EQ2415 – Machine Learning and Data Science
HT22

Tutorial 1

A. Honoré, A. Ghosh

1 Inference in linear models

1.1 Projection on a line.

The set $\mathcal{L} = \{\beta \mathbf{u} \mid \beta \in \mathbb{R}\}$ where $\mathbf{u} \in \mathbb{R}^d$ is a unit vector, defines a line of points that may be obtained by varying the value of β.

Question 1. Derive an expression for the point $\mathbf{y}$ that lies on this line $\mathcal{L}$, and that is as close as possible (according to the Euclidean distance) to an arbitrary point $\mathbf{x} \in \mathbb{R}^d$. This operation of replacing a point by its nearest member within some set is called projection.

Question 2. Write a small Python program that calculates the projection of random points $\mathbf{y}$ on the line generated by a unit vector $\mathbf{u}$. Use a space of dimension $d = 2$.

1.2 Some matrix algebra

Let $m, n > 0$.

Vector by scalar Suppose that a vector $\mathbf{y} \in \mathbb{R}^m$ is dependent upon a scalar $\alpha \in \mathbb{R}$. Then the derivative of $\mathbf{y}$ with respect to α is the vector:

$$J = \frac{\partial \mathbf{y}}{\partial \alpha} = \begin{bmatrix} \frac{dy_1}{d\alpha} \\ \vdots \\ \frac{dy_m}{d\alpha} \end{bmatrix} \tag{1}$$

Scalar by vector Suppose that a scalar $x \in \mathbb{R}$ depends upon a vector $\mathbf{y} \in \mathbb{R}^m$. Then the derivative of x with respect to $\mathbf{y}$ is the (row) vector:

$$J = \frac{\partial x}{\partial \mathbf{y}} = \begin{bmatrix} \frac{\partial x}{\partial y_1} & \dots & \frac{\partial x}{\partial y_m}. \end{bmatrix} \tag{2}$$

Vector by vector Suppose we have m real valued multivariate functions $f_i : \mathbb{R}^n \to \mathbb{R}$. Suppose also that we have a multivariate function $f : \mathbb{R}^n \to \mathbb{R}^m$ is such that for some $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^m$,

$$\mathbf{y} = f(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_m(\mathbf{x})). \tag{3}$$

The Jacobian matrix J, of the multivariate function f, has elements

$$J_{ij} = \frac{\partial f_i(\mathbf{x})}{\partial x_j}, \text{ for } i = 1, \dots, m \text{ and } j = 1, \dots, n, \tag{4}$$

i.e. can be written in matrix form:

$$J = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial f_m(\mathbf{x})}{\partial x_n} \end{bmatrix}. \tag{5}$$

Scalar by matrix Suppose a scalar $x \in \mathbb{R}$ is dependent upon a matrix $M \in \mathbb{R}^{m \times n}$. Then the derivative of x wrt that matrix is written in matrix form:

$$J = \frac{\partial x}{\partial M} = \begin{bmatrix} \frac{\partial x}{\partial M_{11}} & \cdots & \frac{\partial x}{\partial M_{1m}} \\ \vdots & & \vdots \\ \frac{\partial x}{\partial M_{n1}} & \cdots & \frac{\partial x}{\partial M_{nm}} \end{bmatrix} \tag{6}$$

Suppose that for $m, n > 0$, we have $\mathbf{a} \in \mathbb{R}^n$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{c} \in \mathbb{R}^m$ and $M \in \mathbb{R}^{n \times m}$.

Question 1. Calculate the Jacobian:

1. $\dfrac{\partial \mathbf{x}^T \mathbf{a}}{\partial \mathbf{x}} =$
2. $\dfrac{\partial \mathbf{a}^T \mathbf{x}}{\partial \mathbf{x}} =$
3. $\dfrac{\partial \mathbf{a}^T M \mathbf{b}}{\partial M} =$
4. $\dfrac{\partial \mathbf{b}^T M^T M \mathbf{c}}{\partial M} =$
5. $\dfrac{\partial ||\mathbf{x}||^2}{\partial \mathbf{x}} =$

1.3 Minimum mean square error

Suppose that we can observe two random variables $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{y} \in \mathbb{R}^q$. Suppose also that these variables are related, and that we model this relation by a linear model parameterized with a matrix $A \in \mathbb{R}^{q \times d}$, i.e. such that

$$\mathbf{y} = A\mathbf{x}. \tag{7}$$

Question 1. Find $A^\star$ leading to the minimum mean square error, i.e. find $A^\star$ such that

$$A^\star = \arg\min_A \mathbb{E}[||\mathbf{y} - A\mathbf{x}||^2]. \tag{8}$$

Question 2. Suppose that you are given n data points for $\mathbf{x}$ and $\mathbf{y}$, in the form of matrices $X \in \mathbb{R}^{d \times n}$ and $Y \in \mathbb{R}^{q \times n}$ respectively.

Express $A_n^\star$, the value of A leading to minimum mean square error, as a function of the matrices X and Y.

Question 3. The solution above can lead to overfitting, especially when a few data points are provided. Also XX^T may not be invertible. We resort to regularization in these cases. We find $A_n^\star$ such that

$$A_n^\star = \arg\min_A ||Y - AX||_F^2 + \lambda ||A||_F^2, \tag{9}$$

where $\lambda > 0$.

How is $A_n^\star$ calculated in this case ?

Question 4. Implement the solutions to Questions 2 and 3 in Python. Generate data with a linear model and make sure that you are able to recover the linear transformation.

1.4 Kernel based predictions

Similarly to the previous questions, we suppose that we are given n data points for $\mathbf{x}$ and $\mathbf{y}$, in the form of matrices $X \in \mathbb{R}^{d\times n}$ and $Y \in \mathbb{R}^{q\times n}$ respectively.

A kernel based predictor differs from a linear predictor in that it performs linear prediction on a transformed version of the input rather than of the input directly. The transformation is performed by a function (let's call it $\boldsymbol{\phi}$) mapping input vectors to vectors in a space of arbitrary dimension N, i.e. $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^N$. The prediction for a vector $\mathbf{x} \in \mathbb{R}^d$ is written as

$$\hat{\mathbf{y}} = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}), \tag{10}$$

where $\mathbf{w}$ are parameters of the predictor. In other words, kernel predictors are linear predictors in higher dimensional spaces.

Question 1. Introducing the design matrix.

Derive the MSE solution for $\mathbf{w}$, when a constant weighting factor $r_i > 0$ is introduced for each of our samples $\mathbf{x}_i$, i.e. find $\mathbf{w}^\star$ that minimizes:

$$E(\mathbf{w}) = \frac{1}{2n} \sum_{i=1}^{n} r_i \left(y_i - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i)\right)^2 \tag{11}$$

Question 1a. Write the derivative of E wrt $\mathbf{w}$

Question 1b. Find $\mathbf{w}^*$ that minimizes E as a function of

$$\boldsymbol{\Phi}' = \begin{bmatrix} \sqrt{r_1}\boldsymbol{\phi}(\mathbf{x}_1)^T \\ \vdots \\ \sqrt{r_n}\boldsymbol{\phi}(\mathbf{x}_n)^T \end{bmatrix} \text{ and } Y' = \begin{bmatrix} \sqrt{r_1}y_1 \\ \vdots \\ \sqrt{r_n}y_n \end{bmatrix} \tag{12}$$

Question 1c. How can you interpret the coefficients r_i ?

Question 2. Introducing the Gram matrix.

We now assume that $r_i = 1$ for $i = 1, \dots, n$. Also, we assume that we introduce a regularization parameter $\lambda > 0$ in our MSE.

Question 2a. Write the MSE (similar to equation 11) with a regularization term for $||\mathbf{w}||_2$ and without r_i.

The design matrix $\boldsymbol{\Phi}$ can be problematic to compute for some choices of $\boldsymbol{\phi}$. Instead let us introduce a way to perform predictions on a new point $\mathbf{x}$, without explicitely writting the design matrix. For this we need another parameter vector :

$$\mathbf{a} \in \mathbb{R}^n, \text{ where } a_i = -\frac{1}{\lambda}(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n) - y_n), \text{ for } i = 1, \dots, n \tag{13}$$

Using this in the expression of the gradient of equation (11), we find that this new parameter vector is related to $\mathbf{w}$ as follows: $\mathbf{w} = \boldsymbol{\Phi}^T \mathbf{a}$.

Question 2b. By introducing the Gram matrix $K = \boldsymbol{\Phi}\boldsymbol{\Phi}^T$, with elements $K_{i,j} = \boldsymbol{\phi}(\mathbf{x}_i)^T \boldsymbol{\phi}(\mathbf{x}_j)$ show that the prediction for a vector $\mathbf{x} \in \mathbb{R}^d$ can be obtained as (Eq (6.9) in Bishop):

$$\hat{\mathbf{y}} = Y\left(K + \lambda I_n\right)^{-1} \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}) \\ \vdots \\ k(\mathbf{x}_n, \mathbf{x}) \end{bmatrix} \tag{14}$$

Question 3. Implement a function in Python that calculates the Gram matrix associated with a linear kernel. Your function should take as argument two sets of vectors in $\mathbb{R}^d$ in the form of two matrices, e.g. $X_1 \in \mathbb{R}^{d\times n}$ and $X_2 \in \mathbb{R}^{d\times m}$, and return the Gram matrix $K \in \mathbb{R}^{n\times m}$.

Question 4. Similarly, implement a function in Python that calculates the Gram matrix associated with a RBF kernel.

Question 5. Implement (14). Your function should take a matrix with input points columnwise and return a matrix with the predicted vectors columnwise.

Question 6. Compare the performances of a kernel predictor with a linear kernel and with a RBF kernel. You can use a toy dataset for this, e.g.:

```
from sklearn.datasets import make_circles;
X,Y  = make_circles(n_samples=1_000, factor=0.3, noise=0.05, random_state=0);
```