

Iliya Valchanov

Neural Networks Overview

Course Notes

365  DataScience

Table of Content:

Abstract	3
1. The Layer	4
2. What is a Deep Net?	5
3. Why Do we Need Non-Linearities to Stack Layers?	7
4. Activation Functions.....	8
4.1 Common Activation Functions	9
4.2 Softmax Activation	10
5. Backpropagation.....	11
5.1 Backpropagation formula	12

Abstract

In these course notes, we are going to cover the advanced machine learning algorithm known as deep learning which is capable of creating highly accurate predictive models without having to give it any explicit instructions . It accomplishes that by working with large amounts of unstructured data, and mimics the human learning process by building a hierarchy of complex abstractions.

Keywords: deep net, backpropagation formula, Softmax activation, layers

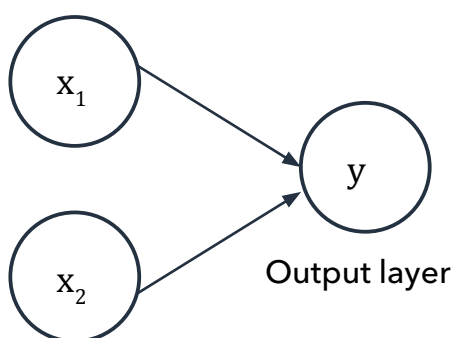
1. The Layer

An initial linear combination and the added non-linearity form a

The layer is the building block of neural networks

Minimal example (a simple neural network)

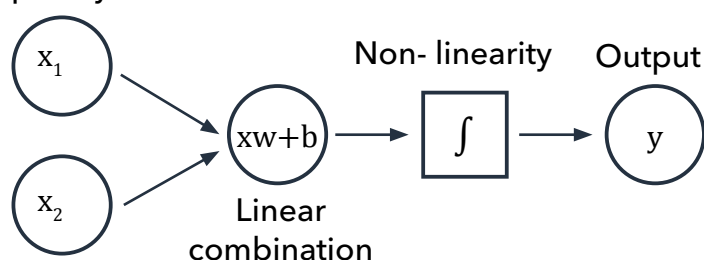
Input layer



In the minimal example we trained a neural network which had no depth. There were solely an input layer and an output layer. Moreover, the output was simply **a linear combination** of the input.

Neural networks

Input layer



Neural networks step on linear combinations, but add a non- linearity to each one of them. Mixing linear combinations and non-linearities allows us to model arbitrary functions.

2. What is a Deep Net?

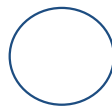
This is a deep neural network (deep net) with 5 layers.

How to read this diagram:

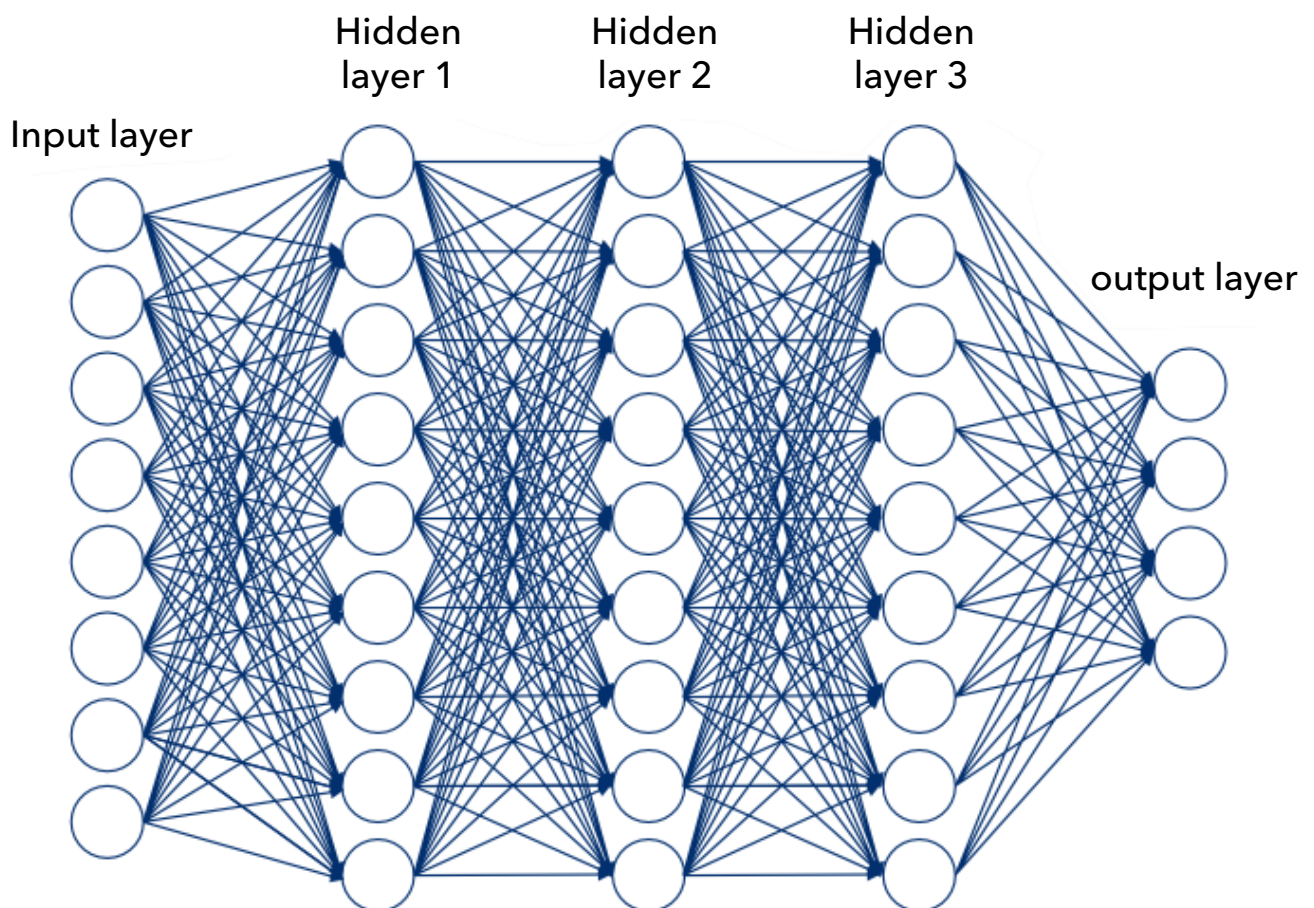
A layer



A unit (a neuron)



