Nearest Neighbors with Learned Distances for Phonetic Frame Classification

John Labiak

Department of Statistics, University of Chicago
Chicago, IL 60637 USA
labiak@galton.uchicago.edu

Karen Livescu

Toyota Technological Institute at Chicago
Chicago, IL 60637 USA
klivescu@ttic.edu

Abstract

Nearest neighbor-based techniques provide an approach to acoustic modeling that avoids the often lengthy and heuristic process of training traditional Gaussian mixture-based models. Here we study the problem of choosing the distance metric for a *k*-nearest neighbor (*k*-NN) phonetic frame classifier. We compare the standard Euclidean distance to two learned Mahalanobis distances, based on large-margin nearest neighbors (LMNN) and locality preserving projections (LPP). We use locality sensitive hashing for approximate nearest neighbor search to reduce the test time of *k*-NN classification. We compare the error rates of these approaches, as well as of baseline Gaussian mixture-based and multilayer perceptron classifiers, on the task of phonetic frame classification of speech from the TIMIT database. The *k*-NN classifiers outperform Gaussian mixture models, but not multilayer perceptrons. We find that the best *k*-NN classification performance is obtained using LPP, while LMNN is close behind.

Index Terms: Phonetic classification, nearest neighbors, distance learning, multilayer perceptrons

1. Introduction

State-of-the-art speech recognition systems often use Gaussian mixture models (GMMs) for acoustic modeling. However, the use of GMMs for acoustic modeling in speech recognition has drawbacks. GMMs usually require a very large number of components, whose parameters can take days or weeks to train. This training typically involves a fair amount of trial and error to initialize and grow the mixtures and to tune their hyperparameters (number of components, regularization parameters). In addition, GMMs are generative models, whereas discriminative approaches are often more successful in classification. Multilayer perceptron (MLP) phone classifiers are sometimes used as well; these are inherently discriminative, but integrating them into a speech recognizer is most often done by training a GMM density model of their outputs, reintroducing some of the drawbacks of GMMs.

One alternative that has recently been investigated [1, 2, 3] is nearest neighbor (NN) approaches. These approaches search through the training set for the examples most similar to a given test example, and use the result to either assign a class label to the test example or estimate a density at test time. NN-based approaches require minimal or no training, and use information from individual examples in the training set when making assignments, thereby avoiding the drawbacks of GMMs.

In this paper, we study the performance of a nearest neighbor-based approach, k-nearest neighbors (*k*-NN), on the task of phonetic frame classification of speech from the TIMIT database [4], and compare them to GMMs and MLPs. A crucial aspect of nearest neighbor-based approaches is the similarity or distance measure used to define the nearest neighbors. We investigate several distance metrics for *k*-NN classification, all of which can be defined as Mahalanobis distances: Given two d-dimensional feature vectors x_1 and x_2, their Mahalanobis distance is $\|x_1 - x_2\|_M = \sqrt{(x_1 - x_2)^T M (x_1 - x_2)}$, where M is a $d \times d$ symmetric positive-semidefinite matrix. A common choice is to set M to the identity matrix, in which case the metric is the standard Euclidean distance. We consider learning M from training data, using the large-margin nearest neighbors (LMNN) [5] and locality-preserving projections (LPP) [6] algorithms, both of which learn projections that are optimized, in some sense, with respect to the *k*-NN decision rule [7]. We also consider the performance of *k*-NN classifiers after reducing the dimensionality of the features either using linear discriminant analysis (LDA) or as part of the LMNN and LPP algorithms.

While the Euclidean distance metric often achieves good results, it is not likely to be optimal to use the same measure of similarity between examples for different classification tasks, such as phonetic classification and speaker identification, even when computing distances between the same fixed-length feature vectors in each task. We hope to improve performance by using a distance metric that is adapted to the specific classification task of interest. If new test examples come from the same distribution as the training examples, then we expect that the learned distance will be well adapted to the task of inferring the class labels of these test examples.

By using *k*-NN, we are shifting the burden from training time to test time. In order to reduce the test-time burden, we employ locality-sensitive hashing (LSH) [8], a fast approximate search technique which can be used to efficiently search through the training set at test time.

