Examen Métodos Funcionales

Blanca Cano Camarero

March 18, 2023

Contents

Chapter 1

Introduction

To accelerate the training of kernel machines, they propose to map the input data to a randomized low-dimensional feature space and then apply existing fast linear methods.

SVMs are a powerful class of machine learning algorithms that can be used for both classification and regression tasks. SVMs use a kernel function to map the input data into a higher dimensional space, where it becomes easier to separate the classes with a hyperplane. SVMs are known for their ability to handle high-dimensional data and nonlinear relationships between variables. However, SVMs can be computationally expensive, especially when dealing with large datasets, because they involve solving a convex optimization problem.

Linear learning techniques, on the other hand, are a family of algorithms that are used for linear regression and classification problems. These techniques work by fitting a linear function to the data, which can then be used to make predictions. Linear learning techniques are generally faster and simpler than SVMs, more quickly when the dimensionality of the data is small because they operate on the covariance matrix rather than the kernel matrix of the training data. However, they may not perform as well as SVMs when dealing with complex relationships between variables.

The authors suggest a method to merge the benefits of both linear and nonlinear approaches. By drawing inspiration from randomized algorithms that estimate kernel matrices, they effectively convert the training and assessment of any kernel machine into the corresponding actions of a linear machine. This is achieved by converting the data into a relatively low-dimensional randomized feature space. Based on their experiments, the results show that the application of random features, together with simple linear learning techniques, achieve competitive results when compared to modern kernel-based classification and regression algorithms. Implementing random features greatly reduces the computational burden required for training, while obtaining similar or even better testing accuracy.

Chapter 2

Random Fourier Features

2.1 Random Fourier Features

Let be $x \in \mathbb{R}^d$ (a column vector), the first set of random features consists of random Fourier bases

$$\cos(w^T x + b) \tag{2.1}$$

where $w \in \mathbb{R}^d$ and b are random variables.

See that $T(x) = w^t x + b$ is affine transformation $T : \mathbb{R}^d \longrightarrow \mathbb{R}$ and then cos function maps $\cos : \mathbb{R} \longrightarrow S^1$.

The algorithm is defined as:

Algorithm 1: Random Fourier Features

Data: K a positive definite shift-invariant kernel $k(x, y) = k(x - y)$.

Result: A randomized feature map $z(x) : \mathbb{R}^d \longrightarrow \mathbb{R}^D$ so that $z(x)^T z(y) \approx k(x - y)$

1 Compute the Fourier transform p of the kernel k:

$$p(w) = \frac{1}{2\pi} \int e^{-jw^T \delta} k(\delta) d\Delta. \tag{2.2}$$

2 Draw D iid samples $\{w_1, \ldots, w_D\} \subset \mathbb{R}^d$ from p and D iid samples $b_1, \ldots, b_D \in \mathbb{R}$ from the uniform distribution on $[0, 2\pi]$.

3 Let

$$z(x) \equiv \sqrt{\frac{2}{D}} \left[\cos(w_1^T x + b_1) \ \ldots \cos(w_D^T x + b_D)\right]^T. \tag{2.3}$$

2.1.1 Implementation considerations

Fourier transformation

In order to compute (2.2) we have different alternatives:

- Use (2.2) and a numerical method to compute the integral (as Montecarlo).
- Use a library *scipy Fourier Transforms*[1]

[1] See `https://docs.scipy.org/doc/scipy/tutorial/fft.html` (last consult on 03-07-23).

- Write explicitly its formula.

For simplicity and since we use widely known kernels we are going to follow the third alternative. The implementation of the kernel and its Fourier Transformation can be found at `/Code/random_feature/kernels.py`.

How to draw examples from a probability distribution p

Let $p : \mathbb{R}^d \longrightarrow \mathbb{R}$ a probability function. In order to draw samples from this distribution we are going to use the following algorithm:

Algorithm 2: Generate example from a probability distribution

Data:

- Probability density function $p : \mathbb{R}^d \longrightarrow \mathbb{R}$,
- dimension of the output vectors d,
- number of vectors: n,
- maximum absolute value of the coefficients of the vectors: K.

Result: Radom variables: $\{w_1, \ldots, w_n\} \subset \mathbb{R}^d$ r that are examples of p.

