Deep Learning

Machine Learning – Perceptron

Edgar F. Roman-Rangel.
edgar.roman@itam.mx

Digital Systems Department.
Instituto Tecnológico Autónomo de México, ITAM.

January 15th, 2021.

Outline

Basic formulation

Let us think of neural networks as approximatio functions of the form:

$y = Ax$

$$y = f(\mathbf{x}; \mathbf{w}),$$

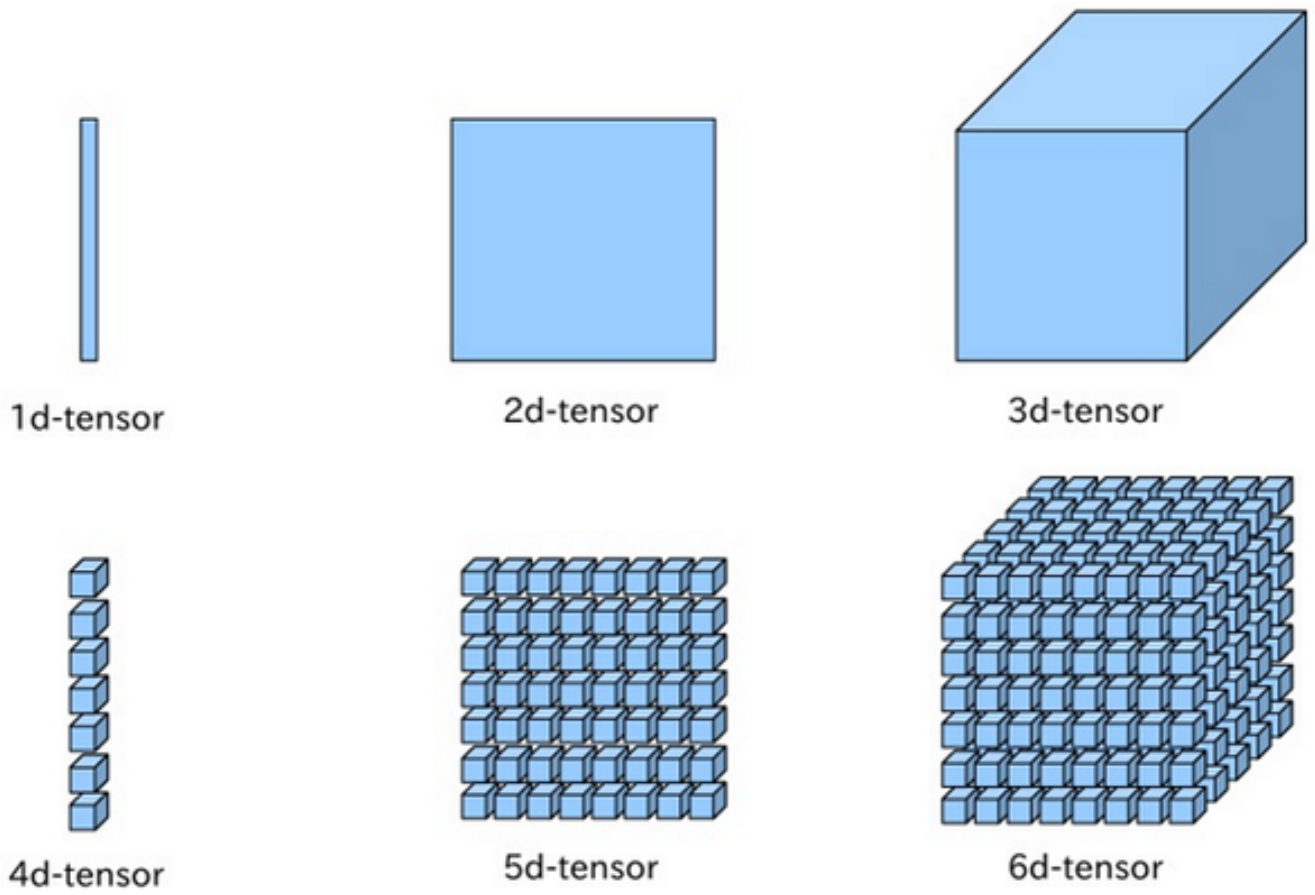
where,

- $\mathbf{x}$ is the input data (vector, text, image, audio, etc). We often use the term "document".
- $f(\cdot)$ is a mapping function (neural network).
- $\mathbf{w}$ is the set $\{\omega_i\}$ of parameters of the network (weights).
- y is the expected output of the mapping function (usually a scalar). We often use the term "label or ground truth".
- $\hat{y}$ is the actual output provided by our model.

Train a model: learning or estimating the values of the parameters $\mathbf{w}$ that best map $\mathbf{x}$ on y.

Notation, I

We will use the term "tensor".



Notation, II

- x a scalar (0th order tensor).
- $\mathbf{x}$ a vector (1st order tensor).
- $\mathbf{X}$ a matrix (or any higher degree tensor).

