clustering* for $\mathcal{FL}$ is defined as

$$\hat{\mathbf{y}} \quad := \quad \arg\max_{\mathbf{y}\in\mathcal{FL}} |\mathbf{y}|$$

where $|\mathbf{y}|$ is the gap size for clustering $\mathbf{y}$, as in Definition 1.

The achieved maximum gap value defining $\hat{\mathbf{y}}$ is finite. Indeed, gap sizes for $\mathbf{y} \in \mathcal{FL}$ are uniformly bounded by the diameter of the dataset $\{X_i\}_{i=1}^N$

$$|\mathbf{y}| \quad \leq \quad \max_{1\leq i<j\leq N} \|X_i - X_j\| \qquad \forall \mathbf{y} \in \mathcal{FL}$$

assuming that $\mathcal{FL}$ excludes trivial clusterings.

Our next theorem extends Theorem 1 to clustering based on R_∞-decisiveness. We define the total decisiveness as

$$R_\infty(\mathbf{v}) \;:=\; \overline{R_\infty(\sigma(\mathbf{v}^\top X))} \;\equiv\; -\overline{\ln \sigma^{y_X^{\mathbf{v}}}(\mathbf{v}^\top X)} \tag{3.27}$$

where the bar indicates averaging over all data points $\{X_i\}$. The proof uses the equivalent expression on the right, which is a self-labeling version of NLL (3.26) based on (3.22) and (3.20).

Theorem 2 (max-margin clustering for R_∞). Consider any set $\mathcal{FL}$ of allowed or feasible linearly separable binary labelings $\mathbf{y}$ for dataset $\{X_i\}_{i=1}^N$, e.g. restricted to “fair” clusterings of the data. Assuming linear classifier $\mathbf{v}(\gamma)$ minimizes regularized decisiveness over classifiers $\mathbf{v} \in \mathbf{V}^{\mathcal{FL}}$ consistent with $\mathcal{FL}$

$$\mathbf{v}(\gamma) \quad := \quad \arg \min_{\mathbf{v} \in \mathbf{V}^{\mathcal{FL}}} \gamma \|\mathbf{v}\|^2 + R_\infty(\mathbf{v})$$

then

$$\frac{\mathbf{v}(\gamma)}{\|\mathbf{v}(\gamma)\|} \quad \xrightarrow{\gamma \to 0} \quad \mathbf{u}^{\hat{\mathbf{y}}}$$

for the maximum margin clustering $\hat{\mathbf{y}}$ in $\mathcal{FL}$, as in Definition 2.

Proof. The proof requires analysis of the loss $R_\infty(\sigma(\mathbf{v}^\top X))$, but due to identity (3.22), it can follow the same steps as the analysis of the supervised loss (3.24) from Theorem 2.1 in [179] (case "a"). Indeed, they consider supervised classification with a given true linearly separable labeling $\mathbf{y}$, but it appears in their proof only in the margin expression $d_X^{\mathbf{v}}(y)$ setting its right sign, see (3.25). Their analysis is restricted to classifiers consistent with $\mathbf{y}$, but we observe that their supervised loss (3.24) has an equivalent *constrained* formulation

$$R_\infty(\sigma(\mathbf{v}^\top X) \,|\, y) \quad \equiv \quad r(d_X^{\mathbf{v}} \|\mathbf{v}\|) \qquad \text{for } \mathbf{v} \in \mathbf{V}^{\mathbf{y}} \tag{3.28}$$

using a simpler unsigned expression for the margins

$$d_X^{\mathbf{v}} \;:=\; \frac{|\mathbf{v}^\top X|}{\|\mathbf{v}\|} \tag{3.29}$$

as (3.25) is guaranteed to be non-negative for all consistent classifiers, see Figure 3.6(a). Our formulation of $R_\infty(\cdot \,|\, y)$ in (3.28) does not depend on labeling $\mathbf{y}$, as as long as $\mathbf{v} \in \mathbf{V}^{\mathbf{y}}$.

In the context of clustering, i.e. unsupervised classification without given (true) labeling $\mathbf{y}$, we show that entropy values $R_\infty(\sigma(\mathbf{v}^\top X))$ relate to logistic loss and distances to the clustering boundary, a.k.a. *margins*. Indeed, as implied by self-labeling identity (3.22) and (3.28), we get

$$R_\infty(\sigma(\mathbf{v}^\top X)) \quad \equiv \quad r(d_X^{\mathbf{v}} \|\mathbf{v}\|) \qquad \text{for any } \mathbf{v} \tag{3.30}$$

for any linear classifier $\mathbf{v}$, which is always consistent with its "hard-max" labeling $\mathbf{y}^{\mathbf{v}}$, by definition. Our equation (3.30) uses the unsigned expression for margins (3.29), see Figure 3.6(b).

The formal max-margin analysis for classifiers $\mathbf{v} \in \mathbf{V}^{\mathbf{y}}$ in [179] uses only logistic loss values, which we expressed as (3.28). The same analysis directly applies to the identical decisiveness values (3.30) for classifiers $\mathbf{v} \in \mathbf{V}^{\mathcal{FL}}$, even when such classifiers may have different labelings $\mathbf{y}^{\mathbf{v}} \in \mathcal{FL}$. $\square$

Generalization for $a > 0$

Binary Renyi entropy $R_\alpha(\sigma) \equiv R_\alpha(\varsigma)$, defined in (3.16-3.19) for distribution $\sigma = (\varsigma, 1-\varsigma)$, is symmetric $R_\alpha(\varsigma) = R_\alpha(1-\varsigma)$ around $\varsigma = 0.5$ for any order $\alpha \geq 0$, see Figure 3.5. We define a monotone non-increasing function, see Figure 3.7(b)

$$f_\alpha(\varsigma) \;:=\; \begin{cases} R_\alpha(\varsigma) & \text{if } \varsigma \geq 0.5 \\ \ln 2 & \text{if } \varsigma \leq 0.5 \end{cases} \tag{3.31}$$

that can be used for training softmax classifier $\sigma(\mathbf{v}^\top X)$ based on the following supervision loss

$$R_\alpha(\sigma \,|\, y) \;:=\; f_\alpha(\sigma^y). \tag{3.32}$$

Function f satisfies the following identities, see Figure 3.7(b),

$$R_\alpha(\sigma) \;\equiv\; \min\{f_\alpha(\varsigma), f_\alpha(1-\varsigma)\} \;\equiv\; f_\alpha(\max\{\varsigma, 1-\varsigma\})$$

analogous to the relation (3.21) between $R_\infty(\sigma)$ and logistic regression, see Figure 3.7(a). As in (3.22), the identities above imply the "self-labeling" identity

$$R_\alpha(\sigma) \;=\; R_\alpha(\sigma \,|\, y^\sigma) \tag{3.33}$$

between decisiveness $R_\alpha(\sigma)$ and the supervised loss (3.32).

Similarly to Section 3.2.2, we will prove the margin maximization property for decisiveness $R_\alpha(\sigma)$ using the self-labeling relation (3.33) and the standard theories for supervised losses [179]. Instead of logistic loss $r(x)$ in (3.23), we use

$$\begin{aligned} g_\alpha(x) \quad &:= \quad f_\alpha(\varsigma(x)) \\ &\overset{(3.31,3.16)}{=} \begin{cases} \frac{\ln\left((1+e^{-x})^{-\alpha} + (1+e^{x})^{-\alpha}\right)}{1-\alpha} & \text{if } x \geq 0 \\ \ln 2 & \text{if } x \leq 0 \end{cases} \end{aligned} \tag{3.34}$$

which is a monotone non-increasing supervision loss enabling an expression for $R_\alpha(\sigma|y)$ (3.32) analogous to (3.24)

$$R_\alpha(\sigma(\mathbf{v}^\top X) \,|\, y) \;\equiv\; g_\alpha(d_X^{\mathbf{v}}(\mathbf{y})\|\mathbf{v}\|) \tag{3.35}$$

where $d_X^{\mathbf{v}}(\mathbf{y})$ are (signed) margins (3.25). Loss $g_\alpha(x)$ satisfies the "fast-enough decay" condition from Theorem 2.1 in [179].

PROPERTY 2 (**exponential decay for g_α**). For any positive value of parameter $\alpha > 0$, loss function g_α in (3.34) satisfies

$$\lim_{x\to\infty} \frac{g_\alpha(x\cdot(1-\epsilon))}{g_\alpha(x)} = \infty, \qquad \forall\epsilon$$

which is the sufficient condition for margin-maximizing classification according to Theorem 2.1 from [179].

Proof. With loss of generality, we only prove the case when $x \to +\infty$. First, let's define

$$e^{x_\epsilon} := e^{x\cdot(1-\epsilon)} \equiv \frac{e^x}{e^{\epsilon x}}.$$

Then when $x \to +\infty$, we have:

$$\begin{aligned} \frac{g_\alpha(x\cdot(1-\epsilon))}{g_\alpha(x)} &= \frac{\ln((1+e^{-x_\epsilon})^{-\alpha} + (1+e^{x_\epsilon})^{-\alpha})}{\ln((1+e^{-x})^{-\alpha} + (1+e^{x})^{-\alpha})} \\ &= \frac{\ln(1-\alpha e^{-x_\epsilon} + e^{-\alpha x_\epsilon})}{\ln(1-\alpha e^{-x} + e^{-\alpha x})} \\ &= \frac{-\alpha e^{-x_\epsilon} + e^{-\alpha x_\epsilon}}{-\alpha e^{-x} + e^{-\alpha x}} \\ &= e^{\epsilon x} \; \frac{\alpha e^{-x} - e^{(\alpha-1)\epsilon x - \alpha x}}{\alpha e^{-x} - e^{-\alpha x}} \end{aligned} \tag{3.36}$$

From (3.36), we will only keep the dominant terms in both nominator and denominator for different α. Note that when $0 < \alpha < 1$, we have

$$(3.36) = e^{\epsilon x} \; \frac{-e^{(\alpha-1)\epsilon x - \alpha x}}{-e^{-\alpha x}} = e^{\alpha\epsilon x} \to \infty.$$

When $\alpha \geq 1$, we have

$$(3.36) = e^{\epsilon x} \; \frac{\alpha e^{-x}}{\alpha e^{-x}} = e^{\epsilon x} \to \infty.$$

Hence, when $\alpha > 0$, the limitation goes to infinity and the proof is complete. □

Property 2 allows us to generalize our max-margin clustering Theorem 2 to the total R_α-decisiveness loss

$$\begin{aligned} R_\alpha(\mathbf{v}) \;&:=\; \overline{R_\alpha(\sigma(\mathbf{v}^\top X))} \;\equiv\; \overline{R_\alpha(\sigma(\mathbf{v}^\top X)\,|\,y_X^{\mathbf{v}})} \\ &\equiv\; \overline{f_\alpha(\sigma^{y_X^{\mathbf{v}}}(\mathbf{v}^\top X))} \end{aligned} \tag{3.37}$$

where f_α replaces "$-\ln$" in the *self-labeling NLL* loss (3.27).

Theorem 3 (**max-margin clustering for R_α**)**.** Consider any set $\mathcal{FL}$ of allowed linearly separable binary clusterings/labelings $\mathbf{y}$ for dataset $\{X_i\}_{i=1}^N$, e.g. restricted to "fair" clusterings. Given any $\alpha > 0$, assume that $\mathbf{v}(\gamma)$ minimizes regularized decisiveness over linear classifiers $\mathbf{v} \in \mathbf{V}^{\mathcal{FL}}$ consistent with $\mathcal{FL}$

$$\mathbf{v}(\gamma) \quad := \quad \arg \min_{\mathbf{v}\in\mathbf{V}^{\mathcal{FL}}} \gamma\|\mathbf{v}\|^2 + R_\alpha(\mathbf{v}).$$

Then

$$\frac{\mathbf{v}(\gamma)}{\|\mathbf{v}(\gamma)\|} \quad \xrightarrow{\gamma\to 0} \quad \mathbf{u}^{\hat{\mathbf{y}}}$$

for the maximum margin clustering $\hat{\mathbf{y}}$ in $\mathcal{FL}$, as in Definition 2.

Proof. We follow the same arguments as in the proof of Theorem 2. Equation (3.35) and self-labeling relation (3.33) imply identical expressions for R_α-values w.r.t. unsigned margins $d_X^{\mathbf{v}} = \frac{|\mathbf{v}^\top X|}{\|\mathbf{v}\|}$ in two cases: (supervision) for classifiers $\mathbf{v} \in \mathbf{V}^{\mathbf{y}}$ consistent with some given (true) labeling $\mathbf{y}$

$$R_\alpha(\sigma(\mathbf{v}^\top X)\,|\,y) \quad \equiv \quad g_\alpha(d_X^{\mathbf{v}}\|\mathbf{v}\|) \qquad\qquad \text{for } \mathbf{v} \in \mathbf{V}^{\mathbf{y}}$$

and (decisiveness) for any classifier

$$R_\alpha(\sigma(\mathbf{v}^\top X)) \quad \equiv \quad g_\alpha(d_X^{\mathbf{v}}\|\mathbf{v}\|) \qquad\qquad \text{for any } \mathbf{v}$$

which are analogous to logistic loss (3.28) and decisiveness (3.30). The only difference between these supervised and unsupervised cases is the set of allowed classifiers. Since classification loss $R_\alpha(\mathbf{v}\,|\,\mathbf{y})$ and decisiveness $R_\alpha(\mathbf{v})$ have identical dependence on unsigned margins $d_i^{\mathbf{v}} = \frac{|\mathbf{v}^\top X_i|}{\|\mathbf{v}\|}$, all arguments from Theorem 2.1 [179] proving the margin maximizing property for the (exponential-decay) classification loss $R_\alpha(\mathbf{v}\,|\,\mathbf{y})$ directly apply to the clustering loss $R_\alpha(\mathbf{v})$. □

Note that there is no margin maximization for $\alpha = 0$ since $R_0(\sigma(\mathbf{v}^\top X)) = const$ for any finite data point X.

Generalization for $K > 2$

We use the max-margin property for multi-label classification in Theorem 4.1 [179] to prove the max-margin property for clustering with regularized R_α-decisiveness when $K > 2$. We need to introduce some additional terminology.

First, the regularization for max-margin K-clustering is based on the squared Frobenius norm of classifier matrix $\mathbf{v}$

$$\|\mathbf{v}\|_{\mathcal{F}}^2 := \sum_{k=1}^{K} \|\mathbf{v}_k\|^2$$

where $\|\mathbf{v}_k\|$ are L_2 norms of linear discriminators $\mathbf{v}_k$ for each class k, which are the columns of matrix $\mathbf{v}$. As before, we exclude the bias parameters in the norms of $\mathbf{v}_k$.

We use the unit-norm multi-class classifiers consistent with any given labeling $\mathbf{y}$

$$\mathbf{U}^{\mathbf{y}} := \{\mathbf{v} \,|\, \mathbf{y}^{\mathbf{v}} = \mathbf{y},\ \|\mathbf{v}\|_{\mathcal{F}} = 1\}$$

as in the binary case in Section 3.2.2. The next definition of margin size for multi-class labelings is consistent with [179].

DEFINITION 3. Assume $\mathbf{y} = \{y_i\}_{i=1}^N$ is a linearly separable multi-class labeling for $\{X_i\}_{i=1}^N$ such that $y_i \in \{1, \cdots, K\}$. Then, the *gap* or *margin size* for labeling/clustering $\mathbf{y}$ is

$$\begin{aligned} |\mathbf{y}| &:= \max_{\mathbf{u}\in\mathbf{U}^{\mathbf{y}}} \min_i \min_{k\neq y_i} |(\mathbf{u}_{y_i} - \mathbf{u}_k)^\top X_i| \\ &\equiv \min_i \min_{k\neq y_i} |(\mathbf{u}^{\mathbf{y}}_{y_i} - \mathbf{u}^{\mathbf{y}}_k)^\top X_i| \end{aligned}$$

where $\mathbf{u}^{\mathbf{y}}$ is the max-margin unit-norm linear classifier for $\mathbf{y}$.

The multi-class margin in Definition 3 is determined by the two closest classes/clusters. The gaps between the other pairs of clusters are ignored as long as they are larger.

DEFINITION 4. Consider any set $\mathcal{FL}$ of allowed linearly separable multi-class labelings $\mathbf{y}$ for dataset $\{X_i\}_{i=1}^N$ where $y_i \in \{1, \cdots, K\}$. Then, *max-margin clustering* for $\mathcal{FL}$ is

$$\hat{\mathbf{y}} := \arg\max_{\mathbf{y}\in\mathcal{FL}} |\mathbf{y}|$$

where $|\mathbf{y}|$ is the gap size for clustering $\mathbf{y}$, as in Definition 3.

Figure 3.8(b) is an example of multi-class clustering achieving the max-margin (Def. 3) among fair solutions. The tightest spot between the red and blue clusters determines $|\hat{\mathbf{y}}|$.

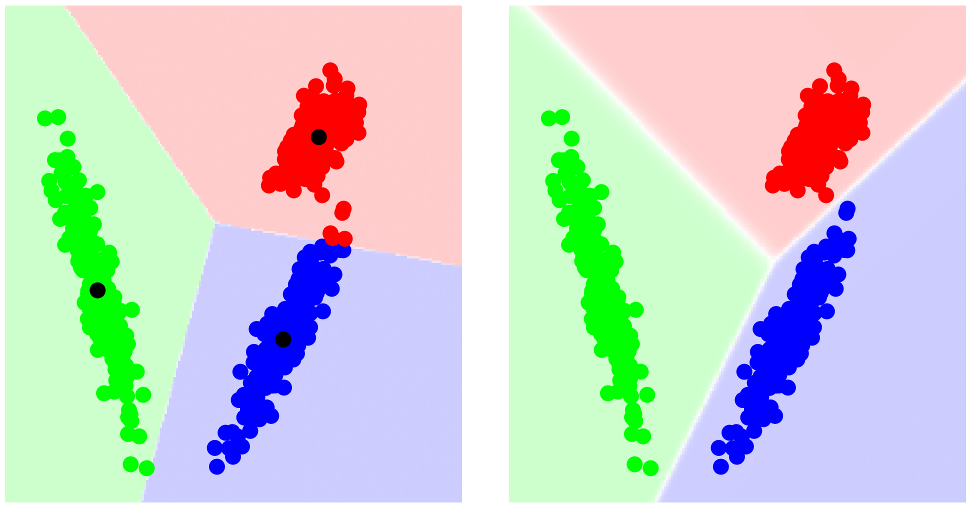
We state max-margin clustering theorem for multi-class decisiveness of order α consistent with the binary case (3.37)

$$R_\alpha(\mathbf{v}) := \overline{R_\alpha(\sigma(\mathbf{v}^\top X))}$$

using the general definition of Renyi entropy [176] for $K \geq 2$

$$R_\alpha(\sigma) := \frac{\ln \sum_{k=1}^K (\sigma^k)^\alpha}{1-\alpha} \tag{3.38}$$

extending (3.16) to multi-class predictions $\sigma = (\sigma^1, \cdots, \sigma^K)$.



(a) soft K-means (3.11) (b) entropy clustering (3.14)

Figure 3.8: Multi-label clustering: K-means vs. entropy clustering. Consistently with Theorem 4, loss (3.14) produces max-margin clustering, as in Definitions 3-4, satisfying the fairness.

Theorem 4 (max-margin multi-class clustering for R_α). Consider any set $\mathcal{FL}$ of allowed or feasible linearly separable multi-class labelings $\mathbf{y}$ for $\{X_i\}_{i=1}^N$, e.g. restricted to "fair" clusterings. Given any order $\alpha > 0$, assume that $\mathbf{v}(\gamma)$ minimizes regularized decisiveness over linear classifiers $\mathbf{v} \in \mathbf{V}^{\mathcal{FL}}$ consistent with $\mathcal{FL}$

$$\mathbf{v}(\gamma) \quad := \quad \arg \min_{\mathbf{v} \in \mathbf{V}^{\mathcal{FL}}} \gamma \|\mathbf{v}\|_{\mathcal{F}}^2 + R_\alpha(\mathbf{v}).$$

Then

$$\frac{\mathbf{v}(\gamma)}{\|\mathbf{v}(\gamma)\|} \quad \xrightarrow{\gamma \to 0} \quad \mathbf{u}^{\hat{\mathbf{y}}}$$

for the maximum margin clustering $\hat{\mathbf{y}}$ in $\mathcal{FL}$, as in Definition 4.

Idea of proof: Definition 3 above is consistent with multi-class L_2-margin (10) in [179]. Their multi-class max-margin classification Theorem 4.1 extends to clustering with regularized decisiveness $R_\alpha(\mathbf{v})$ due to self-labeling relation (3.33), which also works for $K > 2$ with a properly constructed multi-class supervised loss $R_\alpha(\sigma \,|\, y)$. For example, generalizing (3.31,3.32) to the case $K > 2$, one can use supervised loss

$$R_\alpha(\sigma \,|\, y) \;\; := \begin{cases} R_\alpha(\sigma) & \text{if } y^\sigma = y \\ \ln K & \text{o.w.} \end{cases} \tag{3.39}$$

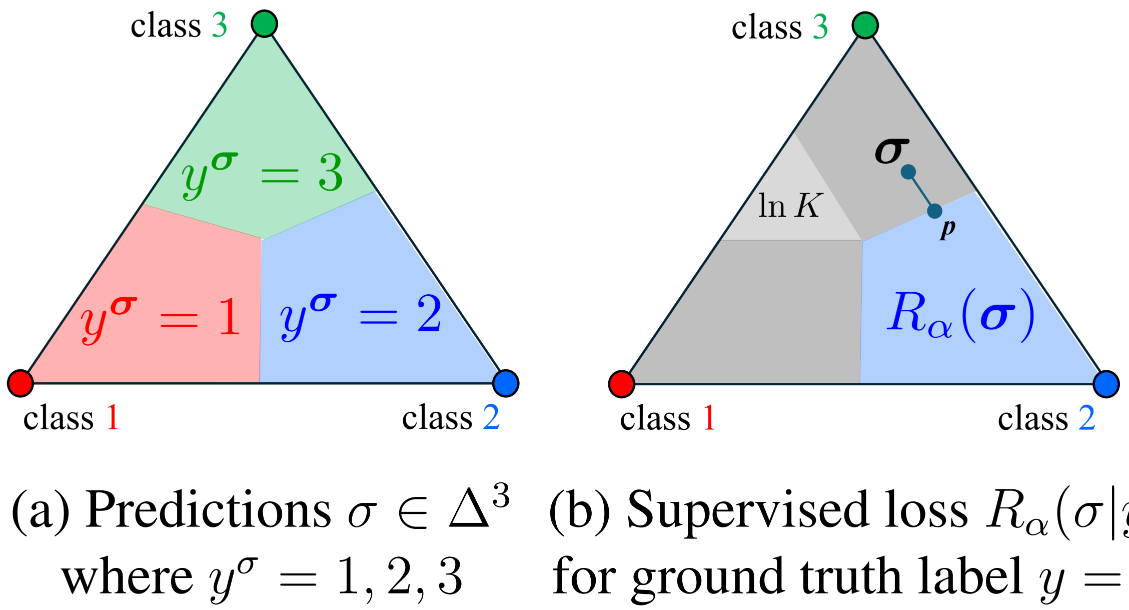

class 3
$y^{\boldsymbol{\sigma}} = 3$
$y^{\boldsymbol{\sigma}} = 1$
$y^{\boldsymbol{\sigma}} = 2$
class 1
class 2
(a) Predictions $\sigma \in \Delta^3$ where $y^\sigma = 1, 2, 3$
class 3
$\boldsymbol{\sigma}$
$\ln K$
$\boldsymbol{p}$
$R_\alpha(\boldsymbol{\sigma})$
class 1
class 2
(b) Supervised loss $R_\alpha(\sigma|y)$ for ground truth label $y = 2$

Figure 3.9: Supervised loss $R_\alpha(\sigma \,|\, y)$: extending Figure 3.7(b) to $K = 3$. Three colored areas of probability simplex Δ^3 in (a) mark predictions σ with different hard-max labels $\mathbf{y}^\sigma$. Assuming $y = 2$, the blue region in (b) shows where supervised losses $R_\alpha(\sigma \,|\, y)$ in (3.39) and (3.40) are consistent with entropy $R_\alpha(\sigma)$. Loss (3.39) is constant $\ln K$ in all gray regions. It is discontinuous at the blue/gray boundary, except in the center. The continuous variant (3.40) is constant $\ln K$ only in the light gray region. For σ in the dark gray area, it is defined by the entropy $R_\alpha(\mathbf{p})$ at the closest point $\mathbf{p}$ in the blue region.

or its continuous variant, see Figure 3.9,

$$R_\alpha(\sigma \,|\, y) \;:=\; R_\alpha\left(\arg \min_{\mathbf{p}\in\Delta^K:\, y^{\mathbf{p}}=y} \|\mathbf{p} - \sigma\|\right) \tag{3.40}$$

where the argument of entropy $R_\alpha(\mathbf{p})$ is distribution $\mathbf{p} \in \Delta^K$ consistent with label y that is the closest to prediction σ. It is necessary to check that, as a function of $(K-1)$-dimensional *margins*, supervised loss $R_\alpha(\cdot \,|\, y)$ satisfies the exponential decay condition (11) in Theorem 4.1 [179]. The remaining proof of Theorem 4 follows the same arguments as in Theorem 3.

Discussion: entropy clustering vs SVM clustering

After establishing the max-margin property for regularized decisiveness (3.13) or (3.15), it is interesting to review the implications for entropy clustering and juxtapose it with SVM clustering. In particular, the formulation in [227] is highly relevant. They use the the regularized *hinge loss*, as in *soft SVM*, combined with a hard *fairness* constraint

$$\begin{aligned} L_{mm} \;\; &= \;\; \gamma\|\mathbf{v}\|^2 \; + \; \overline{\max\{0, 1 - t\,\mathbf{v}^\top X\}} \\ & \quad\; \text{subject to} \quad -\epsilon \leq \bar{t} \leq \epsilon \end{aligned} \tag{3.41}$$

where, unlike supervised SVM [217], binary targets $t \in \{\pm 1\}$ are optimization variables here. They are estimated jointly with the linear model parameters $\mathbf{v}$. Following our terminology from Section 3.1.4, objective (3.41) is a self-labeling loss with binary pseudo-labels t. It resembles (3.8) where hard constraints represent fairness and pseudo-labels are discrete/hard. One can also relate the hinge loss in (3.41) to a truncated variant of decisiveness[5] encouraging linear discriminant $\mathbf{v}$ such that $|\mathbf{v}^\top X| \geq 1$ for all data points X.

However, one significant difference exists between (3.41) and (3.8). Minimization of the norm $\|\mathbf{v}\|^2$ in supervised or unsupervised SVM, as in (3.41), is known to be central to the problem. It is well understood that it directly corresponds to margin maximization. In contrast, entropy clustering methods [30, 98, 74, 102, 8] typically skip $\|\mathbf{v}\|^2$ in their losses, e.g. (3.8), though it may appear implicitly due to the omnipresence of the *weight decay*. It is included in (3.12) [101], but for the wrong reason discussed in Section 3.2.1. It is also a part of (3.6), but it is motivated only by a generic model simplicity argument in the context of complex models [116]. It is also argued in [79] that decisiveness $\overline{H(\sigma)}$ encourages "decision boundary away from the data points", which may informally suggest margin maximization, but norm $\|\mathbf{v}\|^2$ is not present in their argument. In fact, margin maximization is not guaranteed without norm regularization.

Similarly to SVM, our max-margin theories for regularized decisiveness are focused on a linear binary case where guarantees are the strongest, but the multi-label case is also discussed. Non-linear extensions are possible due to high-dimensional embeddings $f_{\mathbf{w}}(X)$ in (3.4), but the quality guarantees are weak, as in kernel SVM. One notable difference is that non-linear SVMs typically use *fixed* data kernels, i.e. implicit high-dimensional embeddings. In contrast, discriminative entropy clustering can learn $f_{\mathbf{w}}(X)$.

Also, note that our max-margin theories apply to entropy clustering losses combining regularized decisiveness with fairness, e.g. (3.14), only if fairness is not compromised in the solution. If decisiveness and fairness require some trade-off, the max-margin guarantee is less clear [6]. However, four toy examples varying the cardinalities of one of the clusters in Figure 3.10 empirically

[5]Minimizing out t in the hinge loss (3.41) gives $\max\{0, 1 - |\mathbf{v}^\top X|\}$.

[6]This is similar to soft SVM when the margin size is traded off with margin violations.

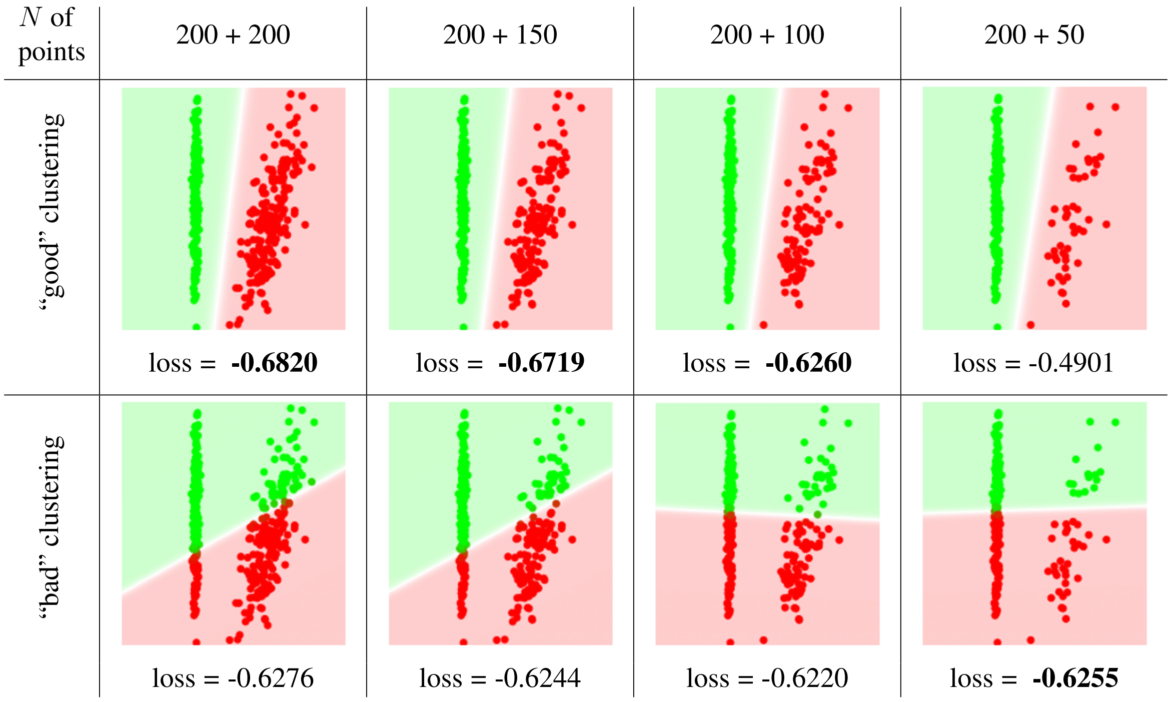


Figure 3.10: Balance vs margin size for loss (3.14). The data is generated from two Gaussian distributions. The cardinalities are equal in the leftmost column, with 200 points in each Gaussian. In the other three columns, the cardinality of one of the Gaussian blobs is gradually reduced to 50. The rows show two local minima for each example and the corresponding loss values. The lower loss value is highlighted in boldface font. Note that loss (3.14) favors the clustering balance (fairness) over the maximum margin only in the last column.

demonstrate that loss (3.14) prefers the maximum margin clustering even for highly unbalanced data. The optimum favors the balance over the margin size only in the last column.

Empirically, the max-margin property is also evident for self-labeling surrogates of (3.14), see Sec.3.1.4 and Sec.3.3. We conjecture that margin maximization could be formally extended to many self-labeling formulations of entropy clustering loss including the norm $\|\mathbf{v}\|^2$, but leave this research for future work.

3.3 Our self-labeling loss and optimization algorithms

The conceptual properties discussed in the previous section may improve the general understanding of entropy clustering, but their new practical benefits are limited. For example, margin maximization implicitly happens in prior entropy methods since norm regularization (weight-decay) is omnipresent.

This section addresses some specific limitations of prior entropy clustering formulations that do affect the practical performance. We focus on self-labeling (Section 3.1.4 and Chapter 2). As introduced in Chapter 2, we observe that the standard cross-entropy formulation of decisiveness is sensitive to pseudo-label errors while reverse cross-entropy and collision cross-entropy are more robust to the errors in the estimated pseudo-labels. We propose *strong fairness* in Section 3.3.1, which is critical to deriving efficient EM algorithms for minimizing the loss w.r.t. pseudo-labels. The EM algorithms for reverse cross-entropy and collision cross-entropy are derived in Section 3.3.2 and 3.3.3 respectively.

3.3.1 Self-labeling surrogate loss formulation

We start from the (2.5) as introduced in Chapter 2. We observe that the standard fairness term in ((2.5),3.6,3.9) is the *reverse* KL divergence w.r.t. cluster volumes, i.e. the average predictions $\bar{\sigma}$. It can tolerate highly unbalanced solutions where $\bar{\sigma}_k = 0$ for some cluster k, see the dashed curves in Figure 3.11(a). We propose the *forward*, a.k.a. *zero-avoiding*, KL divergence $KL(u \,\|\, \bar{\sigma})$, see the solid curves Figure 3.11(a), which assigns infinite penalties to highly unbalanced clusters. We refer to this as *strong fairness*. More importantly, this switch facilitates the derivation of efficient EM algorithms for estimating the pseudo-labels. Our formulation of self-labeling loss can be written as

$$L_{our} \quad := \quad \overline{H(\sigma, y)} \; + \; \lambda \, KL(u \,\|\, \bar{y}) \; + \; \gamma \, \|\mathbf{v}\|^2. \tag{3.42}$$

where $H(\cdot, \cdot)$ will be replaced by H_{RCE} as shown in the formula and H_{CCE}.

3.3.2 EM algorithm for optimizing y with H_{RCE}

Minimization of a self-labeling loss w.r.t pseudo-labels y for given predictions σ is a critical operation in iterative self-labeling techniques [8, 101], see Section 3.1.4. Besides well-motivated numerical properties of our new loss (3.42), in practice it also matters that it has an efficient solver for pseudo-labels. While (3.42) is convex w.r.t. y, optimization is done over a probability

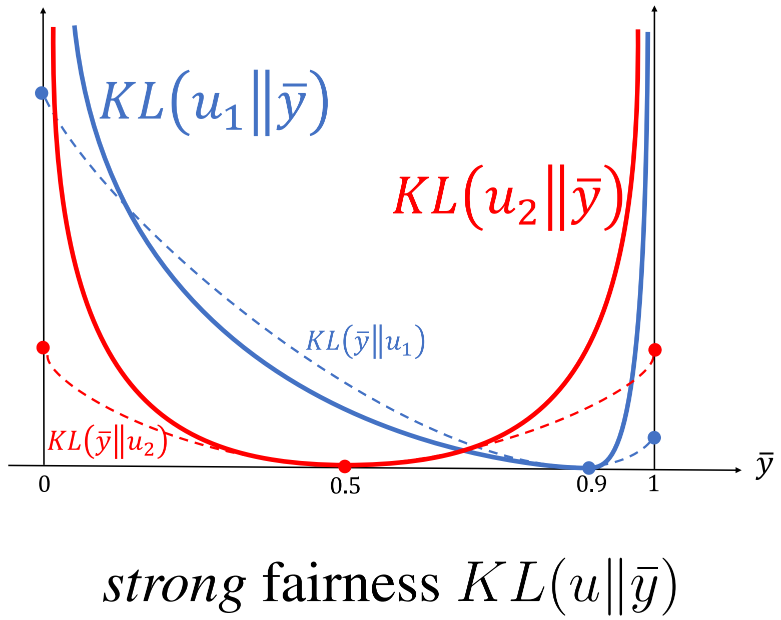


Figure 3.11: "Forward" vs "reverse": KL-divergence. Assuming binary classification $K = 2$, probability distributions σ or $\bar{\sigma}$ are represented as points on [0,1]. The solid curves illustrate the *forward* KL-divergence $KL(u\|\bar{y})$ for average pseudo-labels $\bar{y}$ in (3.42), though the same points could be made with average predictions $\bar{\sigma}$ instead of $\bar{y}$. We show two examples of volumetric prior $u_1 = (0.9, 0.1)$ (blue) and $u_2 = (0.5, 0.5)$ (red). The reverse KL-divergence $KL(\bar{\sigma}\|u)$ (dashed curves), commonly representing fairness in prior work, tolerates extremely unbalanced clustering, i.e. the end points of the interval [0,1].

simplex and a good practical solver is not a given. Note that $H(\sigma, y)$ works as a *log barrier* for the constraint $y \in \Delta^K$. This could be problematic for the first-order methods, but a basic Newton's method is a good match, e.g. [107]. The overall convergence rate of such second-order methods is fast, but computing the Hessian's inverse is costly. Instead, we derive a more efficient *expectation-maximization* (EM) algorithm, see Table 3.1.

Assume that model parameters and predictions in (3.42) are fixed, *i.e.* $\mathbf{v}$ and σ. Following *variational inference* [16], we introduce K auxiliary latent variables, distributions $S^k \in \Delta^N$ representing normalized support of each cluster k over N data points. In contrast, N distributions $y_i \in \Delta^K$ show support for each class at every point X_i. We refer to each vector S^k as a *normalized cluster* k. Note that here we focus on individual data points and explicitly index them by $i \in \{1, \dots, N\}$. Thus, we use $y_i \in \Delta^K$ and $\sigma_i \in \Delta^K$. Individual components of distribution $S^k \in \Delta^N$ corresponding to data point X_i is denoted by scalar S_i^k.

First, we expand our loss (3.42) using our new latent variables $S^k \in \Delta^N$

$$\begin{aligned} L_{our} &\overset{c}{=} \overline{H_{\text{RCE}}(y, \sigma)} + \lambda\, H(u, \bar{y}) + \gamma\, \|\mathbf{v}\|^2 && (3.43) \\ &= \overline{H(\sigma, y)} - \lambda \sum_k u^k \ln \sum_i S_i^k \frac{y_i^k}{S_i^k N} + \gamma\, \|\mathbf{v}\|^2 \\ &\leq \overline{H(\sigma, y)} - \lambda \sum_k \sum_i u^k S_i^k \ln \frac{y_i^k}{S_i^k N} + \gamma\, \|\mathbf{v}\|^2 && (3.44) \end{aligned}$$

Due to the convexity of negative $\log$, we apply Jensen's inequality to derive an upper bound, i.e. (3.44), to L_{our}. Such a bound becomes tight when:

$$\text{E-step}: \qquad S_i^k = \frac{y_i^k}{\sum_j y_j^k} \tag{3.45}$$

Then, we fix S_i^k as (3.45) and solve the Lagrangian of (3.44) with simplex constraint to update y as:

$$\text{M-step}: \qquad y_i^k = \frac{\sigma_i^k + \lambda N u^k S_i^k}{1 + \lambda N \sum_c u^c S_i^c} \tag{3.46}$$

We run these two steps until convergence with respect to some predefined tolerance. Note that the minimum y is guaranteed to be globally optimal since (3.43) is convex w.r.t. y. The empirical convergence rate is within 15 steps on MNIST. The comparison of computation speed on synthetic data is shown in Table 3.1. While the number of iterations to convergence is roughly the same as Newton's methods, our EM algorithm is much faster in terms of running time and is extremely easy to implement using the highly optimized built-in functions from the standard PyTorch library that supports GPU.

	number of iterations (to convergence)			running time in Section (to convergence)		
K	**2**	**20**	**200**	**2**	**20**	**200**
Newton	3	3	4	$2.8e^{-2}$	$3.3e^{-2}$	$1.7e^{-1}$
EM	2	2	2	$9.9e^{-4}$	$2.0e^{-3}$	$4.0e^{-3}$

Table 3.1: Our EM algorithm vs Newton's methods [107].

Inspired by [197, 98], we also adapted our EM algorithm to allow for updating y within each batch. In fact, the mini-batch approximation of (3.43) is an upper bound. Considering the first two terms of (3.43), we can use Jensen's inequality

$$\overline{H(\sigma,y)} + \lambda\, H(u,\bar{y}) \quad \leq \quad \mathbb{E}_B[\overline{H_B(\sigma,y)} + \lambda\, H(u,\bar{y}_B)] \tag{3.47}$$

where B is the batch randomly sampled from the whole dataset and the bar operator is the average over data points in the batch. Now, we can apply our EM algorithm to update y in each batch, which is even more efficient. Compared to other methods [74, 8, 101] using auxiliary variables y, we can efficiently update y on the fly while they only update once or just a few times per epoch due to the cost of computing y for the whole dataset. Interestingly, we found that it is important to update y on the fly, which makes convergence faster and improves the performance significantly, as shown in Figure 3.12. We use this "batch version" EM throughout all the experiments.

3.3.3 EM algorithm for optimizing y with H_{CCE}

As discussed in Chapter 2, H_{CCE} has more good properties than H_{RCE}, such as symmetry, and the allowance of one-hot distribution. However, the derivation of the EM algorithm is not as straightforward as that for H_{RCE}. Since the second-order Newton's method can not be easily applied, unlike H_{RCE} that acts like a log barrier, an efficient EM algorithm is more necessary. As shown in Table 3.2, our derived EM algorithm is much faster than the general projected gradient descent algorithm. Next, we show the detailed derivation of the EM algorithm.

We derive the EM algorithm introducing latent variables, K distributions $S^k \in \Delta^M$ representing normalized support for each cluster over M data points. We refer to each vector S^k as a *normalized cluster* k. Note the difference with distributions represented by pseudo-labels $y \in \Delta^K$ showing support for each class at a given data point. Since we explicitly use individual data points below, we will start to carefully index them by $i \in \{1, \dots, M\}$. Thus, we will use $y_i \in \Delta^K$ and

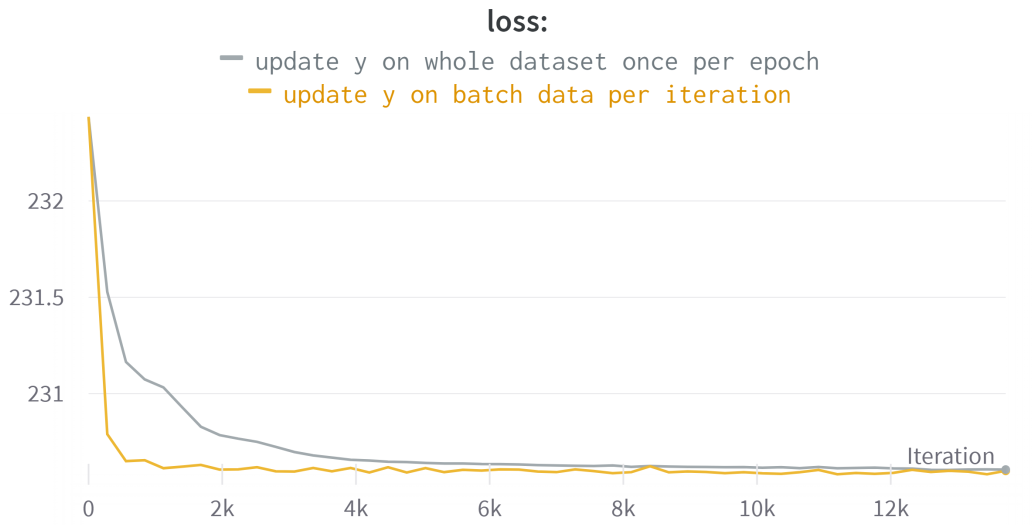


Figure 3.12: Loss (3.42) for two different update strategies for y. These plots are generated with a linear classifier on MNIST. We use the same initialization and run both strategies for 50 epochs. Note that once-per-epoch updates (corresponding to the gray curve) achieve 52% accuracy, while once-per-batch updates (corresponding to the yellow curve) achieve 63%.

$\sigma_i \in \Delta^K$. Individual components of distribution $S^k \in \Delta^M$ corresponding to data point i will be denoted by scalar S_i^k. In the following derivation, we omit the l_2 norm for brevity.

First, we introduce the latent variables $S^k \in \Delta^M$

$$L_{CCE+} \overset{c}{=} \overline{H_{\text{CCE}}(y,\sigma)} + \lambda\, H(u,\bar{y}) \tag{3.48}$$

$$= \overline{H_{\text{CCE}}(y,\sigma)} - \lambda \sum_k u^k \ln \sum_i S_i^k \frac{y_i^k}{S_i^k M} \leq \overline{H_{\text{CCE}}(y,\sigma)} - \lambda \sum_k \sum_i u^k S_i^k \ln \frac{y_i^k}{S_i^k M} \tag{3.49}$$

Due to the convexity of negative $\log$, we apply the Jensen's inequality to derive an upper bound, i.e. (3.49), to L_{CCE+}. Such bound becomes tight when:

$$\textbf{E step} : \qquad S_i^k = \frac{y_i^k}{\sum_j y_j^k} \tag{3.50}$$

Next, we derive the M step. Introducing the hidden variable S breaks the fairness term into the sum of independent terms for pseudo-labels $y_i \in \Delta_K$ at each data point i. The solution for S

	running time in Section per iteration			**number of iterations (to convergence)**			**running time in Section (to convergence)**		
K	**2**	**20**	**200**	**2**	**20**	**200**	**2**	**20**	**200**
PGD (η_1)	$7.8e^{-4}$	$2.9e^{-3}$	$6.7e^{-2}$	326	742	540	0.25	2.20	36.25
PGD (η_2)	$9.3e^{-4}$	$3.3e^{-3}$	$6.8e^{-2}$	101	468	344	0.09	1.55	23.35
PGD (η_3)	$9.9e^{-4}$	$3.2e^{-3}$	$7.0e^{-2}$	24	202	180	0.02	0.65	12.60
our EM	$1.8e^{-3}$	$1.6e^{-3}$	$5.1e^{-3}$	25	53	71	0.04	0.09	0.36

Table 3.2: Comparison of our EM algorithm to Projected Gradient Descent (PGD). η is the step size. For $K = 2$, $\eta_1 \sim \eta_3$ are 1, 10 and 20 respectively. For $K = 20$ and $K = 200$, $\eta_1 \sim \eta_3$ are 0.1, 1 and 5 respectively. Higher step size leads to divergence of PGD.

does not change (E step). Lets focus on the loss with respect to y. The collision cross-entropy (CCE) also breaks into the sum of independent parts for each y_i. For simplicity, we will drop all indices i in variables y_i^k, S_i^k, σ_i^k. Then, the combination of CCE loss with the corresponding part of the fairness constraint can be written for each $y = \{y_k\} \in \Delta_K$ as

$$-\ln \sum_k \sigma_k y_k \quad - \quad \lambda \sum_k u_k S_k \ln y_k. \tag{3.51}$$

First, observe that this loss must achieve its global optimum in the interior of the simplex if $S_k > 0$ and $u_k > 0$ for all k. Indeed, the second term enforces the "log-barier" at the boundary of the simplex. Thus, we do not need to worry about KKT conditions in this case. Note that S_k might be zero, in which case we need to consider the full KKT conditions. However, the Property 3 that will be mentioned later eliminates such concern if we use positive initialization. For completeness, we also give the detailed derivation for such case and it can be found at the end of this section.

Adding the Lagrange multiplier γ for the simplex constraint, we get an unconstrained loss

$$-\ln \sum_k \sigma_k y_k \quad - \quad \lambda \sum_k u_k S_k \ln y_k \quad + \quad \gamma \left(\sum_k y_k - 1 \right)$$

that must have a stationary point inside the simplex.

Computing partial derivatives w.r.t. y_k and simplify the equations, then we get

$$y_k \left(1 + \lambda u^\top S - \frac{\sigma_k}{\sigma^\top y} \right) \quad = \quad \lambda u_k S_k$$

This equation is necessarily solved by pseudo-labels $y = \{y_k\} \in \Delta_K$ such that

$$y_k \;=\; \frac{\lambda u_k S_k}{\lambda u^\top S + 1 - \frac{\sigma_k}{x}} \qquad \text{for some } \; x \in \left(\frac{\sigma_{max}}{1+\lambda u^\top S} \; , \; \sigma_{max} \right] \tag{3.52}$$

where σ_{max} is the largest σ_k. The specified interval domain for x is necessary for the solution.

Theorem 5. [M-step solution]: The sum $\sum_k y_k$ as in (3.52) is positive, continuous, convex, and monotonically decreasing function of x on the specified interval, see Fig.3.13(a). Moreover, there exists a unique solution $\{y_k\} \in \Delta_k$ and x such that

$$\sum_k y_k \;\equiv\; \sum_k \frac{\lambda u_k S_k}{\lambda u^\top S + 1 - \frac{\sigma_k}{x}} \;=\; 1 \quad \text{and} \quad x \in \left(\frac{\sigma_{max}}{1+\lambda u^\top S} \; , \; \sigma_{max} \right] \tag{3.53}$$

Proof. All y_k in (3.52) are positive, continuous, convex, and monotonically decreasing functions of x on the specified interval. Thus, $\sum y_k$ behaves similarly. Assuming that max is the index of prediction σ_{max}, we have $y_{max} \to +\infty$ when approaching the interval's left endpoint $x \to \frac{\sigma_{max}}{1+\lambda u^\top S}$. Thus, $\sum y_k > 1$ for smaller values of x. At the right endpoint $x = \sigma_{max}$ we have $y_k \leq \frac{\lambda u_k S_k}{\lambda u^\top S}$ for all k implying $\sum y_k \leq 1$. Monotonicity and continuity of $\sum y_k$ w.r.t. x imply the theorem. $\square$

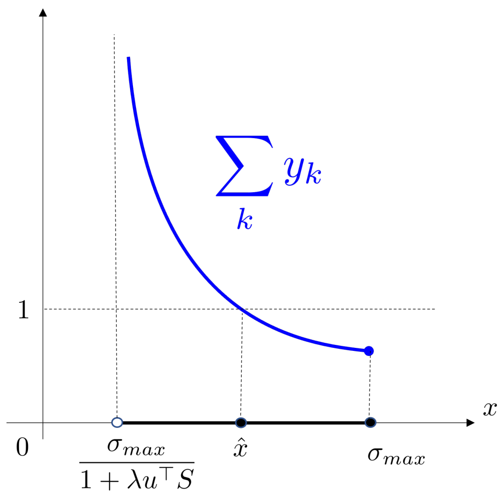
The monotonicity and convexity of $\sum_k y_k$ with respect to x suggest that the problem (3.53) formulated in Theorem 5 allows efficient algorithms for finding the corresponding unique solution. For example, one can use the iterative Newton's updates to search for x in the specified interval. The following Lemma gives us a proper starting point for Newton's method.

Lemma 1. *Assuming $u_k S_k$ is positive for each k, then the reachable left end point in Theorem 5 can be written as*

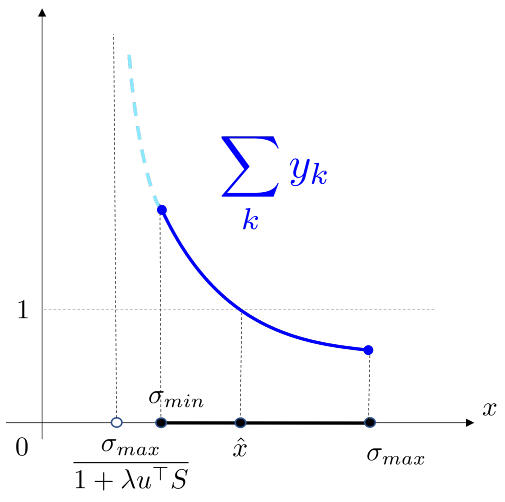
$$l := \max_k \frac{\sigma_k}{1 + \lambda u^\top S - \lambda u_k S_k}.$$

Proof. Firstly, we prove that l is (strictly) inside the interior of the interval in Theorem 5. For the left end point, we have

$$\begin{aligned} l := \max_k & \frac{\sigma_k}{1 + \lambda u^\top S - \lambda u_k S_k} \\ \geq \quad & \frac{\sigma_{max}}{1 + \lambda u^\top S - \lambda u_{max} S_{max}} \\ > \quad & \frac{\sigma_{max}}{1 + \lambda u^\top S} && u_{max} S_{max} \text{ is positive} \end{aligned}$$



(a) general case



(b) special case for $\sigma_{min} > \frac{\sigma_{max}}{1+\lambda u^\top S}$

Figure 3.13: Collision Cross Entropy - M step (3.53): the sum $\sum_k y_k$ is a positive, continuous, convex, and monotonically decreasing function of x on the specified interval (a). In case $\sigma_{min} > \frac{\sigma_{max}}{1+\lambda u^\top S}$ (b), one can further restrict the search interval for x. The fact that $\sum_k y_k \geq 1$ for $x = \sigma_{min}$ follows from an argument similar to the one below (3.52) showing that $\sum_k y_k \leq 1$ for $x = \sigma_{max}$.

For the right end point, we have

$$\begin{aligned} l := \max_k \frac{\sigma_k}{1+\lambda u^\top S - \lambda u_k S_k} & \\ < \quad \max_k \sigma_k & \qquad\qquad 1+\lambda u^\top S - \lambda u_k S_k > 1 \\ = \quad \sigma_{max} & \end{aligned}$$

Therefore, l is a reachable point. Moreover, any $\frac{\sigma_{max}}{1+\lambda u^\top S} < x < l$ will still induce positive y_k for any k and we will also use this to prove that x should not be smaller than l. Let

$$c := \arg\max_k \frac{\sigma_k}{1+\lambda u^\top S - \lambda u_k S_k}$$

then we can substitute l into the x of y_c. It can be easily verified that $y_c = 1$ at such l. Since y_c is monotonically decreasing in terms of x, any x smaller than l will cause y_c to be greater than 1. At the same time, other y_k is still positive as mentioned just above, so the $\sum_k y_k$ will be greater than 1. Thus, l is a reachable left end point. $\square$

The algorithm for M-step solution is summarized as below. Note that we present the algorithm for only one data point, and we can easily and efficiently scale up for more data in a batch by using the Numba compiler.

Algorithm 1 Newton's method for M-step

```
Input  : ${\sigma_k}$, ${S_k}$, $\lambda$, $\epsilon$
Output : ${y_k}$
Initialize $x \leftarrow \max_k \frac{\sigma_k}{1+\lambda u^\top S-\lambda u_k S_k}$
 calculate $f(x) \leftarrow \sum_k \frac{\lambda u_k S_k}{\lambda u^\top S+1-\frac{\sigma_k}{x}} - 1$
 while $f(x) \geq \epsilon$ do
    calculate $f'(x) \leftarrow \sum_k \frac{-\lambda u_k S_k \sigma_k}{(\lambda u^\top Sx+x-\sigma_k)^2}$
    $x \leftarrow x - \frac{f(x)}{f'(x)}$
    calculate $f(x) \leftarrow \sum_k \frac{\lambda u_k S_k}{\lambda u^\top S+1-\frac{\sigma_k}{x}} - 1$
end
$y_k \leftarrow \frac{\lambda u_k S_k}{\lambda u^\top S+1-\frac{\sigma_k}{x}}$
```

In the following, we give the property about the positivity of the solution. This property implies that if our EM algorithm has only (strictly) positive variables S_k or y_k at initialization, these variables will remain positive during all iterations.

PROPERTY 3. For any category k such that $u_k > 0$, the set of strictly positive variables y_k or S_k can only grow during iterations of our EM algorithm for the loss (3.51) based on the collision cross-entropy.

Proof. As obvious from the E-step (3.50), it is sufficient to prove this for variables y_k. If $y_k = 0$, then the E-step (3.50) gives $S_k = 0$. According to the M-step for the case of collision cross-entropy, variable y_k may become (strictly) positive at the next iteration if $\sigma_k = \sigma_{max}$. Once y_k becomes positive, the following E-step (3.50) produces $S_k > 0$. Then, the fairness term effectively enforces the log-barrier from the corresponding simplex boundary making M-step solution $y_k = 0$ prohibitively expensive. Thus, y_k will remain strictly positive at all later iterations. □

Note that Property 3 does not rule out the possibility that y_k may become arbitrarily close to zero during EM iterations. Empirically, we did not observe any numerical issues. The complete algorithm is given below. As discussed in Section 3.3.2, we also update our y in each batch. Intuitively, updating y on the fly can prevent the network from being easily trapped in some local minima created by the incorrect pseudo-labels.

Complete Solutions for M step The main case when $u_k S_k > 0$ for all k in (3.51) is presented above. Here we derive the case when there exists some k such that $u_k S_k = 0$. Assume a non-empty subset of categories/classes

$$K_o := \{k \,|\, u_k S_k = 0\} \quad \neq \quad \emptyset$$

and its non-empty complement

$$\bar{K}_o := \{k \,|\, u_k S_k > 0\} \quad \neq \quad \emptyset.$$

In this case the second term (fairness) in our loss (3.51) does not depend on variables y_k for $k \in K_o$. Also, note that the first term (collision cross-entropy) in (3.51) depends on these variables only via their linear combination $\sum_{k\in K_o} \sigma_k y_k$. It is easy to see that for any given confidences y_k for $k \in \bar{K}_o$ it is optimal to put all the remaining confidence $1 - \sum_{k\in\bar{K}_o} y_k$ into one class $c \in K_o$ corresponding to the larges prediction among the classes in K_o

$$c \;:=\; \arg\max_{k\in K_o} \sigma_k$$

so that

$$y_c \;=\; 1 - \sum_{k\in\bar{K}_o} y_k \qquad \text{and} \qquad y_k = 0, \;\; \forall k \in K_o \setminus c.$$

Then, our loss function (3.51) can be written as

$$-\ln \sum_{k\in\bar{K}_o\cup\{c\}} \sigma_k y_k \;\;-\;\; \lambda \sum_{k\in\bar{K}_o} u_k S_k \ln y_k \tag{e}$$

that gives the Lagrangian function incorporating the probability simplex constraint

$$-\ln \textstyle\sum_{k\in\bar{K}_o\cup\{c\}} \sigma_k y_k \;\;-\;\; \lambda \textstyle\sum_{k\in\bar{K}_o} u_k S_k \ln y_k \;\;+\;\; \gamma \left(\textstyle\sum_{k\in\bar{K}_o\cup\{c\}} y_k - 1\right).$$

The stationary point for this Lagrangian function should satisfy equations

$$-\tfrac{\sigma_k}{\sigma^\top y} - \lambda u_k S_k \tfrac{1}{y_k} + \gamma \;\;=\;\; 0, \quad \forall k \in \bar{K}_o \qquad \text{and} \qquad -\tfrac{\sigma_c}{\sigma^\top y} + \gamma \;\;=\;\; 0$$

which could be easily written as a linear system w.r.t variables y_k for $k \in \bar{K}_o \cup \{c\}$.

We derive a closed-form solution for the stationary point as follows. Substituting γ from the right equation into the left equation, we get

$$\frac{\sigma_c - \sigma_k}{\sigma^\top y} y_k \;\;=\;\; \lambda u_k S_k, \qquad \forall k \in \bar{K}_o\,. \tag{f}$$

Summing over $k \in \bar{K}_o$ we further obtain

$$\frac{\sigma_c(1-y_c)-\sum_{k\in\bar{K}_o}\sigma_k y_k}{\sigma^\top y} = \lambda u^\top S \qquad \Rightarrow \qquad \frac{\sigma_c-\sigma^\top y}{\sigma^\top y} = \lambda u^\top S$$

giving a closed-form solution for $\sigma^\top y$

$$\sigma^\top y = \frac{\sigma_c}{1+\lambda u^\top S}.$$

Substituting this back into (f) we get closed-form solutions for y_k

$$y_k = \frac{\lambda u_k S_k}{(1+\lambda u^\top S)(1-\frac{\sigma_k}{\sigma_c})}, \qquad \forall k \in \bar{K}_o .$$

Note that positivity and boundedness of y_k requires $\sigma_c > \sigma_k$ for all $k \in \bar{K}_o$. In particular, this means $\sigma_c = \sigma_{max}$, but it also requires that all σ_k for $k \in \bar{K}_o$ are strictly smaller than σ_{max}. We can also write the corresponding closed-form solution for y_c

$$y_c = 1 - \sum_{k\in\bar{K}_o} y_k = 1 - \frac{\sigma_c}{1+\lambda u^\top S}\sum_{k\in\bar{K}_o}\frac{\lambda u_k S_k}{\sigma_c-\sigma_k}.$$

Note that this solution should be positive $y_c > 0$ as well.

In case any of the mentioned constraints ($\sigma_c > \sigma_k, \forall k \in \bar{K}_o$ and $y_c > 0$) is not satisfied, the *complimentary slackness* (KKT) can be used to formally prove that the optimal solution is $y_c = 0$. That is, $y_k = 0$ for all $k \in K_o$. This reduces the optimization problem to the earlier case focusing on resolving y_k for $k \in \bar{K}_o$. This case is guaranteed to find a unique solution in the interior of the simplex $\Delta_{\bar{K}_o}$. Indeed, since inequality $u_k S_k > 0$ holds for all $k \in \bar{K}_o$, the strong fairness enforces a log-barrier for all the boundaries of this simplex.

3.3.4 Discussion

Here we briefly discuss the differences between our method for (3.42) and the algorithm for (3.12) in [101]. Their loss is convex with respect to $\mathbf{v}$. In contrast, the reverse cross-entropy in our loss (3.42) is non-convex as a natural consequence of our focus on robustness to pseudo-label errors. However, both losses are non-convex w.r.t. parameters $\mathbf{w}$ of the interior network layers responsible for representation $f_{\mathbf{w}}(X)$ in (3.4). As for the optimization w.r.t. the pseudo-labels y, both losses are convex. Our EM algorithm converges to the global optima for y, while they use an approximate closed-form solution for y, but it does not have optimality guarantees. Both self-labeling algorithms iteratively optimize the corresponding surrogate losses over pseudo-labels y and model parameters $\mathbf{v}$, $\mathbf{w}$.

3.4 Experimental results

We test our approach on standard benchmark datasets with different network architectures. We also provide a comparison of different losses under semi-supervised settings. All the results are reproduced using either public codes or our own implementation under the same experimental settings for fair comparison.

Dataset For the clustering problem, we use four standard benchmark datasets: MNIST [123], CIFAR10/100 [209] and STL10 [48]. We follow [102] to use the whole dataset for training and testing unless otherwise specified. As for the semi-supervised setting, we conduct experiments on CIFAR10 and STL10. We split the data into training and test sets as suggested by the official instructions of the datasets.

Evaluation As for the evaluation of clustering, we set the number of clusters to the number of ground-truth category labels. To calculate the accuracy, we use the standard Hungarian algorithm [119] to find the best one-to-one mapping between predicted clusters and ground-truth labels. We don't need this matching step if we use other metrics, i.e. NMI, and ARI. For the evaluation of semi-supervised classification, we directly calculate the accuracy of the test set without the mapping step.

Implementation details For clustering with fixed features, we set λ in our loss to 100. The learning rate of the stochastic gradient descent is set to 0.1 and training takes 10 epochs. The batch size was set to 250. The coefficients for the l_2 norm of the weights are set to 0.001, 0.02, 0.009, and 0.02 for MNIST, CIFAR10, CIFAR100, and STL10 respectively. We also tuned the hyperparameters for all other methods.

As for the training of VGG4 in Section 3.4.3, we use Adam [108] with learning rate $1e^{-4}$ for optimizing the network parameters. We set the batch size to 250 for CIFAR10, CIFAR100, and MNIST, and we used 160 for STL10. For each image, we generate two augmentations sampled from "horizontal flip", "rotation" and "color distortion". We set the λ to 100 in our loss. The weight decay coefficient is set to 0.01. We report the mean accuracy and Std from 6 runs with different initializations while we use the same initialization for all methods in each run. We use 50 epochs for each run and all methods with tuned hyperparameters reach convergence within 50 epochs. As for the training of ResNet-18, we still use the Adam optimizer, and the learning rate is set to $1e^{-1}$ for the linear classifier and $1e^{-5}$ for the backbone. The weight decay coefficient is set to $1e^{-4}$. The batch size is 200 and the number of total epochs is 50. The λ is still set to 100. We only use one augmentation per image, and the coefficient for the augmentation term is set to 0.5, 0.2 and 0.4 respectively for STL10, CIFAR10, and CIFAR100. Such coefficient is tuned separately for IMSAT and MIADM.

3.4.1 Effect of α in Renyi entropies for clustering

In Section 3.2.2, we provide theoretical results that any α in Renyi entropies leads to the maximum margin under certain conditions. However, it is unclear how different α affects the results in real-world settings. Here, we use the standard STL10 dataset to test the clustering performance for different α following the experimental settings from [216]. We tuned the hyperparameters for

α	0.1	0.5	1	5	∞
ACC	62%(3.5)	63.45%(3.2)	70.23%(2.0)	71.05%(1.6)	73.34%(1.6)

Table 3.3: Effects of different α in Renyi entropy as the decisiveness term.

each column and used the stochastic gradient descent as the optimizer. While our self-labeling approach can give a performance boost for $\alpha = 1$, it does not work well empirically for other α. As one can easily check, the subproblems in the self-labeling formulation for $\alpha \neq 1$ can not be easily optimized. From the results in Table 3.3, we can see that larger α gives better results. Note that larger α induces smaller gradients as the network prediction gets more confident. If the network prediction is wrong, there is a chance for correction due to small gradients. On the contrary, the smaller α has larger gradients as network predictions become more confident, which creates strong local minima that are hard to get out of.

3.4.2 Clustering with fixed features

In this section, we test our loss as a proper clustering loss and compare it to the widely used Kmeans (generative) and other closely related losses (entropy-based and discriminative). We use the pre-trained (ImageNet) Resnet-50 [90] to extract the features. For Kmeans, the model is parameterized by K cluster centers. Comparably, we use a one-layer linear classifier followed by softmax for all other losses including ours. Kmeans results were obtained using the scikit-learn package in Python. To optimize the model parameters for other losses, we use stochastic gradient descent. Here we report the average accuracy and standard deviation over 6 randomly initialized trials in Table 3.4.

3.4.3 Joint clustering and feature learning

In this section, we train a deep network to jointly learn the features and cluster the data. We test our method on both a small architecture (VGG4) and large ones (ResNet-18, ResNet-50). The

	STL10	CIFAR10	CIFAR100 (20)	MNIST
K-means	85.20%(5.9)	67.78%(4.6)	42.99%(1.3)	47.62%(2.1)
MIGD [30, 116]	89.56%(6.4)	72.32%(5.8)	43.59%(1.1)	52.92%(3.0)
SeLa [8]	90.33%(4.8)	63.31%(3.7)	40.74%(1.1)	52.38%(5.2)
MIADM [101]	81.28%(7.2)	56.07%(5.5)	36.70%(1.1)	47.15%(3.7)
MIADM$\star$ [101]	88.64%(7.1)	60.57%(3.3)	41.2%(1.4)	50.61%(1.3)
w/ H_{RCE}	92.2%(6.2)	73.48%(6.2)	**43.8%(1.1)**	58.2%(3.1)
w/ H_{CCE}	**92.33%(6.4)**	**73.51%(6.3)**	43.72%(1.1)	**58.4%(3.2)**

Table 3.4: Comparison of different methods using fixed features. The numbers are the average accuracy and the standard deviation over 6 trials. $\star$: our “batch version" implementation of their method.

only extra standard technique we add here is self-augmentation as discussed in Section 3.1.3.

To train the VGG4, we use random initialization for network parameters. From Table 3.5, it can be seen that our approach consistently achieves the most competitive results in terms of accuracy (ACC). Most of the methods we compared in our work (including our method) are general concepts applicable to single-stage end-to-end training. To be fair, we tested all of them on the same simple architecture. But, these general methods can be easily integrated into other more complex systems with larger architecture such as ResNet-18.

	STL10	CIFAR10	CIFAR100 (20)	MNIST
IMSAT [98]	25.28%(0.5)	21.4%(0.5)	14.39%(0.7)	92.90%(6.3)
IIC [102]	24.12%(1.7)	21.3%(1.4)	12.58%(0.6)	82.51%(2.3)
SeLa [8]	23.99%(0.9)	24.16%(1.5)	**15.34%(0.3)**	52.86%(1.9)
MIADM [101]	17.37%(0.9)	17.27%(0.6)	11.02%(0.5)	17.75%(1.3)
MIADM$\star$ [101]	23.37%(0.9)	23.26%(0.6)	14.02%(0.5)	78.88%(3.3)
w/ H_{RCE}	25.33%(1.4)	24.16%(0.8)	15.09%(0.5)	93.58%(4.8)
w/ H_{CCE}	**25.98%(1.1)**	**24.26%(0.8)**	15.14%(0.5)	**95.11%(4.3)**

Table 3.5: Quantitative results of accuracy for unsupervised clustering methods with VGG4. We only use the 20 coarse categories for CIFAR100 following [102].

As for the training of ResNet-18, we found that random initialization does not work well. Following [216], we use the pre-trained weight from the self-supervised learning. For a fair comparison, we followed their experimental settings and compared ours to their second-stage results. Note that they split the data into training and testing. We also report two additional evaluation metrics, i.e. NMI and ARI.

In Table 3.6, we show the results using their pretext-trained network as initialization for our entropy clustering. We use only our clustering loss together with the self-augmentation (one augmentation per image this time) to reach higher numbers than SCAN, as shown in the table below. More importantly, we consistently improve over the most related method, MIADM, by a large margin, which clearly demonstrates the effectiveness of our proposed loss together with the optimization algorithm.

	CIFAR10			CIFAR100 (20)			STL10		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
SCAN [216]	81.8 (0.3)	71.2 (0.4)	66.5 (0.4)	42.2 (3.0)	**44.1 (1.0)**	26.7 (1.3)	75.5 (2.0)	65.4 (1.2)	59.0 (1.6)
IMSAT [98]	77.64% (1.3)	71.05% (0.4)	64.85% (0.3)	43.68% (0.4)	42.92% (0.2)	26.47% (0.1)	70.23% (2.0)	62.22% (1.2)	53.54% (1.1)
MIADM⋆ [101]	74.76% (0.3)	69.17% (0.2)	62.51% (0.2)	43.47% (0.5)	42.85% (0.4)	27.78% (0.4)	67.84% (0.2)	60.33% (0.5)	51.67% (0.6)
w/ H_{RCE}	83.09 (0.2)	71.65 (0.1)	68.05 (0.1)	46.79 (0.3)	43.27 (0.1)	28.51 (0.1)	77.67 (0.1)	67.66 (0.3)	61.26 (0.4)
w/ H_{CCE}	**83.27% (0.2)**	**71.95% (0.2)**	**68.15% (0.1)**	**47.01% (0.2)**	43.28% (0.1)	**29.11% (0.1)**	**78.12% (0.1)**	**68.11% (0.3)**	**62.34% (0.3)**

Table 3.6: Quantitative comparison using network ResNet-18.

We also follow the experimental settings in [216] to test our method against some closely related ones in the case where the number of classes is relatively large, say 100. We used a subset of the ImageNet dataset [58] where the number of classes is 100, following [216].

	Top1	Top5	NMI	ARI
SCAN [216]	66.2	88.1	78.7	54.4
IMSAT [98]	65.1	87.3	78.2	53.5
MIADM⋆ [101]	63.4	86.1	77.1	52.9
w/ H_{RCE}	67.1	89.7	79.5	55.2
w/ H_{CCE}	**68.5**	**89.9**	**79.8**	**55.3**

Table 3.7: Quatitative comparison using ResNet-50 on ImageNet-100 subset [216].

3.4.4 Semi-supervised classification

Although our paper is focused on self-labeled classification, we find it also interesting and natural to test our loss under a semi-supervised setting where partial data is provided with ground-truth labels. We use the standard cross-entropy loss for labeled data and directly add it to the self-labeled loss to train the network initialized by the pretext-trained network following [216]. Note that the pseudo-labels for the labeled data are constrained to be the ground-truth labels.

	0.1		**0.05**		**0.01**	
	STL10	**CIFAR10**	**STL10**	**CIFAR10**	**STL10**	**CIFAR10**
Only seeds	78.4%	81.2%	74.1%	76.8%	68.8%	71.8%
+ IMSAT [98]	88.1%	91.5%	81.1%	85.2%	74.1%	80.2%
+ IIC [102]	85.2%	90.3%	78.2%	84.8%	72.5%	80.5%
+ SeLa [8]	86.2%	88.6%	79.5%	82.7%	69.9%	79.1%
+ MIADM [101]	84.9%	86.1%	77.9%	80.1%	69.6%	77.5%
+ w/ H_{RCE}	88.7%	92.2%	82.6%	85.8%	75.6%	81.9%
+ w/ H_{CCE}	**88.9%**	**92.3%**	**82.9%**	**86.2%**	**75.7%**	**82.4%**

Table 3.8: Quantitative results for semi-supervised classification on STL10 and CIFAR10 using ResNet18. The numbers 0.1, 0.05 and 0.01 correspond to different ratio of labels used for supervision. “Only seeds” means we only use standard cross-entropy loss on seeds for training.

Chapter 4

Potts relaxations and weakly-supervised semantic segmentation

This Chapter studies unsupervised losses in the context of weakly-supervised semantic segmentation (WSSS). It combines entropy-based margin-maximizing clustering losses, *decisiveness* discussed in the Chapter 3, with segmentation-specific unsupervised loss, the Potts model. It is one of the simplest pairwise losses, which are common in semi-supervised learning [37, 240] and could be seen as a pixel-based *contrastive loss*. In the context of self-labeling with soft pseudo-labels and collision cross-entropy, see Chapter 2, we systematically study second-order relaxations of the Potts model. We discuss the numerical properties of such relaxations and provide an empirical comparison. We also develop an efficient general self-labeling algorithm.

4.1 Introduction

Full supervision for semantic segmentation requires thousands of training images with complete pixel-accurate ground truth masks. Their high costs explain the interest in weakly-supervised approaches based on image-level class *tags* [112, 7], pixel-level *scribbles* [130, 206, 205], or *boxes* [120]. This chapter is focused on weak supervision with *scribbles*, which we also call *seeds* or *partial masks*. While only slightly more expensive than image-level class tags, scribbles on less than 3% of pixels were previously shown to achieve accuracy approaching full supervision without any modifications of the segmentation models. In contrast, tag supervision typically requires highly specialized systems and complex multi-stage training procedures, which are hard to reproduce. Our interest in the scribble-based approach is motivated by its practical simplicity and mathematical clarity. The corresponding methodologies are focused on the design of unsupervised or self-supervised loss functions and stronger optimization algorithms. The corresponding solutions are often general and can be used in different weakly-supervised applications.

4.1.1 Scribble-supervised segmentation

Assume that a set of image pixels is denoted by Ω and a subset of pixels with ground truth labels is $S \subset \Omega$, which we call *seeds* or *scribbles* as subset S is typically marked by mouse-controlled UI for image annotations, e.g. see seeds over an image in Figure4.3(a). The ground truth label at any given pixel $i \in S$ is an integer

$$\bar{y}_i \in \{1, \dots, K\} \tag{4.1}$$

where K is the number of classes including the background. Without much ambiguity, it is convenient to use the same notation $\bar{y}_i$ for the equivalent *one-hot* distribution

$$\bar{y}_i \;\equiv\; (\bar{y}_i^1, \dots, \bar{y}_i^K) \in \Delta_{0,1}^K \qquad \text{for} \quad \bar{y}_i^k := [k = \bar{y}_i] \in \{0, 1\} \tag{4.2}$$

where $[\,\cdot\,]$ is the *True* operator for the condition inside the brackets. Set $\Delta_{0,1}^K$ represents K possible one-hot distributions, which are vertices of the K-class *probability simplex*

$$\Delta^K \;:=\; \{p = (p^1, \dots, p^K) \mid p^k \geq 0, \; \sum_{k=1}^K p^k = 1\}$$

representing all K-categorical distributions. The context of specific expressions should make it obvious if $\bar{y}_i$ is a class index (4.1) or the corresponding one-hot distribution (4.2).

Loss functions for weakly supervised segmentation with scribbles typically use *negative log-likelihoods* (NLL) over scribbles $i \in S \subset \Omega$ with ground truth labels $\bar{y}_i$

$$-\sum_{i \in S} \ln \sigma_i^{\bar{y}_i} \tag{4.3}$$

where $\sigma_i = (\sigma_i^1, \dots, \sigma_i^K) \in \Delta^K$ is the model prediction at pixel i. This loss is a standard in full supervision where the only difference is that $S = \Omega$ and usually, no other losses are needed for training. However, in a weakly supervised setting the majority of pixels are unlabeled, and unsupervised losses are needed for $i \notin S$.

The most common unsupervised loss in image segmentation is the Potts model and its relaxations. It is a pairwise loss defined on pairs of *neighboring* pixels $\{i, j\} \in \mathcal{N}$ for a given neighborhood system $\mathcal{N} \subset \Omega \times \Omega$, typically corresponding to the *nearest-neighbor* grid (NN) [26, 78], or other *sparse* (SN) [219] and *dense* neighborhoods (DN) [115]. The original Potts model is defined for discrete segmentation variables, e.g. as in

$$\sum_{\{i,j\} \in \mathcal{N}} P(\sigma_i, \sigma_j) \qquad \text{where} \quad P(\sigma_i, \sigma_j) = [\sigma_i \neq \sigma_j]$$

assuming integer-valued one-hot predictions $\sigma_i \in \Delta^K_{0,1}$. This *regularization* loss encourages smoothness between the pixels. Its popular *self-supervised* variant is

$$P(\sigma_i, \sigma_j) = w_{i,j} \cdot [\sigma_i \neq \sigma_j]$$

where pairwise affinities w_{ij} are based on local intensity edges [26, 78, 115]. Of course, in the context of network training, one should use relaxations of P applicable to (soft) predictions $\sigma_i \in \Delta^K$. Many types of its relaxation [175, 230] were studied in segmentation, e.g. *quadratic* [78], *bi-linear* [206], *total variation* [170, 32], and others [51].

Another unsupervised loss highly relevant for training segmentation networks is the entropy of predictions, which is also known as *decisiveness* [29, 80]

$$\sum_i H(\sigma_i)$$

where H is the Shannon's entropy function. This loss can improve generalization and the quality of representation by moving (deep) features away from the decision boundaries. Widely known in the context of unsupervised or semi-supervised classification, this loss also matters in weakly-supervised segmentation where it is used explicitly or implicitly[1].

Other unsupervised losses (e.g. contrastive), clustering criteria (e.g. K-means), or specialized architectures can be found in weakly-supervised segmentation [220, 160, 106, 39]. However, a lot can be achieved simply by combining the basic losses discussed above

$$L_{ws}(\sigma) \;\; := \;\; -\sum_{i \in S} \ln \sigma_i^{\bar{y}_i} \; + \; \eta \sum_{i \notin S} H(\sigma_i) \; + \; \lambda \sum_{ij \in \mathcal{N}} P(\sigma_i, \sigma_j) \tag{4.4}$$

which can be optimized directly by gradient descent [206, 219] or using *self-labeling* techniques [130, 142, 141] incorporating optimization of auxiliary *pseudo-labels* as sub-problems.

[1]Interestingly, a unary decisiveness-like term is the difference between convex quadratic and *tight*, but non-convex, bi-linear relaxations [175, 141] of the discrete pairwise Potts model.

4.1.2 Self-labeling and hard pseudo-labels

One argument motivating self-labeling approaches to weakly-supervised segmentation comes from well-known limitations of gradient descent when optimizing the Potts relaxatons, e.g. [142]. But even when using convex Potts relaxations [78, 170, 32], they are combined with the concave entropy term in (4.4) making their optimization challenging.

Typical self-labeling methods, including one of the first works on scribble-based semantic segmentation [130], introduce a sub-problem focused on the estimation of *pseudo-labels* over unlabeled points, separately from the network training by such labels. Pseudo-labeling is typically done by optimization algorithms or heuristics balancing unsupervised or self-supervised criteria, e.g. the Potts, and proximity to current predictions. Then, network fine-tuning from pseudo-labels and pseudo-labeling steps are iterated.

