Report HW1

1. **K Nearest Neighbor:** K nearest Neighbor is a non-parametric classifier, given k, number of neighbors, it tries to find out closest k neighbor points n_o from the given test point, using distance function. Frequently used distance functions are, Euclidean Distance and Manhattan Distance.

1. Euclidean Distance:

$$d(x_j, x_k) = \sqrt{\sum_i (x_{j,i} - x_{k,i})^2}$$

2. Manhattan Distance:

$$d(x_j, x_k) = \sum_i |x_{j,i} - x_{k,i}|$$

Later KNN uses Bayes Rule and determine the classification of test point. Using formula:

$$p(y = j | x = x_o) = \frac{1}{k} \sum_{i \in n_o} I(y_i = j)$$

KNN takes neighbors(k) as hyper parameter, thus K needs to be set for this classification. This classifier will work good, as it has low bias so it performs better when there is large amount of data set.

Logistic Regression: Logistic Regression is a Probabilistic Discriminant Classifier, which classifies the data set with categorical Output. The probability function is defined as:

$$p(X) = \frac{e^{\beta_o + \beta_1 x}}{1 + e^{\beta_o + \beta_1 x}}$$

It uses sigmoid function so that probability always lie in [0,1]. Regression Coefficients β_o & β_1 are usually determined by Maximum Likelihood function.
Here, Hyper-Parameter is C, inverse of Regularization constant. As, Logistic Regression have complex Probabilistic Classifier, it is said to have higher capacity. Higher capacity can be controlled using Regularization Technique, as shown below.

$\theta_* = \arg max_\theta = C \sum_{i=1}^{n} \log P(Y = y_i | X = x_i) - ||w||^2$ Here, C=$\frac{1}{\lambda}$, Inverse of Regularization Strength.

Logistic Regression can also be used for multiple categories using:

$$log \frac{p(x)}{1 - p(x)} = \beta_o + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

Logistic Regression is based on certain assumption that the conditional distribution is a Bernoulli distribution as the dependent variable is binary. It has Linear Decision Boundary in case of Binary Classification and piece wise linear boundary in case of Multiple Classification. This Classifier is one of best for Binary Output Values just like in Semiconductor and Email Spam Dataset.

Support Vector Machines: Support Vector Machine is a Discriminant classifier thus will work best for Email Spam and Semiconductor Dataset. SVM segregate data points using Hyper-Planes, it can be written as:

$y_i(w^T X + b) > 0$ where, $y_i \epsilon \{-1, 1\}$, i=1,2,3…n

Here, $w^T X + b = 0$ is a Hyperplane. To select a maximal Margin classifier, we need to determine the perpendicular distance of the closest points to the planes, known as Margin(M). Plane which has largest margin gets selected. $Maximize_{w^T} M$

$$y_i(w^T X + b) \geq M \quad \text{i=1,2,3…n}$$

In many scenarios, separating Hyperplanes doesn't exist, which leads to misclassification; to avoid it, slack variables are introduced along with Margin in equation.

$y_i(w^T X + b) \geq M(1 - \varepsilon_i)$ and $C \geq \sum_{i=1}^{n} \varepsilon_i$

C is known as non-negative Tuning Parameter, it determines the severity of violations that will be allowed. It is used as Hyper Parameter for SVM. If C is small, there will be narrow margins which gets rarely violated. It results in high fit classifier which has low bias and high variance. If C is larger, there will be wider margins, many support vectors. It will result in more bias and low variance.

Kernels: It is a method to change the decision boundaries from Linear form to High dimensional form, by enlarging feature space. $k(x_i, x_i') = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d$

SVM function with Kernel Implementation can be written as, $f(x) = \beta_o + \sum_{i \epsilon S} \alpha_i k(x, x_i)$