Moreover,

- $\mathbf{x}_i$ is the i-th element of the vector.
- $\mathbf{x}^j$ is the j-th vector.

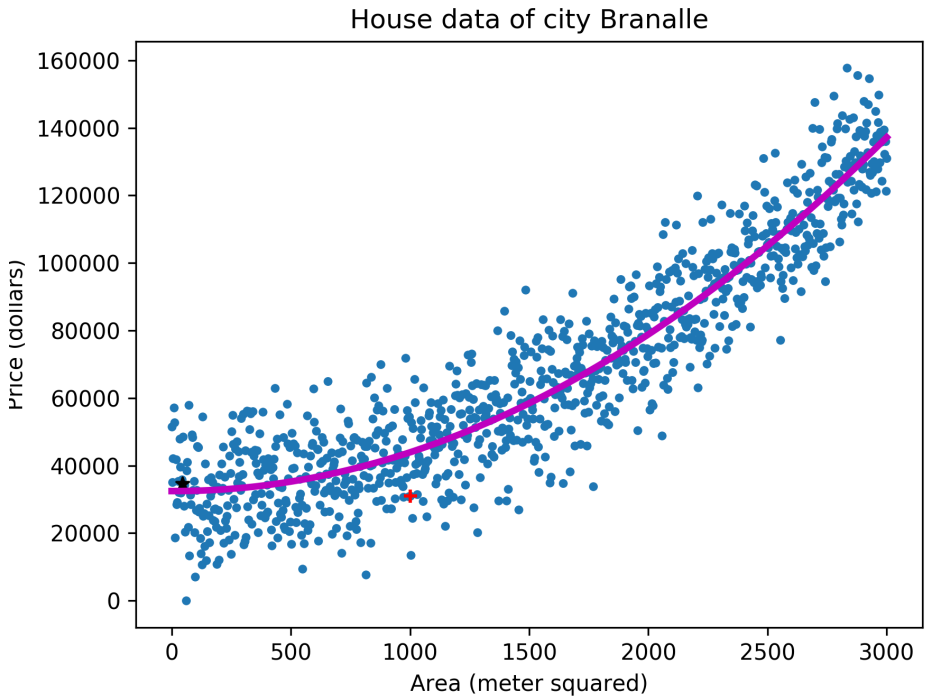
Therefore, we can define a dataset as a collection of pairs (document, label),

$$\{\mathbf{x}^m, y^m\}_{m=1}^{M}.$$

$\mathbf{X}$ would be a matrix, whose rows correspond to data points, and columns to data features.

Regression

- A problem where the output y lives in a continuos space.
- $f(\cdot)$ returns a real value $y \in \mathbb{R}$.



