National Chiao Tung University
Spring 2019
Deep Learning
Instructor: Jen-Tsung Chien

Deep Learning HW1 Report

Alfons Hwu
Student ID: 0416324

alfons.cs04@g2.nctu.edu.tw
Dept of Computer Science
Writing with LaTeXon Overleaf

Deep Learning HW1 Report

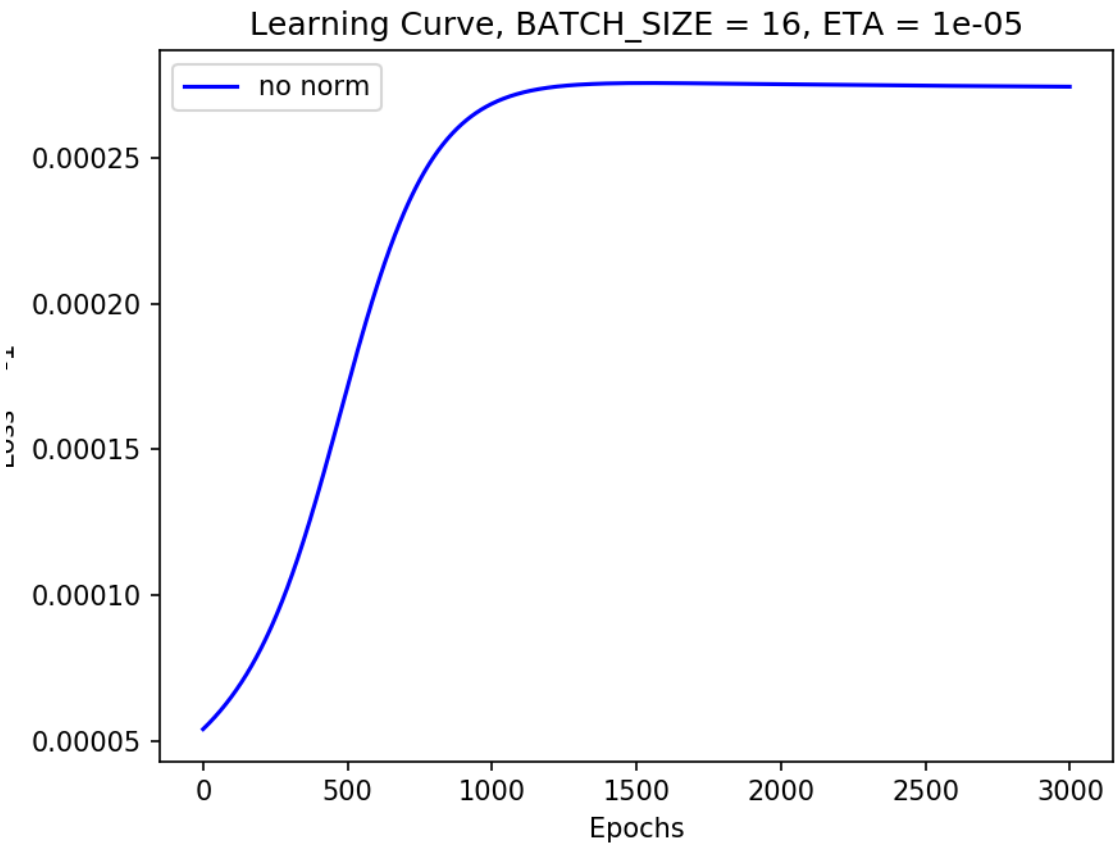
April 5, 2019

1 Self-designed DNN for binary classification

Loss is defined by

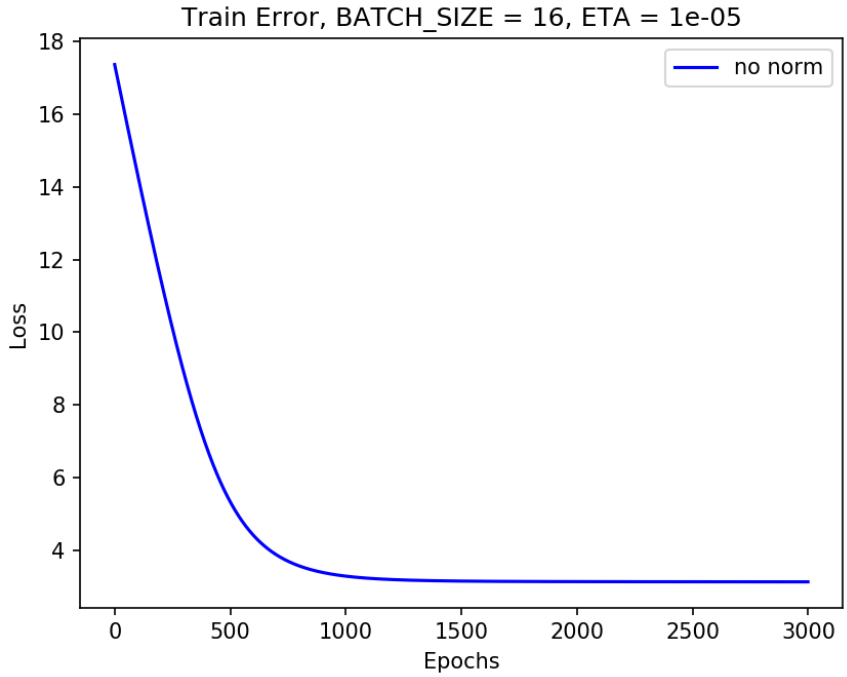
$$E(w) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_k(X_n, w)$$

1.1 Learning Curve



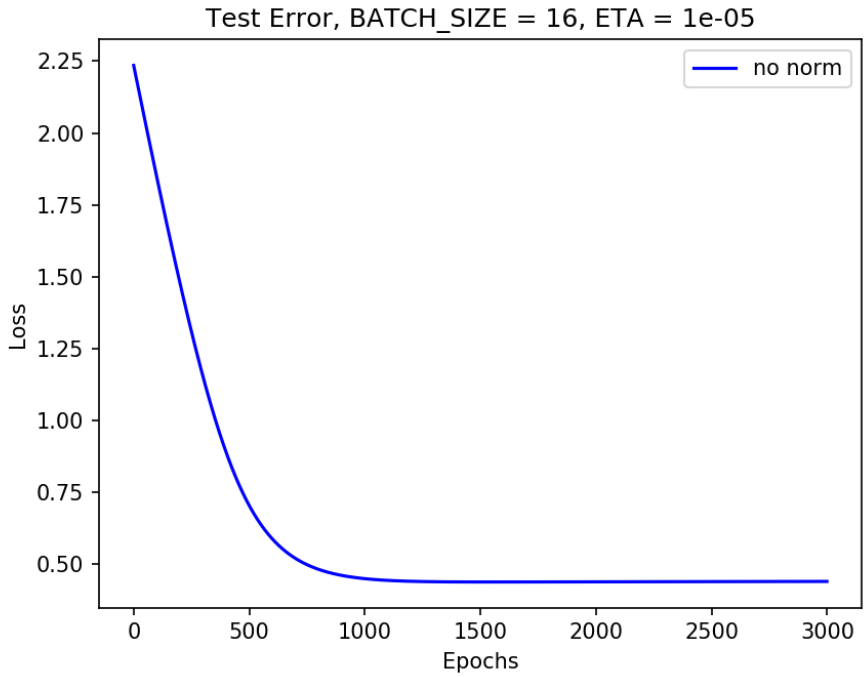
The Y-axis represents 1/Loss, telling the learning curve(loss is dropping and finally converged)

1.2 Training Loss



The Y-axis represents loss, telling the learning curve(loss is dropping and finally converged)

1.3 Testing Loss



The Y-axis represents loss, telling the learning curve(loss is dropping and finally converged)

1.4 Explanation of my neural network architecture

- Layer: Input 6unit (input including 6 features, which is 6 dimensions).
  Hidden 4unit, since in my opinion, what influences the result most lies in the learning rate, the neuron set 4 will be optimal for computation (too many neurons merely increases the matrix computation time)
  Click for ref
  Output: 1unit for alive or death.
  For the following two items, I select them based on the combination with shell script, and comparing the optimal results using the **same random seed** for benchmarking the learning curve vs batch size and learning rate. (Choose the normal learning rate first (such as 0.5) and slightly decrease by empirical method (automated training with shell script and comparing with pyplot figure).
  According to this article the **high learning rate will cause the non-convergent of learning curve.**
  With the aforementioned site, I start up with learning rate 0.5 and empirically dropped down to about **0.00001 +- 0.000005**

```
if [ $sel -eq 1 ];
then
    batch_size=$2
    for learning_rate in 0.000001 0.000003 0.000005 0.000008 0.00001 0.000
    do
        echo $learning_rate
        python3 dnn_1.py $1\_$batch_size\_$learning_rate $batch_size $lea
    done
else
    python3 dnn_1.py $1_8_0.000005 8 0.000005
fi

mkdir -p $1_1
mv $1*\.png $1_1/
```

