Learning Nearest-Neighbor Classifiers with Hyperkernels

Hua Ouyang
College of Computing
Georgia Institute of Technology
houyang@cc.gatech.edu

Alexander Gray
College of Computing
Georgia Institute of Technology
agray@cc.gatech.edu

Abstract

We consider improving the performance of k-Nearest Neighbor classifiers. A regularized kNN is proposed to learn an optimal dissimilarity function to substitute the Euclidean metric. The learning process employs hyperkernels and shares a similar regularization framework as support vector machines (SVM). Its performance is shown to be consistently better than kNN, and is competitive with SVM.

1 Introduction

k-Nearest Neighbor(kNN) Classifier is one of the simplest and most straightforward classifier. Nevertheless, it has been adopted in many applications, and in some circumstances can yield competitive results. The kNN classifier involves voting among the k "nearest" training samples that surround the test sample **x**. The local area that covers these k samples is called a *vicinity region*, denoted by $\mathcal{R}$. Careful studies reveal drawbacks of kNN that are due to the limited sample size N and some underlying assumptions.

The kNN classifier can be formulated as a generative method. It first estimates the conditional densities by $p(\mathbf{x}|y) \approx \frac{k_y}{Nv}$, then predicts according to the Bayes decision rule. Here k_y is the number of samples in $\mathcal{R}$ with label y, and v is the volume of $\mathcal{R}$. The weak consistency of kNN classifiers states that if k is allowed to grow and $k/N \to 0$ then they are weakly universally consistent [2]. Hence it puts a strong assumption that $v \to 0$.

Formations of vicinity regions for classical kNN classifiers are metric-based. The Euclidean metric is often employed to measure how "near" (or how "far") two samples are. Although not explicitly indicated, the underlying assumption for Euclidean metric is that the input space $\mathbb{X} = \mathbb{R}^D$ is *isotropic*, meaning that any direction in $\mathbb{X}$ is equally important. Consequently, $\mathcal{R}$ is a D-hypersphere centered at the test sample. This strong assumption is often violated in practice.

Since $p(\mathbf{x}|y)$ is estimated by taking average over the vicinity region $\mathcal{R}$. In order to obtain an accurate estimation, $\mathcal{R}$ must be small enough, such that $p(\mathbf{x}|y)$ does not change much within it. However, training samples are always sparse in practice. $\mathcal{R}$ can not be too small, otherwise it may contain no sample.

Learning a metric from training samples [2,3,4,5,6,7,8] has been shown to be a practical way to tackle the above problems. Most of these methods share the same idea: *instead of the Euclidean metric, an alternative distance metric is sought, such that the distances across decision boundaries (within-class distances d_w) are large, and/or the distances within a classes (between-class distances d_w) are small.*

2 Regularized k-Nearest Neighbor Classifier

The goal of the proposed *regularized k-nearest neighbor classifier* (RkNN) is to learn an alternative dissimilarity $d(\mathbf{x}_i, \mathbf{x}_j) : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ in substitution of the Euclidean metric. Unlike metric learning methods, RkNN is explicitly regularized.

In order to minimize the empirical error, metric learning and subspace methods try to find an explicit representation of the feature space, where they can minimize the overlapped area of $p(d_w)$ and $p(d_b)$ by pushing the two densities apart. RkNN attempts to approach this goal from an different way: it learns a suitable measure of dissimilarity $d(\mathbf{x}_i, \mathbf{x}_j)$ such that the overlapped area is *directly* penalized.

We want to put the proposed RkNN into the framework of regularized empirical risk minimization such that the complexity of learned dissimilarity functions can be fully controlled. More precisely, we want to fit each ordered pair of training samples $\{\dot{\mathbf{x}}_i, \mathbf{x}_j\}$ with an appropriate dissimilarity $d(\dot{\mathbf{x}}_i, \mathbf{x}_j)$, while at the same time use Tikhonov's regularization to control the capacities of $d(\dot{\mathbf{x}}_i, \mathbf{x}_j)$. Classification can then be carried out as the classical kNN using the new dissimilarity measure.

We propose to solve the following optimization problem:

$$\min_{d \in \mathbb{H}} \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} L\big(y_i y_j, d(\dot{\mathbf{x}}_i, \mathbf{x}_j)\big) + \lambda \|d\|_{\mathbb{H}}^2, \tag{1}$$

where L is the loss function, the regularizer $\|d\|_{\mathbb{H}}^2$ is chosen to be a squared norm in the hypothesis space $\mathbb{H}$, and $\lambda > 0$ is an adjustable positive number that controls the trade-off between the fitting of training samples and the regularizer.

