Homework Assignment 4

COGS 118A: Supervised Machine Learning Algorithms

Due: Nov 13th, 2020, 11:59pm (Pacific Time).

Instructions: Combine your answer to the questions below with your notebook to create a PDF file; submit your file via Gradescope.

1 (8 points) Conceptual Questions

1. Choose **all** the valid answers to the description about **linear regression** and **logistic regression** from the options below:

   A. Linear regression is an unsupervised learning problem; logistic regression is a supervised learning problem.

   B. Linear regression deals with the prediction of continuous values; logistic regression deals with the prediction of class labels.

   C. We cannot use gradient descent to solve linear regression. Instead, we can only use the closed-form solution to tackle the linear regression problem.

   D. Linear regression is a convex optimization problem whereas logistic regression is not.

2. Choose **all** the valid answers to the description about **gradient descent** from the options below:

   A. The global minimum is guaranteed to be reached by using gradient descent.

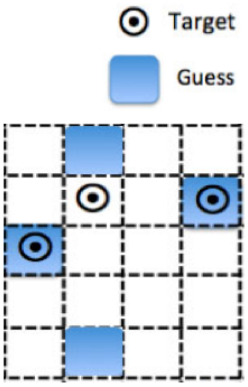
   B. Every gradient descent iteration can always decrease the value of loss function even when the gradient of the loss function is zero.

   C. When the learning rate is very large, it is possible that some iterations of gradient descent may not decrease the value of loss function.

   D. With different initial weights, it is possible for the gradient descent algorithm to obtain to different local minimum.

2 (12 points) Error Metrics

The grid below shows $5 \times 4 = 20$ possible locations for targets to appear. We make guesses at the locations where targets are located, and mark those cells in blue. For the rest of the locations, we guess that those locations are non-targets, which are marked in white. The actual locations for the target are marked with a circle in cells, while the actual locations for non-target are marked with empty cells. Evaluate the following metrics for your guesses.



1. Compute the Recall using the following formula:

$$\text{Recall} = \frac{\text{number of true positives (correct guesses)}}{\text{number of actual targets}}$$

2. Compute the Precision using the following formula:

$$\text{Precision} = \frac{\text{number of true positives (correct guesses)}}{\text{number of guesses}}$$

3. Compute the F1 metric on this data. F1 is a way to measure classification accuracy. It's the harmonic mean of the precision and recall. F1 value is calculated using the following formula:

$$F_1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

3 (10 points) Logistic Regression Inference

We have a logistic regression classifier for a 2-dimensional feature vector $\mathbf{x} = (x_1, x_2)$:

$$p(y = +1|\mathbf{x}) = \frac{1}{1 + e^{-(2x_1 - x_2 + 1)}}$$

$$f(\mathbf{x}) = \begin{cases} 1, & p(y = +1|\mathbf{x}) \geq 0.5, \\ -1, & \text{otherwise.} \end{cases}$$

where $f(\mathbf{x}) \in \{-1, 1\}$ is the predicted label. Please solve the following problems.

1. Draw the decision boundary of the logistic regression classifier and shade the region where the classifier predicts 1. Make sure you have marked the x_1 and x_2 axes and the intercepts on those axes.

2. There are two new data points $\mathbf{x}_1 = (0.5, 3)$ and $\mathbf{x}_2 = (0.5, -2)$. Use the logistic regression defined above to:

   a. Calculate the probability that each data point belongs to the positive class. You may use a calculator or computer to get the numeric answer.

   b. Predict the labels for the new data points.

4 (40 points) Logistic Regression

Assume in a binary classification problem, we need to predict a binary label $y \in \{-1, 1\}$ for a feature vector $\mathbf{x} = [x_0, x_1]^\top$. In logistic regression, we can reformulate the binary classification problem in a probabilistic framework: We aim to model the distribution of classes given the input feature vector $\mathbf{x}$. Specifically, we can express the conditional probability $p(y|\mathbf{x})$ parameterized by $(\mathbf{w}, b)$ using logistic function as:

$$p(y|\mathbf{x}) = \frac{1}{1 + e^{-y(\mathbf{w}\cdot\mathbf{x}+b)}} \tag{1}$$

where $\mathbf{w} = [w_0, w_1]^\top$ is the weight vector, and b is the bias scalar. Given a training dataset $S_{\text{training}} = \{(\mathbf{x}_i, y_i)\}, i = 1, \ldots, n\}$, we wish to optimize the negative log-likelihood loss $\mathcal{L}(\mathbf{w}, b)$:

$$\mathcal{L}(\mathbf{w}, b) = -\sum_{i=1}^{n} \ln p(y_i|\mathbf{x}_i) \tag{2}$$

and find the optimal weight vector $\mathbf{w}$ and bias b to build the logistic regression model:

$$\mathbf{w}^*, b^* = \arg\min_{\mathbf{w},b} \mathcal{L}(\mathbf{w}, b) \tag{3}$$

- In this problem, we attempt to obtain the optimal parameters $\mathbf{w}^*$ and b^* by using a standard gradient descent algorithm. Assume $p_i = p(y_i|\mathbf{x}_i)$, the gradient for $\mathbf{w}$ and the gradient for b are shown as following:

$$\frac{\partial\mathcal{L}(\mathbf{w}, b)}{\partial\mathbf{w}} = -\sum_{i=1}^{n}(1 - p_i)y_i\mathbf{x}_i, \qquad \frac{\partial\mathcal{L}(\mathbf{w}, b)}{\partial b} = -\sum_{i=1}^{n}(1 - p_i)y_i. \tag{4}$$

