CS 573 Homework 3

Yifan Fei

Computer Science
Purdue University

Instructor: Prof. Bruno Ribeiro

Spring 2018

Section 0:

1

Yes, I discussed the homework with others but came up with my own answers. Their name(s) are Yupeng Han and Meng Liu.

2

I have used online resources to help me answer this question, but I came up with my own answers. Here is a list of the websites I have used in this homework:

Vanishing gradient: https://datascience.stackexchange.com/questions/5706/what-is-the-dying-relu-problem-in-neural-networks

Vanishing gradient: https://ayearofai.com/rohan-4-the-vanishing-gradient-problem-ec68f76ffb9b

Q1: Deep Learning

1: New equations of the new neural network

Forward pass:

$$z_1 = xW_1 + b_1$$
$$h_1 = ReLu(z_1)$$
$$z_2 = h_1W_2 + b_2$$
$$h_2 = ReLu(z_2)$$
$$z_3 = h_2W_3 + b_3$$
$$\hat{y} = softmax(z_3)$$

back propagation:

The derivative of the last layer parameters:

$$\frac{\partial L(i)}{\partial W_3} = \frac{\partial L(i)}{\partial z_3}\frac{\partial z_3}{\partial W_3} = {h_{2,i}}^T(y_i - \hat{y_i})$$

$$\frac{\partial L(i)}{\partial h_2} = \frac{\partial L(i)}{\partial z_3}\frac{\partial z_3}{\partial h_2} = W_3(y_i - \hat{y_i})$$

$$\frac{\partial L(i)}{\partial b_3} = \frac{\partial L(i)}{\partial z_3}\frac{\partial z_3}{\partial b_3} = (y_i - \hat{y_i})$$

$$\frac{\partial L(i)}{\partial z_2} = \frac{\partial L(i)}{\partial h_2}\frac{\partial h_2}{\partial z_2}$$

$$\frac{\partial L(i)}{\partial b_2} = \frac{\partial L(i)}{\partial h_2}\frac{\partial h_2}{\partial b_2} = \frac{\partial L(i)}{\partial h_2}$$

$$\frac{\partial L(i)}{\partial W_2} = \frac{\partial L(i)}{\partial h_2}\frac{\partial h_2}{\partial z_2}\frac{\partial z_2}{\partial W_2} = \frac{\partial L(i)}{\partial h_2}\frac{\partial h_2}{\partial z_2}h_1$$

where

$$\frac{\partial h_2}{\partial z_2} = \begin{cases} 1, & \text{if } h_2 > 0. \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

$$\frac{\partial L(i)}{\partial h_1} = \frac{\partial L(i)}{\partial z_2}\frac{\partial z_2}{\partial h_1} = W_2\frac{\partial L(i)}{\partial z_2}$$

$$\frac{\partial L(i)}{\partial z_1} = \frac{\partial L(i)}{\partial h_1}\frac{\partial h_1}{\partial z_1}$$

$$\frac{\partial L(i)}{\partial b_1} = \frac{\partial L(i)}{\partial h_1}\frac{\partial h_1}{\partial b_1} = \frac{\partial L(i)}{\partial h_1}$$

where

$$\frac{\partial h_1}{\partial z_1} = \begin{cases} 1, & \text{if } h_1 > 0. \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

$$\frac{\partial L(i)}{\partial W_1} = \frac{\partial L(i)}{\partial h_1}\frac{\partial h_1}{\partial z_1}\frac{\partial z_1}{\partial W_1} = \frac{\partial L(i)}{\partial h_1}\frac{\partial h_2}{\partial z_2}x$$

The back propagation output:

$$dW_1 = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L(i)}{\partial W_1}$$

$$dW_2 = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L(i)}{\partial W_2}$$

$$dW_3 = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L(i)}{\partial W_3}$$

$$db_1 = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L(i)}{\partial b_1}$$

$$db_2 = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L(i)}{\partial b_2}$$

$$db_3 = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L(i)}{\partial b_3}$$

2: Accuracy with the new neural network

Accuracy after 10 iterations with new neural network: 0.9973333333333333

This result is better than the first neural network with 1 hidden layer.

The reason why the model is more accurate now is that it adds one more layer to the network, making the model more complicated and fits the data better. Plus adding the bias enlarge the search space.

