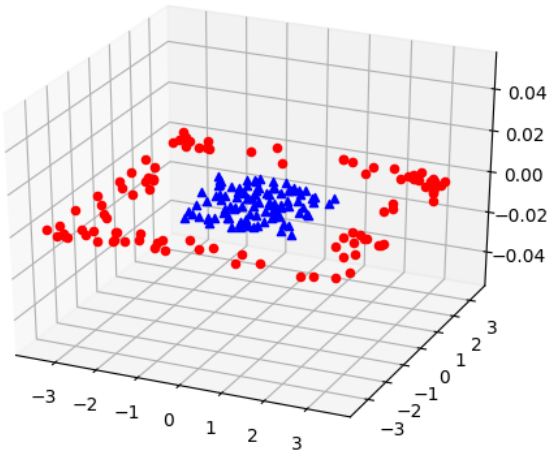
Statistical Methods in AI

Gokul B. Nair
201502034

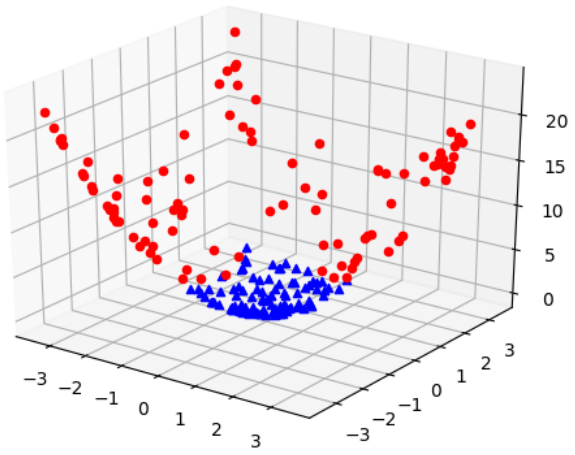
1 Problem 1

1.1 Section 1

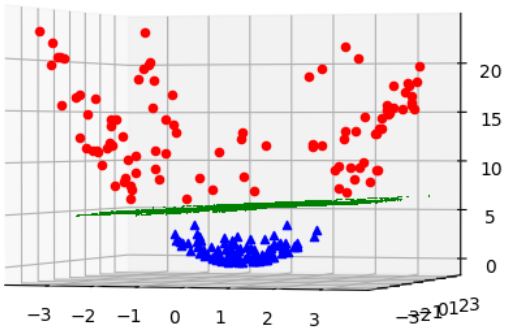
The required code snippets have been written into the file 'kernel_trick_perceptron_submission.py'. The given data corresponds to the following representation:



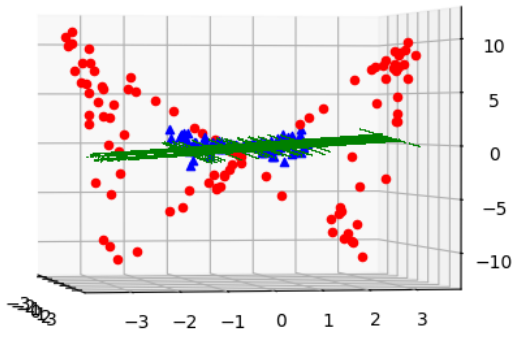
The data clearly is linearly inseperable. I used a kernel defined as $Z=X^2+Y^2$ and obtained the linearly seperable data whose representation is as follows:



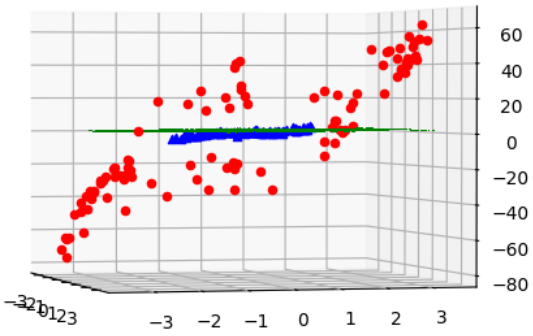
The seperating plane is shown in:



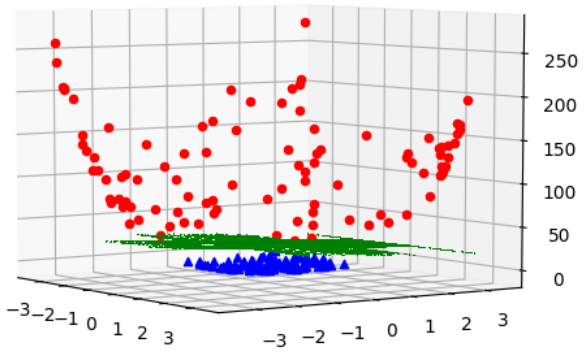
The accuracy obtained here is 1.0.
Using the kernel Z=XY, we get a linearly inseperable set of data which works with accuracy 0.58. The representation is as follows:



Using the kernel $Z=X^3+Y^3$, we again get linearly inseperable set of data with accuracy 0.66. It's representation is as follows:



Using the kernel $Z=X^4+Y^4$, we again get linearly inseperable set of data with accuracy 1.0. It's representation is as follows:



