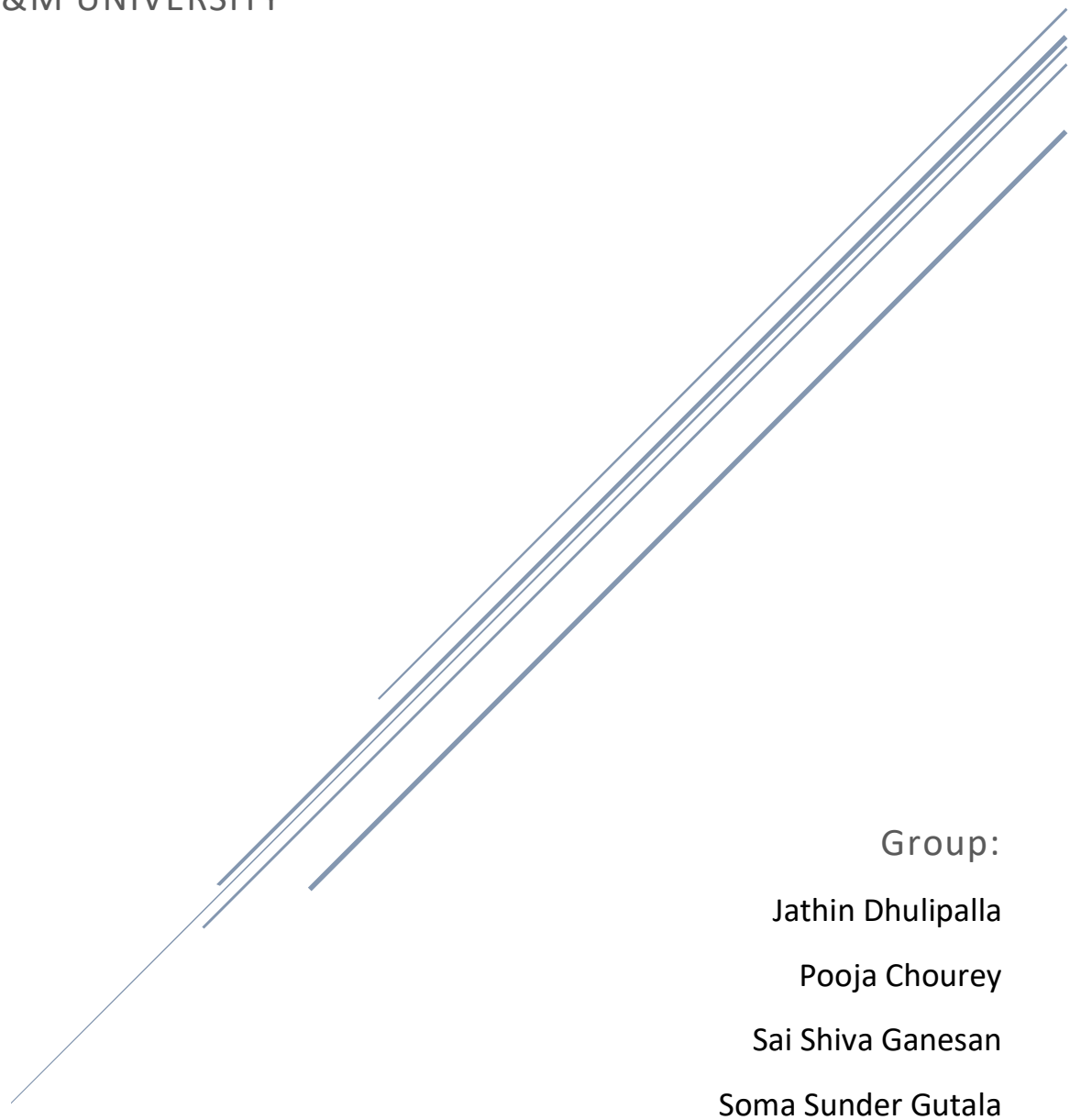
ISEN 613: Engineering Data Analysis

Fall 2019 – Course Project

Image Classification on CIFAR 10 Dataset

TEXAS A&M UNIVERSITY



Group:

