Learning embeddings: efficient algorithms and applications

Thèse n. 8348
présentée le 20 Décembre 2017
à la Faculté des Sciences et Techniques de l'Ingénieur
Programme Doctoral en Génie Électrique (EDEE)
Laboratoire LIDIAP (Idiap Research Institute)
École Polytechnique Fédérale de Lausanne

pour l'obtention du grade de Docteur ès Sciences
par

Cijo Jose

acceptée sur proposition du jury:

Prof Pascal Frossard, président du jury
Dr François Fleuret, directeur de thèse
Dr Olivier Bousquet, rapporteur
Dr Moustapha Cissé, rapporteur
Dr Mathieu Salzmann, rapporteur

Lausanne, EPFL, 2017

Abstract

Learning to embed data into a space where similar points are together and dissimilar points are far apart is a challenging machine learning problem. In this dissertation we study two learning scenarios that arise in the context of learning embeddings and one scenario in efficiently estimating an empirical expectation. We present novel algorithmic solutions and demonstrate their applications on a wide range of data-sets.

The first scenario deals with learning from small data with large number of classes. This setting is common in computer vision problems such as person re-identification and face verification. To address this problem we present a new algorithm called Weighted Approximate Rank Component Analysis (WARCA), which is scalable, robust, non-linear and is independent of the number of classes. We empirically demonstrate the performance of our algorithm on 9 standard person re-identification data-sets where we obtain state of the art performance in terms of accuracy as well as computational speed.

The second scenario we consider is learning embeddings from sequences. When it comes to learning from sequences, recurrent neural networks have proved to be an effective algorithm. However there are many problems with existing recurrent neural networks which makes them data hungry (high sample complexity) and difficult to train. We present a new recurrent neural network called Kronecker Recurrent Units (KRU), which addresses the issues of existing recurrent neural networks through Kronecker matrices. We show its performance on 7 applications, ranging from problems in computer vision, language modeling, music modeling and speech recognition.

Most of the machine learning algorithms are formulated as minimizing an empirical expectation over a finite collection of samples. In this thesis we also investigate the problem of efficiently estimating a weighted average over large data-sets. We present a new data-structure called Importance Sampling Tree (IST), which permits fast estimation of weighted average without looking at all the samples. We show successfully the evaluation of our data-structure in the training of neural networks in order to efficiently find informative samples.

Keywords: Learning embedding, Metric learning, Orthonormal, Unitary regularizer, Person re-identification, Recurrent neural networks, Kronecker product, Importance sampling.

Résumé

Apprendre à représenter des données dans un espace où les points similaires sont proches les un des autres et les points dissimilaires sont éloignés les uns des autres est un problème d'apprentissage machine difficile. Cette thèse s'attèle à deux scénarios d'apprentissage que l'on retrouve dans le contexte des transformations d'apprentissage, ainsi quà un scénario concernant l'estimation d'une espérance empirique de façon efficace. Nous présentons de nouvelles solutions algorithmiques et démontrons leurs applications sur un large éventail d'ensembles de données.

Le premier scénario s'articule autour de l'apprentissage á partir de faibles ressources sur un grand nombre de classes. Ce [setting] est courant dans les problèmes de vision par ordinateur comme par exemple la réidentification de personnes ou encore la reconnaissance faciale. Nous proposons un algorithme pour résoudre ce problème : WARCA (pour Weighted Approximate Rank Component Analysis, i.e. analyse pondérée des composantes de rang approximatif). Cette méthode est robuste, non linéaire, indépendante du nombre de classes et évolutive. Nous démontrons empiriquement les performances de notre algorithme sur 9 bases de données standard de réidentification de personnes : nous obtenons des performances de pointe en termes de précision et de vitesse de calcul.

Le deuxième scénario que nous étudions est l'apprentissage de représentations à partir de séquences. Les réseaux de neurones récurrents ont démontré leur capacité à apprendre à partir de séquences. Toutefois, un grand nombre de problèmes se présentent dans l'utilisation des réseaux de neurones récurrents existants. Ils nécessitent alors de larges volumes de données (haute complexité des échantillons) et sont difficiles à entraîner. Nous introduisons un nouveau type de réseau de neurones récurrent que nous appelons KRU (Kronecker Recurrent Units, en français unités récurrentes Kronecker). Ce réseau utilise des matrices de Kronecker pour résoudre les problèmes rencontrés dans l'utilisation des réseaux récurrents existants. Les performances de notre méthode sont démontrées sur 7 applications allant de problèmes en vision par ordinateur à la reconnaissance vocale en passant par la modélisation de langue ou encore la modélisation musicale.

La plupart des algorithmes d'apprentissage machine sont formulés comme la minimisation de l'espérance empirique sur une collection finie d'échantillons. Dans cette thèse nous étudions également le problème de l'estimation efficace d'une moyenne pondérée sur de

larges volumes de données. Nous proposons une nouvelle structure de données appelée IST (Importance Sampling Tree, en français arbre d'importance d'échantillonage), laquelle permet d'estimer rapidement cette moyenne pondérée sans avoir à regarder tous les échantillons. Nous démontrons l'évaluation de notre structure de données dans la formation de réseaux neuronaux afin de trouver efficacement des échantillons informatifs.

Mots-clés: Inférer un plongement, apprentissage métrique, régularisateur orthonormé, régularisateur unitaire, ré-identification de personne, réseaux de neurones récurrents, produit de Kronecker, échantillonnage d'importance.

Acknowledgements

Firstly, I would like to express my gratitude to my enthusiastic adviser and mentor, François Fleuret. He gave me complete freedom of research topics, and was always willing and patient to hear any of my ideas. It was because of this academic freedom and encouragement that I was able work across diverse topics in machine learning. François taught me how good machine learning research is being done and never stops to amuse me by his ability to transform my vague intuitions into sound research ideas. I could not have asked for a better adviser for my Ph.D research.

I would like to thank my thesis jury members: Pascal Frossard, Olivier Bousquet, Moustapha Cissé, and Mathieu Salzmann for their insightful comments and encouragement.

I sincerely thank Swiss National Science Foundation (SNSF) for supporting my research through the grant, CRSII2-147693 – WILDTRACK. My gratitude also goes to Facebook AI Research (FAIR) and my mentor at FAIR, Moustapha Cissé, for giving me an internship opportunity. Internship at FAIR and the discussions with researchers there with diverse background enabled me to widen my research perspective.

Before coming to Idiap, I already had some research experience in machine learning as a masters student at IIT Delhi and as an intern at Microsoft. I am thankful to Manik Varma and SVN Vishwanathan for introducing me to the world of machine learning research.

The colleagues I have had at Idiap have been exceptional. James Newling, who showed how to really be focused on good research and stay out of distractions. Olivier Canévet, whose insights and engineering skills resulted in a research collaboration. I would like to thank all the system and the administrative team at Idiap for their support. I thank all the wonderful people that I met during my time at Idiap for their friendship and all the fun times. I cherish those coffee breaks, Friday beers and dinners. Switzerland is a playground for the people who like outdoor sports and the nature. I started skiing, mountain biking and climbing after coming here and the weekends were always fun with those activities. I thank all the people who were partners in it.

Finally, I thank my family, especially my parents for their patience, encouragement and freedom to pursue whatever interests me.

Martigny, January 2018 C. J.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

Machine learning is the science of automatically discovering meaningful patterns from data and using this information for prediction. In contrast to the traditional view of computing, in machine learning a human programmer does not write an exact specification of how patterns should be detected from data. Instead machine learning algorithms take inspiration from intelligent beings by learning a set of parameters to discover meaningful patterns based on a set of heuristic specifications and use these learned parameters for prediction. The core motivation behind such a paradigm for computing is that designing exact specification for many interesting pattern discovery problems is enormously difficult. Examples of such problems include image classification, automatic speech recognition, weather prediction, spam classification and analysis of genome data.

In this thesis we are interested in exploring a class of machine learning algorithms which learn a compact space where similar points in the data are nearby and dissimilar points are far apart. Thus this thesis fits in the paradigm of learning embeddings. Many of the machine learning algorithms are formulated as minimizing an empirical expectation over the data, including the models for learning embeddings. In this thesis, we also study the problem of efficiently estimating an empirical expectation over a large collection of data-samples. Before we delve into details let us discuss the core motivations for this dissertation.

1.1 Motivations

This thesis is inspired from three core motivating scenarios that arise in machine learning:

1.1.1 Learning from small data with large number of classes

Many state of the art machine learning algorithms such as neural networks or support vector machines require huge amounts of data to achieve good performance. With the availability of large scale data-sets such as Imagenet (Deng et al., 2009) performance of these methods has increased dramatically over the past decade (Krizhevsky et al., 2012b). However it is

unknown how this dramatic progress paves the way for learning from few examples or few-shot learning and reasoning about unseen categories using the regularities learned from the known categories (zero-shot learning). The reason for this is that the standard algorithms require all the categories to be known in advance during training and also require a large amount of examples per class to learn the category without over-fitting. Moreover, for these algorithms, the training and prediction time complexity as well as space complexity(model size) grows linearly (some times quadratically eg: 1vs1 classifiers) with the number of classes and thus they are limited to fairly small number of categories (on the order of 1000). Hence, these algorithms are ill-suited for the small sample learning problems where the number of examples per category is very small and the number of classes is unknown during training and is very large. This problem is prevalent in many important computer vision scenarios such as person re-identification for video surveillance and face verification for security applications.

A promising direction to pursue in order to address this scenario is to learn an embedding where the points from the same classes are together and dissimilar classes are far-apart and use this embedding to reason about unknown categories. In Chapter 3 we are interested in computationally efficient learning of an embedding under Mahalanobis distances. As an application we focus on person re-identification. We present a metric learning model called Weighted Approximate Rank Component Analysis (WARCA). WARCA optimizes the precision at top ranks by combining the Weighted Approximate Rank Pairwise(WARP) loss (Usunier et al., 2009; Weston et al., 2011; Lim and Lanckriet, 2014) with a regularizer that favors orthonormal linear mappings and avoids rank-deficient embeddings. Using this new regularizer allows us to efficiently exploit stochastic gradient descent, which results in an algorithm that scales gracefully to data-sets with large number of classes and training points. Also, we derive a kernel space WARCA which allows to take advantage of state-of-the-art features for re-identification when data-set size permits kernel computation. Benchmarks on recent and standard re-identification data-sets show that our method out-performs existing state-of-the-art techniques both in terms of accuracy and speed. We also provide experimental analysis to shed light on the properties of the regularizer we use, and how it improves performance.

1.1.2 Learning to embed sequences

Many natural signals like speech, language, video appear as sequences. Recently, Recurrent Neural Networks (RNNs) have emerged to be effective algorithms in learning embeddings from sequences. However there are several challenges in learning with recurrent neural networks.

In Chapter 4 we addresses two of these challenges with recurrent neural networks: (1) they are over-parametrized, and (2) the recurrence matrix is ill-conditioned. The former increases the sample complexity of learning and the training time. The latter causes the vanishing and exploding gradient problem. We present a flexible RNN model called Kronecker Recurrent Units (KRU). KRU achieves parameter efficiency in RNNs through a Kronecker factored recurrent matrix. It overcomes the ill-conditioning of the recurrent matrix by enforcing soft unitary

constraints on the factors. Thanks to the small dimensionality of these factors, maintaining these constraints is computationally efficient. Our experimental results on five standard data-sets reveal that KRU can reduce the number of parameters by three orders of magnitude in the recurrent weight matrix compared to the existing recurrent models, without trading the statistical performance. These results in particular show that while there are advantages in having a high dimensional recurrent space, the capacity of the recurrent part of the model can be dramatically reduced.

1.1.3 Efficient estimation of an empirical expectation

Efficiently estimating an empirical risk is a very important problem in machine learning as most of the machine learning methods are formulated as the minimization of an empirical expectation.

In Chapter 5 we present a tree-based data-structure called Importance Sampling Tree (IST) inspired by the Monte-Carlo Tree Search (Browne et al., 2012) that dynamically modulates an importance-based sampling to prioritize computation, while getting unbiased estimates of weighted sums. We apply this generic method to perform learning on very large training sets. The core idea is to reformulate the estimation of a score – whether a loss or a prediction estimate – as an empirical expectation, and to use a tree whose leaves carry the samples to focus efforts over the problematic "heavy weight" samples. We illustrate the potential of this approach on two problems: 1) To improve a multi-layer perceptron on $2D$ synthetic tasks with several million points and 2) To train a large-scale convolutional network on several millions deformations of the CIFAR data-set. In each case, we show how IST allows us to get better loss estimates.

1.2 Summary of contributions

The main contributions of this thesis are summarised as follows:

- We present a new scalable metric learning algorithm called WARCA to learn a Mahalanobis distance according to a weighted sum of the precisions at different ranks. This criterion in particular encompasses the Area Under the Curve (AUC) (uniform weighting) and the precisions at individual ranks (The Dirac weighting).
- We also present a non-linear WARCA by using kernel trick.
- We present a simple regularizer for preventing matrix rank degeneration in low rank matrix optimization by approximately enforcing the orthonormality constraint on the matrix being learned and demonstrate its effectiveness in learning algorithms.
- We present a new recurrent neural network model called KRU. KRU allows fine grained control over the number of parameters. Hence it permits a fine grained control over

Table 1.1: Notation

$\mathbb{R}$	The set of real numbers
$\mathbb{C}$	The set of complex numbers
Non bold capital letters indicate size or functions	
M	Number of training points
D	Dimension of training samples
Q	Number of classes
$\mathbb{R}^D$	The set of D dimensional vectors over $\mathbb{R}$
$\mathbb{R}^{D\times N}$	The set of $D \times N$ dimensional matrices over $\mathbb{R}$
Non-bold small letters indicate scalars or functions	
Bold capital letters indicate matrices	
Bold small letters indicate column vectors	
$\mathbf{x}_1, \ldots, \mathbf{x}_M$	A sequence of M vectors
$x_1, \ldots, x_M$	A sequence of M scalars
$\mathbb{1}_{\text{condition}}$	is equal to 1 if the condition is true, 0 otherwise

the computation and statistical performance of RNN. It is also robust to vanishing and exploding gradients.

- We introduce a new data-structure called IST which can efficiently sample points from a large collection of points according to a weight distribution over the collection and at the same time efficiently modulate the sampling weights for future sampling.

1.3 Thesis outline

This thesis is organized as follows. In Chapter 2 we review some basic concepts in machine learning. We describe our WARCA metric learning algorithm in Chapter 3 . In Chapter 4 we present Kronecker Recurrent Units (KRU). Chapter 5 presents IST data-structure and Chapter 6 concludes our work.

1.4 Notation

Table 1.1 summarizes the general notation that we use in this thesis. This notation is consistent across chapters and when needed a chapter specific notation is provided.

2 Background on Machine Learning

Contents

2.1 Introduction

This chapter aims at giving a concise introduction to machine learning. Most of the concepts discussed here are presented more thoroughly in the text book by Shalev-Shwartz and Ben-David (2014).

Let's imagine we want to design an algorithm to automatically label digital images with 1 or -1 depending upon whether the images contain a cat or not. If there is a cat our algorithm tag the image with 1 and -1 otherwise. One way to tackle this problem is to come up with an exact formal algorithmic specification of how a cat looks like in a digital photo and use this algorithm for labeling. However designing such an exact specification would be an enormously difficult task because of the variations among different cats and the photographic nuisances such as illumination, view-point changes and occlusion. Another way to tackle this problem is to first get a set of images with cat (1) or not cat (-1) labels and design an algorithm to learn what distinguishes an image with cats from the rest. Once we have learned this information we can then use it for image label prediction. Machine learning takes this approach to solve problems for which a formal specification is difficult to derive. Let's formalize a simple model for machine learning problems using our photo tagger as an example. We define the following notation.

- $\mathcal{X}$ The domain set. All natural images in the photo tagging problem.
- $\mathcal{Y}$ The label set (1 or -1 in the example)
- $\mathcal{D}$ Data generating distribution. That is, the probability distribution over $\mathcal{X}$. This distribution is assumed to be unknown. In the example, it is the probability distribution of all natural images in the set of all images.
- $\mathcal{S} = \{(\mathbf{x}_1, y_1), .., (\mathbf{x}_M, y_M)\}$ A set of M training data-label pairs in $\mathcal{X} \times \mathcal{Y}$ (A set of M image label pairs in the example).
- $f : \mathcal{X} \to \mathcal{Y}$ The correct labeling function which is unknown. (Function which maps the photo to the label). The generalization error (which we define later) of the optimal classifier is assumed to be 0 for simplicity.
- $h : \mathcal{X} \to \mathcal{Y}$ The prediction function or the hypothesis that is used to predict the label of data-points from the domain set. During learning, the machine learning algorithm evaluates the performance of many different hypothesis from a set of hypotheses and outputs the best hypothesis it possibly could from the set according to some performance measure. The performance measure quantifies how well the hypothesis is doing at predicting the label.
- $\mathcal{H} = \{h : \mathcal{X} \to \mathcal{Y}\}$ Hypothesis class or the domain of all functions that the learning algorithm considers for choosing a hypothesis.

2.2 Empirical Risk Minimization (ERM) principle

In order to select a hypothesis from the hypothesis class $\mathcal{H}$ we need to define a performance measure quantifying how well the chosen hypothesis is doing in terms of correctly predicting the label. This performance measure is often called as loss function in machine learning. In our photo labeling example it is the probability that a randomly sampled image from the data generating distribution $\mathcal{D}$ is incorrectly labeled by the current hypothesis h. This defines the generalization error or the expected risk $R(h)$ under a hypothesis h:

$$R(h) = \mathop{\mathbb{P}}_{\mathbf{x}\sim\mathcal{D}}(f(\mathbf{x}) \neq h(\mathbf{x})) = \mathop{\mathbb{E}}_{x\sim\mathcal{D}}[1_{h(\mathbf{x})\neq f(\mathbf{x})}] = \int\limits_{\mathbf{x}\sim\mathcal{D}} 1_{h(\mathbf{x})\neq f(\mathbf{x})} d\mathbf{x}. \tag{2.1}$$

Given a set of training data-label pairs $\mathcal{S}$, where the training data is sampled from an unknown probability distribution $\mathcal{D}$ and labeled by an unknown labeling function f, the goal of a machine learning algorithm is to output a prediction function from the hypothesis class $\mathcal{H}$, such that it minimizes the generalization error defined in 2.1. That is, ideally, the algorithm would like to minimize the expected risk by choosing the best hypothesis it possibly could from the hypothesis class $\mathcal{H}$.

However since the distribution $\mathcal{D}$ and the labeling function f is unknown, directly minimizing the expected risk is impossible. Instead what an algorithm can do is to minimize the training error or the empirical risk $\hat{R}(h)$ which is defined as follows:

$$\hat{R}(h) = \frac{1}{M}\sum_{m=1}^{M} 1_{h(\mathbf{x}_m)\neq f(\mathbf{x}_m)}. \tag{2.2}$$

Machine learning algorithms minimize the empirical risk as a tractable proxy for the expected risk. This learning principle of finding a hypothesis by minimizing the empirical risk is called Empirical Risk Minimization (ERM). That is, given a training set $\mathcal{S}$, a learning algorithm using the ERM principle does the following:

$$h_{\mathcal{S}} = \operatorname*{argmin}_{h\in\mathcal{H}} \frac{1}{M}\sum_{m=1}^{M} 1_{h(\mathbf{x}_m)\neq f(\mathbf{x}_m)}. \tag{2.3}$$

There are two errors that come up when we design learning algorithms using the ERM principle: 1) approximation error and 2) estimation error.The approximation error (ϵ_{app}) is the minimum generalization error achievable by a hypothesis from the considered hypothesis class:

$$\epsilon_{app} = \min_{h\in\mathcal{H}} R(h). \tag{2.4}$$

The approximation error occurs because the hypothesis class may not contain the true labeling

function f.

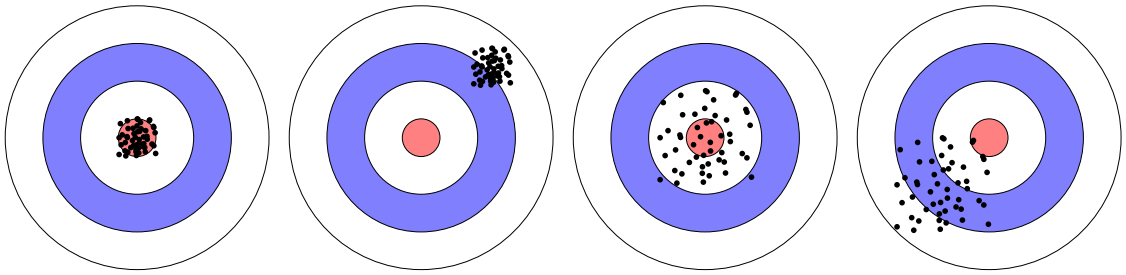
The estimation error (ϵ_{est}) is the difference between the generalization error achieved by the hypothesis selected by the learning algorithm over the training set $\mathcal{S}$ and the approximation error:

$$\epsilon_{est} = R(h_{\mathcal{S}}) - \epsilon_{app}. \tag{2.5}$$

Estimation error happens because the empirical risk is just a proxy for the expected risk and so the algorithm minimizing the empirical risk is just a proxy for the algorithm minimizing the expected risk.

2.2.1 Bias-variance trade-off

In order to generalize well,the learning algorithm should have low approximation error and low estimation error. When the approximation error is high, we say that the algorithm is under-fitting or has a high bias towards a particular hypothesis across different training sets. High bias arises when the considered hypothesis class is not flexible enough to capture the relevant information in the data. This can be avoided by considering a rich hypothesis class. Figure 2.1 illustrates bias and variance in a dart throwing game.



(a) Low bias, low variance **(b)** High bias, low variance **(c)** Low bias, high variance **(d)** High bias, high variance

Figure 2.1: Bias and variance illustration in a dart throwing game.

However when we increase the complexity of the hypothesis class it may lead to high estimation error or high variance in the hypothesis selected by the algorithm on different training sets. High variance arises because the considered hypothesis class is too flexible and it fits to the noise in the data but not to the correct information. In this case the algorithm will fit very well the training data but fails to generalize to the data outside of the training set. This problem of over-fitting on the training set can be avoided by controlling the search space of hypotheses within the hypothesis class by exploiting some prior knowledge about the data. This strategy is often called ERM with inductive bias. However high inductive bias will lead to under-fitting. So the goal of any learning algorithm using ERM principle is to obtain the right trade-off between the approximation error and the estimation error or the right trade-off between the bias and

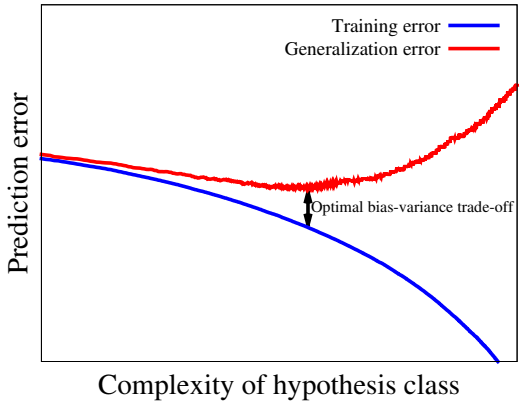


Figure 2.2: Illustration of bias and variance in machine learning algorithms. As the complexity of the hypothesis class gets high, the approximation error decreases but the estimation error increases. That is, the algorithm starts fitting to the noise in the training data which causes the training error to go down. And since the algorithm is explaining the noise in the training data its error outside the training data (generalization error) goes up. The goal of any machine learning algorithm is to obtain the optimal trade-off between the approximation error and the estimation error or the bias and variance.

variance. This trade-off is achieved in practice by using cross-validation (Shalev-Shwartz and Ben-David, 2014). Figure 2.2 illustrates bias and variance in machine learning algorithms.

2.3 Perceptron learning algorithm

Let's consider our photo labeling example and imagine each image being represented as a D dimensional vector. Now consider a training set $\mathcal{S}$ of image / label pairs:

$$\mathcal{S} = (\mathbf{x}_m, y_m) \in \mathbb{R}^D \times \{-1, 1\},\ m = 1, \ldots, M. \tag{2.6}$$

Let's consider the hypothesis class $\mathcal{H}$ to be the set of all linear classifiers in $\mathbb{R}^D$ passing through the origin. Linear classifiers are sign thresholded linear functions. An illustration of a linear classifier in $\mathbb{R}^2$ is shown in Figure 2.3. The prediction function of a linear classifier is of the form:

$$h(\mathbf{x}; \mathbf{w}) = \text{sign}(\mathbf{w}^T \mathbf{x}) \tag{2.7}$$

where $\mathbf{w} \in \mathbb{R}^D$ is the parameter of the linear classifier and $h(\mathbf{x}; \mathbf{w}) \in \{-1, 1\}$ is the response of the linear classifier on the input $\mathbf{x}$. Different settings of $\mathbf{w}$ give different functions in the hypothesis class. Given $\mathcal{S}$, our goal is to find a setting of the parameter $\mathbf{w}$ such that it minimizes the

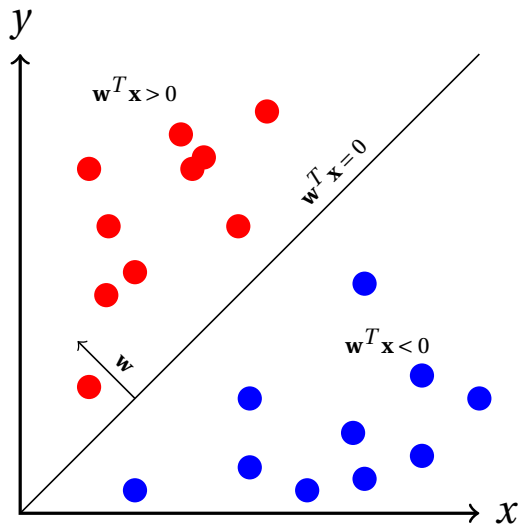


Figure 2.3: Illustration of a linear classifier passing through the origin in $\mathbb{R}^2$, separating the red dots from the blue dots. The red dots are labeled 1 and the blue dots are labeled -1. The line separating the two classes ($\mathbf{w}^T\mathbf{x} = 0$) is called as the decision boundary and $\mathbf{w}$ which is the normal to the line $\mathbf{w}^T\mathbf{x} = 0$ is the parameter of the linear classifier.

empirical risk. The corresponding ERM problem can be written as:

$$\min_{\mathbf{w}} \frac{1}{M} \sum_{m=1}^{M} 1_{y_m \text{sign}(\mathbf{w}^T\mathbf{x}_m) \leq 0}. \tag{2.8}$$

Unfortunately the optimization problem in Equation 2.8 is difficult to solve using standard numerical methods because the discrete 0-1 step loss function used as the performance measure is not differentiable and non-convex. In order to make ERM practical the Perceptron algorithm approximates the step function with a convex rectified linear function:

$$\min_{\mathbf{w}} \frac{1}{M} \sum_{m=1}^{M} \max(0, -y_m(\mathbf{w}^T\mathbf{x}_m)). \tag{2.9}$$

This loss function is differentiable everywhere except at 0. The Perceptron learning algorithm is shown in Algorithm 1. It minimizes the empirical risk 2.9 by stochastic sampling of the training points one at a time. If the data is linearly separable by a margin and have bounded l_2-norm then it can be shown that the Perceptron converges to 0 empirical risk in a finite number of iterations.

Theorem 1. *If the data is linearly separable with a margin $\gamma > 0$, that is, there exists $\mathbf{w}^*$: $y_m(\mathbf{w}^{*T}\mathbf{x}^*_m) \geq \gamma$ and $\|\mathbf{x}_m\| \leq R$ for all $m \in \{1,\ldots,M\}$ then the Perceptron converges to 0 empirical risk in $t \leq \frac{R^2\|\mathbf{w}^*\|^2}{\gamma^2}$ steps (Novikoff, 1962).*

Algorithm 1 Perceptron algorithm

Input: Training set $\mathcal{S} = (\mathbf{x}_m, y_m) \in \mathbb{R}^D \times \{-1, 1\},\ m = 1, \ldots, M$, Number of iterations T.

$\mathbf{w}_0 = \{0\}^D, b = 0$
$t = 0$
while $t < T$ **do**
  Sample $(\mathbf{x}_r, y_r)$ uniformly at random from $\mathcal{S}$
  if $y_t(\mathbf{w}_t^T \mathbf{x}_r) \leq 0$ **then**
    $\mathbf{w}_{t+1} = \mathbf{w}_t + y_r \mathbf{x}_r$
  end if
  t = t + 1
end while

Output: $\mathbf{w}_t$

Table 2.1: Convex approximations of 0-1 loss function ($1_{y(\mathbf{w}^T\mathbf{x}) \leq 0}$).

$\max(0, 1 - y(\mathbf{w}^T\mathbf{x}))$	Hinge loss
$\log(1 + e^{-y(\mathbf{w}^T\mathbf{x})})$	Logistic loss
$e^{-y(\mathbf{w}^T\mathbf{x})}$	Exponential loss
$(y - \mathbf{w}^T\mathbf{x})^2$	Least squares loss

Proof. The proof is presented in Appendix A.1. □

2.3.1 Remarks

There are many convex approximations to the 0-1 loss function. A few of them are listed in Table 2.1. Depending upon the loss function we choose, we get a class of learning algorithms which is known under different names. Let us denote all these loss functions by a general function: $\mathcal{L}(\mathbf{w}^T\mathbf{x}, y)$.

2.4 Stochastic gradient descent (SGD)

The Perceptron learning algorithm that we saw in the previous section can be generalized to arbitrary loss functions. Consider the ERM problem under the loss function $\mathcal{L}(\mathbf{w}^T\mathbf{x}, y)$.

$$\min_{\mathbf{w},b} \frac{1}{M} \sum_{m=1}^{M} \mathcal{L}(\mathbf{w}^T\mathbf{x}_m, y_m). \tag{2.10}$$

If the loss function is convex and differentiable then we can use gradient descent to minimize the above empirical risk. In gradient descent we start from an initial value and at each step t we move along the negative direction of the gradient with a learning rate $\eta > 0$:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\eta}{M} \sum_{m=1}^{M} \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}_t^T\mathbf{x}_m, y_m)\big|_{\mathbf{w}=\mathbf{w}_t} \mathbf{x}_m. \tag{2.11}$$

It can be proved that a sequence of gradient descent steps will converge to the minima for convex-Lipschitz functions (Shalev-Shwartz and Ben-David, 2014). When the loss function is convex but not differentiable we can use a sub-gradient from the sub-differential set at the non-differentiable points.

However when the size of the training data M is very large, gradient descent updates are expensive $\Theta(M)$. So in order to save the computation we can sample a point or a subset of points uniformly at random from the training data and compute the gradient on this set and do an update. Since the gradient is a linear function, by the linearity of expectation, the gradient on the uniformly sampled data points in expectation will be equal to the gradient of the ERM problem. This algorithm where we use a point or a subset of points for the parameter update is called stochastic gradient descent (SGD) and it can be shown that SGD converges to the minima in expectation (Shalev-Shwartz and Ben-David, 2014). SGD has become the *de-facto* algorithm for ERM problems. It enjoys many nice properties suitable for ERM problems at scale (Bousquet and Bottou, 2008) and it is simple, fast and easy to implement. The SGD algorithm for ERM is illustrated in Algorithm 2.

Algorithm 2 Stochastic gradient descent for ERM

Input: Training set $\mathcal{S} = (\mathbf{x}_m, y_m) \in \mathbb{R}^D \times \{-1, 1\},\ m = 1, \ldots, M$, Loss function $\mathcal{L}(.,.)$, Learning rate $\eta > 0$, Number of iterations T.

1: $\mathbf{w}_0 = \{0\}^D$
2: $t = 0$
3: **while** $t < T$ **do**
4: Sample $(\mathbf{x}_r, y_r)$ uniformly at random from $\mathcal{S}$
5: $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}^T \mathbf{x}_r, y_r)\big|_{\mathbf{w}=\mathbf{w}_t}$
6: t = t + 1
7: **end while**

Output: $\mathbf{w}_t$

2.5 Representer theorem and non-linear learning algorithms

When the data is linearly separable but noisy the linear classifiers using the ERM rule, such as the Perceptron algorithm over-fits. As we discussed earlier over-fitting can be avoided by controlling the complexity of the hypothesis class by introducing some inductive bias about the data. This achieved by a learning paradigm called Regularized Loss Minimization (RLM) which jointly minimizes the empirical risk $\hat{R}(h_{\mathbf{w}})$ and a regularization function $\Omega(\mathbf{w})$:

$$\min_{\mathbf{w}} \ \Omega(\mathbf{w}) + \frac{1}{M} \sum_{m=1}^{M} \mathcal{L}(\mathbf{w}^T \mathbf{x}_m, y_m). \tag{2.12}$$

A simple regularizer for the linear classifiers is the squared l_2 norm of $\mathbf{w}$:

$$\Omega(\mathbf{w}) = \lambda \|\mathbf{w}\|^2, \tag{2.13}$$

where $\lambda \geq 0$ is a scalar that controls the strength of the regularization. This gives us the following RLM problem:

$$\min_{\mathbf{w}} \ \lambda \|\mathbf{w}\|^2 + \frac{1}{M} \sum_{m=1}^{M} \mathcal{L}(\mathbf{w}^T \mathbf{x}_m, y_m). \tag{2.14}$$

When the data is not linearly separable, the linear classifiers under-fits. In order to tackle this problem, we can map the data to a high-dimensional feature space using a non-linear transform. In the high-dimensional space the data might be linearly separable and there we can learn a linear classifier. Often, this high-dimensional feature space is very large or infinite and an explicit access to that space is computationally prohibitive. If we can access that space through the dot products between the points in that space and the loss function $\mathcal{L}(z, y))$ is convex in z, then the Representer theorem enables us to derive powerful non-linear learning algorithms:

Theorem 2. *The optimization problem in 2.14 has a minimizer of the form:*

$$\mathbf{w}^* = \sum_{m=1}^{M} a_m x_m. \tag{2.15}$$

Proof. This is an intuitive proof. Please refer to Schölkopf and Smola (2002) for a general statement of the theorem and a rigorous proof.

$$G_\lambda(\mathbf{w}) = \lambda \|\mathbf{w}\|^2 + \frac{1}{M} \sum_{m=1}^{M} \mathcal{L}(\mathbf{w}^T x_m, y_m). \tag{2.16}$$

Taking the gradient of $G_\lambda(\mathbf{w})$ with respect to $\mathbf{w}$:

$$\nabla_{\mathbf{w}} G_\lambda(\mathbf{w}) = 2\lambda \mathbf{w} + \frac{1}{M} \sum_{m=1}^{M} \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}^T \mathbf{x}_m, y_m) \mathbf{x}_m. \tag{2.17}$$

The gradient is 0 at all stationary points, which implies:

$$\mathbf{w} = -\frac{1}{2\lambda M} \sum_{m=1}^{M} \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}^T \mathbf{x}_m, y_m) \mathbf{x}_m, \tag{2.18}$$

$$= \sum_{m=1}^{M} a_m \mathbf{x}_m, \tag{2.19}$$

where

$$a_m = -\frac{1}{2\lambda M} \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}^T \mathbf{x}_m, y_m). \tag{2.20}$$

□

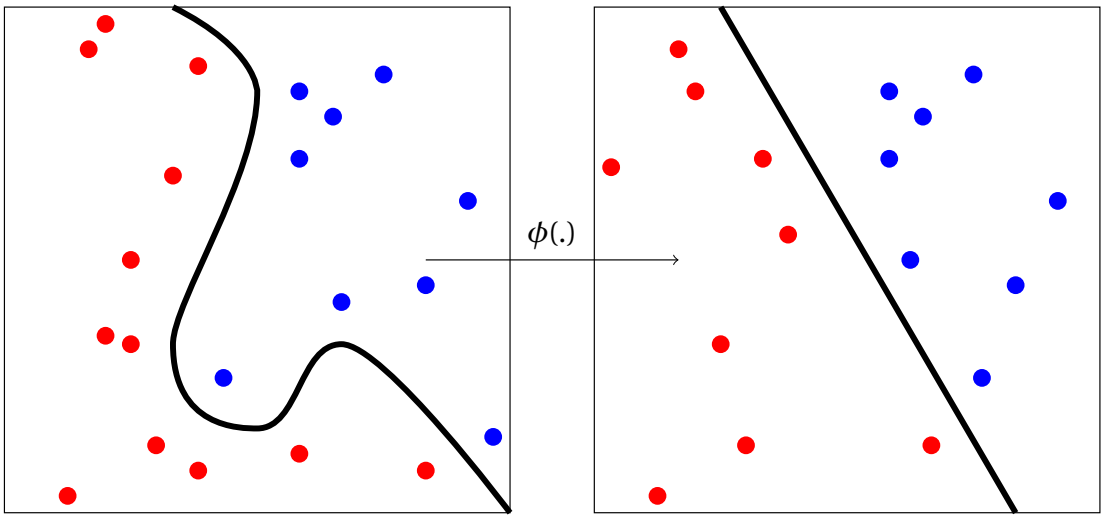


Figure 2.4: Illustration of kernel trick. The data-set is not linearly separable in the input space and it is mapped to a high-dimensional feature space through mapping function $\phi(.)$ where the data-set is linearly separable. The Representer theorem allows us to learn linear classifiers in this high dimensional space without accessing it explicitly.

