Artificial Neural Network & Deep Learning Q & A Summary

Hicham Mohamad, hi8826mo-s
hsmo@kth.se

October 3, 2022

1 Introduction

This summary is intended to give an introduction to artificial neural networks and deep learning. Artificial neural networks (ANN) represents a technology that has connections to many disciplines such as neuroscience, computer science, mathematics, statistics and physics. This course will provide you with enough knowledge to use neural networks and deep learning in practical applications. We will study ANNs from both a theoretical and a computational/practical point of view. The course material is divided into different parts:

1. Feed-forward neural networks
2. Ensemble machines
3. CNN, Autoencoder and GAN
4. Recurrent neural networks
5. Self-organizing neural networks

There are two computer exercises in this course. The exercises are application oriented, meaning that we will focus on the methods/application and not so much on the programming part. The amount of programming required is therefore limited. These notes are meant as a complement to the lectures. The lectures will also use material from the "Deep Learning Book" (DLB) [1].

1.1 Modelling - Biological Motivation and Connections

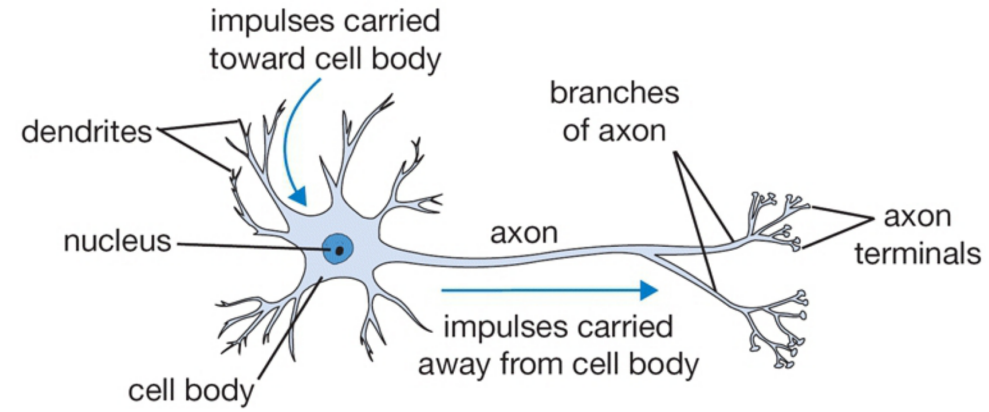
As explained in the course Convolutional Neural Networks for Visual Recognition, CS231N at Stanford [2], we review the analogy between the biological and mathematical mode of a human neuron in the nervous system. The basic computational unit of the brain is a **neuron**. Approximately *86 billion neurons* can be found in the *human nervous system* and they are connected with approximately $10^{14} - 10^{15}$ **synapses**. The diagram below shows a cartoon drawing of a biological neuron (left) and a common mathematical model (right).

Each neuron receives input signals from its **dendrites** and produces output signals along its (single) **axon**. The axon eventually branches out and connects via synapses to dendrites of other neurons. In the computational model of a neuron, the signals that travel along the axons (e.g. x_0) interact multiplicatively (e.g. $w_0 x_0$) with the dendrites of the other neuron based on the synaptic strength at that synapse (e.g. w_0). The idea is that the *synaptic strengths* (the weights w) are learnable and control the strength of influence (and its direction: *excitory* (positive weight) or *inhibitory* (negative weight)) of one neuron on another.

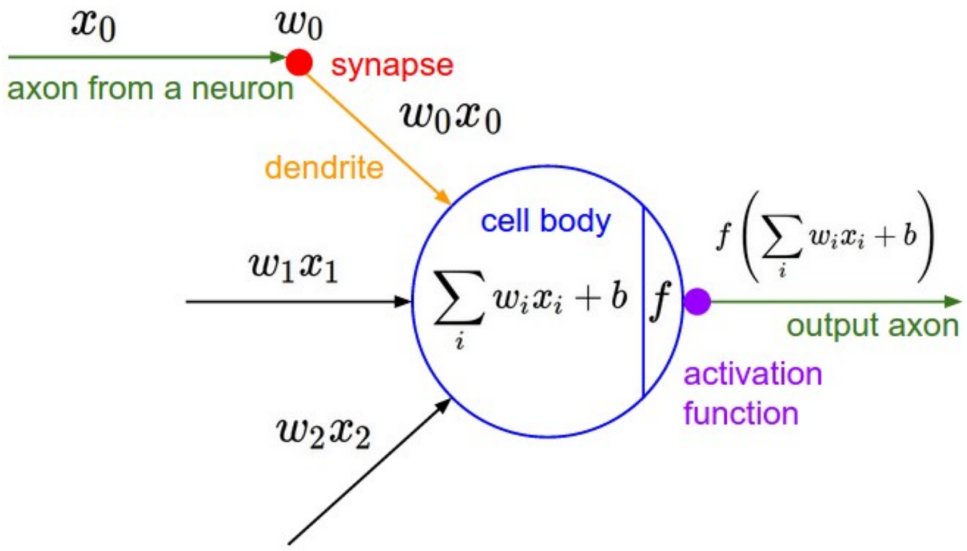
In the basic model, the dendrites carry the signal to the cell body where they all get summed. If the final sum is above a certain *threshold*, the *neuron can fire*, sending a *spike* along its axon. *In the computational model, we assume that the precise timings of the spikes do not matter, and that only the frequency of*

[1] `www.deeplearningbook.org`
[2] `https://cs231n.github.io/neural-networks-1/`

the firing communicates information. Based on this rate code interpretation, we model the *firing rate* of the neuron with an **activation function f**, which represents the *frequency of the spikes* along the axon. Historically, a common choice of activation function is the **sigmoid function σ**, since it takes a real-valued input (the signal strength after the sum) and squashes it to range between 0 and 1. We will see details of these activation functions later in this section.



(a)



(b)

Figure 1: (a) A cartoon drawing of a biological neuron. (b) Neuron mathematical model . Courtesy: Convolutional Neural Networks for Visual Recognition, CS231N Stanford

In other words, each neuron performs a *dot product* with the input and its weights, adds the *bias* and applies the non-linearity (or activation function), in this case the sigmoid $\sigma(x) = 1/(1 + e^{-x})$. We have different activation functions to take in consideration.

Coarse model It's important to stress that this model of a biological neuron is very coarse: For example, *there are many different types of neurons, each with different properties.* The dendrites in biological neurons perform *complex nonlinear computations.* The synapses are not just a single weight, they're a *complex non-linear dynamical system.* The exact timing of the output spikes in many systems is known to be important, suggesting that the rate code approximation may not hold. Due to all these and many other simplifications, be prepared to hear groaning sounds from anyone with some neuroscience

background if you draw analogies between Neural Networks and real brains. See this review (pdf) [3], or more recently this review [4] if you are interested.

2 Chap 3 - Feed-forward Neural Networks

1. What is linear separability, i.e. when a problem is linearly separable? When the classes in a classification problem can be separated by a hyperplane.

2. The simple perceptron cannot solve the XOR problem, but it can solve the AND. What is the simple perceptron? A linear classifier.

3. What is the error for the linear single perceptron? Summed squared error. 4. Explain how gradient descent works. 5. What is batch updating, online updating and stochastic gradient descent?

6. Define the Kronecker delta?

$$\delta_{ij} = \begin{cases} 1 & \text{if } j = i \\ 0 & \text{if } j \neq i \end{cases} \tag{1}$$

7. What is important to remember if the linear simple perceptron is no longer linear, i.e. the activation function isn't linear? That the inner derivative of the activation function needs to be taken into consideration.

8. Can be the multi-layered perceptron solve the XOR problem that the simple perceptron cannot, and if so why? Yes, because each node in the hidden layer defines a hyperplane and thus the XOR can be separated.

9. What's the derivative of the tangent hyperbolic function? It is the hyperbolic secant, i.e.

$$f'(x) = \tanh(x))' = sech(x) = \frac{2}{e^x + e^{-x}} \tag{2}$$

10. Backpropagation (not through time) for a 1-hidden layer perceptron needs derivatives of the error function for the input-to-hidden and the hidden-to-output. How are these defined? The hidden-to-output is just "one step" and thus does not require any chain rule. It is pretty straightforward and contains the net output from the hidden layer as variable h. The input-to-hidden requires the chain rule. Remember that it goes EY, YH, HJ and thus does not take into consideration the hidden-to-output weights.

