Assignment 4

**Answer 2**

1. **I have submitted the code as *“Kernel Linear.py”*.**
2. **I tried implementing the given linear kernel with different kernel size in range (5,40) choosing different regularizer weight.**

**The best accuracy was at** kernel size: 30, regularizer weight: 1 = **77.2**

Below are my observations

C:\Users\sohil\AppData\Local\Enthought\Canopy\User\python.exe D:/Downloads/Kernel1.py

Accuracy for kernel size: 5, regularizer weight: 0.001 = 76.6

Accuracy for kernel size: 5, regularizer weight: 0.01 = 76.6

Accuracy for kernel size: 5, regularizer weight: 0.1 = 76.0666666667

Accuracy for kernel size: 5, regularizer weight: 1 = 76.9333333333

Accuracy for kernel size: 10, regularizer weight: 0.001 = 76.6

Accuracy for kernel size: 10, regularizer weight: 0.01 = 76.6

Accuracy for kernel size: 10, regularizer weight: 0.1 = 76.4

Accuracy for kernel size: 10, regularizer weight: 1 = 76.6

Accuracy for kernel size: 15, regularizer weight: 0.001 = 76.6

Accuracy for kernel size: 15, regularizer weight: 0.01 = 76.6

Accuracy for kernel size: 15, regularizer weight: 0.1 = 76.7333333333

Accuracy for kernel size: 15, regularizer weight: 1 = 76.6666666667

Accuracy for kernel size: 20, regularizer weight: 0.001 = 76.6

Accuracy for kernel size: 20, regularizer weight: 0.01 = 76.6

Accuracy for kernel size: 20, regularizer weight: 0.1 = 76.5333333333

Accuracy for kernel size: 20, regularizer weight: 1 = 77.1333333333

Accuracy for kernel size: 25, regularizer weight: 0.001 = 76.6

Accuracy for kernel size: 25, regularizer weight: 0.01 = 76.6

Accuracy for kernel size: 25, regularizer weight: 0.1 = 76.4

Accuracy for kernel size: 25, regularizer weight: 1 = 77.0666666667

Accuracy for kernel size: 30, regularizer weight: 0.001 = 76.6

Accuracy for kernel size: 30, regularizer weight: 0.01 = 76.6

Accuracy for kernel size: 30, regularizer weight: 0.1 = 76.6666666667

Accuracy for kernel size: 30, regularizer weight: 1 = 77.2

Accuracy for kernel size: 35, regularizer weight: 0.001 = 76.6

Accuracy for kernel size: 35, regularizer weight: 0.01 = 76.6

Accuracy for kernel size: 35, regularizer weight: 0.1 = 76.6

Accuracy for kernel size: 35, regularizer weight: 1 = 76.8666666667

Accuracy for kernel size: 40, regularizer weight: 0.001 = 76.6

Accuracy for kernel size: 40, regularizer weight: 0.01 = 76.6

Accuracy for kernel size: 40, regularizer weight: 0.1 = 76.4666666667

Accuracy for kernel size: 40, regularizer weight: 1 = 76.9333333333

**(C)** I have tried three different kernels on the same dataset namely - Linear, Polynomial, Gaussian and written down the results in file ‘*Kernel Comparision Results.txt”.* The choice of a Kernel depends on the problem at hand because it depends on what we are trying to model. My main observations when I ran all the three kernels for same dataset are –

* The results of accuracy was best for Gaussian, second for Linear and last for Polynomial.
* The Gaussian took noticeably more time for execution, Linear and Polynomial time execution was not noticeable
* I observed huge variation in the accuracies of Gaussian Kernel, the others gave quite consistent accuracies for different settings.

Upon running the kernels separately, I tried optimizing the accuracy for Gaussian kernel by different settings of the constant term sigma and noted my observations in file ‘*Gaussian Kernel.txt’*

* Performed very badly with very low and high values of sigma
* On the given dataset, it performed badly with higher regularizer weight of more than 0.1
* I received very good accuracy of **79.0** after some parameter tuning at regularizer weight: 0.001, sigma: 6

Other theoretical concepts I came across about the different kernels –

* A [polynomial](http://en.wikipedia.org/wiki/Polynomial) kernel allows us to model feature conjunctions up to the order of the polynomial.
* Radial basis functions allows to pick out circles (or hyperspheres) – in contrast with the Linear kernel, which allows only to pick out lines (or [hyperplanes](http://en.wikipedia.org/wiki/Hyperplane)).
* The Rational Quadratic kernel is less computationally intensive than the Gaussian kernel and can be used as an alternative when using the Gaussian becomes too expensive