Data Mining: Learning from Large Data Sets - Fall Semester 2015

{usamuel, adavison, kurmannn}@student.ethz.ch

November 4, 2015

Large Scale Image Classification

Problem formulation. The goal of this project was to use a SVM to classify a set of images into whether they are images of people, or images of nature. The images were provided in the form of pre-computed feature vectors.

Approach and Results. We computed a classifier by running SGD on each mapper. All mappers read values (y_t, x_t) for $1 <= t <= T$ and then compute w_t using the SGD algorithm that was presented in the lecture. The final result of the mapper is the average of all w_t. The reducer collects all these values and averages them again to output the final classifier w.

We tried several approaches to compute a good classifier for the image classification problem.

At first, we used SGD to compute a linear classifier. The features were used "as is" with the addition of a constant 1. We chose $\eta = 1/\sqrt{(t)}$ and settled for $\lambda = 1$. This approach yielded a score of 60.

To improve on this, we tried to process the feature vectors uniformely at random. We adapted the mapper to first store the entire input in a maxtrix. This matrix was then processd uniformely at random. To make the most of our computation time, we continuted reading from the matrix in random order until the timelimit is reached. In addition to that, we chose $\lambda = 1/T$ and $\eta = 1/(\lambda \cdot t)$. This approach yielded a score of 70.

We were not able to improve on this and therefore assumed that the feature space was not linearely separable. Therefore we experimented with several non-linear transformations of the feature space. We were able to boost our score to 80 with the following transition using built in numpy functions: $f(x) := [numpy.sqrt(x)\ ;\ numpy.diff(x)\ ;\ numpy.gradient(x)]$

Finally we were able to beat the hard baseline by using random fourier features on top of our non-linear transformation $f(x)$ e.g. we took $f(x)$ as input and then computed the random fourier features $z(f(x))$ as shown in the lecture. This gave us a score of 84 (using 3500 dimensions).

Workload distribution. Nico and Alexander wrote the initial structure for the mapper and reducer. Samuel implemented the basic SGD algorithm. Alexander extended it to process samples uniformly at

random and he also came up with the non-linear transition f. Samuel implemented random fourier samples (with errors). Alexander corrected the errors and applied the random fourier features to f. Everybody contributed to the final report.