Machine Learning Assignment 2

Tung-Chun, Chiang R05922027

October 21, 2016

1 Logistic regression function

```
def sigmoid(x):
  return 1.0 / (1.0 + np.exp(-x))

def gradient_descent(x, y):
  # x is feature vector and y is target value
  # W, b are weight vector and bias of this model
  infer = sigmoid(np.dot(x, W) + b)
  # cost = cross entropy
  cost = -np.sum(y * np.log(infer) + (1.0 - y) * np.log(1.0 -
     infer))
  # cost is not used in compute gradients, so we also compute
     cost_partial
  cost_partial = sigmoid(np.dot(x, W) + b) - y

  # gw is the gradient of W
  gw = np.dot(cost_partial, x)

  # gb is the gradient of b
  gb = np.sum(cost_partial)

  # update parameters
  W -= learningRate * gw
  b -= learningRate * gb

  return
```

2 My methods

2.1 Feature normalization

$x_{normalized} = (x - mean(x))/std(x)$

2.2 Models

2.2.1 Neuron Network

We use a fully connected neuron network with neurons in 3 hidden layers = [50, 50, 50], 1 neuron in output layer, the activation function is sigmoid and the cost function is cross entropy. The output value from output layer can be treated as probabilities $P(is\_spam|x)$. To predict test data, we assign predict y = 1 if $P(is\_spam|x) >= 0.5$ and 0 otherwise.(The sample model is shown in Figure 1)

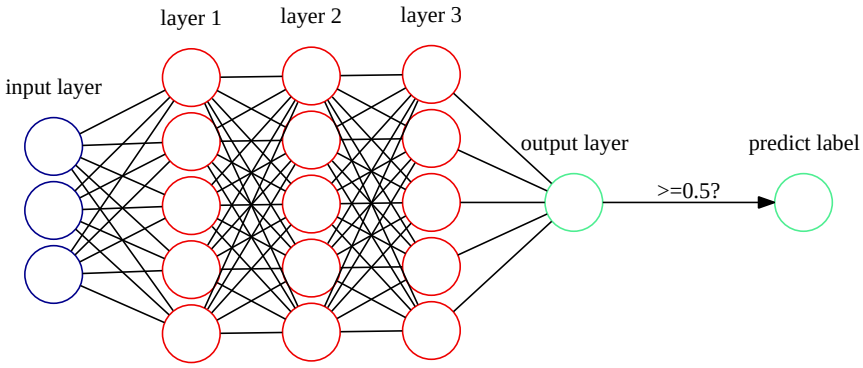

Figure 1: Neuron Network for input dimension = 3 and 3 hidden layers

2.2.2 Ensemble Neuron Network

To avoid overfitting, ensemble models are often used, such as Random Forest or Adaboost. In this section, we introduce our ensemble model, ensemble Neuron Network. First we use bagging method to randomly sample features from origin features(we sample 0.9 of origin features as new features and the whole instances). We create some estimators(in this case, some NNs as above section), and each estimator has its feature space sampled by bagging. To inference, a test instance x will be predicted by all estimators with corresponding feature space. Each estimator gives a predict label for x as voting. And we use the class with max votes as final predict label. e.g., if there are 100 estimators, 55 of them give test instance x label 1 and 45 give label 0, we use label 1 as the final predict label.(The sample model is shown in Figure 2)

3 Experiments

3.1 Overfitting

3.1.1 Settings

In this experiment, we use 57-dim features ($x_{normalized}$), and train each model to appropriate E_{in}, batch size 100 and Adagrad method. Error function is $0/1error$.

3.1.2 Result & Analysis

In Table 1, we show E_{kaggle} of each model with close E_{in}. For logistic regression model(logReg), it's hard to get close error rate, so we just post the best result of it. You can see that even though

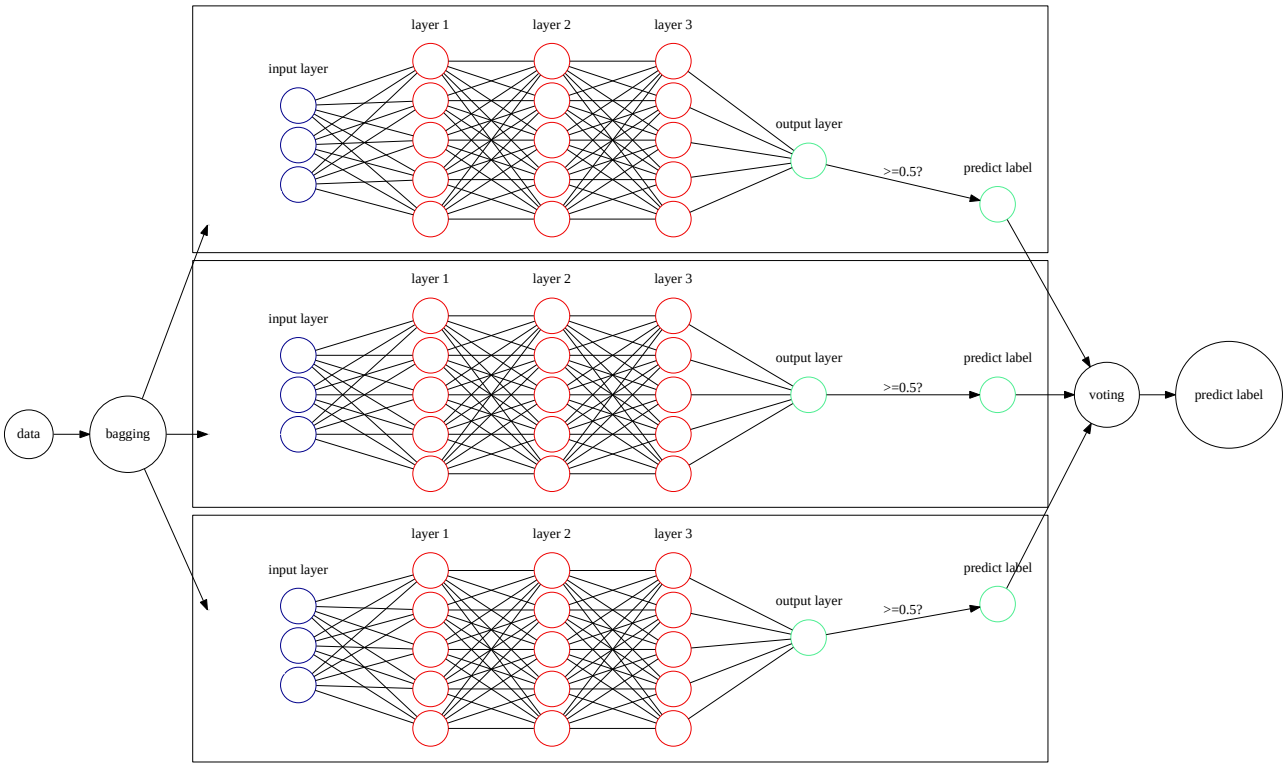

Figure 2: Ensemble Neuron Network for 3 estimators(NN), and each NN has input dimension = 3 and 3 hidden layers

model, result	E_{in}	E_{kaggle}
enNN	0.01200	0.04333
NN	0.00775	0.05667
logReg	0.06148	0.07333
enlogReg.	0.00582	0.06333

Table 1: E_{in} and E_{kaggle} of different models

enNN(ensemble NN)'s E_{in} is worse than NN and enlogReg, enNN's E_{kaggle} is the best. And enlogReg(ensemble logistic regression) outperforms the origin logReg. It is apparent that ensemble methods are better. (but it costs more time of training)