Machine Learning 2014: Project 2 - Support Vector Machine Report

anguyen@student.ethz.ch
spark@student.ethz.ch
frenaut@student.ethz.ch

November 28, 2014

Experimental Protocol

We first imported the data, added some combination of features, then ran cross validation over candidate C and σ values (see section 2).

Within the cross validation, we normalised the training and validation subsets using the following formula:

$$\hat{x}_{i,j} = (x_{i,j} - \mu_j) / \sigma_j \tag{1}$$

After attaining optimal hyperparameters, we then ran the trainer on the whole dataset to acquire our final model.

Mean and standard deviation of the whole dataset were then calculated to normalise the validation dataset in a method similar to the following:

$$\hat{x}_{i,j}^{\mathrm{val}} = \left(x_{i,j}^{\mathrm{val}} - \mu_j^{\mathrm{train}} \right) / \sigma_j^{\mathrm{train}} \tag{2}$$

1 Tools

Initially plain `MATLAB` and the `MATLAB` statistics toolbox were used but finally we decided to use `MATLAB` with libsvm (written in C) for performance.

2 Algorithm

Custom gradient descents were initially used in the optimisation step of SVM. We tried implementing standard gradient-descent, SGD, batched SGD, and pegasos methods. However we found the resulting errors and time-to-solution unsatisfactory. In this step we were using the primal formulation of soft-margin SVM.

We next used SMO to solve soft-margin SVM for the dual problem. This enabled us to perform feature transformations using the kernel trick.

We tested linear, polynomial, sigmoid kernels but decided to use the radial basis function due to better results.

3 Features

A histogram was made of every given feature, colour-coded depending on class. It was then possible to see that mean intensity values are the only features for which it is relatively easy to see the two distinct classes as separable distributions.

Therefore, we attempted to combine the other two feature types in different ways and monitored the resulting predicted generalisation errors.

We finally decided on adding the following new features:

$$\text{gradient mean} - \sqrt{\text{gradient variance}} \tag{3}$$

4 Parameters

The cost-hyperparameter C for soft-margin SVM and standard deviation parameter σ (for the rbf kernel) were then determined using joint cross-validation.

We noticed cross validation results tended to be quite random when using the built-in `crossval` function. This turned out to be due to the random partitioning employed by the built-in function. We thus implemented our own function which partitions the training dataset in a predictable manner.

To incorporate the variance of errors across different subsets, we added the standard deviation to the mean of errors as a final measure in evaluating the optimality of our hyperparameters. This was to avoid overfitting and create a more robust algorithm.

We used 10-fold cross validation as it performed better and resulted in lower generalisation errors than 15-fold or higher cross validation.

5 Lessons Learned

We focused too much on implementing custom optimisation methods such as SGD as we did not know we could use existing libraries.

Using library functions helped us focus better on other aspects such as tuning cross validation and feature transformations.