Jathin Dhulipalla

Pooja Chourey

Sai Shiva Ganesan

Soma Sunder Gutala

Contents

1. Executive Summary

The CIFAR-10 dataset consists of 50,000 32x32x3 images labeled into 10 classes. Each class of images corresponds to a physical object. The aim of this project is to develop a machine learning model to classify the object in each image provided in the data set. Each picture is stored as one observation in the dataset. Data preprocessing was an essential step to reduce the computation times. Hence, we ended up having a handful of features (220) to train our model with compared to the original 3072 features. These features were then used to train various statistical models such as linear discriminant analysis, quadratic discriminant analysis, bagging, boosting, random forests and support vector machines.

To assess each model, validation approach was primarily used with a split of 80-20 resulting in training and test data with 40,000 and 10,000 observations respectively. Ultimately, QDA was chosen as the best model and the entire dataset was used to train the final QDA model.

The test data was run using the QDA model that was trained finally. One of the findings was that the label 9 (ship) was classified with the most accuracy of being predicted correctly 7 out of 10 times. The second highest accuracy was achieved on label 2 (automobile) with almost 6 out of 10 predictions being correct. Interestingly, both these were man made objects. Labels 3 (bird) and 4 (cat) were the labels that were predicted with least accuracy with just a little over 3 out of 10 predictions being correct. In case of 4 (cat) for instance, it was classified as label 6 (dog) almost 2 out of 10 times and label 5 (deer) once every 10 times. Surprisingly, label 3 (bird) was classified as label 5(deer) over 2 times for every 10 predictions perhaps due to backgrounds in which they tend to be.

The contribution from each team member has been summarized below:

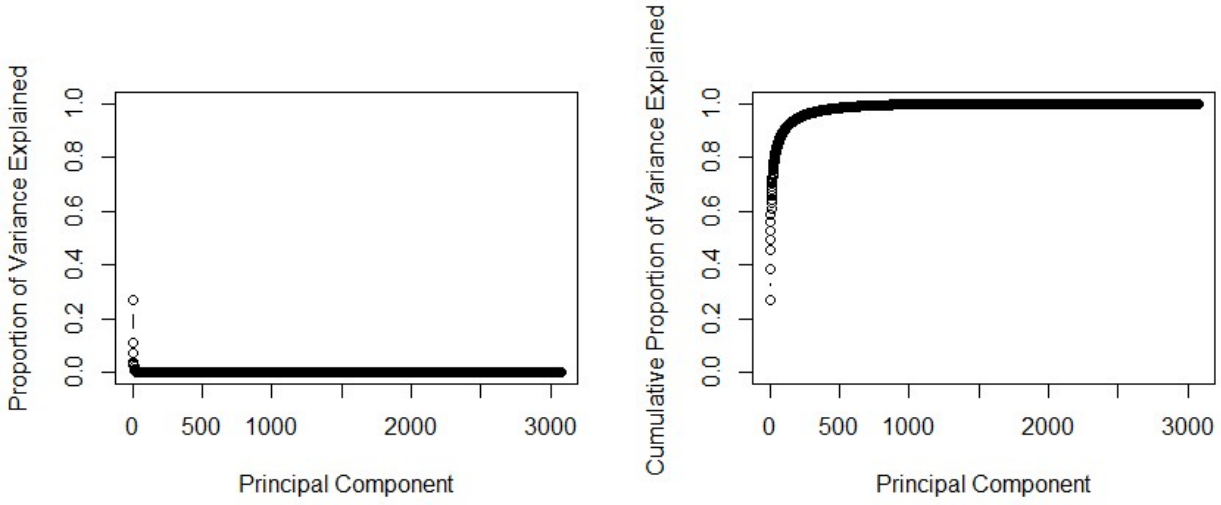
Team Member Name	**Tasks Performed**
Pooja Chourey	Executive Summary. Training, validation and technical report of QDA
Jathin Dhulipalla	Executive Summary, Training, validation and technical report of Boosting model. Improvement upon best method.
Sai Shiva Ganesan	Executive Summary. Training, validation and technical report of Linear SVM.
Soma Sunder Gutala	Executive Summary, Principal Component Analysis

