Neuron

A diagram of a function

Description automatically generated

# Preceptron

A Perceptron takes several binary inputs,x1,x2,…., and produces a single binary output . So you can see that we have got 3 inputs x1,x2 and all of these have got random weight “w0,w1,w2” and the output will be the sum of “x\*w1+w2+b” and we add bias in this. This is how a perceptron works.

A screenshot of a computer

Description automatically generated

A diagram of a machine

Description automatically generated

The goal of the perceptron is to correctly classify the set of externally applied stimuli (x1, x2,..., xm) into one of two classes,c1 or c2. The decision rule for the classification is to assign the point represented by the inputs x1, x2,..., xm to class c1 if the perceptron output y is +1 and to class c2 if it is -1

In the simplest form of the perceptron, there are two decision regions separated by a hyperplane, which is defined by A group of symbols with a plus and a cross

Description automatically generated

A screenshot of a math problem

Description automatically generated

Types of transfer function

A diagram of a graph

Description automatically generated with medium confidence

Threshold function - perceptron

A math equations on a white background

Description automatically generated

A graph of a slope

Description automatically generated

Sigmoid:

* For big potential values the value of sigmoid function – 1. For smaill – 0

Parameter lambda – how quickly sigmoid function changes around zero. Lambda > 1 – changes quickly.

Function is continuous and smooth.

Radial basic (RBF) - gaussian (?)

Wavelet - far away from separating plan now activity can be seen.

**Neuron (Def) – formal definition**

A math equations and formulas

Description automatically generated

Neuron states:

A white paper with black text

Description automatically generated

Training

Supervised – training set of the form [ input / desired output]

Self-organization – no desired output ⇒ Goal: setting (adaptation) of the synaptic weights

Recall: of newly presented input patterns => Goal: get the response (output) of the neural network

A green and yellow text

Description automatically generated

**Training patterns (Def)**

**A white background with black text

Description automatically generated**

**Perceptron (Def)**

**A white paper with text on it

Description automatically generated**

**Linear Separability (def)**

Two sets of points 𝑨 and B are called linearly separable in an 𝒏-dimensional space, if 𝒏 + 𝟏 real numbers 𝒘𝟏, … , 𝒘𝒏, , 𝝑 exist, such that every point 𝑥1, 𝑥2, … , 𝑥𝑛 ∈ 𝐴 satisfies A black text on a white background

Description automatically generated and every point 𝑥1, 𝑥2, … , 𝑥𝑛 ∈ 𝐵 satisfies A black text on a white background

Description automatically generated

**Absolute linear separability: (Def)**

Two sets A and B are called absolutely linearly separable in an 𝒏-dimensional space, if 𝒏 + 𝟏 real numbers w1 , …, wn , ϑ exist, such that every point 𝑥1, 𝑥2, … , 𝑥𝑛 ∈ 𝐴 satisfies A black text on a white background

Description automatically generated and every point 𝑥1, 𝑥2, … , 𝑥𝑛 ∈ 𝐵 satisfies A black text on a white background

Description automatically generated

Theorem: (proof on slides)

Two finite sets of points A and B , that are linearly separable in an n-dimensional space, are also absolutely linearly separable

A close-up of a text

Description automatically generated

Separating hyperplane – for the extended weight and feature space

**A close-up of a text

Description automatically generated**

Problem: Find such weights and threshold capable of absolutely separating two sets => e.g., PERCEPTRON LEARNING ALGORITHM

SEPARATION of A and B:

**A math equations and formulas

Description automatically generated with medium confidence**

The goal of learning: minimize E(w) in the weight space (E(w) = 0 )

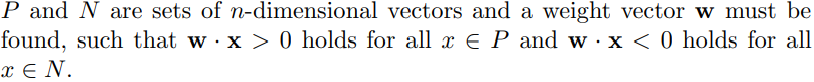
**Perceptron learning algorithm**

**A diagram of a graph

Description automatically generated**

**A close-up of a text

Description automatically generated**

****

**A white background with black text

Description automatically generated**

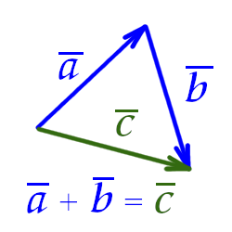
**A text on a white background

Description automatically generated**

**A text on a page

Description automatically generated**

**A close up of a text

Description automatically generated** 

**A graph of lines and arrows

Description automatically generated with medium confidence**

**A screenshot of a computer program

Description automatically generated**

Convergence

If the sets P and N are finite and linearly separable, the perceptron learning algorithm updates the weight vector a finite number of times.

