MAD 2018-19, Assignment 4

Jonas Peters, Fabian Gieseke, Aasa Feragen

hand in until: 02.01.2019 at 23:59
(note: this is the first Wednesday in 2019)

General comments: The assignments in MAD must be completed and written individually. You are allowed (and encouraged) to discuss the exercises in small groups. If you do so, you are required to list your group partners in the submission. The report must be written completely by yourself. In order to pass the assignment, you will need to get at least 40% of the available points. The data needed for the assignment can be found in the assignment folder that you download from Absalon.

Submission instructions: Submit your report as a PDF, not zipped up with the rest. Please add your source code to the submission, both as executable files and as part of your report. To include it in your report, you can use the `lstlisting` environment in LaTeX, or you can include a "print to pdf" output in your pdf report.

Non-Linear Regression

For the following two exercises, use `numpy.linalg.solve` to solve the induced systems of linear equations instead of inverting the corresponding matrices (as discussed during the lecture, inverting matrices can become numerically unstable). Note that you **are allowed** to make use of all the code discussed during the lectures!

Exercise 1 (**Polynomial Fitting with Linear Regression, 4 points**).

Baron Münchhausen has mounted a cannonball and tries to fly to the moon! He reports his coordinates (horizontal distance from the cannon, height):

$$(1, 8.57), (2, 14.33), (3, 21.13), (4, 24.29), (5, 23.85), (6, 25.19), (7, 22.19), (8, 15.96)$$

It is well-known that, in presence of gravitation, bodies fly on parabolas (i.e., their flight path can be modelled via a function $f(x) = c + bx + ax^2$ with $a, b, c \in \mathbb{R}$). It is also known that the position of the cannon is $(0, 0)$, which means that your model must *pass through (0,0). Solve the following questions under the assumption that the horizontal distances are exact, but the height observations are noisy.*

a) Use linear regression to estimate the parameters of the parabolic trajectory of the flight. Remember that the model must pass through (0,0). However, it does not have to pass exactly through the observed points.

b) Estimate the location (i.e., distance from the cannon) where Baron Münchhausen will fall down.

c) Plot the position of the cannon, the reported coordinates, and the estimated trajectory in one figure to see how good your approximation is.

Deliverables. *a) Brief explanation of how you solved the problem (either the equations you have used or your code) and the parameters you got, b) brief explanation of how you computed the distance (the equation you used) and the outcome of your calculation, c) the plot. Also add your source code to your submission!*

Exercise 2 (Polynomial Fitting with Regularized Linear Regression and Cross-Validation, 4 points). *In this exercise, you will apply leave-one-out cross-validation to study the influence of the regularization parameter λ on the predictive performance of regularized linear regression. In the* `men-olympic-100.txt` *file, you will find data on the men's Olympic 100m running times. Take the first place running times (second column) as target values $t_1, \ldots, t_{27}$. The input variables $x_1, \ldots, x_{27}$ are given in the first column.*

a) *Apply polynomial fitting with regularized linear regression to fit a* ***first order polynomial*** *to the data. Plot the leave-one-out cross-validation error (y-axis) as a function of λ for some values $\lambda \in [0, 10]$ (x-axis); use* `numpy.logspace(-20, 1, 200, base=10)` *to generate the values to be tested. Report the lowest cross-validation error as well as the associated best value of λ. Further, report the optimal regression coefficients $\hat{\mathbf{w}}$ corresponding to both $\lambda = 0$ (no regularization) and the best value of λ.*

   Hint: Be aware of the fact that the values outputted via `print` are usually rounded. Make use of a higher precision when printing, e.g., via `print("lam=%.15f and loss=%.15f" % (lam, loss))`.

b) *Repeat the same for fitting a* ***third order polynomial*** *to the data.*

Deliverables. *a) The plot, the lowest cross-validation error, the best value of λ, and the two sets of regression coefficients; b) the plot, the lowest cross-validation error, the best value of λ, and the two sets of regression coefficients. Add your source code to your submission!*

Principal Components Analysis (PCA)

In the following two exercises we shall use PCA for dimensionality reduction of the MNIST handwritten image dataset, and show how the image content can be stored using far less than their 784 coordinates corresponding to the image pixels. **See the appendix for a description of the data.**

