Bilkent University

Department of Computer Engineering

CS - 464
Introduction to Machine Learning

Homework 2

Name : Osman Burak İntişah
Id : 2160243
Section : 2
Date : 3.05.2020

Question 1 - SVM and Logistic Regression

Superclass Classification Using Logistic Regression

Question 1.1

Output for HOG (mini-batch gradient ascent algorithm):

```
When learning rate is 1e-06 ..
CPU times: user 39.3 s, sys: 1.06 s, total: 40.4 s
Wall time: 39.2 s
Accuracy : 0.56
Precision : 0.5335195530726257
Recall : 0.955
NPV : 0.7857142857142857
FPR : 0.835
FDR : 0.4664804469273743
F1 Score : 0.6845878136200716
F2 Score: 1.25
Confusion Matrix: [[191, 167], [9, 33]]

When learning rate is 1e-05 ..
CPU times: user 39.3 s, sys: 1.08 s, total: 40.3 s
Wall time: 39.1 s
Accuracy : 0.66
Precision : 0.686046511627907
Recall : 0.59
NPV : 0.6403508771929824
FPR : 0.27
FDR : 0.313953488372093
F1 Score : 0.6344086021505376
F2 Score: 1.25
Confusion Matrix: [[118, 54], [82, 146]]

When learning rate is 0.001 ..
CPU times: user 39.3 s, sys: 1.1 s, total: 40.5 s
Wall time: 39.2 s
Accuracy : 0.6575
Precision : 0.6632124352331606
Recall : 0.64
NPV : 0.6521739130434783
FPR : 0.325
```

```
FDR : 0.33678756476683935
F1 Score : 0.6513994910941475
F2 Score: 1.2500000000000002
Confusion Matrix: [[128, 65], [72, 135]]

When learning rate is 0.1 ..
CPU times: user 38.9 s, sys: 1.06 s, total: 39.9 s
Wall time: 38.7 s
Accuracy : 0.625
Precision : 0.6488095238095238
Recall : 0.545
NPV : 0.6077586206896551
FPR : 0.295
FDR : 0.35119047619047616
F1 Score : 0.5923913043478262
F2 Score: 1.25
Confusion Matrix: [[109, 59], [91, 141]]

When learning rate is 1 ..
CPU times: user 38.1 s, sys: 1.08 s, total: 39.2 s
Wall time: 37.9 s
Accuracy : 0.595
Precision : 0.5562130177514792
Recall : 0.94
NPV : 0.8064516129032258
FPR : 0.75
FDR : 0.4437869822485207
F1 Score : 0.6988847583643122
F2 Score: 1.25
Confusion Matrix: [[188, 150], [12, 50]]

Best learning rate for mini batch trained with hog: 1e-05
```

Output for Neural Network Generated Features (mini-batch gradient ascent algorithm):

```
When learning rate is 1e-06 ..
CPU times: user 3min 14s, sys: 2min 14s, total: 5min 29s
Wall time: 2min 51s
Accuracy : 0.8775
Precision : 0.8544600938967136
Recall : 0.91
NPV : 0.9037433155080213
```

```
FPR : 0.155
FDR : 0.14553990610328638
F1 Score : 0.8813559322033899
F2 Score: 1.25
Confusion Matrix: [[182, 31], [18, 169]]

When learning rate is 1e-05 ..
CPU times: user 3min 20s, sys: 2min 16s, total: 5min 37s
Wall time: 2min 55s
Accuracy : 0.875
Precision : 0.8537735849056604
Recall : 0.905
NPV : 0.898936170212766
FPR : 0.155
FDR : 0.14622641509433962
F1 Score : 0.8786407766990291
F2 Score: 1.2500000000000002
Confusion Matrix: [[181, 31], [19, 169]]

When learning rate is 0.001 ..
CPU times: user 3min 30s, sys: 2min 19s, total: 5min 49s
Wall time: 3min 4s
Accuracy : 0.8775
Precision : 0.8682926829268293
Recall : 0.89
NPV : 0.8871794871794871
FPR : 0.135
FDR : 0.13170731707317074
F1 Score : 0.8790123456790123
F2 Score: 1.25
Confusion Matrix: [[178, 27], [22, 173]]

When learning rate is 0.1 ..
CPU times: user 3min 34s, sys: 2min 19s, total: 5min 53s
Wall time: 3min 9s
Accuracy : 0.88
Precision : 0.8762376237623762
Recall : 0.885
NPV : 0.8838383838383839
FPR : 0.125
FDR : 0.12376237623762376
F1 Score : 0.880597014925373
```

```
F2 Score: 1.25
Confusion Matrix: [[177, 25], [23, 175]]

When learning rate is 1 ..
CPU times: user 3min 34s, sys: 2min 20s, total: 5min 55s
Wall time: 3min 8s
Accuracy : 0.875
Precision : 0.8676470588235294
Recall : 0.885
NPV : 0.8826530612244898
FPR : 0.135
FDR : 0.1323529411764706
F1 Score : 0.8762376237623763
F2 Score: 1.2499999999999998
Confusion Matrix: [[177, 27], [23, 173]]

Best learning rate for mini batch trained with neural network:
0.1
```

