Random Fourier Features for Kernel Ridge Regression

Approximation Bounds and Statistical Guarantees

Kailong Wang[1]

[1]Ph.D. of ECE
Rutgers University University

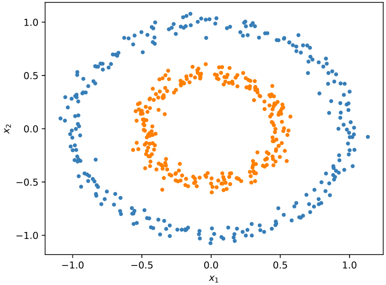
ECE 539 HDP, May 3, 2023

Table of Contents

Linear Classification with Non-linear Input

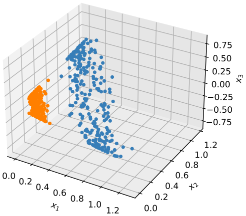
Consider a binary classification problem with non-linear (e.g. polynomial) samples. This is not separable with linear function.

(e.g. $X = \begin{bmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \\ & \dots \\ x_{N,1} & x_{N,2} \end{bmatrix} \in \mathbb{R}^{N \times 2}$.)



Lifting

One idea is to **LIFT** the samples into a higher dimensional space in which the samples are linearly separable.



The Lifting function in this case is $\phi(X) = \begin{bmatrix} x_{1,1}^2 & x_{1,2}^2 & \sqrt{2}x_{1,1}x_{1,2} \\ x_{2,1}^2 & x_{2,2}^2 & \sqrt{2}x_{2,1}x_{2,2} \\ & \cdots & \\ x_{N,1}^2 & x_{N,2}^2 & \sqrt{2}x_{N,1}x_{N,2} \end{bmatrix}$.

Curse of Dimensionality

Consider solving the above problem with *support vector machine* (SVM).

$$\mathcal{L}(\mathbf{w}, \alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n}^{N} \sum_{m}^{N} \alpha_n \alpha_m y_n y_m ({x_n}^\intercal x_m).$$

The $\mathbf{w}$ is the linear decision boundary and α is a vector of Lagrange multipliers.

Curse of Dimensionality

Consider solving the above problem with *support vector machine* (SVM).

$$\mathcal{L}(\mathbf{w}, \alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n}^{N} \sum_{m}^{N} \alpha_n \alpha_m y_n y_m ({x_n}^\intercal x_m).$$

The $\mathbf{w}$ is the linear decision boundary and α is a vector of Lagrange multipliers.

We need to use lifting function $\phi(X)$ to make the samples linearly separable. Specifically, we replace $({x_n}^\intercal x_m)$ with $(\phi(x_n)^\intercal \phi(x_m))$.

$$\begin{aligned}\phi(x_n)^\intercal \phi(x_m) &= \begin{bmatrix} x_{n,1}^2 & x_{n,2}^2 & \sqrt{2} x_{n,1} x_{n,2} \end{bmatrix} \begin{bmatrix} x_{m,1}^2 & x_{m,2}^2 & \sqrt{2} x_{m,1} x_{m,2} \end{bmatrix}^\intercal \\ &= x_{n,1}^2 x_{m,1}^2 + x_{n,2}^2 x_{m,2}^2 + 2 x_{n,1} x_{n,2} x_{m,1} x_{m,2}\end{aligned}$$

Curse of Dimensionality

Consider solving the above problem with *support vector machine* (SVM).

$$\mathcal{L}(\mathbf{w}, \alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n}^{N} \sum_{m}^{N} \alpha_n \alpha_m y_n y_m ({x_n}^\intercal x_m).$$

The $\mathbf{w}$ is the linear decision boundary and α is a vector of Lagrange multipliers.

