Logistic Regression with Stochastic Gradient Descent - Report

1. To solve dual SVM problem, we have data matrix and kernel function. Therefore, we can use

$$max_{\alpha \in \mathbb{R}^N} \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m K(x_n, x_m)$$

$$subject\ to\ \sum_{n=1}^{N} y_n \alpha_n = 0\ and\ \alpha_n \geq 0\ for\ all\ n.$$

a. We have 4 data points and their labels. Our kernel function is $K(x_1, x_2) = (x_1^T x_2 + 1)^2$. Therefore, our problem becomes

$$max_{\alpha \in \mathbb{R}^4} \sum_{n=1}^{4} \alpha_n - \frac{1}{2} \sum_{n=1}^{4} \sum_{m=1}^{4} \alpha_n \alpha_m y_n y_m (x_n^T x_m + 1)^2$$

$$subject\ to\ \sum_{n=1}^{4} y_n \alpha_n = 0\ and\ \alpha_n \geq 0\ for\ all\ n.$$

If we write the our constraints by using the data points and labels, then we will get

$$\begin{aligned} max_{\alpha \in \mathbb{R}^4} \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2} \Big[& \alpha_1\alpha_1 y_1 y_1 (x_1^T x_1 + 1)^2 + \alpha_2\alpha_1 y_2 y_1 (x_2^T x_1 + 1)^2 \\ & + \alpha_3\alpha_1 y_3 y_1 (x_3^T x_1 + 1)^2 + \alpha_4\alpha_1 y_4 y_1 (x_4^T x_1 + 1)^2 \\ & + \alpha_1\alpha_2 y_1 y_2 (x_1^T x_2 + 1)^2 + \alpha_2\alpha_2 y_2 y_2 (x_2^T x_2 + 1)^2 \\ & + \alpha_3\alpha_2 y_3 y_2 (x_3^T x_2 + 1)^2 + \alpha_4\alpha_2 y_4 y_2 (x_4^T x_2 + 1)^2 \\ & + \alpha_1\alpha_3 y_1 y_3 (x_1^T x_3 + 1)^2 + \alpha_2\alpha_3 y_2 y_3 (x_2^T x_3 + 1)^2 \\ & + \alpha_3\alpha_3 y_3 y_3 (x_3^T x_3 + 1)^2 + \alpha_4\alpha_3 y_4 y_3 (x_4^T x_3 + 1)^2 \\ & + \alpha_1\alpha_4 y_1 y_4 (x_1^T x_4 + 1)^2 + \alpha_2\alpha_4 y_2 y_4 (x_2^T x_4 + 1)^2 \\ & + \alpha_3\alpha_4 y_3 y_4 (x_3^T x_4 + 1)^2 + \alpha_4\alpha_4 y_4 y_4 (x_4^T x_4 + 1)^2 \Big] \end{aligned}$$

$$subject\ to\ y_1\alpha_1 + y_2\alpha_2 + y_3\alpha_3 + y_4\alpha_4 = 0\ and\ \alpha_n \geq 0\ for\ all\ n.$$

b. Let our kernel matrix be **K**. Then, we can write it as

$$\mathbf{K} = \begin{array}{c} \\ \\ \end{array} \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \end{array}$$

$$\mathbf{K} = \left[\begin{array}{cccc} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{array}\right] \begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \end{array}$$

In Jordan et al. (2004) lecture notes, Mercer's condition says that a symmetric function $K(x, y)$ can be expressed as inner product

$$K(x, y) = \phi(x)^T \phi(y)$$

for some ϕ if and only if K(x,y) is positive semi-definite. To show that the matrix $\mathbf{K}$ satisfies the Mercer's condition, we can show that all its eigenvalues are positive; that is, it is a positive semi-definite matrix (Mittal Lecture 11 notes). In python, we can use np.linalg.eig(); thus, we can find its eigenvalues and eigenvectors. According to this, we get

$$eigenvalues\ of\ \mathbf{K} = [8.12.8.8.].$$

We see that all eigenvalues are positive; therefore, $\mathbf{K}$ satisfies Mercer's consition.