**The pocket algorithm**

If the learning set is not linearly separable the perceptron learning algorithm does not terminate

The main idea of the algorithm is to store the best weight vector found so far by perceptron learning (in a “pocket”) while continuing to update the weight vector itself. If a better weight vector is found, it supersedes the one currently stored and the algorithm continues to run

**A white background with blue text

Description automatically generated**

**A close up of text

Description automatically generated**

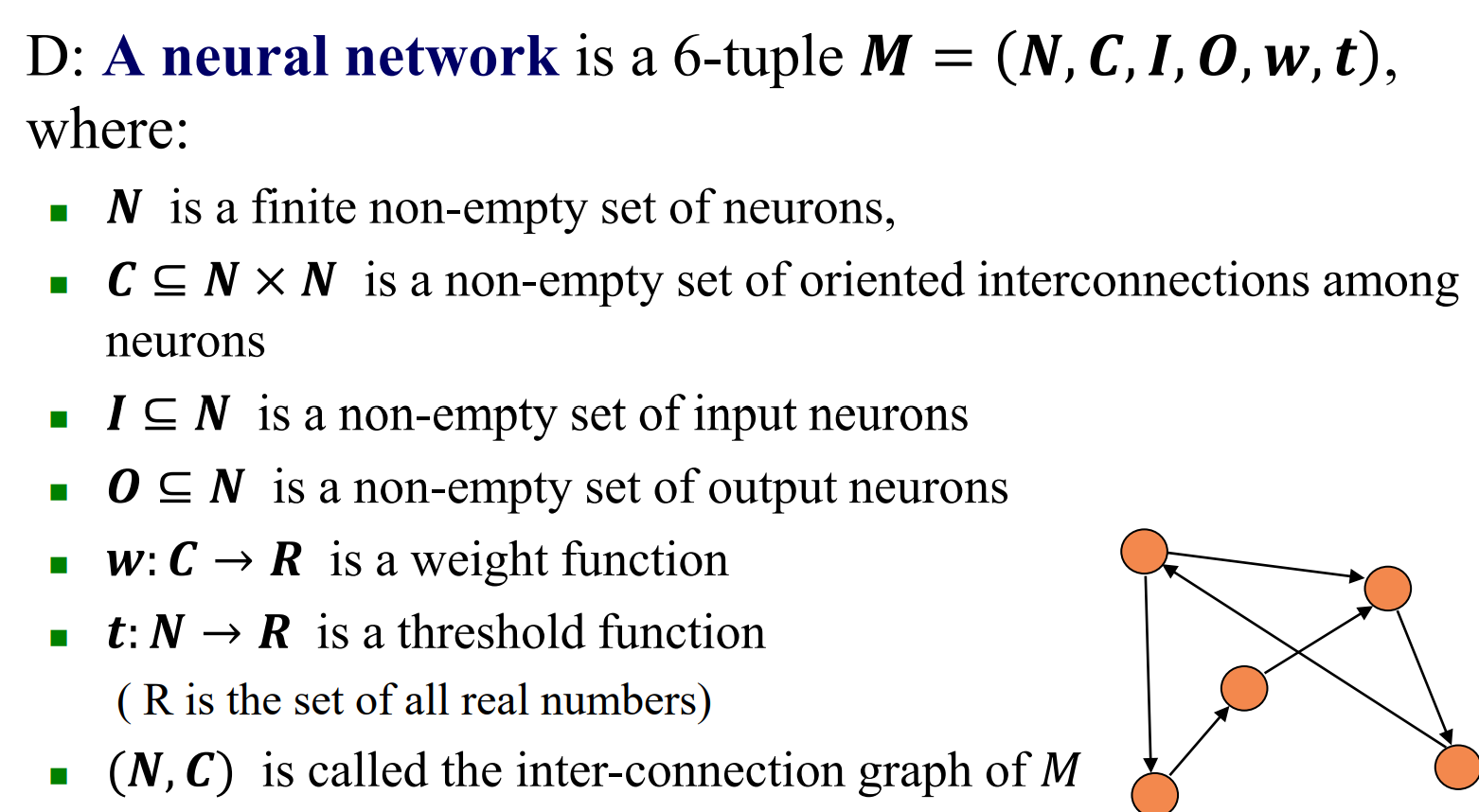
The algorithm can occasionally change a good stored weight vector for an inferior one, since only information from the last run of selected examples is considered. The probability of this happening, however, becomes smaller and smaller as the number of iterations grows. If the training set is finite and the weights and vectors are rational, it can be shown that this algorithm converges to an optimal solution with probability 1