11. The universal approximation theorem. What does it state? That the MLP can approximate any continuous function.

12. What error function follows naturally for regression problems? The summed squared error function.

13. What error function follows for two-class classification problem? Cross-entropy error.

14. For classification problems, which activation function is used for the output node? Logistic activation function.

15. How many outputs are used when we are talking about a classification problem with $c > 2$? c.

16. What error functions follows for classification problems with $c > 2$? Cross-entropy, but a modified version.

17. Now let's talk about the MLP "A brief conclusion"-table:

- a) Let's say we have a regression problem. What is the activation function for the hidden layer? The logistic function or the tangent hyperbolic.
- b) Still regression. What activation function for the output? Linear.
- c) Still regression. What error function shall be used? Summed squared error function.
- d) Classification problem, $c = 2$. What act. function for the hidden layer? Logistic or tangent hyperbolic.
- e) What act. function for the output? Logistic. Classification, $c > 2$. What act. function for the hidden layer? Logistic or tanh.

[3] https://physics.ucsd.edu/neurophysics/courses/physics_171/annurev.neuro.28.061604.135703.pdf
[4] http://www.sciencedirect.com/science/article/pii/S0959438814000130

- f) For the output? Softmax.
- g) What error functions for the classification problems? Cross-entropy.

18. Why do we normalize inputs? A few problems with no normalization. Consider two inputs, where one of them is very large compared to the other. The large one will dominate and drown the smaller one out, even though they might be equally significant. Furthermore, it is a good idea to normalize the input in order to avoid having to initiate the weights differently for each input. This give a standard procedure that will be used for all input.

19. Name two ways of normalizing. Place the input in a fixed interval or transform it to unit variance and zero mean.

20. The normal gradient descent formula contains the negative gradient and a learning rate. Name three enhancements to this method.

- a) Momentum term (poor man's conjugate gradient): It adds a part of the previous update to the current one, i.e. the current weight multiplied with a factor. This will accelerate the minimization when several updates in a row have the same sign and decelerate when they have opposite sign.
- b) Dynamical Learning Rate (bold driver): Bold driver modifies the learning rate based on the sign of the error changes.
- c) Individual and adaptive learning factors – RPROP (Resilient PROPagation): Individual learning rates can be used. One such method is the RPROP which uses the sign on the gradient and not its size. Negative aspect: Has to tune two new parameters.

21. Conjugate gradient. Name an updating rule. Polak-Ribiere.

22. Define generalization performance. Generalization performance = the performance of a trained ANN model on new data, i.e. data that was not part of the training procedure (validation performance).

23. Name a complication that can limit the generalization performance and explain it. The problem can be overtraining, which means that the ANN becomes very good at estimation the training data, but the validation data will be bad.

24. Explain bias-variance tradeoff. The summed squared error (for example) can be decomposed into a decomposition of two terms. These two terms are the bias and the variance. The bias-variance tradeoff means that there is a conflict between the bias term and the variance term. For ANN models this usually means that in order to have low bias one need a flexible model, e.g. a large number of hidden nodes. A flexible model however means that we can fit the ANN model very well to a specific dataset, i.e. the variance increases. In order to have low variance we can restrict the ANN model by e.g. having very few hidden nodes, but this also means that we may increase the bias since the ANN model may be too limited.

25. What is the bias? It is the average of how much the model differs from the target function.

26. What is the variance? It measures the sensitivity of the output for every dataset.

27. What is it called to control the bias-variance tradeoff? Regularization.

28. Generally speaking, how does the error function being modified when doing regularization?

$$E_{mod} = E + \alpha\Omega \tag{3}$$

where omega is the regularization term and alpha controls the amount of regularization.

29. Name four regularization terms and their pros and cons.

- a) L2 norm regularization (weight decay) Omega is half of the summed squared weights. This term will force excess weights to zero while keeping necessary weights to a non-zero value. + Easy to implement + Often increases generalization performance + Theoretical interpretation - alpha needs tuning - Sometimes large weights are necessary
- b) Modified L2 norm (weight elimination)
- c) L1 norm regularization (lasso) Half of the sum of the absolute values of the weights. Lasso can be used as a feature selection method.

- d) Early stopping When the validation error starts to increase, the training is stopped.

