ASI assessed exercise 2016/2017

Garnot Lucas

Instructions to run the code:

Run all the scripts that begin by A0, A1, A2, … in this order for the naïve Bayes classifier and its improvements. Then run the code that begin with B1, B2 for the linear regression. The functions used start with f1, f2, f3 for the classifier and with g1, g2 for the regression. The 'M' in a function means MNIST data and the 'C' means CIFAR data.

1. (code) Download the MNIST and CIFAR10 datasets and import them. [3]

RUN:
A1_load_data
A1_preprocess_data

2. (text) Comment on the distribution of class labels and the dimensionality of the input and how these may affect the analysis. [7]

In the two datasets, the labels are almost equally distributed across the dataset. The computation of the Bayes formula will be easier because we can consider that the probability of one class is one divided by the number of class.

The handwriting image are in gray level (double) and the size is 28x28. The CIFAR images have three channel of unit8 pixels. The size of each image is 32x32. The dimension of the CIFAR images is bigger so the analysis will take more time to compute.

3. Classification

a) (code) Implement the Naïve Bayes classifier. [10]

Functions:
f1_train_naive_bayes_classifier
f2_predict_naive_bayes_classifier

b) (text) Describe a positive and a negative feature of the classifier for these tasks [5]

The positive aspect of this classifier is that it is easy to use and implement because the hypothesis over the independency of the parameters simplifies a lot the Bayes formula. However, if we does not take into account the correlation between the pixels which is something really important in an image. Because of this, this classifier can just detect the value of each pixels but it cannot detect shapes. Moreover, if there is a rotation in the image, the classifier will be totally enable to detect the class anymore.

This classifier is based on the analysis of the value of each pixels. It suppose that there is a little intra-class variance over the value of each pixel. This is more or less the case in the MNIST dataset because each number is centered in the picture, there is no rotation and the images are in grayscale levels. However in the CIFAR dataset, the images are in color and the color itself is not a great indicator of the class. In fact, we can find object of the same class with very different colors, for example a car can be red, blue, gray,… This method is not appropriate for this dataset. We should rather use techniques as pattern recognition, feature extractor with convolutional neural network,…

c) (text) Describe any data pre-processing that you suggest for this data and your classifier [5]

For the MNIST data it is possible to transform the images into binary images. In fact we can still recognize the number with binary images. It will be also easier to compute the probability because we just do not need to use a Gaussian model anymore.

We can also delete the pixels that are in the border of the image because they are always black in the MNIST dataset. It can speed up our computations.

It is also possible to search the best features, that means the features that have a big inter-class variance and a little intra-class variance.

d) (code) Apply your classifier to the two given datasets. Make sure your optimization is clearly commented. Use classification accuracy and test log-likelihood as your figures of merit [15]

e) (code) Display the confusion matrix on the test data [5]

RUN:
A2_train_naive_bayes_classifier
A3_test_M
A4_test_C

f) (text) Discuss the performance, compare them against a classifier that outputs random class labels, and suggest ways in which performance could be improved [5]

Finally I obtained a performance of 84.08% for MNIST and 26.40% for CIFAR. This is better than a random classifier (10%) but the classifier is not really efficient for the CIFAR dataset for the reasons we discussed previously.

In this model, we did not take into account the special feature. In fact we considered independently each pixel. In order to avoid that we can add some features that contain special information to the original data. For example I tried to add the mean of each row and each column (result: 83.92%) and the mean of certain areas (result: 84.68%). The result are slightly better in the second improvement.

Then I try to add features of the PCA transformation of the image. However, the results are not better than the original classifier (82.54% for the data with PCA features added). We observe that the first two PCA features does not separate the classes, except one or two classes.

RUN:
A5_new_features_means_row
A6_new_features_means_areas
A7_new_features_PCA

4. Linear regression

a) (code) Implement Bayesian linear regression (you should already have an implementation from the lab sessions) [10]

Functions:
g1_train_linear_regression
g2_predict_linear_regression

b) (code) Treat class labels as continuous and apply regression to the training data. [15]

c) (code) Produce a scatter plot showing the predictions versus the true targets for the test set and compute the mean squared error on the test set [5]

RUN:
B1_train_linear_regression
B2_test_C
B2_test_M

d) (text) Suggest a way to discretize predictions and display the confusion matrix on the test data and report accuracy [5]

To discretize the predictions, we can round it to the closest integer. Nevertheless I noticed that the linear regression model had difficulties to find the biggest and the smallest labels (0 and 9). When the prediction is above five, it is more likely to be below the true label and inversely the prediction is often above the true label when it is under five. Therefore I decided to round up for predictions above 5.5 and I round down for predictions below 4.5. With this method I improved the results by 8%.

e) (text) Discuss regression performance with respect to classification performance [5]

I obtained 32.88% for MNIST and 10% for CIFAR. Those results are really bad compare to the classifier technique.

f) (text) Describe one limitation of using regression for this particular task. [5]

The poor performances of this method come from the fact that this model is not appropriate for this situation. A linear regression suppose that there is a link between the features (pixels) and the labels and it tries to model this link by a linear dependency. The predicted labels are a linear combination of the features. However in that case, the value of the labels itself have no particular meaning and there is no link between this value and the pixels. We tried to compute artificially a link but we saw in the results that it is not coherent.

5. Bonus question

a) (text / code) The state-of-the-art in these image classification problems suggests that convolutional layers in convolutional neural networks yield most of the improvements compared to standard neural networks. The reason is that they are capable of modeling spatial patterns through the hierarchical analysis of patches of images. Propose and implement ways to exploit patch information in the Naïve Bayes classifier or linear regression. A couple of suggestions are: (i) apply Naïve Bayes classification to the output of convolutional layer in the LeNet architecture (ii) construct the Naïve Bayes classifier by calculating patch-specific statistics and extend this by stacking multiple of these

To try to improve the model, I continued on searching new features. I tried to model the first part of a convolutional neural network in order to extract new features. I did not use back propagation algorithm to train the filters of the convolutional layer so I used edge detector filters and random filters. I did not use ReLu layer because the result was worse with this layer and I used a mean pooling layer.

With just one filter and one layer I obtain almost the same result than before. When I use three edge detector and two layers, the results are worse (68.82%). When I use three random filters with two layers, the results are usually between 45% and 65%.

To improve this model the best thing to do would be to train the filters with the back propagation algorithm and then use the naïve Bayes classifier on the extracted features. But we can also try a more naïve method with a genetic algorithm. For example we initialize three random filters and we compute the predictions with each pair of features. We keep the pair that gives the best results and then we generate a new third filter with a combination of the two filters that we kept. We continue this method a certain number of time and we keep the three filters that give the best results. This method may take a lot of time because there is an important random component. Moreover, we should choose a clever way to generate the third filter.

Functions:
h1_compute_next_layer
h2_extract_feature

RUN:
A8_convnet