Artificial Intelligence: HW 3

Jeong Min Lee

November 13, 2023

1 Linear Regression

1.a

Let f be the target function. I'll use the superscript with parenthesis to describe the n-th sample vector.

$$f(\boldsymbol{\omega}, \overline{\boldsymbol{x}^{(n)}}) = \sum_n \frac{1}{2}\left(t^{(n)} - \boldsymbol{\omega}^T \overline{\boldsymbol{x}^{(n)}}\right)^2 \tag{1}$$

To minimize f, differentiate it by $\boldsymbol{\omega}$ and find the $\boldsymbol{\omega}_0$ which makes the derivative zero. To make expression simple, I used the Einstein notation.

$$\begin{aligned}\frac{\partial f}{\partial \omega_j} &= -(t^{(n)} - \boldsymbol{\omega}^T \overline{\boldsymbol{x}^{(n)}}) \cdot \frac{\partial}{\partial \omega_j} \boldsymbol{\omega}^T \bar{\boldsymbol{x}}^{(n)} \\ &= -(t^{(n)} - \boldsymbol{\omega}^T \overline{\boldsymbol{x}^{(n)}}) x_j^{(n)} \\ &= 0\end{aligned}$$

By enumerating the $\frac{\partial f}{\partial \omega_j}$ horizontally, one can get $\frac{\partial f}{\partial \boldsymbol{\omega}}$.

$$\sum_n t^{(n)} \begin{pmatrix} x_1^{(n)} \\ \vdots \\ x_M^{(n)} \end{pmatrix}^T = \sum_n \begin{pmatrix} \boldsymbol{\omega}^T \overline{\boldsymbol{x}^{(n)}} x_1^{(n)} \\ \vdots \\ \boldsymbol{\omega}^T \overline{\boldsymbol{x}^{(n)}} x_M^{(n)} \end{pmatrix}^T \tag{2}$$

The left hand side is simply $\sum_n t^{(n)} \overline{\boldsymbol{x}^{(n)}}^T$. From the linearity of vector summation rule, the right hand side is $\left(\left(\sum_n \overline{\boldsymbol{x}^{(n)}} \cdot \overline{\boldsymbol{x}^{(n)}}^T\right) \boldsymbol{\omega}\right)^T$. By taking transpose to both sides, one can get the following equation.

$$\left[\sum_n \overline{\boldsymbol{x}^{(n)}} \cdot \overline{\boldsymbol{x}^{(n)}}^T\right] \boldsymbol{\omega} = \sum_n t^{(n)} \overline{\boldsymbol{x}^{(n)}} \tag{3}$$

Therefore, $\boldsymbol{A} = \sum_n \overline{\boldsymbol{x}^{(n)}} \cdot \overline{\boldsymbol{x}^{(n)}}^T$ and $\boldsymbol{b} = \sum_n t^{(n)} \overline{\boldsymbol{x}^{(n)}}$

1.b

$\overline{\boldsymbol{x}^{(1)}} = (1, 0)^T, t^{(1)} = 1$. $\overline{\boldsymbol{x}^{(2)}} = (1, \epsilon)^T, t^{(2)} = 1$. $\boldsymbol{A} = \overline{\boldsymbol{x}^{(1)}} \cdot \overline{\boldsymbol{x}^{(1)}}^T + \overline{\boldsymbol{x}^{(1)}} \cdot \overline{\boldsymbol{x}^{(2)}}^T = \begin{pmatrix} 2 & \epsilon \\ \epsilon & \epsilon^2 \end{pmatrix}$ $\boldsymbol{b} = \overline{\boldsymbol{x}^{(1)}} + \overline{\boldsymbol{x}^{(2)}} = (2, \epsilon)^T$. Since $\boldsymbol{A}$ is invertible(determinant is nonzero.),

$$\begin{aligned}\boldsymbol{\omega} &= \boldsymbol{A}^{-1}\boldsymbol{b} \qquad (4) \\ &= \frac{1}{\epsilon^2}\begin{pmatrix} \epsilon^2 & -\epsilon \\ -\epsilon & 2 \end{pmatrix}\begin{pmatrix} 2 \\ \epsilon \end{pmatrix} \qquad (5) \\ &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad (6)\end{aligned}$$

1.c

$\boldsymbol{A}$ is same to the above one. $\boldsymbol{b} = (1+\epsilon)\cdot\overline{\boldsymbol{x}^{(1)}} + \overline{\boldsymbol{x}^{(2)}} = (2+\epsilon, \epsilon)^T$

$$\begin{aligned}\boldsymbol{\omega} &= \boldsymbol{A}^{-1}\boldsymbol{b} &(7)\\ &= \frac{1}{\epsilon^2}\begin{pmatrix}\epsilon^2 & -\epsilon\\ -\epsilon & 2\end{pmatrix}\begin{pmatrix}2+\epsilon\\ \epsilon\end{pmatrix} &(8)\\ &= \begin{pmatrix}1+\epsilon\\ -1\end{pmatrix} &(9)\end{aligned}$$

1.d

$\boldsymbol{\omega}_b = (1,0)^T, \boldsymbol{\omega}_c = (1.1,-1)^T$. The difference of $\Delta\boldsymbol{\omega} = \boldsymbol{\omega}_c - \boldsymbol{\omega}_b = (\epsilon,-1)^T = (0.1,-1)^T$

2 Linear Regression with Regularization

2.a

Claim 1 : $\boldsymbol{A}$ is positive semi-definite.
proof

$\boldsymbol{A}$ is trivially symmetry matrix. $\forall \boldsymbol{v}\in\mathbb{R}^n, \boldsymbol{v}^T\boldsymbol{A}\boldsymbol{v} = \sum_n \boldsymbol{v}^T\overline{\boldsymbol{x}^{(n)}}\cdot\overline{\boldsymbol{x}^{(n)}}^T\boldsymbol{v} = \sum_n \|\boldsymbol{v}^T\overline{\boldsymbol{x}^{(n)}}\|^2 \geq 0.$■

Claim 2 : $\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{x} \iff \boldsymbol{A}^{-1}\boldsymbol{x} = \lambda^{-1}\boldsymbol{x}$ where $\lambda \neq 0$ and $\boldsymbol{A}$ is invertible.
proof

$\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{x} \iff \boldsymbol{A}^{-1}\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{A}^{-1}\boldsymbol{x} \iff \lambda^{-1}\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{x}.$■

Let S(A) be the set of all eigenvalues of A.

$$S(\boldsymbol{A}) \equiv \{\lambda_i | \text{for some } \boldsymbol{x}\in\mathbb{R},\ \boldsymbol{A}\boldsymbol{x} = \lambda_i\boldsymbol{x}\} \qquad (10)$$

For $\forall\tilde{\lambda}\in S(\boldsymbol{A}+\lambda\boldsymbol{I}), (\boldsymbol{A}+\lambda I)\boldsymbol{x} = \tilde{\lambda}\boldsymbol{x}$.
By multiplying $\boldsymbol{x}^T, \boldsymbol{x}^T(\boldsymbol{A}+\lambda\boldsymbol{I})\boldsymbol{x} = \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} + \lambda = \tilde{\lambda} \geq \lambda.(\because \boldsymbol{A}$ is positive semi-definite.)
This implies that $\min(S(\boldsymbol{A}+\lambda\boldsymbol{I})) \geq \lambda$. Equivalently,due to the **Claim 2**, this also means that $\max(S((\boldsymbol{A}+\lambda I)^{-1})) \leq \lambda^{-1}$. By noticing that $\max(S((\boldsymbol{A}+\lambda I)^{-1})) = \rho((\boldsymbol{A}+\lambda I)^{-1})$, the proof is done. Note that for the equality, $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$ must have nontrivial solution.

2.b

For both problems, the $\boldsymbol{A}+\lambda\boldsymbol{I}$ is following.

$$\boldsymbol{A}+\lambda\boldsymbol{I} = \begin{pmatrix}2+\lambda & \epsilon\\ \epsilon & \epsilon^2+\lambda\end{pmatrix} \qquad (11)$$

Since (11) is invertible, one can get $\boldsymbol{\omega_b}, \boldsymbol{\omega_c}$.

$$\begin{aligned}\boldsymbol{\omega_b} &= \frac{1}{(1+\lambda)\epsilon^2+\lambda(\lambda+2)}\cdot\begin{pmatrix}\epsilon^2+\lambda & -\epsilon\\ -\epsilon & 2+\lambda\end{pmatrix}\cdot\begin{pmatrix}2\\ \epsilon\end{pmatrix}\\ &= \frac{1}{(1+\lambda)\epsilon^2+\lambda(\lambda+2)}\cdot\begin{pmatrix}\epsilon^2+2\lambda\\ \epsilon\lambda\end{pmatrix}\\ &= \begin{pmatrix}0.973\\ 0.044\end{pmatrix}\end{aligned}$$

$$\begin{aligned}\boldsymbol{\omega_c} &= \frac{1}{(1+\lambda)\epsilon^2+\lambda(\lambda+2)}\cdot\begin{pmatrix}\epsilon^2+\lambda & -\epsilon\\ -\epsilon & 2+\lambda\end{pmatrix}\cdot\begin{pmatrix}2+\epsilon\\ \epsilon\end{pmatrix}\\ &= \frac{1}{(1+\lambda)\epsilon^2+\lambda(\lambda+2)}\cdot\begin{pmatrix}\epsilon^3+\epsilon^2+\lambda\epsilon+2\lambda\\ -\epsilon^2+\lambda\epsilon\end{pmatrix}\\ &= \begin{pmatrix}1.026\\ -0.044\end{pmatrix}\end{aligned}$$

Furthemore, $\Delta\boldsymbol{\omega} = \boldsymbol{\omega_c} - \boldsymbol{\omega_b}$ can be obtained.

$$\Delta\boldsymbol{\omega} = \frac{1}{(1+\lambda)\epsilon^2+\lambda(\lambda+2)}\begin{pmatrix}\epsilon^3+\lambda\epsilon\\ -\epsilon^2\end{pmatrix} = \begin{pmatrix}0.0531\\ -0.088\end{pmatrix} \tag{12}$$

2.c

One can notice that $\|\Delta\boldsymbol{\omega}\|$ with regularization is much smaller than $\|\Delta\boldsymbol{\omega}\|$ without regularization. This implies that regularization makes the parameters less variable with respect to small noise in input data. This can be verified by the following figure.

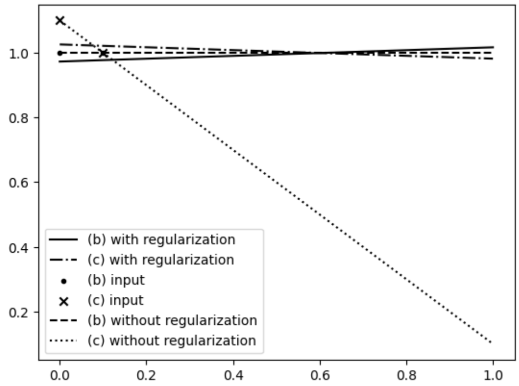

Figure 1: The linear regression result of (b) and (c)

According to the figure 1, the linear regression with regularization tends to be stable for the noised sample (c) and unnoised sample (b). However, the regression without the regularization has more fluctuation for noised sample.

3 LR with Regularization: A Probabilistic Perspective

The Gaussian prior implies the following.

$$\Pr(\boldsymbol{\omega}) = \mathcal{N}(\mathbf{0}, \frac{1}{\lambda}\boldsymbol{I}) = \frac{\lambda^{N/2}}{2\pi^{N/2}}\exp\left(-\frac{1}{2}\boldsymbol{\omega}^T\lambda\boldsymbol{\omega}\right)$$

In general, the $\epsilon^{(i)}$ also tends to have the Gaussian distribtuion. Therefore, by substituting $\epsilon^{(i)}$ to $t^{(i)} - \boldsymbol{\omega}^T\boldsymbol{x}^{(i)}$, $\Pr\left(t^{(i)} \mid \boldsymbol{\omega}, \boldsymbol{x}^{(i)}\right)$ is possible to be dervied as follow.

