Machine Learning Algorithms

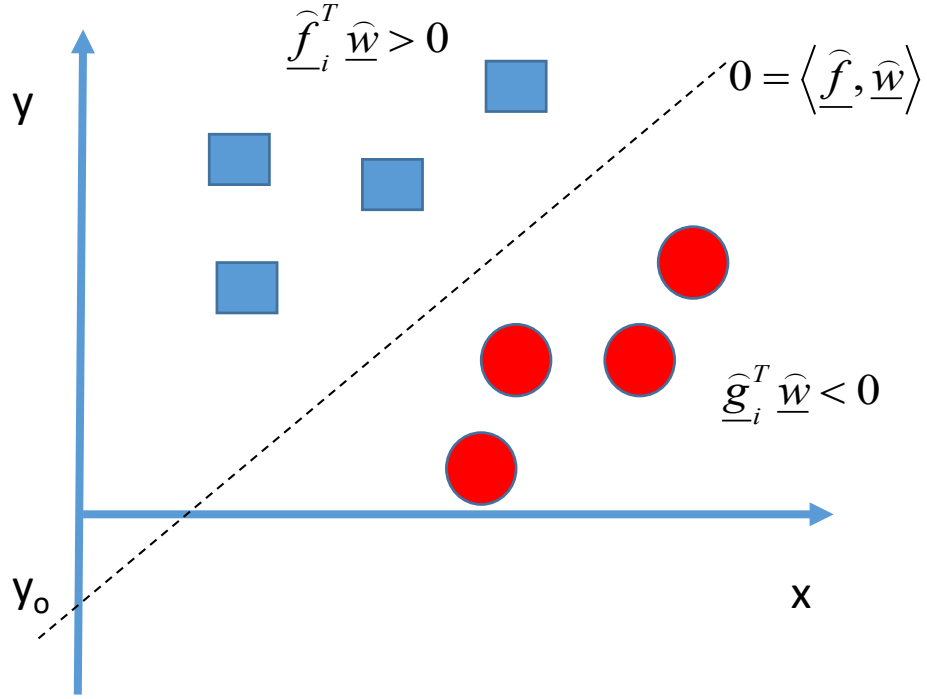
Neural Networks and the Backpropagation Algorithm

