M.S. THESIS

Anaylsis and Improvement of Softmax-Loss in Deep Metric Learning

딥 메트릭 러닝의 소프트맥스 손실 함수의 대한 분석과 개선 방안

BY

Junghyun Lee

February 2021

DEPARTMENT OF MATHEMATICAL SCIENCES
COLLEGE OF NATURAL SCIENCES
SEOUL NATIONAL UNIVERSITY

M.S. THESIS

Anaylsis and Improvement of Softmax-Loss in Deep Metric Learning

딥 메트릭 러닝의 소프트맥스 손실 함수의 대한 분석과 개선 방안

BY

Junghyun Lee

February 2021

DEPARTMENT OF MATHEMATICAL SCIENCES
COLLEGE OF NATURAL SCIENCES
SEOUL NATIONAL UNIVERSITY

Anaylsis and Improvement of Softmax-Loss in Deep Metric Learning

딥 메트릭 러닝의 소프트맥스 손실 함수의 대한 분석과 개선 방안

지도교수 강 명 주
이 논문을 이학석사 학위논문으로 제출함

2021년 2월

서울대학교 대학원

수리 과학부

이 정 현

이정현의 이학석사 학위 논문을 인준함

2021년 2월

위 원 장: 국 웅
부위원장: 강 명 주
위 원: 곽 지 훈

Abstract

Deep metric learning is a widely used learning method in the field of image search and face verification. This learning method learns the semantic distance between features while mapping each data to a specific embedding space. For learning metrics in the embedding space, various loss functions and data sampling methods have been developed recently. In particular, the proxy-based loss function is widely used because of the excellent computation speed. Previous researches on the proxy-based loss have assumed the feature vectors and the corresponding proxies coming close after enough training. Although this assumption highly related to the performance of classification, the actual behavior of vectors in embedding space has not been analyzed explicitly. This study analyzes the softmax loss function used as the basic frame in proxy-based loss to determine when the assumption will match the real training result. The analyzed results indicate the new loss will improve performance. The experimental result confirms that this new loss function can effectively improve the performance of the ResNet-50 model.

keywords: Deep Metric Learning, Image Retrieval, Fine-tuning, Feature Vector, Intra-Class Distance, Inter-Class Distance

student number: 2018-29521

Contents

Chapter 1

Introduction

Deep learning has played a major role in discovering the semantic meanings hiding in data that machines cannot easily discover. In particular, many developments have been made in image analysis. It was not an easy task to discover the meaning inherent in the image and classify it accordingly, but as the deep neural network developed, it became possible to effectively analyze large image sets such as ImageNet [1].

There are several types of image classification tasks: identification and verification. The identification task is a problem of checking whether a given image and testing gallery are entered, and the verification task is a problem of checking whether two given images belong to the same class or different classes. Learning of deep neural networks enables these tasks to distinguish not only the known classes, but also the features of images corresponding to classes not used for learning. Various studies on face-verification have been developed [2, 3].

Among the various methods proposed to solve image tasks, there is *deep metric learning* that learns the semantic metric. In deep metric learning, images are converted into vectors of the same length through a neural network, and images are identified through distances in the space where this vector is embedding. The main purpose here is to embedding the vector of the image corresponding to the same class closer together and the vector of the image corresponding to different class farther away.

The loss of deep metric learning used to learn the metric is largely divided into pair-based loss and proxy-based loss. Pair-based loss is a loss that contains the purpose of moving images of the same class closer and farther away. Triplet loss [3], Contrastive loss [4] and Multi-Similarity loss [5] are such examples of pair-based loss.

On the other hand, proxy-based loss is a method in which the motion of image vectors is determined based on the proxy vector. The concept of a proxy vector corresponding to each class has been introduced [6]. Among them, cross-entropy loss, which is widely used in classification tasks, is commonly applied [2, 7, 8]. The weight of the fully connected layer used for cross-entropy loss corresponds to the concept of proxy. This operation method is called softmax loss [7]. In fact, lots of proxy-based loss is a variation of softmax loss.

Paired-based loss has a relatively long learning time, such as $\mathcal{O}(N^2)$ or $\mathcal{O}(N^3)$ [9], due to the feature of considering various pairs of data. Here, N is the total number of data. On the other hand, since proxy-based loss only needs to consider the relationship between data and proxy, the learning time is $\mathcal{O}(NC)$ or $\mathcal{O}(NC^2)$ [10], which requires relatively little learning time. Here, C is the number of classes of training data, and in most tasks, $C \ll N$. Therefore, there are cases in which proxy-based loss is preferred due to the advantage of learning time [10].

Paired-based loss requires a lot of learning time, but because the loss function is intuitive and simple. It is possible to accurately analyze the gradient of the loss, and there are many studies that improve performance by using the analysis result [11]. However, since proxy-based loss is based on the method of using the classification, it is not clear what behaviors the vectors specifically show in the embedding space in the learning process. Various assumptions are used, such as that proxy and image vectors become closer [8, 10], but there are not many cases where detailed analysis has been discussed. The purpose of this study is to clearly analyze proxy-based loss, especially softmax loss, to check whether there is any posibilities for improvement.

Chapter 2

Related Works in Deep Metric Learning

2.1 Basic Concepts of Proxy-based Loss and Notations

Before describing the contents, we begin by introducing a basic concepts of proxy-based loss defining the symbols to be used in this paper.

Suppose that we have a training data with images of C classes. The model embeds each images in D-dimensional space. For example, ResNet-50 [12] model extracts each image to 2048-dimensional vector so we can consider ResNet-50 embeds each image in 2048-dimensional space. This space is called as *embedding space*.

Definition 1. *The feature vector is an embedded vector of training image in embedding space. We denote* $\mathbf{f}_{c,i} \in \mathbb{R}^D$ *as an feature vector of* i*th image in* c*th class.*

In deep metric learning, the commonly used metric of the embedding space is cosine-similarity [13]. If two feature vectors $\mathbf{x}_1,\ \mathbf{x}_2 \in \mathbb{R}^n$ are given, the cosine-similarity of two vectors, $\left(\frac{\mathbf{x}_1}{\|\mathbf{x}_1\|_2}\right)^\top \frac{\mathbf{x}_2}{\|\mathbf{x}_2\|_2}$, is a measurement of distance between two vectors where $\|\cdot\|_2$ is a Euclidean distance in embedding space $\mathbb{R}^D$. (We do not embed the images to the zero vector.) If the cosine-similarity is small, two vectors are considered close. Therefore the norm of each vectors has no effect on training. From now on, we assume

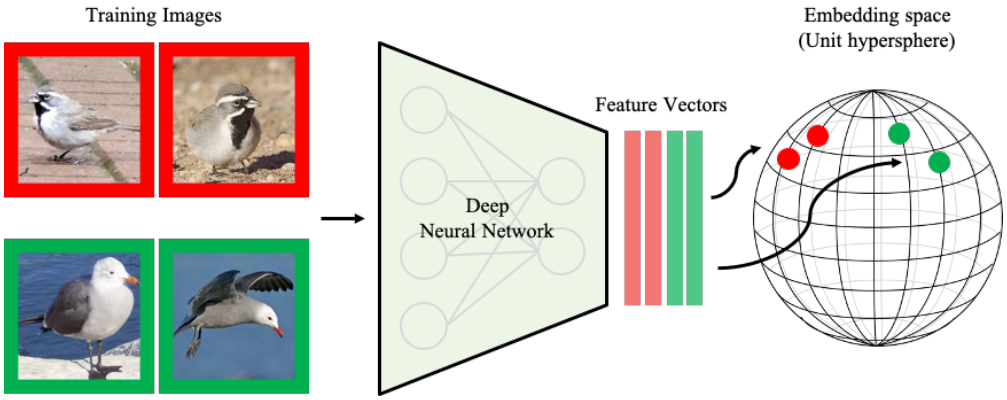

Figure 2.1: Diagram of image training in deep metric learning.