Output for HOG (stochastic gradient ascent algorithm):

```
When learning rate is 1e-06 ..
CPU times: user 39.3 s, sys: 1.09 s, total: 40.4 s
Wall time: 39.2 s
Accuracy : 0.575
Precision : 0.5431034482758621
Recall : 0.945
NPV : 0.7884615384615384
FPR : 0.795
FDR : 0.45689655172413796
F1 Score : 0.6897810218978102
F2 Score: 1.25
Confusion Matrix: [[189, 159], [11, 41]]

When learning rate is 1e-05 ..
CPU times: user 39 s, sys: 1.13 s, total: 40.1 s
Wall time: 38.9 s
Accuracy : 0.66
Precision : 0.6818181818181818
Recall : 0.6
NPV : 0.6428571428571429
FPR : 0.28
FDR : 0.3181818181818182
```

```
F1 Score : 0.6382978723404256
F2 Score: 1.2499999999999998
Confusion Matrix: [[120, 56], [80, 144]]

When learning rate is 0.001 ..
CPU times: user 39.4 s, sys: 1.09 s, total: 40.5 s
Wall time: 39.3 s
Accuracy : 0.66
Precision : 0.6666666666666666
Recall : 0.64
NPV : 0.6538461538461539
FPR : 0.32
FDR : 0.3333333333333333
F1 Score : 0.6530612244897959
F2 Score: 1.25
Confusion Matrix: [[128, 64], [72, 136]]

When learning rate is 0.1 ..
CPU times: user 39.5 s, sys: 1.14 s, total: 40.6 s
Wall time: 39.4 s
Accuracy : 0.6225
Precision : 0.6449704142011834
Recall : 0.545
NPV : 0.6060606060606061
FPR : 0.3
FDR : 0.35502958579881655
F1 Score : 0.5907859078590786
F2 Score: 1.25
Confusion Matrix: [[109, 60], [91, 140]]

When learning rate is 1 ..
CPU times: user 39.2 s, sys: 1.12 s, total: 40.3 s
Wall time: 39.1 s
Accuracy : 0.6075
Precision : 0.6972477064220184
Recall : 0.38
NPV : 0.5738831615120275
FPR : 0.165
FDR : 0.30275229357798167
F1 Score : 0.49190938511326854
F2 Score: 1.2500000000000002
Confusion Matrix: [[76, 33], [124, 167]]
```

```
Best learning rate for stochastic trained with hog: 1e-05
```

Output for Neural Network Generated Features(stochastic gradient ascent algorithm):

```
When learning rate is 1e-06 ..
CPU times: user 3min 17s, sys: 2min 15s, total: 5min 32s
Wall time: 2min 51s
Accuracy : 0.875
Precision : 0.8472222222222222
Recall : 0.915
NPV : 0.907608695652174
FPR : 0.165
FDR : 0.1527777777777778
F1 Score : 0.8798076923076923
F2 Score: 1.25
Confusion Matrix: [[183, 33], [17, 167]]

When learning rate is 1e-05 ..
CPU times: user 4min 13s, sys: 2min 26s, total: 6min 40s
Wall time: 3min 47s
Accuracy : 0.875
Precision : 0.8537735849056604
Recall : 0.905
NPV : 0.898936170212766
FPR : 0.155
FDR : 0.14622641509433962
F1 Score : 0.8786407766990291
F2 Score: 1.2500000000000002
Confusion Matrix: [[181, 31], [19, 169]]

When learning rate is 0.001 ..
CPU times: user 4min 46s, sys: 2min 32s, total: 7min 19s
Wall time: 4min 17s
Accuracy : 0.8775
Precision : 0.8682926829268293
Recall : 0.89
NPV : 0.8871794871794871
FPR : 0.135
FDR : 0.13170731707317074
F1 Score : 0.8790123456790123
F2 Score: 1.25
```

```
Confusion Matrix: [[178, 27], [22, 173]]

When learning rate is 0.1 ..
CPU times: user 4min 39s, sys: 2min 31s, total: 7min 11s
Wall time: 4min 12s
Accuracy : 0.88
Precision : 0.8762376237623762
Recall : 0.885
NPV : 0.8838383838383839
FPR : 0.125
FDR : 0.12376237623762376
F1 Score : 0.880597014925373
F2 Score: 1.25
Confusion Matrix: [[177, 25], [23, 175]]

When learning rate is 1 ..
CPU times: user 4min 11s, sys: 2min 27s, total: 6min 38s
Wall time: 3min 43s
Accuracy : 0.8725
Precision : 0.8860103626943006
Recall : 0.855
NPV : 0.8599033816425121
FPR : 0.11
FDR : 0.11398963730569948
F1 Score : 0.8702290076335878
F2 Score: 1.25
Confusion Matrix: [[171, 22], [29, 178]]

Best learning rate for stochastic trained with neural net: 0.1
```

