Deep Learning

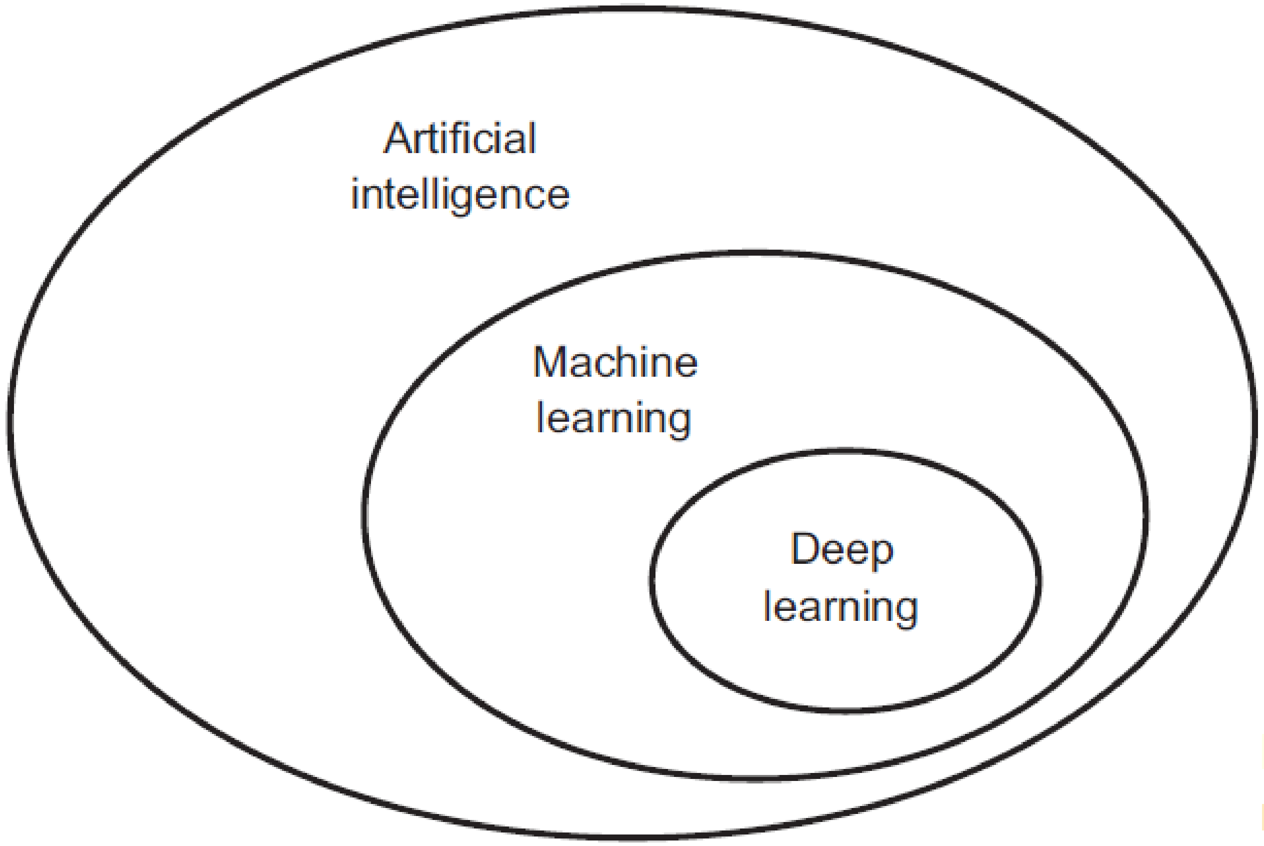


Figure 1.1 Artificial intelligence, machine learning, and deep learning

Introduction

- Deep learning is a specific subfield of machine learning: a new take on learnin representations from data that puts an emphasis on learning successive layers o increasingly meaningful representations.
- The deep in deep learning isn't a reference to any kind of deeper understandin achieved by the approach; rather, it stands for this idea of successive layers o representations. How many layers contribute to a model of the data is called th depth of the model. Other appropriate names for the field could have bee layered representations learning and hierarchical representations learning.
- In deep learning, these layered representations are (almost always) learned vi models called neural networks, structured in literal layers stacked on top of eac other

Deep learning and brain

- The term "neural network" draws inspiration from neurobiology, and while certain fundamental concepts in deep learning were influenced by our knowledge of the brain, it's essential to note that deep-learning models are not accurate representations of the brain. There is no supporting evidence indicating that the brain employs learning mechanisms akin to those used in contemporary deep-learning models. Despite popular science articles asserting that deep learning operates similarly to the brain or is modelled after it, this assertion is not accurate.

- You may forget anything you may have read about hypothetical links between deep learning and biology. You should view deep learning as a mathematical framework for learning representations from data.

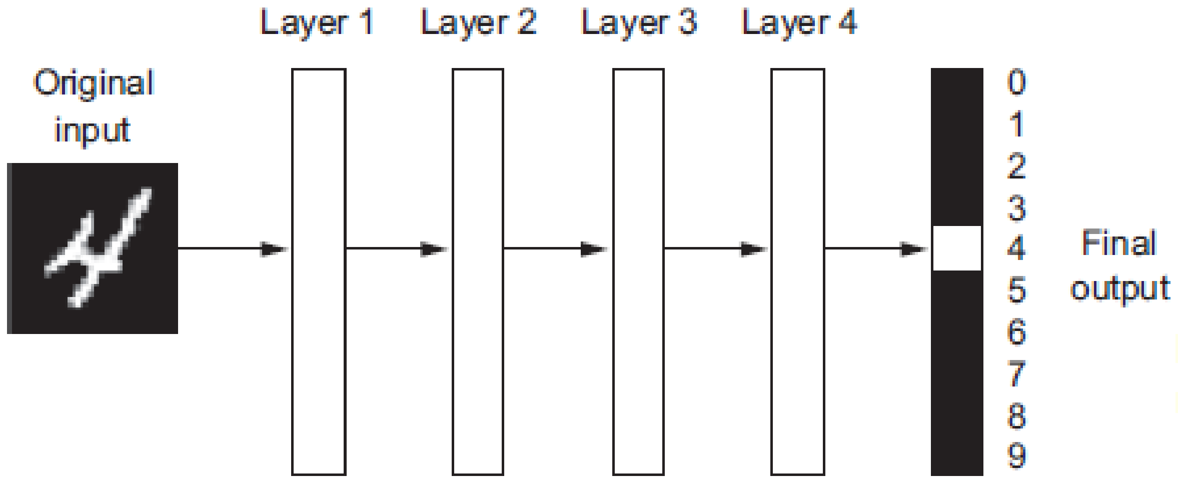


Figure 1.5 A deep neural network for digit classification

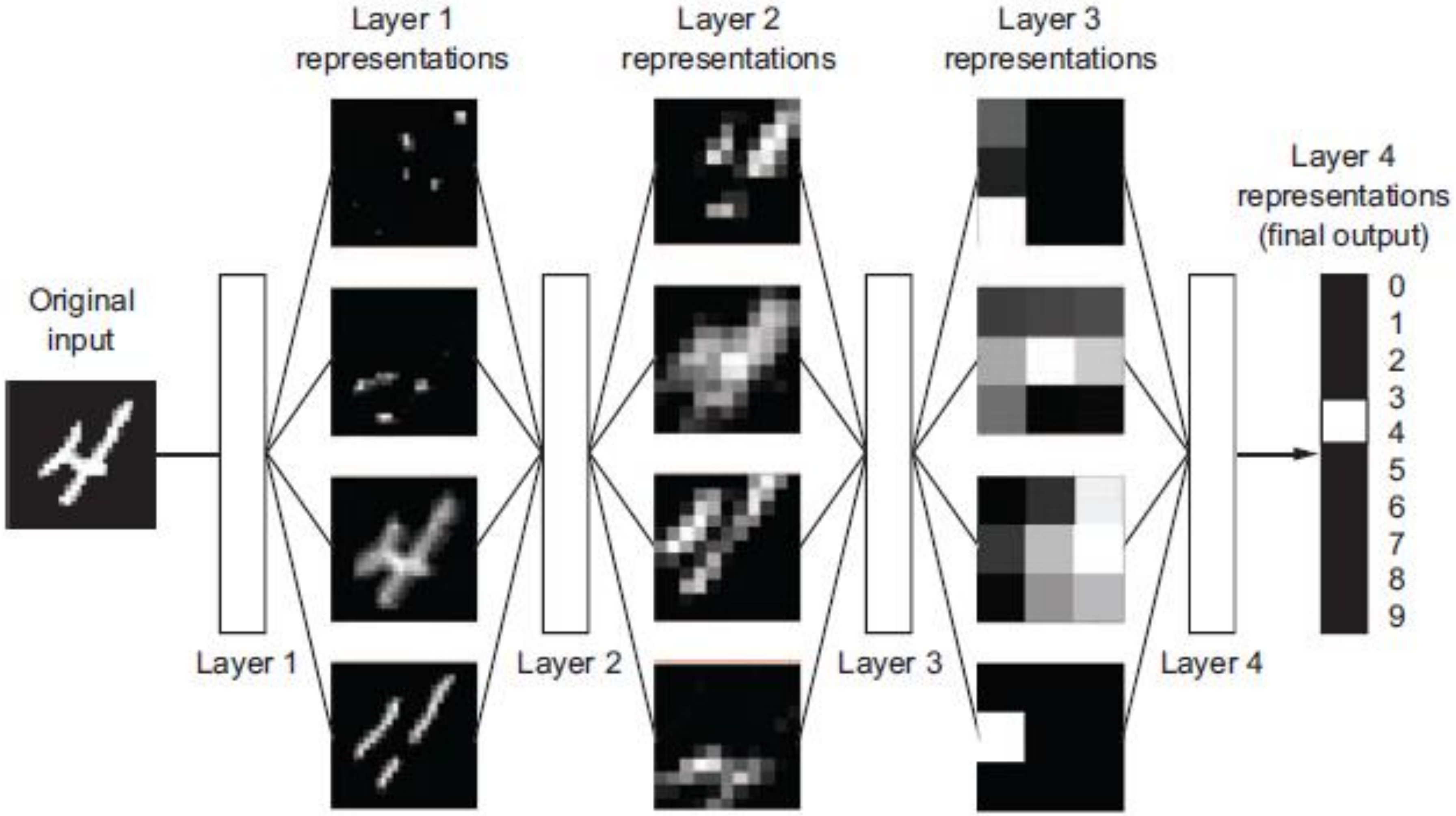


Figure 1.6 Deep representations learned by a digit-classification model

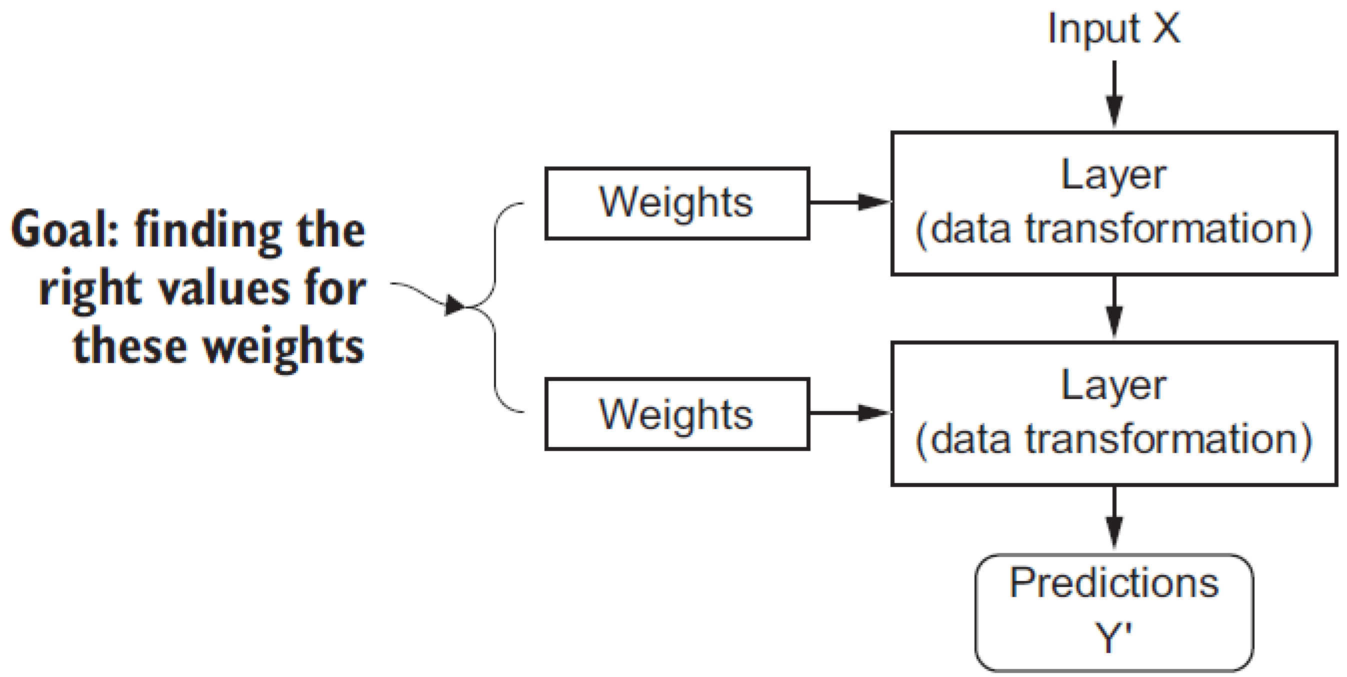


Figure 1.7 A neural network is parameterized by its weights.

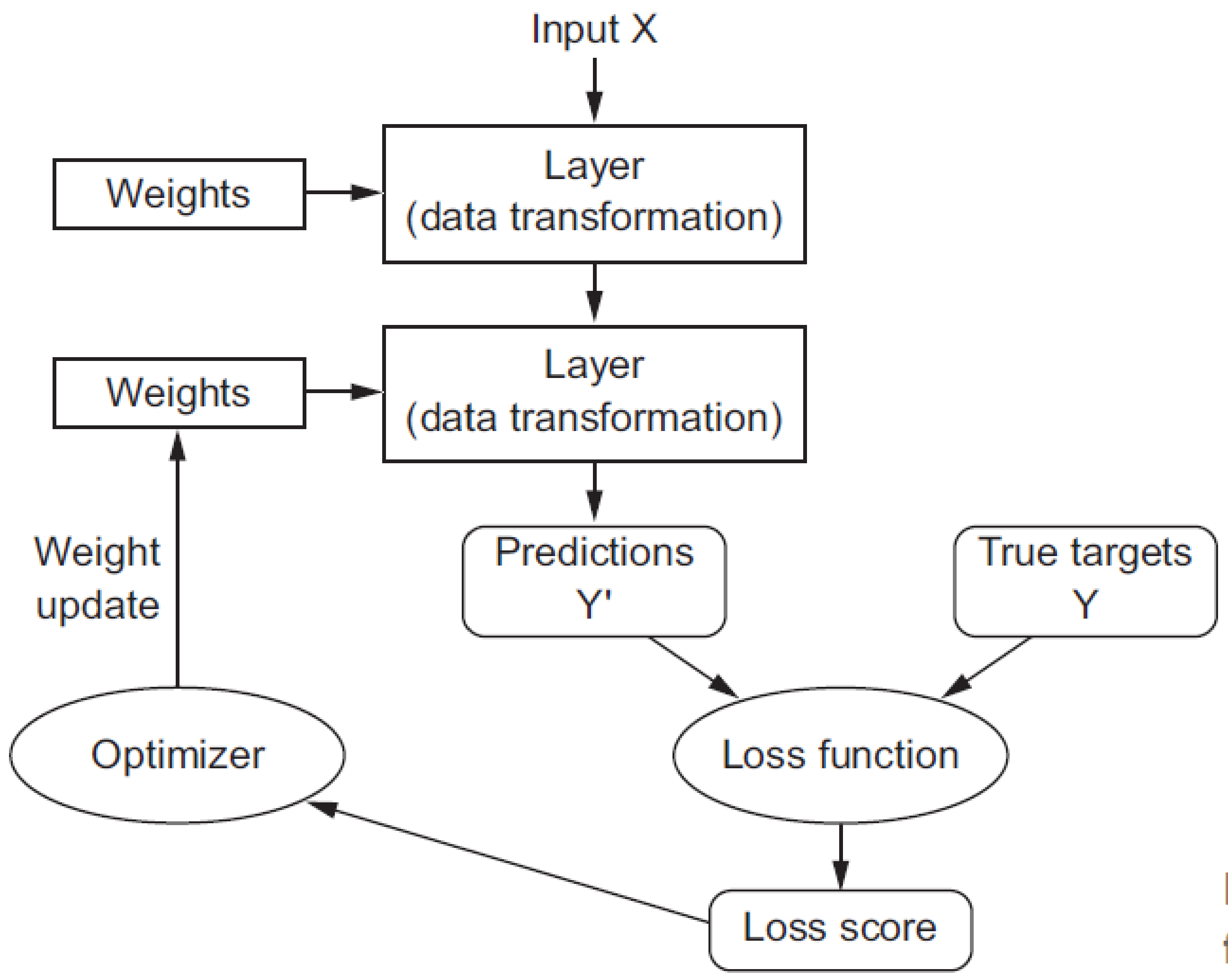


Figure 1.9 The loss score is used as a feedback signal to adjust the weights.

- *Learning* means finding a combination of model parameters that minimizes a loss function for a given set of training data samples and their corresponding targets.
- Learning happens by drawing random batches of data samples and their targets, and computing the gradient of the network parameters with respect to the loss on the batch. The network parameters are then moved a bit (the magnitude of the move is defined by the learning rate) in the opposite direction from the gradient.
- The entire learning process is made possible by the fact that neural networks are chains of differentiable tensor operations, and thus it's possible to apply the chain rule of derivation to find the gradient function mapping the current parameters and current batch of data to a gradient value.

What deep learning has achieved so far?

In particular, deep learning has achieved the following breakthroughs, all in historically difficult areas of machine learning:

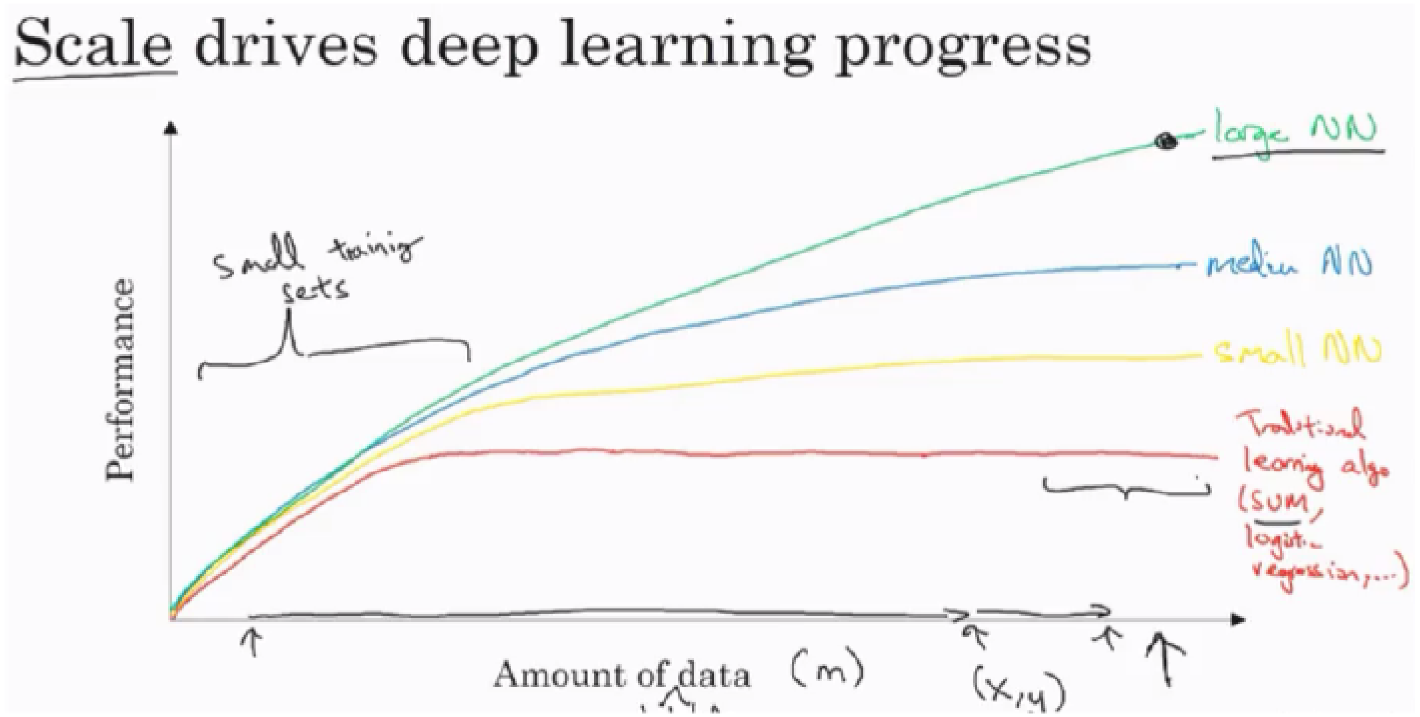
- Near-human-level image classification
- Near-human-level speech recognition
- Near-human-level handwriting transcription
- Improved machine translation
- Improved text-to-speech conversion
- Digital assistants such as Google Now and Amazon Alexa
- Near-human-level autonomous driving
- Improved ad targeting, as used by Google, Baidu, and Bing
- Improved search results on the web
- Ability to answer natural-language questions
- Superhuman Go playing

Supervised learning with neural networks

- Different types of neural networks for supervised learning which includes:
  - CNN or convolutional neural networks (Useful in computer vision)
  - RNN or Recurrent neural networks (Useful in Speech recognition or NLP)
  - Standard NN (Useful for Structured data)
  - Hybrid/custom NN or a Collection of NNs types
- Structured data is like the databases and tables.
- Unstructured data is like images, video, audio, and text.

Why is deep learning taking off?

- Deep learning is taking off for 3 reasons:

  i. Data:
    - Using this image we can conclude:
      - 

    - For small data NN can perform as Linear regression or SVM (Support vector machine)
    - For big data a small NN is better that SVM
    - For big data a big NN is better that a medium NN is better that small NN.
    - Hopefully we have a lot of data because the world is using the computer a little bit more
      - Mobiles
      - IOT (Internet of things)

ii. Computation:

- GPUs.
- Powerful CPUs.
- Distributed computing.
- ASICs

iii. Algorithm:

a. Creative algorithms has appeared that changed the way NN works.

  - For example using RELU function is so much better than using SIGMOID function in training a NN becaus it helps with the vanishing gradient problem.

Neural networks

A neural network (NN), just like a regression or an SVM model, is a mathematical function:

$$y = f_{NN}(\mathbf{x}).$$

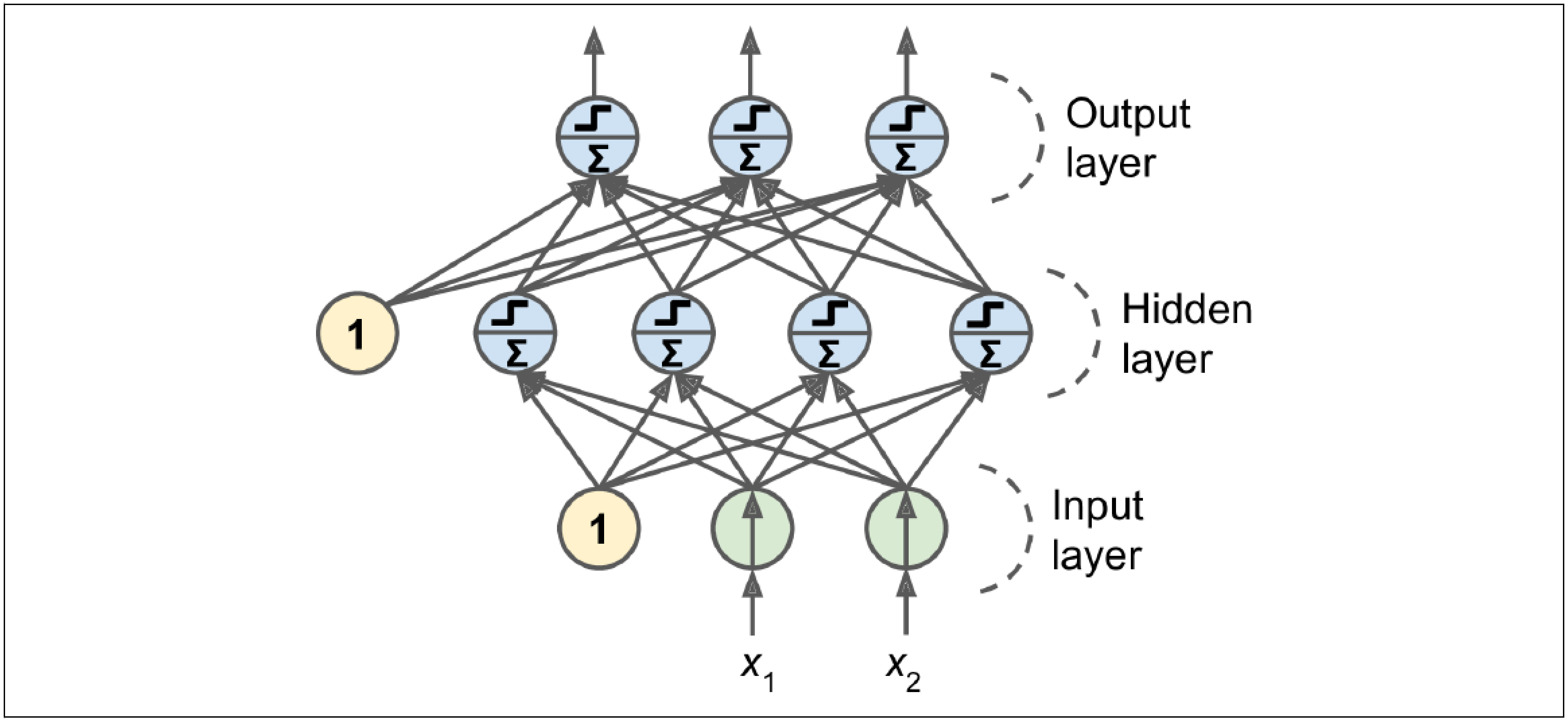
The function f_{NN} has a particular form: it's a *nested function*. You have probably already heard of neural network **layers**. So, for a 3-layer neural network that returns a scalar, f_{NN} looks like this:

$$y = f_{NN}(\mathbf{x}) = f_3(\boldsymbol{f}_2(\boldsymbol{f}_1(\mathbf{x}))).$$

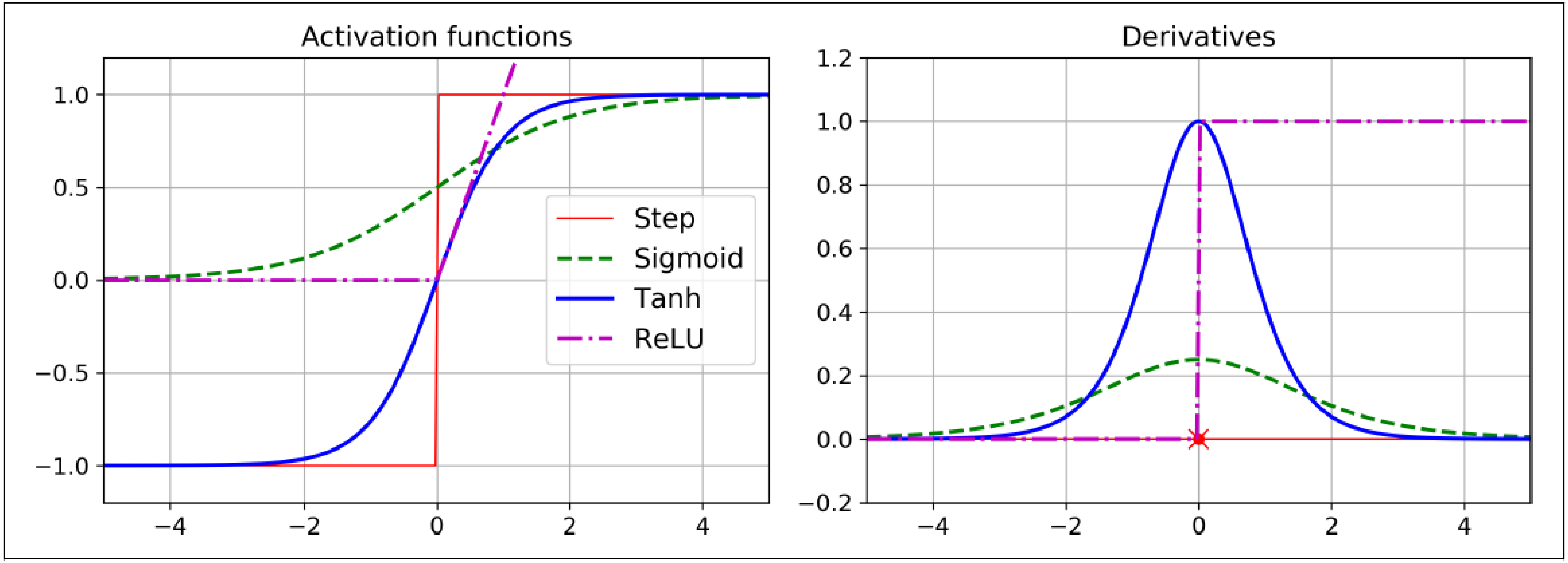
In the above equation, $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ are vector functions of the following form:

$$\boldsymbol{f}_l(\mathbf{z}) \stackrel{\text{def}}{=} \boldsymbol{g}_l(\mathbf{W}_l\mathbf{z} + \mathbf{b}_l), \tag{1}$$

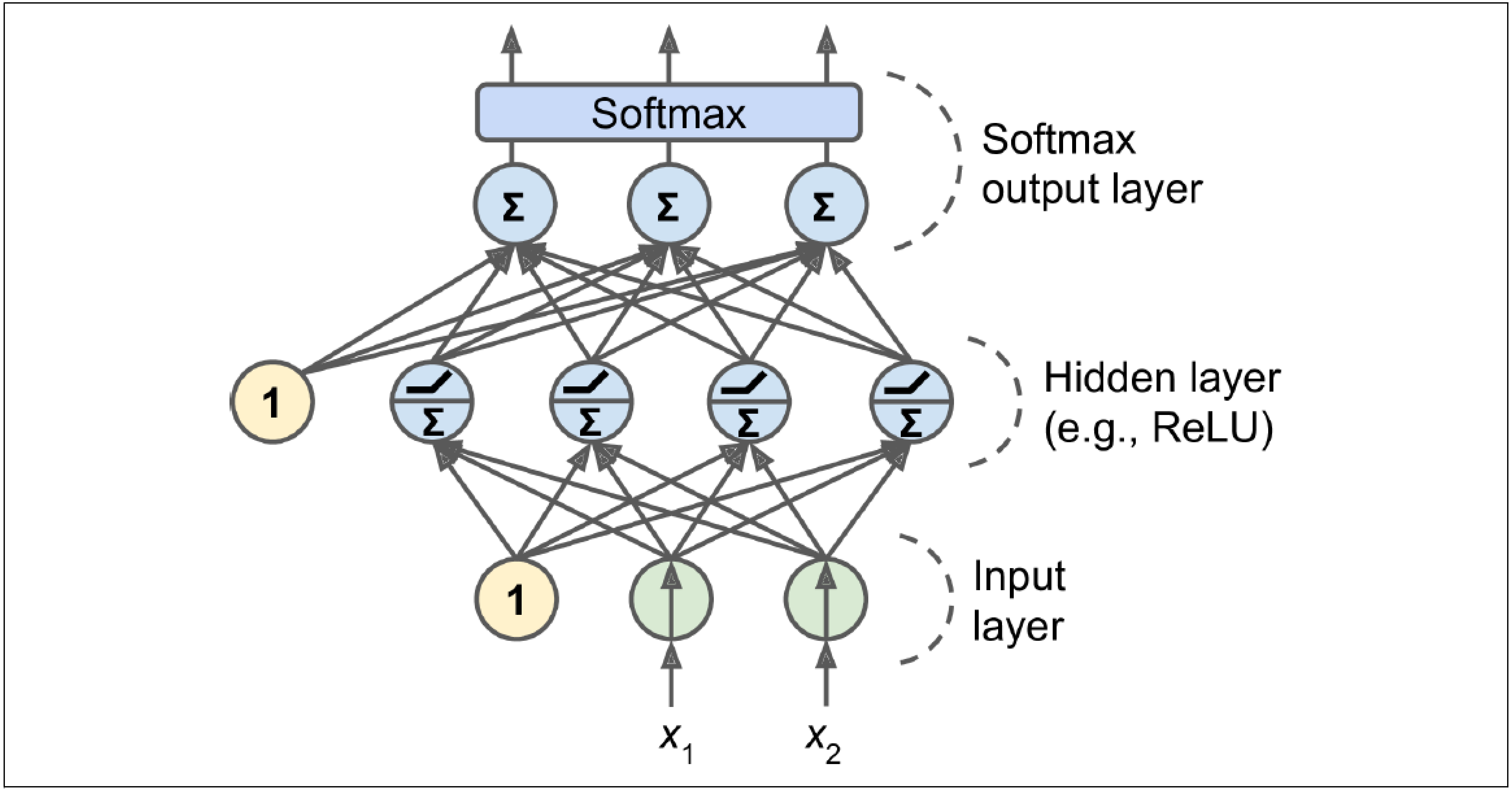
where l is called the layer index and can span from 1 to any number of layers. The function $\boldsymbol{g}_l$ is called an **activation function**. It is a fixed, usually nonlinear function chosen by the data analyst before the learning is started. The parameters $\mathbf{W}_l$ (a matrix) and $\mathbf{b}_l$ (a vector) for each layer are learned using the familiar gradient descent by optimizing, depending on the task, a particular cost function (such as MSE).



Multi-Layer Perceptron



Activation functions and their derivatives



A modern MLP (including ReLU and softmax) for classification

Typical Classification MLP Architecture

Hyperparameter	Binary classification	Multilabel binary classification	Multiclass classification
Input and hidden layers	Same as regression	Same as regression	Same as regression
# output neurons	1	1 per label	1 per class
Output layer activation	Logistic	Logistic	Softmax
Loss function	Cross-Entropy	Cross-Entropy	Cross-Entropy

Typical Regression MLP Architecture

Hyperparameter	Typical Value
# input neurons	One per input feature (e.g., 28 x 28 = 784 for MNIST)
# hidden layers	Depends on the problem. Typically 1 to 5.
# neurons per hidden layer	Depends on the problem. Typically 10 to 100.
# output neurons	1 per prediction dimension
Hidden activation	ReLU (or SELU, see Chapter 11)
Output activation	None or ReLU/Softplus (if positive outputs) or Logistic/Tanh (if bounded outputs)
Loss function	MSE or MAE/Huber (if outliers)

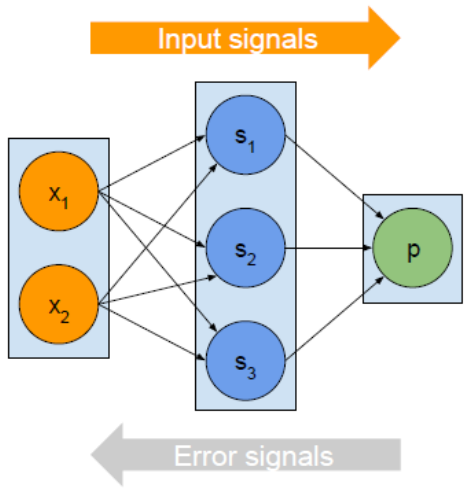
Training a neural network: Backward propagation and gradient descent algorithm

The backpropagation algorithm is a Gradient Descent based optimization step (assuming convex function profile) using an efficient technique for computing the gradients automatically. It is able to compute the gradient of the network's error with regards to every single model parameter using just two passes through the network (one forward, one backward). In other words, it can find out how each connection weight and each bias term should be tweaked in order to reduce the error. Once it has these gradients, it just performs a regular Gradient Descent step, and the whole process is repeated until the network converges to the solution.

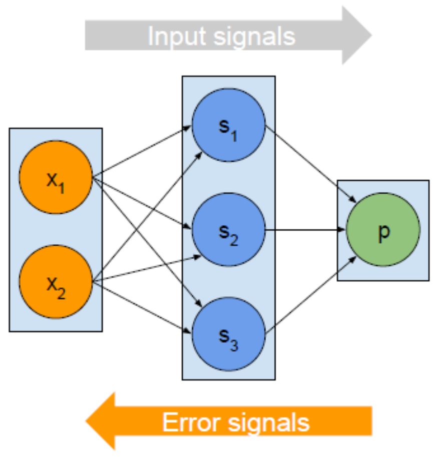
Let's discuss the steps of the algorithm in details:

- **Forward pass:** It handles one mini-batch of training data at a time (batch size is usually set to power of 2, i.e., 16,32, 64,…), and it goes through the full training set multiple times. Each pass is called an *epoch.* Each mini-batch is passed to the network's input layer, which just sends it to the first hidden layer. The algorithm then computes the output of all the neurons in this layer (for every instance in the mini-batch). The result is passed on to the next layer, its output is computed and passed to the next layer, and so on until we get the output of the last layer (the output layer). This is the forward pass. It is exactly like making predictions, except all intermediate results are preserved since they are needed for the backward pass.
- **Error calculation:** Next, the algorithm measures the network's output error (i.e., it uses a loss function that compares the desired output and the actual output of the network, and returns some measure of the error.
- **Apply chain rule:** Then it computes how much each output connection contributed to the error. This is done analytically by simply applying the chain rule (perhaps the most fundamental rule in calculus), which makes this step fast and precise. The algorithm then measures how much of these error contributions came from each connection in the layer below, again using the chain rule—and so on until the algorithm reaches the input layer.
- **Gradient Descent step:** Finally, the algorithm performs a Gradient Descent step to tweak all the connection weights in the network, using the error gradients it just computed.

Mathematical interpretation

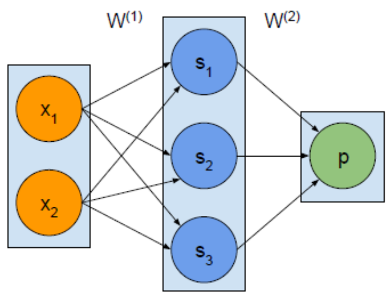


1. Get prediction
2. Calculate error
3. Calculate gradient of error function over the weights
4. Update parameters



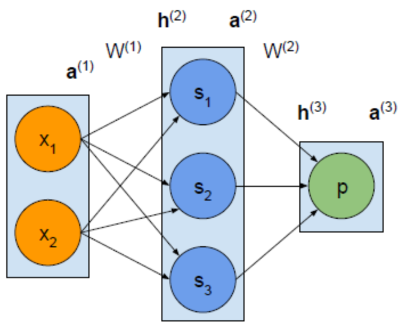
1. Get prediction
2. Calculate error
3. Calculate gradient of error function over the weights
4. Update parameters

Calculate gradient of error function



$$F = F(\mathbf{x}, W)$$
$$E = E(\mathbf{p}, \mathbf{y}) = E(F(\mathbf{x}, W), \mathbf{y})$$

Calculate gradient of error function



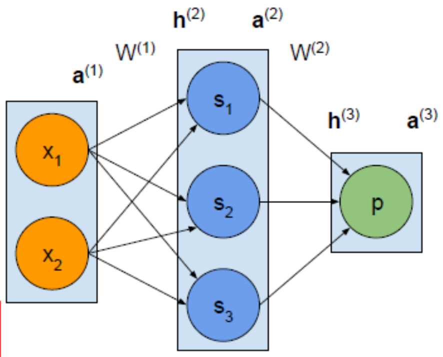
$$\frac{\partial E}{\partial W^{(2)}} = \boxed{\frac{\partial E}{\partial a^{(3)}}} \frac{\partial a^{(3)}}{\partial h^{(3)}} \frac{\partial h^{(3)}}{\partial W^{(2)}}$$

Calculate gradient of error function

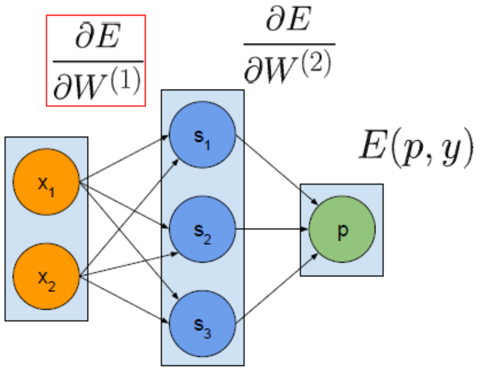
$$\frac{\partial E}{\partial W^{(1)}} = \frac{\partial E}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial h^{(2)}} \frac{\partial h^{(2)}}{\partial W^{(1)}}$$

$$\frac{\partial E}{\partial a^{(2)}} = \frac{\partial E}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial h^{(3)}} \frac{\partial h^{(3)}}{\partial a^{(2)}}$$

$$\boxed{\frac{\partial E}{\partial W^{(1)}} = \frac{\partial E}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial h^{(3)}} \frac{\partial h^{(3)}}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial h^{(2)}} \frac{\partial h^{(2)}}{\partial W^{(1)}}}$$



Backpropagation


$\boxed{\frac{\partial E}{\partial W^{(1)}}}$
$\frac{\partial E}{\partial W^{(2)}}$
$E(p, y)$
x_1
x_2
s_1
s_2
s_3
p