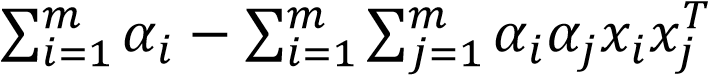
Data Conversion:

* The file was parsed to obtain the entire dictionary of words.
* Then each mail was parsed and a feature vector corresponding to the mail containing the whole dictionary with frequencies was constructed. A column vector was constructed with 1 for not spam and -1 for spam.

A python script was written to do data processing which give output in the

corresponding files which were loaded by matlab.

a) **Linear kernel**

Dual SVM Problem to optimize: 

𝑄(𝑖, 𝑗) = 𝑥𝑖𝑥𝑗𝑇

C=0 and a is the alpha

Criteria for choosing alpha’s for support vectors:

𝛼𝑖 > 0.0001

This condition is necessary as due to some floating point all 𝛼𝑖′𝑠 are greater than 0 and majority are of the order of 10^-5 to 10^-6 .

By enforcing the above criteria the no. of support vectors obtained is **151 for train small** and **452 for train**

The set of email number which are support vector are in Support Vector ID folder for both train and train-small.

The average accuracy reported was **91.3%** when **train-small** was used to train the

SVM and **98.7%** when **train** was used to train the SVM

b) **Gaussian Kernel**

𝑄(𝑖, 𝑗) = exp⁡(−𝛾 ∗ (||𝑥𝑖 − 𝑥𝑗 ||^2)

The no. of support vectors obtained in through cvx is **536** when trained using **trainsmall** dataset and **1996** when trained using **train** dataset.

The accuracy obtained is **89.3 %** for **train-small** and **97.7%** when SVM was trained using **train** dataset.

The accuracies in Gaussian case are slightly less than linear kernel case. This is due to the fact that the data is linearly separable as none of the 𝛼𝑖′𝑠 are exactly equal to C (i.e. 1 in our case). This implies zi’s are zero for every example. As the linear kernel does not change the space and since the data is separable, a better separation is learnt in linear case. Gaussian kernel transforms the space to infinite dimension and therefore it is not able to utilize the linear separability in the linear space and tires to classify using exponential space. And also gaussian has large space and the dataset is small, so gaussian over fits the training data, which results in poorer accuracy.

c) LibSVM:

LibSVM for linear case:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.no.** | **Accuracy** | **No. Support**  **Vectors** | **Kernel** | **Dataset** |
| **1** | 91.3% | 152 | Linear | train-small |
| **2.** | 89.3% | 530 | Gaussian | train-small |
| **3.** | 98.7 | 452 | Linear | train |
| **4.** | 98.7 | 1982 | Gaussian | train |

The accuracy of linear and gaussian obtained through libSVM are in agreement with those obtained through cvx.

The no. of support vectors are almost same. Though time taken by libSVM is much smaller than CVX package. This may be due to the fact that libSVM is customized to optimize SVM problems but cvx is a general purpose which lacks special SVM specific optimizations. The cvx code takes 13-15 minutes to run for train-small and hours for train whereas libSVM outputs comes within 2-3 minutes for both trainsmall and train.