We denote pseudo-labels y_i slightly differently from the ground truth labels $\bar{y}_i$ by omitting the bar. It is important to distinguish them since the ground truth labels $\bar{y}_i$ for $i \in S$ are given, while the pseudo-labels y_i for $i \in \Omega \backslash S$ are estimated. The majority of existing self-labeling methods [130, 3, 142, 2, 125, 141, 221] estimate *hard* pseudo-labels, which could be equivalently represented either by class indices

$$y_i \in \{1, \dots, K\} \tag{4.5}$$

or by the corresponding one-hot categorical distributions

$$y_i \quad \equiv \quad (y_i^1, \dots, y_i^K) \in \Delta_{0,1}^K \qquad \text{for} \quad y_i^k := [k = y_i] \ \in \{0, 1\} \tag{4.6}$$

analogously with the hard ground truth labels in (4.1) and (4.2). In part, hard pseudo-labels are motivated by the network training where the default is NLL loss (4.3) assuming discrete labels. Besides, there are powerful discrete solvers for the Potts model [26, 170, 32]. We discuss the potential advantages of soft pseudo-labels in the next Section 4.1.3.

Joint loss vs "proposal generation": The majority of self-labeling approaches can be divided into two groups. One group designs pseudo-labeling and the network training sup-problems that are not formally related, e.g. [130, 129, 226]. While pseudo-labeling typically depends on the current network predictions and the network fine-tuning uses such pseudo-labels, the lack of a formal relation between these sub-problems implies that iterating such steps does not guarantee any form of convergence. Such methods are often referred to as *proposal generation* heuristics.

Alternatively, the pseudo-labeling sub-problem and the network training sub-problem can be formally derived from a weakly-supervised loss like (4.4), e.g. by ADM *splitting* [142] or as high-order *trust-region* method [141]. Such methods often formulate a *joint loss* function w.r.t network predictions and pseudo-labels and iteratively optimize it in a convergent manner that is guaranteed to decrease the loss. We consider this group of self-labeling methods as better motivated, more principled, and numerically safer.

4.1.3 Motivation and contributions

We observe that self-labeling with hard pseudo-labels y_i in (4.5) and (4.6) is inherently limited as such labels can not represent the uncertainty of class estimates at unlabeled pixels $i \in \Omega \backslash S$. Instead, we focus on *soft* pseudo-labels

$$y_i \quad = \quad (y_i^1, \ldots, y_i^K) \in \Delta^K \tag{4.7}$$

which are general categorical distributions p over K-classes. It is possible that the estimated pseudo-label y_i in (4.7) could be a one-hot distribution, which is a vertex of Δ^K. In such a case, one can treat y_i as a class index (4.5), but we avoid this. However, the ground truth labels $\bar{y}_i$ are always hard and we use them either as indices (4.1) or one-hot distributions (4.2), as convenient.

Soft pseudo-labels can be found in prior work on weakly-supervised segmentation [129, 226] using the “soft proposal generation”. In contrast, we formulate soft self-labeling as a principled optimization methodology where network predictions and soft pseudo-labels are variables in a joint loss, which guarantees convergence of the training procedure. Our pseudo-labels are auxiliary variables for ADM-based [23] splitting of the loss (4.4) into two simpler optimization sub-problems: one focused on the Potts model over unlabeled pixels, and the other on the network training. While similar to [142], instead of hard, we use soft auxiliary variables for the Potts sub-problem. Our work can be seen as a study of the relaxed Potts sub-problem in the context of weakly-supervised semantic segmentation. The related prior work is focused on discrete solvers fundamentally unable to represent class estimate uncertainty. Our contributions can be summarized as follows:

- convergent *soft self-labeling* framework based on a simple joint self-labeling loss (2.6)
- systematic evaluation of Potts relaxations and (cross-) entropy terms in our loss (2.6)
- state-of-the-art in scribble-based semantic segmentation that does not require any modifications of semantic segmentation models and is easy to reproduce
- using the same segmentation model, our self-labeling loss with 3% scribbles may outperform standard supervised cross-entropy loss with full ground truth masks.

bi-linear $\sim$ "graph cut" $P_{\text{BL}}(p,q) \;:=\; 1 - \; p^\top q$	**quadratic** $\sim$ "random walker" $P_{\text{Q}}(p,q) \;:=\; \frac{1}{2}\|p-q\|^2$
normalized quadratic $P_{\text{NQ}}(p,q) \;:=\; 1 - \frac{p^\top q}{\|p\|\|q\|} \;\equiv\; \frac{1}{2}\left\|\frac{p}{\|p\|} - \frac{q}{\|q\|}\right\|^2$	

Table 4.1: Second-order Potts relaxations, see Figure4.1(a,b,c)

4.2 Second-order relaxations of the Potts model

We focus on second-order relaxations for two reasons. First, to manage the scope of this study. Second, this includes several important baseline cases (see Table 4.1): *quadratic*, the simplest convex relaxation popularized by the *random walker* algorithm [78], and *bi-linear*, which is non-convex but *tight* [175] w.r.t. the original discrete Potts model. The latter implies that optimizing it over relaxed variables will lead to a solution consistent with a discrete Potts solver, e.g. *graph cut* [26]. On the contrary, the quadratic relaxation will produce a significantly different soft solution. We investigate such soft solutions.

Figure 4.2 shows two examples illustrating local minima for (a) the bi-linear and (b) quadratic relaxations of the Potts loss. In (a) two neighboring pixels attempt to jointly change the common soft label from $y_i = y_j = (1,0,0)$ to $y_i'' = y_j'' = (0,1,0)$, which corresponds to a "move" where the whole object is reclassified from A to B. This move does not violate smoothness within the region represented by the Potts model. But, the soft intermediate state $y_i' = y_j' = (\frac{1}{2},\frac{1}{2},0)$ will prevent this move in bi-linear case

$$P_{\text{BL}}(y_i', y_j') = \frac{1}{2} \quad > \quad 0 = P_{\text{BL}}(y_i, y_j) = P_{\text{BL}}(y_i'', y_j'')$$

while quadratic relaxation assigns zero loss for all states during this move. On the other hand, the example in Figure 4.2(b) shows a move problematic for the quadratic relaxation. Two neighboring pixels have labels $y_i = (1,0,0)$ and $y_j = (0,0,1)$ corresponding to the boundary of objects A and C. The second object attempts to change from C to B. This move does not affect the discontinuity between two pixels, but quadratic relaxation prefers that the second object is stuck in the intermediate state $y_j' = (0,\frac{1}{2},\frac{1}{2})$

$$P_{\text{Q}}(y_i, y_j') = \frac{3}{4} \quad < \quad 1 = P_{\text{Q}}(y_i, y_j) = P_{\text{Q}}(y_i, y_j'')$$

while bi-linear relaxation $P_{\text{BL}}(y_i, y_j) = 1$ remains constant as y_j changes.

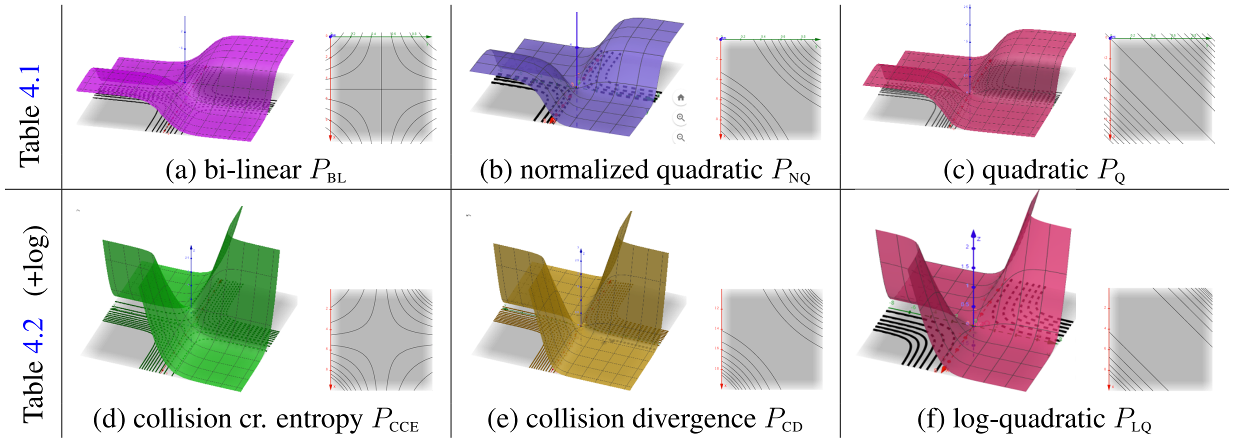
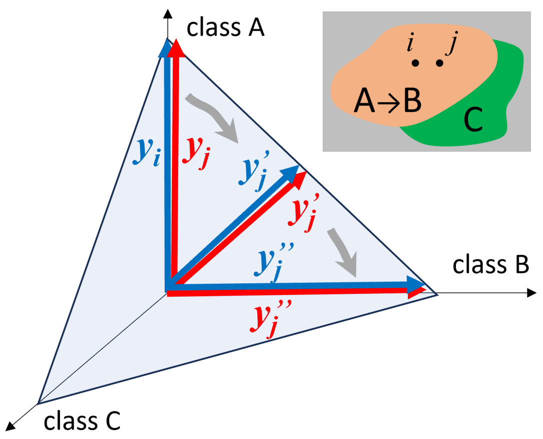


Figure 4.1: Second-order Potts relaxations in Tables 4.1 and 4.2: interaction potentials P for pairs of predictions (σ_i, σ_j) in (4.4) or pseudo-labels (y_i, y_j) in (2.6) are illustrated for $K = 2$ when each prediction σ_i or label y_i, i.e. distribution in Δ^2, can be represented by a single scalar as $(x, 1-x)$. The contour maps are iso-levels of $P((x_i, 1-x_i), (x_j, 1-x_j))$ over domain $(x_i, x_j) \in [0,1]^2$. The 3D plots above illustrate the potentials P as functions over pairs of "logits" $(l_i, l_j) \in \mathbb{R}^2$ where each scalar logit l_i defines binary distribution $(x_i, 1-x_i)$ for $x_i = \frac{1}{1+e^{-2l_i}} \in [0,1]$.

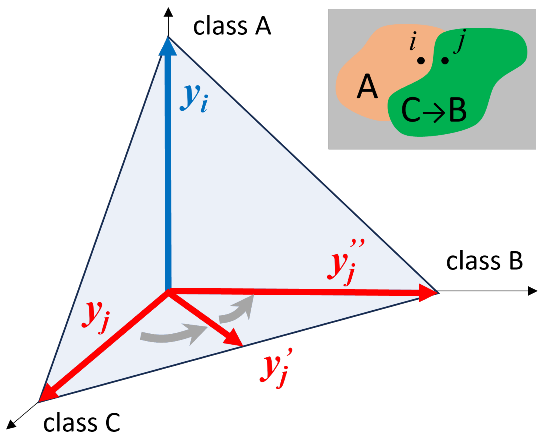
We propose a new relaxation, *normalized quadratic* in Table 4.1. Normalization leads to equivalence between quadratic and bi-linear formulations combining their benefits. As easy to check, normalized quadratic relaxation P_{NQ} does not have local minima in both examples of Figure 4.2. Table 4.2 also proposes "logarithmic" versions of the relaxations in Table 4.1 composing them with function $-\ln(1-x)$. As illustrated by Figure 4.1, the logarithmic versions in (d-f) addresses the "vanishing gradients" evident in (a-c).

collision cross entropy	**log-quadratic**
$P_{\text{CCE}}(p,q) \;:=\; -\ln p^\top q$	$P_{\text{LQ}}(p,q) \;:=\; -\ln\left(1 - \frac{\|p-q\|^2}{2}\right)$
collision divergence	
$P_{\text{CD}}(p,q) \;:=\; -\ln \frac{p^\top q}{\|p\|\|q\|} \;\equiv\; -\ln\left(1 - \frac{1}{2}\left\|\frac{p}{\|p\|} - \frac{q}{\|q\|}\right\|^2\right)$	

Table 4.2: Log-based Potts relaxations, see Figure4.1(d,e,f)



(a) both pixels change (A $\to$ B)



(b) only pixel q changes (C $\to$ B)

Figure 4.2: Examples of "moves" for neighboring pixels $\{i, j\} \in \mathcal{N}$. Their (soft) pseudo-labels y_i and y_j are illustrated on the probability simplex Δ^K for $K = 3$. In (a) both pixels i and j are inside a region/object changing its label from A to B. In (b) pixels i and j are on the boundary between two regions/objects; one is fixed to class A and the other changes from class C to B.

4.3 Optimization algorithms

In this section, we will focus on the optimization of (2.6) in steps iterating optimization of y and σ. The network parameters are optimized by standard stochastic gradient descent in all our experiments. Pseudo-labels are also estimated online using a mini-batch. To solve y at given σ for the loss below, it is a large-scale constrained convex problem.

$$\eta \sum_{i \notin S} H(\sigma_i, y_i) \; + \; \lambda \sum_{ij \in \mathcal{N}} P(y_i, y_j)$$

While there are existing general solvers to find global optima, such as projected gradient descent, it is often too slow for practical usage. Instead, we reformulate our problem to avoid the simplex constraints so that we can use standard gradient descent in PyTorch library accelerated by GPU. Specifically, instead of directly optimizing y, we optimize a set of new variables $\{l_i \in \mathbb{R}^K, i \in \Omega\}$ where y_i is computed by $softmax(l_i)$. Now, the simplex constraint on y will be automatically satisfied. Note that the hard constraints on scribble regions still need to be considered because the interaction with unlabeled regions through pairwise terms will influence the optimization process. Inspired by [239], we can reset $softmax(l_i)$ where $i \in S$ back to the ground truth at the beginning of each step of the gradient descent.

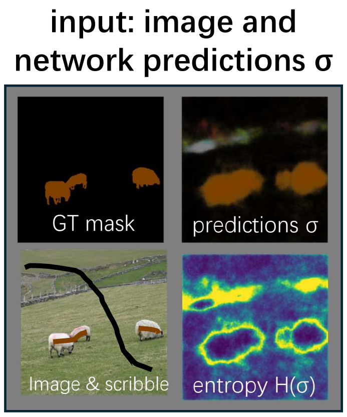
However, the original convex problem now becomes non-convex due to the Softmax operation. Thus, initialization is important to help find better local minima or even the global optima. Empirically, we observed that the network output logit can be a fairly good initialization. The quantitative comparison uses a special quadratic formulation where closed-form solution and efficient solver [1, 78] exist. We compute the standard soft Jaccard index for the pseudo-labels between the solutions given by our solver and the global optima. The soft Jaccard index is 99.2% on average over 100 images. Furthermore, our experimental results for all other formulations in Figure 4.3, 4.5, and the experimental section confirm the effectiveness of our optimization solver. In all experiments, the number of gradient descent steps for solving y is 200 and the corresponding learning rate is 0.075. To test the robustness of the number of steps here, we decreased 200 to 100 and the mIoU on the validation set just dropped from 71.05 by 0.72. This indicates that we can significantly accelerate the training without much sacrifice of accuracy.

4.4 Experimental results

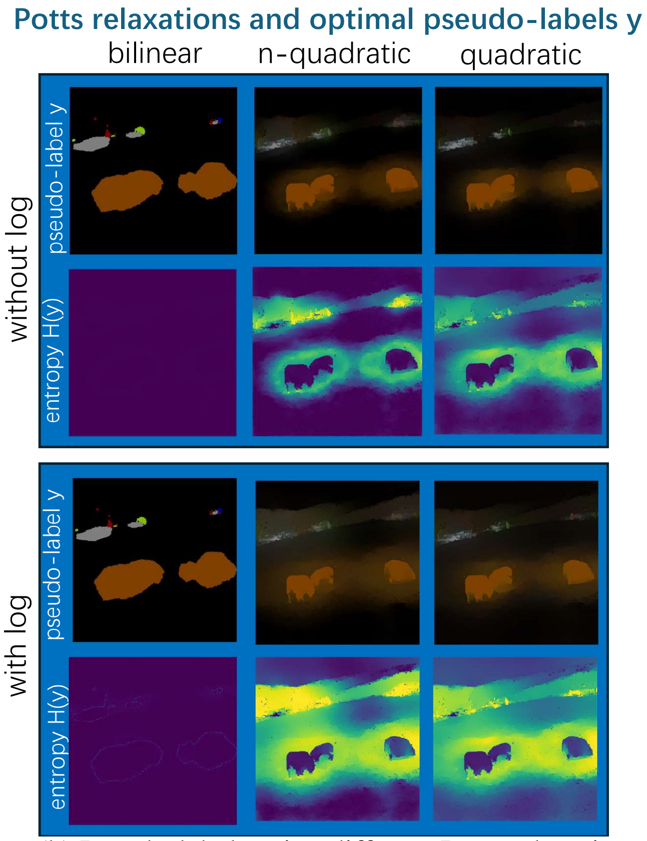
We conducted comprehensive experiments to demonstrate the choice of each element (cross-entropy, pairwise term, and neighborhood) in the loss and compare our method to the state-of-the-art. In Section 4.4.2, quantitative results are shown to compare different Potts relaxations. The qualitative examples are shown in Figure 4.3. Then we compare several cross-entropy terms in Section 4.4.3. Besides, we also compare our soft self-labeling approach on the nearest and dense neighborhood systems in Section 4.4.4. We summarized the results in Section 4.4.5. In the last section, we show that our method achieves the SOTA and even can outperform the fully-supervised method.

4.4.1 Experimental settings

Dataset and evaluation We mainly use the standard PASCAL VOC 2012 dataset [65] and scribble-based annotations for supervision [130]. The dataset contains 21 classes including background. Following the common practice [40, 205, 206], we use the augmented version which has 10,582 training images and 1449 images for validation. We employ the standard mean Intersection-over-Union (mIoU) on validation set as the evaluation metric. We also test our method on two additional datasets in Section 4.4.6. One is Cityscapes [49] which is built for urban scenes and consists of 2975 and 500 fine-labeled images for training and validation. There are 19 out of 30 annotated classes for semantic segmentation. The other one is ADE20k [235] which has 150 fine-grained classes. There are 20210 and 2000, images for training and validation. Instead of scribble-based supervision, we followed [129] to use the block-wise annotation as a form of weak supervision.



(a) Image, GT & input



(b) Pseudo-labels using different Potts relaxation

Figure 4.3: Illustration of the difference among Potts relaxations. The visualization of soft pseudo-labels uses the convex combination of RGB colors for each class weighted by pseudo-label itself.

Implementation details We adpoted DeepLabv3+ [42] framework with two backbones, ResNet101 [90] and MobileNetV2 [183]. We use ResNet101 in Section 4.4.6, and use MobileNetV2 in other sections for efficiency. All backbone networks (ResNet-101 and MobileNetV2) are pre-trained on Imagenet [58]. Unless stated explicitly, we use batch 12 as the default across all the experiments. Following [205], we adopt two-stage training, unless otherwise stated, where only the cross-entropy loss on scribbles is used in the first stage. The optimizer for network parameters is SGD. The learning rate is scheduled by a polynomial decay with a power of 0.9. Initial learning is set to 0.007 in the first stage and 0.0007 in the second phase. 60 epochs are used to train the model with different losses where hyperparameters are tuned separately for them. For our best result, we use $\eta = 0.3$, $\lambda = 6$, H_{CCE} and P_{CD}.

	scribble length ratio				
	0	0.3	0.5	0.8	1.0
P_{BL}	56.42	61.74	63.81	65.73	67.24
P_{NQ}	59.01	65.53	67.80	70.63	71.12
P_{Q}	58.92	65.34	67.81	70.43	71.05
P_{CCE}	56.40	61.82	63.81	65.81	67.41
P_{CD}	59.04	65.52	67.84	70.93	71.22
P_{LQ}	59.03	65.44	67.81	70.80	71.21

Table 4.3: Comparison of Potts relaxations with self-labeling. mIoUs on validation set are shown here.

4.4.2 Comparison of Potts relaxations

To compare different Potts relaxations under the self-labeling framework, we need to choose one cross-entropy term. Motivated by the properties and empirical results shown in Section 4.4.3, we use H_{CCE}. The neighborhood system is the nearest neighbors. The color intensity bandwidth in the Potts model is set to 9 across all the experiments on Pascal VOC 2012 and 3 for Cityscapes and ADE20k datasets. To determine the bandwidths, we first randomly picked 100 images from the training set. Then we calculated the soft pseudo-labels with H_{CCE} and P_{Q} based on pretrained network outputs. Then we calculated the soft mIoU and used such metric to select the bandwidths. The quantitative results are in Table 4.3. First, One can see that the pairwise terms with logarithm are better than those without the logarithm because the logarithm may help with the gradient vanishing problem in softmax operation. Moreover, the logarithm does not like abrupt change across the boundaries, so the transition across the boundaries is smoother (see Figure 4.3). Note that it is reasonable to have higher uncertainty around the boundaries. Second, the results prefer the normalized version, which confirms the points made in Figure 4.2. Third, the simplest quadratic formulation P_{Q} can be a fairly good starting point to obtain decent results. Additionally, we specifically test $H_{\text{Q}} + P_{\text{Q}}$ due to the existing closed-form solution [1, 78]. Since the pseudo-labels generated from this formula tend to be overly soft, we explicitly add entropy terms during the training of network parameters and the mIoU goes up to 68.97% from 67.8%.

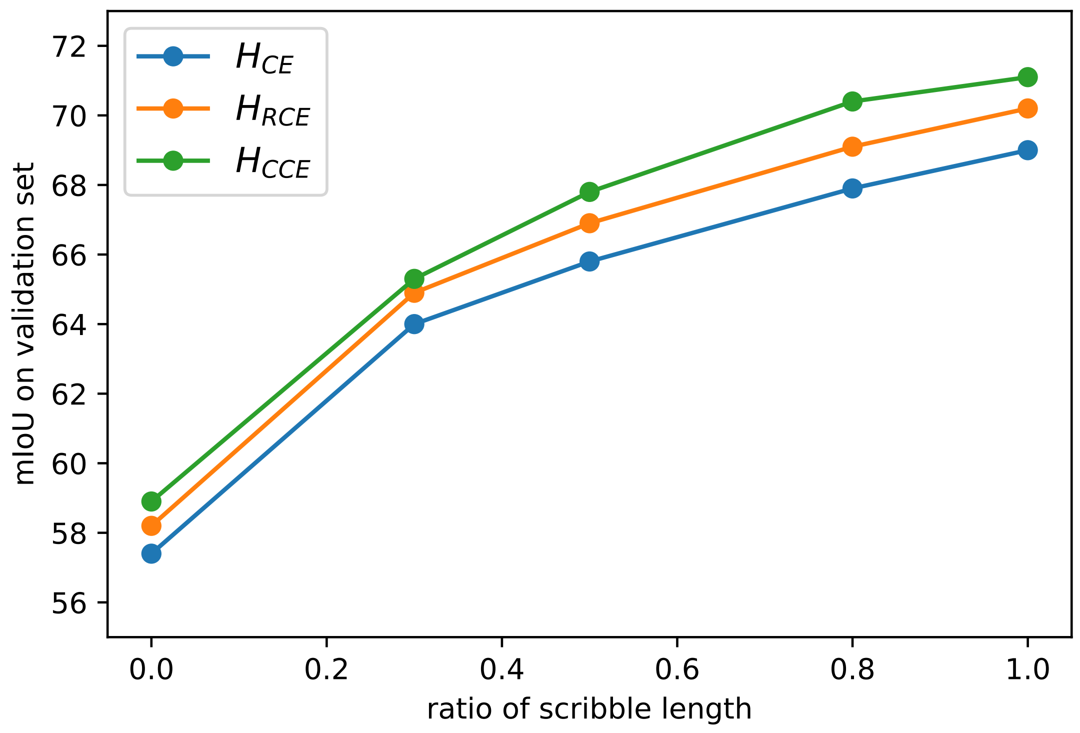


Figure 4.4: Comparison of cross-entropy terms.

4.4.3 Comparison of cross-entropy terms

In this section, we compare different cross-entropy terms while fixing the pairwise term to P_{Q} due to its simplicity and using the nearest neighborhood system. The results are shown in Figure 4.4. One can see that H_{CCE} performs the best consistently across different supervision levels, i.e. scribble lengths. Both H_{CCE} and H_{RCE} are consistently better than standard H_{CE} with a noticeable margin because they are more robust, as explained in Chapter 2, to the uncertainty in soft pseudo-labels when optimizing network parameters. We also test the performance of using $H_{\text{CCE}} + P_{\text{Q}}$ with hard pseudo-labels obtained via the $argmax$ operation on the soft ones. The mIoU on the validation set is 69.8% under the full scribble-length supervision.

4.4.4 Comparison of neighborhood systems

Until now, we only used the four nearest neighbors for the pairwise term. In this section, we also use the dense neighborhood and compare the results under the self-labeling framework.

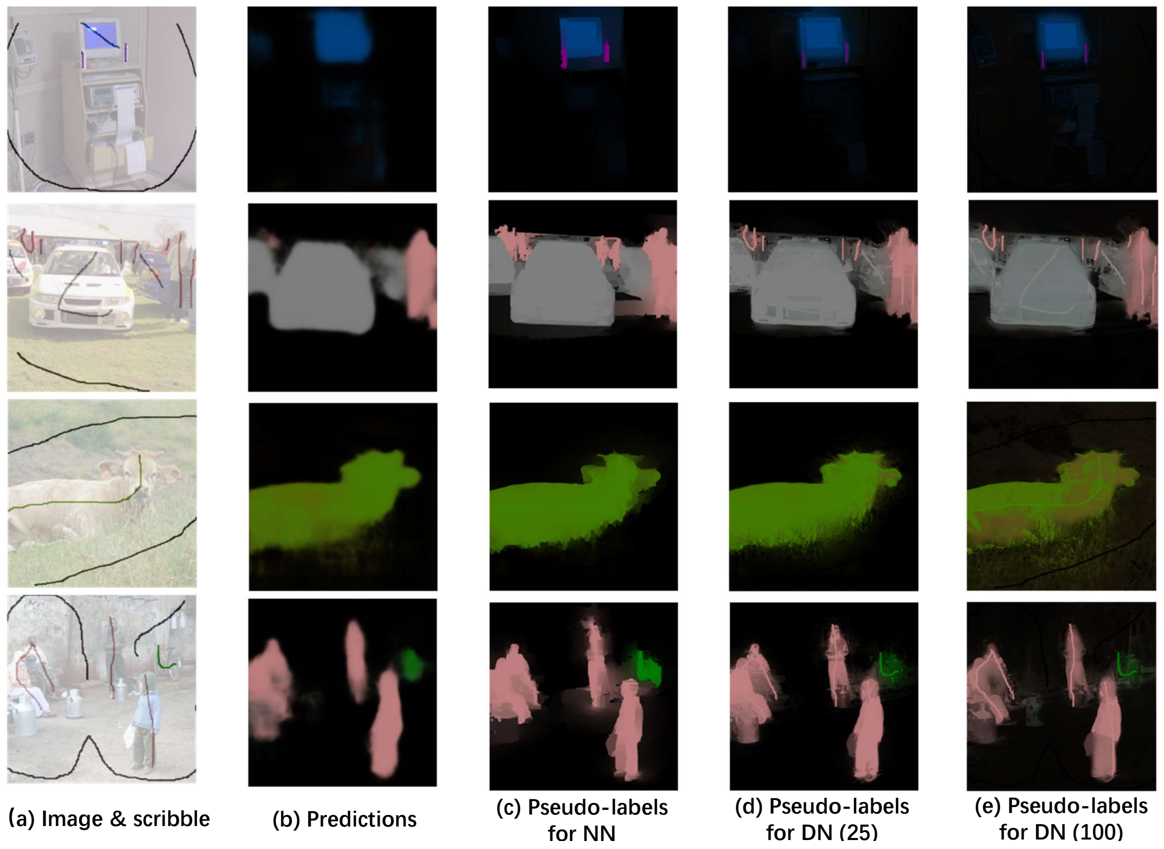


Figure 4.5: Pseudo-labels generated from given network predictions using different neighborhoods.

