What to read in the course book

The course book, by Andries P. Engelbrecht (*Computational Intelligence: An Introduction*) provides a computational and practical view of the field. The book includes chapters on neural networks, evolutionary computing, swarm intelligence, artificial immune systems and fuzzy systems. We cannot cover all of it during this course, though.

There are two editions of the book. This course, and the reading instructions below, assume that you have the second edition.

In addition to the book, the handout material provided through the student portal is also expected to be read by all students.

PART 1: INTRODUCTION

Read it all. The history lesson in section 1.2 is interesting, but not important. You may browse this section.

Section 1.1.1

Engelbrecht defines an artificial neural network as a layered network of artificial neurons. While most ANNs can be described this way, this is not true for all ANNs (i.e. not all ANNs are layered).

In the bulleted list in the same section, Engelbrecht lists a number of different neural networks types, including "standard back propagation". This is not a network, it is a learning algorithm (independent from a network structure). What he means, I believe, is the multilayer perceptron (the MLP).

Section 1.1.2

The second bullet in the bulleted list: Genetic programming does not always use trees to represent individuals. But usually, yes.

PART 2: ARTIFICIAL NEURAL NETWORKS

This is the most important part of the book, since neural networks is the main focus of this course. You should read everything in this part of the book, unless otherwise stated below.

Chapter 2: The Artificial Neuron

Page 17

> The mapping f suggests that neurons are always bounded (between 0 and 1 or between -1 and 1). This is not true. Example in fig 2.2(a).

Section 2.1

> Production units (PU's) are very uncommon and will not be considered in this course.
>
> Engelbrecht's "activation function" is also called "transfer function". I usually call it transfer function, but I admit that activation function is probably better. I'll try to change, but I hope you'll excuse me if my tongue slips.

Section 2.2

> In Figure 2.2, the Ramp function (c) should go through origo, and the Gaussian function (f) should not have that sharp tip at x=0. Many graphs throughout the book contains similar anomalies, unfortunately.
>
> Same section, page 20: There are two common ways to denote non-inclusive intervals. The non-inclusive interval here, (0,1), could also have been written]0,1[. I prefer the latter notation, but it is only a matter of taste.
>
> Equation 2.13: The *lambda* in the exponent must be doubled if you want the the same slope as in equation 2.12.

Section 2.3, Figure 2.3

> The hyperplane illustrated here assumes that the neuron's activation is a step function, i.e. a binary/bipolar one. With a sigmoidal activation function, the boundary becomes fuzzy.

Section 2.4, Figure 2.4

In (a) Any threshold value between 1 and 2 would work. Setting it to the limit (2) like this is not recommended, since the reader then must know whether the "greater-than" test in the activation function is strict, or not. Same in (b). Also, in (b) Engelbrecht makes an OR gate by increasing the input weights. The same effect could be achieved by keeping the weights from the AND gate, but lowering the threshold instead (to any value between 0 and 1.

Section 2.4.1, on augmented vectors, equation 2.10

It is more common to augment the vector in its other end, i.e. sum from 0 to *I* instead of from 1 to *I*+1 as it is done here. Both ways work, of course.

Section 2.4.2 on Gradient Descent Learning Rule

I usually define the error (equation 2.17) as half the squared error, which in effect removes the constant 2 equation 2.20.

Same section, immediately after the reference. Just to be clear (common mistake): The *learning rule* assumes that *f*==*net*, i.e. that the nodes are linear. This is not to say that the nodes *are* linear when producing their output, however.

Section 2.4.4

The Generalized Delta Rule is equivalent to the Backpropagation algorithm. Just another name.

Section 2.4.5, Error-Correction Learning Rule

This is equivalent to the Perceptron Convergence Procedure, talked about in the lectures.

Section 2.6, Assignments

I insert a subtle hint here, for those of you who bother to read my reading instructions: All the assignments here are possible exam questions.

Chapter 3: Supervised Learning Neural Networks

Again, in my opinion, Engelbrecht confuses learning with structure (the title). The network structures described in this chapter could also be trained using other training paradigms than supervised learning.

Section 3.1.1 Feedforward Neural Networks

> Confusing title and acronym (FFNN = FeedForward Neural Networks), since the following two network stuctures are also feedforward networks. When Engelbrecht write about FFNN he really means MLP (MultiLayer Perceptrons), i.e. feedforward networks with non-linear summation units. FFNN is a more general concept.
>
> Same section, in the equations, the variable *v* is never explained. But it is fairly obvious, from figure 3.1, that is is the input-to-hidden weights.

3.1.2. Functional Link Neural Networks

3.1.3. Product Unit Networks

> Only browse these two sections. They are interesting variants, but they are not very common and I won't bring them up on the course.

Section 3.1.5, first sentence

> Scratch the sub-clause. Backpropagation through time and TDNN are not the same.

Section 3.1.6

> I would avoid using CNN as an acronym for Cascaded Neural Networks since CNN already has other meanings in neural networks (Cellular Neural Networks, for example).

Section 3.2.1, equation 3.25

> Engelbrecht is not fully consistent in his error definitions. Sometimes he subtracts the target from from the network output, sometimes (in the next section, for example) it is the other way around. Of course, since the error is squared, this makes no difference.
>
> Same section. "Stochastic/online learning" is also called "pattern learning". "Batch/off-line learning" is also called "epoch learning".

Section 3.2.2

Curiosum. Engelbrecht gives credit to Werbos for backpropagation. Many authors fail to recognize Werbos' PhD thesis for what it is/was, and quote Rumelhart and McClelland (1986) as the inventors of backpropagation (they did invent the name, though).

Same paragraph, second sentence. As "learning iteration" is defined here, one learning iteration is not an epoch. An epoch is one pass through the whole training set, and Engelbrecht correctly uses the word in that, latter, meaning later in the rest of the book.

About the derivation of backprop in this section. Compare this to the derivation made on lecture. They are almost the same, but not quite. (Both are correct, though). The update rules (equations 3.37 and 3.44) are usually presented without the minus signs, i.e. the minus signs are baked in the definition of delta. However, this is just a cosmetic difference.

In equation 3.38. f' should be written to the left of the summation sign, or it should be made clear, using parenthesis, that f' is not part of the sum.

In the algorithm on page 41: Note that the deltas are computed for all layers before any weights are updated! This is to avoid backpropagating deltas through the wrong (updated) set of weights. (Common mistake when implementing the algorithm)

Same page, last point in the bulleted list. Engelbrecht has not yet defined what validation error is. He does so in section 7.1.1.

The subsection beginning on page 42, on product unit networks can be browsed, for reasons mentioned earlier.

Section 3.2.3. Scaled Conjugate Gradient

Section 3.2.4. LeapFrog Optimization

Skip these two sections, unless you want to propose a project on SCG or LeapFrog.

Section 3.3

Figure 3.8 assumes threshold units (step activation functions). The borders would not look like this if sigmoids were used. The combination of sveral sigmoids bends the borders and rounds the corners. For example, a circular region can be formed by only three hidden nodes.

Last paragraph. "the number of turning points plus one" is equal to the number of monotonic regions.

Section 3.4 on Ensemble Neural Networks

Note that the third alternative mentioned (p.51) on how to combine the outputs of several neural networks - to form a linear combination of them - is to make one large network of them all. The linear combination is just an extra (and common) output layer. The individual networks are still trained separately, though.

Last paragraph, same page. A clarification. For this to work, the patterns must be sampled with replacement.

Last paragraph in this section. The idea referred to as boosting here, is older than implied by the references. We will see an example of this on lecture.

Section 3.6 Assignments

Question 3 does not belong to this chapter. The question is about concepts that are introduced in chapter 7.

Each of the 5 "aspects" under consideration in Question 4 is a possible project on this course.

Chapter 4: Unsupervised Learning Neural Networks

Section 4.1

> The example in the last sentence of the first paragraph is not entirely clear to me. I think he means the ability to recognize the pitch of a note, irrespective of the*sound* of that note (e.g. irrespective of what instrument was played).
>
> Page 56, second paragraph. This is a generalization. Whether "supervised learning NNs have to retrain on all the information when new data becomes available" or not, depends on the algorithm used. But, OK, in general it is true.

Section 4.2, equation 4.2

> This is actually not Hebbs rule, but an extension to it. Hebbs postulate was that two nodes that are active simulatenously should have the connection between them reinforced. To my knowledge, Hebb said nothing about the counter case - inhibiting the connection between uncorrelated nodes, which this equation also implements (given bipolar node values).

Section 4.3

> Skip.

Section 4.4

> In the text immediately after equation 4.19, Engelbrecht mentions the case where LVQ-1 does not make use of a neighbourhood function, "thereby updating only the weights of the winning output unit". This is equivalent to SCL (Standard Competitive Learning).
>
> Throughout the section, it is not entirely clear what a "neighbor" is. What is usually meant is a node that is close in the network structure, not in weight space.
>
> The square root in equation 4.20 is not necessary. The winning node is the same whether this square root is computed or not.

Section 4.5, Self-Organizing Feature Maps

> Engelbrecht does not mention that the SOM is normally trained in two phases. First a coarse phase and then a fine-tuning phase.

Section 4.5.1, first paragraph, last two sentences

I think the last sentence was meant to replace the one immediately before it.

Section 4.5.2, Batch Map

Algorithm 4.3 is equivalent to K-Means.

Section 4.5.3, Algorithm 4.5

- The "furthest immediate neighbor" is the one among the structural neighbours which has a weight vector furthest from this one. In other words, "furthest" in this case refers to the weight space.
- In the insertion step the column and row are mixed up. Swap.

Section 4.5.4

The acronym BMN is not explain, as far as I can see. Engelbrecht means the winning node (Best Matching Neuron).

In the subsection on Shortcut Winner Search (nice idea, by the way).

- A clarification (within square brackets): "The search is based on the premise that the BMN of a pattern is in the vicinity of the BMN for [the same pattern in] the previous epoch".
- For the same reason, the first step in the following algorithm should read "Retrieve the previous BMN for this pattern".

Chapter 5: Radial Basis Function Networks

Section 5.2.2

Many alternative basis functions are proposed here. On this course, however, we only consider the Gaussian version (Eq. 5.9).

Section 5.2.3

The first two suggested ways to train a RBF network are not very good. The focus on this course is on the "two phase" training.

Section 5.2.4

Skip.

Chapter 6: Reinforcement Learning

This chapter contains a lot of misconception about Reinforcement learning (RL). The subject is better described in your compendium. Section 6.1 here is a good summary, though.

Introduction, second paragraph

> Engelbrecht refers to LVQ-II (described in chapter 5) as a RL algorithm. It is not. RL, by definition (see fig. 6.1), learns by interaction with an environment. It is true that LVQ-II learns from bipolar feedback, "rights" and "wrongs", but the sequence of patterns it encounters does not depend on its actions. That is not RL.

Section 6.1, second paragraph

> The rewards may be positive or negative, yes, but it is misleading to associate this sign to good or bad, respectively. RL tries to maximize the long-term reward received, disregarding the sign - the maximum may very well be negative!

Equation 6.2 and 6.3

> The discount factor in the infinite-horizon discounted model "enforces a bound on the infinite sum", as Engelbrech writes. It also has the side effect that the agent will favour solutions which give earlier rewards. That may, or may not, be the right thing to do, depending on the application. Consequently, the "problem" with the average reward model that it does *not* favour short-term rewards is not necessarily a problem at all.

Section 6.2

> This is a good, but too brief, summary of TD(0) and Q-Learning.

Section 6.3

> Engelbrecht repeats the statement that LVQ-II is RL, which it isn't.

Section 6.3.1

> This is a good description of RPROP, but it does not belong in this chapter. RPROP is a *supervised* learning algorithm, a modification to Backpropagation, and belongs in section 3.2. There is no connection to RL at all, except that any supervised learning algorithm may be used within the RL framework of course).

Section 6.3.2 and 6.3.3

> Skip. 6.3.2. is peripheral (though a good example of RL without explicit value functions) and 6.3.3 is better described in lecture.

Section 6.4, assignment 4

> Does not belong in this chapter (see comment to section 6.3.1 above)

Chapter 7: Performance Issues (Supervised Learning)

This is a very nice chapter. Knowing what to expect in terms of accuracy, convergence, performance, etc. is crucial in practical problem solving with neural networks, as is knowing how to manipulate data to facilitate training, without destroying information or giving the network unwanted bias.

