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

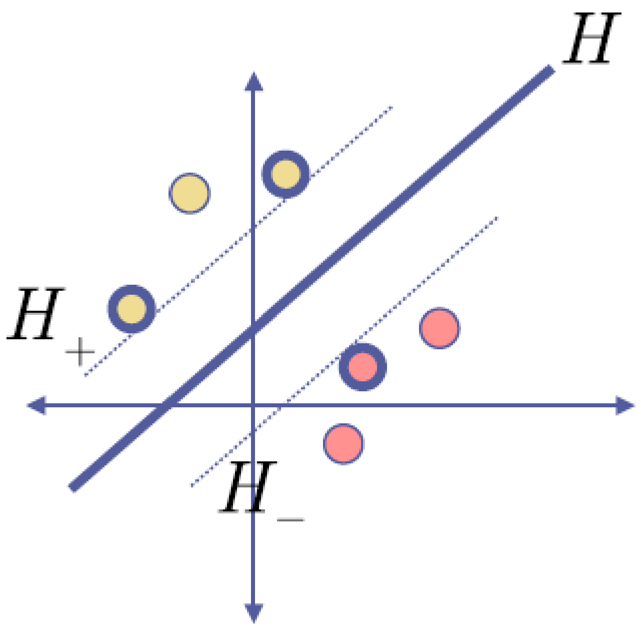
- Background
  - Kernel methods
  - Kernel approximation
    - Random Fourier Features
  - Acoustic modeling overview
- Our work
  - Kernel composition
  - Parallel training
  - Experimental results
- Future work
  - Nystrom method
  - Image data

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:

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

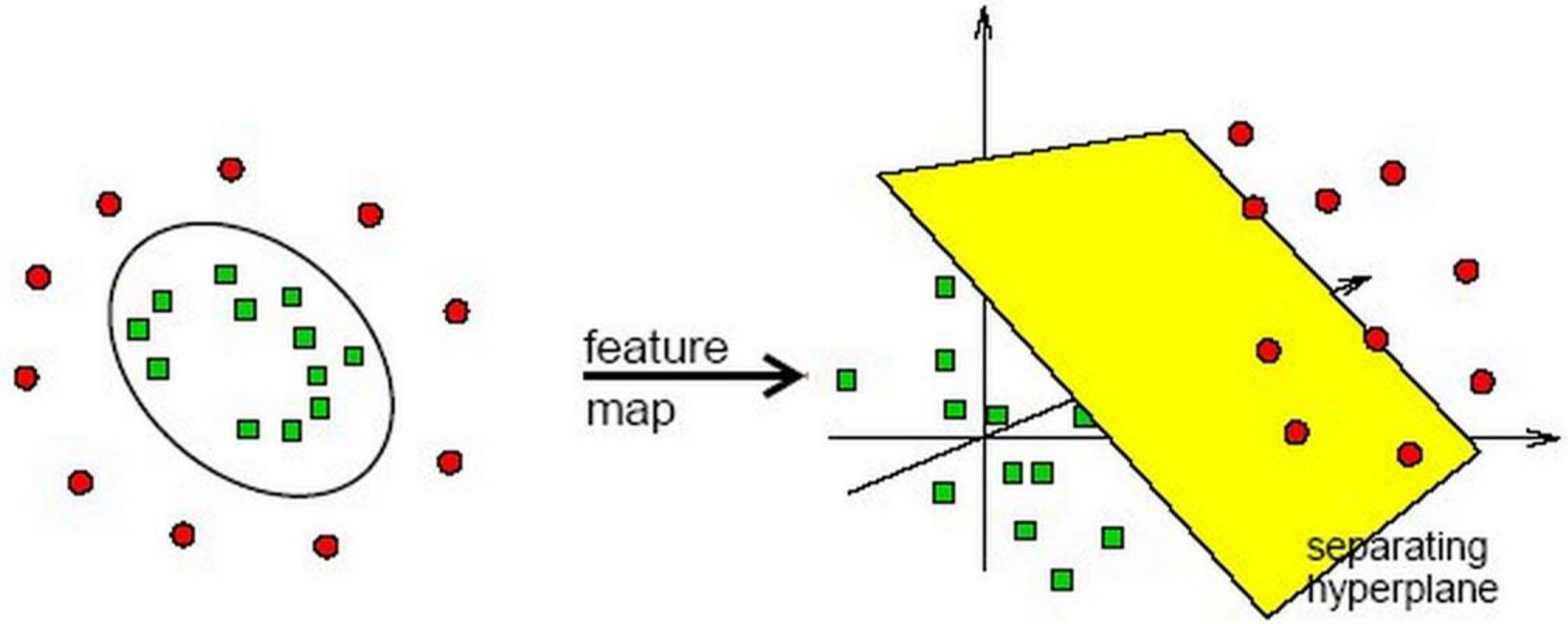
$$\implies \text{Classifier: } f(x \mid w, b) = sign(w^T x + b)$$

