Homework – DSC 540 Week Three

Colton Proctor

Grand Canyon University

Part One

1.

When handling a binary classification task, the metric that you use as your performance measure depends on the algorithm that you choose to implement. For example, take an SVM as the model that is being evaluated. Using the number of misclassifications as a performance metric is adequate as a performance measure but does not tell the full story.

Visualize the decision boundary of an SVM as a line on a 2D grid of points, there is an infinite number of lines that can be drawn between two separate classes to create a distinction for classification. However, most of these lines would not be optimal as they could be too close to one class or the other.

To deal with this issue, it is better to maximize the margin between two classes instead. Maximizing the margin finds the line that optimally splits two classes apart. This means that any new data that is input will have the greatest likelihood of belonging to the class that the decision boundary assigns to it. Where if any old line that gave the same number of misclassifications wouldn't necessarily have the same success on unknown data.

2.

For a binary classification task, I would choose to use linear SVM over utilizing a perceptron algorithm such as a Neural Network. As far as machine learning problems go, binary classification is on the lower end of complexity. This makes it suited for a less complex model such as a Linear SVM. It is also the type of problem perfectly suited to linear SVM based off how they make classifications. Namely, by finding an optimal decision

boundary by maximizing the distance between two classes. Linear SVMs can be used for multi-classification problems as well, but they are very well suited to the binary case.

Neural Networks on the other hand are very complicated models which utilize graphs to emulate the process used by neurons in a human brain. They dynamically optimize weights between neurons in a network so that they can assign a classification that best fits the data. This lends itself more aptly to multiclassification problems. Neural Networks also have the downfall of requiring lots of data to be trained appropriately. This is both not always feasible as well as computationally expensive.

The Linear SVM does not have these downfalls. They can find a decision boundary efficiently using matrices, as well as not needing a lot of data to find the boundary. Due to this I would choose to use an SVM for binary classification as its very well suited to the task. I would save Neural Networks for a time where there are many different classes as well as many data points, such as image recognition.

3.

The optional parameters for the SVM algorithm are the kernel, C, and epsilon. The choice of kernel is a major decision, and the choices are Linear, Sigmoid, RBF, Poly, or Precomputed. For my application I chose to use the default kernel which is RBF. This is because the underlying data that I am using is not linear in nature due to the structure of the sinc function. The RBF kernel imposes onto the data a non-linear structure which accounts for the sinc function.

The second parameter that can be adjusted is C, which defines the tradeoff for classification error over maximizing the decision functions margin. I found that as I increased C the classifications became more accurate. At C = 1 the default the classifications were

wildly incorrect. At C = 5 I reached good accuracy without risk of overfitting too closely. At C = 10 the predictions were even more accurate, but I risked overfitting. In the end I chose to use a C of 5 as a good tradeoff for accuracy and maximizing the margin.

The final term is epsilon, which is the error term that allows for no penalty when predictions fall within epsilon of the correct value. I found that halving the initial value from .1 to .05 increased the accuracy of the model. Any further adjustments lower didn't have any effect, and raising the epsilon caused the predictions to fall in a straight line toward the center of the graph. The higher the epsilon became the more inaccurate the predictions ended up being.

Overall, the regressor is not very accurate. I wouldn't use it to approximate the sinc function, as it gets the gerneral shape but with the noise it has a hard time predicting correctly. Only a few of the values that were predicted in the test set were actually very close to their true value. With more training data I do think that the underlying structure of the sinc function could more closely be approximated with this type of model.

Regarding the findings in the signal theory approach to support vector learning classification the model could be improved by choosing better parameters for its creation. As stated in the beginning of the article the strength and downfall of SVM models is that they are highly customizable through use of their parameters. Even though you see improvements while adjusting the parameters there is no way to know for sure if the model you are using is optimal or not.

The only way to know is to create an unreasonable number of models and test their proficiency against one another. That isn’t really feasible in the way I have it implemented; it would require much more automation. Guiding the choice of the parameters however would

reduce that search space and make it so that it would be possible to test a smaller range of parameters against one another.

In the article it states that you can use the sinc function itself to find the optimal parameters. You choose the maxima in the function and utilize them while also utilizing sinc as a kernel function as well. Changing the parameters of the SVM function to fit closely to the underlying structure of your data helps in its accuracy. Therefore, I think that using a different kernel, such as the sinc kernel, and guiding the optimization of its parameters would assist in creating a much more accurate model.

Works Cited

An Idiot's Guide to support Vector Machines (svms) - MIT. (n.d.). Retrieved November 25, 2021, from https://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf.

K, G. M. (2020, July 18). *Machine learning basics: Support vector regression*. Medium. Retrieved November 25, 2021, from https://towardsdatascience.com/machine-learning-basics-support-vector-regression-660306ac5226.

James D.B. Nelson, Robert I. Damper, Steve R. Gunn, Baofeng Guo,
A signal theory approach to support vector classification: The sinc kernel, Neural Networks, Volume 22, Issue 1, 2009, Pages 49-57, ISSN 0893-6080, https://doi.org/10.1016/j.neunet.2008.09.016
(https://www.sciencedirect.com/science/article/pii/S0893608008002347)

V. Gavriilidis and A. Tefas, "Random Walk Kernel Applications to Classification Using Support Vector Machines," 2014 22nd International Conference on Pattern Recognition, 2014, pp. 3898-3903, doi: 10.1109/ICPR.2014.668.