HW3 Report

Burak Akgül

June 12, 2021

1 Part 1: Decision Tree

1.1 Information Gain

Test accuracy for info gain without prepruning: 93.3%. *info_gain_no_preprune.pdf.pdf* is the pdf file for the tree.

1.2 Average Gini Index

Test accuracy for average gini index without prepruning: 90%. *avg_gini_index_no_preprune.pdf.pdf* is the pdf file for the tree.

1.3 Information Gain with Chi-squared Pre-pruning

Test accuracy for info gain with prepruning: 96.66%. *info_gain_preprune.pdf.pdf* is the pdf file for the tree.

1.4 Average Gini Index with Chi-squared Pre-pruning

Test accuracy for average gini index with prepruning: 90%. *avg_gini_index_preprune.pdf.pdf* is the pdf file for the tree.

2 Part 2: Support Vector Machine

2.1 First Part

As can be seen from the below figures, increasing the C value lowers the margin of the classifier and results in a tighter fit. This lowers the generalization of the classifier but avoids missclassfications of training data.

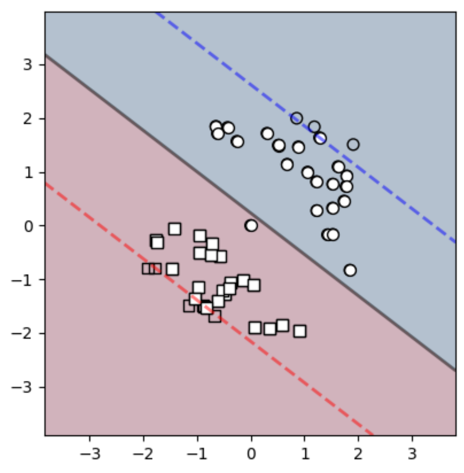

Figure 1: Linear kernel with C=0.01

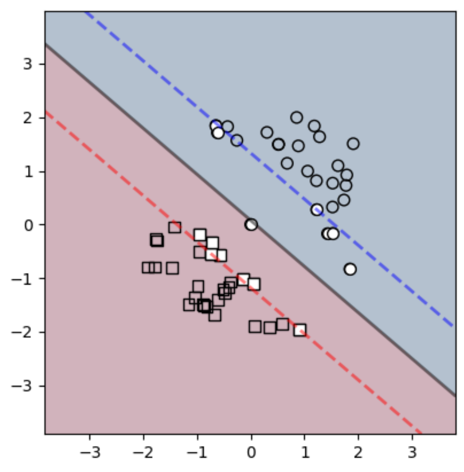

Figure 2: Linear kernel with C=0.1

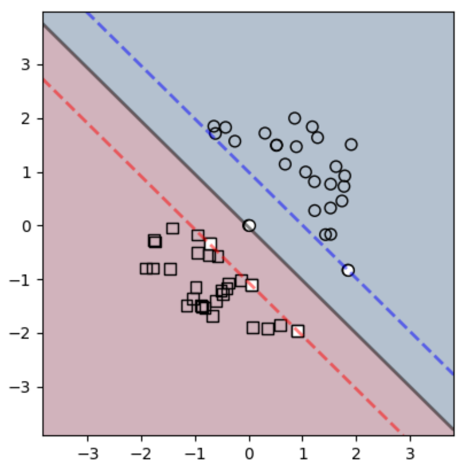

Figure 3: Linear kernel with C=1

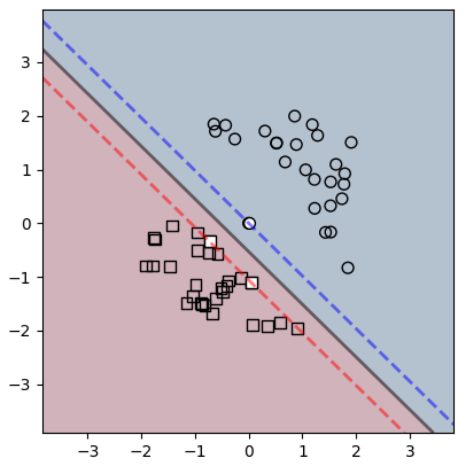

Figure 4: Linear kernel with C=10

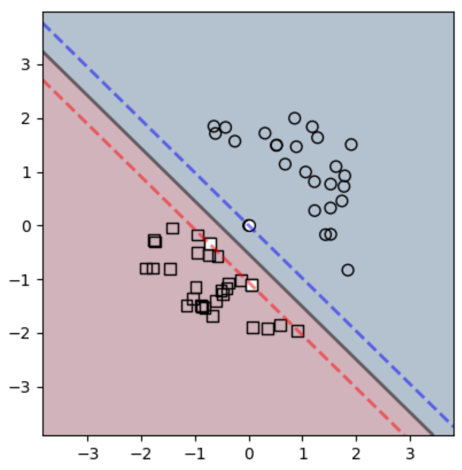

Figure 5: Linear kernel with C=100

2.2 Second Part

Each kernel use different strategies to divide the data. Linear classifier uses a line to divide the data, which performs poorly in our non-linear data. Polynomial, RBF and sigmoid kernels did better jobs but the best one is RBF kernel because one class in our data is collected in the middle of the cartesian space, where RBF kernel can divide it from the rest.

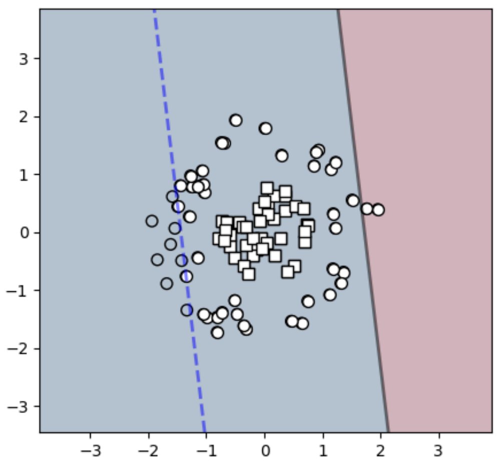

Figure 6: Linear kernel with C=1

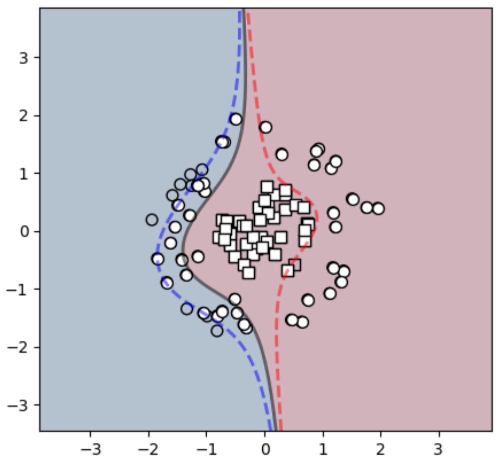

Figure 7: Polynomial kernel with C=1

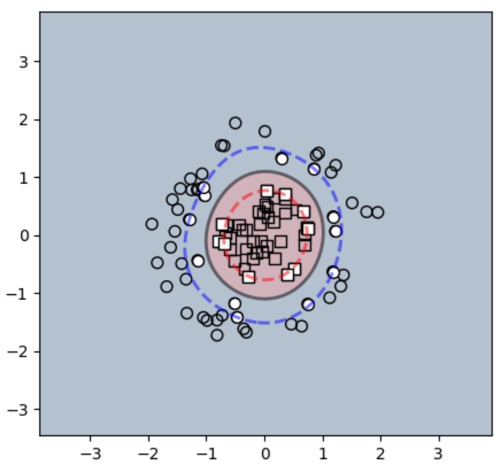

Figure 8: RBF kernel with C=1

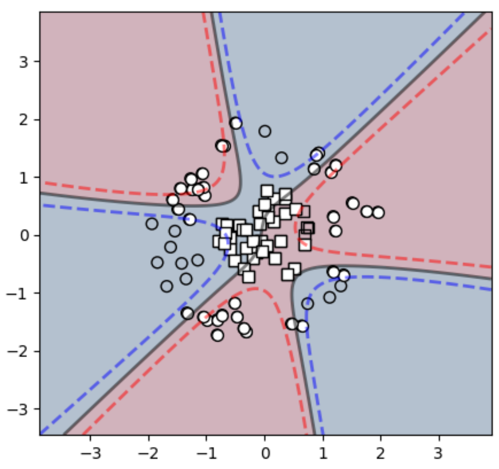

Figure 9: Sigmoid kernel with C=1

2.3 Third Part

Test accuracy for the best model (RBF) with best hyperparameters (C=100, gamma=0.01) is: 0.8806.

gamma	C				
	0.01	0.1	1	10	100
-	0.774	**0.8113**	0.7786	0.7379	0.73

Table 1: Linear kernel

gamma	C				
	0.01	0.1	1	10	100
0.00001	0.51	0.51	0.51	0.5113	0.7473
0.0001	0.51	0.51	0.5113	0.7486	0.796
0.001	0.51	0.5106	0.7506	0.828	0.8566
0.01	0.51	0.7613	0.8693	0.8939	**0.896**
0.1	0.51	0.51	0.8833	0.8853	0.8853
1	0.51	0.51	0.51	0.51	0.51

Table 2: RBF kernel

gamma	C				
	0.01	0.1	1	10	100
0.00001	0.51	0.51	0.51	0.51	0.51
0.0001	0.51	0.51	0.51	0.51	0.51
0.001	0.51	0.51	0.5506	0.7373	0.81
0.01	0.7373	0.81	**0.8746**	0.8673	0.8673
0.1	0.8673	0.8673	0.8673	0.8673	0.8673
1	0.8673	0.8673	0.8673	0.8673	0.8673

Table 3: Polynomial kernel

gamma	C				
	0.01	0.1	1	10	100
0.00001	0.51	0.51	0.51	0.51	0.736
0.0001	0.51	0.51	0.51	0.736	0.774
0.001	0.51	0.51	0.736	0.768	**0.7873**
0.01	0.51	0.5106	0.336	0.6873l	0.6813
0.1	0.51	0.51	0.51	0.51	0.51
1	0.51	0.51	0.51	0.51	0.51

Table 4: Sigmoid kernel

2.4 Fourth part

2.4.1 Without handling the imbalance problem

Test accuracy is 94.9%.

Accuracy may not be a good performance metric, as the data that to be tested may contain much more examples from one class and much fewer examples from the other class, and it may be the case that our trained model was already favoring that class with more examples, leading to high accuracy even when our model is trained poorly.

Confusion matrix (rows are test, columns are predictions)

3	76
0	1421

2.4.2 Oversampling the minority class

Test accuracy is 95.8%. Confusion matrix (rows are test, columns are predictions)

31	48
15	1406

As can be seen, our accuracy rose up. The reason for it is the model now can differentiate the class with fewer examples because we trained it with data that have nearly equal amount of instances in each classes. In the previous case (without handling the imbalance problem), our model was favoring class 1, but now our model can differentiate to an extent whether an instance belongs to class 0. (3 of them was labeled as class 0 with our model previously, but now 46 of them are labeled as 0; of course the model mistakes 15 of them as 0 too but overal, we can say that the model generalizes better, it does not sat class 1 to nearly every result).

2.4.3 Undersampling the majority class

Test accuracy is 79.46%. Confusion matrix (rows are test, columns are predictions)

54	25
283	1138

In this case, our model failed to predict the majority class because we deleted some important features by undersampling, leading to classfier cannot differentiating class 0 from class 1 properly. Classes are even in this case but there isn't enough data now.

2.4.4 Setting the class_weight to balanced

Test accuracy is 95.86%. Confusion matrix (rows are test, columns are predictions)

$$\begin{array}{cc} 38 & 41 \\ 21 & 1400 \end{array}$$

In this case, accuracy and predictions are similar to the oversampling part, becuase the updates coming from the minority class affect the model as much as the majority class as the model uses weighted updates.