CS:E4830 Kernel Methods in Machine Learning

Lecture 7 : Algorithms - Kernel Logistic Regression and Bochner's Theorem (for Large-scale Kernel Methods)

Rohit Babbar

21st April, 2021

Some Announcements

- Lecture slides of this (7th) lecture - uploaded to Mycourses
- Assignment 2 deadline next week (28th April)
- Tutorial session for assignment 2 will be tomorrow at 4:15pm

Recall from lecture 3 - Least Square Regression

- Let f be the prediction function, then squared error is given by

$$\ell(f(x), y) = (y - f(x))^2$$

- Let $\mathcal{F}$ be a function class (not necessarily an RKHS) from which we are choosing the desired function f
- Least Square regression finds a function with smallest squared error

$$\hat{f} \in \arg\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} (y_i - f(x_i))^2$$

- Possible problems :
  - Can be unstable in high dimensions
  - Can overfit if the function space $\mathcal{F}$ is too large

Kernel Ridge Regression

- Finding f in an RKHS with a kernel $k(x, x')$
- $$\hat{f} \in \arg\min_{f \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} (y_i - f(x_i))^2 + \lambda ||f||_{\mathcal{H}}$$
- The above formulation has two advantages :
  - Prevents overfitting
  - Representer theorem enables an efficient solution of the form

$$\hat{f}(.) = \sum_{i=1}^{N} \alpha_i k(., x_i)$$

Solving Kernel Ridge Regression

- For the input instances, the prediction by the desired function can be written as follows :

$$(\hat{f}(x_1), \ldots, \hat{f}(x_N))^T = K\boldsymbol{\alpha}$$

- We also know that

$$||f||^2_{\mathcal{H}} = \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j k(x_i, x_j) = \boldsymbol{\alpha}^T K \boldsymbol{\alpha}$$

- Solving Kernel Ridge Regression involves solving

$$\arg\min_{\boldsymbol{\alpha} \in \mathbb{R}^N} \frac{1}{N}(K\boldsymbol{\alpha} - \mathbf{y})^T(K\boldsymbol{\alpha} - \mathbf{y}) + \lambda \boldsymbol{\alpha}^T K \boldsymbol{\alpha}$$

Kernel Ridge Regression - Solution

- Desired optimization problem

$$\arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^N} \frac{1}{N}(K\boldsymbol{\alpha}-\mathbf{y})^T(K\boldsymbol{\alpha}-\mathbf{y})+\lambda\boldsymbol{\alpha}^T K\boldsymbol{\alpha}$$

- The above is convex and differntiable w.r.t to $\boldsymbol{\alpha}$, and can be analytically found by setting the gradient

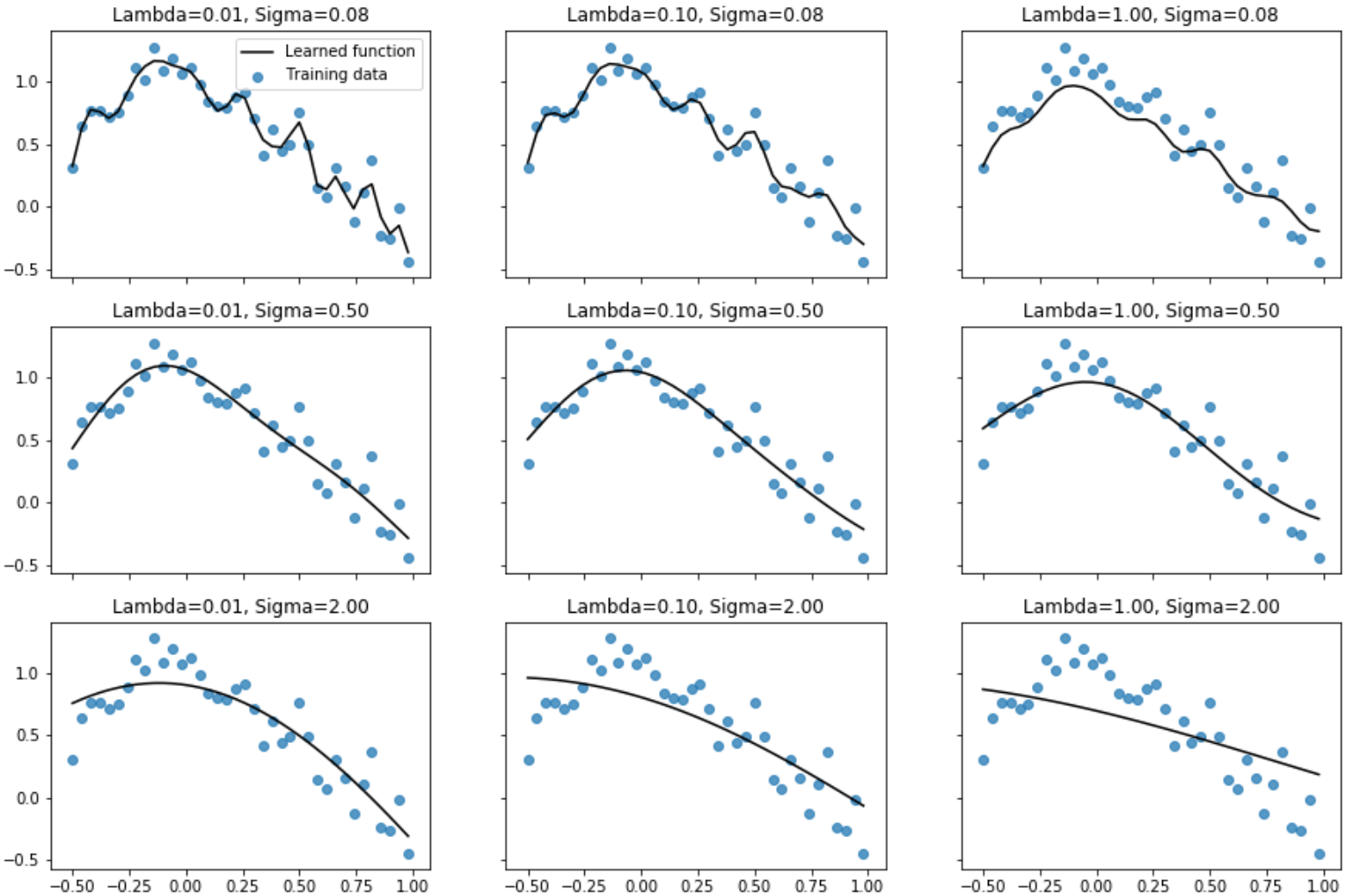
$$\frac{2}{N}K(K\alpha-\mathbf{y})+2\lambda K\alpha=\mathbf{0}$$

- Since K is positive definite (from previous lecture), we can invert $K+\lambda NI$, and hence the solution is given by

$$\boldsymbol{\alpha}=(K+N\lambda I)^{-1}\mathbf{y}$$

where I is the identity matrix

Kernel Ridge Regression with Gaussian Kernel



Weighted Regression

- In ridge regression, we weight each error uniformly
- Suppose, we weigh the error at each training point differently, such that $\beta_i > 0$ is weight of error at point i, then
- The corresponding objective function is

$$\arg\min_{f \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} \beta_i (y_i - f(x_i))^2 + \lambda ||f||_{\mathcal{H}}$$

- How do we solve it?
- Using Representer Theorem, noticing that solution is of the form $\sum_{i=1}^{N} \alpha_i K(x_i, .)$, where $\boldsymbol{\alpha}$ is obtained by solving the following :

$$\arg\min_{\boldsymbol{\alpha} \in \mathbb{R}^N} \frac{1}{N} (K\boldsymbol{\alpha} - y)^T B (K\boldsymbol{\alpha} - y) + \lambda \boldsymbol{\alpha}^T K \boldsymbol{\alpha}$$

where B is a diagonal matrix with weight β_i at the $i-$th diagonal entry

Weighted Regression

- Setting the gradient to $\mathbf{0}$

$$\mathbf{0} = \frac{2}{N}(KBK\boldsymbol{\alpha} - KBy) + 2\lambda K\boldsymbol{\alpha}$$
$$2KBy = 2K(BK + N\lambda I)\boldsymbol{\alpha}$$

- Therefore, desired solution is given by

$$\boldsymbol{\alpha} = B^{\frac{1}{2}}\left(B^{\frac{1}{2}}KB^{\frac{1}{2}} + N\lambda I\right)^{-1}B^{\frac{1}{2}}y$$

Kernel Logistic Regression

Kernel Logistic Regression - logistic loss

- Under the logistic regression model, we model the probability $p(y|x)$ as follows :

$$y \in \{-1, +1\}, p(y|x) = \frac{1}{1 + \exp(-yf(x))} = \sigma(yf(x))$$

  where $f(.) \in \mathcal{H}$ is the desired function
- How does f look like:
  - How did f look like in *Machine Learning : Supervised Methods* course ?

Kernel Logistic Regression - logistic loss

- Under the logistic regression model, we model the probability $p(y|x)$ as follows :

$$y \in \{-1, +1\}, p(y|x) = \frac{1}{1 + \exp(-yf(x))} = \sigma(yf(x))$$

  where $f(.) \in \mathcal{H}$ is the desired function
- How does f look like:
  - How did f look like in *Machine Learning : Supervised Methods* course ?
  - For kernel logistic regression, $f \in \mathcal{H}$ (an RKHS corresponding to a kernel $k(.,.)$.
- Role of $yf(x)$ on training data
  - case y_i and $f(x_i)$ have the same sign
  - case y_i and $f(x_i)$ have opposite sign

Logistic regression

- Logistic regression model assumes a underlying conditional probability:

$$Pr(y|\mathbf{x}) = \frac{\exp(+\frac{1}{2}y\mathbf{w}^T\mathbf{x})}{\exp(+\frac{1}{2}y\mathbf{w}^T\mathbf{x}) + \exp(-\frac{1}{2}y\mathbf{w}^T\mathbf{x})}$$

  where the denominator normalizes the right-hand side to be between zero and one.
- Dividing the numerator and denominator by $\exp(+\frac{1}{2}y\mathbf{w}^T\mathbf{x})$ reveals the logistic function

$$Pr(y|\mathbf{x}) = \phi_{logistic}(y\mathbf{w}^T\mathbf{x}) = \frac{1}{1+\exp(-y\mathbf{w}^T\mathbf{x})}$$

- The margin $y\mathbf{w}^T\mathbf{x}$ is thus interpreted as the log odds of label y vs. label $-y$ given input $\mathbf{x}$:

$$y\mathbf{w}^T\mathbf{x} = \log \frac{Pr(y|\mathbf{x})}{Pr(-y|\mathbf{x})}$$

6

Kernel Logistic Regression - formulation

- To convert the above probability (related to likelihood in MLE) into a loss function,

$$\ell_{logistic}(f(x), y) = -\log(p(y|x)) = \log\left(1 + \exp(-yf(x))\right)$$

  - Converting product of probabilities over samples to sum of log probabilities
- The formulation of Kernel logistic regression, when the desired function f comes from an RKHS $\mathcal{H}$ is given by :

$$\begin{aligned}\hat{f} &= \arg\min_{f\in\mathcal{H}} \frac{1}{N}\sum_{i=1}^{N} \ell_{logistic}(f(x_i), y_i) + \frac{\lambda}{2}||f||^2_{\mathcal{H}} \\ &= \arg\min_{f\in\mathcal{H}} \frac{1}{N}\sum_{i=1}^{N} \log(1 + \exp\left(-y_i f(x_i)\right)) + \frac{\lambda}{2}||f||^2_{\mathcal{H}}\end{aligned}$$

- How do we solve it?

Kernel Logistic Regression - solution

- By representer theorem, any solution to kernel logistic regression is given by

$$\hat{f}(x) = \sum_{i=1}^{N} \alpha_i k(x_i, x)$$

- Also, we have the following :
  - For the input instance, the prediction by the desired function can be written as follows :

$$(\hat{f}(x_1), \ldots, \hat{f}(x_N))^T = K\boldsymbol{\alpha}$$

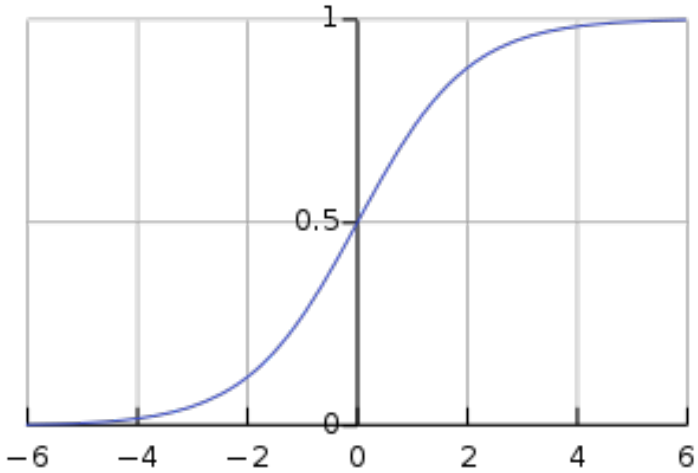
  - We also know that

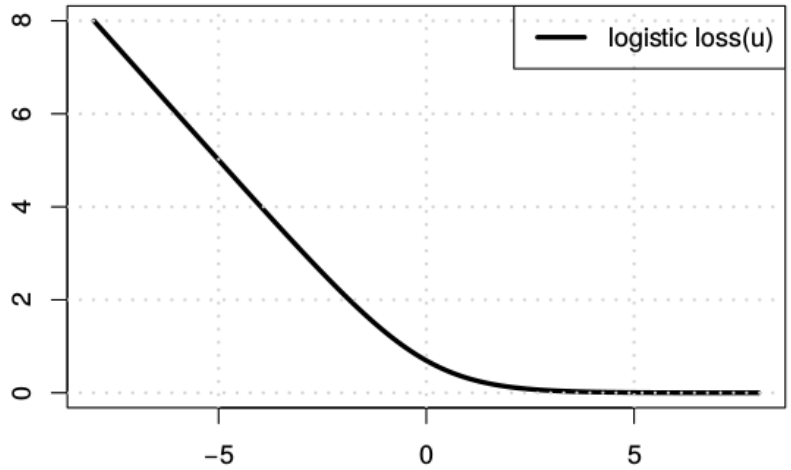
$$||f||_{\mathcal{H}}^2 = \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j k(x_i, x_j) = \boldsymbol{\alpha}^T K \boldsymbol{\alpha}$$

- Therefore, we need to solve the following :

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^N} \frac{1}{N} \sum_{i=1}^{N} \log(1 + \exp(-y_i [K\boldsymbol{\alpha}]_i)) + \frac{\lambda}{2} \boldsymbol{\alpha}^T K \boldsymbol{\alpha}$$

Some facts related to Sigmoid and logistic loss





Sigmoid Function

- $\sigma(z) = \frac{1}{1+\exp(-z)}$
- $\sigma(-z) = 1 - \sigma(z)$
- $\sigma'(z) = \sigma(z)\sigma(-z) \geq 0$

Logistic loss

- $\ell_{logistic}(z) = \log(1 + \exp(-z))$
- $\ell'_{logistic}(z) = -\sigma(-z)$
- $\ell''_{logistic}(z) = \sigma(z)\sigma(-z) \geq 0$

KLR - Optimization

- Recall the objective :

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^N} \frac{1}{N}\sum_{i=1}^{N} \log(1+\exp(-y_i[K\boldsymbol{\alpha}]_i)) + \frac{\lambda}{2}\boldsymbol{\alpha}^T K\boldsymbol{\alpha}$$

  - Can we set the derivative equal to $\mathbf{0}$ as before?
  - No closed form (KRR or Weighted KRR)!
- Newton's method - 2nd order Taylor series approximation [1]

$$J_q(\boldsymbol{\alpha}) = J(\boldsymbol{\alpha}_0) + (\boldsymbol{\alpha}-\boldsymbol{\alpha}_0)^T\nabla J(\boldsymbol{\alpha}_0) + \frac{1}{2}(\boldsymbol{\alpha}-\boldsymbol{\alpha}_0)^T\nabla^2 J(\boldsymbol{\alpha}_0)(\boldsymbol{\alpha}-\boldsymbol{\alpha}_0)$$

  where
  - $\nabla J(\boldsymbol{\alpha}_0) \in \mathbb{R}^N$ is the gradient of the original objective function at $\boldsymbol{\alpha}_0$, and
  - $\nabla^2 J(\boldsymbol{\alpha}_0) \in \mathbb{R}^{N\times N}$ is the Hessian matrix of the original objective function at $\boldsymbol{\alpha}_0$
- The famous gradient descent makes a first order approximation. Which one is better, and under what conditions ?

[1] `https://en.wikipedia.org/wiki/Taylor_series`

KLR - Gradient and Hessian

- Gradient $\nabla J(\boldsymbol{\alpha})$

$$\frac{\partial J}{\partial \boldsymbol{\alpha}_j} = \frac{1}{N}\sum_{i=1}^{N} \underbrace{\ell'_{logistic}(y_i[K\boldsymbol{\alpha}]_i)}_{P_i(\boldsymbol{\alpha})} y_i K_{ij} + \lambda[K\boldsymbol{\alpha}]_j$$

  - In vector notation,

$$\nabla J(\boldsymbol{\alpha}) = \frac{1}{N} KP(\boldsymbol{\alpha})y + \lambda K\boldsymbol{\alpha}$$

  where $P(\boldsymbol{\alpha}) = diag(P_1(\boldsymbol{\alpha}), \ldots, P_N(\boldsymbol{\alpha}))$ and $P_i(\boldsymbol{\alpha}) = \ell'_{logistic}(y_i[K\boldsymbol{\alpha}]_i)$

- Hessian $\nabla^2 J(\boldsymbol{\alpha})$

$$\frac{\partial^2 J}{\partial \boldsymbol{\alpha}_j \partial \boldsymbol{\alpha}_l} = \frac{1}{N}\sum_{i=1}^{N} \underbrace{\ell''_{logistic}(y_i[K\boldsymbol{\alpha}]_i)}_{B_i(\boldsymbol{\alpha})} y_i K_{ij} y_i K_{il} + \lambda[K]_{jl}$$

  - In matrix notation,

$$\nabla^2 J(\boldsymbol{\alpha}) = \frac{1}{N} KB(\boldsymbol{\alpha})K + \lambda K \text{ note that } y_1 \times y_i = 1$$

  where $B(\boldsymbol{\alpha}) = diag(B_1(\boldsymbol{\alpha}), \ldots, B_n(\boldsymbol{\alpha}))$

KLR - Computing the quadratic approximation

- Recall the quadratic approximation :

$$J_q(\boldsymbol{\alpha}) = J(\boldsymbol{\alpha}_0) + (\boldsymbol{\alpha} - \boldsymbol{\alpha}_0)^T \nabla J(\boldsymbol{\alpha}_0) + \frac{1}{2}(\boldsymbol{\alpha} - \boldsymbol{\alpha}_0)^T \nabla^2 J(\boldsymbol{\alpha}_0)(\boldsymbol{\alpha} - \boldsymbol{\alpha}_0)$$