We need to use lifting function $\phi(X)$ to make the samples linearly separable. Specifically, we replace $({x_n}^\intercal x_m)$ with $(\phi(x_n)^\intercal \phi(x_m))$.

$$\begin{aligned}\phi(x_n)^\intercal \phi(x_m) &= \begin{bmatrix} x_{n,1}^2 & x_{n,2}^2 & \sqrt{2} x_{n,1} x_{n,2} \end{bmatrix} \begin{bmatrix} x_{m,1}^2 & x_{m,2}^2 & \sqrt{2} x_{m,1} x_{m,2} \end{bmatrix}^\intercal \\ &= x_{n,1}^2 x_{m,1}^2 + x_{n,2}^2 x_{m,2}^2 + 2 x_{n,1} x_{n,2} x_{m,1} x_{m,2}\end{aligned}$$

Calculate the inner product in the $\mathbb{R}^3$ across all N pairs of samples is acceptable. However, the lifting function $\phi(X)$ is usually very high dimensional.

Kernel Trick

Consider the following derivation,

$$\begin{aligned}
({x_n}^\intercal x_m)^2 &= ([x_{n,1}\ x_{n,2}][x_{m,1}\ x_{m,2}]^\intercal)^2 \\
&= (x_{n,1}x_{m,1} + x_{n,2}x_{m,2})^2 \\
&= x_{n,1}^2 x_{m,1}^2 + x_{n,2}^2 x_{m,2}^2 + 2x_{n,1}x_{n,2}x_{m,1}x_{m,2} \\
&= \phi(x_n)^\intercal \phi(x_m)
\end{aligned}$$

Kernel Trick

Consider the following derivation,

$$
\begin{aligned}
({x_n}^\intercal x_m)^2 &= ([x_{n,1}\ x_{n,2}][x_{m,1}\ x_{m,2}]^\intercal)^2 \\
&= (x_{n,1}x_{m,1} + x_{n,2}x_{m,2})^2 \\
&= x_{n,1}^2 x_{m,1}^2 + x_{n,2}^2 x_{m,2}^2 + 2x_{n,1}x_{n,2}x_{m,1}x_{m,2} \\
&= \phi(x_n)^\intercal \phi(x_m)
\end{aligned}
$$

Instead of computing inner product in the high dimensional space, we compute the inner product in the original space.

Kernel Trick

Consider the following derivation,

$$
\begin{aligned}
({x_n}^\intercal x_m)^2 &= ([x_{n,1}\ x_{n,2}][x_{m,1}\ x_{m,2}]^\intercal)^2 \\
&= (x_{n,1}x_{m,1} + x_{n,2}x_{m,2})^2 \\
&= x_{n,1}^2 x_{m,1}^2 + x_{n,2}^2 x_{m,2}^2 + 2x_{n,1}x_{n,2}x_{m,1}x_{m,2} \\
&= \phi(x_n)^\intercal \phi(x_m)
\end{aligned}
$$

Instead of computing inner product in the high dimensional space, we compute the inner product in the original space.

The function

$$K(x_n, x_m) = ({x_n}^\intercal x_m)^2 = \phi(x_n)^\intercal \phi(x_m)$$

is called a **kernel function**.

There must be disadvantages. . .

Given training data $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N) \in \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} \subseteq \mathbb{R}$. Consider *Kernel Ridge Regression* (KRR), with $\phi(\mathcal{X}) \subseteq \mathbb{R}^k$, where $k \to \infty$

$$\mathcal{L}(\mathbf{w}, \lambda) = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{n}^{N} (y_n - \mathbf{w}^\intercal \phi(\mathbf{x}_n))^2 + \lambda \mathbf{w}^\intercal \mathbf{w}.$$

Solving it with Lagrange multipliers α, which is the solution of

$$(\mathbf{K} + \lambda \mathbf{I}_k)\alpha = \mathbf{y},$$

requires $\Theta(k^3)$ time and $\Theta(k^2)$ memory. Here $\mathbf{K} \in \mathbb{R}^{k \times k}$ is the kernel matrix or Gram matrix defined by $\mathbf{K}_{nm} \equiv K(\mathbf{x}_n, \mathbf{x}_m)$.

There must be disadvantages...