Let $\Lambda \leftarrow \{\}$ be the solution.
while *The size of* Λ *is less than* n **do**
    Let $w \leftarrow U([-k, k]^d)$; `/* random vector of` d `vector that follow a uniform distribution in` $[-k, k]$ `interval */`
    $u \leftarrow U([0, 1])$
    if $u \leq p(w)$ **then**
        $\Lambda \leftarrow \Lambda \cup \{w\}$.
return Λ

Ideally, K should be as big as possible. However, the bigger K is, the bigger the norm of the vectors w and the lower their probability.

If the probability of acceptance is too low, we would end up rejecting too many vectors, making the algorithm slow. Therefore, so we need to find a balance between mathematical rigor and pragmatism choosing k.

This function is done at `random_feature/random_samples.py`.

2.1.2 Some popular shift-invariant kernels and their Fourier transforms

Kernel name	$k(x-y)$	p(w)
Gaussian	$e^{-\frac{\|x-y\|_2^2}{2}}$	$(2\pi)^{-\frac{D}{2}} e^{-\frac{\|w\|_2^2}{2}}$
$\mathbb{R}^2$ Laplacian	$e^{-\|\Delta\|_1}$	$\prod_{d=1}^{D} \frac{1}{\pi(1+w_d^2)}$

Table 2.1: Some popular shift-invariant kernels and their Fourier transforms

2.1.3 Convergence proof

Our objective is to proof that $z(x)^T z(y)$ is close to $k(x-y)$. Mathematically, we want to garante the uniform convergence of the Fourier Features.

Theorem 1 (Bochner's theorem)**.** *A continuos kernel $k(x,y) = k(x-y)$ on $\mathbb{R}^d$ is positive if and only if $k(\delta)$ is the Fourier transforme of a non-negative measure.*

Proof. If a shift-invariant kernel $k(\delta)$ is properly scaled, Bochner's theorem guarantees that its Fourier transform $p(w)$ is a proper probability distribution.

Defining $\zeta_w(x) = e^{jw^t x}$, we have

$$k(x-y) = \int_{\mathbb{R}^d} p(w) e^{jw^t(x-y)} dw = E_w\left[\zeta_w(x)\zeta_w(y)^*\right], \tag{2.4}$$

where $\zeta_w(y)^* = e^{-jw^t x}$ is the conjugate. We have proof in (2.4) that $\zeta_w(x)\zeta_w(y)^*$ is a unbiased estimate of $k(x,y)$ when w is drawn from p.

Notice that $p : \mathbb{R}^d \longrightarrow [0,1]$ is a real function, and the $k(\Delta)$ is real too, so the integral (2.4) converges when $\zeta_w(x)\zeta_w(y)^*$ is real.

Defining

$$z_w(x) = \sqrt{2}\cos\left(w^T x + b\right), \tag{2.5}$$

where w is drawn from a $p(w)$ and b is drawn uniformly from $[0, 2\pi]$ we obtain that

$$E\left[z_w(x) z_w(y)\right] = k(x,y). \tag{2.6}$$

Firstly, for $s \in \{1,-1\}$ notice that using chain rule and $p(w)$ is a probability function and therefore $\int_{\mathbb{R}^d} p(w)dw = 1$.

$$\begin{aligned}
E\left[e^{sjw^T(x+y)+s2b}\right] &= \int_{\mathbb{R}^d} e^{sjw^T(x+y)+s2b} p(w) dw \\
&= e^{s2b}\int_{\mathbb{R}^d} e^{sjw^T(x+y)} p(w) dw \\
&= e^{s2b}\left\{e^{sjw^T(x+y)} - \int j(x+y) e^{sjw(x+y)} dw\right\}_{\mathbb{R}^d} \\
&= e^{s2b}\left\{e^{sjw^T(x+y)} - e^{sjw(x+y)}\right\} = 0.
\end{aligned} \tag{2.7}$$

Secondly, as a consequence of the sum of angles:

$$\begin{aligned}
z_w(x) z_w(y) &= 2\cos\left(w^T x + b\right)\cos\left(w^T y + b\right) \\
&= \left(\cos\left(w^T x + b\right)\cos\left(w^T y + b\right) + \sin\left(w^T x + b\right)\sin\left(w^T y + b\right)\right) \\
&\quad + \left(\cos\left(w^T x + b\right)\cos\left(w^T y + b\right) - \sin\left(w^T x + b\right)\sin\left(w^T y + b\right)\right) \\
&= \cos\left(w^T(x-y)\right) + \cos\left(w^T(x+y) + 2b\right).
\end{aligned} \tag{2.8}$$