- Batch size: 16 for optimal, (too large will consume too much memory resource, and too small will cause the unstable learning curve. Although more randomness, less chance to converge)
  Ref link

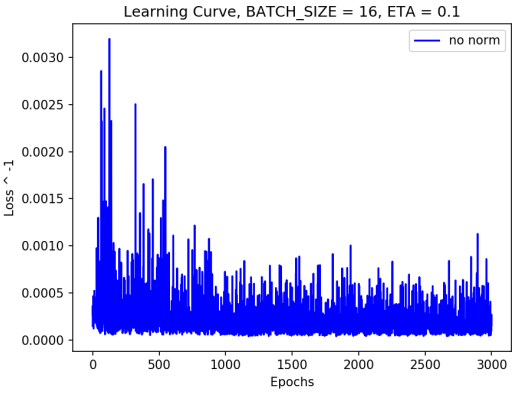


Figure 1: Learning rate = 0.1

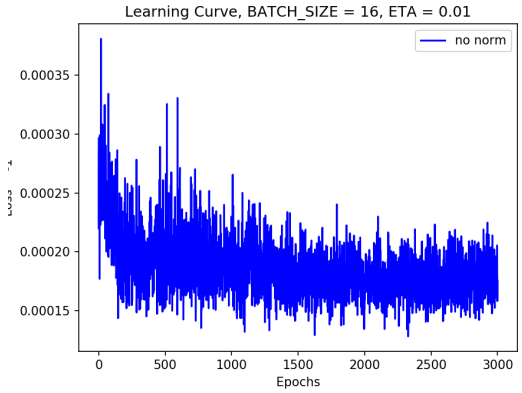
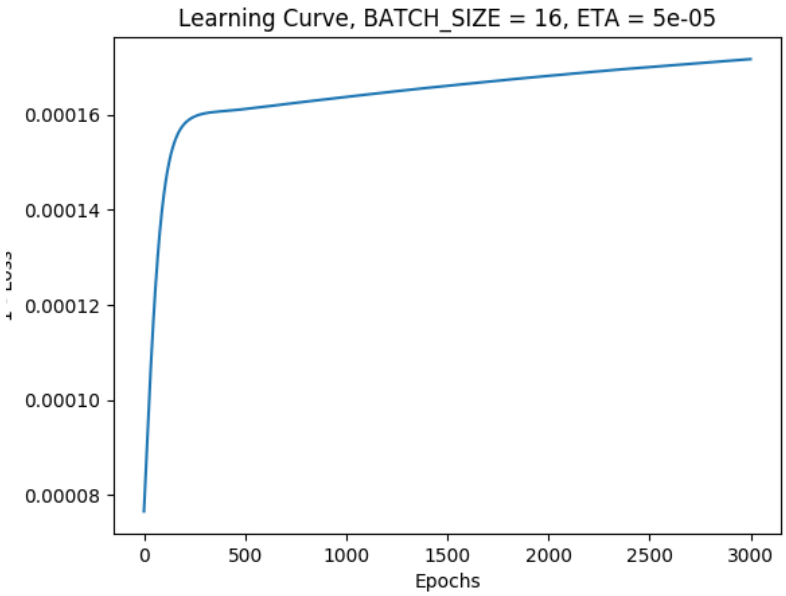