that all feature vectors $\mathbf{x}$ has norm 1, i.e. $\|\mathbf{x}\|_2 = 1$. Then, every extracted feature vector will be contained in the unit-hypersphere $\mathbb{S}_D = \{\mathbf{v} \in \mathbb{R}^D \mid \|\mathbf{x}\|_2 = 1\}$. Also, we denote cosine-similarity as

$$s(\mathbf{x}_1, \mathbf{x}_2) := \mathbf{x}_1^\top \mathbf{x}_2. \tag{2.1}$$

The main object of the metric learning is to embed train feature vector of the same class gather together. Moreover, it can be said well trained when feature vectors of each class is better distinguished [14]. The feature vectors from same class should become more closer, which can be called *intra-class variation* is small. Also the feature vectors from different class should become more further, which can be called *inter-class variation* is large. The behavior of is determined by training loss. For example, consider the standard *triplet loss* [3]

$$\mathcal{L}_t(\mathbf{x}, \mathbf{x}', \mathbf{y}) = g(\|\mathbf{x} - \mathbf{x}'\|_2) - g(\|\mathbf{x} - \mathbf{y}\|_2), \tag{2.2}$$

where $\mathbf{x}$ and $\mathbf{x}'$ are feature vectors from the same class, $\mathbf{y}$ is another feature vector from another class, and g is a non-decreasing differentiable function. If the model is trained by triplet loss, $\mathbf{x}'$ will be close to $\mathbf{x}$, and $\mathbf{y}$ will move away from $\mathbf{x}$ to reduce the triplet

loss. Note that if $g(x) = x^2$, from $\|\mathbf{x}\|_2 = \|\mathbf{x}'\|_2 = \|\mathbf{y}\|_2 = 1$, we can rewrite (2.2) as

$$\mathcal{L}_t(\mathbf{x}, \mathbf{x}', \mathbf{y}) = -\mathbf{x}^\top \mathbf{x}' + \mathbf{x}^\top \mathbf{y}, \tag{2.3}$$

which is the term of cosine-similarity.

Unlike the loss function that is designed in consideration of a pair of data, [6] suggests the concept of *proxy*, which is a representation vector of each classes. Here, we describe a two motivations of introducing proxies.

2.1.1 The View of Proxy - Fully Connected Layer in Classification

To classify data with C classes, $D \times C$ fully connected layer and cross-entropy loss are commonly used where D is the length of extracted vector from training model. Let $\mathbf{v_I} \in \mathbb{R}^D$ be an extracted vector from given image tensor $\mathbf{I}$. Then, $\mathbf{W} \in \mathbb{R}^{D \times C}$ fully connected layer with bias $\mathbf{b} \in \mathbb{R}^C$ extracts new vector $\mathbf{W}^\top \mathbf{v_I} + \mathbf{b} \in \mathbb{R}^C$. Here, $\mathbf{W}$ and $\mathbf{b}$ are trainable parameters. Now, the cross-entropy loss is applied which is defined as

$$\mathcal{L}(\mathbf{I}) = -\log \frac{\exp(u_{y_\mathbf{I}})}{\sum_{j=1}^{C} \exp(u_j)} \tag{2.4}$$

where $y_\mathbf{I}$ denotes the class index of image tensor $\mathbf{I}$, and $\mathbf{u} = [u_1,\ u_2,\ \cdots,\ u_C]^\top$. Note that if the model is trained with loss (2.4), then $u_{y_\mathbf{I}}$ becomes larger than other y_j's, and $\arg\max_j u_j$ might become $y_\mathbf{I}$. After training, we can determine the class of test image tensor $\mathbf{I}'$ by $\arg\max_j u'_j$ where $\mathbf{u} = [u_1,\ u_2,\ \cdots,\ u_C]^\top = \mathbf{W}^\top \mathbf{v_{I'}} + \mathbf{b}$.

For simplification, if we set $\mathbf{b} = \mathbf{0}$, the cross-entropy loss (2.4) becomes

$$\begin{aligned}\mathcal{L}(\mathbf{I}) &= -\log \frac{\exp(u_{y_\mathbf{I}})}{\sum_{j=1}^{C} \exp(u_j)} \\ &= -\log \frac{\exp(\mathbf{w}_{y_\mathbf{I}}^\top v_\mathbf{I})}{\sum_{j=1}^{C} \exp(\mathbf{w}_j^\top v_\mathbf{I})},\end{aligned}$$

when we write $\mathbf{W}$ as $[\mathbf{w}_1,\ \mathbf{w}_2,\ \cdots,\ \mathbf{w}_C]^\top$. Note that $\mathbf{w}_j \in \mathbb{R}^D$ $(1 \le j \le C)$. In this view, $\mathbf{v_I}$ can be considered as feature vector, and $\mathbf{w}_j^\top \mathbf{v_I}$ as a "relation" between class j and image tensor $\mathbf{I}$. In the context of the classification above, we can train the fully

connected layer $\mathbf{W}$ to make "relation" between class $y_{\mathbf{I}}$ and image tensor $\mathbf{I}$ larger than other relations.

Moreover, if $\|\mathbf{v}_{\mathbf{I}}\|_2 = 1$ and $\|\mathbf{w}_j\|_2 = 1$ for all j, $\mathbf{w}_j^\top \mathbf{v}_{\mathbf{I}}$ becomes a cosine-similarity between $\mathbf{w}_j$ and $\mathbf{v}_{\mathbf{I}}$, which is a metric on the D-dimensional unit hypersphere described in (2.1). The metric learning and classification can be related in this sense. From now on, $\mathbf{W}$ and $\mathbf{w}_j$ will be called *proxy* (or *proxy of class* j).

2.1.2 The View of Proxy - Learning a Semantic Vector

To distinguish features from image data, we have to determine some semantic object which is trainable. Thinking independently of classification, we should consider each point of the unit-hypersphere described above contains semantic meaning. [6] suggests a small data set $\mathcal{P} \subset \mathbb{S}_D$, which has a major feature of image and $|\mathcal{P}| = C$. Each feature vector have to be corresponded to one semantic meaning, i.e., element of $\mathcal{P}$. The main object of training is to learn a semantic meaning of feature vector $\mathbf{x}$ which can be defined as

$$p(\mathbf{x}) = \arg\max_{\mathbf{p} \in \mathcal{P}} s(\mathbf{x}, \mathbf{p}). \tag{2.5}$$

It can be seen that the D-dimensional vector proxy is proposed for both concepts. A *proxy-based loss* is a loss computed by proxies. In deep metric learning, the equation (2.4) is referred as *softmax-loss*, the process of forward-passing fully connected layer with zero biases and applying cross-entropy loss.

2.2 Examples of Proxy-based Loss

This section will observe some proxy-based losses and their advantages studied so far. We already introduced softmax loss in (2.4).

2.2.1 Angular Softmax Loss

To improve an intra-class and inter-class variation, [2] suggested angular margin between each classes. Angular softmax loss(A-softmax) uses angles between proxy and feature vector $\mathbf{x}$,

$$\theta_j = \arccos(\mathbf{w}_j^\top \mathbf{x}).$$

Note that the class of feature vector can be determined as $\arg\min_j \theta_j$. In other words, the class of feature vector $\mathbf{x}$ is c when

$$\theta_c < \theta_j, \ \ \forall j \neq c.$$

A-softmax gives more strict decision boundary than (2.4), the normal softmax loss. In A-softmax, the class of feature vector $\mathbf{x}$ is c when

$$m\theta_c < \theta_j, \ \ \forall j \neq c. \tag{2.6}$$

Here, m is an integer hyperparameter. The condition above means that the angle θ_c at which the class of vector $\mathbf{x}$ becomes c must be smaller.