- Terms depending on $\boldsymbol{\alpha}$
  - $\boldsymbol{\alpha}^T \nabla J(\boldsymbol{\alpha}_0) = \frac{1}{N}\boldsymbol{\alpha}^T K P(\boldsymbol{\alpha}_0) y + \lambda \boldsymbol{\alpha}^T K \boldsymbol{\alpha}_0$,
  - $\frac{1}{2}\boldsymbol{\alpha}^T \nabla^2 J(\boldsymbol{\alpha}_0)\boldsymbol{\alpha} = \frac{1}{2N}\boldsymbol{\alpha}^T K B(\boldsymbol{\alpha}_0) K \boldsymbol{\alpha} + \frac{\lambda}{2}\boldsymbol{\alpha}^T K \boldsymbol{\alpha}$,
  - $-\boldsymbol{\alpha}^T \nabla^2 J(\boldsymbol{\alpha}_0)\boldsymbol{\alpha}_0 = -\frac{1}{N}\boldsymbol{\alpha}^T K B(\boldsymbol{\alpha}_0) K \boldsymbol{\alpha}_0 - \lambda \boldsymbol{\alpha}^T K \boldsymbol{\alpha}_0$,
- Aggregating terms,

$$\begin{aligned} 2J_q(\boldsymbol{\alpha}) &= -\frac{2}{N}\boldsymbol{\alpha}^T K B(\boldsymbol{\alpha}_0) \underbrace{(K\boldsymbol{\alpha}_0 - B^{-1}(\boldsymbol{\alpha}_0) P(\boldsymbol{\alpha}_0) y)}_{:=u} + \frac{1}{N}\boldsymbol{\alpha}^T K B(\boldsymbol{\alpha}_0) K \boldsymbol{\alpha} \\ &\quad + \lambda \boldsymbol{\alpha}^T K \boldsymbol{\alpha} + constant \\ &= \frac{1}{N}(K\boldsymbol{\alpha} - u)^T B(\boldsymbol{\alpha}_0)(K\boldsymbol{\alpha} - u) + \lambda \boldsymbol{\alpha}^T K \boldsymbol{\alpha} + const. \end{aligned}$$

The above is same as Weighted Kernel Ridge regression with weight matrix $B(\boldsymbol{\alpha}_0)$!

Random Fourier Features

Kernel Matrix Bottleneck - from Lecture 3

In many problem scenarios in modern day big data setup, it is not difficult to get millions (or even bigger) of training samples :

- Computing a million $\times$ million kernel matrix is not trivial
  - For instance, one needs to invert the kernel matrix for kernel ridge regression (as we will see later) - complexity $O(N^3)$ for N training data points
- The computational bottle-neck also limits hyper-parameter tuning

Explicit feature map for Polynomial kernel with unigram + bi-grams features - From lecture 3

Prediction function can involve non-linear combination of features

- For the classification function f_2 below, which is linear in weights and **non-linear in input features**

$$f_2(x) = w^{(1)}x^{(1)} + w^{(2)}x^{(2)} + w^{(3)}x^{(1)}x^{(2)} + w^{(4)}x^{(2)}x^{(1)}$$

- Here, the decision function $f_2(x)$ is trying to capture **non-linear combination** of the input components as well such as $x^{(1)}x^{(2)}, x^{(2)}x^{(1)}$
- Non-linear feature map $\phi_2 : \mathbb{R}^2 \mapsto \mathbb{R}^4$, and is given by $\phi_2(x) = (x^{(1)}, x^{(2)}, x^{(1)}x^{(2)}, x^{(2)}x^{(1)})^T$
  - $\phi_2(x) \in \mathcal{H}$, which is referred to as the feature space
- **Importantly**, it is computationally much easier to solve a linear classification or regression problem than a non-linear problem such as doing pairwise computation as required for the kernel matrix

More on Explicit Feature Maps

- The **exact** feature map for Gaussian kernel is infinite dimensional
- Is it still possible to **approximate** it somehow ?

Fourier transforms

- **Fourier transform** $S(\omega)$ of a function $f(x)$,

$$S(\omega) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x \omega} dx$$

where

- i is the imaginary number with $i^2 = -1$ and $i^0 = 1$
- ω is a frequency

Fourier transforms

- **Fourier transform** $S(\omega)$ of a function $f(x)$,

$$S(\omega) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x \omega} dx$$

  where
  - i is the imaginary number with $i^2 = -1$ and $i^0 = 1$
  - ω is a frequency
- **Inverse Fourier transform** $f(x)$ of spectral density $S(\omega)$,

$$f(x) = \int_{-\infty}^{\infty} S(\omega) e^{2\pi i x \omega} d\omega$$

Fourier transforms

- **Fourier transform** $S(\omega)$ of a function $f(x)$,

$$S(\omega) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x \omega} dx$$

  where
  - i is the imaginary number with $i^2 = -1$ and $i^0 = 1$
  - ω is a frequency
- **Inverse Fourier transform** $f(x)$ of spectral density $S(\omega)$,

$$f(x) = \int_{-\infty}^{\infty} S(\omega) e^{2\pi i x \omega} d\omega$$

- Euler's identity helps compute Fouriers in practice

$$e^{ix} = \underbrace{\cos x}_{\text{real part}} + \underbrace{i \cdot sinx}_{\text{complex part}}$$

Fourier transforms

- **Fourier transform** $S(\omega)$ of a function $f(x)$,

$$S(\omega) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x \omega} dx$$

  where
  - i is the imaginary number with $i^2 = -1$ and $i^0 = 1$
  - ω is a frequency
- **Inverse Fourier transform** $f(x)$ of spectral density $S(\omega)$,

$$f(x) = \int_{-\infty}^{\infty} S(\omega) e^{2\pi i x \omega} d\omega$$

- Euler's identity helps compute Fouriers in practice

$$e^{ix} = \underbrace{\cos x}_{\text{real part}} + \underbrace{i \cdot sinx}_{\text{complex part}}$$

- Hence,

$$e^{-2\pi i x \omega} = \cos(2\pi x \omega) - i \sin(2\pi x \omega)$$
$$e^{2\pi i x \omega} = \cos(2\pi x \omega) + i \sin(2\pi x \omega)$$

Fourier Transform

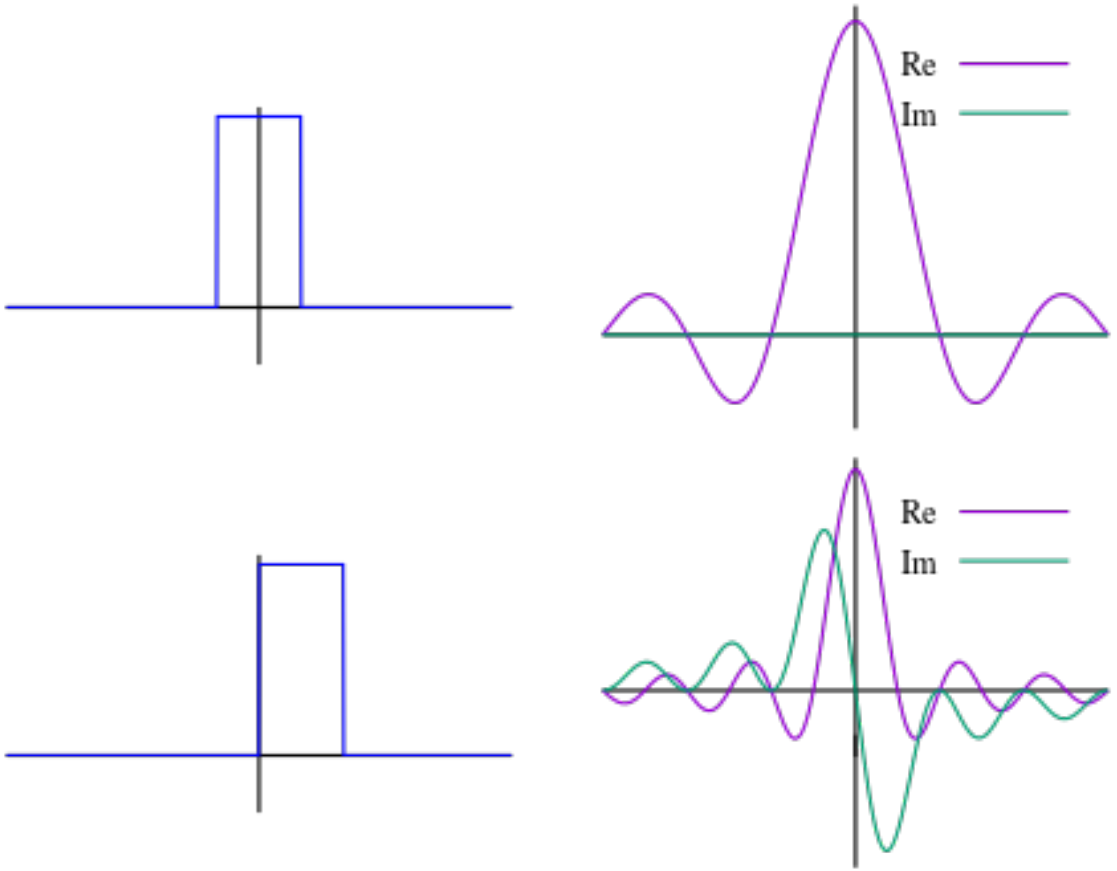


Figure: Symmetric pulse (top left) and a translated version (bottom left), and their respective Fourier transforms on the right (picture from Wikipedia)

Bochner's Theorem

Let the input space $\mathcal{X} = \mathbb{R}^D$

- *Definition* - A kernel $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is said to be translation invariant if it satisfies $k(x, x') := \psi(x - x')$ for a positive definite function $\psi(.)$ (with one argument) on $\mathbb{R}^D$.

Bochner's Theorem

Let the input space $\mathcal{X} = \mathbb{R}^D$

- *Definition* - A kernel $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is said to be translation invariant if it satisfies $k(x, x') := \psi(x - x')$ for a positive definite function $\psi(.)$ (with one argument) on $\mathbb{R}^D$.
- Example of translation invariant kernel ?

Bochner's Theorem

Let the input space $\mathcal{X} = \mathbb{R}^D$

- *Definition* - A kernel $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is said to be translation invariant if it satisfies $k(x, x') := \psi(x - x')$ for a positive definite function $\psi(.)$ (with one argument) on $\mathbb{R}^D$.
- Example of translation invariant kernel ?
- Bochner theorem (stated in somewhat simpler terms below) gives a complete characterization of a translation invariant kernel function

Theorem (Bochner)

A continuous function $\psi : \mathbb{R}^D \mapsto \mathbb{R}$ is positive definite if and only if it is the inverse fourier transform of a probability density function.

$$\psi(\tau) = \int_{-\infty}^{\infty} p(\omega) e^{2\pi i \omega^T \tau} d\omega \qquad (p(\omega) \text{ is a probability density function})$$

Bochner's Theorem

Let the input space $\mathcal{X} = \mathbb{R}^D$

- *Definition* - A kernel $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is said to be translation invariant if it satisfies $k(x, x') := \psi(x - x')$ for a positive definite function $\psi(.)$ (with one argument) on $\mathbb{R}^D$.
- Example of translation invariant kernel ?
- Bochner theorem (stated in somewhat simpler terms below) gives a complete characterization of a translation invariant kernel function

Theorem (Bochner)

A continuous function $\psi : \mathbb{R}^D \mapsto \mathbb{R}$ is positive definite if and only if it is the inverse fourier transform of a probability density function.

$$\psi(\tau) = \int_{-\infty}^{\infty} p(\omega) e^{2\pi i \omega^T \tau} d\omega \qquad (p(\omega) \text{ is a probability density function})$$

What other characterizations of kernels we have seen already?

Bochner's Theorem

Let the input space $\mathcal{X} = \mathbb{R}^D$

- *Definition* - A kernel $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is said to be translation invariant if it satisfies $k(x, x') := \psi(x - x')$ for a positive definite function $\psi(.)$ (with one argument) on $\mathbb{R}^D$.
- Example of translation invariant kernel ?
- Bochner theorem (stated in somewhat simpler terms below) gives a complete characterization of a translation invariant kernel function

Theorem (Bochner)

A continuous function $\psi : \mathbb{R}^D \mapsto \mathbb{R}$ is positive definite if and only if it is the inverse fourier transform of a probability density function.

$$\psi(\tau) = \int_{-\infty}^{\infty} p(\omega) e^{2\pi i \omega^T \tau} d\omega \qquad (p(\omega) \text{ is a probability density function})$$

What other characterizations of kernels we have seen already?

- Existence of a feature map $\phi(.)$ and a feature space $\mathcal{H}$ (recall lecture 1)
- Postive definiteness (recall lecture 2)

Simplified Representation for Symmetric Distributions

- Assume symmetric distribution $p(\omega) = p(-\omega)$
- Euler's identity $e^{\pm ix} = \cos x \pm i \sin x$
- Sine identity $\sin(-x) = -\sin(x)$
- Then we can solve the inverse Fourier representation on previous slide as follows

Simplified Representation for Symmetric Distributions

- Assume symmetric distribution $p(\omega) = p(-\omega)$
- Euler's identity $e^{\pm ix} = \cos x \pm i \sin x$
- Sine identity $\sin(-x) = -\sin(x)$
- Then we can solve the inverse Fourier representation on previous slide as follows

$$
\begin{aligned}
\psi(\tau) &= \int_{-\infty}^{\infty} p(\omega) e^{2\pi i \omega^T \tau} d\omega \\
&= \int_{-\infty}^{\infty} p(\omega) \cos(2\pi\omega^T\tau) d\omega + \int_{-\infty}^{\infty} i \cdot p(\omega) \sin(2\pi\omega^T\tau) d\omega \\
&= \mathbb{E}_\omega[\cos(2\pi\omega^T\tau)] + \int_{-\infty}^{0} i \cdot p(\omega) \sin(2\pi\omega^T\tau) d\omega + \int_{0}^{\infty} i \cdot p(\omega) \sin(2\pi\omega^T\tau) d\omega \\
&= \mathbb{E}_\omega[\cos(2\pi\omega^T\tau)] + \int_{0}^{\infty} i \cdot p(\omega) \sin(2\pi(-\omega)^T\tau) d\omega + \int_{0}^{\infty} i \cdot p(\omega) \sin(2\pi\omega^T\tau) d\omega \\
&= \mathbb{E}_\omega[\cos(2\pi\omega^T\tau)] - \int_{0}^{\infty} i \cdot p(\omega) \sin(2\pi\omega^T\tau) d\omega + \int_{0}^{\infty} i \cdot p(\omega) \sin(2\pi\omega^T\tau) d\omega \\
&= \mathbb{E}_\omega[\cos(2\pi\omega^T\tau)]
\end{aligned}
$$