Classification

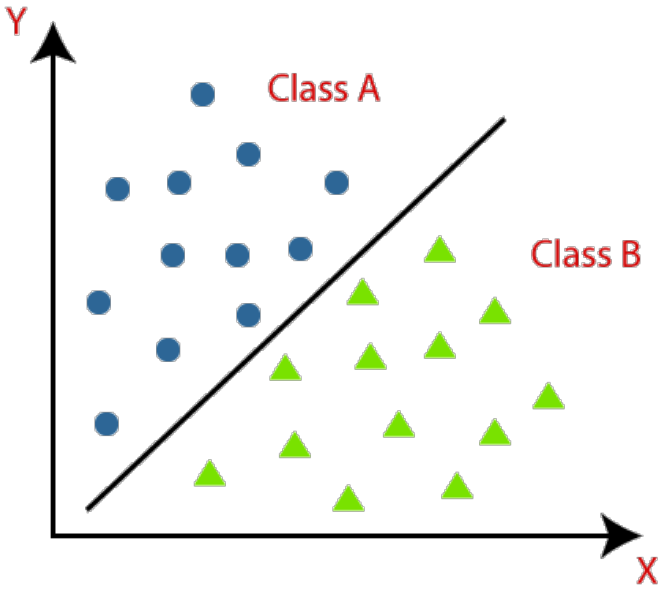
$y = w_1 x_1 + w_2 x_2 + \ldots + b$

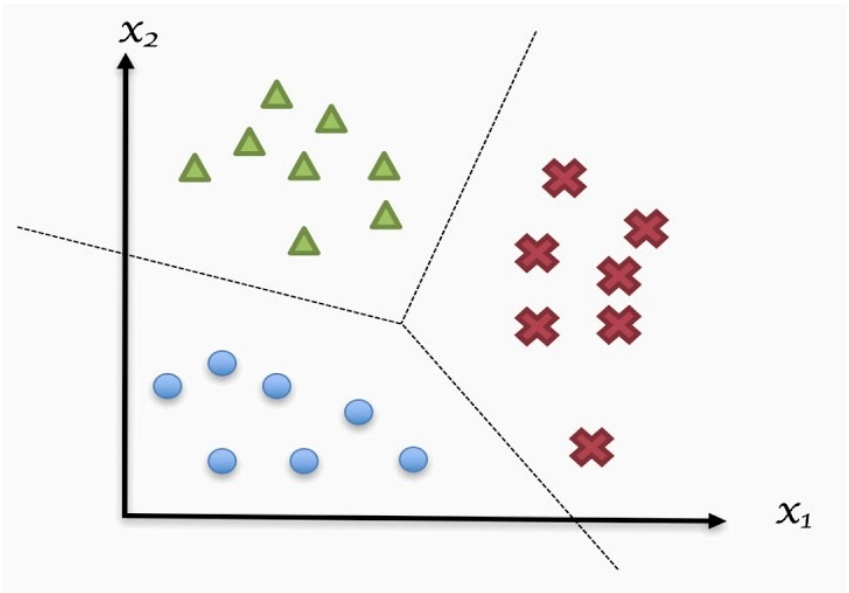
$b = 1 \rightarrow x_0 w_0$

$= w_0 x_0 + w_1 x_1 + w_2 x_2 + \ldots$

$= W^T X$

- A problem where the output y lives in a discrete space.
- $f(\cdot)$ returns a categorical value $y \in \{c_1, c_2, \ldots, c_{|C|}\}$.



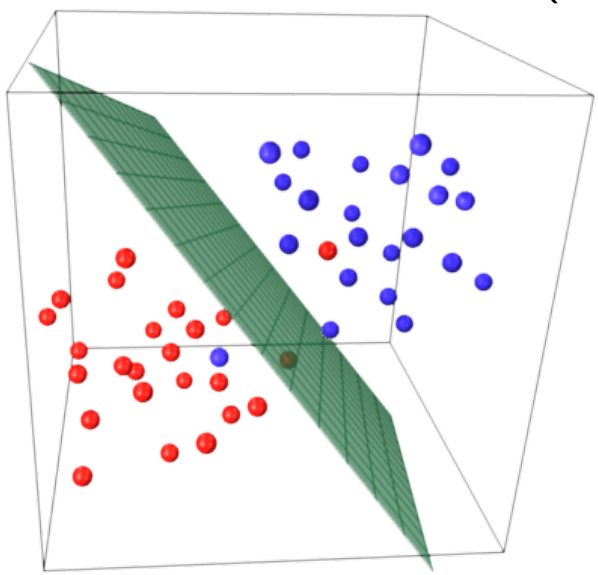


Regression and Classification

In both cases, we must learn the parameters that define a line,

$$y = \mathbf{w}^T \mathbf{x},$$

either a regression line or a separation line (or an hyperplane).



Let us treat both problems with the same approach.

Parameters and hyper-parameters

$y = w^T x$ $w^T = x$

Parameters

"Internal" variables of the model. Estimated from data.

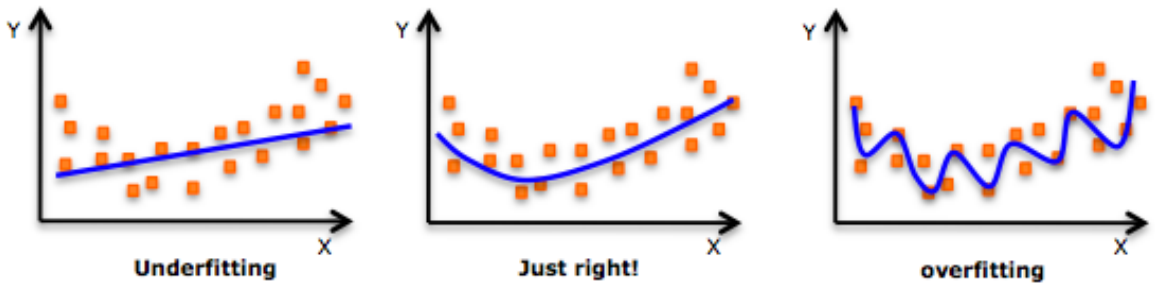
e.g., weights $\{\omega_i\}$ in a regression model.

Hyperparameters

"External" variables of the model. Set manually through validation (trial and error).

e.g., degree of the polynomial in a regression model, or the number of neurons in a neural network.

$y = w_0 x_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + w_5 x_1 x_2$



Types of learning

Supervised learning

$y = f(x)$

Most common type of learning, we train our model using pairs $\{\mathbf{x}, y\}$. E.g., regression, classification.

