ELL409   
Assignment 2

Vansh Gupta  
2019EE10143

**Appendix:**

* CVXOPT linear kernel derivation:

**Part 1A**

* **Binary Classification**:
* **Linear Kernel**
* Note, here’s my understanding of the parameter C, based on [this](https://stats.stackexchange.com/questions/31066/what-is-the-influence-of-c-in-svms-with-linear-kernel): *The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points. For very tiny values of C, you should get misclassified examples, often even if your training data is linearly separable.* i.e., it kind of acts as a regularization constant

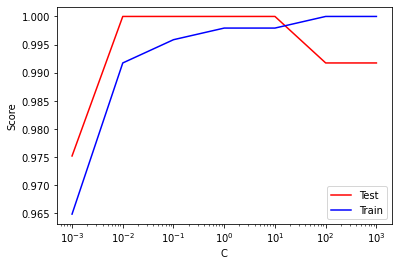
First, I take the linear kernel, for which, the only parameter is C (Because the degree is introduced in the separate poly kernel). Further, I first consider only 1st 10 features for this sub-part and report the results in table 1.

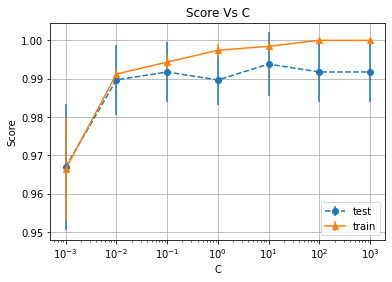
**10 features**:

I take first 10 features, split the dataset in 4:1 training:test set and run a 5-fold CV on the training set for the results. The plots are of 2 types. First is the score (mean accuracy) for different values of C, and the second is an error bar for the mean score for the 5-fold CV, spanning the standard deviation about mean

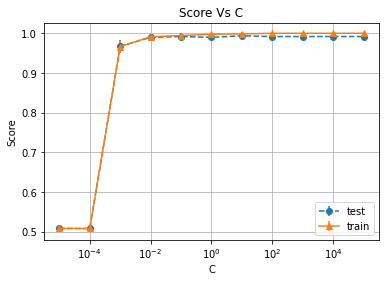
-----------------------------------------------------------------------------------------

Classes: 0 & 1  
Training size: 484 instances  
Test size: 121 instances  
Grid search returned: {'SVM\_\_C': 10.0}  
Training Accuracy: 99.8%  
Test Accuracy: 100%

  
Fig 1. Score vs regularization constant, giving best performance at C=10

  
Fig 2. Score vs regularization constant averaged over different cross validation sets

For the range that I took, one can hardly see any over-fitting, but if I increase the range to , it becomes slightly more evident:



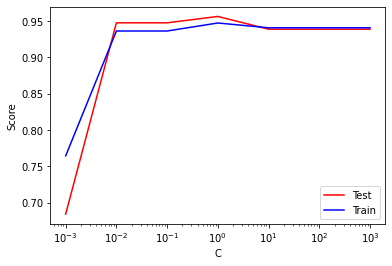
Weights: [-3.12930875, -1.66418532, -0.4952001, 0.69124859, 0.64139166, 0.72032653, 0.925925, -0.61276059, 1.27476009, -1.50301441]  
Bias: [-3.06011821]

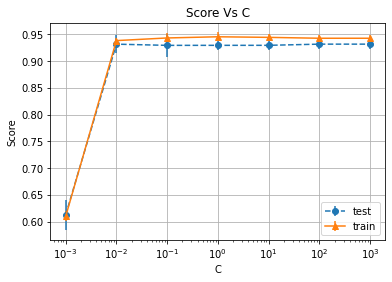
Further, setting C=10, I get the following output for the **cvxopt**:

Weights: [-3.12925705, -1.66458371, -0.49527914, 0.6912746, 0.64136747, 0.72033966, 0.92583145, -0.61279811, 1.27460542, -1.50272523]  
Bias: [-3.06034436]

-----------------------------------------------------------------------------------------

Classes: 8 & 9  
Training size: 454 instances  
Test size: 114 instances  
Grid search returned: {'SVM\_\_C': 100}  
Training Accuracy: 94.06%  
Test Accuracy: 93.86%

  
Fig 3. Score vs regularization constant, giving best performance at C=100

  
Fig 4. Score vs regularization constant averaged over different cross validation sets

We can see that even though the *best* performance is indeed at 100, C=1 also gives almost equally good performance

