LARGE-SCALE KERNEL METHODS

MDI 341, MS Big Data, Télécom ParisTech

Aurélien Bellet

Inria Lille

February 25, 2020

OUTLINE

1. Reminder on Kernel Methods
2. Scalability Issues
3. Random Kernel Features
4. Nyström Approximation
5. Conclusion

REMINDER ON KERNEL METHODS

- Training observations $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$
- For now, we assume $\mathcal{X} \subset \mathbb{R}^p$
- Training labels $y_1, \ldots, y_n \in \{-1, 1\}$
- Linear classifier with parameters $\mathbf{w} \in \mathbb{R}^p$, $b \in \mathbb{R}$:

$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x} + b)$$

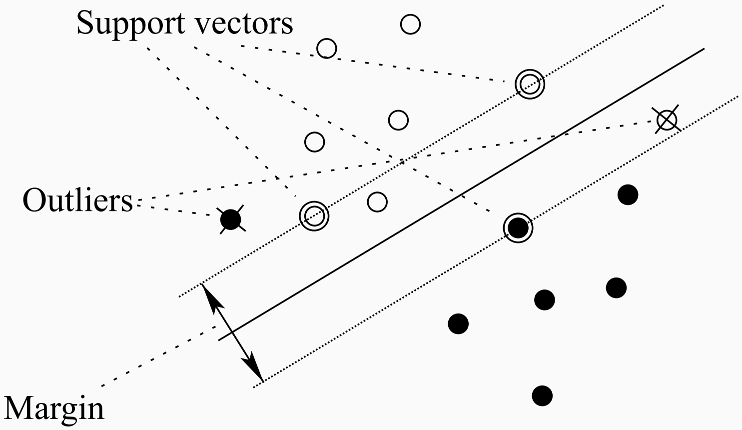
SVM: primal formulation

$$\min_{\boldsymbol{w}\in\mathbb{R}^p, b\in\mathbb{R}, \boldsymbol{\xi}\in\mathbb{R}^n} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 \quad + \quad C\sum_{i=1}^{n}\xi_i$$

$$\text{s.t.} \quad y_i(\boldsymbol{w}^T\boldsymbol{x} + b) \geq 1 - \xi_i \quad i = 1, \dots, n$$

$$\xi_i \geq 0 \quad i = 1, \dots, n$$

- Key principle: margin maximization
- Convex optimization problem


Support vectors
Outliers
Margin

SVM: Lagrange dual formulation

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

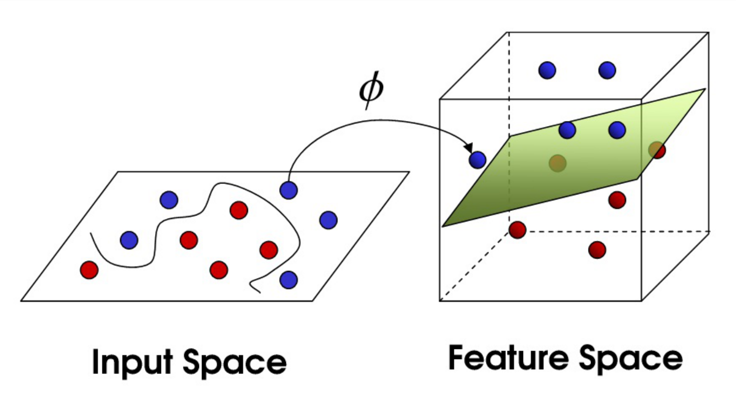
$$\text{s.t.} \quad 0 \leq \alpha_i \leq C \quad i = 1, \ldots, n$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

- Also convex
- We have $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x_i}$ and thus

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}^T \mathbf{x_i} + b \right)$$

- Note: b can be deduced from support vectors


ϕ
Input Space
Feature Space

Definition (Kernel function)

A symmetric function $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a *kernel* if there exists a mapping function $\phi : \mathcal{X} \rightarrow \mathbb{H}$ from the instance space $\mathcal{X}$ to a Hilbert space $\mathbb{H}$ such that K can be written as an inner product in $\mathbb{H}$:

$$K(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle .$$

Equivalently, K is a *kernel* if it is symmetric positive semi-definite (PSD), i.e.,

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j K(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

for all finite sequences of $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$ and $c_1, \ldots, c_n \in \mathbb{R}$.