Result Discussion:
I have run mini-batch gradient ascent and stochastic gradient ascent algorithms for bot HOG and Neural Network generated features. For each run I have tried several learning rates. Which are **[1e-6, 1e-5, 1e-3, 1e-1, 1].** For each run with learning rates I calculated running time, accuracy, precision, recall, npv, fpr, fdr, f1 score, f2 score and confusion matrix. I compared the accuracy results while deciding on which learning rate is better than others. At the end I have concluded that best learning rate for HOG features is **1e-5** for both algorithm. And for neural network generated features best learning rate is found as **0.1** for both algorithms.

From the given performance metrics, it is seen that we can obtain better results when we use neural network generated features for both algorithm. However, running the algorithms with neural network generated features takes much more time than the HOG generated features. It is because of the size of the array is bigger than the other one.

Question 1.2

For this question I have used the learning rates obtained from the question 1.1. Which is **1e-5** HOG features, **0.1** for neural network generated features.

Output for HOG Generated Features:

```
When learning rate is 1e-05 ..
CPU times: user 886 ms, sys: 183 ms, total: 1.07 s
Wall time: 555 ms
10 most important feature indices:
[164, 289, 29, 195, 288, 154, 155, 153, 60, 61]
Accuracy : 0.6675
Precision : 0.6936416184971098
Recall : 0.6
NPV : 0.6475770925110133
FPR : 0.265
FDR : 0.3063583815028902
F1 Score : 0.6434316353887399
F2 Score: 1.25
Confusion Matrix: [[120, 53], [80, 147]]
```

Output for Neural Network Generated Features:

```
When learning rate is 0.1 ..
CPU times: user 6.18 s, sys: 326 ms, total: 6.5 s
Wall time: 3.35 s
10 most important feature indices:
[123, 993, 1948, 1783, 1189, 1108, 1928, 187, 1020, 287]
Accuracy : 0.8875
Precision : 0.8934010152284264
Recall : 0.88
NPV : 0.8817733990147784
FPR : 0.105
FDR : 0.1065989847715736
F1 Score : 0.8866498740554157
F2 Score: 1.25
Confusion Matrix: [[176, 21], [24, 179]]
```

Result Discussion:
For each run with given learning rates I calculated running time, accuracy, precision, recall, npv, fpr, fdr, f1 score, f2 score, confusion matrix and 10 most important features(in terms of indices).

From these information, it is seen that this algorithm runs quite faster than other two algorithms for both feature set. Additionally accuracy and other performance metrics are better for the neural network generated features as it was for the first two algorithms.

Superclass Classification Using SVM

Question 1.4 - Soft Margin SVM Model with linear kernel

For HOG Features
Mean of the selected performance metric: [0.6665 0.6735 0.6615 0.6435 0.639]
Optimal value of C: 0.1
Output for best C value on Test Set:

```
When c= 0.1, Accuracy: 0.68
When c= 0.1, Precision: 0.7
When c= 0.1, Recall: 0.63
When c= 0.1, F1 Score: 0.663157894736842
[[146  54]
 [ 74 126]]

CPU times: user 1.88 s, sys: 11.6 ms, total: 1.89 s
Wall time: 1.89 s
```

For Neural Network Generated Features
Mean of the selected performance metric: [0.8825 0.8685 0.8665 0.8665 0.8665]
Optimal value of C: 0.01
Output for best C value on Test Set:

```
When c= 0.01, Accuracy: 0.8925
When c= 0.01, Precision: 0.8792270531400966
When c= 0.01, Recall: 0.91
When c= 0.01, F1 Score: 0.8943488943488944
[[175  25]
 [ 18 182]]

CPU times: user 6.22 s, sys: 34.9 ms, total: 6.26 s
Wall time: 6.26 s
```

Question 1.5 - Hard Margin SVM Model with radial basis function (rbf) kernel

For HOG Features
Mean of the selected performance metric: [0.681 0.687 0.6875 0.7 0.705 0.6995 0.502]
Optimal value of Gamma: 1
Output for best Gamma value on Test Set:

```
When gamma= 1, Accuracy: 0.7225
When gamma= 1, Precision: 0.7192118226600985
When gamma= 1, Recall: 0.73
When gamma= 1, F1 Score: 0.7245657568238213
[[143  57]
 [ 54 146]]

CPU times: user 2.5 s, sys: 4 ms, total: 2.51 s
Wall time: 2.51 s
```

