
BSC

Introduction and Limitations of Deep Learning

ISUM 2018, MARCH 5 - 9 2018

E. Ulises Moya-Sánchez, Abraham Sánchez and Ulises Cortés
eduardo.moyasanchez@bsc.es

A little learning is a dangerous thing; drink deep, or taste not the Pierian spring: there shallow draughts intoxicate the brain, and drinking largely sobers us again.

Alexander Pope

We want to acknowledge to the creators of the images and simulations. We cite most of them, however, some of them we can't find the original source.

Marenostrum IV

1 Introduction

2 Introduction to Neural Networks

3 Hands on Session 1

1 Introduction

Abstract

This course presents an introduction of the bases and limits of Deep Learning. We will explain some basics questions such as, What is Deep Learning?, Why is deep learning so powerful and what can it be used for?. However, we present as well as some advanced topics such as some Deep Learning limitations and open problems.
Additionally, we will cover the necessary topics to learn how to apply or adapt these ideas into new solutions in the hands-on sessions. We will focus in Convolutional Neural Networks and see how them work for yourself.

Introduction

- E. Ulises Moya-Sánchez Post-Doc CONACYT at BSC, SNI Member
  1. Monte Carlo X-ray simulation
  2. Hypercomplex NN and monogenic signal [1, 2]
  3. DL for medical image aplications.
  4. Netica `http://newsnet.conacytprensa.mx/index.php/fotostock/mediaitem/8820-dr-ulises-moya-sa-nchez-1`
- Abraham Sánchez Master student Computer Science. UAG student (PNPC-CONACYT)
- Ulises Cortés. Full Profesor-UPC, PI AI-HPC group.

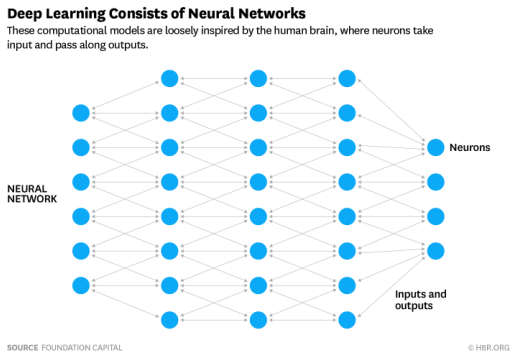
1 Introduction

Why did you hear about Deep Learning (DL)?

- Google buying DeepMind for 400 million dollars,
- Self-driving Cars, Alpha Go, Play Atari
- Toyota's billion dollar AI research investment.
- 4IR = No more jobs for humans

Introduction

- DL is about (Artificial) Neural Networks NN (Bio-inspired)
- DL is related with hierarchy of concepts.
- NN have the potential to revolutionize the AI
- Become more useful as the amount of available training data has increased



Introduction

We are so good at this that its hard to appreciate how difficult it is

- Invariance. Its one of the main difficulties in making computers perceive.
- General solution. We still don't have generally accepted solutions

Figure: Biologist: "It sure looks to me like there is a universal neural network to me."

Introduction

Definition

Deep Learning is an approach of AI, it is a type of machine learning, a technique that allows computer systems to improve with experience and data [3].

