Fundamental concepts of neural modeling

LOT Winter School 2024

Grzegorz Chrupała

What is a model?

What are the goals of modeling?

What is the approach to modeling in CogSci vs AI?

Simplified representation of an object or process.

Reasons to model

- Understand the nature and behavior of the modeled entity by manipulating model.
  - Also, ability to predict the behavior of modeled entity.
  - Example: modeling weather and/or climate

- In AI, we often want to replicate useful behaviors. Modeling as engineering practical applications.
  - Examples:
    - Transcribing spoken language to text. Translating text. Answering questions.

Connectionism

Neural networks

Deep learning

Different names for the same basic family of approaches to modeling, based on distributed representations and gradient-based learning.

Often opposed to symbolic models (symbolic models may be probabilistic).

Several waves of popularity.

Psychological Review
Vol. 65, No. 6, 1958

THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN[1]

F. ROSENBLATT

Cornell Aeronautical Laboratory

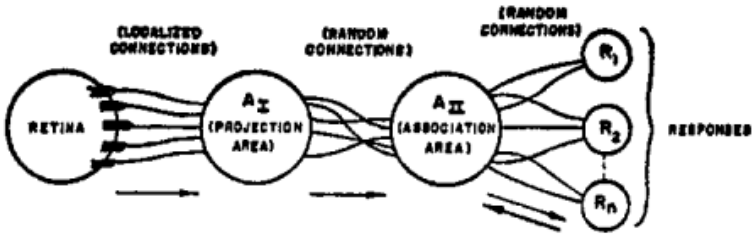
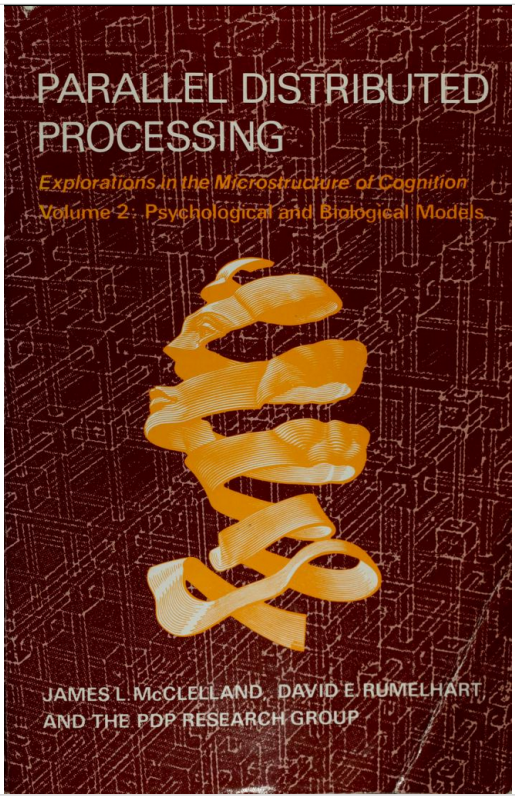


FIG. 1. Organization of a perceptron.

Rosenblatt introduced the basic idea of a computing unit inspired by a neuron. It was later shown (by Minsky & Papert 69) that this simple model cannot represent certain basic mathematical operations such as XOR.

The paradigm lost popularity.



Innovations, including hidden layers.

Repeated discovery of backpropagation – way of learning model parameters in the presence of multiple hidden layers.

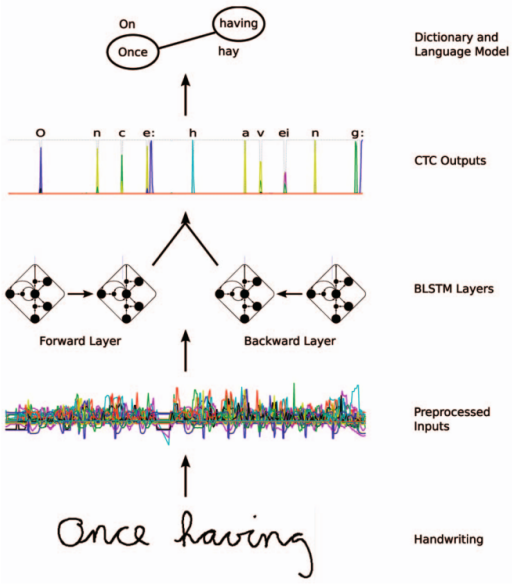


Fig. 13. The complete RNN handwriting recognition system. First, the online or offline handwriting data is preprocessed with the techniques described in Sections 2 and 3. The resulting sequence of feature vectors is scanned in opposite directions by the forward and backward BLSTM hidden layers. The BLSTM layers feed forward to the CTC output layer, which produces a probability distribution over character transcriptions. This distribution is passed to the dictionary and language model, using the token passing algorithm, to obtain the final word sequence.

IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 31, NO. 5, MAY 2009

A Novel Connectionist System for Unconstrained Handwriting Recognition

Alex Graves, Marcus Liwicki, Santiago Fernández,
Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber

Another wave of popularity started around in the second decade of 21 century, as neural models started outcompeting other approaches on benchmarks such as handwriting recognition and image classification.

The term Deep Learning gained popularity.

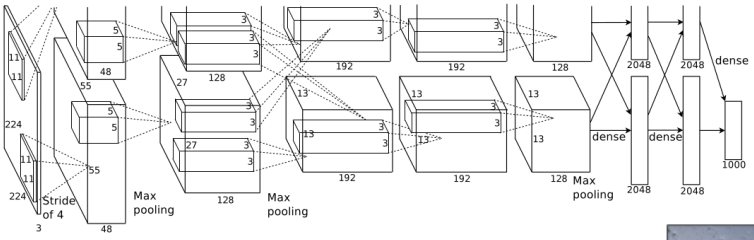


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of re between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs tl at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dim the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64, 4096–4096–1000.

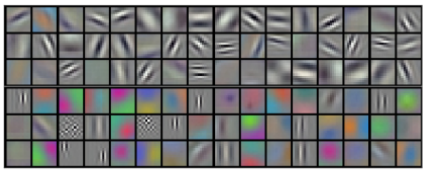


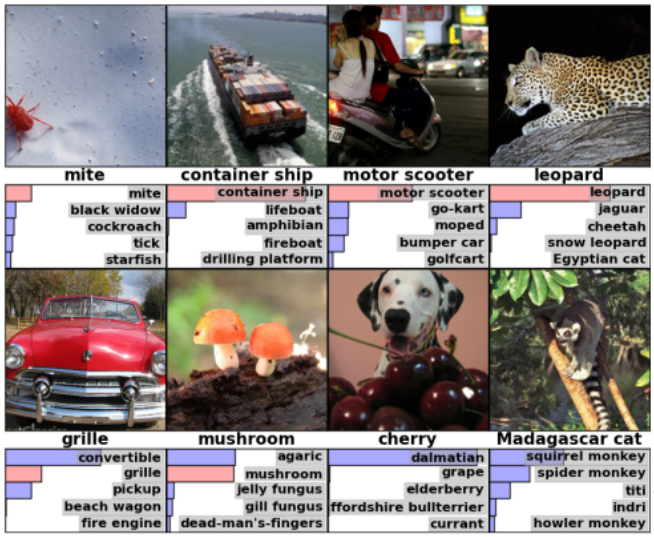
Figure 3: 96 convolutional kernels of size $11\times11\times3$ learned by the first convolutional layer on the $224\times224\times3$ input images. The top 48 kernels were learned on GPU 1 while the bottom 48 kernels were learned on GPU 2. See Section 6.1 for details.

ImageNet Classification with Deep Convolutional Neural Networks

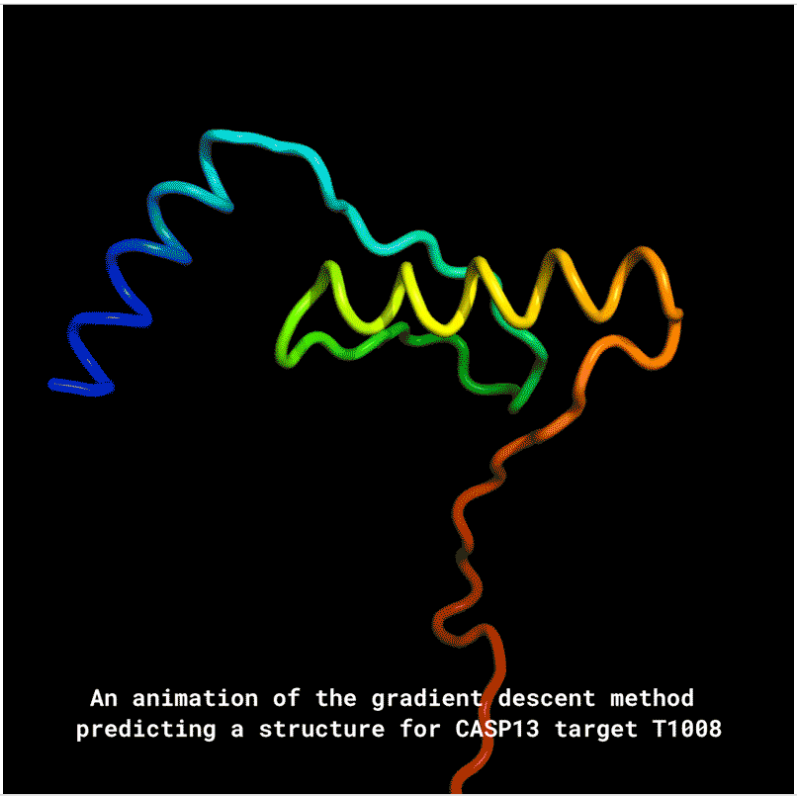
Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca



What are some successful current application of deep learning?



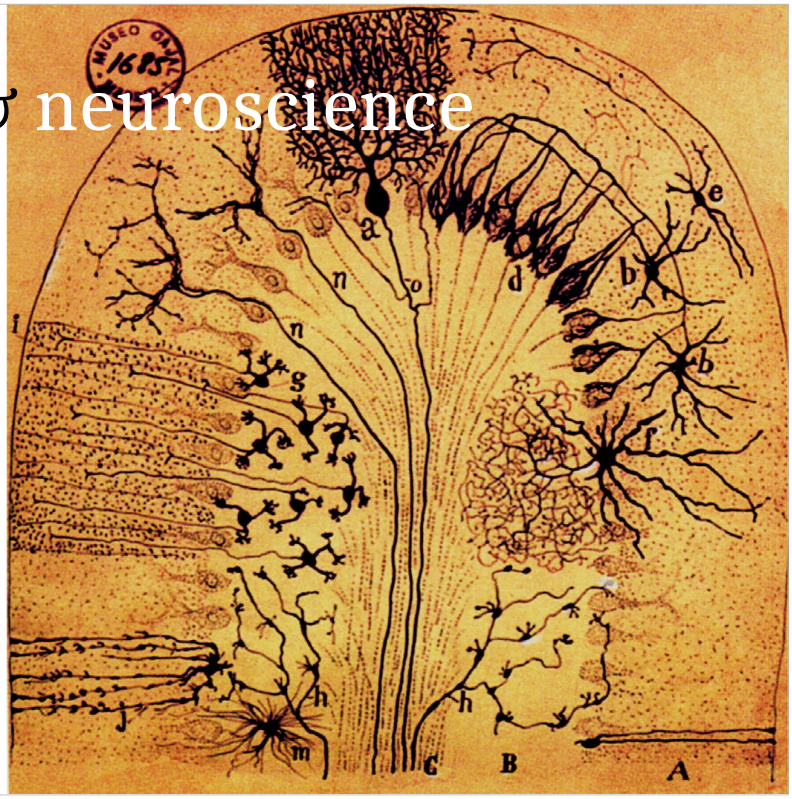
Alpha Fold: prediction of three-dimensional protein structure based the sequence of amino acids.

What are some motivations behind deep learning?

Biology & neuroscience

Humans and other animals are good at learning.