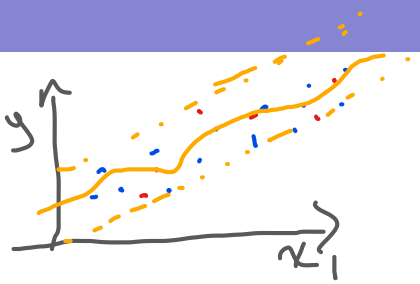
Unsupervised learning

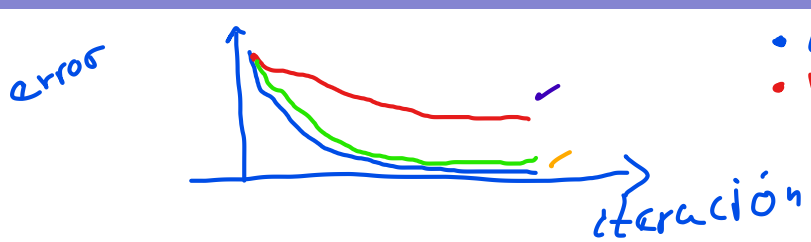
We do not know the labels y, and we are interested in understanding or manipulatig our data. E.g., cluster analysis, dimensionality reduction, anomaly detection.

Reinforcement learning

There is no notion of document and label, but rather states, actions and rewards. And there is an agent that must learn to interact with the environment to maximize its reward.

Datasets, I





Learning makes sense if knowledge can be extrapolated to new events. Therefore we define three data subsets.

$y = wx$

Training set

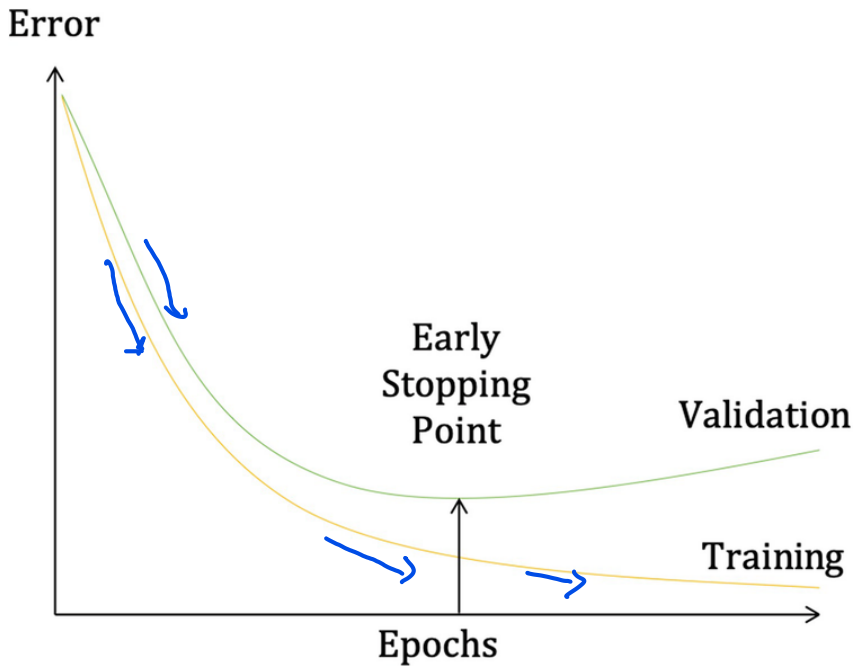
Initial set used to estimate the values for our parameters $\{\omega_i\}$.

Validation set

Once we have adjusted the values of $\{\omega_i\}$, we use this set to validate how well the model performs on data never seen before (how well it generalizes).

If the validation performance is poor in comparison with the training performance, then we must change some hyper-parameters to improve it (shorthen the generalizatio gap).

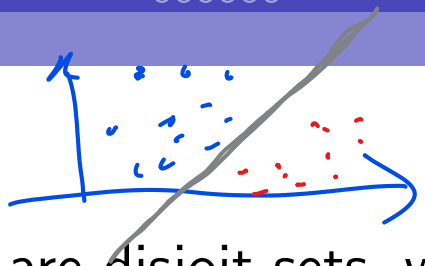
Datasets, II



Test set

Used for final performance evaluation. Data samples in this set were never considered neither to learn the parameters nor the hyperparameters. Therefore, generalization is valid.
This evaluation gives us the expected level of performance for our model once in production.

Datasets, III



Notice, although subsets are disjoit sets, we assume that all data is generated by the same phenomenon.