For Neural Network Generated Features
Mean of the selected performance metric: [0.585 0.517 0.503 0.501 0.5005 0.5 0.5]
Optimal value of Gamma : 0.0625
Output for best Gamma value on Test Set:

```
When gamma= 0.0625, Accuracy: 0.5875
When gamma= 0.0625, Precision: 0.547945205479452
When gamma= 0.0625, Recall: 1.0
When gamma= 0.0625, F1 Score: 0.7079646017699115
[[ 35 165]
 [  0 200]]

CPU times: user 18.1 s, sys: 17.1 ms, total: 18.1 s
Wall time: 18.1 s
```

Question 1.6 - Soft Margin SVM Model with radial basis function (rbf) kernel

For HOG Features
Mean of the selected performance metric: [0.666 0.676 0.501 0.693 0.7005 0.501 0.681 0.7045 0.501]
Optimal value of (Gamma,C): (2,100)
Output for best (Gamma,C) value on Test Set:

```
When gamma= 2, c = 100.0, Accuracy: 0.7075
When gamma= 2, c = 100.0, Precision: 0.6966824644549763
When gamma= 2, c = 100.0, Recall: 0.735
When gamma= 2, c = 100.0, F1 Score: 0.7153284671532847

[[136  64]
 [ 53 147]]

CPU times: user 2.54 s, sys: 1e+03 µs, total: 2.54 s
Wall time: 2.55 s
```

For Neural Network Generated Features
Mean of the selected performance metric:[0.5005 0.837 0.5   0.504 0.5   0.5   0.5065 0.5   0.5   ]
Optimal value of (Gamma,C): (2,0.01)
Output for best (Gamma,C) value on Test Set:

```
When gamma= 2, c = 0.01, Accuracy: 0.84
When gamma= 2, c = 0.01, Precision: 0.8333333333333334
When gamma= 2, c = 0.01, Recall: 0.85
When gamma= 2, c = 0.01, F1 Score: 0.8415841584158417

[[166  34]
 [ 30 170]]

CPU times: user 18.1 s, sys: 14.3 ms, total: 18.1 s
Wall time: 18.1 s
```

Result Discussion:
In order to compare the results for each gamma and C values in the algorithms I have used accuracy performance metric. Because after I have done my search I found out that accuracy is better when there are not imbalance for the classes in the data set. And since we were using the algorithm that we wrote for part 1.3, we were sure that classes are balanced for each fold.

Except the hard margin SVM training we obtained better results for neural network generated features. But when we used hard margin SVM for HOG features we obtained better accuracy results.

Additionally, it was also seen that it takes much more time when we use neural network generated features.

Note: I have give the execution time for training each fold. But I didn't add it to the report since the output for the best gives us the insight of the execution time.

Subclass Classification Using SVM

Question 1.7

For HOG Features
Mean of the selected performance metric: [0.277 0.329 0.31   0.258 0.348 0.3495 0.1665 0.1065 0.1095]
Optimal value of (Gamma,C): (2,100)
Output for best (Gamma,C) value on Test Set:

```
When gamma= 2, c = 100.0, Accuracy: 0.36

Accuracy : 0.36
Confusion Matrix :
[[21  2  4  2  0  4  3  3  0  1]
 [ 0  8  3  4 14  3  1  0  7  0]
 [ 6  5  5  7  5  2  3  2  4  1]
 [ 4  4  2 11  9  1  3  2  1  3]
 [ 2  6  0  3 20  0  1  2  5  1]
 [ 3  1  1  5  3 16  6  1  0  4]
 [ 2  1  0  6  1  2 20  4  2  2]
 [ 2  1  1  3  2  4  5 12  9  1]
 [ 1  5  3  4  7  1  2  6 11  0]
 [ 0  2  1  1  3  6  0  0  7 20]]
              precision    recall  f1-score   support

  subclass 0       0.51      0.53      0.52        40
  subclass 1       0.23      0.20      0.21        40
  subclass 2       0.25      0.12      0.17        40
  subclass 3       0.24      0.28      0.26        40
  subclass 4       0.31      0.50      0.38        40
  subclass 5       0.41      0.40      0.41        40
  subclass 6       0.45      0.50      0.48        40
  subclass 7       0.38      0.30      0.33        40
  subclass 8       0.24      0.28      0.26        40
```

```
  subclass 9       0.61      0.50      0.55        40

    accuracy                           0.36       400
   macro avg       0.36      0.36      0.36       400
weighted avg       0.36      0.36      0.36       400

CPU times: user 4.29 s, sys: 112 µs, total: 4.29 s
Wall time: 4.3 s
```