Section 7.1.1

> Accuracy. The first two sentences are misleading. Generalization is a more general (sic!) concept than interpolation. Generalization is a measure of how well the network performs on previously unseen data, whether that data can be interpolated or not. Extrapolation is also to generalize.
>
> Equation 7.1, and the surrounding text. I don't know why Engelbrecht calls this "the true risk function", as if there were only one. This is not *the* objective, it is *an*objective, made under (at least) two assumptions: That the squared error is the most apropriate measure and that the distribution (*Omega*) is stationary.
>
> Engelbrecht introduces the mean squared error (MSE) in equation 7.3 when, in fact, he has already assumed that this was the objective in the preceeding equations. This shows how easy it is to make unintentional or "hidden" assumptions, or to forget that you made them.
>
> After equation 7.4, Engelbrecht argues that, in classification, counting the number of correctly classified patterns is a better performance measure than the MSE. Intuitively yes, but remember that the network still minimizes the MSE (unless you make drastic changes to the algorithm). Is it fair to judge the network's performance on a different criterion than was given during training?
>
> Immediately after Figure 7.1, generalization set = test set.

To clarify: The *training set* is used to compute weight changes. The *validation set* is used to plot the error during training (on the training set), in order to find a minimum. The performance (the results of the experiment) is then measured using the *test set* (generalization set).

Section 7.2

You don't have to understand the details on how to compute confidence intervals. The *t*-distribution, by the way, was invented by an employee at the Guinness brewing company in 1908. It (the distribution, not the brewery) is close to a normal distribution, but it changes its shape with the sample size - the "degrees of freedom").

Section 7.3.1, on missing values

the first paragraph after the bulleted list begins

"While missing values present a problem to supervised neural networks, ..."

Whether the network is trained by supervised learning or not is not relevant. The sentence should begin

"While missing values present a problem to summation unit neural networks, ..." Correspondingly, that the SOM does not have this problem is not because it is trained unsupervised, but because its units are distance measure units, not summation units.

on outliers, first bullet: Funny choice of words - "it is believed" (that removing outliers may also remove important data). Of course it may.

third bullet. "If the original training set contains no outliers, the method simply reduces to standard learning." I don't know this algorithm, but I wonder how one can decide if a set contains no outliers. Is the outlier concept that well defined?

on scaling and normalization. In practice, the "active domain" of a sigmoid is at least twice as wide as the one mentioned here ([-sqrt(3),sqrt(3)]). Engelbrecht does not say which "simple mathematical calculations" lead to this result.

Engelbrecht describes some scaling mechanisms using matrix notation. In my opinion, these methods are better described looking at one input (or target) at the time. For example, the first, mean centering, is simply to offset each input by its average, so that

the value is 0 at the average, all values above the average are positive, and all values below the average are negative.

In the section on Noise Injection, note that this applies to the *inputs* only. Don't inject noise on the targets (that would destroy the functional relationship).

Same paragraph, last sentence: The idea to use noise injection to generate new training patterns is a much older trick than this reference implies.

On training set manipulation, fourth paragraph: The idea to start small and gradually increase complexity is usually credited to a paper by Elman from 1993: "Learning and Development in Neural Networks: The Importance of Starting Small".

Section 7.3.2

Remember that this chapter applies to *supervised* learning. There are better initialization strategies for unsupervised learning.

Section 7.3.3, on learning rates

In the second paragraph it says that the learning rate should be increased if convergence is slow and decreased if the error does not decrease fast enough. This is contradictory. The criterion for decreasing would be better to base on the error variance (decrease if the error curve is noisy).

Next paragraph: Compare this to RPROP (in the handouts)

on momentum: Quickprop is more than a fancy momentum strategy. It is a second order method. See handouts.

Section 7.3.5: Equation 7.26. Another interpretation of the regularization term is that it implements a smoothness criterion (on the approximated function).

Browse the different pruning techniques. The ideas are of interest, but details and implementation are not important. You may skip the Information matrix pruning techniqe completely - I don't expect you, the reader, to know what a Fisher matrix is (nor should the author, in my opinion).

Section 7.3.6 Adaptive Activation Functions

Skip.

