Assignment-2
ELL 409

Jaskeerat Singh Saluja (2019MT60752)

October 25, 2021

Code Overview

SVM class has been implemeted in 4 different ways

1. **SVM_LIBSVM** : Uses LIBSVM pacakge , supports function fit ,pred ,score ,describe.
2. **SVM_CVX** : using CVXOPT packages for solving dual problem of SVM.
3. **SVM_SMO_SIMPLE** : SVM optimization using **Simple SMO of cs229 lecture notes**
4. **SVM_SMO_FULL** :SVM optimization using **John platt's paper**

Note

1. All of above implementation are present as separate files in **utils** directory
2. Data normalization is performed automatically in all above classes when fit is called. MINMAX normalization is used which is implemeted in the normalize.py in utils directory.

1. Other directory are P1 and P2 , which contains .ipynb file where the Part 1 and part-2 problems are solved and anlysed .

Feeding into CVXOPT

Format required : $\frac{1}{2}x^TPx + q^Tx, Gx <= h, Ax = b$, required P,q,G A,b and h
dual problem : $\min(\alpha) = \alpha^T(yy^TK(X))\alpha - e^T\alpha$, s.t $\alpha^Ty = 0$ and $\forall i 0 \leq \alpha_i \leq C$

1. K(X) : Kernel of matrix X
2. $P = yy^TK(X)$
3. $q = -e^T$
4. $A = Y^T, b = 0$
5. $g1 = -e^T, b1 = 0$
6. $g2 = e^T, b2 = C$
7. G = g1 and g2 stacked vertically , b : b1 and b2 stacked vertically

The above is fed to **CVXOPT** package to get optimal value of α

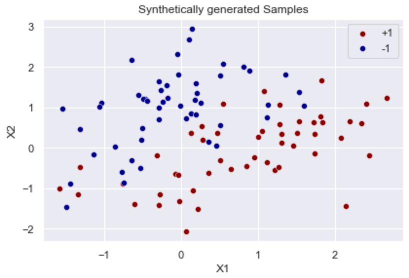
1. get support vectors : $SV = alpha > 1e^{-5}$
2. If linear kernel : $w = \sum_{\forall i \in sv} y_i\alpha_i sv_i$

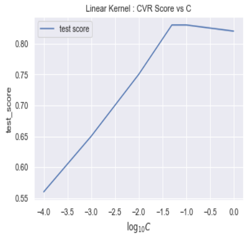
Table of Content

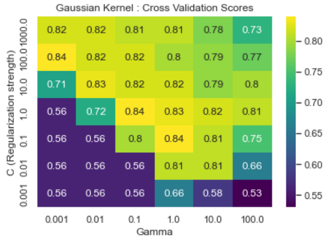
Linearly Separable (Overlapping Data)

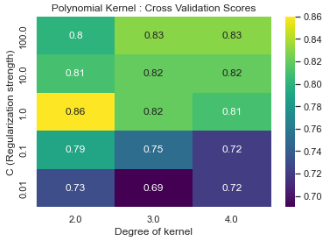
Taking Synthetically generated distribution of 2d data with partial Overlaps.

1. RUN SVM classifier using differrent **Kernels**
2. Hyperparameter tuning using **Grid Search CV** . (Implemented own version)
3. Visaulizing cases of **overfit and underfitting**