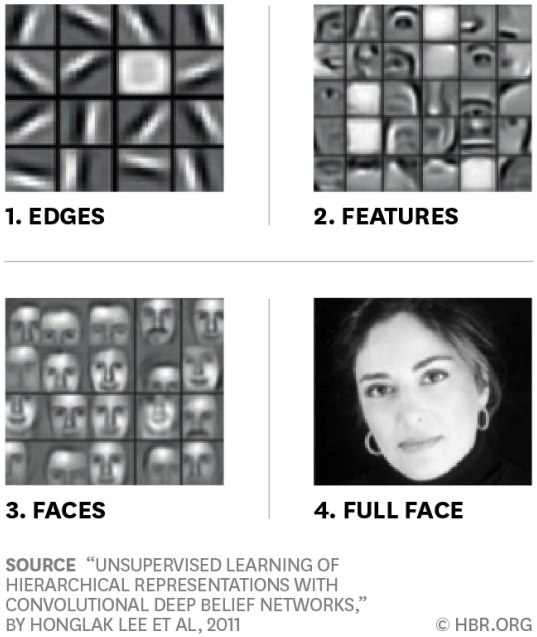
Introduction

DL, breaks complex patterns into simpler ones

pixel $->$ edge $->$ texton $->$ super-pixel $->$ part $->$ object

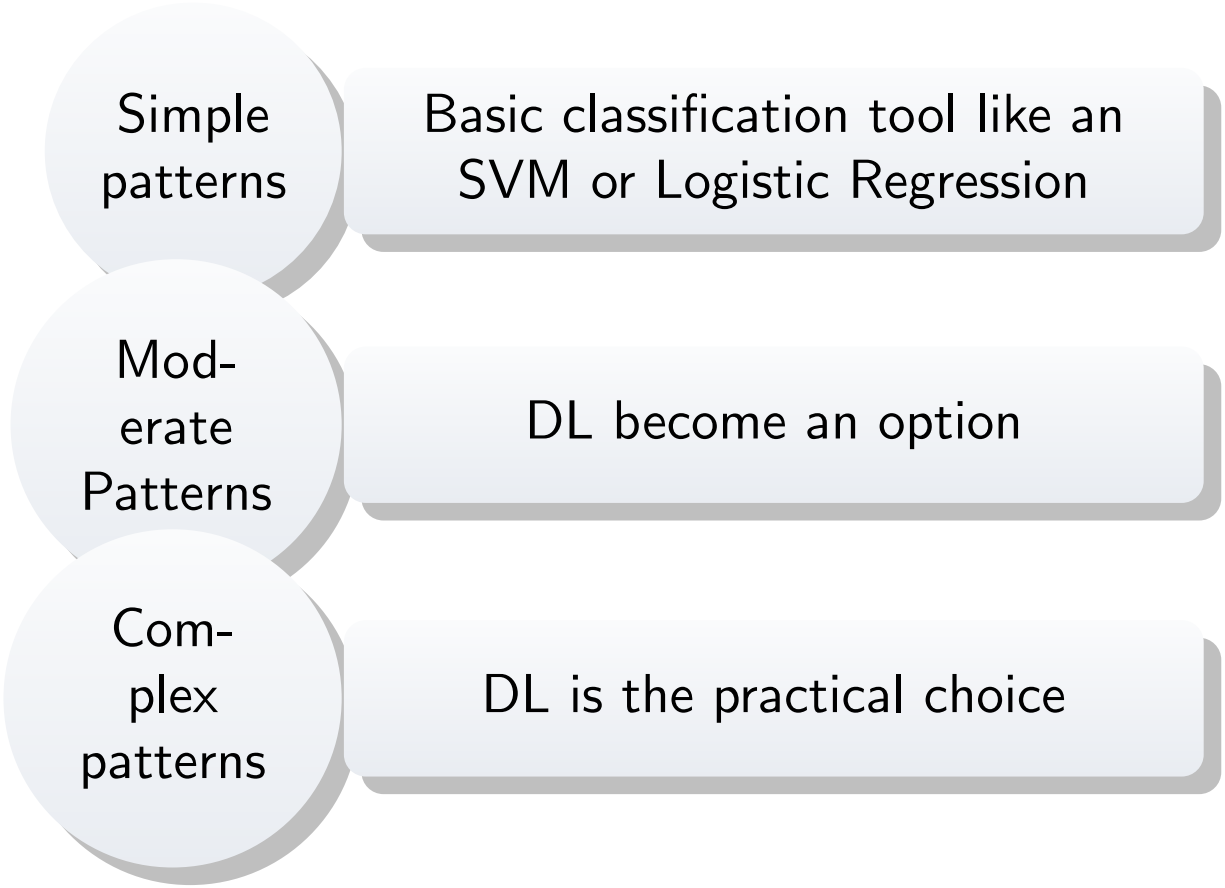
Deep Learning Relies on Multiple "Layers" of Training

In this case, the first layer recognizes edges, the second recognizes facial features like a nose or an ear, until eventually the final layer recognizes full faces.



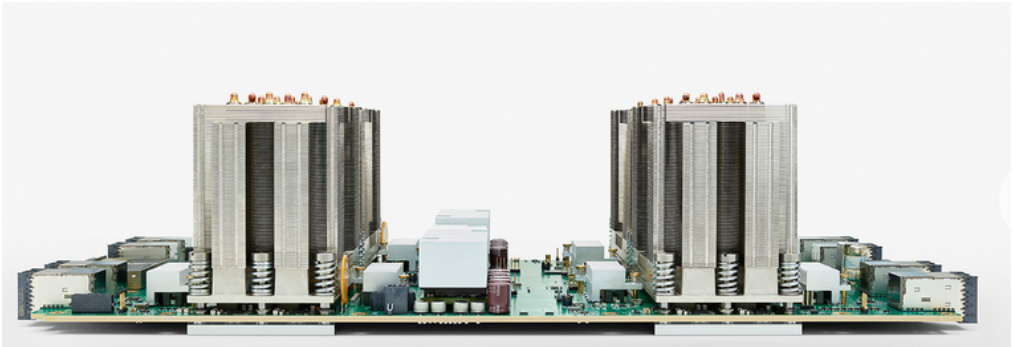
SOURCE "UNSUPERVISED LEARNING OF HIERARCHICAL REPRESENTATIONS WITH CONVOLUTIONAL DEEP BELIEF NETWORKS," BY HONGLAK LEE ET AL, 2011

Introduction



Introduction

- Downside – DL nets take much longer to train.
- Not embedded training so far.
- Good news: High Performance Computing.
- CPUs weeks, GPUs a week, TPU days, Intel-Nervana NNP ??



2 Introduction to Neural Networks

1. Introduction to Neural Networks
   - 1.1 What is a (artificial) neural network ?
   - 1.2 Supervise learning: regression and classification
   - 1.3 Perceptron: linear regression, hyperparameters
   - 1.4 Back Propagation and Vanishing gradient
   - 1.5 Hand on session 1: Perceptron and Backpropagation (Jupyter Notebook Python 3)

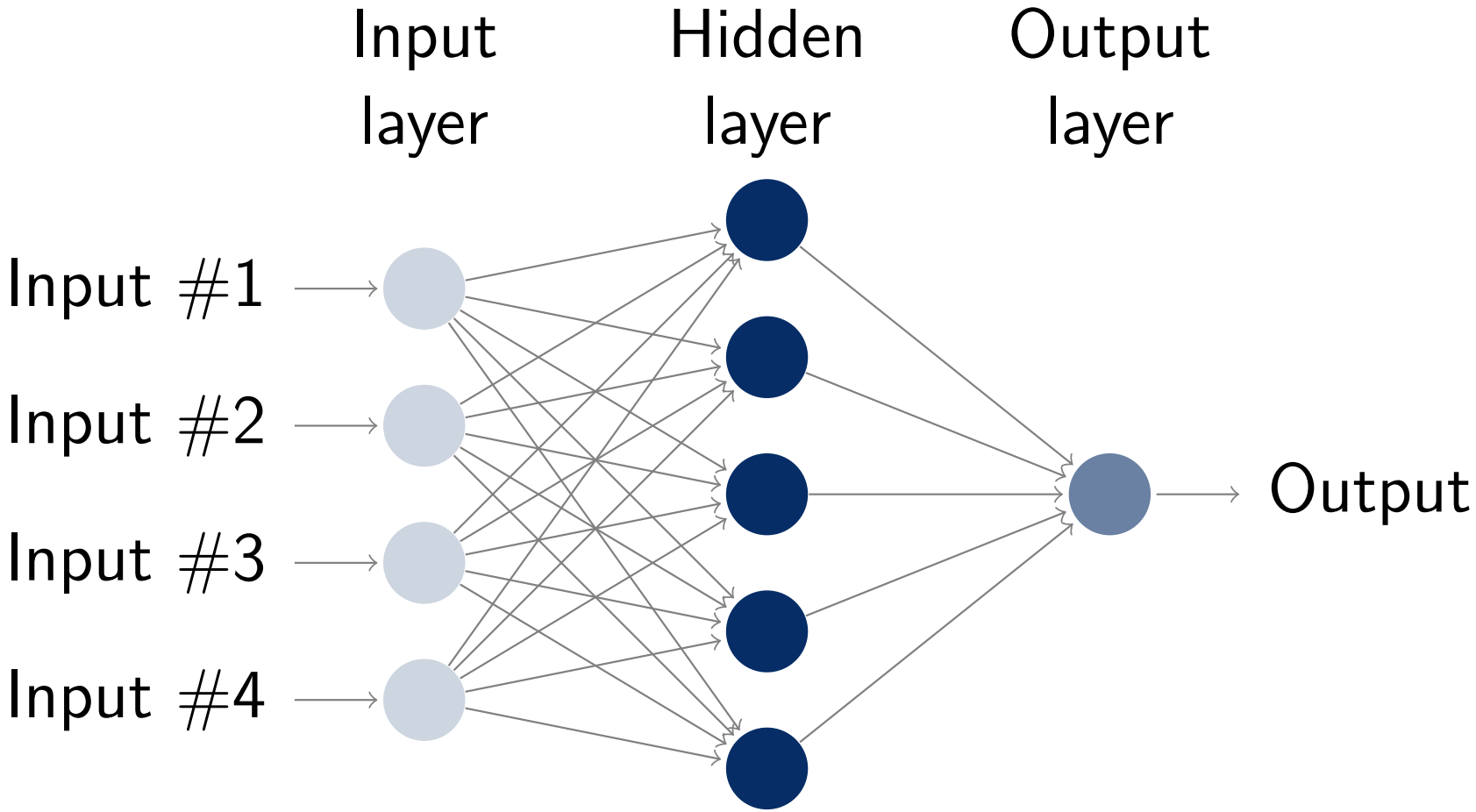
Introduction to Neural Networks

- 1^{st} work about NN McCulloch *et al* 1943
- NN main function $f()$ is to receive a set of inputs $\mathbf{x} = x_i$, perform calculations, and then get an output $\mathbf{y} = y_i$ to solve the problem.