$$\begin{aligned} t^{(i)} = \boldsymbol{\omega}^T\boldsymbol{x}^{(i)} + \epsilon^{(i)} \;\;\&\;\; \Pr(\epsilon^{(i)}) &= \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-{\epsilon^{(i)}}^2/2\sigma^2\right)\\ \Rightarrow \Pr\left(t^{(i)} \mid \boldsymbol{\omega}, \boldsymbol{x}^{(i)}\right) &= \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-(t^{(i)} - \boldsymbol{\omega}^T\boldsymbol{x}^{(i)})^2/2\sigma^2\right)\end{aligned}$$

By producting Pr $\left(t^{(i)} \mid \boldsymbol{\omega}, \boldsymbol{x}^{(i)}\right)$ for all possible i, one can be able to get $\Pr(\boldsymbol{t} \mid \boldsymbol{\omega}^T \boldsymbol{x}^{(i)})$.

$$\Pr(\boldsymbol{t} \mid \boldsymbol{\omega}^T \boldsymbol{x}^{(i)}) = \prod_{i=1}^{N} \Pr(t^{(i)} \mid \boldsymbol{\omega}^T, \boldsymbol{x}) = \frac{1}{(2\pi)^{N/2}\sigma^N} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(t^{(i)} - \boldsymbol{\omega}^T \boldsymbol{x}^{(i)})^2\right)$$

From the discussion above and Bayes' rule, the posterior probability is following.

$$\Pr(\boldsymbol{\omega} \mid \boldsymbol{x}, \boldsymbol{t}) \propto \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{N}\left(t^{(i)} - \boldsymbol{\omega}^T \boldsymbol{x}^{(i)}\right)^2 - \frac{\lambda}{2}\boldsymbol{\omega}^T \boldsymbol{\omega}\right) \tag{13}$$

Minimizing (13) is equivalent to minimizing its logarithm.

$$\log\left(\Pr(\boldsymbol{\omega} \mid \boldsymbol{x}, \boldsymbol{t})\right) \propto -\frac{1}{2\sigma^2}\sum_{i=1}^{N}\left(t^{(i)} - \boldsymbol{\omega}^T \boldsymbol{x}^{(i)}\right)^2 - \frac{\lambda}{2}\boldsymbol{\omega}^T \boldsymbol{\omega} \tag{14}$$

One can see from (14) that maximizing logarithm of posterior is equivalent to minimizing the target function of linear regression with L2 regularization.

4 Logistic Regression

$$\log L(\boldsymbol{\omega}) = -\sum_{i}\left[t^{(i)} \log\left(\frac{1}{1+e^{-\boldsymbol{\omega}^T \boldsymbol{x}^{(i)}}}\right) + (1-t^{(i)})\log\left(\frac{e^{-\boldsymbol{\omega}^T \boldsymbol{x}^{(i)}}}{1+e^{\boldsymbol{\omega}^T \boldsymbol{x}^{(i)}}}\right)\right] \tag{15}$$

To minimize (15), one can differentiate it about $\boldsymbol{\omega}$ to find the $\boldsymbol{\omega_0}$ that makes the derivative zero. By denoting $P^{(n)} = (1 + e^{\boldsymbol{\omega}^T \boldsymbol{x}^{(n)}})^{-1} = \sigma(\boldsymbol{\omega}^T \boldsymbol{x}^{(n)})$

$$\begin{aligned}
\frac{\partial}{\partial \omega_i}\log L(\boldsymbol{\omega}) &= -\sum_{n}\frac{\partial}{\partial \omega_i}\left[t^{(n)}\log P^{(n)} + (1-t^{(n)})\log(1-P^{(n)})\right] \\
&= -\sum_{n}\left[\frac{t^{(n)}}{P^{(n)}} - \frac{1-t^{(n)}}{1-P^{(n)}}\right]\cdot\frac{\partial P^{(n)}}{\partial \omega_i} \\
&= -\sum_{n}\left[t^{(n)}/P^{(n)} - (1-t^{(n)})/P^{(n)}\right]\cdot P^{(n)}(1-P^{(n)})\cdot\frac{\partial}{\partial \omega_i}\boldsymbol{\omega}^T \boldsymbol{x}^{(n)} \\
&= -\sum_{n}\left[t^{(n)}(1-P^{(n)}) - (1-t^{(n)})P^{(n)}\right]x_i^{(n)} \\
&= -\sum_{n}(t^{(n)} - P^{(n)})x_i^{(n)} = 0
\end{aligned}$$

Note that $d\sigma(x)/dx = \sigma(x)(1-\sigma(x))$. The result of the last equation above can be written in vector form.

$$\sum_{n}\left(t^{(n)} - \sigma(\boldsymbol{\omega}^T \boldsymbol{x})\right)\boldsymbol{x}^{(n)} = 0 \tag{16}$$

The equation (16) can be written as following equation.

$$\sum_{n} t^{(n)}\boldsymbol{x}^{(n)} = \sum_{n}\sigma(\boldsymbol{\omega}^T \boldsymbol{x})\boldsymbol{x}^{(n)} \tag{17}$$

Unfortunately, since sigmoid function σ is non-linear function, the equation (17) is non-linear equation that doesn't have the closed form of solution in general.

5 KNN

In this section, I will briefly introduce my KNN algorithm. I have implemented the following functions: **predict_knn()**, **eval_knn()**, and **cross_validation_knn()**.

5.a predict_knn

To handle potential errors in **load_knn_data()**, I consistently used a **try-except** clause for each implemented function. It's important to note that the usage of these functions may not be consistent, as **predict_knn()** is called not only in **eval_knn()** but also in **main()**. In **main()**, the input parameters are in the form of tuples, while in **eval_knn()**, they are in the form of 2-dimensional numpy arrays.

Now, let's discuss **predict_knn()**. After the **try-except** clause, I defined **dist_arr**, which contains the square of the Euclidean norm from x to every sample in $inputs$. Subsequently, I sorted the index of **dist_arr** with respect to **dist_arr** and obtained the index of first $k\_neighbors$ features that are the closest $k\_neighbors$ samples to x. Next, from that index, I retrieved the labels of x's neighbors. Finally, **np.unique()** enabled me to find the labels of x's neighbors and the count of each label. I used the **argmax()** function to get the label that appeared most frequently among the $k\_neighbors$ samples. Due to the characteristics of **argmax()** function, the formost label is returned for the tied label.

5.b eval_knn

This function simply checks whether the given input data points are classified correctly. For each iteration and every row of $inputs$, I counted the cases where the given $labels$ and the result of **predict_knn()** matched.

5.c cross_validation_knn

I treated a portion of the training dataset as a validation set, with the portion determined by k_folds. By factorizing the training dataset into k_folds sets, I obtained the average performance (or accuracy) of the KNN results for each hyperparameter. This process was repeated until every given hyperparameter was evaluated.

5.d Results

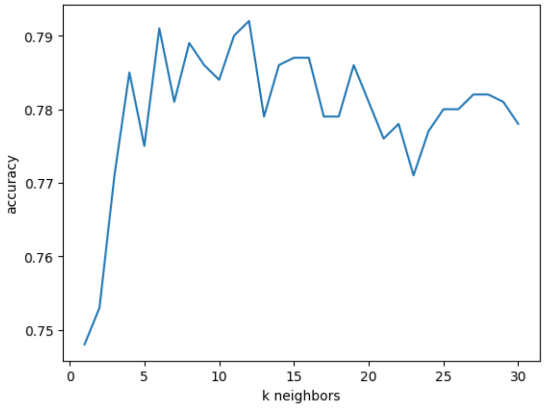

Figure 2: The final result of **main()**

In my algorithm, the optimal number of neighbors is 12, with a cross-validation accuracy of 0.792. For the test set, it resulted in an accuracy of 0.7545.