Markus Rupp

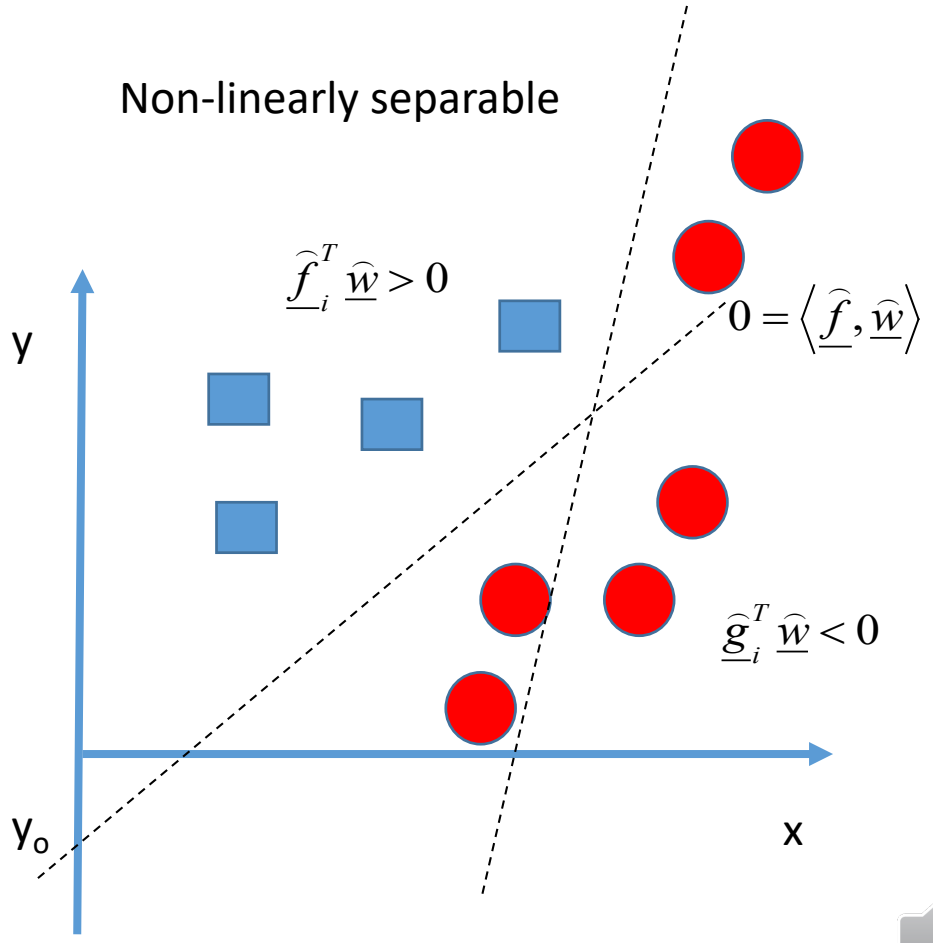
16.9.2020

Recall Classification

- Consider simple example

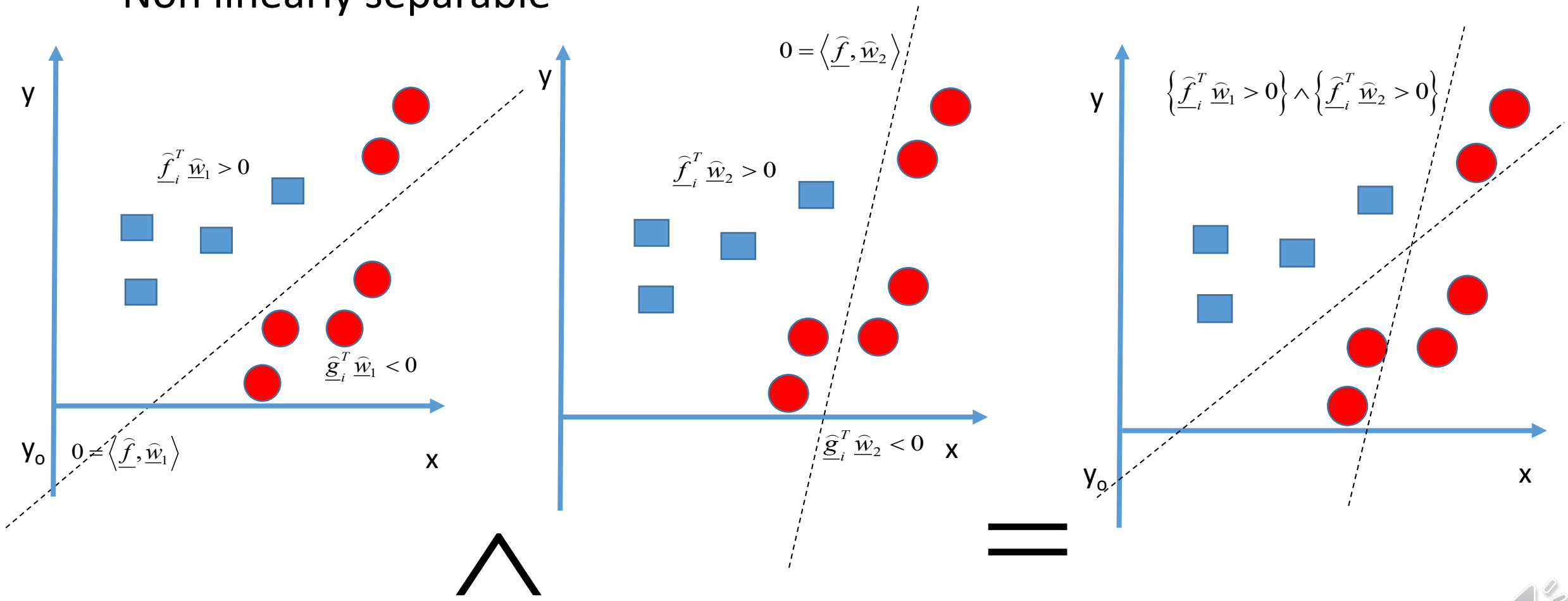


Non-linearly separable



Recall Classification

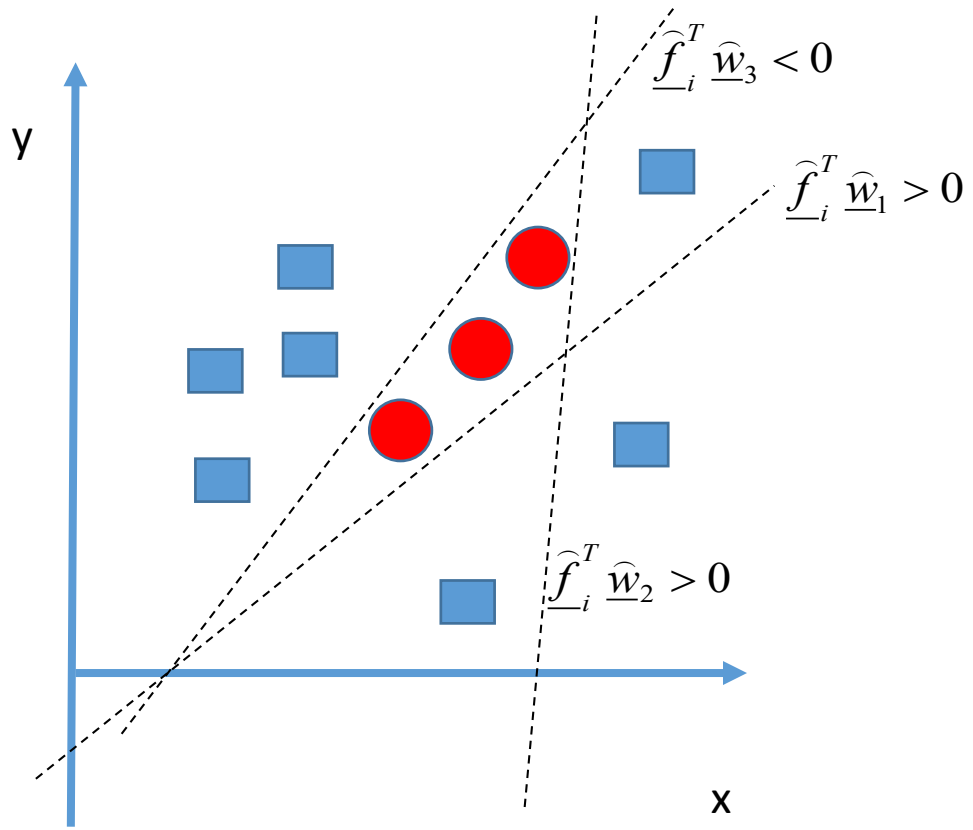
- Non linearly separable



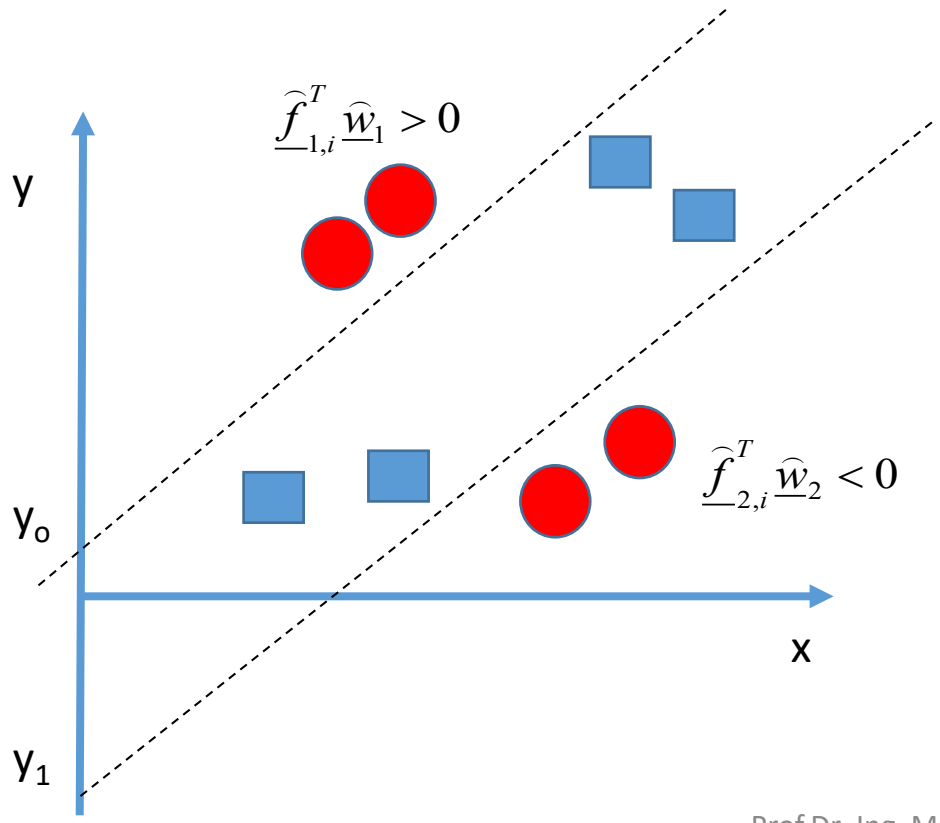
Multilinear Classification

- We can thus combine several half planes and form (arbitrary) sets
- Such sets can even be closed
- We thus „only“ have to combine several sets

$$\left\{\underline{\hat{f}}_i^T \underline{\hat{w}}_1 > 0\right\} \wedge \left\{\underline{\hat{f}}_i^T \underline{\hat{w}}_2 > 0\right\} \wedge \left\{\underline{\hat{f}}_i^T \underline{\hat{w}}_3 < 0\right\}$$



Classical nonlinearly separable Problem: XOR.



$$z_{1,i} = \underline{\hat{f}}_{1,i}^{T}\underline{\hat{w}}_1 > 0$$

$$\left[1, x_{1,i}, y_{1,i}\right]^T \left[w_{1,0}, w_{1,1}, w_{1,2}\right] > 0$$

$$y_{1,i} = \underbrace{-\frac{w_{1,0}}{w_{1,2}}}_{y_0} \underbrace{-\frac{w_{1,1}}{w_{1,2}}}_{>0} x_{1,i}$$

$$\text{if} \quad w_{1,2} > 0 \rightarrow w_{1,1}, w_{1,0} < 0$$

$$z_{2,i} = \underline{\hat{f}}_{2,i}^{T}\underline{\hat{w}}_2 < 0$$

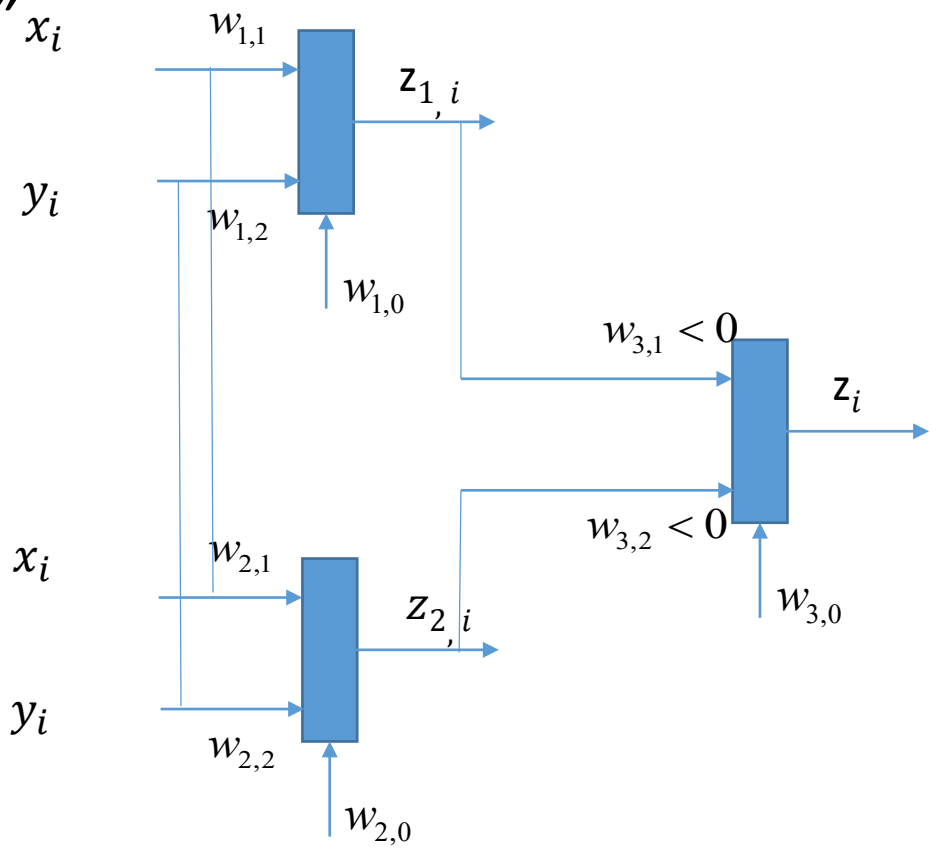
$$\left[1, x_{2,i}, y_{2,i}\right]^T \left[w_{2,0}, w_{2,1}, w_{2,2}\right] > 0$$

$$y_{2,i} = \underbrace{-\frac{w_{2,0}}{w_{2,2}}}_{y_1} \underbrace{-\frac{w_{2,1}}{w_{2,2}}}_{>0} x_{2,i}$$

$$\text{if} \quad w_{2,2} > 0 \rightarrow w_{2,1} < 0, w_{2,0} > 0$$

Classical nonlinearly separable Problem: XOR.

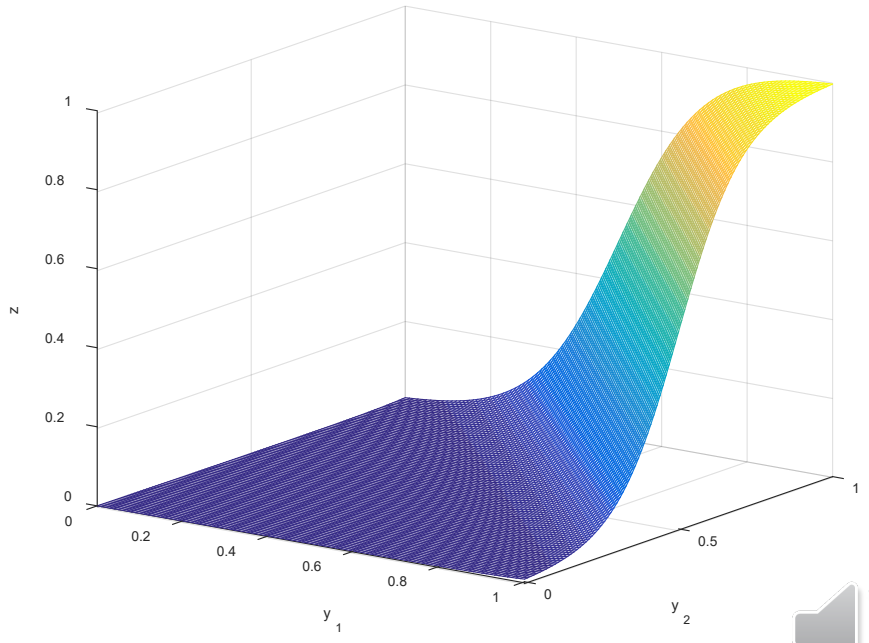
- Select blue squares „1“



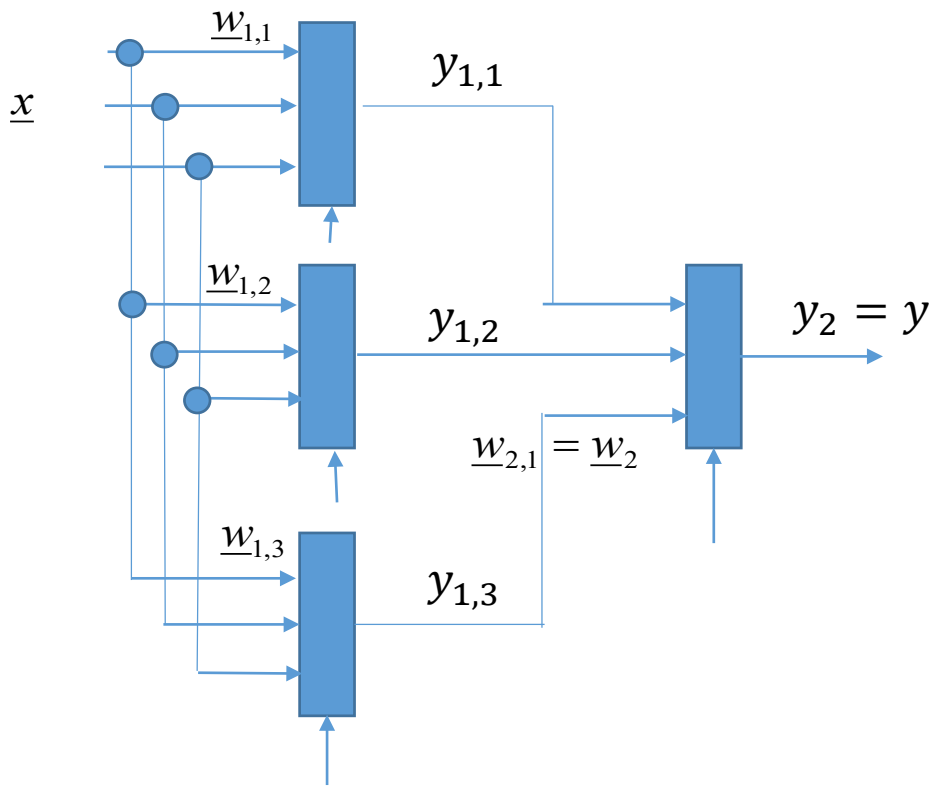
How to Combine Sets

- Let‘s assume both neurons deliver a positive value z_1,z_2 close to 1, indicating that each of them satisfy their conditions.
- This can be linearly combined and send to another neuron as input.
- Setting the bias of the second neuron sufficiently high, it will act as a logical „and“
- The output z of the second neuron is

$$z = \frac{1}{1 + e^{-10\left(w_{3,1}z_1 + w_{3,2}z_2 - 1.5\right)}}$$



Combining Sets



- Two layered neural net with input layer and output layer
- Each neuron comprises of a linear weight vector and a sigmoid function to make a decision
- The output neuron may require an additional input to adjust the „bias“ term.

The PLA for a two layered Network

- Updating the output layer weights $\underline{w}_{2,i}$: i=0,1,2,…

$$\underline{\hat{w}}_{2,i} = \underline{\hat{w}}_{2,i-1} + \mu \underline{\tilde{x}}_i \left(y_i - f\left(\underline{\tilde{x}}_i^T \underline{\hat{w}}_{2,i-1} \right) \right) f'\left(\underline{\tilde{x}}_i^T \underline{\hat{w}}_{2,i-1} \right)$$

$$\underline{\tilde{x}}_i^T = \left[1, y_{1,1,i}, y_{1,2,i}, y_{1,3,i} \right]$$

- Updating the input layer weights: $\underline{w}_{1,l,i}$: i=0,1,2,…, l=1,2,3

$$\underline{\hat{w}}_{1,l,i} = \underline{\hat{w}}_{1,l,i-1} + \mu \left(y_i - f\left(\underline{\tilde{x}}_i^T \underline{\hat{w}}_{2,i-1} \right) \right) f'\left(\underline{\tilde{x}}_i^T \underline{\hat{w}}_{2,i-1} \right) \underbrace{\frac{\partial \underline{x}_i^T \underline{\hat{w}}_{2,i-1}}{\partial \underline{\hat{w}}_{1,l,i-1}}}$$

$$\frac{\partial y_l}{\partial \underline{\hat{w}}_{1,l,i-1}} \left(\underline{\hat{w}}_{2,i-1} \right)_l = \left(\underline{\hat{w}}_{2,i-1} \right)_l f'(\underline{x}_i^T \underline{\hat{w}}_{1,l,i-1}) \underline{x}_i$$

$$= \underline{\hat{w}}_{1,l,i-1} + \mu \left(y_i - f\left(\underline{\tilde{x}}_i^T \underline{\hat{w}}_{2,i-1} \right) \right) f'\left(\underline{\tilde{x}}_i^T \underline{\hat{w}}_{2,i-1} \right) \left(\underline{\hat{w}}_{2,i-1} \right)_l f'(\underline{x}_i^T \underline{\hat{w}}_{1,l,i-1}) \underline{x}_i$$

The PLA for a two layered Network

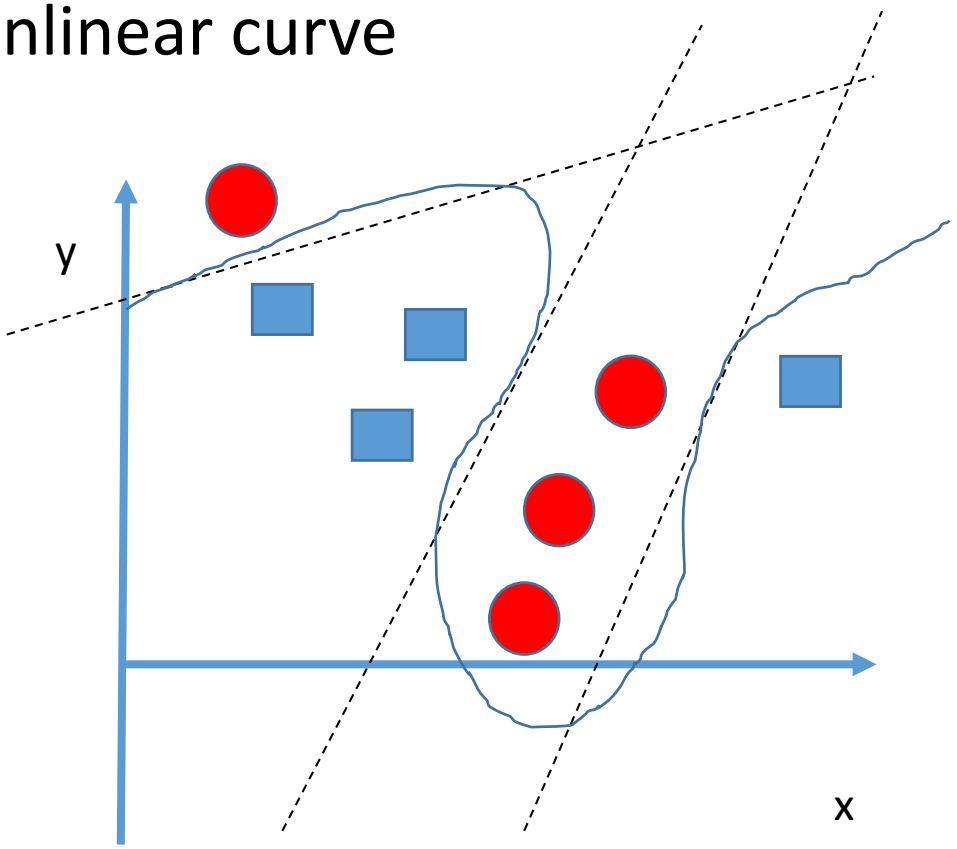
- We have added the index i to denote the iteration number.
- For each iteration a data pair $(y_i, \underline{x}_i)$ is available. These can be the same values repeated several times or simply one by one from the training set.

Problems for two layered Networks

- The obtained solution is not unique due to the symmetries of such network (swap two neurons, and you can obtain the same behaviour)
- Due to such symmetries, the initial values for an iterative algorithm may need to be selected carefully (i.e., different to ensure they do not select the same „line“)
- Due to the non-uniqueness of the solution, we cannot take the parameter error vector as a performance measure, only the output error is feasible.

How many Layers do we need?

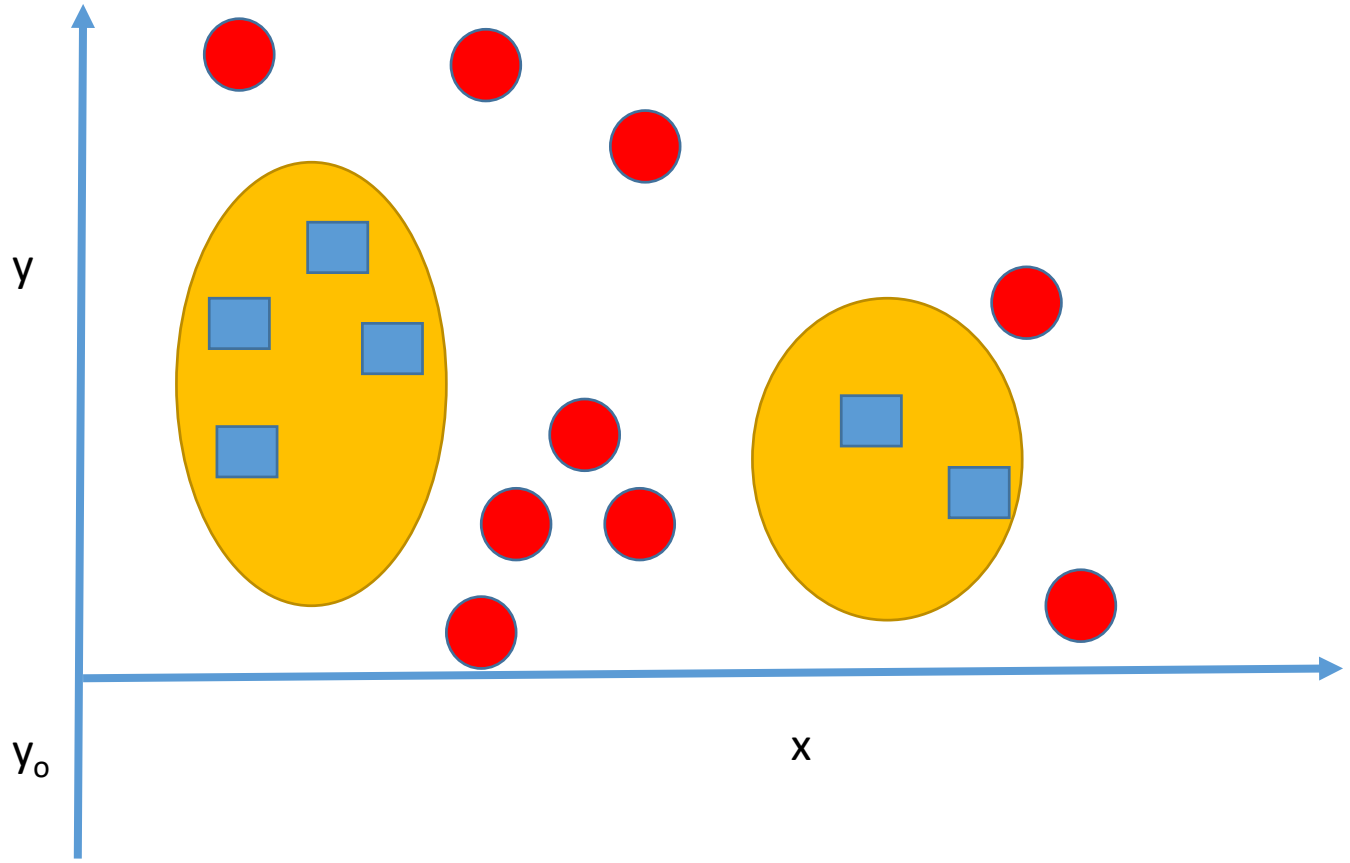
- Note that we a combination of two layers we can approximate any nonlinear curve



Thus, are 2 layers sufficient?

