Logistic Regression and Softmax Regression to Separate Images of Different Car Types

Xinyuan Lu
xil069@ucsd.edu

Hanyang Xu
hax032@ucsd.edu

Abstract

We trained two of our networks using logistic regression and softmax regression to predict a class of a 200 * 300 pixel image using a combination of PCA to reduce the dimension and logistic regression and softmax regression to make the prediction. For the aligned set of data, we have achieved the following result. with PCA n_component = 10, using logistic regression, we have achieved an average accuracy of 60% and n_component = 40 using softmax regression cross validation we have achieved an average performance of 65%. For two class classification, anything better than 50% is better than random guessing and for four class classification, the average accuracy is more than 25% is better than random guessing. Thus, we conclude that our network successfully trained under PCA.

1 Data Processing, Cross Validation, and PCA

1.1 Introduction

In this section, we first load the four car types data from The Comprehensive Cars (CompCars) dataset. For each classifier we trained in this project, there are resized and aligned datasets. The resized set is simply re-sized from the original images. The aligned set has been preprocessed to align the cars. The four car types included in the dataset are: convertible, minivan, sedan, and pickup.

After loading the dataset, we implemented the k-fold cross-validation framework to estimate our model's performance on unseen data. k-fold cross-validation divides the dataset into training set, validation set, and test set. We trained our model on the training set, adjusted the model's parameters based on the validation set, and obtained the model's performance result on the testing set.

For this project, we performed dimensionality reduction on the data images by using a linear dimensionality reduction technique called Principal Components Analysis (PCA). The PCA is only performed on the training set and applied to the holdout and test sets without change.

1.2 Method

The PCA we use is given by the professor, Which will transform m n-dimensional vector dataset to a m k-dimensional vector dataset. For each of the four different sets of classes, the top principal component (PC) shows the differences between the images the dataset contains most clearly, as seen with the highly contrasting colors in the PC visualizations in the section below. The top principal component represents the dimension that captures the largest variance across the images in the dataset, so we are able to see the most data contrast from the top principal components. We also observe different behavior between the first two set principal components: In the first set of PCA, we have Minivan and Convertible and the second set of PCA we have all four different car types. In the second PCA visualization we believed we can see a bit more details compared to the first one. There is not much to say here since we only used the code from the professor and correctly applied it.

1.4 Results

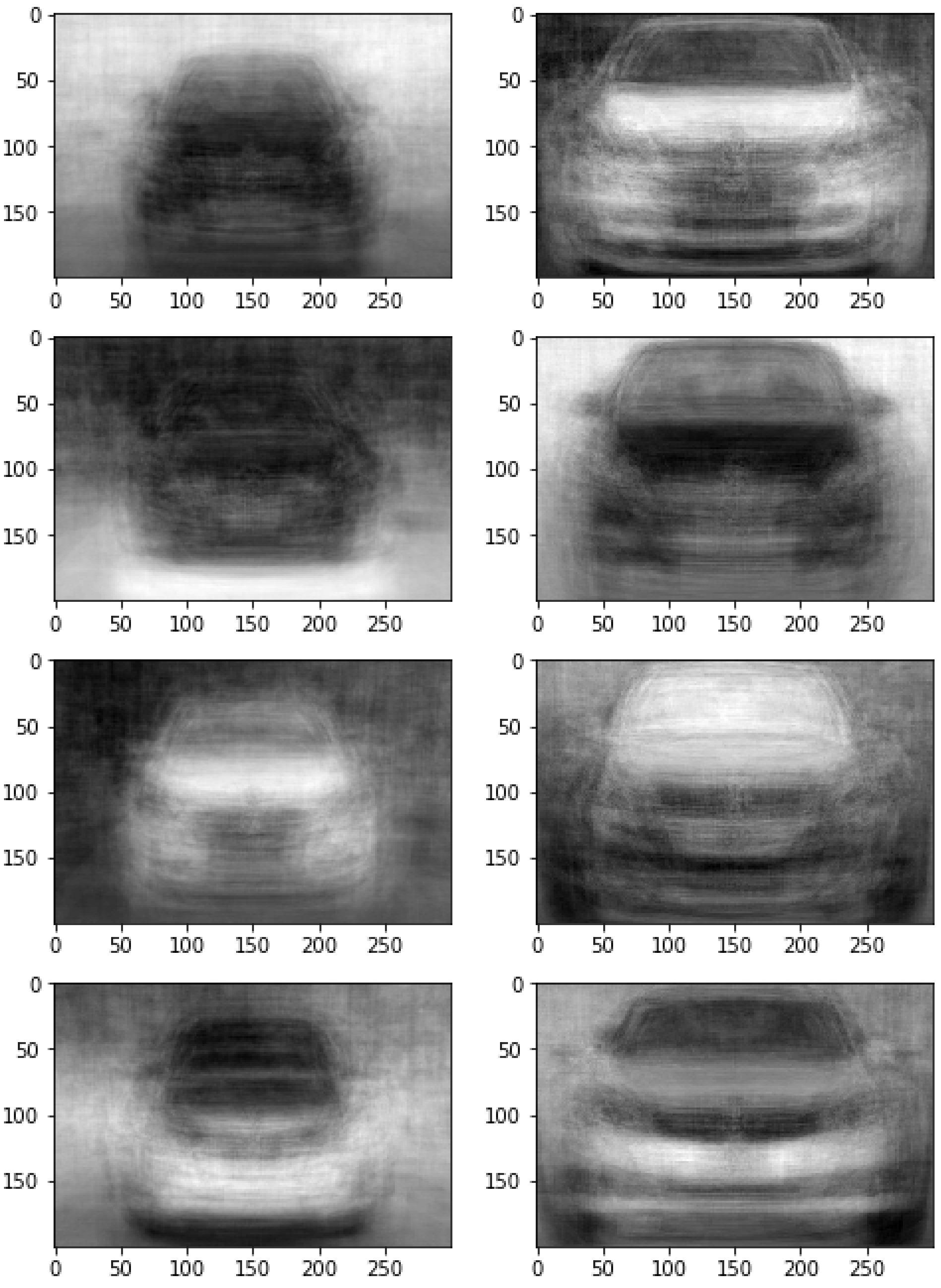


Figure.1 First four PCA Component on Resize Dataset(Left) vs Aligned Dataset (Right)

1.5 Discussion

The first four PCA components on resized data set and aligned dataset indicate the quality difference between these two datasets. The principal components of aligned data are clearer than the components of resized data. Such differences cause accuracy differences as we will see later.

2 Logistic regression

2.1 Introduction

In this section, we implemented Logistic Regression via Batch Gradient Descent. We first evaluated the model on Convertible vs Minivan using the resized dataset without cross-validation, we then evaluated our model on Convertible vs Minivan on the aligned dataset with 10-fold cross validation. Finally, we evaluated our model on Sedan vs Pickup on the aligned dataset.

2.2 Background

We implemented logistic regression for the binary classification task. We use the Binary Cross Entropy (BCE) as loss function:

$$E(w) = -\sum_{n=1}^{N}\{t^n \ln(y^n) + (1 - t^n)\ln(1 - y^n)\}$$

We used gradient descent with the gradient of the loss function:

$$-\frac{\partial E(w)}{\partial w_j} = \sum_{n=1}^{N}(t^n - y^n)x_j^n$$

2.3 Method

The logistic regression class consists of several important methods: updateweight(), predict(), calculateLoss(), and score(). For our implementation, we have all the function support vectorization to speed things up using np internal math library. The function itself is fairly simple. The prediction is made by summing up the element wise product and using a sigmoid function to map it onto the range (0, 1). The loss function will calculate the loss of a given feature set and label. The update function we use is a standard weight update function given by the professor in the lecture note. We have tried several different combinations of hyperparameters including n_component, learning rate and stop early method and mini batch size. But in the end, we decided to run batch updates simply because we think the for loop is very slow and using numpy built-in matrix vector subroutines is much more efficient. Our test accuracy for the resulting model was 0.65 with n_component = 10.. Through analyzing the output of the validation accuracy on each set of hyperparameters, we found that batch sizes and learning rates were not as important in the model performance, but an increase in principal components and always led to a better model performance, moreover using aligned data set is always better than using the resized data set . One thing worth mentioning is that logistic regression does not train well with n_componet = 4 for resized data sets. Which itself will not learn any correct information even with a huge number of epochs.

2.4 Results

Evaluate the model on Convertible vs Minivan using the resized dataset (Problem 5b)

PCA Component, Learning Rate	**Accuracy**
5, 0.02	**56%**

10, 0.02	**70%**
20, 0.02	**76%**

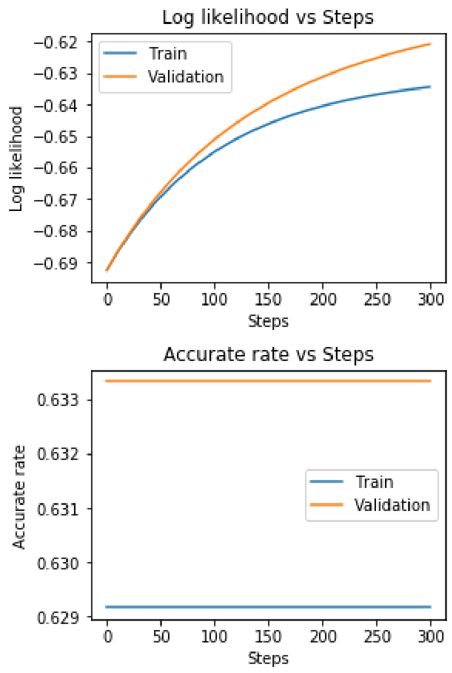


Figure 2. Loss and Performance Plots for 20 Component and 0.02 Learning Rate

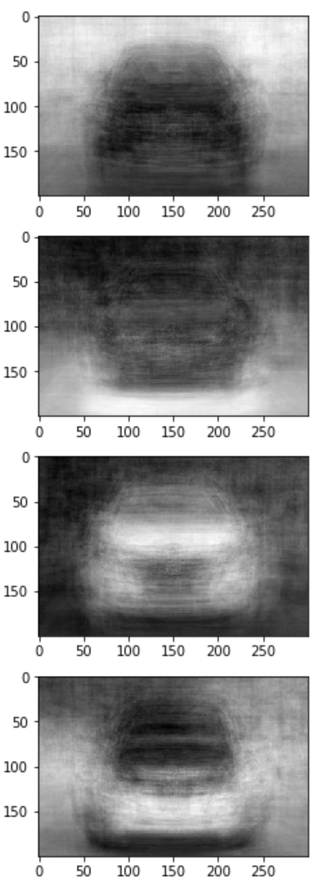


Figure 3. First Four PCA Components for the Best Model

Evaluate on Convertible vs Minivan on the aligned dataset

PCA Component, Learning Rate	**Accuracy**
5, 0.008	**64%**
10, 0.008	**77%**
20, 0.008	**78%**