Figure 2: Learning rate = 0.01

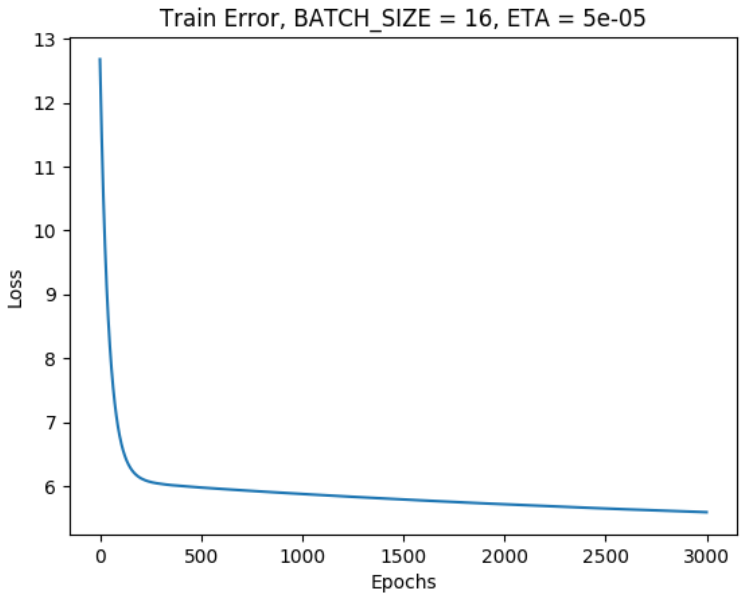
- Learning rate: 0.00001 (1e-5) will be optimal (The above figures shows some high learning rate causing non-convergent of learning curve)

2 TA-designed DNN

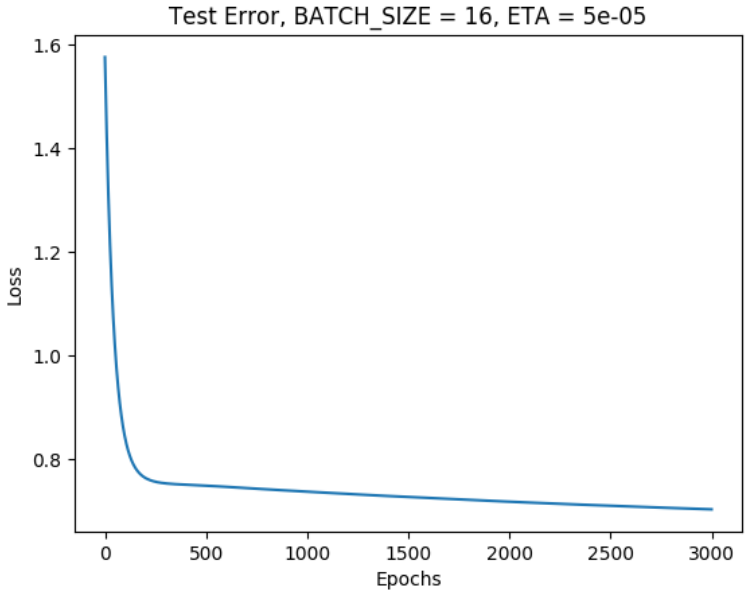
The selection of learning rate and training batch size is the same as above