In terms of angles, the equation (2.4) equals to

$$\mathcal{L}(\mathbf{x}) = -\log \frac{\exp(\cos(\theta_c))}{\sum_{j=1}^{C} \exp(\cos(\theta_j))}.$$

To adapt the condition (2.6) in the softmax loss, the loss would have a form such as

$$\mathcal{L}(\mathbf{x}) = -\log \frac{\exp(\cos(m\theta_c))}{\exp(\cos(m\theta_c)) + \sum_{j\neq c} \exp(\cos(\theta_j))}.$$

However the term $\cos(m\theta_c)$ cannot be generalized well, since $\cos(m\theta_c)$ can be maximized for not only $\theta_c = 0$, but also $\theta_c = \frac{2\pi k}{m}$ for any integer k. To overcome this problem, [] suggested a monotonically decreasing function

$$\psi(x) = (-1)^{\left\lfloor \frac{mx}{\pi} \right\rfloor} \cos(mx) - 2 \left\lfloor \frac{mx}{\pi} \right\rfloor,$$

and defined A-softmax loss as

$$\mathcal{L}_A(\mathbf{x}) = -\log \frac{\exp(\psi(m\theta_c))}{\exp(\psi(m\theta_c)) + \sum_{j\neq c} \exp(\cos(\theta_j))}. \tag{2.7}$$

Also, since m is an integer, $\psi(m\theta_c)$ can be represented as a differentiable function of $\cos(\theta_c) = \mathbf{w}_c^\top \mathbf{x}$ with Chebyshev polynomials of the first kind as follows:

$$\begin{aligned}\cos(m\theta_c) &= \sum_{k=0}^{\lfloor m/2 \rfloor} \binom{m}{2k} (\cos^2\theta_c - 1)^k \cos^{m-2k}\theta_c \\ &= \sum_{k=0}^{\lfloor m/2 \rfloor} \binom{m}{2k} ((\mathbf{w}_c^\top \mathbf{x})^2 - 1)^k (\mathbf{w}_c^\top \mathbf{x})^{m-2k}.\end{aligned}$$

Hence the gradient $\frac{\partial \mathcal{L}_A}{\partial \mathbf{w}_j}$ and $\frac{\partial \mathcal{L}_A}{\partial \mathbf{s}}$ can be computed well, and backpropagation of the loss is possible. With this margin m, the feature vectors become more distinguishable.

2.2.2 Proxy-NCA Loss

Neighbourhood Components Analysis(NCA) [15] proposes NCA loss

$$\mathcal{L}_{\text{NCA}}(\mathbf{x}, \mathbf{x}', \mathcal{Y}) = -\log \frac{\exp(-d(\mathbf{x}, \mathbf{x}'))}{\sum_{\mathbf{y}\in\mathcal{Y}} \exp(-d(\mathbf{x}, \mathbf{y}))}, \tag{2.8}$$

where $\mathbf{x}'$ denotes the feature vector which is from the same class of x, $\mathcal{Y}$ denotes the set of feature vectors of class different from $\mathbf{x}$, and d is a proper metric. Note that if we use a cosine-distance as

$$d(\mathbf{x}, \mathbf{y}) = 1 - s(\mathbf{x}, \mathbf{y}),$$

where s is from (2.1), the equation (2.8) becomes

$$\mathcal{L}_{\text{NCA}}(\mathbf{x}, \mathbf{x}', \mathcal{Y}) = -\log \frac{\exp(s(\mathbf{x}, \mathbf{x}'))}{\sum_{\mathbf{y}\in\mathcal{Y}} \exp(s(\mathbf{x}, \mathbf{y}))}. \tag{2.9}$$

The equation (2.9) is pair-based loss, and it is natural to change to proxy-based loss as follows:

$$\mathcal{L}_{\text{Proxy-NCA}}(\mathbf{x}, \mathcal{P}) = -\log \frac{\exp(\mathbf{p}_\mathbf{x}^\top \mathbf{x})}{\sum_{\mathbf{p}'\in\mathcal{P},\, \mathbf{p}'\neq\mathbf{p}_\mathbf{x}} \exp(\mathbf{p}'^\top \mathbf{x})}. \tag{2.10}$$

[6] defined the loss (2.10) as Proxy-NCA loss. Note that it has a similar form with softmax-loss.

2.2.3 Proxy-Anchor Loss

The motivation of Proxy-Anchor loss [9] is similar to Proxy-NCA loss, and [9] improved the gradient of the loss to make train more efficient. The loss is given as

$$\mathcal{L}_{\text{Proxy-Anchor}}(\mathcal{X}, \mathcal{P}) = \frac{1}{|\mathcal{P}^+|} \sum_{\mathbf{p} \in \mathcal{P}^+} \log \left(1 + \sum_{\mathbf{x} \in \mathcal{X}_{\mathbf{p}}^+} \exp(-\alpha s(\mathbf{x}, \mathbf{p}) - \delta) \right) + \frac{1}{|\mathcal{P}|} \sum_{\mathbf{p} \in \mathcal{P}} \log \left(1 + \sum_{\mathbf{x} \in \mathcal{X}_{\mathbf{p}}^-} \exp(\alpha s(\mathbf{x}, \mathbf{p}) + \delta) \right),$$

where $\mathcal{X}$ is a minibatch, $\mathcal{P}^+ = \{\mathbf{p} \in \mathcal{P} | \exists \mathbf{x} \in \mathcal{X} \text{s.t.} \mathbf{p} = \mathbf{p}_{\mathbf{x}}\}$, $\mathcal{X}_{\mathbf{p}}^+ = \{\mathbf{x} \in \mathcal{X} | \mathbf{p}_{\mathbf{x}} = \mathbf{p}\}$, $\mathcal{X}_{\mathbf{p}}^- = \mathcal{X} \backslash \mathcal{X}_{\mathbf{p}}^+$ and α, δ are hyperparameters. [9] focused on the gradient of the loss with respect to cosine-similarity between the feature vector and its corresponding proxy, $s(\mathbf{x}, \mathbf{p}_{\mathbf{x}}) + \delta$. Comparison of the gradient of Proxy-NCA and Proxy-Anchor as follows:

2.3 The Relation Between Proxy-based Loss and Pair-based Loss

Although the concept contained in two kinds of losses, proxy-based loss and pair-based loss are different, they are closely related to each other. If this fact can be discovered, we prefer the proxy-based loss since it has a good advantage of training time. In this section, we describe the relation between two kinds of losses.

The neural network is trained in the direction to reduce the value of the loss function. Therefore if the pair-based loss can be dominated by proxy-based loss, the reduction of pair-based loss can be guaranteed when proxy-based loss decreases. The idea of the propositions and explanations in this chpater below comes from [6].

Let's consider a simplest form of proxy-based loss

$$\mathcal{L}_p(\mathbf{x}, \mathcal{P}, \mathbf{y}) = g(\|\mathbf{x} - \mathbf{p}_{\mathbf{x}}\|_2) - g(\|\mathbf{x} - \mathbf{p}_{\mathbf{y}}\|_2),$$

where g is a same function in the equation (2.2). This loss trains feature vector $\mathbf{x}$ to get closer to $\mathbf{p}_\mathbf{x} \in \mathcal{P}$ and get further to another proxy $\mathbf{p}_\mathbf{y} \in \mathcal{P}$.