---

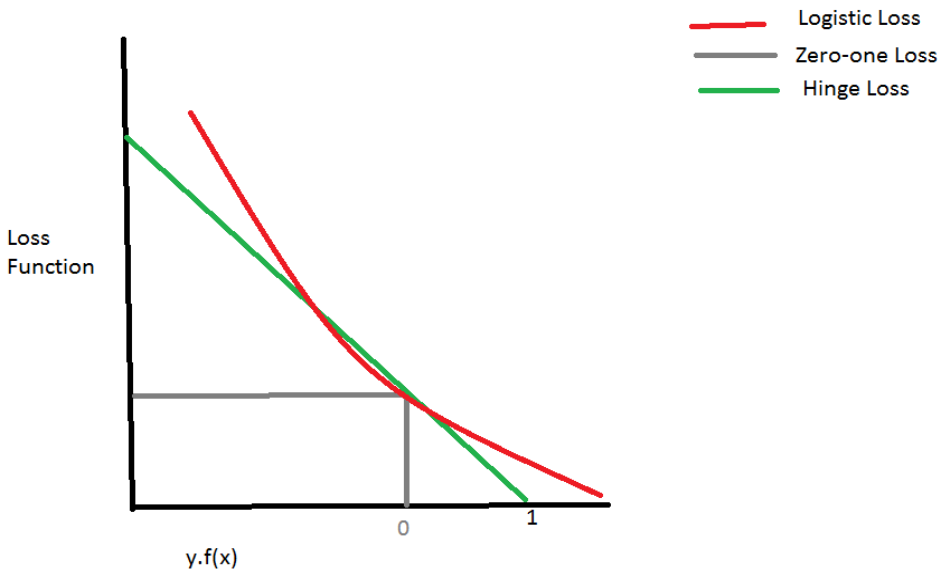
2. Support Vector Machines and Logistic Regression are related as both are Discriminative classifier. Their Loss Function, Hinge Loss and Logit Loss have almost same convergence. Although in Support Vectors Case Hinge Loss Function can be null if Observation is on correct side, but in logistic Regression Loss Function is not exactly zero. Due to this similarity, they almost have same results on a data set.

Errors are as follows:

Logistic Loss: $C \sum_{i=1}^{n} \log P(Y = y_i | X = x_i) - ||w||^2$, C *is inverse of Regularization parameter*

Hinge Loss: $C \sum_{i=1}^{n} \max[0, 1 - y_i(\beta_o + \beta_1 x_{i1} + \cdots + \beta_p x_{ip})] + ||w||^2$, C is tuning parameter

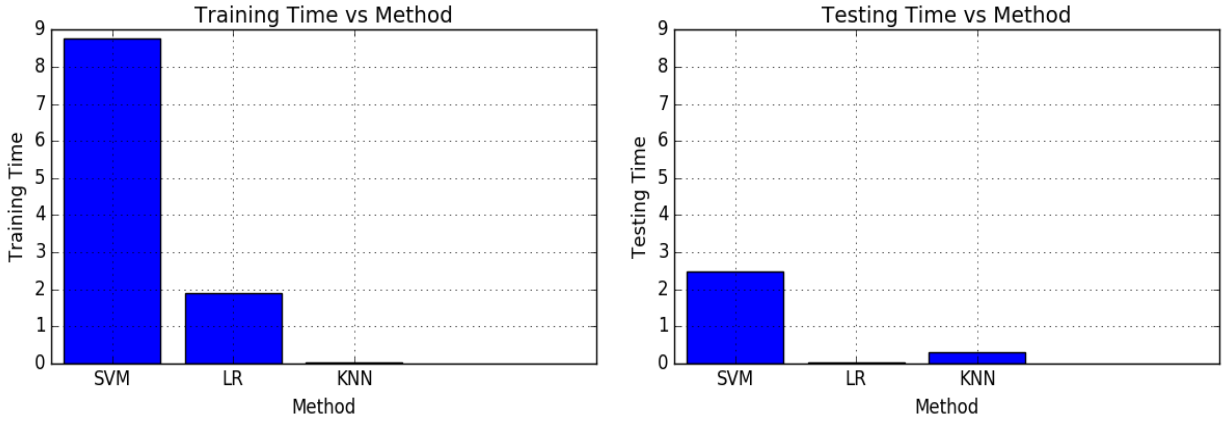
Their Convergence along with 0-1 optimization can be visualized as:



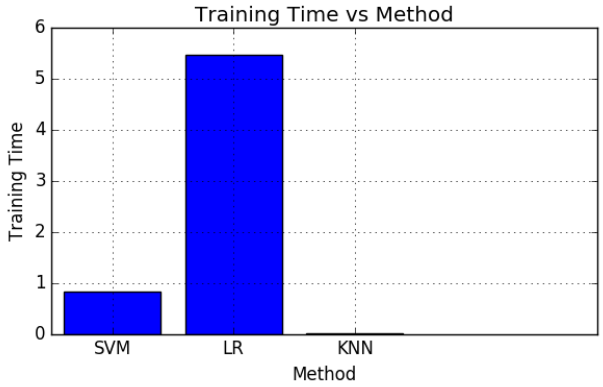
Features	K Nearest Neighbors	Logistic Regression	Support Vector Machine
Decision Boundary	Arbitrary, but as K increases it becomes almost Linear.	Binary: Linear Multiple Categories: Piece-Wise Linear	Without Kernels: Linear With Kernels: As per Polynomial Dimension
Speed/Time	It takes huge amount of time to store and calculate distance.	Fitting a logistic Model is slower, as it is fit by an iterative process using Maximum Likelihood.	Takes somewhat equal time as Logistic Regression for Linear SVM, but for Polynomial it takes more time to determine the coefficients.
Storage	we need to store all data, and calculate distance for all.	It considers all data points, thus require more space.	Needs less space to store the classifier as It depends on support vectors only.
Flexibility	As K grows, this method becomes less flexible.	Logistic regression considers all the points in the data set, thus it gets affected with any change in data set.	Classifier only depends upon support vectors, so it is flexible even if data increases.
Variance	High	High	Depends upon Tuning Parameter. If C is small, high variance. If C is large, less variance.
Capacity	High	High	High
Bias	Low Bias	Low Bias	Depends upon Tuning Parameter. If C is small, it will have low bias. If C is large, it will have more bias.