For Neural Network Generated Features
Mean of the selected performance metric:
Optimal value of (Gamma,C): (2,0.01)
Output for best (Gamma,C) value on Test Set:

```
When gamma= 2, c = 0.01, Accuracy: 0.58

Accuracy : 0.58
Confusion Matrix :
[[23  1  5  5  0  0  2  1  0  3]
 [ 0 20  0  2  6  0  0  3  8  1]
 [11  1 13  8  1  1  0  1  1  3]
 [ 4  4  5 21  2  0  1  0  3  0]
 [ 0  6  1  2 25  0  0  0  5  1]
 [ 3  0  0  0  0 32  1  0  1  3]
 [ 1  2  0  0  0  2 27  4  0  4]
 [ 1  2  1  0  0  3  1 22  8  2]
 [ 2 10  0  0  1  1  1  2 21  2]
 [ 3  2  0  2  0  2  1  0  2 28]]
              precision    recall  f1-score   support

  subclass 0       0.48      0.57      0.52        40
  subclass 1       0.42      0.50      0.45        40
  subclass 2       0.52      0.33      0.40        40
  subclass 3       0.53      0.53      0.53        40
  subclass 4       0.71      0.62      0.67        40
  subclass 5       0.78      0.80      0.79        40
  subclass 6       0.79      0.68      0.73        40
  subclass 7       0.67      0.55      0.60        40
  subclass 8       0.43      0.53      0.47        40
  subclass 9       0.60      0.70      0.64        40

    accuracy                           0.58       400
```

```
   macro avg       0.59      0.58      0.58       400
weighted avg       0.59      0.58      0.58       400

CPU times: user 26 s, sys: 14.1 ms, total: 26 s
Wall time: 26.1 s
```

Question 1.8

For HOG Features
Mean of the selected performance metric: [0.3295 0.3495 0.361 0.3295 0.3495 0.361 0.3295 0.3495 0.361]
Optimal value of (Gamma,Degree): (0.25,7)
Output for best (Gamma,Degree) value on Test Set:

```
When gamma= 64, degree = 3, Accuracy: 0.3775

Accuracy : 0.3775
Confusion Matrix :
[[18  2  3  5  1  1  4  4  0  2]
 [ 1 16  4  2 10  3  1  1  2  0]
 [10  7  9  3  1  2  3  1  3  1]
 [ 4  4  4  8  9  2  4  2  2  1]
 [ 4  6  1  3 22  0  1  0  1  2]
 [ 5  2  2  4  1 15  4  4  0  3]
 [ 3  1  0  5  1  3 20  2  3  2]
 [ 3  1  1  6  1  4  5 12  6  1]
 [ 4  7  3  1  5  4  1  5 10  0]
 [ 1  0  2  1  5  3  0  1  6 21]]
              precision    recall  f1-score   support

  subclass 0       0.34      0.45      0.39        40
  subclass 1       0.35      0.40      0.37        40
  subclass 2       0.31      0.23      0.26        40
  subclass 3       0.21      0.20      0.21        40
  subclass 4       0.39      0.55      0.46        40
  subclass 5       0.41      0.38      0.39        40
  subclass 6       0.47      0.50      0.48        40
  subclass 7       0.38      0.30      0.33        40
  subclass 8       0.30      0.25      0.27        40
  subclass 9       0.64      0.53      0.58        40

    accuracy                           0.38       400
```

```
   macro avg       0.38      0.38      0.37       400
weighted avg       0.38      0.38      0.37       400

CPU times: user 3.69 s, sys: 1 ms, total: 3.69 s
Wall time: 3.69 s
```

For Neural Network Generated Features
Mean of the selected performance metric:[0.6765 0.6715 0.6315 0.6765 0.6715 0.6315 0.6765 0.6715 0.6315]
Optimal value of (Gamma,Degree): (0.25, 3)
Output for best (Gamma,Degree) value on Test Set:

```
When gamma= 0.25, degree = 3, Accuracy: 0.7275

Accuracy : 0.7275
Confusion Matrix :
[[31  1  6  2  0  0  0  0  0  0]
 [ 0 30  0  1  2  0  1  1  4  1]
 [10  1 17  8  1  1  1  1  0  0]
 [ 1  0  6 28  3  0  1  0  1  0]
 [ 0  4  0  1 31  0  0  2  2  0]
 [ 0  0  0  0  0 35  2  1  2  0]
 [ 0  1  2  0  0  1 31  3  0  2]
 [ 0  0  2  2  0  0  2 33  1  0]
 [ 0  3  1  0  3  0  1  4 24  4]
 [ 1  1  2  0  0  2  1  0  2 31]]
              precision    recall  f1-score   support

  subclass 0       0.72      0.78      0.75        40
  subclass 1       0.73      0.75      0.74        40
  subclass 2       0.47      0.42      0.45        40
  subclass 3       0.67      0.70      0.68        40
  subclass 4       0.78      0.78      0.78        40
  subclass 5       0.90      0.88      0.89        40
  subclass 6       0.78      0.78      0.78        40
  subclass 7       0.73      0.82      0.78        40
  subclass 8       0.67      0.60      0.63        40
  subclass 9       0.82      0.78      0.79        40

    accuracy                           0.73       400
   macro avg       0.73      0.73      0.73       400
weighted avg       0.73      0.73      0.73       400
```

```
CPU times: user 13 s, sys: 999 µs, total: 13 s
Wall time: 13.1 s
```