- Dual Problem

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

$$\implies \text{Classifier: } f(x \mid \alpha) = sign(\sum_i \alpha_i y_i x_i^T x + b)$$

Background: Kernel Methods



Kernelized Primal vs. Dual

- Kernelized Primal problem:

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

- Kernelized Dual Problem

Only use dot-products ☺

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

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

Kernel Trick!

N^2 sum… ☹

$$\Longrightarrow \text{Classifier: } f(x \mid \alpha) = sign(\sum_i \alpha_i y_i k(x_i,x) + b)$$

Kernel Trick

$$x \in \mathbb{R}^2, \quad \phi(x) = [x_1^2 \;\; \sqrt{2}x_1x_2 \;\; x_2^2]^T$$

$$k(x,y) = \phi(x)^T\phi(y)$$

$$k(x,y) = x_1^2y_1^2 + 2x_1y_1x_2y_2 + x_2^2y_2^2$$

$$k(x,y) = (x^Ty)^2$$

Kernel Examples

$$\begin{aligned}
\text{Polynomial kernel (degree p): } k(x,y) &= (x^Ty+1)^p \\
\text{Radial Basis Function (RBF) Kernel: } k(x,y) &= \exp(-\frac{1}{2\sigma^2}\|x-y\|_2^2) \\
\text{Laplacian Kernel: } k(x,y) &= \exp(-\frac{1}{\sigma}\|x-y\|_1^2) \\
\text{Sigmoid kernel: } k(x,y) &= \tanh(\kappa x^Ty-\delta)
\end{aligned}$$

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

$$
\begin{aligned}
\text{No Kernel:} \quad G &= X^T X \\
\Longrightarrow G_{i,j} &= x_i^T x_j \\
\text{Exact Kernel:} \quad G &= \Phi^T \Phi \\
\Longrightarrow G_{i,j} &= \phi(x_i)^T \phi(x_j) \\
\text{Approximate Kernel:} \quad G &\approx \tilde{G} = Z^T Z \\
\Longrightarrow \tilde{G}_{i,j} &= z(x_i)^T z(x_j)
\end{aligned}
$$

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 $[0, 2\pi]$

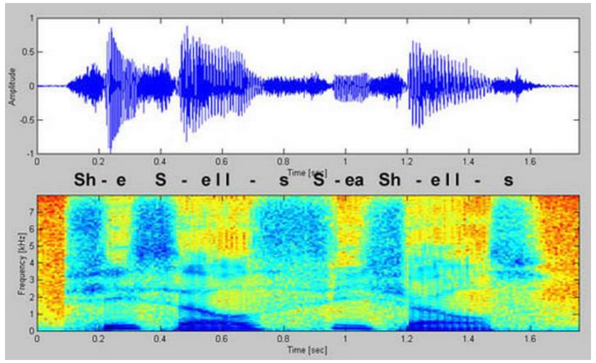
Kernel Name	$k(\Delta)$	$p(\omega)$
Gaussian	$e^{-\frac{\|\Delta\|_2^2}{2}}$	$(2\pi)^{-\frac{D}{2}} e^{-\frac{\|\omega\|_2^2}{2}}$
Laplacian	$e^{-\|\Delta\|_1}$	$\prod_d \frac{1}{\pi(1+\omega_d^2)}$
Cauchy	$\prod_d \frac{2}{1+\Delta_d^2}$	$e^{-\|\Delta\|_1}$

[1] Random Features for Large-Scale Kernel Machines. Ali Rahimi and Benjamin Recht. NIPS 2007, Vancouver.