Given training data $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N) \in \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} \subseteq \mathbb{R}$. Consider *Kernel Ridge Regression* (KRR), with $\phi(\mathcal{X}) \subseteq \mathbb{R}^k$, where $k \to \infty$

$$\mathcal{L}(\mathbf{w}, \lambda) = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{n}^{N} (y_n - \mathbf{w}^\intercal \phi(\mathbf{x}_n))^2 + \lambda \mathbf{w}^\intercal \mathbf{w}.$$

Solving it with Lagrange multipliers α, which is the solution of

$$(\mathbf{K} + \lambda \mathbf{I}_k)\alpha = \mathbf{y},$$

requires $\Theta(k^3)$ time and $\Theta(k^2)$ memory. Here $\mathbf{K} \in \mathbb{R}^{k \times k}$ is the kernel matrix or Gram matrix defined by $\mathbf{K}_{nm} \equiv K(\mathbf{x}_n, \mathbf{x}_m)$.

Intuition: Can we find a kernel function which lifts $\mathcal{X}$ to $\mathbb{R}^s$, where $d < s \ll k$, while not sacrifices model performance?

Some Prerequisites

Shift Invariant Kernel (Radial Basis Kernel (RBK))

A kernel function $K(\mathbf{x_n}, \mathbf{x_m})$ is called **shift invariant** if it can be written as $K(\mathbf{x_n}, \mathbf{x_m}) = g(\mathbf{x_n} - \mathbf{x_m})$ for some function $g(\cdot)$ (e.g. $K_{Gaussian}(\mathbf{x_n}, \mathbf{x_m}) = \exp(-\gamma\|\mathbf{x_n} - \mathbf{x_m}\|_2^2)$).

Mercer's Theorem

A continuous function $K(\mathbf{x_n}, \mathbf{x_m})$ is a valid kernel function if and only if the kernel matrix $\mathbf{K}$ is **positive definite**.

Bochner's Theorem

A continuous function $g(\cdot)$ is **positive definite** if and only if it is the Fourier transform of a non-negative measure.

Random Fourier Features

Conclusion

A continuous **shift invariant** kernel $K(\mathbf{x_n}, \mathbf{x_m})$, which is **positive definite** (Mercer's Theorem), is the Fourier transform of a non-negative measure $p(\cdot)$.

$$\begin{aligned} \phi(\mathbf{x_n})^\intercal \phi(\mathbf{x_m}) &= K(\mathbf{x_n}, \mathbf{x_m}) = K(\mathbf{x_n} - \mathbf{x_m}) && (1) \\ &= \int_{\mathbb{R}^s} p(\omega) \exp(i\omega^\intercal (\mathbf{x_n} - \mathbf{x_m})) d\omega && (2) \\ &= \mathbb{E}_\omega [\xi(\mathbf{x_n}) \xi(\mathbf{x_m})^*] && (3) \end{aligned}$$

Here $\xi(\mathbf{x}) = \exp(i\omega^\intercal \mathbf{x}) = \begin{bmatrix} \cos(\omega^\intercal \mathbf{x}) \\ \sin(\omega^\intercal \mathbf{x}) \end{bmatrix}$ and hence $\xi(\mathbf{x_n})\xi(\mathbf{x_m})^*$ is an unbiased estimator of $K(\mathbf{x_n}, \mathbf{x_m})$.

Algorithm

Since both the $p(\cdot)$ and $K(\triangle)$ are real-valued, we can replace $\xi(\mathbf{x})$ with $z_\omega(\mathbf{x}) = [\sqrt{2}\cos(\omega^\intercal \mathbf{x} + b)]$ where ω is drawn from $p(\omega)$ and b is uniformly drawn from $[0, 2\pi]$. Then (3) becomes $\mathbb{E}_\omega[z(\mathbf{x_n})z(\mathbf{x_m})^\intercal]$

Algorithm 1 An algorithm with caption

```
Require: n ≥ 0
Ensure: y = x^n
  y ← 1
  X ← x
  N ← n
  while N ≠ 0 do
      if N is even then
          X ← X × X
          N ← N/2                                  ▷ This is a comment
      else if N is odd then
          y ← y × X
          N ← N − 1
      end if
```

Two-column slide

This is a subtitle

This is a text in first column.

$$E = mc^2$$

- First item ℓ
- Second item

This text will be in the second column and on a second thought this is a nice looking layout in some cases.