c. If we put the data points and labels in our constraints from part a, then we have

$$\begin{aligned} max_{\alpha\in\mathbb{R}^4}\alpha_1+\alpha_2+\alpha_3+\alpha_4-\frac{1}{2}\Big[&9\alpha_1\alpha_1-\alpha_2\alpha_1\\ &+\alpha_3\alpha_1-\alpha_4\alpha_1\\ &-\alpha_1\alpha_2+9\alpha_2\alpha_2\\ &-\alpha_3\alpha_2+\alpha_4\alpha_2\\ &+\alpha_1\alpha_3-\alpha_2\alpha_3\\ &+9\alpha_3\alpha_3-\alpha_4\alpha_3\\ &-\alpha_1\alpha_4+\alpha_2\alpha_4\\ &-\alpha_3\alpha_4+9\alpha_4\alpha_4\Big]\end{aligned}$$

$$subject\ to\ \alpha_1-\alpha_2+\alpha_3-\alpha_4=0\ and\ \alpha_n\geq 0\ for\ all\ n.$$

where

$$\mathbf{X}=\begin{bmatrix}x_1\\x_2\\x_3\\x_4\end{bmatrix}=\begin{bmatrix}-1&-1\\-1&1\\1&1\\1&-1\end{bmatrix}\ and\ y=\begin{bmatrix}y_1\\y_2\\y_3\\y_4\end{bmatrix}=\begin{bmatrix}1\\-1\\1\\-1\end{bmatrix}$$

In short,

$$max_{\alpha\in\mathbb{R}^4}\Big[\alpha_1+\alpha_2+\alpha_3+\alpha_4-\frac{1}{2}\Big(9\alpha_1^2+9\alpha_2^2+9\alpha_3^2+9\alpha_4^2-2\alpha_1\alpha_2+2\alpha_1\alpha_3-2\alpha_1\alpha_4-2\alpha_2\alpha_3+2\alpha_2\alpha_4-2\alpha_3\alpha_4\Big)\Big]$$

$$subject\ to\ \alpha_1+\alpha_3=\alpha_2+\alpha_4\ and\ \alpha_n\geq 0\ for\ all\ n.$$

Let say

$$\mathcal{L}(\alpha)=\alpha_1+\alpha_2+\alpha_3+\alpha_4-\frac{1}{2}\Big(9\alpha_1^2+9\alpha_2^2+9\alpha_3^2+9\alpha_4^2-2\alpha_1\alpha_2+2\alpha_1\alpha_3-2\alpha_1\alpha_4-2\alpha_2\alpha_3+2\alpha_2\alpha_4-2\alpha_3\alpha_4\Big)$$

Therefore, we can find α_i's by taking gradient of $\mathcal{L}(\alpha)$ with respect to each α_i, $i=1,2,3,4$.Then, these gradients will set to zero.

$$-\frac{1}{2}\Big(18\alpha_1-2\alpha_2+2\alpha_3-2\alpha_4\Big)+1\Rightarrow 9\alpha_1-\alpha_2+\alpha_3-\alpha_4-1=0$$

$$-\frac{1}{2}\Big(18\alpha_2-2\alpha_1-2\alpha_3+2\alpha_4\Big)+1\Rightarrow 9\alpha_2-\alpha_1-\alpha_3+\alpha_4-1=0$$

$$-\frac{1}{2}\Big(18\alpha_3+2\alpha_1-2\alpha_3-2\alpha_4\Big)+1\Rightarrow 9\alpha_3+\alpha_1-\alpha_2-\alpha_4-1=0$$

$$-\frac{1}{2}\Big(18\alpha_4-2\alpha_1+2\alpha_2-2\alpha_3\Big)+1\Rightarrow 9\alpha_4-\alpha_1+\alpha_2-\alpha_3-1=0$$