Result Discussion:
In order to compare the results for each gamma and C values in the algorithms I have used accuracy performance metric. Because after I have done my search I found out that accuracy is better when there are not imbalance for the classes in the data set. And since we were using the algorithm that we wrote for part 1.3, we were sure that classes are balanced for each fold.

Training hard margin SVM with polynomial kernel gives us better accuracy. And also to increase the accuracy we should use neural network generated features.

Additionally, it was also seen that it takes much more time when we use neural network generated features.

Note: I have give the execution time for training each fold. But I didn't add it to the report since the output for the best gives us the insight of the execution time.

Comparison of Models

Question 1.9

- **Which feature extraction method yields better results.**

For mini-batch gradient ascent algorithm:
    Neural network generated features
For stochastic gradient ascent algorithm:
    Neural network generated features
For full-batch gradient ascent algorithm:
    Neural network generated features
For Soft Margin SVM Model with linear kernel (superclass):
    Neural network generated features
For Hard Margin SVM Model with radial basis function (rbf) kernel (superclass):
    HOG features
For Soft Margin SVM Model with radial basis function (rbf) kernel (superclass):
    Neural network generated features
For Soft Margin SVM Model with radial basis function (rbf) kernel (subclass):
    Neural network generated features
For Hard Margin SVM Model with polynomial kernel (subclass):
    Neural network generated features

- **Which model and feature combination performed best/worst.**

Best:
Soft Margin SVM Model with linear kernel - Neural Network Generated Features
%89.25 Accuracy
Worst:
Hard Margin SVM Model with radial basis function (rbf) kernel - Neural Network Generated Features
%58.75

- **The effect of C on the decision boundary of SVM.**

C effects the margin length of the decision boundary. When C is small it becomes soft margin and when it is high it becomes hard margin. This means it decides on the trade off between errors on training data set and margin maximization.

- **The effect of Gamma on the decision boundary of SVM with RBF kernel.**

Gamma controls the effect of the new features on decision boundary. When the gamma is large, the features will have more influence on decision boundary so it will wiggle more.

- **The effect of d on the decision boundary of SVM with polynomial kernel.**

Higher degree allow a more flexible decision boundary for polynomial kernels.

- **The effects of different (C; Gamma) pairs on the decision boundary and tolerance of SVM.**

Since C changes the margin maximization, and gamma changes the influence of the features to the decision boundary when c is small and gamma is large we cannot have a proper boundary because we cannot have a proper tolerance. Therefore, we need to pick the proper pair.

- **The effects of different (d; Gamma) pairs on the decision boundary of SVM with polynomial kernel.**
- **The effect of batch size to performance and training time of logistic regression models.**

When batch size is too much we have less training time for logistic regression models. However, when we increase the batch size performance generally decreasing for both feature set.
Even though accuracies are generally close to each other when comparing svm and logistic regression, I have obtained better results with the predictions on SVM trained models.
Disadvantage of Logistic Regression:

- Assumption of linearity between variables

Advantages of SVM:

- Works well with unstructured data so we have used different kernels and obtained different results and see how data separate from each other.

- The overfitting is less.
- Better performance faster.

Question 1.10

F1 Score, NPV, FPR, FDR might be better choices when there is an uneven class distribution. For instance we can use F1 score measure to use if we need to seek a balance between Precision and Recall AND there is an uneven class distribution (large number of Actual Negatives).

Accuracy is a reliable metric for both classification tasks because accuracy measure treats all misclassifications the same. And for this task we are not actually focus on the false positive or false negatives. So that for this task we can use accuracy for both subclass and superclass classifications.

Question 2

PCA

Question 2.1 - MSE by SVD based implementation

Output:

```
Calculating Principal components with svd...

Reconstructed image shape: (150, 10625)

MSE for svd based implementation: 0.006597526837140322

CPU times: user 576 ms, sys: 66.7 ms, total: 643 ms
Wall time: 346 ms
```

Result Discussion:
We have obtained very low mean squared error. Which means that the reconstructed image is very close the original ones.