2. Technical Report: Linear kernel SVM (SVC)

Data Preprocessing:

The aim of building this model was to predict the object in each image provided in the data set. We have been given a training data set of 50,000 images with RGB pixel values which directly correspond to 3072 (1024 X 3) features. Since the computation time required to train a model across all the given features is extremely long, we decided to use the **Principal Component Analysis** dimensionality reduction technique with scaling set to 'TRUE' to work with lesser number of features.

In order to choose the number of principal components, we fixed a threshold variance of 95%. To explain 95% of variance in the data, we ended up choosing the first 220 principal components. The same has been illustrated by the scree plot and cumulative PVE (proportion of variance explained) plot.



With 220 features and respective training labels available for each observation, a support vector machine with a linear kernel function (support vector classifier) was fit to 80% of shuffled data. An interesting point to note here is that the linear kernel SVM seemed to perform much better than other nonlinear kernel functions like radial and polynomial SVMs. Training data corresponded to 40,000 observations in the given dataset. We are using the validation set approach to assess the model performance. 20% of data is used as a holdout set that the model will make predictions on.

Training the model

Since SVC is a soft margin classifier and margin violations are kept possible to provide robustness to the model, the cost parameter had to be set while training the model. The cost

value acts like a budget for the magnitude of violations that will be acceptable by the model. We decided to go with a cost parameter value of 0.1. This particular cost value yielded better results compared to other values in our testing. A seed value is set so that the model results are reproducible.

```
> set.seed(10)
> svmlin.fit = svm(as.factor(train_Y)~., data=train_PCA.df, kernel='linear',
cost=0.1)
> summary(svmlin.fit)
```

Taking a look at the summary of our fit linear SVM model, we see a total of 35,583 support vectors from all 10 classes. We now run the fit model on our training data to check the training accuracy.

```
> svmlin.train.predict = predict(svmlin.fit, train_PCA.df)
```

	Target									
Predicted	1	2	3	4	5	6	7	8	9	10
1	2086	237	341	184	219	157	62	174	642	295
2	207	2070	161	186	107	123	121	143	229	594
3	194	132	1339	359	579	406	261	303	114	96
4	130	104	304	1247	258	632	466	216	95	100
5	102	78	398	227	1324	288	361	340	62	69
6	109	123	332	666	313	1441	297	273	127	85
7	114	192	571	565	577	443	2121	210	60	174
8	198	166	317	186	420	268	142	1954	49	175
9	577	221	162	168	99	157	60	98	2254	251
10	309	673	92	221	112	114	95	259	354	2134

From the table, we can see that we get a training accuracy of **44.92%** from linear SVM. The model has low variance but a relatively higher bias.

Validating the model

Now running the fit model on validation set:

```
> svmlin.val.predict = predict(svmlin.fit, test_PCA.df)
```

	Target									
Predicted	1	2	3	4	5	6	7	8	9	10
1	466	62	92	46	56	34	22	62	176	79
2	45	486	43	47	23	48	29	37	56	176
3	55	30	304	90	157	115	76	88	29	36
4	43	32	85	279	74	185	108	52	25	25
5	19	18	100	62	279	58	92	103	17	25
6	24	41	82	176	63	306	90	93	30	14
7	32	45	136	146	165	117	520	62	20	43
8	58	43	79	38	114	46	35	445	14	49
9	170	62	34	55	21	40	16	27	552	77
10	62	185	28	52	40	22	26	61	95	503

We end up with a validation set accuracy of **41.40%** from our model.

3. Technical Report: Quadratic Discriminant Analysis (QDA)

Data Preprocessing:

For Quadratic Discriminant Analysis, data preprocessing was done in a similar way as it was done for the linear SVM model. Since the number of features in our dataset is 3072, we performed principal component analysis to identify the features that explain the maximum proportion of variance in data. We found that 95% of the variance in data could be explained by 220 features. These features were used to train our model using quadratic discriminant analysis and then further to validate the model. Following the validation set approach, the data was divided into an 80-20 ratio to train and validate, respectively. There was a total of 40,000 rows in our training data and 10,000 rows in the validation data. Since the labels are readily available for this data, it was convenient to check the prediction accuracy of the model on our validation set.

We preferred QDA for this dataset because the training data is large and as compared to some other models that we tried, QDA seemed to perform well in terms of computational speed as well as accuracy.

Training the model

After performing the principal component analysis and split on data, the QDA model was trained. There are no hyper parameters to tune when it comes to training a QDA classifier. It is important to note that the QDA classifier assumes observations from each class take the form of Gaussian distribution and each class has its own co-variance matrix. Hence the resulting classifier boundary is a much more flexible quadratic decision boundary compared to an LDA classifier. A seed value is also being set before training so that the results from our model are reproducible.

```
> set.seed(10)
> qda.fit = qda(as.factor(train_Y)~., data=train_PCA.df)
```

The training accuracy of the model was obtained as:

```
> qda.train.pred = predict(qda.fit, train_PCA.df)
```

	Target									
Predicted	1	2	3	4	5	6	7	8	9	10
1	2413	158	220	92	113	47	60	100	110	145
2	144	3211	93	70	76	37	74	34	185	178
3	164	24	1986	310	110	377	198	121	76	39
4	84	57	168	2402	180	216	124	148	97	93
5	225	28	804	432	2945	391	468	427	117	49
6	84	35	251	234	122	2603	89	225	43	37
7	24	54	134	179	74	110	2799	32	40	27
8	87	34	105	76	213	111	28	2757	30	51
9	612	262	133	116	92	41	72	46	3105	280
10	189	133	123	98	83	96	74	80	183	3074

We got a training accuracy of **68.19%** from QDA.

Validating the model

After performing fit on the training data, the model was run on the other 20% data - validation set. The validation set accuracy that we got was **50.69%**

```
> qda.val.pred = predict(qda.fit, newdata=test_PCA.df)
```

	Target									
Predicted	1	2	3	4	5	6	7	8	9	10
1	435	37	74	31	44	7	24	23	67	59
2	52	651	24	32	25	16	49	12	90	147
3	50	3	307	104	64	120	69	28	24	7
4	29	20	59	318	67	155	75	56	23	41
5	66	10	245	118	572	94	152	145	29	15
6	26	13	107	176	43	417	51	97	20	12
7	6	16	66	71	36	43	512	13	11	15
8	29	18	30	37	73	60	11	568	10	27
9	217	81	24	28	36	19	21	13	665	80
10	64	155	47	76	32	40	50	75	75	624

These results were significantly better than the results we achieved with an LDA classifier. Hence this turned out to be one of our top 3 models on the CIFAR 10 dataset.

4. Technical Report: Boosting

Data Preprocessing:

For Boosting Analysis, data preprocessing has been done using Principal Component Analysis. This is similar to what has been done for the above two methods. The total number of features in the data is 3072 which causes problems for computing the models and associated predictions. It was found that 95% of the variance in data could be explained by 220 features which were used to train the model. The complete data set has been split into training data that contains 80% of observations and a test set of data containing 20% of the observations. Using the given labels, test error has been used to compare this model with the other models.

After using random forests, boosting and bagging since they fall under the same family of methods, we found that boosting had the most potential.

Training the model

Boosting model has been trained after Principal Component Analysis of the data as described in the previous section. Boosting has three tuning parameters that makes this assessment one of the most time consuming. These parameters are the number of trees, shrinkage parameter and the depth of trees grown. For the depth, we take 4 as the value since shorter trees have better results as boosting grows the trees based on the information from previous trees. For the number of trees, we take a value of 2000. We consider the tuning of overall model mainly by focusing on the tuning parameter. Since the number of trees is not being changed, the shrinkage parameter is picked by evaluating test errors for the values of 0.001, 0.005, 0.01, 0.05, 0.1 and 0.5. We found that 0.005 gives a good amount of accuracy compared to other shrinkage parameters.

After this, we finally train the model with the chosen parameters using a multinomial distribution since we require classification on more than two classes. We then calculate the training and test errors.

```
> boost.fit
gbm(formula = as.factor(train_Y) ~ ., distribution = "multinomial",
    data = train_PCA.df, n.trees = 4000, interaction.depth = 4,
    shrinkage = 0.005)
A gradient boosted model with multinomial loss function.
4000 iterations were performed.
There were 220 predictors of which 220 had non-zero influence.
```

The accuracy on training set has been found to be **60.0825%.**

In both instances of predicting on training and validation sets, we have used the label that came out with the maximum probability.

```
> boost.train.predprob = predict(boost.fit,newdata=train_PCA.df,type="respons
e", n.trees=4000)
> boost.train.pred = apply(boost.train.predprob, 1, which.max)
```

	Target									
Predicted	1	2	3	4	5	6	7	8	9	10
1	2485	155	242	120	188	83	41	114	339	163
2	172	2716	136	141	67	104	102	110	222	428
3	199	44	2020	273	423	288	230	170	84	45
4	100	85	248	1910	204	555	240	227	81	74
5	107	44	422	198	2136	256	276	294	64	38
6	61	72	158	484	160	1986	143	174	77	83
7	91	110	375	427	402	318	2755	141	42	98
8	121	93	187	168	225	194	79	2452	33	138
9	485	207	132	122	97	115	50	84	2862	195
10	205	470	97	166	106	130	70	204	182	2711

Validating the model

After performing fit on the training data, the model was run on the other 20% data - validation set. The validation set accuracy that we got was **45.27%.** Also, note that the predictions were bases on which label had he highest probability.

```
> boost.val.predprob = predict(boost.fit,newdata = test_PCA.df,type="response
",n.trees=2000)
> boost.val.pred = apply(boost.val.predprob, 1, which.max)
```

	Target									
Predicted	1	2	3	4	5	6	7	8	9	10
1	506	50	91	39	56	19	20	32	120	70
2	57	538	42	53	21	34	24	42	69	183
3	50	12	298	73	140	100	77	71	25	17
4	36	31	86	320	46	187	91	72	25	26
5	25	9	150	53	393	53	123	122	17	18
6	19	37	62	171	53	330	48	92	23	30
7	33	31	115	119	136	106	555	50	12	25
8	31	32	54	59	84	66	29	443	16	61
9	174	68	37	43	27	35	13	27	632	85
10	43	196	48	61	36	41	34	79	75	512

The training and more importantly the validation error turned out to be one of the best on the CIFAR 10 dataset using boosting.

5. Choosing the Best Model: QDA

After considerable analysis as outlined in the previous sections, the following table has been compiled comprising of the training and test accuracies for each of the models. Note that these accuracies are achieved after tuning the models.

Model	Training Set Accuracy	Validation Set Accuracy
Support Vector Machine (SVM) - Linear Kernel	44.9200%	41.4000%
Quadratic Discriminant Analysis (QDA)	68.1900%	50.6900%
Boosting	60.0825%	45.2700%

The primary methodology that has been used in selecting and training the best model is validation set approach. We have calculated both test and training accuracies for multiple models and the above three have been found to be performing the best for the given dataset. We have used Principal Component Analysis (PCA) for preprocessing before each of the models since the number of features in the given dataset is overwhelming to perform the computations.