Definition

Neural Network is a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs [4]

- NN topology:


Input layer
Hidden layer
Output layer
Input #1
Input #2
Input #3
Input #4
Output

Neural Network Applications

NN are used to do classification or regression among others

- Classification. Categorizing a group of objects, predict a discrete number of values
- Regression. Predict continuous valued output

In this course we only see (Supervised Learning) and mostly classification problems

ML types

Y. LeCun : The Next Frontier in AI: Unsupervised Learning

- **"Pure" Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Unsupervised/Predictive Learning (cake)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

Types of ML

- Supervised Learning (training set (x_i, y_i))

$$\mathbf{y} = f(\mathbf{x}), \mathbf{x} \in \mathbb{R} \tag{1}$$

  Aim, to fit a function. Regression $\mathbf{y} \in \mathbb{R}$ and Classification $\mathbf{y} \in \mathbb{Z}$

- Unsupervised Learning

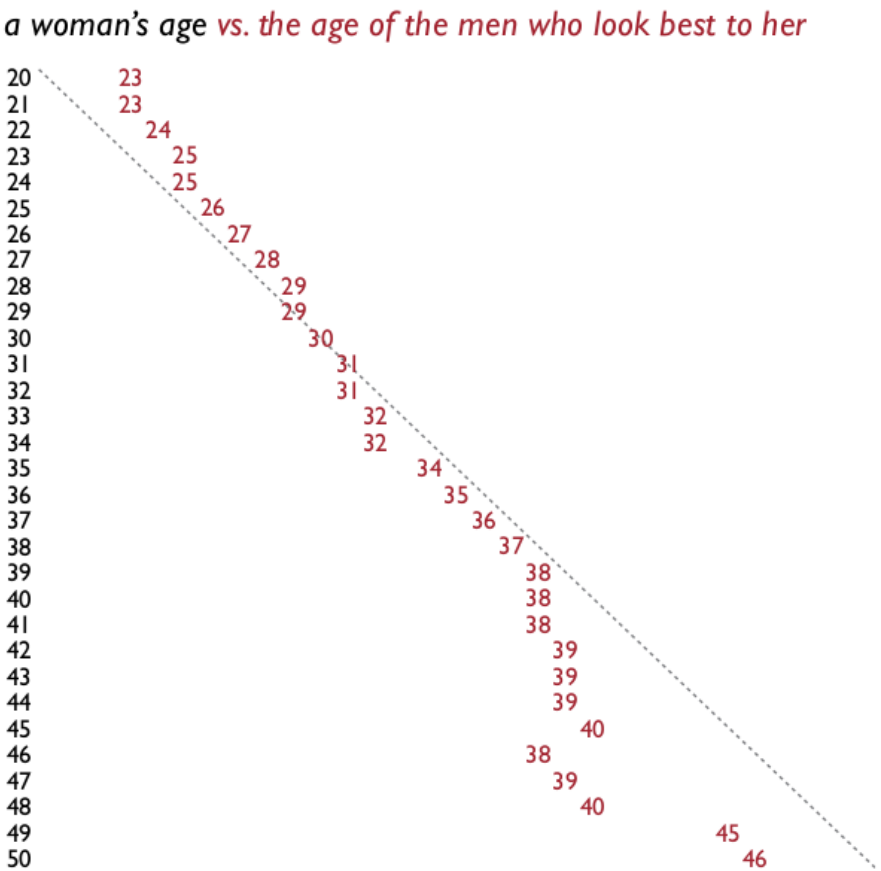
$$f(\mathbf{x}) \tag{2}$$

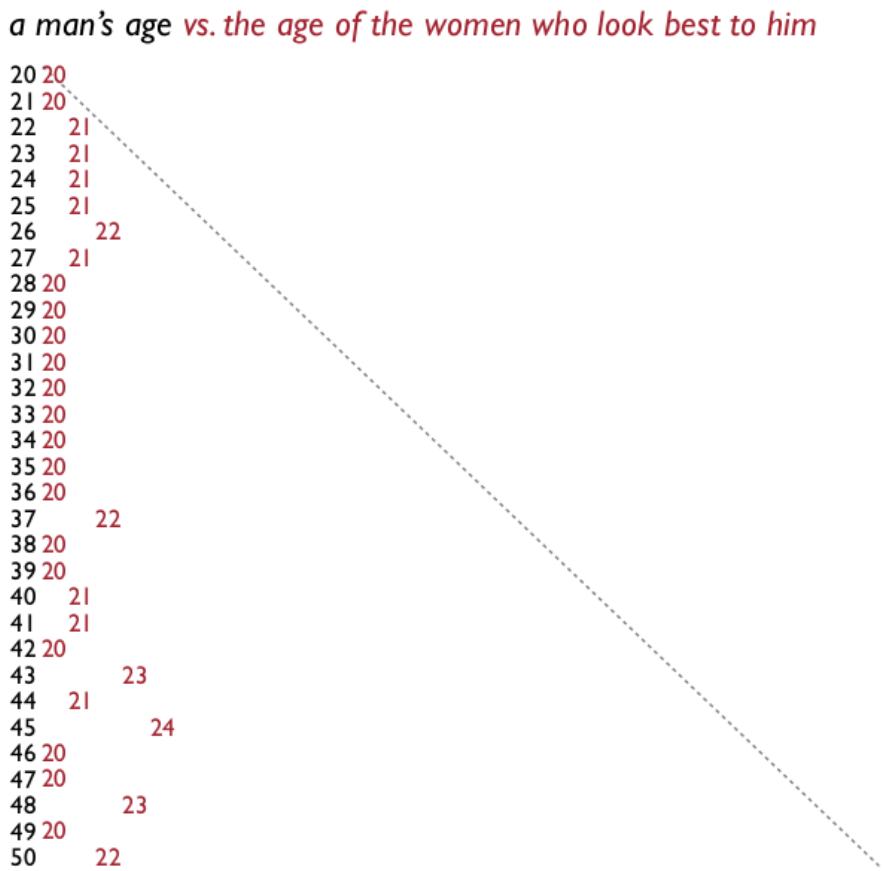
- Reinforcement Learning

$$\mathbf{y} = f(\mathbf{x}) \tag{3}$$

$$\mathbf{z} \tag{4}$$

Clasiffication vs Regression





Neurons: buildings block of NN

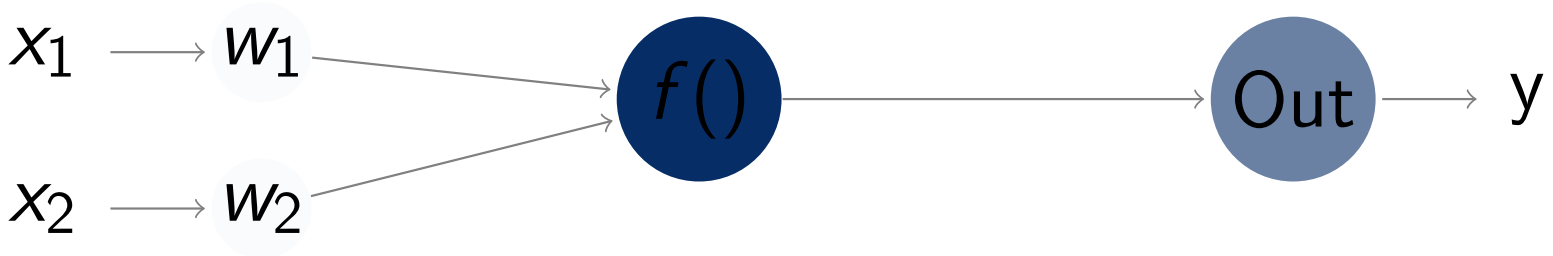
Perceptron

$$\sum_i w_i x_i =$$

$$w_1 x_1 + w_2 x_2$$

$$sgn(\sum_i w_i x_i) = \begin{cases} 1, & \text{if } \sum_i w_i x_i \geq \theta \\ 0, & \text{otherwise} \end{cases}$$

$$y = \begin{cases} 1, & class1 \\ 0, & class2 \end{cases}$$



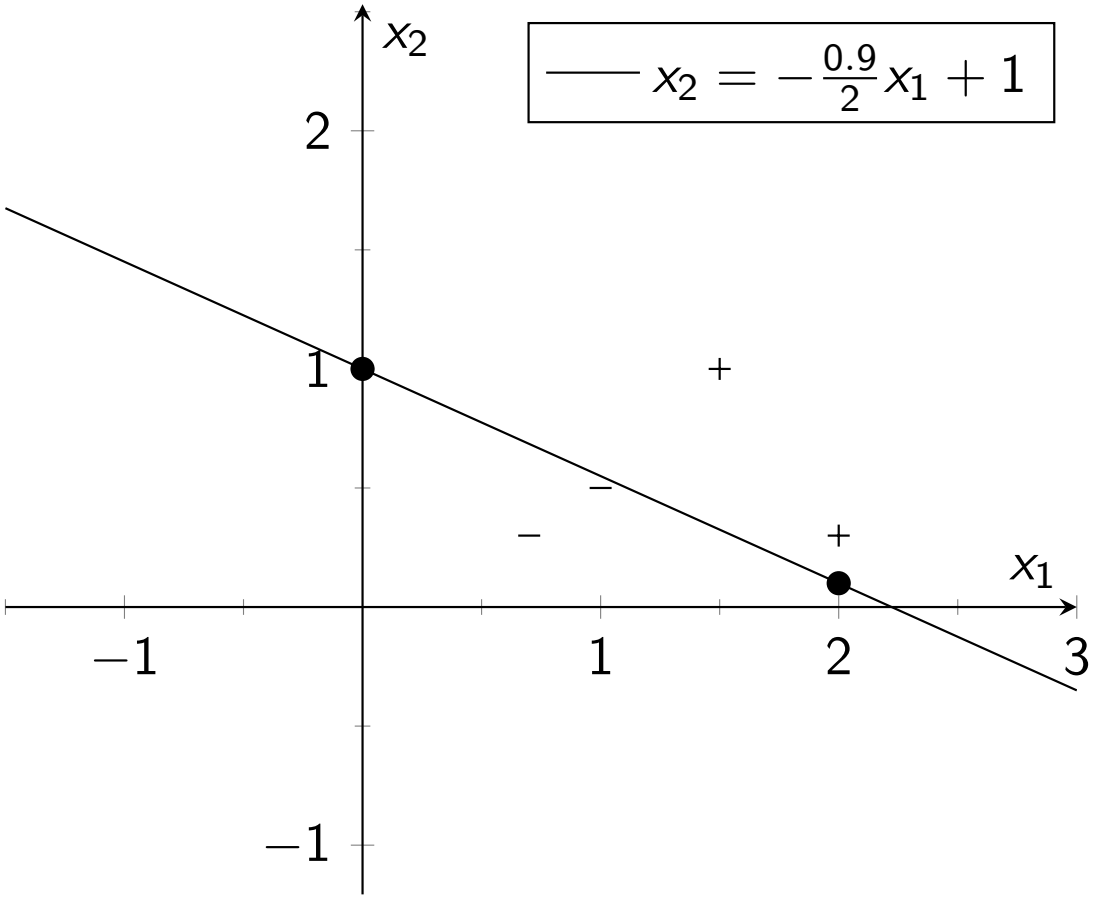
Geometrical meaning in 2-D the decision boundaries are always straight lines.

