INF264 - Homework 5

Pierre Gillot Natacha Galmiche*

Week 39 - 2021

1 Polynomial kernel SVM

In this exercise, we learn how to use a polynomial kernel SVM classifier to fit on non-linear data. Answer the following questions:

1. Load and visualize the dataset contained in the file 'svm_data_3.csv'. Can you hope to learn anything reasonable with a linear SVM classifier ?

2. You can create a polynomial kernel SVM classifier with the class 'sklearn.svm.SVC', where the argument 'kernel' is set to 'poly'. Four parameters are particularly important for this model: 'C', 'degree', 'gamma' and 'coef0'. Perform a model selection on the aforementioned parameters of the polynomial kernel SVM classifier (remember to evaluate the performance of the best model).

3. In a 2D plot, display the training data, the support vectors, the decision boundaries, the decision hyperplane and the separating hyperplanes of the best model. **Hint:** Since the data features are two-dimensional, hyperplanes are simply lines. You can display these lines by plotting the levels $\{-1, 0, 1\}$ of the 'decision_function' method of your model applied to a grid, using the 'matplotlib.pyplot.contour' function.

4. In a 3D plot, display the decision surface and the decision values of the training data using the 'matplotlib.pyplot.plot_surface' and 'matplotlib.pyplot.scatter3D' functions, respectively. **Hint:** The three dimensions correspond to the two features and the decision values returned by the 'decision_function' method.

5. How different are the decision hyperplane and separating hyperplanes (from question 3) compared to what you obtained with the linear SVM classifier on linearly separable data ? Explain why.

6. Implement your own polynomial kernel as described in `https://scikit-learn.org/stable/modules/svm.html#svm-kernels`.

7. Implement your own decision function and predict function as described in the slides of the course, and compare them to the 'decision_function' and 'predict' methods of a polynomial kernel SVM classifier from sklearn.

2 Boosting

Boosting classifiers are a type of ensemble classifiers whose main objective is to reduce the bias by training a sequence of weak learners so that subsequent models compensate the errors made by the earlier ones. This approach is motivated by a theorem stating that weak learners (that is to say learners that are barely better than a random classifier) can in theory be 'boosted' to perform as well as 'strong learner' (that is to say, classifiers whose accuracy is almost 1).

Typically a weak learner could be a decision tree with a max depth set to 1 or 2.

*Not a TA for this course this year.

The dataset we will use in this exercise is the Sonar dataset. This dataset describes sonar chirp returns bouncing off different surfaces. The 60 input variables are the strength of the returns at different angles. It is a binary classification problem with 208 observations that requires a model to differentiate rocks (labeled R) from metal (labeled M) cylinders.

There is a template available, you can choose whether you want to use it or not.

1. Load the sonar dataset and store the features in a matrix X and labels in a vector y. M and R labels should be converted into 1 and -1 labels.

2. Boosting training function:
   (a) This function should take as input a training dataset X_train and y_train; a number of classifiers n_cfls; a classifier $Classifier$ (could be any sklearn class implementing a classifier) and a dictionary $clfs\_args$ containing arguments specific to the $Classifier$ class given as argument.
   (b) This function should return a list of trained classifiers composing the boosting classifier as well as a list of their associated weights.

3. Boosting predict function:
   (a) This function should take as input a list of trained classifiers $clfs$ composing a boosting classifier, their associated weights $alphas$ and a matrix of features X.
   (b) This function should return a vector of predictions made by the boosting classifier with the rule of (weighted) majority vote.

4. Evaluation of boosting classifiers based on DecisionTrees:
   (a) Train and evaluate using cross validation a boosting classifiers based on n_clfs Decision Trees classifiers whose max depth is set to 1 for different number of classifiers n_clfs.
   (b) Compare the performance on the test set of a boosting classifier consisting of a single Decision Tree whose max depth is set to 1 (that is to say the boosting classifier is actually a decision tree of depth 1) with a boosting classifier consisting of 400 Decision Trees.
   (c) What can we say about the variance and bias of a simple decision tree compared to a boosting classifier with 400 Decision Trees?
   (d) What are the disadvantages of a boosting classifier then?

5. Evaluate boosting classifiers based on different numbers of Logistic regression models.