Project 2 Report

1 Description of Algorithm

In this project, SVM (support vector machine) with cross validation is used to build the prediction model.

1.1 SVM

1.1.1 General Definition

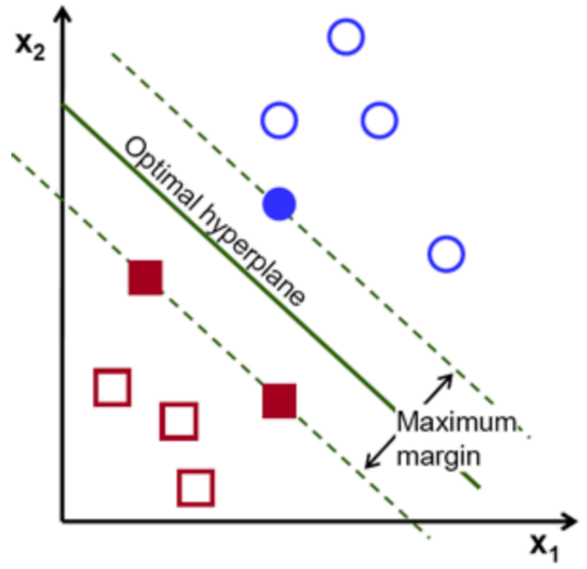
Support vector machines (SVMs) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis[1]. Given a set of labeled training data, the algorithm outputs an optimal hyperplane which categorizes new examples into one or the other of two categories(labels).
The operation of the SVM algorithm is based on finding the hyperplane that gives the largest minimum distance (**margin**) to the training examples. The data points from each class that lie closest to the hyperplane are known as **support vectors**.

1.1.2 Discriminative Algorithm

A SVM is a **discriminative** classifier defined by a separating hyperplane. Instead of assuming a probabilistic model for the data as in the generative algorithm (e.g. MAP hypothesis in Naïve Bayes), SVM does not care about how the data is generated, it simply categorizes the data. The goal of SVM is to find a "rule" to distinguish the examples in one class from the examples in the other class. Thus, it's a discriminative algorithm.

1.1.3 Mathematical Formula

The optimal separating hyperplane maximizes the margin of the training data. Figure below shows the optimal hyperplane and the maximum margin.



Given hyperplane g(x)=0 where g(x) $= w^T x + w_0$, the distance of x to the hyperplane is: $\frac{g(x)}{||w||} = \frac{g(x)}{\sqrt{w_1{}^2+w_2{}^2+\cdots+w_d{}^2}}$

Define canonical weights for the hyperplane to be w and w$_0$ such that $w^T x^{(i)} + w_0 = 1$ when $x^{(i)}$ is the closest point. Thus the goal is to find w and w$_0$ that maximize $\frac{1}{||w||}$, equivalently, $\min \frac{1}{2}||w||^2,\ subject\ to\ y^{(i)}\left(w^T x^{(i)} + w_0\right) \geq 1, i = 1, \ldots, n$

To avoid overfitting, a new slack variable is introduced to allow outliers. This is called soft margin SVM. Then the problem becomes:

$\min \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i$

$subject\ to\colon\ y^{(i)}\left(w^T x^{(i)} + w_0\right) \geq 1 - \xi_i, i = 1, \ldots, n$

Using L2 regularization, objective function is $\mathrm{L} = \sum_{i=1}^{n} \max\left(0,1 - w^T x_i y_i\right) + \frac{C}{2}||w||^2$

C is the tunable/penalty parameter. When C is large, classifier is not willing to have outliers, thus if C is too large, the classifier will try to make as less mistakes as possible on the training data, this will lead to overfitting. When C is too small, it will care little about the classification error. The performance of the model is weak (under fitting). Thus, choosing an appropriate C value is important for a good SVM model.

In this project, C=0.005,0.05,0.1 are evaluated to get a best model.

1.1.4 Assumptions

a) Examples are *independent and identically distributed*(IID)
b) The margin should be as large as possible.
c) The support vectors are the most useful data points because they are the ones most likely to be incorrectly classified.

1.2 Cross Validation

K- fold cross validation divides a dataset into K subsets and perform an analysis repeatedly using K-1 subsets as training set and 1 as the testing set. Each round of the validation process involves training the classifier using the training set and evaluating the classifier performance on the testing set.

In this project, a 5-fold cross validation is applied to evaluate the C value of the estimator. The training set is split into 5 smaller sets. For each fold i, all other 4 folds are training data and the resulting model will be validated on this fold i. The performance measure reported by 5-fold cross-validation is then the average of the accuracy computed in the loop.

2 Restrictions on Data

In this project, the "titanic-train" data is used. "Survived" is the label, and {Pclass, Sex, Age, Fare} are features.

The independent variable in R can be both numerical and categorical (feature "sex"). We can set scale=TRUE without factoring the data. However in spark, values of column "sex"{male, female} will be mapping to {1, 2} first. Then all data will be standardized, with variance equal to 1.

The dependent variable does not need to be scaled. In R, it is categorical but it's fine to do training without factoring it. (Though the output predicted label is categorical). In spark, it's mapping to Double type.