$$x_2 = -\frac{w_1}{w_2} x_1 + \frac{\theta}{w_2} \tag{5}$$

Example forward prop: Let $w_1 = 0.9, w_2 = 2$ and $\theta = 2$. then $\rightarrow$

$$\begin{cases} 1, & \text{if } 0.9x_1 + 2x_2 \geq 2 \\ 0, & \text{otherwise} \end{cases} \rightarrow x_2 = -\frac{0.9}{2}x_1 + 1, \text{ for } \mathbf{x} = (x_1 = 1.5, x_2 = 1)$$

$0.9(1.5) + 2(1) \geq 1$ then $\in$ class $+$



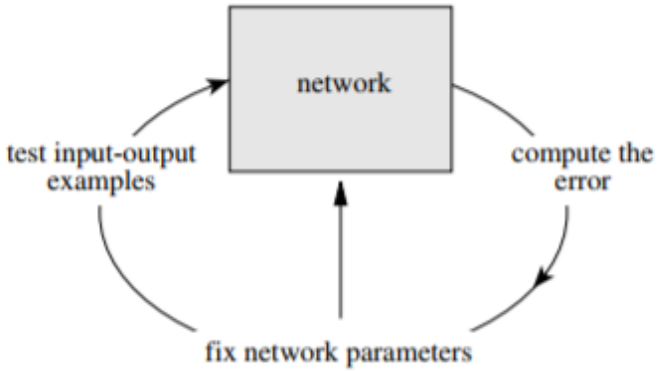
Perceptron

Definition

Perceptron is a computing unit with threshold θ which, when receiving the n real inputs $x_1, x_2, ..., x_n$ through edges with the associated weights $w_1, w_2, ..., w_n$, outputs 1, if the inequality $\sum_i w_i x_i \geq \theta$, holds and otherwise 0 [5].

Perceptron Learning/Training Algorithm

How to find the parameters (w_i) adequate for a given task? Learning algorithm is a closed loop with examples to generate corrections to the network parameters executed iteratively until the network learns to produce the desired response, Fig from [5].



Perceptron Learning Method

The learning algorithm must minimize perceptron error function δ.

- Initialize weights at random.
- Compute output **y** (forward prop)
- Compute error $\delta = (y_{target} - y_i)$
- Update weights, using the learning perceptron learning rate η

$$w_{n+1} = w_n + \eta \delta x \tag{6}$$

- Repeat the last step until convergence (i.e. error δ is zero)

Example

Create a Perceptron that will produce a **'1'** when the inputs are both **'1'**, and a **'zero'** otherwise. This problem represents the **AND** logical gate.
Input X and target t:

$$X = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix} \qquad t = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Initial weights W and b:

$$W = \begin{pmatrix} 0.038 & 0.21 \end{pmatrix} \qquad \theta = -0.32$$

Learning rate $\eta = 0.05$

Epoch 1

Iteration (1):

$$y(1) = step(X(1)W^T(1)+\theta(1)) = \begin{pmatrix}0 & 0\end{pmatrix}\begin{pmatrix}0.038\\0.21\end{pmatrix}-0.32 = step(-0.32) = 0$$

Expected 0. Correct classification!, keep weights.

Iteration (2):

$$y(2) = step(X(2)W^T(2)+\theta(2)) = \begin{pmatrix}0 & 1\end{pmatrix}\begin{pmatrix}0.038\\0.21\end{pmatrix}-0.32 = step(-0.11) = 0$$

Expected 0. Correct classification!, keep weights.