Now we can write the prediction function as:

$$h(\mathbf{x};\mathbf{a}) = \sum_{m=1}^{M} a_m \langle \mathbf{x}_i, \mathbf{x} \rangle,$$

where $\mathbf{a} = [a_1, ..., a_M]^T$ is the parameter vector in this new representation. Substituting the expression 2.19 in 2.21 gives a regularized risk in $\mathbf{a}$:

$$G_\lambda(\mathbf{a}) = \lambda \sum_{i=1}^{M} \sum_{j=1}^{M} a_i \langle \mathbf{x}_i, \mathbf{x}_j \rangle a_j + \frac{1}{M} \sum_{i=1}^{M} \mathcal{L}(\sum_{j=1}^{M} a_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle, y_i). \tag{2.21}$$

That is, we can write the entire RLM problem in terms of dot products between data points. This is a powerful paradigm because we can replace the dot product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ by $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$. Where $\phi(.)$ is a non-linear mapping function to a high dimensional space, where the data is linearly separable. If we could compute $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$ through a simple kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ we do not need an explicit access to $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$. This technique is often called the "kernel trick". An illustration of the kernel trick is shown in Figure 2.4. For example the Gaussian RBF kernel whose feature expansion is infinite dimensional can be computed efficiently as $k(\mathbf{x}_i, \mathbf{x}_j) = e^{\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\tau^2}}$. The Gaussian RBF kernel is one of the most popular kernel functions and works well on practical problems. It has been shown to be an universal approximator, that is, a kernel regularized risk minimizer using a Gaussian RBF kernel can learn any complex functions given a sufficient number of training points.

Now we can write the regularized risk in terms of the kernel matrix: $\mathbf{K} = [k(\mathbf{x}_i, \mathbf{x}_j)]_{i,j=1}^{M} \in \mathbb{R}^{M \times M}$.

$$G_\lambda(\mathbf{a}) = \lambda \mathbf{a}^T K \mathbf{a} + \frac{1}{M} \sum_{m=1}^{M} \mathcal{L}(\mathbf{a}^T \mathbf{k}_m, y_m). \tag{2.22}$$

where $\mathbf{k}_m$ is the m^{th} column in the kernel matrix $\mathbf{K}$ and thus the RLM problem becomes:

$$\min_{\mathbf{a}} \ (\lambda \mathbf{a}^T \mathbf{K} \mathbf{a} + \frac{1}{M} \sum_{m=1}^{M} \mathcal{L}(\mathbf{a}^T \mathbf{k}_m, y_m)). \tag{2.23}$$

In Algorithm 3 we give a simple stochastic gradient descent algorithm for solving the Kernel RLM (KRLM) problem in Equation 2.23.

Algorithm 3 Stochastic gradient descent for KRLM

Input: Training set $\mathcal{S} = (\mathbf{x}_m, y_m) \in \mathbb{R}^D \times \{-1, 1\},\ m = 1, \ldots, M$, Kernel function $k(.,.)$, Loss function $\mathcal{L}(.,.)$, Learning rate $\eta > 0$, Number of iterations T.

1: $\mathbf{a}_0 = \{0\}^M$
2: $\mathbf{K} = [k(\mathbf{x}_i, \mathbf{x}_j)]_{i,j=1}^M \in \mathbb{R}^{M \times M}$
3: $t = 0$
4: **while** $t < T$ **do**
5: Sample a column $\mathbf{k}_r$ and it's label y_r uniformly at random.
6: $\mathbf{a}_{t+1} = \mathbf{a}_t - \eta(2\lambda a_{tr} \mathbf{k}_r + \nabla_{\mathbf{a}} \mathcal{L}(\mathbf{a}^T \mathbf{k}_r, y_r)\big|_{\mathbf{a}=\mathbf{a}_t})$
7: t = t + 1
8: **end while**

Output: $\mathbf{a}_t$

2.6 Kernel learning and neural networks

In the previous section we have seen that the Representer theorem allows us to derive powerful non-linear learning algorithms. But there are a few practical issues in applying KRLM when the number of data-points M is very large relative to its dimension D, that is $M \gg D$. This type of problems are very common in machine learning and examples include image classification (photo tagger), speech recognition and natural language processing. The computational complexity of KRLM grows as $\mathcal{O}(M^{2+\epsilon})$ where $0 \leq \epsilon \leq 1$. Also the computational complexity of prediction is $\mathcal{O}(M)$ and moreover, we have to keep the training-points for prediction which increases the memory footprint. On the other hand we have computationally efficient linear algorithms, where the training computational complexity is $\mathcal{O}(M)$ and the prediction complexity is $\mathcal{O}(1)$ but they are not as powerful as KRLM.

At this point we can ask a question: Can we have the best out of both worlds for the problems where $M \gg D$? That is, can we have a learning algorithm which is powerful like a kernel

method and at the same time computationally efficient? A way to approach this question is to start with a kernel function that works well on real world problems, approximate this kernel function by an easy to compute explicit feature map, and then train a linear classifier in that approximate explicit feature map. That is consider a kernel function:

$$k(\mathbf{x},\mathbf{y}) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle. \tag{2.24}$$

We approximate it by a finite dimensional approximate feature map $\hat{\phi}(.)$:

$$\hat{k}(\mathbf{x},\mathbf{y}) = \hat{\phi}(\mathbf{x})^T \hat{\phi}(\mathbf{y}). \tag{2.25}$$

And learn a linear classifier in $\hat{\phi}(.)$:

$$h(\mathbf{x}) = \mathbf{w}^T \hat{\phi}(\mathbf{x}). \tag{2.26}$$

Where $\mathbf{w}, \hat{\phi}(\mathbf{x}) \in \mathbb{R}^N : D < N << M$.

2.6.1 Random feature map for Gaussian RBF kernel

We have discussed that KRLM problems with Gaussian RBF kernels can learn complex non-linear functions. If we could efficiently get a reasonable approximation of a Gaussian RBF kernel in low dimensional space, then we can learn complex non-linear functions with a linear function in that space. Let's derive an explicit feature map for the Gaussian RBF kernel. Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^D$ be two vectors. Then the Gaussian RBF kernel between $\mathbf{x}$ and $\mathbf{y}$ is given by:

$$k(\mathbf{x},\mathbf{y}) = e^{\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\tau^2}}. \tag{2.27}$$

Let's write this kernel function in the Fourier basis:

$$k(\mathbf{x},\mathbf{y}) = k(\mathbf{x}-\mathbf{y}) = \int_{\mathbb{R}^D} p(\mathbf{u}) e^{i\mathbf{u}^T(\mathbf{x}-\mathbf{y})} d\mathbf{u}, \tag{2.28}$$

where $p(\mathbf{u})$ is the inverse Fourier transform of $k(\mathbf{x}-\mathbf{y})$. Since $k(\mathbf{x},\mathbf{y}) \in [0,1]$ by Bochner's theorem p is a probability density. In the case of the Gaussian RBF kernel, p is a Gaussian density. This implies that we can do Monte-carlo sampling from p to approximate the kernel:

$$k(\mathbf{x}-\mathbf{y}) = \int_{\mathbb{R}^D} p(\mathbf{u}) e^{i\mathbf{u}^T(\mathbf{x}-\mathbf{y})} d\mathbf{u} = \mathop{\mathbb{E}}_{\mathbf{u}\sim p}[e^{i\mathbf{u}^T(\mathbf{x}-\mathbf{y})}] \approx \frac{1}{N}\sum_{n=1}^{N} e^{i\mathbf{u}_n^T(\mathbf{x}-\mathbf{y})} : \mathbf{u}_i \sim p. \tag{2.29}$$

That is:

$$k(\mathbf{x}-\mathbf{y}) \approx \frac{1}{N}\sum_{n=1}^{N} e^{i\mathbf{u}_n^T(\mathbf{x}-\mathbf{y})} : \mathbf{u}_i \sim p = \hat{\phi}(\mathbf{U}^T\mathbf{x})^H \hat{\phi}(\mathbf{U}^T\mathbf{y}). \tag{2.30}$$

where $\mathbf{U} \in \mathbb{R}^{D\times N}$ is a random matrix whose entries are sampled i.i.d from p, and $\hat{\phi}(\mathbf{z})$ is the

point-wise projection on to the unit circle in the complex plane normalized by the square-root of the length of the vector. That is:

$$\hat{\phi}(\mathbf{z}) = \frac{[e^{iz_1}, ..., e^{iz_N}]^H}{\sqrt{N}}. \tag{2.31}$$

So we have an N dimensional approximation of the Gaussian RBF kernel and now we can learn a linear classifier in this N dimensional space to get non-linear functions. As $N \to \infty$ we recover the exact Gaussian RBF kernel. Usually many problems of practical interest require only a few feature expansions which make the algorithm computationally efficient. This simple paradigm of approximating kernels works with any translation invariant kernel ($k(\mathbf{x}, \mathbf{y}) = k(\mathbf{x} - \mathbf{y})$) as long as we can sample efficiently from its spectral density. This technique of approximating shift invariant kernels using random features was shown by Rahimi and Recht (2008).

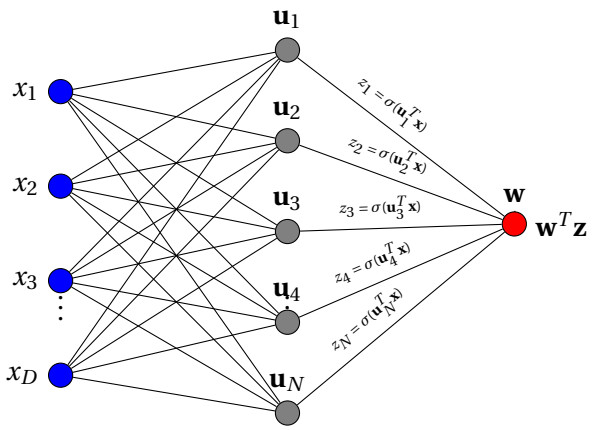


Figure 2.5: A diagrammatic illustration of non-linear learning algorithm viewed through the lens of kernel approximation. The input point whose component is represented as a blue node is connected to each of the N grey nodes where each grey node represents a D dimensional vector sampled i.i.d from the spectral density of the kernel we want to approximate. Each of these grey nodes receives the input point $\mathbf{x}$, computes its projection and passes it through a non-linear function $\sigma(.)$ to get an N dimensional approximate random feature map under the kernel. Further, each of these N gray nodes is connected to a red node representing a linear function in $\mathbb{R}^N$.

2.6.2 Neural networks

We have seen a simple paradigm for approximating translation invariant kernels with random features. The steps involved in this paradigm are illustrated in Figure 2.5. However, when we are just using random projections to approximate kernels, we are wasting the modeling power because these random features are sampled independently of the data. Instead of just

using random feature maps to approximate the kernels we can consider these as parameters and learn them together with the linear function from the data. This has the advantage that, since we are adapting the random parameters to the data, we might just need fewer dimension expansion to learn a good non-linear function, thereby improving the prediction time. This also comes with several disadvantages: 1) the optimization problem is now non-convex and requires some hyper-parameter tuning to make it work and 2) It increases the sample complexity of the learning algorithm as we have more parameters.

Now we can take these ideas further. Instead of learning just one layer of features we learn a hierarchy of features using function composition as illustrated in Figure 2.6. The reasoning behind this strategy is that certain functions can be represented compactly if we have a hierarchical architecture. This class of learning algorithms which learn a hierarchy of non-linear feature maps are called Neural Networks. Each layer of these feature hierarchies is called hidden layer and each node is called a Neuron. The input nodes are called input layer and the output nodes output layer. These algorithms aim at learning the kernel along with the linear classifier from the data. Historically, neural networks were developed before kernel methods. The equivalence between certain kernel methods and a one hidden layer neural network networks with infinite number of random neurons was first shown in Neal (1996).

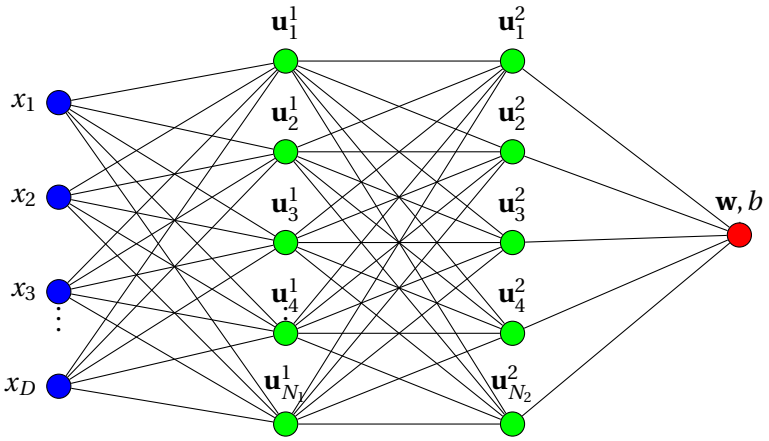


Figure 2.6: A neural network learns a hierarchy of features together with the linear function to capture regularities in data. The core intuition for such a method is that certain functions can be compactly represented by such a function composition. These learning algorithms often work well in practice when combined with inductive biases in the data. Examples of such neural networks with inductive biases include convolutional neural networks for visual data and recurrent neural networks for sequence data.

With the advent of very large data-sets, progress in computer hardware and clever algorithms, neural networks have emerged as effective algorithms for problems in machine learning. Recently they achieved breakthrough results in image classification (Krizhevsky et al., 2012c; He et al., 2016), speech recognition (Hinton et al., 2012), machine translation (Bahdanau et al.,

2014), game playing (Mnih et al., 2013; Silver et al., 2017), image synthesis (Goodfellow et al., 2014) and speech synthesis (Oord et al., 2016).

Neural networks pose interesting research challenges. They have shown to be vulnerable to small perturbations in the data-set and this raises questions regarding their robustness (Szegedy et al., 2013; Fawzi, 2016; Cisse et al., 2017a). The classical learning theory tools using uniform convergence is proved to be inadequate in explaining the generalization in neural networks (Zhang et al., 2016). Algorithm specific analysis (Bousquet and Elisseeff, 2002) appears to be more suitable here (Hardt et al., 2015). Another interesting research avenue is about better understanding the properties of the functions realized by the neural networks in modeling natural signals so that we can analytically construct those functions (Bruna and Mallat, 2013). Moreover, one of the strong reasons why neural networks work is because of the inductive biases built into the algorithm, such as convolutions for natural images. It calls for a better understanding of the data so that we can incorporate the knowledge (inductive bias) about the data into the neural networks. This will reduce the model complexity of the neural networks and make them generalize better.

Machine learning and neural networks also pose significant and interesting engineering challenges. They disrupts traditional engineering practices which are based on abstracting specifications (Sculley et al., 2014; Bottou, 2015). Abstraction leaks are common in machine learning systems because the specifications used to build these systems are weak and not exact, unlike the traditional software systems. Recently there have been attempts to build systems which abstract away learning algorithms and learning systems. Examples of such systems include Theano (Bergstra et al., 2011), Torch (Collobert et al., 2011) Tensorflow (Abadi et al., 2016) and Pytorch (Paszke et al., 2017). At low level engineering, the major challenge associated with neural networks include efficient implementations of primitive operations across a wide range of hardware ranging from cell phones to super-computers by exploiting fast changing advances in the hardware.

2.7 Discussion

We use many concepts discussed here in later chapters to derive learning algorithms. In Chapter 3 we use the ERM principle to formulate our WARCA metric learning algorithm and exploit a weak form of the Representer theorem to derive a kernelized WARCA. In Chapter 4 we address a few issues of recurrent neural networks, which are neural networks for learning from sequence data. We design suitable regularizers for biasing our learning algorithms to avoid over-fitting. All our RLM problems are optimized using SGD or its variants. Chapter 5 is aimed at efficiently estimating an empirical expectation, such as the empirical risk.

3 Weighted Approximate Rank Component Analysis

Contents

In this chapter we present the Weighted Approximate Rank Component Analysis (WARCA) algorithm for metric learning. WARCA is scalable, robust, non-linear and ideal for learning from data with large number of classes where the number of samples per class is small. WARCA is based on the following publication:

> Cijo Jose and François Fleuret. Scalable metric learning via weighted approximate rank component analysis. In *European Conference on Computer Vision*, pages 875–890. Springer, 2016

3.1 Introduction

Our core motivation for this chapter is small data learning with large number of classes and also reasoning about classes which were not available during training using the regularities learned from the data (zero-shot learning). The line of research we follow to tackle this problem is metric learning. Metric learning methods aim at learning a parametrized distance embedding from a labeled set of samples, so that under the learned embedding, samples with the same labels are nearby and samples with different labels are far apart (Weinberger and Saul, 2009). Many fundamental questions in computer vision such as "How to compare two images? and for what information?" boil down to this problem. Among them, person re-identification is the problem of recognizing individuals at different physical locations and times, on images captured by different devices.

It is a challenging problem which recently received a lot of attention because of its importance in various application domains such as video surveillance, biometrics, and behavior analysis (Gong et al., 2014).

The performance of person re-identification systems relies mainly on the image feature representation and the distance measure used to compare them. Hence the research in the field has focused either on designing features (Cheng et al., 2011; Liao et al., 2015) or on learning a distance function from a labeled set of images (Mignon and Jurie, 2012; Köstinger et al., 2012; Li et al., 2013; Xiong et al., 2014; Liao et al., 2015; Liao and Li, 2015).

It is difficult to analytically design features that are invariant to the various non-linear transformations that an image undergoes such as illumination, viewpoint, pose changes, and occlusion. Furthermore, even if such features were provided, the standard Euclidean metric would not be adequate as it does not take into account dependencies on the feature representation. This motivates the use of metric learning for person re-identification. As a consequence many metric learning methods have been developed and applied to computer vision problems and person re-identification is one of them.

Re-identification models are commonly evaluated by the cumulative match characteristic (CMC) curve (Köstinger et al., 2012). This measure indicates how the matching performance of the algorithm improves as the number of returned images increases. Given a matching algorithm and a labeled test set, each image is compared against all the others, and the position of the first correct match is recorded. The CMC curve indicates for each rank the fraction of test samples which had that rank or better. A perfect CMC curve would reach the value 1 for rank #1, that is the best match is always of the correct identity.

In this chapter we are interested in learning a Mahalanobis distance by minimizing a weighted rank loss such that the precision at the top rank positions of the CMC curve is maximized. When learning the metric, we directly learn the low-rank projection matrix instead of the PSD matrix because of computational efficiency and scalability to high dimensional data-sets (see § 3.3.1). But naively learning the low-rank projection matrix suffers from the problem of

matrix rank degeneration and non-isolated minima (Lim and Lanckriet, 2014). We address this problem by using a simple regularizer which approximately enforces the orthonormality of the learned matrix efficiently (see § 3.3.2). We extend the WARP loss (Usunier et al., 2009; Weston et al., 2011; Lim and Lanckriet, 2014) and combine it with our approximate orthonormal regularizer to derive a metric learning algorithm which approximately minimizes a weighted rank loss efficiently using stochastic gradient descent (see § 3.3.3).

We extend our model to kernel space to handle distance measures which are more natural for the features we are dealing with (see § 3.3.4). We also show that in kernel space SGD can be carried out more efficiently by using preconditioning (Chapelle, 2007; Mignon and Jurie, 2012).

We validate our approach on nine person re-identification data-sets: Market-1501 (Zheng et al., 2015), CUHK03 (Li et al., 2014), OpeReid (Liao et al., 2014), CUHK01 (Li et al., 2012), VIPeR (Gray and Tao, 2008), CAVIAR (Cheng et al., 2011), 3DPeS (Baltieri et al., 2011), iLIDS (Zheng et al., 2009) and PRI450s (Roth et al., 2014), where we outperform other metric learning methods proposed in the literature, both in speed and accuracy.

3.2 Related work

Metric learning is a well studied research problem (Yang and Jin, 2006). Metric learning algorithms learn an embedding of the input data, where the points from the similar classes are together and dissimilar classes are far apart. Formally, Given M data points $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_M) : x_m \in \mathbb{R}^D$ and labels $(y_1, y_2, ..., y_M) : y_i \in \{1, ..., Q\}$, where Q is the number of classes, metric learning methods aims at learning a function $f(\mathbf{x}; \mathbf{W}) : \mathbb{R}^D \mapsto \mathbb{R}^{D'}$ such that the intra-class distance in $\mathbb{R}^{D'}$ is minimized and the inter-class distance is maximized. $\mathbf{W}$ is the parameters that are being learned. This is in contrast to multi-class classification where we learn a class specific decision boundary for each of the classes. Most of the existing approaches to metric learning have been developed in the context of the Mahalanobis distance learning paradigm (Xing et al., 2002; Weinberger and Saul, 2009; Davis et al., 2007; Mignon and Jurie, 2012; Köstinger et al., 2012), where $f(\mathbf{x}; \mathbf{W}) = \mathbf{W}\mathbf{x} : \mathbf{W} \in \mathbf{R}^{D' \times D}$. This consists in learning distances of the form:

$$\mathcal{D}^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M} (\mathbf{x}_i - \mathbf{x}_j), \tag{3.1}$$

where $\mathbf{M}$ is a positive semi-definite matrix. Based on the way the problem is formulated the algorithms for learning such distances involve either optimization in the space of positive semi-definite (PSD) matrices, or learning the projection matrix $\mathbf{W}$, in which case $\mathbf{M} = \mathbf{W}^T\mathbf{W}$.

Large margin nearest neighbours (Weinberger and Saul, 2009) (LMNN) is a metric learning algorithm designed to maximize the performance of k-nearest neighbor classification in a large margin framework. Information theoretic metric learning (Davis et al., 2007) (ITML) exploits the relationship between the Mahalanobis distance and Gaussian distributions to learn the metric. Many researchers have applied LMNN and ITML to re-identification problems with

varying degree of success (Roth et al., 2014).

Pairwise Constrained Component Analysis (PCCA) (Mignon and Jurie, 2012) is a metric learning method that learns the low rank projection matrix $\mathbf{W}$ in kernel space from sparse pairwise constraints. Xiong et al. (2014) extended PCCA with an L_2 regularization term and showed that it further improves the performance.

Köstinger et al. (2012) proposed the KISS ("Keep It Simple and Straight forward") metric learning abbreviated as KISSME. Their method enjoys very fast training and they show good empirical performance and scaling properties with the number samples. However this method suffers from the Gaussian assumptions on the model.

Li et al. (2013) consider learning a local thresholding rule for metric learning. This method is computationally expensive to train, even with as few as 100 dimensions. Their paper discusses solving the problem in dual form to decouple the dependency on the dimension but in practice it is solved in primal form with off-the-shelf solvers.

The performance of many kernel-based metric learning methods for person re-identification was evaluated by Xiong et al. (2014). In particular the authors evaluated PCCA (Mignon and Jurie, 2012), variants of kernel Fisher discriminant analysis (KFDA) and reported that the KFDA variants consistently outperform all other methods. The KFDA variants they investigated were Local Fisher Discriminant Analysis (LFDA) and Marginal Fisher Discriminant Analysis (MFA).

Recently several metric learning algorithms have been presented for person re-identification. Chen et al. (2015a) attempt to learn a metric in the polynomial feature map exploiting the relationship between the Mahalanobis metric and the polynomial features. Ahmed et al. (2015) propose a deep learning model which learns the features as well as the metric jointly. Liao et al. (2015) propose XQDA exploiting the benefits of Fisher discriminant analysis and KISSME to learn a metric. However like FDA and KISSME, XQDA's modelling power is limited because of the Gaussian assumptions on the data. In another work Liao and Li (2015) apply accelerated proximal gradient descent (APGD) to a Mahalanobis metric under a logistic loss similar to the loss of PCCA (Mignon and Jurie, 2012). The application of APGD makes this model converge fast compared to existing batch metric learning algorithms but still it suffers from scalability issues because all the pairs are required to take one gradient step and the projection step on to the PSD cone is computationally expensive.

None of the above mentioned techniques explicitly models the objective that we are looking for in person re-identification, that is to optimize a weighted rank measure. We show that modeling this in the metric learning objective improves the performance. We address scalability through stochastic gradient descent (SGD) and our model naturally eliminates the need for asymmetric sample weighting as we use triplet based loss function.

There is an extensive body of work on optimizing ranking measures such as AUC, precision at k, F_1 score, etc. Most of this work focuses on learning a linear decision boundary in the original

Table 3.1: Notation

M Number of training samples
D Dimension of training samples
Q Number of classes
$(\mathbf{x}_m, y_m) \in \mathbb{R}^D \times \{1, \dots, Q\}$ m-th training sample
$\mathbb{1}_{\text{condition}}$ is equal to 1 if the condition is true, 0 otherwise
$\mathcal{S}$ The set pairs of indices of samples from the same class
$\mathcal{T}_y$ The set of indices of samples not from class y
$\mathcal{F}_{\mathbf{W}}$ The distance function under the linear map $\mathbf{W}$
$rank_{i,j}(\mathcal{F}_{\mathbf{W}})$ For i and j of same class, number of miss-labeled examples closer to i than j is
$\mathcal{L}(r)$ The rank weighting function

input space, or in the feature space for ranking a list of items based on the chosen performance measure. A well known such model is the structural SVM (Tsochantaridis et al., 2004). In contrast here we are interested in ranking pairs of items by learning a metric. A related work by McFee and Lanckriet (2010) studies metric learning with different rank measures in the structural SVM framework. Wu et al. (2011) used this framework to do person re-identification by optimizing the mean reciprocal rank criterion. Outside the direct scope of metric learning from a single feature representation, Paisitkriangkrai et al. (2015) developed an ensemble algorithm to combine different base metrics in the structural SVM framework which leads to excellent performance for re-identification. Such an approach is complementary to ours, as combining heterogeneous feature representations requires a separate additional level of normalization or the combination with a voting scheme.

We use the WARP loss from WSABIE (Weston et al., 2011), proposed for large-scale image annotation problem, that is, a multi-label classification problem. WSABIE learns a low dimensional joint embedding for both images and annotations by optimizing the WARP loss. This work reports excellent empirical results in terms of accuracy, computational efficiency, and memory footprint.

The work that is closely related to us is FRML (Lim and Lanckriet, 2014) where the authors learn a Mahalanobis metric by optimizing the WARP loss function with SGD. However there are some key differences with our approach. FRML is a linear method using an L_2 or LMNN regularizer, and relies on an expensive projection step in the SGD. Beside, this projection requires to keep a record of all the gradients in the mini-batch, which results in high memory footprint. A rationale for the projection step is to accelerate the SGD because directly optimizing a low rank matrix may result in rank deficient matrices and thus in non-isolated minima which might generalize poorly to unseen samples. We propose a computationally cheap solution to this problem by using a regularizer which approximately enforces the rank of the learned matrix efficiently.

3.3 Weighted Approximate Rank Component Analysis (WARCA)

This section presents our metric learning algorithm, Weighted Approximate Rank Component Analysis (WARCA). Table 3.1 summarizes some important notation that we use in this chapter.

Let us consider a training set of data point / label pairs:

$$(\mathbf{x}_m, y_m) \in \mathbb{R}^D \times \{1,\dots,Q\},\ m = 1,\dots,M. \tag{3.2}$$

and let $\mathcal{S}$ be the set of pairs of indices of samples from the same class:

$$\mathcal{S} = \left\{(i,j) \in \{1,\dots,M\}^2,\ y_i = y_j\right\}. \tag{3.3}$$

For each class label y we define the set $\mathcal{T}_y$ of indices of samples not from class y:

$$\mathcal{T}_y = \left\{k \in \{1,\dots,M\},\ y_k \neq y\right\}. \tag{3.4}$$

In particular, to each $(i,j) \in \mathcal{S}$ corresponds a set $\mathcal{T}_{y_i} = \mathcal{T}_{y_j}$.

Let $\mathbf{W}$ be a linear transformation that maps the data points from $\mathbb{R}^D$ to $\mathbb{R}^{D'}$, with $D' \leq D$. For the ease of notation, we do not distinguish between matrices and their corresponding linear mappings. The distance function under the linear map $\mathbf{W}$ is given by:

$$\mathcal{F}_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{W}(\mathbf{x}_i - \mathbf{x}_j)\|_2. \tag{3.5}$$

3.3.1 Problem formulation

For a pair of points (i,j) of same label $y_i = y_j$, we define a ranking error function:

$$\forall (i,j) \in \mathcal{S},\ err(\mathcal{F}_{\mathbf{W}}, i, j) = \mathcal{L}\left(rank_{i,j}(\mathcal{F}_{\mathbf{W}})\right). \tag{3.6}$$

where:

$$rank_{i,j}(\mathcal{F}_{\mathbf{W}}) = \sum_{k \in \mathcal{T}_{y_i}} \mathbb{1}_{\mathcal{F}_{\mathbf{W}}(\mathbf{x}_i,\mathbf{x}_k) \leq \mathcal{F}_{\mathbf{W}}(\mathbf{x}_i,\mathbf{x}_j)}. \tag{3.7}$$

is the number of samples $\mathbf{x}_k$ of different labels which are closer to $\mathbf{x}_i$ than $\mathbf{x}_j$ is.

Formulating our objective that way, following closely the formalism of Weston et al. (2011), shows how training a multi-class predictor shares similarities with our metric-learning problem. The former aims at avoiding, for any given sample to have incorrect classes with responses higher than the correct one, while the latter aims at avoiding, for any pair of samples $(\mathbf{x}_i, \mathbf{x}_j)$ of the same label, to have samples $\mathbf{x}_k$ of other classes in between them.

Minimizing directly the rank treats all the rank positions equally, and usually in many problems including person re-identification we are interested in maximizing the correct match within

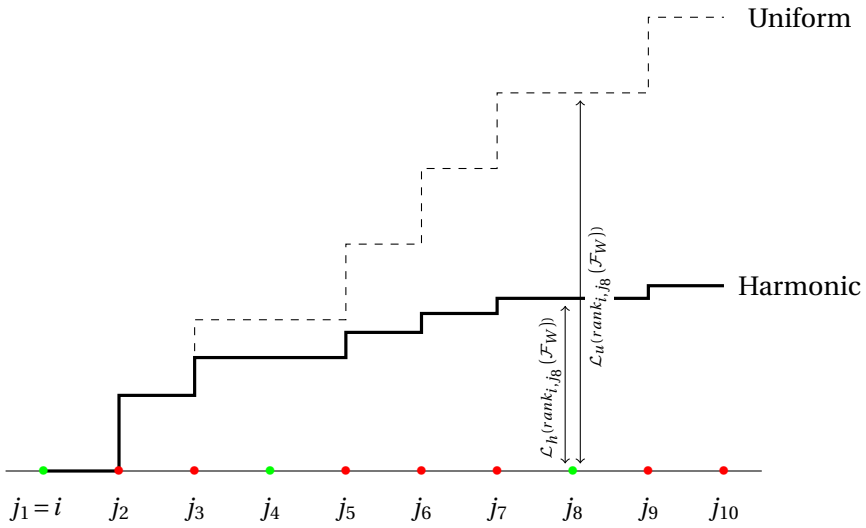


Figure 3.1: Given a query sample i, we define the rank of another sample j of the same class, with respect to that query, as a function of the number of samples of other classes closer to the query. The horizontal axis here corresponds to indexes of the samples sorted according to their similarities to the query sample. Green dots stand for samples of the same class as the query, and red circles for samples of different classes. The dashed line shows the uniform weighting of the rank $\mathcal{L}_u\left(rank_{i,j}\left(\mathcal{F}_{\mathbf{W}}\right)\right)$, which increases by one unit each time we cross a sample of an incorrect class, and the thick line is the harmonic weighting $\mathcal{L}_h\left(rank_{i,j}\left(\mathcal{F}_{\mathbf{W}}\right)\right)$, which increases as $1/n$. We used harmonic weighting for all our experiments.

the top few rank positions. This can be achieved by a weighting function $\mathcal{L}(\cdot)$ which penalizes more a drop in the rank at the top positions than at the bottom positions. In particular we use the rank weighting function presented by Usunier et al. (2009), of the form:

$$\mathcal{L}(R) = \sum_{r=1}^{R} \alpha_r,\ \alpha_1 \geq \alpha_2 \geq \ldots \geq 0. \tag{3.8}$$

For example, using $\alpha_1 = \alpha_2 = \cdots = \alpha_N$ will treat all rank positions equally, and using higher values of αs in top few rank positions will weight top rank positions more. We use the harmonic weighting, which has such a profile and was also used by Weston et al. (2011) as it yielded state-of-the-art results on their application.

Finally, we would like to solve the following empirical risk minimization problem:

$$\min_{\mathbf{W}} \frac{1}{|\mathcal{S}|} \sum_{(i,j)\in\mathcal{S}} \mathcal{L}\left(rank_{i,j}\left(\mathcal{F}_{\mathbf{W}}\right)\right). \tag{3.9}$$

3.3.2 Approximate OrthoNormal (AON) regularizer

The optimization problem of Equation 3.9 may lead to severe over-fitting on small and medium scale data-sets. Regularization terms are central in re-identification for that reason.

The standard way of regularizing a low-rank metric learning objective function is by using an L_2 penalty, such as the Frobenius norm (Lim and Lanckriet, 2014). However, such a regularizer tends to push toward rank-deficient linear mappings, which we observe in practice (see § 3.4.4, and in particular Figure 3.3a).