Learning curve

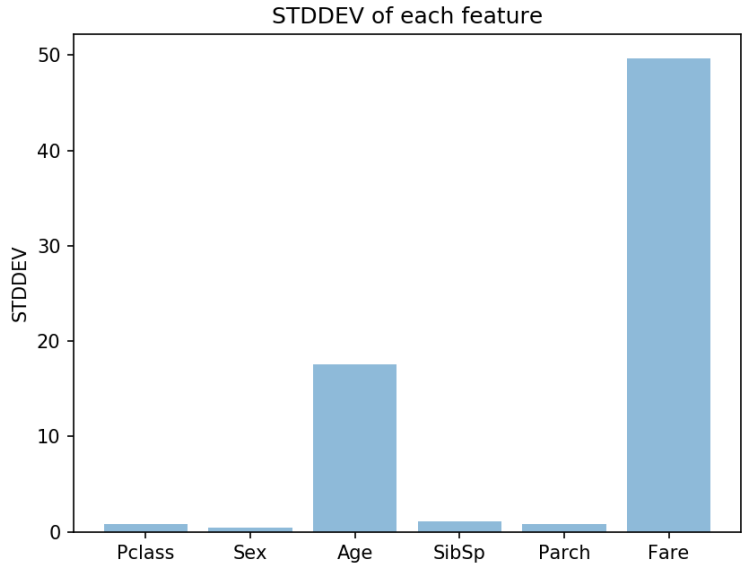


Training error



Testing error

3 The normalization of features

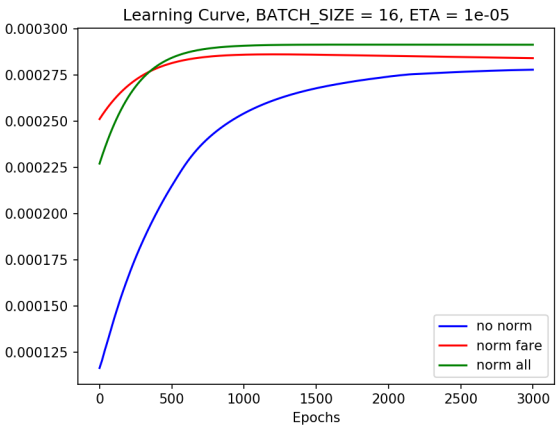


Why normalize the data?

Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. For machine learning, every dataset does not require normalization. It is required only when features have different ranges.

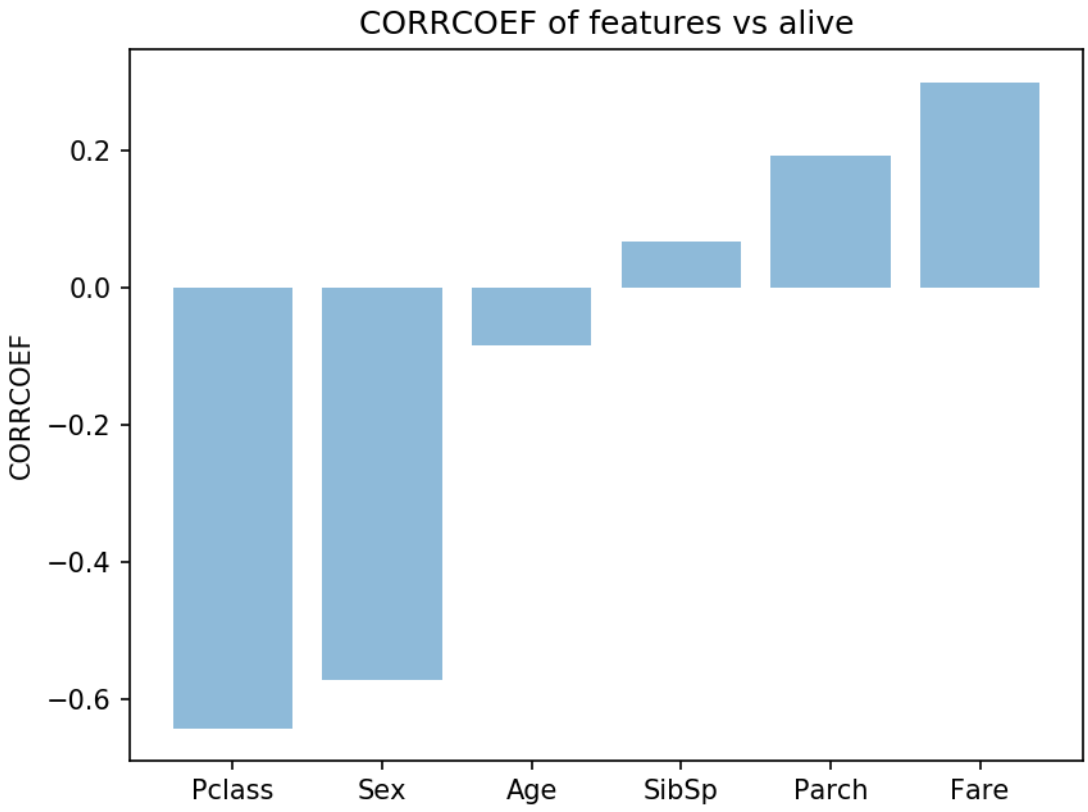
Aside from the 'fare' feature, the standard deviation of 'age' is big as well, hence for convenience and performance purpose, in the third test, I normalize all the classes.

The following figure is the result of normalized fare and normalized all the features, which shows the better performance under normalization (use the same random seed, same batch size and same learning rate for fairness), the normalized results converged better and quicker . Note: Same cross entropy formula is used.



