Random Function Classes

Kernel (recall)

$$k(x,x') = \langle \ell(x), \ell(x') \rangle = \sum_i \lambda_i \ell_i(x) \ell_i(x')$$

where $\langle \ell_i, \ell_j \rangle = \delta_{ij}$ (in L_2)

Corollary:

Sampling yields a good rate:

draw i with $p(i) = \frac{\lambda_i}{\sum_j \lambda_j}$ has

variance $\frac{1}{m}\left[\sum_i \lambda_i \left(\ell_i(x)\ell_i(x')\right)^2 - k^2(x,x')\right]$

Useful Lemmas: (Neal, 1994)

given ℓ_z with $z \sim p(z)$

and $\alpha_z \sim \mathcal{N}(0,1)$

yields functions $f(x) := \frac{1}{m}\sum_i \alpha_{z_i} \ell_{z_i}(x)$

such that

$$\mathbb{E}\{f(x)\} = \mathbb{E}_z\left[\sum_i \mathbb{E}[\alpha_{z_i}] \ell_{z_i}(x)\right] = 0$$

$$\mathbb{E}_{f}\left[f(x) f(x')\right] = \mathbb{E}_z\left[\frac{1}{m}\sum_{i=1}^m \mathbb{E}[\alpha_{z_i}^2] \ell_{z_i}^2(x)\right]$$

$$= \mathbb{E}_{z\sim p(z)}\left[\ell_z(x)\ell_z(x')\right]$$

$$=: k(x,x')$$

⇒ This is a Gaussian Process

If $|\ell_z(x)| \le R$

then $\Pr_z\left[\left|k(x,x') - \frac{1}{m}\sum_{i=1}^m \ell_{z_i}(x)\ell_{z_i}(x')\right| > \varepsilon\right]$

$$\le 2e^{-m\varepsilon^2/R^4}$$

by Application of Hoeffding $\Box \Rightarrow \varepsilon = O\left(\sqrt{\frac{\log\delta}{m}}\right)$

Much Stronger result via Matrix Hoeffding

(we will prove this later in class)

Thm: given matrices Y_k with $\mathbb{E}[Y_k] = 0$

and $Y_k^2 \le A_k^2 \quad \forall k$

then $\Pr\left\{\lambda_{max}\left(\sum_k Y_k\right) \ge \varepsilon\right\} \le d\, e^{-\varepsilon^2/2\sigma^2}$

where $\sigma^2 = \frac{1}{2}\left\|\sum_k A_k^2 + \mathbb{E}[Y_k^2]\right\|$

Application $Y_k := \frac{1}{m}\left[\ell_z(x)\ell_z(x') - k\right]$

$\mathbb{E}[Y_k] = 0$

$Y_k^2 = \frac{1}{m^2}\left[\ell_z \ell_z^+ \ell_z \ell_z^+ - \ell_z \ell_z^+ k - k \ell_z \ell_z^+ + k^2\right]$

$\ell_z \ell_z^+ \ell_z = \ell_z \|\ell_z\|^2$

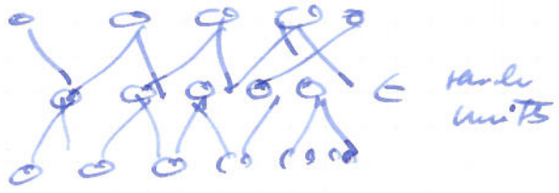
assume that $\|\ell_z\|^2 \le m R^2$

then we pick $A_k = m R^2 \cdot 1$

Random Function Classes

This has been (re) invented many times:

a) Neural Networks with random activation potential
~ 1980s



b) Empirical kernel Map (Tsuda, Schölkopf, ~ 1990)

