2^{st} mini project - K-means

Sebastien Duc

November 27, 2011

About the project

To run the project, type `python kmeans.py`. Note that convergence can take some time. There is just one python file. All functions needed for this project are in this file. There are 3 main functions. The first is `normal(improved)` which takes `improved` (a `Boolean`) as parameter. This is used for the first part of the project. When `improved = False`, the vanilla k-means is used on the data set. When `improved = True`, the dead unit reducing version is used.

The second function is `reduced(improved)` which takes again a `Boolean` as parameter. This function is used for part 1.1. It first reduces the data set using PCA and then applies k-means on it. Again `improved` allows us to choose between the standard and the improved version of the algorithm.

The third function is `blind(reduction)`. `reduction` is either `None`, in that case we do not reduced the data set before applying k-means. Or `reduction` is an `Int`, in that case we reduce the data set by keeping only `reduction` dimensions and then apply k-means.

Part 1.1

The basic algorithm

The algorithm used for this project is the batch version of k-means. First we initialize the prototypes with random elements from the data set. Then, while the clusters have not converged, first we assign for each data element of the data set the closest prototype. Then we update the prototypes using the batch rule

$$\text{For each prototype } p \text{ that represent cluster } C_p \quad p \leftarrow p + \mu \sum_{e \in C_p} p - \text{DataSet}[e]$$

The criterion of convergence we used is the number of data elements of the data set that changes cluster. If no data elements changed cluster then we say that the algorithm has converged. See Figure 1 to have a plot of the time course of our convergence measure.

For the prototypes, see Figure 2. To get all the digit, $k = 10$ is not enough because some clusters represent the same digit. With $k = 20$ we have most of the time all digits.

To detect dead units, we look at the number of data elements that are in the cluster. If this number is less than some threshold (for example 50), then we say that it is a dead unit. Dead units appear only when k is big.

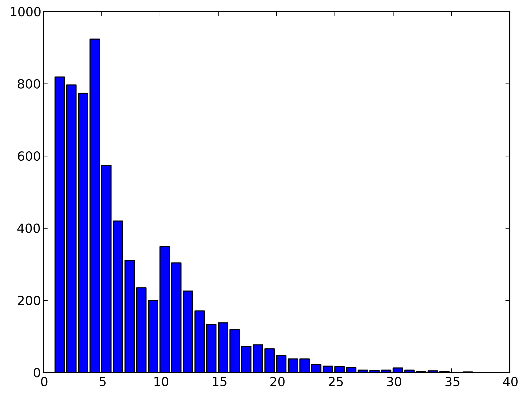


Figure 1: Time course of convergence measure

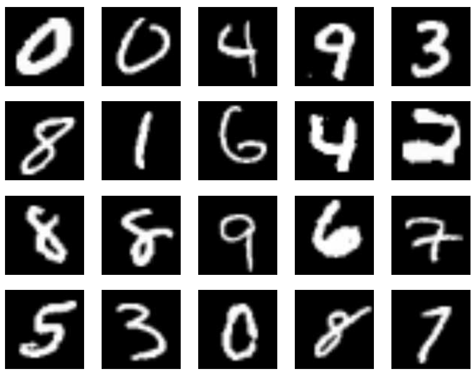


Figure 2: Prototypes with $k = 20$

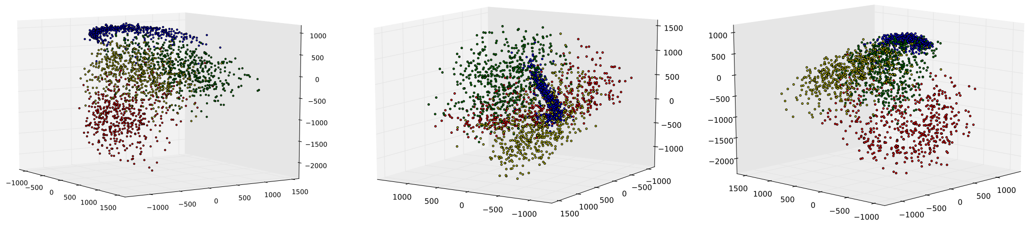


Figure 3: Subsets of the data set after PCA

We also computed the shannon entropy of the clusters, but we need to have the label to do this. The entropy is computed as following: for each cluster C find the number of data elements n_i with label i (i.e. that represent digit i) normalize them ($p_i \leftarrow \frac{n_i}{\sum_i n_i}$). Then $H(C) = \sum_i p_i \log \frac{1}{p_i}$. A low entropy means that the cluster represent a single digit and a high entropy means that the cluster represent a mix of digits.

The improved algorithm

The purpose of this new version is to avoid dead units. To achieve this, we changed a bit the initial algorithm. First each time we update the clusters, we assign with low probability the data element to a random cluster. Also, we decrease the learning rate ν.

We also tried to improve the initialization, so that the initial prototypes are far away for each other. For the first prototype, pick one data element at random. Then pick a next data element with probability proportional to the square of the distance to the nearest prototype already chosen until we have k prototypes. This method didn't really increased the preformances. When doing statistics on 20 trials, we got on average 2.5 dead units for the initial algorithm and 2.0 for the improved one.

To detect clusters that represent the same class, we can compute the correlation matrix of the prototypes. A high correlation means the the clusters represent the same cluster.

Part 1.2

In this part we reduced the size of the data set. First we kept only data with label 0,1,2,3. And then we applied PCA on this set, by keeping only 4 dimensions. In Figure 3 we plotted some subsets of the 4-dimensional reduced set. Label 0 is in red, label 1 in blue, label 2 in green and label 3 in yellow. One can clearly see some clusters in these plots.

Then when we apply k-means on this reduced set we get 4 by 4 digit pictures. One can see them in Figure 4 Then we also applied k-means on the reduced set but keeping 25,64 and 400 dimensions. One can see the prototypes in Figure 5.

Part 2.1

The prototypes are available in the file `blindprototypes.txt`. The algorithm was run with $k = 20$. The prototype 7 is probably a dead unit

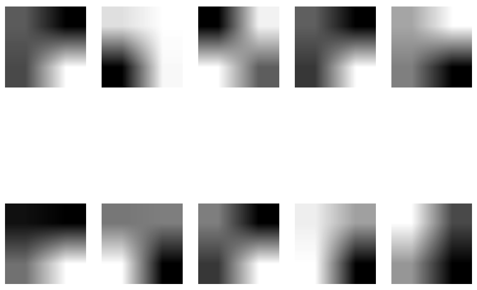
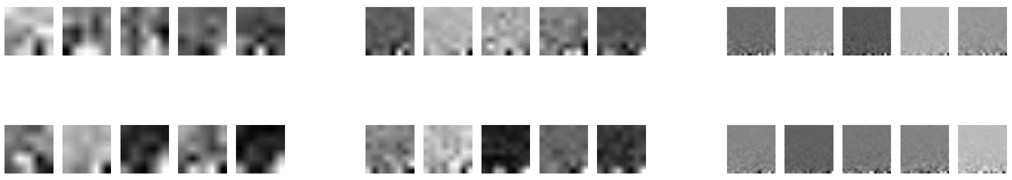


Figure 4: Reduced prototypes with $k = 10$, and keeping 4 dimensions



(a) Dimension is 5×5 (b) Dimension is 8×8 (c) Dimension is 20×20

Figure 5: Reduced prototypes with $k = 10$, and keeping 25,64,400 dimensions