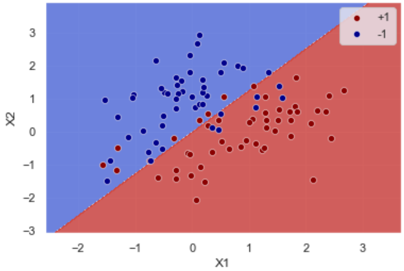




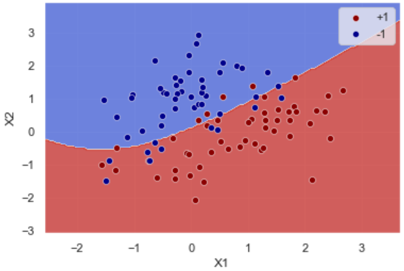


Surface boundry visualization

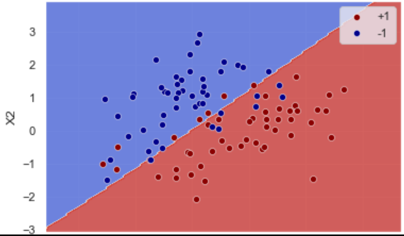
Descion boundry Linear SVM



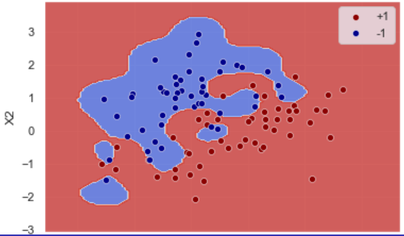
Descion boundry Polynomial(kernel),degree =2 C=1.0



Descion boundry rbf(gaussian) kernel,γ =0.1 C=1.0



Overfitting rbf(gaussian) kernel,γ =100 C=1000.0



Observation of the Synthetic 2d Data

1. Higher values of γ cause the rbf kernel svm to **overfit**
2. Best Hyperparameter Configuration are present in the **opposite diagonal** of the heatmap
3. C **regularization cofficient** : low value of C implies underfitting , higher values of C cause overfitting

$$\min \frac{1}{2}||w||^2 + C\sum \alpha_i$$

High values of C cause w to dominate , hence the optimzed solution cause high w and hence low margin . For low value of C , we have opposite scenario

Binary Classification

Problem Statment

1. Select Any two classes out of 10 , and create a binary classifier.
2. **Chosen classes number 1 and 4**
3. Try tuning the model with different Hyperparameter for various kernels.
4. Interested Kernels are **gaussian(rbf), linear, polynomial**

Hyperparameter tuning

1. Using **GridSearchCV** class implemeted in **search.py**. (My own implementation)
2. Since for each kernel there are **atmost 2 hyperparamters** thus using **HEATMAP** analysis we get the best hyperparamters for our binary Classification problem

Binary Classification :(rbf)-10 features

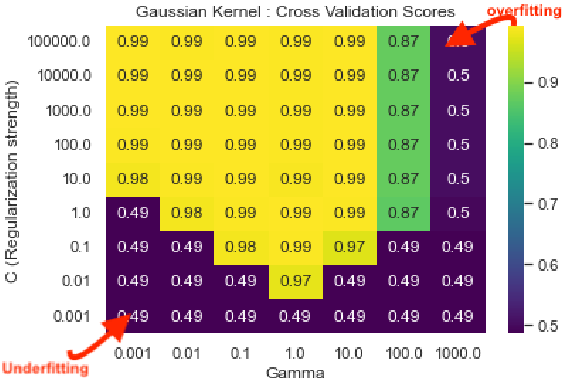
On right is the **HeatMap** for cross validation scores , for Interested range of $(\gamma, \mathcal{C})$. Calculated use **LIBSVM** python package.