Lim and Lanckriet (2014) in their FRML algorithm, address this problem by using a Riemannian manifold update step in their SGD algorithm, which is computationally expensive and induces a high memory footprint. We propose an alternative approach that maintains the rank of the matrix by pushing toward orthonormal matrices. This is achieved by using as a penalty term the L_2 divergence of $\mathbf{W}\mathbf{W}^T$ from the identity matrix $\mathbf{I}$:

$$\|\mathbf{W}\mathbf{W}^T - \mathbf{I}\|^2. \tag{3.10}$$

This orthonormal regularizer can also be seen as a strategy to mimic the behavior of approaches such as PCA or FDA, which ensure that the learned linear transformation is orthonormal. For such methods, this property emerges from the strong Gaussian prior over the data, which is beneficial on small data-sets but degrades performance on large ones where it leads to under-fitting. Controlling the orthonormality of the learned mapping through a regularizer weighted by a meta-parameter λ allows us to adapt it on each data-set individually through cross-validation.

With this regularizer the ERM problem of Equation 3.9 becomes the following regularized loss minimization problem:

$$\min_{\mathbf{W}} \frac{\lambda}{2}\|\mathbf{W}\mathbf{W}^T - \mathbf{I}\|^2 + \frac{1}{|\mathcal{S}|}\sum_{(i,j)\in\mathcal{S}} \mathcal{L}\left(rank_{i,j}\left(\mathcal{F}_{\mathbf{W}}\right)\right). \tag{3.11}$$

3.3.3 Max-margin reformulation

The RLM problem in Equation 3.11 aims at minimizing the 0-1 loss and as we discussed in Chapter 2 that minimizing the 0-1 loss is a difficult optimization problem. Applying the reasoning behind the WARP loss to make it tractable, we approximate the 0-1 loss with the hinge loss with margin $\gamma \geq 0$. This is equivalent to minimizing the following regularized loss function:

$$G_\lambda(\mathbf{W}) = \frac{\lambda}{2}\|\mathbf{W}\mathbf{W}^T - \mathbf{I}\|^2 + \frac{1}{|\mathcal{S}|}\sum_{(i,j)\in\mathcal{S}}\sum_{k\in\mathcal{T}_{y_i}} \mathcal{L}(rank^{\gamma}_{i,j}(\mathcal{F}_{\mathbf{W}}))\frac{\left|\gamma + \xi_{ijk}\right|_+}{rank^{\gamma}_{i,j}(\mathcal{F}_{\mathbf{W}})}, \tag{3.12}$$

where:

$$\xi_{ijk} = \mathcal{F}_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j) - \mathcal{F}_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_k) \tag{3.13}$$

and $rank^{\gamma}_{i,j}(\mathcal{F}_{\mathbf{W}})$ is the margin penalized rank:

$$rank^{\gamma}_{i,j}(\mathcal{F}_{\mathbf{W}}) = \sum_{k \in \mathcal{T}_{y_i}} \mathbb{1}_{\gamma + \xi_{ijk} > 0}. \tag{3.14}$$

The loss function in Equation 3.12 is the WARP loss (Usunier et al., 2009; Weston et al., 2011; Lim and Lanckriet, 2014). It was shown by Weston et al. (2011) that the WARP loss can be efficiently minimized by using stochastic gradient descent and we follow the same approach. An unbiased estimator of $\frac{|\gamma + \xi_{ijk}|_+}{rank^{\gamma}_{i,j}(\mathcal{F}_{\mathbf{W}})}$ can be obtained with the following sampling procedure:

1. Sample (i, j) uniformly at random from $\mathcal{S}$.
2. For the selected (i, j) uniformly sample k from $\{k \in \mathcal{T}_{y_i} : \gamma + \xi_{ijk} > 0\}$, *i.e.* from the set of incorrect matches scored higher than the correct match $\mathbf{x}_j$.

The sampled triplet (i, j, k) has a contribution of $|\gamma + \xi_{ijk}|_+$ because the probability of drawing a k in step 2 from the violating set is $\frac{1}{rank^{\gamma}_{i,j}(\mathcal{F}_{\mathbf{W}})}$ (Weston et al., 2011).

We also get a stochastic approximation of the rank $rank^{\gamma}_{i,j}(\mathcal{F}_{\mathbf{W}})$ from the above sampling procedure. Let Z denote the number of times we have to sample from $\mathcal{T}_{y_i}$ to get a margin violating point k for (i, j) . It follows a geometric distribution:

$$\mathbb{P}(Z = z) = (1 - p)^{z-1} p, \tag{3.15}$$

where

$$p = \frac{rank^{\gamma}_{i,j}(\mathcal{F}_{\mathbf{W}})}{|\mathcal{T}_{y_i}|}. \tag{3.16}$$

Therefore the expected number of times we have to sample until we get a margin violating point $\mathbb{E}(Z)$ is given by:

$$\mathbb{E}(Z) = \frac{1}{p} = \frac{|\mathcal{T}_{y_i}|}{rank^{\gamma}_{i,j}(\mathcal{F}_{\mathbf{W}})}. \tag{3.17}$$

and this motivates the use of the following approximation:

$$\hat{rank}^{\gamma}_{i,j}(\mathcal{F}_{\mathbf{W}}) = \left\lfloor \frac{|\mathcal{T}_{y_i}|}{z} \right\rfloor, \tag{3.18}$$

which can be shown to be an upper bound of the true rank $rank^{\gamma}_{i,j}(\mathcal{F}_{\mathbf{W}})$ by at most a factor

$\frac{\ln p}{p-1} \geq 1$.

We use the above sampling procedure to solve WARCA efficiently using mini-batch stochastic gradient descent. We use the Adam SGD algorithm (Kingma and Ba, 2014a), which is found to converge faster empirically compared to vanilla SGD. Algorithm 4 describes a vanilla stochastic gradient descent algorithm for WARCA.

Algorithm 4 Stochastic gradient descent algorithm for WARCA

Input: Data point/label pairs: $(\mathbf{x}_m, y_m) \in \mathbb{R}^D \times \{1,\dots,Q\},\ m = 1,\dots,M$, Regularizer: $\lambda \geq 0$, Initial solution: $\mathbf{W}_0 \in \mathbb{R}^{D' \times D}$, Step size: η, Margin: γ

$\mathcal{S} = \{(i,j) \in \{1,\dots,M\}^2,\ y_i = y_j\}$
$\forall y_i \in y, \mathcal{T}_{y_i} = \{k \in \{1,\dots,M\},\ y_k \neq y_i\}$
$t = 0$
while (not converged) **do**
  Sample (i,j) uniformly at random from $\mathcal{S}$
  $d_{ij} = \mathcal{F}_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j)$
  $z = 0$
  do
    Sample k uniformly at random from $\mathcal{T}_{y_i}$
    $d_{ik} = \mathcal{F}_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_k)$
    $z = z + 1$
  while $z \leq |\mathcal{T}_{y_i}|$ or $\gamma + d_{ij} > d_{ik}$
  $\mathbf{W}_{t+1} = \mathbf{W}_t - 2\eta\lambda(\mathbf{W}_t\mathbf{W}_t^T - \mathbf{I})\mathbf{W}_t$
  if $\gamma + d_{ij} > d_{ik}$ **then**
    $\mathbf{W}_{t+1} = \mathbf{W}_{t+1} - 2\eta\mathcal{L}\left(\left\lfloor \frac{|\mathcal{T}_{y_i}|}{z} \right\rfloor\right)\nabla_{\mathbf{W}}|\gamma + \mathcal{F}_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j) - \mathcal{F}_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_k)|_+\big|_{\mathbf{W}=\mathbf{W}_t}$
  end if
  $t = t + 1$
end while
Output $\mathbf{W}_t$

3.3.4 WARCA in kernel space

The most commonly used features in person re-identification are histogram-based such as LBP, SIFT BOW, RGB histograms to name a few. The most natural distance measure for histogram-based features is the χ^2 distance. Most of the standard metric learning methods work on the Euclidean distance with PCCA being a notable exception. To plug any arbitrary metric which is suitable for the features, such as χ^2, one has to resort to explicit feature maps that approximate the χ^2 metric. However, it blows up the dimension and the computational cost. Another way to deal with this problem is to do metric learning in kernel space, which is the approach we follow.

Let $\mathbf{X} \in \mathbb{R}^{D \times M}$ be the training data matrix, that is a set of M points in $\mathbb{R}^D$. Let us assume that $\mathbf{W}$

is spanned by the training samples:

$$\mathbf{W} = \mathbf{A}\mathbf{X}^T = \mathbf{A}\begin{pmatrix} \mathbf{x}_1^T \\ \dots \\ \mathbf{x}_N^T \end{pmatrix}. \tag{3.19}$$

Where $\mathbf{A} \in \mathbb{R}^{D' \times M}$. This leads us to expressing the distance function $\mathcal{F}_{\mathbf{W}}(.,.)$ in terms of $\mathbf{A}$, that is:

$$\begin{aligned} \mathcal{F}_{\mathbf{A}}(x_i, x_j) &= \|\mathbf{A}\mathbf{X}^T(\mathbf{x}_i - \mathbf{x}_j)\|_2, & (3.20)\\ &= \|\mathbf{A}(\mathbf{k}_i - \mathbf{k}_j)\|_2. & (3.21) \end{aligned}$$

Where $\mathbf{k}_i$ is the i^{th} column of the kernel matrix $\mathbf{K} = \mathbf{X}^T\mathbf{X}$. Then the regularized loss function in Equation 3.12 becomes:

$$G_\lambda(\mathbf{A}) = \frac{\lambda}{2}\|\mathbf{A}\mathbf{K}\mathbf{A}^T - \mathbf{I}\|^2 + \frac{1}{|\mathcal{S}|}\sum_{(i,j)\in\mathcal{S}}\sum_{k\in\mathcal{T}_{y_i}} \mathcal{L}(rank^{\gamma}_{i,j}(\mathcal{F}_{\mathbf{A}}))\frac{|\gamma + \xi_{ijk}|_+}{rank^{\gamma}_{i,j}(\mathcal{F}_{\mathbf{A}})}, \tag{3.22}$$

with:

$$\xi_{ijk} = \mathcal{F}_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j) - \mathcal{F}_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_k). \tag{3.23}$$

Apart from being able to do non-linear metric learning, kernelized WARCA can be solved efficiently again by using stochastic gradient descent. If we use the inverse of the kernel matrix as the pre-conditioner of stochastic gradient, the computation of the update equation, as well the parameter update, can be carried out efficiently. Mignon and Jurie (2012) used the same technique to minimize PCCA using gradient descent, and showed that it converges faster than vanilla gradient descent. We use the same technique to derive an efficient update rule for our kernelized WARCA. A stochastic sub-gradient of Equation 3.22 with the sampling procedure described in the previous section is given as:

$$\nabla G_\lambda(\mathbf{A}) = 2\lambda(\mathbf{A}\mathbf{K}\mathbf{A}^T - \mathbf{I})\mathbf{A}\mathbf{K} + 2\mathcal{L}(rank^{\gamma}_{i,j}(\mathcal{F}_{\mathbf{A}}))\mathbf{A}\mathbb{1}_{\gamma+\xi_{ijk}>0}\mathbf{U}_{ijk}, \tag{3.24}$$

where:

$$\mathbf{U}_{ijk} = \frac{(\mathbf{k}_i - \mathbf{k}_j)(\mathbf{k}_i - \mathbf{k}_j)^T}{d_{ij}} - \frac{(\mathbf{k}_i - \mathbf{k}_k)(\mathbf{k}_i - \mathbf{k}_k)^T}{d_{ik}}, \tag{3.25}$$

and:

$$d_{ij} = \mathcal{F}_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j), \;\; d_{ik} = \mathcal{F}_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_k). \tag{3.26}$$

Multiplying the right hand side of equation B.14 by $\mathbf{K}^{-1}$:

$$\nabla G_\lambda(\mathbf{A})\mathbf{K}^{-1} = 2\lambda(\mathbf{A}\mathbf{K}\mathbf{A}^T - \mathbf{I})\mathbf{A} + 2\mathcal{L}(rank^\gamma_{i,j}(\mathcal{F}_\mathbf{A}))\mathbf{A}\mathbf{K}\mathbb{1}_{\gamma+\xi_{ijk}>0}\mathbf{V}_{ijk}. \tag{3.27}$$

with:

$$\mathbf{V}_{ijk} = \mathbf{K}^{-1}\mathbf{U}_{ijk}\mathbf{K}^{-1} = \frac{(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^T}{d_{ij}} - \frac{(\mathbf{e}_i - \mathbf{e}_k)(\mathbf{e}_i - \mathbf{e}_k)^T}{d_{ik}}. \tag{3.28}$$

where $\mathbf{e}_l$ is the l^{th} column of the canonical basis that is the vector whose l^{th} component is one and all others are zero. In the preconditioned stochastic sub-gradient descent we use updates of the form:

$$\mathbf{A}_{t+1} = (\mathbf{I} - 2\lambda\eta(\mathbf{A}_t\mathbf{K}\mathbf{A}_t^T - \mathbf{I}))\mathbf{A}_t - 2\eta\mathcal{L}(rank^\gamma_{i,j}(\mathcal{F}_\mathbf{A}))\mathbf{A}_t\mathbf{K}\mathbb{1}_{\gamma+\xi_{ijk}>0}\mathbf{V}_{ijk}. \tag{3.29}$$

Please note that $\mathbf{V}_{ijk}$ is a very sparse matrix with only nine non-zero entries. This makes the update extremely fast. Preconditioning also enjoys faster convergence rates since it exploits second order information through the preconditioning operator, here the inverse of the kernel matrix (Chapelle, 2007).

3.4 Experiments

We evaluate our proposed algorithm on nine standard person re-identification data-sets. All experiments are evaluated in the zero-shot setting when the training and test classes are disjoint. We first describe the data-sets and baseline algorithms and then present our results.

3.4.1 Data-sets and baselines

The largest data-set we experimented with is the **Market-1501** data-set (Zheng et al., 2015) which is composed of 32,668 images of 1,501 persons captured from 6 different view points. It uses DPM (Felzenszwalb et al., 2008) detected bounding boxes as annotations. The **CUHK03** data-set (Li et al., 2014) consists of 13,164 images of 1,360 persons and it has both DPM detected and manually annotated bounding boxes. We use the manually annotated bounding boxes here. The **OpeReid** data-set (Liao et al., 2014) consists of 7,413 images of 200 persons. The **CUHK01** data-set (Li et al., 2012) is composed of 3,884 images of 971 persons, with two pairs of images per person, each pair taken from a different viewpoint. Each image is of resolution 160×60. The **VIPeR** (Gray and Tao, 2008) data-set has 1,264 images of 632 person, with 2 images per person. The images are of resolution 128x48, captured from horizontal viewpoints but from widely different directions. The **PRID450s** data-set (Roth et al., 2014) consists of 450 image pairs recorded from two different static surveillance cameras. The **CAVIAR** data-set (Cheng et al., 2011) consists of 1,220 images of 72 individuals from 2 cameras in a shopping mall. The number of images per person varies from 10 to 20 and image resolution

also varies significantly from 141×72 to 39×17. The **3DPeS** data-set (Baltieri et al., 2011) has 1,011 images of 192 individuals, with 2 to 6 images per person. The data-set is captured from 8 outdoor cameras with horizontal but significantly different viewpoints. Finally the **iLIDS** data-set (Zheng et al., 2009) contains 476 images and 119 persons, with 2 to 8 images per individual. It is captured from a horizontal view point at an airport.

We compare our method against the current state-of-the-art baselines MLAPG (Liao and Li, 2015), rPCCA (Mignon and Jurie, 2012), SVMML (Li et al., 2013), FRML (Lim and Lanckriet, 2014), LFDA (Xiong et al., 2014) and KISSME (Köstinger et al., 2012). A brief overview of these methods is given in section 3.2. rPCCA, MLAPG, SVMML, FRML are iterative methods whereas LFDA and KISSME are spectral methods on the second order statistics of the data. Since WARCA, rPCCA and LFDA are kernel methods we used both the χ^2 kernel and the linear kernel with them to benchmark the performance. Marginal Fisher discriminant analysis (MFA) is proven to give similar result to that of LFDA so we do not use them as the baseline.

We did not compare against other ranking based metric learning methods such as LORETA (Shalit et al., 2012), OASIS (Chechik et al., 2010) and MLR (McFee and Lanckriet, 2010) because all of them are linear methods. In fact we derived a kernelized OASIS but the results were not as good as ours or rPCCA. We also do not compare against LMNN and ITML because many researchers have evaluated them before (Mignon and Jurie, 2012; Köstinger et al., 2012; Li et al., 2013) and found out that they do not perform as well as other methods considered here.

3.4.2 Technical details

For the Market-1501 data-set we used the experimental protocol and features described in Zheng et al. (2015). We used their baseline code and features. As Market-1501 is quite large for kernel methods we do not evaluate them. We also do not evaluate the linear methods such as Linear rPCCA and SVMML because their optimization algorithms were found to be very slow.

All other evaluations where carried out in the single-shot experiment setting (Gong et al., 2014) and our experimental settings are very similar to the one adopted by Xiong et al. (2014). Except for Market-1501, we randomly divided all the data-sets into two subsets such that there are P individuals in the test set. We created 10 such random splits. In each partition one image of each person was randomly selected as a probe image, and the rest of the images were used as gallery images and this was repeated 10 times. The position of the correct match was processed to generate the CMC curve. We followed the standard train-validation-test splits for all the other data-sets and P was chosen to be 100, 119, 486, 316, 225, 36, 95 and 60 for CUHK03, OpeReid, CUHK01, VIPeR, PRID450s, CAVIAR, 3DPeS and iLIDS respectively.

We used the same set of features for all the data-sets except for the Market-1501 and all the features are essentially histogram based. First all the data-sets were re-scaled to 128×48 resolution and then 16 bin color histograms on RGB, YUV, and HSV channels, as well as

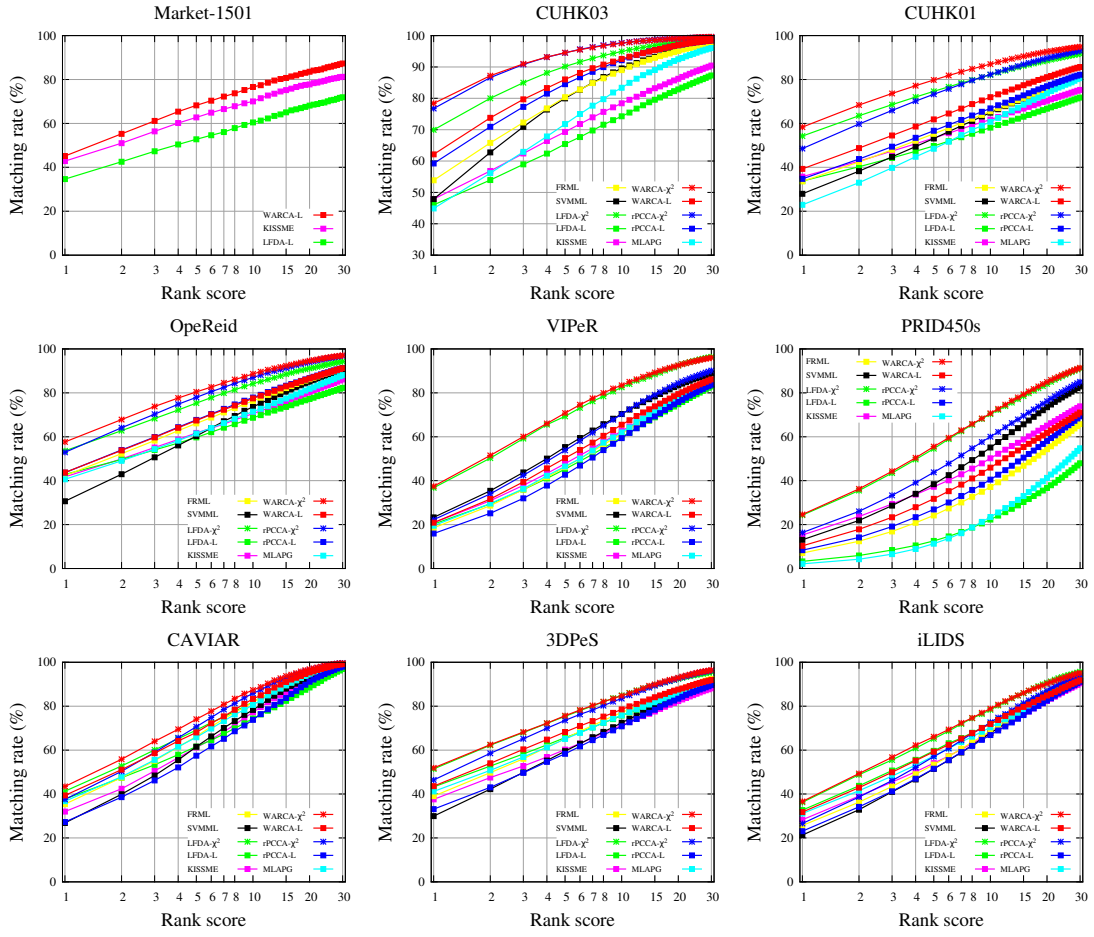


Figure 3.2: CMC curves comparing WARCA against state-of-the-art methods on nine re-identification data-sets.

Table 3.2: Table showing the rank 1, rank 5 and AUC performance measure of our method WARCA against other state-of-the-art methods. Bold fields indicate best performing methods. The dashes indicate computation that could not be run in a realistic setting on Market-1501

(a) Rank 1 accuracy

Dataset	WARCA-χ^2	WARCA-L	rPCCA-χ^2	rPCCA-L	MLAPG	FRML	SVMML	LFDA-χ^2	LFDA-L	KISSME
Market-1501	–	**45.16±0.00**	–	–	–	–	–	–	34.65±0.00	42.81±0.00
CUHK03	**78.38±2.44**	62.12±2.07	76.74±2.06	59.22±2.65	44.90±1.57	53.87±2.31	47.89±2.59	69.94±2.21	46.02±1.55	47.88±1.80
CUHK01	**58.34±1.26**	39.30±0.76	48.55±1.12	34.73±1.06	22.92±0.94	33.58±0.69	27.96±0.86	54.25±1.04	33.74±0.73	35.74±0.95
OpeReid	**57.65±1.60**	43.74±1.34	52.89±1.78	43.66±1.45	40.63±1.31	42.27±1.35	30.63±1.51	53.58±1.65	42.84±1.18	41.76±1.36
VIPeR	**37.47±1.70**	20.86±1.04	22.25±1.91	15.91±1.16	19.49±2.26	18.52±0.78	23.28±1.53	36.77±2.10	20.22±1.85	20.89±1.22
PRID450s	**24.58±1.75**	10.33±1.20	16.35±1.30	8.34±1.25	2.13±0.59	7.05±1.60	13.08±1.63	24.31±1.44	3.24±0.95	15.24±1.56
CAVIAR	**43.44±1.82**	39.35±1.98	37.56±2.17	27.26±2.15	36.74±1.96	35.40±2.67	26.82±1.64	41.29±2.25	37.72±2.08	31.99±2.17
3DPeS	**51.89±2.27**	43.57±2.18	46.42±2.25	33.12±1.58	41.17±2.26	39.03±1.85	29.94±2.10	51.44±1.40	43.24±2.57	37.55±1.80
iLIDS	**36.61±2.40**	31.77±2.77	26.57±2.60	23.07±3.07	31.13±1.57	25.68±2.25	21.32±2.89	36.23±1.89	32.70±3.12	28.29±3.59

(b) Rank 5 accuracy

Dataset	WARCA-χ^2	WARCA-L	rPCCA-χ^2	rPCCA-L	MLAPG	FRML	SVMML	LFDA-χ^2	LFDA-L	KISSME
Market-1501	–	**68.23±0.00**	–	–	–	–	–	–	52.76±0.00	62.74±0.00
CUHK03	**94.55±1.31**	86.03±1.62	94.50±1.29	84.52±1.41	71.80±1.52	80.36±1.22	79.97±2.08	90.15±1.27	65.41±1.66	69.29±2.35
CUHK01	**79.76±0.69**	61.84±0.98	73.29±1.32	56.67±1.20	48.48±1.49	55.27±0.83	53.11±0.78	74.60±1.00	49.73±0.91	53.34±0.69
OpeReid	**80.43±1.71**	67.39±1.02	77.95±1.82	67.68±1.25	61.45±1.61	66.08±1.30	60.32±1.31	75.34±1.76	59.70±1.37	61.74±1.55
VIPeR	**70.78±2.43**	50.29±1.61	53.82±2.32	42.71±2.02	46.49±2.23	46.15±1.62	55.28±1.99	69.30±2.23	45.25±1.90	47.73±2.28
PRID450s	**55.52±2.23**	31.73±3.08	43.82±2.18	26.89±2.21	11.29±1.66	24.16±3.04	38.38±1.77	54.58±2.06	12.55±1.41	37.22±1.81
CAVIAR	**74.06±3.13**	68.06±2.44	70.62±2.26	57.44±2.48	65.83±2.73	66.24±3.08	61.53±3.64	69.12±3.02	61.60±2.94	61.17±3.21
3DPeS	**75.64±2.80**	68.26±1.91	73.54±2.26	58.34±2.31	65.06±1.89	65.20±2.15	59.52±2.62	75.36±1.91	65.64±1.91	60.22±2.05
iLIDS	**66.09±2.31**	59.27±3.12	57.07±2.93	51.55±3.59	57.31±3.12	53.42±2.17	51.45±4.30	65.20±2.68	59.66±2.51	54.08±3.63

(c) AUC score

Dataset	WARCA-χ^2	WARCA-L	rPCCA-χ^2	rPCCA-L	MLAPG	FRML	SVMML	LFDA-χ^2	LFDA-L	KISSME
Market-1501	–	**75.41±0.00**	–	–	–	–	–	–	60.53±0.00	70.02±0.00
CUHK03	**93.94±0.76**	89.67±0.80	93.92±0.81	89.17±0.69	82.30±1.01	86.64±0.65	86.64±1.07	91.66±0.68	74.23±1.51	77.68±1.83
CUHK01	**84.99±0.65**	71.88±0.67	81.00±0.88	67.56±0.93	62.84±1.51	66.39±0.76	65.73±1.07	80.84±0.80	58.92±1.08	62.36±0.95
OpeReid	**86.47±1.08**	77.17±0.94	85.25±1.16	77.42±1.01	72.34±1.11	76.51±0.88	73.88±1.04	82.67±1.30	68.96±1.53	71.33±1.14
VIPeR	**81.87±1.07**	67.00±1.11	71.30±1.50	62.40±1.43	64.71±1.15	64.19±1.39	71.04±1.63	81.34±1.21	62.67±1.35	64.74±1.20
PRID450s	**72.13±1.49**	50.07±2.25	63.10±2.16	46.19±1.89	30.81±2.19	42.97±2.84	59.54±1.25	71.55±1.70	28.18±1.22	53.83±1.86
CAVIAR	**85.76±1.48**	83.01±1.44	84.41±1.28	76.57±1.29	81.58±1.50	81.88±1.85	79.38±2.19	81.94±2.32	76.76±1.69	78.85±1.54
3DPeS	**83.89±1.53**	78.07±1.57	82.84±1.44	72.27±1.96	75.98±1.28	76.89±1.44	73.38±1.70	83.49±0.95	75.87±1.49	72.22±1.31
iLIDS	**79.04±1.60**	73.42±1.96	74.10±2.04	69.60±2.44	72.45±1.99	71.26±1.55	70.25±2.09	78.98±1.43	74.26±2.02	70.33±2.90

texture histogram based on Local Binary Patterns (LBP) were extracted on 6 non-overlapping horizontal patches. All the histograms are normalized per patch to have unit L_1 norm and concatenated into a single vector of dimension 2,580 (Mignon and Jurie, 2012; Xiong et al., 2014).

The source codes for LFDA, KISSME and SVMML are available from their respective authors website, and we used those to reproduce the baseline results (Xiong et al., 2014). The code for PCCA is not released publicly. A version from Xiong et al. (2014) is available publicly but the memory footprint of that implementation is very high making it impossible to use with large data-sets (e.g. it requires 17GB of RAM to run on the CAVIAR data-set). Therefore to reproduce the results in (Xiong et al., 2014) we wrote our own implementation, which uses 30 times less memory and can scale to much larger data-sets. We also ran sanity checks to make sure that it behaves the same as that of the baseline code. All the implementations were done in Matlab with mex functions for the acceleration of the critical components.

In order to fairly evaluate the algorithms, we set the dimensionality of the projected space to be same for WARCA, rPCCA and LFDA. For the Market-1501 data-set the dimensionality used is 200 and for VIPeR it is 100 and all the other data-sets it is 40. We choose the regularization parameter and the learning rate through cross-validation across the data splits using grid search in $(\lambda, \eta) \in \{10^{-8}, \ldots, 1\} \times \{10^{-3}, \ldots, 1\}$. Margin γ is fixed to 1. Since the size of the parameter matrix scales in $O(D^2)$ for SVMML and KISSME we first reduced the dimension of the original features using PCA keeping 95% of the original variance and then applied these algorithms. In our tables and figures WARCA$-\chi^2$, WARCA-L, rPCCA$-\chi^2$, rPCCA-L, LFDA$-\chi^2$ and LFDA-L denote WARCA with χ^2 kernel, WARCA with linear kernel, rPCCA with χ^2 kernel, rPCCA with linear kernel, and LFDA with χ^2 kernel, LFDA with linear kernel respectively.

For all experiments with WARCA we used harmonic weighting for the rank weighting function of Equation 3.8. We also tried uniform weighting which gave poor results compared to the harmonic weighting. For all the data-sets we used a mini-batch size of 512 in the SGD algorithm and we ran the SGD for 2000 iterations (A parameter update using the mini-batch is considered as 1 iteration).

Tables 3.2a and 3.2b summarize respectively the rank-1 and rank-5 performance of all the methods, and Table 3.2c summarizes the Area Under the Curve (AUC) performance score. Figure 3.2 reports the CMC curves comparing WARCA against the baselines on the nine data-sets. The square and the star markers denote linear and kernel methods respectively.

WARCA improves over all other methods on all the data-sets. On VIPeR, 3DPeS, PRID450s and iLIDS data-sets LFDA come very close to the performance of WARCA. The reason for this is that these data-sets are too small and consequently simple methods such as LFDA which exploits strong prior assumptions on the data distribution work nearly as well as WARCA.

3.4.3 Comparison against state-of-the-art

We also compare against the state-of-the-art results reported using recent algorithms such as MLAPG on LOMO features (Liao and Li, 2015), MLPOLY (Chen et al., 2015a) and IDEEP (Ahmed et al., 2015) on VIPeR, CUHK01 and CUHK03 data-sets. The reason for not including these comparisons in the main results is because apart from MLAPG the code for other methods is not available, or the features are different which makes a fair comparison difficult. Our goal is to evaluate experimentally that, given a set of features, which is the best off-the-shelf metric learning algorithm for re-identification.

In this set of experiments we used the state-of-the-art LOMO features (Liao et al., 2015) with WARCA for VIPeR and CUHK01 data-sets. The results are summarized in the Table 3.3. We improve the rank1 performance by 21% on CUHK03 by 1.40% on CUHK01 data-set.

Table 3.3: Comparison of WARCA against state-of-the-art results for person re-identification

Data-set	WARCA(Ours)				MLAPG (Liao and Li, 2015)				MLPOLY (Chen et al., 2015a)				IDEEP (Ahmed et al., 2015)			
	rank=1	rank=5	rank=10	rank=20	rank=1	rank=5	rank=10	rank=20	rank=1	rank=5	rank=10	rank=20	rank=1	rank=5	rank=10	rank=20
VIPeR	40.22	68.16	80.70	91.14	**40.73**	**69.94**	**82.34**	**92.37**	36.80	70.40	83.70	91.70	34.81	63.61	75.63	84.49
CUHK01	**65.64**	85.34	90.48	**95.04**	64.24	**85.41**	**90.84**	94.92	-	-	-	-	47.53	71.60	80.25	87.45
CUHK03	**78.38**	**94.5**	**97.52**	**99.11**	57.96	87.09	94.74	98.00	-	-	-	-	54.74	86.50	94.02	97.02

3.4.4 Analysis of the AON regularizer

Here we present an empirical analysis of the AON regulariser against the standard Frobenius norm regularizer. We used the VIPeR data-set with LOMO features for the experiments shown in the first two rows of Figure 3.3. With very low regularization strength AON and Frobenius behave the same. As the regularization strength increases, Frobenius results in rank deficient mappings (Figure 3.3a), which is less discriminant and perform poorly on the test set (Figure 3.3b). The AON regularizer on the contrary pushes towards orthonormal mappings, and results in an embedding well conditioned, which generalizes well to the test set. It is also worth noting that training with the AON regularizer is robust over a wide range of the regularization parameters which is not the case for the Frobenius norm. Finally, the AON regularizer was found to be very robust to the choice of the SGD step size η (Figure 3.3c) which is a crucial parameter in large-scale learning. A similar behavior was observed by Lim and Lanckriet (2014) with their orthonormal Riemannian gradient update step in the SGD but it is computationally expensive and not trivial to use with modern SGD algorithms.