2. Methods

2.1. k-NN classification

The classification problem is defined as follows: Given a training set, containing N labeled examples $\{(x_i, y_i)\}_{i=1}^N$, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$, where $\mathcal{Y}$ denotes a finite set containing C discrete class labels, assign a new test example $x_0 \in \mathcal{X}$ to one of the C classes of $\mathcal{Y}$. Usually, $\mathcal{X}$ is taken to be $\mathbb{R}^d$, with $\mathcal{Y} \equiv \{1, ..., C\}$.

The *k*-nearest neighbors decision rule is based on the assumption that examples that are "similar," according to some reasonable similarity measure, are likely to belong to the same class. It is often assumed that a distance metric, $\mathcal{D} : (\mathcal{X} \times \mathcal{X}) \mapsto \mathbb{R}$, between examples serves as a reasonable proxy for their similarity. Thus, given a training set of labeled examples, the *k*-NN decision rule assigns a class label to a new test example by first finding the *k* closest examples in the training set, according to $\mathcal{D}$, to the test example. The test example is then predicted to belong to the class with the largest representation among those *k* nearest neighbors.

One type of distance metric is a Mahalanobis distance, which is a norm induced by a symmetric positive-semidefinite matrix $M \in \mathbb{R}^{d\times d}$, i.e. the distance is: $\mathcal{D}(x_1, x_2) = \|x_1 - x_2\|_M = \sqrt{(x_1 - x_2)^T M (x_1 - x_2)}$, where $M = M^T$ and $M \succeq 0$. We consider the *k*-NN decision rule for (1) $M \equiv I_d$, where I_d is the $d \times d$ identity matrix, corresponding to the Euclidean distance, and (2) $M \equiv L^T L$, where L is a linear transformation learned by a distance learning algorithm. Next we describe two such distance learning algorithms.

2.2. Learned distances

2.2.1. Large Margin Nearest Neighbors (LMNN)

The large margin nearest neighbors (LMNN) algorithm [5] is a supervised approach for learning the matrix of a Mahalanobis distance metric, $M \in \mathbb{R}^{d\times d}$, or, equivalently, a linear transformation, $L : \mathbb{R}^d \mapsto \mathbb{R}^p$ $(p \leq d)$, that is optimized with respect to *k*-NN classification. LMNN seeks a transformation such that, in the embedding space, the *k* nearest neighbors to an example are from the same class and examples from different classes are separated by a large margin. This gives rise to the cost function:

$$\sum_{ij} \quad \eta_{ij} \|L(x_i - x_j)\|^2$$

$$+c \sum_{ijl} \quad \eta_{ij}(1 - y_{il})[1 + \|L(x_i - x_j)\|^2 - \|L(x_i - x_l)\|^2]_+$$

where $[z]_+ = \max(z, 0)$,

$$y_{ij} = \begin{cases} 1 & \text{if } y_i = y_j; \\ 0 & \text{o.w.} \end{cases}$$

and $\eta_{ij} = 1$ if and only if x_j is an n "target" neighbor of x_i; that is, if x_j and x_i are from the same class ($y_{ij} = 1$) and x_j is among the n closest examples to x_i in the training set, as measured by the Euclidean distance. The parameter n is known as the "neighborhood parameter" of the LMNN algorithm (not to be confused with *k* in the *k*-NN decision rule).

The first term in the cost function penalizes large distances between an example and its n target neighbors. The second term penalizes small distances between an example and examples from different classes. In particular, the second term incorporates the idea of a margin, through its use of the hinge loss $[z]_+ = \max(z, 0)$, such that a penalty is incurred for examples from different classes whose distance to an example does not exceed, by one absolute unit of distance, the distance between the example and any one of its target neighbors. The cost function above can be reformulated as a semidefinite program (SDP). In this work we use a Matlab implementation provided by Kilian Weinberger [5].