Thanks to Euler formula

$$\cos\left(w^T(x-y)\right) = \frac{1}{2}\left(e^{jw^T(x-y)} + e^{-jw^T(x-y)}\right) = \frac{1}{2}\left(e^{jw^T(x-y)} + e^{jw^T(y-x)}\right), \tag{2.9}$$

and its expected value is

$$E\left[\cos\left(w^T(x-y)\right)\right] = \frac{1}{2}\left(E\left[e^{jw^T(x-y)}\right] + E\left[e^{jw^T(y-x)}\right]\right) = k(x,y), \tag{2.10}$$

since k is symmetric and shift invariant and (2.4).

Finally, as a result of Euler formula and (2.7)

$$E\left[\cos\left(w^T(x+y)+2b\right)\right]=0, \tag{2.11}$$

so we have proved (2.6).

□

We can lower the variance of the estimate of the kernel by concatenating D randomly chosen z_w into one $D-dimensional$ vector and normalizing each component by $\sqrt{2}$. The inner product

$$z(x)^T z(y)=\frac{1}{D}\sum_{j=1}^{D} z_{w_j}(x)z_{w_j}(y) \tag{2.12}$$

is a sample average of z_w and is therefore a lower variance approximation to the expectation (2.4).

Now we are going to achieve a first bound for the distance between $z(x)^T z(y)$ and the kernel $k(x,y)$.

Theorem 2. *Hoeffding's inequality Let $X_1,\ldots,X_n$ be independent random variables such that $a_i \le X_i \le b_i$ almost surely. Consider $S_n = X_1+\ldots+X_n$.*

The Hoeffding's theorem states that, for all $t>0$,

$$P\left(|S_n - E[S_n]| \ge t\right) \le 2\exp\left(-\frac{2t^2}{\sum_{i=1}^{n}(b_i-a_i)^2}\right) \tag{2.13}$$

(See the proof at [Hoe94]).

Let $z_{w_j}(x)z_{w_j}(y)$ our normalized D independent random variables which are bounded:

$$\frac{-2}{\sqrt{D}} \le \frac{-2}{D} \le z_{w_j}(x)z_{w_j}(y) \le \frac{2}{D} \le \frac{2}{\sqrt{D}}. \tag{2.14}$$

Using Hoeffding's inequality

$$P\left(|z(x)^T z(y) - k(x,y)| > \varepsilon\right) \le 2\exp\left[-\frac{2\varepsilon^2}{(4/\sqrt{D})^2}\right] \le 2\exp\left[-\frac{D\varepsilon^2}{8}\right] \tag{2.15}$$

Now in order to proof the correctness of our algorithm we need to guarantees that guarantees that $z(x)'z(y)$ is close to $k(x-y)$ for the centers of an ϵ-net over $M\times M$.

Claim 1. *(Uniform convergence of Fourier features) Let M be a compact subset of $\mathbb{R}^d$ with diameter* $\operatorname{diam}(M)$*. Then, for the mapping z defined in Algorithm 1, we have*

$$P\left[\sup_{x,y\in M}|z(x)^T z(y)-k(y,x)|\ge\varepsilon\right] \le 2^8\left(\frac{\sigma_p \operatorname{diam}(M)}{\varepsilon}\right)\exp\left(-\frac{D\varepsilon^2}{4(d+2)},\right) \tag{2.16}$$

where $\sigma_p^2 \equiv \mathbb{E}_{p(\omega)}[\omega'\omega]$ is the second moment of the Fourier transform of k.

Further,

$$\sup_{x,y\in M}|z(x)^T z(y)-k(y,x)|\ge\varepsilon$$

with any constant probability when $D=\Omega\left(\frac{d}{\epsilon^2}\log\frac{\sigma_p diam M}{\epsilon}\right)$.

Proof. Define $s(x,y) \equiv z(x)^T z(y)$, and $f(x,y) \equiv s(x,y) - k(y,x)$, and for a bigger enough D in the first inequality (2.15) and by construction we would have $|f(x,y)| \leq 2$ and $E[f(x,y)] = 0$.

