Kernels, Random Embeddings and Deep Learning

Vikas Sindhwani
IBM Research, NY

October 28, 2014

Acknowledgements

- At IBM: Haim Avron, Tara Sainath, B. Ramabhadran, Q. Fan
- Summer Interns: Jiyan Yang (Stanford), Po-sen Huang (UIUC)
- Michael Mahoney (UC Berkeley), Ha Quang Minh (IIT Genova)
- IBM DARPA XDATA project led by Ken Clarkson (IBM Almaden)

Setting

- Given labeled data in the form of input-output pairs,

$$\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n, \quad \mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^d, \quad \mathbf{y}_i \in \mathcal{Y} \subset \mathbb{R}^m,$$

  estimate the unknown dependency $f : \mathcal{X} \mapsto \mathcal{Y}$.

Setting

- Given labeled data in the form of input-output pairs,

$$\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n, \quad \mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^d, \quad \mathbf{y}_i \in \mathcal{Y} \subset \mathbb{R}^m,$$

  estimate the unknown dependency $f : \mathcal{X} \mapsto \mathcal{Y}$.
- Regularized Risk Minimization in a suitable hypothesis space $\mathcal{H}$,

$$\arg\min_{f \in \mathcal{H}} \sum_{i=1}^n V(f(\mathbf{x}_i), \mathbf{y}_i) + \Omega(f)$$

Setting

- Given labeled data in the form of input-output pairs,

$$\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n, \quad \mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^d, \quad \mathbf{y}_i \in \mathcal{Y} \subset \mathbb{R}^m,$$

  estimate the unknown dependency $f : \mathcal{X} \mapsto \mathcal{Y}$.
- Regularized Risk Minimization in a suitable hypothesis space $\mathcal{H}$,

$$\arg\min_{f \in \mathcal{H}} \sum_{i=1}^n V(f(\mathbf{x}_i), \mathbf{y}_i) + \Omega(f)$$

- Large $n \implies$ Big models: $\mathcal{H}$ "rich" /non-parametric/nonlinear.

Setting

- Given labeled data in the form of input-output pairs,

$$\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n, \quad \mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^d, \quad \mathbf{y}_i \in \mathcal{Y} \subset \mathbb{R}^m,$$

  estimate the unknown dependency $f : \mathcal{X} \mapsto \mathcal{Y}$.
- Regularized Risk Minimization in a suitable hypothesis space $\mathcal{H}$,

$$\arg\min_{f \in \mathcal{H}} \sum_{i=1}^n V(f(\mathbf{x}_i), \mathbf{y}_i) + \Omega(f)$$

- Large $n \implies$ Big models: $\mathcal{H}$ "rich" /non-parametric/nonlinear.
- Two great ML traditions around choosing $\mathcal{H}$:

Setting

- Given labeled data in the form of input-output pairs,

$$\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n, \quad \mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^d, \quad \mathbf{y}_i \in \mathcal{Y} \subset \mathbb{R}^m,$$

  estimate the unknown dependency $f : \mathcal{X} \mapsto \mathcal{Y}$.
- Regularized Risk Minimization in a suitable hypothesis space $\mathcal{H}$,

$$\arg\min_{f \in \mathcal{H}} \sum_{i=1}^n V(f(\mathbf{x}_i), \mathbf{y}_i) + \Omega(f)$$

- Large $n \implies$ Big models: $\mathcal{H}$ "rich" /non-parametric/nonlinear.
- Two great ML traditions around choosing $\mathcal{H}$:
  - Deep Neural Networks: $f(\mathbf{x}) = s_n(\ldots s_2(\mathbf{W}_2 s_1(\mathbf{W}_1 \mathbf{x}))\ldots)$

Setting

- Given labeled data in the form of input-output pairs,

$$\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n, \quad \mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^d, \quad \mathbf{y}_i \in \mathcal{Y} \subset \mathbb{R}^m,$$

  estimate the unknown dependency $f : \mathcal{X} \mapsto \mathcal{Y}$.
- Regularized Risk Minimization in a suitable hypothesis space $\mathcal{H}$,

$$\arg\min_{f \in \mathcal{H}} \sum_{i=1}^n V(f(\mathbf{x}_i), \mathbf{y}_i) + \Omega(f)$$

- Large $n \implies$ Big models: $\mathcal{H}$ "rich" /non-parametric/nonlinear.
- Two great ML traditions around choosing $\mathcal{H}$:
  - Deep Neural Networks: $f(\mathbf{x}) = s_n(\ldots s_2(\mathbf{W}_2 s_1(\mathbf{W}_1 \mathbf{x}))\ldots)$
  - Kernel Methods: general nonlinear function space generated by a kernel function $k(\mathbf{x}, \mathbf{z})$ on $\mathcal{X} \times \mathcal{X}$.
- This talk: Thrust towards scalable kernel methods, motivated by the recent successes of deep learning.

Outline

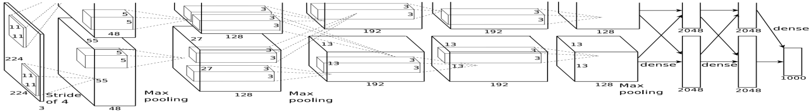
Motivation and Background

Scalable Kernel Methods
- Random Embeddings+Distributed Computation (ICASSP, JSM 2014)
- libSkylark: An open source software stack
- Quasi-Monte Carlo Embeddings (ICML 2014)

Synergies?

Deep Learning is “Supercharging” Machine Learning

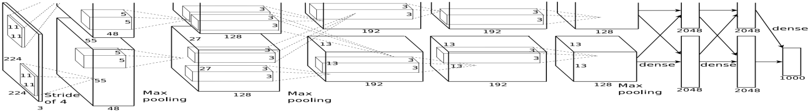
Krizhevsky et.al. won the 2012 ImageNet challenge (ILSVRC-2012) with top-5 error rate of 15.3% compared to 26.2% of the second best entry.



- Many statistical and computational ingredients:
  - Large datasets (ILSVRC since 2010)

Deep Learning is "Supercharging" Machine Learning

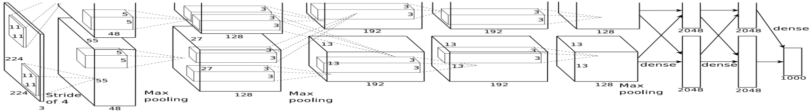
Krizhevsky et.al. won the 2012 ImageNet challenge (ILSVRC-2012) with top-5 error rate of 15.3% compared to 26.2% of the second best entry.



- Many statistical and computational ingredients:
  - Large datasets (ILSVRC since 2010)
  - Large statistical capacity ($1.2M$ images, $60M$ params)

Deep Learning is "Supercharging" Machine Learning

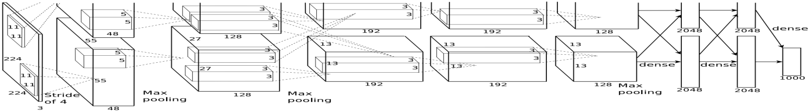
Krizhevsky et.al. won the 2012 ImageNet challenge (ILSVRC-2012) with top-5 error rate of 15.3% compared to 26.2% of the second best entry.



- Many statistical and computational ingredients:
  - Large datasets (ILSVRC since 2010)
  - Large statistical capacity ($1.2M$ images, $60M$ params)
  - Distributed computation

Deep Learning is “Supercharging” Machine Learning

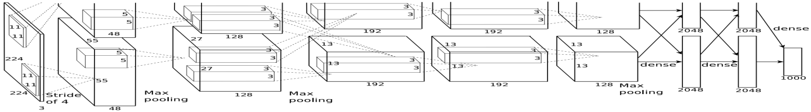
Krizhevsky et.al. won the 2012 ImageNet challenge (ILSVRC-2012) with top-5 error rate of 15.3% compared to 26.2% of the second best entry.



- Many statistical and computational ingredients:
  - Large datasets (ILSVRC since 2010)
  - Large statistical capacity ($1.2M$ images, $60M$ params)
  - Distributed computation
  - Depth, Invariant feature learning (transferrable to other tasks)

Deep Learning is "Supercharging" Machine Learning

Krizhevsky et.al. won the 2012 ImageNet challenge (ILSVRC-2012) with top-5 error rate of 15.3% compared to 26.2% of the second best entry.