3

the 20 accuracy values using a boxplot

The original code with 20 runs 100 iterations each over validation data:

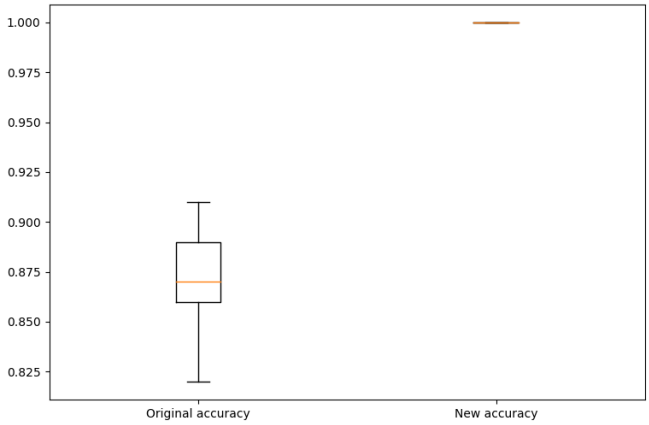


Figure 1: the 20 accuracy values using a boxplot

Null and alternative hypothesis, t-test

Suppose the original accuracy value and new accuracy value are a_1, a_2 here the experiment accuracy values:

$x_1 = 0.873, std_1 = 0.022383, x_2 = 1.0, std_2 = 0,$

Null hypothesis: There is no significant difference to merit using test data. i.e. the true mean difference is zero, $H_0 : a_2 - a_1 = 0$.

Alternative hypothesis: There is significant difference to merit using test data. i.e. $H_1 : a_2 - a_1 \geq 0$.

t-test Calculation:

$$t = \frac{(x_2 - x_1) - (a_2 - a_1)}{SE}$$

Here $a_2 - a_1 = 0$,

$$SE = \sqrt{\frac{{s_1}^2}{n_1} + \frac{{s_2}^2}{n_2}} = \sqrt{\frac{0.022383^2}{20} + \frac{0^2}{20}} = 0.0050$$

Degree of freedom is:

$$DF = \left\lfloor \frac{({\sigma_1}^2/n_1 + {\sigma_2}^2/n_2)^2}{({\sigma_1}^2/n_1)^2/(n_1 - 1) + ({\sigma_2}^2/n_2)^2/(n_2 - 1)} \right\rfloor = \left\lfloor \frac{(0.022383^2/20 + 0^2/20)^2}{(0.022383^2/20)^2/(19) + (0^2/20)^2/(19)} \right\rfloor = 19$$

So

$$t_d = \frac{(1 - 0.873) - 0}{0.0050} = 25.4000$$

$$p = P(T_{DF} \geq t_d) = 6.4988e - 16$$

which is very small. So confidence level $= 1 - p$, we have about 100% confidence to reject H_0. So there is difference to merit using the test data.

4: learning curves

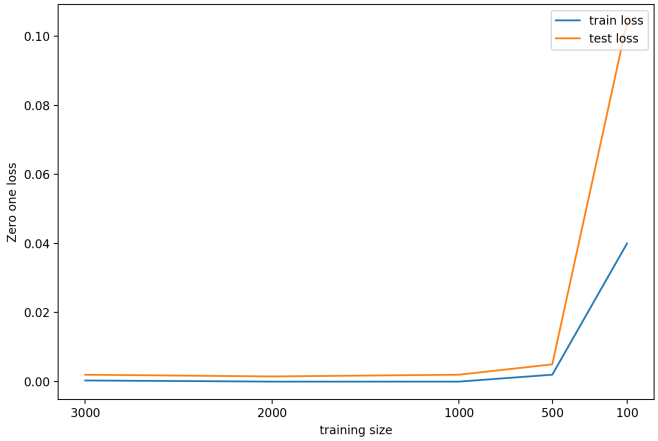


Figure 2: learning curves using 0-1 loss

The zero one loss is the opposite of accuracy. As is shown, the test loss is a little bigger than the train loss, which is reasonable due to the overfitting. Also, as the training data size increases from 100 to 3000, the test loss and train loss both decrease.

5: Gradient Vanishing

ReLUs "die" (output zero) when the input to it is negative. This can completely block back propagation because the gradients will just be zero after one negative value has been inputted to the ReLU function. If at least one input has our ReLU on the steep side, then the ReLU is still alive because there's still learning going on and weights getting updated for this neuron. If all inputs put the ReLU on the flat side, there's no hope that the weights change at all and the neuron is dead.