Finally, after deciding the required model, we have used PCA once again to train the model on the entire dataset as this would give more data for the model to be trained. Since we would be using this model on data that is completely new, there is no concern that the data is going to cause an underestimate of errors on the new data.

We cannot do much than speculate on how QDA turned out to be the best models among all the classification models we have tried out like Linear Discriminant Analysis, KNN, Random Forests among others. We found that the least test error was obtained for boosting, linear SVM and Quadratic Discriminant Analysis. However, it is important to note that boosting is extremely powerful and showed great potential. If we could have more computations, perhaps we could have achieved higher accuracy. This observation is being made since we noticed a trend of increasing accuracy while increasing the number of trees and reducing the shrinkage parameter. Boosting as a method learns slowly to make better predictions. Based on our literature review, we found that image classification works best when using machine learning models that learn slowly like neural networks. Boosting could come close to perhaps mimic such a learning pattern and shows promise if provided enough resources.

For now, based on the given conditions, data and constraints, we have found Quadratic Discriminant Analysis to be the model to go forward with.

6. Evaluating Test Results

The test set provided consists of 10000 observations (images). This section of the project analyzes the performance of the best model selected previously (QDA) on the unseen test data.

Data Preprocessing

The test set which was provided in a bin format was extracted and stored as data frames for prediction by our model. The PCA function which was fit on our full training set was used to extract the first 220 principal components (95% variance) for our test set. The new data frame with 10000 observations and 220 features were used to make predictions.

Testing the best model (QDA):

The QDA model was trained on the entire training data set before it was used to make predictions on the unseen test set.

```
> qda_full.fit = qda(as.factor(full_y)~., data=PCA_full.df)

> qda.test.pred = predict(qda_full.fit, newdata=testset_PCA.df)
> qda.test.table = table(Predicted=qda.test.pred$class, Target=testset_Y)
```

Comparing our model predictions with given target labels for test set, we end up with the following table.

	Target									
Predicted	1	2	3	4	5	6	7	8	9	10
1	473	50	72	40	42	17	25	30	67	41
2	37	637	15	27	23	13	34	19	74	137
3	48	6	327	99	63	112	76	40	28	12
4	40	22	72	328	58	151	70	50	20	46
5	78	6	223	112	579	103	138	108	23	14
6	18	10	126	180	61	461	47	94	6	17
7	10	23	50	71	41	47	530	9	11	10
8	21	13	37	47	79	56	14	572	15	33
9	211	89	41	25	27	8	26	13	703	95
10	64	144	37	71	27	32	40	65	53	595

We get a test set accuracy of **52.05%**.

This translates to a test set error of **0.4795.**

7. Potential Improvements (Bonus)

The approach we have taken in our image classification effort has been mostly inspired from CNN (Convolutional Neural Networks). The main idea is that instead of focusing on the entire image (the features in this case), most of the images can be classified with a smaller frame within the image. By preprocessing our data using Principal Component Analysis, we were trying to analogously find the important features of the images. However, using neural networks and the corresponding layers could have greatly improved the accuracy.

However, another approach that can increase improve the model training is to increase the data from existing set of data. This process is called data augmentation where we use the same image and randomly crop, resize and rotate it so that the main characteristics of a certain class can be extracted. With proper tools and resources, we would have liked to perform this step on top of what we did to increase the data available for training our model.

Even though we have selected Quadratic Discriminant Analysis as our star model, we would have perhaps found more success on Boosting algorithm as there were many approaches that could have been used to fine tune the parameters we used. QDA somewhat is limiting in the sense that it has very few parameters that could have been changed. Particularly the growth parameter and number of trees turned out to be quite sensitive in our initial explorations while choosing the models. Also, multinomial distribution was used to get probabilities of variables while we were evaluating boosting. Other distributions were not explored as multi-level classification beyond 2 classes was not as available in literature as others.