ARCCOSINE KERNELS: ACOUSTIC MODELING WITH INFINITE NEURAL NETWORKS

Chih-Chieh Cheng ∗
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093, USA
chc028@cs.ucsd.edu

Brian Kingsbury
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598, USA
bedk@us.ibm.com

ABSTRACT

Neural networks are a useful alternative to Gaussian mixture models for acoustic modeling; however, training multilayer networks involves a difficult, nonconvex optimization that requires some “art” to make work well in practice. In this paper we investigate the use of arccosine kernels for speech recognition, using these kernels in a hybrid support vector machine/hidden Markov model recognition system. Arccosine kernels approximate the computation in a certain class of infinite neural networks using a single kernel function, but can be used in learners that require only a convex optimization for training. Phone recognition experiments on the TIMIT corpus show that arccosine kernels can outperform radial basis function kernels.

Index Terms— kernel methods, support vector machines, neural networks, hybrid systems, speech recognition

1. INTRODUCTION

While hidden Markov models (HMMs) with Gaussian mixture models (GMMs) are the dominant acoustic model in automatic speech recognition, there are advantages to models that do not rely on density estimation. Because they make minimal assumptions about their inputs, neural networks are a promising alternative, and architectures that use neural networks directly as acoustic models (hybrid systems) [1] or as feature-extraction modules (tandem systems) [2] have been proposed. Recent results coming from the machine learning community have shown the promise of a greedy, layer-wise initialization procedure based on restricted Boltzmann machines for building deep models [3].

Their advantages notwithstanding, neural networks do have one significant problem: their training involves a challenging, nonconvex optimization. This problem motivates the exploration of alternative learners that are easier to train, but share some of the same properties. One option is support vector machines (SVMs). Like neural networks, SVMs make minimal assumptions about their inputs, can work in a highly nonlinear feature space, and are discriminative. Unlike neural networks, they require only a convex optimization in training. Recently, a new kernel family, arccosine kernels [4], was introduced. Arccosine kernels connect SVMs and neural networks by mimicking the computation in multilayer neural networks with a kernel function.

In this paper we investigate the use of arccosine kernels in speech recognition, integrating SVMs and HMMs in a hybrid acoustic model [5]. To connect the binary outputs from SVMs to HMMs, we train a multinomial regression module to obtain posterior probabilities, which are converted to emission probabilities by Bayes’ rule. By incorporating a trainable multinomial regression module, we add one more discriminative layer on top of the SVM outputs, which is different from previous HMM/SVM hybrid systems. We perform phone recognition experiments on the TIMIT corpus to evaluate different kernels, feature sets, and methods for improving the multinomial regression.

2. KERNEL METHODS

2.1. Arccosine Kernels

We briefly describe the recently introduced arccosine [4] kernel family: kernels that merge shallow and deep learning architectures by mimicking the computation in a certain class of infinite neural networks.

Let \( \Theta(z) = 0.5(1 + \text{sign}(z)) \) denote the Heaviside step function. Given input vectors \( x \) and \( y \), the arccosine kernel function is defined as

\[
k_n(x, y) = 2 \int dw \frac{e^{\frac{i|w|^2}{2}}}{(2\pi)^{d/2}} \Theta(w \cdot x) \Theta(w \cdot y) (w \cdot x)^n (w \cdot y)^n,
\]

and is derived to approximate the inner product between the outputs of two single-layer neural networks having an infinite number of elements.

To see the connection, consider a single-layer threshold network with input \( x \) and weights \( W \), with \( W_{ij} \) denoting the weight connecting the \( i \)th output to the \( j \)th input. The output of the network, \( f(x) \), is defined as

\[
f(x) = g(Wx),
\]

where \( g(z) \) is the nonlinear activation function. If we consider piecewise-smooth activation functions

\[
g_n(z) = \Theta(z) z^n,
\]

then the inner product between the \( m \)-dimensional outputs of this network is

\[
f(x) \cdot f(y) = \sum_{i=1}^{m} \Theta(w_i \cdot x) \Theta(w_i \cdot y) (w_i \cdot x)^n (w_i \cdot y)^n,
\]

where \( n \) is the coefficient controlling the degree of nonlinearity in the activation function. In the limit, as \( m \) approaches infinity, and assuming that the weights \( W_{ij} \) are Gaussian distributed with zero mean and unit variance, Eq. (4) reduces to Eq. (1), up to a multiplicative factor.

∗ The author performed the work while interning at the IBM T. J. Watson Research Center.
The computation of the kernel function in Eq. (1) comes in a very neat form, and the full derivation can be found in [4]. Denote θ as the angle between the input vectors, which can be computed by

$$θ = \cos^{-1}\left(\frac{x \cdot y}{\|x\|\|y\|}\right).$$

Eq. (1) can be rewritten in terms of θ as

$$k_n(x, y) = \frac{1}{\pi}\|x\|^n\|y\|^n J_n(\theta),$$

where n is coefficient setting the activation function nonlinearity (Eq. (3)), and

$$J_n(\theta) = (-1)^n(\sin \theta)^{2n+1}\left(\frac{\partial}{\partial \theta} \frac{\partial}{\partial \theta} \left(\frac{\pi - \theta}{\sin \theta}\right)^n\right)$$

for $\forall n \in \{0, 1, 2, \ldots\}$.

Arccosine kernels have several interesting properties. Note that there is a degree coefficient, n, that tunes the shape of the kernel.

For n = 0, Eq. (6) simplifies to

$$k_0(x, y) = 1 - \frac{1}{\pi}\cos^{-1}\left(\frac{x \cdot y}{\|x\|\|y\|}\right),$$

which is simply a linear function of the angle between the input vectors. The n = 0 kernel is very similar to RBF kernels, but is scale-invariant. The n = 1 kernel is the only one preserving the original magnitude of the input vector, while n > 1 kernels have an expanding property similar to polynomial kernels.

### 2.2. Extension to Multilayer Networks

One of the most compelling properties of arccosine kernels is that we can compose them to mimic the computation in multilayer, infinite neural networks. Recall that a kernel function induces a nonlinear mapping from input vectors x to feature vectors $\Phi(x)$, and the kernel function defines the inner product in the induced feature space:

$$k(x, y) = \Phi(x) \cdot \Phi(y).$$

If we apply another nonlinear mapping on the induced feature space $\Phi(x)$, a new feature space $\Phi(\Phi(x))$ can be induced. Based on this idea, if $l$ successive nonlinear mappings $\Phi(\cdot)$ are applied, the corresponding kernel function should be

$$k^{(l)}(x, y) = \Phi(\Phi(\ldots \Phi(x)))) \cdot \Phi(\Phi(\ldots \Phi(x)))).$$

By substituting into Eq. (6), an $l$-fold recursive computation for multilayer arccosine kernels can be derived:

$$k^{(l)}_n(x, y) = \frac{1}{\pi}\left[k^n_n(x, x)k^n_n(y, y)\right]^{n/2} J_n(\theta^n_n),$$

where $\theta^n_n = \cos^{-1}\left(\frac{k^n_n(x, y)}{\sqrt{k^n_n(x, x)k^n_n(y, y)}}\right)$.

To simplify the notation, we assume that each kernel has the same degree n, but the extension to the general case is straightforward. Note that arccosine kernels are unique in that function composition induces interesting, new feature spaces. Composition of polynomial kernels produces higher-degree polynomials; composition of RBF kernels, while non-trivial, does not produce a substantially different computation because the RBF kernel is a soft vector quantizer [4].

### 2.3. Multiway Classification

The most easily used kernel classifier is the support vector machine (SVM); however, the original formulation of the SVM is as a binary classifier, and acoustic modeling requires many classes. While there are alternatives such as multiclass SVMs and kernel logistic regression that are inherently multiway classifiers, we chose to use a hierarchical structure in which SVMs perform pairwise classification (a one-vs.-one design), then the pairwise classifications are aggregated into a single, multiway classification. We opted for this approach because of the availability of open-source software, libSVM [6], for training binary SVMs, and because this design sped up acoustic model training. SVM training is quadratic in the number of training samples, so reducing the classification problem to one between phone pairs leads to much faster training. While this design requires the training of a large number of SVMs, the training is readily parallelized across a large compute cluster.