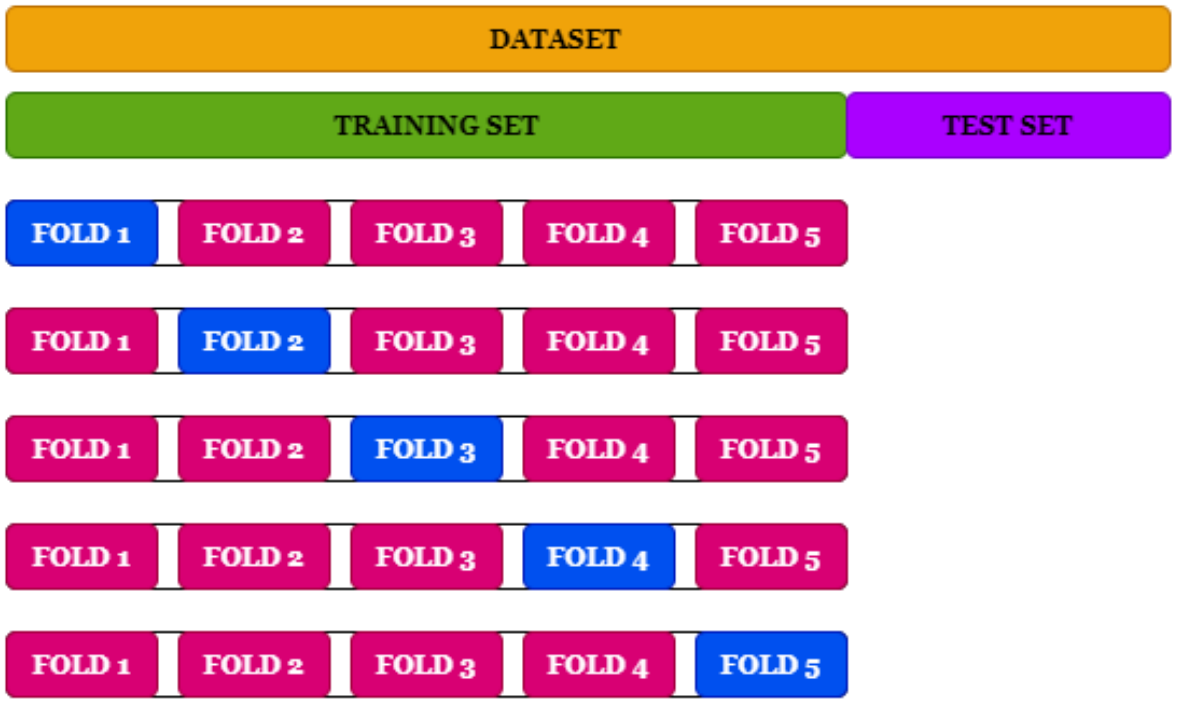
Think of data as random variables, and assume they are generated by the same *pdf*.

Counter example: if we train a model to recognize golden retriever dogs, we cannot expect it to recognize well chihuahuas.

Cross-validation

Some times, we might not have enough data for training a large model properly. If such is the case, we can rely on a cross-validation approach.

Train with subsets, and select the best performing model.



Outline

Linear regression

Approximate y from the input data $\mathbf{x}$, using the set of weigths $\mathbf{w} = \{\omega_i\}$,

$$\mathbf{y} = \mathbf{X}\mathbf{w}.$$

We could learn $\mathbf{w}$ using the normal equation (least squares):

$$\mathbf{w} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}.$$

$-\infty \leq \hat{y} \leq \infty$

Logistic regression

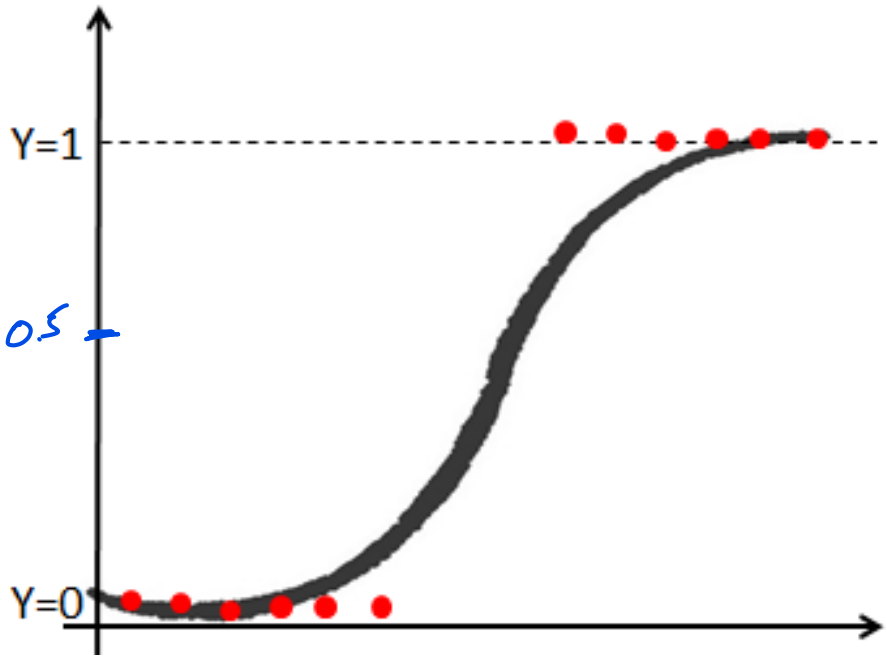
Similarly, we could fit a logistic function to perform binary classification: true vs false (0 vs 1).

$$z = \mathbf{w}^T \mathbf{x},$$
$$y = \sigma(z),$$

where,

$$\sigma(z) = \frac{1}{1 + \exp^{-z}},$$

is the sigmoid function.



