Zhang, Jinhan PID A53211930
University of California, San Diego

February 5, 2017

HOMEWORK TWO

Problem 1. Solution

Results after we re-shuffle the data are shown as Table 1:

Table 1: Results after randomly re-shuffling the data

λ	Train Accuracy	Validate Accuracy	Test Accuracy
0	0.751225490196	0.757501530925	0.738518064911
0.01	0.748774509804	0.757501530925	0.738518064911
1	0.729166666667	0.753827311696	0.72933251684
100	0.66237745098	0.681567666871	0.680342927128

Listing 1: Key code for Prob.1

```
random.seed(0)

print "Reading data..."
dataFile = urllib.urlopen("http://archive.ics.uci.edu/ml/machine-learning-databases/
                                                    wine-quality/winequality-white.csv")
header = dataFile.readline()
fields = ["constant"] + header.strip().replace('"', '').split(';')
featureNames = fields[:-1]
labelName = fields[-1]
lines = [[1.0] + [float(x) for x in l.split(';')] for l in dataFile]
random.shuffle(lines)        ##randomly shuffle
X = [l[:-1] for l in lines]
y = [l[-1] > 5 for l in lines]
print "done"
```

Problem 2. Solution

Classifiers Evaluation Results are shown as Table 2:

Table 2: Classifiers Evaluation Value

True Positives	1129
True Negatives	145
False Positives	321
False Negatives	38
Balanced Error Rate	0.360701663412

Listing 2: Key code for Prob.2

```
def performance2(theta):

    scores_test = [inner(theta, x) for x in X_test]

    predictions_test = [s > 0 for s in scores_test]
    TP = sum([(a == 1 and b == 1) for (a, b) in zip(predictions_test, y_test)])
    TN = sum([(a == 0 and b == 0) for (a, b) in zip(predictions_test, y_test)])
    FP = sum([(a == 1 and b == 0) for (a, b) in zip(predictions_test, y_test)])
    FN = sum([(a == 0 and b == 1) for (a, b) in zip(predictions_test, y_test)])
    TPR = TP * 1.0 / (TP + FN)
    TNR = TN * 1.0 / (TN + FP)
    BER = 1 - (TPR + TNR) / 2

    return TP,TN,FP,FN,BER


lam = 0.01
theta = train(lam)
TP,TN,FP,FN,BER = performance2(theta)
print("TP = " + str(TP) + ";TN=" + str(TN) + "; FP=" + str(FP) + "; FN=" + str(
FN) + "; BER=" + str(BER))
```

Problem 3. Solution

Results are shown as Table 3:

Table 3: Ranking Evaluation Results

Number of Return	Precision	Recall
10	1.0	0.00856898029135
500	0.956	0.409597257926
1000	0.864	0.740359897172

Listing 3: Key code for Prob.3

```
def performance3(theta,numR):

    scores_test = [inner(theta, x) for x in X_test]
    combine = [[scores_test[i]] + [y_test[i]] for i in range(0, len(y_test))]
    combine.sort(key=lambda x: x[0], reverse=True)
    total = sum(l[1]==True for l in combine)
    rel = sum(combine[i][1]==True for i in range(0,numR))
    precision = rel * 1.0 / numR
    recall = rel *1.0 / total

    return precision,recall

lam = 0.01
theta = train(lam)
for numReturn in [10,500,1000]:
    precision,recall = performance3(theta,numReturn)
    print("numReturn = " + str(numReturn) + ";\tprecision=" + str(precision) + ";
                                                recall=" + str(recall))
```

Problem 4. Solution

Plot is shown as Fig.1 below:

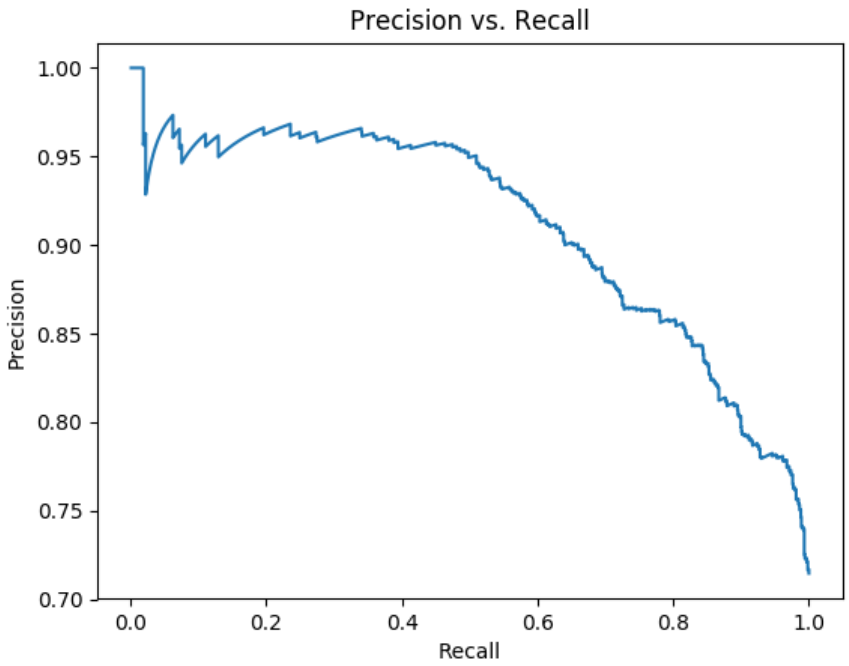

Figure 1: Precision vs. recall as the number of results considered varies

Listing 4: Key code for Prob.4

```
def performance3(theta,numR):

    scores_test = [inner(theta, x) for x in X_test]
    combine = [[scores_test[i]] + [y_test[i]] for i in range(0, len(y_test))]
    combine.sort(key=lambda x: x[0], reverse=True)
    total = sum(l[1]==True for l in combine)
    rel = sum(combine[i][1]==True for i in range(0,numR))
    precision = rel * 1.0 / numR
    recall = rel *1.0 / total

    return precision,recall


lam = 0.01
theta = train(lam)
result_pre = []
result_rec = []
for numReturn in range(1,len(y_test)+1):
    precision,recall = performance3(theta,numReturn)
    result_pre.append(precision)
    result_rec.append(recall)
plt.plot(result_rec,result_pre)
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision vs. Recall')
plt.show()
```

Problem 5. Solution

The 'reconstruction error' of the naive 'compression' is: 3675818.61688

Listing 5: Key code for Prob.5

```
X_mean=numpy.mean(X_train,axis=0)
diff=X_train-X_mean
re=sum(sum(numpy.multiply(diff,diff)).T)
print re
```

Problem 6. Solution

PCA components of winequality dataset are shown as Fig.2 below:

```
[[ -3.23636346e-04   1.42201752e-04   3.17030713e-04   5.36390435e-02
    9.30284526e-05   2.54030965e-01   9.65655009e-01   3.19990241e-05
   -2.95831396e-04   3.84043646e-04  -1.00526693e-02]
 [ -7.57985623e-03  -1.66366340e-03   1.04742899e-03   5.21677266e-02
    4.49425600e-05   9.65020304e-01  -2.56793964e-01   7.90089050e-06
    5.24900596e-04  -1.09699394e-03  -2.89827657e-03]
 [  1.82124420e-02   2.54680710e-03   3.31838657e-03   9.93221259e-01
   -1.51888372e-04  -6.42297821e-02  -3.91682592e-02   4.30929482e-04
   -6.93199060e-03  -2.85216045e-03  -8.62920933e-02]
 [  1.56811999e-01   3.28220652e-03   1.66866136e-02   8.28549640e-02
   -6.91822288e-03   1.13029682e-03   5.39110108e-03  -9.49080503e-04
    2.68027305e-03   1.30498102e-03   9.83955205e-01]
 [  9.81360642e-01  -1.45890108e-02   5.92643662e-02  -3.17546064e-02
    5.07483182e-04   8.43759364e-03  -1.77578042e-03   6.03725221e-04
   -9.05011239e-02  -9.35630845e-03  -1.54417839e-01]
 [  7.76578401e-02  -2.37665885e-01   2.23406619e-02   5.04113878e-03
   -1.43564098e-02  -2.14210997e-04  -2.22913844e-04   3.36617054e-03
    8.77254205e-01   4.08570175e-01  -1.54145486e-02]
 [ -7.36289612e-02  -2.61563804e-01   9.43067566e-01  -2.14514264e-03
    1.19104298e-02  -1.68808905e-03   1.42294158e-04  -1.17203197e-04
   -1.45895558e-01   1.23868963e-01  -2.88797236e-03]
 [ -1.37617196e-02   2.11129619e-01  -1.16514121e-01   5.30670319e-04
    1.05181628e-02   1.36446528e-03  -8.21179429e-04   3.09221855e-04
   -3.58358431e-01   9.01728510e-01   3.27758247e-03]
 [  1.74575775e-02   9.10890084e-01   3.04081497e-01  -2.89763923e-03
    2.34615054e-02   1.17406025e-03  -3.85957239e-04   1.23176271e-03
    2.68927937e-01  -6.70756658e-02  -1.12101920e-02]
 [  2.31513441e-03  -2.38717789e-02  -1.67445603e-02   8.92206499e-04
    9.99462734e-01  -9.81109101e-05  -3.32812875e-05   4.14235255e-03
    1.18483756e-02  -3.51543098e-03   6.92344110e-03]
 [  7.48312160e-04   3.08204153e-04   2.55232500e-04   3.49846801e-04
    4.12943179e-03  -6.96565372e-06   4.16951216e-06  -9.99984215e-01
    3.17948604e-03   1.53436134e-03  -1.10029138e-03]]
```

Figure 2: PCA components of winequality dataset

Listing 6: Key code for Prob.6

```
print "Reading data..."
dataFile = urllib.urlopen("http://archive.ics.uci.edu/ml/machine-learning-databases/
                                                    wine-quality/winequality-white.csv")
header = dataFile.readline()
fields = ["constant"] + header.strip().replace('"', '').split(';')
featureNames = fields[:-1]
labelName = fields[-1]
lines = [[float(x) for x in l.split(';')] for l in dataFile]
```

```
X = [l[:-1] for l in lines]
y = [l[-1] > 5 for l in lines]
print "done"

X_train = X[:int(len(X) / 3)]
y_train = y[:int(len(y) / 3)]
X_validate = X[int(len(X) / 3):int(2 * len(X) / 3)]
y_validate = y[int(len(y) / 3):int(2 * len(y) / 3)]
X_test = X[int(2 * len(X) / 3):]
y_test = y[int(2 * len(X) / 3):]

pca = PCA(n_components=11)
pca.fit(X_train)
print pca.components_
```

Problem 7. Solution

The reconstruction error is: 1345.47557409

We can use two ways to compute the reconstruction error, one is to use the projection matrix we get to reconstruct original daata and then compute the error, the other is use the sum of the variance we drop to compute the reconstruction error.

Listing 7: Key code for Prob.7

```
pca = PCA(n_components=4)
pca.fit(X_train)


X_PCA=pca.fit_transform(X_train)      ##Way 1
x_rec2=pca.inverse_transform(X_PCA)
x_rec2=numpy.matrix(x_rec2)
error2=sum(sum(numpy.multiply(x_rec2-X_train,x_rec2-X_train)).T).tolist()[0][0]
print error2

print pca.noise_variance_*len(x_rec2)   ##Way 2
```

Problem 8. Solution

Plot is shown as Fig.3 below. We can see from the plot that both training and test set MSE decrease with the increment of the number of dimensions in general. And the test set sometimes gets higher MSE with a higher number of dimensions.