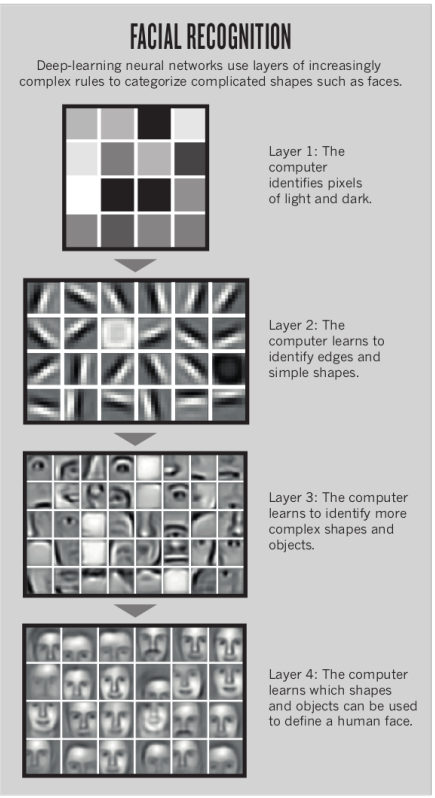
Mimicing brain and nervous system may lead to similar capabilities in machines.

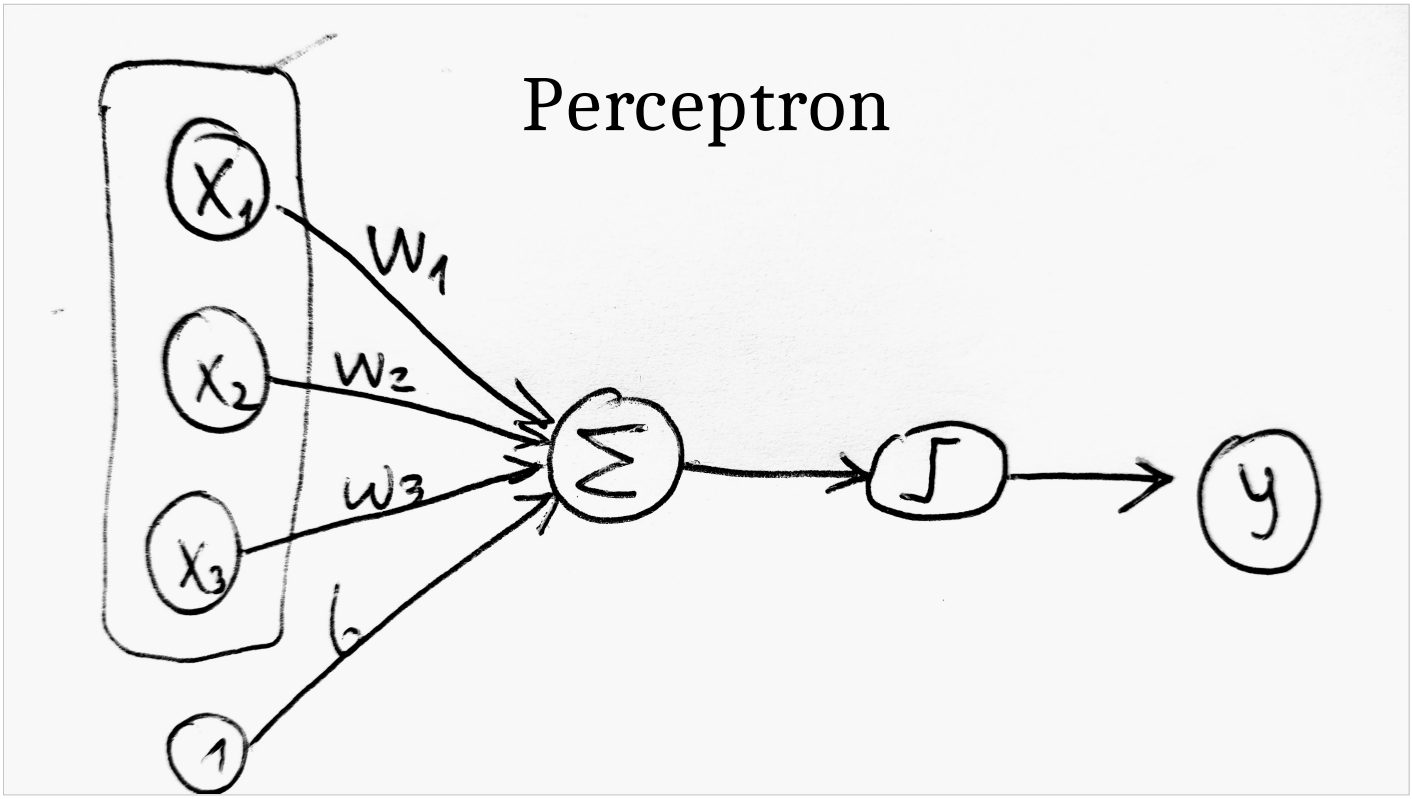


Machine learning

Classical ML doesn't work well with high-dimensional feature spaces: images, sounds, videos.

DL uses a layered architecture to learn to build a hierarchy of features.




Perceptron
x_1
x_2
x_3
1
w_1
w_2
w_3
b
Σ
f
y

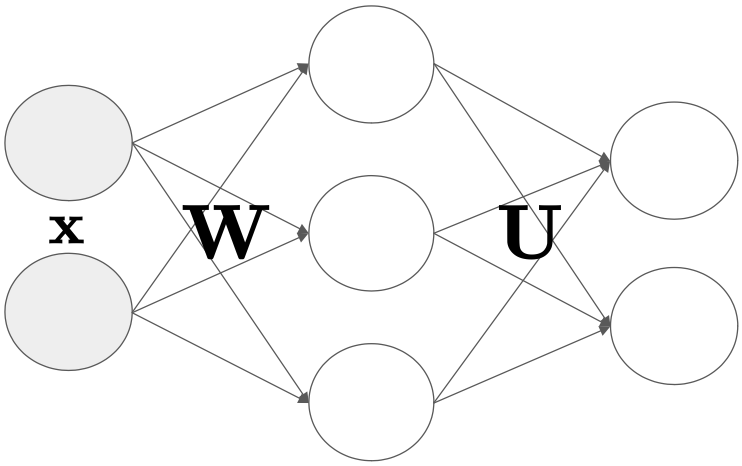
$$f(\mathbf{x}) = \left(\sum_{i=1}^{D} w_i x_i \right) + b$$

$$y = \begin{cases} +1 & \text{if } f(\mathbf{x}) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$$

$$y = \begin{cases} +1 & \text{if } f(\mathbf{x}) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

Multi-layer perceptrons



$$\mathbf{y} = \mathbf{U} f(\mathbf{W}\mathbf{x})$$

Multilayer perceptrons have multiple perceptron-like units arranged into layers.

In addition to the input layer (features) and output layer, there is at least one **hidden layer**.

In this picture the function f is an elementwise nonlinear **activation** function. Its inclusion is crucial in order for the MLP to be able to model non-linear computations.

As function composition

$$h = u \circ f \circ w$$

where

$$w(\mathbf{z}) = \mathbf{W}\mathbf{z}$$
$$u(\mathbf{z}) = \mathbf{U}\mathbf{z}$$

This network is a composition of three functions:

- Apply multiplication with matrix **W**
- Apply function f
- Apply multiplication with matrix **U**

Linearity

$$f(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}$$

In the absence of a non-linear activation function an MLP is equivalent to a linear model.

And thus has the same limitations as simple Perceptron.

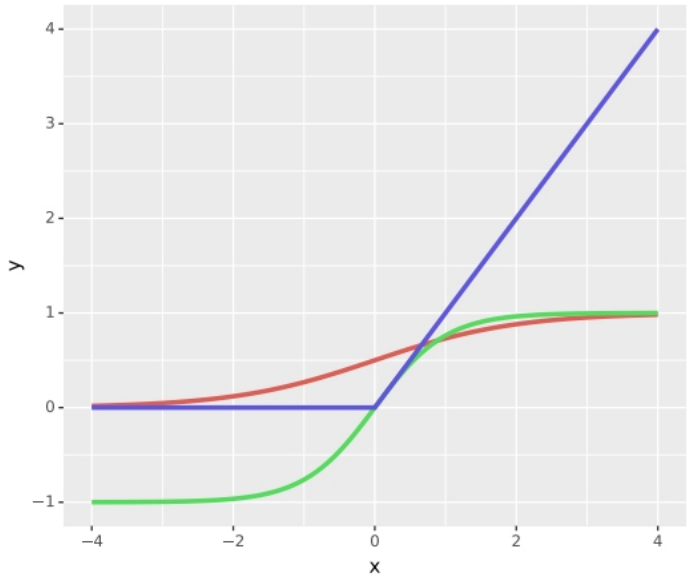
$$w(\mathbf{x}) = \mathbf{W}\mathbf{x}$$

$$u(\mathbf{x}) = \mathbf{U}\mathbf{x}$$

Composing *w* and *u* we get:

$$h(\mathbf{x}) = w(u(\mathbf{x}))$$
$$h(\mathbf{x}) = \mathbf{U}\mathbf{W}\mathbf{x}$$
$$h(\mathbf{x}) = \mathbf{Z}\mathbf{x} \text{ where } \mathbf{Z} = \mathbf{U}\mathbf{W}$$

Activation functions



$$\tanh(z) = 2 \times \text{sigmoid}(2z) - 1$$

Non-linear elementwise activation functions.

Relu is the most commonly used in the hidden layers.

Relu is piecewise linear and does not saturate.

Math. Control Signals Systems (1989) 2: 303–314

Mathematics of Control, Signals, and Systems

Approximation by Superpositions of a Sigmoidal Function*

G. Cybenko†

5. Summary

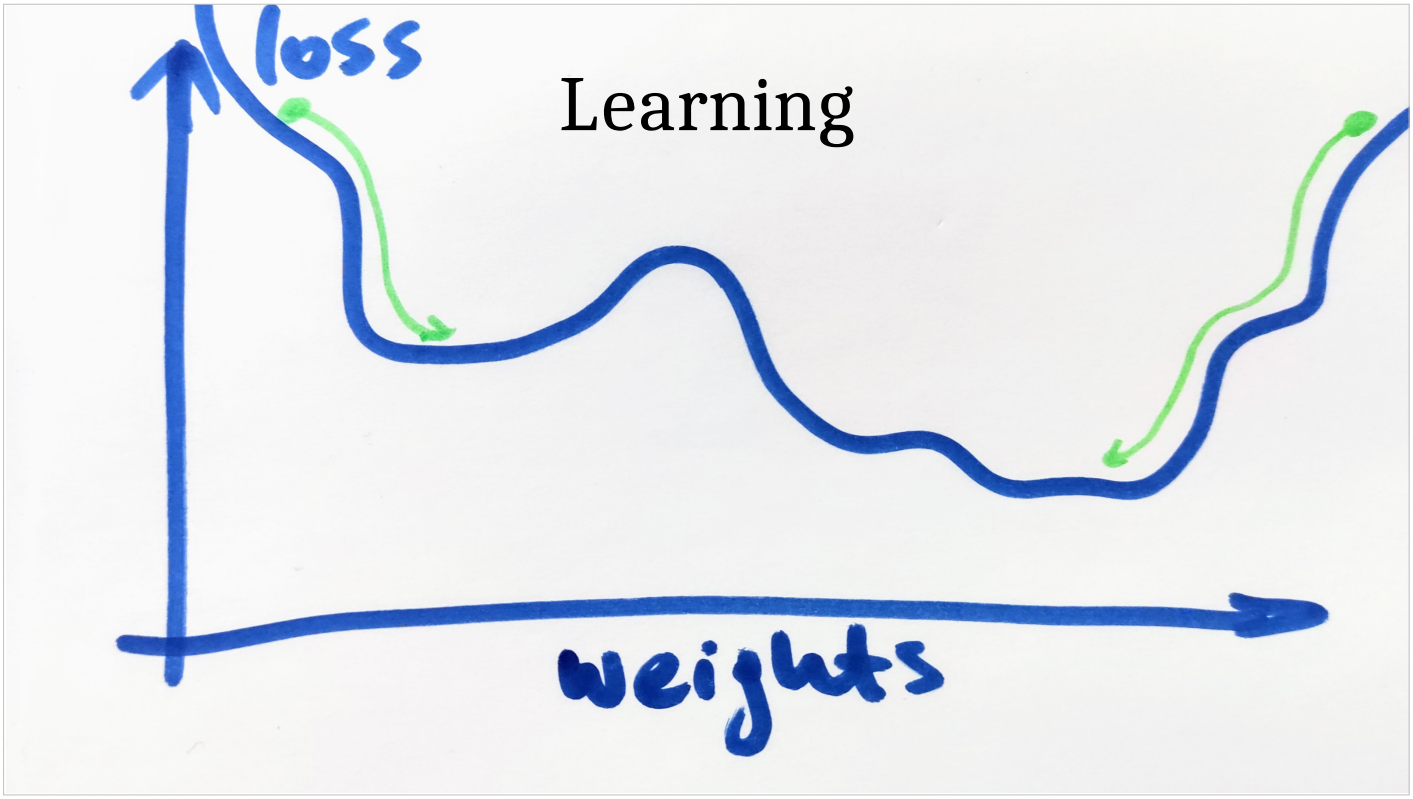
We have demonstrated that finite superpositions of a fixed, univariate function that is *discriminatory* can uniformly approximate any continuous function of n real variables with support in the unit hypercube. Continuous sigmoidal functions of the type commonly used in real-valued neural network theory are discriminatory.

This combination of results demonstrates that any continuous function can be uniformly approximated by a continuous neural network having only one internal, hidden layer and with an arbitrary continuous sigmoidal nonlinearity (Theorem 2). Theorem 3 and the subsequent discussion show in a precise way that arbitrary decision functions can be arbitrarily well approximated by a neural network with one internal layer and a continuous sigmoidal nonlinearity.

The proof concerns the ability to represent, not learn these function.

Existence proof: there is some set of weights which enable the approximation. But finding such weights may be hard.

Another issue: bounds on network width and depth needed.



Finding a good set of weights for a neural network is known as training it.

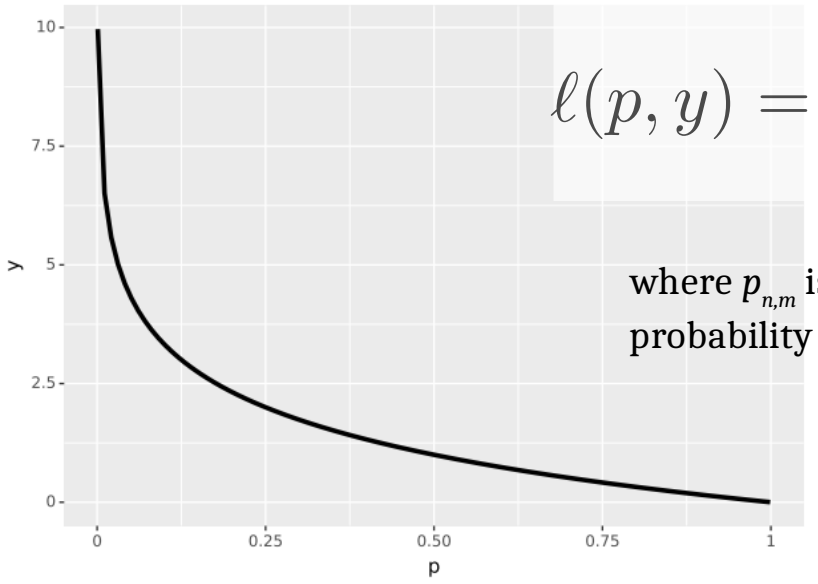
Training relies on a function which quantifies how badly the network is performing its task, as we vary the weights: the **loss** function.

The goal is to minimize the loss function.

This is done based on the **gradient** of the loss function.

The gradient of function f of several variables is the vector field (or vector-valued function) ∇f whose value at a point gives **the direction and the rate of fastest increase.** Thus the components of ∇f are partial derivatives of f.

Cross-entropy loss



$$\ell(p, y) = -\sum_{n=1}^{N} \log_2(p_{n,y_n})$$

where $p_{n,m}$ is the predicted probability of class m for example n.

There are many kinds of loss functions depending on the task. For **classification** the one commonly used is the **cross-entropy** loss.

This loss is the **negative logarithm of the probability** assigned to the correct class, aggregated over training examples.

Automatic differentiation

$$h' = (f \circ g)' = (f' \circ g) \cdot g'$$

The chain rule gives the derivative of a composition of two functions.

Automatic differentiation involves keeping track of the gradient information of complex computations, based on the repeated application of the chain rule.

Several softare toolkits implement this functionality for the purposes of training deep learning models.

In the context of neural networks training via gradient descent is also known as backpropagation.