Sketching Algorithms for Big Data Fall 2017

Lecture 24 — November 28, 2017

Christopher Musco *Scribe: Akshat Agrawal*

1 Overview

In this lecture MIT graduate student Chris Musco went over a technique known as random feature maps for Kernel learning. The outline is as follows:

1. Review of Kernel Methods
2. Rahimi-Recht Algorithm
3. Cleanup, improvements

2 Review of Kernel Methods

Kernel methods turn "linear" learning algorithms into nonlinear ones. Examples of such are algorithms are:

- Linear Regression
- Support Vector Machine
- Principal Components Analysis (Linear Dimensionality Reduction)

In this lecture we use regression as the example.

2.1 Overview of Regression

Goal: Given $A \in \mathbb{R}^{n \times d}$, $b \in \mathbb{R}^n$, we wish to learn a function f such that $f(a_i) \approx b_i$. We assume that $f(z) = x^T z$ and we wish to minimize $\min_{x \in \mathbb{R}^d} ||Ax - b||_2$.

What happens if we want to learn polynomial f? We have two options.

2.2 Learning Nonlinear Mappings

Option 1: Explicit Basis Functions

Suppose we want to learn quadratic f. Given

$$z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_d \end{bmatrix} \in \mathbb{R}^d$$

we let the feature map be

$$\phi(z) = \begin{bmatrix} z_1 \\ z_1^2 \\ z_1 z_2 \\ \vdots \\ z_d^2 \end{bmatrix} \in \mathbb{R}^{d^2+d}$$

We then run normal linear regression on $\phi(z)$. The problem is that as you increase dimension of the polynomial f this quickly becomes infeasible as the vector becomes too large.

Option 2: Kernel Trick

Two observations:

1. For linear learning, we need access only to all **pairwise dot products** $\langle a_i, a_j \rangle$
2. We can compute these pairwise dot products faster than through explicit feature computation

We first prove the first observation:

Proof.

$$\min_{x \in \mathbb{R}^d} ||Ax - b||_2 = \min_{y \in \mathbb{R}^n} ||AA^T y - b||_2$$

Note that we can let $x = A^T y$ for some y since x (under optimality) needs to be in the rowspan of A. The matrix $K := AA^T$ is called the **Kernel matrix** with $K_{ij} = \langle a_i, a_j \rangle$ as desired. □

Next we prove the second observation.

Proof. Consider quadratic f for clarity:

$$\begin{aligned}\langle \phi(x), \phi(y) \rangle &= \begin{bmatrix} 1 \\ \sqrt{2}x_1 \\ x_1^2 \\ \sqrt{2}x_1x_2 \\ \vdots \end{bmatrix} \begin{bmatrix} 1 \\ \sqrt{2}y_1 \\ y_1^2 \\ \sqrt{2}y_1y_2 \\ \vdots \end{bmatrix} \\ &= 1 + x_1y_1 + 2x_1^2y_1^2 + 2x_1x_2y_1y_2 + \cdots \\ &= (x_1y_1 + x_2y_2 + \cdots + x_dy_d + 1)^2 \\ &= (\langle x, y \rangle + 1)^2\end{aligned}$$

It extends further that for a polynomial of degree q, the dot product is given by $(\langle x, y \rangle + 1)^q$. This is called the **Kernel function**. In this case it can be computed in $O(d)$ time. □

A few examples of notable Kernel functions include:

- Gaussian Kernel: $K(x, y) = e^{-||x-y||^2}$
- Exponential Kernel: $K(x, y) = e^{-||x-y||}$
- Laplacian Kernel: $K(x, y) = e^{-||x-y||_1}$

These Kernels are all **shift invariant**, i.e. they only depend on $\Delta := x - y$. They define a sort of similarity score that goes towards 0 if x, y are far apart, and towards 1 if they are close.

3 Rahimi-Recht Algorithm

Now onto algorithms. Since linear regression with Kernels requires computing and inverting K as the major operations, we should note the following complexities:

- Constructing K: $O(n^2d)$
  - This is a problem, and the subject of what follows
- Inverting K: $O(n^3)$
  - This can be made faster using methods we have seen previously, such as iterative algorithms, sketching, etc.

3.1 Rahimi-Recht for a Gaussian Kernel

What follows comes from the 2007 NIPS paper by Ali Rahimi and Benjamin Recht [1].

Goal: For a positive definite shift invariant kernel function, give a rank $\frac{\log n}{\epsilon^2}$ approximation to K.

This is done by producing a mapping from A to Z where Z has dimensions n by $\frac{\log n}{\epsilon^2}$, and $ZZ^T \approx K$. The algorithm leads to overall $O(nd\frac{\log n}{\epsilon^2})$ compute time for Z. One can also then invert ZZ^T (use the SVD to see this) in time $O(n\frac{\log^2 n}{\epsilon^4})$ time. The authors proved the claim for all PD shift invariant Kernels, but we restrict ourselves to Gaussian Kernels for simplicity.