1.2 Section 2

The required code snippets have been written into the file 'letter_classification_svm_submission.py'. I have tried the use of a various kernel types and values of paramter 'C'.
The results are as follows:

kernel type = 'linear' i.e. linear type kernel
– For **C=1.0**, various scores obtained are as follows:

```
Fold1 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.8744
Fold2 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.8690
Fold3 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.8724
Fold4 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.8740
Fold5 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.8698
Fold1 -> Accuracy=0.8419 Precision=0.8439 Recall=0.8401 f1=0.8420
Fold2 -> Accuracy=0.8547 Precision=0.8560 Recall=0.8533 f1=0.8546
Fold3 -> Accuracy=0.8541 Precision=0.8561 Recall=0.8529 f1=0.8545
Fold4 -> Accuracy=0.8475 Precision=0.8498 Recall=0.8457 f1=0.8477
Fold5 -> Accuracy=0.8631 Precision=0.8639 Recall=0.8620 f1=0.8629
```

– For **C=0.01**, various scores obtained are as follows:

```
Fold1 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.8004
Fold2 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.7985
Fold3 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.7988
Fold4 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.7972
Fold5 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.7945
Fold1 -> Accuracy=0.7775 Precision=0.7907 Recall=0.7754 f1=0.7830
Fold2 -> Accuracy=0.7831 Precision=0.7973 Recall=0.7814 f1=0.7893
Fold3 -> Accuracy=0.7844 Precision=0.7961 Recall=0.7823 f1=0.7892
Fold4 -> Accuracy=0.7941 Precision=0.8062 Recall=0.7922 f1=0.7991
Fold5 -> Accuracy=0.7953 Precision=0.8065 Recall=0.7935 f1=0.7999
```

kernel type = 'poly' i.e. polynomial type kernel

– For **C=1.0**, various scores obtained are as follows:

```
Fold1 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.9013
Fold2 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.9020
Fold3 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.8998
Fold4 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.8988
Fold5 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.8997
Fold1 -> Accuracy=0.8728 Precision=0.9078 Recall=0.8720 f1=0.8895
Fold2 -> Accuracy=0.8706 Precision=0.9101 Recall=0.8698 f1=0.8895
Fold3 -> Accuracy=0.8797 Precision=0.9117 Recall=0.8787 f1=0.8949
Fold4 -> Accuracy=0.8756 Precision=0.9058 Recall=0.8744 f1=0.8898
Fold5 -> Accuracy=0.8706 Precision=0.9008 Recall=0.8696 f1=0.8849
```

– For **C=0.1**, various scores obtained are as follows:

```
Fold1 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.7042
Fold2 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.7046
Fold3 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.7023
Fold4 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.6996
Fold5 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.6993
Fold1 -> Accuracy=0.6831 Precision=0.8432 Recall=0.6801 f1=0.7529
Fold2 -> Accuracy=0.6844 Precision=0.8486 Recall=0.6815 f1=0.7559
Fold3 -> Accuracy=0.6878 Precision=0.8439 Recall=0.6851 f1=0.7563
Fold4 -> Accuracy=0.6878 Precision=0.8448 Recall=0.6849 f1=0.7565
Fold5 -> Accuracy=0.6972 Precision=0.8399 Recall=0.6944 f1=0.7603
```

kernel type = 'rbf' i.e. Radial Basis Function kernel

– For **C=1.0**, various scores obtained are as follows:

```
Fold1 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.9558
Fold2 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.9561
Fold3 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.9566
Fold4 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.9571
Fold5 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.9550
Fold1 -> Accuracy=0.9425 Precision=0.9447 Recall=0.9421 f1=0.9434
Fold2 -> Accuracy=0.9359 Precision=0.9393 Recall=0.9352 f1=0.9373
Fold3 -> Accuracy=0.9341 Precision=0.9361 Recall=0.9335 f1=0.9348
Fold4 -> Accuracy=0.9416 Precision=0.9452 Recall=0.9411 f1=0.9431
Fold5 -> Accuracy=0.9366 Precision=0.9383 Recall=0.9360 f1=0.9371
```

– For **C=0.5**, various scores obtained are as follows:

```
Fold1 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.9327
Fold2 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.9319
Fold3 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.9323
Fold4 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.9325
Fold5 -> Number of training samples: 12800 | Number of testing samples: 3200
Training Accuracy: 0.9333
Fold1 -> Accuracy=0.9122 Precision=0.9159 Recall=0.9117 f1=0.9138
Fold2 -> Accuracy=0.9253 Precision=0.9287 Recall=0.9248 f1=0.9268
Fold3 -> Accuracy=0.9184 Precision=0.9219 Recall=0.9177 f1=0.9198
Fold4 -> Accuracy=0.9134 Precision=0.9180 Recall=0.9128 f1=0.9154
Fold5 -> Accuracy=0.9094 Precision=0.9149 Recall=0.9088 f1=0.9118
```

'sigmoid' and 'precomputed' type of kernels were observed to provide bad results for all values of parameter 'C' that I tried. Clearly we see that the RBF Kernel provided best results at C=1.0. The definition for RBF Kernel is as follows:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$$