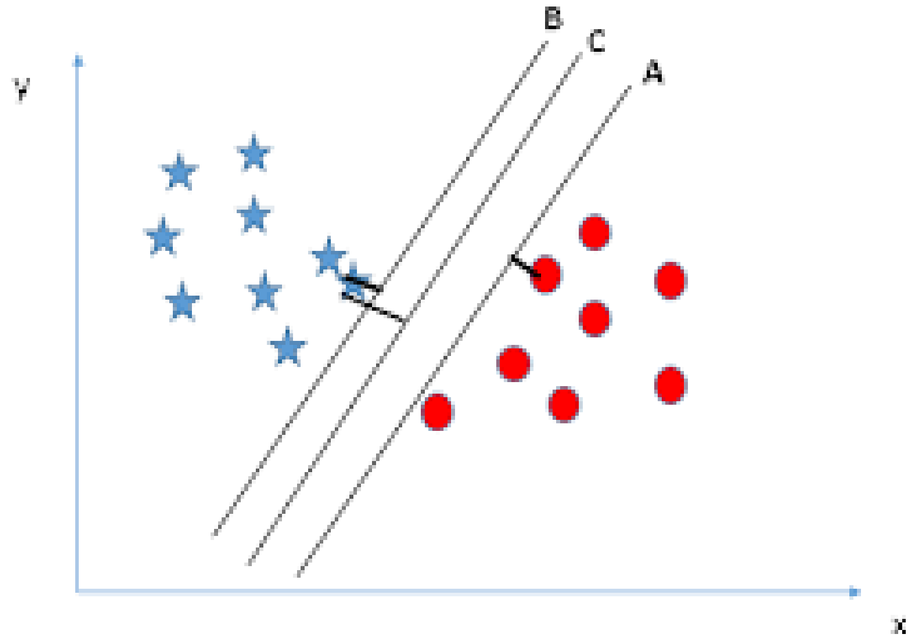
Practice 4

Support Vector Machine

Problem

- Predict whether income exceeds $50K/yr
- Use linear SVM in mllib



- Use predefined function in pyspark.**mllib.classification**

Dataset

➢ Dataset description

- We encoded the features of data points like following

➢ 14 Statistic Features

1. >50K or <=50K
2. Age
3. workclass
4. Weight
5. Eductation
6. Education-num
…

➢ Preprocessed 123 features

1. >50K or <=50K
2. Age1
3. Age2
4. Age3
5. workclass1
…
124. native-country

❖ **The first column of the data matrix indicates the class labels.**

* UCI Machine Learning Repository :

https://archive.ics.uci.edu/ml/datasets/adult

➢ You can download the pre-processed train and test dataset on i-campus

Dataset

➢ **We encoded original feature(Continuous, Categorical, etc) to have True/False value.**

- 1. **Continuous feature**: Whether this value is in specific range

  For example, if the value of age variable is bigger than 30 and smaller than 40, then the value of new range feature “30~40” becomes true.

- 2. **Categorical feature**: We made new features (for example, ***A***, ***B*** and ***C***) which are same with original category, and if one data had ***A*** category, then this data has **true** value in ***A*** feature but **false** value in ***B*** and ***C*** feature (OneHotEncoding).

	AGE	Categorical
Data 1	35	A

	10~20	20~30	30~40	A	B	C
Data 1	False	False	True	True	False	False

Practice 4

1. **Use predefined classes in *pyspark.mllib.classification : SVMwithSGD()***

   Parameters for the method (default)

   - ***iterations = 100, step = 1.0, regParam = 0.01, regTypre = “l2”***

2. **After training the models, calculate the F1 score, precision, recall for each label and accuracy using test data points.**

3. **Due date: May 7th 23:59**

https://spark.apache.org/docs/latest/api/python/pyspark.mllib.html#pyspark.mllib.classification.SVMWithSGD

https://spark.apache.org/docs/latest/mllib-linear-methods.html

Submission

- **You need to submit *result.txt* file**
  - ✓ *Write **f1 score**, **precision**, **recall** value of SVM result for **label 0**, **NOT** using predefined function but using **filter()** function*
  - ✓ *Write **f1 score, precision, recall** value of SVM result for **label 1**, **NOT** using predefined function but using **filter()** function*
  - ✓ *Write **accuracy** for **all labels**, using **TP(true positive), TN(true negative), FN(false negative),** and **FP(false positive)** values.*

