Course: Complex Data - Analysis and Visualization, 2016

©Computational Medicine, Dept. Medical Sciences, Uppsala University (MG+EC)

Version: 161126

Exercise #4- Deep Learning

This week you will be studying an alternative way of initializing networks for deep learning!

Task A - PCA for linear data compression

Working towards speeding up and improve the neural network learning using the image data sets from the previous assignment, perform PCA of the training set and then compress all the images from 784 dimensions to M dimensions where M is selected so that the images can be reproduced reasonably well. What is reasonable you can evaluate yourself simply by visual inspection of some randomly selected pairs of images obtained before and after reconstruction. When you have determined the suitable number of dimensions M which produces acceptable reconstruction according to your visual inspection, determine the corresponding mean squared error E_M you have on the test data. I other words, determine the mean squared error E_M on the test data obtained when the compression yields visually acceptable reconstructions. The values M and E_M will be used in tasks below.

Reminder: For each image represented by a 784-dimensional column vector $\mathbf{y}_n$ you should compress it to a M-dimensional vector as $\mathbf{t}_n = B^T(\mathbf{x} - \mathbf{m})$ where the matrix B consists of M eigenvectors (as columns) and $\mathbf{m}$ is the mean vector. The reconstruction of $\mathbf{x}_n$ is then $\mathbf{y}_n = \mathbf{m} + B\mathbf{t}_n$ and you get the mean squared error as the mean across all the squared errors $||\mathbf{x}_n - \mathbf{y}_n||^2$.

Task B - linear data compression using a ”bottle-neck” neural network

Build a ”bottle-neck” neural network having structure 784-M-784 having only linear nodes (both in the hidden and output layer). Determine if you are able to get the same reconstruction errors on the test examples using this approach as when using PCA above. Run the same code 10 times each time with different random start guesses on the values of the weights and biases in order to determine if you can achieve the same small reconstruction error as by means of the PCA based initialization.

Remark. There is a high risk that some of the M hidden neurons will become almost identical whereas the goal is to make them correspond to different projections of the input data. When some neurons become identical, this corresponds to a local minumum where the mean squared reconstruction error will be larger than E_M from task A. Therefore also consider testing different learning algorithms and changing the learning (optimization) termination parameters, if needed.

Task C Network initialisation based on stacked PCA

Write code that initializes the weights and biases of a network with structure 784-100-M-100-784 (where M is what you selected in Task A) having $\tanh(x)$ as the non-linearity, by means of PCA (principal component analysis) as described below (i-vi). You should confirm that your code for initialization works properly by comparing the reconstructions produced by the network directly after initialising the weights and biases (no further training) with the corresponding reconstructions obtained when using PCA to first reduce the input to 3 dimensions and then directly reconstruct the compressed vectors back to 784 dimensions. The reconstruction errors should be more or less identical.

(i) Perform PCA based on the input vectors $\mathbf{x}_n$ and their mean vector $\mathbf{m}_1$). Consider the resulting compression to 100 dimensions $\mathbf{v}_1(n) = B_1^T(\mathbf{x}_n - \mathbf{m}_1)$ obtained by keeping as columns in the matrix B_1 the 100 eigenvectors corresponding to the 100 largest eigenvalues λ_i of the covariance matrix ordered according to the size of the eigenvalues. Then rewrite this expression as $\mathbf{v}_1(n) = W_1\mathbf{x}_n + \mathbf{b}_1$ and note that the weight matrix W_1 and the bias vector $\mathbf{b}_1$ in the last expression are defined as $W_1 = B_1^T$ and $\mathbf{b}_1 = -B_1^T\mathbf{m} = -W_1\mathbf{m}_1$. This weight matrix and bias vector might seem okay to use in the first hidden layer of the neural network to be initialised but we will ensure that there will not be any severe saturation problems, due to the node non-linearity $\tanh(x)$. This will be achieved in the next step (ii) by making all the inputs to the first layer have the standard deviations equal to or below s_{in} where s_{in} is a user-defined parameter.

(ii) To ensure that the values that go into the first hidden layer of neurons all have the same standard deviation s_{in}, create a diagonal matrix D_1 with diagonal elements $D_1(i,i) = s_{in}/\sqrt{\lambda_1}$ where λ_1 is the largest eigenvalue. Let $s_{in} = 1$ and then determine

$$W_1 = D_1(B_1)^T \text{and } \mathbf{b}_1 = -W_1\mathbf{m}.$$

By calculating the covariance matrix for the resulting vectors $\mathbf{v}_1(n) = W_1\mathbf{x}_n + \mathbf{b}_1$, confirm by means of Matlab computations that this choice will result in vectors an consisting of uncorrelated elements having standard deviations equal to or smaller than 1.