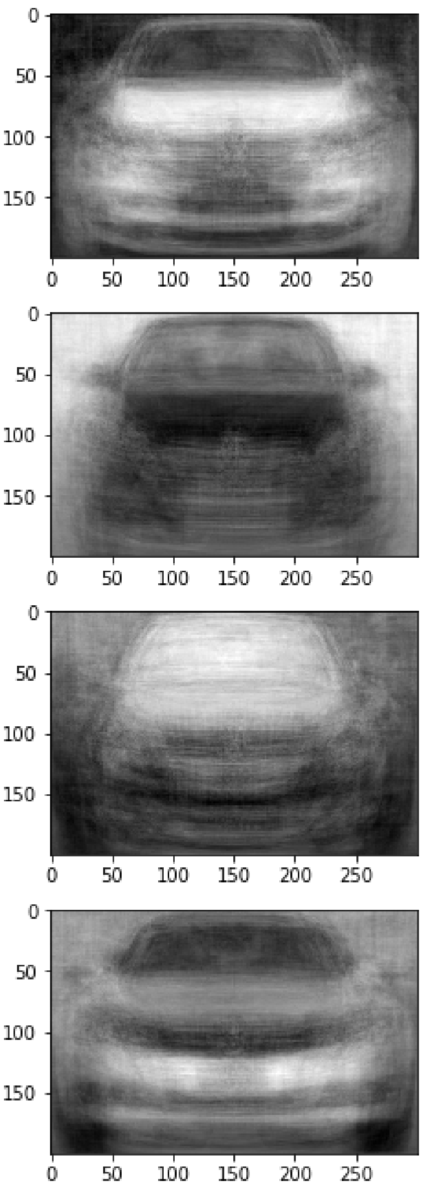


Figure 4. First Four PCA Components for the Best Model on Aligned Data

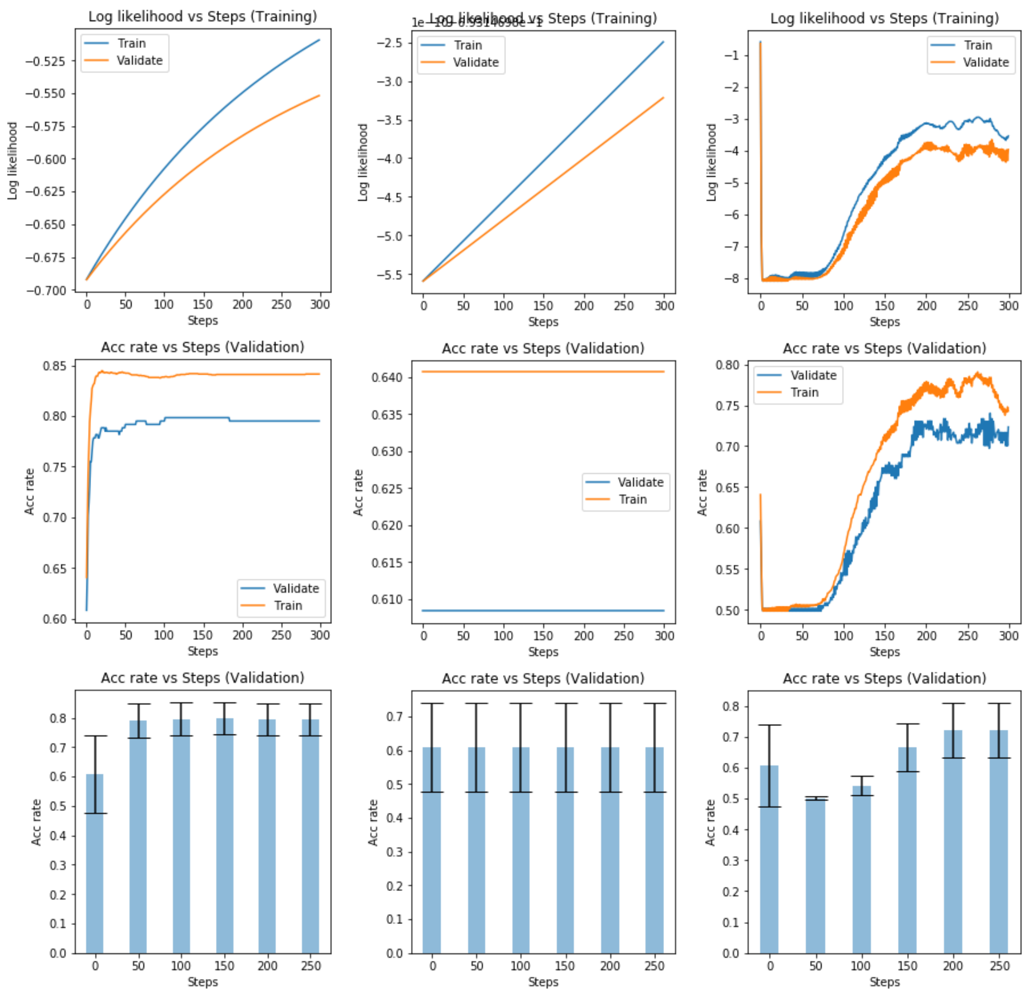


Figure 5. Loss and Performance Plots for 20 Component and three different Learning Rate: Left, just right(0.008); Middle, too small (0.000000000008); Right, too large(2)

Evaluate on Sedan vs Pickup on the Aligned Dataset

PCA Component, Learning Rate	Accuracy
5, 0.01	73%
10, 0.01	80%
20, 0.01	81%

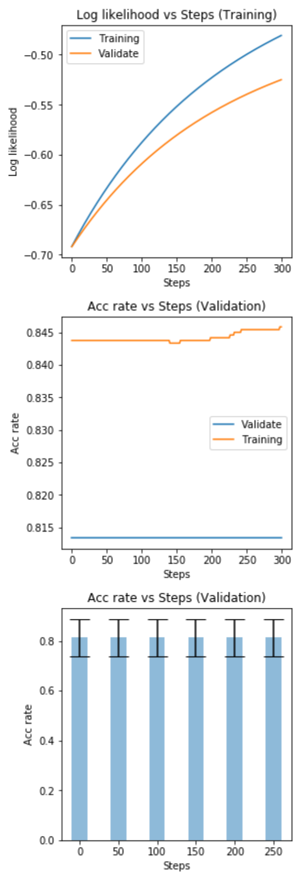


Figure 6. Loss and Performance Plots for the best Model

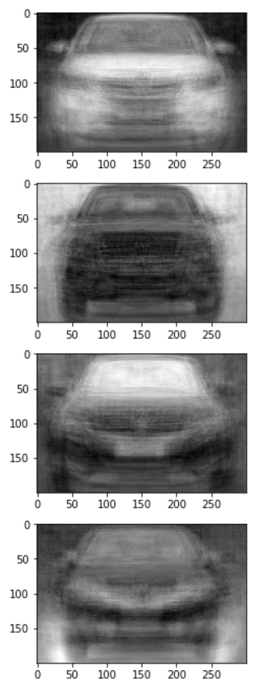


Figure 7. First Four PCA Components for the Best Model on Aligned Data

2.5 Discussion

Evaluate the model on Convertible vs Minivan using the resized dataset (Problem 5b)
We achieved 76% accuracy with PCA component = 20 and learning rate 0.02. The result is relatively poor since the PCA components as shown in figure 3 are very generic without much distinct features, therefore, the algorithm will not learn as much from these inputs compared to a better PCA result.

Evaluate on Convertible vs Minivan on the aligned dataset
We achieved 78% accuracy with PCA component = 20 and learning rate 0.008. The PCA image in Figure 4 is much clearer with more distinct features compared to the PCA image of resized data in Figure 1. Compare three different learning rates in Figure 5, we found that a too small learning rate will cause a loss curve that is almost a straight line which indicates the weight still has a lot of room to change. A too large learning rate will cause the weight to ostiliate during the gradient descent, and the both accuracy and loss ostiliate in the graph.

Evaluate on Sedan vs Pickup on the Aligned Dataset
We achieved 81% accuracy with PCA component = 20 and learning rate 0.01. The accuracy for Sedan vs Pickup is better than Convertible vs Minivan since we can visually the PCA components of Sedan vs Pickup are more distinctive with each other than the PCA components of Convertible vs Minivan.

3 Softmax logistic regression

3.1 Introduction

In this section, we implemented Softmax regression via Batch and Stochastic Gradient Descent. We first evaluated the model on all four car types using the aligned dataset, Batch Gradient Descent, and 10-fold cross validation. We then evaluated our model on all four car types using the aligned dataset, Stochastic Gradient Descent, and 10-fold cross validation. Finally, we visualized the weights of our model by reverse-PCA and scaling each weight to grayscale value.

3.2 Background

Softmax regression is the generalization of logistic regression for multiple classes. Given an input vector x of size n, n being the possible categories that x could belong to, the Softmax function will produce a vector y of size n, with each element k with value a being the probability of x belonging to the k category and the sum of y vector being 1.

$$y_k^n = \frac{exp(a_k^n)}{\sum_{k'} exp(a_{k'}^n)}$$
$$a_k^n = w_k^T x^n$$

We use the the Cross-Entropy Cost Function as loss function:

$$E = -\sum_n \sum_{k=1}^{c} t_k^n \ln y_k^n$$

We used gradient descent with the gradient of the loss function:

$$-\frac{\partial E^n(w)}{\partial w_{jk}} = (t_k^n - y_k^n)x_j^n$$

3.3 Method

The softmax regression class also consists of the same methods as logistic regression to achieve similar API calls: updateweight(), predict(), calculateLoss(), and score(). In our implementation for softmax regression however, due to the limited amount of time, we do not have all the function support vectorization to speed things up. These functions itself are fairly similar to that of logistic regression, with only differences with update rule and loss calculation. As the name suggested, softmax regression uses softmax function and the prediction is made by finding the largest probable classes among all the possible classes. We have tried several different combinations of hyperparameters including n_component, learning rate, stop early method, mini batch size as well as the descent methods. Through analyzing the output of the validation accuracy on each set of hyperparameters, we found that stochastic gradient descent must have a randomized input for different epochs in order for it to learn properly. However, after we randomized the input, both stochastic and batch gradient descent have roughly the same loss descent.

3.4 Results

The result for Softmax regression is as follows for Q6A with different PCA n_componenets, namely, n_component=40, 80, 120.

