Machine Learning, 2024 Spring
Assignment 3

Notice

Plagiarizer will get 0 points.
LaTeXis highly recommended. Otherwise you should write as legibly as possible.

Problem 1 For logistic regression, show that

$$\nabla E_{\text{in}}(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N}\frac{y_n\mathbf{x}_n}{1+e^{y_n\mathbf{w}^{\mathrm{T}}\mathbf{x}_n}} = \frac{1}{N}\sum_{n=1}^{N}-y_n\mathbf{x}_n\theta\left(-y_n\mathbf{w}^{\mathrm{T}}\mathbf{x}_n\right) \tag{1}$$

Argue that a 'misclassified' example contributes more to the gradient than a correctly classified one. (15pt) Solution

- **Show that** $\nabla E_{\text{in}}(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N}\frac{y_n\mathbf{x}_n}{1+e^{y_n\mathbf{w}^{\mathrm{T}}\mathbf{x}_n}} = \frac{1}{N}\sum_{n=1}^{N}-y_n\mathbf{x}_n\theta\left(-y_n\mathbf{w}^{\mathrm{T}}\mathbf{x}_n\right)$ (7pt)

  The $E_{in}(w)$ is shown as

  $$E_{in}(w) = \frac{1}{N}\sum_{n=1}^{N}\ln(\theta(y_n\mathbf{w}^T\mathbf{x}_n)) = \frac{1}{N}\sum_{n=1}^{N}\ln(1+e^{-y_n\mathbf{w}^T\mathbf{x}_n}) \tag{2}$$

  For formula/equation (1), the equation can be rewritten by moving the minus (-) into the summation. Therefore we have

  $$\begin{aligned}\nabla E_{in}(\mathbf{w}) &= \frac{1}{N}\sum_{n=1}^{N}-\frac{y_nx_n}{1+e^{-y_n\mathbf{w}^Tx_n}} \\ &= \frac{1}{N}\sum_{n=1}^{N}(-y_nx_n\times\frac{1}{1+e^{-y_n\mathbf{w}^Tx_n}})\end{aligned} \tag{3}$$

  It is clear that $\frac{1}{1+e^{y_n\mathbf{w}^Tx_n}}$ is in the form of $\theta(u) = \frac{1}{1+e^{-x}}$, $\frac{1}{1+e^{y_n\mathbf{w}^Tx_n}}$ is equal to $\theta(-y_n\mathbf{w}^Tx_n)$, so we have

  $$\nabla E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}-y_n\mathbf{x}_n\theta\left(-y_n\mathbf{w}^{\mathrm{T}}\mathbf{x}_n\right) \tag{4}$$

- **Argue that a 'misclassified' example contributes more to the gradient than a correctly classified one:** (8pt)

  For a sample (x_n, y_n),when the sign of y_n and $\mathbf{w}^T\mathbf{x}_n$ are the same, $y_n\mathbf{w}^T\mathbf{x}_n > 0$, means that the sample is correctly classified. When $y_n\mathbf{w}^T\mathbf{x}_n < 0$, it implies the sample is misclassified.

  - **Correctly Classified:** For $y_n\mathbf{w}^T\mathbf{x}_n > 0$, $\theta(-y_n\mathbf{w}^T\mathbf{x}_n) < \frac{1}{2}$, thus this sample contributed less to the gradient.
  - **Misclassified:** For $y_n\mathbf{w}^T\mathbf{x}_n < 0$, $\theta(-y_n\mathbf{w}^T\mathbf{x}_n) > \frac{1}{2}$, thus this sample contributed more to the gradient.

Problem 2 For linear regression, the out-of-sample error is

$$E_{\text{out}}(h) = \mathbb{E}[(h(x) - y)^2]. \tag{5}$$

Show that among all hypotheses, the one that minimizes E_{out} is given by

$$h^*(x) = \mathbb{E}[y|x]. \tag{6}$$

The function h^* can be treated as a deterministic target function, in which case we can write $y = h^*(x) + \epsilon(x)$ where $\epsilon(x)$ is an (input dependent) noise variable. Show that $\epsilon(x)$ has expected value zero. (15pt)
Solution

- **Show that among all hypotheses, the one that minimizes E_{out} is given by $h^*(x) = \mathbb{E}[y|x]$.** (8pt)
  The out-of-sample error is expanded as follows:

$$E_{out}(h) = \mathbb{E}[(h(x) - y)^2] = \mathbb{E}[h(x) - 2h(x)y + y^2] \tag{7}$$

  Rewrite this expression using $\mathbb{E}[y|x]$, we have

$$\begin{aligned} E_{out}(h) &= \mathbb{E}[(h(x) - \mathbb{E}[y|x] + \mathbb{E}[y|x] - y)^2] \\ &= \mathbb{E}[(h(x) - \mathbb{E}[y|x])^2 + 2(h(x) - \mathbb{E}[y|x])(\mathbb{E}[y|x] - y) + (\mathbb{E}[y|x] - y)^2] \end{aligned} \tag{8}$$

  $\mathbb{E}[y|x] - y$ doesn't depend on the choice of $h(x)$. Therefore we have

$$\mathbb{E}[\mathbb{E}[y|x] - y] = 0. \tag{9}$$

  With which we have,

$$E_{out}(h) = \mathbb{E}[(h(x) - \mathbb{E}[y|x])^2] + \mathbb{E}[(\mathbb{E}[y|x] - y)^2] \tag{10}$$

  Since $\mathbb{E}[(\mathbb{E}[y|x] - y)^2]$ is not related to $h(x)$, it is obvious that when $h(x) = \mathbb{E}[y|x]$, $E_{out}(h)$ attains minimum. Therefore, we have $h^*(x) = \mathbb{E}[y|x]$.

- **Show that $\epsilon(x)$ has expected value zero** (7pt)
  With $y = h^*(x) + \epsilon(x)$, we have $\epsilon(x) = y - h^*(x)$, then we have:

$$\mathbb{E}[\epsilon(x)] = \mathbb{E}[y - h^*(x)] = \mathbb{E}[y] - \mathbb{E}[h^*(x)] \tag{11}$$

  Given that $h^*(x) = \mathbb{E}[y|x]$, we have

$$\mathbb{E}[h^*(x)] = \mathbb{E}[\mathbb{E}[y|x]] = \mathbb{E}[y] \tag{12}$$

  Therefore,

$$\mathbb{E}[\epsilon(x)] = \mathbb{E}[y] - \mathbb{E}[y] = 0. \tag{13}$$

  This completes the proof that the expected value of $\epsilon(x)$ is zero, thereby also verifying that $h^*(x) = \mathbb{E}[y|x]$ is the optimal solution for minimizing E_{out}.

Problem 3 According to the iterative format provided as follow:

- Use the SUV dataset to implement (using Python or MATLAB) the Gradient Descent method to find the optimal model for logistic regression. (30pt)
- What is your test error? What are the sizes of the training set and test set, respectively? (10pt)
- What is your learning rate? How was it chosen? How many steps were iterated in total? (10pt)
- Present the results of the last 10 steps produced by your algorithm, including the loss, learning rate, the L2 norm of the gradient, and the number of function evaluations and gradient evaluations.(20pt)

Fixed learning rate gradient descent:

1: Initialize the weights at time step $t = 0$ to $\mathbf{w}(0)$.
2: **for** $t = 0, 1, 2, \ldots$ **do**
3: Compute the gradient $\mathbf{g}_t = \nabla E_{\text{in}}(\mathbf{w}(t))$.
4: Set the direction to move, $\mathbf{v}_t = -\mathbf{g}_t$.
5: Update the weights: $\mathbf{w}(t+1) = \mathbf{w}(t) + \eta \mathbf{v}_t$.
6: Iterate to the next step until it is time to stop.
7: Return the final weights.

Dataset reference and description
Dataset and download

Solution

- **Use the SUV dataset to implement (using Python or MATLAB) the Gradient Descent method to find the optimal model for logistic regression.** (30pt)
  - Students should provide a code, otherwise deduct 10pt.
  - Different coding languages such as C or C++ don't influence scores.
  - The Key point is to check the gradient function. The framework could referred to in Figure 1.
- **What is your test error? What are the sizes of the training set and test set, respectively?** (10pt)
  - The value of the test error depends on the training set division and the iteration times. So any reasonable answer is acceptable.(6pts)
  - Any size of the training set and test set could give the scores, as long as it is reasonable. (eg. 320;80 is OK, but 80:320 is not OK) (4pt)
- **What is your learning rate? How was it chosen? How many steps were iterated in total?** (10pt)
  - Any reasonable learning rate including 0.01,0.3,0.1... is OK. (2pt)
  - But students should give an appropriate reason otherwise the reason part (5pt) will get 0pt.(For example, the reason can be choosing the learning rate with the Checkerboard Method or another reasonable answer).
  - The steps were iterated in total are also decided by students, anything reasonable is fine (3pt).
- **Present the results of the last 10 steps produced by your algorithm, including the loss, learning rate, the L2 norm of the gradient, and the number of function evaluations and gradient evaluations.**(20pt)

  Anything that can show this algorithm converging is fine.

