Kernels vs. DNNs for Speech Recognition

Joint work with:
Columbia: Linxi (Jim) Fan, Michael Collins (my advisor)
USC: Zhiyun Lu, Kuan Liu, Alireza Bagheri Garakani, Dong Guo, Aurélien Bellet, Fei Sha
IBM: Brian Kingsbury
Outline

• Background
  • Kernel methods
  • Kernel approximation
    • Random Fourier Features
    • Acoustic modeling overview

• Our work
  • Kernel composition
  • Parallel training
  • Experimental results

• Future work
  • Nystrom method
  • Image data
Outline

• Background
  • Review: Kernel methods
    • Kernel approximation
      • Random Fourier Features
    • Acoustic modeling overview

• Our work
  • Kernel composition
  • Parallel training
  • Experimental results

• Future work
  • Nystrom method
  • Image data
Linear SVM Review

Trying to find separating hyperplane with largest margin
Primal vs. Dual

• Primal problem:
\[
\min_{w, b, \xi} \frac{1}{2} \|w\|^2 + C \sum_i \xi_i \quad \text{subject to} \quad y_i(w^T x_i + b) - 1 + \xi_i \geq 0
\]

\[\implies \text{Classifier: } f(x \mid w, b) = \text{sign}(w^T x + b)\]

• Dual Problem
\[
\max_\alpha \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j \quad \text{subject to} \quad \sum_i \alpha_i y_i = 0 \text{ and } \alpha_i \in [0, C]
\]

\[\implies \text{Classifier: } f(x \mid \alpha) = \text{sign}(\sum_i \alpha_i y_i x_i^T x + b)\]
Background: Kernel Methods
Kernelized Primal vs. Dual

• Kernelized Primal problem:

\[
\min_{w,b,\xi} \frac{1}{2} \|w\|^2 + C \sum_i \xi_i \quad \text{subject to} \quad y_i (w^T \phi(x_i) + b) - 1 + \xi_i \geq 0
\]

• Kernelized Dual Problem

\[
\max_\alpha \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j) \quad \text{subject to} \quad \sum_i \alpha_i y_i = 0 \text{ and } \alpha_i \in [0, C]
\]

\[
\max_\alpha \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \quad \text{subject to} \quad \sum_i \alpha_i y_i = 0 \text{ and } \alpha_i \in [0, C]
\]

\[\Rightarrow \text{Classifier: } f(x \mid \alpha) = \text{sign}(\sum_i \alpha_i y_i k(x_i, x) + b)\]
Kernel Trick

Kernel Examples
Outline

• Background
  • Review: Kernel methods
  • Kernel approximation
    • Random Fourier Features
    • Acoustic modeling overview

• Our work
  • Kernel composition
  • Parallel training
  • Experimental results

• Future work
  • Nystrom method
  • Image data
Kernel Approximation

No Kernel: \[ G = X^T X \]

\[ \implies G_{i,j} = x_i^T x_j \]

Exact Kernel: \[ G = \Phi^T \Phi \]

\[ \implies G_{i,j} = \phi(x_i)^T \phi(x_j) \]

Approximate Kernel: \[ G \approx \tilde{G} = Z^T Z \]

\[ \implies \tilde{G}_{i,j} = z(x_i)^T z(x_j) \]

Can use \( z(x) \) in primal, instead of \( \phi(x) \)!
How to construct approximation?

**Theorem (Bochner):** A continuous kernel $k(x, y) = k(x - y)$ on $\mathbb{R}^d$ is positive definite if and only if $k(\delta)$ is the Fourier transform of a non-negative measure.

$$k(x, y) = \mathbb{E}[z(x)^T z(y)],$$

where:

- $z(x)_i = \sqrt{\frac{2}{D}} \cos(w_i^T x + b)$
- $w_i$ drawn from $p(w)$, the probability distribution computed as the Fourier transform of $k(\delta)$
- $b$ is drawn uniformly from $[0, 2\pi]$