Since

$$\frac{\partial h_1}{\partial z_1} = \begin{cases} 1, & \text{if } h_1 > 0. \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

$$dW_1 = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L(i)}{\partial W_1}$$

$$db_1 = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L(i)}{\partial b_1}$$

for all training sample x_i in the mini batch, we must have all $z_{1,i} = x_i W_1 + b_1 \leq 0$

Q2: Cross Validation

1: Learning curve with AUC scores

Fixed the learning rate as 0.01, and the mini-batch size as 200, we got the Library with AUC in sklearn is used. The learning curve is like:

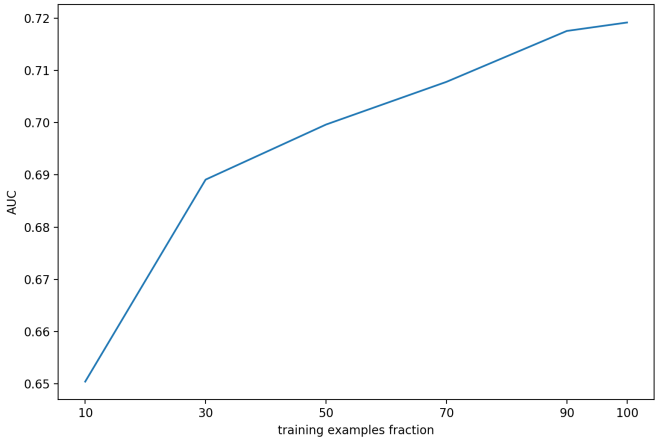
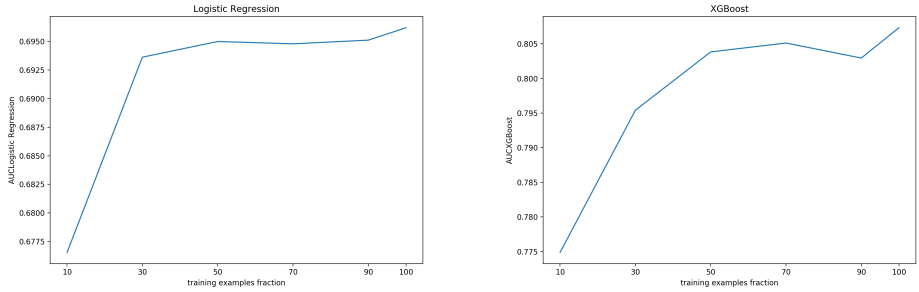


Figure 3: Learning curve using AUC scores

As is shown, the AUC scores go up when the training size gets bigger.

2: Learning curves of logistic regression and boosted decision trees



(a) Learning curves of logistic regression

(b) Learning curves of boosted decision trees

Figure 4: Learning curves of logistic regression and boosted decision trees

As is shown, both AUC scores go up when the training size gets bigger.

3: k-fold, paired t-test

k-fold for Neural network

With minibatch size as 200, epoch as 100, learning rate as 0.01, we get:

[Fold 1/10 Prediciton, AUC score: 0.719663126912]

[Fold 2/10 Prediciton, AUC score: 0.720314516745]

[Fold 3/10 Prediciton, AUC score: 0.706347834129]

[Fold 4/10 Prediciton, AUC score: 0.72865542282]

[Fold 5/10 Prediciton, AUC score: 0.72399028205]

[Fold 6/10 Prediciton, AUC score: 0.715989693816]

[Fold 7/10 Prediciton, AUC score: 0.713633735629]

[Fold 8/10 Prediciton, AUC score: 0.712252291998]

[Fold 9/10 Prediciton, AUC score: 0.726857419707]

[Fold 10/10 Prediciton, AUC score: 0.714873156984]

Mean AUC score for NN: 0.718257748079

Std AUC score for NN: 0.00661079341745

k-fold for Logistic Regression

[Fold 1/10 Prediciton, AUC score: 0.685363632343]

[Fold 2/10 Prediciton, AUC score: 0.690829074504]

[Fold 3/10 Prediciton, AUC score: 0.687819412221]

[Fold 4/10 Prediciton, AUC score: 0.70599952151]

[Fold 5/10 Prediciton, AUC score: 0.714927548494]

[Fold 6/10 Prediciton, AUC score: 0.67140225342]

[Fold 7/10 Prediciton, AUC score: 0.70433732003]

[Fold 8/10 Prediciton, AUC score: 0.699915491709]

[Fold 9/10 Prediciton, AUC score: 0.711582470624]

[Fold 10/10 Prediciton, AUC score: 0.694750816376]

Mean AUC score for LogisticRegression: 0.696692754123

Std AUC score for LogisticRegression: 0.0126273560628

k-fold for Xgboost

[Fold 1/10 Prediciton, AUC score: 0.837212855701]

[Fold 2/10 Prediciton, AUC score: 0.847836139269]