### 2.4. Hybrid HMM/SVM Model

Because we use a one-vs.-one approach for SVM training, for a set of K phones, we must train $K(K - 1)/2$ SVMs. Each SVM computes $f_{ij}(x)$, the distance from the input, x, to the decision boundary between phones i and j. This distance is converted to a posterior probability estimate via logistic regression [7, 8].

We then transform the pairwise posteriors, $p_{ij}(q = i|x)$, to multiway phone posteriors, $p(q = i|x)$, using a trainable softmax function (multinomial regression),

$$p(q = i|x) = \frac{\exp(\sum_j \alpha_{ij}p_{ij} + b_i)}{\sum_i \exp(\sum_j \alpha_{ij}p_{ij} + b_i)} \quad \forall i,$$

where $\alpha_{ij}$ is a trainable weight for $p_{ij}$, and $b_i$ is a trainable bias for phone i. The weights are trained using stochastic gradient descent.

Finally, we transform the posteriors into HMM state emission probabilities [1] by $p(x|q = i) \propto p(q = i|x)/p(q = i)$. The state priors, $p(q = i)$, are estimated from the training data. A standard Viterbi decoding is then used for phone recognition.

### 3. EXPERIMENTS

We perform phone recognition experiments on TIMIT [9] to evaluate the performance of arccosine and RBF kernels in an acoustic modeling task, to evaluate the performance of three different feature sets, and to compare the performance of hybrid HMM/SVM models to other models. We use the standard division of TIMIT into training, development, and test sets comprising 3696, 400, and 944 utterances, respectively. In this work we report results in terms of phone error rate measured on the 196-utterance core test set. All system tuning is done on the development set.

For training, TIMIT’s 61 phonetic labels are collapsed to a set of 49 classes [10], including [q], which is ignored in the scoring procedure. For each experiment, we train 1176 SVM classifiers, each of which makes a pairwise classification between two phonetic targets. SVM training is done using the open-source libSVM library [6], and takes place in two steps. The first step tunes the hyperparameters of the SVM, performing a two-stage grid search in which SVMs are trained on a subsampled version of the training data and the hyperparameters are selected based on frame classification performance on the development data. The subsampling was random, and selected 2426 samples for each of the 49 phone classes. In the second step, SVMs are trained on the full training set, with a weighting of the
### Table 1. TIMIT core test phone error rates for radial basis function and single-layer arccosine kernels with different degrees (Eq. (6)), using LDA features.

<table>
<thead>
<tr>
<th>Kernel</th>
<th>RBF</th>
<th>Single-Layer ARCCOSINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree</td>
<td>—</td>
<td>0</td>
</tr>
<tr>
<td>PER (%)</td>
<td>29.3</td>
<td>29.9</td>
</tr>
</tbody>
</table>

Table 2. TIMIT core test phone error rates for arccosine kernels having 1–3 layers and varying kernel degrees (Eq. (11)), using LDA features.

<table>
<thead>
<tr>
<th>Kernels</th>
<th>ARCCOSINE</th>
</tr>
</thead>
<tbody>
<tr>
<td># Layers</td>
<td>1</td>
</tr>
<tr>
<td>PER (%)</td>
<td>29.4</td>
</tr>
</tbody>
</table>

3.1. Arccosine kernels

We first examine the effect of the kernel degree in single-layer arccosine kernel acoustic models (Eq. (6)), and compare their performance to a baseline acoustic model that uses radial basis function (RBF) kernels. RBF kernels have shown good acoustic modeling performance in prior studies [12, 5, 13]. For all kernels, the margin regularization factor is treated as a hyperparameter to be optimized; for the RBF kernels, we also optimize the kernel width for each pairwise classification problem. We trained models on the LDA features, using the RBF kernel and degree 0, 1, and 2 arccosine kernels \((n = 0, 1, 2)\) in Eq. (6). Table 1 shows core test set phone error rates for the four models. The degree 1 and 2 arccosine kernels outperform the RBF kernel, and the best performance is obtained with the degree 2 arccosine kernel: a kernel with expanding properties on a 9-frame context window.

### Table 3. TIMIT core test phone error rates for different feature sets using RBF or optimized arccosine kernels.

<table>
<thead>
<tr>
<th>System</th>
<th>RBF</th>
<th>ARCCOSINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>29.3</td>
<td>28.7</td>
</tr>
<tr>
<td>+ 9 frames context</td>
<td>27.7</td>
<td>27.5</td>
</tr>
<tr>
<td>+ realignment</td>
<td>27.1</td>
<td>27.2</td>
</tr>
<tr>
<td>+ BMMI training</td>
<td>27.2</td>
<td>26.7</td>
</tr>
</tbody>
</table>

3.2. Comparison between different features

So far we have shown results with LDA features. We also tested the MFCC+Δ and spliced feature sets. Table 3 shows error rates for the different feature sets. For each set, we compare the best result with arccosine kernels to the result with the RBF kernel. Among features, the LDA features are best for both RBF and arccosine kernels; among kernels, arccosine kernels outperform RBF kernels with LDA features, and are somewhat worse than RBF kernels for the MFCC+Δ features. Interestingly, the best arccosine models for the MFCC+Δ and spliced features were 3-layer kernels. We did not have enough time to explore still deeper models on these feature sets.

3.3. Improved multiway classification

In addition to optimizing the initial pairwise SVM classifiers as in the previous experiments, we also investigate improvements to the softmax (Eq. (13)) that produces multiway phone posteriors. We study three enhancements: 1) using a context window in the softmax to exploit information in neighboring frames, 2) realigning the training and development data with a model and then retraining the model using the automatic alignment, and 3) using a sequence-discriminative criterion [14] to train the softmax. All of these improvements are tested on the single-layer, degree 2 arccosine kernel model. In the context window experiments, we test different context windows \((\pm 1 \text{ frame}, \pm 2 \text{ frames}, \pm 3 \text{ frames}, \text{ etc.})\) and choose the size that gives the best phone error rate on the development set. For both kernels, the 9-frame context window \((\pm 4 \text{ frames})\) gave the best performance. In the discriminative training experiments, we found that the minimum phone error (MPE) [15] and boosted maximum mutual information (BMMI) [16] criteria had similar performance.

The results of these experiments are summarized in Table 4. We see that the use of temporal context significantly improves phone...
error rates, while the realignment produces a smaller improvement for both kernels. The discriminative training results are not as consistent: we observe a small degradation on the core test set for the RBF kernel (although development set PER improved from 25.9% to 25.4%) and an improvement for the arccosine kernel. These results illustrate two of the difficulties of the TIMIT task: the core test set is small enough that only large changes in performance can be considered statistically significant, and the training corpus is small enough that model overfitting is hard to avoid.

3.4. Overall comparison

To further analyze the performance of the arccosine kernel acoustic model, we compare our results to others in the literature in Table 5. To make a meaningful comparison, we look only at results on the core test set for context-independent systems that use the same phone set as we do, and that are trained only on the TIMIT training set. We also list the features used by each system. We see that the arccosine kernel acoustic models do not beat the best systems on TIMIT, but that they do compare well to recent results on the corpus.

4. CONCLUSION

In this paper we investigate the use of arccosine kernels for acoustic modeling in a hybrid HMM/SVM system that integrates pairwise classifications from the SVMs using a trainable multinomial regression. We experimented extensively with different kernel settings and feature sets to optimize the SVM classifiers. We also explored improvements to the multinomial regression: using temporal context, discriminative training, and automatic alignments of the training data. We showed that the single layer arccosine kernel with LDA features yields the best results, improving over RBF kernels. Future directions for research include testing additional layers in the kernels, and training algorithms that jointly optimize the pairwise SVMs and multinomial regression.

5. ACKNOWLEDGMENT

We thank Youngmin Cho and Lawrence Saul for their guidance in this project, and we thank Tara Sainath for her advice on working with TIMIT and for supplying us with the front-end processing parameters used in this work.

6. REFERENCES