Bochner's Theorem - Consequence I

Given an integrable function $\psi(\tau)$, i.e. $\int_{\mathbb{R}^D} |\psi(\tau)| < \infty$, compute its Fourier transform as follows

Checking for positive definiteness of a function

$$p(\omega) = \int_{-\infty}^{\infty} \psi(\tau) e^{-2\pi i \omega^T \tau} d\tau,$$

If $p(\omega)$ is non-negative $\forall \omega \in \mathbb{R}^D$, then $\psi(.)$ is a positive definite function, and hence $k(\mathbf{x}, \mathbf{x}')$ is a kernel

Bochner's Theorem - Consequence II

Recall from our notation $\psi(\tau) := \psi(\mathbf{x} - \mathbf{x}') := k(\mathbf{x}, \mathbf{x}')$

$$
\begin{aligned}
\psi(\tau) &= \mathbb{E}_\omega[\cos(2\pi\omega^T\tau)] \\
&= \mathbb{E}_\omega[\cos(2\pi\omega^T(\mathbf{x} - \mathbf{x}'))] \\
&\approx \frac{1}{L}\sum_{\ell=1}^{L}[\cos(2\pi\omega_\ell^T(\mathbf{x} - \mathbf{x}'))] \\
&= \frac{1}{L}\sum_{\ell=1}^{L}[\cos(2\pi\omega_\ell^T\mathbf{x})\cos(2\pi\omega_\ell^T\mathbf{x}') + \sin(2\pi\omega_\ell^T\mathbf{x})\sin(2\pi\omega_\ell^T\mathbf{x}')] \\
&= \frac{1}{L}\sum_{\ell=1}^{L}\left\langle[\cos(2\pi\omega_\ell^T\mathbf{x}), \sin(2\pi\omega_\ell^T\mathbf{x})], [\cos(2\pi\omega_\ell^T\mathbf{x}'), \sin(2\pi\omega_\ell^T\mathbf{x}')]\right\rangle \\
&= \langle\phi(\mathbf{x}), \phi(\mathbf{x}')\rangle
\end{aligned}
$$

where $\phi(\mathbf{x}) = \frac{1}{\sqrt{L}}[\ldots, \cos(2\pi\omega_\ell^T\mathbf{x}), \sin(2\pi\omega_\ell^T\mathbf{x}), \ldots] \in \mathbb{R}^{2L}$ is the approximate non-linear feature map. L is the number of samples drawn in order to approximate the calculation of the expectation.

Algorithm for Computing Random Fourier Features

Require: Training data $\{\mathbf{x}_i\}_{i=1}^{N}$ such that $\mathbf{x}_i \in \mathbb{R}^D$and a kernel $k(.,.)$ (in the form of translation invariant function $\psi(\tau)$)

Ensure: Non-linear Feature maps $\phi(\mathbf{x})$ for instance $\mathbf{x} \in \mathbf{X}$

1: Compute Fourier transform of the kernel to get the probability density $p(w) = \int_{-\infty}^{\infty} \psi(\tau) e^{-2\pi i \omega^T \tau} d\tau$

2: Draw L i.i.d. samples $\omega_1, \ldots, \omega_L \in \mathbb{R}^D$ from the distribution p

3: For each training instance $\mathbf{x}_i$, return the corresponding feature map $\phi(\mathbf{x}_i) = \frac{1}{\sqrt{L}}[\ldots, \cos(2\pi\omega_\ell^T \mathbf{x}_i), \sin(2\pi\omega_\ell^T \mathbf{x}_i), \ldots]$

This idea here is similar to the Polynomial kernel example we covered in Lecture 3. Namely, for the case of large sample sizes, N, one can train a linear classifier on top of the random feature representation (explicit feature space representation) as obtained above.

Summary

- Review of Kernel Ridge Regression
- Weighted version of Kernel Ridge Regression
- Logistic Regression
  - Classification setup
  - Solving via Second order Newton's method
- Random Fourier features
  - Approximation for Gaussian kernel in large-scale regimes

References

- The first part of the lecture is based on a similar course by Julien Mairal's at ENS Paris
- Random fourier Features - Random Features for Large-Scale Kernel Machines, Neurips 2007
  - `https://people.eecs.berkeley.edu/~brecht/papers/07.rah.rec.nips.pdf`