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

Acoustic modeling



Extract Frame

Feature Extraction

$x \in \mathbb{R}^d$

Concatenate Neighbor Frames

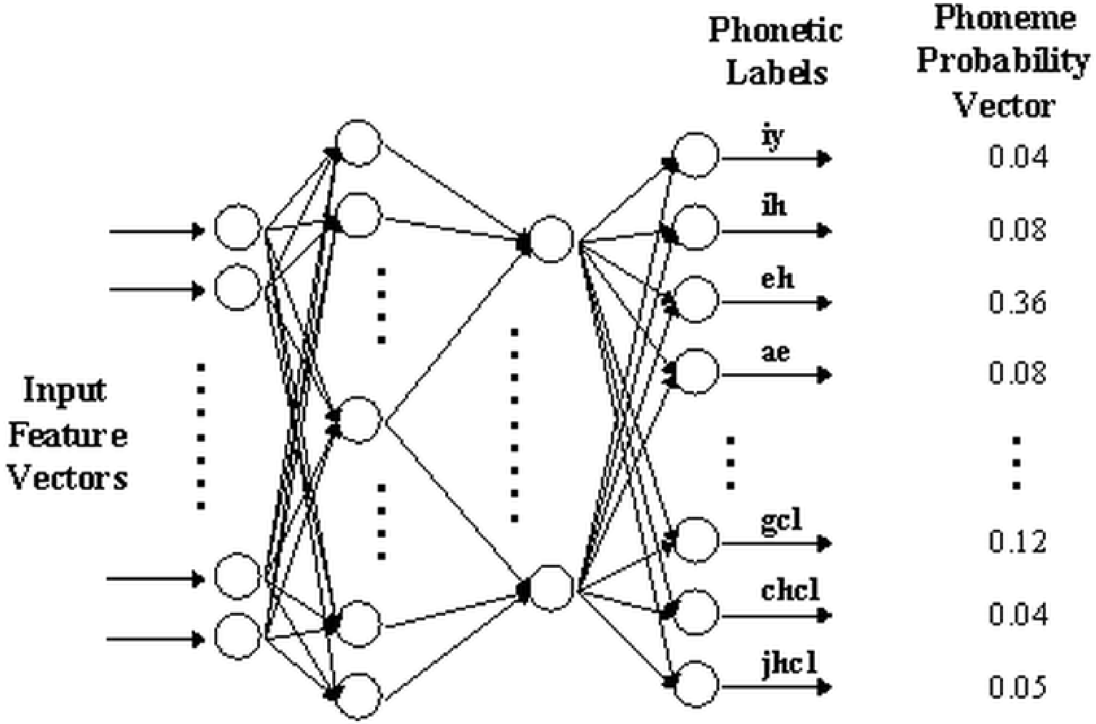
$x \in \mathbb{R}^{d(1+2n)}$

- 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

$$\max_W \sum_i \log(\mathbb{P}(y_i \mid x_i, W)) - \frac{\lambda}{2}\|W\|^2$$

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 3] A COMPARISON OF THE PERCENTAGE WERs USING DNN-HMMs AND GMM-HMMs ON FIVE DIFFERENT LARGE VOCABULARY TASKS.