Proposition 1. *Suppose that there exists a real number $\epsilon > 0$ such that $\|\mathbf{x} - \mathbf{p}_\mathbf{x}\|_2 < \epsilon$ for all feature vector $\mathbf{x}$. Let $M = \sup_{t \in [0,\, 2]} g'(t)$. Then*

$$\mathcal{L}_t(\mathbf{x}, \mathbf{x}', \mathbf{y}) \leq \mathcal{L}_p(\mathbf{x}, \mathcal{P}, \mathbf{y}) + 2\epsilon M.$$

Proof. Since g is a non-decreasing function, and $\mathbf{p}_\mathbf{x} = \mathbf{p}_{\mathbf{x}'}$,

$$\begin{aligned}
\mathcal{L}_t(\mathbf{x}, \mathbf{x}', \mathbf{y}) &= g(\|\mathbf{x} - \mathbf{x}'\|_2) - g(\|\mathbf{x} - \mathbf{y}\|_2) \\
&\leq g(\|\mathbf{x} - \mathbf{p}_\mathbf{x}\|_2 + \|\mathbf{p}_\mathbf{x} - \mathbf{x}'\|_2) - g(\|\mathbf{x} - \mathbf{p}_\mathbf{y}\|_2 - \|\mathbf{p}_\mathbf{y} - \mathbf{y}\|_2) \\
&= g(\|\mathbf{x} - \mathbf{p}_\mathbf{x}\|_2 + \|\mathbf{p}_{\mathbf{x}'} - \mathbf{x}'\|_2) - g(\|\mathbf{x} - \mathbf{p}_\mathbf{y}\|_2 - \|\mathbf{p}_\mathbf{y} - \mathbf{y}\|_2) \\
&\leq g(\|\mathbf{x} - \mathbf{p}_\mathbf{x}\|_2 + \epsilon) - g(\max\{\|\mathbf{x} - \mathbf{p}_\mathbf{y}\|_2 - \epsilon, 0\}) \\
&= g(\|\mathbf{x} - \mathbf{p}_\mathbf{x}\|_2) - g(\|\mathbf{x} - \mathbf{p}_\mathbf{y}\|_2) + \epsilon(g'(a) + g'(b)) \\
&\leq g(\|\mathbf{x} - \mathbf{p}_\mathbf{x}\|_2) - g(\|\mathbf{x} - \mathbf{p}_\mathbf{y}\|_2) + 2\epsilon M \\
&= \mathcal{L}_p(\mathbf{x}, \mathcal{P}, \mathbf{y}) + 2\epsilon M
\end{aligned}$$

where $a \in [\|\mathbf{x} - \mathbf{p}_\mathbf{x}\|_2, \|\mathbf{x} - \mathbf{p}_\mathbf{x}\|_2 + \epsilon]$ and $b \in [\max\{\|\mathbf{x} - \mathbf{p}_\mathbf{y}\|_2 - \epsilon, 0\}, \|\mathbf{x} - \mathbf{p}_\mathbf{y}\|_2]$. □

Also, dominating proxy-based loss can be applied in NCA loss.

Proposition 2. *Suppose that there exists a real number $\epsilon > 0$ such that $\|\mathbf{x} - \mathbf{p}_\mathbf{x}\|_2 < \epsilon$ for all feature vector $\mathbf{x}$. Then,*

$$\mathcal{L}_{\textit{NCA}}(\mathbf{x}, \mathbf{x}', \mathcal{Y}) \leq \mathcal{L}_{\textit{Proxy-NCA}}(\mathbf{x}, \mathcal{P}) + 2\epsilon.$$

Proof. Note that

$$\mathbf{x}^\top(\mathbf{y} - \mathbf{p}_\mathbf{y}) \leq \|\mathbf{y} - \mathbf{p}_\mathbf{y}\|_2 < \epsilon$$

gives $\mathbf{x}^\top \mathbf{y} < \mathbf{x}^\top \mathbf{p}_\mathbf{y} + \epsilon$, and

$$\mathbf{x}^\top(\mathbf{x}' - \mathbf{p}_\mathbf{x}) \geq -\|\mathbf{x} - \mathbf{p}_{\mathbf{x}'}\|_2 > -\epsilon$$

gives $\mathbf{x}^\top \mathbf{x}' > \mathbf{x}^\top \mathbf{p}_\mathbf{x} - \epsilon$. Therefore,

$$\begin{aligned}
\mathcal{L}_{\text{NCA}}(\mathbf{x}, \mathbf{x}', \mathcal{Y}) &= -\log \frac{\exp(\mathbf{x}^\top \mathbf{x}')}{\sum_{\mathbf{y} \in \mathcal{Y}} \exp(\mathbf{x}^\top \mathbf{y})} \\
&= -\mathbf{x}^\top \mathbf{x}' + \log \left(\sum_{\mathbf{y} \in \mathcal{Y}} \exp(\mathbf{x}^\top \mathbf{y}) \right) \\
&< -\mathbf{x}^\top \mathbf{p}_\mathbf{x} + \epsilon + \log \left(\sum_{\mathbf{p} \in \mathcal{P} - \{\mathbf{p}_\mathbf{x}\}} \exp(\mathbf{x}^\top \mathbf{p}_\mathbf{y} + \epsilon) \right) \\
&= \mathcal{L}_{\text{Proxy-NCA}}(\mathbf{x}, \mathcal{P}) + 2\epsilon.
\end{aligned}$$

□

Chapter 3

Analysis of Softmax-Loss

One of the main point in deep learning is loss function and its gradient. As we have seen in Chapter 2, there are many variations of softmax-loss, and training has been improved by these losses in intra-class and inter class variation, computation time, and gradient. Even with these advantages, proxy-based losses are not guaranteed by strict mathematical contributions. The main gap in proxy-based losses are following:

Assumption: Feature vectors and proxies will be sufficiently close to each other [6,8,10], i.e.,

$$\|\mathbf{f}_{c,i} - \mathbf{w}_c\|_2 \to 0.$$

This assumptions highly depends on the explaination of the intra-class and inter-class variation since these are closely related to behavior of feature vectors. In this chapter, we will try to discover this assumption by analyzing softmax-loss.

3.1 Relative Location Between Feature Vectors and Proxies

First we have to check is the global minimum point of training loss in unit hypersphere $\mathbb{S}$. Recall the softmax loss

$$\mathcal{L}(\mathbf{f}) = -\log \frac{\exp(\mathbf{w}_{y_\mathbf{f}}^\top \mathbf{f})}{\sum_{j=1}^{C} \exp(\mathbf{w}_j^\top \mathbf{f})}. \tag{3.1}$$

where $y_{\mathbf{f}}$ denotes the class of the feature vector $\mathbf{f}$. The importance of determining relative location comes from the criterion of softmax-loss. In classification, the class of the feature vector can be determined as

$$y_{\mathbf{f}} = c \text{ if and only if } s(\mathbf{w}_c, \mathbf{f}) \geq s(\mathbf{w}_j, \mathbf{f}) \text{ for every } j. \tag{3.2}$$

However, the softmax loss doesn't always train feature vectors toward to (3.2) actually. For example, consider the case $\mathbf{w}_1 = [1, 0]^\top$, $\mathbf{w}_2 = [\cos(\epsilon), \sin(\epsilon)]^\top$, and $\mathbf{w}_3 = [0, -1]^\top$ with the dimension of embedding space $D = 2$. Then, the loss would be

$$\begin{aligned}\mathcal{L}(\cos\theta, \sin\theta) &= -\log \frac{\exp(\cos\theta)}{\exp(\cos\theta) + \exp(\cos\theta\cos\epsilon + \sin\theta\cos\epsilon) + \exp(-\sin\theta)} \\ &= -\log \frac{1}{1 + \exp(\cos(\theta - \epsilon) - \cos\theta) + \exp(-\cos\theta - \sin\theta)}\end{aligned}$$

where the feature vector $\mathbf{f}$ is located at $(\cos\theta, \sin\theta)$, and its class is 1. Here, the minimum point is closer to $\mathbf{w}_2$ than $\mathbf{w}_1$, even though its class is 1. (Numerical computation shows this result.) In this view, it is necessary to analyze the relative location between feature vectors and proxy more clearly.

The points we are interested in is the minimum point of the loss $\mathcal{L}$. Let $\mathbf{x}_c$ be such point, i.e.,

$$\mathbf{x}_{y_{\mathbf{f}}} = \arg\min_{\mathbf{f} \in \mathbb{S}} \mathcal{L}(\mathbf{f}).$$

Since the final state of feature vector in class c would be $\mathbf{x}_c$, it is enough to analyze the point $\mathbf{x}_c$. The statement below gives a relation between $\mathbf{x}_c$ and $\mathbf{w}_c$.