Kernel SVM formulation

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq C \quad i = 1, \ldots, n$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

- We have $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \phi(\mathbf{x}_i)$ and thus

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b \right)$$

KERNELS: EXAMPLES

- Kernels between vectors
  - Linear kernel: $K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$
  - Polynomial kernel: $K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + c)^d$
  - Gaussian RBF kernel: $K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|_2^2)$
  - Laplace RBF kernel: $K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|_1)$
- Kernels on structured data
  - Many string, tree and graph kernels [Gärtner, 2003]
  - Frameworks to design structured kernels: convolution kernels [Haussler, 1999], mapping kernels [Shin and Kuboyama, 2008]
- Use closure properties to build new kernels from existing ones
  - Sums, positive combinations, etc

KERNELS: APPLICATIONS

- Kernels allow to obtain nonlinear variants for many linear machine learning algorithms
- A few examples
  - SVM
  - Ridge regression
  - PCA
  - CCA
  - K-Means
- Today's topic: scalability of kernel methods
  - Identify general issues
  - Study general solutions

SCALABILITY ISSUES

TRAINING TIME COMPLEXITY

- All kernel methods rely on the Gram matrix $\boldsymbol{G} \in \mathbb{R}^{n \times n}$

$$\boldsymbol{G} = \begin{pmatrix} K(\boldsymbol{x}_1, \boldsymbol{x}_1) & K(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ K(\boldsymbol{x}_2, \boldsymbol{x}_1) & K(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_2, \boldsymbol{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_n, \boldsymbol{x}_1) & K(\boldsymbol{x}_n, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{pmatrix}$$

- Assuming complexity of one kernel evaluation is constant
  - Constructing $\boldsymbol{G}$ takes $O(n^2)$ time
  - If we need to invert $\boldsymbol{G}$ (e.g., Kernel Ridge Regression): $O(n^3)$ time
- **Training time of popular algorithms:** $O(n^2)$ or $O(n^3)$
- This is infeasible for large n

TEST TIME COMPLEXITY

- Kernel methods are nonparametric: the learned model relies on the training data points
- To process a test point $\mathbf{x}$, one generally needs to evaluate the kernel between $\mathbf{x}$ and the training points
  - For instance in SVM:

$$f(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b\right)$$

  - Number of support vectors grows linearly with n [Steinwart, 2003]
- **Prediction time of kernel methods:** $O(n)$
- This is slow when n is large

- Today we will study two techniques which can be used to scale up any kernel method
- Random Kernel Features: approximate kernel function / map
- Nyström Approximation: approximate Gram matrix
- Both methods are very popular and successful in practice
- Still an active area of research

RANDOM KERNEL FEATURES

Kernel SVM : primal formulation

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|_2^2 \quad + \quad C\sum_{i=1}^{n}\left[y_i(\mathbf{w}^T\phi(\mathbf{x})+b)\right]_+$$

where $[a]_+ = \max(0, 1-a)$ is the hinge loss function

- When $\phi(\mathbf{x})$ is known and finite-dimensional (e.g., linear kernel)
  - Training time linear in n (see e.g., [Shalev-Shwartz et al., 2011])
  - Prediction complexity independent of n
- But $\phi(\mathbf{x})$ is usually unknown and potentially infinite-dimensional
- In this case we can only solve the problem in dual form
  - Training time quadratic or cubic in n
  - Prediction complexity linear in n

- **Idea:** find a finite-dimensional feature map $\hat{\phi}(x) \in \mathbb{R}^c$ such that

$$\left\langle \hat{\phi}(x), \hat{\phi}(x') \right\rangle \approx K(x, x')$$

- We can then solve in primal form to get $w \in \mathbb{R}^c$ and $b \in \mathbb{R}$
- We can predict using

$$f(x) = \mathrm{sign}(w^T \hat{\phi}(x) + b)$$

- If $c \ll n^2$, training is much faster
- If $c \ll n$, prediction is also much faster

Definition (Shift-invariant kernel)

Let $K : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ be a positive definite kernel. K is said to be *shift-invariant* if for any $\mathbf{a} \in \mathbb{R}^p$ and any $(\mathbf{x}, \mathbf{x'}) \in \mathbb{R}^p \times \mathbb{R}^p$

$$K(\mathbf{x} - \mathbf{a}, \mathbf{x}' - \mathbf{a}) = K(\mathbf{x}, \mathbf{x}').$$

For simplicity we denote $K(\mathbf{x}, \mathbf{x}') = K(\mathbf{x} - \mathbf{x}') = K(\boldsymbol{\Delta})$.

