Kernel Approximation Methods for Speech Recognition

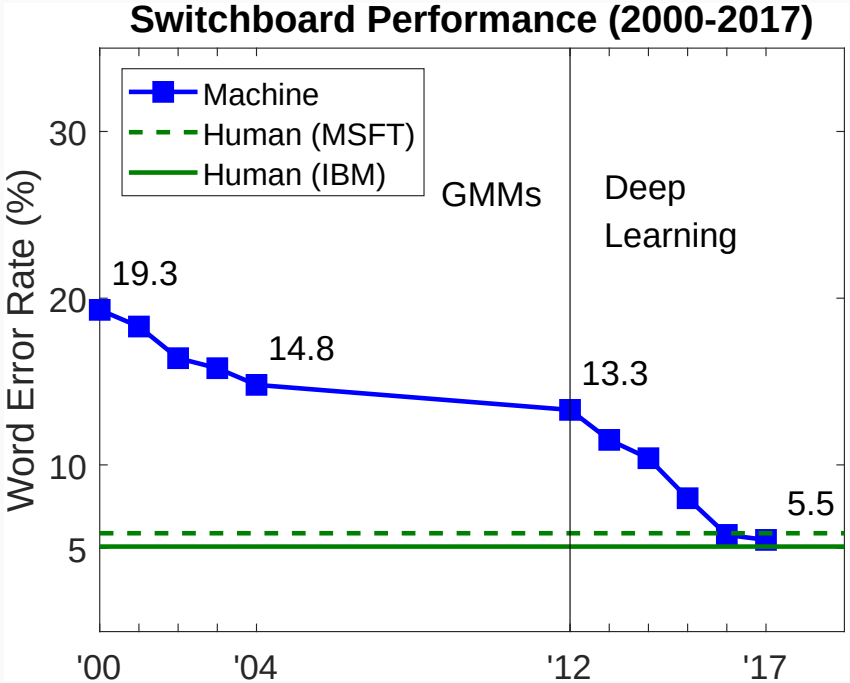
Avner May

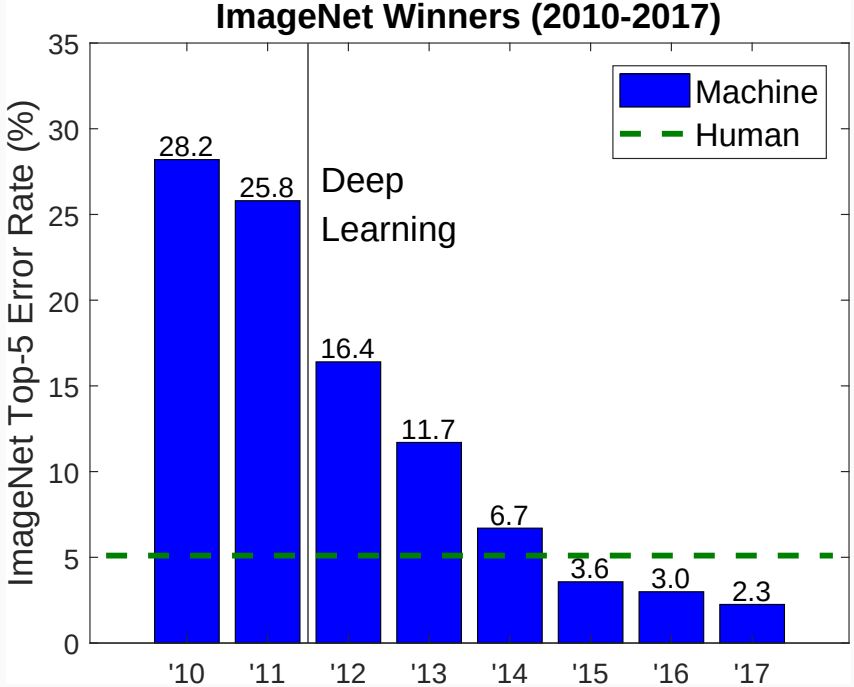
Thesis Defense Committee:
David Blei, Shih-Fu Chang, Michael Collins, Daniel Hsu, Brian Kingsbury

December 12, 2017

Rise of Deep Learning

Deep learning methods have dramatically improved the state of the art performance in various domains.





Rise of Deep Learning

Problems:

- Optimization is non-convex.
- Models are hard to interpret.

Rise of Deep Learning

Open problem:

Can other methods compete?

Motivation

Primary contribution of this dissertation:
To demonstrate that kernel methods can be scaled to compete with deep neural networks in speech recognition.

Outline

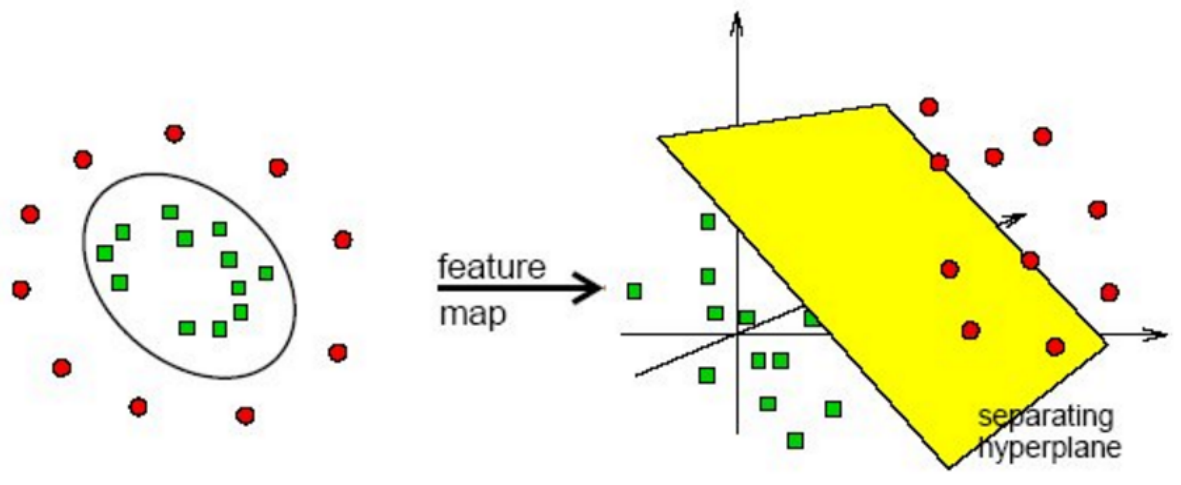
I. Background
II. Random Fourier features (RFFs) for acoustic modeling
III. Improved RFFs through feature selection
IV. Nyström method vs. RFFs
V. Concluding remarks

Outline

I. **Background**
II. Random Fourier features (RFFs) for acoustic modeling
III. Improved RFFs through feature selection
IV. Nyström method vs. RFFs
V. Concluding remarks

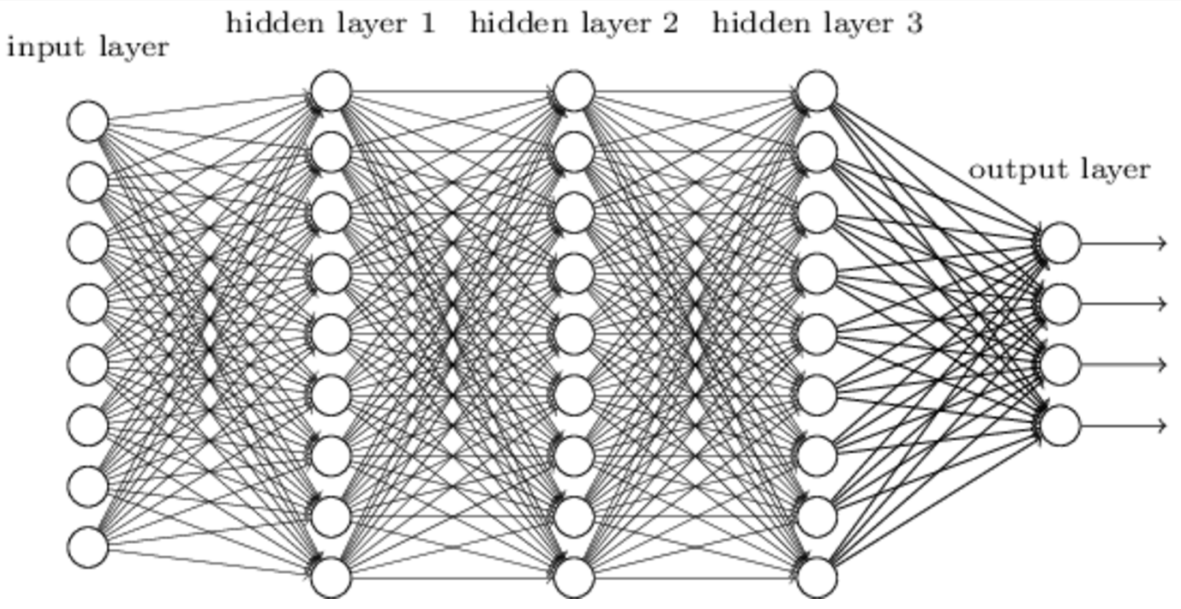
Background: Kernel Methods

- Non-linear transformation of data ($\varphi : \mathcal{X} \to \mathcal{H}$), followed by linear learning algorithm.
- $k(x, y) = \langle \varphi(x), \varphi(y) \rangle$.
  $\to k(x, y)$ can be seen as similarity score.
- Can use "kernel trick" to avoid working in $\mathcal{H}$, but at cost of $O(N^2)$ computation.



Background: DNNs

- Transforms input using composition of linear and non-linear functions.
- Feedforward network:
  $f(x) = g_L(W_L \cdot g_{L-1}(W_{L-1} \cdot g_{L-2}(W_{L-2} \cdots g_1(W_1 x))))$
- Trained using gradient methods (backpropagation).



Background: DNNs vs. Kernels

	DNNs	**Kernels**
Training Time	$O(Np)$	$O(N^2 d)$
Test Time	$O(p)$	$O(Nd)$
Universal Approximator	yes	yes
Convex	no	yes

- N is the number of training points.
- p is the number of parameters in the DNN.
- d is the dimension of the input space.
- We assume $K(x, y)$ can be computed in $O(d)$.

Background: Kernel Approximation

- Feature expansion $z : \mathbb{R}^d \to \mathbb{R}^m$ such that

$$z(x) \cdot z(y) \approx k(x, y).$$

Background: Kernel Approximation

- Feature expansion $z : \mathbb{R}^d \to \mathbb{R}^m$ such that

$$z(x) \cdot z(y) \approx k(x, y).$$

- Learn linear model over $z(x)$.

Background: Kernel Approximation

- Feature expansion $z : \mathbb{R}^d \to \mathbb{R}^m$ such that

$$z(x) \cdot z(y) \approx k(x, y).$$

- Learn linear model over $z(x)$.
- Scalability: Gives large efficiency gains at both train/test time when $m << N$.

Background: Kernel Approximation

"Random Fourier Features" (RFF) [1]:

$$z_i(x) = \sqrt{\frac{2}{m}} \cdot \cos(w_i \cdot x + b_i),$$

where $\mathbb{E}\left[z(x) \cdot z(y)\right] = k(x, y)$ if draw $w_i \sim p_k(w)$, and $b_i \sim \mathcal{U}(0, 2\pi)$.

[1] A. Rahimi, B. Recht. Random Features for Large-Scale Kernel Machines. 2007

Background: Kernel Approximation

"Random Fourier Features" (RFF) [1]:

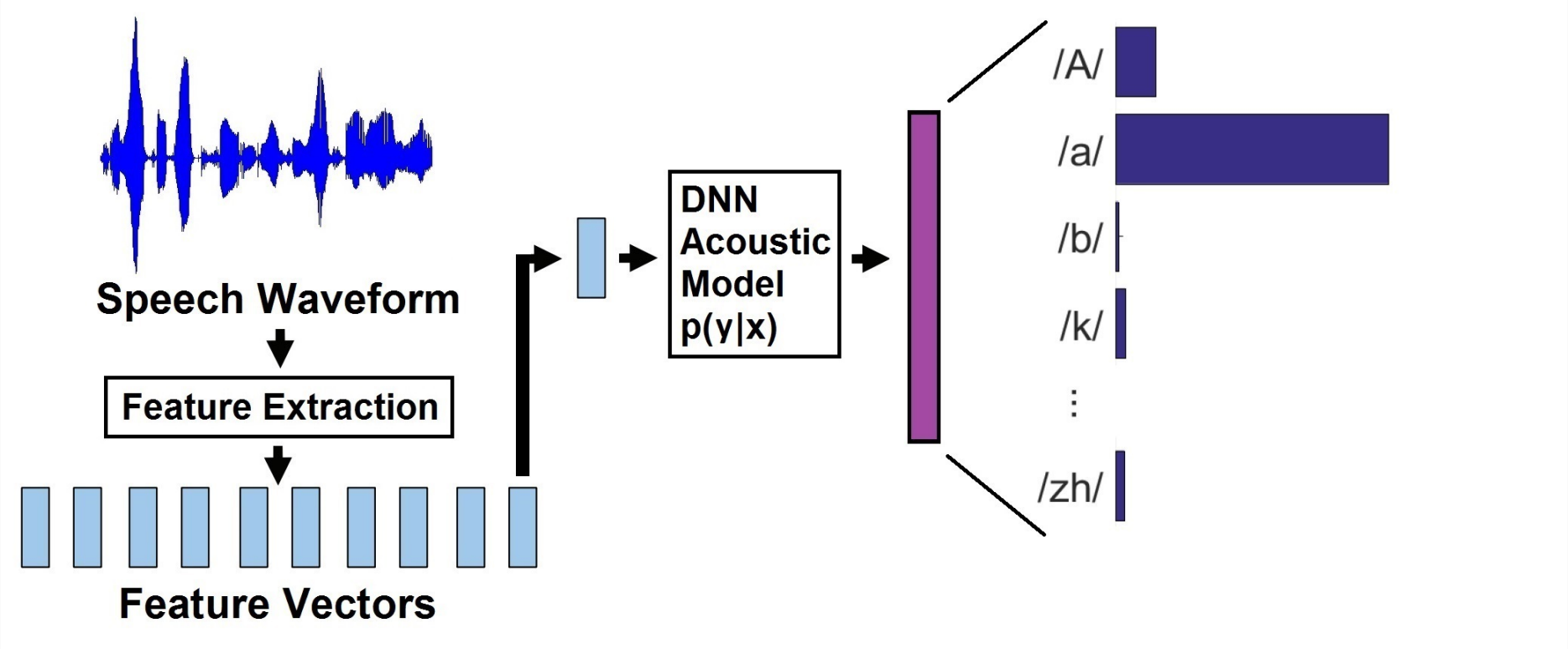
$$z_i(x) = \sqrt{\frac{2}{m}} \cdot \cos(w_i \cdot x + b_i),$$

where $\mathbb{E}\left[z(x) \cdot z(y)\right] = k(x, y)$ if draw $w_i \sim p_k(w)$, and $b_i \sim \mathcal{U}(0, 2\pi)$.

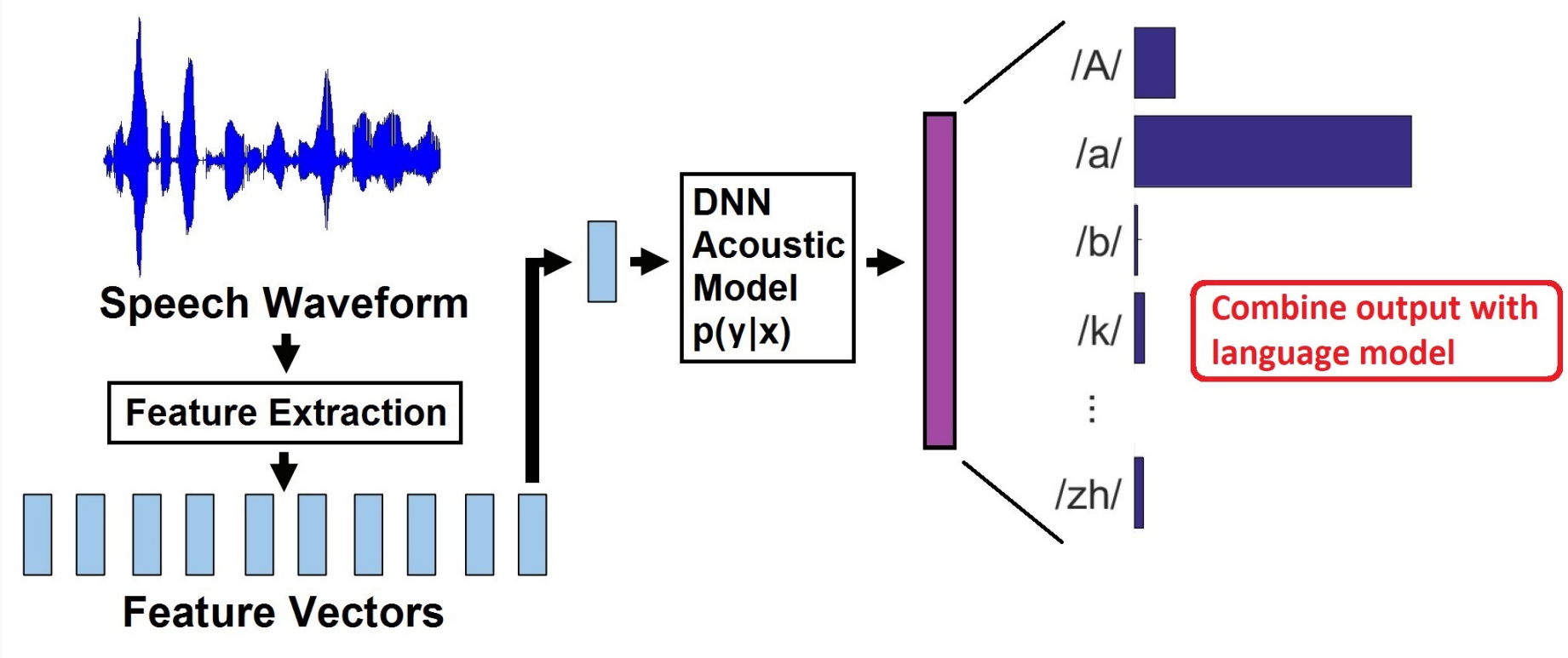
Kernel	$K(x, y)$	$p_k(w)$
Gaussian	$e^{-\|x-y\|_2^2/2\sigma^2}$	Normal$(0_d, \frac{1}{\sigma^2} \mathrm{I}_d)$
Laplacian	$e^{-\lambda\|x-y\|_1}$	Cauchy$(0_d, \lambda)$

[1] A. Rahimi, B. Recht. Random Features for Large-Scale Kernel Machines. 2007

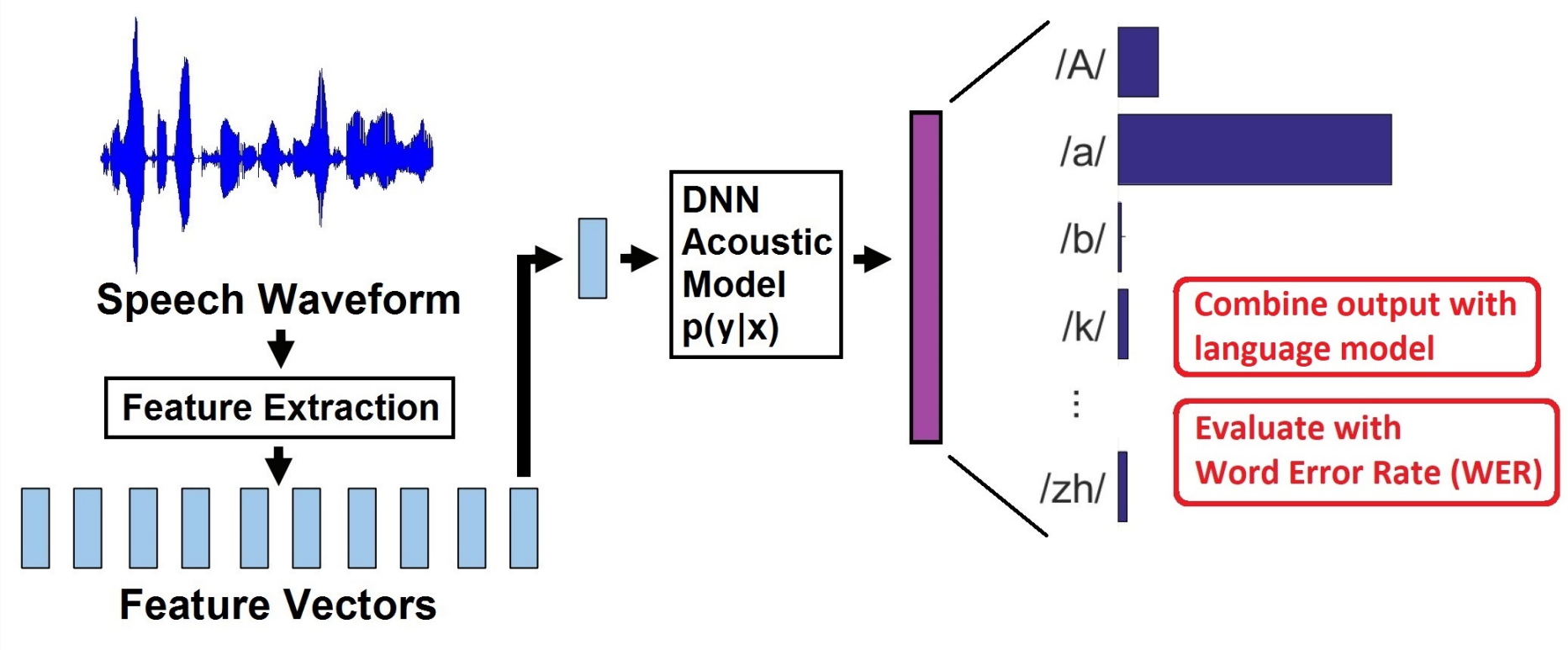
Background: Speech Recognition



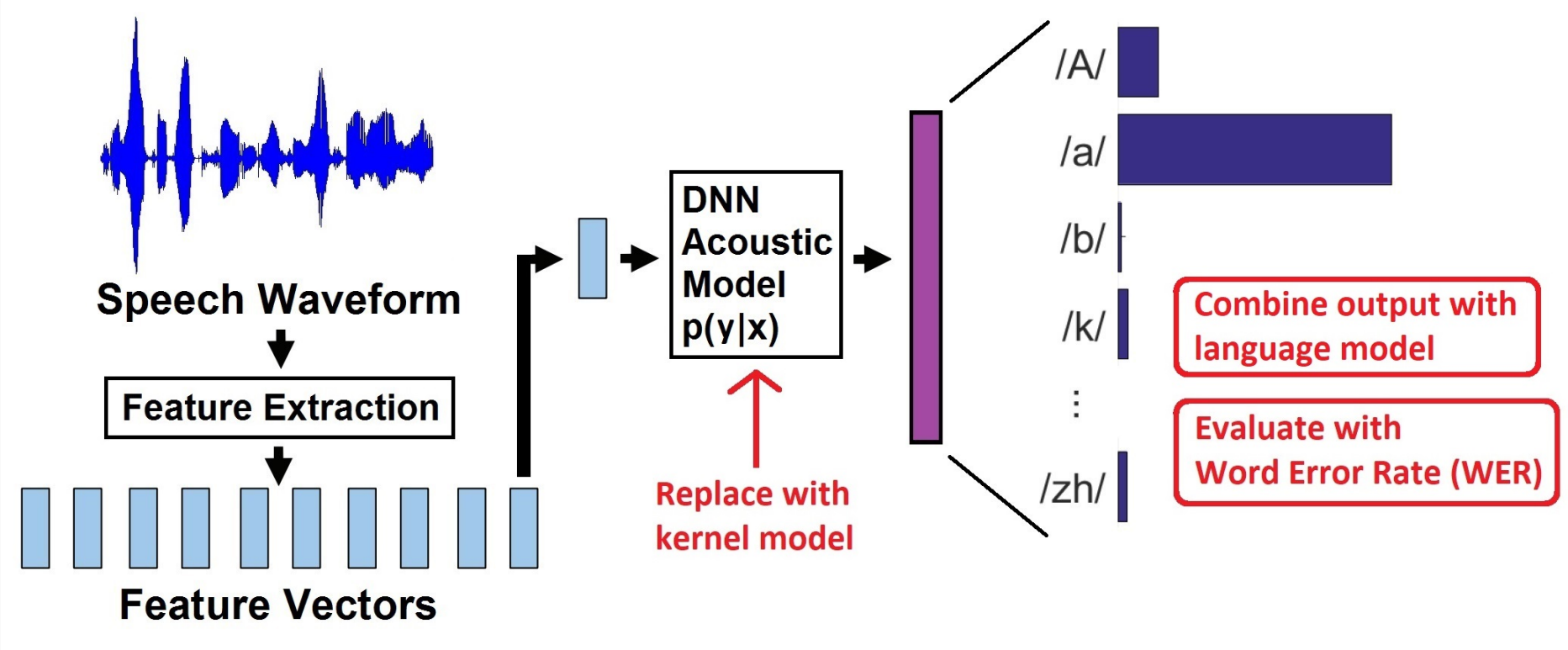
Background: Speech Recognition



Background: Speech Recognition



Background: Speech Recognition



Outline

I. Background
II. **Random Fourier features (RFFs) for acoustic modeling**
III. Improved RFFs through feature selection
IV. Nyström method vs. RFFs
V. Concluding remarks

Random Fourier Features for Acoustic Modeling

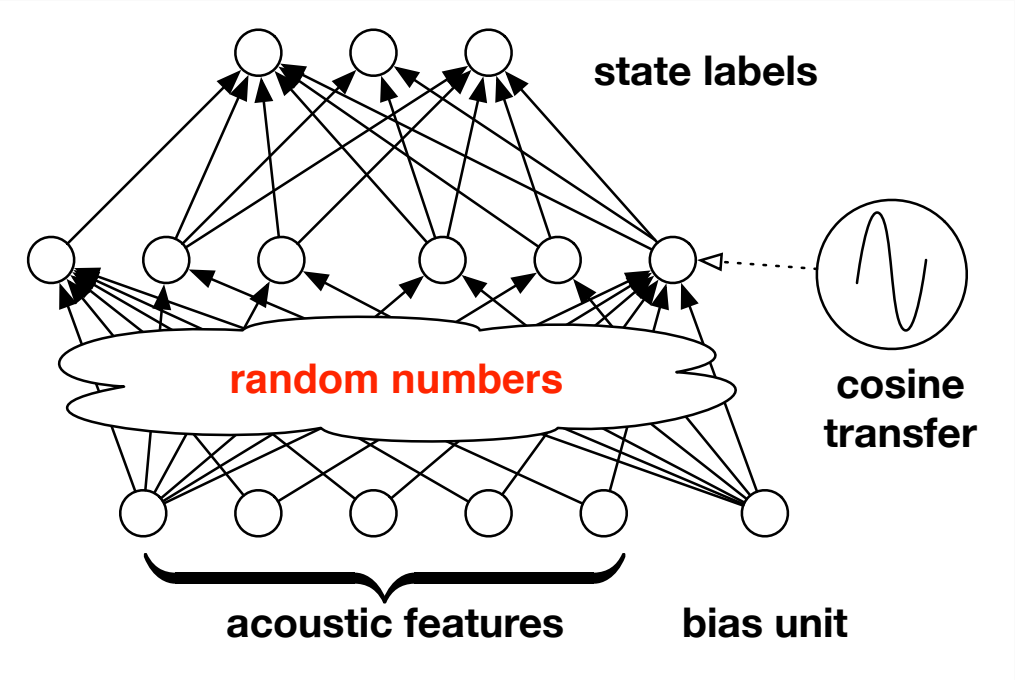
We use RFFs $z(x)$ in a log-linear model for $p(y|x)$:

$$\text{Model}: \qquad p(y|x; W) = \frac{\exp\left(\langle w_y,\ z(x)\rangle + b_y\right)}{\sum_{y'} \exp\left(\langle w_{y'},\ z(x)\rangle + b_{y'}\right)}$$

Random Fourier Features for Acoustic Modeling

We use RFFs $z(x)$ in a log-linear model for $p(y|x)$:

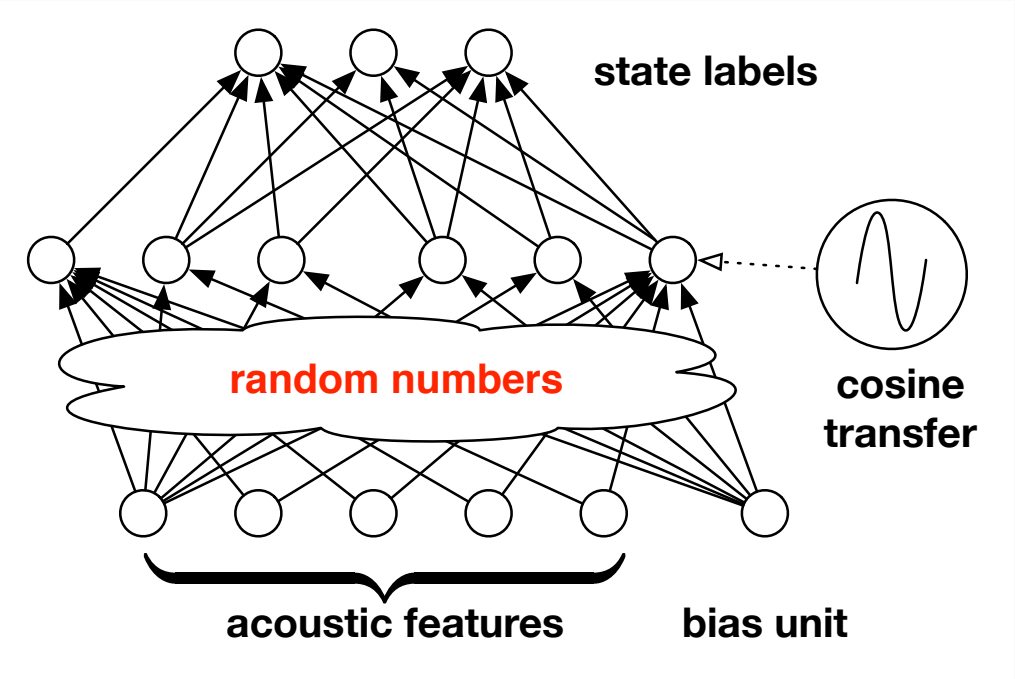
$$\text{Model}: \qquad p(y|x; W) = \frac{\exp\left(\langle w_y,\ z(x)\rangle + b_y\right)}{\sum_{y'} \exp\left(\langle w_{y'},\ z(x)\rangle + b_{y'}\right)}$$



Random Fourier Features for Acoustic Modeling

We use RFFs $z(x)$ in a log-linear model for $p(y|x)$:

$$\text{Model}: \qquad p(y|x; W) = \frac{\exp\left(\langle w_y,\ z(x)\rangle + b_y\right)}{\sum_{y'} \exp\left(\langle w_{y'},\ z(x)\rangle + b_{y'}\right)}$$



$$\text{Training}: \qquad \arg\max_W \frac{1}{N} \sum_{i=1}^{N} \log(p(y_i|x; W))$$

Random Fourier Features for Acoustic Modeling

We use 2 tricks to improve our kernel models:

1. “Linear bottleneck” in output matrix [1].
2. Early stopping using novel metric for improved WER performance [2].

---

[1] Low-rank Matrix Factorization for Deep Neural Network Training with High-dimensional Output Targets. T. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, B. Ramabhadran. ICASSP 2013.

[2] A Comparison Between DNNs and Kernel Acoustic Models for Speech Recognition. Z. Lu, D. Guo, A. Garakani, K. Liu, **A. May**, A. Bellet, L. Fan, M. Collins, B. Kingsbury, M. Picheny, F. Sha. ICASSP 2016.

Task Details

We use 4 speech recognition datasets:

- Bengali and Cantonese from IARPA Babel program ($\sim$24 hours)
- TIMIT benchmark task ($\sim$3.5 hours)
- 50-hour subset of Broadcast News dataset (BN50).

Dataset	Train	# Features	# Classes
Broadcast News	16M	360	5000
Bengali	7.7M	360	1000
Cantonese	7.5M	360	1000
TIMIT	2.3M	440	147

Results: Best DNN vs. Best Kernel

- Kernels: 100k random features (200k for TIMIT).
- DNNs: 4 layers, $\tanh$ non-linearity.

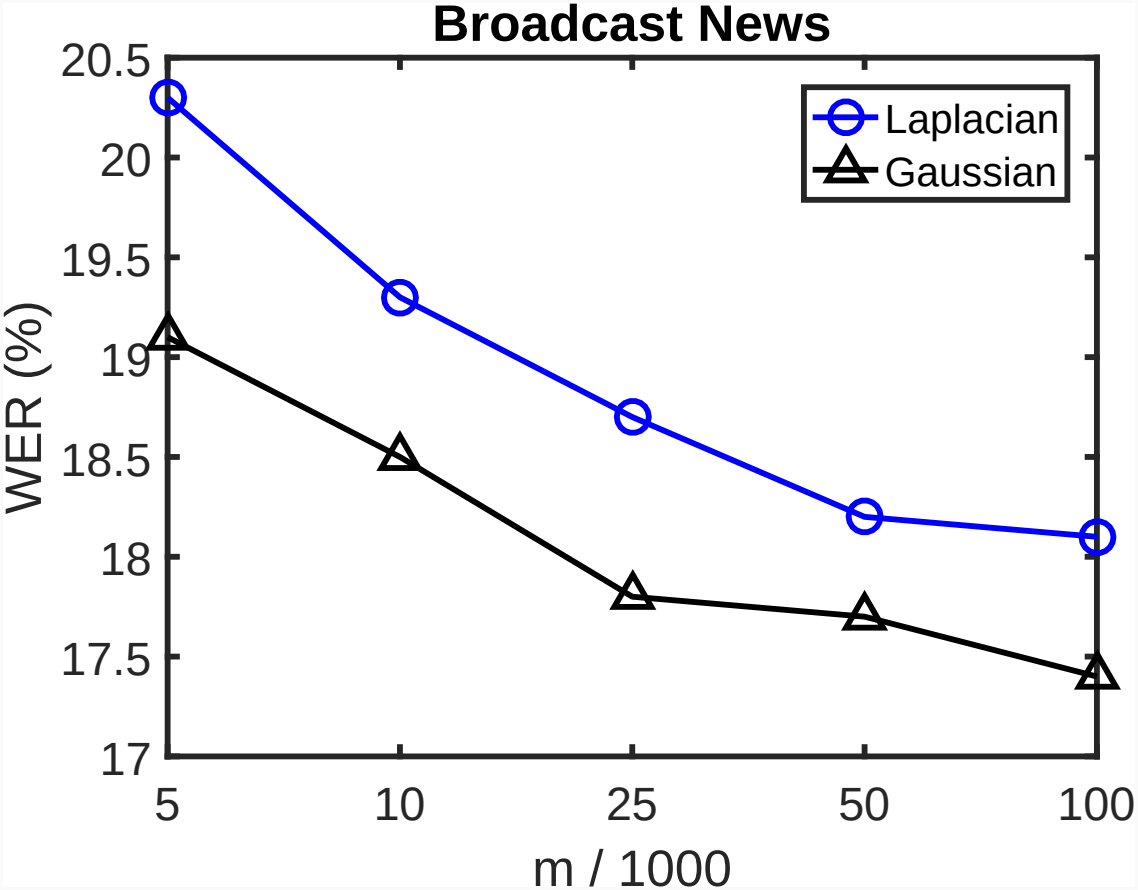
Results: Best DNN vs. Best Kernel

- Kernels: 100k random features (200k for TIMIT).
- DNNs: 4 layers, $\tanh$ non-linearity.

	DNN	Kernel
Broadcast News	**16.4**	17.1
Bengali	**70.2**	71.4
Cantonese	67.1	67.1
TIMIT	18.6	18.6

Table 1: Development set results (WER %). Best DNN vs. best kernel.

Results: Role of Number of Random Features



$$\textit{Laplacian} : k(x,y) = \exp\Big(-\lambda\|x-y\|_1\Big)$$

$$\textit{Gaussian} : k(x,y) = \exp\Big(-\|x-y\|_2^2/2\sigma^2\Big)$$

Random Fourier Features for Acoustic Modeling

Primary contributions:

1. Effectively scaled RFFs to acoustic modeling, and compared with DNNs.
2. Matched performance on two of four datasets.

Random Fourier Features for Acoustic Modeling

Primary contributions:

1. Effectively scaled RFFs to acoustic modeling, and compared with DNNs.
2. Matched performance on two of four datasets.

Remaining problems:

1. We need a ton of random features.
2. We still lag well-behind DNNs on two datasets.

Outline

I. Background
II. Random Fourier features (RFFs) for acoustic modeling
III. **Improved RFFs through feature selection**
IV. Nyström method vs. RFFs
V. Concluding remarks

Feature Selection: First try

Goal: Fast online algorithm for feature selection in multi-class setting.

- **First try**: Use "FOBOS," Duchi and Singer's Forward-Backward Splitting algorithm [1] with ℓ_1/ℓ_2 regularization.

---

[1] Efficient Online and Batch Learning Using Forward Backward Splitting. J. Duchi, Y. Singer. JMLR 2009.

Feature Selection: First try

Goal: Fast online algorithm for feature selection in multi-class setting.

- **First try**: Use "FOBOS," Duchi and Singer's Forward-Backward Splitting algorithm [1] with ℓ_1/ℓ_2 regularization.
- **Problem**: Attaining row-sparsity requires extremely strong regularization $\Rightarrow$ model unable to learn.

---

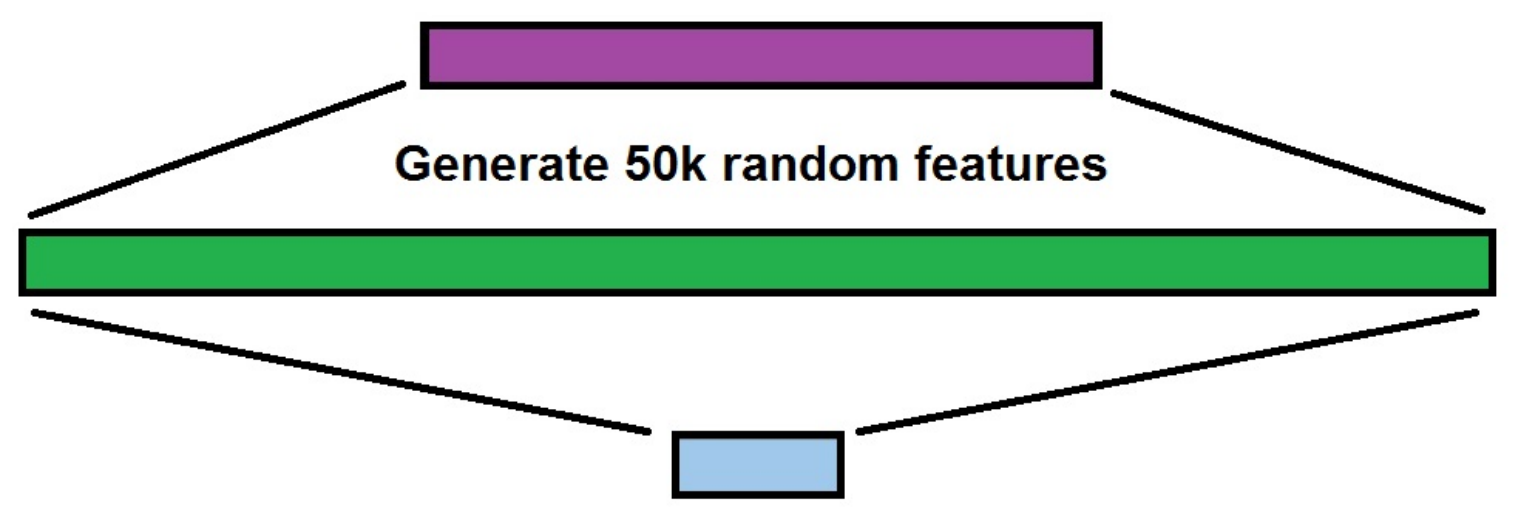
[1] Efficient Online and Batch Learning Using Forward Backward Splitting. J. Duchi, Y. Singer. JMLR 2009.

Our Proposed Feature Selection Algorithm[1]

Say you want to select a total of 50k features...

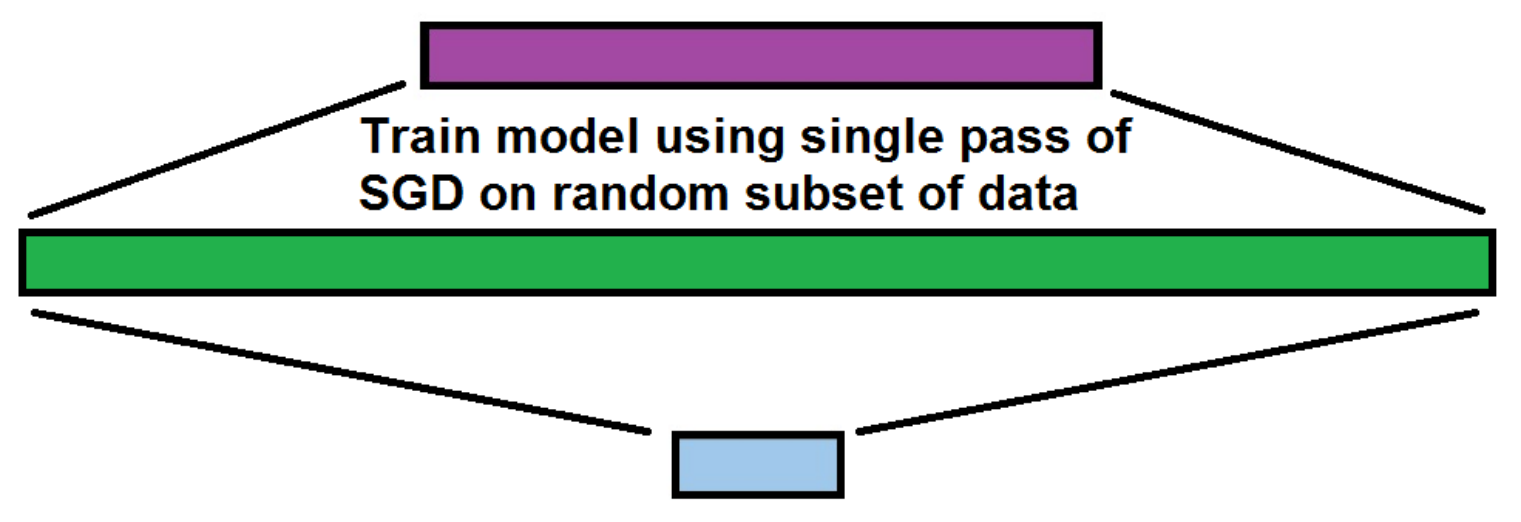
[1] Compact Kernel Models for Acoustic Modeling via Random Feature Selection. **A. May**, M. Collins, D. Hsu, B. Kingsbury. ICASSP 2016.

Our Proposed Feature Selection Algorithm[1]



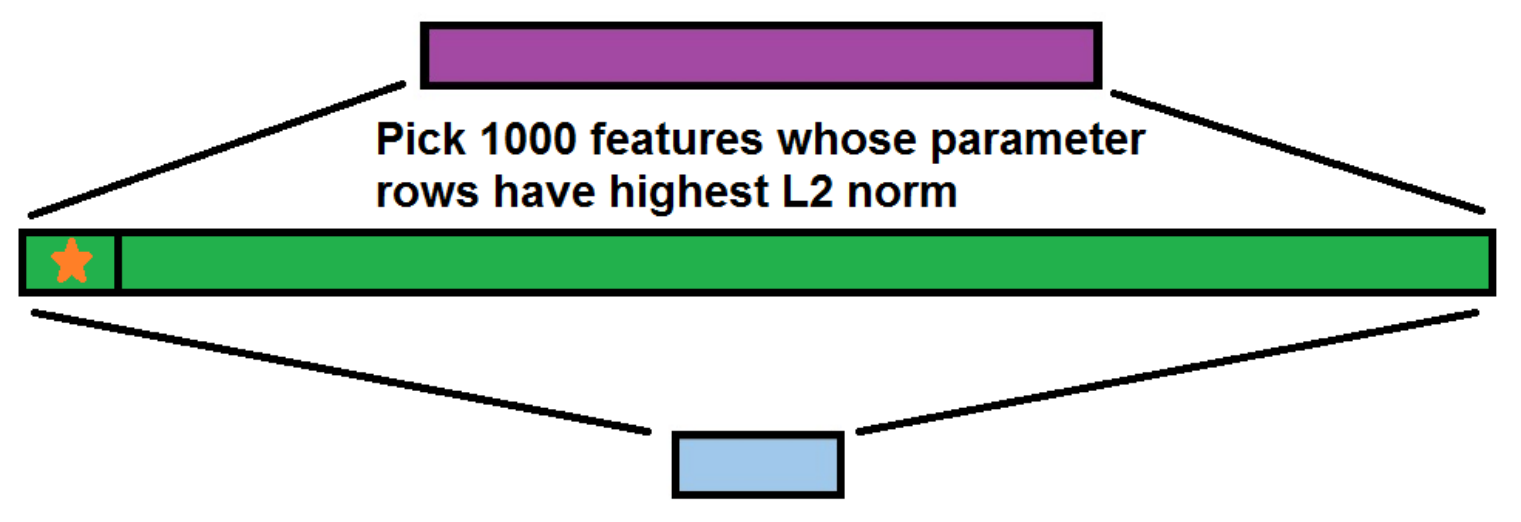
[1] Compact Kernel Models for Acoustic Modeling via Random Feature Selection. **A. May**, M. Collins, D. Hsu, B. Kingsbury. ICASSP 2016.

Our Proposed Feature Selection Algorithm[1]



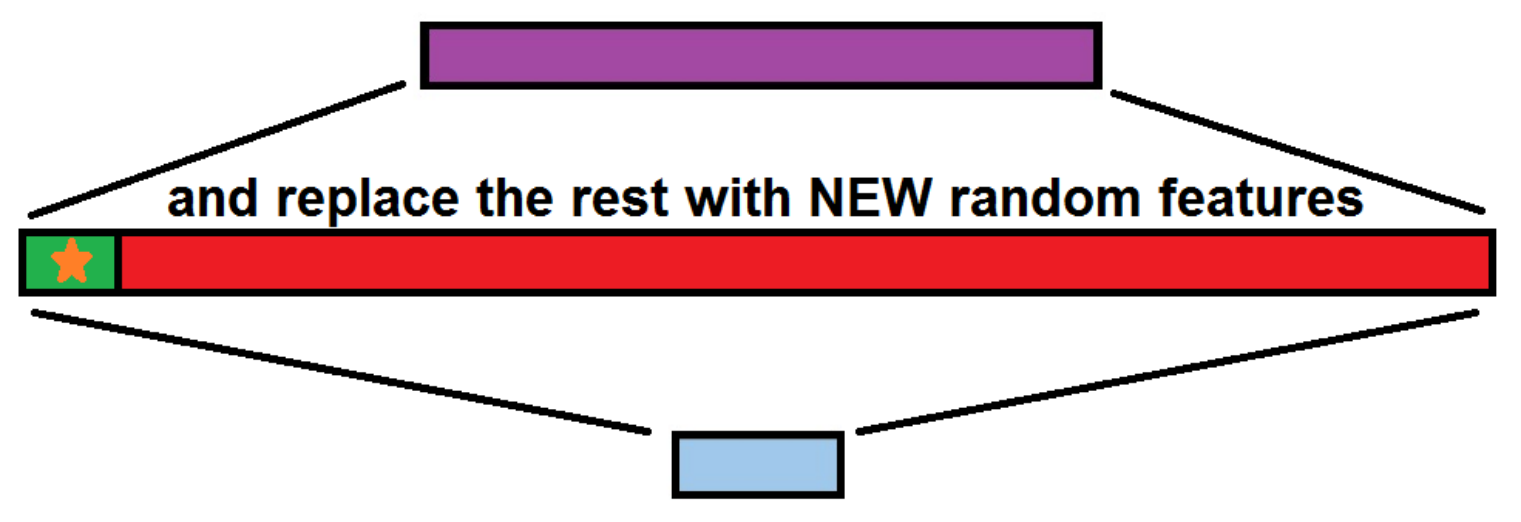
[1] Compact Kernel Models for Acoustic Modeling via Random Feature Selection. **A. May**, M. Collins, D. Hsu, B. Kingsbury. ICASSP 2016.

Our Proposed Feature Selection Algorithm[1]



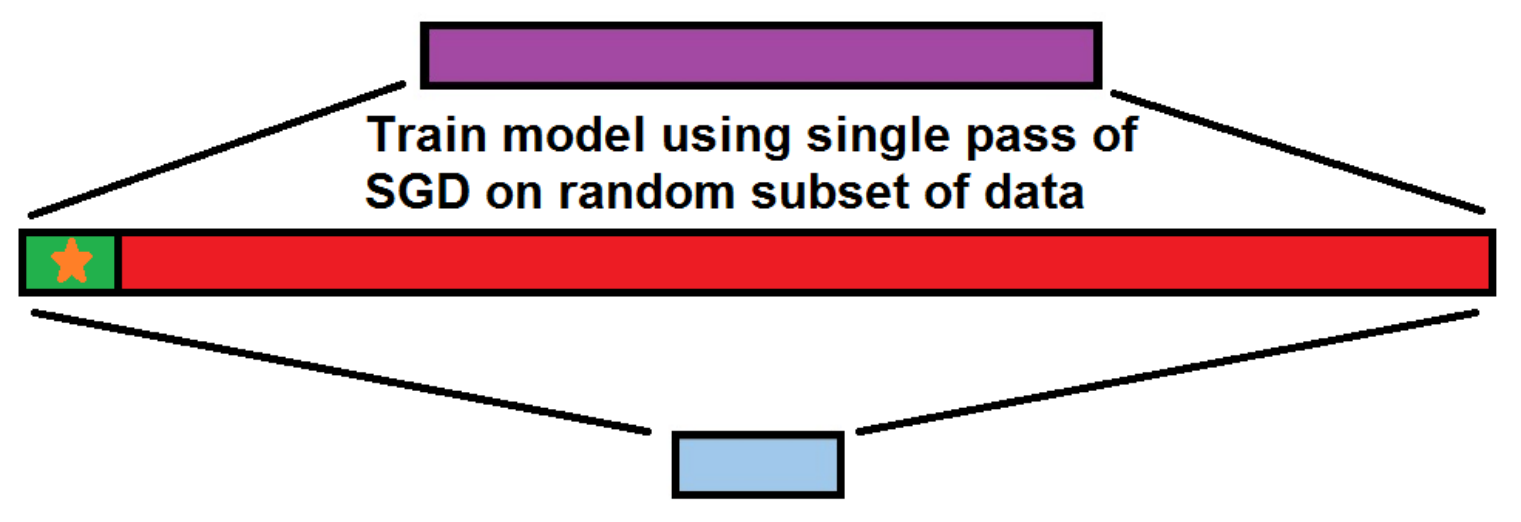
[1] Compact Kernel Models for Acoustic Modeling via Random Feature Selection. **A. May**, M. Collins, D. Hsu, B. Kingsbury. ICASSP 2016.

Our Proposed Feature Selection Algorithm[1]



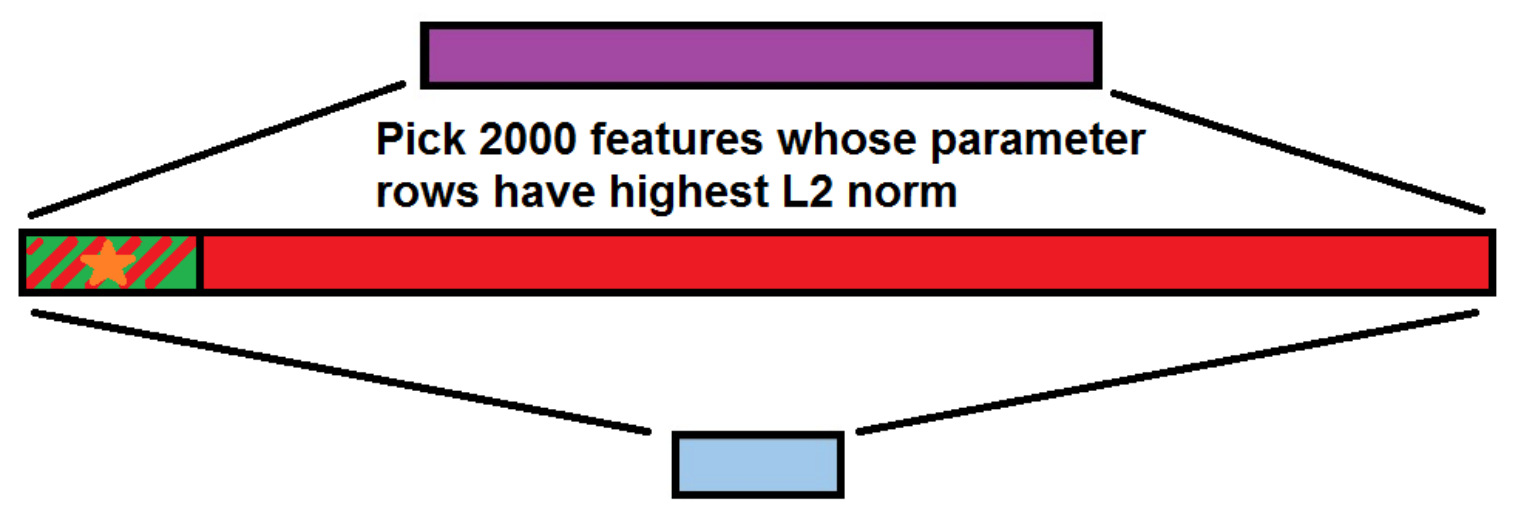
[1] Compact Kernel Models for Acoustic Modeling via Random Feature Selection. **A. May**, M. Collins, D. Hsu, B. Kingsbury. ICASSP 2016.

