CS 461 Homework 1

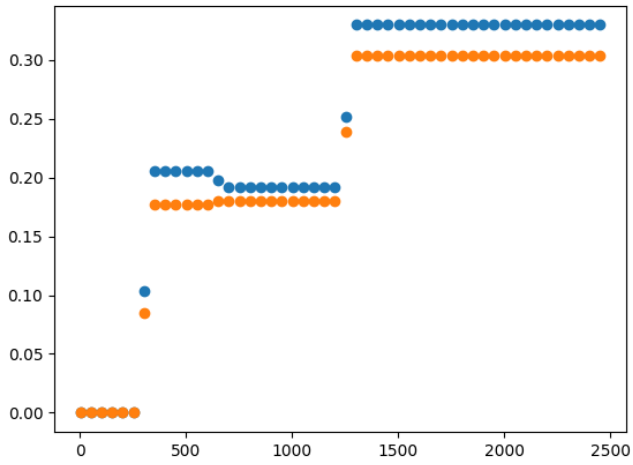
Ben Garcia

November 2022

1 Decision Trees for Classification

1) Write a program to generate data in accordance with the above, with $\sigma = 0.05$. Generate a training data set of size 5000 , and a test data set of size 500 . Plot the number of misclassifications on the training data and testing data, as a function of the minimum sample size you grow the tree to. What is the optimal sample size to grow the tree to, according to your data? Note - only consider odd sample sizes, so you can resolve terminal nodes by majority vote.

The plot bellow shows the percentage of misclassifications plotted over the sample size. Orange is the percent of misclassifications on the training data and blue is the percent of misclassifications on the testing data.



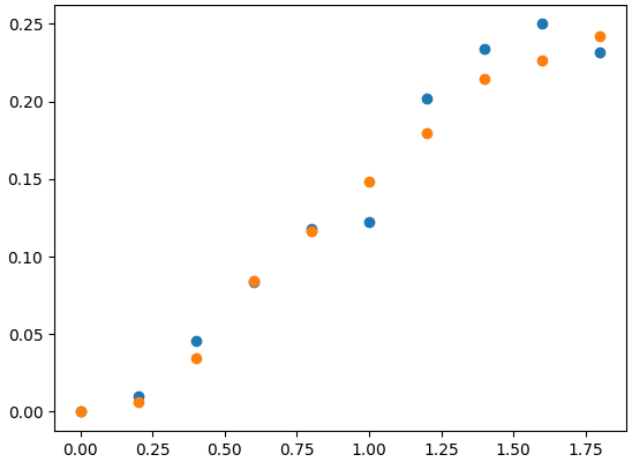
The optimal sample size appears to be any value under 201.

2) Repeat this a few times - is this optimal sample size consistent? Take the average optimal sample size, call its.

The sample size appears to be consistent. Any sample size s, where $3 < s < 250$ appears to perform optimally.

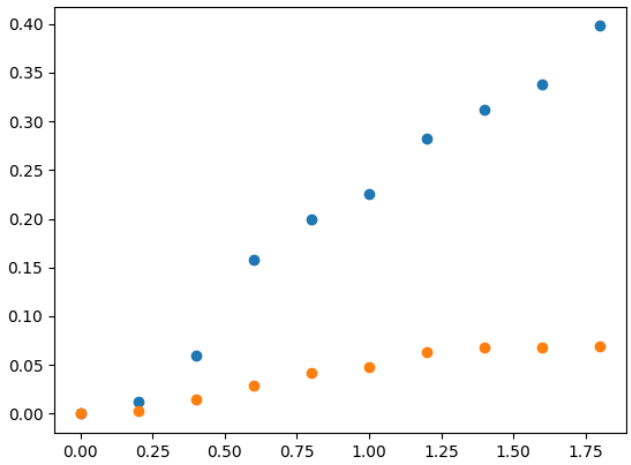
3) Generate again a training set and texting set for the $\underline{X}$ data as above, but we want to consider how the error on the model changes as a function of σ. Plot, as a function of σ, the training and testing error on a decision tree grown to sample size s. Note, for each σ you test, you'll need to recalculate Y based on that σ, and use those Y values to grow the decision tree. How does the noise influence the effectiveness of the tree?

The plot bellow shows the percentage of misclassifications plotted over σ. A sample size of 201 is used. Orange is the percent of misclassifications on the training data and blue is the percent of misclassifications on the testing data.



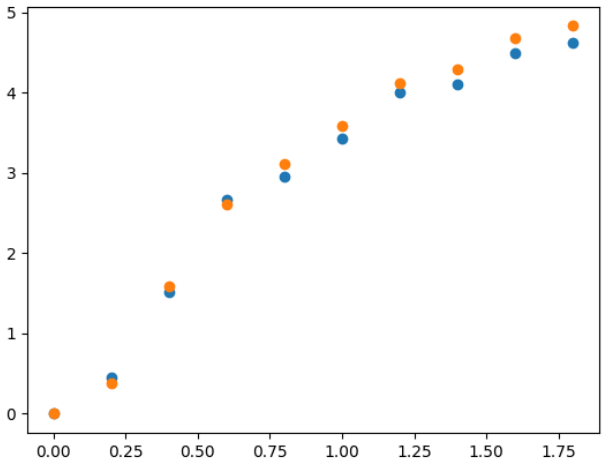
As the variance increases the accuracy of the decision tree reduces. This makes sense, it will be harder to compute information gain from each variable with greater amounts of noise. This results in less accurate splits on the data.

One thing of interest, if a smaller sample size in the optimal range is used, for example $s = 3$, then the variance will have a different effect on the testing and training data. I believe this is because the small sample size allows the training data to over fit and account for the noise. The plot bellow demonstrates this:



4) In parallel with Part 3 above, also plot as a function of σ the number of times irrelevant features show up in the tree (counting duplicates). How does the noise influence the likelihood of including irrelevant features?

The plot bellow shows the number of irrelevant features visited by the tree when predicting on the test and training data, at each σ. Orange represents the training data and blue represents the testing data.



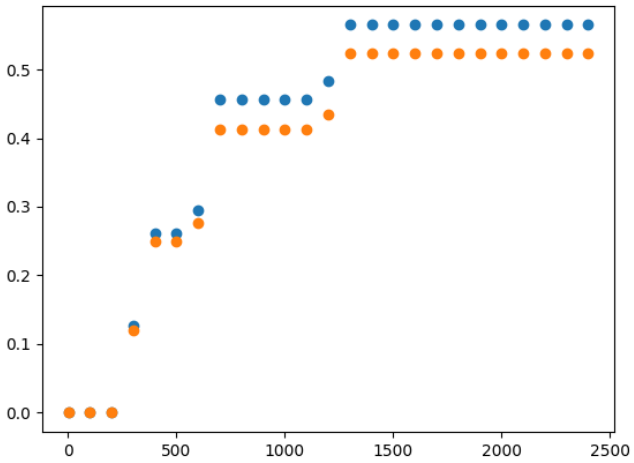
The number of irrelevant features used increases as σ increases. This intuitively makes sense because as σ increases the noise increases. This makes it harder to

distinguish which features actually provide information on y.

2 Logistic Regression

1) Generating a training data set of size 5000 and a test data set of size 500 with $\sigma = 0.05$, grow a decision tree on this data, but instead of resolving terminal nodes by majority vote, let the output of the tree be the fraction of data points in the terminal node that correspond to class Y= +1, i.e., an estimate for the probability a data point belongs to class Y= +1. Plot the logistic error of the decision tree on the test data and the training data as a function of the minimum sample size you grow the tree to. What is the optimal sample size to grow the tree to, according to your data? How does it compare to the previous section, when trying to predict the class label itself?

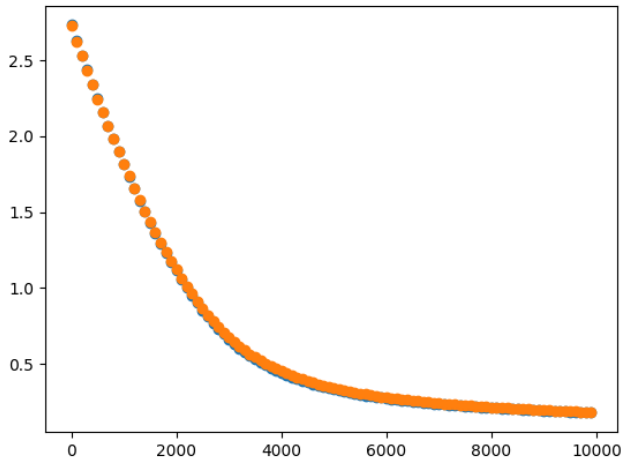
The plot bellow shows the logistic error plotted over the sample size. Orange is the logistic error on the training data and blue is the logistic error on the testing data.



The optimal sample size appears to be the same for both decision tree methods. A near perfect logistic error and percent missclassifications is achieved when $3 < s < 250$ for both methods respectively. Since the metrics differ it's not apparent if the majority vote or probability terminated decision tree works better. The logistic error does seem to increase more quickly though.

2) Consider fitting a logistic model to the data, i.e., $F(x) = \text{sigmoid}\,(w_0 + w_1x_1 + w_2x_2 + \ldots + w_{15}x_{15})$, taking $F(x)$ to be the probability that data point $\underline{x}$ belongs in class $Y = +1$. Plot the logistic error of the model on the test data and the training data as a function of time as you fit the model. Is overfitting an issue here?

The plot bellow shows the logistic error plotted over the training time. Orange is the logistic error on the training data and blue is the logistic error on the testing data.



Overfitting is not an issue here. First, the model is too simple to fit the training data perfectly. Further, from the plot it's apparent the model performs equivalently on the testing and training data, implying it's generalizing properly.

3) For the best decision tree model and the best logistic model - which one is better for modeling the data? Which one is better at minimizing the influence of irrelevant features - and how can you measure this?

The decision tree is better at modeling the data when there is low noise, while the logistic regression model is better when there is higher noise. With low noise or variance, the decision tree can perform perfectly on the testing and training data. The logistic model is also able to perform well with low noise, but it still has a logistic error of about .1 to .2. As the variance or noise increases the decision tree begins to perform worse. Since the decision tree can overfit on the noise, it's training error will remain low, but it's testing error will eventually increase past the logistic model. The influence of irrelevant features increases as noise increases. This is because more noise obfuscates which features are impacting the class. Thus, the model that performs better under similar amounts of noise is better at minimizing irrelevant features.

3 Support Vector Machines

We will formulate a polynomial kernel with $d = 2$ and $c = 1$ for the data. The kernel will have the form:

$$K(x_i, x_j) = (1 + x_i \cdot x_j)^2$$

We will need to account for all combinations of the data in the kernel. Luckily we know $K(x_i, x_j) = K(x_j, x_i)$ by properties of the dot product. Computing for all data points give:

$$\begin{aligned}
K(x_1, x_1) &= 9 \\
K(x_1, x_2) &= 1 \\
K(x_1, x_3) &= 1 \\
K(x_1, x_4) &= 1 \\
K(x_2, x_2) &= 9 \\
K(x_2, x_3) &= 1 \\
K(x_2, x_4) &= 1 \\
K(x_3, x_3) &= 9 \\
K(x_3, x_4) &= 1 \\
K(x_4, x_4) &= 9
\end{aligned}$$

We formulate the dual SVM as:

$$\begin{aligned}
&\max_{\underline{\alpha}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i y_i K\left(\underline{x}^i \cdot \underline{x}^j\right) y_j \alpha_j \\
&\text{(s.t.)} \ \sum_{i=1}^{m} \alpha_i y_i = 0 \\
&\forall i : \alpha_i \geq 0.
\end{aligned}$$

Notice that,

$$\sum_{i=1}^{m} \alpha_i y_i = -\alpha_1 + \alpha_2 + \alpha_3 - \alpha_4 = 0$$

$$\sum_{i=1}^{m} \alpha_i = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4$$

$$\frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i y_i K\left(\underline{x}^i \cdot \underline{x}^j\right) y_j \alpha_j = \frac{1}{2}(9\alpha_1^2 + 9\alpha_2^2 + 9\alpha_3^2 + 9\alpha_4^2 + (2)\alpha_1\alpha_4 + (2)\alpha_2\alpha_3 -$$

$$2\alpha_1\alpha_2 - 2\alpha_1\alpha_3 - 2\alpha_2\alpha_4 - 2\alpha_3\alpha_4)$$

Thus we want to maximize the following:

$$\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2}(9\alpha_1^2 + 9\alpha_2^2 + 9\alpha_3^2 + 9\alpha_4^2 - (2)\alpha_1\alpha_4 - (2)\alpha_2\alpha_3 +$$

$$2\alpha_1\alpha_2 + 2\alpha_1\alpha_3 + 2\alpha_2\alpha_4 + 2\alpha_3\alpha_4)$$

Such that $\forall i : \alpha_i \geq 0$ and $\alpha_2 + \alpha_3 = \alpha_1 + \alpha_4$

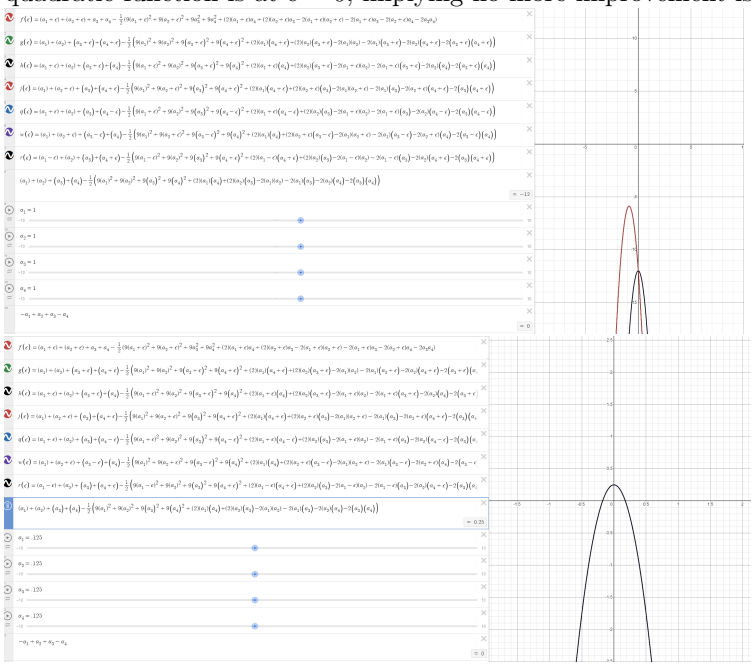
Notice that $\forall i : a_i = 1$ is a solution. Starting from this solution we can tweak pairs of variables to find an optimal α. We redefine the problem as:

$$\begin{aligned}\max \quad & (\alpha_1 + \epsilon) + (\alpha_2 + \epsilon) + \alpha_3 + \alpha_4 - \frac{1}{2}(9(\alpha_1 + \epsilon)^2 + 9(\alpha_2 + \epsilon)^2 \\ & + 9\alpha_3^2 + 9\alpha_4^2 - (2)(\alpha_1 + \epsilon)\alpha_4 - (2)(\alpha_2 + \epsilon)\alpha_3 + 2(\alpha_1 + \epsilon)(\alpha_2 + \epsilon) \\ & + 2(\alpha_1 + \epsilon)\alpha_3 + 2(\alpha_2 + \epsilon)\alpha_4 + 2\alpha_3\alpha_4)\end{aligned}$$

Such that $(\alpha_2 + \epsilon) + \alpha_3 = (\alpha_1 + \epsilon) + \alpha_4$ and $\epsilon > -1$.

Further, $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4$ and equivalent terms exist for all paired terms in the equation to be maximized. This implies that adding epsilon to any term from each side of the constraint $\alpha_2 + \alpha_3 = \alpha_1 + \alpha_4$ is equivalent. Thus, adding ϵ to α_1 and α_2 is equivalent to adding α_3 and α_4 and so on.

We can numerically compute the optimal α values by checking how epsilon effects each pair of α values. This Desmos graph visualizes the process. We start with a valid solution. For example, $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 1$. Then select the function with the maximum value, find ϵ for that value, and update the respective pair of α values. This process is repeated until the maximum of every quadratic function is at $\epsilon = 0$, implying no more improvement is possible.



Starting with all α at 1, we reach the maximum value at $a_1 = .125, a_2 = .125, a_3 = .125, a_4 = .125$ and result in a maximum value of .25

We can now reconstruct the w vector. Assume $\phi(x_1, x_2)$ to be the function mapping our two features into a quadratic embedding space.

$$\begin{aligned} w &= -.125\phi(-1,-1) + .125\phi(1,-1) + .125\phi(-1,1) - .125\phi(1,1) \\ &= -.125(1,-\sqrt{2},-\sqrt{2},\sqrt{2},1,1) + .125(1,\sqrt{2},-\sqrt{2},-\sqrt{2},1,1) + \\ &.125(1,-\sqrt{2},\sqrt{2},-\sqrt{2},1,1) - .125(1,\sqrt{2},\sqrt{2},\sqrt{2},1,1) = (0,0,0,-\frac{1}{2}\sqrt{2},0,0) \end{aligned}$$

We now compute the bias as:

$$1 - \left(-\frac{1}{2}\sqrt{2}\right)\left(\sqrt{2}(-1)(1)\right) = 0$$

Thus we get the simple classifier:

$$\text{sign}(x_1 x_2 \left(-2\sqrt{2}\right))$$

We can now construct the decision boundary by graphing $x_1 x_2(-2\sqrt{2}) > 0$ and $x_1 x_2(-2\sqrt{2}) < 0$: