Image Processing and Pattern Recognition

Assignment 5 - Sparse Coding and Classification

January 16, 2017

Deadline: January 30, 2017 at 23:59h.
Submission: Upload report and implementation zipped (*[surname].zip*) to the TeachCenter using the "My Files" extension.

1 Goal

In this exercise we aim to do classification with the help of sparse coding. Classification is one of the most important research areas in computer vision, with applications ranging from interpreting number plates of cars to automatic text recognition.
The dataset we will use for this exercise is the well-known MNIST [2] database of handwritten digits. Instead of using the images directly for learning a classifier, we first compute a sparse coding of the training data. Then we learn the classifier on the representation coefficients of the sparse code, which ideally should give a boost in classification performance.

2 Method

The sparse coding problem is stated as

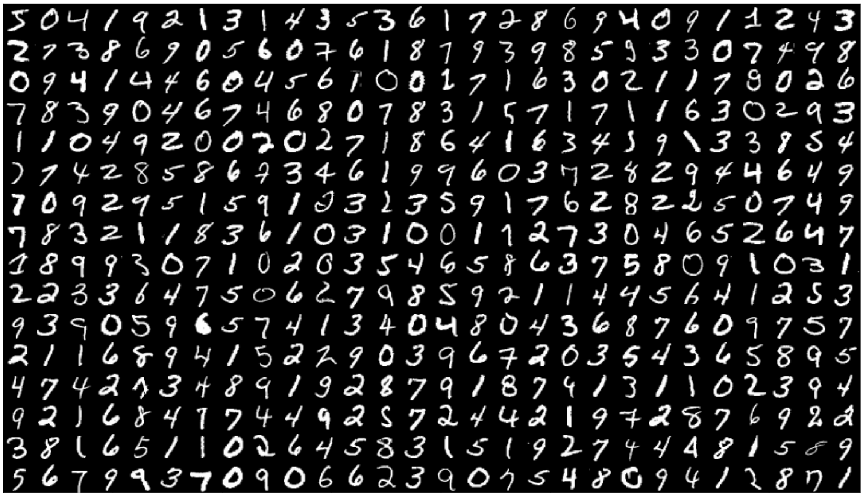
$$\min_{D,C} \frac{1}{2}\|DC - B\|_2^2 + \lambda\|C\|_1, \tag{1}$$

where $D \in \mathbb{R}^{MN\times L}$ is the dictionary, $C \in \mathbb{R}^{L\times K}$ are the representation coefficients and $B \in \mathbb{R}^{MN\times K}$ is the training data. The 2-norm for a matrix $A \in \mathbb{R}^{m\times n}$ is defined as $\|A\|_2 = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} A_{ij}^2}$.
This model is a variant of the LASSO-model, where we optimize for both the dictionary D and the representation C. The meaning of the variables is as follows:

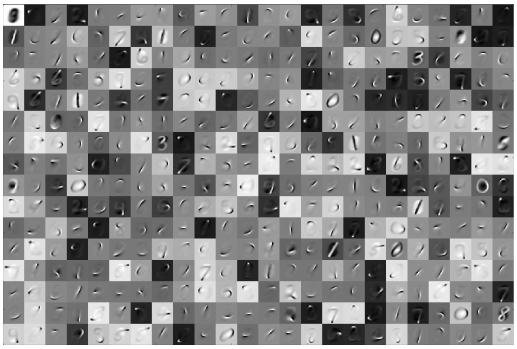
- The columns of B contain the training data. In our case, each column is a vectorized training image of size $M \times N$ from the MNIST database.
- The columns of D are the dictionary atoms.
- Each column of C contains the sparse code for one of the images w.r.t. the dictionary D. In total there are K images, and typically we have $L < MN$.
- The compression comes from the fact that using the optimal basis D, the data has a lower-dimensional representation.

In order to optimize (1), we constrain the atoms of the dictionary to have norm less or equal to one. Additionally we allow the coefficients for the first dictionary atom to freely take any values, i.e. we do not penalize the first row of C. The modified model for sparse coding reads

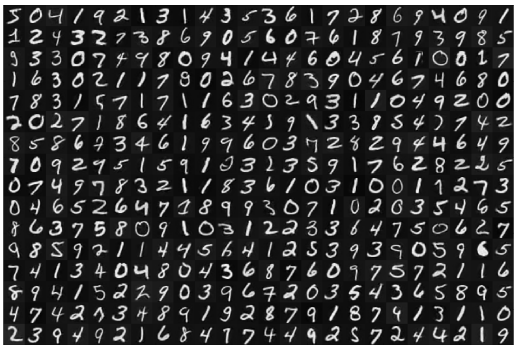
$$\min_{D,C} \frac{1}{2}\|DC - B\|_2^2 + \lambda\sum_{l=2}^{L}\|C_l\|_1 \quad \text{s.t. } D \in A \tag{2}$$