Section 7.3.7 Active Learning

> Browse or skip. The concept is interesting but not brought up on this course, except for the lectures on reinforcement learning which is an active learning technique.

PART 3: EVOLUTIONARY COMPUTING

This is a nice overview of evolutionary computing. However, the field as such needs a unified view - there are too many different names for very similar concepts. For example, Evolutionary Stratiegies (ES) and Differential Evolution (DE) are, in my view, just simple extensions to Genetic Algorithms (GA) and should perhaps have been presented within the GA chapter.

For this course, the first three chapters, chapters 8-10, are relevant (there is also a paper in your handouts by John Koza, on genetic algorithms and genetic programming). Chapters 11-15 can be skipped.

Chapter 8: Introduction to Evolutionary Computing

This is a good overview. All comments I had on this chapter in the first edition have been considered in this edition.

Chapter 9: Genetic Algorithms

9.2.2

> Skip

9.3.1

> We only consider uniform mutation on this course.

9.3.2 - 9.3.3

> Skip

9.5

Browse. There are some interesting variants here. Island Genetic Algorithms is a particularly interesting concept, in my opinion, but the details are not important for this course.

9.6-9.7

Skip

Chapter 10: Genetic Programming

Representing programs as trees is common in genetic programming, but it is not the only way. There are, for example, genetic programming techniques that operate on bit string representations (in which case, the difference between GA and GP is mainly phenotypic).

Chapter 11: Evolutionary Programming

Skip unless you are particularly interested.

The fact that mutation is the only operator used to produce offspring in EP (no crossover) makes this very close to random search. Not quite, though, since there is still a selection mechanism.

Section 11.6.1

The FSM example is in fact also a GA example, since it uses a bit-string representation. The only difference is that there are constraints on how the bitstring can be manipulated (mutated).

Chapter 12: Evolutionary Strategies

Skip unless you are particularly interested.

Evolutionary strategies could just as well be called meta-evolution - evolution not only of the population, but also of the genetic process as such. This is done by including some of the parameters that control the process in the genotype. For example, to include som extra bits to control the rate of mutation in the bit string used to represent an individual in a genetic algorithm.

Some of the claimed differences between ES and EC are superficial. In ES, changes due to mutation are only accepted in the case of success and offspring can be produced by more than two parents. But both of these ideas are possible (and common) in EC too.

Chapter 13: Differential Evolution

Skip unless you are particularly interested.

This is 'just' another reproducion mechanism. As such it is very interesting, though, in that it implements reproduction and mutation in one.

Section 13.1

> Typographic error (spelling), second paragraph, page 168: "propulation".

Chapter 14: Cultural Algorithms

Skip unless you are particularly interested.

Chapter 15: Coevolution

Skip unless you are particularly interested.

PART 4: SWARM INTELLIGENCE

Chapter 16: Particle Swarm Optimization

Note that in this chapter is it implicitly assumed everywhere, as far as I can tell, that the optimization problem at hand is a *minimization* problem.

Read until the second part of 16.1.6 (about termination, which can be skipped).

16.2-16.3

> Browse (you can skip 16.3.5 about velocity models)
>
> There is some confusion on variables in 6.3.3 (on constriction). The way Engelbrect defines phi1 and phi2 (as c1r1 and c2r2 respectively), equation 16.31 is correct. This is

however not the way phi1 and phi2 are usually defined, so his definition below of phi=phi1+phi2 is wrong (not even a constant). Should be phi=c1+c2.

16.4-16.6

Skip

16.7

Browse. Applications are always interesting, but details are not important. Of particular interest to you, perhaps, is the idea to train neural networks with PSO.

Chapter 17: Ant Colony Optimization

Read until (and including) section 17.1.4 (Ant System).

17.1.5-17.1.12

Browse. You might find section 17.1.7 about Ant-Q particularly interesting, since it is very close to (based on) Q-Learning.

17.2-17.4

Skip.

17.5

Read section 17.5.1 about TSP. Skip the rest.

PART 5: ARTIFICIAL IMMUNE SYSTEMS

Skip. Not part of this course (unless you make it so in the form of a project)

PART 6: FUZZY SYSTEMS

Skip. Not part of this course (unless you make it so in the form of a project)