- Many statistical and computational ingredients:
  - Large datasets (ILSVRC since 2010)
  - Large statistical capacity ($1.2M$ images, $60M$ params)
  - Distributed computation
  - Depth, Invariant feature learning (transferrable to other tasks)
  - Engineering: Dropout, ReLU ...
- Very active area in Speech and Natural Language Processing.

Machine Learning in 1990s

- Convolutional Neural Networks (Fukushima 1980; Lecun et al 1989)

Machine Learning in 1990s

- Convolutional Neural Networks (Fukushima 1980; Lecun et al 1989)
  - 3 days to train on USPS ($n = 7291$; digit recognition) on Sun Sparcstation 1 (33MHz clock speed, 64MB RAM)

Machine Learning in 1990s

- Convolutional Neural Networks (Fukushima 1980; Lecun et al 1989)
  - 3 days to train on USPS ($n = 7291$; digit recognition) on Sun Sparcstation 1 (33MHz clock speed, 64MB RAM)
- Personal history:

Machine Learning in 1990s

- Convolutional Neural Networks (Fukushima 1980; Lecun et al 1989)
  - 3 days to train on USPS ($n = 7291$; digit recognition) on Sun Sparcstation 1 (33MHz clock speed, 64MB RAM)
- Personal history:
  - 1998: First ML experiment - train DNN on UCI Wine dataset.

Machine Learning in 1990s

- Convolutional Neural Networks (Fukushima 1980; Lecun et al 1989)
  - 3 days to train on USPS ($n = 7291$; digit recognition) on Sun Sparcstation 1 (33MHz clock speed, 64MB RAM)
- Personal history:
  - 1998: First ML experiment - train DNN on UCI Wine dataset.
  - 1999: Introduced to Kernel Methods - by DNN researchers!

Machine Learning in 1990s

- Convolutional Neural Networks (Fukushima 1980; Lecun et al 1989)
  - 3 days to train on USPS ($n = 7291$; digit recognition) on Sun Sparcstation 1 (33MHz clock speed, 64MB RAM)
- Personal history:
  - 1998: First ML experiment - train DNN on UCI Wine dataset.
  - 1999: Introduced to Kernel Methods - by DNN researchers!
  - 2003-4: NN paper at JMLR rejected; accepted in IEEE Trans. Neural Nets with a kernel methods section!

Machine Learning in 1990s

- Convolutional Neural Networks (Fukushima 1980; Lecun et al 1989)
  - 3 days to train on USPS ($n = 7291$; digit recognition) on Sun Sparcstation 1 (33MHz clock speed, 64MB RAM)
- Personal history:
  - 1998: First ML experiment - train DNN on UCI Wine dataset.
  - 1999: Introduced to Kernel Methods - by DNN researchers!
  - 2003-4: NN paper at JMLR rejected; accepted in IEEE Trans. Neural Nets with a kernel methods section!
- Why Kernel Methods?

Machine Learning in 1990s

- Convolutional Neural Networks (Fukushima 1980; Lecun et al 1989)
  - 3 days to train on USPS ($n = 7291$; digit recognition) on Sun Sparcstation 1 (33MHz clock speed, 64MB RAM)
- Personal history:
  - 1998: First ML experiment - train DNN on UCI Wine dataset.
  - 1999: Introduced to Kernel Methods - by DNN researchers!
  - 2003-4: NN paper at JMLR rejected; accepted in IEEE Trans. Neural Nets with a kernel methods section!
- Why Kernel Methods?
  - Local Minima free - stronger role of Convex Optimization.

Machine Learning in 1990s

- Convolutional Neural Networks (Fukushima 1980; Lecun et al 1989)
  - 3 days to train on USPS ($n = 7291$; digit recognition) on Sun Sparcstation 1 (33MHz clock speed, 64MB RAM)
- Personal history:
  - 1998: First ML experiment - train DNN on UCI Wine dataset.
  - 1999: Introduced to Kernel Methods - by DNN researchers!
  - 2003-4: NN paper at JMLR rejected; accepted in IEEE Trans. Neural Nets with a kernel methods section!
- Why Kernel Methods?
  - Local Minima free - stronger role of Convex Optimization.
  - Theoretically appealing

Machine Learning in 1990s

- Convolutional Neural Networks (Fukushima 1980; Lecun et al 1989)
  - 3 days to train on USPS ($n = 7291$; digit recognition) on Sun Sparcstation 1 (33MHz clock speed, 64MB RAM)
- Personal history:
  - 1998: First ML experiment - train DNN on UCI Wine dataset.
  - 1999: Introduced to Kernel Methods - by DNN researchers!
  - 2003-4: NN paper at JMLR rejected; accepted in IEEE Trans. Neural Nets with a kernel methods section!
- Why Kernel Methods?
  - Local Minima free - stronger role of Convex Optimization.
  - Theoretically appealing
  - Handle non-vectorial data; high-dimensional data
  - Easier model selection via continuous optimization.

Machine Learning in 1990s

- Convolutional Neural Networks (Fukushima 1980; Lecun et al 1989)
  - 3 days to train on USPS ($n = 7291$; digit recognition) on Sun Sparcstation 1 (33MHz clock speed, 64MB RAM)
- Personal history:
  - 1998: First ML experiment - train DNN on UCI Wine dataset.
  - 1999: Introduced to Kernel Methods - by DNN researchers!
  - 2003-4: NN paper at JMLR rejected; accepted in IEEE Trans. Neural Nets with a kernel methods section!
- Why Kernel Methods?
  - Local Minima free - stronger role of Convex Optimization.
  - Theoretically appealing
  - Handle non-vectorial data; high-dimensional data
  - Easier model selection via continuous optimization.
  - Matched NN in many cases, although didnt scale wrt n as well.
- So what changed?

Machine Learning in 1990s

- Convolutional Neural Networks (Fukushima 1980; Lecun et al 1989)
  - 3 days to train on USPS ($n = 7291$; digit recognition) on Sun Sparcstation 1 (33MHz clock speed, 64MB RAM)
- Personal history:
  - 1998: First ML experiment - train DNN on UCI Wine dataset.
  - 1999: Introduced to Kernel Methods - by DNN researchers!
  - 2003-4: NN paper at JMLR rejected; accepted in IEEE Trans. Neural Nets with a kernel methods section!
- Why Kernel Methods?
  - Local Minima free - stronger role of Convex Optimization.
  - Theoretically appealing
  - Handle non-vectorial data; high-dimensional data
  - Easier model selection via continuous optimization.
  - Matched NN in many cases, although didnt scale wrt n as well.
- So what changed?
  - More data, parallel algorithms, hardware? Better DNN training? ...

Kernel Methods and Neural Networks (Pre-Google)

1. Jackel bets (one fancy dinner) that by March 14, 2000, people will understand
quantitatively why big neural nets working on large databases are not so bad.
(Understanding means that there will be clear conditions and bounds)

Vapnik bets (one fancy dinner) that Jackel is wrong.

But .. If Vapnik figures out the bounds and conditions, Vapnik still wins the bet.

****

2. Vapnik bets (one fancy dinner) that by March 14, 2005, no one in his right mind will use neural nets that are essentially like those used in 1995.

Jackel bets (one fancy dinner) that Vapnik is wrong

____________________ 3/14/95
V. Vapnik

____________________ 3/14//95
L. Jackel

____________________ 3/14/95
Witnessed by Y. LeCun

Kernel Methods and Neural Networks

Geoff Hinton facts meme maintained at
`http://yann.lecun.com/ex/fun/`

Kernel Methods and Neural Networks

Geoff Hinton facts meme maintained at
`http://yann.lecun.com/ex/fun/`

- All **kernels** that ever dared approaching Geoff Hinton woke up convolved.
- The only **kernel** Geoff Hinton has ever used is a **kernel** of truth.
- If you defy Geoff Hinton, he will maximize your entropy in no time. Your free energy will be gone even before you reach equilibrium.

Kernel Methods and Neural Networks

Geoff Hinton facts meme maintained at
`http://yann.lecun.com/ex/fun/`

- All **kernels** that ever dared approaching Geoff Hinton woke up convolved.
- The only **kernel** Geoff Hinton has ever used is a **kernel** of truth.
- If you defy Geoff Hinton, he will maximize your entropy in no time. Your free energy will be gone even before you reach equilibrium.

