CSE 514A Data Mining: Assignment 2

Danni Beaulieu (ID 497567)

[1] http://archive.ics.uci.edu/ml/datasets/letter+recognition

Introduction

In the UCI Letter Recognition Data Set, there are 16 features created from “statistical moments and edge counts” of black and white letters of the English alphabet; the characters themselves were derived through randomly distorting 20 different fonts [1]. While the dataset itself is not composed of handwritten characters, it is not far fetched to imagine that it could assist with identifying them. The fonts and their distortions may act as a simulation of different handwriting styles. Machine learning models may train on these if a dearth of handwriting data exists or is not accessible for legal, privacy, or other reasons. Another real life situation these characters may simulate even more closely is one in which characters have been printed onto paper. Applications of the character recognition models may be scanning documents like receipts, recognizing signs for traffic or businesses, or reading lettering from photographs.

To determine the “best” classifier for deciding between two specific letters, say ‘H’ and ‘K’ or ‘M’ and ‘Y’, there are a few metrics to consider. We can use model accuracy or loss, however variance explained (R-squared) does not make sense for classification problems. AUC would be a better way to evaluate than R-squared, especially considering the dataset is relatively balanced. If it were severely imbalanced, it might be better to look at the PR curve. We should also examine complexity and interpretability. For instance, if these models are deployed in real time systems, minimal gains may not be worth the expense of a deep neural network or even a random forest. Similarly, it can be easier to iterate on systems and explain them to others when the type of model matches the task. That is, at least intuitively, as a human, letter recognition does not seem to match with a decision tree since no sequences of choices are consciously made and no chain of consequences come to mind when thinking about identifying one letter versus another. It may seem to match more easily with a Bayes model based on probabilities or an SVM which leverages margins of separability for classification problems. Indeed it is for these reasons, I selected Bayes and SVM for my experiments. KNN may also be a fair choice, given an appropriate distance metric, but again could venture into unintuitive territory and generally requires continual storage of all data, which could limit deployability.

Another step to consider is dimensionality reduction. Without any missing values to remove or impute and without a strong understanding of the features themselves from domain knowledge, simple quality filtering and filter methods do not seem to be the best choice. Likewise, since this is not a regression problem and I have elected not to use decision trees or a random forest, embedded methods are not well-suited. Wrapper feature selection is a possibility, although it is somewhat of a brute force technique. Lastly, we have feature extraction and a range of options including PCA, LDA, ICA, SVD, NMF, and t-SNE. Originally, LDA seemed like a front-runner

since it performs well with classification and does not require standardization, which makes Gaussian assumptions about the data. Unfortunately, the number of features it creates is the number of classes minus one; for binary classification, this would result in only one feature. ICA also does not necessarily seem ideal since it assumes the features are statistically independent and non-Gaussian; in letter classification it seems unlikely the features would be wholly independent of each other. SVD is most popularly applied with sparse data, which is not the case for us here. NMF has been said to be more or less a constrained version of SVD in which between the two, SVD optimizes predictions while NMF optimizes explainability. A benefit of PCA is that since it uses orthogonal transformation which maximizes the variance of the dataset, the resulting extracted features should not be correlated with each other. Finally, t-SNE did not seem like a good fit since it is generally used more for visualization and can be incredibly sensitive to parameter choices, possibly producing misleading results. Given that we know the features will contain correlations, overall, PCA seems like the most promising technique for this dataset. Additionally, it includes noise reduction and does not require additional parameters.

Framing the alphabetical classification problem as pairs of binary problems is interesting. It seems like discriminating between ‘H’ and ‘K’ could operate as an intersection detector, whereas discriminating between ‘Y’ and ‘M’ could operate as a detector for additional line segments. I say this because one way to think of the difference between ‘H’ and ‘K’ is where the middle to right line segments meet. That is, do they meet in the middle or on the right-hand side of the letter? One way to think of the difference between ‘Y’ and ‘M’ is whether the left-hand line segment and right-hand line segment are noticed or not and whether the short middle line segment is noticed or not. In this vein, I chose to predict whether letters are ‘O’ or ‘D’ and think of this differentiator as whether roundness is detected or not. I also feel these two letters will be the most difficult to classify against each other and therefore will be a good metric for my models to ensure they can perform well. The easiest pair to classify may be a pair like ‘O’ and ‘X’ wherein the visual disparity between the two seems very high. ‘O’ is a round circle while ‘X’ consists of line segments only.