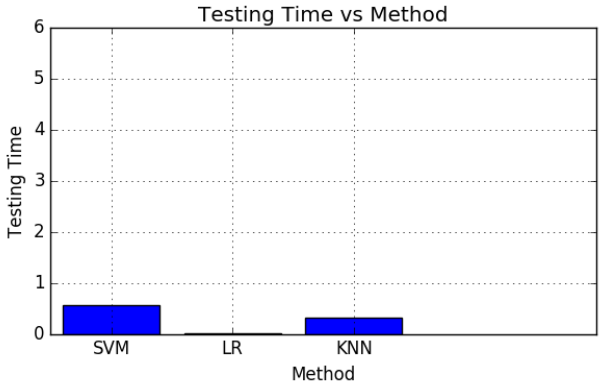
3.a. Training time Vs Testing time

For Digits Data set

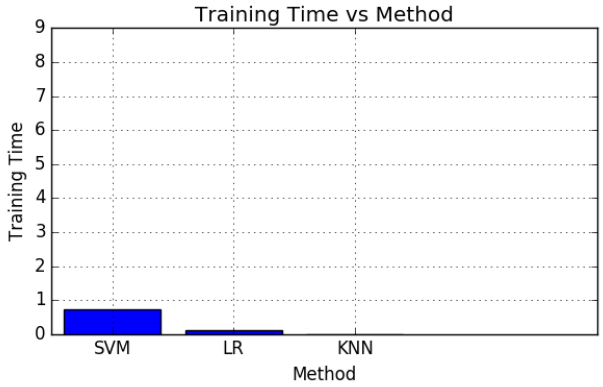


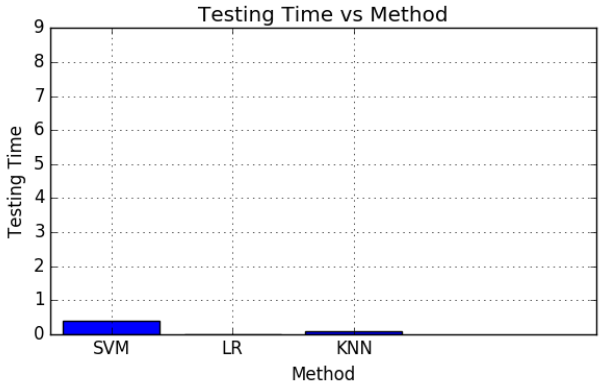
For Semiconductor Dataset



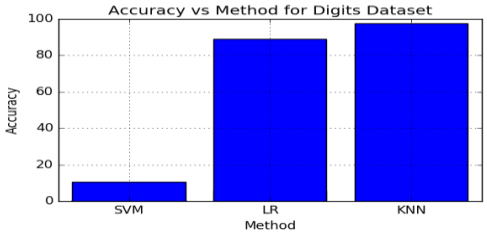


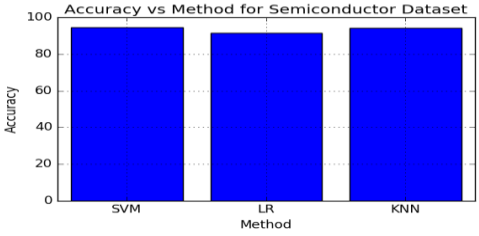
For Email Spam Dataset

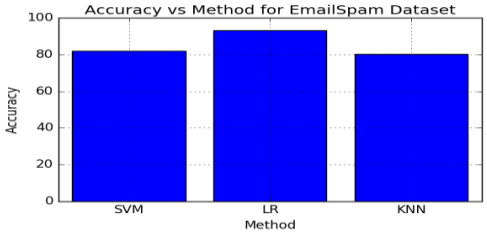




Accuracy:







3.b. K Nearest Neighbor turns out to be the best both in terms of Speed and Accuracy. Although we can see that for Email Dataset its Accuracy is around 80%, whereas Support Vector Machine (81.98%) and Logistic Regression (93.37%) have better Accuracy. Similarly, in case of Semiconductor data set, K nearest neighbor have Accuracy of 94.45%, but SVM has 94.80%.

As per current scenario, KNN would be more apt in terms of best speed-accuracy, as it takes much less time to train and test data, also its accuracy is almost close to that of Support Vector Machine and Logistic Regression in Semiconductor and Email Spam Data Set.