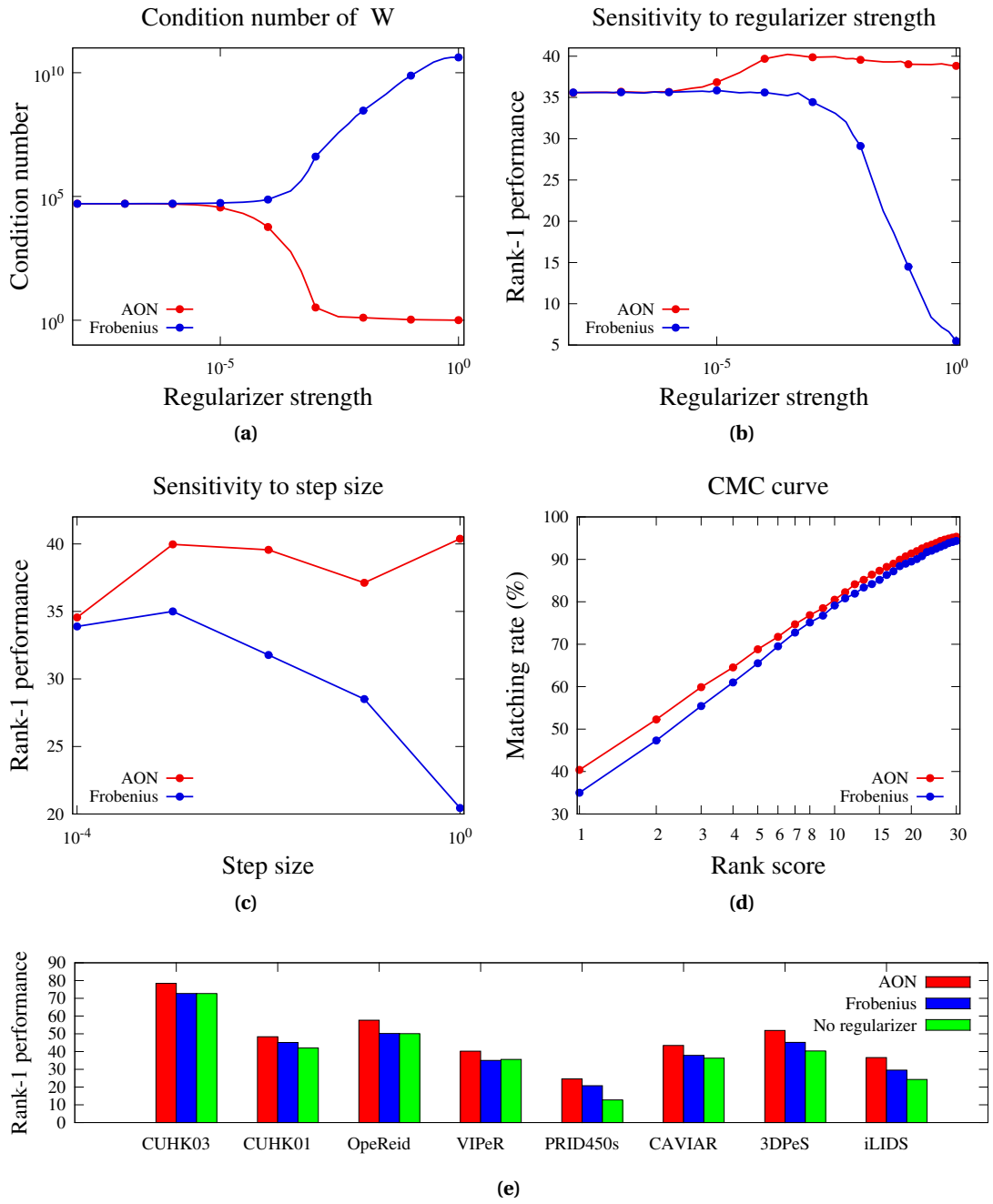


Figure 3.3: Comparison of the Approximate OrthoNormal (AON) regularizer we use in our algorithm to the standard Frobenius norm (L_2) regularizer. Graph (a) shows the condition number (ratio between the two extreme eigenvalues of the learned mapping) vs. the weight λ of the regularization term. As expected, the AON regularizer pushes this value to one, as it eventually forces the learning to chose an orthonormal transformation, while the Frobenius regularizer eventually kills the smallest eigenvalues to zero, making the ratio extremely large. Graph (b) shows the Rank-1 performance vs. the regularizer weight λ, graph (c) the Rank-1 performance vs. the SGD step size η, graph (d) CMC curve with the two regularizers and finally graph (e) shows the Rank-1 performance on different data-sets

3.4.5 Analysis of the training time

Figure 3.4 illustrates how the performance in test of WARCA and rPCCA increase as a function of training time on 3 data-sets. We implemented both the algorithms entirely in C++ to have a fair comparison of running times. In this set of experiments we used 730 test identities for CUHK03 data-set to have a quick evaluation. Experiments with other data-sets follow the same protocol described above. Please note that we do not include spectral methods in this plot because the solutions are found analytically. Linear spectral methods are very fast for low dimensional problems but the training time scales quadratically in the data dimension. In case of kernel spectral methods the training time scales quadratically in the number of data points. We also do not include iterative methods MLAPG and SVMML because they proved to be very slow and not giving good performance.

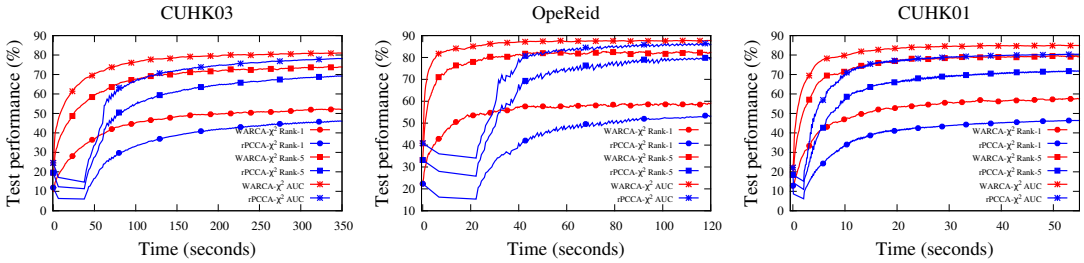


Figure 3.4: WARCA performs significantly better than the state-of-the-art rPCCA on large data-sets for a given training time budget

3.5 Discussion

We have presented a simple and scalable approach to metric learning that combines a new and simple regularizer to a proxy for a weighted sum of the precision at different ranks. The later can be used for any weighting of the precision-at-k metrics. Experimental results show that it outperforms state-of-the-art methods on standard person re-identification data-sets, and that contrary to most of the current state-of-the-art methods, it allows for large-scale learning.

WARCA is a shallow metric learning algorithm. It works well when we have a vector space embedding of the data such as the different feature descriptors we discussed in section 3.4. When we extract those features from raw pixel data we discard a lot of information which might be useful for learning an informative embedding. This might hurt the performance of shallow learning algorithms like WARCA. This is one of the limitations of WARCA. This limitation can tackled by deriving a deep metric learning algorithm with WARCA loss, where the layers are parametrized by convolutions. However deep learning algorithms requires large amount of data to train and the current data-sets available for person re-identification are limited in size.

From a more theoretical perspective, we are interested also in looking at the relations between

the behavior of the learning with the orthonormal regularizer, and the recent residual networks (He et al., 2016). In both case, strong regularization pushes toward full-rank mappings instead of null transformations, as standard L_2 penalty does, which appears to be a very reasonable behavior to expect in general.

4 Kronecker Recurrent Units

Contents

In this chapter we present a new recurrent neural network (RNN) called Kronecker Recurrent Units (KRU). Unlike classical RNNs, KRU is parameter efficient and robust to vanishing and exploding gradients. KRU is based on the following preprint:

> Cijo Jose, Moustpaha Cisse, and Francois Fleuret. Kronecker recurrent units. *arXiv preprint arXiv:1705.10142*, 2017

4.1 Introduction

Deep neural networks have defined the state-of-the-art in a wide range of problems in computer vision, speech analysis, and natural language processing (Krizhevsky et al., 2012c; Hinton et al., 2012; Mikolov, 2012). However, these models suffer from two key issues. (1) They are over-parametrized; thus it takes a very long time for training and inference. (2) Learning deep models is difficult because of the poor conditioning of the matrices that parametrize the model. These difficulties are especially problematic to recurrent neural networks. Indeed, the number of distinct parameters in RNNs grows as the square of the size of the hidden state conversely to convolutional networks which enjoy weight sharing. Moreover, poor conditioning of the recurrent matrices results in the gradients to explode or vanish exponentially fast along the time horizon. This problem prevents RNN from capturing long-term dependencies (Hochreiter, 1991; Bengio et al., 1994).

There exists an extensive body of literature addressing over-parametrization in neural networks. LeCun et al. (1990) first studied the problem and proposed to remove unimportant weights in neural networks by exploiting the second order information. Several techniques which followed include low-rank decomposition (Denil et al., 2013), training a small network on the soft-targets predicted by a big pre-trained network (Ba and Caruana, 2014), low bit precision training (Courbariaux et al., 2014), hashing (Chen et al., 2015b), etc. A notable exception to the above approaches is the deep fried convnets (Yang et al., 2015) which explicitly parametrize the fully connected layers in a convnet with a computationally cheap and parameter-efficient structured linear operator, the Fastfood transform (Le et al., 2013). These techniques are primarily aimed at feed-forward fully connected networks and very few studies have focused on the particular case of recurrent networks (Arjovsky et al., 2016).

The problem of vanishing and exploding gradients has also received significant attention. Hochreiter and Schmidhuber (1997) proposed an effective gating mechanism in their seminal work on LSTMs. Later, this technique was adopted by other models such as the Gated Recurrent Units (GRU) (Chung et al., 2015) and the Highway networks (Srivastava et al., 2015) for recurrent and feed-forward neural networks respectively. Other popular strategies include gradient clipping (Pascanu et al., 2013), and orthogonal initialisation of the recurrent weights (Le et al., 2015). More recently (Arjovsky et al., 2016) proposed to use a unitary recurrent weight matrix. The use of norm preserving unitary maps prevents the gradients from exploding or vanishing, and thus helps to capture long-term dependencies. The resulting model called unitary RNN (uRNN) is computationally efficient since it only explores a small subset of general unitary matrices. Unfortunately, since uRNN can only span a reduced subset of unitary matrices their expressive power is limited (Wisdom et al., 2016). We denote this restricted capacity unitary RNN as RC uRNN. Full capacity unitary RNN (FC uRNN) (Wisdom et al., 2016) proposed to overcome this issue by parameterising the recurrent matrix with a full dimensional unitary matrix, hence sacrificing computational efficiency. Indeed, FC uRNN requires a computationally expensive projection step which takes $\mathcal{O}(N^3)$ time (N being the size of the hidden state) at each step of the stochastic optimization to maintain the unitary

constraint on the recurrent matrix. Mhammedi et al. (2016) in their orthogonal RNN (oRNN) avoided the expensive projection step in FC uRNN by parametrizing the orthogonal matrices using Householder reflection vectors. Their parametrization allows a fine-grained control over the number of parameters by choosing the number of Householder reflection vectors. When the number of these vectors approaches N, this parametrization spans the full reflection set, which is one of the disconnected subset of the full orthogonal set. Jing et al. (2017) also presented a way of parametrizing unitary matrices which allows fine-grained control on the number of parameters. This work, called Efficient Unitary RNN (EURNN), exploits the continuity of the unitary set to have a tunable parametrization ranging from a subset to the full unitary set.

Although the idea of parametrizing recurrent weight matrices with a strict unitary linear operator is appealing, it suffers from several issues: (1) Strict unitary constraints severely restrict the search space of the model, thus making the learning process unstable. (2) Strict unitary constraints make forgetting irrelevant information difficult. While this may not be an issue for problems with non-vanishing long term influence, it causes failure when dealing with real world problems that have vanishing long term influence. Henaff et al. (2016) have previously pointed out that the good performance of strict unitary models on certain synthetic problems is because it exploits the biases in these data-sets which favors a unitary recurrent map and these models may not generalize well to real world data-sets. More recently Vorontsov et al. (2017) have also studied this problem of unitary RNNs and the authors found out that relaxing the strict unitary constraint on the recurrent matrix to a soft unitary constraint improved the convergence speed as well as the generalization performance.

Our motivation is to address the problems of existing recurrent networks mentioned above. We present a new model called Kronecker Recurrent Units (KRU). At the heart of KRU is the use of Kronecker factored recurrent matrices which provide an elegant way to adjust the number of parameters to the problem at hand. This factorization allows us to finely modulate the number of parameters required to encode $N \times N$ matrices, from $\mathcal{O}(\log(N))$ when using factors of size 2×2, to $\mathcal{O}(N^2)$ parameters when using a single factor of the size of the matrix itself. We tackle the vanishing and exploding gradient problem through a soft unitary constraint **??** (Henaff et al., 2016; Cisse et al., 2017b; Vorontsov et al., 2017). Thanks to the properties of Kronecker matrices (Van Loan, 2000), this constraint can be enforced efficiently. Please note that KRU can readily be plugged into vanilla real space RNN, LSTM and other variants in place of standard recurrent matrices. However in the case of LSTMs we do not need to explicitly enforce the approximate orthogonality constraints as the gating mechanism is designed to prevent vanishing and exploding gradients. Our experimental results on seven standard data-sets reveal that KRU and KRU variants of real space RNN and LSTM can reduce the number of parameters drastically (hence the training and inference time) without trading the statistical performance. Our core contribution in this work is a flexible, parameter efficient and expressive recurrent neural network model which is robust to vanishing and exploding gradient problem.

The chapter is organized as follows: in section 2 we restate the formalism of RNN and detail the core motivations for KRU. In section 3 we present the Kronecker recurrent units (KRU). We present our experimental findings in section 4 and section 5 concludes our work.

Table 4.1: Notation

D, N, M Input, hidden and output dimensions
$\mathbf{x}_t \in \mathbb{R}^D$ or $\mathbb{C}^D$, $\mathbf{h}_t \in \mathbb{C}^D$ Input and hidden state at time t
$\mathbf{y}_t \in \mathbb{R}^M$ or $\mathbb{C}^M$, $\hat{\mathbf{y}}_t \in \mathbb{R}^M$ or $\mathbb{C}^M$ Prediction targets and RNN predictions at time t
$\mathbf{U} \in \mathbb{C}^{N\times D}, \mathbf{W} \in \mathbb{C}^{N\times N}, \mathbf{V} \in \mathbb{C}^{M\times N}$ Input, recurrent amd output weight matrices
$\mathbf{b} \in \mathbb{R}^N$ or $\mathbb{C}^N, \mathbf{c} \in \mathbb{R}^M$ or $\mathbb{C}^M$ Hidden and output bias
$\sigma(.), \mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$ Point-wise non-linear activation function and the loss function

4.2 Recurrent neural network formalism

Table 4.1 summarizes some notation that we use in this chapter. We consider the field to be complex rather than real numbers. We will motivate the choice of complex numbers later in this section. Consider a standard recurrent neural network (Elman, 1990). Given a sequence of T input vectors: $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$, at a time step t the RNN performs the following:

$$\mathbf{h}_t = \sigma(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t + \mathbf{b}) \tag{4.1}$$

$$\hat{\mathbf{y}}_t = \mathbf{V}\mathbf{h}_t + \mathbf{c}, \tag{4.2}$$

where $\hat{\mathbf{y}}_t$ is the predicted value at time step t. An illustration of a recurrent neural network unrolled along the time is shown in Figure **??**.

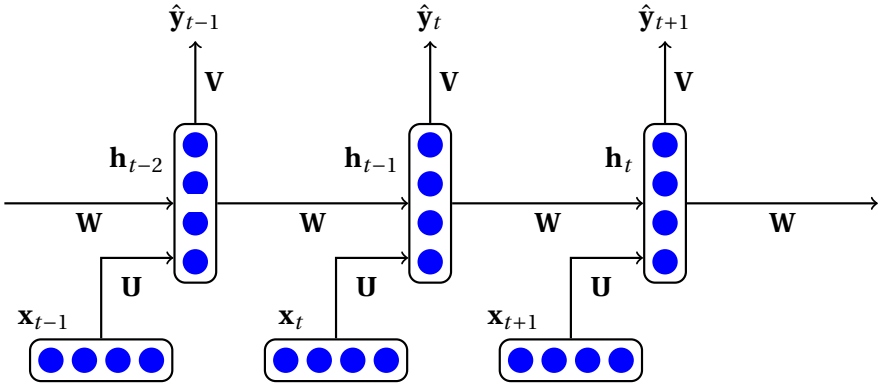


Figure 4.1: A slice of a recurrent neural network unrolled along the time. RNN is a very deep network along the time with shared parameters. At time step t RNN receives the input $\mathbf{x}_t$ and this input is encoded via the input to hidden matrix $\mathbf{U}$. This encoded input is combined with the information from the previous hidden state $\mathbf{h}_{t-1}$ using the recurrent weight matrix $\mathbf{W}$ to get the next hidden state. The information from the hidden state is extracted using the hidden to output matrix $\mathbf{V}$ also called decoder to predict the targets $\hat{\mathbf{y}}_t$.

4.2.1 Over parametrization and computational efficiency

The total number of parameters in an RNN is $c(DN + N^2 + N + M + MN)$, where c is 1 for real and 2 for complex parametrizations. As we can see, the number of parameters grows quadratically with the hidden dimension, *i.e.*, $\mathcal{O}(N^2)$. We show in the experiments that this quadratic growth is an over parametrization for many real world problems. Moreover, it has a direct impact on the computational efficiency of RNNs because the evaluation of $\mathbf{W}\mathbf{h}_{t-1}$ takes $\mathcal{O}(N^2)$ time and it recursively depends on previous hidden states. However, other components $\mathbf{U}\mathbf{x}_t$ and $\mathbf{V}\mathbf{h}_t$ can usually be computed efficiently by a single matrix-matrix multiplication for each of the components. That is, we can perform $\mathbf{U}[\mathbf{x}_1, \ldots, \mathbf{x}_T]$ and $\mathbf{V}[\mathbf{h}_1, \ldots, \mathbf{h}_T]$, which is efficient using modern BLAS libraries. So to summarize, if we can control the number of parameters in the recurrent matrix $\mathbf{W}$, then we can control the computational efficiency.

4.2.2 Poor conditioning implies gradients explode or vanish

The vanishing and exploding gradient problem refers to the decay or growth of the partial derivative of the loss $\mathcal{L}(.)$ with respect to the hidden state $\mathbf{h}_t$ *i.e.* $\frac{\partial \mathcal{L}}{\partial \mathbf{h}_t}$ as the number of time steps T grows (Arjovsky et al., 2016). By the application of the chain rule, the following can be shown (Arjovsky et al., 2016):

$$\left\| \frac{\partial \mathcal{L}}{\partial \mathbf{h}_t} \right\| \leq \|\mathbf{W}\|^{T-t}. \tag{4.3}$$

The derivation of the above inequality (4.3) is shown in appendix C.1. From Equation 4.3, it is clear that if the absolute value of the eigenvalues of $\mathbf{W}$ deviates from 1 then $\frac{\partial \mathcal{L}}{\partial \mathbf{h}_t}$ may explode or vanish exponentially fast with respect to $T-t$. So a strategy to prevent vanishing and exploding gradient is to control the spectrum of $\mathbf{W}$.

4.2.3 Why complex field?

Although Arjovsky et al. (2016) and Wisdom et al. (2016) use complex valued networks with unitary constraints on the recurrent matrix, the motivations for such models are not clear. We give a simple but compelling reason for complex-valued recurrent networks.

The absolute value of the determinant of a unitary matrix is 1. Hence in the real space, the set of all unitary (orthogonal) matrices have a determinant of 1 or -1, *i.e.*, the set of all rotations and reflections respectively. Since the determinant is a continuous function, the unitary set in real space is disconnected. Consequently, with real-valued networks we cannot span the full unitary set using the standard continuous optimization procedures. On the contrary, the unitary set is connected in the complex space as its determinants are the points on the unit circle and we do not have this issue.

As we mentioned in the introduction Jing et al. (2017) use this continuity of unitary space to

have a tunable continuous parametrization ranging from subspace to full unitary space. Any continuous parametrization in real space can only span a subset of the full orthogonal set. For example, the Householder parametrization (Mhammedi et al., 2016) suffers from this issue.

4.3 Kronecker recurrent units (KRU)

We consider parametrizing the recurrent matrix $\mathbf{W}$ as a Kronecker product of F matrices $\mathbf{W}_1, \ldots, \mathbf{W}_F$,

$$\mathbf{W} = \mathbf{W}_1 \otimes \cdots \otimes \mathbf{W}_F = \otimes_{f=1}^{F} \mathbf{W}_f. \tag{4.4}$$

Where each $\mathbf{W}_f \in \mathbb{C}^{P_f \times Q_f}$ and $\prod_{f=1}^{F} P_f = \prod_{f=1}^{F} Q_f = N$. $\mathbf{W}_f$'s are called as Kronecker factors.

To illustrate the Kronecker product of matrices, let us consider the simple case where $\forall_f \{P_f = Q_f = 2\}$. This implies $F = \log_2 N$, and $\mathbf{W}$ is recursevly defined as follows:

$$\mathbf{W} = \otimes_{f=1}^{\log_2 N} \mathbf{W}_f = \begin{bmatrix} \mathbf{w}_1(1,1) & \mathbf{w}_1(1,2) \\ \mathbf{w}_1(2,1) & \mathbf{w}_1(2,2) \end{bmatrix} \otimes_{f=2}^{\log_2 N} \mathbf{W}_f, \tag{4.5}$$

$$= \begin{bmatrix} \mathbf{w}_1(1,1)\mathbf{W}_2 & \mathbf{w}_1(1,2)\mathbf{W}_2 \\ \mathbf{w}_1(2,1)\mathbf{W}_2 & \mathbf{w}_1(2,2)\mathbf{W}_2 \end{bmatrix} \otimes_{f=3}^{\log_2 N} \mathbf{W}_f. \tag{4.6}$$

When $\forall_f \{P_f = Q_f = 2\}$ the number of parameters is $8 \log_2 N$ and the time complexity of hidden state computation is $\mathcal{O}(N \log_2 N)$. When $\forall_f \{P_f = Q_f = N\}$ then $F = 1$ and we will recover standard complex valued recurrent neural network. We can span every Kronecker representation in between by choosing the number of factors and the size of each factor. In other words, the number of Kronecker factors and the size of each factor give us fine-grained control over the number of parameters and hence over the computational efficiency. This strategy allows us to design models with the appropriate trade-off between computational budget and statistical performance. All the existing models lack this flexibility.

The idea of using Kronecker factorization for approximating Fisher matrix in the context of natural gradient methods has recently received much attention. The algorithm was originally presented in Martens and Grosse (2015) and was later extended to convolutional layers (Grosse and Martens, 2016), distributed second order optimization (Ba et al., 2016) and for deep reinforcement learning (Wu et al., 2017). However Kronecker matrices have not been well explored as learnable parameters except by Zhang et al. (2015) who used its spectral property for fast orthogonal projections and Zhou et al. (2015) who used it as a layer in convolutional neural networks.

4.3.1 Soft unitary constraint

Poor conditioning results in vanishing or exploding gradients. Unfortunately, the standard solution of optimizing on the strict unitary set suffers from the retention of noise over time. Indeed, the small eigenvalues of the recurrent matrix can represent a truly vanishing long-term influence on the particular problem and in that sense, there can be good or bad vanishing gradients. Consequently, enforcing strict unitary constraint (forcing the network to never forget) can be a bad strategy. A simple solution to get the best of both worlds is to enforce the unitary constraint approximately by using the following regularization:

$$\left\| \mathbf{W}_f^H \mathbf{W}_f - \mathbf{I} \right\|^2, \forall f \in \{1, \ldots, F\} \tag{4.7}$$

Please note that these constraints are enforced on each factor of the Kronecker factored recurrent matrix. This procedure is computationally very efficient since the size of each factor is typically small. It suffices to do so because if each of the Kronecker factors $\{\mathbf{W}_1, \ldots, \mathbf{W}_F\}$ are unitary then the full matrix $\mathbf{W}$ is unitary (Van Loan, 2000) and if each of the factors are approximately unitary then the full matrix is approximately unitary.

This type of regularizer has recently been exploited for real-valued models. Cisse et al. (2017b) showed that enforcing approximate orthogonality constraints on the weight matrices make the network robust to adversarial samples as well as improve the learning speed. As we have shown in the previous chapter, in metric learning (Jose and Fleuret, 2016) it better conditions the projection matrix thereby improving the robustness of stochastic gradient over a wide range of step sizes as well as the generalisation performance. Henaff et al. (2016) and Vorontsov et al. (2017) have also used this soft unitary constraints on standard RNN after identifying the problems with the strict unitary RNN models. However the computational complexity of naively applying this soft constraint is $\mathcal{O}(N^3)$. This is prohibitive for RNN with large hidden state unless one considers a Kronecker factorisation.

4.4 Experiments

Existing deep learning libraries such as Theano (Bergstra et al., 2011), Tensorflow (Abadi et al., 2016) and Pytorch (Paszke et al., 2017) do not support fast primitives for Kronecker products with arbitrary number of factors. So we wrote custom CUDA kernels for Kronecker forward and backward operations. All our models are implemented in C++. We use tanh as activation function for RNN, LSTM and our model KRU-LSTM. whereas RC uRNN, FC uRNN and KRU use complex rectified linear units (Arjovsky et al., 2016).

4.4.1 Copy memory problem

The copy memory problem (Hochreiter and Schmidhuber, 1997) tests the model's ability to recall a sequence after a long time gap. In this problem each sequence is of length $T + 20$

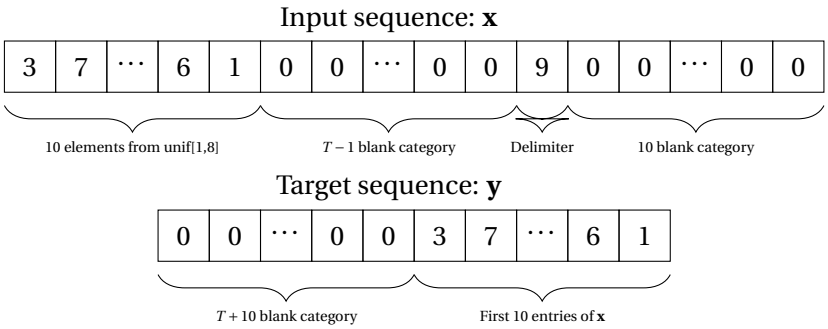


Figure 4.2: An illustration of the copy memory problem. $\mathbf{x}$ is the input sequence of length $T+20$ and $\mathbf{y}$ is the target sequence to be predicted. First 10 entries of $\mathbf{x}$ is filled from unif[1, 8] and the next $T-1$ is 0 - the 'blank' category, followed by 9 - 'the delimiter'. The goal of this task is to predict 'the blank category' for the first $T+10$ steps and as soon as 'the delimter' is encountered it should reproduce the first 10 entries of the input sequence x.

and each element in the sequence come from 10 classes $\{0,\ldots,9\}$. The first 10 elements are sampled uniformly with replacement from $\{1,\ldots,8\}$. The next $T-1$ elements are filled with 0, the 'blank' class followed by 9, the 'delimiter' and the remaining 10 elements are 'blank' category. The goal of the model is to output a sequence of $T+10$ blank categories followed by the 10 element sequence from the beginning of the input sequence. The expected average cross entropy for a memory-less strategy is $\frac{10\log 8}{T+20}$.

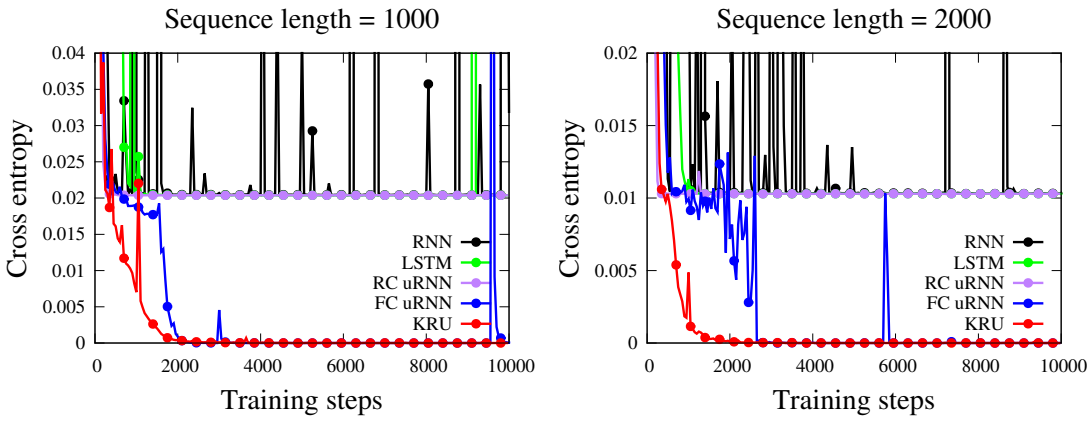


Figure 4.3: Learning curves on copy memory problem for T=1000 and T=2000.

Our experimental setup closely follows (Wisdom et al., 2016) which in turn follows (Arjovsky et al., 2016) but T extended to 1000 and 2000. Our model, KRU, uses a hidden dimension N of 128 with 2x2 Kronecker factors which corresponds to ≈5K parameters in total. We use an RNN of N = 128 (≈ 19K parameters) , LSTM of N = 128 (≈ 72K parameters), RC uRNN of N = 470 (≈ 21K parameters) (Wisdom et al., 2016) , FC uRNN of N = 128 (≈ 37K parameters). All the baseline models are deliberately chosen to have slighly more parameters than KRU. Following Wisdom et al. (2016); Arjovsky et al. (2016), we choose the training and test set size

to be 100K and 10K respectively. All the models were trained using RMSprop with a learning rate of 1e−3, decay of 0.9 and a batch size of 20. For both the settings $T = 1000$ and $T = 2000$, KRU converges to zero average cross entropy faster than FC uRNN. All the other baselines are stuck at the memory-less cross entropy.

The results are shown in Figure 4.3. For this problem we do not learn the recurrent matrix of KRU, We initialize it to a random unitary matrix and just learn the input to hidden matrix, hidden to output matrix and the bias. We found out that this strategy already solves the problem faster than all other methods. Our model in this case is similar to a parametrized echo state networks (ESN). ESNs are known to be able to learn long-term dependencies if they are properly initialized (Jaeger, 2001). We argue that this data-set is not an ideal benchmark for evaluating RNNs in capturing long term dependencies. Just a unitary initialization of the recurrent matrix would solve the problem.

4.4.2 Adding problem

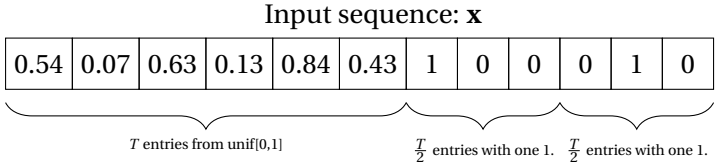


Figure 4.4: Adding problem of sequence length $T = 6$. It is consists of two sequences of length $T = 6$. The first sequence entries ($T = 6$) are sampled uniformly at random from [0,1]. In the next sequence exactly 2 of the entries are 1, each of them located uniformly at random in each half of the sequence. The goal of the adding problem is to predict sum of the numbers in the first sequence corresponding to the marked two locations after seeing the whole sequence. In this example the marked locations in the second sequence are 7 and 11, which corresponds to 1 and 5 in the first sequence so the prediction target is $y = x[1] + x[5] = 0.54 + 0.84 = 1.38$.

Following Arjovsky et al. (2016) we describe the adding problem (Hochreiter and Schmidhuber, 1997). Each input vector is composed of two sequences of length T. The first sequence is sampled from $\mathcal{U}[0,1]$. In the second sequence exactly two of the entries are 1, the 'marker' and the remaining are 0. The first 1 is located uniformly at random in the first half of the sequence and the other 1 is located again uniformly at random in the other half of the sequence. The network's goal is to predict the sum of the numbers from the first sequence corresponding to the marked locations in the second sequence.

We evaluate four settings as in (Arjovsky et al., 2016) with T=100, T=200, T=400, and T=750. For all four settings, KRU uses a hidden dimension N of 512 with 2x2 Kronecker factors which corresponds to ≈3K parameters in total. We use an RNN of $N = 128$ (≈ 17K parameters) , LSTM of $N = 128$ (≈ 67K parameters), RC uRNN of $N = 512$ (≈ 7K parameters) , FC uRNN of $N = 128$ (≈ 33K parameters). The train and test set sizes are chosen to be 100K and 10K respectively.

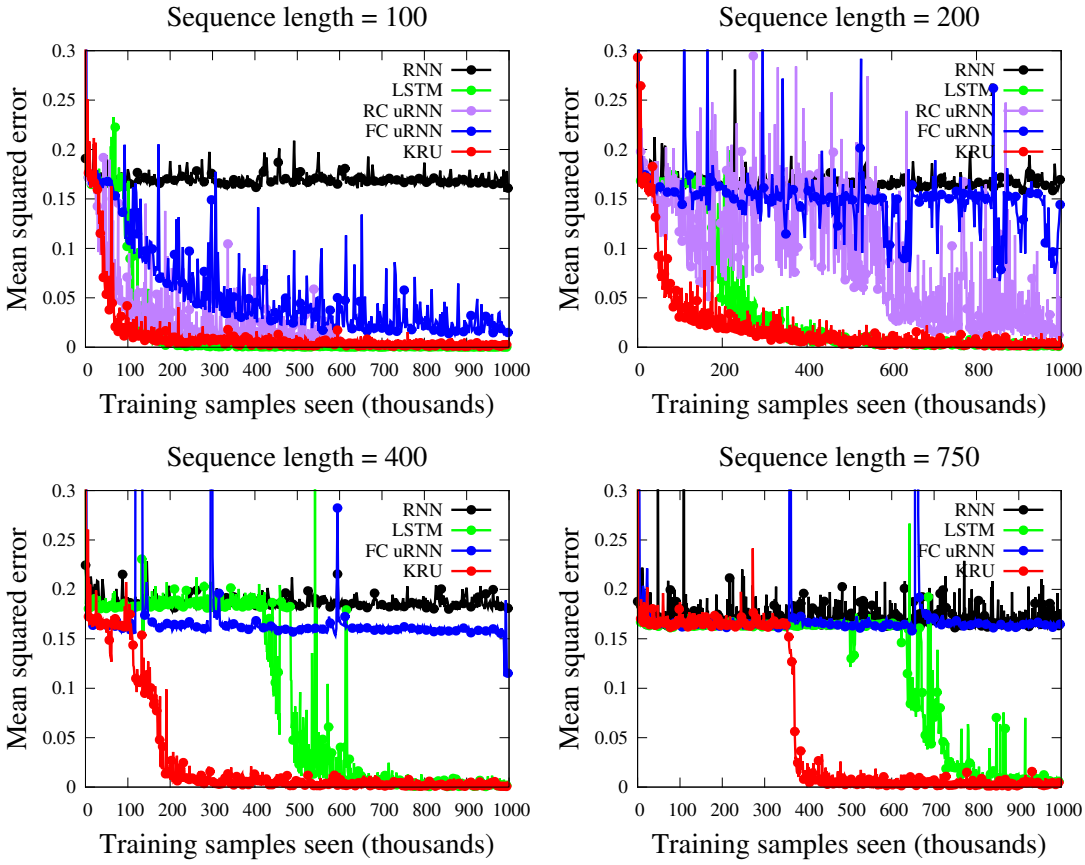


Figure 4.5: Results on adding problem for T=100, T=200, T=400 and T=750. KRU consistently outperforms the baselines on all the settings with fewer parameters.

All the models were trained using RMSprop with a learning rate of 1e−3 and a batch size of 20 or 50 with the best results being reported here.

The results are presented in Figure 4.5. KRU converges faster than all other baselines even though it has much fewer parameters. This shows the effectiveness of soft unitary constraint which controls the flow of gradients through very long time steps and thus deciding what to forget and remember in an adaptive way. LSTM also converges to the solution and this is achieved through its gating mechanism which controls the flow of the gradients and thus the long term influence. However LSTM has 10 times more parameters than KRU. Both RC uRNN and FC uRNN converges for $T = 100$ but as we can observe, the learning is not stable. The reason for this is that RC uRNN and FC uRNN retain noise since they are strict unitary models. Please note that we do not evaluate RC uRNN for $T = 400$ and $T = 750$ because we found out that the learning is unstable for this model and is often diverging.

4.4.3 Pixel by pixel MNIST

As outlined by Le et al. (2015), We evaluate the Pixel by pixel MNIST task. MNIST digits are shown to the network pixel by pixel and the goal is to predict the class of the digit after seeing all the pixels one by one. We consider two tasks: (1) Pixels are read from left to right from top or bottom and (2) Pixels are randomly permuted before being shown to the network. The sequence length for these tasks is $T = 28 \times 28 = 784$. The size of the MNIST training set is 60K among which we choose 5K as the validation set. The models are trained on the remaining 55K points. The model which gave the best validation accuracy is chosen for test set evaluation. All the models are trained using RMSprop with a learning rate of 1e−3 and a decay of 0.9.