We use the Fourier Transform for a Multidimensional Gaussian and compute:

$$\begin{aligned}\phi(x)^T\phi(y) &= e^{-||\Delta||^2} \\ &= \int_{\mathbb{R}} \pi^{d/2} e^{-||\eta||^2\pi^2} e^{-2\pi i\eta^T\Delta} d\eta \\ &= \int_{\mathbb{R}} g(\eta) e^{-2\pi i\eta^T\Delta} d\eta \\ &= \mathbb{E}_{\eta\sim g}[e^{-2\pi i\eta^T\Delta}]\end{aligned}$$

Where $g(\eta) > 0\ \forall\eta$ is a valid probability density function (to see why just consider the expression when $\Delta = 0$). Note that **Bochner's theorem** states that $g \geq 0$ for all shift invariant positive definite kernel functions, and this allows the proof to generalize beyond Gaussians. We note also that g is a multivariate Gaussian, and so we can sample from g efficiently.

By Monte Carlo Integration, we take m independent samples of η from g and approximate:

$$\begin{aligned}\mathbb{E}_{\eta\sim g}[e^{-2\pi i\eta^T\Delta}] &\approx \frac{1}{m}\sum_{j=1}^{m} e^{-2\pi i\eta_j^T\Delta} \\ &= \frac{1}{m}\sum_{j=1}^{m} e^{-2\pi i\eta_j^T x} e^{-2\pi i\eta_j^T(-y)} \\ &= \langle\tilde{\phi}(x), \tilde{\phi}(y)\rangle\end{aligned}$$

Where we use the complex inner product in the last line and

$$\tilde{\phi}(x) = \begin{bmatrix} \frac{1}{\sqrt{m}} e^{-2\pi i\eta_1^T x} \\ \vdots \\ \frac{1}{\sqrt{m}} e^{-2\pi i\eta_m^T x} \end{bmatrix}$$

Claim: If $m = O(\frac{\log\frac{1}{\delta}}{\epsilon^2})$, then with probability at least $1-\delta$, $\langle\tilde{\phi}(x), \tilde{\phi}(y)\rangle \in [K(x,y)-\epsilon, K(x,y)+\epsilon]$. The proof follows directly from a simple complex-number extension to Chernoff, noting that each term in the sum has norm 1.

4 Cleanup, Improvements

4.1 Removing Complex Numbers

We note that since imaginary terms have 0 expectation:

$$\begin{aligned}\mathbb{E}_{\eta\sim g}\Big[e^{-2\pi i\eta^T x}e^{-2\pi i\eta^T(-y)}\Big] &= \mathbb{E}\Big[(\cos(-2\pi\eta^T x)+i\sin(-2\pi\eta^T x))(\cos(-2\pi\eta^T y)+i\sin(2\pi\eta^T y))\Big]\\ &= \mathbb{E}\Big[\cos(2\pi\eta^T x)\cos(2\pi\eta^T y)+\sin(2\pi\eta^T x)\sin(2\pi\eta^T y)\Big]\end{aligned}$$

It then follows that we can let

$$\tilde{\phi}(x)=\frac{1}{\sqrt{m}}\begin{bmatrix}\cos(2\pi\eta_1^T x)\\ \sin(2\pi\eta_1^T x)\\ \vdots\\ \cos(2\pi\eta_m^T x)\\ \sin(2\pi\eta_m^T x)\end{bmatrix}$$

4.2 Faster Multiplication by Gaussians

Note that a significant bottleneck is that to generate the map $\tilde{\phi}(x)$, one must multiply a d-dimensional vector x with m random multivariate Gaussians (which forms a random Gaussian matrix), which is a runtime of $O(dm)$. The question arises as to if one can apply the random Gaussian faster. The answer is yes. The **Fastfood Embeddings** developed by Le, Sarlos, and Smola [2] approximately apply the Gaussian in $O(\max\{m,d\}\log d)$ time, and one can still recover the same probabilisitic guarantee with this approach.

Another relevant paper is Kaprelov, Potluru, and Woodruff's "How to Fake Multiply by a Gaussian" [3]. It seems there is still significant room for improvement in this domain.

4.3 Additive Error Analyis

What is the deal with additive error on each entry?
We have that $\tilde{K}\pm\epsilon=K$, however, this may not be good enough, as if each entry is off by ϵ, then $||\tilde{K}-K||_F\leq\epsilon n$, and this could be huge. With Chernoff, we can show that if $\epsilon=\frac{1}{\sqrt{n}}$ and $m\approx n$, $||\tilde{K}-K||\leq\sqrt{n}$, where we are taking the spectral norm. However, using stronger Matrix Norm concentration inequalities, we can get the same bound using $m\approx\sqrt{n}$ samples.

References

[1] Ali Rahimi and Benjamin Recht. Random Features for Large-Scale Kernel Machines. *Advances in Neural Information Processing Systems*, 20:1177–1184, 2007.

[2] Quoc V. Le and Tamás Sarlós and Alexander J. Smola. Fastfood - Computing Hilbert Space Expansions in loglinear time. *Proceedings of the 30th International Conference on Machine Learning*, 30:244–252, 2013.

[3] Michael Kapralov and Vamsi K. Potluru and David P. Woodruff. How to Fake Multiply by a Gaussian Matrix. *Proceedings of the 33nd International Conference on Machine Learning*, 33: 2101–2110, 2016.