Our Proposed Feature Selection Algorithm[1]



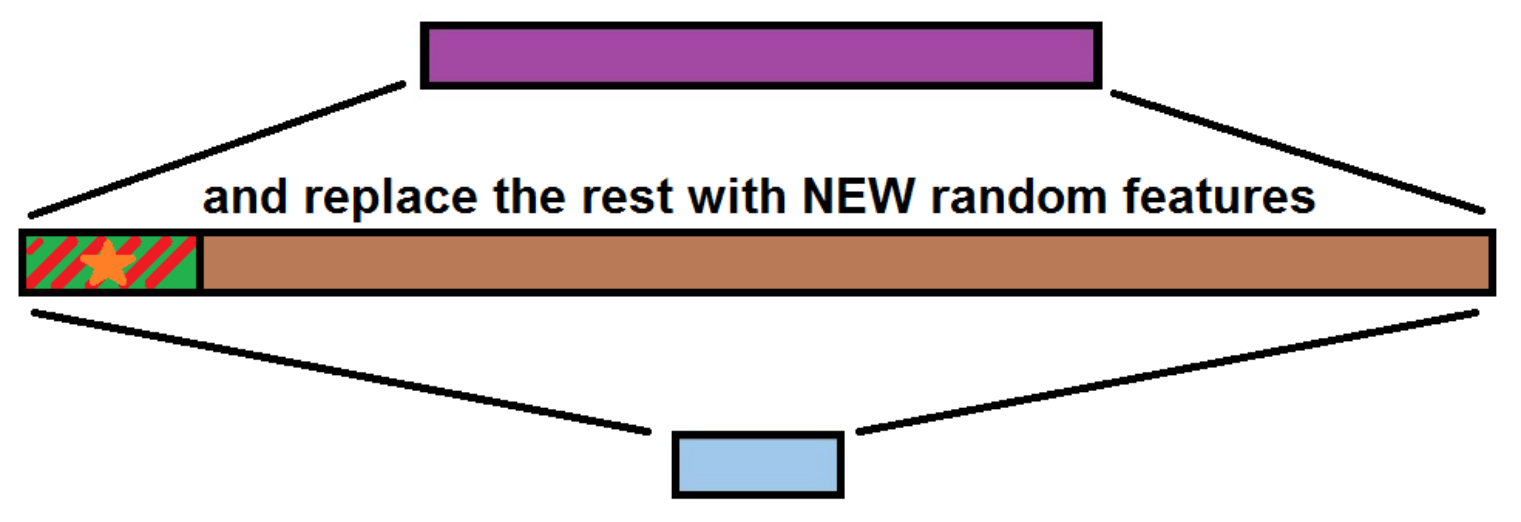
[1] Compact Kernel Models for Acoustic Modeling via Random Feature Selection. **A. May**, M. Collins, D. Hsu, B. Kingsbury. ICASSP 2016.

Our Proposed Feature Selection Algorithm[1]



[1] Compact Kernel Models for Acoustic Modeling via Random Feature Selection. **A. May**, M. Collins, D. Hsu, B. Kingsbury. ICASSP 2016.

Our Proposed Feature Selection Algorithm[1]



[1] Compact Kernel Models for Acoustic Modeling via Random Feature Selection. **A. May**, M. Collins, D. Hsu, B. Kingsbury. ICASSP 2016.

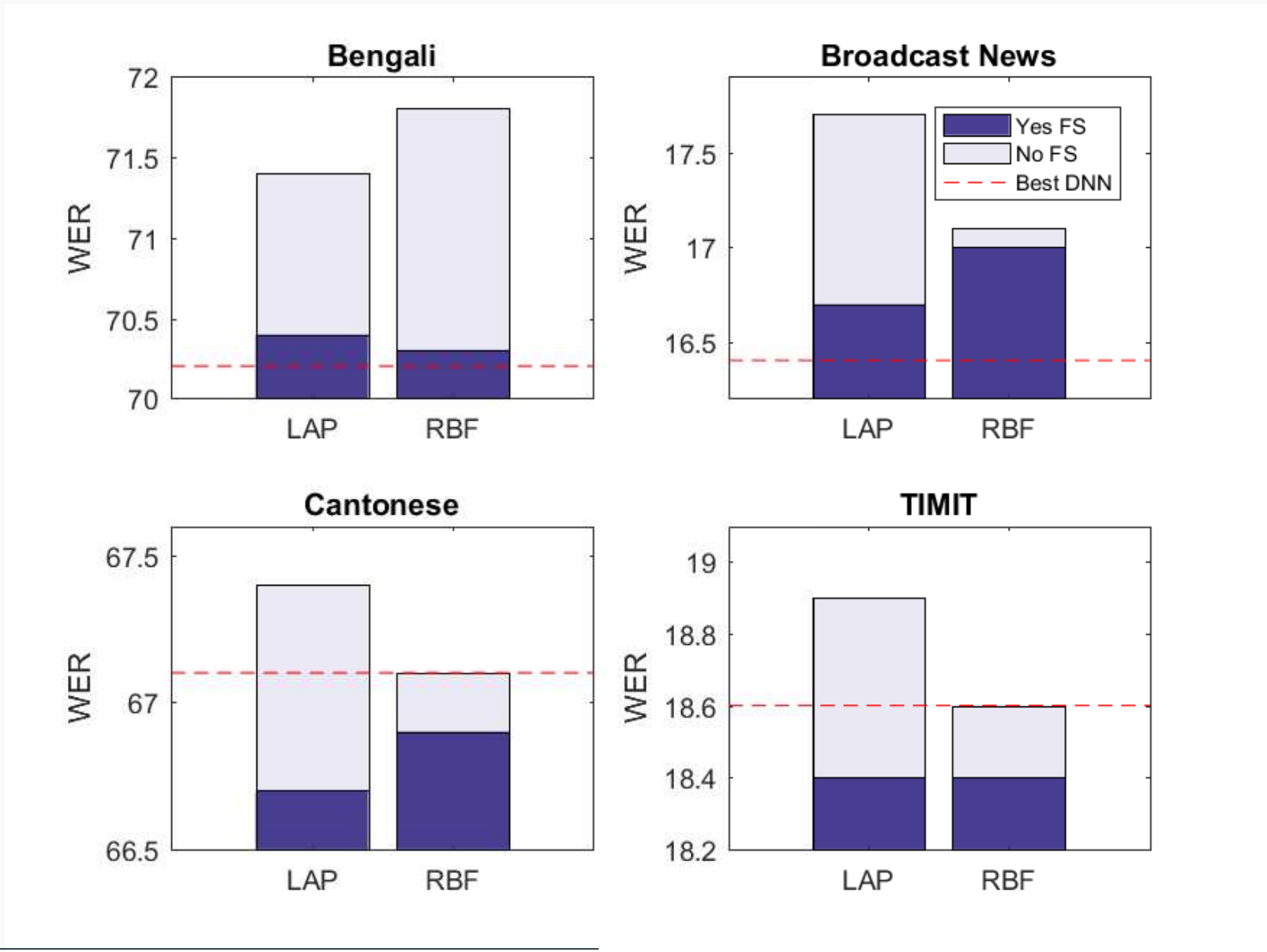
Our Proposed Feature Selection Algorithm[1]

And so on...

until we build up to 50k random features

[1] Compact Kernel Models for Acoustic Modeling via Random Feature Selection. **A. May**, M. Collins, D. Hsu, B. Kingsbury. ICASSP 2016.

Development Set Results: Word Error Rate*

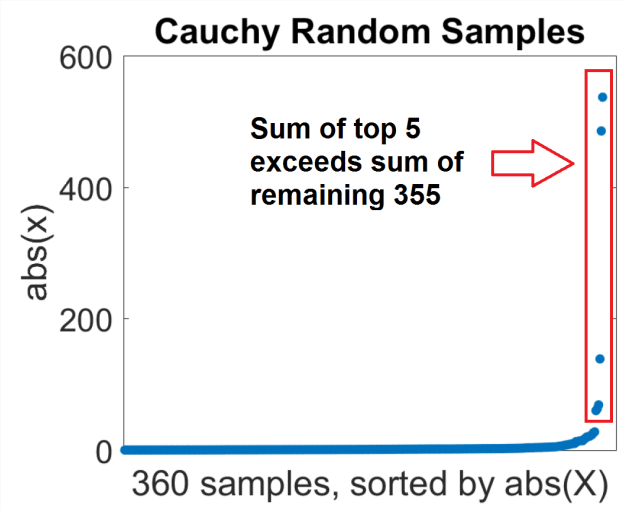


* All of these models use bottleneck + ERP for LR decay

Feature Selection: A Deeper Look

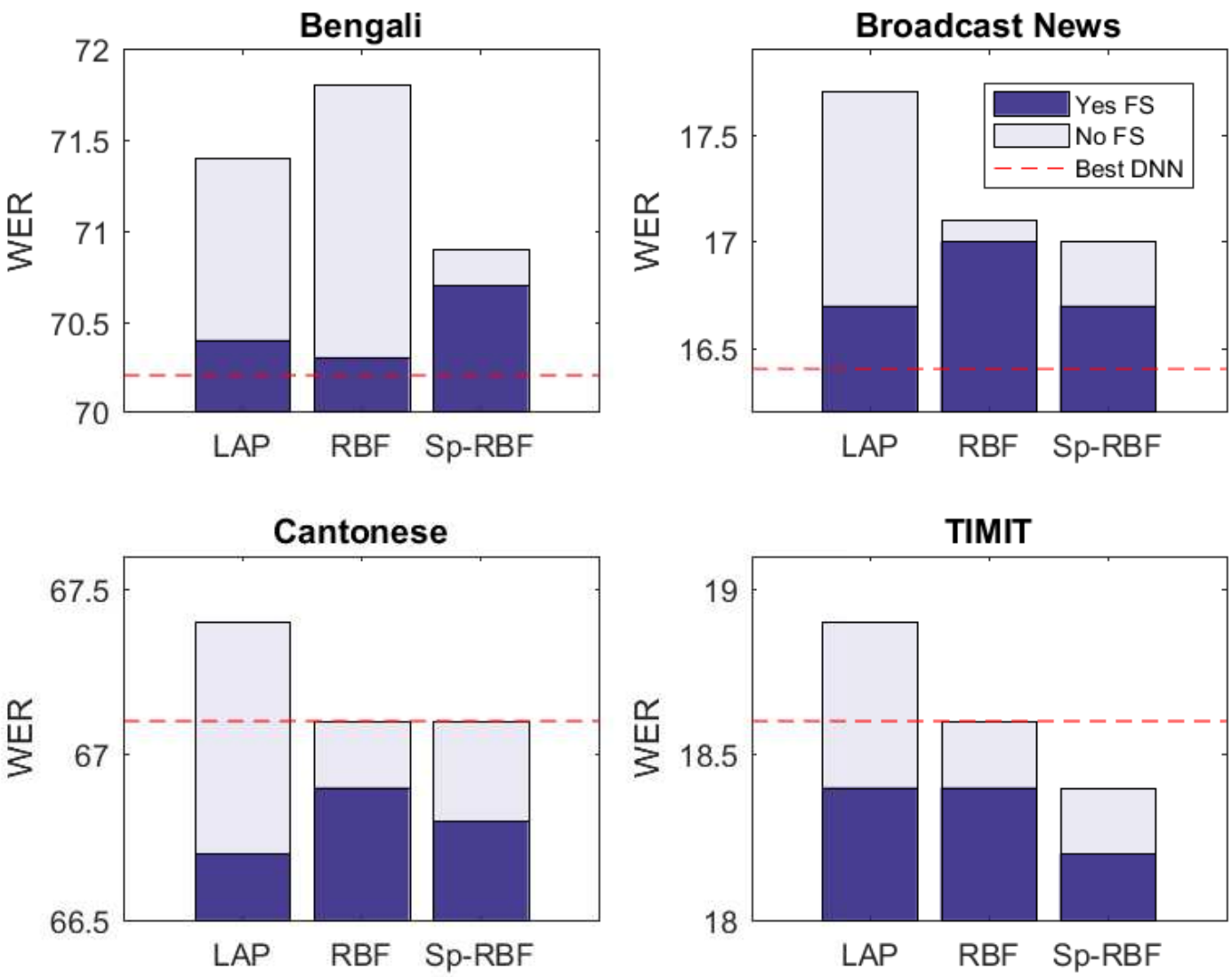
Why does the Laplacian kernel benefit so much from feature selection, while the Gaussian kernel does not?

- To approximate Laplacian kernel, draw from Cauchy. Fat tailed!



- Projections are effectively “sparse”.
- We can construct sparse random projections explicitly.
  $\rightarrow$ draw $k = 5$ non-zero weights from Gaussian, rest are 0.
- We call this the “Sparse Gaussian kernel.”

Development Set Results: Word Error Rate*



∗ All of these models use bottleneck + ERP for LR decay

Results: Can we use even MORE random features?

	$100k$	$400k$
Laplacian	17.7	17.4
+FS	16.7	**16.4**

Table 2: Broadcast News Development set results.

Results: Best DNN vs. Best Kernel

	DNN	Kernel
Broadcast News	11.7	**11.6**
Bengali	**69.1**	69.2
Cantonese	63.7	**63.2**
TIMIT	20.5	**20.4**

Table 3: Test set results (WER %). Best DNN vs. best kernel.

Results: Best DNN vs. Best Kernel

	DNN	Kernel
Our Results	20.5	**20.4**
Prior Work [1]	**20.5**	21.3
Prior Work [2]	N/A	20.9

Table 4: Test set results on TIMIT.

[1] Kernel Methods Match Deep Neural Networks on TIMIT. P.S. Huang, H. Avron, T. Sainath, V. Sindhwani, B. Ramabhadran. ICASSP 2014.

[2] Efficient One-vs-One Kernel Ridge Regression for Speech Recognition. J. Chen, L. Wu, K. Audhkhasi, B. Kingsbury, B. Ramabhadran. ICASSP 2016.

Improved RFFs Through Feature Selection

Our contributions:

1. We propose a novel feature selection algorithm, and demonstrate its effectiveness for acoustic modeling.
2. We propose the new “Sparse Gaussian” kernel, which does well at acoustic modeling, both with and without feature selection.
3. We attain WER performance comparable to DNNs, on all 4 datasets.

Outline

I. Background
II. Random Fourier features (RFFs) for acoustic modeling
III. Improved RFFs through feature selection
IV. **Nyström method vs. RFFs**
V. Concluding remarks

Motivation: Nyström vs. RFF

Previous work argues Nyström is better than RFF [1]:

- Better at approximating kernel.
- Better generalization bounds.
- Better empirical performance on regression/classification tasks.

---

[1] Nyström Method vs Random Fourier Features: A Theoretical and Empirical Comparison. Yang et al. NIPS 2012.

Motivation: Nyström vs. RFF

Previous work argues Nyström is better than RFF [1]:

- Better at approximating kernel.
- Better generalization bounds.
- Better empirical performance on regression/classification tasks.
  $\rightarrow$ **Experiments**: Use up to 1000 features on UCI data.

---

[1] Nyström Method vs Random Fourier Features: A Theoretical and Empirical Comparison. Yang et al. NIPS 2012.

Motivation: Nyström vs. RFF

Previous work argues Nyström is better than RFF [1]:

- Better at approximating kernel.
- Better generalization bounds.
- Better empirical performance on regression/classification tasks.
  $\rightarrow$ **Experiments**: Use up to 1000 features on UCI data.

Awesome! Let's try them at our tasks...

- Will Nyström be able to scale to our larger, harder tasks?
- Will it outperform RFF for a fixed computational budget?

---

[1] Nyström Method vs Random Fourier Features: A Theoretical and Empirical Comparison. Yang et al. NIPS 2012.

Background: Nyström method

Idea: Use low rank approximation of kernel matrix $K \approx Z^T Z$ to generate vectors $z(x)$ for kernel approximation.

$$\begin{aligned} K &= KK^{-1}K \\ &= (KK^{-1/2})(K^{-1/2}K) \end{aligned}$$

Now, letting $K_{x_j} = [k(x_1, x_j), \ldots, k(x_N, x_j)]^T$, we have:

$$\begin{aligned} k(x_i, x_j) &= \left(K_{x_i}^T K^{-1/2}\right)\left(K^{-1/2} K_{x_j}\right) \\ &= z(x_i)^T z(x_j) \\ z(x_j) &= K^{-1/2} K_{x_j} \end{aligned}$$

Background: Nyström method

Idea: Use low rank approximation of kernel matrix $K \approx Z^T Z$ to generate vectors $z(x)$ for kernel approximation.

$$\begin{aligned} K &= KK^{-1}K \\ &= (KK^{-1/2})(K^{-1/2}K) \end{aligned}$$

Now, letting $K_{x_j} = [k(x_1, x_j), \ldots, k(x_N, x_j)]^T$, we have:

$$\begin{aligned} k(x_i, x_j) &= \left(K_{x_i}^T K^{-1/2}\right)\left(K^{-1/2} K_{x_j}\right) \\ &= z(x_i)^T z(x_j) \\ z(x_j) &= K^{-1/2} K_{x_j} \end{aligned}$$

Computational cost?

- Pre-processing: $O(N^3)$ for SVD.
- During training, for each datapoint:
  - $O(Nd)$ to compute K_{x_j}.
  - $O(N^2)$ to multiply by $K^{-1/2}$.

Background: Nyström method

Nyström method: Use the representation $z(x)$ corresponding to the kernel matrix $\hat{K}$ for a random subset of the training data $\hat{x}_1, \ldots, \hat{x}_m$.

$$\begin{aligned} k(x_i, x_j) &\approx z(x_i)^T z(x_j) \\ z(x_j) &= \hat{K}^{-1/2} \hat{K}_{x_j} \\ \hat{K}_{x_j} &= [k(\hat{x}_1, x_j), \ldots, k(\hat{x}_m, x_j)]^T \end{aligned}$$

Background: Nyström method

Nyström method: Use the representation $z(x)$ corresponding to the kernel matrix $\hat{K}$ for a random subset of the training data $\hat{x}_1, \ldots, \hat{x}_m$.

$$\begin{aligned} k(x_i, x_j) &\approx z(x_i)^T z(x_j) \\ z(x_j) &= \hat{K}^{-1/2} \hat{K}_{x_j} \\ \hat{K}_{x_j} &= [k(\hat{x}_1, x_j), \ldots, k(\hat{x}_m, x_j)]^T \end{aligned}$$

We can visualize this as follows:

$$K \approx \begin{bmatrix} \hat{K}_{x_1}^T \\ \hat{K}_{x_2}^T \\ \vdots \\ \hat{K}_{x_N}^T \end{bmatrix} \begin{bmatrix} \\ \hat{K}^{-1/2} \\ \\ \end{bmatrix} \begin{bmatrix} \\ \hat{K}^{-1/2} \\ \\ \end{bmatrix} \begin{bmatrix} \hat{K}_{x_1} & \hat{K}_{x_2} & \cdots & \hat{K}_{x_N} \\ & & & \end{bmatrix}$$

Background: Nyström method

Nyström method: Use the representation $z(x)$ corresponding to the kernel matrix $\hat{K}$ for a random subset of the training data $\hat{x}_1, \ldots, \hat{x}_m$.

$$\begin{aligned} k(x_i, x_j) &\approx z(x_i)^T z(x_j) \\ z(x_j) &= \hat{K}^{-1/2} \hat{K}_{x_j} \\ \hat{K}_{x_j} &= [k(\hat{x}_1, x_j), \ldots, k(\hat{x}_m, x_j)]^T \end{aligned}$$

We can visualize this as follows:

$$K \approx \begin{bmatrix} \hat{K}_{x_1}^T \\ \hat{K}_{x_2}^T \\ \vdots \\ \hat{K}_{x_N}^T \end{bmatrix} \begin{bmatrix} \\ \hat{K}^{-1/2} \\ \\ \end{bmatrix} \begin{bmatrix} \\ \hat{K}^{-1/2} \\ \\ \end{bmatrix} \begin{bmatrix} \hat{K}_{x_1} & \hat{K}_{x_2} & \cdots & \hat{K}_{x_N} \end{bmatrix}$$

****Computing $\mathbf{z(x_j)}$ takes time $\mathbf{O(md + m^2)}$.****

Task Details: Nyström vs. RFF

Datasets:

- **Classification**: *TIMIT*, Forest, CovType, Cod-RNA, Adult
- **Regression**: YearPred, Census, CPU

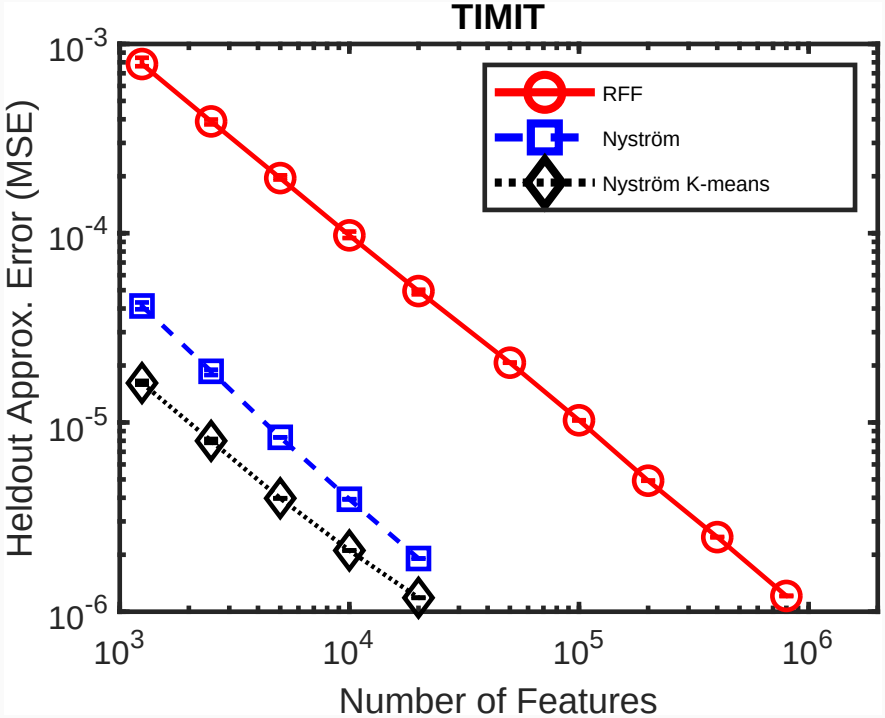
Training Details:

- Use up to 1.6M RFF features, and up to $20k$ Nyström features.
- Models trained using SGD, without any additional tricks.
- Nyström: choose the "landmark points" randomly, or with k-means [1].
- We use the Gaussian kernel for all experiments.

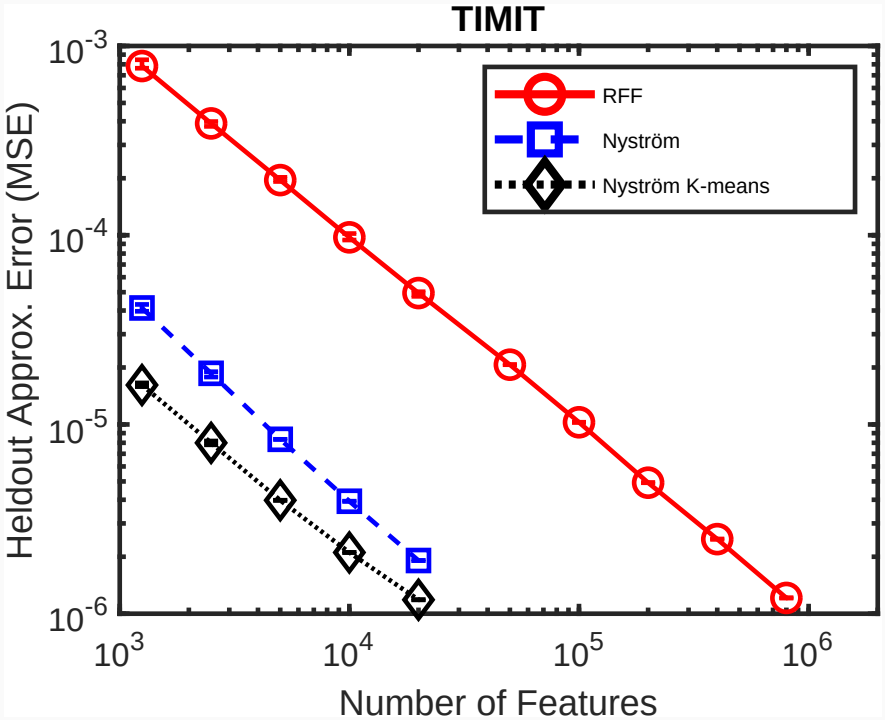
---

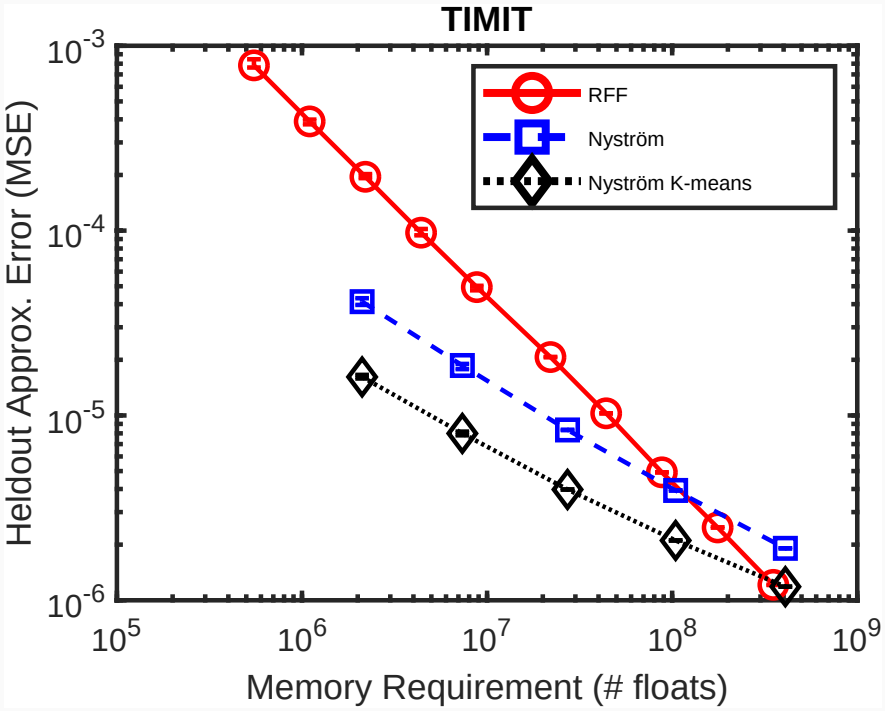
[1] Improved Nystrm low-rank approximation and error analysis. Kai Zhang, Ivor W. Tsang, James T. Kwok. ICML 2008.