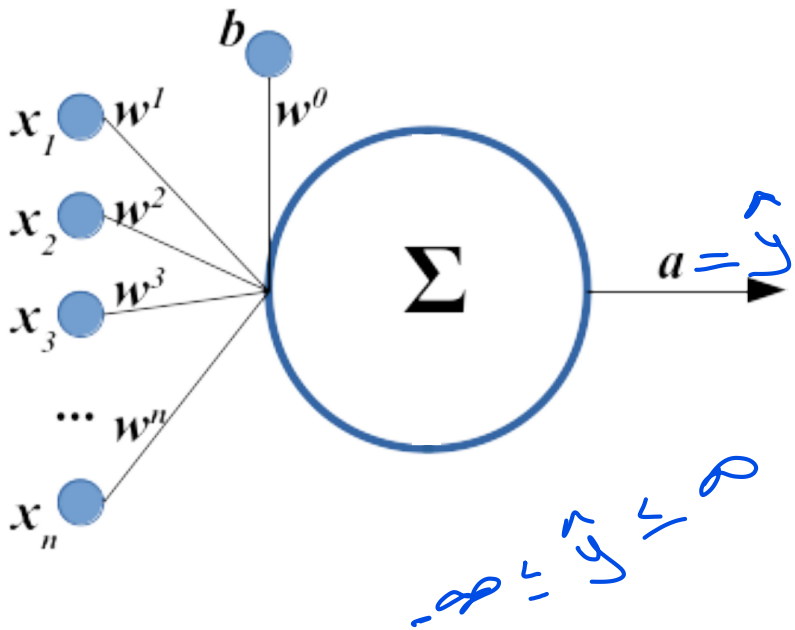
It actually, gives the probability of $\mathbf{y} = 1$.

Linear perceptron

Another formulation for regression problems.

$$
\begin{aligned}
y &= \mathbf{w}^T \mathbf{x}, \\
&= \sum_{n=0}^{N} \omega_n x_n, \\
&= \sum_{n=1}^{N} \omega_n x_n + \omega_0 x_0,
\end{aligned}
$$

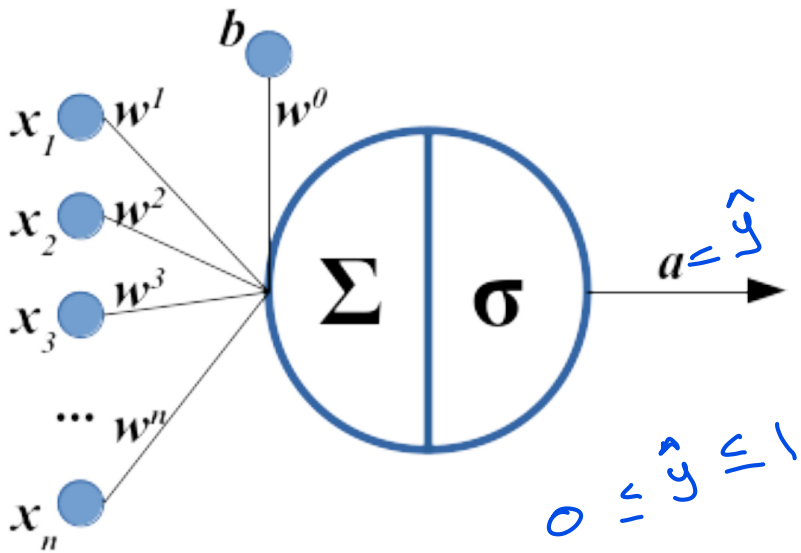
where, $\omega_0 = b$ and $x_0 = 1$.