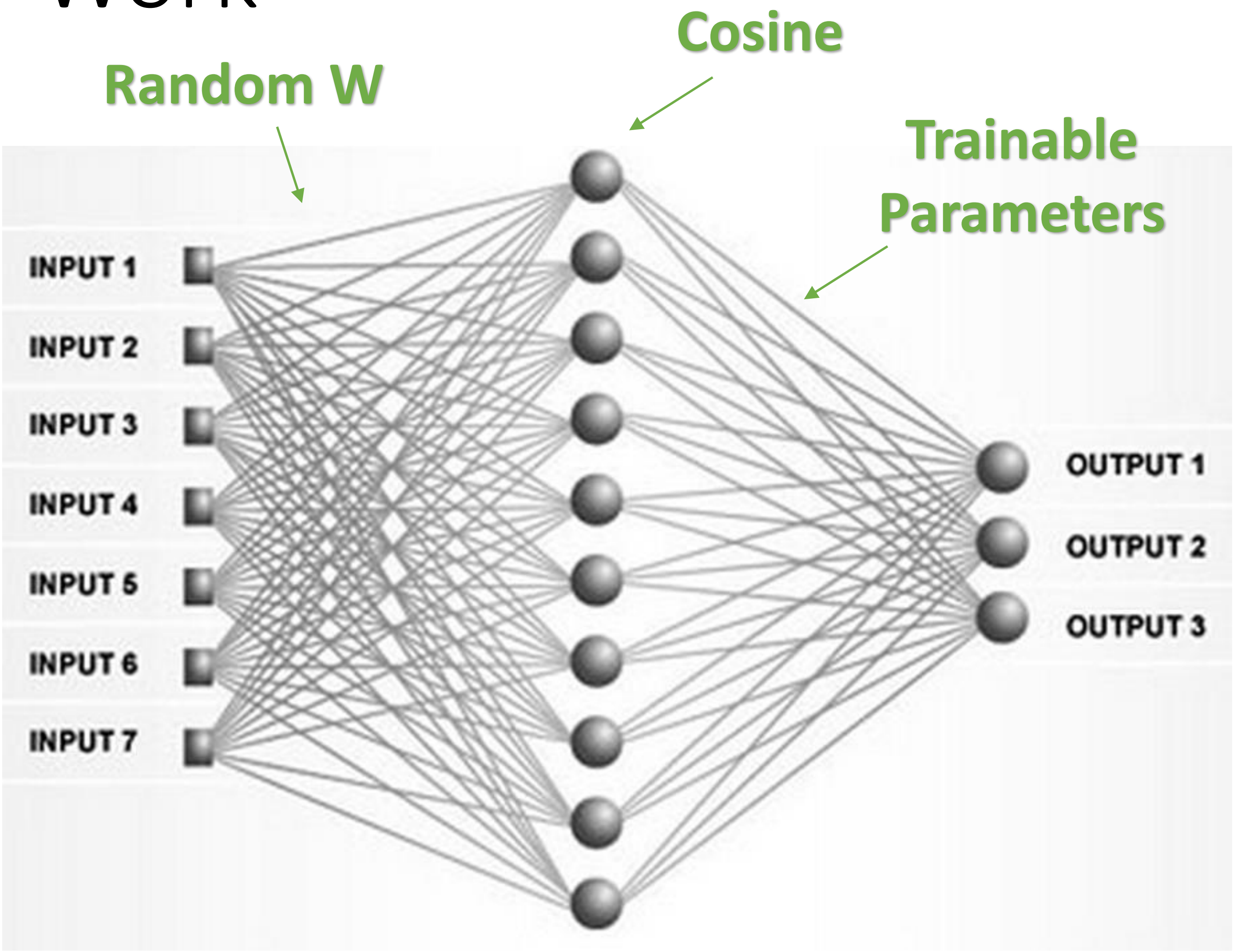
TASK	HOURS OF TRAINING DATA	DNN-HMM	GMM-HMM WITH SAME DATA	GMM-HMM WITH MORE DATA
SWITCHBOARD (TEST SET 1)	309	18.5	27.4	18.6 (2,000 H)
SWITCHBOARD (TEST SET 2)	309	16.1	23.6	17.1 (2,000 H)
ENGLISH BROADCAST NEWS	50	17.5	18.8	
BING VOICE SEARCH (SENTENCE ERROR RATES)	24	30.4	36.2	
GOOGLE VOICE INPUT	5,870	12.3		16.0 (>> 5,870 H)
YOUTUBE	1,400	47.6	52.3	