Are there synergies between these fields towards design of even better (faster and more accurate) algorithms?

The Mathematical Naturalness of Kernel Methods

- Data $\mathcal{X} \in \mathbb{R}^d$, Models $\in \mathcal{H} : \mathcal{X} \mapsto \mathbb{R}$

The Mathematical Naturalness of Kernel Methods

- Data $\mathcal{X} \in \mathbb{R}^d$, Models $\in \mathcal{H} : \mathcal{X} \mapsto \mathbb{R}$
- Geometry in $\mathcal{H}$: inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$, norm $\| \cdot \|_{\mathcal{H}}$ (Hilbert Spaces)

The Mathematical Naturalness of Kernel Methods

- Data $\mathcal{X} \in \mathbb{R}^d$, Models $\in \mathcal{H} : \mathcal{X} \mapsto \mathbb{R}$
- Geometry in $\mathcal{H}$: inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$, norm $\| \cdot \|_{\mathcal{H}}$ (Hilbert Spaces)
- **Theorem** *All* nice Hilbert spaces are generated by a symmetric positive definite function (the kernel) $k(\mathbf{x}, \mathbf{x}')$ on $\mathcal{X} \times \mathcal{X}$
  - if $f, g \in \mathcal{H}$ close i.e. $\|f - g\|_{\mathcal{H}}$ small, then $f(\mathbf{x}), g(\mathbf{x})$ close $\forall \mathbf{x} \in \mathcal{X}$.
  - Reproducing Kernel Hilbert Spaces (RKHSs)

The Mathematical Naturalness of Kernel Methods

- Data $\mathcal{X} \in \mathbb{R}^d$, Models $\in \mathcal{H} : \mathcal{X} \mapsto \mathbb{R}$
- Geometry in $\mathcal{H}$: inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$, norm $\| \cdot \|_{\mathcal{H}}$ (Hilbert Spaces)
- **Theorem** *All* nice Hilbert spaces are generated by a symmetric positive definite function (the kernel) $k(\mathbf{x}, \mathbf{x}')$ on $\mathcal{X} \times \mathcal{X}$
  - if $f, g \in \mathcal{H}$ close i.e. $\|f - g\|_{\mathcal{H}}$ small, then $f(\mathbf{x}), g(\mathbf{x})$ close $\forall \mathbf{x} \in \mathcal{X}$.
  - Reproducing Kernel Hilbert Spaces (RKHSs)
- Functional Analysis (Aronszajn, Bergman (1950s)); Statistics (Parzen (1960s)); PDEs; Numerical Analysis...
  - ML: Nonlinear classification, regression, clustering, time-series analysis, dynamical systems, hypothesis testing, causal modeling, ...

The Mathematical Naturalness of Kernel Methods

- Data $\mathcal{X} \in \mathbb{R}^d$, Models $\in \mathcal{H} : \mathcal{X} \mapsto \mathbb{R}$
- Geometry in $\mathcal{H}$: inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$, norm $\| \cdot \|_{\mathcal{H}}$ (Hilbert Spaces)
- **Theorem** *All* nice Hilbert spaces are generated by a symmetric positive definite function (the kernel) $k(\mathbf{x}, \mathbf{x}')$ on $\mathcal{X} \times \mathcal{X}$
  - if $f, g \in \mathcal{H}$ close i.e. $\|f - g\|_{\mathcal{H}}$ small, then $f(\mathbf{x}), g(\mathbf{x})$ close $\forall \mathbf{x} \in \mathcal{X}$.
  - Reproducing Kernel Hilbert Spaces (RKHSs)
- Functional Analysis (Aronszajn, Bergman (1950s)); Statistics (Parzen (1960s)); PDEs; Numerical Analysis. . .
  - ML: Nonlinear classification, regression, clustering, time-series analysis, dynamical systems, hypothesis testing, causal modeling, . . .
- In principle, possible to compose Deep Learning pipelines using more general nonlinear functions drawn from RKHSs.

Outline

Motivation and Background

Scalable Kernel Methods

- Random Embeddings+Distributed Computation (ICASSP, JSM 2014)
- libSkylark: An open source software stack
- Quasi-Monte Carlo Embeddings (ICML 2014)

Synergies?

Scalability Challenges for Kernel Methods

$$f^\star = \arg\min_{f \in \mathcal{H}_k} \frac{1}{n} \sum_{i=1}^{n} V(y_i, f(\mathbf{x}_i)) + \lambda \|f\|^2_{\mathcal{H}_k}, \ \mathbf{x}_i \in \mathbb{R}^d$$

- Representer Theorem: $f^\star(\mathbf{x}) = \sum_{i=1}^n \alpha_i k(\mathbf{x}, \mathbf{x}_i)$
- Regularized Least Squares

$$(\mathbf{K} + \lambda \mathbf{I})\boldsymbol{\alpha} = \mathbf{Y} \qquad \begin{array}{cc} O(n^2) & \text{storage} \\ O(n^3 + n^2 d) & \text{training} \\ O(nd) & \text{test speed} \end{array}$$

- Hard to parallelize when working directly with $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$

Randomized Algorithms

- Explicit approximate feature map: $\Psi : \mathbb{R}^d \mapsto \mathbb{C}^s$ such that,

$$
\begin{array}{lll}
k(\mathbf{x}, \mathbf{z}) & \approx & \langle \hat{\Psi}(\mathbf{x}), \hat{\Psi}(\mathbf{z}) \rangle_{\mathbb{C}^s} \\
& \Rightarrow & \left(\mathbf{Z}(\mathbf{X})^T \mathbf{Z}(\mathbf{X}) + \lambda \mathbf{I}\right) \mathbf{w} = \mathbf{Z}(\mathbf{X})^T \mathbf{Y}, \quad \begin{array}{cc} O(ns) & \text{storage} \\ O(ns^2) & \text{training} \\ O(s) & \text{test speed} \end{array}
\end{array}
$$

- Interested in *Data-oblivious* maps that depend only on the kernel function, and not on the data.
  – Should be very cheap to apply and parallelizable.

Random Fourier Features (Rahimi & Recht, 2007)

- **Theorem** [Bochner 1930,33] One-to-one Fourier-pair correspondence between any (normalized) shift-invariant kernel k and density p such that,

$$k(\mathbf{x}, \mathbf{z}) = \psi(\mathbf{x} - \mathbf{z}) = \int_{\mathbb{R}^d} e^{-i(\mathbf{x}-\mathbf{z})^T \mathbf{w}} p(\mathbf{w}) d\mathbf{w}$$

  - Gaussian kernel: $k(\mathbf{x}, \mathbf{z}) = e^{-\frac{\|\mathbf{x}-\mathbf{z}\|_2^2}{2\sigma^2}} \iff p = \mathcal{N}(0, \sigma^{-2}\mathbf{I}_d)$

Random Fourier Features (Rahimi & Recht, 2007)

- **Theorem** [Bochner 1930,33] One-to-one Fourier-pair correspondence between any (normalized) shift-invariant kernel k and density p such that,

$$k(\mathbf{x},\mathbf{z}) = \psi(\mathbf{x}-\mathbf{z}) = \int_{\mathbb{R}^d} e^{-i(\mathbf{x}-\mathbf{z})^T\mathbf{w}} p(\mathbf{w}) d\mathbf{w}$$

  – Gaussian kernel: $k(\mathbf{x},\mathbf{z}) = e^{-\frac{\|\mathbf{x}-\mathbf{z}\|_2^2}{2\sigma^2}} \iff p = \mathcal{N}(0, \sigma^{-2}\mathbf{I}_d)$

- Monte-Carlo approximation to Integral representation:

$$k(\mathbf{x},\mathbf{z}) \approx \frac{1}{s}\sum_{j=1}^{s} e^{-i(\mathbf{x}-\mathbf{z})^T\mathbf{w}_j} = \langle \hat{\Psi}_S(\mathbf{x}), \hat{\Psi}_S(\mathbf{z}) \rangle_{\mathbb{C}^s}$$

$$\hat{\Psi}_S(\mathbf{x}) = \frac{1}{\sqrt{s}}\left[e^{-i\mathbf{x}^T\mathbf{w}_1} \ldots e^{-i\mathbf{x}^T\mathbf{w}_s}\right] \in \mathbb{C}^s, \quad S = [\mathbf{w}_1 \ldots \mathbf{w}_s] \sim p$$

DNNs vs Kernel Methods on TIMIT (Speech)

Joint work with IBM Speech Group, P. Huang:

Can "shallow", convex randomized kernel methods "match" DNNs?