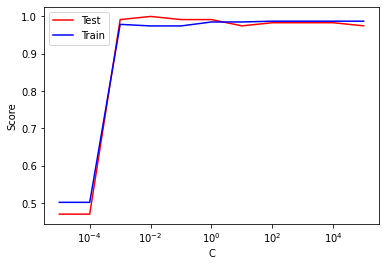
Weights: [-0.33498158, 1.13224457, -0.04446576, -0.04601712, -0.32856846, -0.39540411, -0.20898698, 0.60852265, 0.0013532, -0.41773494]  
Bias: [-0.54310417]

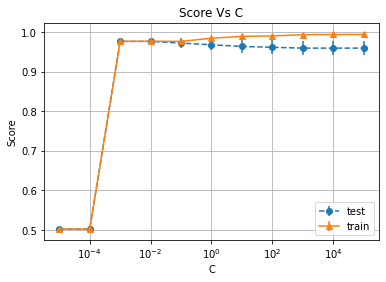
Further, setting C=1, I get the following output for the **cvxopt**:

Weights: [-0.3350519, 1.13316787, -0.04413581, -0.04584269, -0.32911689, -0.39607512, -0.20949239, 0.6096233, 0.00117827, -0.41821986]  
Bias: [-0.4928271]

-----------------------------------------------------------------------------------------

Classes: 4 & 6  
Training size: 472 instances  
Test size: 119 instances  
Grid search returned: {'SVM\_\_C': 0.001}  
Training Accuracy: 97.88%  
Test Accuracy: 99.16%

  
Fig 5. Score vs regularization constant, giving best performance at C=0.001

  
Fig 6. Score vs regularization constant averaged over different cross validation sets

Weights: [ 0.07905413, -0.26257078, 0.06143682, 0.17693943, 0.02516208, 0.16061036, 0.20336535, -0.03289297, -0.06398543, 0.00285935]  
Bias: [-0.08674833]

Further, setting C=0.001, I get the following output for the **cvxopt**:

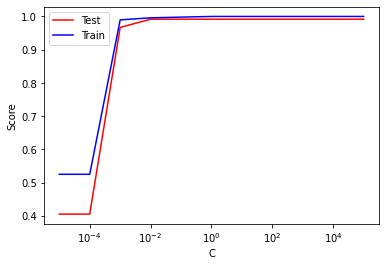
Weights: [.07906718, -0.26258586, 0.06141043, 0.17694032, 0.02520292, -0.16053697, 0.20330742, -0.0328616, -0.06395126, 0.00291391]  
Bias: [1.24360299]

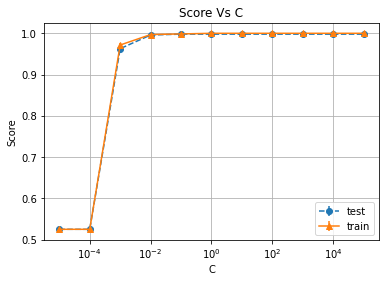
**25 features**:

I take all the features, split the dataset in 4:1 training:test set and run a 5-fold CV on the training set for the results. The plots are of 2 types. First is the score (mean accuracy) for different values of C, and the second is an error bar for the mean score for the 5-fold CV, spanning the standard deviation about mean

-----------------------------------------------------------------------------------------

Classes: 0 & 1  
Training size: 484 instances  
Test size: 121 instances  
Grid search returned: {'SVM\_\_C': 0.1}  
Training Accuracy: 99.8%  
Test Accuracy: 99.17%

  
Fig 7. Score vs regularization constant, giving best performance at C=0.1 and 0.01

  
Fig 8. Score vs regularization constant averaged over different cross validation sets

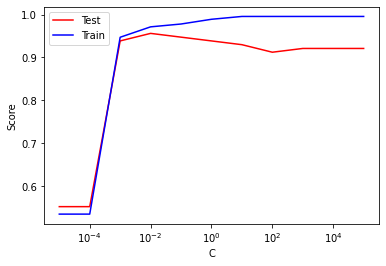
Weights: [-0.29205499, -0.05850226, -0.02846066, 0.02515652, 0.052591, 0.00659001, -0.00351356, -0.03230547, 0.00347567, 0.02422671, 0.02003441, 0.03849303, 0.05418999, 0.02769126, -0.00719097, -0.01669045, -0.02576264, -0.01899277, 0.02262335, 0.00216516, 0.01317834, 0.00493513, 0.0269242, -0.00819326, -0.01116176]  
Bias: [-0.0389515]

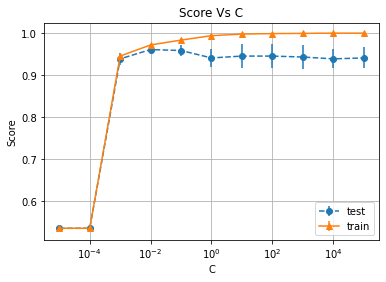
Further, setting C=0.001, I get the following output for the **cvxopt**:

Weights: [-0.29202605, -0.05852558, -0.02842422, 0.02509217, 0.0526775 0.006627 -0.00351456, -0.03229368, 0.00348714, 0.02423127, 0.02001002, 0.03851164, 0.054184, 0.0277176, -0.00718686, -0.01667696, -0.02578883, -0.01898507, 0.02264766, 0.00218381, 0.01319485, 0.00492693, 0.02690514, -0.00820047, -0.01114062]  
Bias: [0.70454935]

-----------------------------------------------------------------------------------------

Classes: 8 & 9  
Training size: 454 instances  
Test size: 114 instances  
Grid search returned: {'SVM\_\_C': 0.01}  
Training Accuracy: 97.14%  
Test Accuracy: 95.61%

  
Fig 9. Score vs regularization constant, giving best performance at C=0.01/0.001

  
Fig 10. Score vs regularization constant averaged over different cross validation sets

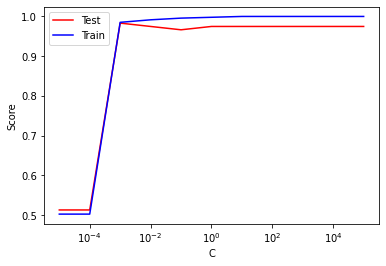
Weights: [-0.05672124, 0.32831456, -0.02751964, 0.02545539, -0.09309066, -0.06510461, 0.01307934, 0.0714284, 0.04263766, -0.02549699, -0.02728191, 0.01993879, -0.16382988, 0.05054793, 0.05283242, -0.09205375, 0.00899802, 0.00502964, -0.01434225, -0.02699369, 0.03141922, -0.02285253, 0.01309503, 0.02937307, -0.00308258]  
Bias: [-0.08552984]

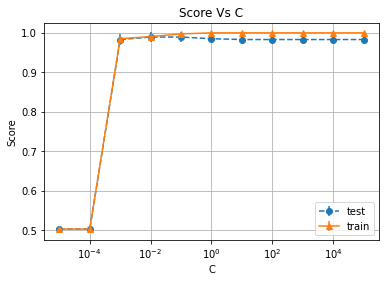
Further, setting C=0.01, I get the following output for the **cvxopt**:

Weights: [-0.07257913, 0.51561707, -0.00123075, 0.00158239, -0.19541067, -0.13615541, 0.00204019, 0.15456693, 0.08936978, -0.10376137, -0.07371073, -0.00554385, -0.2852563, 0.09978655, 0.11380332, -0.28929088, -0.04582913, 0.0021046, -0.09327269, -0.03354809, 0.0550451, -0.20541925, -0.08353213, 0.08993784, 0.02134263]  
Bias: [-0.4928271]

-----------------------------------------------------------------------------------------

Classes: 4 & 6  
Training size: 472 instances  
Test size: 119 instances  
Grid search returned: {'SVM\_\_C': 0.01}  
Training Accuracy: 99.15%  
Test Accuracy: 97.48%

  
Fig 11. Score vs regularization constant, giving best performance at C=0.01/0.001

  
Fig 12. Score vs regularization constant averaged over different cross validation sets

Weights: [0.11717243, -0.40116018, 0.11632333, 0.30692205, 0.03411372, -0.23536823, 0.33034117, -0.01774847, -0.13541525, 0.00437062, -0.10100332, 0.04296125, 0.0747433, 0.02496386, -0.02533199, -0.00231244, 0.26827029, 0.02816113, 0.02320037, 0.05914415, 0.07584339, -0.01022711, 0.06327853, 0.04430409, -0.08353974]  
Bias: [-0.14984723]

Further, setting C=0.01, I get the following output for the **cvxopt**:

Weights: [0.11715816, -0.40104701, 0.11632326, 0.30693711, 0.03432163, -0.23537957, 0.33015983, -0.01780562, -0.13548682, 0.00437479, -0.10104125, 0.04313272, 0.07477161, 0.02475069, -0.02502624, -0.00231084, 0.2683483, 0.02808194, 0.02324129, 0.05937107, 0.07556977, -0.01031471, 0.06317371, 0.04427912, -0.08349651]  
Bias: [-0.31779591]

* **Poly Kernel**
* Note, here’s my understanding of the parameter C, based on [this](https://stats.stackexchange.com/questions/31066/what-is-the-influence-of-c-in-svms-with-linear-kernel): *The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points. For very tiny values of C, you should get misclassified examples, often even if your training data is linearly separable.* i.e., it kind of acts as a regularization constant

First, I take the linear kernel, for which, the only parameter is C (Because the degree is introduced in the separate poly kernel). Further, I first consider only 1st 10 features for this sub-part and report the results in table 1.

**10 features**: