Deep Learning in Computer Vision

Alexei Manso Corrêa Machado
Pontifical Catholic University of Minas Gerais – D. Computer Science

2/2020

Unsupervised Learning

Steps:

- Choose a training set. The class of each element of the set is unknown
- Choose a set of features and the number of classes
- Choose a method for grouping elements
- Determine the parameters from the discovered classes
- Test with objects outside the training set.

Supervised Learning

Steps:

- Choose a training set. The class of each element of the set is known.
- Choose discriminating characteristics.
- Choose a decision method / function.
- Determine the parameters from the training set.
- Test with objects outside the training set.

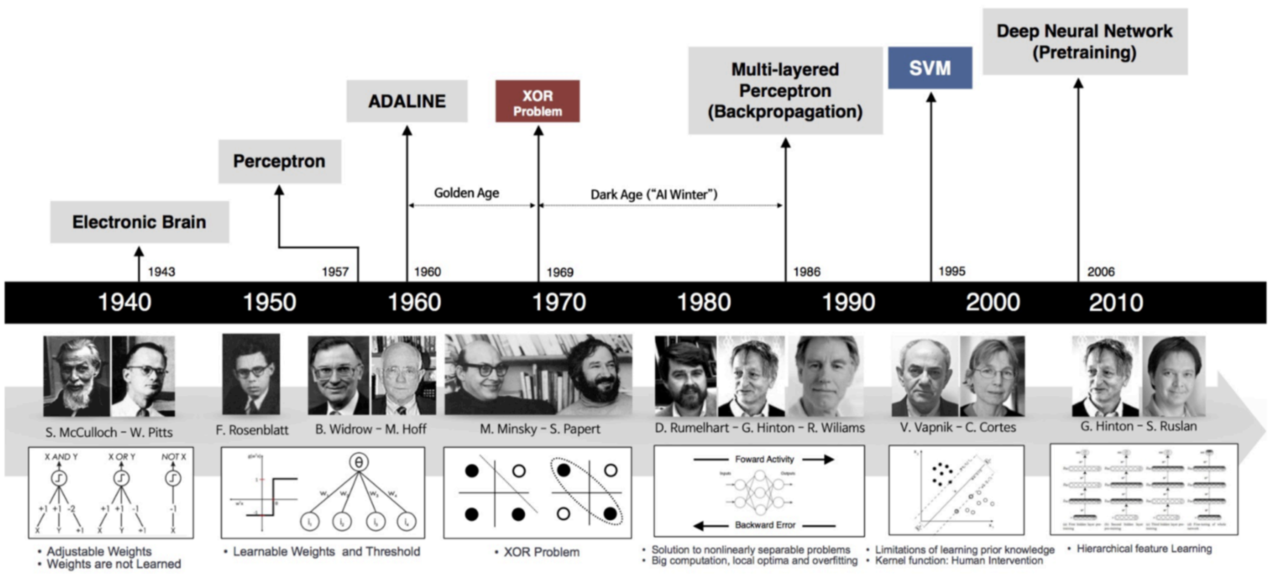
Supervised Learning

Bayes Rule, the optimum decision function:

$$P(y \mid x_1..x_n) = P(x_1..x_n \mid y)P(y) / P(x_1..x_n)$$

- Minimum classification error
- The PDFs are however hard to determine in most cases
- Machine learning algorithms such as SVM, decision trees and neural networks try to approximate Bayes rule

Milestones

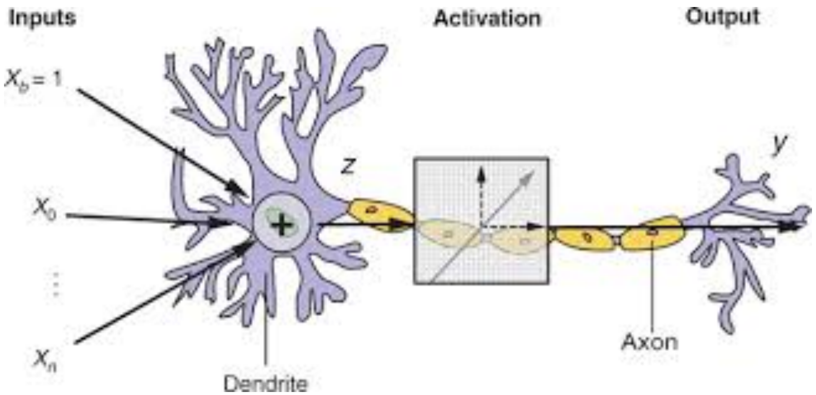


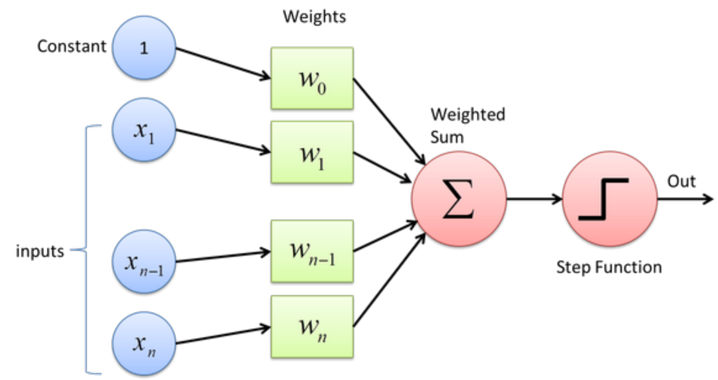
Deep Neural Network (Pretraining)
Multi-layered Perceptron (Backpropagation)
SVM
ADALINE
XOR Problem
Perceptron
Electronic Brain
Golden Age
Dark Age ("AI Winter")
1943
1957
1960
1969
1986
1995
2006
1940
1950
1960
1970
1980
1990
2000
2010
S. McCulloch - W. Pitts
F. Rosenblatt
B. Widrow - M. Hoff
M. Minsky - S. Papert
D. Rumelhart - G. Hinton - R. Wiliams
V. Vapnik - C. Cortes
G. Hinton - S. Ruslan
X AND Y
X OR Y
NOT X
Foward Activity
Backward Error
• Adjustable Weights
• Weights are not Learned
• Learnable Weights and Threshold
• XOR Problem
• Solution to nonlinearly separable problems
• Big computation, local optima and overfitting
• Limitations of learning prior knowledge
• Kernel function: Human Intervention
• Hierarchical feature Learning

Feature selection

- Classical AI and Machine Learning algorithms requires the selection of features from an infinite set
- This is a problem of exponential complexity
- Heuristic solutions imply deeper knowledge of the problem domain
- Features should be uncorrelated
- Too many features can be a confounding aspect
- The “curse of dimensionality”
- Many classifiers require that the number of features be less than the number of training data
- DL algorithms are less sensitive to many of these issues

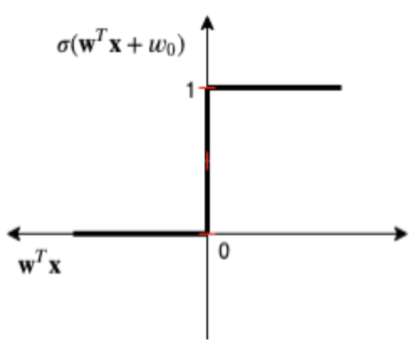
The Perceptron (Rosenblatt 1958)





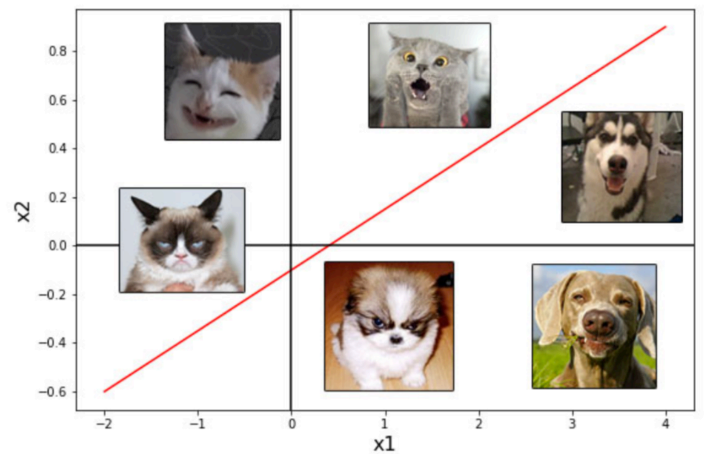
$$z = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n = b + \mathrm{w}^T \mathrm{x}$$

$$a = \hat{y} = \sigma(z)$$



The Perceptron (Rosenblatt 1958)

- The perceptron is a binary classifier: y=0 if z>0 otherwise y=1
- The perceptron defines a linear decision function



- Example with 2 variables: the OR operator
- b= -1 (bias) $$z = -1 + 2x_1 + 3x_2$$

The Perceptron (Rosenblatt 1958)

- Solutions also can be found for the AND and NOT operator
- What about XOR?