(predicting HMM states given short window of coefficients representing acoustic input)

```
G = randn(size(X,1), s);
Z = exp(i*X*G);
I = eye(size(X,2));
C = Z'*Z;
alpha = (C + lambda*I)\(Z'*y(:));
ztest = exp(i*xtest*G)*alpha;
```

DNNs vs Kernel Methods on TIMIT (Speech)

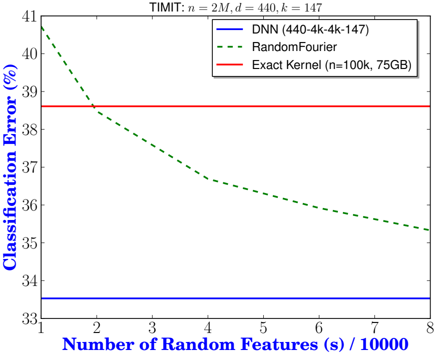
Joint work with IBM Speech Group, P. Huang:

Can "shallow", convex randomized kernel methods "match" DNNs?

(predicting HMM states given short window of coefficients representing acoustic input)

```
G = randn(size(X,1), s);
Z = exp(i*X*G);
I = eye(size(X,2));
C = Z'*Z;
alpha = (C + lambda*I)\(Z'*y(:));
ztest = exp(i*xtest*G)*alpha;
```

- $\mathbf{Z}(\mathbf{X})$: 1.2TB
- Stream on blocks $\mathbf{C}+ = \mathbf{Z}'_B\mathbf{Z}_B$
- But $\mathbf{C}$ also big (47GB).
- Need: Distributed solvers to handle big n, s; $Z(X)$ *implicitly*.



DNNs vs Kernel Methods on TIMIT (Speech)

Kernel Methods match DNNs on TIMIT, ICASSP 2014, with P. Huang and IBM Speech group

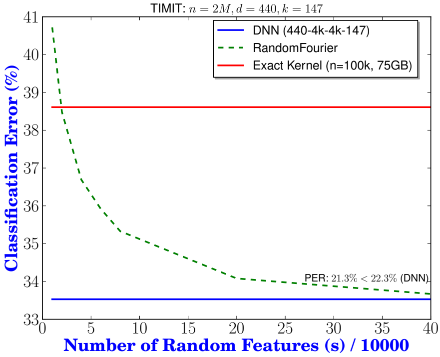
High-performance Kernel Machines with Implicit Distributed Optimization and Randomization, JSM 2014, with H. Avron.

DNNs vs Kernel Methods on TIMIT (Speech)

Kernel Methods match DNNs on TIMIT, ICASSP 2014, with P. Huang and IBM Speech group

High-performance Kernel Machines with Implicit Distributed Optimization and Randomization, JSM 2014, with H. Avron.

- ~ 2 hours on 256 IBM Bluegene/Q nodes.
- Phone error rate of 21.3% - best reported for Kernel methods.
  - Competitive with HMM/DNN systems.
  - New record: 16.7% with CNNs (ICASSP 2014).
- Only two hyperparameters: σ, s (early stopping regularizer).
- $\mathbf{Z} \approx 6.4TB$, $\mathbf{C} \approx 1.2TB$.
- Materialized in blocks/used/discarded on-the-fly, in parallel.



Distributed Convex Optimization

- Alternating Direction Method of Multipliers (50s; Boyd et al, 2013)

$$\arg\min_{x \in \mathbb{R}^n, z \in \mathbb{R}^m} f(x) + g(z) \text{ subject to } Ax + Bz = c$$

- Row/Column Splitting; **Block splitting** (Parikh & Boyd, 2013)

$$\arg\min_{x \in \mathbb{R}^d} \sum_{i=1}^{R} f_i(x) + g(x) \quad \Rightarrow \quad \sum_{i=1}^{R} f_i(x_i) + g(z) \text{ s.t } x_i = z \tag{1}$$

$$x_i^{(k+1)} = \arg\min_x f_i(x) + \frac{\rho}{2}\|x - z^k + \nu_i^k\|_2^2 \tag{2}$$

$$z = prox_{g/(R\rho)}[\bar{x}^{k+1} + \bar{\nu}^k] \quad \text{(comm.)} \tag{3}$$

$$\nu_i^{k+1} = \nu_i^k + x_i^{k+1} - z^{k+1} \tag{4}$$

$$\text{where } prox_f[x] = \arg\min_y \frac{1}{2}\|x - y\|_2^2 + f(y)$$

- Note: extra consensus and dual variables need to be managed.
- Closed-form updates, Extensibility, Code-reuse, Parallelism.