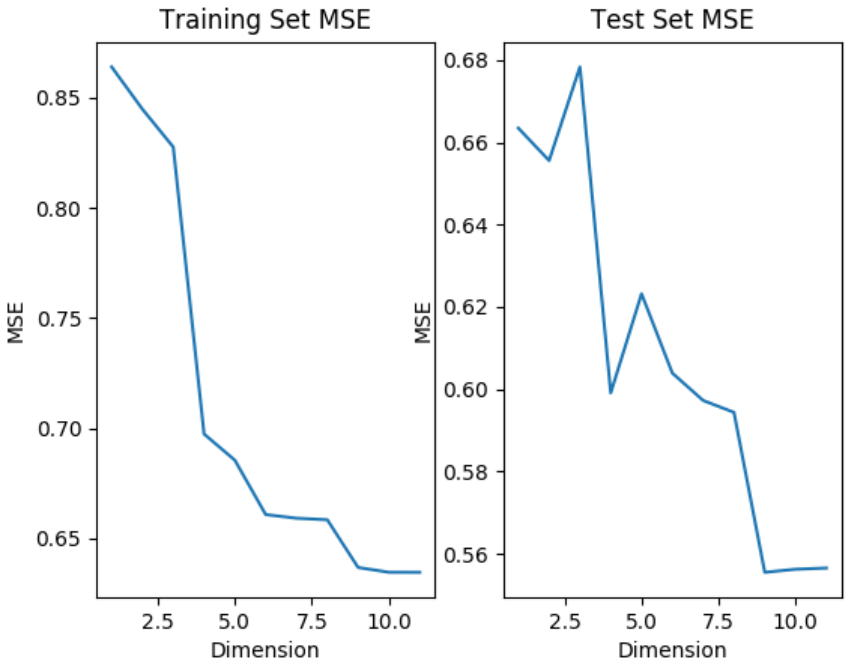

Figure 3: MSE vs. dimensions on the train and test sets

Listing 8: Key code for Prob.8

```
train_mse=[]
test_mse=[]

for dim in range(1,12):
    pca = PCA(n_components=dim)
    X_PCA = pca.fit_transform(X_train).tolist()
    X_PCA = [[1] + elem for elem in X_PCA]
    theta, l, info = scipy.optimize.fmin_l_bfgs_b(f, [0]*(dim+1), fprime, args = (X_PCA
                                                  , y_train, 0))  #lambda
    #print "Final log likelihood =", -l
    X_test_PCA = pca.transform(X_test).tolist()
    X_test_PCA = [[1] + elem for elem in X_test_PCA]
    X_test_PCA=numpy.matrix(X_test_PCA)
    theta=numpy.matrix(theta)
    y_test=numpy.matrix(y_test)
    y=X_test_PCA*theta.T
    k=y-y_test.T
    mse2=(k.T*k/len(y)).tolist()[0][0]
    test_mse.append(mse2)

    X_PCA = numpy.matrix(X_PCA)
    y_train = numpy.matrix(y_train)
```

```
    y = X_PCA * theta.T
    k = y - y_train.T
    mse1 = (k.T * k / len(y)).tolist()[0][0]
    train_mse.append(mse1)
XX=range(1,12)
plt.figure()
ax1 = plt.subplot(121)
ax2 = plt.subplot(122)
plt.sca(ax1)
plt.plot(XX,train_mse)
plt.title('Training Set MSE')
plt.xlabel('Dimension')
plt.ylabel('MSE')
plt.sca(ax2)
plt.plot(XX,test_mse)
plt.xlabel('Dimension')
plt.ylabel('MSE')
plt.title('Test Set MSE')
plt.show()
```

Submitted by Zhang, Jinhan PID A53211930 on February 5, 2017.