Darrion D. Banks

12 June 2017

CS 4641 - Machine Learning

Analysis of Internet Movie Database (IMDB) Ratings Classification

INTRODUCTION

Using the machine learning libraries WEKA and Scikit-learn, I executed decision tree, neural network, boosting, support vector machines, and k-nearest neighbors classification algorithms on data sourced from the movie review and ratings website, imdb.com, and movie appropriateness/ obscenity ratings from Netflix. The data were compiled in a CSV files posted on Kaggle.com. The IMDB dataset consists of approximately 5000 data points with 28 features, such as the director's Facebook likes, the actors' Facebook likes, budget and gross earnings, the IMDB score, and the number of users who scored the movie. The Netflix dataset consists of approximately 700 data points with 16 features, including the Netflix description code (for graphic content), the official rating (e.g., G, PG-13, R), and user's subjective appropriateness/obscenity rating.

From a machine learning perspective, these datasets are interesting because predicting movie scores and movie ratings is a subjective task for humans. Training models to predict scores and ratings would create a computational capability to achieve a subjective task. This is the basic cause for machine learning and artificial intelligence.

Both datasets contained errors and values were missing. To resolve this issue, I executed a Python script on the CSV to "clean" the data. By this, I eliminated inconsequential features (those with low predictive power or high correlation with another feature) and deleted entries with missing data. Cleaning the IMDB dataset with reduce the number of points to 3325. Cleaning the Netflix dataset reduced the number of points to 605.

DECISION TREE

IMDB Data Unpruned Decision Tree Training vs. Validation Scores on Increasing Examples

To implement the decision tree algorithm, I used WEKA, which allows pruning while Scikit-learn does not. For the IMDB movie dataset, executing 10 iterations on a training set, with 10-fold cross-validation, the WEKA generated a pruned decision tree with 9 nodes, of which 5 were leaves. The tree accurately classifies approximately 64 percent of the data. In contrast, an unpruned tree in Scikit-learn results in approximately 52 percent accuracy. In general, these are acceptable accuracy levels, although the pruned tree is preferable. Note, the above chart indicates overfitting by the unpruned tree. The increase in accuracy with the pruned tree indicates the difference is reduced by pruning, and thus the pruned tree is more generalizable.

To achieve these accuracy scores, I discretized the IMDB movie scores. Decision trees work best with discrete data. Thus, converting the continuous values to ranges improved the accuracy from less than 10 percent during initial experimentation to 64 percent now. The ranges I chose were: 1-3, 4-6, 7-9, and 10. Thus the tree is predicting bad movies, good movies, excellent movies, and phenomenal movies. These represent categories, and decision trees are best at categorizing.

For the Netflix movie rating dataset, executing with 10-fold cross-validation, the WEKA generated a pruned decision tree with 37 nodes. The tree accurately classifies approximately 28 percent of the data. The reason is likely that the ratings are highly

subjective judgments and quantification of these judgments is not standard. Two reviewers do not place the same value on an 80 value for content rating, for example. Netflix's coded description and rating according to standards such as G, PG, and R, are based on professional standards. The company has an interest in keeping viewers from offense, whereas ratings by the selected users are based on their personal, subjective value-judgments. Furthermore, a weak correlation of 0.09 between the Netflix rating description code and the user rating supports this conclusion, as well as other weak correlations between features suggests no predictive relationships exist in the data.

Decision trees are among the faster algorithms. Both decision trees executed in approximately 0.20 seconds. Decision trees are fast because of the simple calculations to split or pivot toward outcomes.

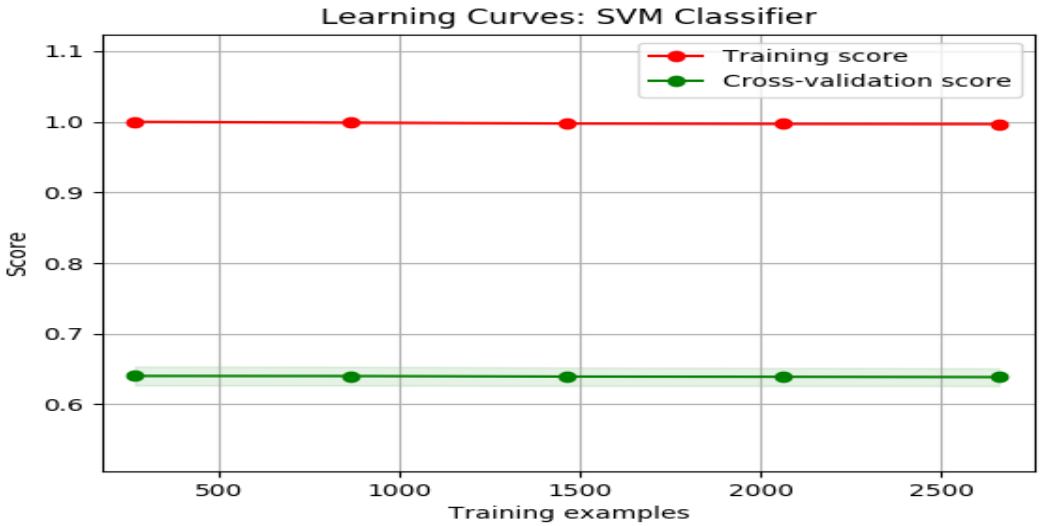
GRADIENT (ADA) BOOSTING

To implement the decision tree algorithm, I used WEKA, which allows pruning while Scikit-learn does not. For the IMDB movie dataset, executing 4 iterations, with 10-fold cross-validation, the WEKA generated pruned decision trees of sizes 9, 41, 75, and 1. The (non-normal) weights were 0.6, 0.23, 0.21, and 0.01, respectively. The boosted tree accurately classifies approximately 64 percent of the data. Note, there was no improvement over the pruned tree without boosting, likely due to the strength or predictive power of multiple features generating trees that are strong learners, whereas Ada Boost only improves accuracy if multiple weak learners can be combined to create a strong learner. Also, this accuracy score is attributable to the discretization of outputs, such that continuous IMDB scores were fit into ranges.

For the Netflix movie rating dataset, executing 10-fold cross validation did not improve the accuracy above 28 percent. Increasing the number of iterations from 10 to 100 also did not improve the accuracy. The likely explanation remains that the features are poorly correlated, or, in other words, there exist no predictive relationships between the features and the labels.

Decision trees are among the faster algorithms. Both decision trees executed in approximately 0.20 seconds. Decision trees are fast because they are simply pivoting or splitting toward set outcomes.

SUPPORT VECTOR MACHINES

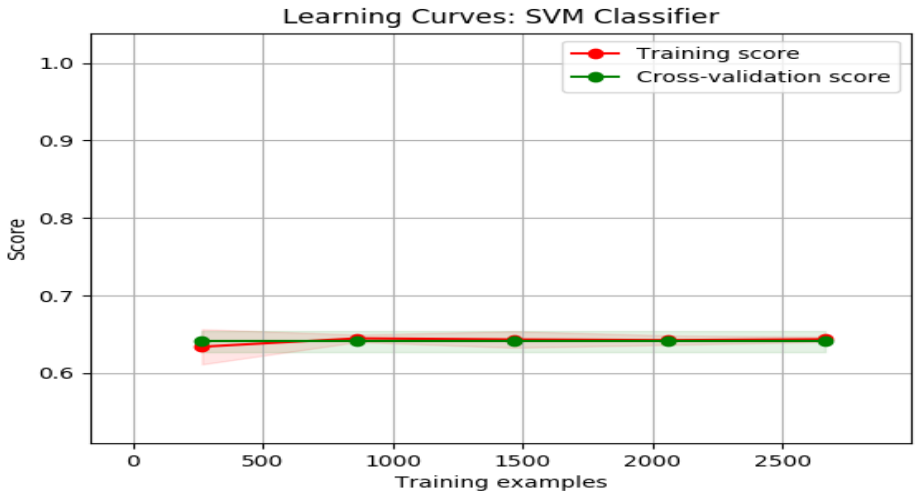


IMDB Data SVM Learning Curves with RBF Kernel

The above training score and validation-score plot resulted from the implementation of the support vector machines algorithm with a radial-basis function. While the polynomial The 100 percent training score of the training score and approximately 68 percent cross-validation score indicate the model is overfitting the data. The overfitting of the training data is likely occurring for two reasons: (1) the radial-basis function precisely encircles clusters of data and (2) these data on movies are likely to be clustered. Intuitively, movie data are likely to be clustered because many of the features are positively correlated with one another and the function the model is approximating is relatively simple. Popular movies are all likely also have high budgets and gross earnings, popular actors (by Facebook likes), more users voting, etc. Likewise, unpopular movies are all likely to have low budgets and gross earnings, less popular actors, fewer users voting, etc.

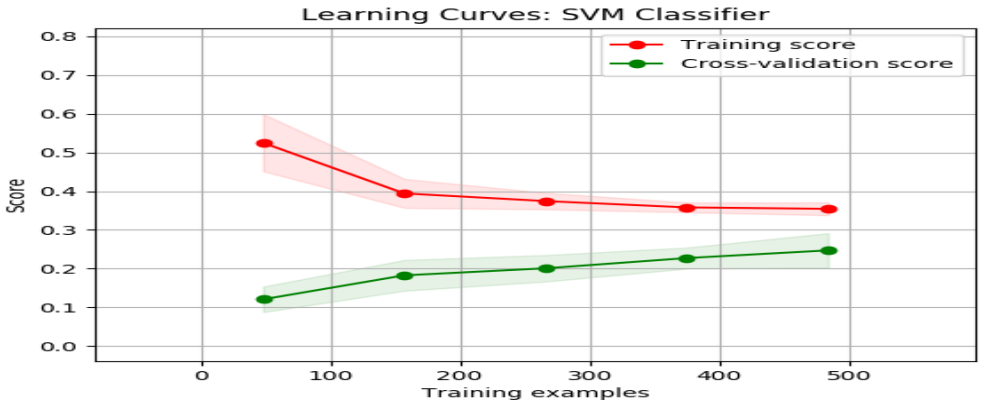
This indicates high similarity throughout the data. Thus, the data are amenable to fitting by a radial-basis function, which measures similarity of data in terms of spread. The

standard deviation of the normalized matrix of features is approximately 0.36. This gives a technical indication that the data are highly similar and explains why the radial-basis function trains the data well.



IMDB Data SVM Learning Curves with Sigmoid Kernel

Interestingly, using a sigmoid kernel generates a more generalizable model, albeit with approximately 64 percent accuracy, lower than the RBF kernel. The sigmoid kernel SVM is more generalizable because of a near 0 margin between the training score and the cross-validation score. This indicates that the SVM model is not overfitting the training data, making it likely that the model would perform on test sets outside this dataset. This kernel likely works because sigmoid functions set smooth thresholds that must be reached before a "decision" is made. Such thresholds exist in the dataset, where a budget would have to cross a threshold before determining a highly successful movie.



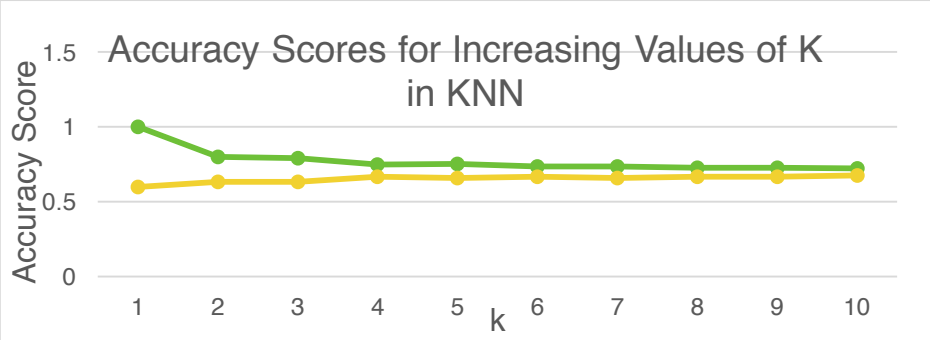
Netflix Data SVM Learning Curves with RBF Kernel