Distributed Block-splitting ADMM

https://github.com/xdata-skylark/libskylark/tree/master/ml

node 1

$\boxed{Y_1, X_1}$

node 2

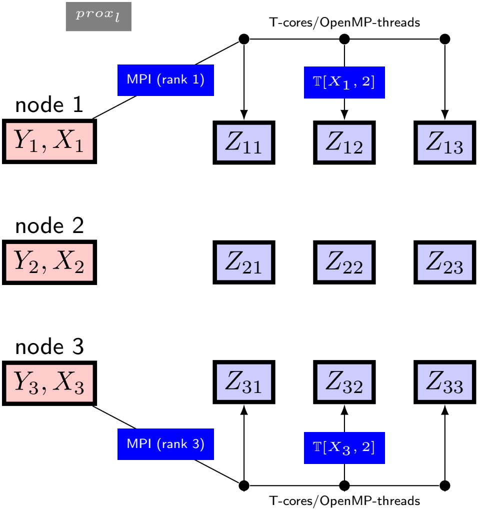
$\boxed{Y_2, X_2}$

node 3

$\boxed{Y_3, X_3}$

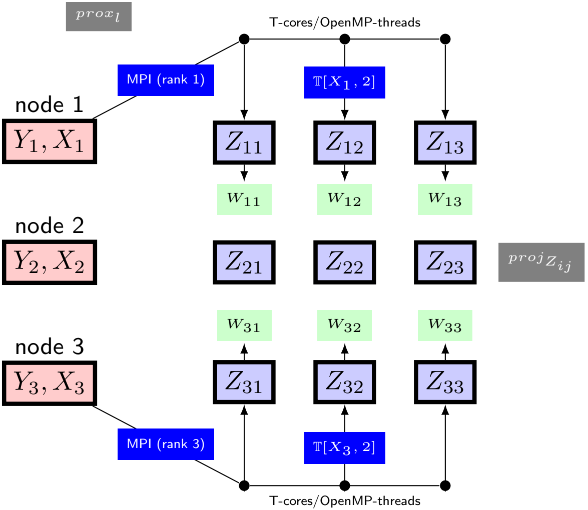
Distributed Block-splitting ADMM

https://github.com/xdata-skylark/libskylark/tree/master/ml



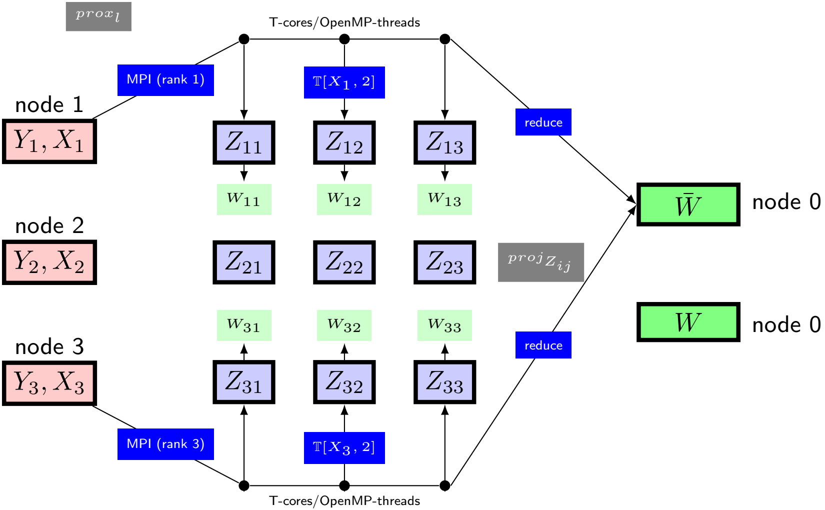
Distributed Block-splitting ADMM

https://github.com/xdata-skylark/libskylark/tree/master/ml



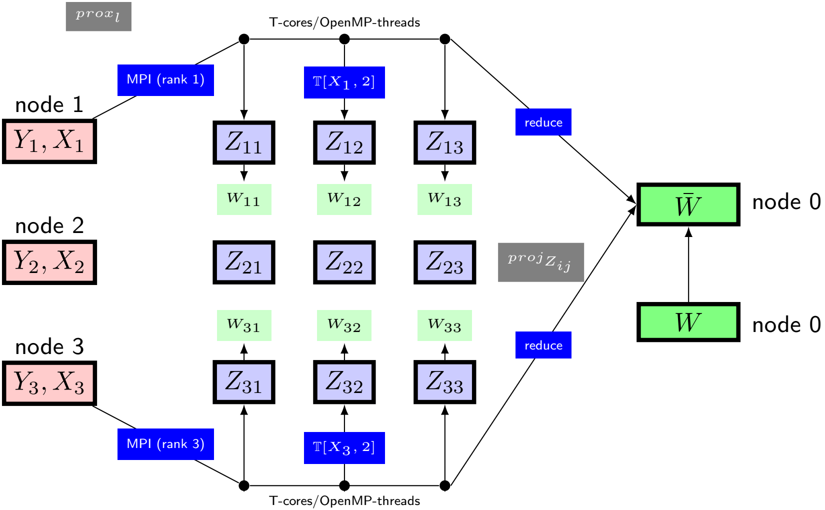
Distributed Block-splitting ADMM

https://github.com/xdata-skylark/libskylark/tree/master/ml



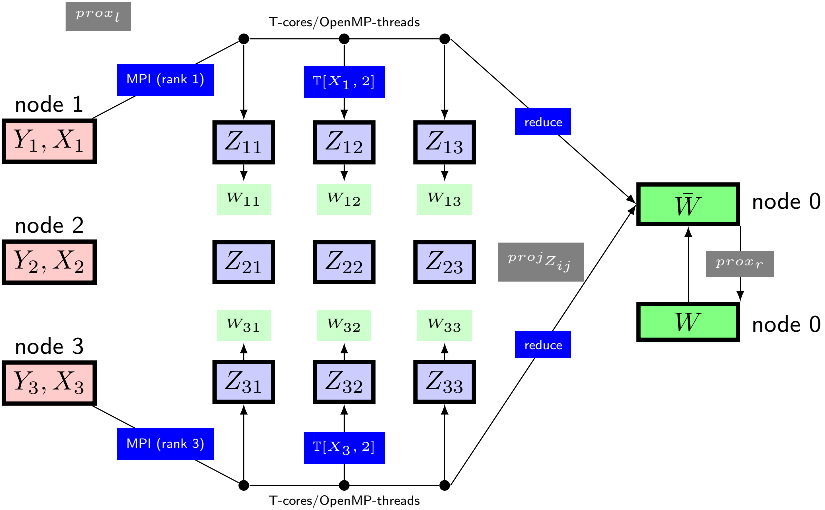
Distributed Block-splitting ADMM

https://github.com/xdata-skylark/libskylark/tree/master/ml



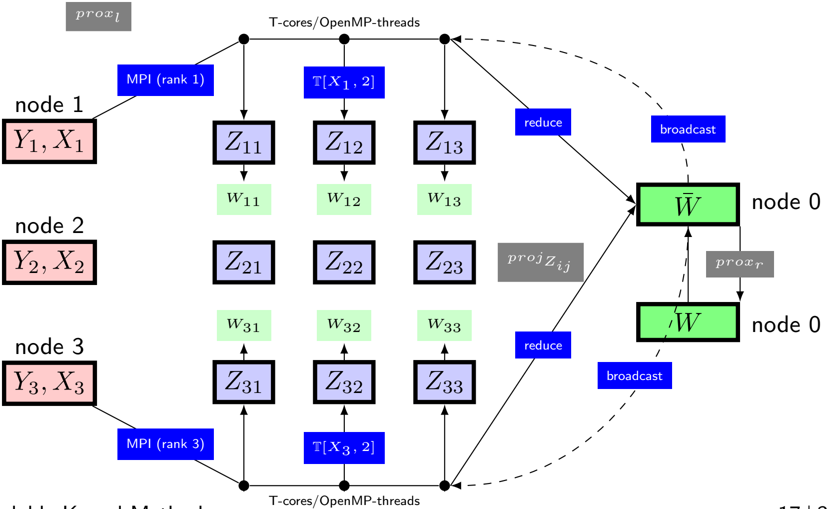
Distributed Block-splitting ADMM

https://github.com/xdata-skylark/libskylark/tree/master/ml



Distributed Block-splitting ADMM

https://github.com/xdata-skylark/libskylark/tree/master/ml



Scalability

- Graph Projection

$$\mathbf{U} = \underbrace{[\mathbf{Z}_{ij}^T\mathbf{Z}_{ij} + \lambda\mathbf{I}]^{-1}}_{\text{cached}}(\mathbf{X} + \underbrace{\mathbf{Z}_{ij}^T\mathbf{Y}}_{\text{gemm}}),\ \ \mathbf{V} = \underbrace{\mathbf{Z}_{ij}\mathbf{U}}_{\text{gemm}}$$

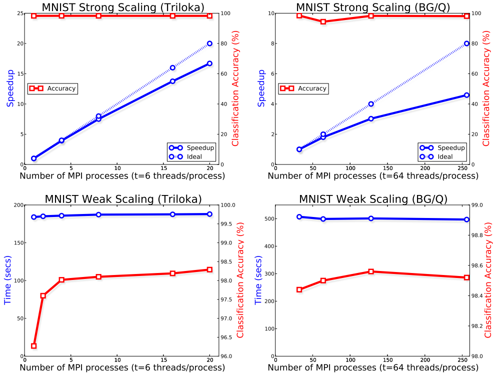
- High-performance implementation that can handle large column splits $C = \kappa\frac{s}{d}$ by reorganizing updates to exploit shared-memory access, structure of graph projection.

$$\begin{array}{rc}
\text{Memory} & \frac{nm}{R}(T+5) + \frac{nd}{R} + 5sm + \frac{1}{\kappa}\left(\frac{Tnd}{R} + Tmd + sd\right) + \kappa\cancel{\frac{nms}{dR}} \\
\text{Computation} & \underbrace{O(\frac{nd^2}{TR\kappa})}_{\text{transform}} + \underbrace{O(\frac{smd}{\kappa T})}_{\text{graph-proj}} + \underbrace{O(\frac{nsm}{TR})}_{\text{gemm}} \\
\text{Communication} & O(s\ m\ \ \log\ \ R)\quad (\text{model reduce/broadcast})
\end{array}$$

- Stochastic, Asynchronous versions may be possible to develop.

Randomized Kernel methods on thousands of cores

- Triloka: 20 nodes/16-cores per node; BG/Q: 1024 nodes/16 (x4) cores per node.
- $s = 100K, C = 200$; strong scaling ($n = 250k$), weak scaling ($n = 250k$ per node)



Comparisons on MNIST

See "no distortion" results at http://yann.lecun.com/exdb/mnist/

Gaussian Kernel	1.4	
Poly (4)	1.1	
3-layer NN, 500+300 HU	1.53	Hinton, 2005
LeNet5	0.8	LeCun et al. 1998
Large CNN (pretrained)	0.60	Ranzato et al., NIPS 2006
Large CNN (pretrained)	0.53	Jarrett et al., ICCV 2009
Scattering transform + Gaussian Kernel	0.4	Bruna and Mallat, 2012
20000 random features	0.45	my experiments
5000 random features	0.52	my experiments
committee of 35 conv. nets [elastic distortions]	0.23	Ciresan et al, 2012

Comparisons on MNIST

See "no distortion" results at http://yann.lecun.com/exdb/mnist/

Gaussian Kernel	1.4	
Poly (4)	1.1	
3-layer NN, 500+300 HU	1.53	Hinton, 2005
LeNet5	0.8	LeCun et al. 1998
Large CNN (pretrained)	0.60	Ranzato et al., NIPS 2006
Large CNN (pretrained)	0.53	Jarrett et al., ICCV 2009
Scattering transform + Gaussian Kernel	0.4	Bruna and Mallat, 2012
20000 random features	0.45	my experiments
5000 random features	0.52	my experiments
committee of 35 conv. nets [elastic distortions]	0.23	Ciresan et al, 2012

- When similar prior knowledge is enforced (invariant learning), performance gaps vanish.
- RKHS mappings can, in principle, be used to implement CNNs.