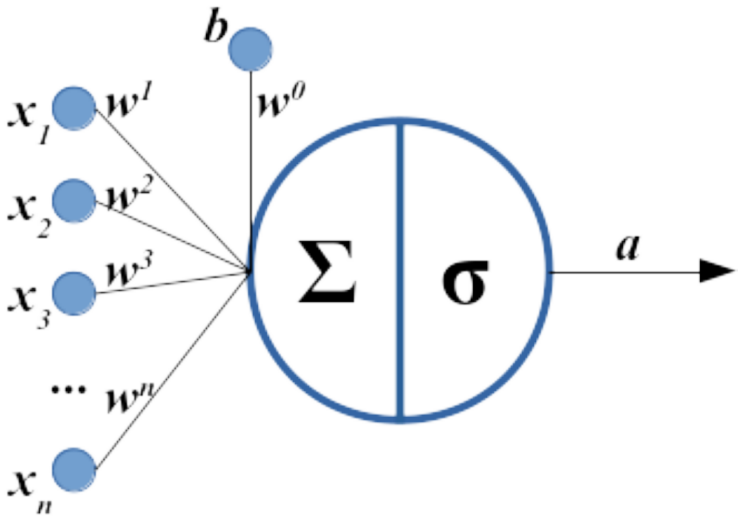
Perceptron

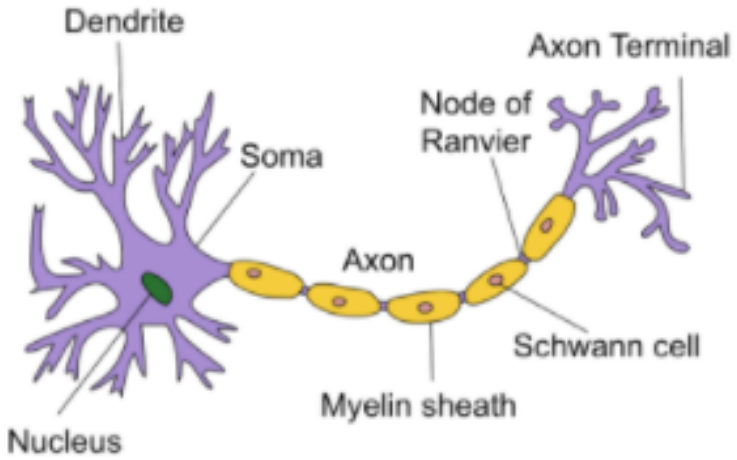
Let's use the sigmoid activation function.

$$s = \mathbf{w}^T \mathbf{x},$$
$$a = \sigma(s).$$



Artificial neuron





Outline

Weights estimation

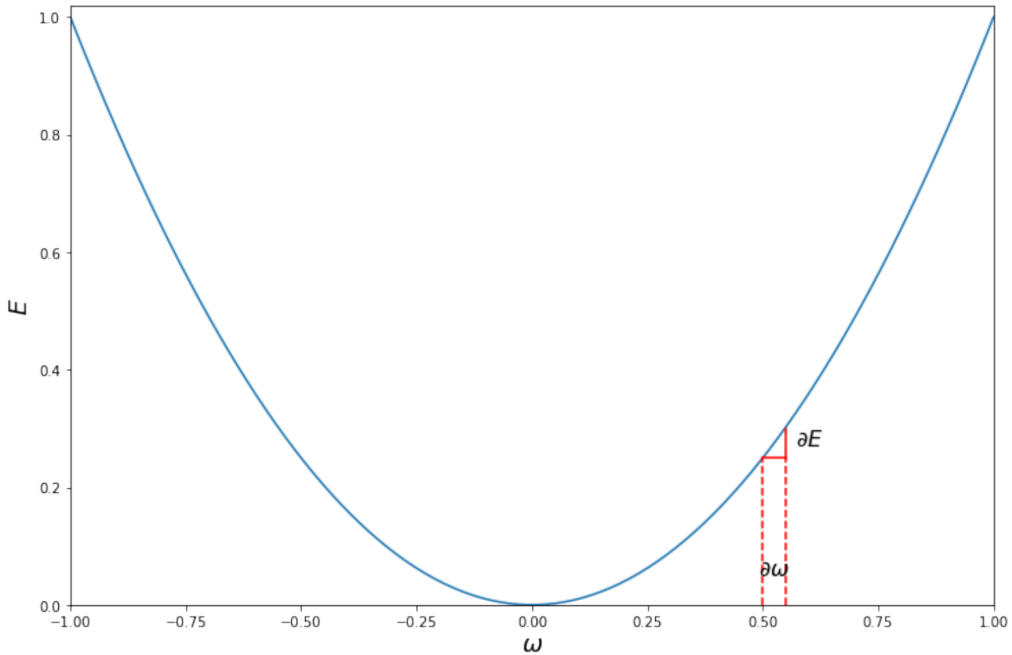
To estimate values for $\{w_i\}$ we use an iterative minization approach termed *Gradient Descent* (GD).

- Most complex problems have no closed-form solution.
- Iterative approaches reach fairly good approximations.
- Risk of getting trapped in local minima.

Gradient descent (GD)

We require a *loss function*. e.g., $E = (y - \hat{y})^2$.

Remeber: relation between derivative, tangent, and direction.



And we can move in the opposite direction of the derivative,

$$\omega_i = \omega_i - \eta \frac{\partial E}{\partial \omega_i}.$$

GD example, I

Consider first only a linear perceptron:

- $\hat{y} = \mathbf{w}^T \mathbf{x} = \sum_{n=0}^{N} \omega_n x_n$,
- $E = (y - \hat{y})^2$.