4 Feature that affects the prediction performance most

By calculating the correlation coefficient for the feature vs output (alive or death), the following figure shows the **fare feature** affects the most (the higher the fare is, the more chance a passenger will survive).



5 Do we need one hot encoding?

I downloaded the **full dataset from kaggle**, and use the dict to collect how many categories lies under the ticket class.

It turns out to be that there are 594 difference b/w the training set and testing set by using the following code.

```
def parse(data):
    data = np.array(data)
    col = int(sys.argv[1])
    to_check = data[:,col]
    m = dict()
```

```
    for i in to_check:
        if i in m:
            m[i] += 1
        else:
            m[i] = 1
    for k, v in m.items():
        print(k, ' ', v)
    return m
def cmp_map(m1, m2):
    diff_dict = m1.keys() - m2.keys()
    print('Total differences ', len(diff_dict))

if __name__ == '__main__':
    label, train_data, test_data, all_data = file_IO()
    m1 = parse(train_data)
    m2 = parse(test_data)
    cmp_map(m1, m2)
```

```
370376   1
Total differences  594
 alfons@mbp  …/HW1  master ● ?
```

For the purpose of one-hot encoding, that is to **quantifies** the class since only some algorithms can work with categorical data directly. For example, a decision tree can be learned directly from categorical data with no data transform required (this depends on the specific implementation).

Many machine learning algorithms cannot operate on label data directly. They require all input variables and output variables to be numeric.

Nonetheless, in this dataset, the difference categories b/w the training set and testing set is 594 in the class 'ticket', rather big for merely a 91 testing set. Suppose on-hot encoding is used, each category is paired with an encoded data.

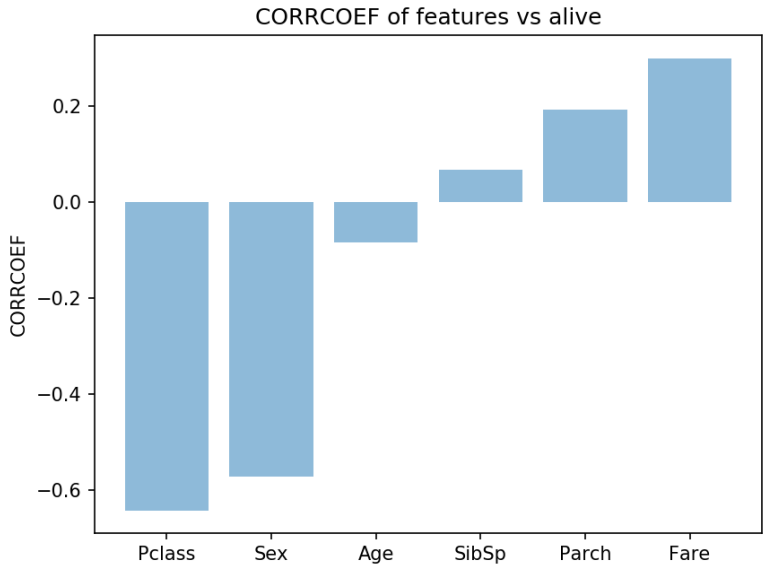
For example, 2427 –> 001 5578 –> 010 7790 –> 100 308889 –> 101 in the training set. Due to so many different categories in the training set, this results in a big sparsity.

Suppose the testing set have input data somewhat like 'B2007T' or '8888' which none of them exists in the training set before, there will be no suitable data for NN to predict correctly

Unless further pre-processing is applied, one hot is not an idel choice here

6 Artificially design 2 samples for one alive and dead

According to the covariance coefficient



What makes one survive the most is the 'fare' class and 'pclass', the former is positively related while the latter is negatively related. This means the more a passenger spends on the ticket and the less of pclass (kind like the first class, business and economy class in the plane), the more opportunity he/she will survive.

Let the chance of survive > 0.5 as alive, otherwise dead to be the split line, here we have the artificially designed two samples for the problem.

```
self sample  [5, 1, 13, 2, 4, 3] chance of survive?  [[0.26384242]]
self sample  [0, 1, 13, 2, 4, 500.0] chance of survive?  [[0.62685314]]
 alfons@alfons  …/HW1  master ● ? ⊛3
```