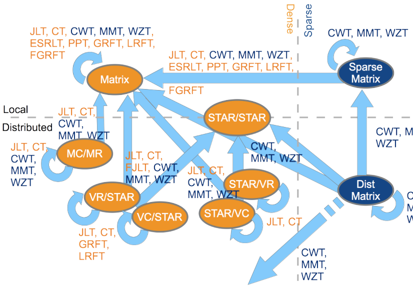
Aside: LibSkylark

http://xdata-skylark.github.io/libskylark/docs/sphinx/

- C/C++/Python library, MPI, Elemental/CombBLAS containers.
- Distributed Sketching operators

$$\|\mathbf{Ax} - \mathbf{b}\|_2 \Rightarrow \|\mathbf{S}\left(\mathbf{Ax} - \mathbf{b}\right)\|_2$$

- Randomized Least Squares, SVD
- Randomized Kernel Methods:
  - Modularity via Prox operators
  - Sqr, hinge, L1, mult. logreg.
  - Regularizers: $L1, L2$



Kernel	Embedding	$\mathbf{z}(\mathbf{x})$	Time
Shift-Invariant	RFT	$e^{-i\mathbf{Gx}}$	$O(sd), O(s\ nnz(\mathbf{x}))$
Shift-Invariant	FRFT	$e^{i\mathbf{SGHPHBx}}$	$O(s\ \log(d))$
Semigroup	RLT	$e^{-\mathbf{Gx}}$	$O(sd), O(s\ nnz(\mathbf{x}))$
Polynomial (deg q)	PPT	$F^{-1}\left(F(\mathbf{C}_1\mathbf{x}) \ldots . * F(\mathbf{C}_q\mathbf{x})\right)$	$O(q(nnz(\mathbf{x}) + s\ \log\ s)$

Rahimi & Recht, 2007; Pham & Pagh 2013; Le, Sarlos and Smola, 2013

Random Laplace Feature Maps for Semigroup Kernels on Histograms, CVPR 2014, J. Yang, V.S., M. Mahoney, H. Avron, Q. Fan.

Efficiency of Random Embeddings

- TIMIT: $58.8M$ $(s = 400k, m = 147)$ vs DNN $19.9M$ parameters.
- Draw $S = [\mathbf{w}_1 \dots \mathbf{w}_s] \sim p$ and approximate the integral:

$$k(\mathbf{x}, \mathbf{z}) = \int_{\mathbb{R}^d} e^{-i(\mathbf{x}-\mathbf{z})^T \mathbf{w}} p(\mathbf{w}) d\mathbf{w} \approx \frac{1}{|S|} \sum_{\mathbf{w} \in S} e^{-i(\mathbf{x}-\mathbf{z})^T \mathbf{w}} \tag{6}$$

Efficiency of Random Embeddings

- TIMIT: $58.8M$ ($s = 400k, m = 147$) vs DNN $19.9M$ parameters.
- Draw $S = [\mathbf{w}_1 \ldots \mathbf{w}_s] \sim p$ and approximate the integral:

$$k(\mathbf{x}, \mathbf{z}) = \int_{\mathbb{R}^d} e^{-i(\mathbf{x}-\mathbf{z})^T \mathbf{w}} p(\mathbf{w}) d\mathbf{w} \approx \frac{1}{|S|} \sum_{\mathbf{w} \in S} e^{-i(\mathbf{x}-\mathbf{z})^T \mathbf{w}} \tag{6}$$

- Integration error

$$\epsilon_{p,S}[f] = \left| \int_{[0,1]^d} f(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} - \frac{1}{s} \sum_{\mathbf{w} \in S} f(\mathbf{w}) \right|$$

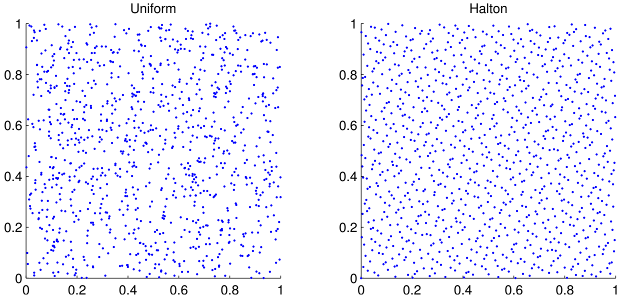
- Monte-carlo convergence rate: $\mathbf{E}[\epsilon_{p,S}[f]^2]^{\frac{1}{2}} \propto s^{-\frac{1}{2}}$
  - 4-fold increase in s will only cut error by half.
- Can we do better with a different sequence S?

Quasi-Monte Carlo Sequences: Intuition

- Weyl 1916; Koksma 1942; Dick et. al., 2013; Caflisch, 1998
- Consider approximating $\int_{[0,1]^2} f(\mathbf{x})d\mathbf{x}$ with $\frac{1}{s}\sum_{\mathbf{w}\in S} f(\mathbf{w})$

Quasi-Monte Carlo Sequences: Intuition

- Weyl 1916; Koksma 1942; Dick et. al., 2013; Caflisch, 1998
- Consider approximating $\int_{[0,1]^2} f(\mathbf{x}) d\mathbf{x}$ with $\frac{1}{s} \sum_{\mathbf{w} \in S} f(\mathbf{w})$



- Deterministic correlated QMC sampling avoid clustering, clumping effects in MC point sets.
- Hierarchical structure: coarse-to-fine sampling as s increases.

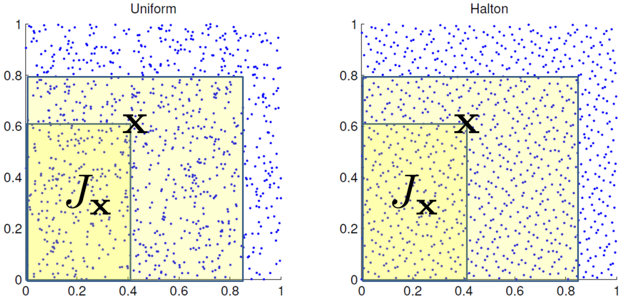
Star Discrepancy

- Integration error depends on variation f and uniformity of S.

Star Discrepancy

- Integration error depends on variation f and uniformity of S.
- **Theorem** (Koksma-Hlawka inequality (1941, 1961))

$$\begin{aligned} \epsilon_S[f] &\leq D^\star(S) V_{HK}[f], \quad \text{where} \\ D^\star(S) &= \sup_{\mathbf{x}\in[0,1]^d} \left| \mathbf{vol}(J_\mathbf{x}) - \frac{|\{i \ : \ \mathbf{w}_i \in J_\mathbf{x}\}|}{s} \right| \end{aligned}$$



Quasi-Monte Carlo Sequences

- We seek sequences with low discrepancy.

Quasi-Monte Carlo Sequences

- We seek sequences with low discrepancy.
- **Theorem** (Roth, 1954) $D^\star(S) \geq c_d \frac{(\log s)^{\frac{d-1}{2}}}{s}$.

Quasi-Monte Carlo Sequences

- We seek sequences with low discrepancy.
- **Theorem** (Roth, 1954) $D^\star(S) \geq c_d \frac{(\log s)^{\frac{d-1}{2}}}{s}$.
- For the regular grid on $[0,1)^d$ with $s = m^d$, $D^\star \geq s^{\frac{1}{d}} \implies$ only optimal for $d = 1$.

Quasi-Monte Carlo Sequences

- We seek sequences with low discrepancy.
- **Theorem** (Roth, 1954) $D^{\star}(S) \geq c_d \frac{(\log s)^{\frac{d-1}{2}}}{s}$.
- For the regular grid on $[0,1)^d$ with $s = m^d$, $D^{\star} \geq s^{\frac{1}{d}} \implies$ only optimal for $d = 1$.
- **Theorem** (Doerr, 2013) Let S with s be chosen uniformly at random from $[0,1)^d$. Then, $\mathbb{E}[D^{\star}(S)] \geq \frac{\sqrt{d}}{\sqrt{s}}$ - the Monte Carlo rate.

Quasi-Monte Carlo Sequences

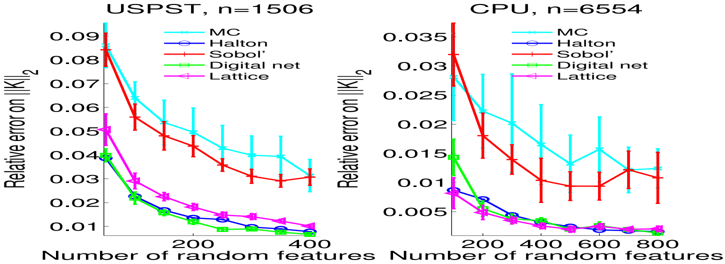
- We seek sequences with low discrepancy.
- **Theorem** (Roth, 1954) $D^\star(S) \geq c_d \frac{(\log s)^{\frac{d-1}{2}}}{s}$.
- For the regular grid on $[0,1)^d$ with $s = m^d$, $D^\star \geq s^{\frac{1}{d}} \implies$ only optimal for $d = 1$.
- **Theorem** (Doerr, 2013) Let S with s be chosen uniformly at random from $[0,1)^d$. Then, $\mathbb{E}[D^\star(S)] \geq \frac{\sqrt{d}}{\sqrt{s}}$ - the Monte Carlo rate.
- There exist *low-discrepancy sequences* that achieve a $C_d \frac{(\log \ s)^d}{s}$ lower bound conjectured to be optimal.
  - Matlab QMC generators: `haltonset`, `sobolset` ...

Quasi-Monte Carlo Sequences

- We seek sequences with low discrepancy.
- **Theorem** (Roth, 1954) $D^\star(S) \geq c_d \frac{(\log s)^{\frac{d-1}{2}}}{s}$.
- For the regular grid on $[0,1)^d$ with $s = m^d$, $D^\star \geq s^{\frac{1}{d}} \implies$ only optimal for $d = 1$.
- **Theorem** (Doerr, 2013) Let S with s be chosen uniformly at random from $[0,1)^d$. Then, $\mathbb{E}[D^\star(S)] \geq \frac{\sqrt{d}}{\sqrt{s}}$ - the Monte Carlo rate.
- There exist *low-discrepancy sequences* that achieve a $C_d \frac{(\log \ s)^d}{s}$ lower bound conjectured to be optimal.
  - Matlab QMC generators: `haltonset`, `sobolset` ...
- With d fixed, this bound actually grows until $s \sim e^d$ to $(d/e)^d$!
  - "On the unreasonable effectiveness of QMC" in high-dimensions.
  - RKHSs and Kernels show up in Modern QMC analysis!

How do standard QMC sequences perform?

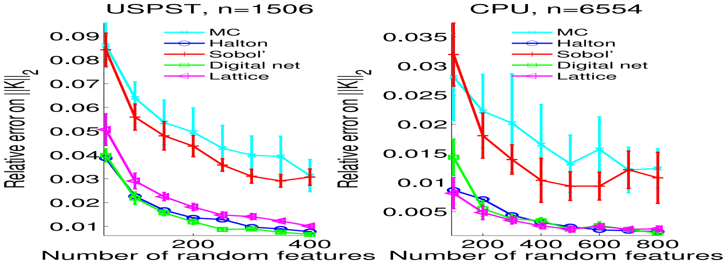
- Compare $\mathbf{K} \approx \mathbf{Z}(\mathbf{X})\mathbf{Z}(\mathbf{X})^T$ where $\mathbf{Z}(\mathbf{X}) = e^{-i\mathbf{X}\mathbf{G}}$ where $\mathbf{G}$ is drawn from a QMC sequence generator instead.



- QMC sequences consistently better.

How do standard QMC sequences perform?

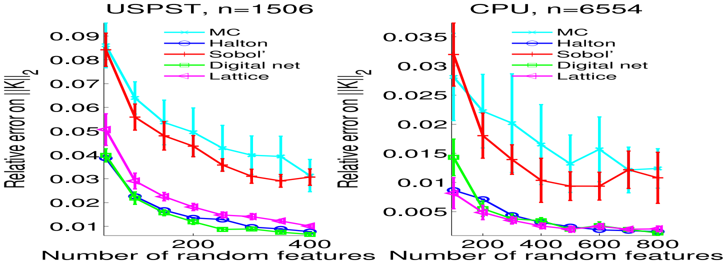
- Compare $\mathbf{K} \approx \mathbf{Z}(\mathbf{X})\mathbf{Z}(\mathbf{X})^T$ where $\mathbf{Z}(\mathbf{X}) = e^{-i\mathbf{X}\mathbf{G}}$ where $\mathbf{G}$ is drawn from a QMC sequence generator instead.



- QMC sequences consistently better.
- Why are some QMC sequences better, e.g., Halton over Sobol?

How do standard QMC sequences perform?

- Compare $\mathbf{K} \approx \mathbf{Z}(\mathbf{X})\mathbf{Z}(\mathbf{X})^T$ where $\mathbf{Z}(\mathbf{X}) = e^{-i\mathbf{X}\mathbf{G}}$ where $\mathbf{G}$ is drawn from a QMC sequence generator instead.



- QMC sequences consistently better.
- Why are some QMC sequences better, e.g., Halton over Sobol?
- Can we learn sequences even better adapted to our problem class?

RKHSs in QMC Theory

- "Nice" integrands $f \in \mathcal{H}_h$, where $h(\cdot, \cdot)$ is a kernel function.

RKHSs in QMC Theory

- "Nice" integrands $f \in \mathcal{H}_h$, where $h(\cdot,\cdot)$ is a kernel function.
- By Reproducing property and Cauchy-Schwartz inequality:

$$\left| \int_{\mathbb{R}^d} f(\mathbf{x})p(\mathbf{x})d\mathbf{x} - \frac{1}{s}\sum_{\mathbf{w}\in S} f(\mathbf{w}) \right| \leq \|f\|_k D_{h,p}(S) \tag{7}$$

where $D^2_{h,p}$ is a **discrepancy** measure:

$$D_{h,p}(S) \quad = \quad \|m_p(\cdot) - \frac{1}{s}\sum_{\mathbf{w}\in S} h(\mathbf{w},\cdot)\|^2_{\mathcal{H}_h}, \qquad \overbrace{m_p = \int_{\mathbb{R}^d} h(\mathbf{x},\cdot)p(\mathbf{x})d\mathbf{x}}^{\text{mean embedding}}$$

RKHSs in QMC Theory

- "Nice" integrands $f \in \mathcal{H}_h$, where $h(\cdot,\cdot)$ is a kernel function.
- By Reproducing property and Cauchy-Schwartz inequality:

$$\left| \int_{\mathbb{R}^d} f(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} - \frac{1}{s} \sum_{\mathbf{w} \in S} f(\mathbf{w}) \right| \leq \|f\|_k D_{h,p}(S) \tag{7}$$

where $D_{h,p}^2$ is a **discrepancy** measure:

$$D_{h,p}(S) = \|m_p(\cdot) - \frac{1}{s} \sum_{\mathbf{w} \in S} h(\mathbf{w}, \cdot)\|^2_{\mathcal{H}_h}, \qquad \overbrace{m_p = \int_{\mathbb{R}^d} h(\mathbf{x}, \cdot) p(\mathbf{x}) d\mathbf{x}}^{\text{mean embedding}}$$

$$= \text{const.} - \underbrace{\frac{2}{s} \sum_{l=1}^{s} \int_{\mathbb{R}^d} h(\mathbf{w}_l, \omega) p(\omega) d\omega}_{\text{Alignment with p } (\mathbf{w}_l \approx \omega)} + \underbrace{\frac{1}{s^2} \sum_{l=1}^{s} \sum_{j=1}^{s} h(\mathbf{w}_l, \mathbf{w}_j)}_{\text{Pairwise similarity in } S}$$

