CMPE 482 - Assignment 5

Spring 2021

Due 23.06.2021, 20:00

In this assignment, we will be solving a *classification problem*.* After conducting a *dimensionality reduction,* we will train a *logistic regression model* and *multilayer perceptron model* using the *gradient descent algorithm*. Then, we will discuss how to compare the two resulting models.

1. **(2 points)** Download the Iris Dataset from the UCI Machine Learning Repository**. We will be using only the last 100 lines of this data set (only 2 classes). The first 4 columns of this data set are the features and the last column is labels. Convert the label 'Iris-versicolor' to 0 and 'Iris-virginica' to 1. Call the 100×4 data matrix A and the 100×1 label vector*** y.
   a. Conduct an SVD of the data matrix $A = U\Sigma V^T$ using `numpy.linalg.svd` function. Compute $C = U\Sigma$. Take the first two columns of C. We will call this 100×2 matrix X. The rows of this matrix will represent the coordinates of our observations in this new space. Produce a scatter plot of X, and color each point according to its label (Use first column of X for the horizontal dimension).
   b. When we use only the two dimensions mentioned above, which subspace of the original 4-dimensional space are we restricting/projecting our data to? You can respond to this question by giving basis vectors for this subspace.

2. **(4 points)** Let us now train a *supervised model* using our processed dataset X. We will use a *logistic regression model* for this task. For labels $y \in \{0, 1\}$ and features $x \in \mathcal{R}^d$, logistic regression can be defined as follows:

$$y \approx \hat{y} = \sigma(\beta + \theta^T x).$$

Here $\beta \in \mathcal{R}$ and $\theta \in \mathcal{R}^d$ are model parameters and σ is the sigmoid function, $\sigma(a) = 1/(1 + e^{-a})$. In our application, we will ignore β (usually called the *intercept parameter*) for simplicity, but note that this is not a common practice.

Previously, we used *error metrics* like mean square error (MSE) to evaluate 'how wrong' a model is on a data set given its parameters. The error metrics used for training a model is called the *loss function* in machine learning. MSE is not a very good loss function in this case as we now try to predict *binary labels*. Therefore, we define a new loss function called the *logistic loss (a.k.a. log loss)*:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} -y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i)$$

.

Unfortunately, logistic regression with log loss does not have an analytical solution, thus we will use the *gradient descent algorithm* to find a solution. Gradient descent is an iterative algorithm and in its description, we will use $\theta^{(k)}$ to denote the parameters in the k'th iteration of the algorithm. In gradient descent, for a randomly initialized $\theta^{(0)}$,

$$\theta^{(k+1)} = \theta^{(k)} - \eta \sum_{i=1}^{N} \nabla_{\theta^{(k)}} L(\theta^{(k)}),$$

where $\eta \in \mathcal{R} > 0$ is called the learning rate or step size. When run for a large enough number of iterations (call K), the model is expected to converge to a *local optimum* (given certain conditions beyond the scope of this course).

- **a.** Derive $\nabla_{\theta_i^{(k)}} L(\theta^{(k)})$ for our model. Note that here we are deriving $\nabla_{\theta_i^{(k)}}$ for a single parameter, the whole gradient would include the results of the operation conducted for all parameters.
- **b.** Initialize $\theta^{(0)}$ using Code Snippet 1. Train your algorithm for $K = 1000$ and $\eta = .001$ using `JAX`. In our example each row of X corresponds to an $x_i, i \in \{1, \dots 100\}$. Print $\theta^{(1000)}$ and $L(\theta^{(1000)})$. **Hint**: In the question above we asked you to provide the derivation for a single parameter for convenience. When training your algorithm do not separately compute the gradient for each parameter. The point of using `JAX` is to use vectorized operations to easily and efficiently computing gradients.
- **c.** Estimate the labels for each observation using $\hat{y}$'s produced by your algorithm: Use 0.5 as our *decision threshold*, that is, predicting y to be 0 if $\hat{y} < .5$, and 1 otherwise. Using your estimated labels, compute and print an accuracy score of your algorithm on the training data. **Hint**: Accuracy score = # of correctly classified observations / # of total observations.
- **d.** Now we will visualize the *decision boundary* between two classes given our trained algorithm. Note that our trained algorithm produces a $\hat{y} \in [0, 1]$ for each input provided to it. Since we now have a trained algorithm and a decision threshold of 0.5, we can look at $\hat{y}$ everywhere in the input space and visualize where it becomes 0.5, effectively demonstrating where our algorithm 'draws the line' between the two classes. An example code for drawing such a graph is supplied in Code Snippet 2, you can use it to demonstrate this decision boundary. **Hint**: For this plot to work correctly, your values for first column of X must lie between -11.5 and -5, and second column must lie between -1.5 and 1.5. If this is not the case, check out and correct your answer to the first question.