Proposition 3. *The point $\mathbf{x}_c$ satisfies the equation below:*

$$\mathbf{x}_c = \lambda_c \sum_j (\mathbf{w}_c - \mathbf{w}_j) \exp(\mathbf{w}_j^\top \mathbf{x}_c) \tag{3.3}$$

where

$$\lambda_c = \left[\sum_{j=1}^{C} (\mathbf{w}_c^\top \mathbf{x}_c - \mathbf{w}_j^\top \mathbf{x}_c) \exp(\mathbf{w}_j^\top \mathbf{x}_c) \right]^{-1}.$$

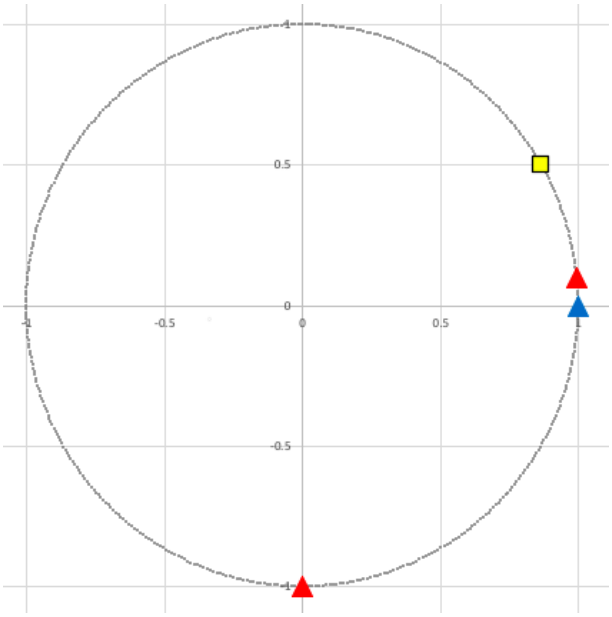

Figure 3.1: The embedding space with dimension 2. The blue point refers to $\mathbf{w}_1$, and red points refer to $\mathbf{w}_2$ and $\mathbf{w}_3$. The yellow point is the global minimum point of softmax-loss. In this figure, ϵ was set to 0.1.

Proof. We have to minimize the function $\mathcal{L}(\mathbf{f})$ with constraint $\mathbf{f} \in \mathbb{S}$, which is equivalent to $\|\mathbf{f}\|_2 = 1$. Let $\phi(\cdot) = \|\cdot\|_2$. By Lagrange multiplier method, there exists a real scalar λ_0 such that

$$\nabla_{\mathbf{f}}\phi(\mathbf{f})\Big|_{\mathbf{f}=\mathbf{x}_c} = \lambda_0 \nabla_{\mathbf{f}}\mathcal{L}(\mathbf{f})\Big|_{\mathbf{f}=\mathbf{x}_c}$$

To get a differentiation, we write $\mathbf{f} = [f_1,\ f_2,\ \cdots,\ f_D]^\top$. Also we use the notation ∂_i which refers to $\frac{\partial}{\partial f_i}$ for convenience. First, from $\|\mathbf{f}\|_2 = 1$,

$$\partial_i \phi(\mathbf{f}) = \partial_i \sqrt{\sum_k f_k^2} = \frac{f_i}{\sqrt{\sum_k f_k^2}} = f_i,$$

therefore $\nabla_{\mathbf{f}}\phi(\mathbf{f}) = \mathbf{f}$. Next,

$$\begin{aligned}
\partial_i \mathcal{L}(\mathbf{f}) &= -\partial_i \log \frac{\exp(\mathbf{w}_{y_\mathbf{f}}^\top \mathbf{f})}{\sum_{j=1}^C \exp(\mathbf{w}_j^\top \mathbf{f})} \\
&= -\frac{\sum_{j=1}^C \exp(\mathbf{w}_j^\top \mathbf{f})}{\exp(\mathbf{w}_{y_\mathbf{f}}^\top \mathbf{f})} \partial_i \left(\frac{\exp(\mathbf{w}_{y_\mathbf{f}}^\top \mathbf{f})}{\sum_{j=1}^C \exp(\mathbf{w}_j^\top \mathbf{f})} \right) \\
&= -\frac{\sum_{j=1}^C \exp(\mathbf{w}_j^\top \mathbf{f})}{\exp(\mathbf{w}_{y_\mathbf{f}}^\top \mathbf{f})} \cdot \frac{1}{\left[\sum_{j=1}^C \exp(\mathbf{w}_j^\top \mathbf{f})\right]^2} \\
&\quad \cdot \left(\partial_i (\exp(\mathbf{w}_{y_\mathbf{f}}^\top \mathbf{f})) \sum_{j=1}^C \exp(\mathbf{w}_j^\top \mathbf{f}) - \exp(\mathbf{w}_{y_\mathbf{f}}^\top \mathbf{f}) \partial_i \left[\sum_{j=1}^C \exp(\mathbf{w}_j^\top \mathbf{f}) \right] \right) \\
&= -\frac{w_{y_\mathbf{f} i} \exp(\mathbf{w}_{y_\mathbf{f}}^\top \mathbf{f}) \sum_{j=1}^C \exp(\mathbf{w}_j^\top \mathbf{f}) - \exp(\mathbf{w}_{y_\mathbf{f}}^\top \mathbf{f}) \sum_{j=1}^C w_{ji} \exp(\mathbf{w}_j^\top \mathbf{f})}{\exp(\mathbf{w}_{y_\mathbf{f}}^\top \mathbf{f}) \sum_{j=1}^C \exp(\mathbf{w}_j^\top \mathbf{f})} \\
&= -\frac{\sum_{j=1}^C (w_{y_\mathbf{f} i} - w_{ji}) \exp(\mathbf{w}_j^\top \mathbf{f})}{\sum_{j=1}^C \exp(\mathbf{w}_j^\top \mathbf{f})}
\end{aligned}$$

where $\mathbf{w}_j = [w_{j1},\ w_{j2},\ \cdots,\ w_{jD}]^\top$. Therefore,

$$\nabla_{\mathbf{f}} \mathcal{L}(\mathbf{f}) = -\frac{\sum_{j=1}^C (\mathbf{w}_{y_\mathbf{f}} - \mathbf{w}_j) \exp(\mathbf{w}_j^\top \mathbf{f})}{\sum_{j=1}^C \exp(\mathbf{w}_j^\top \mathbf{f})}. \tag{3.4}$$

As a result,

$$\mathbf{x}_c = -\lambda_0 \frac{\sum_{j=1}^C (\mathbf{w}_c - \mathbf{w}_j) \exp(\mathbf{w}_j^\top \mathbf{x}_c)}{\sum_{j=1}^C \exp(\mathbf{w}_j^\top \mathbf{x}_c)} = \lambda_c \sum_{j=1}^C (\mathbf{w}_c - \mathbf{w}_j) \exp(\mathbf{w}_j^\top \mathbf{x}_c)$$

where $\lambda_c = -\lambda_0 \frac{1}{\sum_{j=1}^C \exp(\mathbf{w}_j^\top \mathbf{x}_c)}$. If we dot product $\mathbf{x}_c$ on both sides,

$$1 = \mathbf{x}_c^\top \mathbf{x}_c = \lambda_c \sum_{j=1}^C (\mathbf{w}_c^\top \mathbf{x}_c - \mathbf{w}_j^\top \mathbf{x}_c) \exp(\mathbf{w}_j^\top \mathbf{x}_c),$$

$$\lambda_c = \left[\sum_{j=1}^C (\mathbf{w}_c^\top \mathbf{x}_c - \mathbf{w}_j^\top \mathbf{x}_c) \exp(\mathbf{w}_j^\top \mathbf{x}_c) \right]^{-1}.$$

□

Of course, the relation (3.3) does not give an explicit solution of $\mathbf{x}_c$, therefore we must indirectly look for relative positions of feature vectors and proxies. The main

point in this subsection is to determine a gap between feature vectors and proxies, i.e., $\|\mathbf{x}_c - \mathbf{w}_c\|_2$.

Theorem 1. *The sum of gaps between feature vectors and proxies has a lower bound as*

$$\sum_{c=1}^{C} \|\mathbf{x}_c - \mathbf{w}_c\|_2^2 \geq \|\frac{1}{C^2}\sum_{j,j'} \mathbf{s}_{j,j'}\|_2^2. \tag{3.5}$$

where $\mathbf{s}_{j,j'} = \frac{\mathbf{w}_j + \mathbf{w}_{j'}}{8}(1 - \mathbf{w}_j^\top \mathbf{w}_{j'}) \exp(\mathbf{w}_j^\top \mathbf{w}_{j'} - 1)$.

Proof. Suppose that

$$\|\mathbf{x}_c - \mathbf{w}_c\|_2 < \|\frac{1}{C^2}\sum_{j,j'} \mathbf{s}_{j,j'}\|_2 =: \epsilon$$

for every class index c. We introduce new vectors $\tilde{\mathbf{w}}_c$, as

$$\tilde{\mathbf{w}}_c = \sum_{j=1}^{C} (\mathbf{w}_c - \mathbf{w}_j) \exp(\mathbf{w}_j^\top \mathbf{w}_c)$$

for $1 \leq c \leq C$. Note that

$$\sum_{c=1}^{C} \tilde{\mathbf{w}}_c = \sum_{c=1}^{C}\sum_{j=1}^{C} (\mathbf{w}_c - \mathbf{w}_j) \exp(\mathbf{w}_j^\top \mathbf{w}_c) = 0.$$

Therefore,

$$
\begin{aligned}
&\left\|\sum_{j,j'}(\mathbf{w}_j+\mathbf{w}_{j'})(1-\mathbf{w}_j^\top\mathbf{w}_{j'})\exp(\mathbf{w}_j^\top\mathbf{w}_{j'})\right\|_2\\
=&\left\|\sum_c\sum_j\mathbf{w}_c(1-\mathbf{w}_j^\top\mathbf{w}_c)\exp(\mathbf{w}_j^\top\mathbf{w}_c)\right\|_2\\
\le&\left\|\sum_c\sum_j(\mathbf{w}_c-\mathbf{x}_c)(1-\mathbf{w}_j^\top\mathbf{w}_c)\exp(\mathbf{w}_j^\top\mathbf{w}_c)\right\|_2\\
&+\left\|\sum_c\mathbf{x}_c\left[\sum_j(1-\mathbf{w}_j^\top\mathbf{w}_c)\exp(\mathbf{w}_j^\top\mathbf{w}_c)-\frac{1}{\lambda_c}\right]\right\|_2+\left\|\sum_c\frac{\mathbf{x}_c}{\lambda_c}\right\|_2\\
\le&\sum_c\|(\mathbf{w}_c-\mathbf{x}_c)\|_2\sum_j\left\|(1-\mathbf{w}_j^\top\mathbf{w}_c)\exp(\mathbf{w}_j^\top\mathbf{w}_c)\right\|_2\\
&+\left\|\sum_c\mathbf{x}_c\left[\sum_j(1-\mathbf{w}_j^\top\mathbf{w}_c)\exp(\mathbf{w}_j^\top\mathbf{w}_c)-\sum_j(\mathbf{w}_c^\top\mathbf{x}_c-\mathbf{w}_j^\top\mathbf{x}_c)\exp(\mathbf{w}_j^\top\mathbf{x}_c)\right]\right\|_2\\
&+\left\|\sum_c\sum_j(\mathbf{w}_c-\mathbf{w}_j)\exp(\mathbf{w}_j^\top\mathbf{x}_c)\right\|_2\\
\le&2eC(C-1)\epsilon+4eC^2\epsilon+\left\|\sum_c\sum_j(\mathbf{w}_c-\mathbf{w}_j)\exp(\mathbf{w}_j^\top\mathbf{w}_c)\right\|_2\\
&+\left\|\sum_c\sum_j(\mathbf{w}_c-\mathbf{w}_j)[\exp(\mathbf{w}_j^\top\mathbf{w}_c)-\exp(\mathbf{w}_j^\top\mathbf{x}_c)]\right\|_2\\
\le&2eC(C-1)\epsilon+4eC^2\epsilon+\left\|\sum_c\tilde{\mathbf{w}}_c\right\|_2+2eC(C-1)\epsilon=4e(2C^2-C)\epsilon.
\end{aligned}
$$

Therefore, $8eC^2\epsilon\le 4e(2C^2-C)\epsilon$, which leads contradiction. Hence there exist a class index c such that $\|\mathbf{x}_c-\mathbf{w}_c\|_2\ge\|\frac{1}{C^2}\sum_{j,j'}\mathbf{s}_{j,j'}\|_2$, and the conclusion follows. □

The theorem above implies there is a *lower bound* gap, $\|\frac{1}{C^2}\sum_{j,j'}\mathbf{s}_{j,j'}\|_2$, between feature vectors and proxies.

Now we provide *upper bound* of gap. Previously, we need a lemma here.

Lemma 1. *If* $\bar{\mathbf{w}}=\frac{1}{C}\sum_j\mathbf{w}_j$, *then* $\mathbf{w}_c^\top\mathbf{x}_c\ge\bar{\mathbf{w}}^\top\mathbf{x}_c$ *where* $\bar{\mathbf{w}}=\frac{1}{C}\sum_j\mathbf{w}_j$.

Proof. By definition of $\mathbf{x}_c$, $\mathcal{L}(\mathbf{x}_c) \le \mathcal{L}(\mathbf{w}_c)$. This inequality gives

$$\frac{\exp(\mathbf{w}_c^\top \mathbf{x}_c)}{\sum_j \exp(\mathbf{w}_j^\top \mathbf{x}_c)} \ge \frac{\exp(\mathbf{w}_c^\top \mathbf{w}_c)}{\sum_j \exp(\mathbf{w}_j^\top \mathbf{w}_c)} \ge \frac{1}{C}.$$

And by Jensen's inequality,

$$\exp(\mathbf{w}_c^\top \mathbf{x}_c) \ge \frac{1}{C}\sum_j \exp(\mathbf{w}_j^\top \mathbf{x}_c) \ge \exp(\bar{\mathbf{w}}^\top \mathbf{x}_c).$$

which leads the conclusion. □

Theorem 2. *If* $\bar{\mathbf{w}}^\top \mathbf{x}_c > 0$*, then*

$$\|\mathbf{x}_c - \mathbf{w}_c\|_2^2 \le 2 - \sqrt{\frac{2}{e}(1 - \bar{\mathbf{w}}^\top \mathbf{w}_c)} \tag{3.6}$$

Proof. First, by the lemma, $\mathbf{w}_c^\top \mathbf{x}_c \ge \bar{\mathbf{w}}^\top \mathbf{x}_c > 0$.

Consider a function $\varphi(x) = (\mathbf{w}_c^\top \mathbf{x}_c - x)e^x$. The function φ is concave in $[-1, 1]$. This gives

$$\begin{aligned}\frac{1}{\lambda_c} &= \sum_{j=1}^{C} (\mathbf{w}_c^\top \mathbf{x}_c - \mathbf{w}_j^\top \mathbf{x}_c)\exp(\mathbf{w}_j^\top \mathbf{x}_c) \\ &= \sum_{j=1}^{C} \varphi(\mathbf{w}_j^\top \mathbf{x}_c) \le C\varphi\left(\frac{1}{C}\sum_{j=1}^{C} \mathbf{w}_j^\top \mathbf{x}_c\right) = C(\mathbf{w}_c^\top \mathbf{x}_c - \bar{\mathbf{w}}^\top \mathbf{x}_c)\exp(\bar{\mathbf{w}}^\top \mathbf{x}_c)\end{aligned}$$

by Jensen's inequality, where $\bar{\mathbf{w}} = \frac{1}{C}\sum_j \mathbf{w}_j$. On the other hand, the dot product of $\mathbf{w}_c$ in both side of (3.3) gives

$$\mathbf{w}_c^\top \mathbf{x}_c = \lambda_c \sum_{j=1}^{C} (1 - \mathbf{w}_c^\top \mathbf{w}_j)\exp(\mathbf{w}_j^\top \mathbf{x}_c)$$

By Jensen's inequality again, since $1 - \mathbf{w}_c^\top \mathbf{w}_j \ge 0$ for all j,

$$\sum_{j=1}^{C} (1 - \mathbf{w}_c^\top \mathbf{w}_j)\exp(\mathbf{w}_j^\top \mathbf{x}_c) \ge \left[\sum_{j=1}^{C} 1 - \mathbf{w}_c^\top \mathbf{w}_j\right] \exp\left(\mathbf{w}'^\top \mathbf{x}_c\right)$$

where

$$\mathbf{w}' = \sum_{j=1}^{C} \frac{1 - \mathbf{w}_c^\top \mathbf{w}_j}{\sum_{k=1}^{C} 1 - \mathbf{w}_c^\top \mathbf{w}_k} \mathbf{w}_j.$$

Two inequalities give the relation

$$\frac{1}{\mathbf{w}_c^\top \mathbf{x}_c}\left[\sum_{j=1}^{C} 1 - \mathbf{w}_c^\top \mathbf{w}_j\right] \exp\left(\mathbf{w}'^\top \mathbf{x}_c\right) \leq \frac{1}{\lambda_c} \leq C(\mathbf{w}_c - \bar{\mathbf{w}})^\top \mathbf{x}_c \exp(\bar{\mathbf{w}}^\top \mathbf{x}_c)$$

$$(1 - \bar{\mathbf{w}}^\top \mathbf{w}_c)\exp((\mathbf{w}' - \bar{\mathbf{w}})^\top \mathbf{x}_c) \leq (\mathbf{w}_c^\top \mathbf{x}_c)(\mathbf{w}_c - \bar{\mathbf{w}})^\top \mathbf{x}_c$$

$$(1 - \bar{\mathbf{w}}^\top \mathbf{w}_c)\exp(-\|\mathbf{w}' - \bar{\mathbf{w}}\|_2) \leq (\mathbf{w}_c^\top \mathbf{x}_c)^2$$

Therefore,

$$\mathbf{w}_c^\top \mathbf{x}_c \geq \sqrt{1 - \bar{\mathbf{w}}^\top \mathbf{w}_c}\exp(-\frac{1}{2}\|\mathbf{w}' - \bar{\mathbf{w}})\|_2) \geq \sqrt{1 - \bar{\mathbf{w}}^\top \mathbf{w}_c}\exp(-\frac{1}{2})$$

and this gives the result. □

3.2 Improving Softmax Loss

Our main target of this paper is improving softmax loss with using the theorems that we observed in the previous section. The theorems can be concluded as

$$L(\mathbf{W}) \leq \frac{1}{C}\sum_{c=1}^{C} \|\mathbf{x}_c - \mathbf{w}_c\|_2^2 \leq U(\mathbf{W}). \tag{3.7}$$

where

$$L(\mathbf{W}) = \frac{1}{C^3}\|\sum_{j,j'} \mathbf{s}_{j,j'}\|_2^2 \tag{3.8}$$

and

$$U(\mathbf{W}) = \ 2 - \frac{1}{C}\sqrt{\frac{2}{e}}\sum_{j=1}^{C}\sqrt{1 - \bar{\mathbf{w}}^\top \mathbf{w}_j} \tag{3.9}$$

when $\bar{\mathbf{w}}^\top \mathbf{x}_c > 0$ for every c. The proper additional loss term we will going to suggest minimizes $U(\mathbf{W})$. If $U(\mathbf{W})$ goes to zero, the gap $\|\mathbf{x}_c - \mathbf{w}_c\|_2$ will be reduced, and the accuracy of training loss will become better.

For reducing the term $U(\mathbf{W})$, we have to make $\frac{1}{C}\sum_{c=1}^{C}\sqrt{1 - \bar{\mathbf{w}}^\top \mathbf{w}_c}$ large. We can expect that if $\sum_c \bar{\mathbf{w}}^\top \mathbf{w}_c$ become small, then $U(\mathbf{W})$ become small also. The term $\sum_c \bar{\mathbf{w}}^\top \mathbf{w}_c$ is exactly same as $C\|\bar{\mathbf{w}}\|_2^2$.

As a result, if we reduce the norm of the mean vector of proxies $\|\bar{\mathbf{w}}\|_2$, then $U(\mathbf{W})$ will become small. In this sense, we suggest a new loss

$$\mathcal{L}_{\text{improved}}(\mathbf{x}, \mathbf{W}) = \mathcal{L}_{\text{softmax}}(\mathbf{x}, \mathbf{W}) + \alpha\|\bar{\mathbf{w}}\|_2 \tag{3.10}$$

where α is a hyperparameter.

Chapter 4

Experiments and Results

In this section, we train the specific model to check the improvement of our suggested loss, (3.10).

4.1 Datasets

In deep metric learning, CUB-200-2011 [16] and Cars-196 [17] are commonly used benchmark datasets. For CUB-200-2011, there are 5864 images with 100 classes for training, and 5924 images with other 100 classes for testing. For Cars-196, there are 8054 images with 98 classes for training, and 8131 images with other 98 classes for testing.

4.2 Implementation Details

Initialization of Parameters: We use ResNet-50 backbone network with pre-trained weight by ImageNet [1]. We did not change the size of its last layer of ResNet-50 network, so the dimension of the embedding space is 2048. Also, proxies are initialized randomly with Kaiming initialization [18].

Learning Schedule: In every experiment, we used stochastic gradient descent(SGD)

method with nesterov to do backpropagation during training. In SGD, we applied the momentum 0.9, and weight decay 5×10^{-4}. Every parameter used in the experiment trained with initial learning rate 10^{-3}, and learning rates were dropped every 10 epoch with multiplicative factor of learning rate decay 0.1. Every results are maintained after 20-epochs.

Image setting: We followed the standard pre-processing method in deep metric learning test [9]. We centercroped images with size 224×224.

Evaluation Metric: The Recall@K method [19] has been employed in this experiments.

Hyperparameter: We set the hyperparameter α to 1.0 in CUB-200-2011, and 0.01 in Cars-196.

4.3 Results

In our experiments, our loss shows good generalization in both two datasets as Table 4.1. The Recall@1 was improved about 0.5% in CUB-200-2011, and about 1.0% in Cars-196.

Table 4.1: The results of Recall@K in datasets CUB-200-2011 and Cars-196

	CUB-200-2011				Cars-196			
Recall@K	1	2	4	8	1	2	4	8
Softmax-loss	59.69	70.75	**80.47**	87.64	79.49	86.79	91.39	94.83
Ours	**60.16**	**71.22**	80.38	**87.86**	**80.41**	**87.59**	**92.12**	**95.52**

Also, we computed Recall@1 for every epoch and plotted as a graph. Figure 4.1 and 4.2 shows that our improved loss works stable and gives better performance in both two datasets.

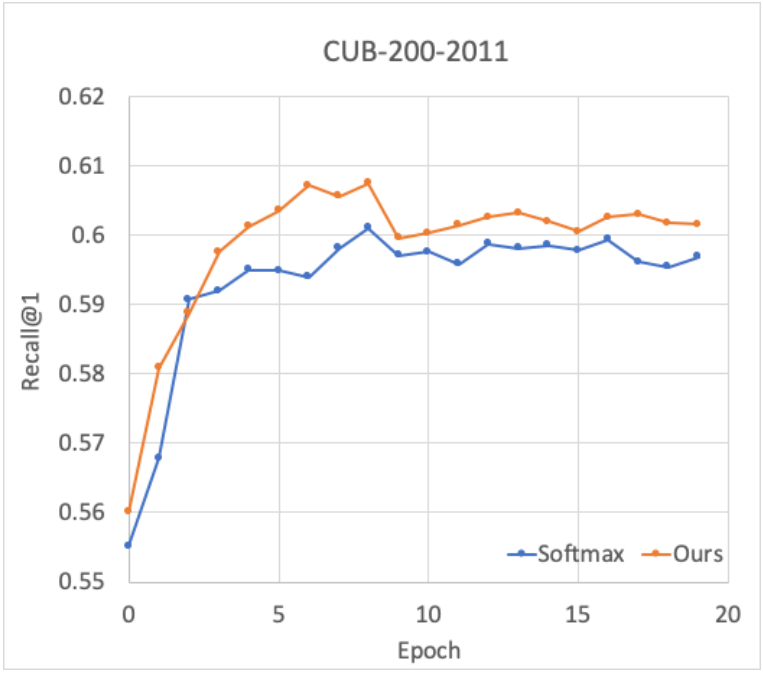

Figure 4.1: Recall@1 graph in the dataset CUB-200-2011.

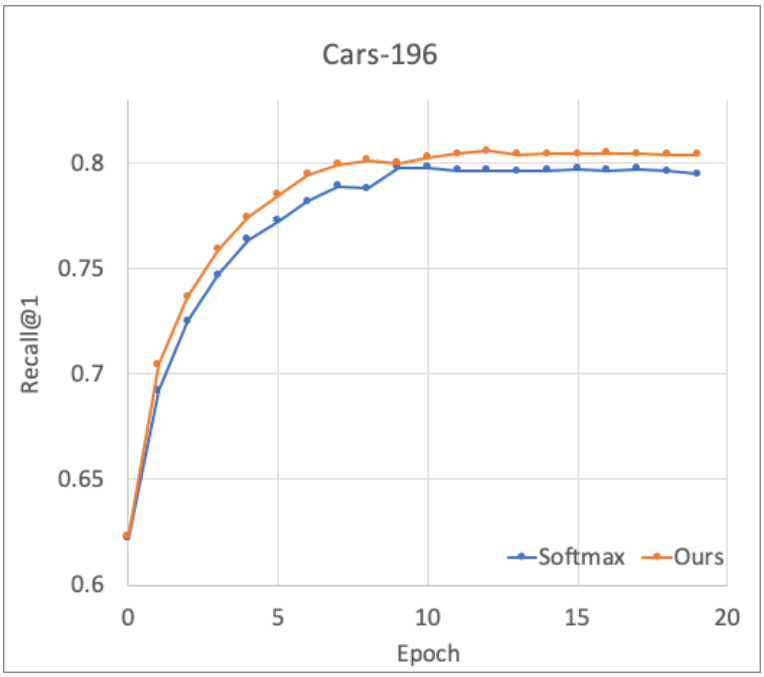

Figure 4.2: Recall@1 graph in the dataset Cars-196.

Chapter 5

Conclusion

Proxy-based loss is widely used due to its benefits of time complexity and convenience of loss control. Although these advantages, there are not much studies about analysis of behavior of feature vectors. It is important to check the behavior of feature vector since the movement of vectors determine the intra-class and inter-class distance, which is highly related to better performance and generalization. In this paper, this study introduced the inequality that shows the variation of the gap between feature vector and corresponding proxy. Moreover, we showed that the variation of the gap can be controlled by the norm of the mean vector of proxies, $\bar{\mathbf{w}}$. In this view, this study suggests a new loss with additional term which role is to reduce the norm of $\bar{\mathbf{w}}$. Several experiments show that our suggested loss improved the test accuracy in two datasets from softmax loss baseline.

It is difficult to measure the behavior of feature vectors because of the complexity of the proxy-based loss function. In further works, other proxy-based losses, such as A-softmax and Proxy-Anchor may be analyzed with similar approach.

Bibliography

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Ieee, 2009.

[2] Sphereface: Deep hypersphere embedding for face recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[3] Dmitry Kalenichenko Florian Schroff and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[4] R. Hadsell S. Chopra and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[5] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R Scott. Multi-similarity loss with general pair weighting for deep metric learning. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5022–5030, 2019.

[6] Thomas K Leung Sergey Ioffe Yair Movshovitz-Attias, Alexander Toshev and Saurabh Singh. No fuss distance metric learning using proxies. In *Proc. IEEE International Conference on Computer Vision (ICCV),*, 2017.

[7] Z. Yu W. Liu, Y. Wen and M. Yang. Large-margin softmax loss for convolutional neural networks. In *ICML*, 2016.

[8] Baigui Sun Juhua Hu Hao Li Qi Qian, Lei Shang and Rong Jin. Softtriple loss: Deep metric learning without triplet sampling. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2019.

[9] Sungyeon Kim, Dongwon Kim, Minsu Cho, and Suha Kwak. Proxy anchor loss for deep metric learning. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[10] Ian Reid Vijay Kumar Tuan Hoang Thanh-Toan Do, Toan Tran and Gustavo Carneiro. A theoretically sound upper bound on the triplet loss for improving the efficiency of deep distance metric learning. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[11] Dennis Fedorishin Srirangaraj Setlur Venu Govindaraju Deen Dayal Mohan, Nishant Sankaran. Moving in the right direction: A regularization for deep metric learning. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[13] Y. Qiang L. Chunjie. Cosine normalization: Using cosine similarity instead of dot product in neural networks. In *arXiv preprint arXiv:1702.05870*, 2017.

[14] M. Kankanhalli Y. Luo, Y. Wong and Q. Zhao. $\mathcal{G}$-softmax: Improving intraclass compactness and interclass separability of features. In *IEEE Trans. Neural Netw. Learn. Syst., vol. 31, no. 2, pp. 685-699*, 2020.

[15] Geoff Hinton Ruslan Salakhutdinov Jacob Goldberger, Sam Roweis. Neighbourhood components analysis. In *Adv. Neural Inf. Process. Syst.(NIPS)*, 2004.

[16] T. Mita C. Wah F. Schroff S. Belongie P. Welinder, S. Branson and P. Perona. Caltech-ucsd birds 200. *Technical Report CNS-TR-2010-001, California Institute of Technology*, 2010.

[17] Jia Deng Jonathan Krause, Michael Stark and Li Fei-Fei. 3d object representations for fine-grained categorization. *Proceedings of the IEEE International Conference on Computer Vision Workshops, pages 554–561*, 2013.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

[19] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *CVPR*, 2016.

초 록

딥 메트릭 학습은 이미지 검색 및 얼굴 검증 분야에서 널리 사용되는 학습 방법이다. 이 학습 방법은 각 데이터를 특정 임베딩 공간에 대응시키면서 특성 간의 의미적 거리를 학습하는 방법을 말한다. 임베딩 공간에서의 메트릭을 배우기 위해, 지금까지 다양한 손실 함수 및 데이터 샘플링 방법이 개발되어 왔다. 특별히, 프록시 기반 손실 함수는 계산 속도에 있어 우수한 장점을 갖고 있어 많이 사용되고 있다. 딥 메트릭 학습에서 성능을 결정하는 중요한 요소 중 하나는 클래스 내 거리와 클래스 간 거리인데, 프록시 기반 손실 함수에서는 이 거리를 분석하는 구체적인 연구가 이루어지지 않았다. 본 연구에서는 프록시 기반 손실에서 기본 프레임으로 사용되는 손실 함수인 소프트 맥스 손실 함수를 분석하여 클래스 내 거리 및 클래스 간 거리를 개선하는 방법을 알아보았다. 그리고 얻은 분석 결과를 사용하여 성능을 향상시킬 수있는 새로운 손실을 제안하였다. 마지막으로 본 연구에서 제안한 손실 함수가 ResNet-50 모델의 성능 향상에 효과적인 것으로 실험을 통해 확인하였다.

주요어: 딥 메트릭 러닝, 이미지 리트리벌, 미세 조정, 특징 벡터, 클래스 내 거리, 클래스 간 거리

학번: 2018-29521