Machine Learning Hw2 B03901036 陳柏文
Collaborator: B03901037 鄭宇強 B03901137 張致綱

1. **Implementation of logistic regression function**

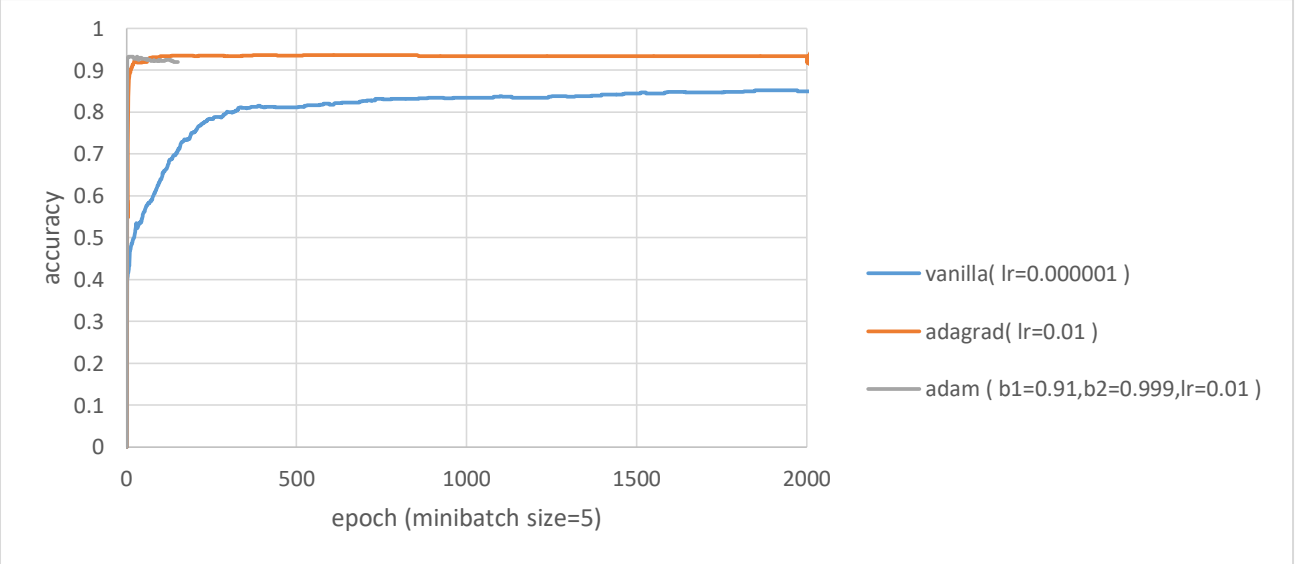
The vanilla implementation of my logistic regression function is constructed with the following setting, optimizer = Adagrad, (If I remove the optimizer and set learning rate= 10^{-6}, the convergence speed rate will be really slow), learning rate = 0.01 and picking cross entropy as objective function for my model. Here, I paste partial code to state my way for implementing sigmoid and feed forward function.

```
def sigmoid(gamma):      # Because the exponential of big positive number isn't computable on python,
    if gamma < 0:        # I let it compute in a conditional way.
        return 1 - 1 / (1 + math.exp(gamma))
    return 1 / (1 + math.exp(-gamma))
def feedForward(sample):
    z = (sum([a*b for a,b in zip(sample,weight[0])]) + bias)
    return sigmoid(z)
```

The following is some setting about my training process.

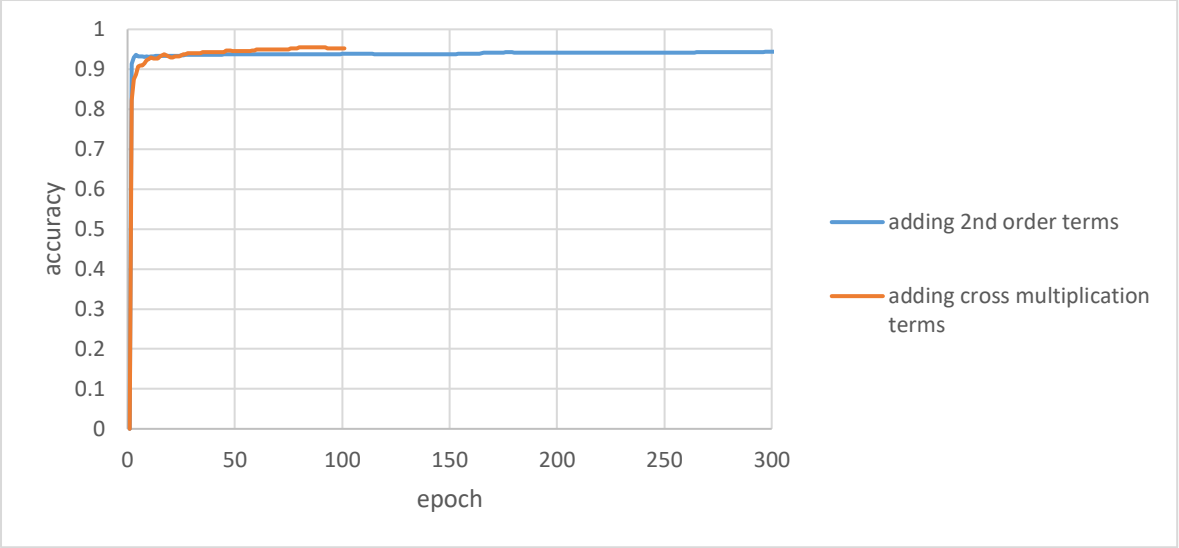
(a) **Use of optimizer – Adam**

Compared to Adagrad, Adam converge faster, but actually it doesn't improve the performance or accuracy. The picture below is my comparison based on three methods. First one is the model trained without any optimizer. Second one is the model trained with Adagrad and setting learning rate equals 0.01. Third one is the model trained with Adam and setting learning rate equals 0.01.



