Using Boosting and Gaussian Processes for classification

Nishant Gupta
Technische Universität München
nishant.gupta@tum.de, Mtk. Nr.: 03646526
https://github.com/nishantgupta91/MLFCV/tree/master/Exercise3

***Abstract*— The first part of this exercise deals with applying Boosting methods to classify handwritten digits of MNIST dataset. In the second part, Gaussian Process classifiers are trained and evaluated with respect to their uncertainty estimation.**

I. Introduction

Boosting is an approach to machine learning based on the idea of creating a highly accurate prediction rule by combining many relatively weak and inaccurate rules [3]. The AdaBoost algorithm of Freund and Schapire [4] was the first practical boosting algorithm, and remains one of the most widely used and studied, with applications in numerous fields. In a Gaussian process, every point in some input space is associated with a normally distributed random variable. Every finite collection of those random variables has a multivariate normal distribution. The distribution of a Gaussian process is the joint distribution of all those random variables, and as such, it is a distribution over functions.

II. Dataset Description

The MNIST dataset [1] contains 60,000 digits ranging from 0 to 255, each value represents the equivalent shade of the pixel. Each image in the dataset has a height of 28 pixels and a width of 28 pixels. This makes the data to have 28 x 28 = 784 features to train on. The dataset also consists of a training set of 10,000 digits. However, the dataset seems to provide best training results when normalized. Two files – loadMNISTImages.m and loadMNISTLabels.m in the source code [6] were used to read and load the MNIST dataset.

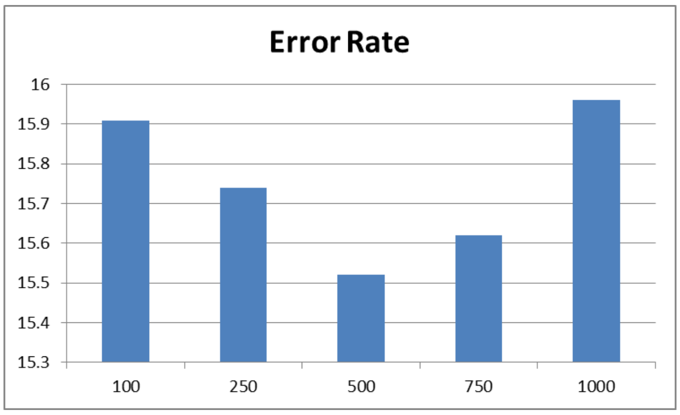
III. AdaBoost Classification

Only 1000 training samples were taken into account to start with. Parameters were tuned by using different kernels and setting their corresponding parameters. At first equal normalized weight was given to every sample. Then AdaBoost classifier is run to train on these 1000 samples. For getting the optimal number of learning rounds required for the classification, training is done multiple times with different number of iterations or learning rounds. Following table represents the training error rate for each given number of learning rounds:

Table 1. Accuracy rate for different number of learning rounds

Number of rounds	Accuracy (1000 samples training set)
100	84.09%
250	84.26%
500	84.48%
750	84.38%
1000	84.04%

Table 2. Error Rate (%) for different number of learning rounds



As observed from the error rates for different number of learning rounds, 500 is most suited value. That's why complete train dataset of 60000 records is trained with **500 learning iterations**. When checked again on entire test set, an accuracy of **94.56%** was achieved.

IV. Gaussian Process Classification

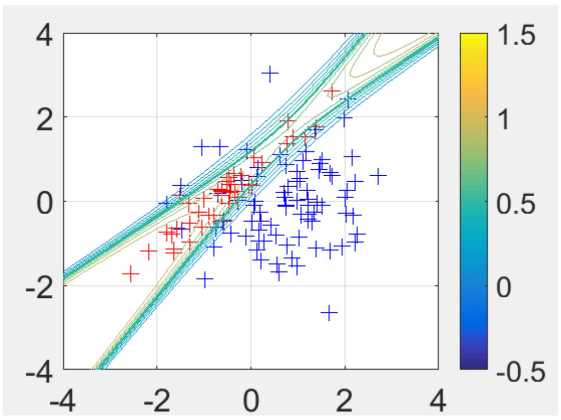
Sample code [5] for Gaussian Process classification in MATLAB was downloaded and installed, after which a script [5] was run to demonstrate the classification process using Gaussian Processes. The default settings for the Gaussian Process model are as follows:

a) A constant mean function, with initial parameter set to 0.