**A close up of a text

Description automatically generated**

# **2.** Multi-layered neural networks

**Neural network (Def)**

****

**A Back-Propagation network (BP-network**) 𝑩 is a neural network with a directed acyclic inter-connection graph. Its set of neurons consists of a sequence of 𝒍 + 𝟐 pairwise disjunctive non-empty subsets called layers.

◼ The first layer called the input layer is the set of all input neurons of 𝑩, these neurons have no predecessors in the interconnection graph; their input value 𝒙 equals their output value. ◼ The last layer called the output layer is the set of all output neurons of 𝑩; these neurons are those having no successors in the inter-connection graph.

◼ All other neurons called hidden neurons are grouped in the remaining 𝒍 hidden layers

**Back-propagation training algorithm (p 152)**

The backpropagation algorithm looks for the minimum of the error function in weight space using the method of gradient descent. The combination of weights which minimizes the error function is considered to be a solution of the learning problem

**https://medium.com/a-year-of-artificial-intelligence/rohan-lenny-1-neural-networks-the-backpropagation-algorithm-explained-abf4609d4f9d**

Aim: find such a set of weights that ensure that for each input vector, the output vector produced by the network is the same as (or sufficiently close to) the desired output vector The actual or desired output values of the hidden neurons are not specified by the task. ω

objective function represents the total error between the desired and actual outputs of all the output neurons in the BP-network taken for all the training patterns

The Error function - corresponds to the difference between the actual and desired network output

A diagram of a mathematical equation

Description automatically generated during training, this difference should be minimized on the given training set => the back-propagation training algorithm

1. produce the actual output for the presented input pattern

2. compare the actual and desired outputs

3. adjust the weights and thresholds

◼ against the gradient of the error function ◼ from the output layer towards the input layer

The backpropagation algorithm is used to find a local minimum of the error function. The network is initialized with randomly chosen weights. The gradient of the error function is computed and used to correct the initial weights. Our task is to compute this gradient recursively

A diagram of a mathematical equation

Description automatically generated with medium confidence

A math equations with numbers and symbols

Description automatically generated with medium confidence

A math equations with numbers and symbols

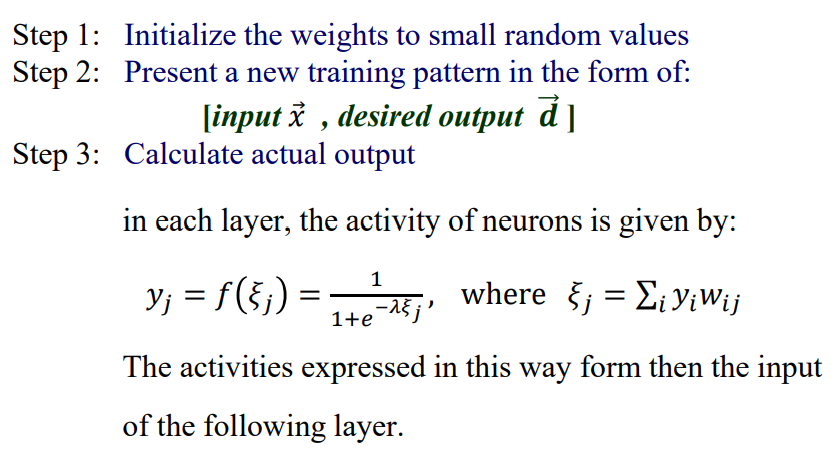
Description automatically generated with medium confidence

**Adjustments rules**

A white board with black text and numbers

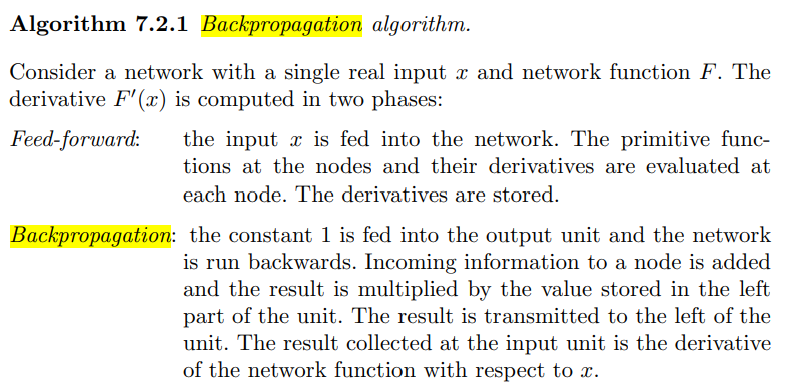
Description automatically generated

Back-propagation training algorithm



A screenshot of a computer program

Description automatically generated



+

ω Simple training algorithm

ω A very often used approach

ω Relatively good results

-

◼ Internal knowledge representation – „black box“

◼ the number of neurons and generalization capabilities

⚫ pruning and retraining

◼ error function (knowledge of the desired outputs)

⚫ „bigger“ and „balanced“ training sets ⚫ assessment of network outputs during recall

The standard back-propagation training algorithm is rather slow → a malicious selection of network parameters can make it even slower

The learning problem for artificial neural networks is NP-complete in the worst case → computational complexity grows exponentially with the number of the variables

Algorithms speeding-up the training process:

ω Keeping a fixed network topology

ω Modular networks ◼ considerable improvement of network approximation abilities

ω Adjustment of both the parameters (weights, thresholds, etc.) and the network topology

initial weight selection:

ω The weights should be uniformly distributed over the interval ⟨−𝜶𝒎 ,𝜶𝒎⟩

ω Zero mean value ◼ leads to an expected zero value of the total input to each node in the network (potential)

ω The derivative of the sigmoidal transfer function is reached its maximum for zero (~ 0.25)

