Assignment 3 - Image Classification

Pritish Sahu - Serena De Stefani

Introduction to AI - CS 520 - 12/18/2015

In this assignment we implement and compare three different algorithms for image classification: Perceptron, Naives Bayes Classifier and Mira.

1 Understanding the problem

1.1

Image classification. In the summer of 1966, Marvin Minsky asked some of his undergraduate students at MIT to connect a computer to a camera and build a program that would have allowed the machine to recognize the content of the pictures taken by the camera. The project was supposed to be suitable for a summer assignment, but fifty years later the problem of image classification is still with us. It turned out to be extremely difficult for a computer to classify images. The problem is hard because when looking at the natural world most of the information is lost, that is, most of the object are partially occluded. In addition, the task is not so much about correctly classifying the light intensities (the inputs) coming from the objects, but to organize the objects in meaningful categories that have only a remote connection with the raw input (the light intensities). Therefore image classification is still a open research question, if not for the simplest situations or for toy problems.

We are going to tackle two of these problems here, the classification of 2D images of digits and the classification of 2D images of faces, where our faces are simplified sketches with the rough appearance of faces.
The classification of digits written with different fonts or writing styles (if handwritten) is of great importance, for example in the automation of tasks such as sorting mail at the post-office. In our case the digits are contained in a text file; each "image" as an area of 28 by 28 pixels. The digits are composed by dashes and pluses, grouped together. The classification of faces is even more relevant in today's society, as millions of photos are uploaded daily on social media and actively searched by users. In this case the dimension of the images will be of 60 by 70 pixels; the faces will be rough sketches assembled putting dashes together in clusters and lines. To solve this problem we will need to work on two fronts. We will need first to identify features that belongs to the different categories we are considering, but in a distinctive way, so that they will help us in the classification process. Then we need to design and implement algorithms that extract the features and weight them on a training set, to return the weights that we will the test with a new set of images.

1.2

The features. The careful selection of features is very important. Feature can range from simple to complex, but complex ones might not necessarily give better results than simpler ones; also adding too many features may be counterproductive, as the accuracy in the recognition task might decrease. We will implement a set of different features in various combinations. We will start with simple features as summing up the numbers of points (that is, of pluses and dashes for digits and of dashes for faces) in all the rows and all the columns. Another feature that we will implement is whether every single pixel in the images is either occupied by a point or by a black space (a binary feature). We will also divide the images into squares of different sizes and we will count the number of points in each square. Another feature that could be particularly useful for faces is that of simmetry, obtained identifying a midline, calculating the numbers of points on each side and comparing them.

1.3

The algorithms. In this report we will implement three algorithm, the perceptron, Naive Bayes classifier and MIRA.

Perceptron. The perceptron algorithm assigns weights to all the different features, for all the possible lables. After reading one image, the perceptron assigns a score to each possible label for that images as:

$$score(f,y) = \sum_i f_i w_i^y$$

Then the perceptron finds the label with the highest score:

$$y^{'} = \arg\max_{y^{''}} score(f, y^{''})$$

And it will adjust the weights accordingly, adding the value of the feature itself to the weights that identify the correct label:

$$w^y = w^y + f$$

And subtracting the value of the feature to the weights the identify a wrong label:

$$w^{y'} = w^{y'} - f$$

Naive Bayes Classifier. The Naive Bayes classifier uses probabilities instead of weights. It models a joint distribution over the label Y and the features. To classify an image, we look for the most probable label given the feature values for each pixel, using Bayes theorem:

$$P(y|f_1, ..., f_m) = \arg\max_y \left\{ logP(y) + \sum_{i=1}^{m} logP(f_i|y) \right\}$$

First we will calculate the prior distribution over label digits, $P(Y)$, estimating it directly from the training data:

$$\hat{P}(y) = \frac{c(y)}{n}$$

then we calculate the conditional probabilities of the features given each label $P(F_i|Y = y)$:

$$\hat{P}(F_i = f_i|Y = y) = \frac{c(f_i, y)}{\sum_{f_i' \in 0,1 C(f_i', y)}}$$

Now we have all the element to apply Bayes theorem. Since the Naive Bayes Classifier calculates and uses conditional probabilities related to the training data, we are making an important assumption: the assumption that the images in the test data will belong to the same population of the images in the training data.