30. Explain the holdout method.

31. Explain the K-fold cross validation.

32. Explain bootstrapping. A bootstrap sample of a dataset means resampling the data set with replacement, to produce a new data set.

3 Chap 4 – Network Ensembles

33. Can we be sure that using ensembles will generate a better performance? Using Cauchy's inequality, it can be shown that the average error for the ensemble is smaller than (or equal to) the average of a single network.

34. The ensemble will of course also generate an error. This error can be decomposed into two terms. Explain these terms. The first term (1) is the average error made by the individual network in the ensemble, and the second term (2) is the average variance of the ensemble members. (1)-(2) is the decomposition.

35. What does this lead to us wanting? We want a set of accurate ensemble members that disagree as much as possible.

36. How should we train these individual ensemble members? Name three strategies.

- a) Different initial conditions. We train all the networks with different random initial values of the weights. This may lead to finding different local minima of the error function, which may produce slightly different networks. + Easy to implement - Networks are not diverse enough
- b) Bagging. Bagging creates diverse networks by training on slightly different training data sets. It is basically bootstrapping. + Simply good - Not diverse enough for very large datasets
- c) K-fold cross splitting + Simply good, faster than bagging - Not really

4 Chap 5 – Recurrent networks

37. When are recurrent typically used? When dealing with sequence data, i.e. e.g. text data, speech data, stock prices on stock market etc.

38. What is a time delay network – does it have feedback connections? It is an ordinary MLP with no feedback connections. The approach is to use a fixed number of previous values of the sequence and use these as input to create a prediction.

39. What is iterated prediction? It's when predicted values are used as inputs. The time delay networks suck at this.

40. Simple recurrent networks, Elman type. How is the architecture built? It is built where a copy of some node is being sent back into the network, and thus creating some recurrence. The network is treated as a normal feed-forward network, though, hence the term simple recurrent network.

41. Simple recurrent networks, general type (true feedback connections). When dealing with recurrence, the temporal dependency in the network can be back traced all the way to the beginning, making the network structure look a lot like an MLP. What is this called? Unfolding in time.

42. Explain backpropagation through time. The backpropagation through time is basically the same procedure as ordinary backpropagation with the key difference that some terms will depend on "previous values". In long sequences, this previous value will also depend on previous value and so on. So, it is unrolling in time and doing standard backpropagation that is called BPTT.

43. Name a problem with BPTT that needs to be taken care of, especially if we are dealing with long sequences. How does this problem arise? It is the vanishing gradient problem. When doing the backpropagation, it becomes clear that a lot of gradients will be multiplied together. Gradients are derivatives which means that they can be very small, and multiplying a large number of small values

means that the gradient will vanish and not contribute to the actual weight update and make the training extremely slow.

42. Name a few methods to combat the vanishing gradient problem.

- a) Properly initialize the weights.
- b) Use rectifier units (ReLU) instead of tanh or sigmoid activation functions.
- c) Truncate the BPTT method, i.e. splitting the sequence.

43. LSTM networks (Long Short-Term Memory networks). Read from truncating to Hopfield.

44. The Hopfield model. It is a fully recurrent network. What does it mean that a pattern is stable in the Hopfield model? This means that if the Hopfield model is initialized with a pattern and we start to update the model, nothing happens.

45. What does it mean with associative memory, and how is it connected to the Hopfield model? It means that if some of the bits of a stable pattern are changed, then when updating the Hopfield model it should retrieve the correct stable pattern. At least half of the bits needs to be correct.

46. If we have 7x5 pixels, how made nodes will be in the Hopfield model? 35.

47. How many patterns can a Hopfield model store, given N nodes? 0.14N.

5 Chap 6 – Deep Learning

48. How is the rectifier activation function defined? g(x) = max(0,x).

49. What is the smooth approximation of the ReLU? The softplus function.

$$g(x) = ln(1 + exp(x)). \tag{4}$$

50. Why has the ReLU become so popular?

- a) Sparse activation.
- b) No vanishing gradients.
- c) Fast computations.
- d) More biological plausible.

51. Explain the dropout method. [. . .]

52. RMSPROP (Root Mean Square Propagation) uses only one common learning rate, but it keeps a running average of the squared gradient for each weight that is used to normalize the magnitude of the gradient and thereby effectively having individual learning rates. What does Adam also have? Adam also keeps a running average of the past gradients.