2.2.2. Locality Preserving Projections (LPP)

The locality preserving projections (LPP) algorithm [6] learns a linear transformation $L : \mathbb{R}^d \mapsto \mathbb{R}^p$ $(p \leq d)$ that aims to preserve the neighborhood structure of the data. The neighborhood structure is encoded as a graph, in which two nodes, x_i and x_j, are connected by an edge if they are *n*-nearest neighbors (*n* is again the "neighborhood parameter" of the algorithm), with edge weight w_{ij} encoding their similarity. Typically, $w_{ij} = e^{-\frac{\|x_i - x_j\|^2}{t}}$ for some t. The basic algorithm considered here is unsupervised, i.e. the training labels are unknown; although a supervised extension exists.

The LPP objective function is a weighted sum of pairwise distances between points in the embedding space, with weights w_{ij}. This objective function incurs a penalty when neighboring points are mapped to vectors that are far apart. The algorithm seeks a linear transformation matrix, $L \in \mathbb{R}^{p\times d}$, such that this objective function is minimized. Intuitively, the optimal L will map neighboring points from $\mathbb{R}^d$ to points that are close in $\mathbb{R}^p$. It can be shown (see [6] for more details) that the solution to this objective function is given by the minimum eigenvalue solutions to the generalized eigenvalue problem:

$$XGX^T\nu = \lambda XDX^T\nu \qquad (1)$$

where $X = [x_1, ..., x_N]$, $G = D - W$ is the graph Laplacian, $W = [w_{ij}]$ is the weight matrix, and $D = [d_{ij}]$ is a diagonal matrix with $d_{ii} = \sum_j w_{ij}$. Letting ν_i for $i = 0, ..., p - 1$ be the vectors corresponding to the p smallest non-zero eigenvalues, $L = [\nu_0; ...; \nu_{p-1}]$ defines the linear transformation matrix that minimizes the LPP objective function. Therefore, L defines a linear transformation such that *k*-NN classification is optimized with respect to the Euclidean distance metric in the embedding space, $\mathbb{R}^p$, and $M = L^T L$ parametrizes a Mahalanobis distance metric that is optimized for *k*-NN classification in the untransformed feature space ($\mathbb{R}^d$).

For this work, the LPP algorithm was implemented in Matlab using code provided by Deng Cai.[1]

2.3. Locality Sensitive Hashing (LSH)

The main drawback of nearest-neighbor methods is the need to search over the training set for each test point, which can be very slow. There are a number of algorithms for approximating the search that can greatly speed up the search, such as kd-trees [9] and locality-sensitive hashing (LSH) [8]. In this work we use LSH, which is particularly well-suited to search in high-dimensional spaces.

An LSH family is a family of hash functions $\mathcal{H} : \mathbb{R}^d \mapsto U$, where U denotes some universe, such that for any two points, p,$q \in \mathbb{R}^d$, and a randomly chosen hash function $h \in \mathcal{H}$, the following two conditions are satisfied: 1. if $\|p - q\| \leq R$ then $Pr_{\mathcal{H}}[h(q) = h(p)] \geq P_1$ and 2. if $\|p - q\| \geq cR$ then $Pr_{\mathcal{H}}[h(q) = h(p)] \leq P_2$. An LSH family is useful if $P_1 > P_2$ and is called (R, cR, P_1, P_2)-sensitive.

Given an LSH family $\mathcal{H}$, and parameters L and k, the LSH construction algorithm is defined as follows: 1. Choose L functions g_j, $j = 1, ..., L$ by setting $g_j = (h_{1j}, h_{2j}, ..., h_{kj})$, where $h_{1j}, h_{2j}, ..., h_{kj}$ are chosen uniformly at random from $\mathcal{H}$. 2. Construct L hash tables, where, for each $j = 1, ..., L$, the j^{th} hash table contains the dataset points hashed using the function g_j.

For a query point q, the search then proceeds as follows: 1. For each $j = 1, 2, ..., L$ (i) Retrieve the points from the bucket $g_j(q)$ in the j^{th} hash table. (ii) For each of the retrieved points p, compute the distance from q to it, and report the point if it is an R-near neighbor (i.e., $\|p - q\| \leq R$). The parameters L and k are the number of hash tables and the length of the hash keys, respectively. While LSH guarantees are with respect to the R-near neighbor search problem, we use the same indexing scheme to find approximate nearest neighbors by selecting from the union of retrieved points across the L tables the set of k points closest to the query.

3. Experiments

Our experiments are based on phonetic classification of speech frames from the TIMIT database of continuous speech [4]. We consider two types of acoustic feature vectors. In the first set of experiments, we used standard 39-dimensional feature vectors consisting of 13 Mel-frequency cepstral coefficients (MFCCs) plus 13 deltas (first derivatives) and 13 delta-deltas (second derivatives). In the second set of experiments, we included 2 context frames on each side, yielding 195-dimensional feature vectors. Each frame was labeled as one of the 39 standard TIMIT phonetic classes. We trained all classifiers on the standard TIMIT training set and tested on the standard "core" test set. We held out a 50,000-frame subset of the training set as development data for tuning hyperparameters of the classifiers and distance learning algorithms.

For the single-frame 39-dimensional features, we evaluated the performance of *k*-NN classifiers using Euclidean and LMNN-based distance metrics. For the Euclidean distance metric, we compared the performance of classifiers using both exact search and LSH [2]. For the LMNN-based distance metric, we used LSH to perform the *k*-NN search, and did not reduce the dimensionality.

Using the five-frame features, we evaluated the performance of *k*-NN classifiers using Euclidean, LMNN-based, and LPP-based distance metrics. In all cases, we allowed for dimensionality reduction before performing *k*-NN classification of the features. For the Euclidean distance metric, we used LDA to reduce the dimensionality of the features to $d = 38$ (one fewer than the number of classes, the maximum possible with LDA). For LMNN, we tuned the dimensionality of the embedding space on the development set and found that full dimensionality yielded the best results. For LPP, we tuned the dimensionality of the embedding space on the development set and found that $d = 130$ yielded the best results. For all experiments on the five-frame features, we performed the *k*-NN search using LSH. For all experiments, we found the optimal value of *k* in the range of 20 to 70 by tuning on the development set. For LMNN and LPP, we used default values for the neighborhood parameter ($n = 3$ and $n = 5$, respectively). Motivated by previous experiments which showed that the distance metrics could be effectively learned on subsets of the training data with substantial reductions in training times, we used 20,000 and 100,000 examples, respectively, to learn the LMNN and LPP metrics. In these experiments, the LMNN and LPP metrics required about 0.5-1.5 hours to train.

We compared *k*-NN classifiers to Gaussian mixture models (GMMs) and multilayer perceptrons (MLPs) [3]. The GMM-based classifiers were implemented using standard Matlab routines. The dimensionality of the features was tuned on the development set and $d = 38$ was found to yield the best results. We assumed diagonal covariances for all Gaussians, and the number of Gaussian components was tuned on the development set for factors of two between 32 and 1024. Within this range, the best number of components (as measured by development set performance) was found to be $C = 512$. The MLPs were trained on the training set, with a softmax output layer and a sigmoid hidden layer with $h = 2000$ hidden nodes, to optimize a cross-entropy based criterion. The MLPs were implemented using the QuickNet toolkit [10].

Table 1 shows our results. k-nearest neighbor classi-

[1] http://www.cs.uiuc.edu/homes/dengcai2/Data/code/LPP.m

[2] We are grateful to Greg Shakhnarovich for providing his LSH code and assistance in tuning LSH.

[3] The MLPs were provided to us by Rohit Prabhavalkar. We gratefully acknowledge his assistance.

fiers in general outperformed GMMs, while MLPs achieved the best performance by a wide margin. Of the k-NN distance learning methods, LPP performed the best, although by only a small margin over the others. It is noteworthy, however, that LPP is unsupervised, while both LDA and LMNN are supervised.

We also compared retrieval times using exact search and LSH with the 195-dimensional 5-frame feature vectors. Depending on the LSH parameters (number of tables, length of hash keys), it improved search speed by factors of up to about 50. For the results in Table 1, we opted for slower LSH structures (closer to exact) to ensure that the performance of different techniques is as comparable as possible. The fastest LSH structure tested, with a speedup of about 50x relative to exact search, resulted in a 2% absolute increase in error rate.