Question 2.2 - MSE by covariance based implementation

Output:

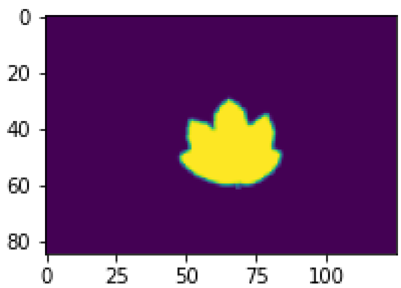
```
Calculating eigenvalues and eigenvalues..
\Reconstructed image shape: (150, 10625)

MSE for cov based implementation: 0.1343346544795466
```

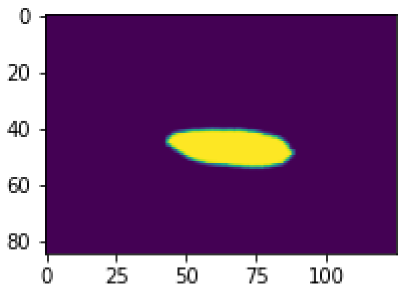
```
CPU times: user 101 ms, sys: 45.1 ms, total: 146 ms
Wall time: 76.6 ms
```

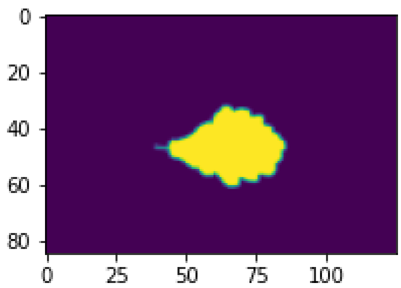
Result Discussion:
We have obtained mean squared error more than SVD based implementation. Which means that the reconstructed image is different than the original images.

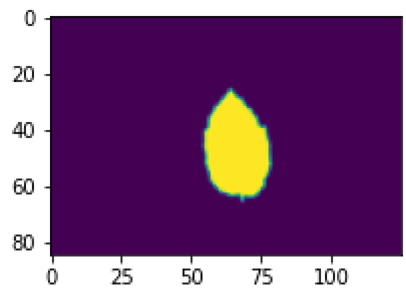
Question 2.3

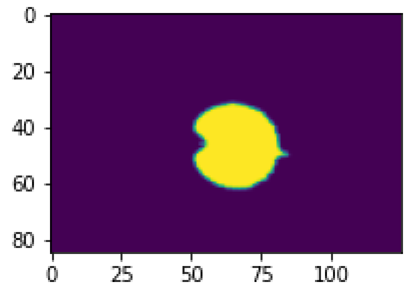
Original Images
Question 2.1)
`Printing original first 5 images from svd based implementation..`





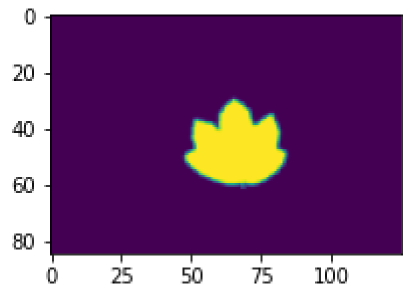


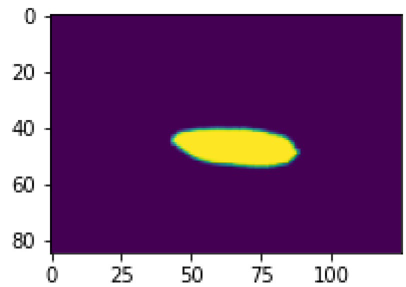


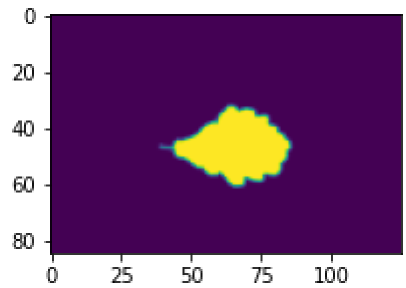


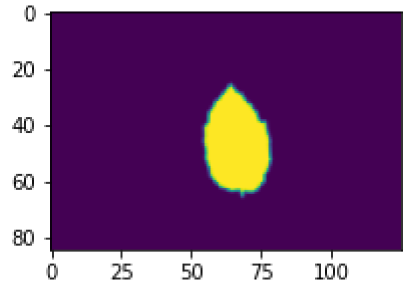
Question 2.2)

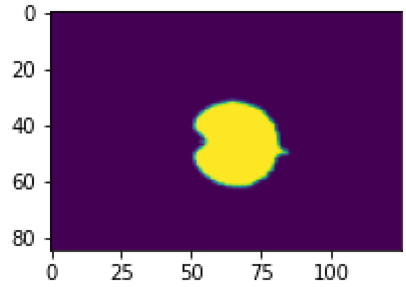
```
Printing original first 5 images from svd based implementation..
```