Arrows represent
mathematical
transformations

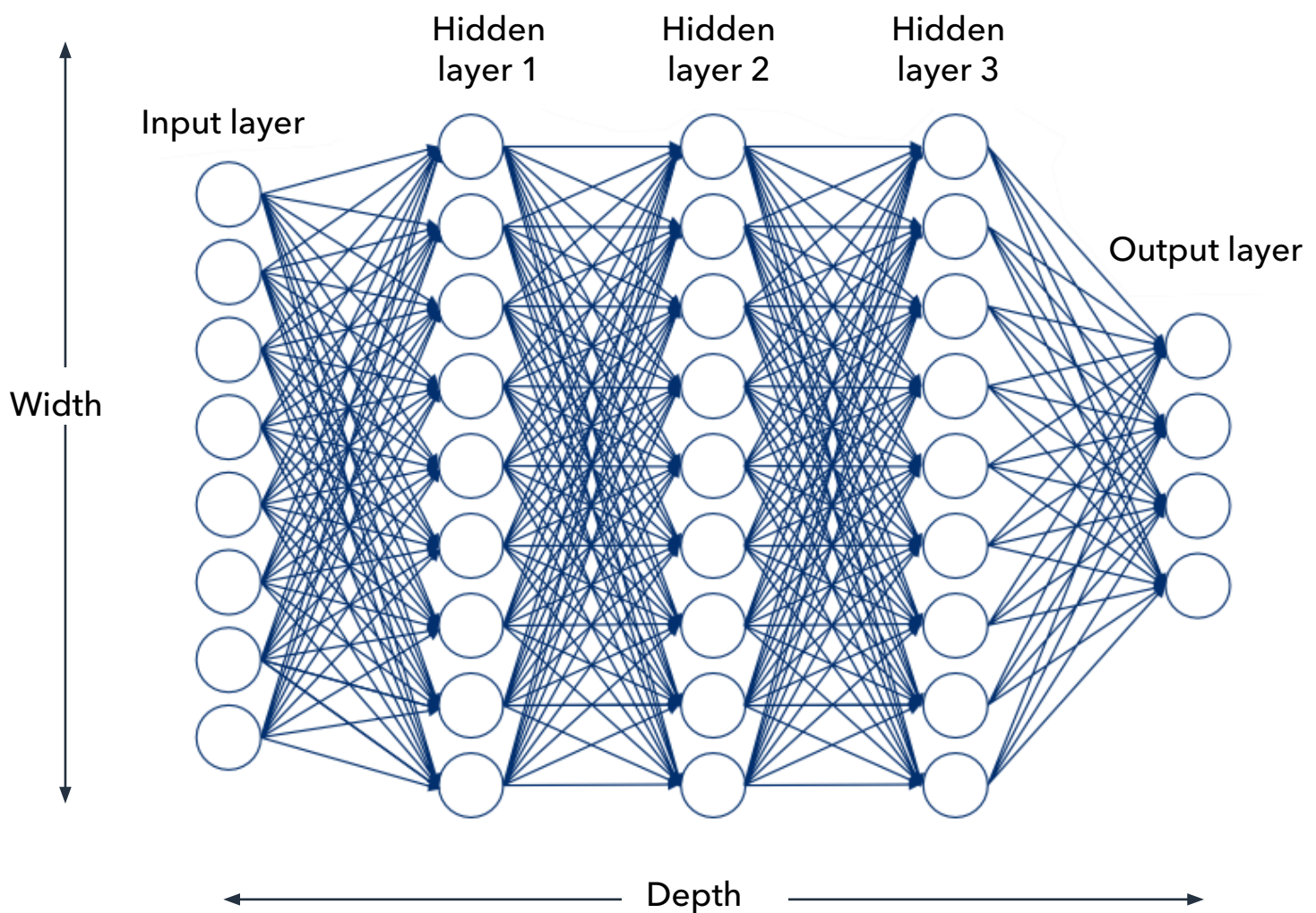


The **width** of a layer is the number of units in that layer

The **width** of the net is the number of units of the biggest layer

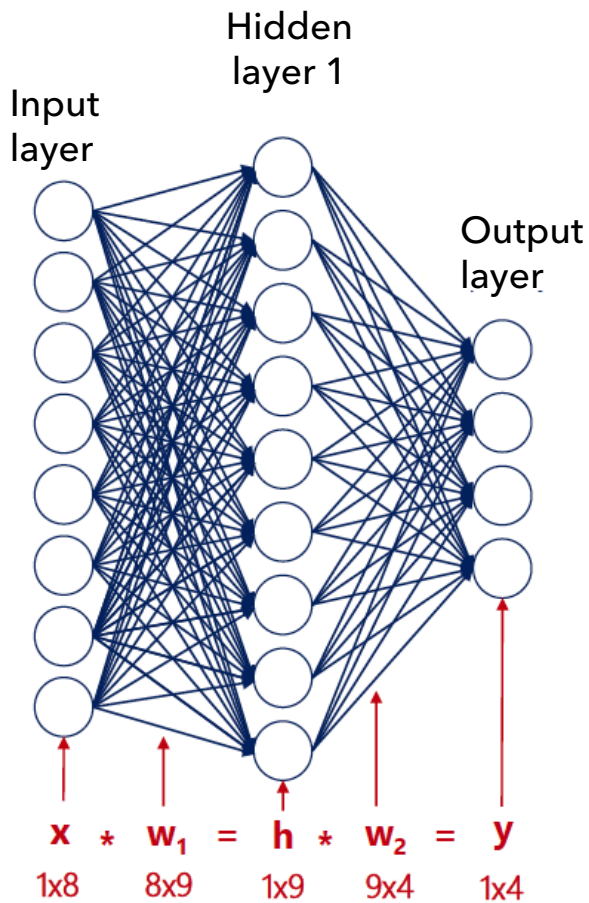
The **depth** of the net is equal to the number of layers or the number of hidden layers. The term has different definitions. More often than not, we are interested in the number of hidden layers (as there are always input and output layers).

The width and the depth of the net are called **hyperparameters**. They are values we manually chose when creating the net.



3. Why do we need non-linearities to stack layers?

You can see a net with no non-linearities: just linear combinations.



$$\mathbf{h} = \mathbf{x} * \mathbf{w}_1$$

$$\mathbf{y} = \mathbf{h} * \mathbf{w}_2$$

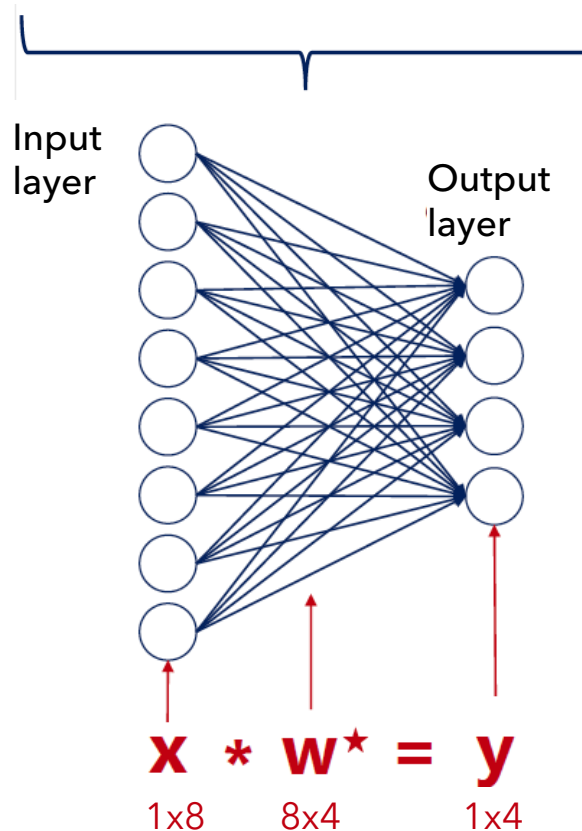
$$\mathbf{y} = \mathbf{x} * \mathbf{w}_1 * \mathbf{w}_2$$

8x9 9x4

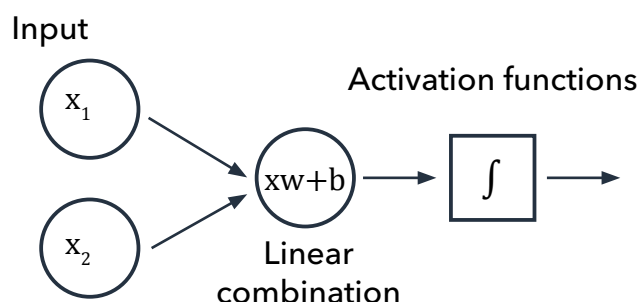
$$\mathbf{y} = \mathbf{x} * \mathbf{w}$$

8x4

Two consecutive linear transformations are equivalent to a single one.



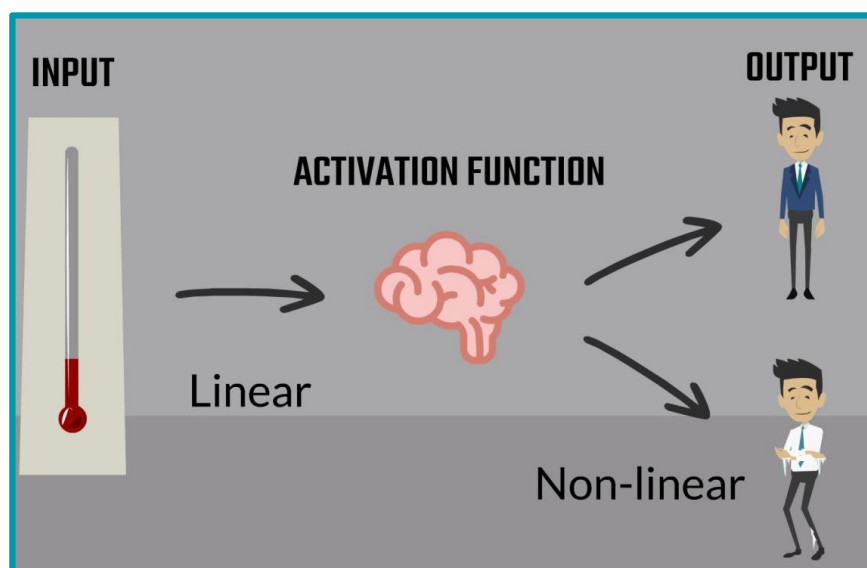
4. Activation Functions



In the respective lesson, we gave an example of temperature change. The temperature starts decreasing (which is a numerical change). Our brain is a kind of an 'activation function'. It tells us whether it is **cold enough** for us to put on a jacket.

Putting on a jacket is a binary action: 0 (no jacket) or 1 (jacket).

This is a very intuitive and visual (yet not so practical) example of how activation functions work.

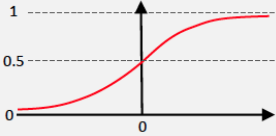
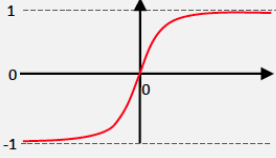
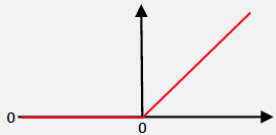


Activation functions (non-linearities) are needed so we can break the linearity and represent more complicated relationships

Moreover, activation functions are required in order to **stack layers**.

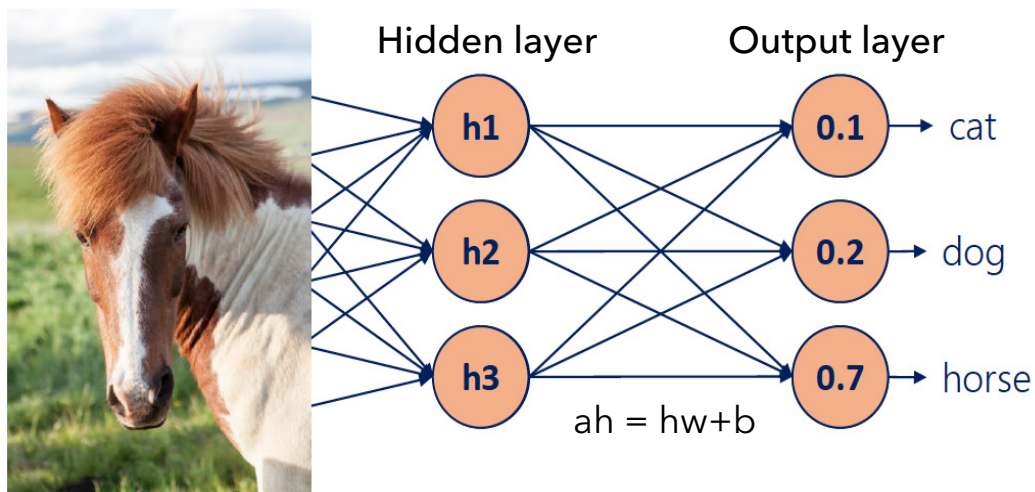
Activation functions transform inputs into outputs of a different kind.

4.1 Common Activation Functions

Name	Formula	Derivative	Graph	Range
sigmoid (logistic function)	$\sigma(a) = \frac{1}{1 + e^{-a}}$	$\frac{\partial \sigma(a)}{\partial a} = \sigma(a)(1 - \sigma(a))$		(0,1)
TanH (hyperbolic tangent)	$\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$	$\frac{\partial \tanh(a)}{\partial a} = \frac{4}{(e^a + e^{-a})^2}$		(-1,1)
ReLu (rectified linear unit)	$\text{relu}(a) = \max(0, a)$	$\frac{\partial \text{relu}(a)}{\partial a} = \begin{cases} 0, & \text{if } a \leq 0 \\ 1, & \text{if } a > 0 \end{cases}$		(0,∞)
softmax	$\sigma_i(\mathbf{a}) = \frac{e^{a_i}}{\sum_j e^{a_j}}$	$\frac{\partial \sigma_i(\mathbf{a})}{\partial a_j} = \sigma_i(\mathbf{a}) (\delta_{ij} - \sigma_j(\mathbf{a}))$		(0,1)

All common activation functions are: monotonic, continuous, and differentiable. These are important properties needed for the optimization.

4.2 Softmax Activation



The softmax activation transforms a bunch of arbitrarily large or small numbers into a valid probability distribution.

While other activation functions get an input value and transform it, regardless of the other elements, the softmax considers the information about the **whole set of numbers** we have.

The values that softmax outputs are in the range from 0 to 1 and their sum is exactly 1 (like probabilities).

Example:

$$\text{softmax}(\mathbf{a}) = \frac{e^{a_i}}{\sum_j e^{a_j}}$$

$$\mathbf{a} = [-0.21, 0.47, 1.72]$$

$$\sum_j e^{a_j} = e^{-0.21} + e^{0.47} + e^{1.72} = 8$$

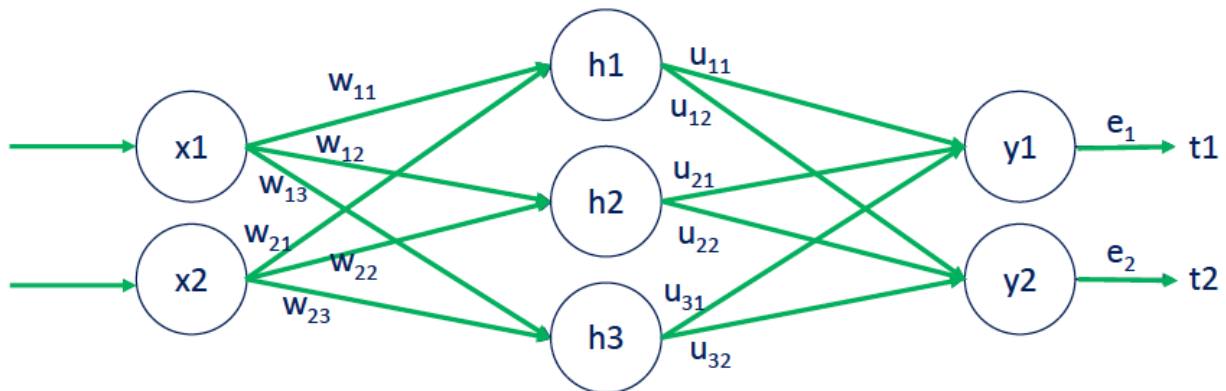
$$\text{softmax}(\mathbf{a}) = \left[\frac{e^{-0.21}}{8}, \frac{e^{0.47}}{8}, \frac{e^{1.72}}{8} \right]$$

$$\mathbf{y} = [0.1, 0.2, 0.7] \quad \text{probability distribution}$$

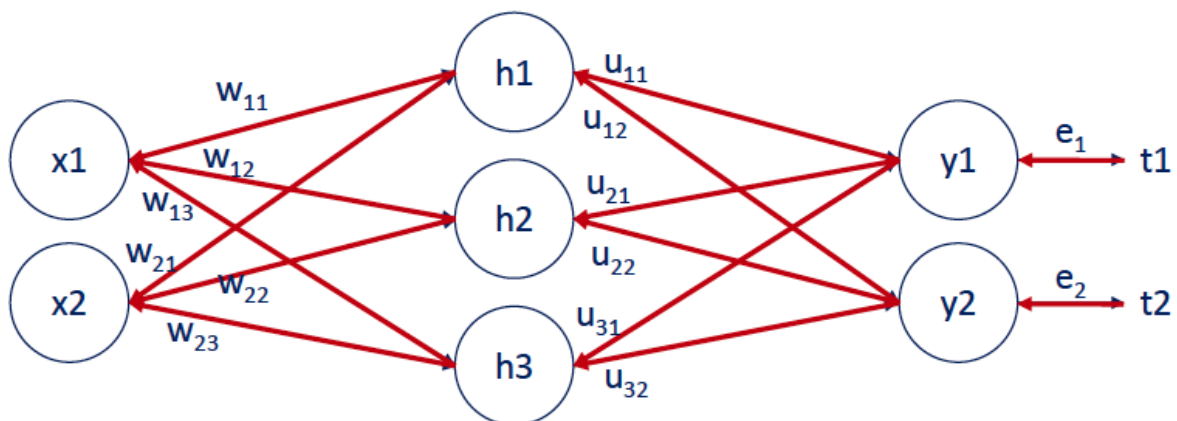
The property of the softmax to output probabilities is so useful and intuitive that it is often used as the activation function for the **final (output) layer**.

However, when the softmax is used prior to that (as the activation of a hidden layer), the results are not as satisfactory. That's because a lot of the information about the variability of the data is lost.

5. Backpropagation

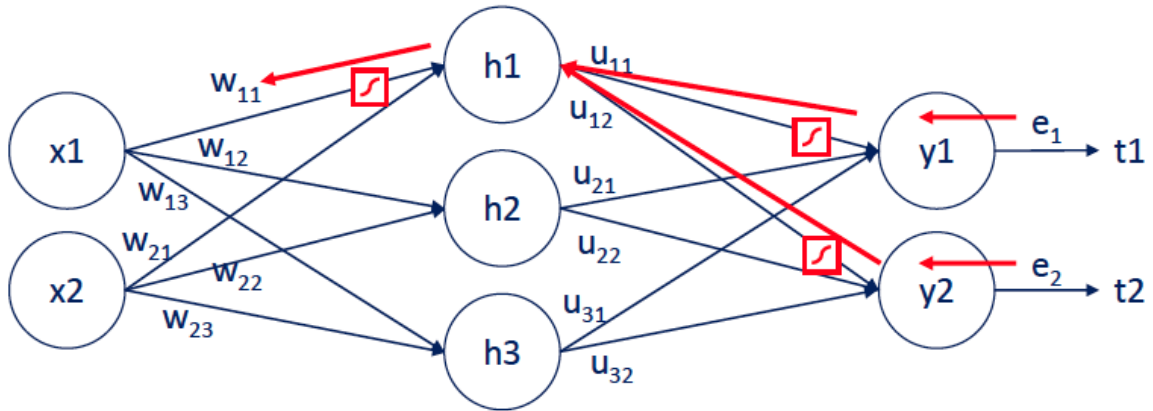


Forward propagation is the process of pushing inputs through the net. At the end of each epoch, the obtained outputs are compared to targets to form the errors



Backpropagation of errors is an **algorithm** for neural networks using gradient descent. It consists of calculating the contribution of each **parameter** to the errors. We backpropagate the **errors** through the net and **update** the parameters (weights and biases) accordingly.

5.1 Backpropagation Formula



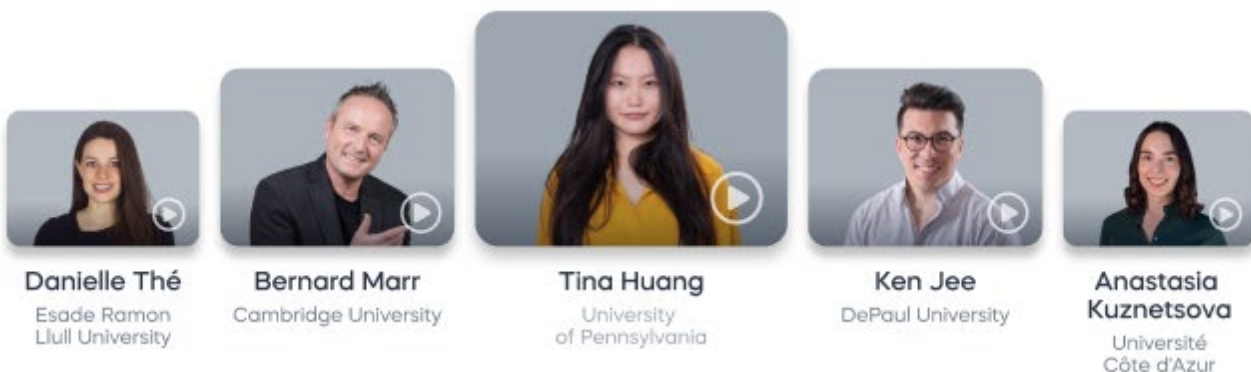
$$\frac{\partial L}{\partial w_{ij}} = \delta_j x_i \quad \text{where} \quad \delta_j = \sum_k \delta_k w_{jk} y_i (1 - y_i)$$

If you want to examine the full derivation, please make use of the PDF we made available in the section: Backpropagation. A peek into the Mathematics of Optimization.

Learn DATA SCIENCE anytime, anywhere, at your own pace.

If you found this resource useful, check out our **e-learning program**. We have everything you need to succeed in data science.

Learn the most sought-after data science skills from **the best experts in the field!**
Earn a **verifiable certificate** of achievement trusted by employers worldwide and future proof your career.



Comprehensive training, exams, certificates.

- ✓ 162 hours of video
- ✓ 599+ Exercises
- ✓ Downloadables
- ✓ Exams & Certification
- ✓ Personalized support
- ✓ Resume Builder & Feedback
- ✓ Portfolio advice
- ✓ New content
- ✓ Career tracks

Join a global community of 1.8 M successful students with an annual subscription
at 60% OFF with coupon code **365RESOURCES**.

~~\$432~~ **\$172.80**/year



Start at 60% Off

VAT may be applied

365[✓]DataScience



Iliya Valchanov

Email: team@365datascience.com

365[°] DataScience