MIRA. MIRA (Margin Infused Relaxed Algorithm) is very similar to the perceptron, as it calculates and assigns weights to features and then chooses the label with the highest score. The main difference with the perceptron is in the way the weights are updated, with a variable step size.

$$w^y = w^y + \tau f$$

$$w^{y'} = w^{y'} - \tau f$$

where tau is such that minimizes the following expression, while being capped by a positive constant C:

$$\tau = min\left\{ C, \frac{(w^{y'} - w^y)f + 1}{2||f||_2^2} \right\}$$

2 Implementation

We ran each algorithm on ten training data samples of different sizes: 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% and finally on the whole training set (100%). For each sample we ran each algorithm five times. Then we averaged the results of images recognized correctly *in the test data* for each cluster of training data used and we plotted the means so obtained, over the percentage of training data analyzed; we also calculated the standard error and we plotted it on the graph. Finally we plotted the line that best fits our results going through the error bars. In addition, we recorded the time necessary for training in each cluster and again we calculated and plotted the means and standard errors over the percentage of training images analyzed.

3 Results

As we started implementing our algorithms, we observed that increasing the number of features did not always bring us a higher accuracy. In particular segmenting the image in squares of different sizes to calculate the numbers of points present in each square did not improve our results but made them worse. Likewise for the symmetry feature, that we thought could be useful for faces. In the end we found that the combination of features that gave consistently the best results was the following: counting the numbers of points in each row and each column and considering each row and column as a feature; considering the individual pixels as features.

In all the figures below, we have on the left graph the number of correct recognitions for the test data as a function of the percentage of training data used; on the right graph the amount of time taken by the CPU to complete the training as a function of the percentage of training data used. The data-points represents the mean values out of five trials; the vertical bars represent the standard errors. After plotting the data-points we fitted, in all cases, both a linear relationship and a quadratic polynomial.

In general all the plots of time over the percentage of training data showed a strong linear relationship, with an almost perfect fit. Instead the plots of number of correct recognition in the test data are best fitted by a quadratic polynomial. The function looks like a truncated parabola with the vertex on top.

Faces - Perceptron. Looking at Fig.1 we see the results obtained for the perceptron for faces. There is a marked improvement as we go from 72.92% of correct recognitions of the test data when 10% of the training data is considered to 84.7% when the whole training set is used for training (see Table 1). The function looks like half a parabola with the vertex at 100% - therefore the improvement is continuous. even though its scale diminishes as the percentage of data analyzed increases. The time plot shows a linear relationship, and there is little variance in the running times obtained.

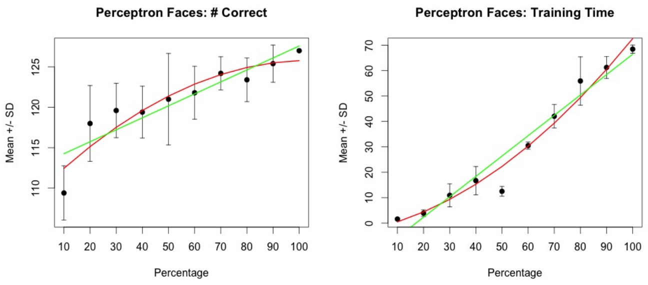

Fig. 1. FACES - PERCEPTRON

Faces - MIRA. Looking at Fig.2 we see the results obtained for the MIRA algorithm for faces. There is an improvement as we go from a mean of 72.66% recognition in the test data for 10% of the training data considered to 86% for the whole set(see Table .1).The line of best fit is markedly curvilinear. In this case the function also resemble a truncated parabola but this time the vertex seems to be reached before the end, at about 70% of training data examined. The time plot shows a linear relationship, but there is slightly more variance in the running times obtained.

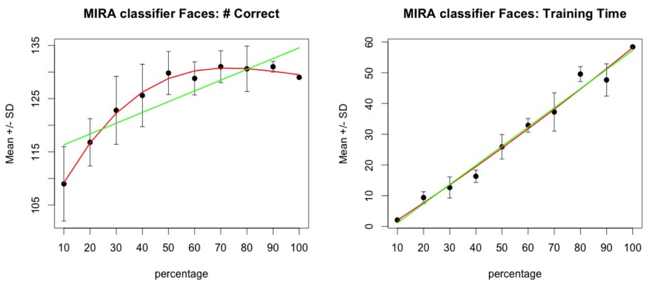

Fig. 2. FACES - MIRA

Faces - Naive Bayes. Looking at Fig.3 we see the results obtained for the Naive Bayes algorithm for faces. There is a marked improvement as we go from a mean of 71.36% of correct recognition in the test data at 10% of the training data tested to 88% at 100% of the training data tested (see Table .1). The function looks like half a parabola with the vertex at 100%. The improvement is very strong at the beginning, but at about 50% of training data analyzed the algorithm reaches a plateau e does not improve much anymore, on the contrary it seems it may get a little worse, and then better again. The time plot follows a linear relationship, and there is almost no variance in the times recorded.

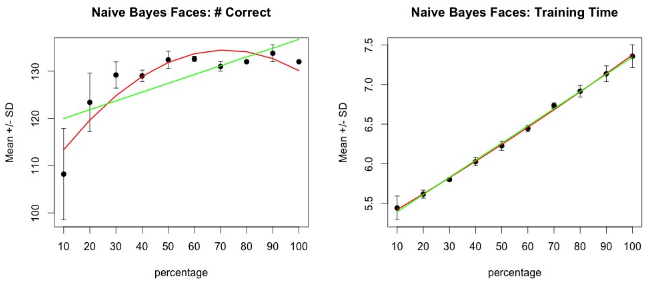

Fig. 3. FACES - NAIVE BAYES

Digits - Perceptron. Now let's consider a Perceptron classifier for digits, whose results are shown in Fig. 4. In this case there is a marked improvement, as we go from a mean of successful recognitions in the test data of 71.9% when using a random sample of 10% of the training data to a mean of 81.9% when considering the whole sample. Considering the time plots, the fit is perfectly linear but less optimal, and it seems there is no variance on the running time recorded; but this is only due to the fact that there are huge differences in the running times for the digits (as the training dataset is much bigger than the one for the faces), so the differences in variance are not detected by the graph.

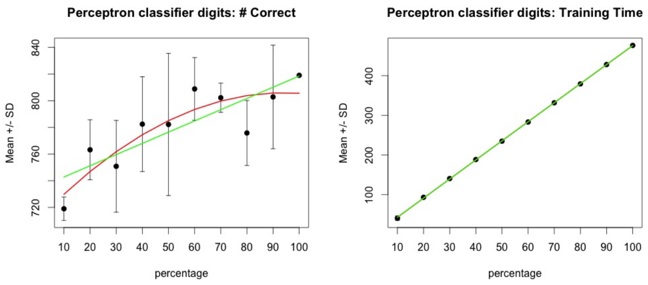

Fig. 4. DIGITS - PERCEPTRON

Digits - MIRA. Here we consider a MIRA classifier for digits, whose results are shown in Fig. 5 (fitting a quadratic equation). In this case there is an improvement, as we go from a mean of successful recognitions in the test data of 77.36% when using a random sample of 10% of the training data to a mean of 81.8% when considering the whole sample. But in this case the percentage gets worse because it gets better again, and the curve looks like a parabola whose vertex is at the bottom. Considering the time plots, the fit is always linear and there seems to be very small variance, but this is also due to the axes scale, as explained above.

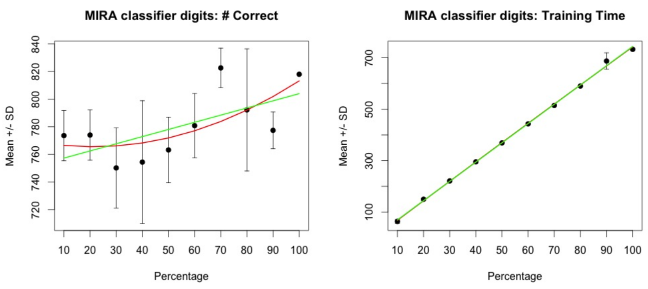

Fig. 5. DIGITS - MIRA

Digits - Naive Bayes. Now let's consider a Naive Bayes classifier for digits, whose results are shown in Fig. 6 (fitting a quadratic equation) and in Fig.8 (fitting a cubic equation). In this case there is just a slight improvement, as we go from a mean of successful recognitions in the test data of 76.6% when using a random sample of 10% of the training data to a mean of 77.4% when considering the whole sample. Considering the time plots, the fit is always linear but less optimal, and there is high variance in the running times recorded.

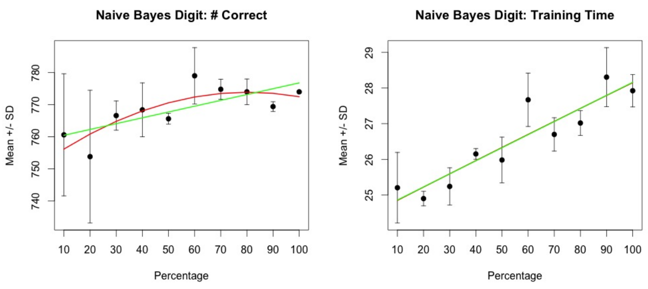

Fig. 6. DIGITS - NAIVE BAYES

4 Summary of results

Table 1. Successful recognition at 10% and 100%

	10% Training	**100% Training**
Perceptron Faces	72.92%	84.7%
Naive Bayes Faces	71.36%	88%
MIRA faces	72.66%	86%
Perceptron Digits	71.9%	81.9
Naive Bayes Digits	76.6%	77.4%
MIRA Digits	77.36%	81.8%