CS:E4830 Kernel Methods in Machine Learning

Lecture 6 : Algorithms - Kernel Ridge and Logistic Regression

Rohit Babbar[1]

13th February, 2019

[1]Parts of the material based on Lectures by Julien Mairal at ENS Paris

Regression

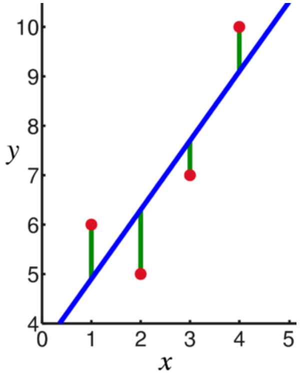


Figure: Linear regression, Picture from Wiki

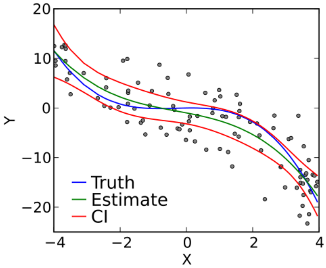


Figure: Non-linear regression, Picture from Wiki

Regression - Numerous applications in various fields

- Original analyses dates back to ealy 1800's - Gauss and Legendre
- Kernel versions are quite recent - couple of decades ago

How to tackle infinite dimensional regression problems - Representer Theorem

- For the following optimization

$$f_{\mathcal{H}} := \arg\min_f \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(x_i)) + \lambda\theta(||f||_{\mathcal{H}})$$

  where $\ell(.,.)$ is the loss function and $\theta : [0, \infty) \mapsto \mathbb{R}$ is non-decreasing function
- Even though the above problem is potentially an infinite dimensional optimization problem, **Representer Theorem** states its solution can be expressed in the following form

$$f_{\mathcal{H}}(.) = \sum_{i=1}^{n} \alpha_i k(., x_i)$$

  where $\alpha_i \in \mathbb{R}$, i.e. it is linear combination of kernel evaluations

Implications of Representer Theorem

- Representer Theorem allows us to look for the solutions of the following form:

$$f_{\mathcal{H}}(.) = \sum_{i=1}^{n} \alpha_i k(., x_i)$$

- Implications
  - The desired function just involves kernel computation on **training points only** via $k(., x_i)$ in the above solution
  - It reduces the problem of finding $f \in \mathcal{H}$ which could be infinite dimensional to a finite dimensional problem
    - We just need to find the coefficients of the finite linear combination $\alpha_1, \ldots, \alpha_n$
  - Also, we can reforumlate the original objective function in the new form (more in next slide)

Reformulating the Objective using Representer Theorem - I

- Recall the original objective :

$$f_{\mathcal{H}} := \arg\min_{f} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(x_i)) + \lambda\theta(||f||_{\mathcal{H}})$$

- For the *j*-th **training point**,

$$f_{\mathcal{H}}(x_j) = \sum_{i=1}^{n} \alpha_i k(x_i, x_j) = [K\boldsymbol{\alpha}]_j$$

which is the j-th element of the matrix-vector product $K\boldsymbol{\alpha}$

Reformulating the Objective using Representer Theorem - II

- Rewriting the regularization term :

$$
\begin{aligned}
||f||_{\mathcal{H}}^2 &= \langle f(.), f(.) \rangle \text{ (Think of } f(.) \text{ as your classifier, a shallow network)} \\
&= \left\langle \sum_{i=1}^{n} \alpha_i k(., x_i), \sum_{i=1}^{n} \alpha_i k(., x_i) \right\rangle \text{ (using representer theorem)} \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(x_i, x_j) \text{ (evaluation of dot product)} \\
&= \boldsymbol{\alpha}^T K \boldsymbol{\alpha} \text{ (writing in matrix notation)}
\end{aligned}
$$

Reformulating the Objective using Representer Theorem - III

Using the above substitutions, the original (intractable) objective

$$f_{\mathcal{H}} := \arg\min_{f} \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f(x_i)) + \lambda\theta(||f||_{\mathcal{H}})$$

translates to an equivalent (tractable) form below

$$f_{\mathcal{H}} := \arg\min_{f} \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, [K\boldsymbol{\alpha}]_i) + \lambda\theta(\boldsymbol{\alpha}^T K\boldsymbol{\alpha})$$

Least Squares Regression

- For the squared error as the loss function,

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

- Let $\mathcal{H}$ be a function class (not necessarily an RKHS) from which we are choosing our function
- Least Square regression (without regularization) find a function with smallest sqaured error

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

- Possible problems :
  - Can be unstable in high dimensions (more in next slide)
  - Overfit if the function space $\mathcal{H}$ is too large

Linear Regression - without Representer Theorem

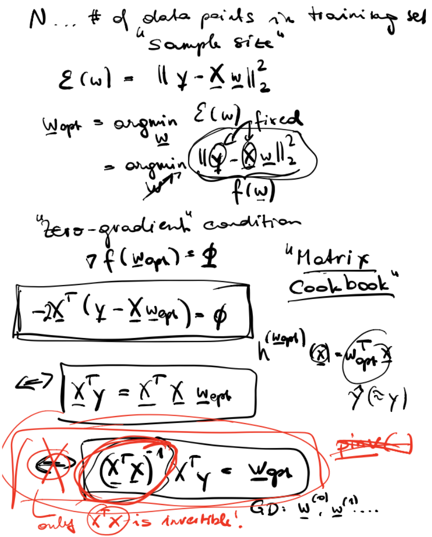


Figure: Picture from Machine Learning Basic Principles Course by Alex Jung

Solving Kernel Ridge Regression

- Let's denote by
  - $y \in \mathbb{R}^n$, the label vector denoting the true values for the inputs
  - The kernel matrix K, where $K_{ij} = K(x_i, x_j)$
  - $\boldsymbol{\alpha} \in \mathbb{R}^n$, the co-efficients we want to find
- 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

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

- Solving Kernel Ridge Regression involves solving

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

Kernel Ridge Regression - Solution

- Desired optimization problem (recall that y is a vector true target values)

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

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

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

- K being a kernel matrix is positive definite and hence invertible
- Also, we can invert $K + \lambda nI$, and hence the solution is given by

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

Kernel Ridge Regression with Gaussian Kernel

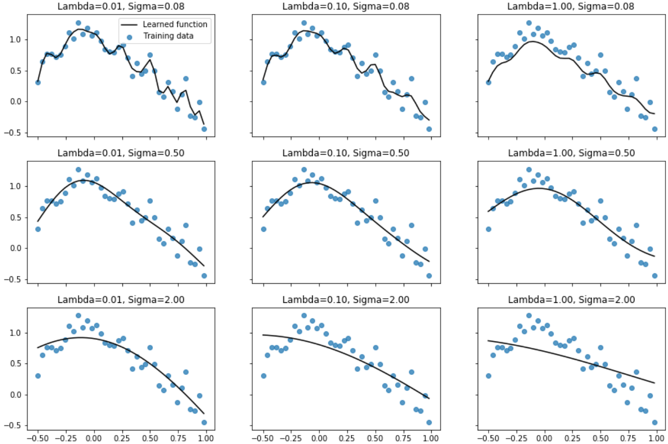


Figure: Impact of varying σ and regularization λ (Figures by Eric Bach)

Kernel Ridge Regression with Linear Kernel

- Let $\mathcal{X} = \mathbb{R}^d$, i.e. we have d-dimensional data,
- Suppose we have n data points, the data matrix $X = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^{n \times d}$,
- For the linear kernel $k(x_i, x_j) = x_i^T x_j$, the kernel matrix is given by $K = XX^T$ (kernel matrix should be $n \times n$, as it is a pairwise similarity measure between data points)
- From Representer Theorem,

$$\boldsymbol{\alpha} = (K + \lambda n I)^{-1} y = (XX^T + \lambda n I)^{-1} y$$

Here $K = XX^T \in \mathbb{R}^{n \times n}$, and $y \in \mathbb{R}^n$ is a vector of the true target values

- Therefore, the weight vector is given by

$$w_{KRR-L} = \sum_{i=1}^{n} \boldsymbol{\alpha}_i x_i = X^T \boldsymbol{\alpha} = X^T (XX^T + \lambda n I)^{-1} y$$

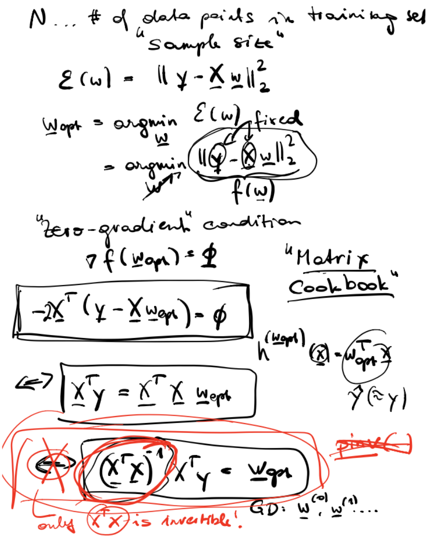


Figure: Picture from Machine Learning Basic Principles Course by Alex Jung

Linear Regression from *Machine Learning Basic Principles*

- Important - without using Representer Theorem
- Recall original problem :