- Examples of shift-invariant kernels
    - Gaussian RBF kernel: $K(\mathbf{x} - \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|_2^2)$
    - Laplace kernel: $K(\mathbf{x} - \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|_1)$

Theorem (Bochner's theorem, see [Rahimi and Recht, 2007])

A continuous shift-invariant kernel $K(\mathbf{x}, \mathbf{x}') = K(\boldsymbol{\Delta})$ is positive definite if and only if $K(\boldsymbol{\Delta})$ is the Fourier transform of a nonnegative probability measure. In particular, if K is properly scaled, we have:

$$K(\mathbf{x} - \mathbf{x}') = \int_{\mathbb{R}^p} P(\boldsymbol{\omega}) e^{i\boldsymbol{\omega}^T\mathbf{x}} e^{-i\boldsymbol{\omega}^T\mathbf{x}'} d\boldsymbol{\omega},$$

where $P(\boldsymbol{\omega})$ is a real-valued probability density function over $\mathbb{R}^p$.

Developing the result of Bochner's theorem:

$$\begin{aligned}
K(\mathbf{x}-\mathbf{x}') &= \int_{\mathbb{R}^p} P(\boldsymbol{\omega}) e^{i\boldsymbol{\omega}^T\mathbf{x}} e^{-i\boldsymbol{\omega}^T\mathbf{x}'} d\boldsymbol{\omega} \\
&= \int_{\mathbb{R}^p} P(\boldsymbol{\omega}) \cos(\boldsymbol{\omega}^T\mathbf{x} - \boldsymbol{\omega}^T\mathbf{x}') d\boldsymbol{\omega} && (1) \\
&= \int_{\mathbb{R}^p} P(\boldsymbol{\omega}) \left(\cos(\boldsymbol{\omega}^T\mathbf{x})\cos(\boldsymbol{\omega}^T\mathbf{x}') + \sin(\boldsymbol{\omega}^T\mathbf{x})\sin(\boldsymbol{\omega}^T\mathbf{x}')\right) d\boldsymbol{\omega} && (2) \\
&= \int_{\mathbb{R}^p} \int_{b=0}^{2\pi} \frac{P(\boldsymbol{\omega})}{2\pi} 2\cos(\boldsymbol{\omega}^T\mathbf{x} + b)\cos(\boldsymbol{\omega}^T\mathbf{x}' + b) d\boldsymbol{\omega} db && (3) \\
&= \mathbb{E}_{\boldsymbol{\omega}\sim P, b\sim\mathcal{U}(0,2\pi)} \left[\sqrt{2}\cos(\boldsymbol{\omega}^T\mathbf{x} + b)\sqrt{2}\cos(\boldsymbol{\omega}^T\mathbf{x}' + b)\right] && (4)
\end{aligned}$$

(1): $K(\mathbf{x}-\mathbf{x}'), P(\boldsymbol{\omega}) \in \mathbb{R}$ so we can ignore imaginary part

(2) and (3): use sum of angles formulas

- We have obtained

$$K(\mathbf{x}-\mathbf{x}') = \mathbb{E}_{\boldsymbol{\omega}\sim P, b\sim\mathcal{U}(0,2\pi)}\left[\sqrt{2}\cos(\boldsymbol{\omega}^T\mathbf{x}+b)\sqrt{2}\cos(\boldsymbol{\omega}^T\mathbf{x}'+b)\right]$$

- K can thus be written as an expectation over $\boldsymbol{\omega}$ drawn from the distribution P
- If we know how to sample from P (the Fourier transform of K), we can approximate K by random sampling

- P is given by the (scaled) Fourier transform of $K(\mathbf{\Delta})$

$$P(\boldsymbol{\omega}) = \frac{1}{(2\pi)^p} \int_{\mathbb{R}^p} K(\mathbf{\Delta}) e^{-i\boldsymbol{\omega}^T \mathbf{\Delta}} d\mathbf{\Delta}$$

- For the Gaussian RBF kernel, P is a Gaussian distribution

$$\begin{aligned}
P^{rbf}(\boldsymbol{\omega}) &= \frac{1}{(2\pi)^p} \int_{\mathbb{R}^p} e^{-\gamma \|\mathbf{\Delta}\|_2^2} e^{-i\boldsymbol{\omega}^T \mathbf{\Delta}} d\mathbf{\Delta} \\
&= \frac{1}{\sqrt{(2\pi)^p 2\gamma}} \int_{\mathbb{R}^p} \frac{1}{\sqrt{(2\pi)^p \frac{1}{2\gamma}}} e^{-\gamma \|\mathbf{\Delta}\|_2^2} e^{-i\boldsymbol{\omega}^T \mathbf{\Delta}} d\mathbf{\Delta} \\
&= \frac{1}{\sqrt{(2\pi)^p 2\gamma}} e^{\frac{\|\boldsymbol{\omega}\|_2^2}{4\gamma}} = \mathcal{N}(0, 2\gamma \mathbf{I}_p)
\end{aligned}$$

- For the Laplace kernel, P is a Cauchy distribution

$$P^{lap}(\boldsymbol{\omega}) = \prod_p \frac{1}{\pi(1 + \omega_p^2 / 2\gamma)}$$

RANDOM KERNEL FEATURES: METHOD

1. Set number of random kernel features c
2. Draw $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_c \sim P(\boldsymbol{\omega})$ and $b_1, \ldots, b_c \sim \mathcal{U}(0, 2\pi)$
3. Map training points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^p$ to their random kernel features $\hat{\phi}(\mathbf{x}_1), \ldots, \hat{\phi}(\mathbf{x}_n) \in \mathbb{R}^c$ where

$$\hat{\phi}_j(\mathbf{x}) = \sqrt{\frac{2}{c}} \cos(\boldsymbol{\omega}_j^T \mathbf{x} + b_j), \quad j \in \{1, \ldots, c\}$$

4. Train linear model (such as linear SVM) on transformed data $\hat{\phi}(\mathbf{x}_1), \ldots, \hat{\phi}(\mathbf{x}_n) \in \mathbb{R}^c$

RANDOM KERNEL FEATURES: APPROXIMATION BOUND

Theorem ([Rahimi and Recht, 2007])

Let $c \geq 1$ and $\hat{\phi} : \mathbb{R}^p \rightarrow \mathbb{R}^c$ be the feature map obtained by drawing $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_c$ from $P(\boldsymbol{\omega})$. Then we have with high probability:

$$\sup_{x, x' \in \mathcal{X}} \left| \left\langle \hat{\phi}(x), \hat{\phi}(x') \right\rangle - K(x, x') \right| \leq O\left(\sqrt{\frac{p}{c}} \right)$$

- Kernel approximation error uniformly decreases in $O(\sqrt{1/c})$
- Can also bound generalization error [Rahimi and Recht, 2008]

ADDITIONAL COMMENTS

- Example of large-scale application: acoustic models for speech recognition [Lu et al., 2016]
    - Trained on 50 hours of speech with $c = 500K$ random features
    - Performance comparable to deep neural nets
- Random features exist for other kernels, such as dot product kernels (including polynomial kernels) [Kar and Karnick, 2012]
- Techniques to speed up prediction further [Le et al., 2013]
    - From $O(cp)$ to $O(c \log p)$ time
- Easy to combine multiple kernels by stacking their random features [Lu et al., 2014]

NYSTRÖM APPROXIMATION

- Let us consider the Gram matrix $\boldsymbol{G} \in \mathbb{R}^{n \times n}$ ($G_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$)
- When n is large, could approximate $\boldsymbol{G}$ with a matrix of rank $k \leq n$
- Consider the spectral decomposition of $\boldsymbol{G}$

$$\boldsymbol{G} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$$

  - $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n]^T \in \mathbb{R}^{n \times n}$ the set of eigenvectors
  - $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ the eigenvalues ($\lambda_1 \geq \cdots \geq \lambda_n$)

- Best rank-k approximation $\boldsymbol{G}_k \in \mathbb{R}^{n \times n}$ is given by

$$\boldsymbol{G}_k = \boldsymbol{U}_k\boldsymbol{\Lambda}_k\boldsymbol{U}_k^T$$

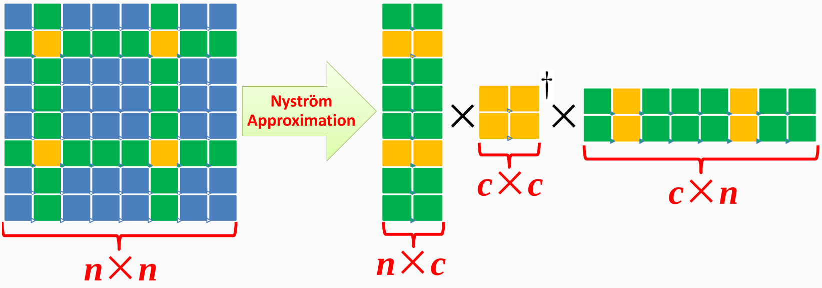
where $\boldsymbol{U}_k = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k]^T \in \mathbb{R}^{n \times k}$ and $\boldsymbol{\Lambda}_k = \mathrm{diag}(\lambda_1, \ldots, \lambda_k)$

- In the context of kernel methods this is useless
  - We still need to construct $\boldsymbol{G}$: $O(n^2)$ time
  - We need to compute its k-thresholded spectral decomposition $\boldsymbol{G}_k$: $O(n^2)$ to $O(n^3)$ time depending on the value of k
- **Goal**: find good approximation $\hat{\boldsymbol{G}}_k$ of $\boldsymbol{G}_k$ in $O(n)$ time

- Nyström approximation [Drineas and Mahoney, 2005]:

$$\hat{G}_k = CW^{\dagger}C^T$$

where $C \in \mathbb{R}^{n \times c}$, $W \in \mathbb{R}^{c \times c}$ and $W^{\dagger}$ is the pseudo-inverse of W



- For intuition: think of approximation of distances between cities

NYSTRÖM APPROXIMATION: GENERAL ALGORITHM

1. Sample a set $\mathcal{I}$ of c indices uniformly in $\{1, \ldots, n\}$
2. Compute $\boldsymbol{C} \in \mathbb{R}^{n \times c}$ with $\boldsymbol{C}_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for $i \in \{1, \ldots, n\}$ and $j \in \mathcal{I}$
3. Form matrix $\boldsymbol{W} \in \mathbb{R}^{c \times c}$ with $\boldsymbol{W}_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for $i, j \in \mathcal{I}$
4. Compute $\boldsymbol{W}_k \in \mathbb{R}^{c \times c}$, the best rank-$k$ approximation of $\boldsymbol{W}$ ($k \leq c$)
5. Final rank k approximation of $\boldsymbol{G}$:

$$\hat{\boldsymbol{G}}_k = \boldsymbol{C}\boldsymbol{W}_k^{\dagger}\boldsymbol{C}^T \in \mathbb{R}^{n \times n}$$

   where $\boldsymbol{W}_k^{\dagger}$ is the Moore-Penrose pseudoinverse of $\boldsymbol{W}_k$

Time complexity: $O(c^3 + nck)$

Theorem ([Drineas and Mahoney, 2005])

Let $\mathbf{G} \in \mathbb{R}^{n \times n}$ be the Gram matrix. Let $\mathbf{G}_k$ be its best rank-k approximation and $\hat{\mathbf{G}}_k$ be its Nyström approximation. We have with high probability:

$$\|\mathbf{G} - \hat{\mathbf{G}}_k\|_F \leq \|\mathbf{G} - \mathbf{G}_k\|_F + O\left(\sqrt{\frac{n}{c}}\right)$$

- If c is large enough, $\hat{\mathbf{G}}_k$ is nearly as good as $\mathbf{G}_k$
- Setting $k < c$ removes the noise contained in smallest eigenvalues of $\mathbf{W}$
- Some nonuniform sampling techniques have been proposed, but uniform sampling tends to work best in practice [Kumar et al., 2009]

- $\hat{G}_k$ can be used directly to speed up training algorithms
- It can also be used to generate an explicit feature map as in RKF
- Recall that $\hat{G}_k = CW_k^{\dagger}C^T$, denote $W_k^{\dagger} = V_k \boldsymbol{\Sigma}_k V_k^T$
- For training points we have an explicit feature map given by:

$$\hat{\phi}(x_i) = C_i V_k \boldsymbol{\Sigma}_k^{1/2}$$

- Natural extension to unseen points:

$$\hat{\phi}(x) = \boldsymbol{k}_{x,\mathcal{I}} V_k \boldsymbol{\Sigma}_k^{1/2}$$

where $\boldsymbol{k}_{x,\mathcal{I}} = [K(x, x_i)]_{i \in \mathcal{I}} \in \mathbb{R}^c$

CONCLUSION

NYSTRÖM APPROXIMATION *VS.* RANDOM KERNEL FEATURES

- Both methods have a $O(1/\sqrt{c})$ convergence rate (c: # of random features for RKF, # of random columns for Nyström)
- Nyström' approximation guarantee is adaptive to the data, while RKF is data-independent
    - In fact, Nyström can achieve $O(1/c)$ convergence when eigengap of $\boldsymbol{G}$ is large [Yang et al., 2012]
- At equal number of random samples (features/columns)
    - Nyström tends to achieve better performance
    - But RKF are generally cheaper to generate (no spectral decomposition or matrix inversion needed)

- Kernel methods: general class of nonlinear algorithms
- Training and prediction time scales badly with n
- Two general techniques to make kernel methods scalable: Random Kernel Features and Nyström approximation
- We can then take advantage of existing fast solvers for linear algorithms

[Drineas and Mahoney, 2005] Drineas, P. and Mahoney, M. W. (2005).
On the nyström method for approximating a gram matrix for improved kernel-based learning.
Journal of Machine Learning Research, 6:2153–2175.

[Gärtner, 2003] Gärtner, T. (2003).
A survey of kernels for structured data.
SIGKDD Explorations, 5(1):49–58.

[Haussler, 1999] Haussler, D. (1999).
Convolution Kernels on Discrete Structure.
Technical Report UCSC-CRL-99-10, University of California at Santa Cruz.

[Kar and Karnick, 2012] Kar, P. and Karnick, H. (2012).
Random feature maps for dot product kernels.
In *AISTATS*, pages 583–591.

[Kumar et al., 2009] Kumar, S., Mohri, M., and Talwalkar, A. (2009).
Sampling techniques for the nystrom method.
In *AISTATS*, pages 304–311.

[Le et al., 2013] Le, Q., Sarlós, T., and Smola, A. (2013).
Fastfood — Approximating Kernel Expansions in Loglinear Time.
In *ICML*.

[Lu et al., 2016] Lu, Z., Guo, D., Bagheri Garakani, A., Liu, K., May, A., Bellet, A., Fan, L., Collins, M., Kingsbury, B., Picheny, M., and Sha, F. (2016).
A Comparison Between Deep Neural Nets and Kernel Acoustic Models for Speech Recognition.
In *ICASSP*.

[Lu et al., 2014] Lu, Z., May, A., Liu, K., Bagheri Garakani, A., Guo, D., Bellet, A., Fan, L., Collins, M., Kingsbury, B., Picheny, M., and Sha, F. (2014).
How to Scale Up Kernel Methods to Be As Good As Deep Neural Nets.
Technical report, arXiv:1411.4000.

[Rahimi and Recht, 2007] Rahimi, A. and Recht, B. (2007).
Random Features for Large-Scale Kernel Machines.
In *NIPS*.

[Rahimi and Recht, 2008] Rahimi, A. and Recht, B. (2008).
Weighted Sums of Random Kitchen Sinks: Replacing minimization with randomization in learning.
In *NIPS*, pages 1313–1320.

[Shalev-Shwartz et al., 2011] Shalev-Shwartz, S., Singer, Y., Srebro, N., and Cotter, A. (2011).
Pegasos: Primal Estimated sub-GrAdient SOlver for SVM.
Mathematical Programming, 127(1):3–30.

[Shin and Kuboyama, 2008] Shin, K. and Kuboyama, T. (2008).
A generalization of Haussler's convolution kernel: mapping kernel.
In *ICML*, pages 944–951.

[Steinwart, 2003] Steinwart, I. (2003).
Sparseness of Support Vector Machines.
Journal of Machine Learning Research, 4:1071–1105.

[Yang et al., 2012] Yang, T., Li, Y.-F., Mahdavi, M., Jin, R., and Zhou, Z.-H. (2012).
Nyström method vs random fourier features: A theoretical and empirical comparison.
In *NIPS*, pages 485–493.