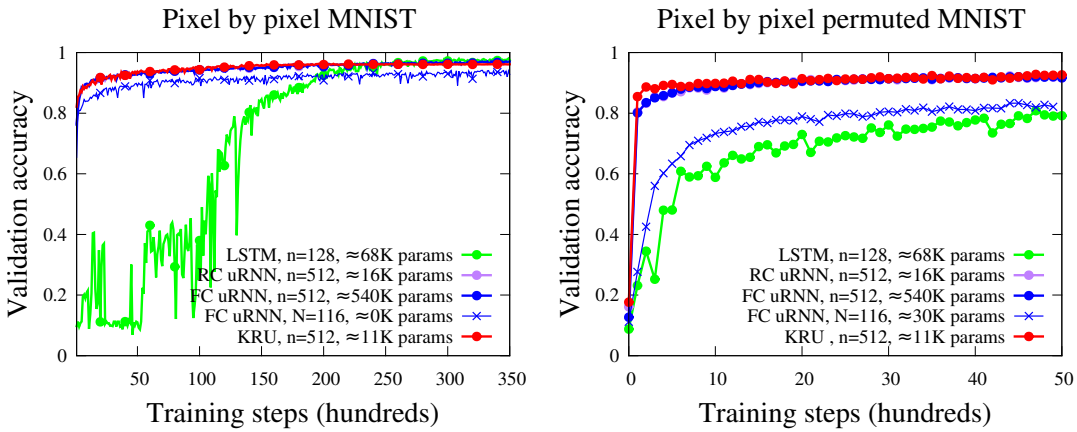


Figure 4.6: Validation accuracy on pixel by pixel MNIST and permuted MNIST class prediction as the learning progresses.

Table 4.2: KRU achieves state of the art performance on pixel by pixel permuted MNIST while having up to four orders of magnitude less parameters than other models.

Model	n	# Parameters		Unpermuted accuracy		Permuted accuracy	
		Total	Recurrent	Valid.	Test	Valid.	Test
LSTM (Arjovsky et al., 2016)	128	≈68K	≈65K	98.1	**97.8**	91.7	91.3
RC uRNN (Wisdom et al., 2016)	512	≈16K	≈3.6K	97.9	97.5	94.2	93.3
FC uRNN (Wisdom et al., 2016)	512	≈540K	≈524K	97.5	96.9	94.7	94.1
FC uRNN (Wisdom et al., 2016)	116	≈30K	≈27K	92.7	92.8	92.2	92.1
oRNN (Mhammedi et al., 2016)	256	≈11K	≈8K	97.0	97.2	-	-
EURNN (Jing et al., 2017)	1024	≈13K	≈4K	-	-	94.0	93.7
KRU	512	≈11K	72	96.6	96.4	94.7	**94.5**

The results are summarized in Table 4.2 and Figure 4.6. On the unpermuted task LSTM achieve the state of the art performance even though the convergence speed is slow. Recently a low rank plus diagonal gated recurrent unit (LRD GRU) (Barone, 2016) has shown to achieves 94.7 accuracy on permuted mnist with 41.2K parameters whereas KRU achieves 94.5 with just 12K parameters i.e KRU has 3x parameters fewer than LRD GRU. Please also note that KRU is a simple model without a gating mechanism. KRU can be straightforwardly plugged into LSTM and GRU to exploit the additional benefits of the gating mechanism which we will show in the

next experiments with a KRU-LSTM.

4.4.4 Character level language modelling on Penn TreeBank (PTB)

We now consider character level language modeling on the Penn TreeBank data-set (Marcus et al., 1993). Penn TreeBank is composed of 5017K characters in the training set, 393K characters in the validation set and 442 characters in the test set. The size of the vocabulary was limited to 10K most frequently occurring words and the rest of the words are replaced by a special <UNK> character (Mikolov, 2012). The total number of unique characters in the data-set is 50, including the special <UNK> character.

Table 4.3: Performance in BPC of KRU variants and other models for character level language modeling on the Penn TreeBank data-set. KRU has fewer parameters in the recurrent matrix which significantly bring down training and inference time.

Model	N	# Parameters		Valid. BPC	Test BPC
		Total	Recurrent		
RNN	300	≈120K	90K	1.65	1.60
LSTM	150	≈127K	90K	1.63	1.59
oRNN (Mhammedi et al., 2016)	512	≈183K	≈130K	1.73	1.68
KRU	411	≈120K	≈38K	1.65	1.60
RNN	600	≈420K	360K	1.56	1.51
LSTM	300	≈435K	360K	**1.50**	**1.45**
KRU	993	≈418K	≈220K	1.53	1.48
KRU-LSTM	500	≈377K	≈250K	1.53	1.47

All our models were trained for 50 epochs with a batch size of 50 and using ADAM (Kingma and Ba, 2014b). We use a learning rate of 1e−3 which was found through cross-validation with default beta parameters (Kingma and Ba, 2014b). If we do not see an improvement in the validation bits per character (BPC) after each epoch then the learning rate is decreased by 0.30. Back-propagation through time (BPTT) is unrolled for 30 time frames on this task.

We did two sets of experiments to have a fair evaluation with the models whose results were available for a particular parameter setting (Mhammedi et al., 2016) and also to see how the performance evolves as the number of parameters is increased. We present our results in Table 4.3. We observe that the strict orthogonal model, oRNN fails to generalize as well as other models even with a high capacity recurrent matrix. KRU and KRU-LSTM perform very close to RNN and LSTM with fewer parameters in the recurrent matrix. Please recall that the computational bottleneck in RNN is the computation of hidden states (4.2.1) and thus having fewer parameters in the recurrent matrix can significantly reduce the training and inference time.

Recently HyperNetworks (Ha et al., 2016) have shown to achieve the state of the art performance of 1.265 and 1.219 BPC on the PTB test set with 4.91 and 14.41 million parameters

respectively. This is respectively 38 and 13 times more parameters than the KRU-LSTM model which achieves 1.47 test BPC. Also Recurrent Highway Networks (RHN) (Zilly et al., 2016) proved to be a promising model for learning very deep recurrent neural networks. Running experiments, and in particular exploring meta-parameters, with models of that size, requires unfortunately computational means beyond what was at our disposal for this work. However, there is no reason that the consistent behavior and improvement observed on the other reference baselines would not generalize to that type of large-scale models.

4.4.5 Polyphonic music modeling

We exactly follow the experimental framework of Chung et al. (2014) for Polyphonic music modelling (Boulanger-Lewandowski et al., 2012) on two data-sets: JSB Chorales and Piano-midi. Similar to (Chung et al., 2014) our main objective here is to have a fair evaluation of different recurrent neural networks. We took the baseline RNN and LSTM models of (Chung et al., 2014) whose model sizes were chosen to be small enough to avoid over-fitting. We choose the model size of KRU and KRU-LSTM in such a way that it has fewer parameters compared to the baselines. As we can see in Table 4.4 both our models (KRU and KRU-LSTM) over-fit less and generalizes better. We also present the wall-clock running time of different methods in Figure 4.7.

Table 4.4: Average negative log-likelihood of KRU and KRU-LSTM compared to the baseline models.

Model	n	# Parameters		JSB Chorales		Piano-midi	
		Total	Recurrent	Train	Test	Train	Test
RNN (Chung et al., 2014)	100	≈20K	10K	8.82	9.10	5.64	9.03
LSTM (Chung et al., 2014)	36	≈20K	≈5.1K	8.15	8.67	6.49	9.03
KRU	100	≈10K	58	7.90	8.59	7.57	8.28
KRU-LSTM	45	≈19K	176	7.47	**8.54**	7.55	**8.18**

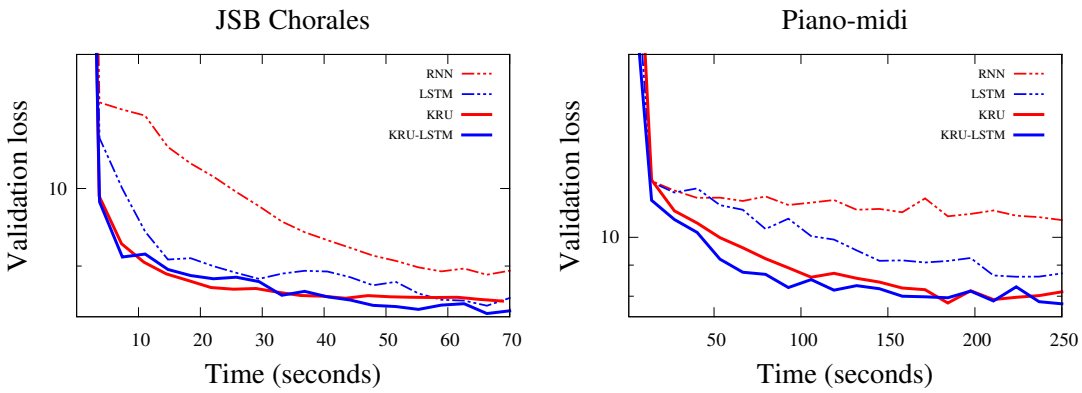
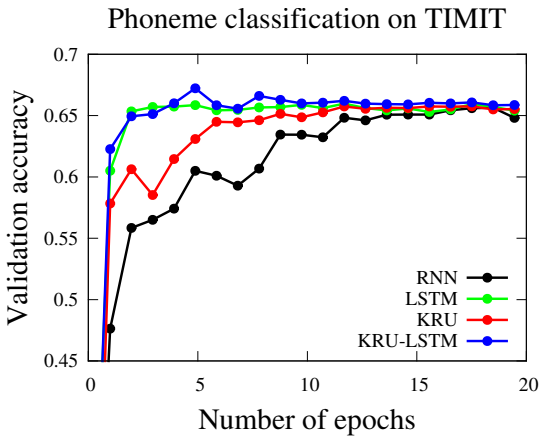


Figure 4.7: Wall clock training time on JSB Chorales and Piano-midi data-set.

4.4.6 Framewise phoneme classification on TIMIT

Framewise phoneme classification (Graves and Schmidhuber, 2005) is the problem of classifying the phoneme corresponding to a sound frame. We evaluate the models for this task on the real world TIMIT data-set (Garofolo et al., 1993). TIMIT contains a training set of 3696 utterances among which we use 184 as the validation set. The test set is composed of 1344 utterances. We extract 12 Mel-Frequency Cepstrum Coefficients (MFCC) (Mermelstein, 1976) from 26 filter banks and also the log energy per frame. We also concatenate the first derivative, resulting in a feature descriptor of dimension 26 per frame. The frame size is chosen to be 10ms and the window size is 25ms.



Model	N	# Parameters		Valid. accuracy	Test accuracy
		Total	Recurrent		
RNN	600	≈406K	360K	65.84	64.53
LSTM	300	≈406K	360K	65.99	64.56
KRU	2048	≈195K	16K	65.91	64.55
KRU-LSTM	2048	≈404	66K	**66.54**	**64.81**

Figure 4.8: KRU and KRU-LSTM performs better than the baseline models with far fewer parameters in the recurrent weight matrix on the challenging TIMIT data-set (Garofolo et al., 1993). This significantly bring down the training and inference time of RNNs. Both LSTM and KRU-LSTM converged within 5 epochs whereas RNN and KRU took 20 epochs. A similar result was obtained by (Graves and Schmidhuber, 2005) using RNN and LSTM with 4 times fewer parameters respectively than our models. However in their work the LSTM took 20 epochs to converge and the RNN took 70 epochs. We have also experimented with the same model size as that of (Graves and Schmidhuber, 2005) and have obtained very similar results as in the table but at the expense of longer training times.

The number of time steps to which back-propagation through time (BPTT) is unrolled corresponds to the length of each sequence. Since each sequence is of different length this implies that for each sample the BPTT steps are different. All the models are trained for 20 epochs with a batch size of 1 using ADAM with default parameters (Kingma and Ba, 2014b). The

learning rate was cross-validated for each of the models from $\eta \in \{1e{-}2, 1e{-}3, 1e{-}4\}$ and the best results are reported here. The best learning rate was found out to be $1e{-}3$ for all the models. Again if we do not observe a decrease in the validation error after each epoch, we decrease the learning rate by a factor of $\gamma \in \{1e{-}1, 2e{-}1, 3e{-}1\}$ which is again cross-validated. Figure 4.8 summarizes our results.

4.4.7 Influence of soft unitary constraints

Here we study the properties of soft unitary constraints on KRU. We use Polyphonic music modeling data-sets (Boulanger-Lewandowski et al., 2012): JSB Chorales and Piano-midi, as well as TIMIT data-set for this set of experiments. We varied the amplitude of soft unitary constraints from $1e-7$ to $1e-1$, the higher the amplitude the closer the recurrent matrix will be to the unitary set. All other hyper-parameters, such as the learning rate and the model size are fixed. We present our studies in Figure 4.9. As we increase the amplitude we can see that the recurrent matrix is getting better conditioned and the spectral norm or the spectral radius is approaching towards 1. As we can see that the validation performance can be improved using this simple soft unitary constraints. For JSB Chorales the best validation performance is achieved at an amplitude of $1e-2$, whereas for Piano-midi it is at $1e-1$.

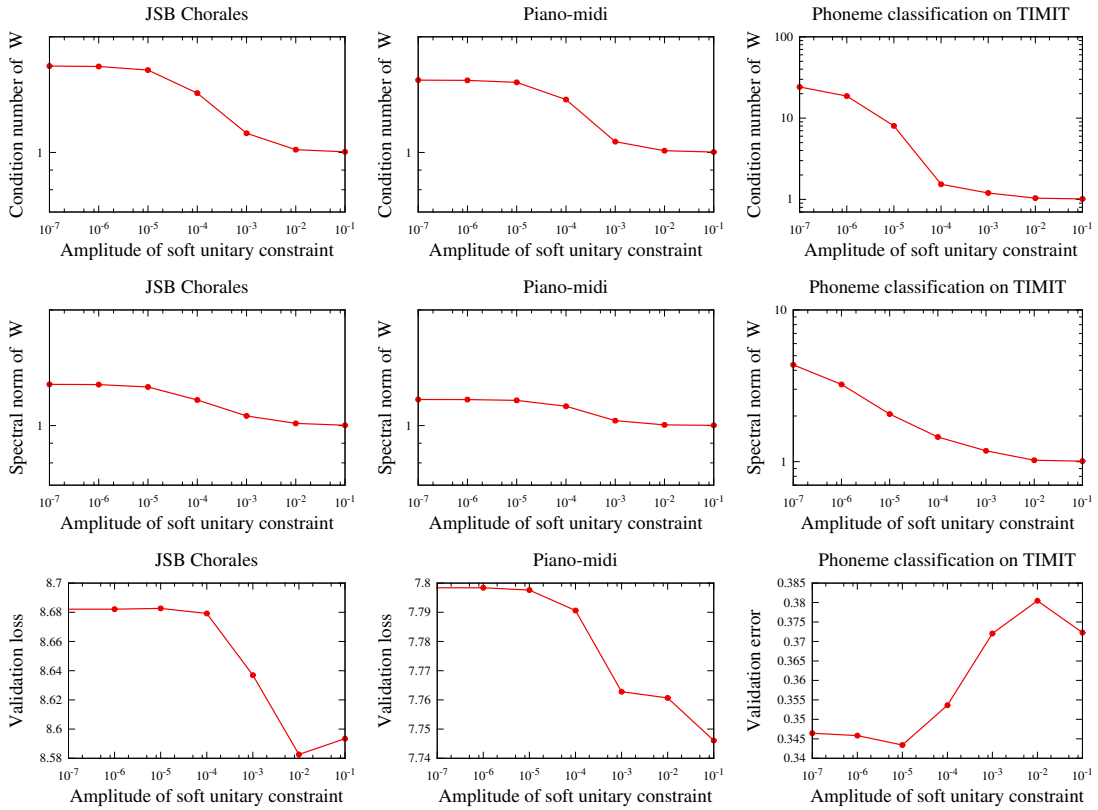


Figure 4.9: Analysis of soft unitary constraints on three data-sets. First, second and the third column presents JSB Chorales, Piano-midi and TIMIT data-sets respectively.

For TIMIT phoneme recognition problem, the best validation error is achieved at $1e-5$ but as we increase the amplitude further, the performance drops. This might be explained by a vanishing long-term influence that has to be forgotten. Our model achieve this by cross-validating the amplitude of soft unitary constraints. These experiments also reveals the problems of strict unitary models such as RC uRNN (Arjovsky et al., 2016), FC uRNN (Wisdom et al., 2016), oRNN (Mhammedi et al., 2016) and EURNN (Jing et al., 2017) that they suffer from the retention of noise from a vanishing long term influence and thus fails to generalize.

A popular heuristic strategy to avoid exploding gradients in RNNs and thereby making their training robust and stable is gradient clipping. Most of the state of the art RNN models use gradient clipping for training. Please note that we are not using gradient clipping with KRU. Our soft unitary constraints offer a principled alternative to gradient clipping.

Moreover Hardt et al. (2016) recently showed that gradient descent converges to the global optimizer of linear recurrent neural networks even though the learning problem is non-convex. The necessary condition for the global convergence guarantee requires that the spectral norm of recurrent matrix is bounded by 1. This seminal theoretical result also inspires to use regularizers which control the spectral norm of the recurrent matrix, such as the soft unitary constraints.

4.5 Discussion

We have presented a new recurrent neural network model based on a Kronecker factored recurrent matrix. Our reason for using a Kronecker factored recurrent matrix stems from its elegant algebraic and spectral properties. Kronecker matrices are neither low-rank nor block-diagonal but they are multi-scale like the FFT matrix. The Kronecker factorization provides a fine control over the model capacity and its algebraic properties enable us to design fast matrix multiplication algorithms. Its spectral properties allow us to efficiently enforce constraints like positive semi-definiteness, unitarity and stochasticity. As we have shown, we used the spectral properties to efficiently enforce a soft unitary constraint.

Our experimental results show that our approach out-perform classical methods which use $\mathcal{O}(N^2)$ parameters in the recurrent matrix. Maybe as important, these experiments show that both on toy problems (§ 4.4.1 and 4.4.2), and on real ones (§ 4.4.3, 4.4.4, , and § 4.4.6), while existing methods require tens of thousands of parameters in the recurrent matrix, competitive or better than state-of-the-art performance can be achieved with far fewer parameters in the recurrent weight matrix. This surprising result provides a new and counter-intuitive perspective on desirable memory-capable architectures: the state should remain of high dimension to allow the use of high-capacity networks to encode the input into the internal state, and to extract the predicted value, but the recurrent dynamic itself can, and should, be implemented with a low-capacity model.

From a practical standpoint, the idea in our method is applicable not only to vanilla recurrent

neural networks and LSTMS as we showed, but also to a variety of machine learning models such as feed-forward networks (Zhou et al., 2015) and boosting weak learners. Our future work encompasses exploring other machine learning models and dynamically increasing the capacity of the models on the fly during training to have a perfect balance between computational efficiency and sample complexity.

5 Importance Sampling Tree for Large-scale Empirical Expectation

Contents

This chapter presents the Importance Sampling Tree data structure (IST) and this is a joint work with Olivier Canévet (2017). Section 5.1, 5.2, 5.3 and 5.4 is a rephrasing of the Chapter 5 in the thesis, Canévet (2017) and the publication, Canévet et al. (2016). The experiments we report are different from Canévet (2017) except one synthetic problem which we reproduced the results using our own implementation. IST is based on the following publication:

> Olivier Canévet, Cijo Jose, and Francois Fleuret. Importance sampling tree for large-scale empirical expectation. In *International Conference on Machine Learning*, pages 1454–1462, 2016

5.1 Introduction

As we have seen in the Chapter 2, virtually every single machine learning algorithm using the ERM principle relies on data-based empirical expectations, either to estimate a loss, or the response of a predictor (see the examples in § 5.3). The larger the data-set, the more accurate the prediction, and many state-of-the-art results have been obtained by enriching the already very large sets using data augmentation, resulting in hundreds of millions of labeled samples Krizhevsky et al. (2012a).

An empirical expectation takes the form of a sum of a quantity evaluated on many data-points, and in practice most of the terms in the sum are negligible. In training it is because most of the samples are far from the decision boundary between populations, and get a "trivially correct answer". Figure 5.1 illustrates this scenario in the training of a deep convolutional neural network on CIFAR10 data-set enriched with data augmentation. In test it is because the prediction on a test point is modulated only by its immediate neighbors in the training set. For example in the case of Gaussian kernel machines trained on large data-sets, even though the model has a very large number of support vectors, in many cases only few of them will actually matter in the final decision score.

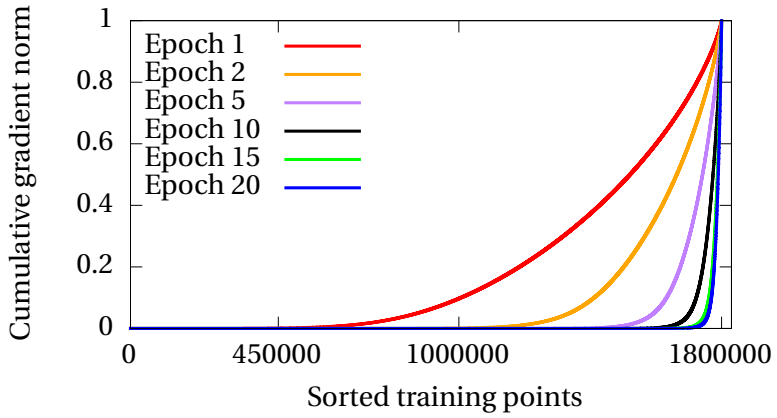


Figure 5.1: Cumulative gradient norms of the training points on CIFAR10 dataset enriched with data augmentation, using a state of the art convolutional neural network as learning algorithm. As the learning progresses most of the contribution to the empirical expectation comes from very few number of points.

Despite this well known state of affairs, algorithms still rely on an exhaustive loop through the samples. Some approaches have been developed to prioritize samples *after they have already been seen* (Kalal et al., 2008; Fleuret and Geman, 2008), or in subsets sampled uniformly (Loosli et al., 2007). But they do not use data structures given *a priori* over the said samples, combining it with statistical observations made over those already observed to reject sets without looking at them.

So if we have an efficient mechanism which gives us the set of points which contribute most to the empirical expectation then we can focus the computation on this set of points, thereby saving computation time. This problem of identifying important points in a set is related to the choice of an optimal move in a strategy game. This has been tackled traditionally with branch-and-bound approaches, going down the tree of possible choices, and discarding sub-trees that can be proven to be bad. But the branch and bound techniques find their limit in games of very large combinatorial complexity and this setting is generally tackled by using data structures which can sample and optimize the sampling over the configuration at the same time. For example, state-of-the-art performance for the game of Go is obtained with Monte-Carlo Tree Search (Gelly et al., 2006) which samples the optimal move and at the same time optimizes over sampling configurations for the future moves.

In this chapter we present a data structure, dubbed IST for "Importance Sampling Tree" for efficiently estimating an empirical expectation. IST organizes the data into a tree structure using the given metric on the data. IST at the same time, samples the data points, provides a correcting factor to compensate for its sampling bias, and optimizes inner statistical estimates to improve sampling over time.

5.2 Related work

Batch learning becomes infeasible when it comes to training predictors on very large data-sets. Therefore, sub-sampling appears to be one solution to make it practical and as we discussed in Chapter 2, stochastic gradient descent (SGD) is the default choice when we have large data-sets. It has been shown that a smart sampling, as opposed to a uniform sampling with SGD, has an impact on the performance of the final classifier.

Bordes et al. (2005) presented LASVM, an online algorithm with importance sampling to train kernel support vector machines. They show that importance sampling reduces the training time and also results in models which are compact with fewer support vectors, while maintaining equal or better prediction accuracy compared to standard algorithms.

Recently, importance sampling has been theoretically studied in the context of stochastic gradient descent (SGD) algorithms (Zhao and Zhang, 2014). Their key observation is that, though sampling observations uniformly at random from the training set results in a stochastic gradient which is an unbiased estimate of the true gradient, the resulting estimator may have high variance. In order to mitigate the convergence issue with high variance they propose an importance sampling scheme. Their theoretical results show that under certain conditions, importance sampling can improve the convergence rate of SGD algorithms.

In the context of boosting the procedures proposed by Fleuret and Geman (2008), and Kalal et al. (2008) select samples to train a weak learner based on their boosting weights. The classifier is thus presented with highly misclassified samples, with a high individual weight, but also with representatives of populations of low weighted individual samples which have a

Table 5.1: Notation

$w_m \in \mathbb{R}_+$ Positive weights we want to approximate through sampling
$\mathcal{T}, \mathcal{L} \subset \mathcal{T}$ Nodes and leaves of IST
D The depth of the tree
$c_0(x), c_1(x)$ Children of node x
U Minimum number of observations to look before biasing the θ_x
$\mathcal{L}(x) \subset \mathcal{L}$ Set of leaves of the sub-tree whose root is x
$n(x) \in \{1, \ldots, M\}$ Index of the weight at leaf x
$\Theta = (\theta_x)_{x \in \mathcal{T} \setminus \mathcal{L}}$ Probabilities to chose the right sub-tree at each node during sampling
$\mu_\Theta(m; x)$ Probability to reach leaf m given that the sampling passes through node x
$w(x) = \sum_{y \in \mathcal{L}(x)} w_{n(y)}$ Sum of the weights of the leaves in the sub-tree whose root is x
$v(x)$ Number of times the sampling has been through node x
$s(x)$ Sum of the individual estimates of $w(x)$
$\hat{w}_x = s(x)/v(x)$ Estimate of $w(x)$
$\mathcal{S}(x)$ Statistics accumulated at node x

large cumulative weight.

Another approach designed for boosting is the reservoir boosting (Lefakis and Fleuret, 2013). Reservoir boosting is designed for the case when the learner cannot use all the samples to choose the next weak learner and can only use a subset called the reservoir because of the memory constraint. At each boosting round, the reservoir is filled efficiently with a sample which maximizes the information of the entire set as projected onto the classifier space.

More generally a sampling approach called Monte Carlo Tree Search (Browne et al., 2012) has gain much interest in the past years for the tremendous improvement for games such as Computer Go. The purpose of MCTS is to find the optimal solution in a potentially huge space organized as a tree by sampling this tree. The tree is traversed from top to bottom by recursively applying a multi-armed bandit on the children of the current node until reaching a leaf. A reward related to the optimal solution is then obtained and the outcome is back-propagated up the root. The next sampling will use the accumulated statistics to prioritize the sampling towards promising branches and eventually find the optimal value.

Contrary to what is the core purpose of MCTS, we are not interested in finding the best leaf or leaves, but to sample among all the leaves, according to their weights. These two objectives are very distinct when an important fraction of the total weight comes from a large population of low-weighting leaves.

5.3 Weighted averages in machine learning

Given M positive weights $w_m \in \mathbb{R}_+,\ m = 1,\dots,M$ and a function $f : \{1,\dots,M\} \to \mathbb{R}^D$, we are interested in the weighted average

$$\sum_{m=1}^{M} w_m f(m) \tag{5.1}$$

when M is too large to allow an exhaustive visit of the weights. Our interest in weighted average stems from the fact most of the important quantities in machine learning can be expressed as an empirical expectation, such as

– The edge of a weak-learner in Adaboost

$$\sum_m \underbrace{\exp(-y_m \psi(\mathbf{x}_m))}_{w_m} \underbrace{y_m h(\mathbf{x}_m)}_{f(m)}. \tag{5.2}$$

– The gradient for training a neural network

$$\sum_m \nabla_\alpha \mathcal{L}(\psi(\mathbf{x}_m;\alpha), y_m) = \sum_m \underbrace{\|\nabla_\alpha \mathcal{L}(\psi(\mathbf{x}_m;\alpha), y_m)\|}_{w_m} \underbrace{\frac{\nabla_\alpha \mathcal{L}(\psi(\mathbf{x}_m;\alpha), y_m)}{\|\nabla_\alpha \mathcal{L}(\psi(\mathbf{x}_m;\alpha), y_m)\|}}_{f(m)}. \tag{5.3}$$

– The prediction function of kernel machines

$$\sum_m \alpha_m y_m k(\mathbf{x}_m, \mathbf{x}) = \sum_n \underbrace{\alpha_m k(\mathbf{x}_m, \mathbf{x})}_{w_m} \underbrace{y_m}_{f(m)}. \tag{5.4}$$

In these examples, a weight can be interpreted as the "importance" of a sample. This importance is large either because the said sample impacts a lot in the training using ERM(induces a large loss, or a large gradient norm), or because it impacts a lot the predicted value (has a large kernel value with the evaluation point). More importantly, in many practical situations, the vast majority of them are negligible or more precisely, large populations of samples contribute to a fraction of the total of the cumulative weights. We aim at devising an approach – relying on a prior tree structure on the weights – that (1) balances computation proportionally to the weights themselves, and (2) does so by looking at a fraction of the full family of weights, opening the way to extremely large samples sets.

5.3.1 Importance sampling for Monte-carlo simulations

Given an arbitrary distribution μ on $\{1,\dots,M\}$ which puts non-zero probabilities on all the values, we can rewrite Equation (5.1) as

$$\sum_m \mu(m)\frac{w_m}{\mu(m)}f(m) = \mathbb{E}_{m\sim\mu}\left[\frac{w_m}{\mu(m)}f(m)\right] \tag{5.5}$$

which we can approximate by generating $m_1,\dots,m_K$ i.i.d $\sim \mu$, and using the empirical expectation

$$\hat{\mathbb{E}}_{m\sim\mu}\left[\frac{w_m}{\mu(m)}f(m)\right] = \frac{1}{K}\sum_{k=1}^{K}\frac{w_{m_k}}{\mu(m_k)}f(m_k). \tag{5.6}$$

The μ which minimizes the variance of the expectation is: $\mu(m) = \frac{w_m}{\sum_k w_k}$, This implies that we should invest the computation proportionally to the weights, and minimize optimally the variance of our estimator.

This choice makes sense if computing $\sum_k w_k$ is tractable, or so cheap to compute compared to the computation of the $f(m)$s that it still provides a substantial gain. However, we are interested in situations where not only w_m is more expensive to compute than $f(m)$, but we also aim at scaling M up to values far greater than the number of CPU operations we have at our disposal.

5.4 Importance Sampling Tree (IST)

We present a data-structure called Importance Sampling Tree (IST), which is a binary tree carrying the weights $w_1,\dots,w_M$ at its leaves, and having at each internal node statistics about the weights in the leaves below it. Given such a tree, we use a recursive sampling procedure that results in a sampling of the leaves, and – in a manner similar to the MCTS – we update the estimates at the nodes every time a sampling is done, and modulate the sampling policy accordingly.

Let $\mathcal{T}$ be the set of tree nodes, $x^* \in \mathcal{T}$ the root node, $\mathcal{L} \subset \mathcal{T}$ the leaves. Since the leaves carry the weights $w_1,\dots,w_M$, for any leaf $x \in \mathcal{L}$ let $m(x) \in \{1,\dots,M\}$ be the index of the weight there. Note that we will often make a confusion between $x \in \mathcal{L}$ and $m(x)$, identifying a leaf with its index.

For any internal node $x \in \mathcal{T}\setminus\mathcal{L}$, let $c_0(x), c_1(x) \in \mathcal{T}$ be its two child nodes. Finally let $\mathcal{L}(x)$ be the leaves of the sub-tree starting at x and $w(x)$ the sum of their weights. Hence, in particular $\forall\, x \in \mathcal{L}, w(x) = w_{m(x)}$.

Given a family of "bifurcation probabilities" $\Theta = (\theta_x)_{x\in\mathcal{T}\setminus\mathcal{L}} \in [0,1]^{\|\mathcal{T}\setminus\mathcal{L}\|}$, that is for each node the probability to "go down on the right", we can derive for each node x and each leaf m a

probability $\mu_\Theta(m;x)$ to reach the leaf m if we start from x and follow the θs at each node we meet. This probability can be defined recursively as

$$\forall x \in \mathcal{T},\ \mu_\Theta(m;x) = \begin{cases} \mathbf{1}_{\{m(x)\}}(m) & \text{if } x \in \mathcal{L} \\ (1-\theta_x)\mu_\Theta(m;c_0(x)) + \theta_x \mu_\Theta(m;c_1(x)) & \text{otherwise.} \end{cases} \tag{5.7}$$

and it is trivial that $\forall x \in \mathcal{T}, \Theta \in [0,1]^{\|\mathcal{T}\|}$, these probabilities sum to 1 and $\mu(.;x,\Theta)$ is a distribution on $\{1,\dots,M\}$. We can easily sample according to these distributions with a recursive procedure: For a leaf, return the leaf number, and for an internal node x recursively pick one of the two child nodes at random according to θ_x. Given a leaf m, the $\mu_\Theta(m;x)$ for all the parents x of m can be computed in $\mathcal{O}(D)$.

5.4.1 Adaptive sampling

We could use many different policies to modulate the θ_x and bias the sampling according to what we have observed. We present a strategy, based on accumulating at every node statistics $\mathcal{S}(x)$ about the weights observed during the previous sampling. Here $\mathcal{S}(x) = (s(x), v(x))$ where $s(x)$ is the sum of the weight estimates $w_m/\mu_\Theta(m;x)$, and $v(x)$ is the number of times the sampling went through x. For $v(x) > 0$, $\hat{w}(x) = s(x)/v(x)$ is an unbiased estimator of $w(x)$. We set θ_x to the ratio of the number of leaves in the child sub-trees if we do not have enough sampling for proper estimations, and to the ratio of the estimations of the weights otherwise. Formally with U a meta-parameter setting the minimum number of observations we request for biasing:

$$\theta_x = \begin{cases} \dfrac{\|\mathcal{L}(c_1(x))\|}{\|\mathcal{L}(c_0(x))\| + \|\mathcal{L}(c_1(x))\|} & \text{if } v(c_0(x)) < U \\ & \text{or } v(c_1(x)) < U, \\ \dfrac{\hat{w}(c_1(x))}{\hat{w}(c_0(x)) + \hat{w}(c_1(x))} & \text{otherwise.} \end{cases} \tag{5.8}$$

Hence, if we need T samples, for $t = 1,\dots,T$:

1. Recursively sample a path down the tree to a leaf m^t, according to $\mu_{\Theta^t}(.;x^*)$.
2. For every node x visited on that path compute $\mathcal{S}^{t+1}(x)$ and θ_x^{t+1} according to Equation (5.8) . All other nodes remain unchanged.

Then, following § 5.3.1, we can use

$$\sum_{m=1}^{M} w_m f(m) \simeq \frac{1}{T}\sum_{t=1}^{T} \frac{w_{m^t}}{\mu_{\Theta^t}(m^t;x^*)} f(m^t). \tag{5.9}$$

Note that the Θ^ts in this expression – and the resulting μs – are the ones that were in the tree when each sampling was done. Also, note that the update of $\mathcal{S}$ and Θ can be delayed,

suppressed, or done in an arbitrary order if implementation constraints impose it.