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

- For linear case, the function is $f(x_i) = w^T x_i$, and the RKHS norm is $||w||^2$, therefore, we have

$$\arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (y_i - w^T x_i)^2 + \lambda ||w||^2$$

- Rewriting $\arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (y_i - w^T x_i)^2 + \lambda ||w||^2$
- Setting, the gradient of above to 0 (to find the minimum), we get

$$w_{LR} = (X^T X + \lambda n I)^{-1} X^T y$$

Equivalence and Matrix Inversion Lemma

Matrix Inversion Lemma

For any matrices P and Q, and $\gamma > 0$, the following is true :

$$P(QP + \gamma I)^{-1} = (PQ + \gamma I)^{-1} P$$

This implies

$$\begin{aligned} w_{KRR-L} &= X^T(XX^T + \lambda n I)^{-1} y \text{ (inverting } n \times n \text{ matrix)} \\ &= (X^T X + \lambda n I)^{-1} X^T y \text{ (inverting } d \times d \text{ matrix)} \\ &= w_{LR} \end{aligned}$$

What should we prefer if we have

- More features than observations?
- More observations than features?

Weighted Regression

- In the previous, 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?

Weighted Regression

- In the previous, 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 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 $\mathbf{0}$

$$\begin{aligned}\mathbf{0} &= \frac{2}{n}(KBK\boldsymbol{\alpha} - KBy) + 2\lambda K\boldsymbol{\alpha} \\ &= \frac{2}{n}KB^{\frac{1}{2}}\left[\left(B^{\frac{1}{2}}KB^{\frac{1}{2}} + n\lambda I\right)B^{-\frac{1}{2}}\boldsymbol{\alpha} - B^{\frac{1}{2}}y\right]\end{aligned}$$

- 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

Logistic Regression - an algorithm for classification

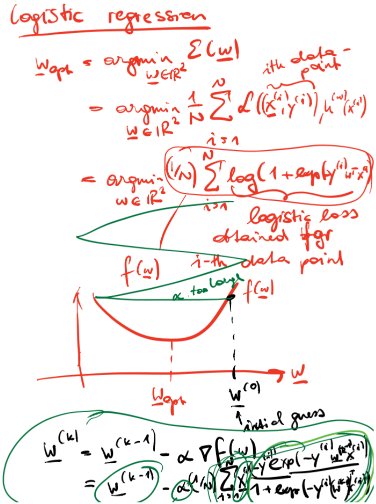


Figure: Picture from Machine Learning Basic Principles Course by Alex Jung

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:
  - For ML Basic principles, f is a linear function,
  - 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

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 fomulation of Kernel logistic regression, when the desired function f comes from an RKHS $\mathcal{H}$ is given by :

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

- 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 :

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

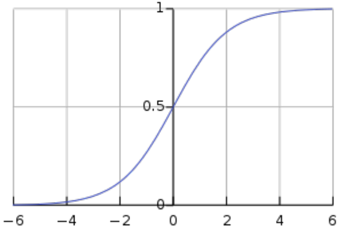
  - We also know that

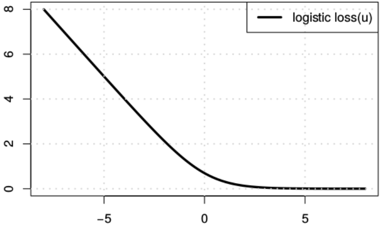
$$||f||_{\mathcal{H}}^2 = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j k(x_i, x_j) = \boldsymbol{\alpha}^T K \boldsymbol{\alpha}$$

- 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?

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)!

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

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

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 $\boldsymbol{\alpha}_0$

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

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

  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 decent makes a first order approximation. Which one is better ?

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,

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

  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,

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

  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,

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

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

Summary

- Review of Representer Theorem
- Kernel Ridge Regression
  - Non-linear regression
  - Relation to Linear Regression
- Logistic Regression
  - Classification setup
  - Solving via Second order Newton's method

- Further details on Kernel Ridge Regression - JST & Christianini book, Chapter 2
- Some of the material is based on Lecture slides by Julien Mairal's lectures notes on a similar course at ENS Paris