Let define

$$M_\Delta = \{x - y : x, y \in M\}. \tag{2.17}$$

Since M is compact, M_Δ is also compact. Moreover, by the triangle inequality, M_Δ has diameter at most twice $\text{diam}(M)$. Since M_Δ is compact, we can construct an ϵ-net that covers M_Δ using at most $T = (4\,\text{diam}(M)/r)^d$ balls of radius r.

Let $\{\Delta_i\} i = 1^T$ denote the centers of these balls, and let L_f be the Lipschitz constant of f. If we ensure that $|f(\Delta_i)| < \epsilon/2$ for all i and $L_f < \epsilon$, then we can guarantee that $|f(\Delta_i)| < \epsilon$ for all $\Delta \in M_\Delta$ by using triangle inequality, Lipschitz definition and all the hypothesis:

$$|f(\Delta)| = |f(\Delta) \pm f(\Delta_i)| \tag{2.18}$$

$$\leq L_f |\Delta - \Delta_i| \tag{2.19}$$

$$\leq L_f r + \frac{\varepsilon}{2} = \varepsilon. \tag{2.20}$$

Since f differentiable, $L_f = \|\nabla f(\Delta^*)\|$, where $\Delta^* = \arg\max_{\Delta \in M_\Delta} \|\nabla f(\Delta)\|$.

By variance expansion in expectations and s gradient,

$$E[\nabla s(\Delta)] = \nabla k(\Delta), \tag{2.21}$$

so

$$E[L_f^2] = E\left[\|\nabla s(\Delta*) - \nabla k(\Delta^*)\|^2\right] = \tag{2.22}$$

$$= E\left[\|\nabla s(\Delta*)\|^2\right] - E\left[\|\nabla k(\Delta*)\|\right]^2 \tag{2.23}$$

$$\leq E\left[\|\nabla s(\Delta*)\|^2\right] \tag{2.24}$$

$$= E\left[w^2 \sin(2\Delta)\right] \tag{2.25}$$

$$\leq E\left[\|w\|^2\right] = \sigma_p^2. \tag{2.26}$$

By Markov's inequality,

$$P\left[L_f^2 \geq t\right] \leq \frac{E[L_f^2]}{t}, \tag{2.27}$$

so

$$P\left[L_f \geq \frac{\epsilon}{2r}\right] \leq \left(\frac{2r\sigma_p}{\epsilon}\right)^2. \tag{2.28}$$

The onion bound followed by Hoeffding's inequality applied to the anchors in the $\epsilon-$net gives

$$P\left[\cup_{i=1}^T \|f(\Delta_i)\| \geq \epsilon/2\right] \leq 2T \exp\left(-D^2/8\right). \tag{2.29}$$

Combining previous inequalities in term of the free variable r:

$$P\left[\sup_{\Delta \in M_\Delta} |f(\Delta)| \leq \epsilon\right] = P\left[\cup_{i=1}^T \|f(\Delta_i)\| \leq \epsilon/2 \wedge L_f \leq \frac{\epsilon}{2r}\right] \tag{2.30}$$

$$= 1 - P\left[\cup_{i=1}^T \|f(\Delta_i)\| \geq \epsilon/2 \vee L_f \geq \frac{\epsilon}{2r}\right] \tag{2.31}$$

$$= 1 - P\left[\cup_{i=1}^T \|f(\Delta_i)\| \geq \epsilon/2\right] - P\left[L_f \geq \frac{\epsilon}{2r}\right] \tag{2.32}$$

$$\geq 1 - 2\left(\frac{4diam(M)}{r}\right)^d \exp(-D\epsilon^2/8) - \left(\frac{2r\sigma_p}{\epsilon}\right)^2. \tag{2.33}$$

This has the form $1 - \kappa_1 r^{-d} - \kappa_2 r^2$. Setting $r = \left(\frac{k_1}{k_2}\right)^{\frac{1}{d+2}}$ turns this to

$$1 - 2k_2^{\frac{d}{d+2}} k_1^{\frac{d}{d+2}}, \tag{2.34}$$

and assuming that $\frac{\sigma_p \operatorname{diam}(M)}{\epsilon} \geq 1$, proves the first part of the claim. To prove the second part of the claim, pick any probability for the right hand side and solve for D.