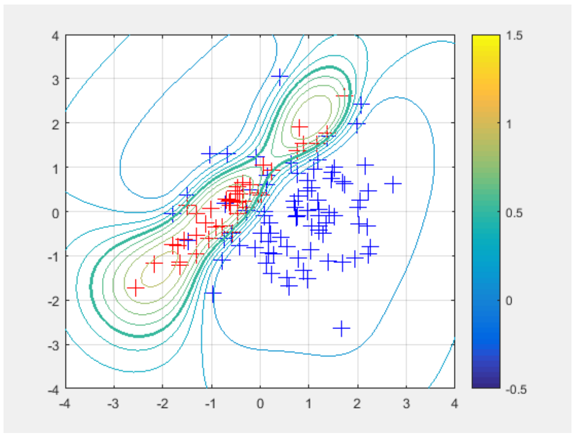
b) a squared exponential with automatic relevance determination covariance function covSEard

c) the likelihood function likErf has the shape of the error-function

Following plot (achieved after running the demo code [6]) shows the data points with the posterior equi-probability contour lines for the probability of class two given complete information about the generating mechanism.



Hyperparameters are trained using function to minimize negative log likelihood. Expectation Propagation (EP) @infEP is the inference method used. The contour plot for the predictive distribution is shown below:



As can be noticed clearly, the predictive probability is close to the probabilities of the generating process in the regions of high data density.

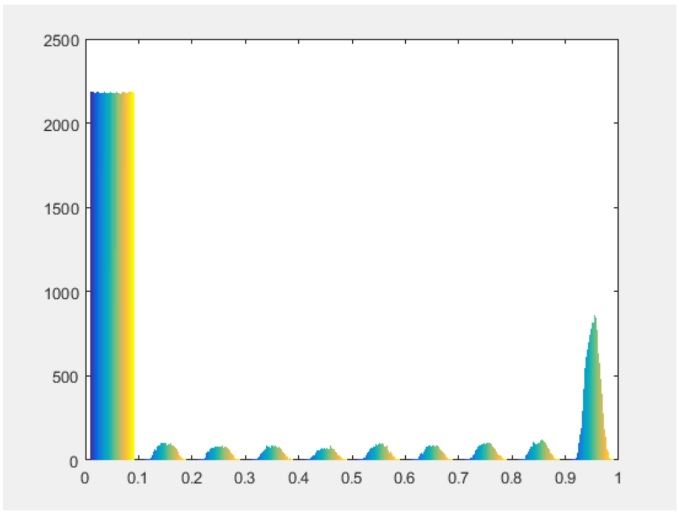
Several other settings were tried for the hyperparameters which are listed here:

Likelihood Function	Inference Method
likUni (Uniform)	infLaplace
likLogistic	infExact
likErf	infExact
likUni	infEP

V. Classifiers' Evaluation

To evaluate Adaboost classifiers with respect to their uncertainty estimation, the test dataset, after classification, was divided into two subsets – one with all records where classifier was correct in prediction, and the other where classifier predicted wrong. hist function [6] of MATLAB with number of bins equal to 10 was used to plot the histograms.

Table 3. Histogram for correctly classified records



References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278-2324, November 1998

[2] Cloah Blog on visualizing the MNIST dataset, http://colah.github.io/posts/2014-10-Visualizing-MNIST.

[3] Robert E. Schapire- "Explaining AdaBoost"
https://www.cs.princeton.edu/~schapire/papers/explaining-adaboost.pdf

[4] Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55(1), 119–139 (1997)

[5] Documentation for GPML Matlab Code version 3.5.
http://www.gaussianprocess.org/gpml/code/matlab/doc/index.html

[6] Source Code: https://github.com/nishantgupta91/MLFCV/tree/master/Exercise3