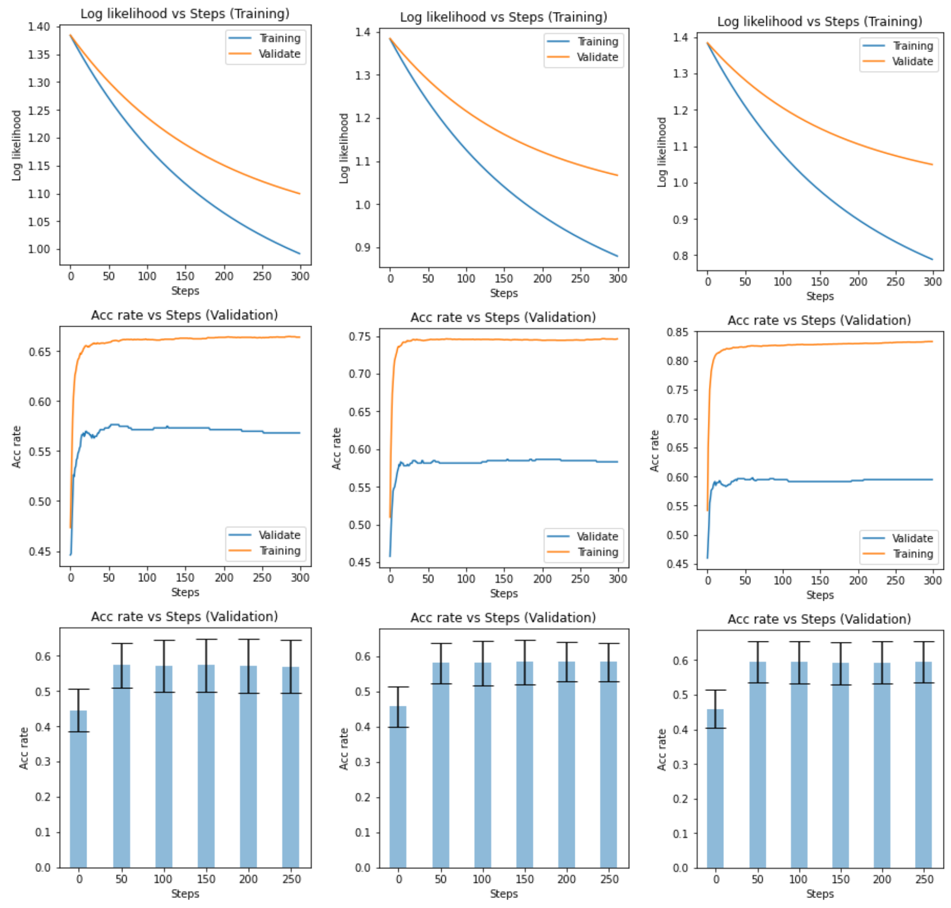
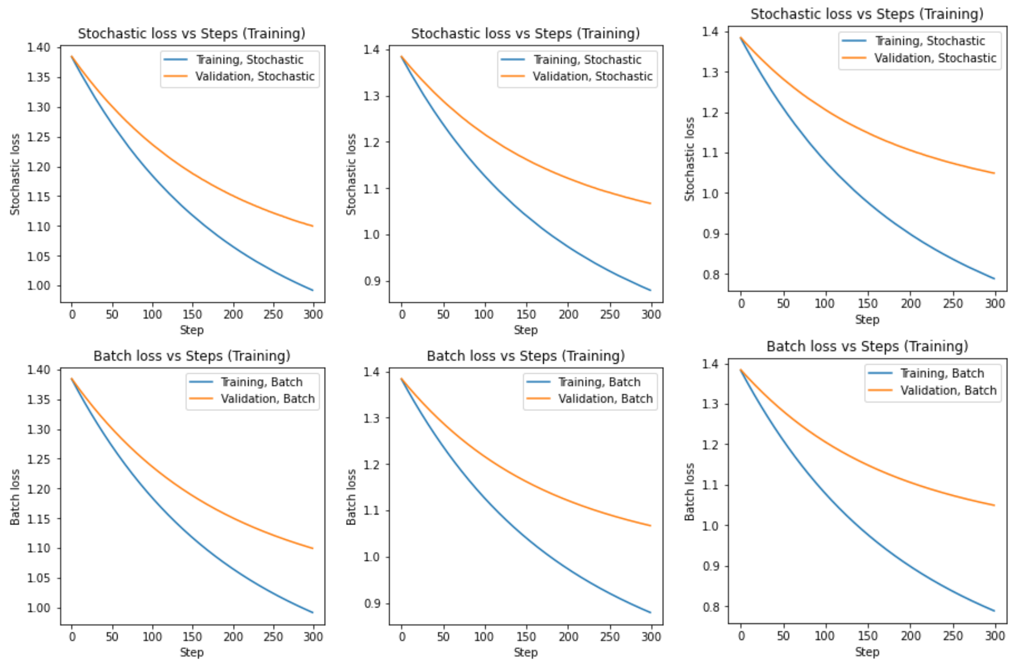


Figure: This table shows the result for softmax regression over 4 different types of vehicles using different PCA n_component. From left to right are 40, 80, 120, separately.

The confusion matrix is given below, as for n_component=120. The best model we have so far.

```
[[0.611   0.111    0.167   0.111]
 [0.086   0.391    0.260   0.260]
 [0.      0.142    0.857   0.     ]
 [0.166   0.25     0.      0.583]]
```

Experient result for Q6B.



3.5 Discussion

As we can see. when stochastic gradient descent is being applied correctly, we can easily achieve the same performance as batch gradient descent. And again, the more n_component the better.

4 Contribution

We employed the method of co-programming and completed this assignment entirely together via Zoom collaboration.