  In reality, we typically tackle this problem in a matrix form: First, we represent data points as matrices $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^T$ and $Y = [y_1, y_2, \ldots, y_n]^T$. Thus, the negative log-likelihood loss $\mathcal{L}(\mathbf{w}, b)$ can be formulated as:

$$P = \text{sigmoid}(Y \circ (X\mathbf{w} + b\mathbf{1})), \qquad \mathcal{L}(\mathbf{w}, b) = -\mathbf{1}^T \ln P \tag{5}$$

  where $\mathbf{1} = [1, 1, ..., 1]^T \in \mathbb{R}^n$ is a n-dimensional column vector, $\ln(\cdot)$ is an element-wise natural logarithm function, $\text{sigmoid}(z) = \frac{1}{1+e^{-z}}$ is an element-wise sigmoid function, and "$\circ$" is an element-wise product operator. Similarly, we can have the gradient for $\mathbf{w}$ and b in the matrix form:

$$\frac{\partial\mathcal{L}(\mathbf{w}, b)}{\partial\mathbf{w}} = -X^T((\mathbf{1} - P) \circ Y), \qquad \frac{\partial\mathcal{L}(\mathbf{w}, b)}{\partial b} = -\mathbf{1}^T((\mathbf{1} - P) \circ Y). \tag{6}$$

- After obtaining the logistic regression model in Eq. 1 with the optimal $\mathbf{w}^*, b^*$ from gradient descent, we can use the following decision rule to predict the label $f(\mathbf{x}; \mathbf{w}^*, b^*) \in \{-1, 1\}$ of the feature vector $\mathbf{x}$:

$$f(\mathbf{x}; \mathbf{w}^*, b^*) = \begin{cases} 1, & \text{if } p(y = +1|\mathbf{x}) \geq 0.5 \\ -1, & \text{otherwise} \end{cases} \tag{7}$$

  Besides, the error e_{training} on the training set $S_{\text{training}} = \{(\mathbf{x}_i, y_i)\}$ is defined as:

$$e_{\text{training}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\big(y_i \neq f(\mathbf{x}_i; \mathbf{w}^*, b^*)\big) \tag{8}$$

  and we can define the test error e_{test} on the test set S_{test} in the same way.

Please download the notebook `logistic-regression.ipynb` from the course Canvas and fill in the missing blanks. Follow the instructions in the skeleton code and report:

1. Your code.
2. Equation of decision boundary corresponding to the optimal $\mathbf{w}^*$ and b^*.
3. Plot of training set along with decision boundary.
4. Plot of test set along with decision boundary.
5. Training error and test error.
6. Training loss curve.

5 (30 points) Perceptron

In this section, we will apply perceptron learning algorithm to solve the same binary classification problem as logistic regression above: We need to predict a binary label $y \in \{-1, 1\}$ for a feature vector $\mathbf{x} = [x_0, x_1]^\top$. The decision rule of the perceptron model is defined as:

$$f(\mathbf{x}; \mathbf{w}, b) = \begin{cases} 1, & \text{if } \mathbf{w}^T\mathbf{x} + b \geq 0, \\ -1, & \text{otherwise.} \end{cases} \tag{9}$$

where $\mathbf{w} = [w_0, w_1]^\top$ is the weight vector, and b is the bias scalar. Given a training dataset $S_{\text{training}} = \{(\mathbf{x}_i, y_i)\}, i = 1, \ldots, n\}$, we define the training error e_{training} as:

$$e_{\text{training}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\big(y_i \neq f(\mathbf{x}_i; \mathbf{w}, b)\big) \tag{10}$$

and the test error e_{test} on the test set S_{test} can be defined in the same way. In the perceptron algorithm, we aim to directly minimize the training error e_{training} in order to obtain the optimal parameters $\mathbf{w}^*, b^*$. If we represent data points in training set S_{training} as matrices $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^T$ and $Y = [y_1, y_2, \ldots, y_n]^T$, the perceptron algorithm is shown as below:

Algorithm 1 Perceptron Learning Algorithm

Data: Training set S_{training}, which contains feature vectors X and labels Y;
Initialize parameters $\mathbf{w}$ and b; pick a constant $\lambda \in (0, 1]$, which is similar to the step size in the standard gradient descent algorithm ($\lambda = 1$ by default).
while *not every data point is correctly classified* **do**
  for *each feature vector* $\mathbf{x}_i$ *and its label* y_i *in the training set* $S_{training}$ **do**
    compute the model prediction $f(\mathbf{x}_i; \mathbf{w}, b)$;
    if $y_i = f(\mathbf{x}_i; \mathbf{w}, b)$ **then**
      continue;
    else
      update the parameters $\mathbf{w}$ and $\mathbf{b}$:
        $\mathbf{w} \Leftarrow \mathbf{w} + \lambda(y_i - f(\mathbf{x}_i; \mathbf{w}, b))\mathbf{x}_i$
        $\mathbf{b} \Leftarrow \mathbf{b} + \lambda(y_i - f(\mathbf{x}_i; \mathbf{w}, b))$
    end
  end
end

Please download the notebook `perceptron.ipynb` from the course website and fill in the missing blanks. Follow the instructions in the skeleton code and report:

1. Your code.
2. Equation of decision boundary corresponding to the optimal $\mathbf{w}^*$ and b^*.
3. Plot of training set along with decision boundary.
4. Plot of test set along with decision boundary.
5. Training error and test error.
6. Training error curve.