□

Chapter 3

Random Binning Features

3.1 Random Binning Features

The algorithm we are going to present is:

Algorithm 3: Random Binning Features

Data:

- A point $x \in \mathbb{R}^d$.
- A kernel function $k(x, y) = \prod_{m=1}^{d} k_m(|x^m - y^m|)$, so that $p_m(\Delta) \equiv \Delta k_m''(\Delta)$ is a probability distribution on $\Delta \geq 0$.
- a randomized feature map $z(x)$ so that $z(x)^T z(y) \approx k(x - y)$.

Result:

$$z(x) \equiv \sqrt{\frac{1}{P}} \left[z_1(x), \ldots, z_p(x) \right]^T .$$

for $p \in \{1, \ldots, P\}$ **do**

Draw grid parameters $\delta, u \in \mathbb{R}^d$ with the pitch $\delta^m \sim p_m$, and shift u^m from the uniform distribution on $[0, \delta^m]$.

Let z return the coordinate of the bin containing x as a binary indicator vector

$$z_p(x) \equiv \text{hash}\left(\left\lfloor \frac{x^1 - u^1}{\delta^1} \right\rfloor, \ldots, \left\lfloor \frac{x^d - u^d}{\delta^d} \right\rfloor \right).$$

As we will see in the deduction of the algorithm, the concept of hashing is a necessary tool for its implementation. Strictly speaking, partitioning the real numbers (or a d-multidimensional real space) into segments (of d-balls) of length r would result in a bijective partition of the natural numbers, and therefore the binary encoding of the real numbers (or space) would have the potential to have as many digits as we want.

However, in real life, the space we are dealing with is not the entire real numbers but rather a compact subset of it, which is therefore finite.

3.2 Algorithm deduction

The second randomized mapping partitions the input space by utilizing grids that are randomly shifted and have random resolutions. It assigns a binary bit string to each input point, based on the bin in which the point falls. Since this mapping uses rectilinear grids, it is particularly suitable for kernels that depend solely on L_1.

The grids are designed such that the likelihood of two points x and y being assigned to the same bin is proportional to $k(x, y)$. The inner product between two transformed points is proportional to the number of times they are binned together, providing an unbiased estimate of $k(x, y)$.

Now we are going to define the kernel. Firstly we define a real (one dimension) approach:

Let $k_{hat}(x, y; \delta) : \mathbb{R} \times \mathbb{R} \times \mathbb{R}^+$ be the kernel defined as:

$$k_{hat}(x, y; \delta) = \max\left(0, 1 - \frac{|x - y|}{\delta}\right) \tag{3.1}$$

Notice that

$$1 - \frac{|x - y|}{\delta} > 0 \Leftrightarrow \delta > |x - y| \tag{3.2}$$

and since $\frac{|x-y|}{\delta} > 0$ for every $x, y \in \mathbb{R}$

$$0 \leq k_{hat}(x, y; \delta) \leq 1. \tag{3.3}$$

Moreover, this induce a real partition of pitch δ and k_{hat} is the probability that two pints x, y falls in the same grid since they would be in the same grid if

$$\hat{x} = \hat{y} \Leftrightarrow \left\lfloor \frac{x - u}{\delta} \right\rfloor = \left\lfloor \frac{y - u}{\delta} \right\rfloor. \tag{3.4}$$

It would happen if

$$\left| \hat{x} - \frac{x - u}{\delta} \right| \leq 0.5 \text{ and } \left| \hat{x} - \frac{y - u}{\delta} \right| \leq 0.5. \tag{3.5}$$

Adding both constraint and using triangle inequality we obtain

$$1 \geq \left| \frac{x - u}{\delta} - \frac{y - u}{\delta} \right| \geq \frac{|x - y|}{\delta}. \tag{3.6}$$

So fixed δ the probability of two points x, y to fall in the same bin is the grid $k_{hat}(x, y, \delta)$.