Geoffrey Hinton, Li Deng, Dong Yu, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath George Dahl, and Brian Kingsbury, **Deep Neural Networks for Acoustic Modeling in Speech Recognition**, in IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 82-97, November 2012

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
INPUT 1
INPUT 2
INPUT 3
INPUT 4
INPUT 5
INPUT 6
INPUT 7
OUTPUT 1
OUTPUT 2
OUTPUT 3

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: $k(x, y) = k_1(x, y) * k_2(x, y)$

$\implies$ 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 >= 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)

	Bengali		Cantonese	
Model	perp	acc (%)	perp	acc (%)
ibm	3.4/3.5	71.5/71.2	6.8/6.16	56.8/58.5
rbm	3.3/3.4	72.1/71.6	6.2/5.7	58.3/59.3
1-k	3.7/3.8	70.1/69.7	6.8/6.2	57.0/58.3
a-2-k	3.6/3.8	70.3/70.0	6.7/6.0	57.1/58.5
m-2-k	3.7/3.8	70.3/69.9	6.7/6.1	57.1/58.4
c-2-k	3.5/3.6	71.0/70.4	6.5/5.7	57.3/58.8

- Best token error rates

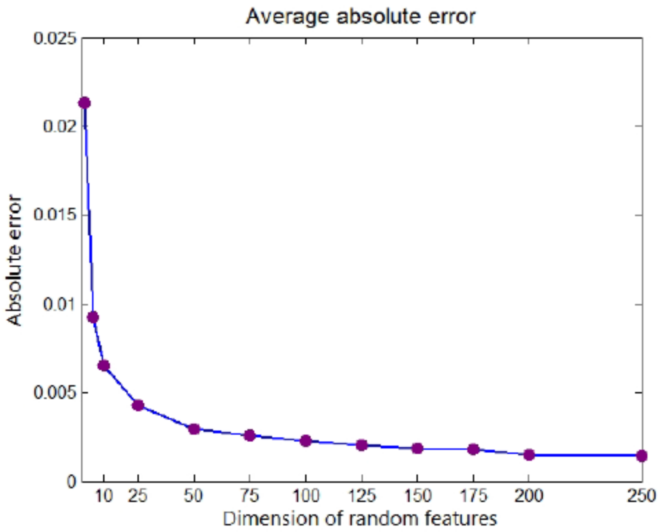
Model	Bengali	Cantonese
ibm	70.4	67.3
rbm	69.5	66.3
1-k	70.0	65.7
a-2-k	73	68.8
m-2-k	72.8	69.1
c-2-k	71.2	68.1

How many random features do we need?

- Single laplacian kernel

	Bengali		Cantonese	
Dim	perp	acc (%)	perp	acc (%)
2k	4.4/4.4	66.5/66.8	8.5/7.4	52.7/54.8
5k	4.1/4.2	67.8/67.8	7.8/7.0	53.9/56.0
10k	4.0/4.1	68.4/68.3	7.5/6.7	54.9/56.6
25k	3.8/3.9	69.2/69.0	7.1/6.4	55.9/57.3
50k	3.8/3.9	69.7/69.4	6.9/6.2	56.5/57.9
100k	3.7/3.8	70.0/69.6	6.8/6.2	56.8/58.2
200k	3.7/3.8	70.1/69.7	6.8/6.2	57.0/58.3

- Kernel Approximation error



Complementary representations?

- Token error rates for combined models:

Model	Bengali	Cantonese
BEST SINGLE SYSTEM	69.5	65.7
rbm $(h = 3, L = 2000)$ + 1-k	69.7	65.3
rbm $(h = 4, L = 1000)$ + 1-k	69.2	64.9
rbm $(h = 4, L = 2000)$ + 1-k	69.1	64.9

THANK YOU!!!