◼ Larger values of the backpropagated errors

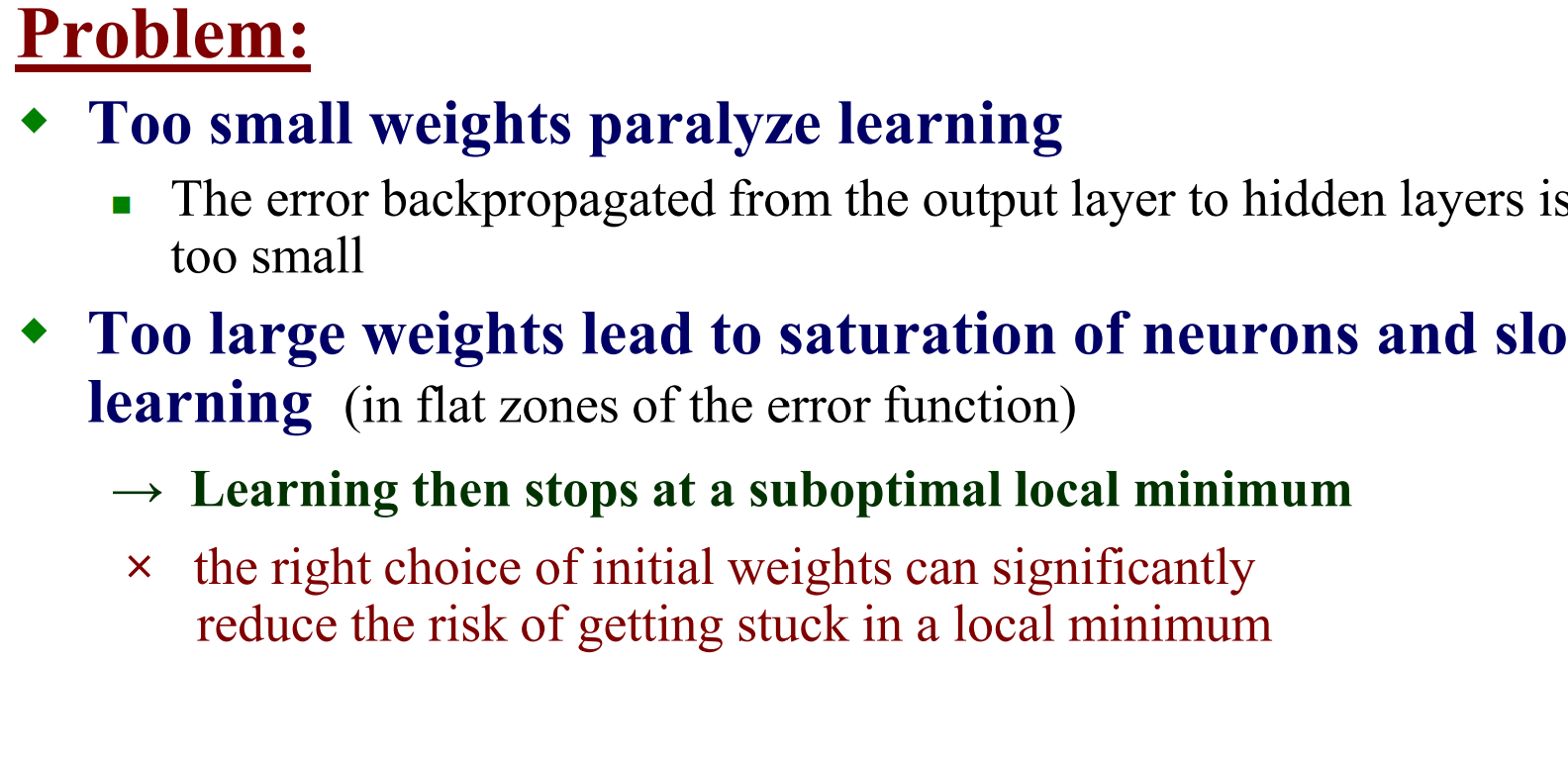
◼ More significant weight updates when training starts