Results: Kernel Approximation Error

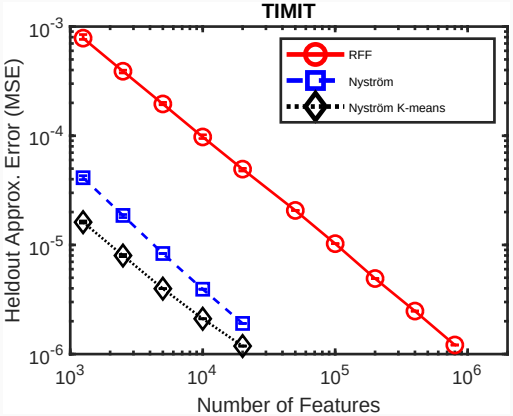


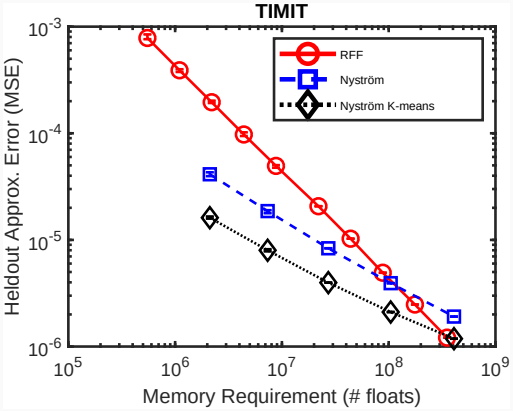
Results: Kernel Approximation Error



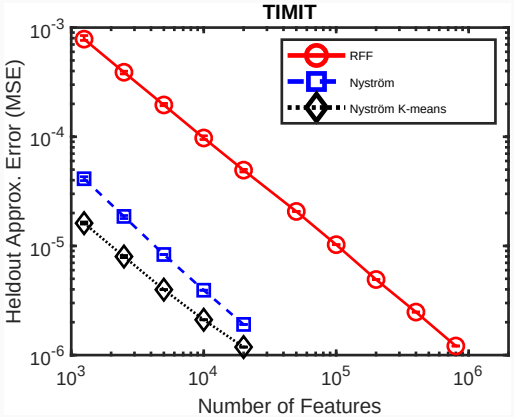


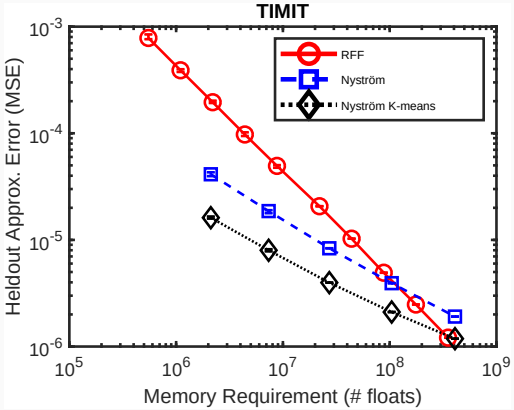
Results: Heldout performance

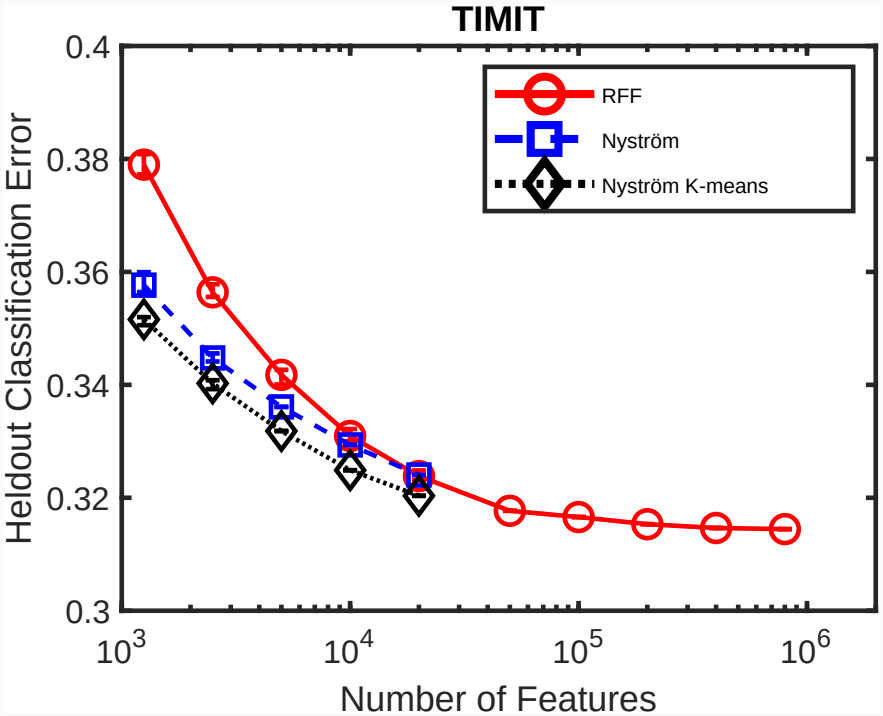


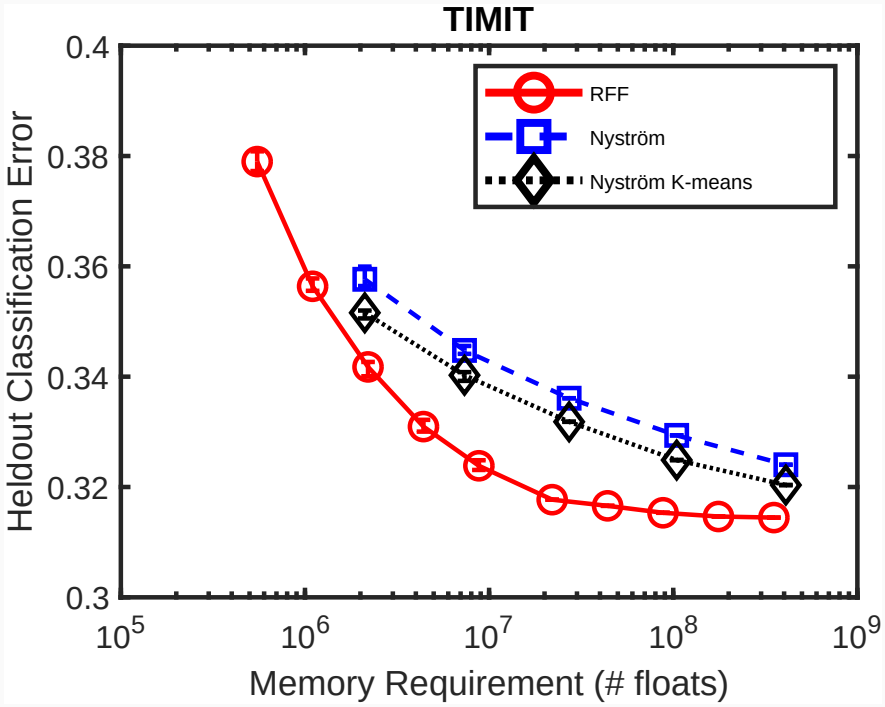


Results: Heldout performance





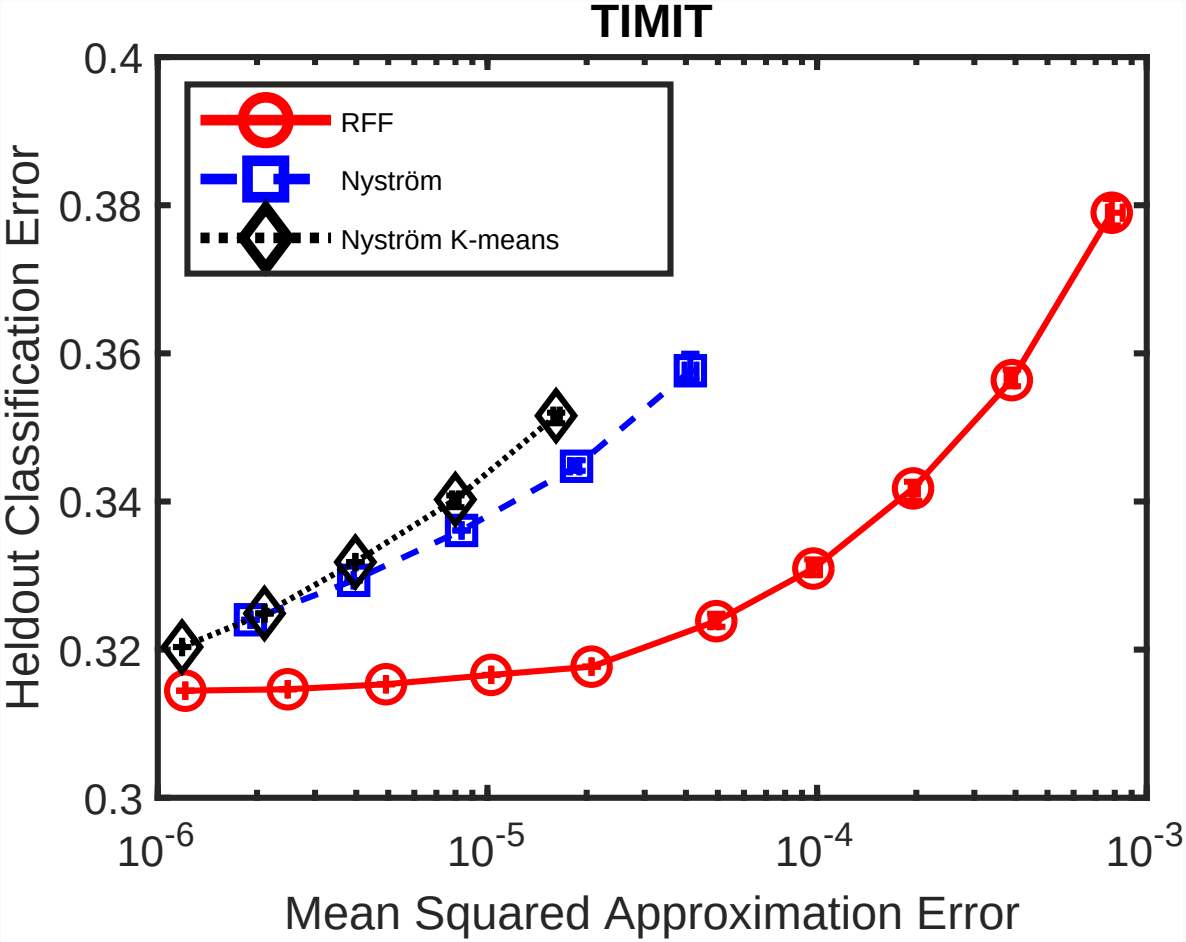




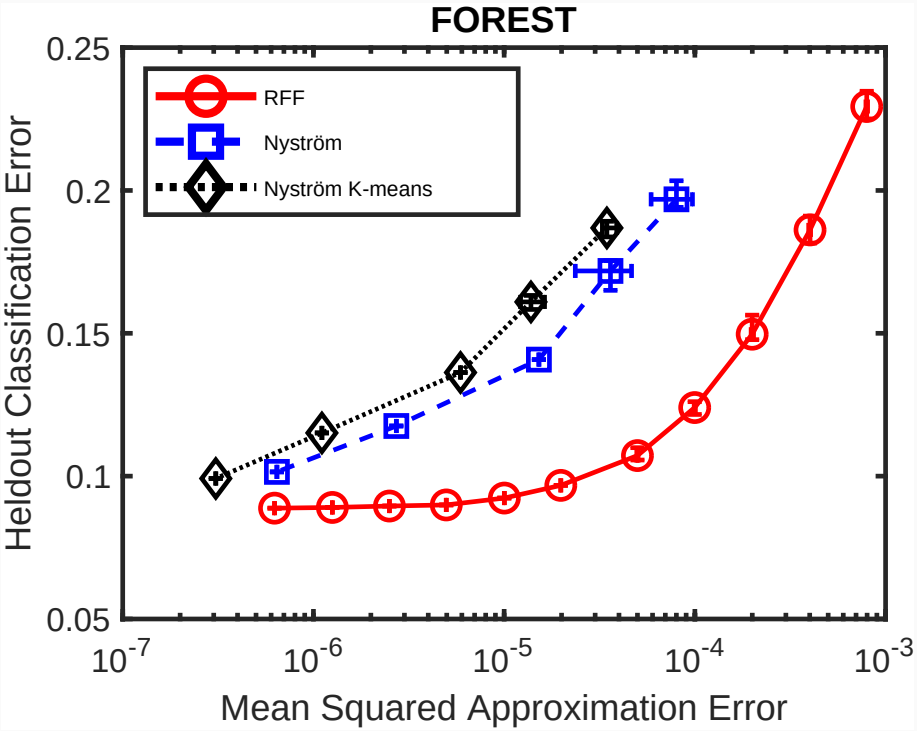
Results: Heldout performance

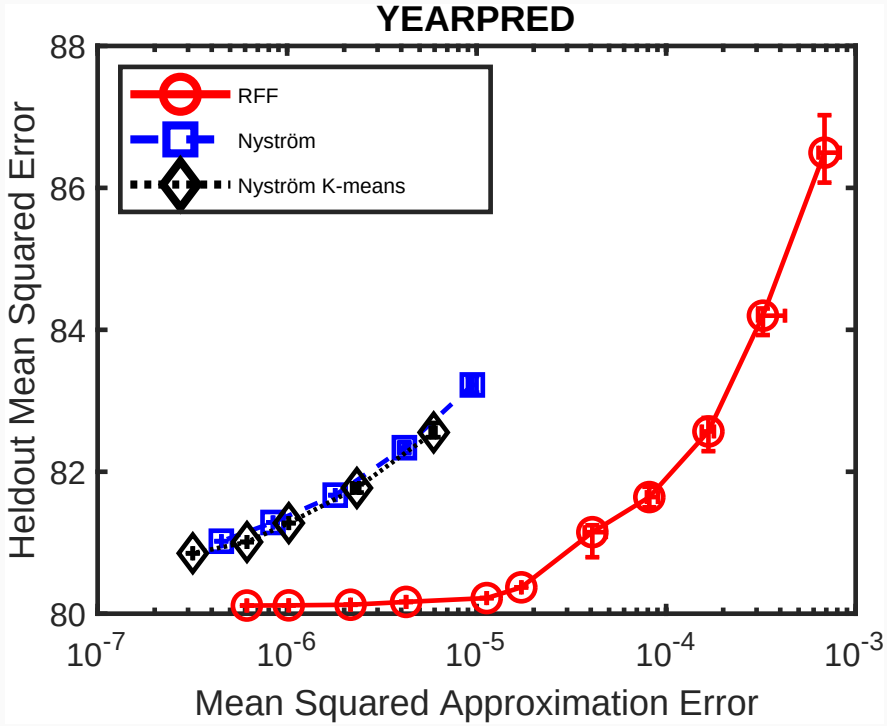
Something funky is going on...

Results: Heldout performance



Results: Other datasets



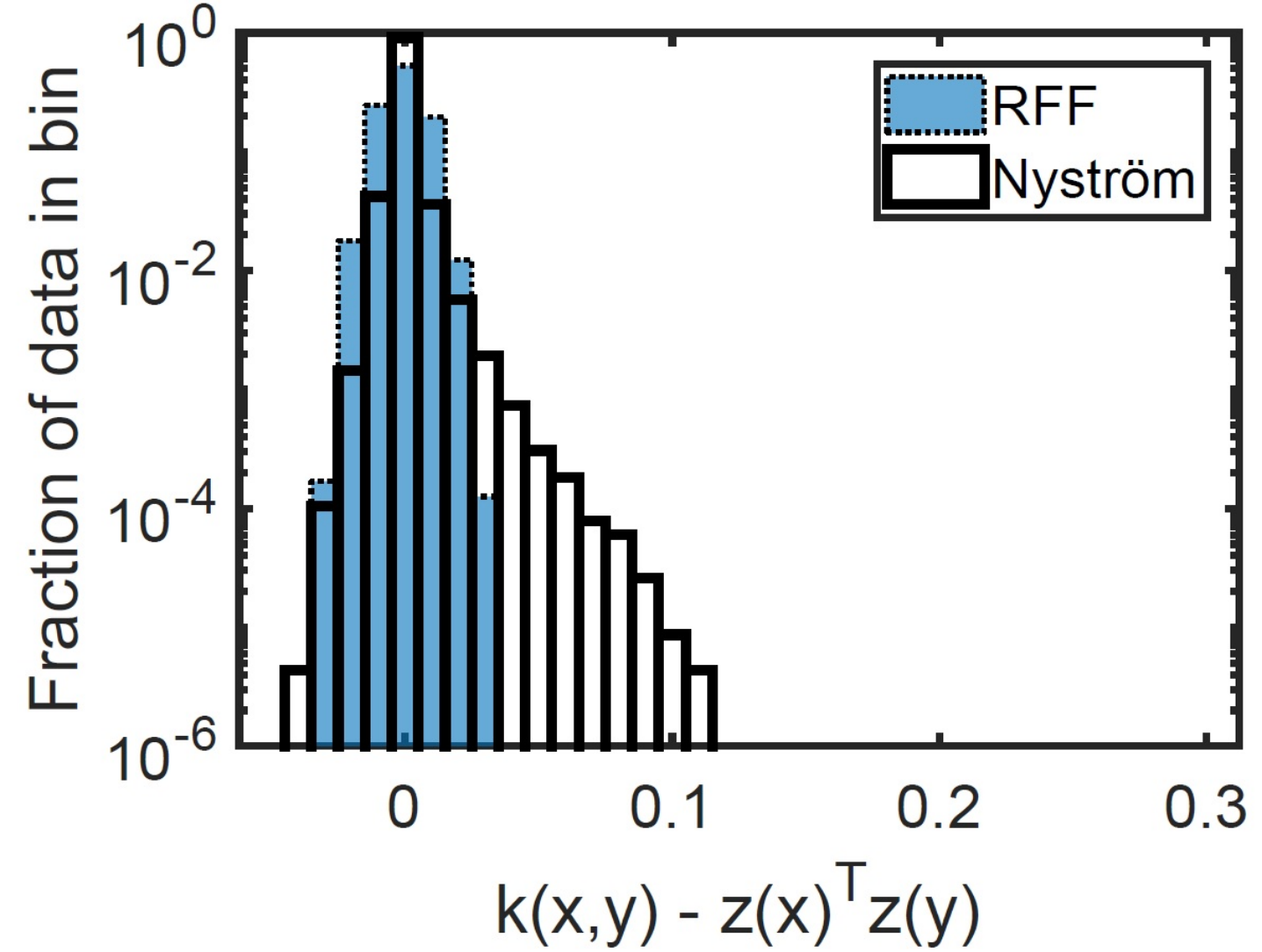


Kernel Approximation Error: A Second Look

Hypothesis: These differences in heldout classification performance are explained by differences in the way these methods make errors approximating the kernel.

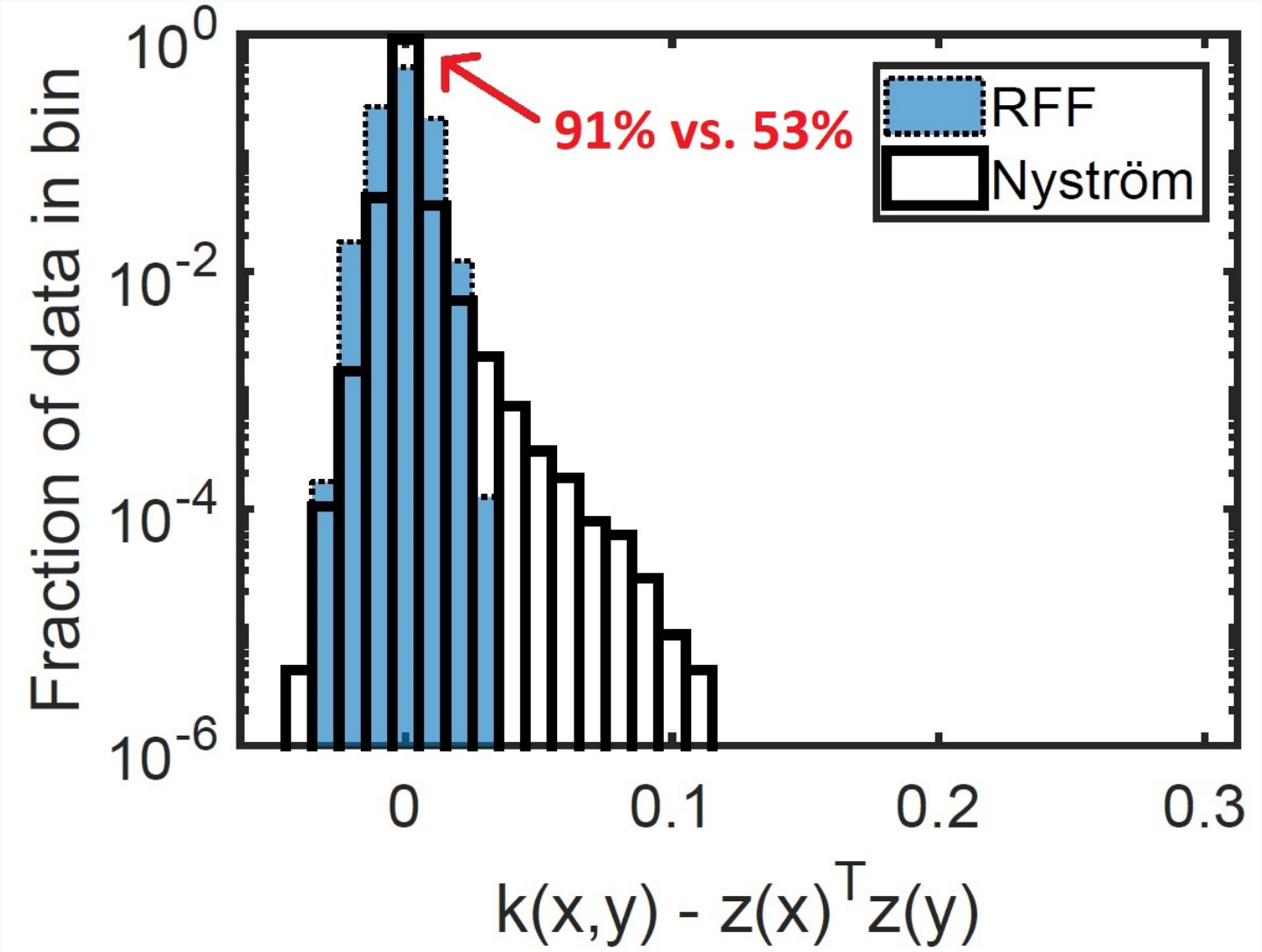
Kernel Approximation Error: A Second Look

$k(x, y) \leq 0.25$ (94.6% of data):



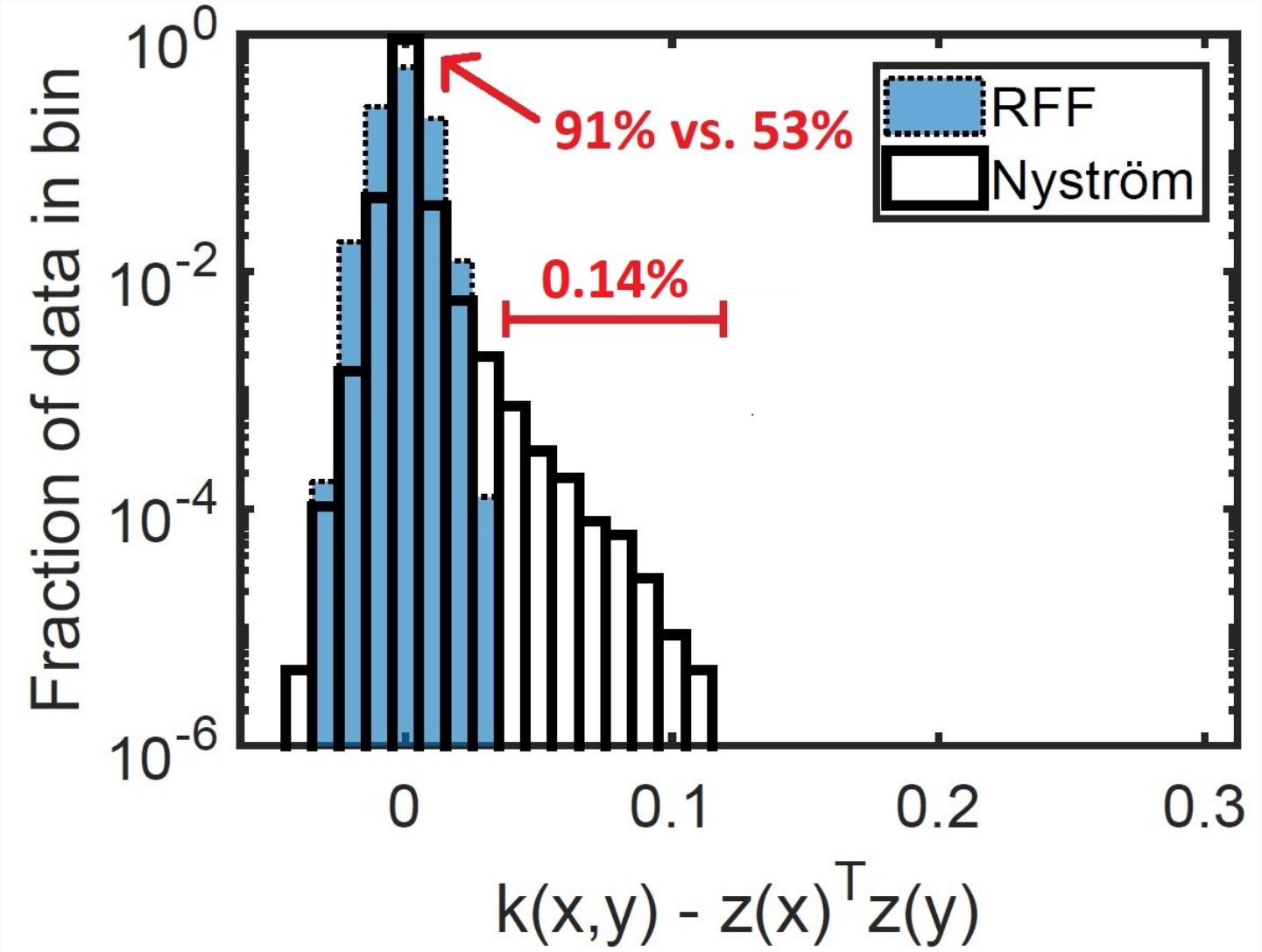
Kernel Approximation Error: A Second Look

$k(x, y) \leq 0.25$ (94.6% of data):



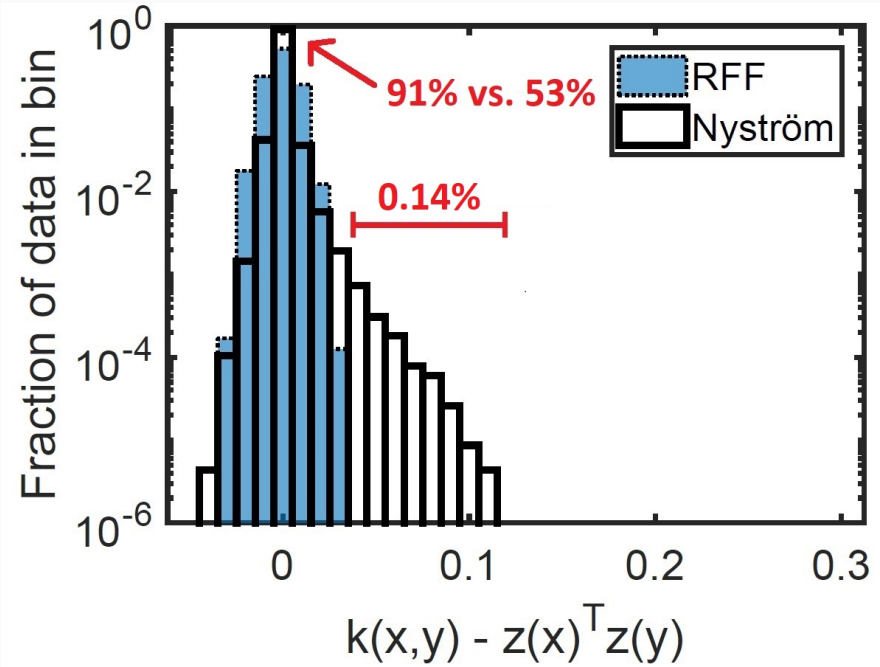
Kernel Approximation Error: A Second Look

$k(x, y) \leq 0.25$ (94.6% of data):

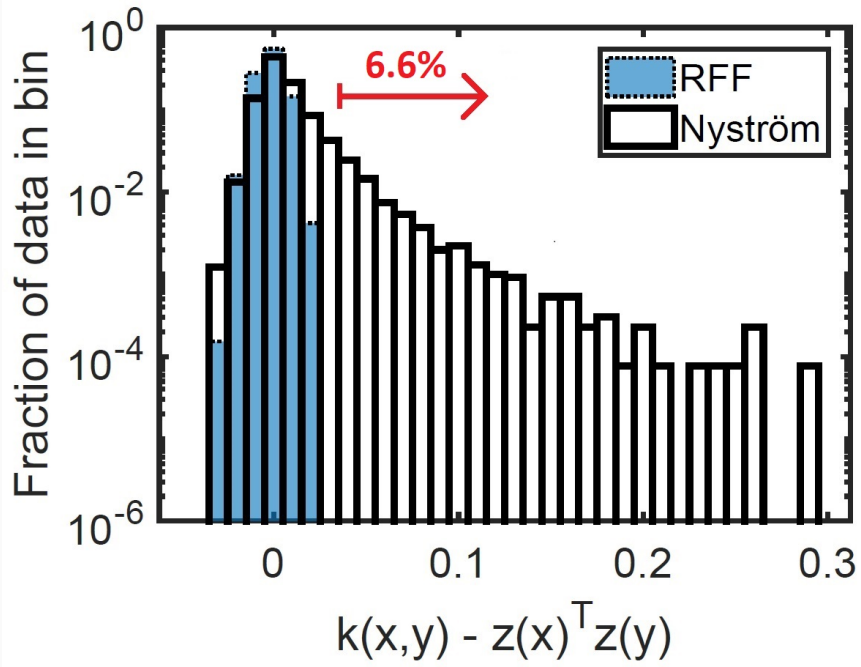


Kernel Approximation Error: A Second Look

$k(x, y) \leq 0.25$ (94.6%)



$k(x, y) \geq 0.25$ (5.4%)

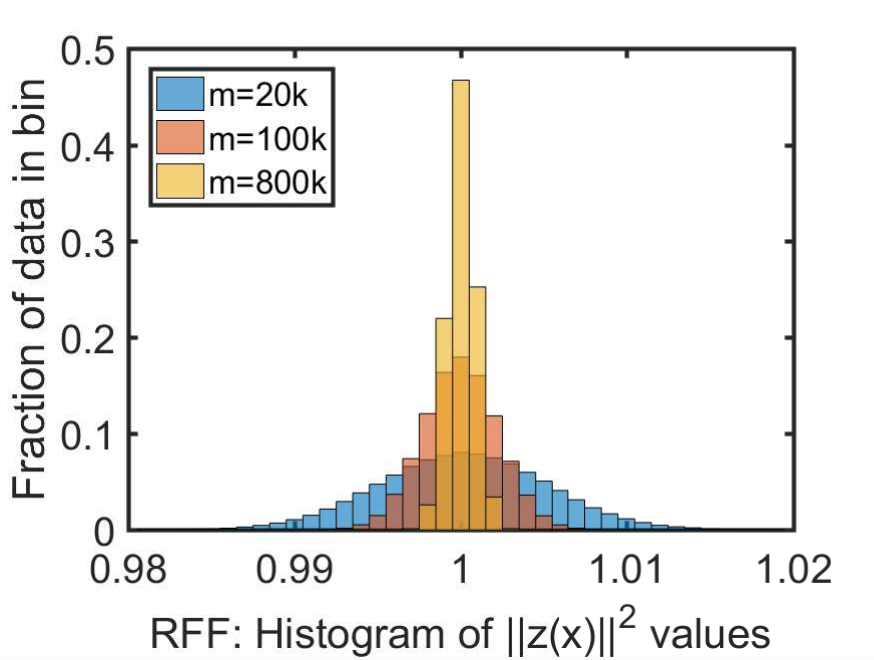


Kernel Approximation Error: A Second Look

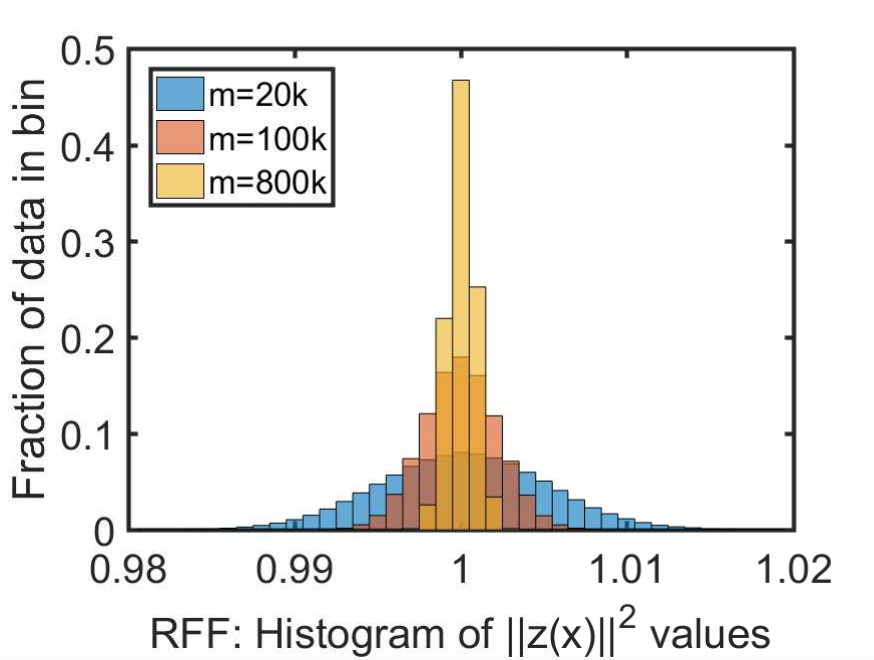
Now let's look at the error approximating $k(x, x)$.

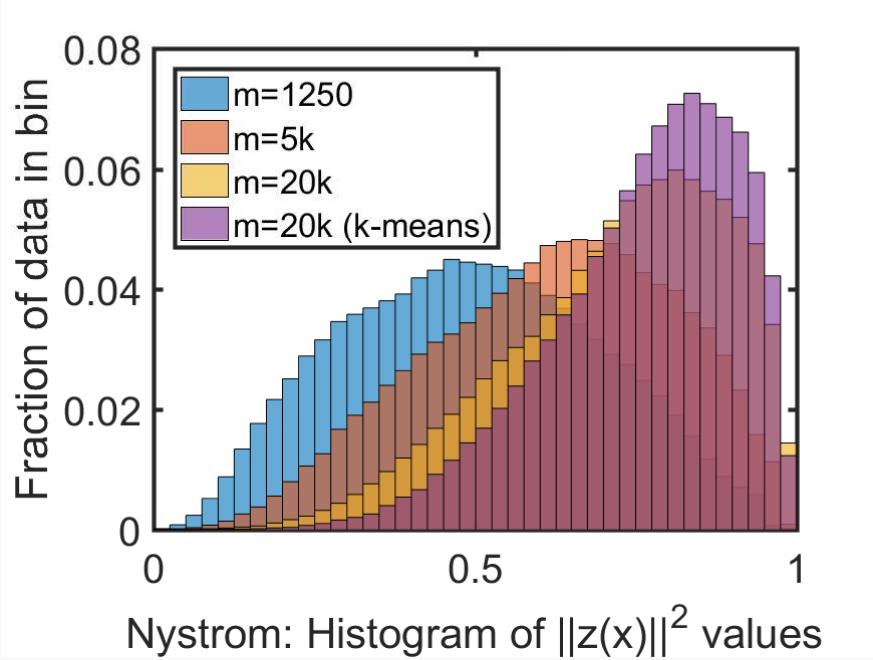
$$1 = k(x, x) \approx z(x)^T z(x) = \|z(x)\|^2$$

Kernel Approximation Error: A Second Look



Kernel Approximation Error: A Second Look





Kernel Approximation Error: Summary

- Nyström is generally more accurate than RFF, but has more outliers.

Kernel Approximation Error: Summary

- Nyström is generally more accurate than RFF, but has more outliers.
- These outliers are concentrated in pairs of points with high true kernel value $k(x, y)$, and generally correspond to *underestimating* the true kernel value.

Kernel Approximation Error: Summary

- Nyström is generally more accurate than RFF, but has more outliers.
- These outliers are concentrated in pairs of points with high true kernel value $k(x, y)$, and generally correspond to *underestimating* the true kernel value.
- Nyström generally underestimates $k(x, x)$ as well, often by a large margin.

Kernel Approximation Error: Summary

- Nyström is generally more accurate than RFF, but has more outliers.
- These outliers are concentrated in pairs of points with high true kernel value $k(x, y)$, and generally correspond to *underestimating* the true kernel value.
- Nyström generally underestimates $k(x, x)$ as well, often by a large margin.
- It appears these large and biased errors are costly in training models.

Nyström Error Analysis

Can we explain these empirical observations theoretically?

Nyström Error Analysis

Recall $\langle \varphi(x), \varphi(y) \rangle_{\mathcal{H}} = k(x, y)$.

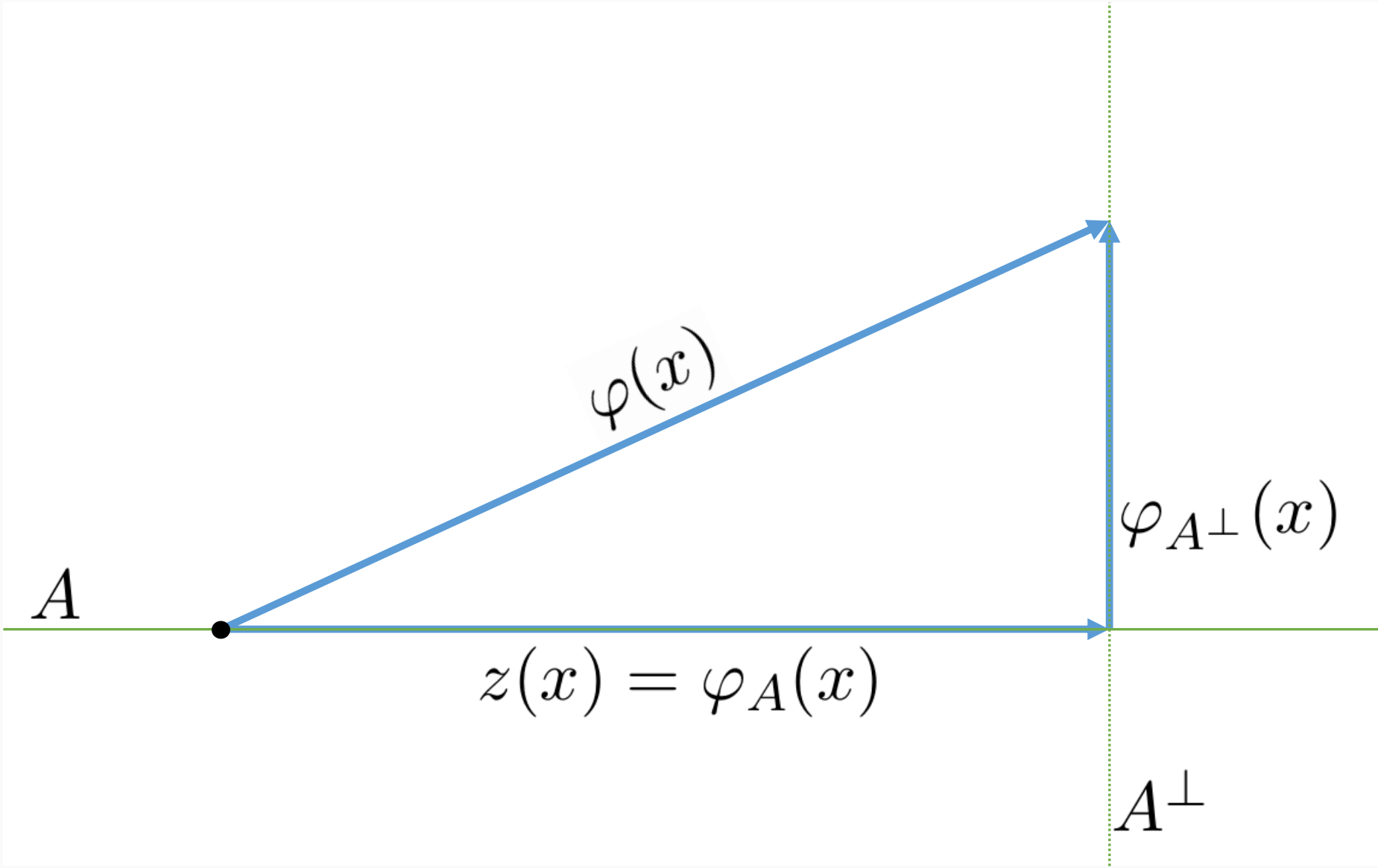
Nyström Error Analysis

Recall $\langle \varphi(x), \varphi(y) \rangle_{\mathcal{H}} = k(x, y)$.

Key Fact:

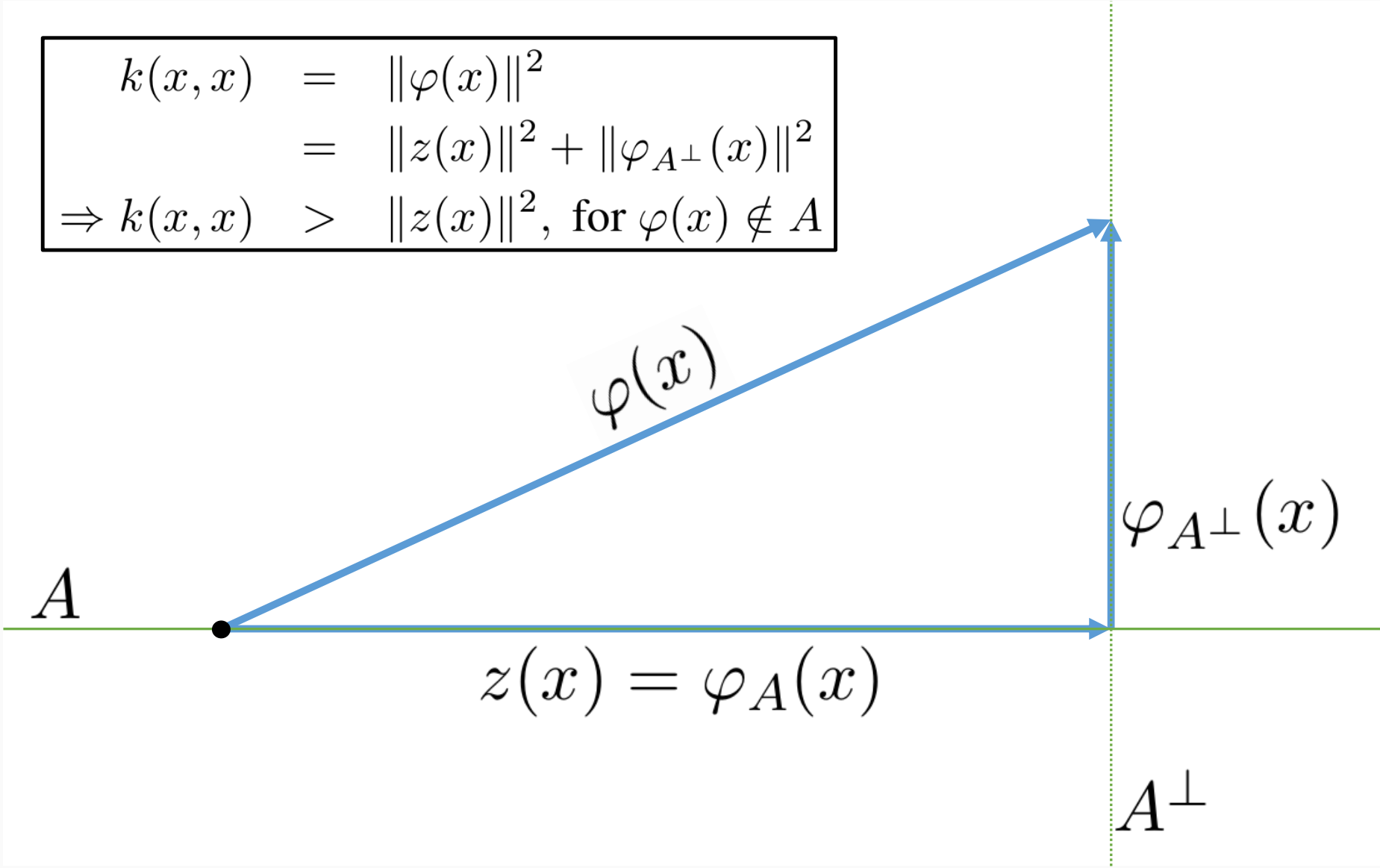
$z(x)$ is a projection of $\varphi(x)$ onto $A = span\Big(\varphi(\hat{x}_1), \ldots, \varphi(\hat{x}_m)\Big)$.

