CS 423 Project 3: Machine Learning

Jared Staman
CS 423: Intro to AI
Fall 2021

Introduction

For this assignment, we were tasked to implement two types of machine learning models: support vector machines and neural networks. Support vector machines create a higher dimensional space so that the data can become linearly separable. Linearly separable data is much easier to classify, specifically when using perceptron algorithms. Neural networks can be thought of as many perceptrons chained together. These two models were used on the mushroom.csv data set. This data set consists of 23 features. The first feature is the target variable, which is labeled either “p” for poisonous or “e” for edible. The other 22 features are independent variables, which are used by the models to determine whether the mushroom is poisonous or edible.

Pre-Processing

Before the models can use the data, the data must be complete and uniform. This means that no data can have missing features or wrong characters. For the mushroom data set, if a feature had a ‘?’, then that data point’s whole row was removed. The next part of pre-processing involved changing the data from strings to floats for the models, which involved applying a “LabelEncoder”. For the neural networks specifically, the data was then normalized between 0 and 1. Normalizing data changes the values of all the features to be put on a common scale. This often dramatically increases the accuracy of the model.

Grid Search

Support vector machines and neural networks both have many options for hyperparameters. Hyperparameters are settings that programmers can change to fine-tune their model. For support vector machines, the hyperparameters that were evaluated were the kernel (linear, poly, rbf), a C value (1, 10, 100, 1000), a degree for the poly kernel (2, 3, 4), and a gamma for the rbf kernel (1e-3, 1e-4). For neural networks, the hyperparameters that were evaluated were the two activation layers used (linear, sigmoid, relu, tanh) and the two neurons used (1, 2, 5, 10, 15, 20, 25, 30). For both models, a coarse grid search was used along with k-fold cross-validation to determine the hyperparameters that led to the most accurate model. The number of splits for the k-fold CV was 5 for both models. This means that the data was split into 5 sections and that the coarse grid search was done 5 times, with each of the 5 sections being the ‘test’ data once. After the coarse grid search, a fine grid search was performed, which involves slightly tweaking the

hyperparameters to find the optimal ones. The following figure shows the fine-tuning for svm.

```
{'C': 34, 'kernel': 'linear'}

Grid scores on development set:

0.997 (+/-0.006) for {'C': 30, 'kernel': 'linear'}
0.997 (+/-0.006) for {'C': 31, 'kernel': 'linear'}
0.997 (+/-0.006) for {'C': 32, 'kernel': 'linear'}
0.997 (+/-0.006) for {'C': 33, 'kernel': 'linear'}
1.000 (+/-0.000) for {'C': 34, 'kernel': 'linear'}
1.000 (+/-0.000) for {'C': 35, 'kernel': 'linear'}
1.000 (+/-0.000) for {'C': 36, 'kernel': 'linear'}
```

Best-Performing Models

The best performing model for svm was the linear model with a C value of 100. The accuracy of this model was 100%. Many of the other models for svm also resulted in a 100% accuracy however. The linear model was chosen and then fine-tuned because it is faster than the other kernels. From the fine-tuning, the C value selected for the best model was 34. A smaller C value was chosen to avoid overfitting the data by keeping a larger margin when separating the data. The following figure shows the accuracy, precision, and recall of the linear C=34 model.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2801
1	1.00	1.00	1.00	1715
accuracy			1.00	4516
macro avg	1.00	1.00	1.00	4516
weighted avg	1.00	1.00	1.00	4516

The best performing model for the neural network was linear for both activations and a 1 for both neurons. The accuracy of this model was 100%. Unlike the svm search where most models were 100%, the linear activations were the only ones that resulted in a 100%. Fine-tuning was not very useful for this model as the smallest neuron value already resulted in a 100% accuracy. The following figure shows the accuracy, precision, and recall of the liner-liner-1-1 model.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1053
1	1.00	1.00	1.00	641
accuracy			1.00	1694
macro avg	1.00	1.00	1.00	1694
weighted avg	1.00	1.00	1.00	1694

Conclusion

From this assignment, I further expanded my understanding of Scikit-learn through the use of support vector machines, and I learned how to create neural networks using Keras for the first time. Because both of the best models had a 100% accuracy, I would choose to use the svm model because neural network models take much longer to train.