COMP4434 Big Data Analytics

Assignment 2

PolyU, Hong Kong

Problem 1 (2 points)
Please answer the following questions briefly.

(a) In the classification task, why we do prefer logistic loss over mean-squared error?

(b) How does a standard Support Vector Machine (SVM) differ from an SVM with a soft margin, and in what way do slack variables (ξ_i) facilitate the handling of non-linearly separable data?

(c) If your multi-layer perceptron model has an overfitting issue, what are the strategies that you could use to handle the issue?

(d) How does backpropagation use the gradient descent algorithm to update the weights in a neural network?

Answer:

(a) It is because the logistic loss punishes the mismatch cases heavily compared with MSE. Also, when we apply MSE to logistic regression, the cost function will become non-convex.

Optional example:

For binary classification, the logistic loss is defined as follows,

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right), \tag{1}$$

$$\begin{aligned}\text{Cost}\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right) &= \begin{cases} -\log\left(h_\theta\left(x^{(i)}\right)\right), & y^{(i)} = 1 \\ -\log\left(1 - h_\theta\left(x^{(i)}\right)\right), & y^{(i)} = 0 \end{cases} \\ &= -y^{(i)} \cdot \log\left(h_\theta\left(x^{(i)}\right)\right) - \left(1 - y^{(i)}\right) \cdot \log\left(1 - h_\theta\left(x^{(i)}\right)\right).\end{aligned} \tag{2}$$

$h_\theta\left(x^{(i)}\right)$ denotes predict value of sample i. And $y^{(i)}$ denotes actual value of sample i. And m denotes the total number of samples. MSE is defined as follows,

$$MSE = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2. \tag{3}$$

Assume $m = 1$, $h_\theta\left(x^{(i)}\right) = 0.01$, $y^{(i)} = 1$. Then, the logistic loss is $J(\theta) = -\log(0.01) \approx 4.6$, while $\text{MSE} = (0.01 - 1)^2 = 0.9801$.

(b) A soft margin SVM introduces slack variables ξ_i, which allow for some misclassification, achieving a better generalization on unseen data.

$$\frac{1}{2}||\mathbf{w}||^2 + C\left(\sum_{i=1}^{m}\xi_i^2\right), \text{ s.t. } y_i(\mathbf{w}^\top\mathbf{x}^{(i)} + b) \geq 1 - \xi_i, \xi_i \geq 0. \tag{4}$$

The constraint changes from $y_i(\mathbf{w}^\top\mathbf{x}^{(i)} + b) \geq 1$ to $y_i(\mathbf{w}^\top\mathbf{x}^{(i)} + b) \geq 1 - \xi_i$. Slack variables ξ_i facilitate the handling of non-linearly separable data by permitting some points to be within the margin or even on the wrong side of the hyperplane.

(c) There are many strategies, such as adding regularization terms, using more training data, reducing the total number of neural network layers, using k-fold cross-validation to split the data, and adding random dropouts.

(d) There are three major steps in one epoch:

1) Forward propagation: Compute the network's activation values and the loss using the current weights.

2) Backward propagation: Calculate the gradient of the loss with respect to each weight using the chain rule, propagating errors backward through the network.

3) Compute the gradients based on the errors and activation values. Adjust each weight using the gradient descent algorithm.

Problem 2 (5 points)
Load dataset in *problem2data.txt* by using "numpy.loadtxt()" in Python, which contains two synthetic classes separated by a non-linear function, with additive noise. The last column contains "0" or "1", indicating the corresponding classes. You are asked to investigate the extent to which polynomial functions can be used to build a logistic regression classifier.

(a) Build a logistic regression classifier by using a polynomial function of order 1 (e.g., $\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$). Apply 5-fold-cross-validation to generate training and test sets. Use the training set to train the model. Compute its five F1 scores on test set (one F1 score for each fold). Compute its five F1 scores on training set. You will need "f1_score" from "sklearn.metrics".

(b) Repeat part (a) for polynomials of order 2 to 10. You will need "PolynomialFeatures" from "sklearn.preprocessing".

(c) Repeat parts (a-b) 20 times, and estimate the average F1 score for each polynomial order (average across its 5×20 runs, both for training and test sets). Generate a plot that shows the F1 scores versus the polynomial order.

(d) Discuss how the F1 scores of the model changes as a function of the polynomial order.

(e) Use Jupyter Notebook to perform implementation. You are required to submit your ".ipynb" file. You could use "LogisticRegression" from "sklearn.linear_model". Do **not** add any regularizations.

Answer: The code and plots are included in "A2Problem2.ipynb". Based on the results, we have two major observations as follows.

1) The average F1 score on training set would keep increasing as the polynomial order increases from 1 to 10. It is because the model has more degrees of flexibility as the polynomial order increases.

2) The average F1 score on test set increases to 0.915 when the polynomial order is 5, and keeps decreasing as the polynomial order increases from 6 to 10. It is because the model to too complicated compared to the data pattern. An overfitting issue occurs.

Answer:

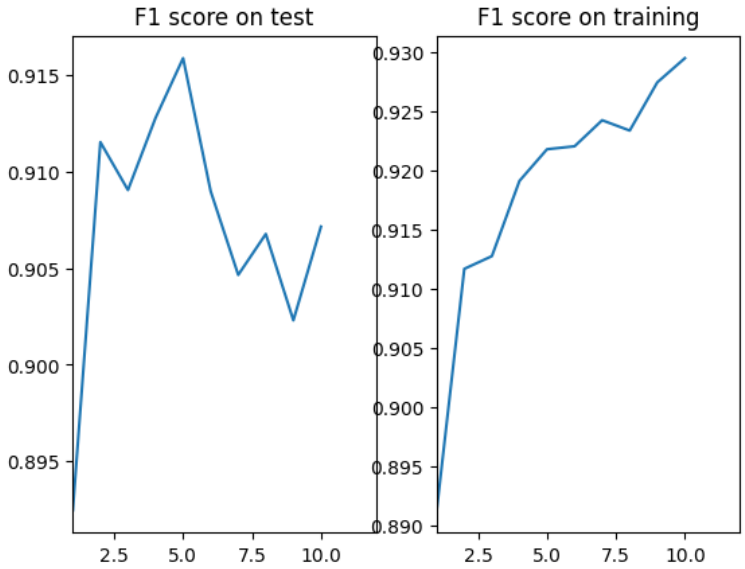

Figure 1: The F1 scores versus the polynomial order.

Problem 3 (2 points)

Consider a set of data points represented by the pairs (x, y). The objective is to apply the k-means algorithm to identify two distinct clusters within the data. Utilizing the 1st data point as the initial center for the first cluster and the 3rd data point as the center for the second cluster, simulate a single iteration of the k-means algorithm (k=2). We employ Euclidean distance as the distance metric.

(1) What are the resultant cluster assignments after completing one iteration?

(2) What are the new centroids after completing one iteration?

Data#	x	y
1	1.2	0.5
2	0.5	2.4
3	3	1.5
4	2	3.1
5	0.5	3.2
6	1.6	4.5

Answer:

Distance between 2 and C1: $\sqrt{((0.5-1.2)^2+(2.4-0.5)^2)} = 2.025$.

Distance between 2 and C2: $\sqrt{((0.5-3.0)^2+(2.4-1.5)^2)} = 2.657$.

Distance between 4 and C1: $\sqrt{((2-1.2)^2+(3.1-0.5)^2)} = 2.720$.

Distance between 4 and C2: $\sqrt{((2-3.0)^2+(3.1-1.5)^2)} = 1.887$.

Distance between 5 and C1: $\sqrt{((0.5-1.2)^2+(3.2-0.5)^2)} = 2.789$.

Distance between 5 and C2: $\sqrt{((0.5-3.0)^2+(3.2-1.5)^2)} = 3.023$.

Distance between 6 and C1: $\sqrt{((1.6-1.2)^2+(4.5-0.5)^2)} = 4.020$.

Distance between 6 and C2: $\sqrt{((1.6-3.0)^2+(4.5-1.5)^2)} = 3.311$.

Cluster 1: Data points 1, 2, and 5. That is, $(1.2, 0.5), (0.5, 2.4), (0.5, 3.2)$.
New centroid C1: $((1.2 + 0.5 + 0.5)/3, (0.5 + 2.4 + 3.2)/3) = (0.733, 2.033)$.
Cluster 2: Data points 3, 4, and 6. That is, $(3, 1.5), (2, 3.1), (1.6, 4.5)$.
New centroid C2: $((3.0 + 2 + 1.6)/3, (1.5 + 3.1 + 4.5)/3) = (2.2, 3.0333)$.