Machine Learning, Fall 2018: Project 3

October 26, 2018

You may use any programming language you like (Matlab, C++, C, Java...). All programming must be done individually from first principles. You are only permitted to use existing tools for simple linear algebra such as matrix multiplication/inversion. **Do NOT use any toolkit that performs machine learning functions and do NOT collaborate with your classmates. Cite any resources that were used.**

1 Project Description

In this exercise, you will implement your own support vector machines (SVM) to solve the Adult Census Income kaggle task: https://www.kaggle.com/uciml/adult-census-income#adult.csv Please submit your code and report along with visualizations and explanations.

2 Dataset preprocessing and interpretation

1. Abandon samples with missing terms: There are several rows of data containing '?'. Abandon the rows that contain '?'.
2. Dealing with discrete (categorical) features: There are some categories that contain discrete features. For example, *marital.status* can be different values: "Widowed", "Divorced", "Never-married" and so on. Find a good representation for them so that they can be used to train a support vector machine.
3. Split the dataset for stratified 10-fold-cross validation.
4. Analyze the features and make a scatter plot with the two features that have the highest information gain.

3 Implement a linear soft-margin SVM

1. Train your SVM with *stratified 10-fold-cross-validation* on the 2 features you selected and visualize your boundary. i.e. plot the support vectors and draw the decision boundary.
2. Change the C parameters from small to larger values. Report your observations on how the value of C would affect SVM's performance. Draw the decision boundaries with smaller and larger values of C to explain its effect.
3. Train the SVM using all the features. Find a way to determine the optimal value of C. Report your accuracy on the test dataset.

4 Implement a kernel SVM

1. Compare the performance (precision, recall, f1-score, and variance) of different kernels: Linear, RBF, and polynomial.
2. Try your best to get higher performance! You can design your own kernel, use bagging or boosting methods, logistic regression, decision trees, Naïve Bayes, or whichever method you prefer. Provide your code and your evaluation method, then explain why the performance is better with your method of choice by using learning curves.