If we encode $\hat{x}$ as a binary indicator vector $z(x)$ over the bins, $z(x)^T z(y) = 1$ if x and y fall in the same bin and zero otherwise, so

$$P\left[z(x)^T z(y) = 1 | \delta\right] = E\left[z(x)^T z(y) = 1 | \delta\right] = k_{hat}(x, y, \delta). \tag{3.7}$$

therefore z is a random map for k_{hat}. Let us now consider shift-invariant kernels that can be expressed as convex combinations of hat kernels on a compact subset of $\mathbb{R} \times \mathbb{R}$. Mathematically, this can be written as:

$$k(x, y) = \int_0^\infty k_{hat}(x, y, \delta) p(\delta) d\delta, \tag{3.8}$$

where $k_{hat}(x, y, \delta)$ is the hat kernel with pitch δ, $p(\delta)$ is a probability distribution over pitch values, and the integral is taken over all positive pitch values. In this formulation, if the pitch δ of the grid is sampled from p, and the shift u is drawn uniformly from $[0, \delta]$, then the probability that x and y are binned together is given by $k(x, y)$.

Before to continue we need to proof the following result:

Lemma 1. *Suppose a function $k(\Delta) : \mathbb{R} \to \mathbb{R}$ is twice differentiable and has the form*

$$k(\Delta) = \int_{\mathbb{R}} p(\delta) \max(0, 1 - \Delta\delta) d\delta. \tag{3.9}$$

Then $p(\delta) = \delta k''(\delta)$.

Proof.

$$k(\Delta) = \int_{\mathbb{R}} p(\delta) \max(0, 1 - \Delta\delta) d\delta \tag{3.10}$$

$$= \int_0^{\Delta} p(\delta) 0 d\delta + \int_{\Delta}^{\infty} p(\delta) \left(1 - \frac{\Delta}{\delta}\right) d\delta \tag{3.11}$$

$$= \int_{\Delta}^{\infty} p(\delta) d\delta - \int_{\Delta}^{\infty} \frac{p(\delta)}{\delta} d\delta. \tag{3.12}$$

Applying the Fundamental theorem of calculus and chain rules:

$$k'(\Delta) = -p(\Delta) - \left[\int_{\Delta}^{\infty} \frac{p(\delta)}{\delta} d\delta - \Delta \frac{p(\Delta)}{\Delta}\right] = -\int_{\Delta}^{\infty} \frac{p(\delta)}{\delta} d\delta. \tag{3.13}$$

Applying again the Fundamental theorem oc calculus:

$$k''(\Delta) = \frac{P(\Delta)}{\Delta}. \tag{3.14}$$

□

Lema 1 1 shows that the function p can be easily recovered from k by setting

$$p(\delta) = \delta k''(\delta). \tag{3.15}$$

For example, when considering the Laplacian kernel with $k_{\text{Laplacian}}(x, y) = \exp(-|x - y|)$, we have that $p(\delta)$ is the Gamma distribution given by $\delta \exp(-\delta)$. However, for the Gaussian kernel, $k(\delta)$ is not convex, resulting in k'' not being everywhere positive, and thus $\delta k''(\delta)$ is not a probability distribution. Therefore, this procedure does not yield a random map for the Gaussian.

Random maps for separable multivariate shift-invariant kernels of the form

$$k(x - y) = \prod_{m=1}^{d} k_m(|x_m - y_m|) \tag{3.16}$$

(such as the multivariate Laplacian kernel) can be constructed similarly if each k_m can be expressed as a convex combination of hat kernels.

The above binning process is applied independently to each dimension of $\mathbb{R}^d$. The probability that x^m and y^m are binned together in dimension m is $k_m(|x^m - y^m|)$.

Since the binning process is independent across dimensions, the probability that x and y are binned together in every dimension is

$$\prod_{m=1}^{d} k_m(|x_m - y_m|) = k(x - y). \tag{3.17}$$

In this multivariate case, $z(x)$ encodes the integer vector $[\hat{x}^1, \ldots, \hat{x}^d]$ corresponding to each bin of the d-dimensional grid as a binary indicator vector. In practice, to prevent

overflows when computing $z(x)$ for large d, unoccupied bins are eliminated from the representation. Since there are never more bins than training points, this ensures that no overflow can occur.

We can again reduce the variance of the estimator $z(x)^T z(y)$ by concatenating P random binning functions z into a larger list of features z and scaling by $\sqrt{\frac{1}{P}}$. The inner product

$$z(x)'z(y) = \frac{1}{P}\sum_{p=1}^{P} z_p(x)'z_p(y) \tag{3.18}$$

is the average of P independent $z(x)^T z(y)$ and therefore has lower variance.

Since z is binary, Hoeffding's inequality guarantees that for a fixed pair of points x and y, $z(x)'z(y)$ converges exponentially quickly to $k(x,y)$ as P increases. Again, a much stronger claim is that this convergence holds simultaneously for all points:

Claim 2. *Let M be a compact subset of $\mathbb{R}^d$ with diameter $diam(M)$. Let $\alpha = \mathbb{E}[1/delta]$ and let L_k denote the Lipschitz constant of k with respect to the L_1 norm. With z as above, we have:*

$$P\left[\sup_{x,y\in M} |z(x)^T z(y) - k(y,x)| \geq \varepsilon\right] \geq 1 - 36dP\alpha\ diam(M)\exp\left(\frac{-\left(\frac{P\epsilon^2}{8} + \ln\frac{\epsilon}{L_k}\right)}{d+1}\right) \tag{3.19}$$

Proof. The argument is similar to that of Claim 1. As z is piecewise constant over M, one can partition $M \times M$ into several L1 balls, such that $k(x,y)$ varies only slightly and $z(x)$ and $z(y)$ are constant within each cell. At the centers of these cells, it is likely that $z(x)^T z(y)$ is close to $k(x,y)$, which implies that $k(x,y)$ and $z(x)^T z(y)$ are close throughout M. □

Chapter 4

Experiments

4.1 Description

In order to reproduce the experiments we are going to train four models in classification and regression problems:

- A lest squares problem with Radom Fourier features.
- A lest squares problem with Radom Binning Features.
- Core Vector Machine,
- and a classic Support Vector Machine.

The implementation we are going to use are:

- For Random Fourier Features we are going to use sklearn: `https://scikit-learn.org/stable/modules/generated/sklearn.kernel_approximation.RBFSampler.html`,
- for a lest squares problem with Random Binning Features `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.KBinsDiscretizer.html`
- For the linear problems for classification `https://scikit-learn.org/stable/modules/linear_model.html`.
- SVM `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`

Datasets

The dataset used in [RR07] is the same as the one used in [TKC05], except for the synthetic dataset.

So our datasets are:

- USPS: `https://datasets.activeloop.ai/docs/ml/datasets/usps-dataset/`.
- CENSUS: `https://www.kaggle.com/datasets/uciml/adult-census-income?resource=download`
- `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_covtype.html`

- TOPO `https://www.openml.org/search?type=data&sort=runs&status=any&qualities.NumberOfClasses=lte_1&qualities.NumberOfFeatures=between_100_1000&qualities.NumberOfInstances=between_1000_10000&id=422`

How to see experiment

As example Census experiment is done in a jupyter notebook. Topo and forest covertypes can be executed typing `make topo` and `make forest_covertypes`, the results will appears in results directory.

4.1.1 Results and conclusions

You can find the mentioned data in the results folder and the census Jupyter notebook.

It is challenging to compare the results with the original article because, despite being relatively well-documented, we have only found the largest dimension databases that we could not execute on our computer due to computational constraints.

The most illustrative result obtained was for CENSUS, where an improvement in execution time was observed, as indicated in the article. However, there was also a reduction in precision compared to the test results. This reduction was mitigated with Random Binning Features.

On the other hand, Forest Covertype showed a significant improvement in execution time but accompanied by a decrease in precision, which opens the possibility to try other kernels or methods to maintain the execution time while improving precision.

The TOPO database turned out to be a negative case in all respects. Neither SVM nor feature methods achieved decent results. Therefore, it is not worth noting the trend in execution times.

Chapter 5

Extensions

5.1 Published articles

5.1.1 HARFE: Hard-Ridge Random Feature Expansion

[SST22] The authors of the article propose a method called the hard-ridge random feature expansion method (HARFE) for approximating high-dimensional sparse additive functions. The method involves utilizing a hard-thresholding pursuit-based algorithm to approximate the coefficients with respect to the random feature matrix, using a random sparse connectivity pattern in the matrix to match the additive function assumption. The approach balances between obtaining sparse models that use fewer terms in their representation and ridge-based smoothing that tends to be robust to noise and outliers. The authors prove that the HARFE method converges with a given error bound depending on the noise and the parameters of the sparse ridge regression model. Numerical results on synthetic data and real datasets show that the HARFE approach obtains lower (or comparable) error than other state-of-the-art algorithms.

5.1.2 Toward Large Kernel Models

[ABP23]

The article maintain the interest in kernel machines, claiming that kernel machines have been additionally bolstered by the discovery of their equivalence to wide neural networks in certain regimes. However, a key feature of DNNs is their ability to scale the model size and training data size independently, whereas in traditional kernel machines model size is tied to data size. Because of this coupling, scaling kernel machines to large data has been computationally challenging.

The article presents a new method for constructing large-scale general kernel models that decouples the model and data, allowing for training on large datasets. The method is called EigenPro 3.0 and is based on projected dual preconditioned SGD. The authors demonstrate that EigenPro 3.0 enables scaling to model and data sizes that were previously not possible with existing kernel methods.

Other older extensions

1 "Convolutional Random Features for Point Cloud Analysis" (2018) - This work extends the idea of random features to point cloud data, which are common in computer vision and robotics applications. The authors propose a technique for extracting random features from point clouds and show that their approach improves performance on point cloud

classification and segmentation tasks. 2 "Distributed Random Features for Large-Scale Kernel Machines" (2013) - This work proposes a technique for distributing the calculation of random features across a cluster of computers, enabling the use of this technique on even larger datasets.

3."Random Fourier Approximations for Positive Definite Kernels: Improved Bounds" (2015) - This work extends the theoretical results presented in the original article, providing better bounds for the approximation error when using random features to approximate positive definite kernels.

4."Random Features for Time-Series Analysis" (2019) - This work proposes a technique for extracting random features from time-series data, which are common in applications such as stock price prediction and anomaly detection. The authors show that their approach improves performance on time-series prediction tasks. These are just some of the most important extensions of the "Random Features for Large-Scale Kernel Machines" article. Since its publication, there have been many other related research efforts that have further explored the possibilities of random features in machine learning and kernel theory.

5.2 My proposal

As we saw at CENSUS random features is an interesting tools to improve time but have some major faults:

The hypotheses required for kernel functions directly impact the class of functions we can explore, which can lead to a fitting error. Another issue is that depending on the chosen method, the results can vary dramatically, and if hyperparameter tuning is added, the search for the optimal fit can become overly laborious. Therefore, we propose exploring the following areas:

1. Discovering new mathematical hypotheses that relax the conditions of kernel functions. 2. Identifying techniques or methods that automate the selection of kernels or other hyperparameters.

Bibliography

[Hoe94] Wassily Hoeffding. "Probability Inequalities for sums of Bounded Random Variables". In: *The Collected Works of Wassily Hoeffding*. Ed. by N. I. Fisher and P. K. Sen. New York, NY: Springer New York, 1994, pp. 409–426. ISBN: 978-1-4612-0865-5. DOI: 10.1007/978-1-4612-0865-5_26. URL: https://doi.org/10.1007/978-1-4612-0865-5_26.

[TKC05] Ivor Wai-Hung Tsang, James Tin-Yau Kwok, and Pak-Ming Cheung. "Core Vector Machines: Fast SVM Training on Very Large Data Sets". In: *J. Mach. Learn. Res.* 6 (2005), pp. 363–392.

[RR07] Ali Rahimi and Benjamin Recht. "Random Features for Large-Scale Kernel Machines". In: *Advances in Neural Information Processing Systems*. Ed. by J. Platt et al. Vol. 20. Curran Associates, Inc., 2007. URL: https://proceedings.neurips.cc/paper/2007/file/013a006f03dbc5392effeb8f18fda755-Paper.pdf.

[SST22] Esha Saha, Hayden Schaeffer, and Giang Tran. *HARFE: Hard-Ridge Random Feature Expansion*. 2022. DOI: 10.48550/ARXIV.2202.02877. URL: https://arxiv.org/abs/2202.02877.

[ABP23] Amirhesam Abedsoltan, Mikhail Belkin, and Parthe Pandit. *Toward Large Kernel Models*. 2023. DOI: 10.48550/ARXIV.2302.02605. URL: https://arxiv.org/abs/2302.02605.