Explicit Feature Methods for Accelerated Kernel Learning

Purushottam Kar

Quick Motivation

- **Kernel Algorithms (SVM, SVR, KPCA) have output**

$$h(x) = \sum_{i=1}^{n} \alpha_i K(x, x_i)$$

- **Number of "support vectors" is typically large**
  - **Provably a constant fraction of training set size***
  - **Prediction time $\Omega(nd)$ where $x_i \in \mathcal{X} \subset \mathbb{R}^d$**
  - **Slow for real time applications**

***[Steinwart NIPS 03, Steinwart-Christmann. NIPS 08]**

The General Idea

- **Approximate kernel using explicit feature maps**

$$Z: \mathcal{X} \longrightarrow \mathbb{R}^D \text{ s.t. } K(x, x_i) \approx Z(x)^\top Z(x_i)$$

- **Speeds up prediction time to $\mathrm{O}(Dd) \ll \mathrm{O}(nd)$**

$$h(x) \approx \sum_{i=1}^{n} \alpha_i \langle Z(x), Z(x_i) \rangle = Z(x)^\top w$$

$$w = \sum_{i=1}^{n} \alpha_i Z(x_i)$$

- **Speeds up training time as well**

Why Should Such Maps Exist?

- **Mercer's theorem***
  Every PSD kernel K has the following expansion

$$K(x,y) = \sum_{i=0}^{\infty} \lambda_i \Phi_i(x)\Phi_i(y)$$

- **The series converges *uniformly* to kernel**
  - **For every $\epsilon > 0, \exists D_\epsilon$ such that if we construct the map**

$$Z_{D_\epsilon} = \left(\Phi_1, \Phi_2, \dots, \Phi_{D_\epsilon}\right) \in \mathbb{R}^{D_\epsilon},$$

  - **then for all $x, y \in X$**

$$\left|K(x,y) - Z_{D_\epsilon}(x)^\top Z_{D_\epsilon}(y)\right| \leq \epsilon$$

  - **Call such maps *uniformly ϵ*-approximate**

***[Mercer 09]**

Today's Agenda

- **Some explicit feature map constructions**
  - **Randomized feature maps**
    e.g. Translation invariant, rotation invariant
  - **Deterministic feature maps**
    e.g. Intersection, scale invariant

- **Some "fast" random feature constructions**
  - **Translation invariant, dot product**

- **The BIG picture?**

Random Feature Maps

Approximate recovery of kernel values with high confidence

Translation Invariant Kernels*

- **Kernels of the form $K(x, y) = K(x - y)$**
  - **Gaussian kernel, Laplacian kernel**
- **Bochner's Theorem****

  For every K there exists a positive function p

$$K(x-y) = \int_{\omega \in \widehat{X}} \cos\left(\omega^\top (x-y)\right) p(\omega) d\omega$$
$$= \mathrm{E}_{\omega \sim p} \left[\!\left[\cos\left(\omega^\top (x-y)\right) \right]\!\right]$$

- **Finding p: take inverse Fourier transform of K**
- **Select $\omega_i \sim p$ for $i = 1, \ldots, D$**

$$Z_i : x \mapsto \left(\cos\left(\omega_i^\top x\right), \sin\left(\omega_i^\top x\right)\right)$$

***[Rahimi-Recht NIPS 07], ** Special case for $\mathcal{X} \subset \mathbb{R}^d$, [Bochner 33]**

Translation Invariant Kernels

- ***Empirical averages* approximate expectations**
- **Let $Z: x \mapsto \left(Z_1(x), Z_2(x), \ldots, Z_D(x)\right)$**

$$
\begin{aligned}
Z(x)^\top Z(y) &= \frac{1}{D}\sum_{i=1}^{D} Z_i(x)^\top Z_i(y) \\
&= \frac{1}{D}\sum_{i=1}^{D} \cos\left(\omega_i^\top (x-y)\right) \\
&\approx \mathrm{E}_{\omega \sim p}\left[\!\left[\cos\left(\omega^\top (x-y)\right)\right]\!\right] \\
&= K(x-y)
\end{aligned}
$$

- **Let us assume points $x, y \in \mathcal{B}(\mathbf{0}, R) \subset \mathbb{R}^d$**
  Then we require $D \geq \frac{30d}{\epsilon^2} \log\left(\frac{RC_K}{\delta\epsilon}\right)$
  C_K depends on spectrum of kernel K

Translation Invariant Kernels

- **For the RBF Kernel**

$$K(x, y) = \exp\left(\frac{-\|x - y\|_2^2}{2}\right)$$

$$p(\omega) = \frac{1}{(2\pi)^{d/2}} \exp\left(\frac{-\|\omega\|_2^2}{2}\right)$$

- **If kernel K offers a γ margin, then we should require**

$$D \gtrsim \frac{30d}{\gamma^2} \log\left(\frac{Rd}{\delta\gamma}\right)$$

Here $C_K \approx d$ where $x, y \in \mathbb{R}^d$

Rotation Invariant Kernels*

- **Kernels of the form $K(x, y) = K(x^\top y)$**
  - **Polynomial kernels, exponential kernel**
- **Schoenberg's theorem****

$$K(x^\top y) = \sum_{p \geq 0} a_p (x^\top y)^p, \qquad a_p \geq 0$$

- **Select $\mathrm{p_i} \sim \mu \in \mathbb{N}$ for $i = 1, \ldots, D$**
  - **Approx. $(x^\top y)^{p_i}$: select $\omega_1, \ldots, \omega_{p_i} \sim \{-1, 1\}^d$**

$$Z_i : x \mapsto \sqrt{a_{p_i}} \prod_{j=1}^{p_i} \omega_j^\top x$$

- **Similar approximation guarantees as earlier**

***[K.-Karnick AISTATS 12], **[Schoenberg 42]**

Deterministic Feature Maps

Exact/approximate recovery of kernel values with certainty

Intersection Kernel*

- **Kernel of the form $K(x, y) = \sum_{j=1}^{d} \min\{x^j, y^j\}$**
- **Exploit *additive separability* of the kernel**

$$h(x) = \sum_{i=1}^{n} \alpha_i \sum_{j=1}^{d} \min\left\{x^j, x_i^j\right\} = \sum_j h_j(x)$$

$$h_j(x) = \sum_i \alpha_i min\left\{x^j, x_i^j\right\}$$

- **Each $h_j(x)$ can be calculated in $\mathrm{O}(\log n)$ time !**
  - **Requires $\mathrm{O}(n \log n)$ preprocessing time per dimension**
- **Prediction time *almost* independent of n**
  - **However, deterministic and exact method – no ϵ or δ**

***[Maji-Berg-Malik CVPR 08]**

Scale Invariant Kernels*

- **Kernels of the form $K(x, y) = \sum_{j=1}^{d} K_j(x^j, y^j)$ where**

$$K_j(x^j, y^j) = (x^j)^\gamma K_j\left(\frac{x^j}{y^j}\right)(y^j)^\gamma, \quad \gamma \geq 0$$

- **Bochner's theorem still applies****
  - **Involves working with $\widetilde{K}_j(x^j, y^j) = \widetilde{K}(\log|x^j| - \log|y^j|)$**
  - **Restrict domain so that we have a Fourier series**

$$\widetilde{K}_j(\lambda) = \sum_{k=-\infty}^{\infty} \widetilde{\mu}_k e^{ij\Delta\lambda}$$

  - **Use only lower frequencies $k \in \{-A, \ldots A\}$**
- **Deterministic ϵ-approximate maps**

***[Vedaldi-Zisserman CVPR 10], **[K. 12]**

Fast Feature Maps

Accelerated Random Feature Constructions

Fast Fourier Features

- **Special case of** $K(x, y) = \exp\left(\frac{-\|x-y\|^2}{2\sigma^2}\right)$
  - **Old method:** $\mathrm{W} \in \mathbb{R}^{D\times d},\ W_{ij} \sim \mathcal{N}\left(0, \sigma^{-2}\right)$**,** $\mathrm{O}(dD)$ **time**
  - **Instead use** $\widetilde{Z}: x \mapsto \cos(Vx)$ **where** $V = SHG\Pi\mathrm{HB}$
  - Π **is the Hadamard transform,** Π **is a random permutation** S, G, B **random diagonal scaling, Gaussian and sign matrices**
- **Prediction time** $\mathrm{O}(\mathrm{D}\log d)$**,** $\mathrm{E}\left[\widetilde{Z}(x)^\top \widetilde{Z}(y)\right] = K(x, y)$
  - **Rows of** V **are (non independent) Gaussian vectors**
  - **Correlations are sufficiently low** $\mathrm{Var}\left[\widetilde{Z}(x)^\top \widetilde{Z}(y)\right] \leq \mathrm{O}\left(\frac{1}{D}\right)$
  - **However, exponential convergence (for now) only for** $D = d$

Fast Taylor Features

- **Special case of $K(x, y) = (x^\top y + c)^p$**
  - Earlier method Z: $x \mapsto \prod_{j=1}^{p} \omega_j^\top x$, takes $\mathrm{O}(pdD)$ time
  - New method* takes $\mathrm{O}\big(p(d + D\log D)\big)$ time
  - Earlier method works (a bit) better for $c > 0$
    - Should be possible to improve new method as well
- Crucial idea $(x^\top y)^p = \langle x^{\otimes p}, y^{\otimes p} \rangle$
  - Count Sketch** $C: x \mapsto C(x)$ such that $C(x)^\top C(y) \approx x^\top y$
  - Create sketch $C(X) \in \mathbb{R}^D$ of tensor $X = x^{\otimes p}$
    - Create p independent count sketches $C_1(x), \dots, C_p(x)$
    - Can show that $C(x^{\otimes p}) \sim \prod_{j=1}^{p} P\left(C_j(x)\right)$
    - Can be done in time $\mathrm{O}\big(p(d + D\log D)\big)$ time using FFT

*[Pham-Pagh KDD 13], **[Charikar *et al* 02]

The BIG Picture

An Overview of Explicit Feature Methods

Other Feature Construction Methods

- **Efficiently evaluable maps for efficient prediction**
  - **Fidelity to a particular kernel not an objective**
  - **Hard(er?) to give generalization guarantees**
- **Local Deep Kernel Learning (LDKL)***
  - **Sparse features speed up evaluation time to $\mathrm{O}(d \log D)$**
  - **Training phase more involved**
- **Pairwise Piecewise Linear Embedding (PL2)****
  - **Encodes (discretization of) individual and pairs of features**
  - **Construct a $\mathrm{D} = O\left(\mathrm{K}(\mathrm{d} + d^2)\right)$ dimensional feature map**
  - **Features are $\mathrm{O}(d + d^2)$ -sparse**

***[Jose *et al* ICML 13], **[Pele *et al* ICML 13]**

A Taxonomy of Feature Methods

Kernel Dependence	Data Dependence: Yes	Data Dependence: No
Yes	**Nystrom Methods** - Slow training - Data aware - Problem oblivious	**Explicit Maps** - Fast training - Data oblivious - Problem oblivious
No	**LDKL, PL2** - Slow(er) training - Data aware - Problem aware	

Discussion

The next big thing in accelerated kernel learning ?