$x \to \left(k(x_1, x) \dots k(x_n, x)\right)$

$\Rightarrow \overline{k}(x,x') = \underset{\tilde{x} \sim p(x)}{\mathbb{E}} \left[k(x, \tilde{x}) \, k(x', \tilde{x}) \right]$

c) Random Kitchen Sinks (Rahimi & Recht, ~ 2008)

for Gaussian / RBF kernel

$$k(x - x') = \int d\varrho(\omega) \underbrace{e^{i\langle \omega, x - x' \rangle}}_{e^{i\langle \omega, x \rangle} \; e^{-i\langle \omega, x' \rangle}}$$

$$\varphi_\omega(x) := \left(\sin\langle \omega, x \rangle, \cos\langle \omega, x \rangle\right)$$

renormalize measure $d\varrho(\omega)$ to get

$$\omega_i \sim \frac{\varrho(\omega)}{\|\varrho(\omega)\|_1}$$

and $\hat{k}(x,x') = \frac{1}{n} \sum_{j=1}^{n} e^{i\langle \omega_j, x \rangle} \; e^{-i\langle \omega_j, x' \rangle}$

Recipe: for Gaussian RBF draw

$$\Omega \sim \mathcal{N}(0,1)$$

$\frac{1}{\sqrt{n}} \exp(i(b \;\; \Omega \cdot x))$ are basis functions
Otherwise simply rescale Ω.

d) inner product kernel $k(x,x') = k(\langle x, x' \rangle)$

Invariances $k(x,x') = k(Ux, Ux')$

This means that in its representation on ℓ_i we need to have $\left[S(U) f(x)\right]^{+} \lambda \left[S(U) f(x')\right] = \langle f(x), f(x') \rangle$

Hence k decomposes into irreducible orthogonal representations $k(x,x') = \sum_j \lambda_j f_j(x)^{+} f_j(x')$

Separating radial / angular parts implies an expansion in terms of spherical harmonics, hence we have $k(x,x') = \sum \lambda_j L_j\left(\frac{\langle x, x' \rangle}{\|x\| \cdot \|x'\|}\right) \|x\|^j \|x'\|^j$

where L_j are adjoint Legendre polynomials (chosen for $\mathbb{R}^\alpha$)

use the fact that

$$L_j(\langle x, x' \rangle) = \int_{S_d} dt \, L_j(\langle x, t \rangle) L_j(\langle x', t \rangle)$$

on the unit sphere (i.e. $\|x\| = \|x'\| = 1$)

Recipe: ⊖ draw $\Omega \sim \mathcal{N}(0,1)$
⊖ rescale rows to unit length
⊖ compute $L_j(\langle x, z \rangle)$
for $j \sim p(j)$ according to choice of kernel

e) Fastfood (Sarlos, Le, Smola '13)

key bottleneck is computing and storing of Ω

but this is a Gaussian random matrix...

Idea: approximate Ω by $SH\Pi GHB$

where $S \in$ Scaling matrix (diagonal)

$H \in$ Hadamard

$\Pi \in$ Permutation

$B \in$ Binary digital (diagonal)

$G \in$ Gaussian (diagonal)

Runtime complexity: diagonal of n and $O(n)$

$$H_{2n} = \begin{bmatrix} H_n & H_n \\ H_n & -H_n \end{bmatrix}$$

can be multiplied in $O(n \log n)$ time.

Proof: assume this is possible for H_n

$$\Rightarrow H_{2n}\begin{pmatrix} v_{2n} \end{pmatrix} = \begin{pmatrix} H_n(v_n + v^n) \\ H_n(v_n - v^n) \end{pmatrix}$$

this is $O\left(\underbrace{2 \cdot (n \log n) + 2n}_{2n \log 2n}\right)$

(do not use Matlab's FHT → it is SLOW!)

$\Rightarrow O(n \log d)$ CPU, $O(n)$ RAM

for $d < n$

→ well-definedness:

$$\left[(H\Pi GHB)^+ (H\Pi GHB)\right]_{ii} =$$

$$n \cdot \left[BH^+ G^2 HB\right]_{ii} = n \operatorname{tr} G^2$$

(all random vectors lie on a sphere) □

→ for a given row of $H\Pi GHB$, we have independent Gaussians:

- binary doesn't change things (Central Gaussian)
- ditto Hadamard ✓

→ rates are a lot of work (maybe cover the details a la Chazelle later).

f) Random set indicators (Davies & Ghahramani '14)

→ generate random set indicators (e.g. trees)

→ compute $k(x,x') = \mathbb{E}_S\left[\delta(s(x), s(x'))\right]$

→ example: random decision trees / forest

all this is useful for

primal space

kernel / function classes (no point in using dual space unless $n \ll d$).