(iii) Determine the node outputs of the first hidden layer "manually" as $\mathbf{z}_1(n) = f(\mathbf{v}_1(n)) = f(W_1\mathbf{x}_n + \mathbf{b}_1)$ where $f(\mathbf{v})$ is a vector valued function that for an input vector $\mathbf{v}$ returns an output vector of the same dimensionality with the i:th element output determined as $\tanh(v_i)$.

(iv) Repeat the same PCA based procedure as above once again, this time applied to the first hidden layer outputs $\mathbf{z}_1(n)$: More specifically, compress the 100-dimensional vectors $\mathbf{z}_1(n)$ using PCA, plus scaling, into M dimensions to create the next layer of weights W_2 and biases $\mathbf{b}_2$ using another eigenvector matrix B_2, mean vector $\mathbf{m}_2$, and diagonal scaling matrix D_2:

$$W_2 = D_2(B_2)^T \text{and } \mathbf{b}_2 = -W_2\mathbf{m}_2$$

Finally calculate the output from the second hidden layer as $\mathbf{z}_2(n) = f(\mathbf{v}_2(n)) = f(W_2\mathbf{z}_1(n) + \mathbf{b}_2)$ after choosing the diagonal matrix D_2 so that the elements of the vectors $\mathbf{v}_2(n) = W_2\mathbf{z}_1(n) + \mathbf{b}_2$ have standard deviation 1.

(v) Determine the weights and biases of the 3rd and 4th hidden layers as follows. Since essentially all input values to the neurons will be in the linear range of the sigmoidal nonlinearity $\tanh(x)$ the neuron as $\tanh(x) \approx x$. Thus we may approximate the neurons as being linear for the current dataset. This means that the total linear inputs $\mathbf{v}_2(n)$ as well as the outputs from the second hidden neurons can be written as $\mathbf{z}_2(n) = \mathbf{v}_2(n) = D_2B_2^T(\mathbf{z}_1(n) - \mathbf{m}_2)$. Multiplying by D_2^{-1} yields $D_2^{-1}\mathbf{z}_2(n) = B_2^T(\mathbf{z}_1(n) - \mathbf{m}_2)$. Thus $D_2^{-1}\mathbf{z}_2(n)$ may be interpreted as score values from a PCA analysis that yielded the mean vector $\mathbf{m}_2$ and the column (eigen) vectors in B_2. From what we know from PCA this shows that the inputs $\mathbf{z}_1(n)$ can be reconstructed as $B_2D_2^{-1}\mathbf{z}_2(n) + \mathbf{m}_2$. In conclusion the weight and biases for the 3rd hidden layer should be

$$W_3 = B_2D_2^{-1} \text{ and } \mathbf{b}_3 = \mathbf{m}_2$$

Analogously, the weights in the 4th layer should be

$$W_4 = B_1D_1^{-1} \text{ and } \mathbf{b}_4 = \mathbf{m}_1$$

(vi) Show that a neural network with structure 784-100-M-100-784 and nodes having the non-linearity tanh(x) in all layers (except the output layer which should have linear nodes) produces quite good reconstructions of the input data when having weights determined using the PCA based approach described above according to (i-v)! Also report the total mean squared error on the training examples.

Task D - Comparing initialisation based on PCA versus using stacked autoencoders

Compare the speed of learning as well as the resulting classification performance of fine tuned (fully trained using supervised learning) networks of structure 784-100-M-1 (designed for detection of the digit ”3” as in the task form last week) when the nets have been initialised using PCA, as suggested in task C above, and when initialised using auto-encoders as in the previous task from last week. Any gains using PCA based weight initialisation?

Task E - Non-linear data compression using a “bottle-neck” neural network

The basic question here is to determine if you can use a non-linear “bottle-neck” neural network with 3 hidden layers to compress the images of your image dataset from last week down to fewer dimensions than M, as obtained with PCA in task A above, while retaining the same level of reconstruction error. Therefore, design a “bottle-neck” neural network with 3 hidden layers having structure M-M-H-M-M having linear nodes only in the last layer. Use the M-dimensional compressed vectors from task A as inputs. Determine how many nodes H you need in the second hidden layer in order to reproduce the 5003 images with a mean squared error less than or equal to the value E_M obtained in task A. In order to avoid local minima you should train the network at least 10 times using different random initialisation values for the weights and biases.

Report

You should write a nice report that explains your solutions and provide well written and well documented **source code** which should be straightforward to download and run by your teacher (including a readme.txt file with a description of your m-files files as well as running instructions). The report should have a clear structure with the following titles or something similar.

Introduction: Introduce the exercise and summarize the subtasks.
Material and Methods: Present an overview of the strategies and tools used to solve the different subtasks.
Results: Present selected results to answer specific questions and/or to show particular observations of interest.
Discussion: Present a discussion of your results in relation to the subtasks assigned.
Conclusion: Present your conclusions from the results presented.
Evaluation: Present a short evaluation (approx half a page) of the exercises in terms of what you have learned and what you think can be improved.

Important Dates

Oral Presentation: Friday, December 2.

Written Report: Monday, December 5.

Good Luck!