53. What is a convolutional neural network? It is a network that does convolution instead of general matrix multiplication.

54. In machine learning applications, what does the input array usually consist of? It is an array of data.

55. What does the kernel usually consist of? It is an array of learnable parameter.

56. Name three important ideas that can help improve a machine learning system.

- a) Sparse interactions: This is accomplished by making the kernel smaller than the input. For example, when processing an image, the input image might have thousands or millions of pixels, but we can detect small, meaningful features such as edges with kernels that occupy only tens or hundreds of pixels. This means we need to store fewer parameters, and thus reduces the memory requirements of the model and improves its statistical efficiency.
- b) Parameter sharing: Refers to using the same parameter for more than one function in a model. In a traditional neural net, each element of the weight matrix is used exactly once. In a CNN, each member of the kernel is used at every position. This doesn't reduce the runtime, rather the storage requirements.

- c) Equivariant to translation: If we move an event later in time in the input, the exact same representation of it will appear in the out.

57. When can equivariance to translation be not so helpful? For example, if we want to recognize a face, some portion of the network needs to vary with spatial location, because the top of a face does not look the same as the bottom of a face.

58. If the function that a layer needs to learn indeed is a local, translation invariant function – what does this mean for us? The layer will be dramatically more efficient if it uses the convolution rather than the matrix multiplication. If the necessary function does not have these properties, using a convolutional layer will cause the model to have large training error.

59. The typical layer of a convolutional network consists of three stages, which ones? a) In the first stage (convolutional stage), the layer performs several convolutions in parallel to produce a set of presynaptic activations. In the second stage, each presynaptic activation is run through a nonlinear activation function, such as the ReLU. This stage is sometimes called the detector stage. In the third stage, we use a pooling function to modify the output of the layer further.

60. Explain pooling? A pooling function replaces the output of the net at a certain location with a summary statistic of the nearby outputs.

61. Name four pooling functions/strategies?

- a) Max pooling operation reports the maximum output within a rectangular neighborhood.
- b) Average of the rectangular neighborhood.
- c) The L2 norm of a rectangular neighborhood.
- d) Weighted average based on the distance from the central pixel.

62. What does pooling accomplish for the network? It makes the representation invariant to small translations of the input.

63. Why is invariance to local translation important sometimes? If we care more about whether a feature is present rather than exactly where it is.

64. Pooling over spatial regions produces invariance to translation. But if we want the features to learn which transformations to be invariant to, what do we do? We pool over the outputs of separately parametrized convolutions.

65. Explain why pooling is essential for many tasks? It handles inputs of varying sizes. For example, if we want to classify images of different sizes, the input to the classification layer must have a fixed size. This is usually accomplished by varying the size and offset between pooling regions so that the classification layer always receives the same number of summary statistics regardless of the input size. For example, the final pooling layer of the network may be defined to output four sets of summary statistics, one for each quadrant of the image, regardless of the image size.

66. More-over, explain this again – but with the kernel in mind. Well, if each image has different width and height, it is unclear how to apply matrix multiplication. Convolution is straight-forward to apply. The kernel is simply applied a different number of times depending on the size of the input, and the output of the convolution operation scales accordingly.

67. Explain the auto-encoder? Key points: - The absolute middle layer, called the bottleneck, always have fewer hidden nodes than the input layer. - First half: encoder, second half: decoder. - Error function: basically, summed squared. - Representing the input efficiently such that the decoder can recreate the data with no error.

68. What error function is used for the training? Basically, a summed squared.

69. When using linear activation functions for the bottleneck layer, what is the autoencoder actually doing? A principle component analysis. If the data contains a lot of linear correlations among the input variables, then this can be successful.

70. What is the architecture of the autoencoder called? Butterfly design.

71. The stacked denoising autoencoder (SdAE) is a modern version (i.e. deep learning version) of the autoencoder. It introduces two new concepts, which ones? - Denoising: This simply means that when

we train the autoencoder we use the dropout technique on the input layer. The main reason is to avoid overtraining when we have a deep autoencoder with many weights. - Pre-training.

