Machine Learning

Homework 2

B02901143 楊筑雅

2016/10/28

I. Logistic regression

1. Logistic regression

   The output of the function is between 0 and 1, which represents the probability of the label belonging to class 1.

```
f = 1 / (1 + safe_exp(-(np.sum(train_X[i] * W) + b)))
```

2. Cross entropy

   I use cross entropy to evaluate the goodness of the function. It is easier to optimize the function using cross entropy than using square error.

```
train_L += -(train_y[i] * safe_ln(f) + (1 - train_y[i]) *
safe_ln(1 - f))
```

3. Gradient descent

   I use gradient descent to optimize the model.

```
dW += -(train_y[i] - f) * train_X[i]
db += -(train_y[i] - f)
```

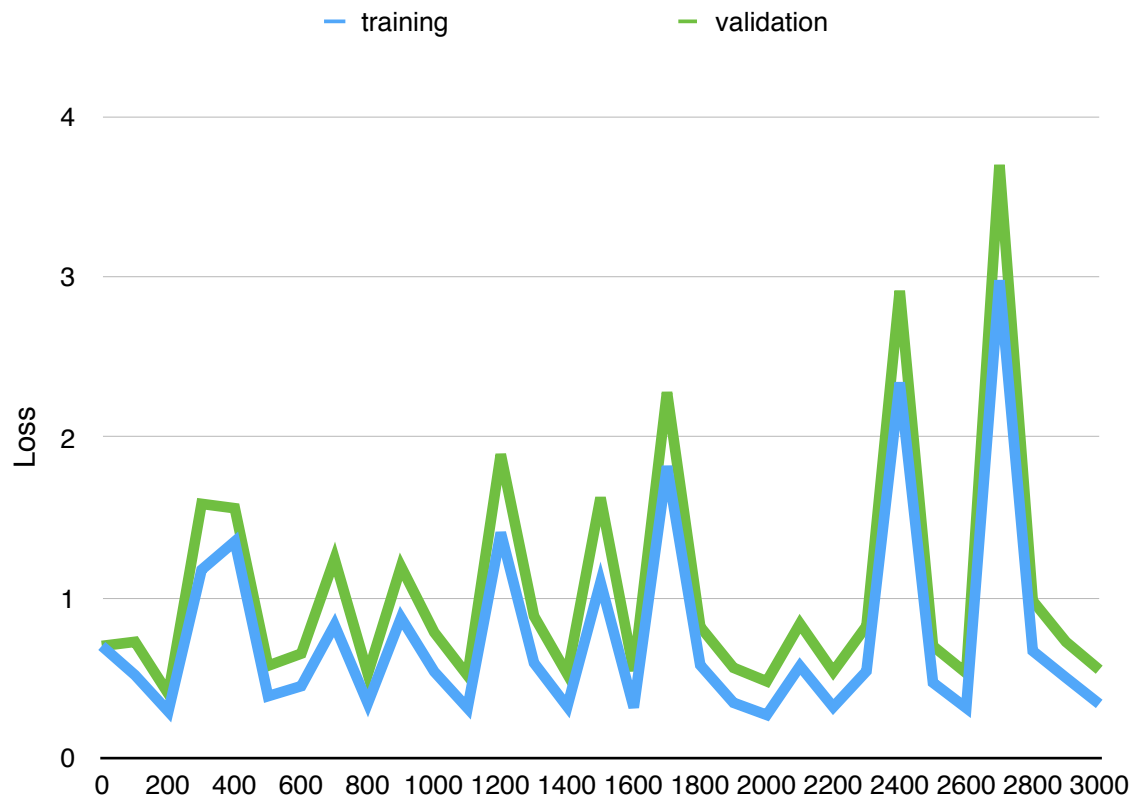
4. Adam update rule

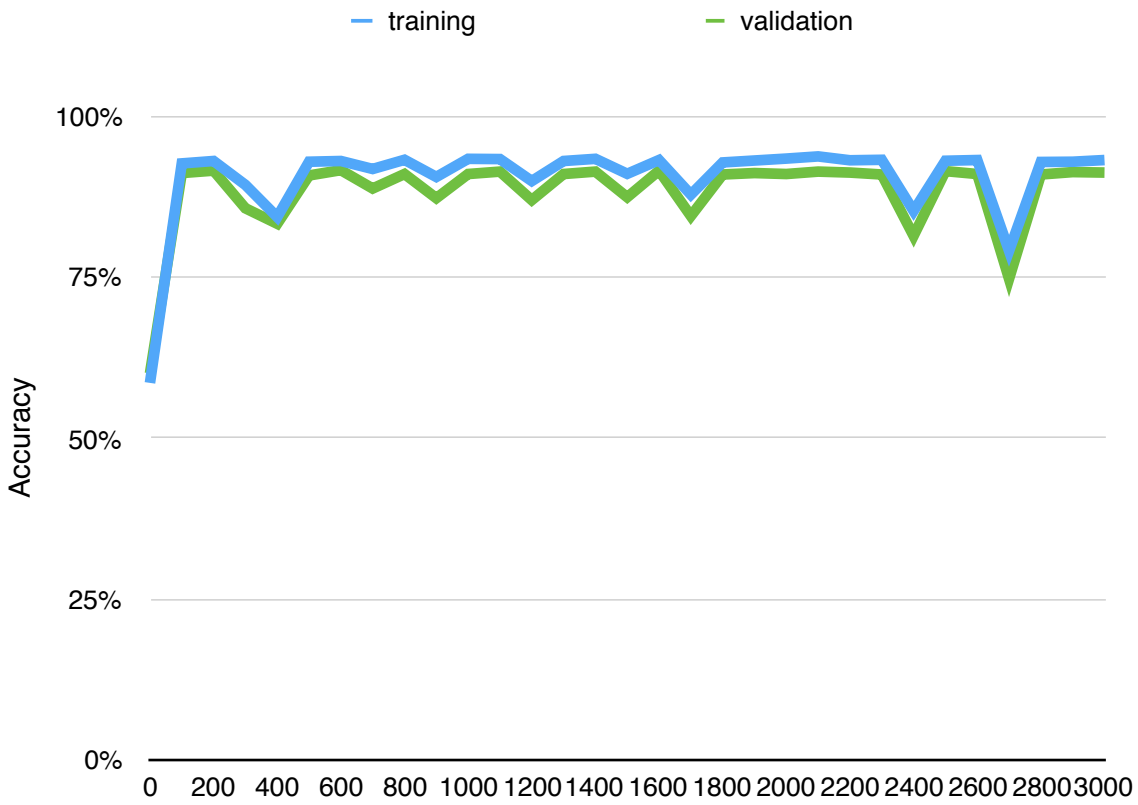
   Adam is the best way to update the parameters up to present. It is like RMSprop with momentum.

```
#adam update
m_W = beta1 * m_W + (1 - beta1) * dW
v_W = beta2 * v_W + (1 - beta2) * (dW ** 2)
m_b = beta1 * m_b + (1 - beta1) * db
v_b = beta2 * v_b + (1 - beta2) * (db ** 2)
#bias correction
m_W_hat = m_W / (1 - beta1 ** (it + 1))
v_W_hat = v_W / (1 - beta2 ** (it + 1))
m_b_hat = m_b / (1 - beta1 ** (it + 1))
v_b_hat = v_b / (1 - beta2 ** (it + 1))
W -= eta * m_W_hat / (np.sqrt(v_W_hat) + eps)
b -= eta * m_b_hat / (np.sqrt(v_b_hat) + eps)
```

5. Results

   Figures below are the loss and the accuracy of training and validation. The highest accuracy is 93.18% and 91.23% for training data and validation data respectively.

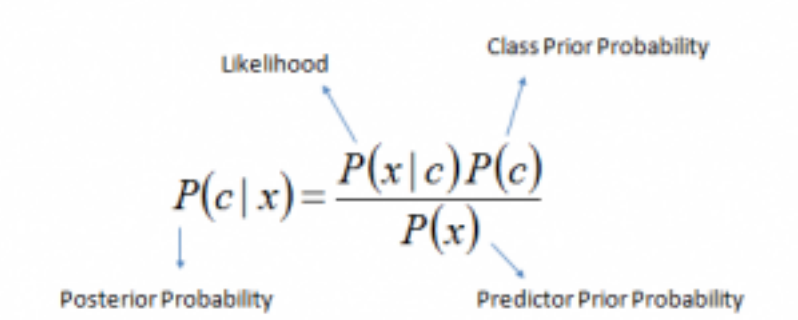




II. Naive Bayes

1. Assumption and theory

Naive bayes assumed that the probability of each attribute belonging to a given class value is independent of all other attributes.



$$P(c \mid \mathbf{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

The probability of a class value given a value of an attribute is called the conditional probability. By multiplying the conditional probabilities together for each attribute for a given class value, I have a probability of a data instance belonging to that class. To make a prediction, I can calculate probabilities of the instance belonging to each class and select the class value with the highest probability.

2. Calculate mean and standard derivation of each attribute
   I assumed that each attribute is in Gaussian distribution, so I need to calculate the mean and the standard derivation at first. Also, the means and the standard derivations are the parameters I stored in the model.

```
#calculate the mean and the std
mean_0 = np.mean(train_0, axis=0)
std_0 = np.std(train_0, axis=0)
mean_1 = np.mean(train_1, axis=0)
std_1 = np.std(train_1, axis=0)
```

3. Calculate class probabilities
   I can use the mean and the standard derivation to calculate the probabilities of a given attribute value. Then I multiply all probabilities to get the class probability.

```
def getClassProb(x, mean, std):
    exponent = np.exp(-(np.power(x - mean, 2) \
               / (2 * np.power(std, 2))))
    prob = (1 / (np.sqrt(2 * np.pi) * std)) * exponent
    return np.prod(prob, axis=1)
```

4. Predict
   The class with the highest class probability is the predicted label of the instance.

```
def predict(x, prob_0, prob_1):
    label = np.ndarray(shape=(x.shape[0], 1))
    for i in range(x.shape[0]):
        if prob_0[i] > prob_1[i]:
            label[i] = 0
        else:
            label[i] = 1
    return label
```

5. Results

I get the accuracy of 82.30%, 82.33% and 85% for training data, pubic testing data and private testing data respectively.

III. Discussion

It is quite obvious that logistic regression has better results than naive bayes. The reason why is that attributes are not independent of all other attributes since some words may appear frequently after some other words. To be fair, with such simple assumption, naive bayes gives robust results in a fast and effective way. However, I should use logistic regression if I want to have better results.