3. **(4 points)** We will now use a different algorithm to solve this problem, a *multilayer perceptron (MLP, a.k.a. artificial neural network)*.This time the algorithm is defined as

$$y \approx \hat{y} = \sigma(w_2^T \sigma(W_1 x)),$$

where $x \in \mathcal{R}^d, W_1 \in \mathcal{R}^{p \times d}, w_2 \in \mathcal{R}^p$, and σ is applied elementwise when applied to a vector. Expressed as such, our MLP has one 'hidden layer' and p 'hidden units'. We use gradient descent with log loss to train the algorithm; think $\theta = (W_1, w_2)$ to see the correspondence with above.

- **a.** Derive the gradient for a single element of W_1 and w_2, which we can generically call $W_{1,i,j}$ and $w_{2,i}$. In other words, derive $\nabla_{W_{1,i,j}^{(k)}}$ and $\nabla_{w_{2,i}^{(k)}}$.

b. Let $p = 5$. Initialize all parameters, i.e. $W_1^{(0)}$ and $w_2^{(0)}$ using Code Snippet 3. Train your algorithm for $K = 1000$ and $\eta = .001$ using `JAX`. Print $W_1^{(1000)}$, $w_2^{(1000)}$, and $L(W_1^{(1000)}, w_2^{(1000)})$. The hint in Question **2b** applies here as well.

c. Estimate the labels for each observation similar to Question **2c**. Using your estimated labels, compute and print an accuracy score of your algorithm on the training data.

d. Plot the decision boundary for the MLP solution, again you can make use of Code Snippet 2. Compare the result to the one you obtained with logistic regression. How/why are they similar/different? The hint in Question **2d** applies here as well.

4. (2 points) Discuss some aspects of the training process and decision making that follows:

a. If we did not provide you with K and η values, what would be a reasonable way of selecting them?

b. If we wanted to do model selection between logistic regression and multilayer perceptron, what would be a correct way of doing this, in terms of generalization?

c. What would be a correct way of selecting p for the MLP, in terms of generalization?

Code Snippet 1

```
import numpy as np
numpy.random.seed(0)
theta_init = np.random.randn(2)/10
```

Code Snippet 2

```
import numpy as np
import matplotlib.pyplot as plt
no_pts = 100
x_1 = np.linspace(-11.5, -5, no_pts)
x_2 = np.linspace(-1.5, 1.5, no_pts)
x_1_grid, x_2_grid = np.meshgrid(x_1, x_2)
p_grid = np.zeros((no_pts, no_pts))
for i in range(no_pts):
      for j in range(no_pts):
      x_1 = x_1_grid[i, j]
      x_2 = x_2_grid[i, j]
      p_grid[i, j] = # Here you produce y^hat using your trained algorithm
given x_1 and x_2 as input
fig, ax = plt.subplots()
# Here you produce an appropriately colored scatter plot of X using the
axis object ax. You can reuse your previous code from the assignment.
ax.contour(x_1_grid, x_2_grid, p_grid, levels=[.5]);
```

Code Snippet 3

```
import numpy as np
numpy.random.seed(0)
W_1_init = np.random.randn(5, 2)/10
w_2_init = np.random.randn(5)/10
```

Notes

* Check out the book Pattern Recognition and Machine Learning for a detailed investigation of machine learning concepts mentioned in this assignment.
** https://archive.ics.uci.edu/ml/index.php
*** In an abuse of notation, we sometimes use y to denote our 100-dimensional label vector and sometimes as a generic label $\in \{0, 1\}$ for convenience. The intended meaning should be clear from the context.