Then,

$$\begin{aligned}
\frac{\partial E}{\partial \omega_n} &= \frac{\partial (y - \hat{y})^2}{\partial \omega_n}, \\
&= 2(y - \hat{y}) \frac{\partial (y - \hat{y})}{\partial \omega_n}, \\
&= 2(y - \hat{y}) \left[0 - \frac{\partial \sum_{n=0}^{N} \omega_n x_n}{\partial \omega_n} \right], \\
&= -2(y - \hat{y}) x_n.
\end{aligned}$$

Therefore,

$$\omega_n = \omega_n + 2\eta(y - \hat{y}) x_n.$$

GD example, II

Consider now a non-linar perceptron:

- $\hat{y} = \sigma(s)$,
- $s = \mathbf{w}^T\mathbf{x} = \sum_{n=0}^{N} \omega_n x_n$,
- $E = 0.5(y - \hat{y})^2$.

The derivative of the sigmoid function is: $\sigma'(s) = \sigma(s)(1 - \sigma(s))$.

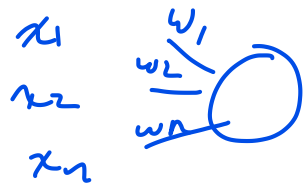
Then,

$$\begin{aligned}\frac{\partial E}{\partial \omega_n} &= \frac{\partial (y - \hat{y})^2}{\partial \omega_n}, \\ &= -(y - \hat{y})\sigma(s)(1 - \sigma(s))x_n.\end{aligned}$$

Therefore,

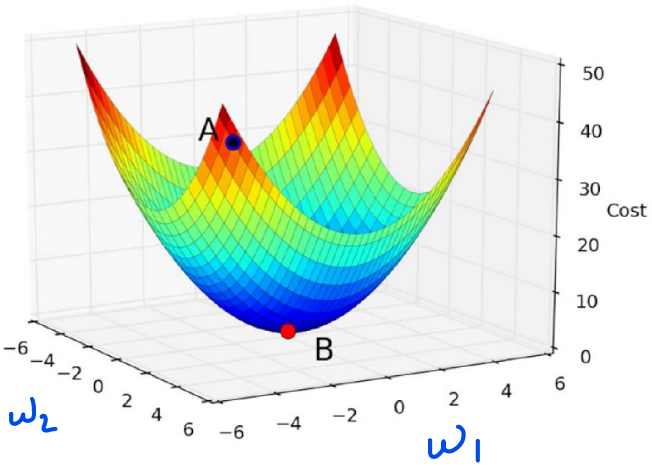
$$\omega_n = \omega_n + \eta(y - \hat{y})\sigma(s)(1 - \sigma(s))x_n.$$

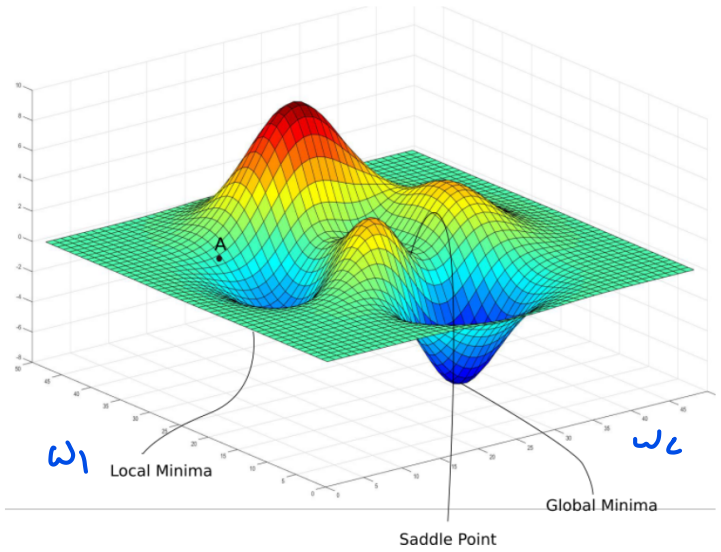
GD multivariado



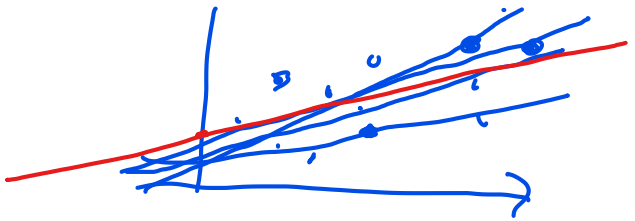
We can use it for multiple parameters.

- We always must move in the direction of the stepest descent, so first compute the all partial derivatives and then update.





GD procedure



$w = \{w_1, \dots w_n\}$

$b = w_0$

For epoch

For training point

1. Random initialization.
2. Forward pass. $\hat{y} = f(x; w)$
3. Error estimation. $E(y, \hat{y})$
4. Gradient computation. $\frac{\partial E}{\partial w_n}$
5. Backward pass (weight adjustment). $w_n = w_n - \eta \frac{\partial E}{\partial w_n}$

$$\frac{\partial E}{\partial w_0} = \frac{\partial E}{\partial b}$$

Q&A

Thank you!

edgar.roman@itam.mx