Box Discrepancy

- Assume that the data (shifted) lives in a box

$$\Box\mathbf{b} = -\mathbf{b} \leq \mathbf{x} - \mathbf{z} \leq \mathbf{b}, \mathbf{x}, \mathbf{z} \in \mathcal{X}$$

- Class of functions we want to integrate:

$$\mathcal{F}_{\Box b} = \{f(\mathbf{w}) = e^{-i\mathbf{\Delta}^T\mathbf{w}}, \Delta \in \Box\mathbf{b}\}$$

Box Discrepancy

- Assume that the data (shifted) lives in a box

$$\Box\mathbf{b} = -\mathbf{b} \leq \mathbf{x} - \mathbf{z} \leq \mathbf{b}, \mathbf{x}, \mathbf{z} \in \mathcal{X}$$

- Class of functions we want to integrate:

$$\mathcal{F}_{\Box b} = \{f(\mathbf{w}) = e^{-i\mathbf{\Delta}^T\mathbf{w}}, \Delta \in \Box\mathbf{b}\}$$

- **Theorem**: Integration error proportional to "Box discrepancy":

$$\mathbb{E}_{f\sim\mathcal{U}(\mathcal{F}_{\Box b})}\left[\epsilon_{S,p}[f]^2\right]^{\frac{1}{2}} \propto D_p^{\Box}(S) \tag{9}$$

where $D_p^{\Box}(S)$ is discrepancy associated with the **sinc** kernel:

$$\mathbf{sinc_b}(\mathbf{u}, \mathbf{v}) = \pi^{-d} \prod_{i=1}^{d} \frac{sin(b_j(u_j - v_j))}{u_j - v_j}$$

Box Discrepancy

- Assume that the data (shifted) lives in a box

$$\square \mathbf{b} = -\mathbf{b} \leq \mathbf{x} - \mathbf{z} \leq \mathbf{b}, \mathbf{x}, \mathbf{z} \in \mathcal{X}$$

- Class of functions we want to integrate:

$$\mathcal{F}_{\square b} = \{f(\mathbf{w}) = e^{-i\mathbf{\Delta}^T \mathbf{w}}, \Delta \in \square \mathbf{b}\}$$

- **Theorem**: Integration error proportional to "Box discrepancy":

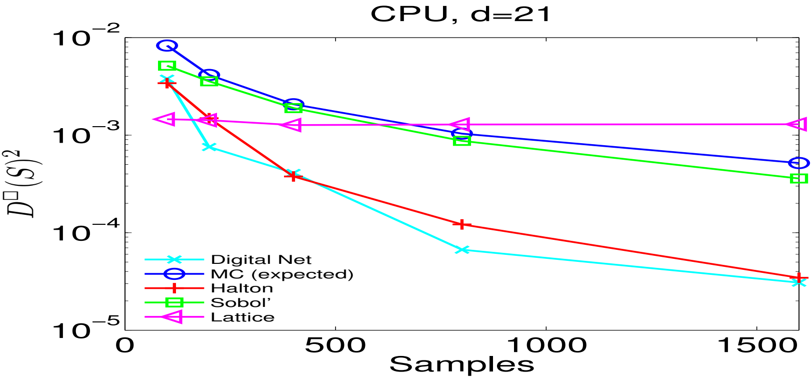
$$\mathbb{E}_{f\sim\mathcal{U}(\mathcal{F}_{\square b})} \left[\epsilon_{S,p}[f]^2\right]^{\frac{1}{2}} \propto D_p^{\square}(S) \tag{9}$$

where $D_p^{\square}(S)$ is discrepancy associated with the **sinc** kernel:

$$\mathbf{sinc_b}(\mathbf{u}, \mathbf{v}) = \pi^{-d} \prod_{i=1}^{d} \frac{sin(b_j(u_j - v_j))}{u_j - v_j}$$

Can be evaluated in closed form for Gaussian density.

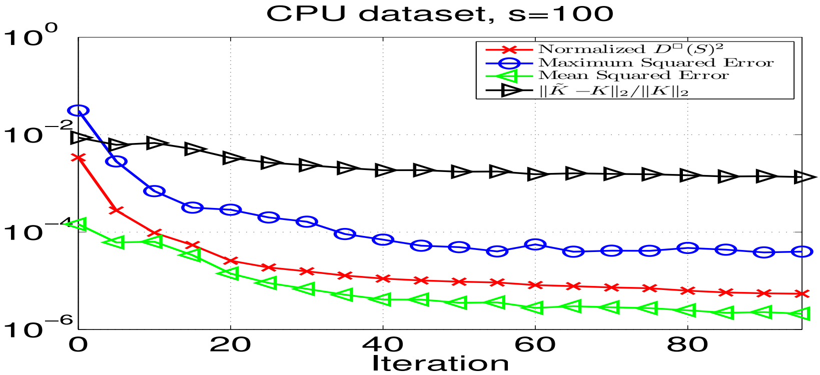
Does Box discrepancy explain behaviour of QMC sequences?



Learning Adaptive QMC Sequences

Unlike Star discrepancy, Box discrepancy admits numerical optimization,

$$S^* = \underset{S=(\mathbf{w}_1 \ldots \mathbf{w}_s) \in \mathbb{R}^{ds}}{\arg\min} D^{\square}(S), \quad S^{(0)} = \text{Halton}. \tag{10}$$



However, full impact on large-scale problems is an open research topic.

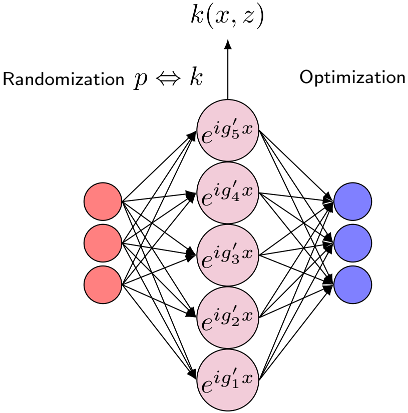
Outline

Motivation and Background

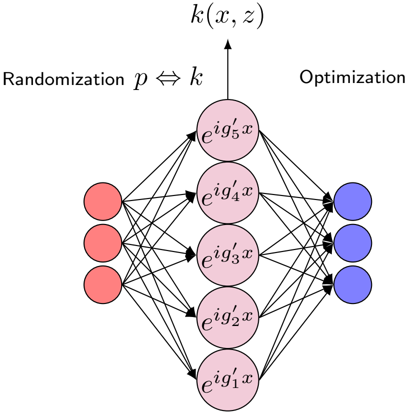
Scalable Kernel Methods
- Random Embeddings+Distributed Computation (ICASSP, JSM 2014)
- libSkylark: An open source software stack
- Quasi-Monte Carlo Embeddings (ICML 2014)

Synergies?

Randomization-vs-Optimization



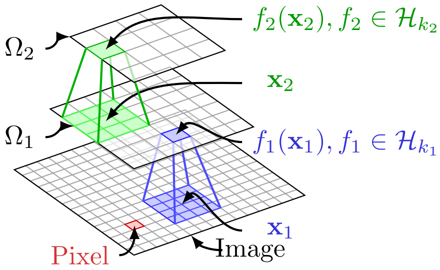
Randomization-vs-Optimization



- Jarret et al 2009, What is the Best Multi-Stage Architecture for Object Recognition?: *"The most astonishing result is that systems with random filters and no filter learning whatsoever achieve decent performance"*
- On Random Weights and Unsupervised Feature Learning, Saxe et al, ICML 2011: "surprising fraction of performance can be attributed to architecture alone."

Deep Learning with Kernels?

Maps across layers can be parameterized using more general nonlinearities (kernels).



- Mathematics of Neural Response, Smale et. al., FCM (2010).
- Convolutional Kernel Networks, Mairal et. al., NIPS 2014.
- SimNets: A Generalization of Conv. Nets, Cohen and Sashua, 2014.
  - learns networks 1/8 the size of comparable ConvNets.

Figure adapted from Mairal et al, NIPS 2014

Conclusions

- Some empirical evidence suggests that once Kernel methods are scaled up and embody similar statistical principles, they are competitive with Deep Neural Networks.
  - Randomization and Distributed Computation *both* required.
  - Ideas from QMC Integration techniques are promising.
- Opportunities for designing new algorithms combining insights from deep learning with the generality and mathematical richness of kernel methods.