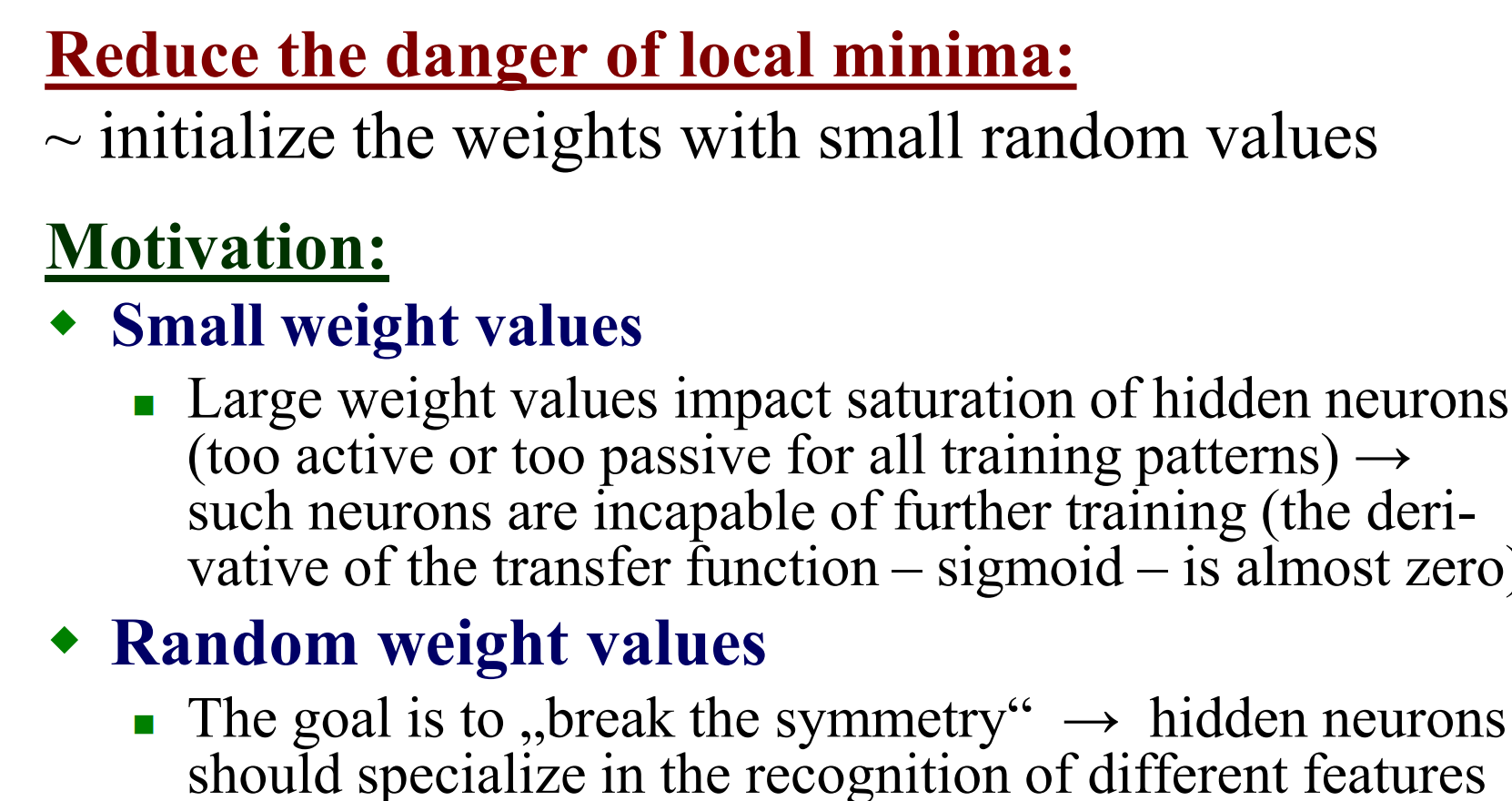
Problem:

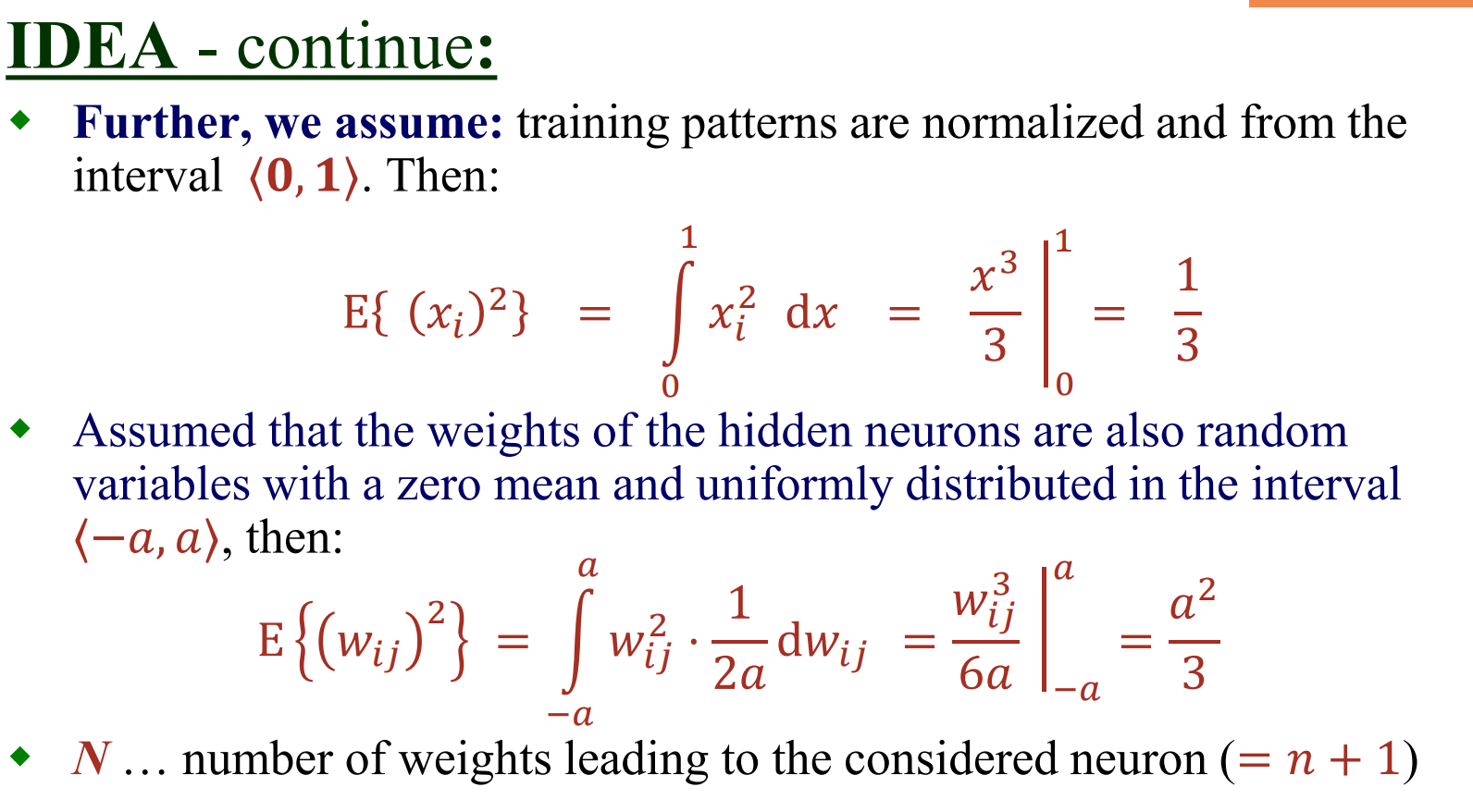
ω Too small weights paralyze learning ◼ The error backpropagated from the output layer to hidden layers is too small