Nyström Error Analysis

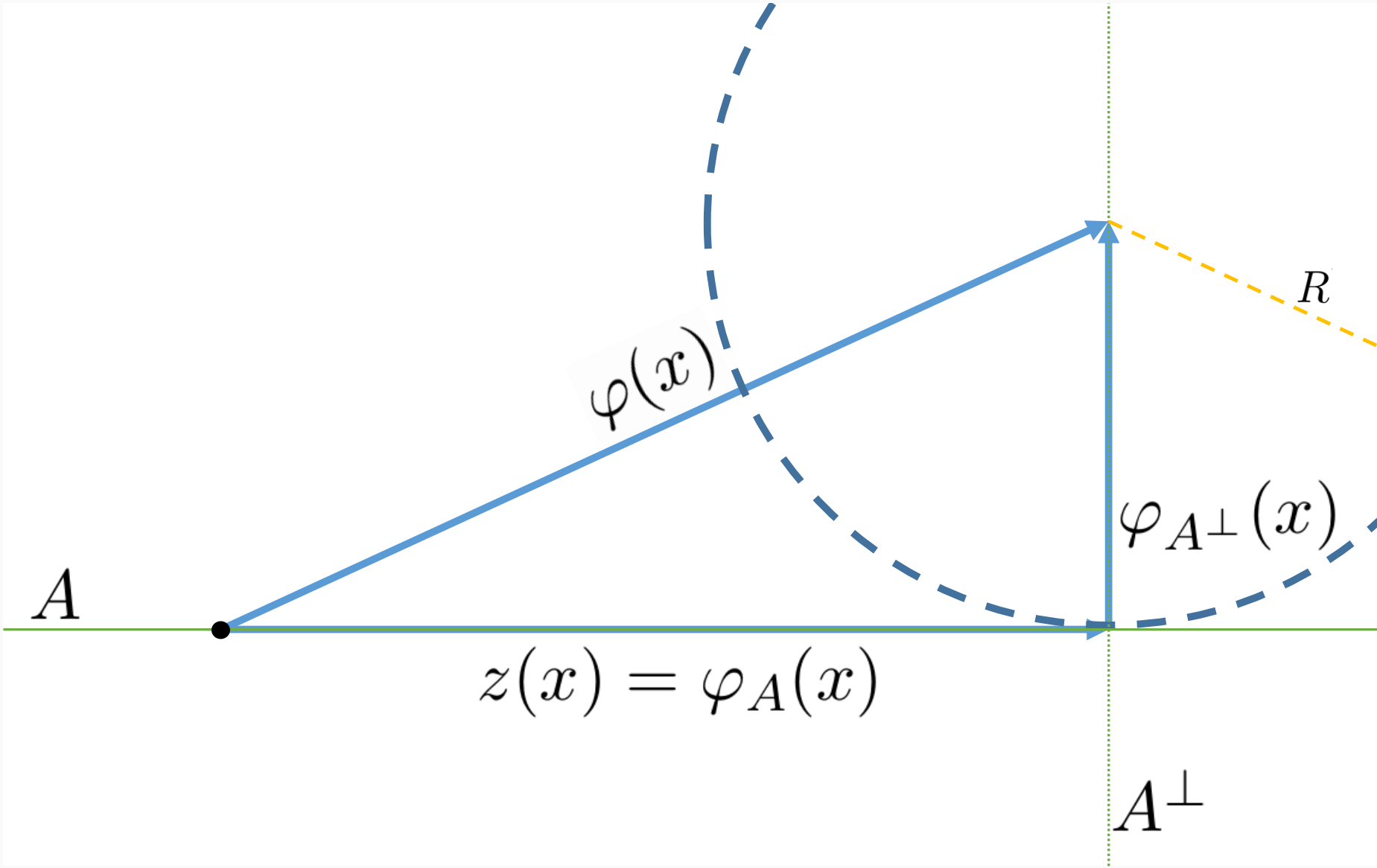


Nyström Error Analysis

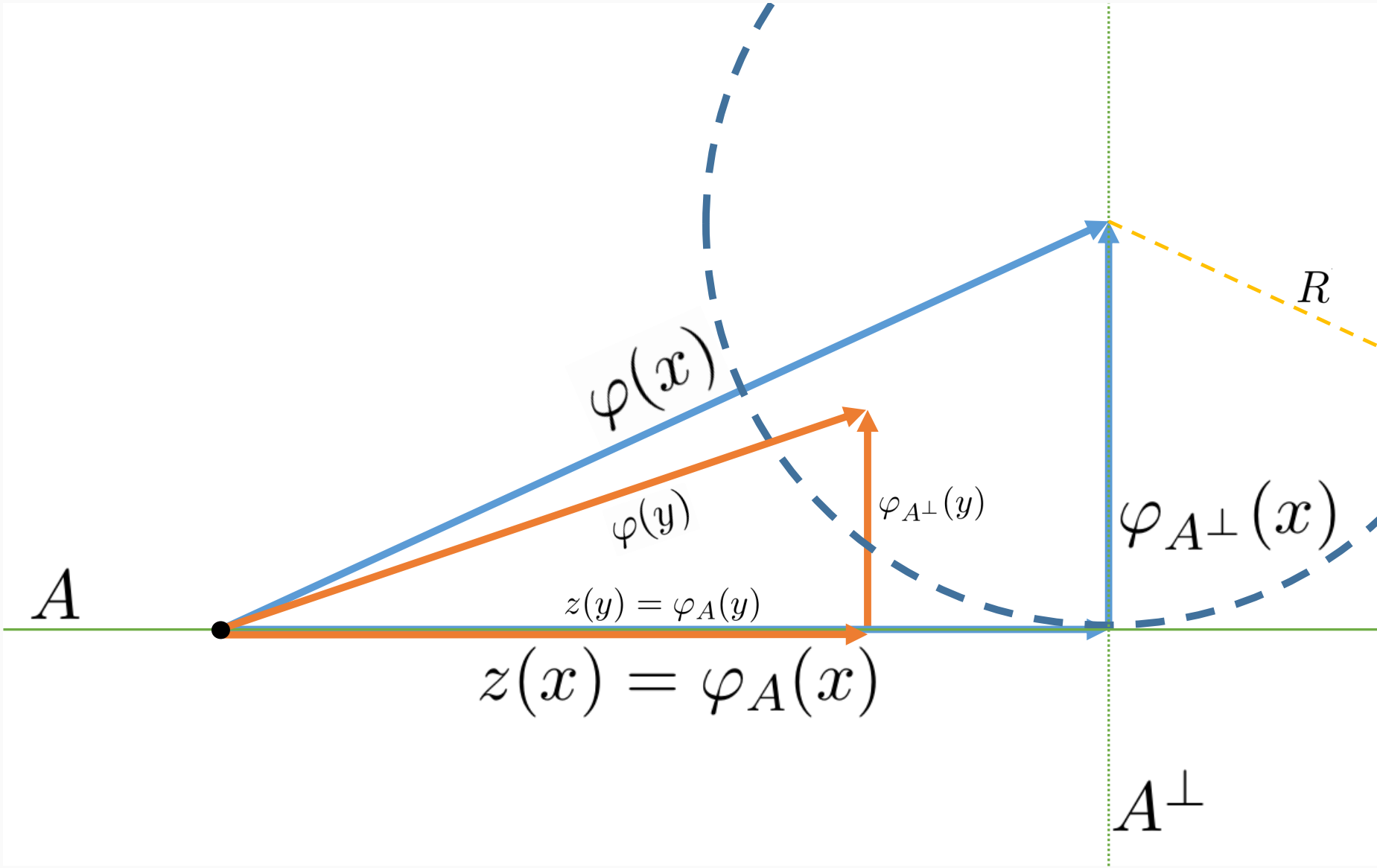
$$
\begin{aligned}
k(x,x) &= \|\varphi(x)\|^2 \\
&= \|z(x)\|^2 + \|\varphi_{A^\perp}(x)\|^2 \\
\Rightarrow k(x,x) &> \|z(x)\|^2, \text{ for } \varphi(x) \notin A
\end{aligned}
$$



Nyström Error Analysis

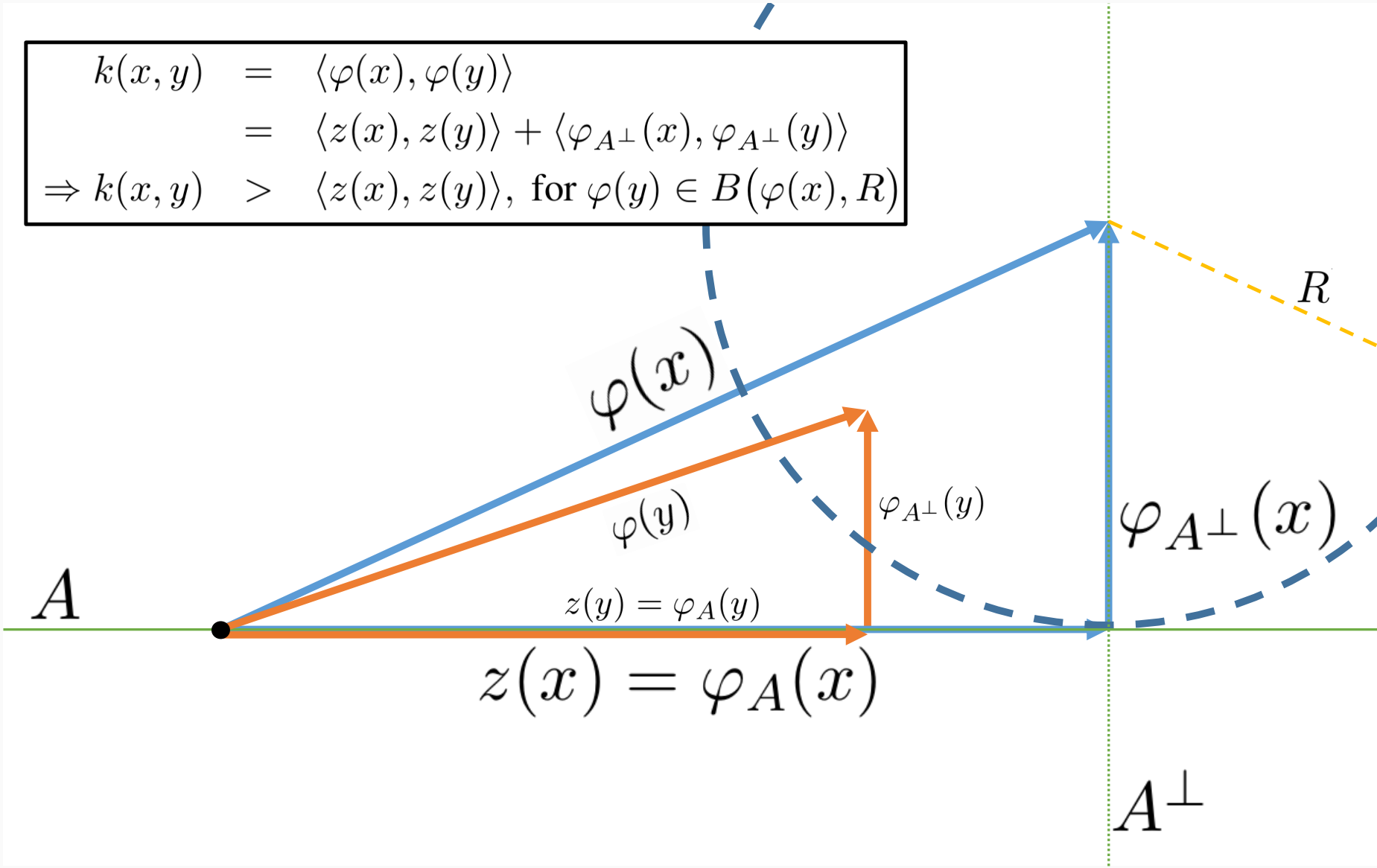


Nyström Error Analysis



Nyström Error Analysis

$$
\begin{aligned}
k(x,y) &= \langle \varphi(x), \varphi(y) \rangle \\
&= \langle z(x), z(y) \rangle + \langle \varphi_{A^\perp}(x), \varphi_{A^\perp}(y) \rangle \\
\Rightarrow k(x,y) &> \langle z(x), z(y) \rangle, \text{ for } \varphi(y) \in B\big(\varphi(x), R\big)
\end{aligned}
$$



Nyström Error Analysis

- Fact: $\varphi(x) \notin A \Rightarrow k(x,x) > \|z(x)\|^2$

Nyström Error Analysis

- Fact: $\varphi(x) \notin A \Rightarrow k(x,x) > \|z(x)\|^2$
- Theorem 1: Given a point $\varphi(x) \notin A$, $\forall \epsilon > 0$, $y \in \mathcal{X}$, $\|x - y\| < R(x,\epsilon) \Rightarrow k(x,y) - \langle z(x), z(y) \rangle > \epsilon$.

Nyström Error Analysis

- Fact: $\varphi(x) \notin A \Rightarrow k(x,x) > \|z(x)\|^2$
- Theorem 1: Given a point $\varphi(x) \notin A$, $\forall \epsilon > 0$, $y \in \mathcal{X}$, $\|x - y\| < R(x,\epsilon) \Rightarrow k(x,y) - \langle z(x), z(y) \rangle > \epsilon$.
- Theorems 2-3: Random x is likely to be far from A.

Nyström Error Analysis

- Fact: $\varphi(x) \notin A \Rightarrow k(x,x) > \|z(x)\|^2$
- Theorem 1: Given a point $\varphi(x) \notin A$, $\forall \epsilon > 0$, $y \in \mathcal{X}$, $\|x - y\| < R(x,\epsilon) \Rightarrow k(x,y) - \langle z(x), z(y) \rangle > \epsilon$.
- Theorems 2-3: Random x is likely to be far from A.
- Theorem 4: $\mathbb{E}_{X,Y}\left[k(X,Y) - \langle z(X), z(Y) \rangle\right] > 0$, assuming $\mathbb{E}_X\left[\varphi(X)\right] \notin A$.
  $\rightarrow$ Thus, Nyström is a biased estimator of $k(x,y)$.

Outline

I. Background
II. Random Fourier features (RFFs) for acoustic modeling
III. Improved RFFs through feature selection
IV. Nyström method vs. RFFs
V. **Concluding remarks**

Summary of Contributions

- Scaled kernel approximation methods to large-scale speech recognition problems, benchmarking performance on four datasets.
- Proposed feature selection algorithm to improve performance, along with a new kernel, matching performance with DNNs across all datasets.
- Compared RFFs to Nyström method, uncovering some important differences between these two methods, and demonstrating the superior performance of RFFs under a computation budget.

Future Work

Theoretically sound methods for large-scale machine learning.

- Next steps on Nyström vs. RFF work:
  - Combining Nyström and RFF features: best of both worlds?
  - Proving large Nyström errors are costly for performance.

Future Work

Theoretically sound methods for large-scale machine learning.

- Next steps on Nyström vs. RFF work:
  - Combining Nyström and RFF features: best of both worlds?
  - Proving large Nyström errors are costly for performance.
- Convolutional/recurrent kernels?

Future Work

Theoretically sound methods for large-scale machine learning.

- Next steps on Nyström vs. RFF work:
  - Combining Nyström and RFF features: best of both worlds?
  - Proving large Nyström errors are costly for performance.
- Convolutional/recurrent kernels?
- Scalable training: How to best leverage parallelization, and modern/future hardware?

Future Work

Theoretically sound methods for large-scale machine learning.

- Next steps on Nyström vs. RFF work:
  - Combining Nyström and RFF features: best of both worlds?
  - Proving large Nyström errors are costly for performance.
- Convolutional/recurrent kernels?
- Scalable training: How to best leverage parallelization, and modern/future hardware?
- Limitations of kernels, and how to overcome them.

Future Work

Theoretically sound methods for large-scale machine learning.

- Next steps on Nyström vs. RFF work:
  - Combining Nyström and RFF features: best of both worlds?
  - Proving large Nyström errors are costly for performance.
- Convolutional/recurrent kernels?
- Scalable training: How to best leverage parallelization, and modern/future hardware?
- Limitations of kernels, and how to overcome them.
- Alternative methods to deep learning, other than kernels?

Future Work

Theoretically sound methods for large-scale machine learning.

- Next steps on Nyström vs. RFF work:
  - Combining Nyström and RFF features: best of both worlds?
  - Proving large Nyström errors are costly for performance.
- Convolutional/recurrent kernels?
- Scalable training: How to best leverage parallelization, and modern/future hardware?
- Limitations of kernels, and how to overcome them.
- Alternative methods to deep learning, other than kernels?
- Better *theoretical* understanding of deep learning.

Thank You!

Questions??