CS244 Exercise 5

Task 1: Comparing Machine Learning Algorithms

You run a large company that receives tens of thousands of job applications each year for positions in your company. You have a database of the hundreds of thousands of applications that you have received over the past decade as well as information about who, from this pool, you have phone interviewed, interviewed in person, and hired. For those people that you hired, you have annual performance evaluation data as well. You are interested in a machine learning algorithm to determine, from the tens of thousands of applications that you receive each year, which small subset you should interview. What machine learning algorithm might be appropriate for this task (circle all that apply)?

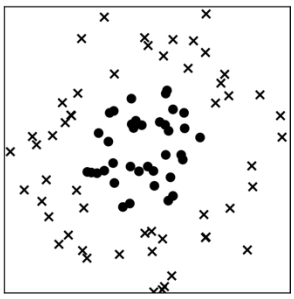
- Decision tree
- *k* nearest neighbors
- Linear regression
- Logistic regression

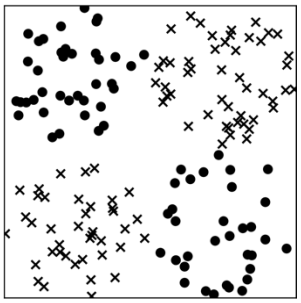
For what algorithms is it important to use feature scaling (circle all that apply)?

- Decision tree
- *k* nearest neighbors
- Linear regression
- Logistic regression

Several of the algorithms that we studied for solving *classification* problems can be adapted to solve *regression* problems. Describe how the *k* nearest neighbors algorithm, which we used previously for solving *classification* problems, can be adapted to solve *regression* problems.

Suppose we have data, such as in the two examples illustrated below, that we have reason to believe are not linearly separable.

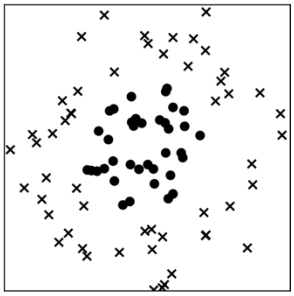


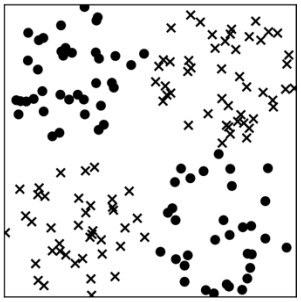


Assuming we do not change or add features, which algorithms might be appropriate to use to classify such data?

- Decision tree
- *k* nearest neighbors
- Perceptron
- Logistic regression

Suppose we have data, such as in the two examples illustrated below, that we have reason to believe are not linearly separable.





Assuming we can change and add features, which algorithms might be appropriate to use to classify such data?

- Decision tree
- *k* nearest neighbors
- Perceptron
- Logistic regression

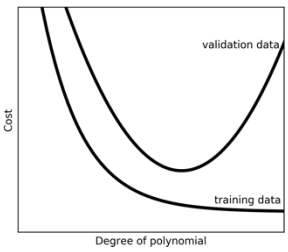
Task 2: Logistic Regression

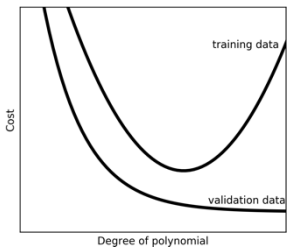
For gradient descent, on the surface the gradient for logistic regression looks the same as that for linear regression:

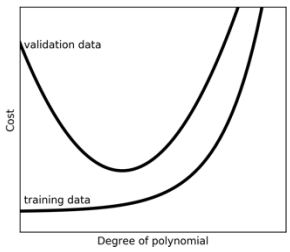
$$w_j = w_j - \alpha \frac{1}{n} \sum_{i=1}^{n} (h(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

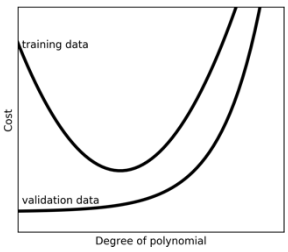
However, the gradient calculation for logistic regression is different from that of the gradient calculation for linear regression. Why?

Suppose you want to classify data that you believe are not linearly separable and you wish to explore using non-linear logistic regression for this purpose. You experiment with adding features to your data that are polynomial combinations of your original data. You add features that correspond to increasingly complex polynomials, i.e., polynomials of increasing degree. The degree of the polynomial may be viewed as a hyperparameter, so you can evaluate polynomials of different degree using validation data. Which of the following figures represents a reasonable expectation for how the cost associated with training data and with validation data might vary as the degree of the polynomial increases?









Task 3: Confusion Matrix

Thus far, when evaluating the performance of a classifier, our measure of accuracy has been the percentage of classifications that the classifier makes correctly, i.e., if it classifies 35 out of 50 data points correctly then we say that the accuracy is 70%. This notion of accuracy may be reasonable when we have approximately the same number of examples in each class, but it can break down when we have skewed classes. For example, imagine we are classifying dollar bills as authentic (0) or counterfeit (1). Suppose we encounter many more authentic bills than counterfeit bills, e.g., 99% of bills are authentic and 1% are counterfeit. If we used a naïve algorithm for classification that *always* classified every bill as authentic, then it would have an accuracy of 99% because it would correctly classify all authentic bills (which account for 99% of all bills) even though it would incorrectly classify all counterfeit bills (which account for 1% of all bills). Such a naïve classifier achieves a high accuracy, 99%, even though it is not particularly useful. This is one motivation for using other measures (besides accuracy) of a classifier's performance, particularly when the data are skewed.

A *confusion matrix* uses a table format to keep track of different aspects of a classifier's performance. For instance, in the binary classification problem of identifying authentic and counterfeit bills, the confusion matrix would keep track of:

- True Positives (TP) - the number of bills predicted to be counterfeit that are actually counterfeit
- False Positives (FP) - the number of bills predicted to be counterfeit that are actually not counterfeit
- True Negatives (TN) - the number of bills predicted not to be counterfeit that are actually not counterfeit
- False Negatives (FN) - the number of bills predicted not to be counterfeit that are actually counterfeit

Confusion matrix

	Actually Counterfeit	Actually Not Counterfeit
Predicted by classifier to be counterfeit	TP	FP
Predicted by classifier not to be counterfeit	FN	TN

Suppose for one thousand bills, a classifier predicts 30 to be counterfeit and 970 to be authentic. Of the 30 counterfeit predictions, 20 are actually counterfeit. Of the 970 authentic predictions, 965 are actually authentic. Fill in a confusion matrix below indicating the number of true positive, false positive, false negative, and true negative classifications.

Two common measures of a classifier's performance are *recall* (a.k.a., *sensitivity*) and *precision*. *Recall* is the percentage of actual positives (counterfeit bills in our example) that were correctly predicted to be positives (counterfeit). *Precision* is the percentage of predicted positives (predicted counterfeit bills) that are actual positives (counterfeit bills).

$$recall = \frac{TP}{TP+FN} \qquad precision = \frac{TP}{TP+FP}$$

In our example, high recall means that we are identifying a lot of the actual counterfeit bills. High precision means that when we predict a bill as counterfeit, our prediction is likely correct.

What is the recall and what is the precision associated with our counterfeit bill classifier?

It is desirable for a classifier to have high recall and high precision. However, there is often a trade-off between the two, where a classifier's recall can be improved at a cost to its precision, or the classifier's precision can be improved at a cost to its recall. For a given classifier, if you tweak its parameters so as to increase the number of positive predictions that the classifier makes (in our example, you cause the classifier to predict a larger number of bills as counterfeit), then which of the following are likely:

- Recall will increase and precision will decrease
- Precision will increase and recall will decrease

It is often helpful to have a single measure, rather than two measures (recall and precision), to evaluate a classifier. The F_1 score is a measure of a classifier's performance that combines recall and precision into one number. The F_1 score is defined as follows:

$$F_1 \text{ score} = 2 * \frac{recall * precision}{recall + precision}$$

What is the F_1 score for our counterfeit bill classifier above?

Instead, suppose we use a naïve classifier that predicts all one thousand bills are counterfeit even though only 30 are actually counterfeit. What is the F_1 score for this naïve classifier?

Task 4: Application of Machine Learning

Your favorite college is considering a change to its admission process. Rather than having staff in the Admission Office read and classify (as *accept* or *reject*) the thousands of applications from aspiring students that the college receives each year, a machine learning algorithm will use the applications as input to recommend who will be offered admittance to the fine institution. While Admission Office staff will oversee the process and decide whether to accept or change the recommendations of the machine learning algorithm, the bulk of the work in reading applications will be done by a machine rather than by people.

Proponents of this admission process change argue that using machines rather than humans will enable a more transparent and equitable system. Humans, the proponents argue, have biases, both conscious and unconscious. Proponents point to research indicating how people will give *different* scores to two otherwise *identical* applications when the applications contain certain information (e.g., race, gender identity, disability, sexual orientation) about the applicants, but people will give the *same* score to two identical applications when the same information (e.g., race, gender identity, disability, sexual orientation) about the applicants is removed from the applications. People's biases can be opaque. It is difficult to audit people about why they gave particular scores to various applications, and it is difficult to eliminate problematic biases in people. In contrast, proponents suggest, it is easier to audit a machine learning algorithm to understand why the algorithm gave particular scores, and it is easier to find and eliminate problematic biases in the algorithm. A further benefit of the admission process change, proponents point out, is that it will dramatically reduce costs in the Admission Office, thereby enabling the college either to charge lower tuition or to offer greater financial aid packages.

Do you support changing the admission process to use machine learning algorithms to determine admittance? What aspects of the change are you comfortable with and uncomfortable with? Are there admission practices/policies/properties that you would insist on?
(minimum 100 word response)

Download the Jupyter Notebook for Exercise 5 from the course website. Open the Notebook in your web browser and work through it. As you work through the Notebook, answer the following questions.

Task 5: Parkinson's Disease

What is the F1 score of your logistic regression classifier on the test data?

Task 6: MLpronto

For comparison, try taking the file, `parkinsons.csv`, that you analyzed in the previous task and analyze it now using MLpronto. Use the same parameters in this task as you used in the previous task.

What is the F1 score reported by MLpronto for the test data when using a logistic regression classifier? Is this the same as the F1 score you observed in the previous task? If not, what is your hypothesis for why the scores differ?