3 Scala Code

This is only **part of the code** to illustrate the most important details of implementation.[2,3]

```
// Linear SVM model
val lsvc = new LinearSVC()
  .setMaxIter(100).setTol(1e-6)
  .setStandardization(true).setFitIntercept(true).setThreshold(0.0)
  .setLabelCol("label").setFeaturesCol("features")

//Parameter C
val paramGrid = new ParamGridBuilder()
  .addGrid(lsvc.regParam, Array(0.005,0.05,0.1)).build()

// Evaluate accuracy at each iteration
val evaluator = new MulticlassClassificationEvaluator()
 .setLabelCol("label").setPredictionCol("prediction").setMetricName("accuracy")

// Cross Validation for lsvc model
val cv = new CrossValidator()
  .setEstimator(lsvc)
  .setEvaluator(evaluator).setEstimatorParamMaps(paramGrid).setNumFolds(5)

// Train on whole training set use best model
val cvModel = cv.fit(training)

// Test on the test set
val predictions = cvModel.transform(test)
```

Due to the need of setting different parameters, org.apache.spark.ml.LinearSVC is used in this project(instead of mllib.SVMWithSGD which includes stochastic gradient decent). Same parameter values in R and Spark are:
Max Iteration = 100;
Tolerance = 1e-6;
Scale = TRUE;
Threshold = 0;

Parameter that is evaluated for the estimator is penalty parameter C ("cost" in R). First, a 5-fold Cross-Validation is operated on the training set, repeating for 3 times with 3 different C, respectively.
Then, a best model with largest accuracy is generated using this C value. Finally, test on the test set and get confusion matrix and AUC number. Details are in the "svm.scala" file.

4 R Code

In R, it's easy to set parameters with svm()[4], change the value of cost and compare the mean accuracy for each C. Then predict use the best model.

```
sv<-svm(Survived ~ ., data=traindata, kernel='linear',type='C-
classification',scale = TRUE,tol=1e-6,max_iter=100,cost=0.1,cross=5)
summary(sv)
# test
sv<-svm(Survived ~ ., data=traindata, kernel='linear',type='C-
classification',scale = TRUE,tol=1e-6,max_iter=100,cost=0.1)
pre <- predict(sv,testdata)
label <- testdata$Survived

# AUC and confusion matrix
prediction <- as.numeric(as.character(pre))
roc_obj <- roc(label, prediction)
auc(roc_obj)
table(label, pre)
```

Other details are in svm.R file.

5 Results in Spark and R

Spark:

1. 5-fold cross validation for C:

Average accuracy for each parameter(C):
0.7817817910035502
0.7817817910035502
0.**78378179100355**   **// C = 0.1**

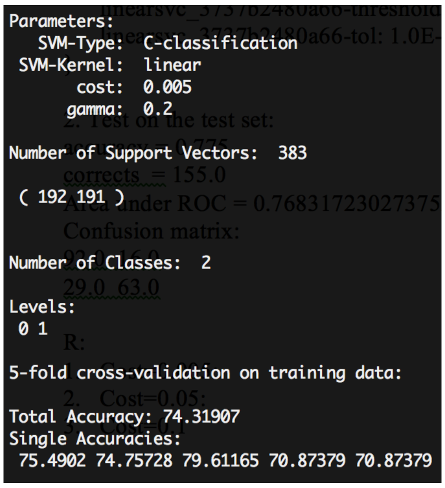
Best Model Parameters:
```
{
    linearsvc_3737b2480a66-aggregationDepth: 2,
    linearsvc_3737b2480a66-featuresCol: features,
    linearsvc_3737b2480a66-fitIntercept: true,
    linearsvc_3737b2480a66-labelCol: label,
    linearsvc_3737b2480a66-maxIter: 100,
    linearsvc_3737b2480a66-predictionCol: prediction,
    linearsvc_3737b2480a66-rawPredictionCol: rawPrediction,
    linearsvc_3737b2480a66-regParam: 0.1,
    linearsvc_3737b2480a66-standardization: true,
    linearsvc_3737b2480a66-threshold: 0.0,
    linearsvc_3737b2480a66-tol: 1.0E-6
}
```

2. Test on the test set:

accuracy = 0.775
corrects = 155.0
Area under ROC = 0.7683172302737521
Confusion matrix:
92.0 16.0
29.0 63.0

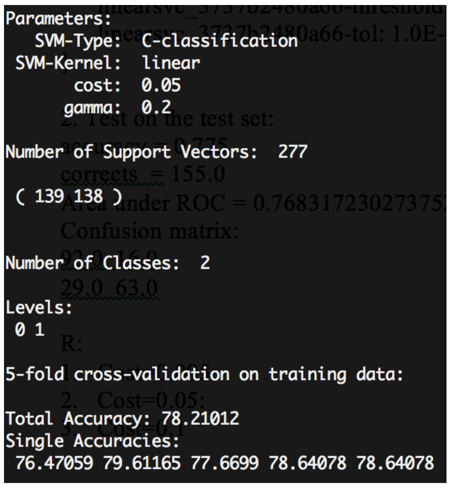
R:

1. Cost=0.005:   **74.31%**



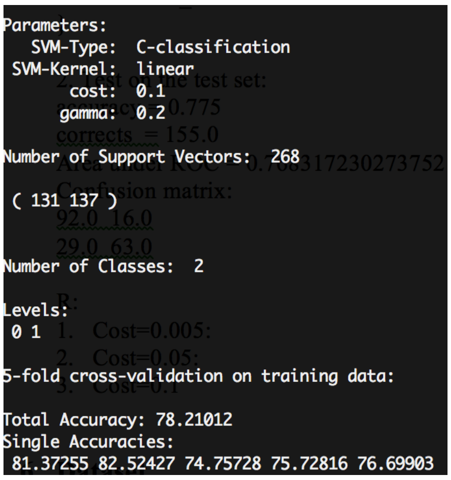
```
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.005
      gamma:  0.2

Number of Support Vectors:  383

 ( 192 191 )


Number of Classes:  2

Levels:
 0 1

5-fold cross-validation on training data:

Total Accuracy: 74.31907
Single Accuracies:
 75.4902 74.75728 79.61165 70.87379 70.87379
```

2. Cost=0.05: **78.21%**



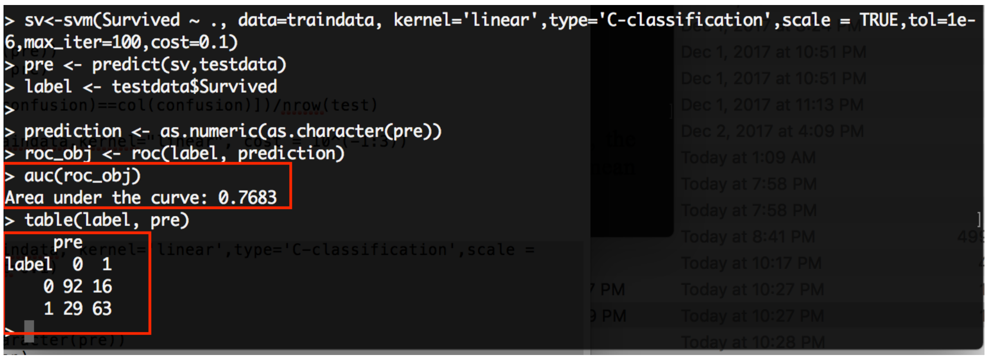
```
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.05
      gamma:  0.2

Number of Support Vectors:  277

 ( 139 138 )


Number of Classes:  2

Levels:
 0 1

5-fold cross-validation on training data:

Total Accuracy: 78.21012
Single Accuracies:
 76.47059 79.61165 77.6699 78.64078 78.64078
```

2. Cost=0.1: **78.21%**



```
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.1
      gamma:  0.2

Number of Support Vectors:  268

 ( 131 137 )


Number of Classes:  2

Levels:
 0 1

5-fold cross-validation on training data:

Total Accuracy: 78.21012
Single Accuracies:
 81.37255 82.52427 74.75728 75.72816 76.69903
```

Then test the model on the test set:



```
> sv<-svm(Survived ~ ., data=traindata, kernel='linear',type='C-classification',scale = TRUE,tol=1e-
6,max_iter=100,cost=0.1)
> pre <- predict(sv,testdata)
> label <- testdata$Survived
>
> prediction <- as.numeric(as.character(pre))
> roc_obj <- roc(label, prediction)
> auc(roc_obj)
Area under the curve: 0.7683
> table(label, pre)
     pre
label  0  1
    0 92 16
    1 29 63
>
```

Due to the differences in implementation(e.g. in R, even with same value of cost, the accuracy can be different. Maybe the data is randomly split into different folds), the mean accuracies in Spark and R are slightly different.

In Spark, C = 0.1 works best, while in R, result is same for C=0.1 and C = 0.05.
C = 0.1 is chosen in both Spark and R to compute the AUC number and confusion matrix.

AUC number and Confusion Matrix is **same** on the test set in both R and Spark.

6 Dataset

Dataset in this project is same as the dataset in slideshare.[5]
Pre-processing is same as slideshare, 200 randomly chosen examples are test set and the remaining is training set.
Save files to “train.csv” and “test.csv”.

References

[1] Support vector machine, Wikipedia,
https://en.wikipedia.org/wiki/Support_vector_machine

[2] Linear Support Vector Machine, Spark 2.2.0 Doc, https://spark.apache.org/docs/2.2.0/ml-classification-regression.html#linear-support-vector-machine

[3] LinearSVC.scala, Github,
https://github.com/apache/spark/blob/v2.2.0/mllib/src/main/scala/org/apache/spark/ml/classification/LinearSVC.scala

[4] svm, R Documentaion, https://www.rdocumentation.org/packages/e1071/versions/1.6-8/topics/svm

[5] Dataset: titanic-train.csv http://christianherta.de/lehre/dataScience/machine Learning/data/titanic-train.csv