STAT G8325
Gaussian Processes and Kernel Methods
§11: Speed and Scaling Part 3

John P. Cunningham

Department of Statistics
Columbia University

Outline

Outline

Administrative interlude

Setup

Random Fourier Features [RR07]

Random Binning Features [RR07]

Results

References

Progress...

Week	Lectures	Content
10	Nov 23, Dec 2	Kernel statistical tests
11	Dec 7	Speed and Scaling Part 3 • [RR07] (intentionally light reading; work on projects)
12	Dec 9	Probabilistic Integration? Random kitchen sinks / fastfood?

- Final project presentations Monday Dec 14, 16
  - Present 5-7 minutes of your project results.
  - Build off of project progress report.
  - Send 1-5 pdf slides to me beforehand.
- Can everyone make Wed Dec 16?
- Final project writeup then due Friday Dec 18 at noon.
  - 8-16 pages pdf, using the tex template from hw3.
  - Deadline strictly enforced.

Outline

Speed and scaling of kernel methods

- §05 and §06 discussed speed and scaling gp methods.
- All boiled down to kernel approximations (the key bottleneck).
- No surprise, kernel methods more generally have scaling methods.
- In some kernel methods (e.g. SVM), K^{-1} is not required, but even still $\mathcal{O}(n^2)$ runtime and storage is burdensome.
- Setup:
  - $\mathcal{X} = \mathbb{R}^d$.
  - $k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$ in the usual way.
  - k stationary: $k(x, x') = k(x - x')$.
- We have some kernel machine admitting the representer theorem:

$$
\begin{aligned}
f(x^*) &= \sum_{i=1}^{n} \alpha_i k(x_i, x^*) \\
e.g. \quad & K_{f^*f}(K_{ff} + \sigma^2 I)^{-1} y \\
&= \sum_{i=1}^{n} \left[(K_{ff} + \sigma^2 I)^{-1} y \right]_i k(x_i, x^*).
\end{aligned}
$$

Speed and scaling of kernel methods

- We have some kernel machine obeying the representer theorem:

$$f(x) \;=\; \sum_{i=1}^{n} \alpha_i k(x_i, x).$$

- Prediction has cost $\mathcal{O}(nd)$; in the large n setting, even this is burdensome.
- The essential idea of [RR07] is to approximate:

$$k(x, x') \;=\; \langle \phi(x), \phi(x') \rangle_{\mathcal{H}} \;\approx\; z(x)^\top z(x')$$

  for some approximating (randomized) feature map $z : \mathbb{R}^d \to \mathbb{R}^D$.
- Note that again we have a low-rank kernel approximation $K \approx Z^\top Z$.
- The subsequent kernel machine then becomes linear in the z feature space:

$$\begin{aligned} f(x) &= \sum_{i=1}^{n} \alpha_i k(x_i, x) \\ &\approx w^\top z(x), \end{aligned}$$

  which is an inner product in $\mathbb{R}^D$ with resulting cost $\mathcal{O}(D + d)$.

Outline

Random Fourier Features

- The essential question is then how to choose that feature map $z : \mathbb{R}^d \to \mathbb{R}^D$.
- Reminder (§04; Bochner): $k(x, x') = k(x - x') = k(\tau)$ is positive definite $\Leftrightarrow$

  $$\gamma p(\omega) = \mathcal{F}\{k\}(\omega) \geq 0 \;\; \forall \;\; \omega.$$

  In other words, the power spectral density $p(\omega)$ is nonnegative everywhere.
- I write $\gamma p(\omega)$ to clarify that $p(\omega)$ is a pdf from which we can sample frequencies. Hereafter assume k normalized such that $\gamma = 1$ (wlog).
- Reminder: a real and even function $k(\tau)$ has:

  $$\begin{aligned} p(\omega) &= \int k(\tau) \exp\left\{-2\pi i \omega^\top \tau\right\} d\tau \\ &= \int k(\tau) \cos(2\pi\omega^\top \tau) d\tau \\ &\quad \text{... and similarly...} \\ k(\tau) &= \int p(\omega) \cos(2\pi\omega^\top \tau) d\omega \\ &= E_p\left(\cos(2\pi\omega^\top \tau)\right). \end{aligned}$$

- Idea: an unbiased estimate of $k(\tau)$ is gained from sampling from $p(\omega)$...