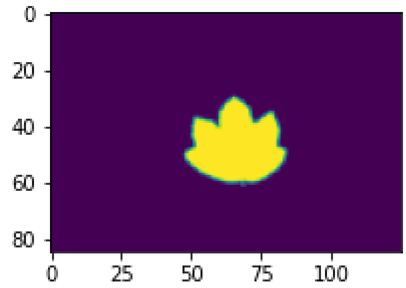


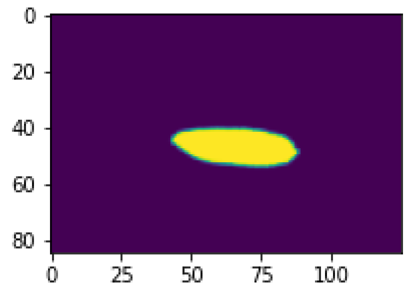


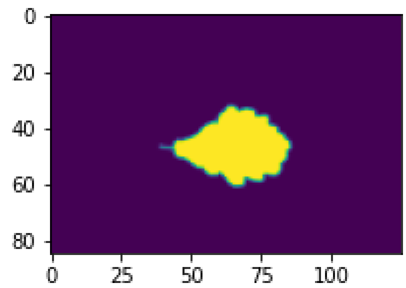


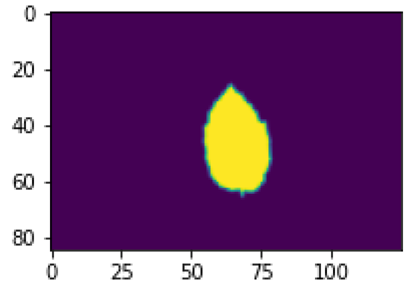
Reconstructed Images
Question 2.1)

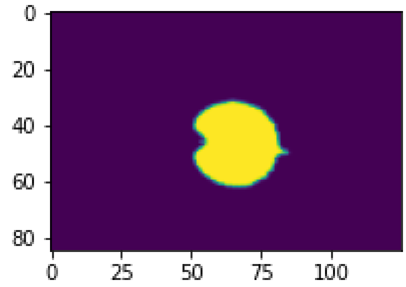
```
Printing reconstructed first 5 images from svd based implementation..
```





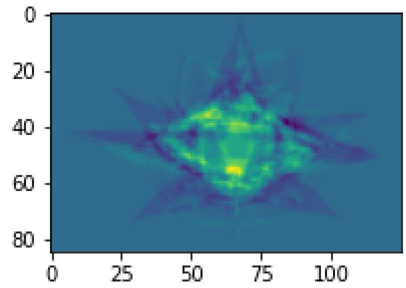


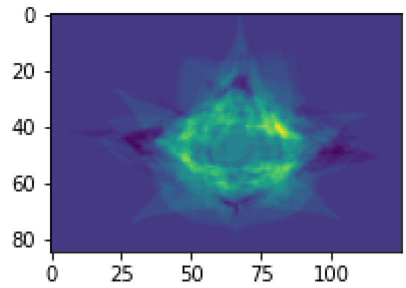


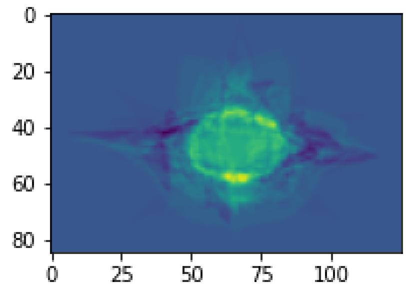


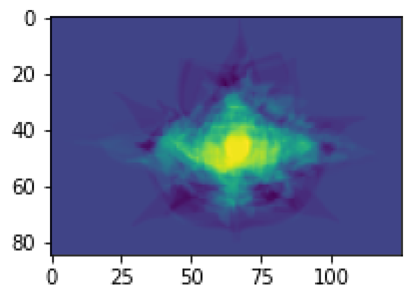
Question 2.2)

```
Printing reconstructed first 5 images from cov based implementation..
```









Result Discussion:
MSE is a reliable metric for comparing image similarity. As it is seen from the images above svd based implementation have less MSE and images are more similar than the ones obtained with the covariance based implementation.

After my research I found out that we can get better result when we use peak signal to noise ratio (PSNR). Which is equal to;

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{\mathrm{Max}^2}{\mathrm{MSE}}\right).$$

Here Max is the maximum possible pixel value of the image. [1]

Question 2.4

Slowest: i
Fastest: ii
So, running time ii > iii > i

Because finding the principal component takes less time if the dataset is more close to the square matrix. If we have a dataset close to the square matrix we can use eigh method from numpy to have substantially decrease in run time. And when we havea matrix like i) , I have done some experience while implementing the homework, it takes more time when we a matrix like iii). It is because finding principal component is faster when we have matrix like iii). So my answer is

ii > iii > i

References

[1] Rajeev Srivastava. 2013. Research Developments in Computer Vision and Image Processing: Methodologies and Applications (1st. ed.). IGI Global, USA.