Combining Islands

- How do we combine such a class?

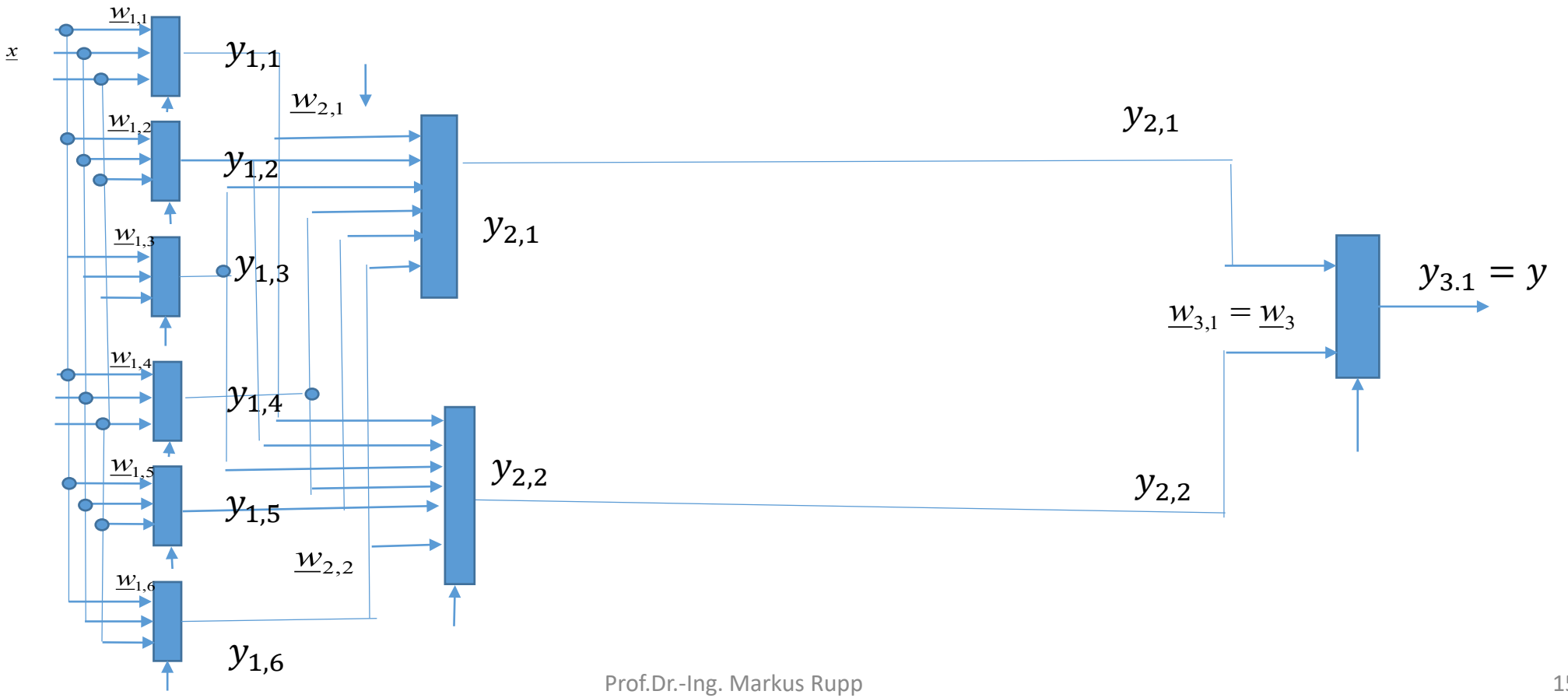


Combining Islands

- We need to have a third layer which allows to combine islands that the second layer defines
- General belief of the 90ies: three layer neurons are sufficient as all shapes can be formed by them
- The later part of this sentence can be proven
- All inner layers that are not input or output layers are called hidden layers. A three layer network has one hidden layer.

Three Layer Network

Are 3 layers sufficient?



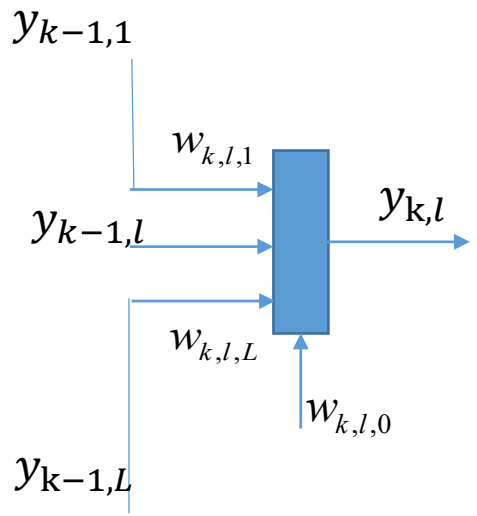
K- layer Network

- Select the right nomenclature:
- Neuron has weights $w_{k,l,m,i}$ to show
  layer k, perceptron l, weight m, epoch i
  bias weight goes with index m=0, forming a vector

$$\underline{w}_{k,l,i}^{T} = \left[w_{k,l,0,i}, w_{k,l,1,i}, w_{k,l,2,i}, \ldots, w_{k,l,m,i}, \ldots, w_{k,l,L,i} \right]$$

- Its output is $y_{k,l,i}$ to show
  layer k, perceptron l, epoch i
- Its inputs are $y_{k-1,m,i}$ the outputs of the previous perceptrons of layer k-1 and form an input vector

$$\underline{x}_{k,i}^{T} = \left[1, y_{k-1,1,i}, y_{k-1,2,i}, \ldots, y_{k-1,l,i}, \ldots, y_{k-1,L,i} \right]$$



Nomenclature

- As the number of indexes is rather large and thus the notation clumsy we will drop some of the indexes if they are not required for the understanding.
- For example, as the last layer has only one perceptron, we do not need to use an index for that layer. $\underline{\hat{w}}_{K,1,i} = \underline{\hat{w}}_{K,i}, \quad y_{K,1,i} = y_{K,i}$
- Similarly, the input vector of the first layer is all the same and given from our observations. There is no need to give it an additional index 1 and we can drop it. $\underline{x}_{1,i} = \underline{x}_i$

The PLA for a two layered Network using new nomenclature

- Updating the output layer weights $\underline{w}_{2,l,i} = \underline{w}_{2,1,i} = \underline{w}_{2,i}$; i=0,1,2,…, l=1

$$\underline{\hat{w}}_{2,i} = \underline{\hat{w}}_{2,i-1} + \mu \underline{x}_{2,i} \left(y_{2,i} - f\left(\underline{x}_{2,i}^T \underline{\hat{w}}_{2,i-1} \right) \right) f'\left(\underline{x}_{2,i}^T \underline{\hat{w}}_{2,i-1} \right)$$

$$\underline{x}_{2,i}^T = \left[1, y_{1,1,i}, y_{1,2,i}, y_{1,3,i} \right]$$

- Updating the input layer weights: $\underline{w}_{1,l,i}$: i=0,1,2,…, l=1,2,3

$$\underline{\hat{w}}_{1,l,i} = \underline{\hat{w}}_{1,l,i-1} + \mu \left(y_{2,i} - f\left(\underline{x}_{2,i}^T \underline{\hat{w}}_{2,i-1} \right) \right) f'\left(\underline{x}_{2,i}^T \underline{\hat{w}}_{2,i-1} \right) \qquad \underbrace{\frac{\partial \underline{x}_{2,i}^T \underline{\hat{w}}_{2,i-1}}{\partial \underline{\hat{w}}_{1,l,i-1}}}$$

$$\frac{\partial y_{1,l}}{\partial \underline{\hat{w}}_{1,l.i-1}} \hat{w}_{2,1,l,i-1} = \hat{w}_{2,l,i-1} f'(\underline{x}_i^T \underline{\hat{w}}_{1,l,i-1}) \underline{x}_i$$

$$= \underline{\hat{w}}_{1,l,i-1} + \mu \left(y_{2,i} - f\left(\underline{x}_{2,i}^T \underline{\hat{w}}_{2,i-1} \right) \right) f'\left(\underline{x}_{2,i}^T \underline{\hat{w}}_{2,i-1} \right) \hat{w}_{2,l,i-1} f'(\underline{x}_i^T \underline{\hat{w}}_{1,l,i-1}) \underline{x}_i$$

The PLA for a three layer Network

- We can now continue the update equations for the output layer: $\underline{w}_{3,l,i}=\underline{w}_{3,1,i}=\underline{w}_{3,i}$;i=0,1,2,…; l=1

$$\underline{\hat{w}}_{3,i}=\underline{\hat{w}}_{3,i-1}+\mu\left(y_{3,i}-f\left(\underline{x}_{3,i}^{T}\underline{\hat{w}}_{3,i-1}\right)\right)f'\left(\underline{x}_{3,i}^{T}\underline{\hat{w}}_{3,i-1}\right)\underline{x}_{3,i}$$

$$\underline{x}_{3,i}^{T}=\left[1,y_{2,1,i},y_{2,2,i},\ldots,y_{2,m,i},\ldots,y_{2,L,i}\right]$$

- 2nd layer:

$$\underline{\hat{w}}_{2,l,i}=\underline{\hat{w}}_{2,l,i-1}+\mu\left(y_{3,i}-f\left(\underline{x}_{3,i}^{T}\underline{\hat{w}}_{3,i-1}\right)\right)f'\left(\underline{x}_{3,i}^{T}\underline{\hat{w}}_{3,i-1}\right)\hat{w}_{3,l,i-1}f'(\underline{x}_{2,i}^{T}\underline{\hat{w}}_{2,l,i-1})\underline{x}_{2,i}$$

$$\underline{x}_{2,i}^{T}=\left[1,y_{1,1,i},y_{1,2,i},\ldots,y_{1,m,i},\ldots,y_{1,L,i}\right]$$

- 1st layer:

$$\underline{\hat{w}}_{1,l,i}=\underline{\hat{w}}_{1,l,i-1}+\mu\left(y_{3,i}-f\left(\underline{x}_{3,i}^{T}\underline{\hat{w}}_{3,i-1}\right)\right)f'\left(\underline{x}_{3,i}^{T}\underline{\hat{w}}_{3,i-1}\right)\sum_{k=1}^{L}\frac{\partial y_{2,k,i-1}}{\partial\underline{\hat{w}}_{1,l,i-1}}\hat{w}_{3,k,i-1}$$

The PLA for a three layer Network

- Obviously, the first neural layer update is more sophisticated

$$\underline{\hat{\mathrm{w}}}_{1,l,i} = \underline{\hat{\mathrm{w}}}_{1,l,i-1} + \mu\left(\mathrm{y}_{3,i} - f\left(\underline{\mathrm{x}}_{3,i}^{H}\,\underline{\hat{\mathrm{w}}}_{3,i-1}\right)\right) f'\left(\underline{\mathrm{x}}_{3,i}^{H}\,\underline{\hat{\mathrm{w}}}_{3,i-1}\right)\sum_{k=1}^{L}\frac{\partial y_{2,k,i-1}}{\partial \underline{\hat{w}}_{1,l,i-1}}\hat{w}_{3,k,i-1}$$

$$\frac{\partial y_{2,k,i-1}}{\partial \underline{\hat{w}}_{1,l,i-1}} = f'\left(\underline{\mathrm{x}}_{2,i}^{T}\,\underline{\hat{\mathrm{w}}}_{2,k,i-1}\right)\hat{w}_{2,k,l,i-1}\frac{\partial y_{1,l,i-1}}{\partial \underline{\hat{w}}_{1,l,i-1}} = f'\left(\underline{\mathrm{x}}_{2,i}^{H}\,\underline{\hat{\mathrm{w}}}_{2,k,i-1}\right)\underbrace{\hat{w}_{2,k,l,i-1}f'\left(\underline{\mathrm{x}}_{i}^{T}\,\underline{\hat{\mathrm{w}}}_{1,l,i-1}\right)}_{\delta_{1,l,i}}\underline{\mathrm{x}}_{i}$$

- We recognize that many parts of the update have been computed in the upper layers and can be reused.

The PLA for a three layer Network

- Using short notations (but full index notation):

$$\delta_{1,k,l,i} = \hat{w}_{2,k,l,i-1} f'\left(\underline{\mathrm{x}}_{1,i}^{T} \underline{\hat{\mathrm{w}}}_{1,l,i-1}\right) \qquad ; l = 1,2,\ldots,L$$

$$\delta_{2,l,i} = \hat{w}_{3,1,l,i-1} f'\left(\underline{\mathrm{x}}_{2,i}^{T} \underline{\hat{\mathrm{w}}}_{2,l,i-1}\right) \qquad ; l = 1,2,\ldots,L$$

$$\delta_{3,i} = \left(\mathrm{y}_{3,1,i} - f\left(\underline{\mathrm{x}}_{3,i}^{T} \underline{\hat{\mathrm{w}}}_{3,1,i-1}\right)\right) f'\left(\underline{\mathrm{x}}_{3,i}^{T} \underline{\hat{\mathrm{w}}}_{3,1,i-1}\right)$$

$$\underline{\hat{\mathrm{w}}}_{3,1,i} = \underline{\hat{\mathrm{w}}}_{3,1,i-1} + \mu\delta_{3,i}\,\underline{\mathrm{x}}_{3,i}$$

$$\underline{\hat{\mathrm{w}}}_{2,l,i} = \underline{\hat{\mathrm{w}}}_{3,l,i-1} + \mu\delta_{3,i}\delta_{2,l,i}\,\underline{\mathrm{x}}_{2,i} \qquad ; l = 1,2,\ldots,L$$

$$\underline{\hat{\mathrm{w}}}_{1,l,i} = \underline{\hat{\mathrm{w}}}_{1,l,i-1} + \mu\delta_{3,i}\sum_{k=1}^{L}\delta_{2,k,i}\delta_{1,k,l,i}\,\underline{\mathrm{x}}_{1,i}\,; l = 1,2,\ldots,L$$

The PLA for a three layer Network

- Many modifications are possible:
  - For example, once $\underline{w}_3$ is computed, δ_3 can be renewed and then $\underline{w}_1$ and $\underline{w}_2$ computed with the new δ_3. The same goes with δ_2 once $\underline{w}_2$ is computed.
  - Different step-sizes for each layer and possibly adaptive step-sizes can be very useful.
  - Extensions to more than three layers are possible.

Probabilistic Interpretation

- As the outputs of each neuron are between zero and one, they can be interpreted as probabilities to obtain 0 and 1
- A measure in terms of probabilities is cross entropy

$$\sum_{i=1}^{M} y_i \ln\left(f\left(\underline{\hat{x}}_i^T \underline{\hat{w}}\right)\right) - (1-y_i)\ln\left(1-f\left(\underline{\hat{x}}_i^T \underline{\hat{w}}\right)\right)$$

- Often to ensure probability measures, new nonlinear activity functions are introduced by proper normalization:

$$y_i = f(\underline{\hat{x}}_i^H \underline{\hat{w}}) = \frac{\exp(\underline{\hat{x}}_i^H \underline{\hat{w}})}{\sum \exp(\underline{\hat{x}}_i^H \underline{\hat{w}})}$$

Choice of nonlinear functions

- Depending on the problem to solve many different nonlinear “activation” functions can be selected throughout the net (one for each layer)
- ReLUs (rectifying Linear Unit) are very popular to form classes or pre-classes
- Sigmoid, hyptan are commonly used for the binary classification problem as choice for the output layer
- Normalized activation functions (as shown on the previous slide) are often selected when a selection “one out of many” is to be achieved.
- A linear outcome (no non-linear mapping) can be selected as the outcome of the latent (center) nodes of an auto-encoder.

Historical Notes

- The term *Deep Learning* was introduced to the machine learning community by Rina Dechter in 1986,[25][13] and to artificial neural networks by Igor Aizenberg and colleagues in 2000, in the context of Boolean threshold neurons.[26][27]
- The first general, working learning algorithm for supervised, deep, feedforward, multilayer perceptrons was published by Alexey Ivakhnenko and Lapa in 1965.[28] A 1971 paper described a deep network with 8 layers trained by the group method of data handling algorithm.[29]
- Other deep learning working architectures, specifically those built for computer vision, began with the Neocognitron introduced by Kunihiko Fukushima in 1980.[30]
- In 1989, **Yann LeCun** et al. applied the standard backpropagation algorithm, which had been around as the reverse mode of automatic differentiation since 1970,[31][32][33][34] to a deep neural network with the purpose of recognizing handwritten ZIP codes on mail. While the algorithm worked, training required 3 days.[35]

Historical Notes

- In 1995, Brendan Frey demonstrated that it was possible to train (over two days) a network containing six fully connected layers and several hundred hidden units using the wake-sleep algorithm, co-developed with Peter Dayan and **Hinton**.[40] Many factors contribute to the slow speed, including the vanishing gradient problem analyzed in 1991 by Sepp Hochreiter.[41][42]
- In 2006, publications by **Geoff Hinton**, Ruslan Salakhutdinov, Osindero and Teh[56] [57][58] showed how a many-layered feedforward neural network could be effectively pre-trained one layer at a time, treating each layer in turn as an unsupervised restricted Boltzmann machine, then fine-tuning it using supervised backpropagation.[59] The papers referred to *learning* for *deep belief nets.*

Historical Notes

- Advances in hardware enabled the renewed interest. In 2009, Nvidia was involved in what was called the “big bang” of deep learning, “as deep-learning neural networks were trained with Nvidia graphics processing units (GPUs).”[79] That year, Google Brain used Nvidia GPUs to create capable DNNs. While there, Andrew Ng determined that GPUs could increase the speed of deep-learning systems by about 100 times.[80] In particular, GPUs are well-suited for the matrix/vector math involved in machine learning.[81][82] GPUs speed up training algorithms by orders of magnitude, reducing running times from weeks to days.[83][84] Specialized hardware and algorithm optimizations can be used for efficient processing.[85]

Historical Notes

- In March 2019, Yoshua Bengio, Geoffrey Hinton and Yann LeCun were awarded the Turing Award for conceptual and engineering breakthroughs that have made deep neural networks a critical component of computing.

- *Bengio, Yoshua; LeCun, Yann; Hinton, Geoffrey (2015). "Deep Learning". Nature.* ***521*** *(7553): 436–444. Bibcode:2015Natur.521..436L. doi:10.1038/nature14539. PMID 26017442.*

Understanding Deep Neural Learning

- Best is to view
- https://www.youtube.com/watch?v=VrMHA3yX_QI&list=PLUl4u3cNGP63gFHB6xb-kVBiQHYe_4hSi&index=13
- Starting at 35:48…

Understanding Deep Neural Learning

- Only through many hidden layers the dimensions are increased and thus many local minima, difficult to distinguish from the desired global minimum, appear now as saddle points and the gradient algorithms keeps running to find good solutions.
- An a-posteriori pruning can reduce the complexity significantly, once the network is fixed (allowing only small tracking steps afterwards)
- Special GPU structures are required to obtain the huge amount of computation for training.