The Perceptron (Rosenblatt 1958)

- How do we find a feasible solution?

1. Initialize W
2. Repeat until W is stable (convergence)

2.1. For each sample (x,*y*) in the dataset

2.1.1 Compute $\hat{y} = \sigma(b+w^T x)$

2.1.2. If $\hat{y} \neq y$ then adjust each weight w_i so that $\hat{y}$ *gets closer to y*

- I.e. We backpropagate the output error in order to get a better estimate of the weights
- If the classes are linearly separable, the algorithm will converge, otherwise it will not!

The Perceptron (Rosenblatt 1958)

Remarks:

- A Sum of Squared error fuction can be used to evaluate convergence (Be careful to overfitting!)
- If the classes are linearly separable, the algorithm will converge, otherwise it will not! (Duda, Hart e Stork)
- The algorithm finds ANY solution that makes it converge. The SVM is an evolution of the perceptron that finds a decision function with maximum separability (Krauth e Mezard, 1987)
- If some input variable is useless its weight should have a small magnitude
- If we increase the number of input variables, linear separabiity may be achieved (not always, of course!)

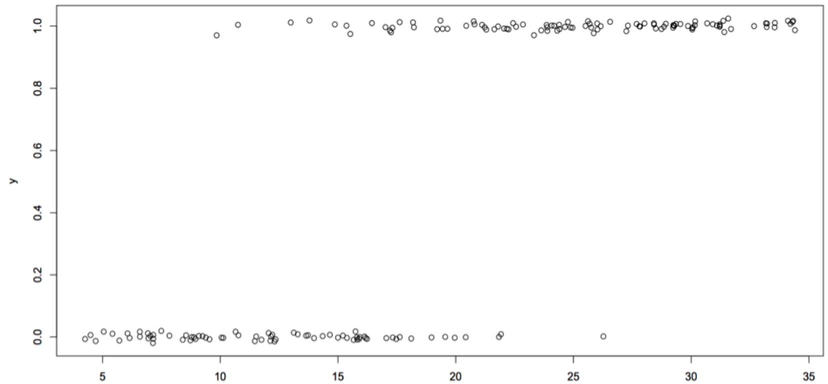
The Perceptron (Rosenblatt 1958)

Remarks:

- In 1969 Minsky and Pappert showed the weakness of the Perceptron to solve the XOR problem
- The research on Connectionism slowed down
- The perceptron model then receives 2 modifications: a different activation function and additional layers.

Logistic regression

- In many cases, the behavior of a variable does not change drastically from one class to the other

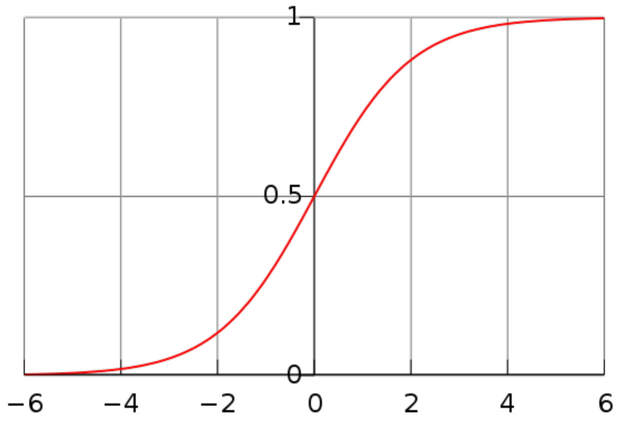


- Therefore we want the output of the classifier to give the probability of the class instead of being a 0/1
- The probability however is not properly defined by a linear function!

Logistic regression

We model the log of the odds ratio as a linear function, from which we get the sigmoid ativation function:

$$\ln \frac{P(y=1)}{1-P(y=1)} = b + \mathrm{w}^T \mathrm{x}$$



Logistic regression

- How to adapt the perceptron's learning algorithm to the new activation function?
  - We need to properly update the weights
  - We need to compute a loss function and know when to stop iterating
- For the logistic activation function, the SSE loss function does not work well
- Since P(y=1|x)=y^ and P(y=0|x)=1-y^ ,we minimize the *cross-entropy loss function:*

$$L(y,\hat{y}) = -\ln P(y\,|\,\mathrm{x}) = -\ln(\hat{y}^{y}(1-\hat{y})^{1-y}) = -(y\ln\hat{y} + (1-y)\ln(1-\hat{y}))$$

$$J(\mathrm{w},b) = \tfrac{1}{m}\sum_{i=1}^{m} L(y^{(i)},\hat{y}^{(i)})$$

A Bit of Calculus

Derivatives:

- Slope of the tangent of a function at a given value
- Variation rate of a function

A Bit of Calculus

Chain rule:

- Derivative of a composite function:

$$\text{If } F(x)=f(g(x)),\ F'(x)=f\,'(g(x))g'(x)$$

If we call $u=g(x)$, $dF/dx = dF/du * du/dx$

A Bit of Calculus

Partial derivative:

- Derivative of a multivariated function with respect to each of the variables separately, the others being cosidered constant

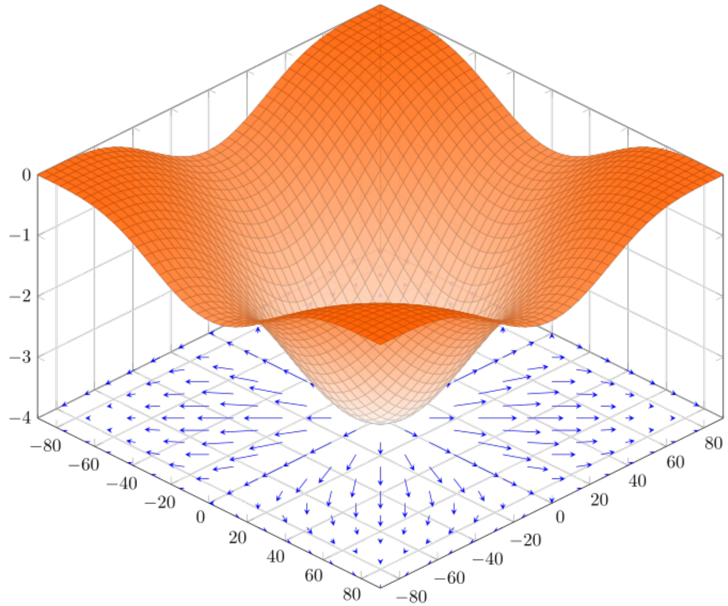
$$\frac{\partial}{\partial x_i} f(x_1,\ldots,x_n)$$

A Bit of Calculus

Gradient:

- Vector of partial derivatives that indicates the direction of maximum variation

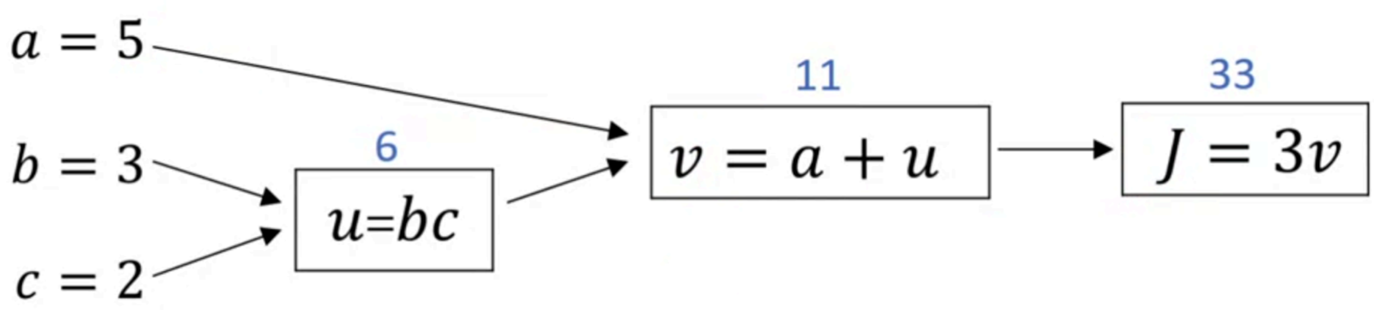
$$\nabla f(x_1,\ldots,x_n) = [\partial f(x_1,\ldots,x_n)/\partial x_1 \ldots, \partial f(x_1,\ldots,x_n)/\partial x_n]^T$$