  It should be noted that all terms should be included in the comparison. The number of function evaluations and gradient evaluations means the number to count the objective function and gradient. The results could refer to Figure 2. It should be noted that these results can be different from the answers of the students.

```python
import pandas
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.preprocessing import StandardScaler

class LogisticRegression:
    def __init__(self,learning_rate=0.1,num_iterations=1000):...
    def sigmoid(self,x):
        return 1/(1+np.exp(-x))
    def fit(self,X_train,y_train):
        print("learning rate:",self.learning_rate)
        nums_samples,num_features=X_train.shape
        self.weight=np.zeros(num_features)
        self.gradient_descent(X_train,y_train)
    def predict(self,x):
        y_pre_prob=self.sigmoid(x.dot(self.weight))
        return np.array(y_pre_prob>=0.5,dtype='int')
    def score(self,x_test,y_test):
        return accuracy_score(y_test,self.predict(x_test))
    def count_y(self,x,weight):
        return self.sigmoid(np.dot(x, weight))
    def count_loss(self,weight, x, y):
        y_pre = self.count_y(x,weight)
        error = y * np.log(y_pre) + (1 - y) * np.log(1 - y_pre)
        loss = -np.sum(error) / len(y)
        self.count_loss_number+=1
        return loss

    def count_gradient(self,x, y):
        self.count_gradient_number+=1
        return x.T.dot(self.count_y(x,self.weight) - y) / len(y)

    def gradient_descent(self,x, y):
        i = 0
        while i < self.num_iterations:
            gradient = self.count_gradient(x, y)
            #new_weight = self.weight
            self.weight = self.weight - self.learning_rate * gradient
            if self.num_iterations-i<=10:
                print("iteration:",i,"loss:",self.count_loss(self.weight,x,y),"the L2 norm of the gradient:"
                      ,np.linalg.norm(gradient),"number of function evaluations",self.count_loss_number,
                      "number of gradient evaluations",self.count_gradient_number)

            i += 1
data_set=pandas.read_csv('suv_data.csv')
X,Y=data_set.iloc[:,2:4],data_set.iloc[:,4]
seed=0
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=seed)
scaler=StandardScaler()
X_train=scaler.fit_transform(X_train)
X_test=scaler.transform(X_test)

model=LogisticRegression()
model.fit(X_train,Y_train)
predictions=model.predict(X_test)
report=classification_report(Y_test,predictions)
print(report)

score=model.score(X_test,Y_test)
print(score)
```

Figure 1: The code of gradient descent algorithm

```
learning rate: 0.1
iteration: 990 loss: 0.4342215123807467 the L2 norm of the gradient: 0.00034798516938237904 number of function evaluations 1 number of gradient evaluations 991
iteration: 991 loss: 0.43422150041971685 the L2 norm of the gradient: 0.00034627332581198207 number of function evaluations 2 number of gradient evaluations 992
iteration: 992 loss: 0.43422148857607257 the L2 norm of the gradient: 0.00034456995582242708 number of function evaluations 3 number of gradient evaluations 993
iteration: 993 loss: 0.43422147684865814 the L2 norm of the gradient: 0.0003428750241122315 number of function evaluations 4 number of gradient evaluations 994
iteration: 994 loss: 0.4342214652363293 the L2 norm of the gradient: 0.000341188481187266 number of function evaluations 5 number of gradient evaluations 995
iteration: 995 loss: 0.4342214537379531 the L2 norm of the gradient: 0.00033951028737807187 number of function evaluations 6 number of gradient evaluations 996
iteration: 996 loss: 0.43422144235240784 the L2 norm of the gradient: 0.0003378404008295O2 number of function evaluations 7 number of gradient evaluations 997
iteration: 997 loss: 0.4342214310785826 the L2 norm of the gradient: 0.00033617877990134274 number of function evaluations 8 number of gradient evaluations 998
iteration: 998 loss: 0.4342214199153778 the L2 norm of the gradient: 0.0003345253831672251 number of function evaluations 9 number of gradient evaluations 999
iteration: 999 loss: 0.43422140886170474 the L2 norm of the gradient: 0.0003328801694134163 number of function evaluations 10 number of gradient evaluations 1000
```

Figure 2: The results of the gradient descent algorithm.