Firstly, to optimize the pseudo-labels for the dense neighborhood, we still use the gradient descent technique as detailed in Section 4.3. The gradient computation employs the bilateral filtering technique following [205]. For the pairwise term, we use P_{Q}. The cross-entropy term is H_{CCE}. Note that the bilateral filtering technique only supports quadratic pairwise terms, i.e. P_{BL} and P_{Q}. Since P_{BL} leads to hard solutions, P_{Q} is the only practical choice for soft self-labeling. We obtained 71.1% mIoU on nearest neighbors while only getting 67.9% on dense neighborhoods (bandwidth is 100). Some qualitative results are shown in Figure 4.5. Clearly from this figure one can see that a larger neighborhood size induces lower-quality pseudo-labels. A possible explanation is that the Potts model gets closer to cardinality/volume potentials when the neighborhood size becomes larger [218]. The nearest neighborhood is better for edge alignment and thus produces cleaner results.

		$\mathcal{N}$	
		NN	DN
GD		67.0	69.5* [206]
SL	hard	69.6* [141]	63.1 [130]
	soft	**71.1**	67.9

Table 4.4: Summary of comparisons. "$*$" stands for the reproduced results from their code repository.

4.4.5 Soft self-labeling vs. hard self-labeling vs. gradient descent

In this section, we give a summary in Table 4.4 as to what is the best framework for the WSSS based on losses regularized by the Potts model. Firstly, to directly optimize the network parameters via stochastic gradient descent on the regularized loss, one needs a larger neighborhood size. One possible explanation is that a larger neighborhood size induces a smoother Potts model and it helps the gradient descent [142]. However, larger neighborhood size is not preferred in the self-labeling framework. If we use Potts model on nearest neighborhoods, the self-labeling optimization should be applied and one should use soft pseudo-labels instead of hard ones. Note that with proper optimization the advantage of the Potts model on small neighborhood size can show up. In Figure 4.6, we also compare these approaches across different scribble lengths.

4.4.6 Comparison to SOTA

In this section, we use a different network architecture, ResNet101, to fairly compare our method with the current state-of-the-art. We only compare the results before applying any post-processing steps. The results are shown in Table 4.5. Note that our results can outperform the fully-supervised method when using 12 as the batch size. We also observe that a larger batch size usually improves the results quite a lot. Our results with 12 batch size can outperform several SOTA methods which use 16 batch size. We also conducted experiments to compare our method to the SOTA on two additional datasets as shown in Table 4.6.

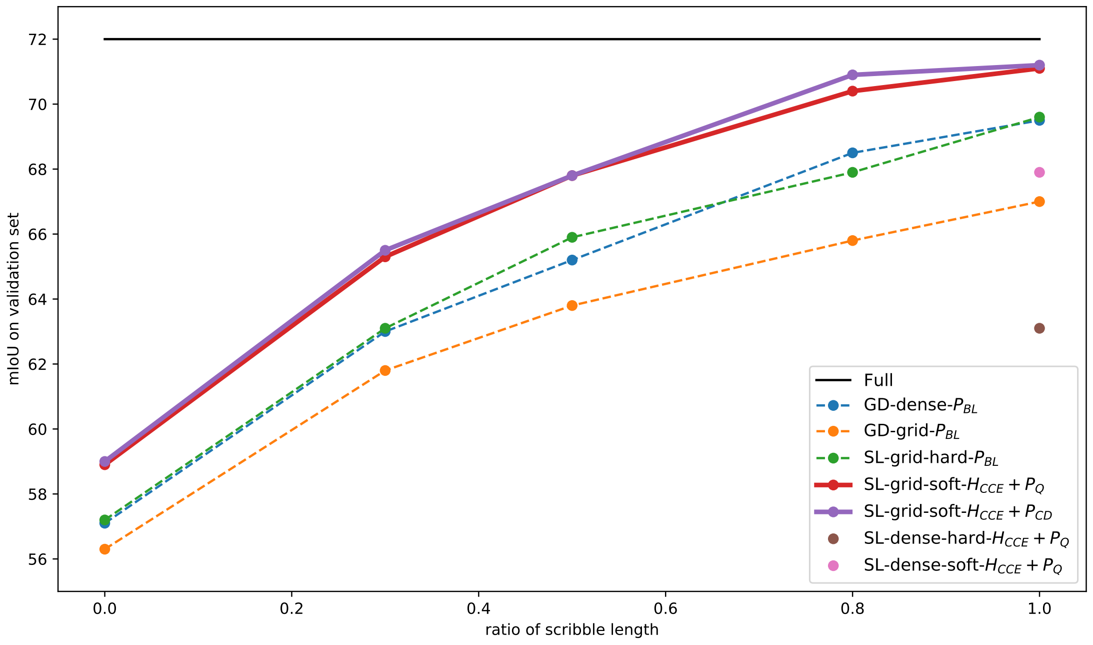


Figure 4.6: Comparison of different methods using Potts relaxations. The architecture is DeeplabV3+ with the backbone MobileNetV2.

Method	Architecture	Batchsize	Optimization			$\mathcal{N}$	mIoU
			GD	SL			
				hard	soft		
Full supervision							
Deeplab* [42]	V3+	16	✓	-	-	-	78.9
Deeplab* [42]	V3+	12	✓	-	-	-	76.6
Deeplab [41]	V2	12	✓	-	-	-	75.6
Scribble supervision							
Architectural modification							
BPG [220]	V2	10	✓	-	-	-	73.2
URSS [160]	V2	16	✓	-	-	-	74.6
SPML [106]	V2	16	✓	-	-	-	74.2
PSI [226]	V3+	-	-	-	✓	-	74.9
SEMINAR [39]	V3+	12	✓	-	-	-	76.2
TEL [129]	V3+	16	-	-	✓	-	77.1
Loss modification - Potts relaxations							
ScribbleSup [130]	VGG16(V2)	8	-	✓	-	DN	63.1
DenseCRF loss* [206]	V3+	12	✓	-	-	DN	75.8
GridCRF loss* [141]	V3+	12	-	✓	-	NN	75.6
NonlocalCRF loss* [219]	V3+	12	✓	-	-	SN	75.7
$\mathbf{H}_{\text{CCE}} + \mathbf{P}_{\text{Q}}$	V3+	12	-	-	✓	NN	77.5
$\mathbf{H}_{\text{CCE}} + \mathbf{P}_{\text{CD}}$	V3+	12	-	-	✓	NN	**77.7**
$\mathbf{H}_{\text{CCE}} + \mathbf{P}_{\text{CD}}$ (no pretrain)	V3+	12	-	-	✓	NN	76.7
$\mathbf{H}_{\text{CCE}} + \mathbf{P}_{\text{CD}}$	V3+	16	-	-	✓	NN	**78.1**
$\mathbf{H}_{\text{CCE}} + \mathbf{P}_{\text{CD}}$ (no pretrain)	V3+	16	-	-	✓	NN	77.6

Table 4.5: Comparison to SOTA methods (without CRF postprocessing) on scribble-supervised segmentation. The numbers are mIoU on the validation dataset of Pascal VOC 2012 and use full-length scribble. The backbone is ResNet101 unless stated otherwise. V2: deeplabV2. V3+: deeplabV3+. $\mathcal{N}$: neighborhood. "$*$": reproduced results. GD: gradient descent. SL: self-labeling. "no pretrain" means the segmentation network is not pretrained using cross-entropy on scribbles.

Method	Architecture	Cityscapes	ADE20k
Full supervision			
Deeplab [42]	V3+	80.2	44.6
Block-scribble supervision			
DenseCRF loss [206]	V3+	69.3	37.4
GridCRF loss* [141]	V3+	69.5	37.7
TEL [129]	V3+	71.5	39.2
$\mathbf{H}_{\text{CCE}} + \mathbf{P}_{\text{CD}}$	V3+	72.4	39.7

Table 4.6: Comparison to SOTA methods (without CRF postprocessing) on segmentation with block-scribble supervision. The numbers are mIoU on the validation dataset of cityscapes [49] and ADE20k [235] and use 50% of full annotations for supervision following [129]. The backbone is ResNet101. "$*$": reproduced results. All methods are trained in a single-stage fashion.

Chapter 5

Confluence prior for unsupervised vessel tree estimation

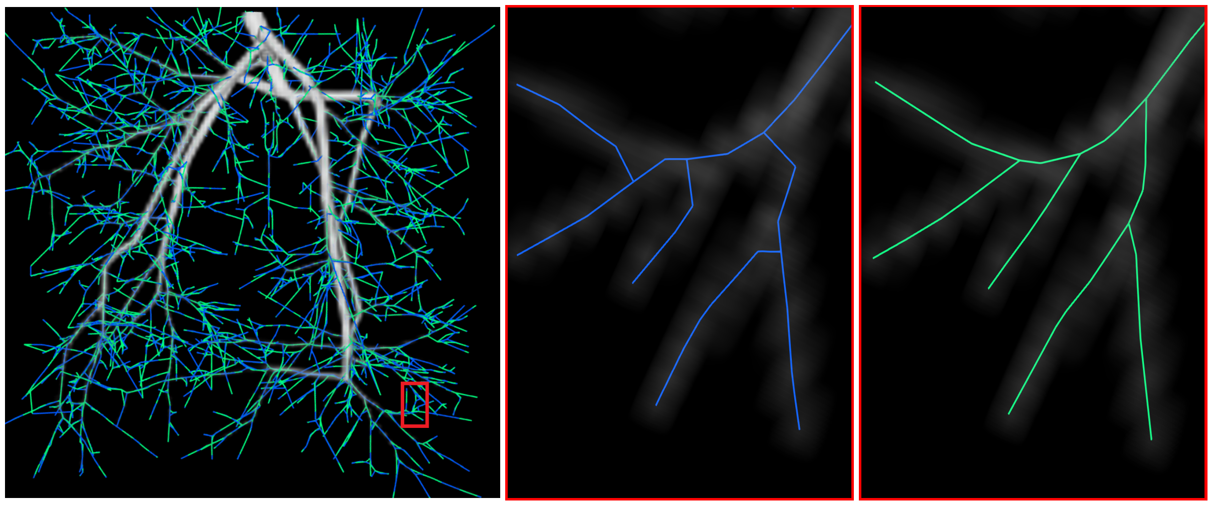
5.1 Introduction

This chapter is focused on unsupervised vessel tree estimation in large volumes containing numerous near-capillary vessels and thousands of bifurcations, see Figure 5.1, 5.13. Around 80% of the vessels in such data have sub-voxel diameter resulting in *partial volume* effects such as contrast loss and gaps. Besides the topological accuracy of trees reconstructed from such challenging imagery, we are particularly interested in the accurate estimation of bifurcations due to their importance in biomedical and pharmaceutical research.

5.1.1 Unsupervised vasculature estimation methods

Unsupervised vessel tree estimation methods for complex high-resolution volumetric vasculature data combine low-level vessel filtering and algorithms for computing global tree structures based on constraints from anatomy, geometry, physics, *etc*. Below we review the most relevant standard methodologies.

Low-level vessel estimation: Anisotropy of tubular structures is exploited by standard vessel filtering techniques, *e.g*. Frangi *et al*. [70]. Combined with non-maximum suppression, local tubularity filters provide estimates for vessel centerline points and tangents, see Figure 5.2(a). Technically, elongated structures can be detected using intensity Hessian spectrum [70], *optimally oriented flux* models [122, 211], steerable filters [71], path operators [146] or other anisotropic



(a) synthetic raw data with two trees (blue & green)

(b) geodesic graph MST [76, 225, 211, 150]

(c) confluent tree reconstruction

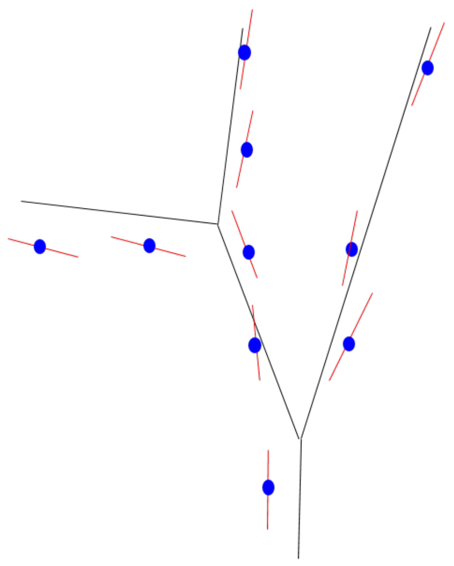
Figure 5.1: *Synthetic example*: (a) raw 3D data with blue & green reconstructed trees, see also zoom-ins (b,c). The blue tree (a,b) is an MST on a geodesic tubular graph. The green tree (a,c), is a *minimum arborescence* on a directed *confluent tubular graph*, see section 5.3.

models. Dense local vessel detections can be denoised using curvature regularization [161, 143]. Prior knowledge about divergence or convergence of the vessel tree (arteries vs veins) can also be exploited to estimate an *oriented* flow pattern [232], see Figure 5.2(b).

Thinning: One standard approach to vessel topology estimation is via *medial axis* [191]. This assumes known vessel segmentation (volumetric mask) [145], which can be computed only for relatively thick vessels. Well-formulated segmentation of thin structures requires Gaussian- or min-curvature surface regularization that has no known practical algorithms. Segmentation is particularly unrealistic for sub-voxel vessels.

Geodesics and shortest paths: Geodesics [60, 38] and shortest paths [67] are often used for AB-interactive reconstruction of vessels between two specified points. A vessel is represented by the shortest path with respect to some anisotropic continuous (Riemannian) or discrete (graph) metric based on a local tubularity measure. Interestingly, the minimum path in an “elevated" search space combining spatial locations and radii can simultaneously estimate the vessel’s centerline and diameter, implicitly representing vessel segmentation [128, 15]. Unsupervised methods widely use geodesics as their building blocks.

Spanning trees: The standard graph concept of a *minimum spanning tree* (MST) is well suited for unsupervised reconstruction of large trees with unknown complex topology [76, 225, 211, 150].


(a) Frangi filtering [70]

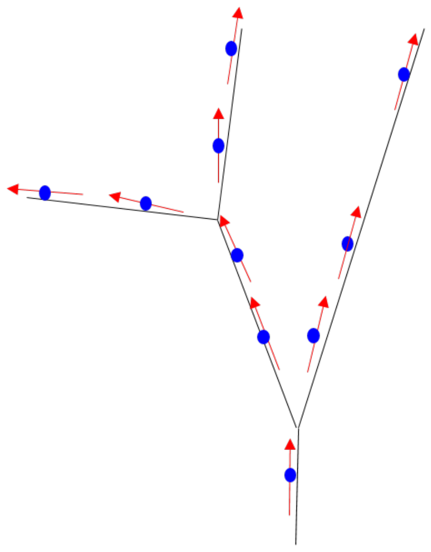
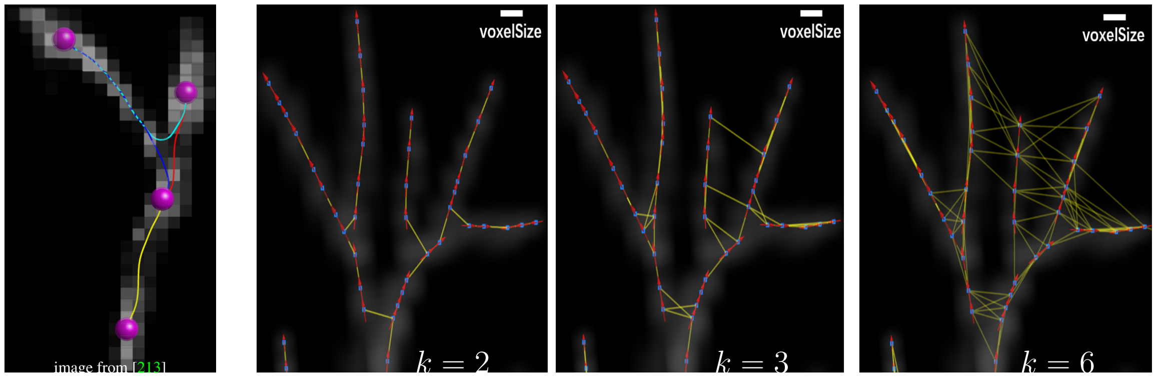

(b) *oriented* flow pattern [232]

Figure 5.2: *Low-level vessel estimation*: True centerline is black. Blue voxels in (a) are local maxima of some tubularity measure [70, 122, 211, 71] in the direction orthogonal to the estimated centerline tangents (red). Regularization [161, 143] can estimate subpixel centerline points (b) and oriented tangents [232] (red flow field).



(a) sparse graph (b) semi-dense geodesic tubular graphs with different size NN systems

Figure 5.3: Examples of standard *geodesic tubular graphs* for vascular image data: A graph from [213] in (a) uses (purple) nodes connected by undirected edges corresponding to shortest paths (geodesics) w.r.t. tubularity-based Riemanninian metric. Alternatively, (b) shows graphs where (blue) nodes correspond to densely-sampled voxels along the vessels [232]. Here the edges correspond to nearest neighbors (KNN) weighted by length of some spline interpolation. In both cases (a) and (b), near- or sub-voxel vessels have sparsely sampled bifurcations.

MST is closely related to the *shortest paths* and *geodesics* since its optimality is defined with respect to its length. Like shortest paths, globally optimal MST can be computed very efficiently. In contrast to the shortest paths, MST can reconstruct arbitrarily complex trees without user interaction.

The quality of MST vessel tree reconstruction depends on the underlying graph construction, see Figure 1.12 and 5.6(a). Graphs designed for reconstructing thin tubular structures as their spanning tree (or sub-tree) are often called *tubular graphs*. Typically, the nodes are "anchor" points generated by low-level vessel estimators, *e.g.* see Figure 5.2. Such anchors represent sparse [212] or semi-dense [143] samples from the estimated tree structure that may be corrupted by noise and outliers. Pairwise edges on a tubular graph typically represent distances or geodesics between the nodes, as in AB-interactive methods discussed earlier. Such graphs are called *geodesic tubular graphs*, see Figure 5.3.

There are numerous variants of tubular graph constructions designed to represent various thin structures as MST [76, 225, 211, 150] or shortest path trees [165]. There are also interesting and useful extensions of MST addressing tubular graph outliers, *e.g.* k-MST [213] and integer programming technique in [212]. Such approaches are more powerful as they seek minimum sub-trees that can automatically exclude outliers. However, the corresponding optimization problems are NP-hard and require approximations. Such methods are expensive compared to the low-order polynomial complexity of MST. They are not practical for dense reconstruction problems in high-resolution vasculature volumes.

5.1.2 Motivation and contributions

We are interested in unsupervised reconstruction of large complex trees from vasculature volumes resolving near-capillary details. Common geodesic approaches can not represent asymmetric smoothness at bifurcations, which have forms sensitive to flow orientation. Hence, standard methods produce vessel tree reconstructions with significant bifurcation artifacts, see Figure 5.1(b), 1.12(b) and 5.6(a). We define a general geometric property for oriented vessels, *confluence*, which is missing in prior art, and propose a practical graph-based reconstruction method enforcing it. The reconstructed confluent vessel trees have significantly better bifurcation accuracy. Our contributions are detailed below.

• We introduce *confluence* as a geometric property for overlapping oriented smooth curves in $\mathbb{R}^3$, *e.g*. representing blood-flow trajectories[1]. It is like “co-differentiability” or “co-continuity”. We define confluent vessel trees formed by overlapping oriented curves.

• We extend confluence to discrete paths and trees on directed tubular graphs where directed arcs/edges represent continuous oriented arcs/curves in $\mathbb{R}^3$. We propose a simple *flow-extrapolating circular arc* construction that guarantees ε-confluence, which approximates confluence. Our confluence constraint implies *directed* tubular graph with asymmetric edge weights, which is in contrast to standard *undirected* geodesic tubular graphs [104, 76, 225, 211, 150, 213, 143, 212, 232].

• We present an efficient practical algorithm for reconstructing *confluent vessel trees*. It uses *minimum arborescence* [63, 207] on our directed *confluent tubular graph* construction.

• Our experiments on synthetic and real data confirm that confluent tree reconstruction significantly improves bifurcation accuracy. We demonstrate qualitative and quantitative improvements via standard and new accuracy measures [2] evaluating tree structure, bifurcation localization, and bifurcation angles.

Our concept of confluent trees is general and our specific algorithm can be modified or extended in many ways, some of which are discussed in section 5.3. To explicitly address outliers, minimum arborescence on our confluent tubular graph can be replaced by optimal sub-tree

[1]Confluence is known in other contexts, *e.g*. rail tracks [99].

[2]For our dataset and implementation of evaluation metrics discussed in this chapter see https://vision.cs.uwaterloo.ca/data.

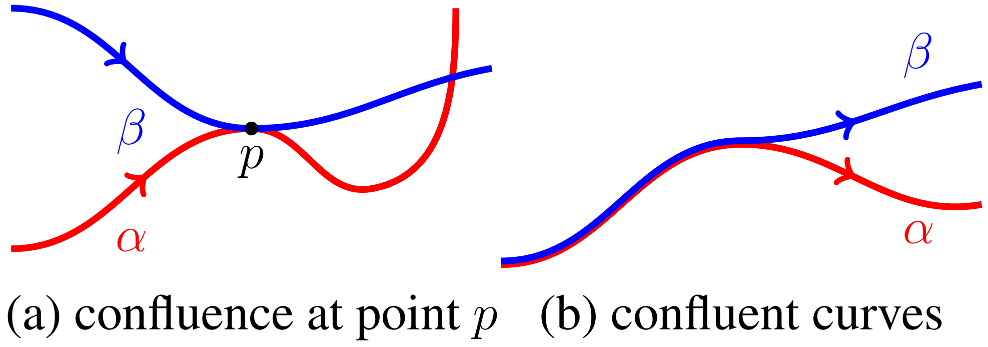


Figure 5.4: Confluence for oriented curves α and β.

algorithms [213, 212][3] or explicit outlier detection [143], but these approximation algorithms address NP-hard problems and maybe too expensive for large semi-dense tubular graphs we study in this work. While outlier detection is relevant, this work is not focused on this problem.

5.2 Confluence of Oriented Curves

This section introduces geometrically-motivated concept of smoothness for objects containing multiple oriented curves, such as vessel trees. We define *confluence* as follows.

DEFINITION 5 (confluence at a point). Two differentiable oriented curves $\alpha(t)$ and $\beta(\tau)$ are called confluent at a shared point p if for some $k > 0$

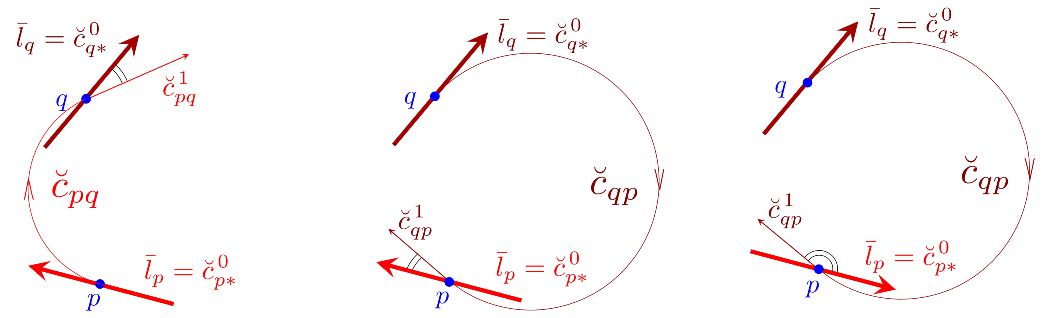
$$\alpha'(t_p) = k\,\beta'(\tau_p) \quad \text{where}$$

t_p and τ_p are s.t. $\alpha(t_p) = \beta(\tau_p) = p$, see Figure 5.4(a).

We will call two oriented curves *confluent* if they are confluent at all points they share, see Figure 5.4(b).

Our concept of confluence is closely related to the geometric $\mathcal{G}^1$*-continuity* [59, 69]. A curve α is called $\mathcal{G}^1$-continuous if at any point on the curve the slope orientation is continuous. Incidentally, the differentiability classes C^k are too restrictive as a $\mathcal{G}^1$-continuous curve can easily be not C^1 due to the curve parameterization. Note that $\mathcal{G}^1$-continuity is only defined for a *single* curve while our confluence extends it for a pair of curves and can be seen as “co-$\mathcal{G}^1$-continuity”.

[3]IP solver in [212] uses *minimum arborescence* as a subroutine.


$\bar{l}_q = \breve{c}^0_{q*}$
q
$\breve{c}^1_{pq}$
$\breve{c}_{pq}$
$\bar{l}_p = \breve{c}^0_{p*}$
p
$\bar{l}_q = \breve{c}^0_{q*}$
q
$\breve{c}_{qp}$
$\breve{c}^1_{qp}$
$\bar{l}_p = \breve{c}^0_{p*}$
p
$\bar{l}_q = \breve{c}^0_{q*}$
q
$\breve{c}_{qp}$
$\breve{c}^1_{qp}$
$\bar{l}_p = \breve{c}^0_{p*}$
p

(a) $\angle(\breve{c}^1_{pq}, \breve{c}^0_{q*}) \leq \varepsilon$ short confluent arc $\breve{c}_{pq}$

(b) $\angle(\breve{c}^1_{qp}, \breve{c}^0_{p*}) \leq \varepsilon$ long confluent arc $\breve{c}_{qp}$

(c) $\angle(\breve{c}^1_{qp}, \breve{c}^0_{p*}) \leq \varepsilon$ non-confluent arc $\breve{c}_{qp}$

Figure 5.5: Examples of directed flow-extrapolating circular arcs: (a) flow $\breve{c}^1_{pq}$ extrapolated by *confluent* arc $\breve{c}_{pq}$ from p to q is consistent with the local flow estimate $\bar{l}_q = \breve{c}^0_{q*}$ at point q as the angle between two vectors is small. Flow extrapolation from q to p in (b) requires another circular arc $\breve{c}_{qp}$ that belongs to a different circle defined by tangent $\bar{l}_q$. Two arcs $\breve{c}_{pq}$ and $\breve{c}_{qp}$ are not even co-planar if tangents $\bar{l}_p$ and $\bar{l}_q$ are not. The extrapolated flow $\breve{c}^1_{qp}$ is also consistent with the local flow estimate $\bar{l}_p = \breve{c}^0_{p*}$ at p, so that the reverse arc $\breve{c}_{qp}$ in (b) is confluent as well. In (c) the local flow estimate $\bar{l}_p$ at p is flipped and arc $\breve{c}_{qp}$ becomes non-confluent since the angle between $\breve{c}^1_{qp}$ and $\bar{l}_p = \breve{c}^0_{p*}$ is large. Note that arc $\breve{c}_{pq}$ in (c) differs from (a) but it must be non-confluent as well, see Theorem 6.

Our concept of confluence allows defining arbitrarily complex (continuous) *confluent vessel trees*. Such trees are formed by multiple oriented curves representing motion trajectories of blood particles from the common root to an arbitrary number of leaves where each pair of curves must be confluent. Figure 5.4(b) shows a simple example of a tree formed by two confluent curves with one bifurcation, which can be formally defined.

5.3 Confluent Tubular Graphs

Our discrete approach to reconstructing *confluent vessel trees* is based on efficient algorithms for directed graphs. Our "tubular" graph nodes correspond to a finite set of detected vessel points. We use discrete representation of oriented vessels as paths along directed edges or *directed arcs* (p, q) connecting the graph nodes. Each arc (p, q) on our tubular graph represents an oriented continuous "flow-extrapolating" curve in $\mathbb{R}^3$ from p to q. Such curves could be obtained from physical models based on fluid dynamics. For simplicity, this chapter is focused on oriented *circular arcs*, see section 5.3.1, motivated as the lowest-order polynomial splines capable of enforcing $\mathcal{G}^1$-continuity and confluence. In general, our confluent tubular graph construction can use higher-order flow-extrapolation models, *e.g*. cubic Hermite splines that are common in computer graphics and geometric modeling of motion trajectories.

By using circular *arcs* as flow-extrapolating curves, we introduce some ambiguity with "arcs" as the standard term for graph edges. However, this should not create confusion since there is a one-to-one relation between *directed arcs* on our tubular graph and the corresponding (circular) *oriented arcs* in $\mathbb{R}^3$. Note that both interpretations are oriented/directed. In all technically formal sentences, continuous or discrete interpretation of the "arc" is clear from the context. In more informal settings, both interpretations are often equally valid.

The rest of this Section is as follows. Oriented flow-extrapolating circular arcs between tubular graph nodes are introduced in section 5.3.1 where ε-confluence constraint is defined in the context of such arcs. We also define directed arc weights to represent the confluence constraint and the local costs of sending flow along these arcs. Geometric properties of confluent circular arcs are discussed in section 5.3.2. The algorithm estimating confluent vessel trees via *minimum arborescence* on our directed tubular graph is presented in section 5.3.3.

5.3.1 Confluent flow-extrapolating arcs

Formally, tubular graph $G = \langle V, A \rangle$ is based on a set of nodes/points V embedded in $\mathbb{R}^3$ representing semi-densely sampled centerlines of a tubular structure. $A \subseteq V^2$ is a set of directed arcs. For our tubular graph construction, each directed arc $(p, q) \in A$ represents some continuous oriented curve in $\mathbb{R}^3$ modelling flow-extrapolation from point p to point q, see Figure 5.5. As discussed earlier, this chapter is focused on oriented circular arcs as the simplest geometric model that can represent confluent vessels, even though higher-order geometric splines or physics-motivated curves are possible. Our specific construction uses flow-extrapolating circular arcs based on a set of unit vectors $\bar{L} = \{\bar{l}_p\}_{p\in V} \subset S^2$ representing flow direction estimates at the nodes, see Figure 5.2(b). Oriented circular arc $\breve{c}_{pq}$ is fit into starting point p, its flow orientation estimate $\bar{l}_p$, and the ending point q, see Figure 5.5. Formally, curve $\breve{c}_{pq}$ corresponds to a differentiable function

$$\breve{c}_{pq} \;:\; [0,1] \;\to\; \mathbb{R}^3$$

traversing points on a circular arc in the plane spanned by p, q, and vector $\bar{l}_p$ so that

$$\breve{c}_{pq}(0) = p\ , \quad \breve{c}_{pq}(1) = q\ , \quad \frac{\breve{c}'_{pq}(0)}{\|\breve{c}'_{pq}(0)\|} = \bar{l}_p \tag{5.1}$$

where derivative $\breve{c}'_{pq}(s)$ gives an oriented tangent.

For shortness, we define (oriented) unit tangents at the beginning and the end points of any flow-extrapolating arc $\breve{c}$ as

$$\breve{c}^{\,0} \;:=\; \frac{\breve{c}'(0)}{\|\breve{c}'(0)\|} \quad , \qquad \breve{c}^{\,1} \;:=\; \frac{\breve{c}'(1)}{\|\breve{c}'(1)\|} \; . \tag{5.2}$$

The definition of arc $\breve{c}_{pq}$ in (5.1) implies $\breve{c}^{0}_{pq} \;\equiv\; \bar{l}_p$ so that tangent $\breve{c}^{0}_{pq}$ is the same for any arc starting at given point p regardless of its end point q. Thus,

$$\breve{c}^{0}_{p*} \;\equiv\; \bar{l}_p \tag{5.3}$$

where the star $*$ represents an arbitrary end point. On the other hand, tangent $\breve{c}^{1}_{pq}$ at the end point q depends on the arc's starting point p. That is, generally,

$$\angle(\breve{c}^{1}_{pq}, \breve{c}^{1}_{rq}) \;\not\equiv\; 0 \qquad \text{if } \; p \neq r.$$

ε-Confluence Constraint: To constrain our tubular graph so that all feasible vessel trees are confluent, it suffices to enforce confluence of the arcs at the nodes where they meet. However, our simple flow-extrapolating circular arcs (5.1) can not be used to enforce the exact confluence. We use some threshold ε to introduce a relaxed version of confluence in Definition 5 for an arbitrary pair of adjacent arcs $\breve{c}_{pq}, \breve{c}_{qr}$ connecting points p, q and r

$$\angle(\breve{c}^{1}_{pq}, \breve{c}^{0}_{qr}) \;\leq\; \varepsilon.$$

In general, this is a high-order (triple clique) constraint. But, property (5.3) of our flow-extrapolating arc construction shows that the end point of the second arc $\breve{c}_{qr}$ is irrelevant. Indeed,

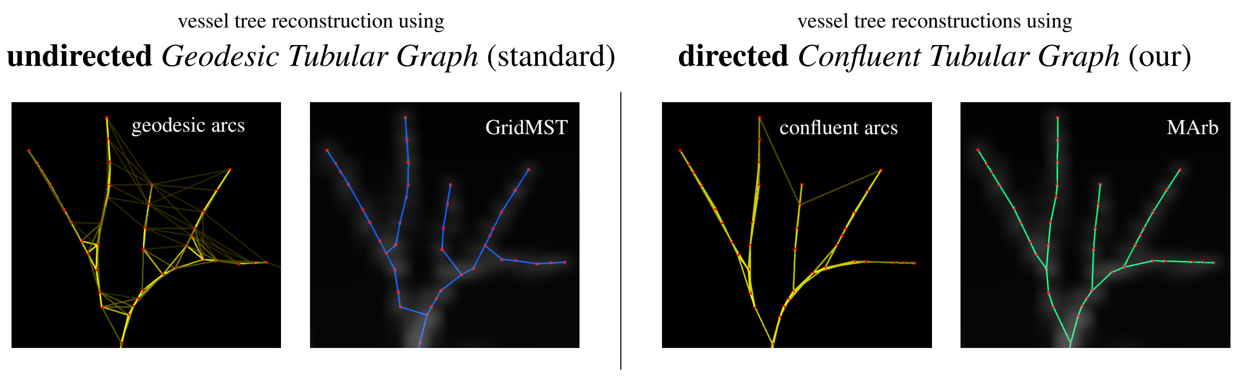
$$\angle(\breve{c}^{1}_{pq}, \breve{c}^{0}_{qr}) \;=\; \angle(\breve{c}^{1}_{pq}, \bar{l}_q) \;\equiv\; \angle(\breve{c}^{1}_{pq}, \breve{c}^{0}_{q*})$$

implying that our specific tubular graph construction allows to express confluence as a pairwise constraint

$$\angle(\breve{c}^{1}_{pq}, \breve{c}^{0}_{q*}) \;\leq\; \varepsilon \tag{5.4}$$

for any pair of points p, q. In essence, this becomes a constraint for our flow extrapolating arcs $\breve{c}_{pq}$ that can be called confluent if $\angle(\breve{c}^{1}_{pq}, \bar{l}_q) \;\leq\; \varepsilon$, see Figure 5.5.

To enforce ε-confluence constraint, our tubular graph can simply drop all non-confluent arcs. Thus, any directed vessel tree on our graph will be confluent by construction. This chapter explores the simplest approach to reconstructing confluent vessel trees as the *minimum arborescence* on our


vessel tree reconstruction using
undirected *Geodesic Tubular Graph* (standard)
geodesic arcs
GridMST
vessel tree reconstructions using
directed *Confluent Tubular Graph* (our)
confluent arcs
MArb

(a) MST (blue) for *Geodesic Tubular Graph* (yellow) [104, 213, 232] (b) Min Arborescence (green) for *Confluent Tubular Graph* (yellow)

Figure 5.6: Typical tree reconstruction examples for standard geodesic (a) and our *confluent* (b) tubular graphs. Arc weights are represented via thickness (yellow). The data is a (representative) crop with near-capillary vessels at a periphery of large volumes, *e.g*. Figure 5.1, 5.13. Sub-voxel vessels have bifurcations sparsely sampled by tubular graph nodes, as in Figure 5.3. MST (blue) on geodesic graph "short-cuts" most bifurcations. *Minimum arborescence* (green) on a directed tubular graph with *confluent* arcs (b), see section 5.3.1, reconstructs flow-consistent bifurcations.

directed tubular graph. In this case, instead of dropping non-confluent arcs, one can incorporate ε-confluence constraint directly into the cost of the corresponding directed graph arcs

$$w_{pq} \quad := \quad \begin{cases} \text{length}(\breve{c}_{pq}) & \text{if } \angle(\breve{c}^1_{pq}, \breve{c}^0_{q*}) \le \varepsilon \\ \infty & \text{otherwise.} \end{cases} \tag{5.5}$$

The reverse edge on our tubular graph has different weight $w_{qp} \neq w_{pq}$ because it corresponds to a different flow extrapolating arc $\breve{c}_{qp}$ that has a different length, see Figure 5.5(b). As an extension, our approach also allows "elastic" arc weights by adding integral of arc's curvature to its length in (5.5). It is also possible to impose soft penalties for the discrepancy between the extrapolated flow $\breve{c}^1_{pq}$ and flow estimate $\breve{c}^0_{q*} \equiv l_q$ in (5.5) based on physical, physiological, or other principles.

Note that higher-order (non-circular) extrapolation arcs $\breve{c}_{pq}$ can be constructed to fit the flow orientation estimates at both ends exactly, implying an exactly confluent graph. However, some non-trivial physiological constraints have to be imposed on the smoothness/curvature of such (non-circular) confluent arcs which should result in very long curves in cases like Figure 5.5(c). Thus, the confluence constraint will manifest itself similarly to the second line in (5.5).

5.3.2 Confluence and co-circularity

Specifically for oriented circular arcs, confluence implies several interesting properties and can be juxtaposed with the standard concept of *co-circularity* [161]. Assume some circular flow-extrapolating arc $\breve{c}_{pq}$ and its reverse $\breve{c}_{qp}$ defined by two oriented tangents $\bar{l}_p, \bar{l}_q$, see section 5.3.1 and Figure 5.5(a,b).

PROPERTY 4. The angle between $\breve{c}^1_{pq}$ and $\bar{l}_q$ is equal to the angle between $\breve{c}^1_{qp}$ and $\bar{l}_p$. That is,

$$\angle(\breve{c}^1_{pq}, \breve{c}^0_{qp}) \equiv \angle(\breve{c}^1_{qp}, \breve{c}^0_{pq}). \tag{5.6}$$

Proof. First, we only consider the circle going through p, q and tangential to some vector $\bar{\tau}_p$ as shown in Figure 5.7. Moving the tangent vector $\bar{\tau}_p$ in its direction along the circle yields another tangent vector $\bar{\tau}_q$ at q. We also translate $\bar{\tau}_p$ to q. By construction, ΔOpq is equilateral. Thus, $\angle Opq = \angle Oqp$. Since $\bar{\tau}_p$ and $\bar{\tau}_q$ are tangential to the circle, we have $\alpha + \angle Opq = \beta + \angle Oqp = \frac{\pi}{2}$, which obviously gives $\alpha = \beta$. WLOG, we assume vectors $\bar{\tau}_p$, $\bar{\tau}_q$ and $\bar{e}$ are all unit vectors. As $\alpha = \beta$, we have:

$$\bar{\tau}_q = 2[(\bar{\tau}_p \cdot \bar{e})\bar{e} - \bar{\tau}_p] + \bar{\tau}_p \tag{5.7}$$

Now, we consider the two circles going through both p and q while one is tangential to $\bar{l}_p$ and the other is tangential to the $\bar{l}_q$ as shown in Figure 5.8. Note that these two circles are not necessarily co-planar. Using (5.7), we can obtain

$$\breve{c}^1_{pq} = 2[(\breve{c}^0_{pq} \cdot \vec{e}_1)\vec{e}_1 - \breve{c}^0_{pq}] + \breve{c}^0_{pq} \tag{5.8}$$

$$\breve{c}^1_{qp} = 2[(\breve{c}^0_{qp} \cdot \vec{e}_1)\vec{e}_1 - \breve{c}^0_{qp}] + \breve{c}^0_{qp} \tag{5.9}$$

To prove (5.6), it is sufficient to prove equality of the two dot products which can be simplified using (5.8) and (5.9):

$$\breve{c}^1_{pq} \cdot \breve{c}^0_{qp} = 2(\breve{c}^0_{pq} \cdot \vec{e}_1)(\breve{c}^0_{qp} \cdot \vec{e}_1) - \breve{c}^0_{pq} \cdot \breve{c}^0_{qp} \tag{5.10}$$

$$\breve{c}^1_{qp} \cdot \breve{c}^0_{pq} = 2(\breve{c}^0_{qp} \cdot \vec{e}_1)(\breve{c}^0_{pq} \cdot \vec{e}_1) - \breve{c}^0_{qp} \cdot \breve{c}^0_{pq} \tag{5.11}$$

It is obvious that the RHS of (5.10) and (5.11) are equal. Therefore, these two angles are equal. □

Identity (5.6) implies the following.

Theorem 6. For circular flow extrapolating arcs, $\breve{c}_{pq}$ is confluent iff the reverse arc $\breve{c}_{qp}$ is confluent.

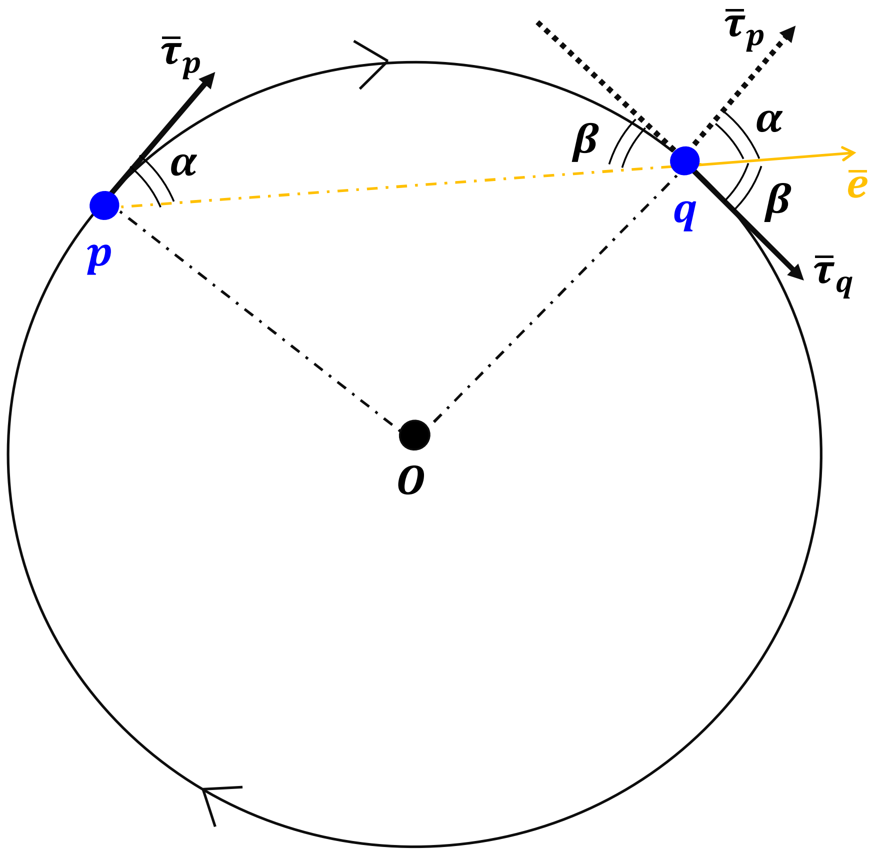


Figure 5.7: Illustration for (5.7). O is the center of the circle. $\bar{e}$ is a unit vector along pq.

This theorem shows that confluence of $\breve{c}_{pq}$ and $\breve{c}_{qp}$ in Figure 5.5(a,b) is not a coincidence. However, in general, such symmetry does not hold for non-circular flow-extrapolating arcs (higher order polynomial curves, etc). Also, Theorem 6 does not imply "undirectedness" of our confluent tubular graph construction using simple circular arcs. As follows from (5.5), $w_{pq} \neq w_{qp}$ since the reverse arcs $\breve{c}_{pq}$ and $\breve{c}_{qp}$ have different lengths regardless of confluence, see Figure 5.5(a,b).

Interestingly, our confluence constraint in case of *circular* oriented arcs can be related to an "oriented" generalization of *co-circularity* that was originally defined in [161] for 2D curves. In $\mathbb{R}^n$ co-circularity constraint can be defined for two *unoriented* tangent lines l_p and l_q at points p and q in a way similar to our definition of confluence for $\breve{c}_{pq}$ that is based on *oriented* tangents $\bar{l}_p$ and $\bar{l}_q$. Assume *unoriented* circle c_{pq} uniquely defined in $\mathbb{R}^n$ by a pair of points p, q and tangent l_p at the first point. If we use c^x to denote an unoriented unit tangent of circle c at any given point x,

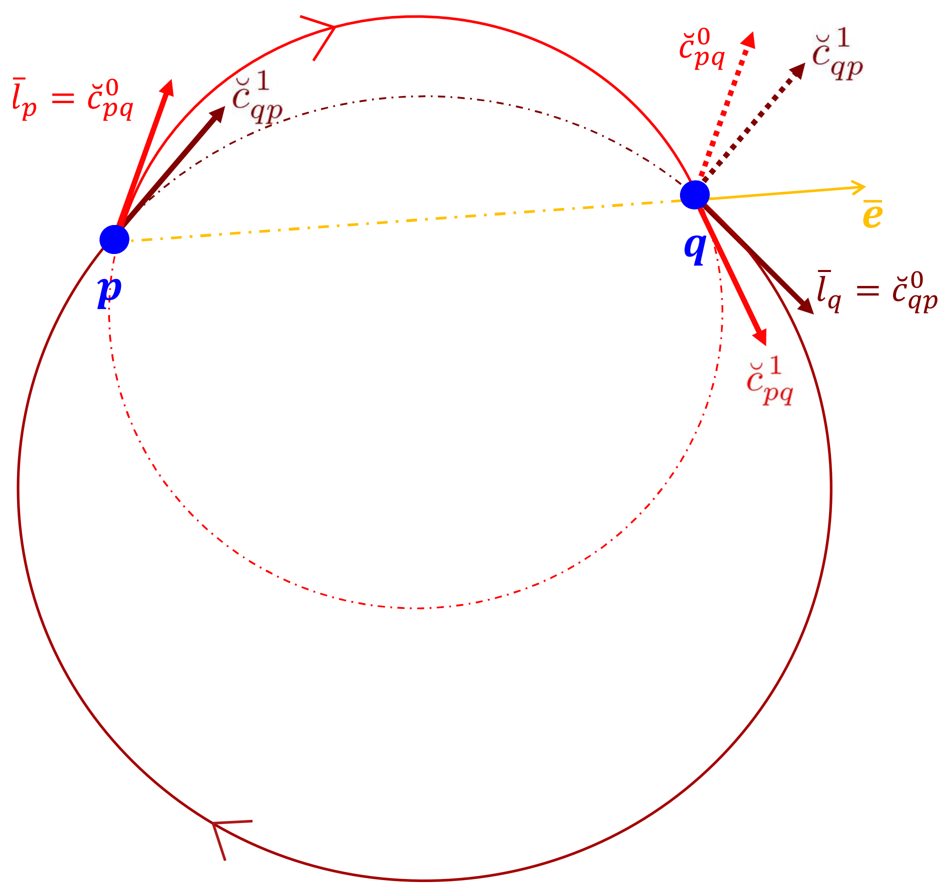


Figure 5.8: Two circles given by p,q, $\breve{c}^0_{pq}$ and $\breve{c}^0_{qp}$. Vectors in red are co-planar with the red circle while those in maroon are co-planar with the maroon circle.

then circle c_{pq} is uniquely defined by three conditions

$$c_{pq} \quad : \quad p \in c_{pq}, \;\; q \in c_{pq}, \;\; c^p_{pq} = l_p. \tag{5.12}$$

In general, circle c_{qp} is different as it is defined by tangent l_q at point q, that is $c^q_{qp} = l_q$. Then, co-circularity constraint for l_p and l_q can be defined as

$$\angle(c^q_{pq}, c^q_{qp}) \;\equiv\; \angle(c^p_{qp}, c^p_{pq}) \;\leq\; \epsilon \tag{5.13}$$

where $\angle(\cdot,\cdot)$ is the angle between two lines in contrast to the angle between vectors in the similar identity (5.6).

The difference between confluence for $\bar{l}_p$, $\bar{l}_q$ and co-circularity for l_p, l_q can be illustrated by the examples in Figure 5.5. Note that unoriented versions of $\bar{l}_p$, $\bar{l}_q$ are identical in all three examples (a,b,c) as they do not depend of the flip of orientation in (c). Thus, they are equally co-circular in (a,b,c). At the same time, oriented tangents are confluent in (a,b) while flipping orientation for $\bar{l}_p$ results in non-confluence in (c). The properties discussed above imply that confluence can be seen as oriented generalization of co-circularity [161].

PROPERTY 5. Confluence of oriented circular arcs $\breve{c}_{pq}$ or $\breve{c}_{qp}$, which are defined by oriented tangents $\bar{l}_p, \bar{l}_q$, implies co-circularity of the corresponding unoriented tangents l_p, l_q, but not the other way around.

Note that co-circularity constraint for *unoriented* circular arcs along a path on a tubular graph can enforce $\mathcal{G}^1$-smoothness within a single vessel branch. But, unoriented co-circularity enforces smoothness indiscriminately in all directions from a bifurcation point without resolving conflicts between multiple branches. This leads to artifacts observed on geodesic tubular graphs, see Figure 5.6(a). In contrast, the confluence constraint discriminates orientations of branches when enforcing smoothness at bifurcations, see Figure 5.6(b).

5.3.3 Confluent tree reconstruction algorithm

Algorithm 2 Confluent Tree Reconstruction

```
Require: Raw volumetric data and root location
1: Estimate a set of centerline points $V$ and directed flow estimates $\bar{L} = \{\bar{l}_p | p \in V\}$.
2: Build a set of oriented arcs $A \subseteq V \times V$, section 5.3.1
3: Build a confluent tubular graph $G$ by computing weights $w_{pq}$ for $(p, q) \in A$ using (5.5).
4: Return the Minimum Arborescence of $G$.
```

Our Confluent Tree Reconstruction Algorithm 2 is discussed below. It inputs raw volumetric data with a marked root of the tree. The algorithm has four steps. First, it runs a subroutine that estimates a set of points on the tree centerline V and oriented flow pattern at these points $\bar{L}$. We use a standard vector field estimation method [232] based on non-negative divergence constraint and regularization over the voxel-grid neighborhood. Since 80% of our large vasculature volumes are near-capillary vessels, the weak sub-voxel signal often results in missing data points and grid-based regularization fails to produce consistent flow orientations, particularly at the tree periphery. We modified [232] by (anisotropically) enlarging their regularization neighborhood, see Section 5.4.3, improving the quality of flow estimates $\bar{L}$ that helps to reconstruct confluent vessel trees.

Second, we build a set of oriented arcs between the points in V that correspond to directed edges on our tubular graph G. Our confluence constraint works well even with a complete graph $A = V \times V$. But, for efficiency, we restrict the neighborhood to K nearest neighbors (KNN). The running time is $\mathcal{O}(K|V|\log|V|)$ with k-d trees.

The last two steps compute a directed weight w_{pq} for all arcs $(p,q) \in A$ as described in section 5.3.1, and invoke a standard minimum arborescence algorithm that has complexity $\mathcal{O}(|A| + |V|\log|V|)$ [72]. In practice, the overall running time of our method for vessel tree reconstruction is dominated by the centerline localization and flow pattern estimation in the first step.

5.4 Experimental results

We use two baselines which we call NMS-MST and GridMST. NMS-MST uses the Frangi method [70] along with Non-Maximum Suppression (NMS) to obtain the centerline points and local unoriented tangent estimates. On top of these, NMS-MST uses the KNN ($K = 500$) graph of the centerline points to build the MST. Note that the KNN graph is symmetric such that a pair of nodes have an arc as long as one is a neighbor of the other. Here, the undirected edge weight is computed by the sum of two shorter arc lengths. GridMST uses [232] for estimating the centerline points and flow direction. Then, it also uses the KNN graph of the centerline points to build the MST.

GridArb uses the set of centerline points and flow estimates produced by [232], but it uses the confluent tubular graph to build the minimum arborescence (discussed in section 5.3.3). MArb exploits a modified version of [232] (see Section 5.4.3) and also uses the confluent tubular graph to build the minimum arborescence. We set $\varepsilon = \frac{\pi}{2}$ in (5.5) for all our experiments.

5.4.1 Validation Measures

Many validation measures rely on matching between the ground truth and the predicted tree. Matching algorithms could be separated into several groups. First, match the nodes of the trees independently based on a distance measure, *e.g.* [232, 211, 214], partial (local) sub-tree matching [75], or global tree matching approaches [110, 231, 46]. We base our evaluation approach on the first group of methods due to their efficiency and the size of our problem.

Centerline reconstruction quality. Our reconstructed tree is ideally the centerline of the vasculature. We compute the recall and fall-out statistics of the centerline points to evaluate the

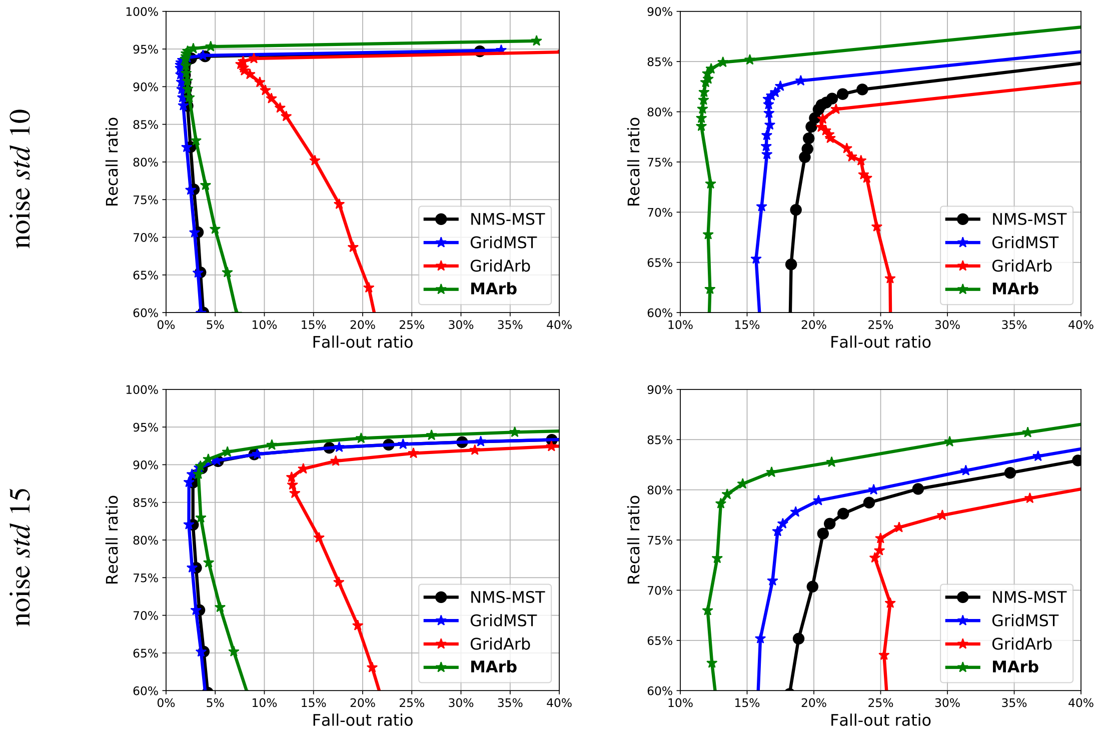
Method	**Flow estimates**	**Graph Weights**	**Tree Extraction**
NMS-MST	Frangi *et al.* [70]	standard geodesic	MST
GridMST	Zhang *et al.* [232]		
GridArb	Zhang *et al.* [232]	our confluent (5.5)	minimum arborescence
MArb	Modified [232], see Section 5.4.3		

Table 5.1: The summary of comparison between different methods.

reconstruction quality. To obtain the *centerline receiver operating characteristic (ROC) curve*, we generate a sequence of *recall/fall-out* points by varying the *detection threshold* parameter for the low-level vessel filter of Frangi *et al.* [70].

Similarly to [232], a specific point on the ground truth centerline is considered detected correctly (recall) iff it is located within $\max(r, \zeta)$ distance of a reconstructed tree where r is the radius of the corresponding ground truth vessel segment and $\zeta = \frac{\sqrt{2}}{2}$ voxel-size. A point on the reconstructed tree that is farther away than this distance is considered incorrectly detected (fall-out). Before computing the ROC curve we re-sample uniformly both the ground truth and reconstructed trees.

Bifurcation reconstruction quality. We introduce two separate metrics. First, we compute the *ROC* curve for only bifurcation points to assess the quality of detection. Second, we measure the median angular error at the reconstructed bifurcations to assess the accuracy, where we match *all* ground truth bifurcations to closet branching points on the detected tree regardless of their proximity and use the median rather than the average for greater stability. The *average* angular error introduced in [232] uses only correctly detected points to compute the bifurcation angular errors. Using such matching to compare different methods is unfair as for a particular detection threshold these methods correctly detect different sets of bifurcations. So, we match *all* ground truth bifurcations to closet branching points on the detected tree regardless of their proximity. For certain thresholds, this causes many incorrectly matched bifurcation and large errors. Despite that such statistic is influenced significantly by random matches, it is meaningful for comparing different reconstruction methods.



(a) centerline detection (b) bifurcation detection

Figure 5.9: Quantitative comparison. Our methods are denoted by MArb and GridArb. GridMST is the best result from [232].

5.4.2 Synthetic Data with Ground Truth

Zhang *et al*. [232] generated and published a dataset with ground truth using [84]. We found that the diversity of bifurcation angles is limited. The mean angle is $68°$ and *std* is $17°$. To increase the angle variance, we introduce a simple modification of vessel tree generation. When a new bifurcation is created from a point and existing line segment, we move the bifurcation towards one of the segment's ends chosen at random decreasing the distance by half. The new mean is $68°$ and *std* is $29°$. We generate 15 volumes $100 \times 100 \times 100$ with intensities between 0 and 512. The voxel size is 0.046 mm. We add Gaussian noise with *std* 10 and 15.

One of the major challenges in large-scale vessel tree reconstruction is the lack of ground truth. That complicates many interactive and supervised learning methods and makes evaluation hard. Zhang *et al*. [232] generated and published a dataset with ground truth using [84]. We used our newly generated 15 volumes $100 \times 100 \times 100$ with intensities between 0 and 512. Our new

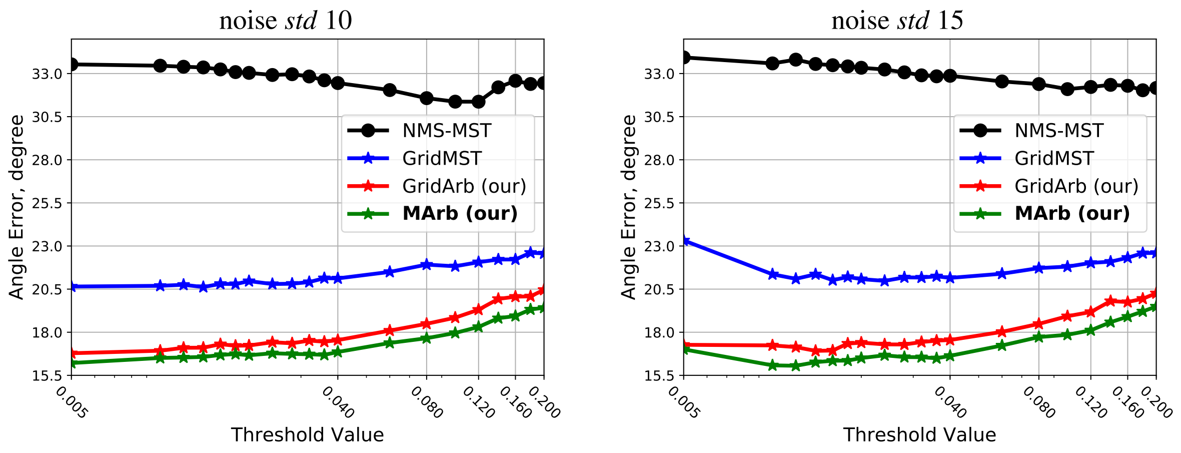


Figure 5.10: Median branching angular error for our methods (GridArb and MArb from section 5.3) outperform all competitors (NMS-MST and GridMST [232])

dataset has a larger variance of bifurcation angles. The voxel size is 0.046 mm. We add Gaussian noise with *std* 10 and 15.

Figure 5.9 compares the results of our methods with two competitors. One is the method of [232], another baseline is simple MST computed over non-maximum suppression of vessel filter output. All methods use essentially the same detection mechanism, *i.e*. Frangi *et al*. filter, so the centerline extraction quality does not differ much. On the other hand, our method significantly outperforms in the quality of bifurcation detection, see Figure 5.9 (b). This result is complemented by superior angular errors in Figure 5.10. We attribute this to the subvoxel accuracy and better reconstruction of bifurcation. A typical example is shown in Figure 5.1(b,c).

The GridArb performs competitively in terms of angular errors but gives the worst results in terms of centerline quality. This is due to the artifact caused by some inconsistent flow estimates near the tree periphery (see Figure 5.6 for concrete examples). In section 5.3.3 we argue that enlarging the regularization neighborhood helps improve the estimation of the flow orientation.

5.4.3 CRF Regularization Neighborhood

Our tree extraction method is based on a directed *confluent tubular graph* construction $G = \langle V, A \rangle$ presented in Section 3 of the paper. We proposed an approach that builds confluent flow-extrapolating arcs $\breve{c}_{pq}$ for our graph from estimated *oriented* flow vectors $\{\bar{l}_p \,|\, p \in V\}$. Specific flow orientations can be computed from Frangi filter outputs using standard MRF/CRF regularization methods [232] enforcing divergence (or convergence) of the flow pattern. However, as mentioned in Section 3.3 and Section 4, we modified [232] by anisotropically enlarging the regularization neighborhood to improve the estimates of flow orientations, which are important for our directed arc construction. The 26-grid neighborhood regularization used in [232] generates too many CRF connectivity gaps near the vessel tree periphery where the signal gets weaker. Such gaps result in flow orientation errors, see white vectors in the zoom-ins in Figure 5.12(a,b). While tree reconstruction on standard undirected geodesic tubular graphs, see Figure 5.12(a), are oblivious to such errors, our directed *confluent tubular graph* construction is sensitive to wrong orientations, see Figure 5.12(b). To address CRF gaps in the flow orientation estimator [232], we modified their 26-grid regularization neighborhood into *anisotropic KNN* based on Frangi's vessel tangents [70]. This significantly reduces orientation errors in $\{\bar{l}_p \,|\, p \in V\}$ and resolves confluent tubular graph artifacts, see Figure 5.12(c). We detail anisotropic KNN below.

CRF connectivity quality: Besides the size of the neighborhood K, anisotropic KNN system has another important hyper-parameter, aspect ratio ar. To select better parameters K and ar, we can evaluate CRF connectivity system $\mathcal{N}$ using ROC curves for synthetic vasculature volumes with ground truth. We consider an edge in $\mathcal{N}$ as *correct* iff the projections of its ends onto the ground truth tree have parent/descendant relationship. The *recall* is the portion of the ground truth tree covered by the correct edges. The *fall-out* is the ratio of incorrect edges to the total edge count.

As shown in Figure 5.11, simply increasing the size of neighborhood closes many gaps but, in the meantime, introduces a lot of spurious connections between different vessel branches. Thus, we propose to use anisotropic neighborhoods. Specifically, the regularization neighborhood is redefined as k anisotropic nearest neighbors instead of regular grid connectivity. This similar to the KNN except Mahalanobis distance is used. This modification addresses the issue giving the state-of-the-art result, see Figure5.12(c). To implement such anisotropic neighborhood system, we first built an isotropic KNN with some large K, eg. K=500. Then, for each node and its neighbors, we transformed the Euclidean distance into Mahalanobis distance based on the tangent direction on the node. After this, we selected K (eg. K=4) nearest neighbors for each node again based on the Mahalnobis distance. Note that such anisotropic neighborhood is symmetric since we consider a pair of nodes as neighbor as long as one is connected to the other.

5.4.4 High Resolution Microscopy CT

We use challenging microscopy computer tomography volume (micro-CT) of size $585 \times 525 \times 892$ voxels to qualitatively demonstrate the advantages of our approach. The data is a high-resolution image of mouse heart obtained *ex vivo* with the use of contrast. The resolution allows detecting nearly capillary level vessels, which are partially resolved (partial volume). Figure 5.13 shows the whole volume and reconstruction.

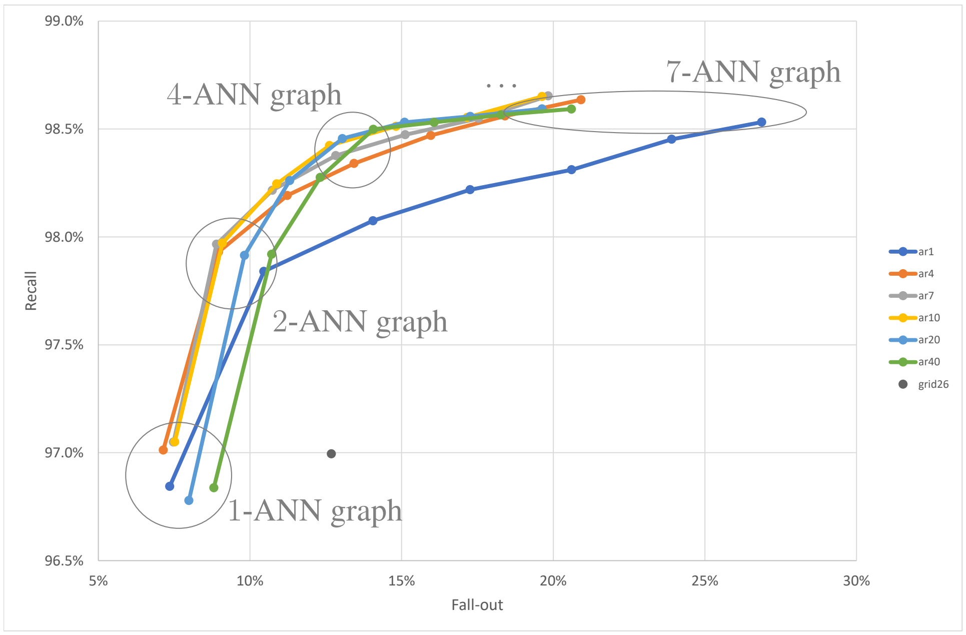
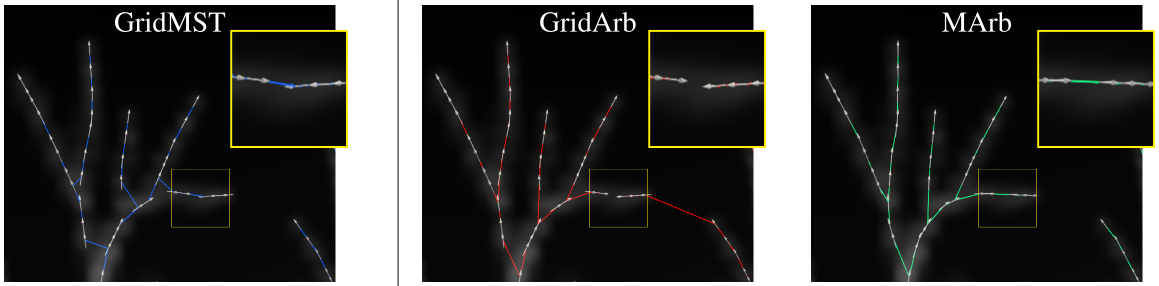


Figure 5.11: (Quasi) ROC curves evaluating accuracy of the neighborhoods N used for flow pattern estimation, as in [232]. We compare anisotropic KNNs and standard 26-grid connectivity (see gray dot). Evaluation is done based on synthetic data with ground truth where correct connectivity is available. “ar" stands for the aspect ratio and the number denotes the square of the aspect ratio. “grid26" represent the regular 26 neighbors on grid. We select “ar10” with 4 anisotropic nearest neighbourhood (ANN) connectivity system.

tree reconstruction (blue) on **undirected** Tubular Graph (standard)

tree reconstructions (red and green) on **directed** Confluent Tubular Graph (our)



(a) Flow pattern estimate (white) [232] + MST on geodesic tubular graph

(b) Flow pattern estimate (white) [232] + min. arb. on confluent tubular graph

(c) *Improved* flow estimate (white) + min. arb. on confluent tubular graph

Figure 5.12: Typical tree reconstruction examples for standard (a) and our *confluent* (b,c) tubular graphs. (a) White vectors represent CRF-based flow pattern estimates [232] using 26-grid regularization neighborhood $\mathcal{N}$. In case of thin sub-voxel vessels, such $\mathcal{N}$ has gaps creating inconsistent flow pattern for isolated small branches (yellow box) lacking bifurcations used by divergence prior to disambiguate orientations. (a) Undirected geodesic tubular graph with large KNN easily bridges such gaps ignoring (inconsistent) flow directions and produces topologically valid vessel MST (blue), even though bifurcations are not accurate. (b) Directed confluent tubular graph is sensitive to flow pattern errors. Minimum arborescence on this graph produces accurate bifurcations, but flow orientation errors (yellow box) lead to wrong topology. (c) CRF-based flow pattern estimator [232] with modified anisotropic KNN system $\mathcal{N}$ addresses the gaps at thin vessels. This improves flow orientations (white vectors in yellow box) and resolves confluent tubular graph artifacts producing trees with accurate both topology and bifurcations.

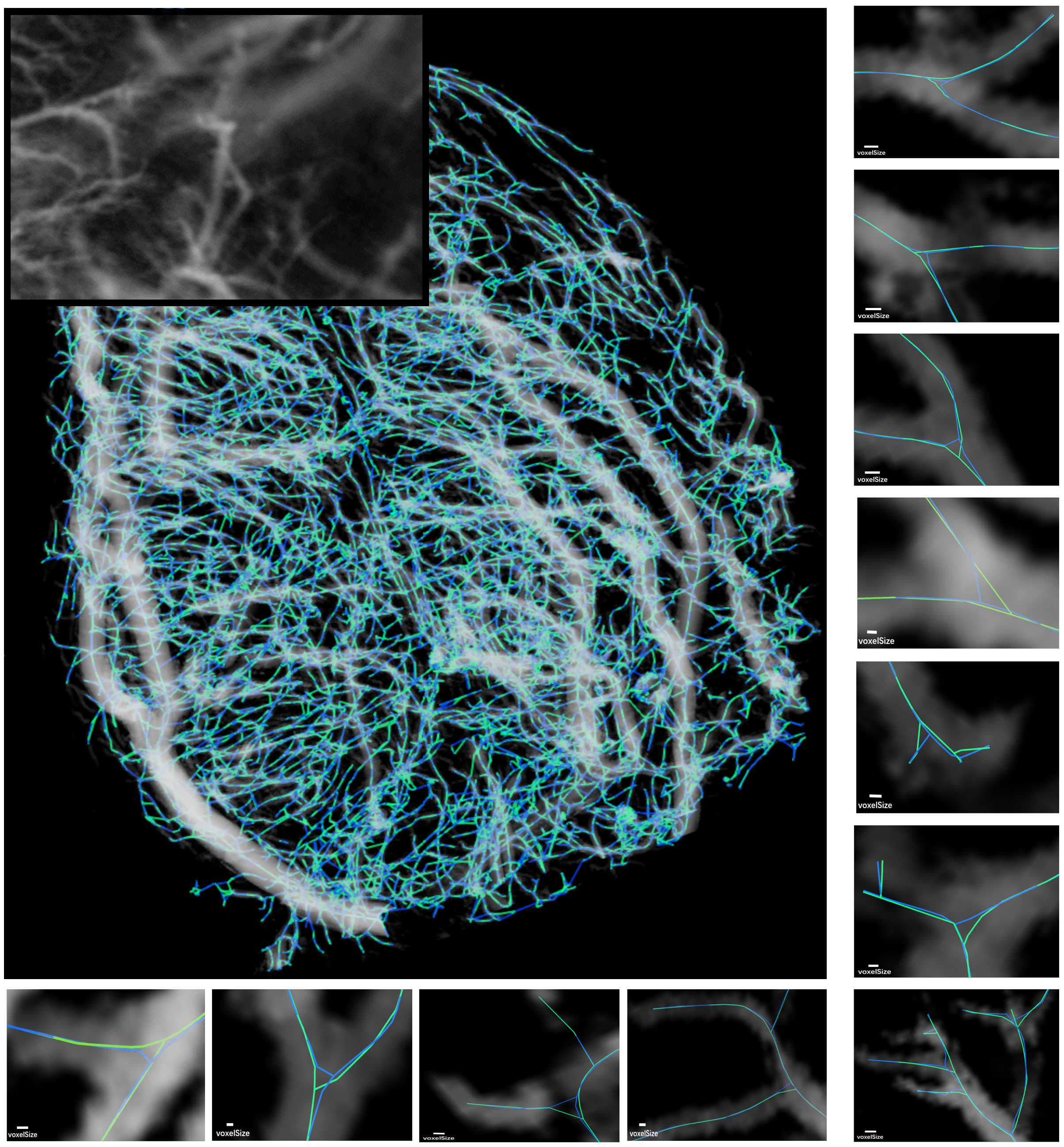


Figure 5.13: Real data micro-CT volume and vessel centerline reconstruction (green) obtained by our method and (blue) by [232]. To reduce clutter we show the result of Frangi *et al*. filter [70] instead of raw input data. The real data zoom-in is at the top left.

Chapter 6

Conclusions and future work

In this chapter, we summarize our contributions and also give the direction of future work.

Chapter 2 introduced novel reverse cross-entropy and collision cross-entropy in the context of self-labeling losses for clustering and weakly-supervised semantic segmentation tasks, which are later discussed in Chapter 3 and 4 respectively. In particular, the collision cross-entropy is naturally interpreted as measuring the probability of equality between two random variables, which perfectly fits the self-labeling framework where both the network predictions and pseudo-labels are estimated. It is symmetric instead of treating one as the target, like the standard cross-entropy. While the latter makes the network copy the uncertainty in estimated pseudo-labels, our collision cross-entropy and reverse cross-entropy naturally weaken the training on data points where pseudo labels are more uncertain, making the losses more robust to labeling errors.

Chapter 3 provides more insights into the properties of entropy-based clustering losses, consisting of decisiveness and fairness terms, and optimization algorithms for the self-labeled version of the losses introduced in Chapter 2. Specifically, we formally proved the maximum margin property of the decisiveness term, represented by Renyi entropy, with norm regularization, distinguishing itself from the variance-based clustering (K-means). For practicability, we derived efficient EM algorithms for optimizing the pseudo-labels in the self-labeling losses with reverse cross-entropy and collision cross-entropy respectively. The discriminative entropy clustering losses we studied are flexible, but there are still limitations that can be taken for the future research. For example, the current assumption for entropy clustering is that the number of clusters is fixed. This assumption is very similar to the fundamental assumption in the basic K-means methodology, which is even reflected in its name. This is not to say that K-means cannot be generalized in a way where K becomes an estimated variable, but this requires additional terms added to the loss (e.g. AIC or BIC information theoretic criteria). In fact, similar ideas can be explored in the context of entropy clustering, but, this is a substantial topic on its own. This direction can be continued in the future work. Also, the fairness term in the loss implies that the clusters should be balanced while in the real world the data imbalance is common. How to adjust/design the loss function to be able to deal with unbalanced datasets can be another future direction.

In Chapter 4 we proposed a convergent soft self-labeling framework based on a simple well-motivated loss based on decisiveness term, discussed in Chapter 3, and Potts model. The soft pseudo-labels were motivated as auxiliary optimization variables, as introduced in Chapter 2, which simplify optimization of scribble-supervised loss. We introduced new relaxations of the Potts model and systematically evaluate them. We also provided an efficient and general optimization algorithm for optimizing the soft pseudo-labels with different Potts relaxations and cross-entropy unary terms. Specifically, we recommend the combination of log-quadratic Potts relaxations, collision cross-entropy and the nearest-neighbor neighborhood, which achieves the best result that may even outperform the fully-supervised method with full pixel-precise masks. Our method does not require any modifications of the semantic segmentation models and it is easy to reproduce. Our general framework can be used for other weakly-supervised segmentation problems based on boxes, class tags, etc. in the future work.

Chapter 5 provides a new geometric prior, employing the estimated flow direction, to compute the tree reconstruction loss followed by the standard minimum arboresence optimization algorithm. Our extensive experiments convincingly demonstrate the advantages of directed *tubular flow graph* for accuracy of unsupervised vessel tree reconstruction, particularly at bifurcations. They validate the introduced anisotropic vessel geometry prior corresponding to the directedness of the flow carried by the vessels on both synthetic and real 3D vasculature data. As for the future work, we can integrate the deep networks and train the network using the pseudo-labels provided by our algorithm. Thus, we can have faster inference. This can be further improved if the problem can be formulated into a self-labeling loss alternating the optimization of pseudo-labels and network parameters.

References

[1] Multilabel random walker image segmentation using prior models. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 763–770. IEEE, 2005.

[2] Jiwoon Ahn, Sunghyun Cho, and Suha Kwak. Weakly supervised learning of instance segmentation with inter-pixel relations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2209–2218, 2019.

[3] Jiwoon Ahn and Suha Kwak. Learning pixel-level semantic affinity with image-level supervision for weakly supervised semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4981–4990, 2018.

[4] Tao D Alter and Ronen Basri. Extracting salient curves from images: An analysis of the saliency network. *IJCV*, 27(1):51–69, 1998.

[5] Luis Alvarez, Pierre-Louis Lions, and Jean-Michel Morel. Image selective smoothing and edge detection by nonlinear diffusion. ii. *SIAM Journal on numerical analysis*, 29(3):845–866, 1992.

[6] Mykhaylo Andriluka, Jasper RR Uijlings, and Vittorio Ferrari. Fluid annotation: a human-machine collaboration interface for full image annotation. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 1957–1966, 2018.

[7] Nikita Araslanov and Stefan Roth. Single-stage semantic segmentation from image labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4253–4262, 2020.

[8] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *International Conference on Learning Representations*, 2020.

[9] Stephen R Aylward and Elizabeth Bullitt. Initialization, noise, singularities, and scale in height ridge traversal for tubular object centerline extraction. *IEEE transactions on medical imaging*, 21(2):61–75, 2002.

[10] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[11] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*, pages 404–417. Springer, 2006.

[12] Ismail Ben Ayed, Lena Gorelick, and Yuri Boykov. Auxiliary cuts for general classes of higher order functionals. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1304–1311, 2013.

[13] A. Ben-Hur, D. Horn, H. Siegelman, and V. Vapnik. Support vector clustering. *Journal of Machine Learning Research*, 2:125 – 137, 2001.

[14] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. Label propagation and quadratic criterion. 2006.

[15] Fethallah Benmansour and Laurent D Cohen. Tubular structure segmentation based on minimal path method and anisotropic enhancement. *IJCV*, 92(2):192–210, 2011.

[16] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[17] Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 4. springer New York, 2006.

[18] Endre Boros and Peter L Hammer. Pseudo-boolean optimization. *Discrete applied mathematics*, 123(1-3):155–225, 2002.

[19] Endre Boros and Peter L Hammer. Discrete optimization: the state of the art. 2003.

[20] Malik Boudiaf, Jérôme Rony, Imtiaz Masud Ziko, Eric Granger, Marco Pedersoli, Pablo Piantanida, and Ismail Ben Ayed. A unifying mutual information view of metric learning: cross-entropy vs. pairwise losses. In *European conference on computer vision*, pages 548–564. Springer, 2020.

[21] Sylvain Bouix, Kaleem Siddiqi, and Allen Tannenbaum. Flux driven automatic centerline extraction. *Medical image analysis*, 9(3):209–221, 2005.

[22] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[23] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[24] Boykov and Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. In *Proceedings Ninth IEEE international conference on computer vision*, pages 26–33. IEEE, 2003.

[25] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11):1222–1239, 2001.

[26] Yuri Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*, volume 1, pages 105–112. IEEE, 2001.

[27] Kristian Bredies, Thomas Pock, and Benedikt Wirth. Convex relaxation of a class of vertex penalizing functionals. *Journal of mathematical imaging and vision*, 47:278–302, 2013.

[28] Kristian Bredies, Thomas Pock, and Benedikt Wirth. Convex relaxation of a class of vertex penalizing functionals. *Journal of Mathematical Imaging and Vision*, 47(3):278–302, 2013.

[29] John Bridle, Anthony Heading, and David MacKay. Unsupervised classifiers, mutual information and'phantom targets. *Advances in neural information processing systems*, 4, 1991.

[30] John S. Bridle, Anthony J. R. Heading, and David J. C. MacKay. Unsupervised classifiers, mutual information and 'phantom targets'. In *NIPS*, pages 1096–1101, 1991.

[31] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 132–149, 2018.

[32] Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of mathematical imaging and vision*, 40:120–145, 2011.

[33] Tony F Chan and Jianhong Shen. Nontexture inpainting by curvature-driven diffusions. *Journal of Visual Communication and Image Representation*, 12(4):436–449, 2001.

[34] Tony F Chan and Luminita A Vese. Active contours without edges. *IEEE Transactions on image processing*, 10(2):266–277, 2001.

[35] Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep adaptive image clustering. In *International Conference on Computer Vision (ICCV)*, pages 5879–5887, 2017.

[36] Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep adaptive image clustering. In *Proceedings of the IEEE international conference on computer vision*, pages 5879–5887, 2017.

[37] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[38] Da Chen, Jean-Marie Mirebeau, and Laurent D Cohen. Global minimum for a finsler elastica minimal path approach. *International Journal of Computer Vision*, 122(3):458–483, 2017.

[39] Hongjun Chen, Jinbao Wang, Hong Cai Chen, Xiantong Zhen, Feng Zheng, Rongrong Ji, and Ling Shao. Seminar learning for click-level weakly supervised semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6920–6929, 2021.

[40] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

[41] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[42] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[43] Qi Chen, Lingxiao Yang, Jian-Huang Lai, and Xiaohua Xie. Self-supervised image-specific prototype exploration for weakly supervised semantic segmentation. In *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4288–4298, 2022.

[44] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[45] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15750–15758, 2021.

[46] Egor Chesakov. Vascular tree structure: Fast curvature regularization and validation. *Electronic Thesis and Dissertation Repository. The University of Western Ontario*, (3396), 2015. Master of Science thesis.

[47] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 539–546. IEEE, 2005.

[48] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

[49] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[50] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20:273–297, 1995.

[51] Camille Couprie, Leo Grady, Laurent Najman, and Hugues Talbot. Power watershed: A unifying graph-based optimization framework. *IEEE transactions on pattern analysis and machine intelligence*, 33(7):1384–1399, 2010.

[52] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[53] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

[54] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[55] Hervé Delingette. On smoothness measures of active contours and surfaces. In *Proceedings IEEE Workshop on Variational and Level Set Methods in Computer Vision*, pages 43–50. IEEE, 2001.

[56] Andrew Delong, Anton Osokin, Hossam N Isack, and Yuri Boykov. Fast approximate energy minimization with label costs. *International journal of computer vision*, 96:1–27, 2012.

[57] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society*, pages 1–38, 1977.

[58] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[59] Anthony D DeRose. Geometric continuity: a parametrization independent measure of continuity for computer aided geometric design. Technical report, CA Univ Berkeley Dept of Electrical Engineering and Computer Sciences, 1985.

[60] Thomas Deschamps and Laurent D. Cohen. Fast extraction of minimal paths in 3d images and applications to virtual endoscopy. *Medical Image Analysis*, 5(4):281 – 299, 2001.

[61] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1(1):269–271, December 1959.

[62] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[63] Jack Edmonds. Optimum branchings. *J. Res. Nat. Bur. Standards*, 71B(4), October-December 1967.

[64] Andinet Enquobahrie, Luis Ibanez, Elizabeth Bullitt, and Stephen Aylward. Vessel enhancing diffusion filter. *The Insight Journal*, 1:1–14, 2007.

[65] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–308, 2009.

[66] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–338, 2010.

[67] M. A. T. Figueiredo and J. M. N. Leitao. A nonsmoothing approach to the estimation of vessel contours in angiograms. *IEEE Transactions on Medical Imaging*, 14(1):162–172, March 1995.

[68] Evelyn Fix. *Discriminatory analysis: nonparametric discrimination, consistency properties*, volume 1. USAF school of Aviation Medicine, 1985.

[69] AH Fowler and CW Wilson. Cubic spline: A curve fitting routine. Technical report, Union Carbide Corp., Oak Ridge, Tenn. Y-12 Plant, 1966.

[70] Alejandro F Frangi, Wiro J Niessen, Koen L Vincken, and Max A Viergever. Multiscale vessel enhancement filtering. In *MICCAI'98*, pages 130–137. Springer, 1998.

[71] William T. Freeman and Edward H Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (9):891–906, 1991.

[72] Harold N Gabow, Zvi Galil, Thomas Spencer, and Robert E Tarjan. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica*, 6(2):109–122, 1986.

[73] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741, 1984.

[74] Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proceedings of the IEEE international conference on computer vision*, pages 5736–5745, 2017.

[75] Todd A Gillette, Kerry M Brown, and Giorgio A Ascoli. The diadem metric: comparing multiple reconstructions of the same neuron. *Neuroinformatics*, 9(2-3):233, 2011.

[76] Germán González, François Fleuret, and Pascal Fua. Automated delineation of dendritic networks in noisy image stacks. In *European Conference on Computer Vision*, pages 214–227. Springer, 2008.

[77] Lena Gorelick, Yuri Boykov, Olga Veksler, Ismail Ben Ayed, and Andrew Delong. Submodularization for binary pairwise energies. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1154–1161, 2014.

[78] Leo Grady. Random walks for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 28(11):1768–1783, 2006.

[79] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. *Advances in neural information processing systems*, 17, 2004.

[80] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. *Advances in neural information processing systems*, 17, 2004.

[81] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

[82] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

[83] Gideon Guy and Gérard Medioni. Inferring global perceptual contours from local features. In *CVPR*, 1993.

[84] Ghassan Hamarneh and Preet Jassi. Vascusynth: simulating vascular trees for generating volumetric image data with ground-truth segmentation and tree analysis. *Computerized medical imaging and graphics*, 34(8):605–616, 2010.

[85] Peter L Hammer and Sergiu Rudeanu. *Boolean methods in operations research and related areas*, volume 7. Springer Science & Business Media, 2012.

[86] Peter L Hammer, Sergiu Rudeanu, Peter L Hammer, and Sergiu Rudeanu. Flows in networks and chains in partially ordered sets. *Boolean Methods in Operations Research and Related Areas*, pages 258–267, 1968.

[87] Chris Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[88] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[89] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[90] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[91] Stefan Heber, Rene Ranftl, and Thomas Pock. Approximate envelope minimization for curvature regularity. In *ECCV*, 2012.

[92] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2018.

[93] Berthold KP Horn. The curve of least energy. *ACM Transactions on Mathematical Software (TOMS)*, 9(4):441–460, 1983.

[94] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[95] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*. John Wiley & Sons, 2013.

[96] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[97] J. Hsieh. Computed tomography: principles, design, artifacts, and recent advances. *SPIE*, 2009.

[98] Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. Learning discrete representations via information maximizing self-augmented training. In *International conference on machine learning*, pages 1558–1567. PMLR, 2017.

[99] Peter Hui, Michael J Pelsmajer, Marcus Schaefer, and Daniel Stefankovic. Train tracks and confluent drawings. *Algorithmica*, 47(4):465–479, 2007.

[100] Alan Julian Izenman. Review papers: Recent developments in nonparametric density estimation. *Journal of the american statistical association*, 86(413):205–224, 1991.

[101] Mohammed Jabi, Marco Pedersoli, Amar Mitiche, and Ismail Ben Ayed. Deep clustering: On the link between discriminative models and k-means. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(6):1887–1896, 2021.

[102] Xu Ji, Joao F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9865–9874, 2019.

[103] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: an unsupervised and generative approach to clustering. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 1965–1972, 2017.

[104] Julien Jomier, Vincent LeDigarcher, and Stephen R Aylward. Automatic vascular tree formation using the mahalanobis distance. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 806–812. Springer, 2005.

[105] Jörg H Kappes, Bjoern Andres, Fred A Hamprecht, Christoph Schnörr, Sebastian Nowozin, Dhruv Batra, Sungwoong Kim, Bernhard X Kausler, Thorben Kröger, Jan Lellmann, et al. A comparative study of modern inference techniques for structured discrete energy minimization problems. *International Journal of Computer Vision*, 115:155–184, 2015.

[106] Tsung-Wei Ke, Jyh-Jing Hwang, and Stella X Yu. Universal weakly supervised segmentation by pixel-to-segment contrastive learning. *arXiv preprint arXiv:2105.00957*, 2021.

[107] Carl T Kelley. *Iterative methods for linear and nonlinear equations*. SIAM, 1995.

[108] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

[109] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[110] Philip N. Klein. Computing the edit-distance between unrooted ordered trees. pages 91–102, 1998.

[111] Timothy L Kline, Mair Zamir, and Erik L Ritman. Accuracy of microvascular measurements obtained from micro-ct images. *Annals of biomedical engineering*, 38(9):2851–2864, 2010.

[112] Alexander Kolesnikov and Christoph H Lampert. Seed, expand and constrain: Three principles for weakly-supervised image segmentation. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 695–711. Springer, 2016.

[113] Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization. In *International Workshop on Artificial Intelligence and Statistics*, pages 182–189. PMLR, 2005.

[114] Vladimir Kolmogorov and Ramin Zabin. What energy functions can be minimized via graph cuts? *IEEE transactions on pattern analysis and machine intelligence*, 26(2):147–159, 2004.

[115] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected CRFs with Gaussian edge potentials. *Advances in neural information processing systems*, 24, 2011.

[116] Andreas Krause, Pietro Perona, and Ryan Gomes. Discriminative clustering by regularized information maximization. *Advances in neural information processing systems*, 23, 2010.

[117] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[118] Joseph B Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

[119] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[120] V. Kulharia, S. Chandra, A. Agrawal, P. Torr, and A. Tyagi. Box2seg: Attention weighted loss and discriminative feature learning for weakly supervised segmentation. In *ECCV'20*.

[121] John Lafferty, Andrew McCallum, Fernando Pereira, et al. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Icml*, volume 1, page 3. Williamstown, MA, 2001.

[122] Max W K Law and Albert C S Chung. Three dimensional curvilinear structure detection using optimally oriented flux. In *European conference on computer vision*, pages 368–382. Springer, 2008.

[123] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[124] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[125] Jungbeom Lee, Eunji Kim, Sungmin Lee, Jangho Lee, and Sungroh Yoon. Ficklenet: Weakly and semi-supervised semantic image segmentation using stochastic inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5267–5276, 2019.

[126] Jesse Levinson, Jake Askeland, Jan Becker, Jennifer Dolson, David Held, Soeren Kammel, J Zico Kolter, Dirk Langer, Oliver Pink, Vaughan Pratt, et al. Towards fully autonomous driving: Systems and algorithms. In *2011 IEEE intelligent vehicles symposium (IV)*, pages 163–168. IEEE, 2011.

[127] Gang Li and Steven W Zucker. Differential geometric inference in surface stereo. *PAMI*, 32(1):72–86, 2010.

[128] Hua Li and Anthony Yezzi. Vessels as 4-d curves: Global minimal 4-d paths to extract 3-d tubular surfaces and centerlines. *IEEE transactions on medical imaging*, 26(9):1213–1223, 2007.

[129] Zhiyuan Liang, Tiancai Wang, Xiangyu Zhang, Jian Sun, and Jianbing Shen. Tree energy loss: Towards sparsely annotated semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16907–16916, 2022.

[130] Di Lin, Jifeng Dai, Jiaya Jia, Kaiming He, and Jian Sun. Scribblesup: Scribble-supervised convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3159–3167, 2016.

[131] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[132] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

[133] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[134] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[135] László Lovász. Submodular functions and convexity. *Mathematical Programming The State of the Art: Bonn 1982*, pages 235–257, 1983.

[136] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004.

[137] David JC MacKay. Bayesian interpolation. *Neural computation*, 4(3):415–447, 1992.

[138] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

[139] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar K-means problem is NP-hard. *Theoretical Computer Science*, 442:13–21, 2012.

[140] Dmitrii Marin. Higher-order losses and optimization for low-level and deep segmentation. 2021.

[141] Dmitrii Marin and Yuri Boykov. Robust trust region for weakly supervised segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6608–6618, 2021.

[142] Dmitrii Marin, Meng Tang, Ismail Ben Ayed, and Yuri Boykov. Beyond gradient descent for regularized segmentation losses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10187–10196, 2019.

[143] Dmitrii Marin, Yuchen Zhong, Maria Drangova, and Yuri Boykov. Thin structure estimation with curvature regularization. In *International Conference on Computer Vision (ICCV)*, 2015.

[144] LW McKeehan. A contribution to the theory of ferromagnetism. *Physical Review*, 26(2):274, 1925.

[145] Odyssée Merveille, Benoît Naegel, Hugues Talbot, and Nicolas Passat. nD variational restoration of curvilinear structures with prior-based directional regularization. *IEEE Transactions on Image Processing*, 28(8):3848–3859, 2019.

[146] Odyssée Merveille, Hugues Talbot, Laurent Najman, and Nicolas Passat. Curvilinear structure analysis by ranking the orientation responses of path operators. *IEEE transactions on pattern analysis and machine intelligence*, 40(2):304–317, 2017.

[147] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3523–3542, 2021.

[148] Sara Moccia, Elena De Momi, Sara El Hadji, and Leonardo Mattos. Blood vessel segmentation algorithms — review of methods, datasets and evaluation metrics. *Computer Methods and Programs in Biomedicine*, 158:71–91, 2018.

[149] Parya MomayyezSiahkal and Kaleem Siddiqi. 3d stochastic completion fields for mapping connectivity in diffusion mri. *PAMI*, 35(4):983–995, 2013.

[150] Stefano Moriconi, Maria A Zuluaga, H Rolf Jäger, Parashkev Nachev, Sébastien Ourselin, and M Jorge Cardoso. Inference of cerebrovascular topology with geodesic minimum spanning trees. *IEEE transactions on medical imaging*, 38(1):225–239, 2018.

[151] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? *Advances in neural information processing systems*, 32, 2019.

[152] C. Nieuwenhuis, E. Toeppe, L. Gorelick, O. Veksler, and Y. Boykov. Efficient squared curvature. In *CVPR*, Columbus, Ohio, 2014.

[153] Claudia Nieuwenhuis, Eno Toeppe, Lena Gorelick, Olga Veksler, and Yuri Boykov. Efficient squared curvature. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4098–4105, 2014.

[154] Natural Scenes Dataset [NSD]. https://www.kaggle.com/datasets/nitishabharathi/scene-classification, 2020.

[155] Carl Olsson and Yuri Boykov. Curvature-based regularization for surface approximation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1576–1583. IEEE, 2012.

[156] Carl Olsson, Johannes Ulén, and Yuri Boykov. In defense of 3d-label stereo. In *CVPR*, pages 1730–1737. IEEE, 2013.

[157] Carl Olsson, Johannes Ulén, Yuri Boykov, and Vladimir Kolmogorov. Partial enumeration and curvature regularization. In *ICCV*, pages 2936–2943. IEEE, 2013.

[158] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[159] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[160] Zhiyi Pan, Peng Jiang, Yunhai Wang, Changhe Tu, and Anthony G Cohn. Scribble-supervised semantic segmentation by uncertainty reduction on neural representation and self-supervision on neural eigenspace. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7416–7425, 2021.

[161] Pierre Parent and Steven W Zucker. Trace inference, curvature consistency, and curve detection. *PAMI*, 11:823–839, 1989.

[162] Deepak Pathak, Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional multi-class multiple instance learning. *arXiv preprint arXiv:1412.7144*, 2014.

[163] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1944–1952, 2017.

[164] Judea Pearl. Reverend bayes on inference engines: A distributed hierarchical approach. In *Probabilistic and Causal Inference: The Works of Judea Pearl*, pages 129–138. 2022.

[165] Hanchuan Peng, Fuhui Long, and Gene Myers. Automatic 3d neuron tracing using all-path pruning. *Bioinformatics*, 27(13):i239–i247, 2011.

[166] Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. 2017.

[167] P Jonathon Phillips, Patrick J Flynn, Todd Scruggs, Kevin W Bowyer, Jin Chang, Kevin Hoffman, Joe Marques, Jaesik Min, and William Worek. Overview of the face recognition grand challenge. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 947–954. IEEE, 2005.

[168] Jean-Claude Picard and H Donald Ratliff. A cut approach to the rectilinear distance facility location problem. *Operations Research*, 26(3):422–433, 1978.

[169] Pedro O Pinheiro and Ronan Collobert. From image-level to pixel-level labeling with convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1713–1721, 2015.

[170] Thomas Pock, Antonin Chambolle, Daniel Cremers, and Horst Bischof. A convex relaxation approach for computing minimal partitions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 810–817, 2009.

[171] Jose C. Principe, Dongxin Xu, and John W. Fisher III. Information-theoretic learning. *Advances in unsupervised adaptive filtering*, 3 2000.

[172] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents.

[173] Sudhir Rao, Allan de Medeiros Martins, and José C. Príncipe. Mean shift: An information theoretic perspective. *Pattern Recognition Letters*, 30:222–230, February 2009.

[174] Pradeep Ravikumar and John Lafferty. Quadratic programming relaxations for metric labeling and markov random field map estimation. In *Proceedings of the 23rd international conference on Machine learning*, pages 737–744, 2006.

[175] Pradeep Ravikumar and John Lafferty. Quadratic programming relaxations for metric labeling and Markov Random Field MAP estimation. In *The 23rd International Conference on Machine Learning*, page 737–744, 2006.

[176] Alfréd Rényi. On measures of entropy and information. *Fourth Berkeley Symp. Math. Stat. Probab.*, 1:547–561, 1961.

[177] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[178] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[179] Saharon Rosset, Ji Zhu, and Trevor Hastie. Margin maximizing loss functions. *Advances in neural information processing systems*, 16, 2003.

[180] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. " grabcut" interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, 23(3):309–314, 2004.

[181] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[182] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[183] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[184] T. Schoenemann, F. Kahl, and D. Cremers. Curvature regularity for region-based image segmentation and inpainting: A linear programming relaxation. In *ICCV*, Kyoto, 2009.

[185] Thomas Schoenemann, Fredrik Kahl, and Daniel Cremers. Curvature regularity for region-based image segmentation and inpainting: A linear programming relaxation. In *2009 IEEE 12th International Conference on Computer Vision*, pages 17–23. IEEE, 2009.

[186] Thomas Schoenemann, Fredrik Kahl, Simon Masnou, and Daniel Cremers. A linear framework for region-based image segmentation and inpainting involving curvature penalization. *arXiv preprint arXiv:1102.3830*, 2011.

[187] Thomas Schoenemann, Fredrik Kahl, Simon Masnou, and Daniel Cremers. A linear framework for region-based image segmentation and inpainting involving curvature penalization. *IJCV*, 2012.

[188] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[189] Matthias Seeger. Learning with labeled and unlabeled data. 2000.

[190] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[191] Kaleem Siddiqi and Stephen Pizer. *Medial representations: mathematics, algorithms and applications*, volume 37. Springer Science & Business Media, 2008.

[192] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[193] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. *Advances in neural information processing systems*, 29, 2016.

[194] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[195] Yan-Yan Song and LU Ying. Decision tree methods: applications for classification and prediction. *Shanghai archives of psychiatry*, 27(2):130, 2015.

[196] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

[197] Jost Tobias Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. In *International Conference on Learning Representations*, 2015.

[198] Petter Strandmark and Fredrik Kahl. Curvature regularization for curves and surfaces in a global optimization framework. In *EMMCVPR*, pages 205–218. Springer, 2011.

[199] BU-Qing Su and Ding-zhe Liu. *Computational geometry: curve and surface modeling*. Academic Press Professional, Inc., 1989.

[200] Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training convolutional networks with noisy labels. *ICLR workshop*, 2015.

[201] W. Sun, Z. Liu, Y. Zhang, Y. Zhong, and N. Barnes. An alternative to wsss? an empirical study of the segment anything model on weakly-supervised semantic segmentation problems, 2023.

[202] Richard Szeliski, Ramin Zabih, Daniel Scharstein, Olga Veksler, Vladimir Kolmogorov, Aseem Agarwala, Marshall Tappen, and Carsten Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE transactions on pattern analysis and machine intelligence*, 30(6):1068–1080, 2008.

[203] Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5552–5560, 2018.

[204] Meng Tang, Ismail Ben Ayed, and Yuri Boykov. Pseudo-bound optimization for binary energies. In *European Conference on Computer Vision*, pages 691–707. Springer, 2014.

[205] Meng Tang, Abdelaziz Djelouah, Federico Perazzi, Yuri Boykov, and Christopher Schroers. Normalized cut loss for weakly-supervised cnn segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1818–1827, 2018.

[206] Meng Tang, Federico Perazzi, Abdelaziz Djelouah, Ismail Ben Ayed, Christopher Schroers, and Yuri Boykov. On regularized losses for weakly-supervised cnn segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 507–522, 2018.

[207] R. E. Tarjan. Finding optimum branchings. *Networks*, 7(1):25–35, 1977.

[208] Ferenc C. Thierrin, Fady Alajaji, and Tamás Linder. Rényi cross-entropy measures for common distributions and processes with memory. *Entropy*, 24(10), October 2022.

[209] Antonio Torralba, Rob Fergus, and William T Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 30(11):1958–1970, 2008.

[210] Michael Tschannen, Josip Djolonga, Paul K Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning. In *International Conference on Learning Representations*, 2019.

[211] Engin Turetken, Carlos Becker, Przemyslaw Glowacki, Fethallah Benmansour, and Pascal Fua. Detecting irregular curvilinear structures in gray scale and color imagery using multi-directional oriented flux. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1553–1560, 2013.

[212] Engin Turetken, Fethallah Benmansour, Bjoern Andres, Przemyslaw Glowacki, Hanspeter Pfister, and Pascal Fua. Reconstructing curvilinear networks using path classifiers and integer programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 38(12):2515–2530, December 2016.

[213] Engin Turetken, German Gonzalez, Christian Blum, and Pascal Fua. Automated reconstruction of dendritic and axonal trees by global optimization with geometric priors. *Neuroinformatics*, 9(2-3):279–302, 2011.

[214] J. A. Tyrrell, E. di Tomaso, D. Fuja, R. Tong, K. Kozak, R. K. Jain, and B. Roysam. Robust 3-d modeling of vasculature imagery using superellipsoids. *IEEE Transactions on Medical Imaging*, 26(2):223–237, Feb 2007.

[215] Francisco J. Valverde-Albacete and Carmen Peláez-Moreno. The case for shifting the Rényi entropy. *Entropy*, 21(1), January 2019.

[216] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Scan: Learning to classify images without labels. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X*, pages 268–285. Springer, 2020.

[217] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[218] Olga Veksler. Efficient graph cut optimization for full crfs with quantized edges. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):1005–1012, 2019.

[219] Olga Veksler and Yuri Boykov. Sparse non-local crf. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4493–4503, 2022.

[220] Bin Wang, Guojun Qi, Sheng Tang, Tianzhu Zhang, Yunchao Wei, Linghui Li, and Yongdong Zhang. Boundary perception guidance: A scribble-supervised semantic segmentation approach. In *IJCAI International joint conference on artificial intelligence*, 2019.

[221] Changwei Wang, Rongtao Xu, Shibiao Xu, Weiliang Meng, and Xiaopeng Zhang. Treating pseudo-labels generation as image matting for weakly supervised semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 755–765, 2023.

[222] Lance R Williams and David W Jacobs. Stochastic completion fields: A neural model of illusory contour shape and salience. *Neural Computation*, 9(4):837–858, 1997.

[223] Lior Wolf, Tal Hassner, and Itay Maoz. Face recognition in unconstrained videos with matched background similarity. In *CVPR 2011*, pages 529–534. IEEE, 2011.

[224] Oliver Woodford, Philip Torr, Ian Reid, and Andrew Fitzgibbon. Global stereo reconstruction under second-order smoothness priors. *PAMI*, 31(12):2115–2128, 2009.

[225] Jun Xie, Ting Zhao, Tzumin Lee, Eugene Myers, and Hanchuan Peng. Automatic neuron tracing in volumetric microscopy images with anisotropic path searching. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 472–479. Springer, 2010.

[226] Jingshan Xu, Chuanwei Zhou, Zhen Cui, Chunyan Xu, Yuge Huang, Pengcheng Shen, Shaoxin Li, and Jian Yang. Scribble-supervised semantic segmentation inference. In

Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 15354–15363, 2021.

[227] Linli Xu, James Neufeld, Bryce Larson, and Dale Schuurmans. Maximum margin clustering. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2004.

[228] Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *international conference on machine learning*, pages 3861–3870. PMLR, 2017.

[229] Xiao-Tong Yuan and Bao-Gang Hu. Robust feature extraction via information theoretic learning. In *International Conference on Machine Learning, (ICML)*, page 1193–1200, June 2009.

[230] Christopher Zach, Christian Häne, and Marc Pollefeys. What is optimized in tight convex relaxations for multi-label problems? In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1664–1671, 2012.

[231] Kaizhong Zhang and Dennis Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*, 18(6):1245–1262, 1989.

[232] Zhongwen Zhang, Dmitrii Marin, Egor Chesakov, Marc Moreno Maza, Maria Drangova, and Yuri Boykov. Divergence prior and vessel-tree reconstruction. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, California, June 2019.

[233] Zhongwen Zhang, Dmitrii Marin, Maria Drangova, and Yuri Boykov. Confluent vessel trees with accurate bifurcations. In *arXiv:2103.14268*, 2021.

[234] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

[235] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017.

[236] Dengyong Zhou, Olivier Bousquet, Thomas Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. *Advances in neural information processing systems*, 16, 2003.

[237] Tianfei Zhou, Meijie Zhang, Fang Zhao, and Jianwu Li. Regional semantic contrast and aggregation for weakly supervised semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4299–4309, 2022.

[238] Song Chun Zhu and Alan Yuille. Region competition: Unifying snakes, region growing, and bayes/mdl for multiband image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 18(9):884–900, 1996.

[239] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. *ProQuest Number: INFORMATION TO ALL USERS*, 2002.

[240] Xiaojin Zhu and Andrew B Goldberg. *Introduction to semi-supervised learning*. Springer Nature, 2022.