En.wikipedia.org

Computational graphs

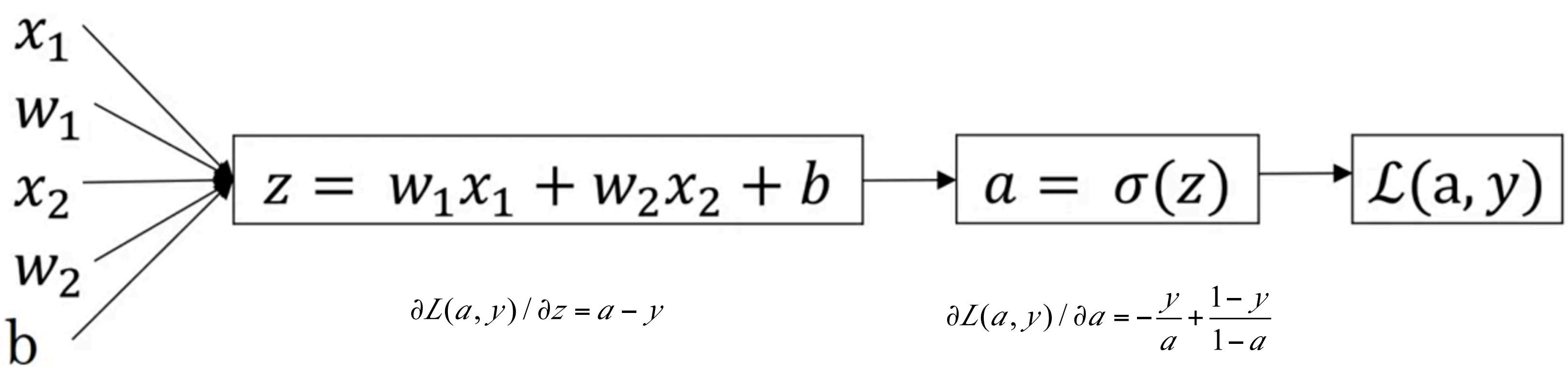
- A model to compute the partial derivatives of the loss function with respect to the input, following the chain rule in backard direction
- Used to implement the backpropagation algorithm in some frameworks
- Ex: J=3(a+bc)



Computational graphs

- Computational graph for logistic Loss function:

$$z = w^T x + b$$
$$\hat{y} = a = \sigma(z)$$
$$\mathcal{L}(a, y) = -(y \log(a) + (1 - y) \log(1 - a))$$



$\partial L(a, y) / \partial z = a - y$

$\partial L(a, y) / \partial a = -\frac{y}{a} + \frac{1 - y}{1 - a}$

$\partial L(a, y) / \partial w_i = x_i \partial L(a, y) / \partial z$

$\partial L(a, y) / \partial b = \partial L(a, y) / \partial z$

Gradient descent

1. Initialize parameters / Define hyperparameters
2. Loop for num_iterations:
    a. Forward propagation
    b. Compute cost function
    c. Backward propagation
    d. Update parameters (using parameters, and grads from backprop)
4. Use trained parameters to predict labels

Gradient descent

```
J = 0, dw₁ = 0, dw₂ = 0, db = 0
for i = 1 to m:
```

$$z^{(i)} = w^T x^{(i)} + b$$
$$a^{(i)} = \sigma(z^{(i)})$$
$$J \mathrel{+}= -\left[y^{(i)} \log a^{(i)} + (1 - y^{(i)}) \log(1 - a^{(i)})\right]$$
$$dz^{(i)} = a^{(i)} - y^{(i)}$$
$$dw_1 \mathrel{+}= x_1^{(i)} dz^{(i)}$$
$$dw_2 \mathrel{+}= x_2^{(i)} dz^{(i)}$$
$$db \mathrel{+}= dz^{(i)}$$

$$J = J/m, \quad dw_1 = dw_1/m, \quad dw_2 = dw_2/m$$
$$db = db/m$$
$$w_1 \mathrel{-}= (\alpha dw_1);\ w_2 \mathrel{-}= (\alpha dw_2);\ b \mathrel{-}= (\alpha db)$$

Complexity?

References and acknowledgements

Some of these slides were inspired or adapted from courses and presentations given by Andrew Ng, Camila Laranjeira, Fei-Fei Li, Flávio Figueiredo, Hugo Oliveira, Jefersson dos Santos, Justin Johnson, Keiller Nogueira, Pedro Olmo, Renato Assunção, Serena Yeung.

Reference courses include *Machine Learning* and *Deep Learning* CS230 and CS231 from Stanford University, *Deep Learning* and *Hands-on Deep Learning* from UFMG, *Deep Learning* CS498 from Un. Of Illinois.