<table>
<thead>
<tr>
<th>Kernel Name</th>
<th>$k(\Delta)$</th>
<th>$p(\omega)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian</td>
<td>$e^{-\frac{|\Delta|_2^2}{2}}$</td>
<td>$(2\pi)^{-\frac{D}{2}} e^{-\frac{|\omega|_2^2}{2}}$</td>
</tr>
<tr>
<td>Laplacian</td>
<td>$e^{-|\Delta|_1}$</td>
<td>$\prod_d \frac{1}{\pi(1+\omega_d^2)} e^{-|\Delta|_1}$</td>
</tr>
<tr>
<td>Cauchy</td>
<td>$\prod_d \frac{2}{1+\Delta_d^2}$</td>
<td></td>
</tr>
</tbody>
</table>

Outline

• Background
  • Review: Kernel methods
  • Kernel approximation
    • Random Fourier Features
  • Acoustic modeling overview

• Our work
  • Kernel composition
  • Parallel training
  • Experimental results

• Future work
  • Nystrom method
  • Image data
Given this $x$, the acoustic modeling problem is to produce a probability distribution over phonemes (or triphones, or senones...) for this frame.

Objective function: Maximize log-probability of training data

$$
\max_{W} \sum_{i} \log(P(y_i | x_i, W)) - \frac{\lambda}{2} \|W\|^2
$$
Training Acoustic Models

• Using DNNs with back-propagation

• Often, some unsupervised or discriminative pre-training
Why use DNNs??

• They are powerful models, that can be trained effectively
• They beat the previous state of the art by a large margin!!!

<table>
<thead>
<tr>
<th>TASK</th>
<th>HOURS OF TRAINING DATA</th>
<th>DNN-HMM</th>
<th>GMM-HMM WITH SAME DATA</th>
<th>GMM-HMM WITH MORE DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWITCHBOARD (TEST SET 1)</td>
<td>309</td>
<td>18.5</td>
<td>27.4</td>
<td>18.6 (2,000 H)</td>
</tr>
<tr>
<td>SWITCHBOARD (TEST SET 2)</td>
<td>309</td>
<td>16.1</td>
<td>23.6</td>
<td>17.1 (2,000 H)</td>
</tr>
<tr>
<td>ENGLISH BROADCAST NEWS</td>
<td>50</td>
<td>17.5</td>
<td>18.8</td>
<td></td>
</tr>
<tr>
<td>BING VOICE SEARCH (SEN)</td>
<td>24</td>
<td>30.4</td>
<td>36.2</td>
<td></td>
</tr>
<tr>
<td>GOOGLE VOICE INPUT</td>
<td>5,870</td>
<td>12.3</td>
<td></td>
<td>16.0 (&gt;&gt; 5,870 H)</td>
</tr>
<tr>
<td>YOUTUBE</td>
<td>1,400</td>
<td>47.6</td>
<td>52.3</td>
<td></td>
</tr>
</tbody>
</table>

Issues with DNNs

• Costly to train (days on GPUs...)
• Sensitive to initialization
• Non-convex optimization problem
• Need to use lots of tricks, like momentum, drop-out, fancy initialization and pre-training, etc.
• Lots of hyper-parameters to tune (# of layers, # hidden units per layer, learning rate, regularization, etc.)
• Not well understood theoretically.
• Model not interpretable: “Magic black box”
Our Work

Random W

Cosine

Trainable Parameters
Kernel Combinations

Additive Kernels: \( k(x, y) = k_1(x, y) + k_2(x, y) \)

\( \implies \) simply concatenate feature representations for each kernel

Multiplicative Kernels: \( k(x, y) = k_1(x, y) \ast k_2(x, y) \)

\( \implies \) Draw \( w_i \) from \( p_i \), and then take \( w = \sum_i w_i \)

Composite Kernels: \( k(x, y) = k_2(\phi_1(x), \phi_1(y)) = \phi_2(\phi_1(x))^T \phi_2(\phi_1(y))^T \)

\( \implies \) Equivalent to having 2 hidden layers, each with random weights

\( \implies \) For efficiency, we perform supervised dimensionality reduction on output of first hidden layer
Parallel training

• When have hidden layer with $\geq 100,000$ units, we split the training into batches, each with 25,000 hidden units.

• We then combine the models trained from all these models by taking the geometric means of their outputs.
Data sets used

- IARPA Babel Program Cantonese/Bengali Language Packs
  - 20-hour train/test sets
  - Approximately 7.5 millions training, 1 million held-out, 7 million test
  - 1000 phone-states to predict (quinphone context-dependent HMM states clustered using decision trees)
  - 360 dimensional frame representations
Baselines

• IBM’s DNN
  • 5 hidden layers, 1024 logistic units each
  • Trained using greedy layer-wise discriminative pretraining.
  • Fine-tuning using SGD with mini-batches of size 250

• RBM-DNN
  • 1, 2, 3, or 4 hidden layers, each with 500, 1000, or 2000 logistic units
  • Unsupervised pre-training using Contrastive Divergence algorithm
  • Fine tuning using SGD
Results

- Best perplexity and accuracy by different models (heldout/test)

<table>
<thead>
<tr>
<th>Model</th>
<th>Bengali</th>
<th>Cantonese</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>perp</td>
<td>acc (%)</td>
</tr>
<tr>
<td>ibm</td>
<td>3.4</td>
<td>71.5/71.2</td>
</tr>
<tr>
<td>rbm</td>
<td>3.3</td>
<td>72.1/71.6</td>
</tr>
<tr>
<td>1-k</td>
<td>3.7</td>
<td>70.1/69.7</td>
</tr>
<tr>
<td>a-2-k</td>
<td>3.6</td>
<td>70.3/70.0</td>
</tr>
<tr>
<td>m-2-k</td>
<td>3.7</td>
<td>70.3/69.9</td>
</tr>
<tr>
<td>c-2-k</td>
<td>3.5</td>
<td>71.0/70.4</td>
</tr>
</tbody>
</table>

- Best token error rates

<table>
<thead>
<tr>
<th>Model</th>
<th>Bengali</th>
<th>Cantonese</th>
</tr>
</thead>
<tbody>
<tr>
<td>ibm</td>
<td>70.4</td>
<td>67.3</td>
</tr>
<tr>
<td>rbm</td>
<td>69.5</td>
<td>66.3</td>
</tr>
<tr>
<td>1-k</td>
<td>70.0</td>
<td>65.7</td>
</tr>
<tr>
<td>a-2-k</td>
<td>73</td>
<td>68.8</td>
</tr>
<tr>
<td>m-2-k</td>
<td>72.8</td>
<td>69.1</td>
</tr>
<tr>
<td>c-2-k</td>
<td>71.2</td>
<td>68.1</td>
</tr>
</tbody>
</table>
How many random features do we need?

- Single laplacian kernel

<table>
<thead>
<tr>
<th></th>
<th>Bengali</th>
<th>Cantonese</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dim</td>
<td>perp</td>
<td>acc (%)</td>
</tr>
<tr>
<td>2k</td>
<td>4.4/4.4</td>
<td>66.5/66.8</td>
</tr>
<tr>
<td>5k</td>
<td>4.1/4.2</td>
<td>67.8/67.8</td>
</tr>
<tr>
<td>10k</td>
<td>4.0/4.1</td>
<td>68.4/68.3</td>
</tr>
<tr>
<td>25k</td>
<td>3.8/3.9</td>
<td>69.2/69.0</td>
</tr>
<tr>
<td>50k</td>
<td>3.8/3.9</td>
<td>69.7/69.4</td>
</tr>
<tr>
<td>100k</td>
<td>3.7/3.8</td>
<td>70.0/69.6</td>
</tr>
<tr>
<td>200k</td>
<td>3.7/3.8</td>
<td>70.1/69.7</td>
</tr>
</tbody>
</table>

- Kernel Approximation error
Complementary representations?

- Token error rates for combined models:

<table>
<thead>
<tr>
<th>Model</th>
<th>Bengali</th>
<th>Cantonese</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEST SINGLE SYSTEM</td>
<td>69.5</td>
<td>65.7</td>
</tr>
<tr>
<td>rbm ($h = 3, L = 2000$) + 1-k</td>
<td>69.7</td>
<td>65.3</td>
</tr>
<tr>
<td>rbm ($h = 4, L = 1000$) + 1-k</td>
<td>69.2</td>
<td>64.9</td>
</tr>
<tr>
<td>rbm ($h = 4, L = 2000$) + 1-k</td>
<td>69.1</td>
<td>64.9</td>
</tr>
</tbody>
</table>
THANK YOU!!!