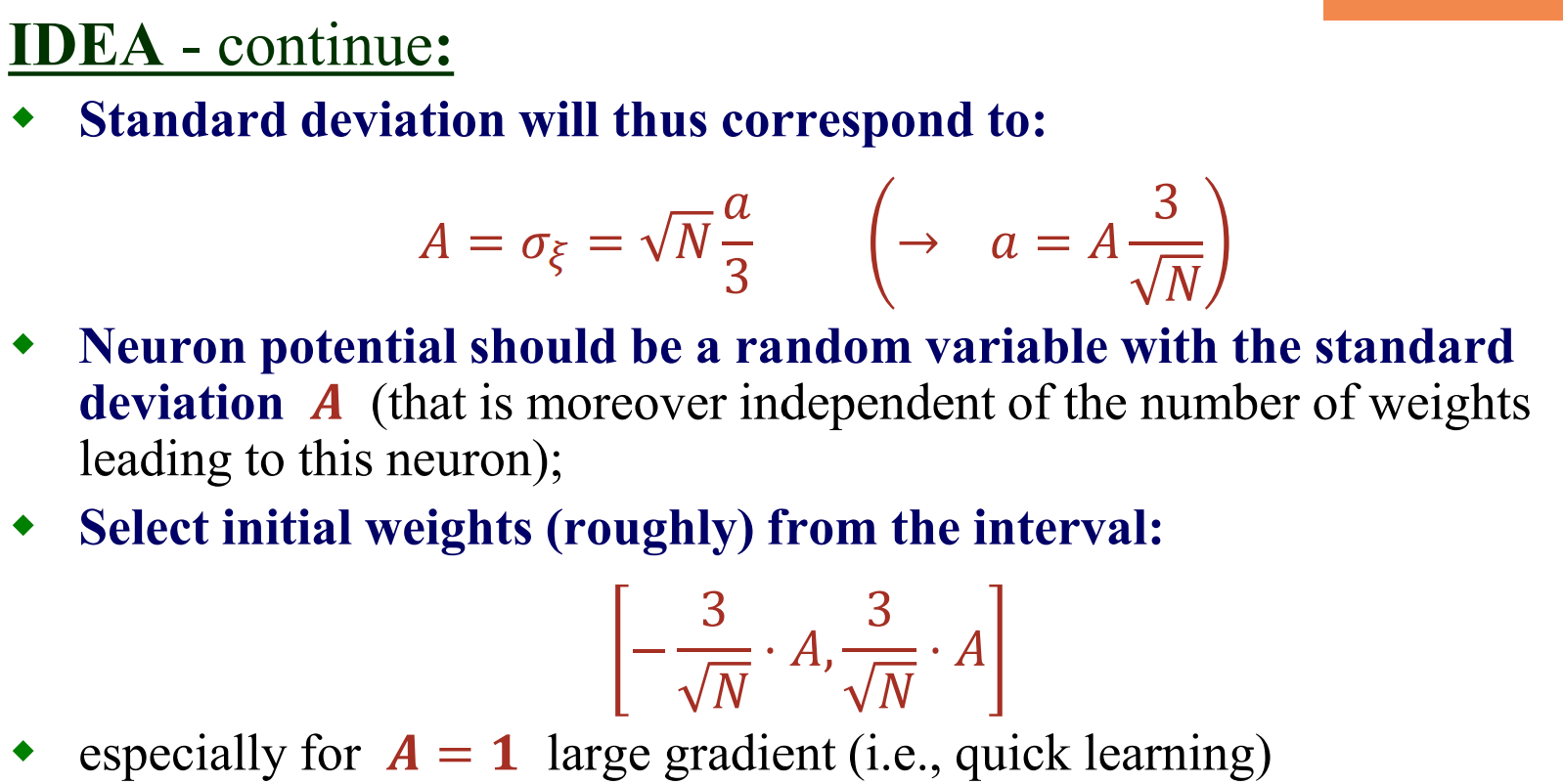
ω Too large weights lead to saturation of neurons and slow learning (in flat zones of the error function) → Learning then stops at a suboptimal local minimum

× the right choice of initial weights can significantly reduce the risk of getting stuck in a local minimum









Back-propagation training algorithm: first-order descent

A white text with black text

Description automatically generatedA math equations and formulas

Description automatically generated with medium confidence

Back-propagation training algorithm with momentum

A diagram of a diagram of a plane

Description automatically generated with medium confidence

When the minimum of the error function for a given learning task lies in a narrow “valley”, following the gradient direction can lead to wide oscillations of the search process. The best strategy in this case is to orient the search towards the center of the valley, but the form of the error function is such that the gradient does not point in this direction. => momentum

Momentum - a weighted average of the current gradient and the previous correction direction is computed at each step

The gradient of the error function is computed for each new combination of weights, but instead of just following the negative gradient direction a weighted average of the current gradient and the previous correction direction is computed at each step. Theoretically, this approach should provide the search process with a kind of *inertia* and could help to avoid excessive oscillations in narrow valleys of the error function.

→ Inertia ~ could help to avoid excessive oscillations in „narrow valleys of the error function“

A math equations on a white background

Description automatically generated

A close-up of a text

Description automatically generatedA white background with blue text

Description automatically generated

A graph of a marble rolling down the hill

Description automatically generated

In the nonlinear case, the gradient of the error function is almost zero in the regions far from local minima – possibility of oscillations → in such a case, larger learning rates could help → return back to „convex“ regions of the error function

Solution: ω Nesterov momentum ω Adaptive learning rates ω Pre-processing of the training set