While solving this linear equations system, we can use this way:

$$Ax=b\Rightarrow x=A^{-1}b$$

Here, A should be invertible. Smith's Math217 lecture notes says "A square matrix is invertible if and only if its determinant is non-zero." In python, we can use np.linalg.det() to calculate determinant.

$$Q_{nm} = y_n y_m \Phi(x_n)^T \Phi(x_m)$$

Therefore,

$$\mathbf{Q} = \begin{bmatrix} 9 & -1 & 1 & -1 \\ -1 & 9 & -1 & 1 \\ 1 & -1 & 9 & -1 \\ -1 & 1 & -1 & 9 \end{bmatrix}$$

$$det\mathbf{Q} = 6144.000000000003$$

Therefore, it is invertible and we can use this way to reach solution.

$$b = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Therefore, we get α by using $\alpha = \mathbf{Q}^{-1}b$. By solving this linear system of equations, we get

$$\alpha_1 = 0.125, \alpha_2 = 0.125, \alpha_3 = 0.125 \ and \ \alpha_4 = 0.125$$

According to our solution, we should control them subject to constraints. Each $\alpha_i, i = 1, 2, 3, 4;$ is greater than 0. It is okay. Also, the equation below should be satisfied

$$\alpha_1 + \alpha_3 = \alpha_2 + \alpha_4$$

$$0.125 + 0.125 = 0.125 + 0.125$$

It is also okay. Therefore, we can say

$$\alpha = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} = \begin{bmatrix} 0.125 \\ 0.125 \\ 0.125 \\ 0.125 \end{bmatrix}$$

d. We know that kernel function $K(x_n, x_m)$ is equal to $\Phi(x_n)^T \Phi(x_m)$. We have $K(x_n, x_m) = (x_n^T x_m + 1)^2$ already. If we write it as

$$\begin{aligned} K(x_n, x_m) &= (x_n^T x_m + 1)(x_n^T x_m + 1) \\ &= (x_n^T x_m)^2 + 2x_n^T x_m + 1 \end{aligned}$$

Therefore, we say

$$\Phi(x_n) = \begin{bmatrix} 1 \\ \sqrt{2}x_{n1} \\ \sqrt{2}x_{n2} \\ x_{n1}^2 \\ x_{n2}^2 \\ \sqrt{2}x_{n1}x_{n2} \end{bmatrix}$$

by using lecture notes page 108. Then, we can write all $\Phi(\mathbf{x}_i)$'s.

$$\Phi(\mathbf{x}_1) = \begin{bmatrix} 1 \\ -\sqrt{2} \\ -\sqrt{2} \\ 1 \\ 1 \\ \sqrt{2} \end{bmatrix}, \Phi(\mathbf{x}_2) = \begin{bmatrix} 1 \\ -\sqrt{2} \\ \sqrt{2} \\ 1 \\ 1 \\ -\sqrt{2} \end{bmatrix}, \Phi(\mathbf{x}_3) = \begin{bmatrix} 1 \\ \sqrt{2} \\ \sqrt{2} \\ 1 \\ 1 \\ \sqrt{2} \end{bmatrix} \ and \ \Phi(\mathbf{x}_4) = \begin{bmatrix} 1 \\ \sqrt{2} \\ -\sqrt{2} \\ 1 \\ 1 \\ -\sqrt{2} \end{bmatrix}$$

e. We have all necessary information for the solution. Therefore, we can use below equation for the primal solution.

$$\mathbf{w}^* = \sum_{n_1}^{4} y_n \alpha_n \Phi(x_n)$$

In python, we can do it easily and we get

$$\mathbf{w}^* = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.70710678 \end{bmatrix}$$

2. Now, we will perform logistic regression with stochastic gradient descent. Here, we can follow Abu-Mostafa et al. (2012) Learning From Data book. At first, let's talk about logistic regression with stochastic gradient descent algorithm and then we will talk about our result for the given data.
Logistic regression helps to classify data points. We can implement it with gradient descent(GD) or stochastic gradient descent(SGD). The thing that SGD is different from GD is to take gradient by using random data points and update gradient for each epoch. Here, epoch helps to use each data point.

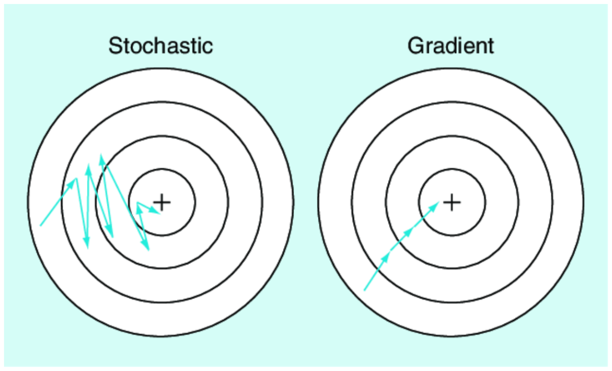

Figure 1: Stochastic Gradient Descent vs Gradient Descent

In Figure 1, we can see the difference between stochastic gradient descent and gradient descent clearly.
Logistic Regression with SGD:

- At first, we should initialize a weight vector including zeros.
- Then, we can compute stochastic gradient descent. For that part, we choose random indexes along length of data, we calculate gradients for this selected data points and then, we repeat this for each epoch. We calculate gradients by using

$$\begin{aligned} \nabla E_{in}(w) = \frac{\partial E(w)}{\partial w} &= \frac{1}{N}\sum_{n_1}^{N}(-y_n x_n exp(-y_n w^T x_n)/1 + exp(-y_n w^T x_n)) \\ &= \frac{1}{N}\sum_{n_1}^{N}(-y_n x_n/1 + exp(y_n w^T x_n)) \\ &= -\frac{1}{N}\sum_{n_1}^{N}(y_n x_n/1 + exp(y_n w^T x_n)) \end{aligned}$$

where E(w) is logistic loss function:

$$E(w) = \frac{1}{N}\sum_{n_1}^{N} ln(1 + exp(-y_n w^T x_n))$$

- Therefore, we can update our weights and we can use them to predictions.

Now, we can discuss our results. We add a column with ones of test and train data for bias because our weights matrix includes bias by first value. We can choose random values for leaning rate and number of epochs. Here, we selected them as 0.15 and 2000 respectively. Therefore, we calculate accuracy for both train and test data. Our results in Table.

Train accuracy	0.9769378603459321
Test accuracy	0.9528301886792453

We can say that our results is good and our model for logistic regression with stochastic gradient descent worked well.

REFERENCE

- Abu-Mostafa, Y. S., Magdon-Ismail, M., & Lin, H. T. (2012). *Learning from data* (Vol. 4). New York, NY, USA:: AMLBook.
- Bishop, C. M. (2006). *Pattern recognition and machine learning.* springer.
- Crowley, J. L. (2016). Kernel functions and support vector machines. *Intelligent Systems: Reasoning and Recognition.*
- Duda, R. O., & Hart, P. E. (2006). *Pattern classification.* John Wiley & Sons.
- Jordan, M. I., & Thibaux, R. (2004). The kernel trick. Lecture Notes.
- Mittal, R. Lecture 11: Positive semidefinite matrix.
- Smith, K. E. Math 217: Proof of Multiplicative Property of Determinant
- Suykens, J. A. (2001). Support vector machines: a nonlinear modelling and control perspective. *European Journal of Control*, 7(2-3), 311-327.
- https://towardsdatascience.com/gradient-descent-in-python-a0d07285742f
- http://binaryplanet.org/2020/04/scratch-implementation-of-stochastic-gradient-descent-using-python/