The decreasing training score and low cross-validation score with increasing training examples supports the conclusion founded in the section on the decision tree for this Netflix dataset. That is, there exist no predictive relationships within the dataset, such that data points with similar features, such as rating, Netflix rating, and Netflix rating description code, and the same class, would be in proximity to one another in a multi-dimensional space.

In general, the SVM algorithms are in the category of faster algorithms given that calculations are not being done on the data to get outputs (e.g., neural network). However, they are slower than decision trees. This is likely due to the complex linear algebra being applied, whereas decision trees are simply pivoting or splitting toward outputs based on information gain.

K-NEAREST NEIGHBORS

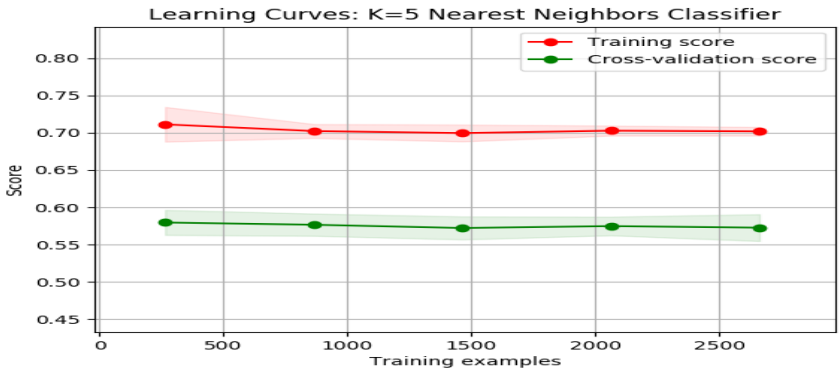
I implemented the k-nearest neighbors algorithm using Scikit-learn. I iteratively tested values of k from 1 to 10, in order to maximize the training and cross validation scores. As the graph above shows. At k = 5, there are no significant returns with increasing values of k. Furthermore, the convergence of the training and cross-validation scores around 70 percent is acceptable, as this indicates the model neither overfits nor underfits the data.



Accuracy Scores for Increasing Values of K on the IMDB Dataset

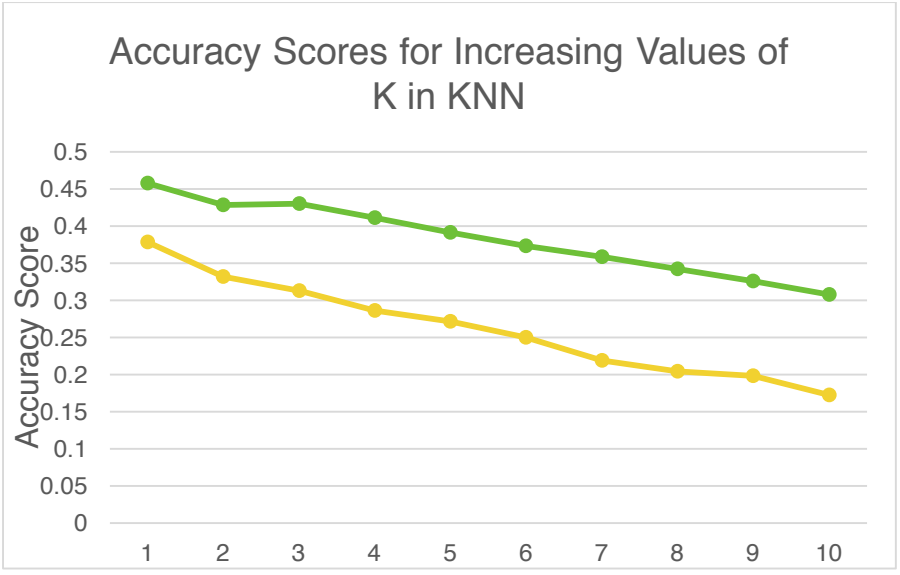
Iterating with a value of k=5, cross-validating, and reserving 20 percent as a test set, resulted in a training score of approximately 70 percent and a cross-validation score of approximately 57 percent. The k-nearest neighbors algorithm performs best on datasets

where points of each class are clustered or near to one another. One way to determine this is with a measure of spread, such as standard deviation. For this dataset, the standard deviation of the normalized matrix of features X is approximately 0.36. This is relatively low, meaning the points are not spread far from one another. One can infer from this general measure that within classes/labels, data are not widely spread from one another. Thus, the k-nearest neighbors algorithm yields acceptable performance.



KNN Training Scores vs. Cross-Validation Scores at K=5 over Increasing IMDB Examples

On the Netflix movie rating dataset, increasing values of k resulted in decreasing accuracy. This affirms the conclusion that these data have no power to predict how users rate content for appropriateness or obscenity. Given accuracy decreases as k increases, I implemented the KNN algorithm on the Netflix dataset for k=1.



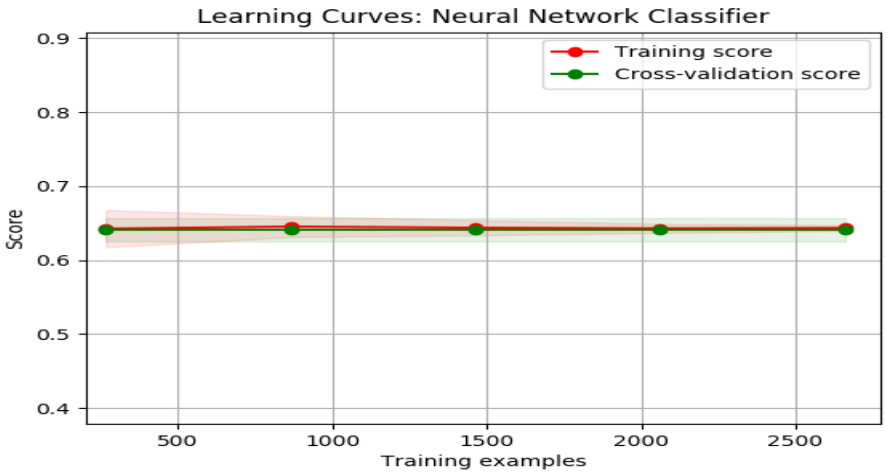
Accuracy Scores for Increasing Values of K on the Netflix Dataset

Like the decision trees algorithm, the KNN algorithm is relatively fast. The implementation on both datasets completed in just over 0.50 seconds. This is because the calculations for KNN, relative to SVM and neural networks, is simple. The KNN simply groups data points near one another into classes or labels.

NEURAL NETWORKS

I implemented the neural network using the MLPClassifier (i.e., multi-layer perceptron) classifier in Scikit-learn with the language Python. On intuition, I executed the neural net on the following features: duration; director Facebook likes; actor 1, actor 2, and actor 3 Facebook likes; total cast Facebook likes; content rating (e.g., G, PG, R, etc.); budget and gross earnings; the number of users who voted or reviewed the movie; and the number of critics who reviewed the movie. To mitigate cultural bias or differences, I limited the movies to those filmed and shown in the United States.

The activation functions available include the logistic (sigmoid) function, the tan(h) function, and the rectified linear unit (ReLU) function. The functions yielded results consistent with one another, with the exception of the ReLU function that yielded approximately 30 percent accuracy at random iterations.

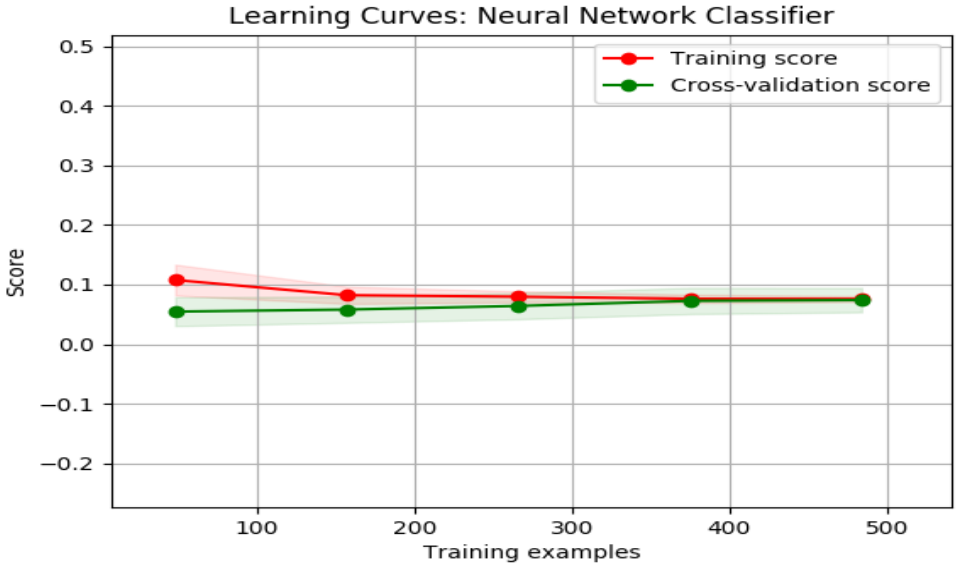


IMDB Dataset Training Scores vs. Cross-Validation Scores for Increasing Examples

Implementing a logistic function in the neural network using 3047 data points, 10 hidden layers with 46 nodes (double the number of features), stochastic gradient decent, cross-validating, and reserving 20 percent as a test set, resulted in a training score of

approximately 64 percent and a cross-validation score of approximately 64 percent. As the number of examples increases, the training score and cross-validation score are approximately the same, even as the number of examples increases. The training score increases to a minor extent as the number of features increases. However, this is insignificant. The proximity of the training score and the cross-validation score indicates that the neural network is not overfitting the data. The training score indicates that the model learns the training data well and at a rate consistent with the accuracy of classifying test data. The same trend results with 100 hidden layers as did with 10 layers. Again, the training score indicates that the model learns the training data well and at a rate consistent with the accuracy of classifying test data.

The logistic function likely works so well because it sets a threshold that must be overcome before a positive decision is made. As asserted with the sigmoid kernel in the SVM, these thresholds exist within the data, where, for example, a threshold – as opposed to a hard line between high and low – must be overcome before a budget is high enough to predict a high-scoring movie.



Netflix Dataset Training Scores vs. Accuracy Scores for 10 Layers and 46 Perceptrons

On the Netflix dataset, the data plotted above show the training scores and cross-validations scores of as a result of implementing a neural network with 10 hidden layers with

46 perceptrons and 100 hidden layers with 46 perceptrons. The low training and cross-validation accuracy scores of approximately 8 percent support the earlier conclusion about the Netflix data: that no predictive relationships exist between the features and the labels.

Both implementations of the neural network completed in approximately 170 seconds. Neural networks are slow primarily due to backpropagation, especially stochastic gradient descent, which requires iterative reduction of error. This repeated backwards and forwards work intuitive takes longer than input-to-output, unidirectional solution algorithms such as Decision Trees, SVM, KNN.

CONCLUSION

The best models with respect to both accuracy and generalizability are the neural network and the SVM with sigmoid kernel on the IMDB dataset. A generalizable model is one with a small margin between the training scores and cross-validation scores. This indicates that the models do not overfit the data. Thus, the models are likely to be accurate on test sets outside the dataset in this experiment.