Random Fourier Features

- Standard trigonometric identities show:

$$\begin{aligned} k(\tau) &= \int p(\omega)\cos\left(2\pi\omega^\top\tau\right)d\omega \\ &= E_{p(\omega)}\left(\cos\left(2\pi\omega^\top\tau\right)\right). \\ &= E_{p(\omega)}\left(\cos\left(2\pi\omega^\top(x-x')\right)\right). \\ &= E_{p(\omega)}E_{U(0,2\pi)}\left(2\cos\left(\omega^\top x+b\right)\cos\left(\omega^\top x'+b\right)\right). \end{aligned}$$

  where $b \sim U(0, 2\pi)$ is the uniform distribution.

- Random fourier features thus defines the approximate feature map:

$$z(x) = \begin{bmatrix} \sqrt{2/D}\cos\left(2\pi\omega_1^\top x' + b_1\right) \\ \vdots \\ \sqrt{2/D}\cos\left(2\pi\omega_D^\top x' + b_D\right) \end{bmatrix}$$

  where $\omega_1, ..., \omega_D \sim_{iid} p(\omega)$ and $b_1, ..., b_D \sim_{iid} U(0, 2\pi)$.

- Then $z(x)^\top z(x') = \frac{1}{D}\sum_{k=1}^{D} z_{\omega_k}(x) z_{\omega_k}(x')$ is an unbiased estimate of $k(\tau)$.

Random Fourier Features

- $k_{rff}(x, x') = z(x)^\top z(x') = \frac{1}{D}\sum_{k=1}^{D} z_{\omega_k}(x) z_{\omega_k}(x') \approx k(\tau)$, where

$$z(x) = \begin{bmatrix} \sqrt{2/D}\cos\left(2\pi\omega_1^\top x' + b_1\right) \\ \vdots \\ \sqrt{2/D}\cos\left(2\pi\omega_D^\top x' + b_D\right) \end{bmatrix}$$

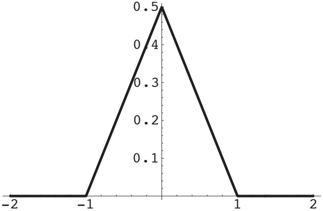
  where $\omega_1, ..., \omega_D \sim_{iid} p(\omega)$ and $b_1, ..., b_D \sim_{iid} U(0, 2\pi)$.

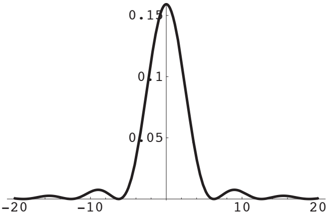
- $k(x, x')$ is approximated to within ϵ with $D = \mathcal{O}(d\epsilon^{-2}\log\epsilon^{-2})$ [RR07].
- RFF replaces a kernel with a low-rank kernel $K \approx Z^\top Z$.
- Allows one to train a linear kernel in the feature space of size D.
- This method is heavily used with good results.

Outline

Random Binning Features

- Random fourier features compared x and x' in terms of how close they are in the $\cos(2\pi\omega^\top(x - x'))$ sense.
- Random binning features asks if x and x' live in the same bin after randomly gridding input space R^d.
- To understand why this is a sensible idea, consider the univariate triangle kernel $k(x, x') = \max(0, 1 - \frac{1}{\delta}|x - x'|)$:





- Notice if we grid up $\mathbb{R}$ with width ('pitch') δ:

$$
\begin{aligned}
k(x, x') = 0 &\quad \Leftrightarrow \quad x, x' \text{ are in different bins.} \\
k(x, x') > 0 &\quad \Leftrightarrow \quad x, x' \text{ are in the same bin.}
\end{aligned}
$$

Random Binning Features

- Next grid $\mathbb{R}$ with some random shift $u \sim U(0, \delta)$, so bins are:

$$[u + n\delta, u + (n+1)\delta].$$

- Notice then that x, x' are in bins $i = \left\lfloor \frac{x-u}{\delta} \right\rfloor$, $i' = \left\lfloor \frac{x'-u}{\delta} \right\rfloor$.
- Define $z(x)$ such that $z(x)^\top z(x') = 1 \Leftrightarrow x, x'$ are in the same bin.
- This feature vector is simply an indicator vector $z(x) = \{\mathbb{1}(\ x \text{ is in bin i }) \}_i$.
- Now notice, for a given δ:

$$\begin{aligned} E_u\left(z(x)^\top z(x')\right) &= Prob\left(z(x)^\top z(x') = 1\right) \\ &= Prob\left(i = i'\right) \\ &= \max\left(0, 1 - \frac{|x - x'|}{\delta}\right) \\ &= k(x, x') \end{aligned}$$

...where the third line is either from the convolution of two uniform r.v.'s or basic reasoning.

- Then D random grid shifts yields an unbiased estimator of $k(x, x')$.

Random Binning Features

- With the triangle $k_\triangle(x, x'; \delta) = E_u(z(x)^\top z(x')|\delta)$, we can consider a more general class of kernels:

$$k(x, x') = \int k_\triangle(x, x'; \delta) p(\delta) d\delta.$$

- This binning trick can then be extended by the law of total expectation:

$$k(x, x') = E_\delta \left(E_u \left(z(x)^\top z(x')|\delta \right) \right).$$

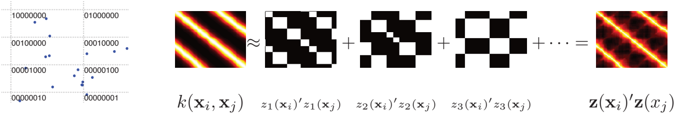
- The multivariate Laplace kernel is such an example. The RBF kernel is not.

This is sensible, as binning feels more ℓ_1 than ℓ_2.

- [RR07] shows that $p(\delta) = \delta \frac{d^2}{d\delta^2} k(\delta)$ recovers p from k.
- Laplace: $p(\delta) = \delta \exp(-\delta)$ for $k(x, x') = \exp(-|x - x'|)$.

Random Binning Features

- An example picture of random binning features:



Outline

Random Binning Features

- Some results:

Dataset	Fourier+LS	Binning+LS	CVM	Exact SVM
CPU regression 6500 instances 21 dims	3.6% 20 secs $D = 300$	5.3% 3 mins $P = 350$	5.5% 51 secs	11% 31 secs ASVM
Census regression 18,000 instances 119 dims	5% 36 secs $D = 500$	7.5% 19 mins $P = 30$	8.8% 7.5 mins	9% 13 mins SVMTorch
Adult classification 32,000 instances 123 dims	14.9% 9 secs $D = 500$	15.3% 1.5 mins $P = 30$	14.8% 73 mins	15.1% 7 mins $\text{SVM}^{\text{light}}$
Forest Cover classification 522,000 instances 54 dims	11.6% 71 mins $D = 5000$	2.2% 25 mins $P = 50$	2.3% 7.5 hrs	2.2% 44 hrs libSVM
KDDCUP99 (see footnote) classification 4,900,000 instances 127 dims	7.3% 1.5 min $D = 50$	7.3% 35 mins $P = 10$	6.2% (18%) 1.4 secs (20 secs)	8.3% < 1 s SVM+sampling

- Extensions include [RR09] and [LSS13], both of which are interesting...

Outline

References

[LSS13] Quoc Le, Tamás Sarlós, and Alex Smola.
Fastfood-approximating kernel expansions in loglinear time.
In *Proceedings of the international conference on machine learning*, 2013.

[RR07] Ali Rahimi and Benjamin Recht.
Random features for large-scale kernel machines.
In *Advances in neural information processing systems*, pages 1177–1184, 2007.

[RR09] Ali Rahimi and Benjamin Recht.
Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning.
In *Advances in neural information processing systems*, pages 1313–1320, 2009.