(a) First 448 images of MNIST



(b) Learned dictionary with 384 atoms



(c) Reconstruction of the first 384 digits

Figure 1: Sparse coding of the MNIST database of handwritten digits.

where the convex set A is defined as $A = \{D \in \mathbb{R}^{MN \times L} \ : \ \|D_l\|_2 \leq 1, \ l = 1 \dots L\}$. Note that $C_l \in \mathbb{R}^K$ denotes the rows of the matrix C, whereas $D_l \in \mathbb{R}^{MN}$ are columns of the matrix D.
To optimize (2), we employ the iPALM algorithm, an accelerated version of the proximal alternating linearized minimization (PALM) [1] algorithm. This algorithm can solve non-convex, non-smooth problems that exhibit a certain structure by executing block coordinate-descent steps: Alternatingly one of the variables is fixed, and a gradient descent step followed by a proximal projection is computed on the other variable. Writing problem (2) as

$$\min_{D,C} f(D,C) + g(C), \tag{3}$$

where $f(D,C) = \frac{1}{2}\|DC - B\|_2^2$ is the smooth part of (2) and $g(C) = \lambda \sum_{l=2}^{L} \|C_l\|_1$ is the non-smooth part, the algorithm is given by the following iterations for $k > 0$

$$\begin{aligned}
\beta^k &= \frac{k-1}{k+2} \\
\tilde{C}^k &= C^k + \beta^k (C^k - C^{k-1}) \\
\tilde{D}^k &= D^k + \beta^k (D^k - D^{k-1}) \\
C^{k+1} &= \operatorname{prox}_{\tau g}\left(\tilde{C}^k - \tau \nabla_C f(D^k, \tilde{C}^k)\right) \\
D^{k+1} &= \pi_A\left(\tilde{D}^k - \sigma \nabla_D f(\tilde{D}^k, C^{k+1})\right)
\end{aligned}$$

Here, $\pi_A(\cdot)$ denotes the projection of D onto the convex set A, which can be computed by dividing every column D_l of D by $\max(1, \|D_l\|)$. $\operatorname{prox}_{\tau g}(\cdot)$ denotes the proximal mapping w.r.t. the function $g(\cdot)$. For a point $\hat{x}$, it is defined as the solution of the following optimization problem

$$\operatorname{prox}_{\tau g}(\hat{x}) = \min_x \frac{\|x - \hat{x}\|^2}{2\tau} + g(x) \tag{4}$$

In case $g(x) = \lambda\|x\|_1$ this is a pointwise problem and can be solved by applying a soft-thresholding to every element of $\hat{x}$.
The timesteps are given by $\tau = \frac{1}{\|(D^k)^T D^k\|}$ and $\sigma = \frac{1}{\|C^{k+1}(C^{k+1})^T\|}$, where $\|\cdot\|$ denotes the operator norm[1].

2.1 Tasks

- Implement the sparse coding as explained in section 2 to compute the optimal dictionary on the training set. Make sure to run enough iterations such that the optimization converges.
- Use the linear SVM from the `scikit-learn` package (`sklearn.svm.LinearSVC`) to train a SVM
  - on the original MNIST training database
  - on the sparse code for the MNIST training database
- Solve the LASSO model to compute the representation coefficients of the test set (use the iPALM algorithm). Classify the test set with your trained SVM.
- Compare the classification error of the SVM trained on the original data with the error on the sparse code. What do you find?

  Before training the SVM, augment the sparse code $C_k \in \mathbb{R}^L,\ k = 1 \dots K$ for each image by $\hat{C}_k = (C_k, C_k \odot C_k)^T$, where $\odot$ denotes element-wise multiplication. How does this affect the classification?
- Try different settings for computing the sparse coding (number of atoms, regularization). Include your findings in the report!
- Show some of the images where classification failed.

Hints

- Scale down the problem for implementing/checking the optimization algorithm (e.g. 1000 digits, 32 dictionary atoms).
- Use the visualization functions in the framework for debugging.
- Plot the energy (2) during iterations to check convergence.

3 Framework

The framework contains the basic structure as well as some functions you might find useful. We use the Linear SVM from the `scikit-learn` module for computing the SVM.

References

[1] Jérôme Bolte, Shoham Sabach, and Marc Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Math. Program.*, 146(1-2):459–494, August 2014.

[2] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[1] The operator norm is given by the largest singular value and can be computed e.g. by using the function `np.linalg.norm()` (Frobenius norm).