5.5 Experiments and results

5.5.1 Multi-layer Neural Network on a $2D$ synthetic data-sets

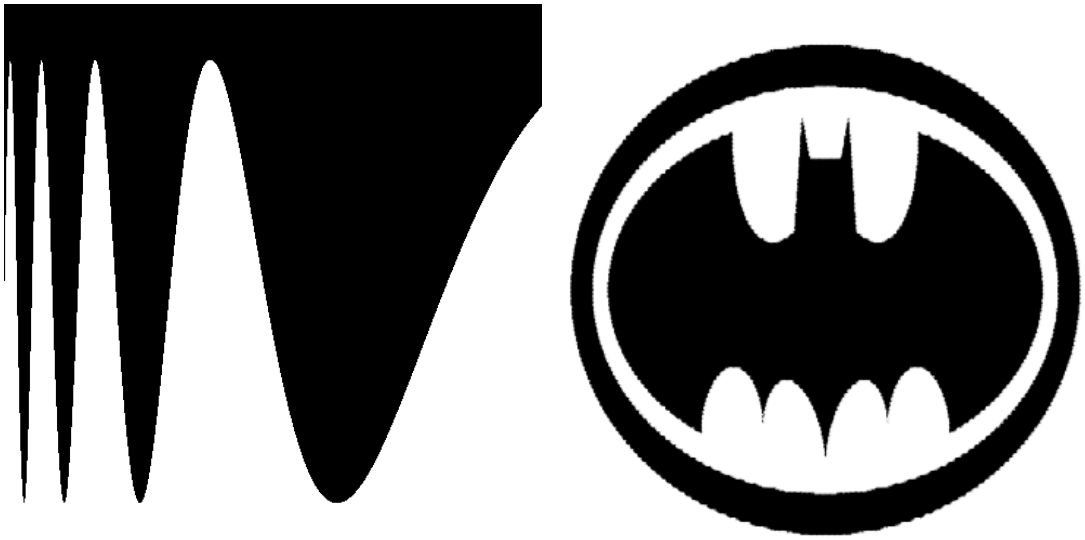


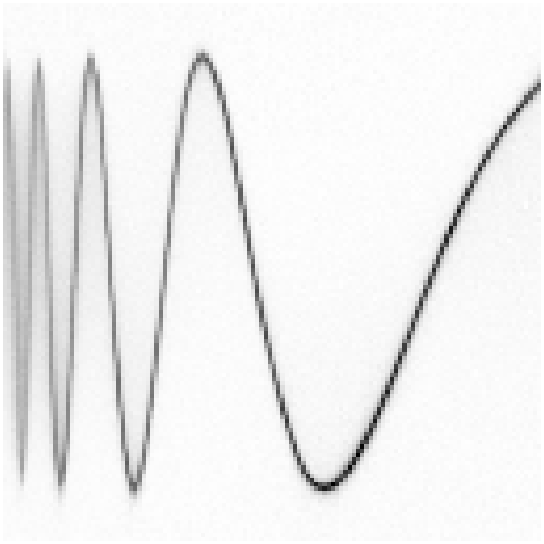
Figure 5.2: Two synthetic data-sets (Sinusoidal and batman) used to evaluate the training of multi-layer perceptron using importance sampling tree. Both of these data-sets require capturing fine details to separate the categories.

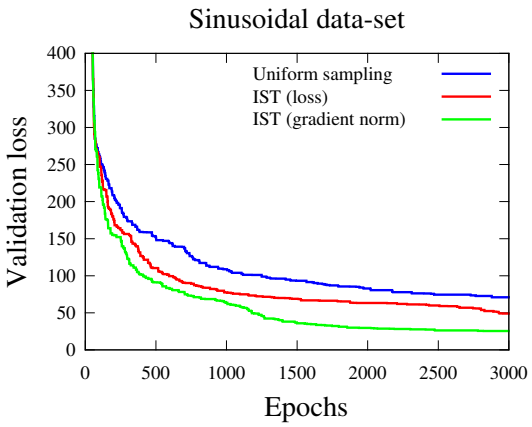
We consider two $2D$ synthetic problem, depicted in Figure 5.2. In Sinusoidal data-set the boundary between the two classes is an oscillation of variable frequency and for the Batman data-set it is the bat symbol which separates the classes. Both of the problems exhibit the difficulty of many real-world data-sets in which the core issue is to capture fine details of the boundary.

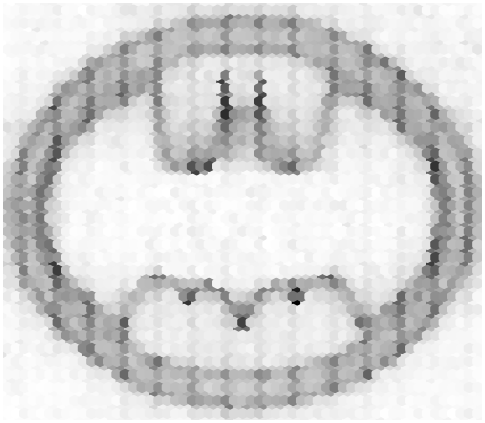
For the Sinusoidal data-set, we train a neural network with two units as input standing for the coordinates in the $[0,1]^2$ domain, two fully connected hidden layers with 40 units each, and one output unit. The transfer function is the hyperbolic tangent, and the weights are initialized layer after layer so that the response of every unit before non-linearity is centered, of standard deviation 0.5. We use the quadratic loss for training, and a pure stochastic gradient descent, one sample at a time. Every $1,000$ gradient steps, we compute a validation loss and adapt the step size. The experiments are repeated for 10 times with different train/validation splits.

For the Batman data-set, we used a 3 layer network with two input units as input standing for the coordinates in the $[0,1]^2$ domain. The two hidden layers have 32 neurons each interleaved

with rectified linear units as non-linear activation functions. The final layer is a linear classifier. We use the binary logistic loss as the loss function. The model is trained for 20 epochs with a batch size of 64 and the validation performance is calculated after each epoch. The experiments are repeated for 10 times with different random seeds and we average out the results.







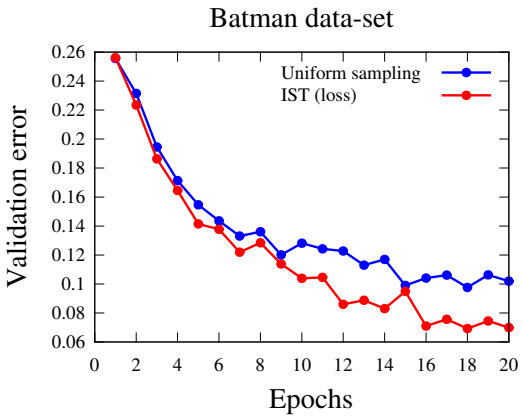


Figure 5.3: A multi-layer neural network on two synthetic $2D$ problems. The first column represents the sampling heat-map on the problems using IST. As we can see the computation focuses near the decision boundaries.

We compare three strategies: 1) **ANN** is the baseline, using uniform sampling for SGD. 2) **ANN-IST** samples with IST using the gradient norm as importance function, following Equation (5.3) and the same tree structure. However we did not evaluate ANN-IST with gradient norm as importance on the Batman data-set because computing the per-sample gradient in a mini-batch requires extra computation. 3) **ANN-IST-L** In this strategy we use the loss per sample instead of the gradient norm as the importance function. Since the gradient norms and loss

are order preserving, using loss as importance is a reasonable proxy. Also it is computationally efficient. Our results are summarized in Figure 5.3 and Table 5.2. IST improves convergence speed of SGD on both the data-sets by focusing the computation on the important points.

Table 5.2: Classification error of different sampling strategies on the synthetic Sinusoidal and Batman data-sets.

Data-set	ANN	ANN-IST	ANN-IST-L
Sinusoidal	0.0264 ± 0.0029	0.0062 ± 0.0014	0.0158 ± 0.0060
Batman	0.0976 ± 0.0170	-	0.0692 ± 0.0126

5.5.2 Deep Convolution Network on CIFAR10

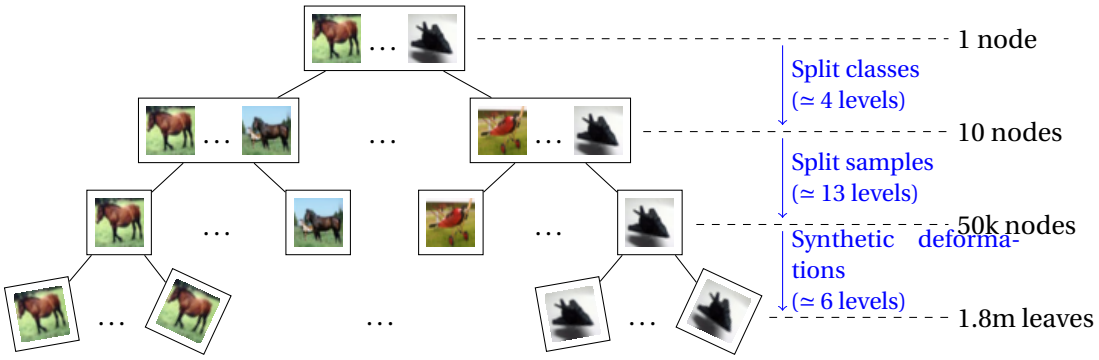


Figure 5.4: Structure of the sample tree we use to train a CNN on the CIFAR10 data-set. The first levels of the tree split the classes uniformly, the next levels split the images of each class according to the L^2 metric, until we get to individual images, from which the tree splits synthesized images using the same "deformation tree" replicated for each single original image.

For the synthetic examples of the previous section, we have exploited the Euclidean structure of the problem to build the IST. The main (potential) weakness for using it in practice is the availability of a tree structure consistent with the gradient norm.

We show in this section that this is not the case, and that a very natural tree structure leads to consistent sampling of training points with large gradient norms on a state-of-the-art large-scale problem of image classification.

Our experiments replicate the training of a network[1] designed for a Kaggle competition on the CIFAR10 data-set (Krizhevsky and Hinton, 2009), which relies on synthetic deformations of the original $50,000$ images with translations and scalings to create a total of 1.8 millions images.

The IST for this data-set has the structure depicted in Figure 5.4: we first split the classes, then the images in each class separately with a top down clustering using the image gradient maps.

[1] https://github.com/nagadomi/kaggle-cifar10-torch7

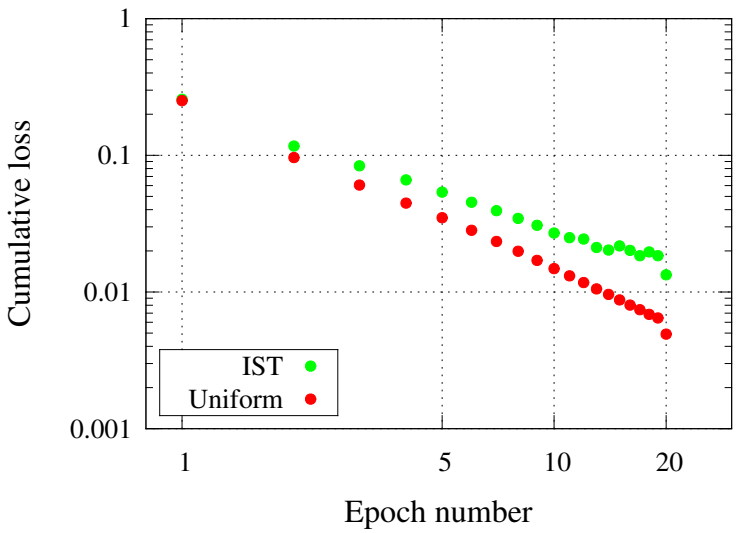


Figure 5.5: CNN experiment on CIFAR10. This scatter plot shows the ratio between the cumulative loss obtained with the IST-based sampling, and the same with the uniform sampling vs. the latter. Each dot corresponds to an epoch. As expected, when learning progresses, the cumulative loss goes down (dots move to the left) and gain with the IST goes up (dots move to the top). This is both due to a better estimate in the IST and a higher variance in the sample individual losses.

That is, we recursively apply a K-means with $K = 2$ at each node: starting from the full set of images at the top, the set is clustered into two sub-sets, which are themselves each clustered in two, etc. until reaching a single image, where we then stop and create a leaf. From that point we split the synthetically generated images by clustering the deformations themselves so that similar deformations are close in the tree.

We use the gradient norm as the importance for each sample in the IST, and we update it after every step of mini-batch gradient descent. We exploit the property of the matrix product as described in Goodfellow (2015) to compute the gradient norms efficiently for the fully connected layers. For the convolutional layers we have the per-sample gradients in the intermediate computations which we use to compute the gradient norms by trading some computational efficiency.

To evaluate performance, for both the uniform sampling that we use as baseline – and allows to replicate the error rate obtained in the original experiments with that network – and the IST sampling, we compute for each epoch the cumulative sample loss. The scatter plot in Figure 5.5 depicts the ratio between the latter quantity and the former as a function of the former. The results show that IST allows to capture up to three times more loss in the late

stages of the optimization.

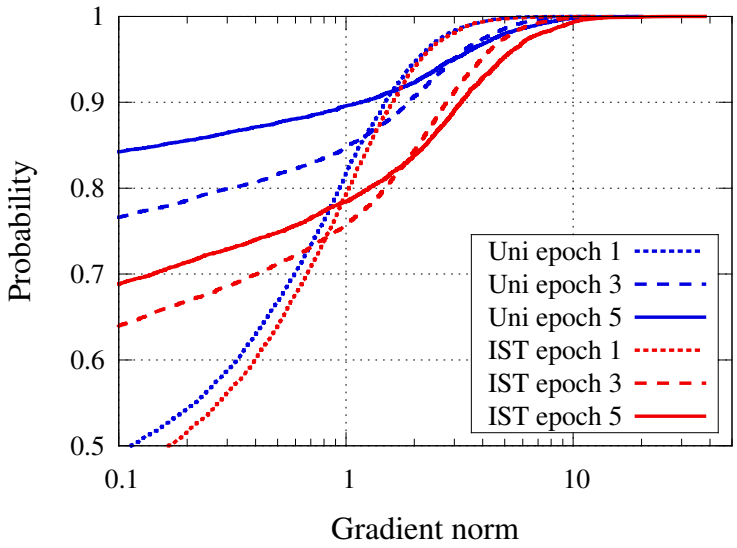


Figure 5.6: Gradient norms of the sampled training points on three different epochs. This plot shows that as the training progresses (1) most of the examples get a small gradient norm, and a small proportion of "hard" examples appears, with greater gradient norm, and (2) the sampling based on IST (red curves) is always better than uniform (blue curves), and improves in time, as reflected by the increasing gap between the curves.

In Figure 5.6 we show that the IST-based sampling is able to leverage the structure of the tree to sample training points with large gradient norms. In particular, after five epochs, the 0.75-quantile of the gradient norm is 2.64×10^{-4} with the uniform sampling, and 0.52 with the IST.

To assess the stability of the convergence of IST, we also computed the correlation between the $\hat{w}$ after each epoch between two randomized runs. The correlation goes from -0.017 after the first epoch, to 0.9897 after 10 epochs and remains above this value after that, showing that the two runs lead to virtually identical weight values. Figure 5.7 shows this result.

5.6 Discussion

We have presented a simple data-structure for the estimation of weighted sums over very large data-sets. Instead of sampling uniformly, or designing *a priori* a sampling scheme, our data-structure adaptively modulates the sampling according to data it has sampled previously. Because it compensates the sampling bias at any time, estimation of the empirical quantities of interest is done at the same time the sampling is improved.

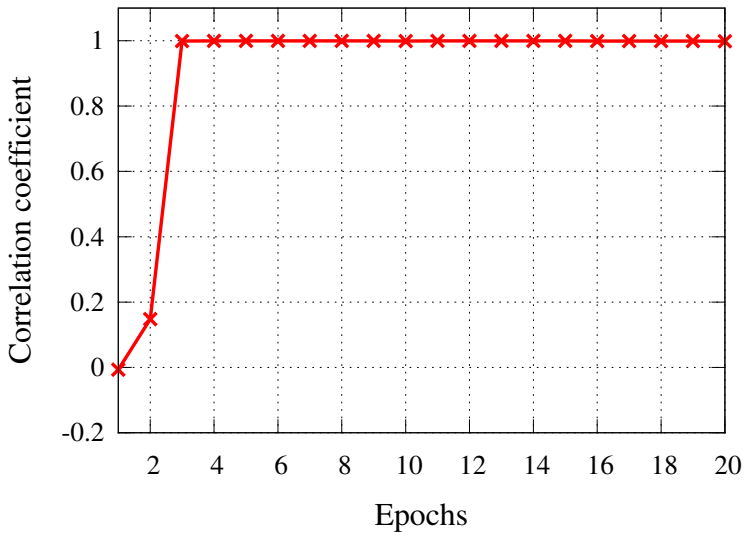


Figure 5.7: Graph showing the correlation coefficient between the sampling weights predicted by IST between two randomized runs along the epochs. As we can see two of the runs converges to the same distribution after few epochs.

Two key elements remain to be investigated thoroughly. The first is the construction of the tree itself. In our experiments on real data, it was constructed with a recursive partitioning based on the Euclidean metric. The rationale behind this strategy is that "close samples" should be "similarly weighted". This makes sense for the problem with a given kernel such as the synthetic data-sets on the Euclidean plane where a Gaussian kernel under a Euclidean metric is suitable. But this is an unsatisfactory proxy for the Neural networks since we are learning the kernel along with the classifier.

The second point is the sampling procedure. We have presented a strategy using empirical weight estimates. What is missing in our approach is a bandit-type use of a confidence interval on the estimate. For instance based on the Hœffding's inequality, to get something similar to the UCB policy used in MCTS.

6 Conclusions

6.1 Summary

In this thesis we have presented new algorithms for learning embeddings and a new data-structure for the efficient estimation of an empirical expectation over a large collection of data points.

In Chapter 3 we devised the WARCA metric learning algorithm which learns an embedding under a Mahalanobis distance by minimizing a loss defined on the weighted sum of the precision at different ranks. WARCA is designed to learn from data with a very large number of classes and few examples per class. It is also designed to tackle the problem settings where not all the classes are available during training (zero-shot learning). Our application of WARCA focused on the person re-identification problem for video surveillance and our results on 9 standard person re-identification data-sets show that WARCA can improves the state of the art. We have also presented a simple regularizer which prevents rank-deficient linear mappings in low rank matrix optimization by encouraging orthonormal linear maps. We have shown its effectiveness in the training of WARCA. This regularizer has later become popular in the deep learning community for training robust and very deep neural networks (Cisse et al., 2017b; Henaff et al., 2016; Xie et al., 2017; Vorontsov et al., 2017).

Chapter 4 was aimed at designing a parameter efficient and robust recurrent neural network which is an effective learning algorithm for embedding sequences. The algorithm we presented called Kronecker Recurrent Units (KRU) is parameter efficient and flexible through a Kronecker factored recurrent matrix and it is robust to the vanishing and exploding gradient problem by enforcing the same regularizer we presented for WARCA on the recurrent weight matrices. We have shown that enforcing this regularizer is computationally efficient for KRU because of the Kronecker factorization. The experimental results on 7 standard data-sets have proved the effectiveness of KRU and the KRU variant of LSTM in bringing down the number of parameters, improving the computational efficiency and the robustness of RNN without trading the statistical performance.

In Chapter 5 we presented the Importance Sampling Tree (IST) data-structure for selecting and modulating the important points in a large data-set to efficiently estimate an empirical expectation over the data-set. IST organizes the data-set into a tree by using the given metric on the data and use this metric regularity to sample efficiently the important points and at the same time modulate the sampling weights using the points it has sampled previously. We have shown that, using IST can improve the performance of stochastic gradient descent by reducing the variance of the stochastic gradients, on the synthetic data-sets. On the CIFAR10 data-set where the metric is being learned along with the classifier using a convolutional neural network, we show that IST can still sample and modulate important points by organizing the data using a hierarchical 2-means clustering on the edge features. However we did not observe an improvement in the performance of SGD compared to uniform sampling.

6.2 Future directions

The WARCA algorithm we developed is a shallow learning algorithm. It works well when we have a good kernel function which captures the similarity between the data points well, such as the χ^2 kernel on the hand designed image features. However WARCA does not perform well when we do not have a good kernel function on our data representation. For example, when the data representation is raw pixels instead of hand designed image features. Convolutional neural networks (CNNs) (LeCun et al., 1998) have proved to be an effective algorithm for learning from the raw pixels. CNNs learn a hierarchical kernel parametrized by convolutions along with the linear discriminator. However CNNs are shown to be vulnerable to adversarial perturbations in the input images (Szegedy et al., 2013; Fawzi, 2016; Cisse et al., 2017b). It would be interesting to see a linear WARCA as a layer on convolutional neural network, in particular whether WARCA could improve the robustness of CNNs. WARCA is not only designed to discriminate classes (maximize inter class distance) but also it explicitly minimizes intra class distance which the standard sample decomposable loss functions used with CNNs does not do. Incorporating this constraint might improve the robustness of CNNs.

The application of Kronecker matrices in machine learning is not just restricted to recurrent neural networks. It is useful for machine learning problems with very high dimensional input or output spaces where even learning a linear model is computationally prohibitive. Examples of such problems include extreme multi-label classification and machine translation. Current approaches in extreme multi-label classification rely on low-rank factorizations. In machine translation the current models restrict the size of the vocabulary or use approximate loss functions. However these models suffer from the problem of long-tail due to the low-rank methods and the use of approximate loss functions. As we go down the tail, the performance of these methods deteriorates rapidly. Predicting accurately the tail labels in extreme multi-label classification and correctly modeling the rare words in natural language processing can be beneficial in improving the performance and Kronecker matrices seem to be a promising choice.

Our importance sampling tree (IST) concludes with interesting avenues for future research. We have empirically showed that IST converges to its stationary distribution. However under what conditions will it converge is an open theoretical question and an answer to this question would guide us to a better design of IST.

A Chapter 2 Appendix

A.1 Perceptron convergence proof

Theorem 3. *If the data is linearly separable with a margin $\gamma > 0$, that is, there exists $\mathbf{w}^*$: $y_m({\mathbf{w}^*}^T \mathbf{x}_m *) \geq \gamma$ and $\|\mathbf{x}_m\| \leq R$ for all $m \in \{1, \ldots, M\}$ then Perceptron converges to 0 empirical risk in $t \leq \frac{R^2 \|\mathbf{w}^*\|^2}{\gamma^2}$ steps (Novikoff, 1962).*

Proof. The proof is a simple application of triangle inequality and Cauchy-Schwarz inequality. Consider a step t of Perceptron which makes mistake on $(\mathbf{x}_t, y_t)$:

$$\mathbf{w}_t = \mathbf{w}_{t-1} + y_t \mathbf{x}_t \tag{A.1}$$

$$\begin{aligned} \|\mathbf{w}_t\|^2 &= \left\|\mathbf{w}_{t-1} + y_t \mathbf{x}_t\right\|^2 && \text{(A.2)} \\ &\leq \|\mathbf{w}_{t-1}\|^2 + \|\mathbf{x}_t\|^2 && \text{(A.3)} \\ &\leq \|\mathbf{w}_{t-1}\|^2 + R^2 && \text{(A.4)} \\ &\leq tR^2 && \text{(A.5)} \end{aligned}$$

Now consider the dot product $\mathbf{w}_t^T \mathbf{w}^*$, the above A.5 implies:

$$\mathbf{w}_t^T \mathbf{w}^* \leq \left\|\mathbf{w}^*\right\| \sqrt{t} R \tag{A.6}$$

$$\begin{aligned} \mathbf{w}_t^T \mathbf{w}^* &= (\mathbf{w}_{t-1} + y_t \mathbf{x}_t)^T \mathbf{w}^* && \text{(A.7)} \\ &\geq (\mathbf{w}_{t-1}^T \mathbf{w}^* + \gamma) && \text{(A.8)} \\ &\geq t\gamma && \text{(A.9)} \end{aligned}$$

Combining A.6 and A.9:

$$t\gamma \leq \sqrt{t}R \left\| \mathbf{w}^* \right\| \tag{A.10}$$

$$\sqrt{t} \leq \frac{R \left\| \mathbf{w}^* \right\|}{\gamma} \tag{A.11}$$

$$t \leq \frac{R^2 \left\| \mathbf{w}^* \right\|^2}{\gamma^2} \tag{A.12}$$

□

B Chapter 3 Appendix

B.1 Metric learning

Most of the existing similarity measure learning approaches have been developed in the context of Mahalanobis distance learning paradigm (Xing et al., 2002; Weinberger and Saul, 2009; Davis et al., 2007; Mignon and Jurie, 2012; Köstinger et al., 2012). These methods essentially find a linear transformation of the data and then do the nearest neighbor search to find the points which are similar to the queried point in the projected space. Formally, Given M data points $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_M) : x_n \in \mathbb{R}^D$ and labels $(y_1, y_2, ..., y_M) : y_i \in \{1, ..., Q\}$, where Q is the number of classes, these methods aims at learning a linear projection matrix $\mathbf{W}$ where the distance $D_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{W}(\mathbf{x}_i - \mathbf{x}_j)\|_2^2$ is small for similar pairs and large for dissimilar pairs. Based on the way the problem is formulated the algorithms for learning $\mathbf{W}$ involve either optimization on the space of positive definite matrices or spectral methods on the second order statistics of the data. An illustration of metric learning using Mahalanobis metric is given in the figure B.1. Here we we present some of the most elegant models.

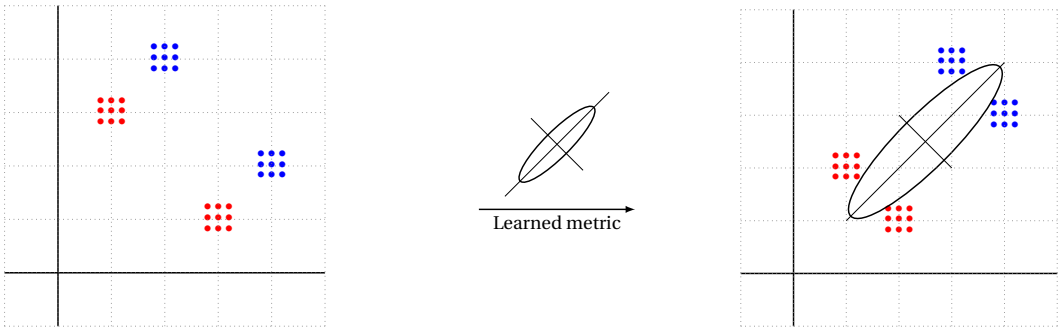


Figure B.1: A schematic illustration of metric learning.

B.1.1 Principal component analysis

PCA computes a linear transformation **W** that projects the D-dimensional data points into D' dimensional subspace ($1 \leq D' \leq D$) that minimizes the L_2 reconstruction error or maximizes the variance of the projected inputs, subject to the constraint that the columns of **W** are orthonormal . Thus the objective of the PCA can be written as an optimization problem in terms of the data covariance matrix **C**:

$$\mathbf{C} = \frac{1}{M} \sum_{m=1}^{M} (\mathbf{x}_m - \mu)(\mathbf{x}_m - \mu)^t \tag{B.1}$$

where $\mu = \frac{1}{M} \sum_{m=1}^{M} \mathbf{x}_i$ is data mean vector. The PCA optimization problem is given as follows.

$$\begin{aligned} &\max_{\mathbf{W}} && trace(\mathbf{W}^T \mathbf{C} \mathbf{W}) \\ &\text{subject to} && \mathbf{W}\mathbf{W}^T = \mathbf{I}. \end{aligned} \tag{B.2}$$

The above equation also called as the Rayleigh-Quotient has an elegant closed form solution in terms of the first K Eigen vectors of the covariance matrix. By using the representation theorem we can kernelize PCA and thus have a non-linear variant, which is very handy for the problems where the data lives in a non-linear low-dimensional manifold. PCA does not exploit the labels to generate more informative projections but it has some good properties in-terms of reducing the noise in the data as well as reducing the dimensionality of the input data thereby improving the performance of nearest neighbors and accelerating the nearest neighbor search.

B.1.2 Fisher discriminant analysis

Fisher discriminant analysis aims at finding a linear transformation of the data such that in the projected space the intra class scatter is minimized and inter class scatter is maximized. The inter class and intra class scatter matrices are defined by:

$$\begin{aligned} \mathbf{C}_{inter} &= \frac{1}{Q} \sum_{q=1}^{Q} \mu_n \mu_n^t \\ \mathbf{C}_{intra} &= \frac{1}{M} \sum_{q=1}^{Q} \sum_{i \in class(q)} (\mathbf{x}_i - \mu_q)(\mathbf{x}_i - \mu_q)^t \end{aligned} \tag{B.3}$$

where μ_q is the class q data mean vector. Thus FDA optimization problem is given as follows.

$$\begin{aligned} \max_{\mathbf{W}} \quad & trace(\frac{\mathbf{W}^T\mathbf{C}_{intra}\mathbf{W}}{\mathbf{W}^T\mathbf{C}_{inter}\mathbf{W}}) \\ \text{subject to} \quad & \mathbf{W}\mathbf{W}^T = \mathbf{I}. \end{aligned} \tag{B.4}$$

Similar to PCA the above equation has a closed form solution in terms of the Eigen vectors of $\mathbf{C}_{inter}^{-1}\mathbf{C}_{intra}$. FDA and its kernel variant (Schölkopf and Smola, 2002) is one of the most widely used algorithms for dimensionality reduction and for metric learning. The current state of the art methods on many person re-identification datasets is some variant of FDA (Xiong et al., 2014).

Large margin nearest neighbors (Weinberger and Saul, 2009)

Weinberger and Saul (2009) proposed a Mahalanobis metric learning approach in a large margin setting by exploiting the local structure in the data. For each data-point $\mathbf{x}$ the points which doesn't belong to its class and are with in the k-nearest neighbor radius is penalized and at the same time the it also penalizes large distance between the k-nearest points having the same label. The above ideas can be realized by solving the following optimization problem

$$\min_{\mathbf{W}} \quad \lambda \sum_{j \rightsquigarrow i} \|\mathbf{W}^T(\mathbf{x}_i - \mathbf{x}_j)\|^2 + (1-\lambda) \sum_{i, j \rightsquigarrow i} \sum_{l} (1 - y_{il}) \Big[1 + \|\mathbf{W}^T(\mathbf{x}_i - \mathbf{x}_j)\|^2 - \|\mathbf{W}^T(\mathbf{x}_i - \mathbf{x}_l)\|^2\Big]_+ \tag{B.5}$$

Where $[x]_+$ is the standard hinge loss, That is, $[x]_+ = max(0, x)$ and λ is the parameter controlling the trade-off between minimizing the k-nearest neighbor distance between similar points and penalizing the invading points in the k-nearest neighbor radius. Though this is an elegant formulation there are some inherent difficulties in solving this problem in large scale. The above formulation is not convex. We can get away from that problem by replacing $\mathbf{W}$ by $\mathbf{M} = \mathbf{W}\mathbf{W}^T$ but then it is computationally expensive because of the positive semi-definite constraint on the matrix $\mathbf{M}$. The authors propose a gradient descent algorithm, with projections of current parameter estimate on to the positive semi-definite cone after each gradient update. This projection step makes the formulation difficult to apply in large scale setting as it involves a spectral decomposition at each gradient iteration.

B.1.3 Information-theoretic metric learning (Davis et al., 2007)

Davis et al. (2007) exploits the relationship between multivariate Gaussian distributions and the set of Mahalanobis distances. The idea is to search for a distance metric $\mathbf{M}$ that satisfies the constraints from the point labels and at the same time close to a distance metric prior $\mathbf{M}_0$. The closeness acts as a regularizer and it is captured through the KL divergence between

the corresponding distributions. The constraints enforce that the distance between similar pairs are below a threshold, that is $D_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \leq l$ and the distance between dissimilar points are large, ie $D_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \geq u$. Solving the resulting optimization involves Bregmen projections (Bregman et al., 2003).

B.1.4 KISS metric learning (Köstinger et al., 2012)

Köstinger et al. (2012) proposes KISS (Keep It Simple and Straight forward) metric learning. This method considers two independent data generating distributions one for similar and other for dissimilar pairs. Then whether a pair of points $(\mathbf{x}_i, \mathbf{x}_j)$ is dissimilar or not can be obtained by a log-odd ratio test. Under Gaussian assumptions the likelihood ratio between two points $\mathbf{x}_i$ and $\mathbf{x}_j$ can be written as

$$\psi_{ij} = (\mathbf{x}_i - \mathbf{x}_j)^t (\Sigma_{\mathcal{S}} - \Sigma_{\mathcal{D}})(\mathbf{x}_i - \mathbf{x}_j) + log(|\Sigma_{\mathcal{D}}|) - log(|\Sigma_{\mathcal{S}}|), \tag{B.6}$$

where $\Sigma_{\mathcal{S}}$ and $\Sigma_{\mathcal{D}}$ are the covariance matrices for similar pairs of points $\mathcal{S}$ and dissimilar pairs of points $\mathcal{D}$ respectively. Assuming zero mean on pairwise differences between points in $\mathcal{S}$ and $\mathcal{D}$. The covariance matrices can be written as follows:

$$\begin{aligned} \Sigma_S &= \frac{1}{|\mathcal{S}|} \sum_{(i,j)\in S} (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^t \\ \Sigma_d &= \frac{1}{|\mathcal{D}|} \sum_{(i,j)\in D} (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^t \end{aligned} \tag{B.7}$$

The smaller the ψ_{ij} is, the similar the points are. By eliminating the constant term and projecting $(\Sigma_{\mathcal{S}} - \Sigma_{\mathcal{D}})$ to the PSD cone they obtain a metric which can be used to match points.

B.1.5 Siamese neural network (Chopra et al., 2005)

Chopra et al. (2005) proposes a non-linear transformation based similarity measure learning using convolutional neural network. This model is specifically designed in the context of face verification and the learning process minimizes a discriminative loss function that drives similarity measure to be small for similar pairs of faces and large for different pairs. The mapping function from raw pixel features to target feature space is a convolutional neural network which is designed to be robust against geometric distortions.

B.1.6 Chopping (Fleuret and Blanchard, 2005)

A notable exception from the Mahalanobis distance based similarity measure learning paradigm is the chopping algorithm (Fleuret and Blanchard, 2005). It works by creating a large number of random binary splits of data labels such that all points having the same label are put together. Classifiers are trained on these binary splits and for prediction, given two samples the response of the split predictors are combined with a Bayesian rule into a posterior probability of similarity. Our current research is mainly based on this approach. The original chopping algorithm was developed to create binary splits of data label and use Perceptron as the predictor but it is straight forward to extend the algorithm to K-ary ($2 \leq k \leq$ number of training classes) splits of data label and use more powerful predictors such as Adaboost, Support vector machine or any other algorithms.

B.2 Person re-identification

Person re-identification (Gong et al., 2014) is the problem of matching people across disjoint camera views. This is a very challenging problem that has tremendous applications in various domains of video-surveillance. As illustrated in Figure B.2 the application that we are primarily interested in is improving the performance of the multi-object tracking by person re-identification to disambiguate tracklets (Shitrit et al., 2011).

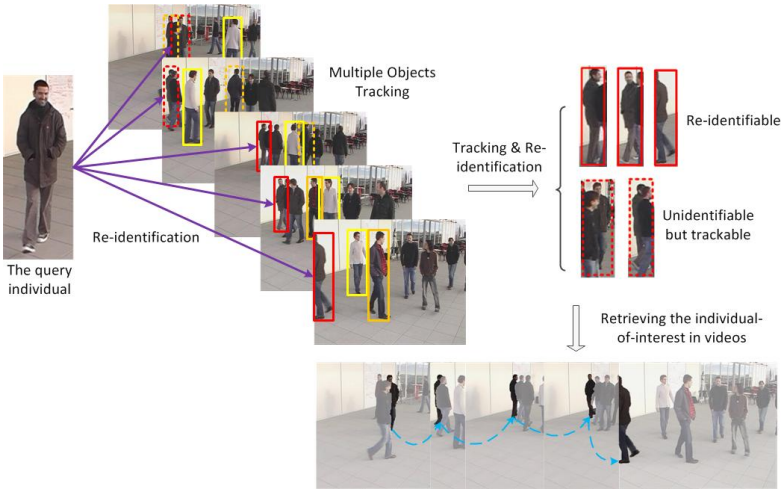


Figure B.2: Person re-identification to improve multi object tracking. Image of the occluded person is matched against a set of gallery images with the help of re-identification model to resolve identity switches, thereby improving tracking.

This problem is challenging because of the illumination, view point changes, occlusions and various other non-linear transformations between the images of the person obtained between multiple camera views. The research efforts for tackling this problem can be broadly classified into two categories (Gong et al., 2014): 1) Design features that are discriminative and invariant to various non-linear transformations. 2)Develop machine learning algorithms that exploits

existing labeled data information to learn the semantics that are invariant across various complex transformations.

B.2.1 Performance measures for person re-identification

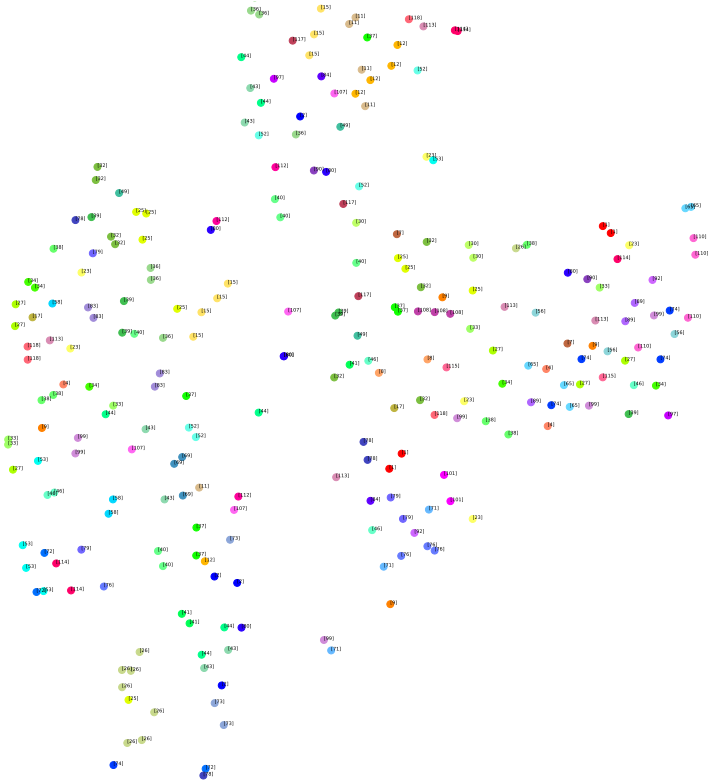
The most commonly used performance metric for person re-identification is the Cumulative Matching Characteristic(CMC) curve (Gong et al., 2014). The CMC curve shows how the matching performance increases as the allowed number of returned image increases. To generate the CMC curve for a particular algorithm, the rank of the correct observation is recorded and aggregated over the entire test set to generate the Match Characteristic $M(r)$, the probability that the rank r choice is right and then this is accumulated to get the CMC curve.

Pedagadi et al. (2013) uses PUR (Proportion of Uncertainty Removed) scores. Let N is the number of images against to which an image is matched against. When we match a person image against a set of images we can assume that each image in the set has equal prior probability of being correct therefore the initial uncertainty or entropy is $log(N)$. After the matching, the posterior probability that the retrieved image at rank r is the correct match is the match characteristic $M(r)$ and expected uncertainty at rank r is $-\sum_{i=1}^{r} M(i)log(M(i))$. Therefore the proportion of uncertainty removed or PUR score at r is the difference between $log(r)$ and the expected uncertainty at rank r. To make it invariant to the gallery size we can normalize it by initial entropy. Thus PUR score is defined as:

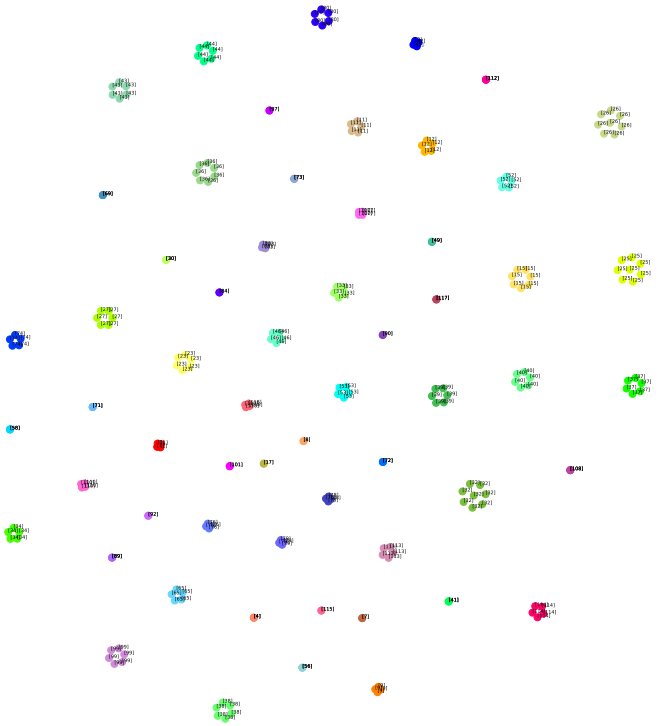
$$PUR = \frac{log(N) + \sum_{r=1}^{N} M(r)log(M(r))}{log(N)} \tag{B.8}$$

B.3 Feature space visualization for WARCA

In Figure B.3 and B.4 we show the visualization of feature space in the χ^2 kernel space and the feature space learned by WARCA on iLIDS and CUHK01 dataset using the tSNE algorithm (Maaten and Hinton, 2008). The tSNE visualizations illustrate that the WARCA embeddings bring together points from the similar class and separates dissimilar class.

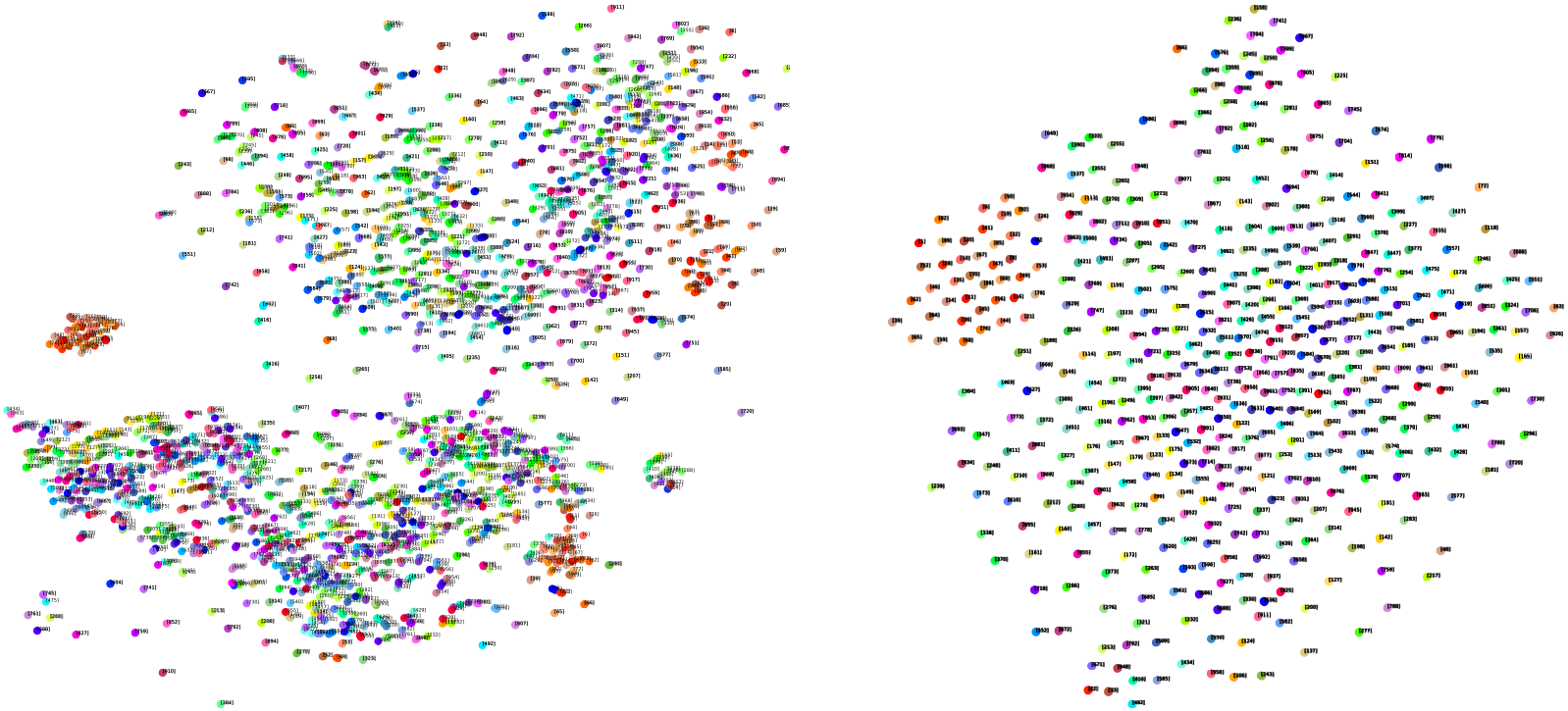


Points in the χ^2 kernel space.



Points in the space learned by WARCA.

Figure B.3: Feature space visualization on iLIDS data-set using tSNE (Maaten and Hinton, 2008).



Points in the χ^2 kernel space.

Points in the space learned by WARCA.

Figure B.4: Feature space visualization on CUHK01 data-set using tSNE (Maaten and Hinton, 2008).

B.4 Maximizing AUC with a Mahalanobis metric

Here we present a simple algorithm to Maximize AUC with Mahalanobis Metric (MAMM). This is the first algorithm we derived in order to learn from small data with very number of classes.

Let us consider a training set of data point / label pairs

$$(\mathbf{x}_m, y_m) \in \mathbb{R}^D \times \{1, \ldots, Q\}, m = 1, \ldots, M$$

Let $\mathcal{T}$ be a set of triplet of indexes from the training set defined as follows:

$$\mathcal{T} = \left\{(i, j, k) \in \{1, \ldots, M\}^3,\ y_i = y_j, y_i \neq y_k\right\}.$$

Let $\mathbf{W}$ be a linear transformation that maps the data points from $\mathbb{R}^D$ to $\mathbb{R}^{D'}$, with $D' \leq D$. The distance function under the linear map W is given by

$$\mathcal{F}_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{W}(\mathbf{x}_i - \mathbf{x}_j)\|_2$$

We are interested in learning the parameter W of the distance function by maximizing the area under the curve. The AUC estimated on the training set using the distance function defined above is:

$$\text{AUC} = \frac{1}{|\mathcal{T}|} \sum_{t=1}^{|\mathcal{T}|} \mathbf{1}_{\xi_{i_t j_t k_t} > 0}.$$

where

$$\xi_{ijk} = \mathcal{F}^2_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_k) - \mathcal{F}^2_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j).$$

Our goal is to find a mapping W^* that maximizes the above equation. This is equivalent to minimizing:

$$\min_{\mathbf{W}} \sum_{t=1}^{|\mathcal{T}|} \mathbf{1}_{\xi_{i_t j_t k_t} \leq 0}.$$

The above optimization problem is not tractable as it aims at minimizing the 0-1 loss. In order to make it tractable we upper bound this loss with the hinge loss Shalev-Shwartz and Ben-David (2014). Also, to control the capacity of the model, we add a L^2 regularizer on the

parameters we are estimating. Thus the optimization problem becomes:

$$\begin{aligned}
&\underset{\mathbf{W}}{\text{minimize}}\ \frac{1}{2}\|\mathbf{W}\|^2 + C\sum_{t=1}^{|\mathcal{T}|}\xi_{i_t j_t k_t} \\
&\text{subject to, } \forall (i,j,k) \in \mathcal{T}, \\
&\quad \mathcal{F}^2_{\mathbf{W}}(\mathbf{x}_i,\mathbf{x}_k) - \mathcal{F}^2_{\mathbf{W}}(\mathbf{x}_i,\mathbf{x}_j) \geq \gamma - \xi_{ijk}, \\
&\quad \xi_{ijk} \geq 0.
\end{aligned}$$

where $\gamma > 0$ is the margin at which similar pairs and dissimilar pairs are separated, $C \geq 0$ is the parameter that controls the capacity of the learner and $\|\mathbf{W}\|^2$ is the square of the Frobenius norm of the matrix $\mathbf{W}$. That is $\|\mathbf{W}\|^2 = Tr(\mathbf{W}\mathbf{W}^T)$. This is equivalent to minimizing the following loss, without constraint:

$$\mathcal{L}(\mathbf{W}) = \frac{1}{2}\|\mathbf{W}\|^2 + C\sum_{t=1}^{|\mathcal{T}|}\left|\gamma + \mathcal{F}^2_{\mathbf{W}}(\mathbf{x}_{i_t},\mathbf{x}_{j_t}) - \mathcal{F}^2_{\mathbf{W}}(\mathbf{x}_{i_t},\mathbf{x}_{k_t})\right|_+. \tag{B.9}$$

This optimization problem has important connections to the popular large margin nearest neighbors (LMNN) Weinberger and Saul (2009). For each point $\mathbf{x}_i$ we consider all of its neighbors to minimize the triplet loss where as in LMNN only the k nearest neighbors are considered and that is equivalent to a surrogate loss maximizing the area under the curve up-to location k or the precision at k Shalev-Shwartz and Ben-David (2014).

B.4.1 Kernelization

Let $\mathbf{W}$ be of the form:

$$\mathbf{W} = \mathbf{A}\mathbf{X}^T = \mathbf{A}\begin{pmatrix} \mathbf{x}_1^T \\ \dots \\ \mathbf{x}_N^T \end{pmatrix}$$

Using the above definition we can write everything in terms of the kernel between samples in the training set.

$$\begin{aligned}
\mathcal{F}_{\mathbf{A}}(\mathbf{x}_i,\mathbf{x}_j) &= \|\mathbf{A}\mathbf{X}^T(\mathbf{x}_i - \mathbf{x}_j)\|_2 &\text{(B.10)}\\
&= \|\mathbf{A}(\mathbf{k}_i - \mathbf{k}_j)\|_2 &\text{(B.11)}
\end{aligned}$$

where $\mathbf{k}_i$ and $\mathbf{k}_j$ are the i^{th} and j^{th} columns of the kernel matrix $\mathbf{L} = \mathbf{X}^T\mathbf{X}$. Then, the loss (B.9) becomes:

$$\mathcal{L}(\mathbf{A}) = \frac{1}{2}\|\mathbf{A}\mathbf{K}\mathbf{A}^T\|^2 + C\sum_{t}^{\mathcal{T}}\left|\gamma + \mathcal{F}^2_{\mathbf{A}}(\mathbf{k}_{i_t},\mathbf{k}_{j_t}) - \mathcal{F}^2_{\mathbf{A}}(\mathbf{k}_{i_t},\mathbf{k}_{k_t})\right|_+ \tag{B.12}$$

B.4.2 Optimization

A sub-gradient of the function $\mathcal{L}(\mathbf{A})$ is given as:

$$\nabla\mathcal{L}(\mathbf{A}) = 2\mathbf{A}\mathbf{K} + 2C\mathbf{A}\sum_{t}^{\mathcal{T}} \mathbf{1}_{h_t(\mathbf{A})>0}\mathbf{G}_t \tag{B.13}$$

$$= 2\mathbf{A}\mathbf{K} + 2C\mathbf{A}\mathbf{K}\mathbf{K}^{-1}\sum_{t}^{\mathcal{T}} \mathbf{1}_{h_t(A)>0}\mathbf{G}_t \tag{B.14}$$

where

$$h_t(\mathbf{A}) = \left|\gamma + \mathcal{F}^2_{\mathbf{A}}(\mathbf{k}_{i_t}, \mathbf{k}_{j_t}) - \mathcal{F}^2_{\mathbf{A}}(\mathbf{k}_{i_t}, \mathbf{k}_{k_t})\right|_+ \tag{B.15}$$

and

$$\mathbf{G}_t = (\mathbf{k}_{i_t} - \mathbf{k}_{j_t})(\mathbf{k}_{i_t} - \mathbf{k}_{j_t})^T - (\mathbf{k}_{i_t} - \mathbf{k}_{k_t})(\mathbf{k}_{i_t} - \mathbf{k}_{k_t})^T. \tag{B.16}$$

Multiplying the right hand side of the equation (B.14) by $\mathbf{K}^{-1}$:

$$\nabla\mathcal{L}(\mathbf{A})\mathbf{K}^{-1} = 2\mathbf{A} + 2C\mathbf{A}\mathbf{K}\mathbf{K}^{-1}\sum_{t}^{\mathcal{T}} \mathbf{1}_{h_t(\mathbf{A})>0}\mathbf{G}_t\mathbf{K}^{-1}. \tag{B.17}$$

By taking the $\mathbf{K}^{-1}$ on the both side of summation inside, we get:

$$\nabla\mathcal{L}(\mathbf{A})\mathbf{K}^{-1} = 2\mathbf{A} + 2C\mathbf{A}\mathbf{K}\sum_{t}^{\mathcal{T}} \mathbf{1}_{h_t(\mathbf{A})>0}\mathbf{U}_t, \tag{B.18}$$

with

$$\mathbf{U}_t = \mathbf{K}^{-1}\mathbf{G}_t\mathbf{K}^{-1} \tag{B.19}$$

$$= (\mathbf{e}_{i_t} - \mathbf{e}_{j_t})(\mathbf{e}_{i_t} - \mathbf{e}_{j_t})^T - (\mathbf{e}_{i_t} - \mathbf{e}_{k_t})(\mathbf{e}_{i_t} - \mathbf{e}_{k_t})^T, \tag{B.20}$$

where $\mathbf{e}_l$ is the l^{th} column of the canonical basis that is the vector whose l^{th} component is one and all others are zero. In the preconditioned sub-gradient descent we use the updates of the form:

$$\mathbf{A}^{p+1} = (1-2\eta)\mathbf{A}^p - \eta 2C\mathbf{A}^p\mathbf{K}\sum_{t}^{\mathcal{T}} \mathbf{1}_{h_t(\mathbf{A}^p)>0}\mathbf{U}_t, \tag{B.21}$$

where $\mathbf{A}^p$ is the parameter obtained after p iterations and η is the step size obtained by a line search algorithm.

Algorithm 5 Preconditioned gradient descent algorithm for MAMM

Input: Label vector $y \in \{1,...,Q\}^M$, Kernel matrix $\mathbf{K} \in \mathbb{R}^{M\times M}$, Regularizer $C \geq 0$, Initial solution $\mathbf{A}^0 \in \mathbb{R}^{D'\times N}$, Step size η

$\mathcal{T} = \left\{(i,j,k) \in \{1,\ldots,N\}^3,\ y_i = y_j, y_i \neq y_k\right\}$.
$p = 0$
while (not converged) **do**
  $\mathbf{X}' = \mathbf{A}^p\mathbf{K}$
  $\mathbf{G} = \underbrace{\begin{bmatrix} 0 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & 0 \end{bmatrix}}_{M} \Big\}M$
  $\mathbf{Z}$ =PairwiseEuclideanDistanceSquared($\mathbf{X}'$)
  for $t = 1$ ... $|\mathcal{T}|$ **do**
    if $1 + \mathbf{Z}[t_i][t_j] - \mathbf{Z}[t_i][t_k] > 0$ **then**
      $\mathbf{G}[t_j][t_j] = \mathbf{G}[t_j][t_j] + 1$
      $\mathbf{G}[t_i][t_j] = \mathbf{G}[t_i][t_j] - 1$
      $\mathbf{G}[t_j][t_i] = \mathbf{G}[t_j][t_i] - 1$
      $\mathbf{G}[t_k][t_k] = \mathbf{G}[t_k][t_k] - 1$
      $\mathbf{G}[t_i][t_k] = \mathbf{G}[t_i][t_k] + 1$
      $\mathbf{G}[t_k][t_i] = \mathbf{G}[t_k][t_i] + 1$
    end if
  end for
  $\mathbf{A}^{p+1} = (1-2\eta)\mathbf{A}^p - 2\eta C\mathbf{X}'\mathbf{G}$
  $p = p + 1$
end while
Output $\mathbf{A}^p$

	Iterative methods					Spectral methods		
Dataset	MAMM-χ^2	MAMM-L	rPCCA-χ^2	rPCCA-L	SVMML	LFDA-χ^2	LFDA-L	KISSME
CUHK01	0.52±0.01 #2	0.35±0.01	0.49±0.01 #3	0.35±0.01	0.28±0.01	0.55±0.01 #1	0.34±0.01	0.36±0.01
VIPeR	0.26±0.01 #2	0.20±0.01	0.22±0.02	0.16±0.01	0.23±0.02 #3	0.31±0.02 #1	0.20±0.02	0.21±0.01
PRID450s	0.19±0.02 #2	0.09±0.01	0.17±0.02 #3	0.08±0.01	0.13±0.02	0.24±0.02 #1	0.03±0.01	0.15±0.02
CAVIAR	0.38±0.03 #2	0.34±0.02	0.37±0.02 #3	0.27±0.02	0.27±0.02	0.41±0.02 #1	0.37±0.02 #3	0.32±0.02
3DPeS	0.47±0.03 #2	0.40±0.03	0.46±0.02 #3	0.33±0.02	0.30±0.02	0.52±0.02 #1	0.43±0.03	0.38±0.02
iLIDS	0.28±0.03 #3	0.26±0.04	0.28±0.03 #3	0.23±0.03	0.21±0.03	0.36±0.02 #1	0.33±0.03 #2	0.28±0.04

Table B.1: Rank 1 performance of different methods on different data-sets. We indicated the ranking of the top-3 methods for each data-set.

	Iterative methods					Spectral methods		
Dataset	MAMM-χ^2	MAMM-L	rPCCA-χ^2	rPCCA-L	SVMML	LFDA-χ^2	LFDA-L	KISSME
CUHK01	0.76±0.01 #1	0.56±0.02	0.74±0.01 #2	0.57±0.02	0.54±0.02	0.76±0.01 #1	0.49±0.01	0.54±0.01
VIPeR	0.58±0.02 #2	0.49±0.02	0.53±0.02 #3	0.43±0.02	0.52±0.02	0.64±0.02 #1	0.45±0.02	0.48±0.03
PRID450s	0.47±0.03 #2	0.30±0.02	0.44±0.03 #3	0.27±0.02	0.38±0.02	0.55±0.02 #1	0.13±0.01	0.37±0.02
CAVIAR	0.72±0.02 #1	0.64±0.02	0.71±0.02 #2	0.57±0.02	0.62±0.04	0.69±0.04 #3	0.62±0.03	0.61±0.03
3DPeS	0.73±0.02 #2	0.66±0.02	0.73±0.02 #2	0.58±0.02	0.60±0.03	0.75±0.02 #1	0.66±0.02	0.60±0.02
iLIDS	0.57±0.02	0.54±0.03	0.58±0.03 #3	0.52±0.04	0.51±0.04	0.66±0.03 #1	0.60±0.03 #2	0.54±0.04

Table B.2: Rank 5 performance of different methods on different data-sets. We indicated the ranking of the top-3 methods for each data-set.

B.4.3 Experiments

In this section we describe the set of experiments we carried out to evaluate our proposed method.

We used the same set of features for all the datasets and all the features are essentially histogram based. First all the datasets were rescaled to 128×48 resolution and then 16 bin color histograms on RGB, YUV, and HSV channels, as well as texture histogram based on Local Binary Patterns (LBP) were extracted on 6 non-overlapping horizontal patches. All the histograms are normalized per patch to have unit L_1 norm and concatenated into a single vector of dimension 2580 Mignon and Jurie (2012); Xiong et al. (2014).

In order to fairly evaluate the algorithms, we set the dimensionality of the projected space to 40 for both MAMM and rPCCA. We chose the regularization parameter through cross validation across the data splits. Since the size of the parameter matrix scales like $O(D^2)$ for SVMML and KISSME we first reduced the dimension of the original features using PCA keeping 95% of the original variance and then applied these algorithms. In our tables and figures MAMM$-\chi^2$, MAMM-L, rPCCA$-\chi^2$, rPCCA-L, LFDA$-\chi^2$ and LFDA-L denote MAMM with χ^2 kernel, MAMM with linear kernel, rPCCA with χ^2 kernel, rPCCA with linear kernel and LFDA with χ^2 kernel, LFDA with linear kernel respectively.

Tables B.1 and B.2 summarize respectively the rank1 and rank5 performance of all the methods, and Table B.3 summarizes the AUC performance score. Figure B.5 reports the CMC curves comparing MAMM against the baselines on all the six data-sets. The dashed curves denote spectral methods and the continuous ones denote the iterative methods, the circle and the star markers denote linear and kernel methods respectively.

	Iterative methods					Spectral methods		
Dataset	MAMM-χ^2	MAMM-L	rPCCA-χ^2	rPCCA-L	SVMML	LFDA-χ^2	LFDA-L	KISSME
CUHK01	0.83±0.01 #1	0.67±0.01	0.81±0.01 #3	0.68±0.01	0.66±0.01	0.82±0.01 #2	0.58±0.01	0.63±0.01
VIPeR	0.74±0.01 #2	0.67±0.01	0.71±0.01 #3	0.62±0.01	0.71±0.02 #3	0.79±0.01 #1	0.63±0.01	0.65±0.01
PRID450s	0.65±0.02 #2	0.49±0.02	0.63±0.02 #3	0.46±0.02	0.60±0.01	0.72±0.02 #1	0.28±0.01	0.54±0.02
CAVIAR	0.85±0.01 #1	0.81±0.01 #3	0.84±0.01 #2	0.77±0.01	0.79±0.02	0.81±0.02 #3	0.77±0.02	0.79±0.02
3DPeS	0.82±0.01 #3	0.77±0.02	0.83±0.02 #1	0.72±0.02	0.73±0.02	0.83±0.01 #1	0.76±0.01	0.72±0.01
iLIDS	0.74±0.02 #3	0.72±0.02	0.75±0.02 #2	0.70±0.02	0.70±0.02	0.79±0.02 #1	0.74±0.02 #3	0.70±0.03

Table B.3: AUC of different methods on different data-sets. We indicated the ranking of the top-3 methods for each data-set.

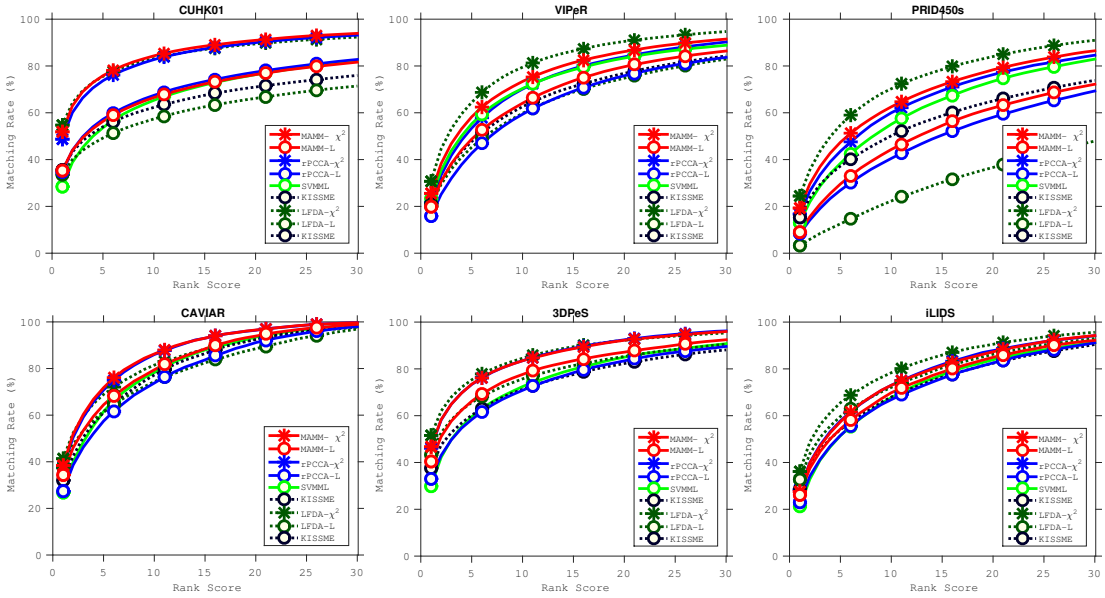


Figure B.5: CMC curves comparing MAMM against other methods on six re-identification datasets.

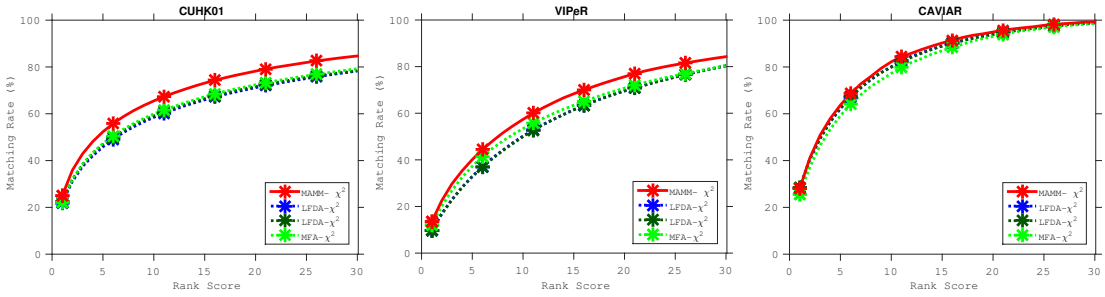


Figure B.6: CMC curves comparing MAMM against FDA variants under low capacity setting.

Comparison to iterative methods

MAMM-χ2 improves over SVMML on all the data-sets and this is coupled by the fact that MAMM is optimizing a better loss function and it can exploit better features through the kernel. MAMM-χ2 performs better than rPCCA on all data-sets except iLIDS. Especially the

performance of MAMM-χ2 over rPCCA and SVMML on the most challenging VIPeR data-set is worth noting.

Comparison to FDA variants

Despite having better performance than recently published algorithms, MAMM does not improve the overall best performance drastically, in spite of explicitly optimizing the performance measure itself. The reason for this is that the existing re-identification data-sets are too small for sophisticated algorithms to learn the invariances without over-fitting. Consequently simple methods such as Fisher Discriminant Analysis (FDA) work nearly as well or better than sophisticated models by optimizing a criterion different from the performance measure.

We did some additional experiments to verify this claim. We limited the capacity of MAMM and FDA by reducing the projected dimension to 5 and evaluated how the models behave under low capacity. Figure B.6 summarizes our results. MAMM works better than FDA and its variants. Please note that even under his low capacity setting there is a lot of over-fitting happening. On the training splits AUC of both the models is nearly 1.

C Chapter 4 Appendix

C.1 Analysis of vanishing and exploding gradients in RNN

Given a sequence of T input vectors: $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{T-1}$, let us consider the operation at the hidden layer t of a recurrent neural network:

$$\mathbf{z}_t = \mathbf{W}_t \mathbf{h}_{t-1} + \mathbf{U}_t \mathbf{x}_t + \mathbf{b} \tag{C.1}$$

$$\mathbf{h}_t = \sigma(\mathbf{z}_t) \tag{C.2}$$

By the chain rule,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}_t} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}_T} \frac{\partial \mathbf{h}_T}{\partial \mathbf{h}_t} \tag{C.3}$$

$$= \frac{\partial \mathcal{L}}{\partial \mathbf{h}_T} \prod_{k=t}^{T-1} \frac{\partial \mathbf{h}_{k+1}}{\partial \mathbf{h}_k} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}_T} \prod_{k=t}^{T-1} \mathbf{J}_{k+1} \mathbf{W}^T \tag{C.4}$$

where σ is the non-linear activation function and $\mathbf{J}_{k+1} = diag(\sigma^{'}(\mathbf{z}_{k+1}))$ is the Jacobian matrix of the non-linear activation function.

$$\left\| \frac{\partial \mathcal{L}}{\partial \mathbf{h}_t} \right\| = \left\| \frac{\partial \mathcal{L}}{\partial \mathbf{h}_T} \prod_{k=t}^{T-1} \mathbf{J}_{k+1} \mathbf{W}^T \right\| \tag{C.5}$$

$$\leq \left\| \frac{\partial \mathcal{L}}{\partial \mathbf{h}_T} \right\| \prod_{k=t}^{T-1} \left\| \mathbf{J}_{k+1} \mathbf{W}^T \right\| \tag{C.6}$$

$$\leq \left\| \frac{\partial \mathcal{L}}{\partial \mathbf{h}_T} \right\| \|\mathbf{W}\|^{T-t} \prod_{k=t}^{T-1} \|\mathbf{J}_{k+1}\| \tag{C.7}$$

From equation C.7 it is clear the norm of the gradient is exponentially dependent upon two factors along the time horizon:

- The norm of the Jacobian matrix of the non-linear activation function $\|\mathbf{J}_{k+1}\|$.
- The norm of the hidden to hidden weight matrix $\|\mathbf{W}\|$.

These two factors are causing the vanishing and exploding gradient problem.

Since the gradient of the standard non-linear activation functions such as tanh and ReLU are bounded between [0, 1], $\|\mathbf{J}_{k+1}\|$ does not contribute to the exploding gradient problem but it can still cause vanishing gradient problem.

C.2 Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997)

LSTM networks presented an elegant solution to the vanishing and exploding gradients through the introduction of gating mechanism. Apart from the standard hidden state in RNN, LSTM introduced one more state called cell state $\mathbf{c}_t$ for controlling the flow of information along the time. LSTM has three different gates whose functionality is described as follows:

- Forget gate ($\mathbf{W}_f, \mathbf{U}_f, \mathbf{b}_f$): Decides what information to keep and erase from the previous cell state.
- Input gate ($\mathbf{W}_i, \mathbf{U}_f, \mathbf{b}_i$): Decides what new information should be added to the cell state.
- Output gate ($\mathbf{W}_o, \mathbf{U}_o, \mathbf{b}_o$): Decides which information from the cell state is going to the output.

In addition to the gates, LSTM prepares candidates for the information from the input gate that might get added to the cell state through the action of input gate. Let's denote the parameters describing the function that prepares this candidate information as $\mathbf{W}_c, \mathbf{U}_c, \mathbf{b}_c$.

Given a sequence of T input vectors: $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{T-1}$, at a time step t LSTM performs the following:

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{h}_{t-1} + \mathbf{U}_f \mathbf{x}_t + \mathbf{b}_f) \tag{C.8}$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{h}_{t-1} + \mathbf{U}_i \mathbf{x}_t + \mathbf{b}_i) \tag{C.9}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{h}_{t-1} + \mathbf{U}_o \mathbf{x}_t + \mathbf{b}_o) \tag{C.10}$$

$$\hat{\mathbf{c}}_t = \tau(\mathbf{W}_c \mathbf{h}_{t-1} + \mathbf{U}_c \mathbf{x}_t + \mathbf{b}_c) \tag{C.11}$$

$$\mathbf{c}_t = \mathbf{c}_{t-1} \odot \mathbf{f}_t + \hat{\mathbf{c}}_t \odot \mathbf{i}_t \tag{C.12}$$

$$\mathbf{h}_t = \tau(\mathbf{c}_t) \odot \mathbf{o}_t \tag{C.13}$$

where $\sigma(.)$ and $\tau(.)$ are the point-wise sigmoid and tanh functions. $\odot$ indicates element-wise multiplication. The first three are gating operations and the 4th one prepares the candidate information. The 5th operation updates the cell-state and finally in the 6th operation the output gate decided what should go into the current hidden state.

C.3 Unitary evolution RNN (Arjovsky et al., 2016)

Unitary evolution RNN (uRNN) proposed to solve the vanishing and exploding gradients through a unitary recurrent matrix, which is for the form:

$$\mathbf{W} = \mathbf{D}_3 \mathbf{R}_2 \mathscr{F}^{-1} \mathbf{D}_2 \Pi \mathbf{R}_1 \mathscr{F} \mathbf{D}_1. \tag{C.14}$$

Where:

- $\mathbf{D}_1, \mathbf{D}_2, \mathbf{D}_3$: Diagonal matrices whose diagonal entries are of the from $\mathbf{D}_{kk} = e^{i\theta_k}$, implies each matrix have N parameters, $(\theta_0, \ldots, \theta_{N-1})$.
- $\mathscr{F}$ and $\mathscr{F}^{-1}$: Fast Fourier operator and inverse fast Fourier operator respectively.
- $\mathbf{R}_1, \mathbf{R}_2$: Householder reflections. $R = \mathbf{I} - 2\frac{vv^H}{\|v\|}$, where $v \in \mathbb{C}^N$.

The total number of parameters for this uRNN operator is $7N$ and the matrix vector can be done $Nlog(N)$ time. It is parameter efficient and fast but not flexible and suffers from the retention of noise and difficulty in optimization due its unitarity.

C.4 Full capacity unitary RNN (Wisdom et al., 2016)

Full capacity unitary RNN (FC uRNN) does optimization on the full unitary set instead on a subset like uRNN. That is FC uRNN's recurrent matrix $\mathbf{W} \in U(N)$. There are several challenges in optimization over unitary manifold especially when combined with stochastic gradient method. The primary challenge being the optimization cost is $\mathcal{O}(N^3)$ per step.

C.5 Orthogonal RNN (Mhammedi et al., 2016)

Orthogonal RNN (oRNN) parametrizes the recurrent matrices using Householder reflections.

$$\mathbf{W} = \mathcal{H}_N(\mathbf{v}_N)...\mathcal{H}_{N-K+1}(\mathbf{v}_{N-k+1}). \tag{C.15}$$

where

$$\mathcal{H}_K(\mathbf{v}_K) = \begin{bmatrix} \mathbf{I}_{N-K} & 0 \\ 0 & \mathbf{I}_K - 2\frac{\mathbf{v}_K\mathbf{v}_K^H}{\|\mathbf{v}_K\|} \end{bmatrix} \tag{C.16}$$

and

$$\mathcal{H}_1(\mathbf{v}) = \begin{bmatrix} \mathbf{I}_{N-1} & 0 \\ 0 & \mathbf{v} \in \{-1, 1\} \end{bmatrix} \tag{C.17}$$

where $\mathbf{v}_K \in \mathbb{R}^K$. The number of parameters in this parameterization is $\mathcal{O}(NK)$. When $N = K = 1$ and $v = 1$, it spans the rotation subset and when $v = -1$, it spans the full reflection subset.

C.6 Properties of Kronecker matrix (Van Loan, 2000)

Consider a matrix $\mathbf{W} \in \mathbb{C}^{N\times N}$ factorized as a Kronecker product of F matrices $\mathbf{W}_1, \ldots, \mathbf{W}_F$,

$$\mathbf{W} = \mathbf{W}_1 \otimes \cdots \otimes \mathbf{W}_F = \otimes_{f=1}^{F} \mathbf{W}_f. \tag{C.18}$$

Where each $\mathbf{W}_f \in \mathbb{C}^{P_f \times Q_f}$ respectively and $\prod_{f=1}^{F} P_i = \prod_{f=1}^{F} Q_f = N$. $\mathbf{W}_f$'s are called as Kronecker factors.

If the factors $\mathbf{W}_i$'s are $\left\{\begin{array}{l}\text{Nonsingular}\\ \text{Symmetric}\\ \text{Stochatsic}\\ \text{Orthogonal}\\ \text{Unitary}\\ \text{PSD}\\ \text{Toeplitz}\end{array}\right\}$ then $\mathbf{W}$ is $\left\{\begin{array}{l}\text{Nonsingular}\\ \text{Symmetric}\\ \text{Stochatsic}\\ \text{Orthogonal}\\ \text{Unitary}\\ \text{PSD}\\ \text{Block Toeplitz}\end{array}\right\}$

Theorem 4. *If* $\forall f \in 1, \ldots, F$, $\mathbf{W}_f$ *is unitary then* $\mathbf{W}$ *is also unitary.*

Proof.

$$\mathbf{W}^H\mathbf{W} = (\mathbf{W}_1 \otimes \cdots \otimes \mathbf{W}_F)^H(\mathbf{W}_1 \otimes \cdots \otimes \mathbf{W}_F) \tag{C.19}$$

$$= (\mathbf{W}_1^H \otimes \cdots \otimes \mathbf{W}_F^H)(\mathbf{W}_1 \otimes \cdots \otimes \mathbf{W}_F) \tag{C.20}$$

$$= \mathbf{W}_1^H\mathbf{W}_1 \otimes \cdots \otimes \mathbf{W}_F^H\mathbf{W}_F = \mathbf{I}. \tag{C.21}$$

□

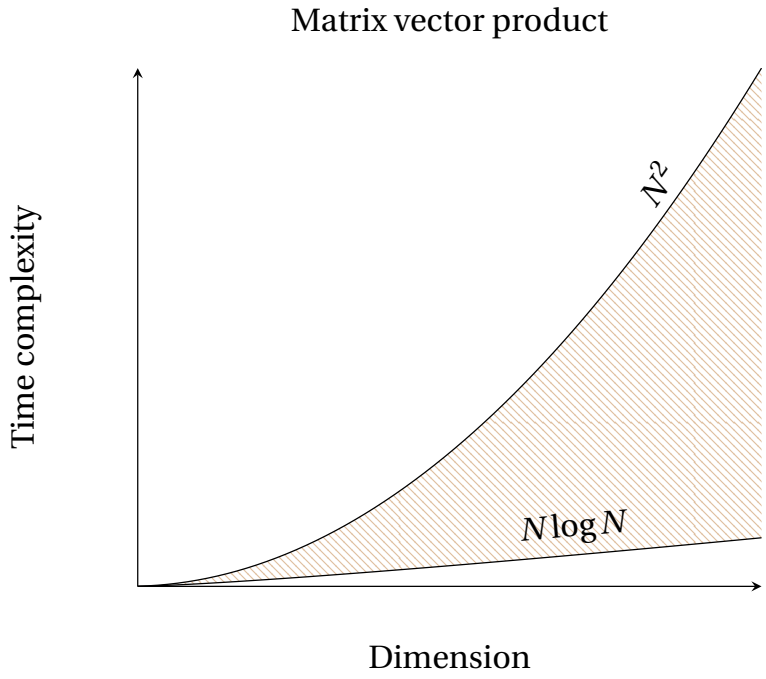


Figure C.1: Graph illustrating the time complexity of dense vector product with Kronecker factored square matrices as a function of the vector dimension. Kronecker factorization allows a fine-grained control over the number of parameters and hence the computational efficiency. By choosing the number of factors and the size of each factors we can span all the Kronecker matrices in the shaded region of the graph and thus can control the amount of computation we want to invest in.

C.7 Product between a dense matrix and a Kronecker matrix

For simplicity here we use real number notations. Consider a dense matrix $\mathbf{X} \in \mathbb{R}^{M \times K}$ and a Kronecker factored matrix $\mathbf{W} \in \mathbb{R}^{N \times K}$. That is $\mathbf{W} = \otimes_{f=1}^{F} \mathbf{W}_f$, where each $\mathbf{W}_f \in \mathbb{R}^{P_f \times Q_f}$ respectively and $\prod_{f=1}^{F} P_f = N$ and $\prod_{f=1}^{F} Q_f = K$. Let us illustrate the matrix product $\mathbf{X}\mathbf{W}^T$ resulting in a matrix $\mathbf{Y} \in \mathbb{R}^{M \times N}$.

$$\mathbf{Y} = \mathbf{X}\mathbf{W}^T. \tag{C.22}$$

The computational complexity first expanding the Kronecker factored matrix and then computing the matrix product is $\mathcal{O}(MNK)$. This can be reduced by exploiting the recursive definition of Kronecker matrices. For examples when $N = K$ and $\forall_f \{P_f = Q_f = 2\}$, the matrix product can be computed in $\mathcal{O}(MN \log N)$ time instead of $\mathcal{O}(MN^2)$.

The matrix product in equation C.22 can be recursively defined as:

$$\mathbf{Y} = (\ldots(\mathbf{X} \odot \mathbf{W}_1^T) \otimes \cdots \otimes \mathbf{W}_F^T). \tag{C.23}$$

Please note that the binary operator $\odot$ is not the standard matrix multiplication operator but instead it denotes a strided matrix multiplication. The stride is computed according to the algebra of Kronecker matrices. Let us define $\mathbf{Y}$ recursively:

$$\mathbf{Y}_1 = \mathbf{X} \odot \mathbf{W}_1 \tag{C.24}$$

$$\mathbf{Y}_f = \mathbf{Y}_{f-1} \odot \mathbf{W}_f. \tag{C.25}$$

Combining equation C.27 and C.25

$$\mathbf{Y} = \mathbf{Y}_F = (\ldots(\mathbf{X} \odot \mathbf{W}_1^T) \otimes \cdots \otimes \mathbf{W}_F^T). \tag{C.26}$$

We use the above notation for $\mathbf{Y}$ in the algorithm. That is the algorithm illustrated here will cache all the intermediate outputs $(\mathbf{Y}_1, \ldots, \mathbf{Y}_F)$ instead of just $\mathbf{Y}_F$. These intermediate outputs are later used to compute the gradients during the back-propagation and this cache will save some computation in that case. If the model is just being used for inference then the algorithm can the organized in such a way that we do not need to cache the intermediate outputs and thus save memory.

Algorithm for computing the product between a dense matrix and a Kronecker factored matrix C.27 is given in 6. All the matrices are assumed to be stored in row major order. For simplicity the algorithm is illustrated in a serial fashion. Please note the lines 4 to 15 except lines 9-11 can be trivially parallelized as they writes to independent memory locations. The GPU implementation exploits this fact. Please also note that in the algorithm all the indices start from 0 instead of 1 in order to be consistent with the C/C++ programming language.

Algorithm 6 Dense matrix product with a Kronecker matrix, $\mathbf{Y} = (\dots(\mathbf{X}\mathbf{W}_0^T) \otimes \cdots \otimes \mathbf{W}_{F-1}^T)$

Input: Dense matrix $\mathbf{X} \in \mathbb{R}^{M \times K}$, Kronecker factors $\{\mathbf{W}_0, \dots, \mathbf{W}_{F-1}\} : \mathbf{W}_f \in \mathbb{R}^{P_f \times Q_f}$, Size of each Kronecker factors $\{(P_0, Q_0), \dots, (P_{F-1}, Q_{F-1})\} : \prod_{f=0}^{F-1} P_f = N, \prod_{f=0}^{F-1} Q_f = K$,

Output: Output matrix $Y_{F-1} \in \mathbb{R}^{M \times N}$

for $f = 0$ to $F-1$ **do**
  $stride = K/Q_f$, $index = 0$
  for $m = 0$ to $M-1$ **do**
    $\mathbf{X}_m = \mathbf{X} + m \times K$
    for $p = 0$ to $P_f - 1$ **do**
      for $s = 0$ to $stride - 1$ **do**
        $\mathbf{Y}_f[index] = 0$
        for $q = 0$ to $Q_k - 1$ **do**
          $\mathbf{Y}_f[index] = \mathbf{Y}_f[index] + \mathbf{X}_m[q \times stride + s] \times \mathbf{W}_f[p \times Q_f + q]$
        end for
        $index = index + 1$
      end for
    end for
  end for
  $K = stride$, $M = M \times P_f$
  $X = Y_f$
end for

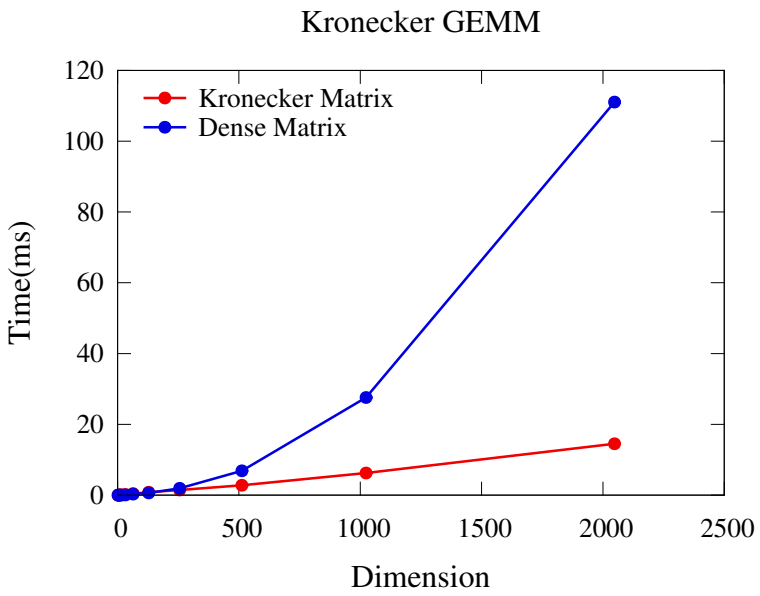


Figure C.2: Comparison of matrix product between a dense matrix and a Kronecker matrix with 2x2 factors vs Dense matrix matrix product (GEMM). We use the standard BLAS notations (M = 64, Dimension = N = K).

C.8 Gradient computation in a Kronecker layer

Following the notations from the above section C.7, here we illustrate the algorithm for computing the gradients in a Kronecker layer. To be clear and concrete the Kronecker layer does the following computation in the forward pass(equation C.25).

$$\mathbf{Y} = \mathbf{Y}_F = (\dots(\mathbf{X} \odot \mathbf{W}_1^T) \otimes \cdots \otimes \mathbf{W}_F^T). \tag{C.27}$$

That is, the Kronecker layer is parametrized by a Kronecker factored matrix $\mathbf{W} = \otimes_{f=1}^{F} \mathbf{W}_f$ stored as it factors $\{\mathbf{W}_1, \dots, \mathbf{W}_F\}$ and it takes an input $\mathbf{X}$ and produces output $\mathbf{Y} = \mathbf{Y}_{F-1}$ using the algorithm 6.

The following algorithm 7 computes the Gradient of the Kronecker factors: $\{\mathbf{gW}_1, \dots, \mathbf{gW}_F\}$ and the Jacobian of the input matrix $\mathbf{gX}$ given the Jacobian of the output matrix: $\mathbf{gY} = \mathbf{gY}_F$. The algorithm is illustrated with all the indices starting from 0 instead of 1 in order to be consistent with the C/C++ programming language.

Algorithm 7 Gradient computation in a Kronecker layer.

Input: Input matrix $\mathbf{X} \in \mathbb{R}^{M \times K}$, Kronecker factors $\{\mathbf{W}_0, \ldots, \mathbf{W}_{F-1}\} : \mathbf{W}_f \in \mathbb{R}^{P_f \times Q_f}$, Size of each Kronecker factors $\{(P_0, Q_0), \ldots, (P_{F-1}, Q_{F-1})\} : \prod_{f=0}^{F-1} P_f = N, \prod_{f=0}^{F-1} Q_f = K$, All intermediate output matrices from the forward pass: $\{\mathbf{Y}_0, \ldots, \mathbf{Y}_{F-1}\}$, Jacobian of output matrix: $\mathbf{gY}_{F-1} \in \mathbb{R}^{M \times N}$

Output: Gradient of Kronecker factors: $\{\mathbf{gW}_0, \ldots, \mathbf{gW}_{F-1}\}$ and Jacobian of input matrix: $\mathbf{gX} \in \mathbb{R}^{M \times K}$.

$T = M \times N$, $stride P = 1$, $stride Q = 1$
$\mathbf{gY} = \mathbf{gY}_{F-1}$
for $f = F-1$ to 0 **do**
  $R = stride P \times P_f$, $S = stride Q \times Q_f$, $T = T / P_f$
  $\mathbf{Z} = nullptr$, $\mathbf{gZ} = nullptr$
  if $f == 0$ **then**
    $\mathbf{Z} = \mathbf{X}$, $\mathbf{gZ} = \mathbf{gX}$
  else
    $\mathbf{gZ} = \mathbf{Y}_{f-1}$, $\mathbf{Z} = \mathbf{gZ}$
  end if
  $index = 0$
  for $t = 0$ to $T-1$ **do**
    $\mathbf{Z}_t = \mathbf{Z} + t \times S$
    for $p = 0$ to $P_f - 1$ **do**
      for $s = 0$ to $stride Q - 1$ **do**
        for $q = 0$ to $Q_k - 1$ **do**
          $\mathbf{gW}_f[p \times Q_k + 1] = \mathbf{gW}_f[p \times Q_k + 1] + \mathbf{Z}_t[q \times stride Q + s] \times \mathbf{gY}[index]$
        end for
        $index = index + 1$
      end for
    end for
  end for
  $index = 0$
  for $t = 0$ to $T-1$ **do**
    $\mathbf{gY}_t = \mathbf{gY} + t \times R$
    for $p = 0$ to $P_f - 1$ **do**
      for $s = 0$ to $stride Q - 1$ **do**
        $\mathbf{gZ}[index] = 0$
        for $q = 0$ to $Q_k - 1$ **do**
          $\mathbf{gZ}[index] = \mathbf{gZ}[index] + \mathbf{gY}[q \times stride Q + s] \times \mathbf{W}_f[q \times P_f + q]$
        end for
        $index = index + 1$
      end for
    end for
  end for
  $\mathbf{gY} = \mathbf{gZ}$ //We reuse the memory for the intermediate outputs to store the gradients.
  $stride Q = S$, $stride P = R \times Q_f / P_f$
end for

Bibliography

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Ejaz Ahmed, Michael Jones, and Tim K Marks. An improved deep learning architecture for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3908–3916, 2015.

Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pages 1120–1128, 2016.

Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014.

Jimmy Ba, Roger Grosse, and James Martens. Distributed second-order optimization using kronecker-factored approximations. 2016.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Davide Baltieri, Roberto Vezzani, and Rita Cucchiara. 3dpes: 3d people dataset for surveillance and forensics. In *Proceedings of the 2011 joint ACM workshop on Human gesture and behavior understanding*, pages 59–64. ACM, 2011.

Antonio Valerio Miceli Barone. Low-rank passthrough neural networks. *arXiv preprint arXiv:1603.03116*, 2016.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

James Bergstra, Olivier Breuleux, Pascal Lamblin, Razvan Pascanu, Olivier Delalleau, Guillaume Desjardins, Ian Goodfellow, Arnaud Bergeron, Yoshua Bengio, and Pack Kaelbling. Theano: Deep learning on gpus with python. 2011.

A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *The Journal of Machine Learning Research*, 6:1579–1619, 2005.

Léon Bottou. Two big challenges in machine learning, 2015. URL http://icml.cc/2015/invited/LeonBottouICML2015.pdf.

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*, 2012.

Olivier Bousquet and Léon Bottou. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pages 161–168, 2008.

Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2(Mar):499–526, 2002.

Lev M. Bregman, Yair Censor, Simeon Reich, and Yael Zepkowitz-Malachi. Finding the projection of a point onto the intersection of convex sets via projections onto half-spaces. *J. Approx. Theory*, 124(2):194–218, October 2003. ISSN 0021-9045. doi: 10.1016/j.jat.2003.08.004. URL http://dx.doi.org/10.1016/j.jat.2003.08.004.

C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *Computational Intelligence and AI in Games, IEEE Transactions on*, 4(1):1–43, 2012.

Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, 2013.

Olivier Canévet. Object detection with active sample harvesting. 2017.

Olivier Canévet, Cijo Jose, and Francois Fleuret. Importance sampling tree for large-scale empirical expectation. In *International Conference on Machine Learning*, pages 1454–1462, 2016.

Olivier Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19(5): 1155–1178, 2007.

Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. *The Journal of Machine Learning Research*, 11:1109–1135, 2010.

Dapeng Chen, Zejian Yuan, Gang Hua, Nanning Zheng, and Jingdong Wang. Similarity learning on an explicit polynomial kernel feature map for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1565–1573, 2015a.

Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *International Conference on Machine Learning*, pages 2285–2294, 2015b.

D. S. Cheng, M. Cristani, M. Stoppa, L. Bazzani, and V. Murino. Custom pictorial structures for re-identification. In *British Machine Vision Conference (BMVC)*, 2011.

Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 539–546, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2372-2. doi: 10.1109/CVPR.2005.202. URL http://dx.doi.org/10.1109/CVPR.2005.202.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Junyoung Chung, Caglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *ICML*, pages 2067–2075, 2015.

Moustapha Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured prediction models. *arXiv preprint arXiv:1707.05373*, 2017a.

Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. *arXiv preprint arXiv:1704.08847*, 2017b.

Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011.

Matthieu Courbariaux, Jean-Pierre David, and Yoshua Bengio. Low precision storage for deep learning. *Arxiv: 1412.7024*, 2014.

Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th annual International Conference on Machine Learning (ICML-07)*, pages 209–216. ACM, 2007.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 2009.

Misha Denil, Babak Shakibi, Laurent Dinh, Nando de Freitas, et al. Predicting parameters in deep learning. In *Advances in Neural Information Processing Systems*, pages 2148–2156, 2013.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Alhussein Fawzi. *Robust Image Classification: Analysis and Applications*. PhD thesis, Ecole Polytechnique Fédérale de Lausanne, 2016.

Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

F. Fleuret and G. Blanchard. Pattern recognition from one example by Chopping. In *Proceedings of the international conference on Neural Information Processing Systems (NIPS)*, pages 371–378, 2005. URL http://fleuret.org/papers/fleuret-blanchard-nips2005.pdf.

F. Fleuret and D. Geman. Stationary features and cat detection. 9:2549–2578, 2008.

John S Garofolo, Lori F Lamel, William M Fisher, Jonathon G Fiscus, and David S Pallett. Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1. *NASA STI/Recon technical report n*, 93, 1993.

S. Gelly, Y. Wang, R. Munos, and O. Teytaud. Modification of UCT with Patterns in Monte-Carlo Go. Technical Report RR-6062, INRIA, 2006.

S. Gong, M. Cristani, S. Yan, and Loy, editors. *Person Re-Identification*. Advances in Computer Vision and Pattern Recognition. Springer, 2014.

Ian Goodfellow. Efficient per-example gradient computations. *arXiv preprint arXiv:1510.01799*, 2015.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005.

Douglas Gray and Hai Tao. Viewpoint invariant pedestrian recognition with an ensemble of localized features. In *Computer Vision–ECCV 2008*, pages 262–275. Springer, 2008.

Roger Grosse and James Martens. A kronecker-factored approximate fisher matrix for convolution layers. In *International Conference on Machine Learning*, pages 573–582, 2016.

David Ha, Andrew Dai, and Quoc Le. Hypernetworks. 2016.

Moritz Hardt, Benjamin Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*, 2015.

Moritz Hardt, Tengyu Ma, and Benjamin Recht. Gradient descent learns linear dynamical systems. *arXiv preprint arXiv:1609.05191*, 2016.

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.

Mikael Henaff, Arthur Szlam, and Yann LeCun. Orthogonal RNNs and long-memory tasks. *arXiv preprint arXiv:1602.06662*, 2016.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

Sepp Hochreiter. *Untersuchungen zu dynamischen neuronalen Netzen*. PhD thesis, diploma thesis, Technische universität münchen, 1991.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9 (8):1735–1780, 1997.

Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148(34):13, 2001.

Li Jing, Yichen Shen, Tena Dubcek, John Peurifoy, Scott Skirlo, Yann LeCun, Max Tegmark, and Marin Soljačić. Tunable efficient unitary neural networks (EUNN) and their application to RNNs. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1733–1741, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/jing17a.html.

Cijo Jose and François Fleuret. Scalable metric learning via weighted approximate rank component analysis. In *European Conference on Computer Vision*, pages 875–890. Springer, 2016.

Cijo Jose, Moustpaha Cisse, and Francois Fleuret. Kronecker recurrent units. *arXiv preprint arXiv:1705.10142*, 2017.

Z. Kalal, J. Matas, and K. Mikolajczyk. Weighted sampling for large-scale boosting. In *BMVC*, 2008.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014a.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014b.

Martin Köstinger, Martin Hirzer, Paul Wohlhart, Peter M Roth, and Horst Bischof. Large scale metric learning from equivalence constraints. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2288–2295. IEEE, 2012.

A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Computer Science Department, University of Toronto, Tech. Rep*, 1(4):7, 2009.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. pages 1097–1105, 2012a.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012b.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012c.

Quoc Le, Tamás Sarlós, and Alex Smola. Fastfood-approximating kernel expansions in loglinear time. In *Proceedings of the international conference on machine learning*, 2013.

Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

L. Lefakis and F. Fleuret. Reservoir boosting : Between online and offline ensemble learning. In *Proceedings of the international conference on Neural Information Processing Systems (NIPS)*, 2013.

Wei Li, Rui Zhao, and Xiaogang Wang. Human reidentification with transferred metric learning. In *ACCV (1)*, pages 31–44, 2012.

Wei Li, Rui Zhao, Tong Xiao, and Xiaogang Wang. Deepreid: Deep filter pairing neural network for person re-identification. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 152–159. IEEE, 2014.

Zhen Li, Shiyu Chang, Feng Liang, Thomas S Huang, Liangliang Cao, and John R Smith. Learning locally-adaptive decision functions for person verification. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3610–3617. IEEE, 2013.

Shengcai Liao and Stan Z Li. Efficient psd constrained asymmetric metric learning for person re-identification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3685–3693, 2015.

Shengcai Liao, Zhipeng Mo, Yang Hu, and Stan Z Li. Open-set person re-identification. *arXiv preprint arXiv:1408.0872*, 2014.

Shengcai Liao, Yang Hu, Xiangyu Zhu, and Stan Z Li. Person re-identification by local maximal occurrence representation and metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2197–2206, 2015.

Daryl Lim and Gert Lanckriet. Efficient learning of mahalanobis metrics for ranking. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1980–1988, 2014.

G. Loosli, S. Canu, and L. Bottou. Training invariant support vector machines using selective sampling. In Léon Bottou, Olivier Chapelle, Dennis DeCoste, and Jason Weston, editors, *Large Scale Kernel Machines*, pages 301–320. MIT Press, Cambridge, MA., 2007. URL http://leon.bottou.org/papers/loosli-canu-bottou-2006.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International Conference on Machine Learning*, pages 2408–2417, 2015.

Brian McFee and Gert R Lanckriet. Metric learning to rank. In *Proceedings of the 27th annual International Conference on Machine Learning (ICML-10)*, pages 775–782, 2010.

Paul Mermelstein. Distance measures for speech recognition, psychological and instrumental. *Pattern recognition and artificial intelligence*, 116:374–388, 1976.

Zakaria Mhammedi, Andrew Hellicar, Ashfaqur Rahman, and James Bailey. Efficient orthogonal parametrisation of recurrent neural networks using householder reflections. *arXiv preprint arXiv:1612.00188*, 2016.

Alexis Mignon and Frédéric Jurie. Pcca: A new approach for distance learning from sparse pairwise constraints. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2666–2672. IEEE, 2012.

Tomáš Mikolov. *Statistical Language Models Based on Neural Networks*. PhD thesis, Ph. D. thesis, Brno University of Technology, 2012.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Radford M Neal. Priors for infinite networks. In *Bayesian Learning for Neural Networks*, pages 29–53. Springer, 1996.

A. B. Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 12, pages 615–622, New York, NY, USA, 1962. Polytechnic Institute of Brooklyn.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

S. Paisitkriangkrai, C. Shen, and A. van den Hengel. Learning to rank in person re-identification with metric ensembles. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318, 2013.

Adam Paszke, Sam Gross, and Soumith Chintala. Pytorch, 2017.

Sateesh Pedagadi, James Orwell, Sergio Velastin, and Boghos Boghossian. Local fisher discriminant analysis for pedestrian re-identification. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, pages 3318–3325, Washington, DC, USA, 2013. IEEE Computer Society. ISBN 978-0-7695-4989-7. doi: 10.1109/CVPR.2013.426. URL http://dx.doi.org/10.1109/CVPR.2013.426.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

Peter M Roth, Martin Hirzer, Martin Köstinger, Csaba Beleznai, and Horst Bischof. Mahalanobis distance learning for person re-identification. In *Person Re-Identification*, pages 247–267. Springer, 2014.

Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

D Sculley, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, and Michael Young. Machine learning: The high-interest credit card of technical debt. 2014.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

Uri Shalit, Daphna Weinshall, and Gal Chechik. Online learning in the embedded manifold of low-rank matrices. *The Journal of Machine Learning Research*, 13(1):429–458, 2012.

H. Ben Shitrit, J. Berclaz, F. Fleuret, , and P. Fua. Tracking Multiple People under Global Appearance Constraints. *International Conference on Computer Vision*, 2011.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the 21st annual International Conference on Machine Learning (ICML-04)*, page 104. ACM, 2004.

Nicolas Usunier, David Buffoni, and Patrick Gallinari. Ranking with ordered weighted pairwise classification. In *Proceedings of the 26th annual International Conference on Machine Learning (ICML-09)*, pages 1057–1064. ACM, 2009.

Charles F Van Loan. The ubiquitous kronecker product. *Journal of computational and applied mathematics*, 123(1):85–100, 2000.

Eugene Vorontsov, Chiheb Trabelsi, Samuel Kadoury, and Chris Pal. On orthogonality and learning recurrent networks with long term dependencies. *arXiv preprint arXiv:1702.00071*, 2017.

Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *The Journal of Machine Learning Research*, 10:207–244, 2009.

Jason Weston, Samy Bengio, and Nicolas Usunier. WSABIE: Scaling up to large vocabulary image annotation. In *IJCAI*, volume 11, pages 2764–2770, 2011.

Scott Wisdom, Thomas Powers, John Hershey, Jonathan Le Roux, and Les Atlas. Full-capacity unitary recurrent neural networks. In *Advances In Neural Information Processing Systems*, pages 4880–4888, 2016.

Yang Wu, Makoto Mukunoki, Takuya Funatomi, Michihiko Minoh, and Shihong Lao. Optimizing mean reciprocal rank for person re-identification. In *Advanced Video and Signal-Based Surveillance (AVSS), 2011 8th IEEE International Conference on*, pages 408–413. IEEE, 2011.

Yuhuai Wu, Elman Mansimov, Shun Liao, Roger Grosse, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. *arXiv preprint arXiv:1708.05144*, 2017.

Di Xie, Jiang Xiong, and Shiliang Pu. All you need is beyond a good init: Exploring better solution for training extremely deep convolutional neural networks with orthonormality and modulation. *arXiv preprint arXiv:1703.01827*, 2017.

Eric P Xing, Michael I Jordan, Stuart Russell, and Andrew Y Ng. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pages 505–512, 2002.

Fei Xiong, Mengran Gou, Octavia Camps, and Mario Sznaier. Person re-identification using kernel-based metric learning methods. In *Computer Vision–ECCV 2014*, pages 1–16. Springer, 2014.

Liu Yang and Rong Jin. Distance metric learning: A comprehensive survey. *Michigan State Universiy*, 2, 2006.

Zichao Yang, Marcin Moczulski, Misha Denil, Nando de Freitas, Alex Smola, Le Song, and Ziyu Wang. Deep fried convnets. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1476–1483, 2015.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Xu Zhang, Felix X Yu, Ruiqi Guo, Sanjiv Kumar, Shengjin Wang, and Shi-Fu Chang. Fast orthogonal projection based on kronecker product. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2929–2937, 2015.

P. Zhao and T. Zhang. Stochastic optimization with importance sampling. *arXiv preprint arXiv:1401.2753*, 2014.

Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, Jiahao Bu, and Qi Tian. Scalable person re-identification: A benchmark. *Computer Vision, IEEE International Conference on*, 2015.

Wei-Shi Zheng, Shaogang Gong, and Tao Xiang. Associating groups of people. In *British Machine Vision Conference (BMVC)*, volume 2, page 6, 2009.

Shuchang Zhou, Jia-Nan Wu, Yuxin Wu, and Xinyu Zhou. Exploiting local structures with the kronecker layer in convolutional networks. *arXiv preprint arXiv:1512.09194*, 2015.

Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. Recurrent highway networks. *arXiv preprint arXiv:1607.03474*, 2016.

Cijo Jose

Rue Marc-Morand 7, 1920 Martigny, Valais, Switzerland

(41) 762935669 cijo.jose@idiap.ch

Objective

A career in machine learning research.

Area of Research Interests

Learning embeddings, Neural networks, Kernel methods, Computational and statistical tradeoffs in machine learning, Software tools for machine learning.

Education

École Polytechnique Fédérale de Lausanne Lausanne, Switzerland. • Doctoral Student in Electrical Engineering., January 2014 - February 2018
Adviser: Dr. François Fleuret

Indian Institute of Technology Delhi Delhi, India. • Master of Technology in Computer Science., July 2010 - September 2012
Adviser: Dr. Manik Varma

College of Engineering Trivandrum Kerala, India. • Bachelor of Technology in Computer Science., September 2004 - June 2008

Publications

- Cijo Jose, and François Fleuret. Scalable Metric Learning via Weighted Approximate Rank Component Analysis. In *Proceedings of the European Conference in Computer Vision* (**ECCV**) *, Amsterdam, Netherlands,* October 2016.
- Olivier Canévet, Cijo Jose, and François Fleuret. Importance Sampling Tree for Large Scale Empirical Expectation. In *Proceedings of the International Conference on Machine Learning* (**ICML**) *, Newyork City, Newyork,* June 2016.
- Cijo Jose, Prasoon Goyal, Parv Agarwal, and Manik Varma. Local Deep Kernel Learning for Efficient Non-linear SVM Prediction. In *Proceedings of the International Conference on Machine Learning* (**ICML**) *, Atlanta, Georgia,* June 2013.

Preprints

- Cijo Jose, Moustapha Cissé and François Fleuret. Kronecker Recurrent Units. In *Arxiv* May 2017.

Professional Experience

Idiap Research Institute (December 2013 - Present)
Research Assistant

- Carrying out research in learning embeddings and importance sampling for the doctoral studies.

Facebook AI Research (January 2017 - April 2017)
Research Intern

- Conceptualised, designed and developed Kronecker recurrent units.

Microsoft Corporation (October 2012 - April 2013)
Research Intern

- Improved the research results on local deep kernel learning.
- Explored different machine learning models for the ad-serving system in Bing search engine.

Huawei Technologies Co. Ltd - Research and Development (October 2008 - July 2009)
Software Engineer

- Delivered the GPRS and 3G platform, called the service delivery platform(SDP) to the TATA-Docomo mobile service in India.
- Designed and developed a report system for the SDP platform.

Skills

Programming Languages: C, C++.
Scripting Languages: Python, Lua, Matlab, Shell Scripting.
Markup Languages: LaTeX, HTML, XML.