[Fold 3/10 Prediciton, AUC score: 0.83449621832]

[Fold 4/10 Prediciton, AUC score: 0.843883570099]

[Fold 5/10 Prediciton, AUC score: 0.846185845717]

[Fold 6/10 Prediciton, AUC score: 0.841705596886]

[Fold 7/10 Prediciton, AUC score: 0.834767153974]

[Fold 8/10 Prediciton, AUC score: 0.841023184885]

[Fold 9/10 Prediciton, AUC score: 0.83593153519]

[Fold 10/10 Prediciton, AUC score: 0.835188688178]

Mean AUC score for Xgboost: 0.839823078822

Std AUC score for Xgboost: 0.0062007457868

paired t-test Calculation: we are testing multiple hypothesis.

1. **First test:** Xgboost model accuracy is better than Logistic Regression.

   Null hypothesis: Xgboost model accuracy is the same as Logistic Regression. i.e. $a_3 - a_2 = 0$.

   Alternative hypothesis: Xgboost model accuracy is better than Logistic Regression. i.e. $a_3 - a_2 \geq 0$.

   t-test Calculation:

   $$t = \frac{(x_3 - x_2) - (a_3 - a_2)}{SE}$$

   Here $a_3 - a_2 = 0$,

   $$SE = \sqrt{\frac{{s_3}^2}{n_3} + \frac{{s_2}^2}{n_2}} = \sqrt{\frac{0.00620074578682}{10} + \frac{0.012627356062 8^2}{10}} = 0.004448588196824$$

   The degree of freedom can be calculated by the the following formula:

   $$DF = \left\lfloor \frac{({\sigma_3}^2/n_3 + {\sigma_2}^2/n_2)^2}{({\sigma_3}^2/n_3)^2/(n_3 - 1) + ({\sigma_2}^2/n_2)^2/(n_2 - 1)} \right\rfloor = \left\lfloor \frac{(0.0062007457868^2/10 + 0.012627356062 8^2/10)^2}{(0.0062007457868^2/10)^2/(9) + (0.012627356062 8^2/10)^2/(9)} \right\rfloor$$

   $$DF = 13$$

   By the table to calculate the critical t-value, we have:

   $$t = \frac{(x_3 - x_2) - (a_3 - a_2)}{SE} = \frac{0.839823078822 - 0.696692754123}{0.004448588196824} = 32.1743$$

   For this is one-tailed test. Since $t_{0.9995} = 4.221 < 25.4000$ when $DF = 13$

   $$p = P(T_{DF} \geq t_d) = 1.11876e - 10$$

   which is very small.

   correcting for the fact that we are testing multiple hypotheses:

   Since we are testing multiple hypotheses, like the Paul Miracle in PPT, setting p-value should be divided by 2 because here we have two models challenging xgboost. So here p-value should be doubled.

   $$p = (1.11876e - 10) * 2 = 2.237e - 10$$

   But even the confidence derease due to multiple hypotheses, we still have about 100% confidence to reject H_0.

2. **Second test:** Xgboost model is better than Neural network.

   Null hypothesis: Xgboost model accuracy is the same as Neural network. i.e. $H_0 : a_3 - a_1 = 0$.

   Alternative hypothesis: Xgboost model accuracy is better than Neural network. i.e. $H_1 : a_3 - a_1 \geq 0$.

   The typical p-value is 0.05. This means that there is 95% confidence that the conclusion of this test will be valid.

   t-test Calculation:

$$t = \frac{(x_3 - x_1) - (a_3 - a_1)}{SE}$$

   Here $a_3 - a_1 = 0$,

$$SE = \sqrt{\frac{{s_3}^2}{n_3} + \frac{{s_1}^2}{n_1}} = \sqrt{\frac{0.0062007457868^2}{10} + \frac{0.00661079341745^2}{10}} = 0.002866214191590$$

   The degree of freedom can be calculated by the the following formula:

$$DF = \left\lfloor \frac{({\sigma_3}^2/n_3 + {\sigma_2}^2/n_2)^2}{({\sigma_3}^2/n_3)^2/(n_3 - 1) + ({\sigma_2}^2/n_2)^2/(n_2 - 1)} \right\rfloor = \left\lfloor \frac{(0.0062007457868^2/10 + 0.00661079341745^2/10)^2}{(0.0062007457868^2/10)^2/(9) + (0.00661079341745^2/10)^2/(9)} \right\rfloor$$

$$DF = 17$$

   By the table to calculate the critical t-value, we have:

$$t = \frac{(x_3 - x_1) - (a_3 - a_2)}{SE} = \frac{0.839823078822 - 0.718257748079}{0.002866214191590} = 42.4132$$

   For this is one-tailed test. Since $t_{0.9995} = 3.965 < 25.4000$ when $DF = 17$

$$p = P(T_{DF} \geq t_d) = 8.00039e - 11$$

   which is very small.

   correcting for the fact that we are testing multiple hypotheses:

   Since we are testing multiple hypotheses, like the Paul Miracle in PPT, setting p-value should be divided by 2 because here we have two models challenging xgboost. So here p-value should be doubled.

$$p = (8.00039e - 11) * 2 = 1.6e - 10$$

   But even the confidence level decrease due to multiple hypotheses, we still have about 100% confidence to reject H_0.

In conclusion, xgboost is the best model based on multiple hypotheses and t-test.

4: Hyper-parameter tuning

As it said on piazza, **I did not use the normalization of the training and testing data.**

Table 1: **Hyper-parameter Tunning**

10 fold	100,0.01	100,0.001	1000,0.01	1000,0.001
1/10 fold	0.72642971	0.71211106	0.68097972	0.7003197
2/10 fold	0.72017242	0.69016398	0.69807681	0.68410502
3/10 fold	0.71302471	0.70866448	0.69696601	0.69631282
4/10 fold	0.71575632	0.7136613	0.69883041	0.68707421
5/10 fold	0.71932839	0.70145295	0.70033761	0.7009405
6/10 fold	0.71268941	0.7126891	0.69886388	0.6919007
7/10 fold	0.71164398	0.70595973	0.69974978	0.69406911
8/10 fold	0.70064722	0.71103104	0.68446566	0.68676546
9/10 fold	0.71003639	0.70425506	0.68160962	0.68731317
10/10 fold	0.69890773	0.69488639	0.69103942	0.6812775

Choosing the best 10 hyperparameter combos to test the data and finally get the result:

This fold use hyperparameter: 100, 0.01 [Fold 1/10 Prediciton, AUC score: 0.71286122984]

This fold use hyperparameter: 100, 0.01 [Fold 2/10 Prediciton, AUC score: 0.717176687482]

This fold use hyperparameter: 100, 0.01 [Fold 3/10 Prediciton, AUC score: 0.711655957603]

This fold use hyperparameter: 100, 0.01 [Fold 4/10 Prediciton, AUC score: 0.733453391449]

This fold use hyperparameter: 100, 0.01 [Fold 5/10 Prediciton, AUC score: 0.726364874366]

This fold use hyperparameter: 100, 0.01 [Fold 6/10 Prediciton, AUC score: 0.707164671304]

This fold use hyperparameter: 100, 0.01 [Fold 7/10 Prediciton, AUC score: 0.715730830518]

This fold use hyperparameter: 100, 0.001 [Fold 8/10 Prediciton, AUC score: 0.722638048674]

This fold use hyperparameter: 100, 0.01 [Fold 9/10 Prediciton, AUC score: 0.729868594141]

This fold use hyperparameter: 100, 0.01 [Fold 10/10 Prediciton, AUC score: 0.695001118048]

Mean AUC score for NN: 0.717191540342

Std AUC score for NN: 0.0108786539055

test 1:

Null hypothesis: Xgboost model accuracy is the same as Neural network. i.e. $H_0 : a_3 - a_1 = 0$.

Alternative hypothesis: Neural network model accuracy is not the same as Xgboost model. i.e. $H_1 : a_1 - a_3 \neq 0$.

test 2:

Null hypothesis: Logistic Regression model accuracy is the same as Neural network. i.e. $H_0 : a_1 - a_2 = 0$.

Alternative hypothesis: Neural network model accuracy is not the same as Logistic Regression model. i.e. $H_1 : a_1 - a_2 \neq 0$.

This procedure does not change the hypothesis test. We still compare model with models. The hyper-parameter tunning is inside the k-fold cross validation.

5: Formalize the null and alternative hypothesis

null hypothesis: This 4 distinct neural network models are the same as logistic regression or xgboost.

alternative hypothesis: This 4 distinct neural network models have different model accuracy with logistic regression or xgboost.

This procedure does change the hypothesis test from question Q2(4). In Q2(4) we first do hyper-parameter tunning then choose the best performance hyper-parameter during each fold split to represent the neural network. Then we say run k-fold to get the performance based on these hyper-parameters. That is to say, we compare model(Neural Network) with model(Other models like xgboost). However, in this problem, the hypothesis becomes to compare some neural network model with specific hyper-parameters with other general models, which is wrong.