4.a. K fold Cross Validation: Cross Validation is a method to measure the accuracy of classifier, via changing the Hyperparameter values. It is used to optimize the Hyperparameter Values. For given data set, K fold Cross Validation approach would be best, its steps are as follows:

1. Divide the given data set in to K parts.
2. For i=1,2,3…, k:
   Choose part I as Cross Validation set, and rest as Training Set.
   Train the data.
   Compute the mean square error between training and cross validation set.
   Return average of the errors for given hyperparameter value.

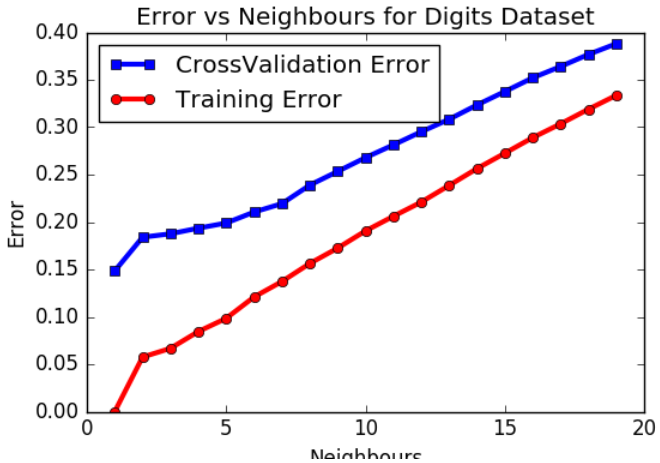
4.b. K Fold Cross Validation is one of the best approach as it considers whole dataset for cross validation as well. K Fold is better because it has low variance, as compared to other available choices such as 3 fold, 2 fold. On, the other hand Cross Validation- Cross Validation might perform better than K Fold but there will be a cost of time.

4.c. Code Cross Validation

```
def kFold(features_train,labels_train,features_test,labels_test,path,DataSet):
n=features_train.shape
k=10 # make it 10 fold test
size = n[0]/k # size of each fold.
errors1={} # to save CrossValidation Errors
e1=[] # to obtain mean square error of whole CrossValidation set
for p in range(1,20): # considering neighbours from range 1 to 20.
print "Considering Neighbor="+str(p)
for i in range(1,k):
# Select the cross Validation set, it will change with changing values of i
Feature_CrossVal = features_train[i*size:][:size]
Label_CrossVal = labels_train[i*size:][:size]
# Add the rest of Training set
Feature_Train = features_train[:i*size]
np.append(Feature_Train ,features_train[(i+1)*size:])
Label_Train = labels_train[:i*size]
np.append(Label_Train ,labels_train[(i+1)*size:])
# K Nearest Neighbor training with testset and testing with Cross Validation set
neigh = KNeighborsClassifier(n_neighbors=p)
neigh.fit(Feature_Train,Label_Train)
predict=np.zeros(Label_CrossVal.shape)
predict=neigh.predict(Feature_CrossVal)
e1.append(mean_squared_error(Label_CrossVal,predict))
errors1[p]=np.mean(e1)
neigh = KNeighborsClassifier(n_neighbors=min(errors1, key=errors1.get))
#fitting the training data
neigh.fit(features_train,labels_train)
predict=np.zeros(labels_test.shape)
```

```
#testing with test features
predict=neigh.predict(features_test)
kaggle.kaggleize(predict,str(path)+"/Submission/Predictions/"+str(DataSet)+"/CrossValidation_KNN.csv")
return errors1,errors2
```

5.a. After applying K fold technique for selecting number of neighbors for K Nearest Neighbor, we found following results on Digits Dataset. As per Graph, when k=1, it has minimal error i.e. optimal value of hyperparameter.



5.b. After Applying K Fold(k=10) method for hyper parameter selection of K nearest neighbor. We obtained following results:

KNN Method	Old Accuracy (%)	New Accuracy (%)
Digits	97.447%	97.684%
Semiconductor	94.481%	94.805%
Email Spam	80.110%	80.442%

Upon using optimal Hyperparameter according to training accuracy, Similar accuracy values were obtained, but it might be because our data is not big enough. For practical purpose Its better to count for test or cross validation accuracy, to avoid problems such as overfitting, or ruling out the possibilities of new learning through new entries.

5.c. For Digits Dataset: While using K Nearest Neighbors with default value of k=5, we were getting accuracy of 97.447%, and by using Optimal value of k=1, we improved accuracy by 0.23%.

For Semiconductor Dataset: Default KNN has given accuracy of 94.48%, whereas optimal k=19 improved classifications by 0.32%

For Email Dataset: Default KNN resulted in 80.442% accuracy. But via using optimal k=1, accuracy gets reduced to 80.110%