Method	**Search**	**ER (%)**	**Params**
k-NN Euclidean	Exact	39.93	(*k*=39,d=39)
k-NN Euclidean	LSH	40.08	(*k*=36,d=39)
k-NN LMNN	LSH	39.90	(*k*=51,d=39)
k-NN Euclidean	Exact	38.01	(*k*=30,d=195)
k-NN Euclidean	LSH	38.03	(*k*=30,d=195)
k-NN Eucl.-LDA	LSH	37.53	(*k*=33,d=38)
k-NN LMNN	LSH	36.92	(*k*=30,d=195,n=3)
k-NN LPP	LSH	**36.72**	(*k*=38,d=130,n=5)
GMM	n/a	**41.95**	(C=512,r=0.0005)
MLP	n/a	**30.16**	(h=2000)

Table 1: Error rates (ER) in %. $d = 39$ indicates 1-frame features; the remainder use 5-frame features. The best results obtained with each type of classifier are in boldface.

4. Discussion

We (and others) have motivated *k*-NN classifiers as an alternative to GMMs for speech recognition because of their simplicity, lack of training, and discriminative nature. In this study, we have (a) reaffirmed the conclusions of prior work (e.g., [1]) with evidence that *k*-NN classifiers are at least a reasonable choice for speech recognition, as their performance is better than that of GMMs, if not as good as that of MLPs, and (b) found that two distance learning techniques, LMNN and LPP, improve over the Euclidean distance with or without dimensionality reduction using LDA. The metrics can be learned on subsets of the training data, leading to quick training. LSH can be used to substantially reduce the burden of the *k*-NN classifiers at test time.

One reason for the superiority of MLPs in our experiments may be their nonlinearity. One direction for future work, therefore, is nonlinear extensions of distance learning such as kernelized versions of the algorithms used here. Other future directions include multiple-metric extensions to LMNN [5] and LPP [11], and supervised learning of embeddings rather than distances [12]).

Finally, frame classification performance is not necessarily predictive of recognition performance, so future work must also include validation in complete speech recognition systems. One approach for incorporating k-NN classifiers into continuous speech recognition is to construct class-conditional distributions using kernel density estimation, as in [1]: Given a feature vector x, the conditional distribution for class c can be computed by first finding the *k*-nearest neighbors to x from class c in the training data, then defining the class-conditional distribution for c as a linear combination of k Gaussian densities with equal variances, each centered at one of the k neighbors. One of our main directions for future work is to combine this idea with learned distance metrics.

5. References

[1] T. Deselaers, G. Heigold, and H. Ney, “Speech recognition with state-based nearest neighbour classifiers,” in *ICSLP*, 2007.

[2] N. Singh-Miller and M. Collins, “Learning label embeddings for nearest-neighbor multi-class classification with an application to speech recognition,” in *NIPS*, 2009.

[3] L. Golipour and D. O’Shaughnessy, “Phoneme classification and lattice rescoring based on a k-NN approach,” in *Interspeech*, 2010.

[4] L. Lamel, R. Kassel, and S. Seneff, “Speech database development: Design and analysis of the acoustic-phonetic corpus,” in *Speech Input/Output Assessment and Speech Databases*, 1989.

[5] K. Q. Weinberger and L. K. Saul, “Distance metric learning for large margin nearest neighbor classification,” *The Journal of Machine Learning Research*, vol. 10, pp. 207–244, 2009.

[6] X. He and P. Niyogi, “Locality preserving projections,” in *NIPS*, 2003.

[7] T. Cover and P. Hart, “Nearest neighbor pattern classification,” *Information Theory, IEEE Transactions on*, vol. 13, no. 1, pp. 21–27, 2002.

[8] A. Andoni and P. Indyk, “Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions,” *Communications of the ACM*, vol. 51, no. 1, pp. 117–122, 2008.

[9] S. Arya, D. N. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, “An optimal algorithm for approximate nearest neighbor searching,” in *Symposium on Discrete Algorithms*, 1994.

[10] D. Johnson *et al.*, “ICSI Quicknet software package,” *http://www.icsi.berkely.edu/Speech/qn.html*, 2007.

[11] Y. Tang and R. Rose, “A study of locality preserving projections for feature extraction in speech recognition,” in *ICASSP*, 2008.

[12] G. Shakhnarovich, “Learning task-specific similarity,” Ph.D. dissertation, Massachusetts Institute of Technology, 2005.