Best hyperparamters

1. $\gamma = 0.1$ and $\mathcal{C} = 1.0$
2. CV Score $= 0.9909318$

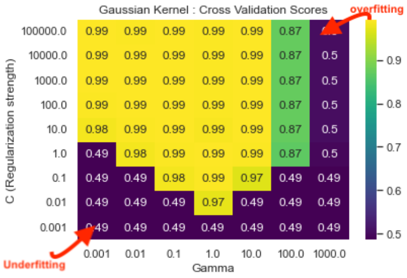
CV Score from SVM-CVXOPT

1. SVM-CVXOPT : SVM impl using convex opt package
2. CV Score 0.99094326

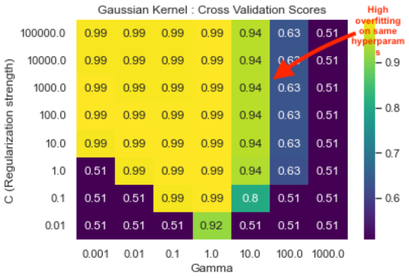


C \ Gamma	0.001	0.01	0.1	1.0	10.0	100.0	1000.0
100000.0	0.99	0.99	0.99	0.99	0.99	0.87	[illegible]
10000.0	0.99	0.99	0.99	0.99	0.99	0.87	0.5
1000.0	0.99	0.99	0.99	0.99	0.99	0.87	0.5
100.0	0.99	0.99	0.99	0.99	0.99	0.87	0.5
10.0	0.98	0.99	0.99	0.99	0.99	0.87	0.5
1.0	0.49	0.98	0.99	0.99	0.99	0.87	0.5
0.1	0.49	0.49	0.98	0.99	0.97	0.49	0.49
0.01	0.49	0.49	0.49	0.97	0.49	0.49	0.49
0.001	0.49	0.49	0.49	0.49	0.49	0.49	0.49

Binary Classification :(rbf)-25 features



10 features



25 features

Observation

1. Due to more number of features , **The model starts overfitting** soon with high value of γ and C.
2. Look for $\gamma = 100$

Binary Classification :(linear) -10 features

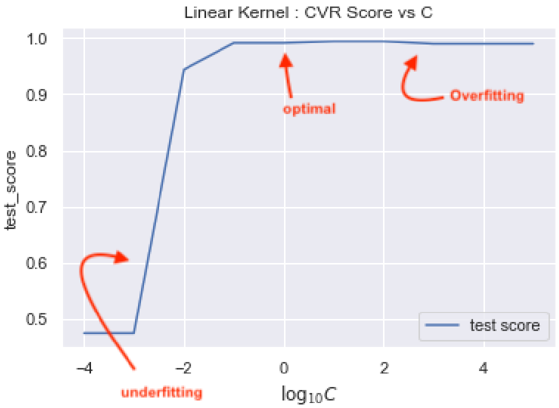
On right is the **PLOT** for cross validation scores , for Interested range of ($\mathcal{C}$). Calculated use **LIBSVM** python package.

Best hyperparamters

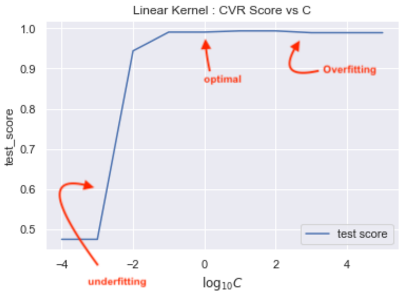
1. $\mathcal{C} = 1.0$
2. CV Score $= 0.9909318$

CV Score from SVM-CVXOPT

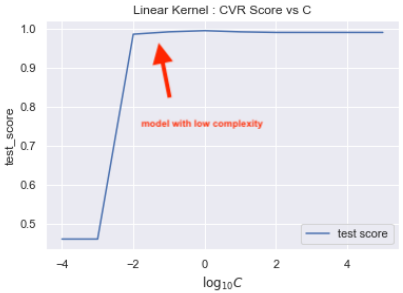
1. SVM-CVXOPT : SVM impl using convex opt package
2. CV Score 0.99092048



Binary Classification :(linear)-25 features



10 features



25 features

Observation

1. Due to more number of features , **The model starts overfitting** soon with high value of γ and C.
2. Thus C=1 worked 10 feature case , not C=0.1 works the same for 25 features.

Binary Classification :(polynomial) -10 features

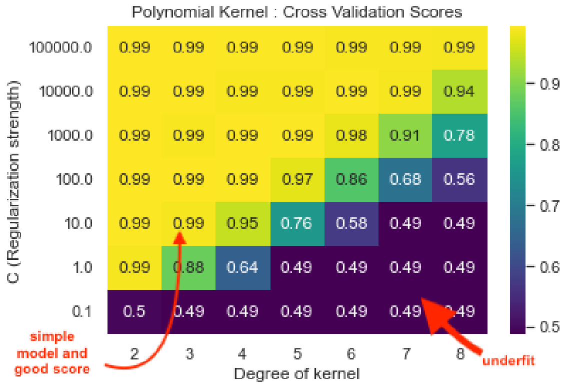
On right is the **PLOT** for cross validation scores , for Interested range of $(degree, \mathcal{C})$.
Calculated use **LIBSVM** python package.

Best hyperparamters

1. $\mathcal{C} = 10$
2. $degree = 3$
3. CV Score $= 0.99244$

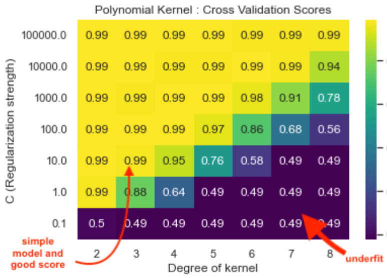
CV Score from SVM-CVXOPT

1. SVM-CVXOPT : SVM impl using convex opt package
2. CV Score 0.9924584

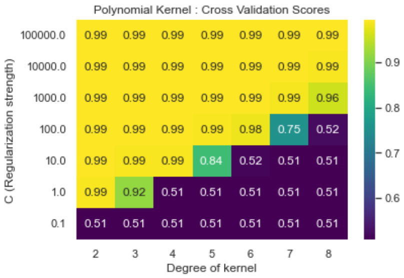


	2	3	4	5	6	7	8
100000.0	0.99	0.99	0.99	0.99	0.99	0.99	0.99
10000.0	0.99	0.99	0.99	0.99	0.99	0.99	0.94
1000.0	0.99	0.99	0.99	0.99	0.98	0.91	0.78
100.0	0.99	0.99	0.99	0.97	0.86	0.68	0.56
10.0	0.99	0.99	0.95	0.76	0.58	0.49	0.49
1.0	0.99	0.88	0.64	0.49	0.49	0.49	0.49
0.1	0.5	0.49	0.49	0.49	0.49	0.49	0.49

Binary Classification :(polynomial)-25 features



10 features

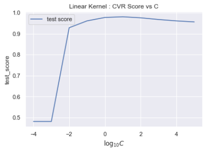


25 features

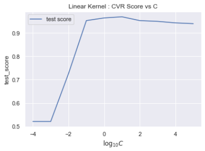
Observation

1. Configurations where **polynomial kernel** had underfitting with 10 features , some of them now have a good fit with increased number of features i.e 25.
2. There is **No overfitting** in polynomial kernel with given data set
3. Increase in feature cause shift of underfit along the primary diagonal of the HEATMAP

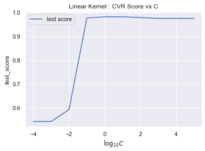
Binary Classification:(linear) More pair of classes



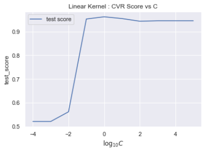
classes 2,7



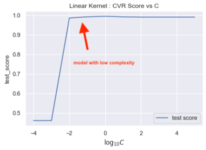
classes 3,9



classes 4,8



classes 5,6

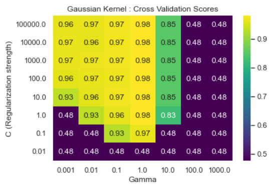


classes 1,4

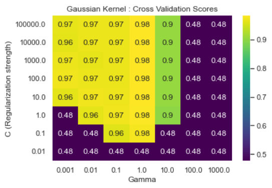
Observation

1. **Same Hyperparameter i.e C=0.1** works for all above pair of classes
2. **Got consistently Best result** with same hyperparamters.

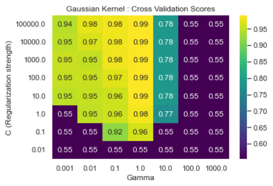
Binary Classification:(gaussian) More pair of classes



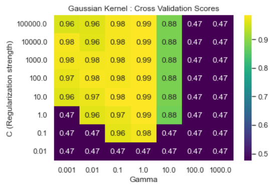
classes 3,9



classes 2,4



classes 2,5



classes 6,8

Observation

1. **Got consistently Best result** with same hyperparamters, evem with gaussian kernel
2. $\gamma = 0.1, C = 1$

Multi-Class Classification

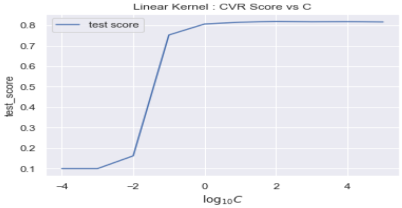
Problem Statment

1. Using LIBSVM implemeted in SVMLIBSVM.py
2. Try tuning the model with different Hyperparameter for various kernels.
3. Interested Kernels are **gaussian(rbf), linear, polynomial**

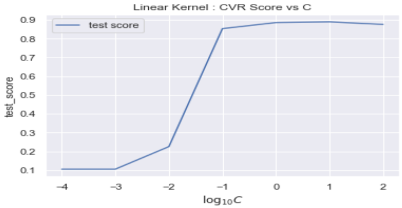
Hyperparameter tuning

1. Using **GridSearchCV** class implemeted in **search.py**. (My own implementation)
2. Since for each kernel there are **atmost 2 hyperparamters** thus using **HEATMAP** analysis we get the best hyperparamters for our binary Classification problem

Multilclass Classification :(linear)



Feature =10 , best cv-score (0.8193)

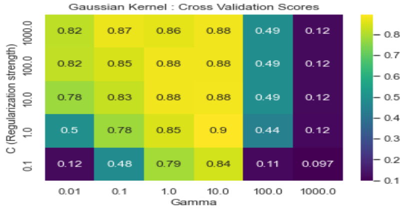


Feature =25 , best cv-score (0.889)

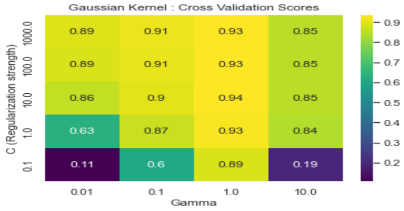
Observation

1. **Best hyperparamters remains same** as they were in case of binary clf.
2. The **best cv-score(0.8193)** in 10-feature case and **0.889** in 25-features case , is less as compared to binary clf(around **0.99**)
3. **More number of features**(25 feature case) **generalizes** the multiclass desicion boundry.

Multilclass Classification :(rbf)



Feature =10 , best cv-score (0.8953)



Feature =25 , best cv-score (0.9445)

Observation

1. **Best hyperparamters remains same** as they were in case of binary clf.
2. The **best cv-score(0.8952)** in 10-feature case and **0.9445** in 25-features case , is less as compared to binary clf(around **0.99**)
3. **More number of features**(25 feature case) **generalizes** the multiclass desicion boundry.

Notice

- Training time of the SVM model is very highly sensitive to values of C . thus while performing GridSearch for 25 feature case , a smaller Hyperparameter space is used.

Multiclass classification :Conclusion

1. Best score of 0.9445 with kernel 'rbf' which is quite less compared to binary Classification

One vs One Classification -LIBSVM

1. One-vs-One is another heuristic method for using binary classification algorithms for multi-class classification.
2. one-vs-one splits a multi-class classification dataset into binary classification problems
3. There are $n(n-1)/2$ models trained
4. Each binary classification model may predict one class label and the model with the most predictions or votes is predicted by the one-vs-one strategy.

SMO

SMO general algorithm

1. Find a Lagrange multiplier α_1 that violates the Karush–Kuhn–Tucker (KKT) conditions for the optimization problem.
2. Pick a second multiplier α_2 and optimize the pair (α_1, α_2).
3. Repeat steps 1 and 2 until convergence.

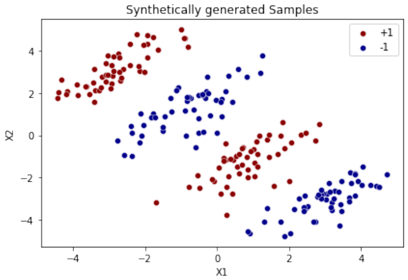
Note

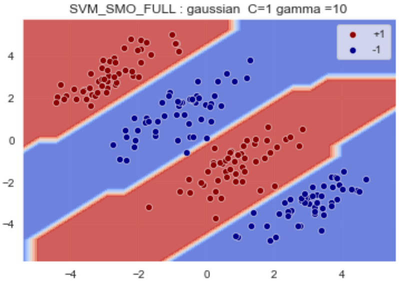
Although this algorithm is guaranteed to converge, heuristics are used to choose the pair of multipliers so as to accelerate the rate of convergence. This is critical for large data sets since there are $\frac{n(n-1)}{2}$ possible choices for α_i and α_j.

Simplified SMO

Implemented from CS229 lecture notes.

Training SVM on non-linear case





Observation

1. Since the simplified SMO uses random heuristics while choosing pairs of alpha thus theoritically we cannot proove convergence of this Algorithm.
2. Simplified SMO is faster than CVXOPT .

SMO Complete- Choosing heuristics

First Choice heuristics

1. As the SMO algorithm progresses, examples that are at the bounds are likely to stay at the bounds, while examples that are not at the bounds will move as other examples are optimized.
2. First langrange multiplier is chosen out of unbounded examples

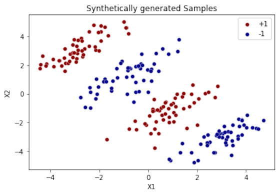
Second choice heuristics

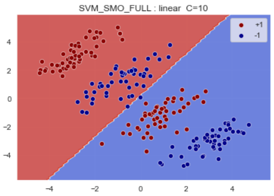
1. Once a first Lagrange multiplier is chosen, SMO chooses the second Lagrange multiplier to maximize the size of the step taken during joint optimization
2. SMO keeps a cached error value E for every non-bound point in the training set and then chooses an error to approximately maximize the step size
3. If E1 is positive, SMO chooses an point with minimum error E2. If E1 is negative, SMO chooses an point with maximum error E2.
4. if SMO cannot make positive progress using the second choice heuristic
    1. then SMO starts iterating through the non-bound examples, searching for an second example that can make positive progress
    2. If none of the non-bound examples make positive progress, then SMO starts iterating through the entire training set until an example is found that makes positive progress
    3. In extremely degenerate circumstances, none of the examples will make an adequate second example. We skip the first langrange multiplier

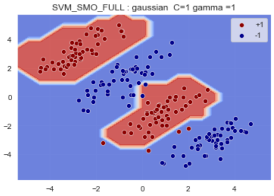
SMO Complete

The code for SMO was implemeted Using the pseudo code in John platt paper . The class is implemeted in SVM_SMO_FULL.py
[, Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines , John Platt]

Training SVM on non-linear case







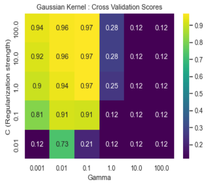
Observation

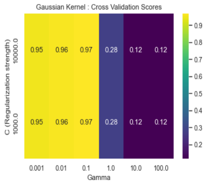
1. My implementation of SMO was faster than SVM_CVX , but slower in run time than standard LIBSVM

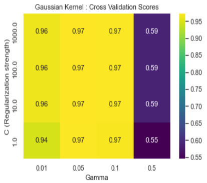
Part-2 Classification Problem

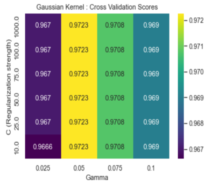
This part of the problem has been solved in the P2.ipynb .

1. Used **HeatMap visualization on GridSearchCV** results to tune the hyperparamters.
2. **kernel : gaussian dominated** after training few models , so I have fine tuned the paramters C and γ in the gaussian kernel









Best Hyperparamters

1. $C = 10$
2. $\gamma = 0.05$
3. kernel : gaussian/rbf
4. Score on Kaggle submission $= 0.96875$