Iteration (3):

$$y(3) = step(X(3)W^T(3)+\theta(3)) = \begin{pmatrix}1 & 0\end{pmatrix}\begin{pmatrix}0.038\\0.21\end{pmatrix}-0.32 = step(-0.282) = 0$$

Iteration (4):

$$y(4) = step(X(4)W^T(4)+\theta(4)) = \begin{pmatrix}1 & 1\end{pmatrix}\begin{pmatrix}0.038\\0.21\end{pmatrix}-0.32 = step(-0.072) = 0$$

Expected 1. Incorrect classification, updating weights:

$$W(5) = W(4)+\eta(t(4)-y(4))X(4) = \begin{pmatrix}0.038\\0.21\end{pmatrix}+0.05(1-0)\begin{pmatrix}1 & 1\end{pmatrix} = \begin{pmatrix}0.088\\0.26\end{pmatrix}$$

$$\theta(5) = \theta(4) + \eta(t(4) - y(4)) = -0.32 + 0.05(1 - 0) = -0.27$$

Epoch 2

Iteration (5):

$$y(5) = step(X(5)W^T(5)+\theta(5)) = \begin{pmatrix}0 & 0\end{pmatrix}\begin{pmatrix}0.088\\0.26\end{pmatrix}-0.27 = step(-0.27) = 0$$

Expected 0. Correct classification!, keep weights.

Iteration (6):

$$y(6) = step(X(6)W^T(6)+\theta(6)) = \begin{pmatrix}0 & 1\end{pmatrix}\begin{pmatrix}0.088\\0.26\end{pmatrix}-0.27 = step(-0.01) = 0$$

Expected 0. Correct classification!, keep weights.

Iteration (7):

$$y(7) = step(X(7)W^T(7)+\theta(7)) = \begin{pmatrix}1 & 0\end{pmatrix}\begin{pmatrix}0.088\\0.26\end{pmatrix}-0.27 = step(-0.182) = 0$$

Expected 0. Correct classification!, keep weights.

Iteration (8):

$$y(8) = step(X(8)W^T(8)+\theta(8)) = \begin{pmatrix}1 & 1\end{pmatrix}\begin{pmatrix}0.088\\0.26\end{pmatrix} - 0.27 = step(0.078) = 1$$

Expected 1. Correct classification!, keep weights.

$$y = \begin{pmatrix}0\\0\\0\\1\end{pmatrix} \quad = \quad t = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

Now the output is equals to the target!!

Finally...weights are:

$$W = \begin{pmatrix} 0.088 & 0.26 \end{pmatrix}$$

$$\theta = -0.27$$

With values above we can create a line which will allow us to classify. In order to graph the line, we can use the general form of the linear equation which is written as:

$$A_x + B_y + C = 0$$

Where A and B are the weights (w) and C is the bias (θ). Then...

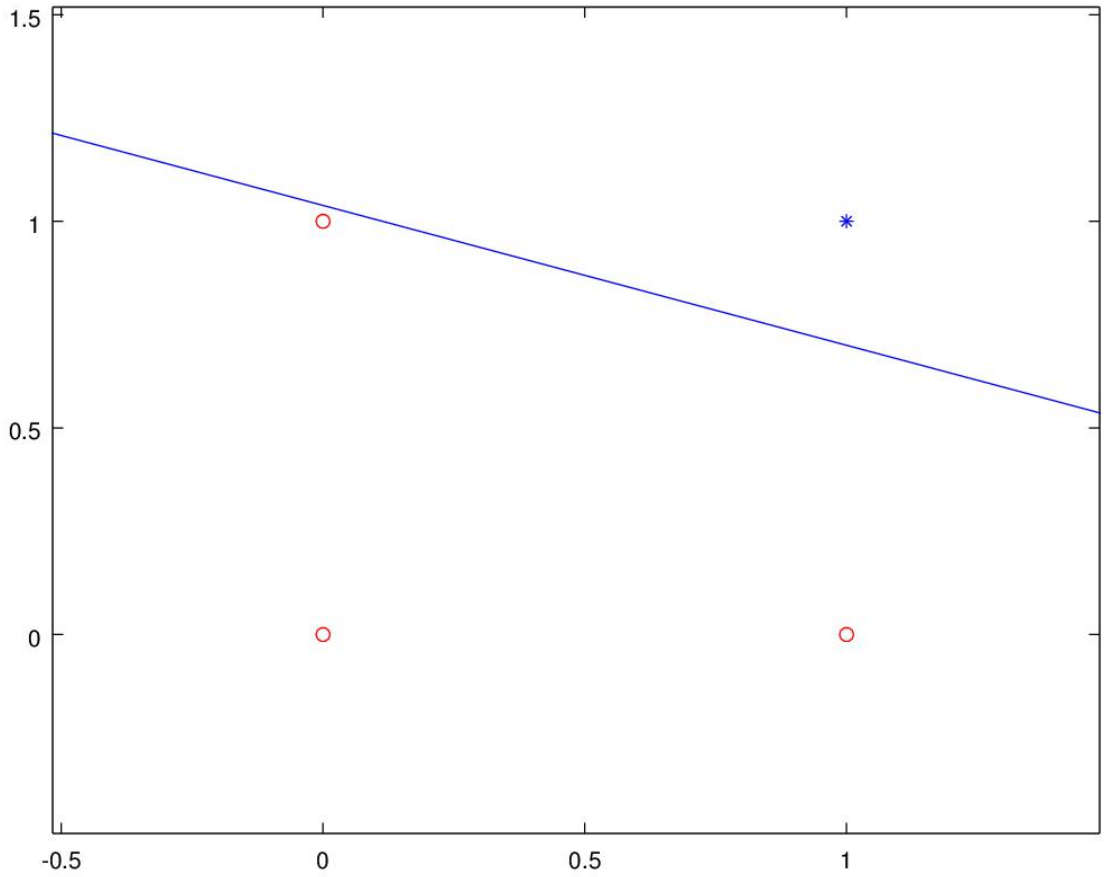
$$w_1 x + w_2 y + \theta = 0 \rightarrow 0.088x + 0.26y - 0.27 = 0$$

We are looking for y variable, so:

$$\begin{aligned} 0.088x + 0.26y - 0.27 &= 0 \\ 0.088x + 0.26y &= 0.27 \\ 0.26y &= 0.27 - 0.088x \\ y &= \frac{0.27 - 0.088x}{0.26} \end{aligned}$$

Now, we can just assign a range of values for x, for example (-2.5, 2.5).

$$0.088x + 0.26y - 0.27 = 0$$



MultiLayerPerceptron (MLP) Versus Perceptron

Multilayer

- 1 or more hidden units
- More complex patterns

Perceptron

- very limited in what they can represent.
- learning problem much simpler

NN Supervised Training Steps

Backpropagation [6]

1. Generate a training pair or pattern $(\mathbf{x}, y_{target})$ and random initialization.
2. Forward prop, **x** and allow it to generate an output **y**
3. Compare y with y_{target} to compute the error
4. Adjust weights, w_i, using the gradient (from right to left), which measures the rate at which the error will change with respect to a change in a weight or a bias
5. Repeat 2-4 until have the expected behavior

Gradient at any point is the product of the previous gradients up to that point.Vanishing gradient: the early layers of the network are the slowest to train. As a result backprop taking a lot of time to train, and the accuracy is often very low.

Gradient Descent

$$y_j^l = f(\sum_i w_{ij}^l x_i^{l-1} + \theta_j^l) \quad (7)$$

$$E = \frac{1}{2}\sum_d (t_d - y_d)^2 \quad (8)$$

$$\nabla E = \frac{\partial E}{\partial w_i}, i = 1, .., n \quad (9)$$

$$\Delta w = -\eta \frac{\partial E}{\partial w_i} \quad (10)$$

Click this link above
http://www.emergentmind.com/neural-network
Backprop example
https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Neural Networks Summary so far

- NN can be viewed as the result of spinning classifiers in a layered web. Each neuron in the hidden and output layers has its own classifier (If Rule approximation).
- Forward prop is a NN way of classifying a set of inputs and computes the error.
- Backward pass computes gradients inputs at each layer and parameters. The process of improving a NN accuracy is training (learning) by adjusting the weights w_i and biases θ_i.
- Why create and train a web of classifiers, when an individual classifier can do the job quite well? Due to problem of pattern complexity. NN is a chain of non-linear operations, implementing highly non-linear functions

NN Hyperparameters

- Variables set, before optimizing the model's parameters
- Structure of the network (Number of Hidden Units, Weight Decay, Activation function)
- Learning rate
- Loss function (error)
- Mini-batch Size
- Number of Training Iterations (Epocs)

https://developers.google.com/machine-learning/glossary/

3 Hands on Session 1

Objective

1. Perceptron implementation
2. AND gate classification
3. Many points classification
4. MLP XOR gate classification
5. MLP Letters classification

Jupyter `http://jupyter.readthedocs.io/en/latest/install.html`
`https://github.com/asp1420/c1.git`
jupyter notebook ann_en.ipynb

Literature Credit

Other courses
`https://upc-mai-dl.github.io/mlp-convnets-theory`
`http://dlai.deeplearning.barcelona/`
`https://deeplearning4j.org`


E. U. Moya-Sánchez and E. Vázquez-Santacruz, "A geometric bio-inspired model for recognition of low-level structures," in *International Conference on Artificial Neural Networks*, pp. 429–436, Springer, 2011.


E. Bayro-Corrochano, E. Vazquez-Santacruz, E. Moya-Sanchez, and E. Castillo-Muñis, "Geometric bioinspired networks for recognition of 2-d and 3-d low-level structures and transformations," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 10, pp. 2020–2034, 2016.


I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*.
MIT Press, 2016.
`http://www.deeplearningbook.org`.


M. Caudill, "Neural networks primer, part i," *AI expert*, vol. 2, no. 12, pp. 46–52, 1987.


R. Rojas, *Neural networks: a systematic introduction*.
Springer Science & Business Media, 2013.


Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*, pp. 9–50, Springer, 1998.


BSC