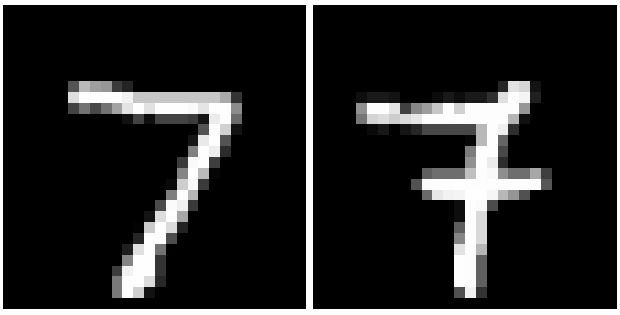

Figure 1: Two different images of the integer seven from the MNIST dataset.

Exercise 3 (PCA I, 4 points). *In this exercise, you will implement PCA and use it for 2-dimensional visualization and analysis of variance as in the lecture.*

a) *Implement PCA using eigenvalue decomposition based on the supplied template* `pca.py`*, and use it to find all principal components of the combined dataset containing all MNIST images of the numbers one and seven, which you find in the files* `MNISTsevens.txt` *and* `MNISTones.txt`*.* **Tip:** *You may want to use* `np.linalg.eigh` *rather than* `np.linalg.eig`*, as it actually sorts the eigenvalues for you. For verification, please include the largest eigenvalue in your report.*

b) *Project the (centered) data onto the two first principal components, and visualize the projected data using a scatter plot, where you give different colors to the ones and the sevens. What do you see?*

c) *Write a function which computes, for any $k \leq 784$, the proportion of variance explained by the first k principal components. Plot the variance explained as a function of k for all $k = 1, \ldots, 784$. How much variance is explained by the first two? How many principal components do you need to explain 95% of the variance? What does this tell you about your visualization in b) and any conclusion you might draw from it?*

Deliverables. *a) The adapted* `pca.py` *file containing your source code, and the largest eigenvalue obtained on the entire dataset; b) A code snippet containing the most crucial lines of code, the resulting scatter plot, and a short discussion; c) The plot, answers to the questions, and a short discussion.*

Exercise 4 (PCA II, 4 points). *In this assignment, we shall use our knowledge about PCA to move beyond replicating what you have already seen, by applying PCA to low-dimensional encoding of high-dimensional data. By projecting the original data onto the subspace spanned by the first k principal components, you will obtain a k-dimensional encoding of the original data. We shall perform the corresponding reconstruction to visually assess how good the encoding is.*

a) Start out working only with the images of the number seven. Using the supplied template `pcarecon.py`*, perform PCA only on these, and make increasingly good reconstructions of the first and tenth image by projecting onto an increasing number of principal components. Please plot your projections onto the first principal component, as well as the first 5 and the first 20 (advice: In LaTeX you can use the package* `subfigure` *to put several figures next to each other). What do you observe as you use more and more principal components?*

b) Also run your PCA and your reconstruction on the MNIST integer ones. Do you see a difference in the number of components needed for a visually good reconstruction? Can you verify your observation by looking at the proportion of variance explained for the two different datasets? Can intuitively explain your observation?

Deliverables. *a) The adapted* `pcarecon.py` *file containing your source code. Include in your report the three reconstructions as images as well as a short discussion. b) Reconstructions, a discussion, description of proportion of variance explained for the two datasets and an intuitive explanation.*

A The MNIST images

MNIST [1] is a standard classification benchmark dataset consisting of a 28×28 grayscale images of handwritten digits. These images are usually stored as 28×28 matrices or arrays, but such an image can also be represented as a 784-dimensional vector consisting of concatenated image columns.

You will find the MNIST data corresponding to digits 1 and 7 in the files `MNISTones.txt` and `MNISTsevens.txt`, which you can download from Absalon. These files are data matrices, where each column is a 784-dimensional vector corresponding to a single image, and you can form the images back by using the function `np.reshape`. Visualize one of the images to make sure you reshaped the vector correctly!

References

[1] Y. LeCun, C. Cortes, C. Burges, *MNIST handwritten digit database*, `http://yann.lecun.com/exdb/mnist/`, 2010.