(b) **Use of 2nd order term in logistic regression model**

Using of 2nd order term for all 57 features helps improve the accuracy, which means adding 57 more weights into my model. So my prediction turns into $\hat{y} = \sigma(b + \sum_{i=0}^{n} w_i x_i + w_i' x_i^2)$, where n is the number of feature and σ is sigmoid function. Compared to vanilla model, accuracy increased from 0.93 to 0.94. And I also try adding **cross multiplication terms** into my model. (For example, for data sample x, $x[2]$ is the values related to feature 2 and $x[3]$ is the values related to feature 3. And I refer those $x[i] * x[j]$ to cross multiplication term. Thus, my prediction turns into $\hat{y} = \sigma(b + \sum_{i=0}^{n} w_i x_i + \sum_{i=0}^{n} \sum_{j=i+1}^{n} w_{ij} x_i x_j)$. However, adding these cross multiplication terms doesn't really help improving accuracy. It overfitted quickly with just few epochs. It can reach accuracy of 0.98 on training data, but only get 0.94 on validation data set.



(c) **Use of k fold cross validation**

In order to solve the problem of overfitting after adding cross multiplication terms. Besides adding regularization term to the objective function, I slice the training data into ten subsets and train ten models based on those subset. And, in the end, I take average of these ten models and make it my final model. I get the accuracy of 0.94667 based on this model.

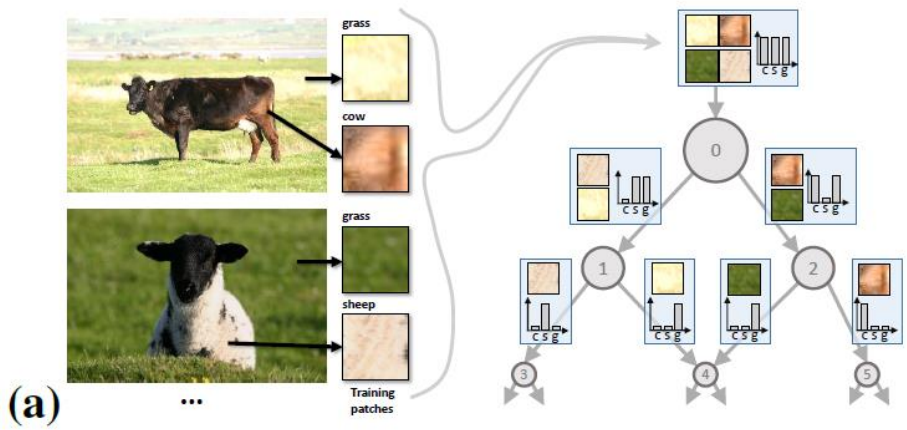
2. **<u>Implementation of second method – Random Forest</u>**

For the second method, I implement decision tree and random forest which is lots of decision trees combining together. In these implementation, we have to slice data set into left and right branches based on one particular feature. For any data sample, if its value related to that feature is over the threshold, then we will pass this data sample to the right branch and vice versa. So we have to determine which feature and the value of threshold. Among all the feature, we will pick the feature and threshold which has the largest impurity gain, which is the difference of **Shannon entropy** of the data set before and after slicing it.

(a) **Random forest**

After several test, I pick the following hyper-parameters, number of levels for each individual tree = 20, number of random selected features = 30 and 2/3 random chosen part of data set as training data for each decision tree. For these settings, I got an accuracy of 0.95.

(b) **Decision jungle** (an implementation of the paper: Jamie Shotton, Toby Sharp, Pushmeet Kohli, Sebastian Nowozin, John Winn, Antonio Criminisi, ***Decision Jungles: Compact and Rich Models for Classification***)



And to explain why decision jungle, there is an example to help describe the motivation, using of a rooted

decision DAG for classifying image patches as belonging to grass, cow or sheep classes. Using DAGs instead of trees reduces the number of nodes and can result in better generalization. For example, differently coloured patches of grass (yellow and green) are merged together into node 4, because of similar class statistics. This may encourage generalization by representing the fact that grass may appear as a mix of yellow and green. So using DAGs can help reduce the nodes in each level of the trees in the jungle, and also improve the accuracy. And I got an accuracy of 0.96 based on DAGs.

3. **Other discussion and detail**

Logistic model is based on the hypothesis that the distribution for data sample is a Gaussian distribution. So I wonder that what if the actual distribution for the data sample is not a Gaussian, if so, we will always have a limit on your best accuracy.