Eq.(1) is similar to the regularized risk of support vector machines (SVM). The specific loss function $L\big(y_i y_j, d(\dot{\mathbf{x}}_i, \mathbf{x}_j)\big)$ is relaxed to the *hinge loss*:

$$L\big(y_i y_j, d(\dot{\mathbf{x}}_i, \mathbf{x}_j)\big) = \begin{cases} 0, & \text{if } \ y_i y_j d(\dot{\mathbf{x}}_i, \mathbf{x}_j) \le -1 \\ 1 + y_i y_j d(\dot{\mathbf{x}}_i, \mathbf{x}_j), & \text{if } \ y_i y_j d(\dot{\mathbf{x}}_i, \mathbf{x}_j) > -1 \end{cases}. \tag{2}$$

2.1 Convex Optimization for RkNN

By combining Eq.(1) and Eq.(2), and by using the definitions of hyperkernel and hyper-RKHS [10], the objective function of RkNN can be written as:

$$\begin{aligned} &\min_{d \in \underline{\mathbb{H}}} \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \xi_{ij} + \lambda \|d\|_{\underline{\mathbb{H}}}^2 \\ &\text{subject to: } y_i y_j d(\dot{\mathbf{x}}_i, \mathbf{x}_j) \le \xi_{ij} - 1, \\ &\qquad \xi_{ij} \ge 0, \text{for all } i, j = 1 \cdots N. \end{aligned} \tag{3}$$

where we introduce a slack variable, since the hinge loss function is non-differentiable.

Utilizing the representer theom in Hyper-RHKS [9] and adding an unregularized bias $b \in \mathbb{R}$, we obtain the *primal problem*:

$$\begin{aligned} &\min_{C \in \mathbb{R}^{N^2 \times 1},\ \xi_{ij} \in \mathbb{R}} \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \xi_{ij} + \lambda \mathbf{c}^T \underline{K} \mathbf{c} \\ &\text{subject to: } y_i y_j \left(\sum_{p=1}^{N} \sum_{q=1}^{N} c_{pq} K\big((\dot{\mathbf{x}}_i, \mathbf{x}_j), (\dot{\mathbf{x}}_p, \mathbf{x}_q)\big) + b \right) \le \xi_{ij} - 1, \\ &\qquad \xi_{ij} \ge 0, \text{for all } i, j = 1 \cdots N \end{aligned} \tag{4}$$

where **c** is a $N^2 \times 1$ vector, with the $(p \times N + q)$th item denoted by c_{pq}, and $\underline{K}$ is the hyperkernel matrix constructed from training examples.

By introducing Langrange multipliers $\boldsymbol{\alpha}$ and utilizing the KKT conditions to ensure strong duality, we can derive the *dual problem*:

$$\begin{aligned} \min_{\boldsymbol{\alpha}\in\mathbb{R}^{N^2\times 1}} \quad & -\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_{ij}+\frac{1}{2}\boldsymbol{\alpha}^T G\boldsymbol{\alpha} \\ \text{subject to:} \quad & \sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_{ij}=0, \\ & 0\le \alpha_{ij}\le \frac{1}{2\lambda N^2}, \text{for all } i,j=1\cdots N, \end{aligned} \tag{5}$$

where $G = Y\underline{K}Y^T$ and Y is a diagonal matrix, with the $(p\times N+q)$th diagonal element $Y_{p\times N+q} = y_p y_q$. This quadratic programming can be solved effectively by specialized toolboxes for SVM.

2.2 Hyperkernel Construction

The construction of hyperkernel is the most important step in achieving a good performance for RkNN. We propose a new family of hyperkernels tailored made for RkNN, named *product hyperkernels*. The construction is based on the following proposition proved in [9].

Proposition 2.1. *Let $k_a(\cdot,\cdot)$ and $k_b(\cdot,\cdot)$ be positive definite kernels, then $\forall \boldsymbol{x}_1, \boldsymbol{x}_1^{'}, \boldsymbol{x}_2, \boldsymbol{x}_2^{'} \in \mathbb{X}$, and $\forall \alpha,\beta>0$, $\left(k_a(\boldsymbol{x}_1,\boldsymbol{x}_2)\right)^{\alpha}\left(k_b(\boldsymbol{x}_1^{'},\boldsymbol{x}_2^{'})\right)^{\beta}$ or $\alpha k_a(\boldsymbol{x}_1,\boldsymbol{x}_2)+\beta k_b(\boldsymbol{x}_1^{'},\boldsymbol{x}_2^{'})$ can give a hyperkernel $\underline{k}$.*

3 Experimental Evaluation

3.1 Synthetic Data Sets

As explained in Section 1, classical kNN classifiers tend to give irregular decision boundaries when the number of training samples is limited. We use some illustrative tasks to demonstrate the regularization effect of RkNN.

The benefit of regularization is shown by a nonlinear separable task in Figure 1(a)(b), where 40 training data are sampled along 2 concentric circles. 1(a) is the result of 1NN, while (b) is the result of R1NN.