Label 0
F1 Score : 0.8958
Precision : 0.8528
Recall : 0.9433

Label 1
F1 Score : 0.5976
Precision : 0.7419
Recall : 0.5003

Accuracy : 0.8345

Windows

```
Label 0
F1 Score : 0.8958
Precision : 0.8528
Recall : 0.9433
```

```
Label 1
F1 Score : 0.5976
Precision : 0.7419
Recall : 0.5003

Accuracy : 0.8345
```

Linux

Solution

- Import package

Import the Spark Package in your program

```
from pyspark import SparkConf, SparkContext
from pyspark.mllib.classification import LabeledPoint, SVMWithSGD
```

- Initialize a SparkContext

```
conf = SparkConf()
conf.set("spark.master", "local")
sc = SparkContext(conf=conf)
```

Configure Spark with SparkConf
Master of Spark as local computer

- Parse data to LabeledPoint

```
def parsePoint(line):
    try:
        values = [float(x) for x in line.replace(',',' ').split(' ')]
        return LabeledPoint(values[0], values[1:])
    except:
        return None
```

Make RDDs have label attribute and feature attribute

Solution

- Create RDDs (Import data) & Parse the data

```
data = sc.textFile("train_practice4.csv", 3)
trainData = data.map(parsePoint)

data = sc.textFile("test_practice4.csv")
testData = data.map(parsePoint)
```

Create RDDs
textFile(): load data from an external storage

3 means we will split data into three partitions

Transform RDDs using user-defined function

- Support Vector Machine

Train the model using Train RDDs which have label and feature together

```
# Support Vector Machine
model_SVM = SVMWithSGD.train(trainData, iterations=100,
                             step=1.0, regParam=0.01, regType="l2")
```

Solution

- Predict & get F1score(Label 0)

Predict the test RDDs' label

```
prediction = testData.map(lambda p: (p.label, model_SVM.predict(p.features)))

f = open('result.txt','w')
```

Find item satisfying lambda condition

```
# Label 0
tp = float(prediction.filter(lambda p: (p[0]==p[1]) & (p[0]==0)).count())
fn = float(prediction.filter(lambda p: (p[0]!=p[1]) & (p[0]==0)).count())
fp = float(prediction.filter(lambda p: (p[0]!=p[1]) & (p[0]==1)).count())
tn = float(prediction.filter(lambda p: (p[0]==p[1]) & (p[0]==1)).count())
```

Calculate F1 Score

```
precision = tp / (tp + fp)
recall = tp / (tp + fn)
f1_score = 2 * precision * recall / (precision + recall)

f.write('Label 0\n')
f.write('F1 Score : {:.4f}\n'.format(f1_score))
f.write('Precision : {:.4f}\n'.format(precision))
f.write('Recall : {:.4f}\n\n'.format(recall))
```

Solution

➢ **Predict & get F1score(Label 1)**

```
# Label 1
tp = float(prediction.filter(lambda p: (p[0]==p[1]) & (p[0]==1)).count())
fn = float(prediction.filter(lambda p: (p[0]!=p[1]) & (p[0]==1)).count())
fp = float(prediction.filter(lambda p: (p[0]!=p[1]) & (p[0]==0)).count())
tn = float(prediction.filter(lambda p: (p[0]==p[1]) & (p[0]==0)).count())

precision = tp / (tp + fp)
recall = tp / (tp + fn)
f1_score = 2 * precision * recall / (precision + recall)

f.write('Label 1\n')
f.write('F1 Score : {:.4f}\n'.format(f1_score))
f.write('Precision : {:.4f}\n'.format(precision))
f.write('Recall : {:.4f}\n\n'.format(recall))

accuracy = (tp + tn) / (tp + fn + fp + tn)
f.write('Accuracy : {:.4f}'.format(accuracy))

sc.stop()
```

Calculate model accuracy for all labels

Solution

➢ Result

- **Your result might be like the following**

Label 0
F1 Score : 0.8958
Precision : 0.8528
Recall : 0.9433

Label 1
F1 Score : 0.5976
Precision : 0.7419
Recall : 0.5003

Accuracy : 0.8345

Windows

```
Label 0
F1 Score : 0.8958
Precision : 0.8528
Recall : 0.9433

Label 1
F1 Score : 0.5976
Precision : 0.7419
Recall : 0.5003

Accuracy : 0.8345
```

Linux