72. Explain pre-training? The pre-training step is done layer by layer. Let's take the first layer of the SdAE. To get an initial set of weights we train a simple autoencoder with only one hidden layer. When training is done, we take the first layer of weights of this training and use that as the first layer of weights for the SdAE. We can now run all the data through this first layer and generate a set of output values for the first hidden nodes. These are now the input data to a second simple autoencoder that will be trained to recover input data. This results in a second simple autoencoder, with weights that will be put on the second layer of the SdAE. And so on! During training, the dropout method is used on the input layer. This denoising is important to avoid identity mapping, especially if the size of the hidden layer is larger than the number of inputs.

73. Applications of SdAE? Pre-train deep networks, imputation of missing data, detection of outliers.

74. The variational autoencoder – what is this type of model? Generative model.

6 Chap 7 – Self-Organizing Neural Networks

75. What is the difference between supervised and unsupervised learning? In unsupervised learning there is no teacher.

76. Explain k-means clustering model? Decide how many clusters to look for (k). Define initial positions for the k clusters. The algorithm now alternates between two steps:

- a) Each data point is assigned to the nearest cluster (which often means the Euclidian distance between the data point and the nearest cluster center).
- b) Update all the cluster centers to be the mean of all data points part of the clusters.

7 Extra

77. (chap 3) Explain radial basis function networks? A radial basis function network has the activation of a hidden node determined by the distance between the input vector and the weight vector.

78. (chap 3) Explain the interpolation problem? You want to interpolate a function using weights and coefficients. The known data points are the center of the radial-basis functions. This problem can be solved by putting up a system of matrices, and thus finding an inverse matrix. Micchelli's theorem states that a large class of radial-basis functions can be used here and where the most important ones, considering ANN, are the Gaussian functions.

79. (chap 3) How is this connected to neural networks? We get a RBF neural network by introducing a set of modifications to the above idea:

- a) The number of basis functions is usually less than the number of data points.
- b) The centers of the basis functions need not be datapoints.
- c) Introducing individual widths for each basis function.
- d) Bias parameters are included in the linear sum.

80. (chap 3) Name a few differences between the MLP and the RBF? a) The MLP can have a complicated structure but the RBF has a simple 1-hidden layer structure. b) RBF suffers the curse of dimensionality more than the MLP does.

81. (chap 3) A very common problem with real world data is missing values, i.e. that the input does not have a full set of data. Name a few ways of handling this?

- a) Average value (for binary replace with most common category)
- b) Estimate with another mother

- c) Random imputation
- d) Auto-encoder

82. (chap 4) Explain briefly optimal ensemble weighting? Every ensemble member is factored with a factor alpha. This needs to be calculated. The error of the whole network can be decomposed into a term depending on the desired alpha and the expected value of residuals. The residuals form a correlation matrix. We wish to minimize the error depending on alpha and the correlation matrix, for a fixed correlation matrix. The solution can be solved with Lagrange multipliers which gives an explicit solution, but since the correlation is not really known, it is approximated by using a data set.

83. (chap 4) A problem with this approach is that we may get solutions with large positive and negative weighting factors. To overcome this, we introduce another constraint on alphas, both that they are summed to one but also that they are larger than zero. What is the drawback now? That the optimization problem is more difficult.

84. (chap 5) LSTM handles long time dependencies and combats the vanishing gradient problem. How is this network built up? In comparison to an MLP which has the node h, we have an extra value c, which is the internal memory of the node. We also have gates, called f, i, and o, for forget, input and output gates, respectively. They are all numbers between 0 and 1, and will be used to filter the new values. The gates have the same structure:

$$i = \sigma(x_t U^i + h_{t-1} W^i) \tag{5}$$

where U and W are new weights. When a candidate value of c has been computed (tanh the above), we are in position to update c and h using gate values. This is done by:

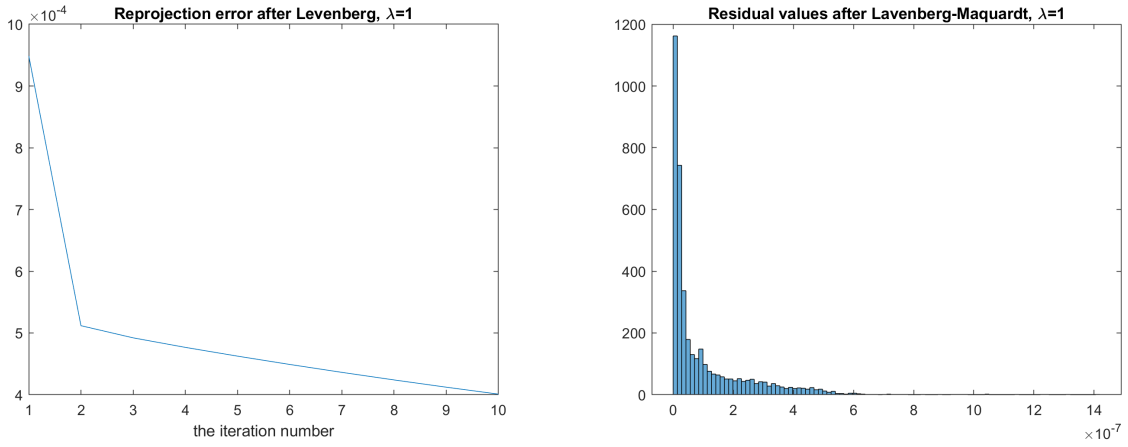
$$c_t = c_{t-1} f + \tilde{c} i; \quad h_t = \tanh(c_t) o \tag{6}$$

85. (chap 6) What is the variational autoencoder (VAE)? It has the ability to generate data.

86. (chap 6) How is the bottleneck divided? It is divided into mean and standard deviation. This will specify a gaussian distribution and this distribution is then sampled from.

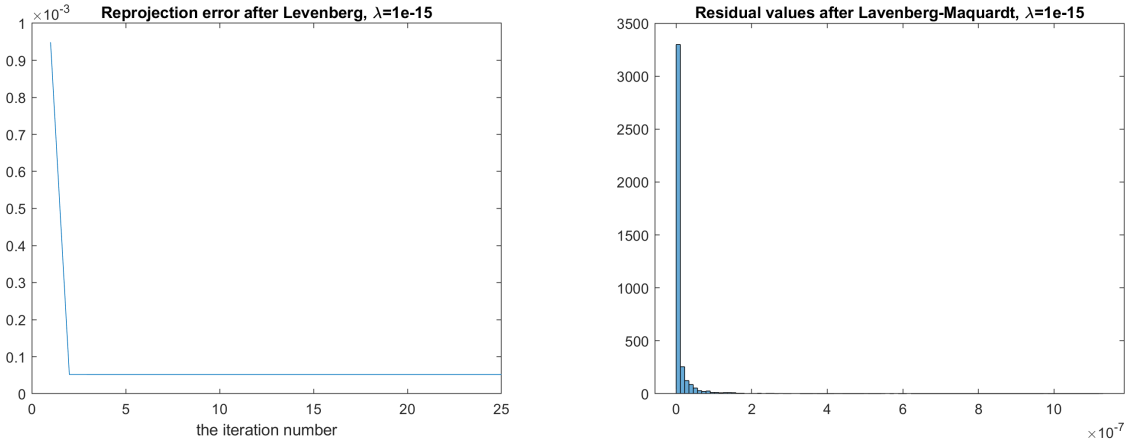
87. (chap 3) What output function shall be used for approx. problems, two-class classification or multi-class classification? Linear, logistic or softmax. 88. What is the update rule for the Hopfield model nodes?

$$si = \begin{cases} s_j, h_i = 0 \\ sgn(h_i), h_i \neq 0 \end{cases} \qquad h_i = \sum_j w_{ij} s_j \tag{7}$$



(a) (b)

Figure 2: Plotting after running the Levenberg-Maquardt method, when $\lambda = 1$. a) Plot of the reprojection error. b) Plot of histogram of all the residual values.



(a) (b)

Figure 3: a) plot of the reprojection error versus the iteration number for $\lambda = 10^{-15}$ after running the Levenberg-Maquardt. b) plot of histograms of all the residual values after running the Levenberg-Maquardt method in when $\lambda = 10^{-15}$.

References

[1] Carl Olsson, Computer Vision - FMAN85, Lectures notes: `https://canvas.education.lu.se/courses/3379`

[2] Hartley, Zisserman, Multiple View Geometry, 2004.

[3] Szeliski, Computer Vision - Algorithms and Applications, Springer.