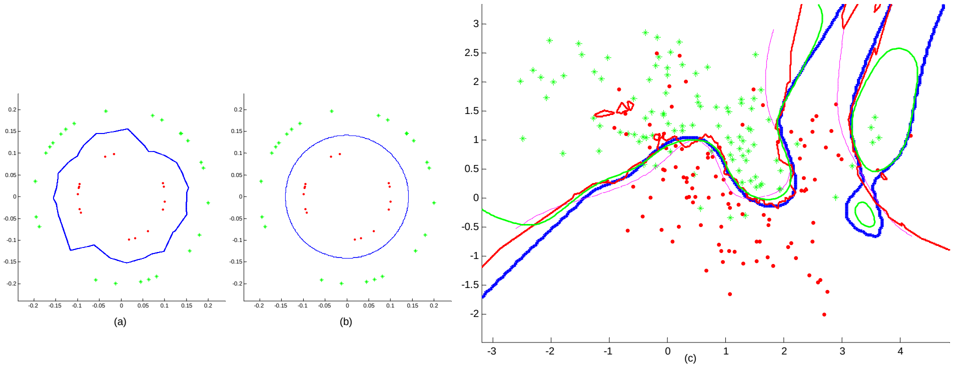

Figure 1: “2 concentric circles” solved by (a)1NN and (b)R1NN; (c) A nonseparable task solved by kNN(thick red line), RkNN(thick blue line), SVM(thick green line) and Bayes decision boundary(thin dotted purple line).

Figure 1(c) shows a 2-dimensional, 2-class nonseparable task solved by kNN, RkNN and SVM. 200 samples are generated from a 10-mixture gaussian distribution. It can be observed that RkNN and SVM have much smoother decision boundaries, while that of kNN is highly irregular. When we compare the results with the Bayes decision boundary, the difference between kNN and RkNN (or SVM) may not be that obvious, since it is a 2-dimensional task with 200 samples, and kNN can yield competitive result. The benefits of RkNN will be more obvious when the samples are drawn from a high-dimensional space, as will be demonstrated in Section 3.2.

3.2 Benchmark Data Sets

Six standard data sets are taken from the UCI Machine Learning Repository to evaluate the performance of RkNN in real-world applications, and to compare with kNN and SVM classifiers. Each data set is randomly divided into a training set (70%) and a test set (30%). The performance for each data set is taken as an average of three experiments, i.e. three random divisions of training and test sets. The training errors and testing errors are shown in Figure

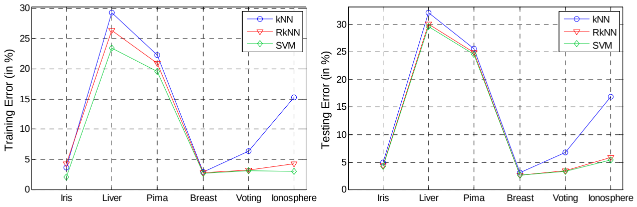

Figure 2: Compare training and testing errors of kNN, RkNN and SVM. Six data sets are employed.

It can be observed that the performance of RkNN is consistently better than kNN. The relative performance gain is more obvious for data sets with higher dimensions, e.g. “Voting”(dimension 16) and “Ionosphere”(dimension 34). RkNN’s performance is very close to SVM.

4 Conclusions

It is found that classical kNN suffers from the averaging scheme in the nonparametric estimation of density functions. It is more severe in small-sample-size and high-dimensional problems. The proposed RkNN classifier is shown to have the ability to circumvent these pitfalls. It learns asymmetric dissimilarity functions to substitute traditional Euclidean metric. The regularized learning framework employed by RkNN ensures it a good generalization ability. Experiments show that it has consistently better performance than kNN, especially in high-dimensional problems. Its performance is competitive with SVM. It is also shown that the proposed hyperkernels have good regularization properties in classification tasks.

Despite the adoption of SMO, the quadratic programming problem of RkNN will still be intractable if the number of training samples gets large. Further reducing the computational complexity will be the future work. It is also interesting to construct more effective hyperkernels.

References

[1] Luc Devroye, László Györfi & Gábor Lugosi, (1996) *A Probablistic Theory of Pattern Recognition*, Springer-Verlag, New York

[2] Trevor Hastie & Robert Tibshirani (1996) Discriminant adaptive nearest neighbor classification. *IEEE Trans. PAMI* **18**(6):607-616

[3] Eric P. Xing, Andrew Y. Ng, Michael I. Jordan & Stuart Russell (2002) Distance Metric Learning with Application to Clustering with Side-Information. *NIPS 15*

[4] James T. Kwok & Ivor W. Tsang (2003) Learning with Idealized Kernels. *ICML 2003*

[5] Goldberger, J., Roweis, S., Hinton, G. (2004) Neighbourhood Components Analysis. *NIPS 17*

[6] Kilian Q. Weinberger, John Blitzer & Lawrence K. Saul (2005) Distance Metric Learning for Large Margin Nearest Neighbor Classification. *NIPS 18*

[7] Amir Globerson & Sam Roweis (2005) Metric Learning by Collapsing Classes. *NIPS 18*

[8] Sumit Chopra, Raia Hadsell & Yann LeCun (2005) Learning a Similarity Metric Discriminatively, with Application to Face Verification. *CVPR 2005*

[9] Hua Ouyang & Alex Gray (2008) Learning Dissimilarities by ranking: from SDP to QP. *ICML 2008*

[10] Cheng Soon Ong, Alexander Smola & Robert Williamson (2005) Learning the Kernel with Hyperkernels. *JMLR*(6):1043-1071