Machine Learning for High Energy Physics

Pattern Recognition and Classification Systems

Prof. Dr. Edmar Egidio Purcino de Souza

15th International Doctorate Network in Particle Physics, Astrophysics, and Cosmology - IDPASC - Natal/2026

Federal University of Bahia
edmar.egidio@ufba.br / edmar.egidio@cern.ch

Classification Systems: Binary Signal Detection

Introduction to Signal Detection

Introduction

- Different areas of High-Energy Physics (HEP) have been explored using tools based on Machine Learning;
- The possibility of exploring non-linear correlations among input variables, statistical learning, and fast execution processing speed has boosted the use of machine learning in various HEP problems.
- Rare event identification, energy estimation, calorimeter calibration, charged particle track reconstruction, trigger, and mass regression are some examples.
- "*Machine Learning Everywhere* " ... But it wasn't always like this.

Introduction

The field of Machine Learning has always been highly questioned in HEP:

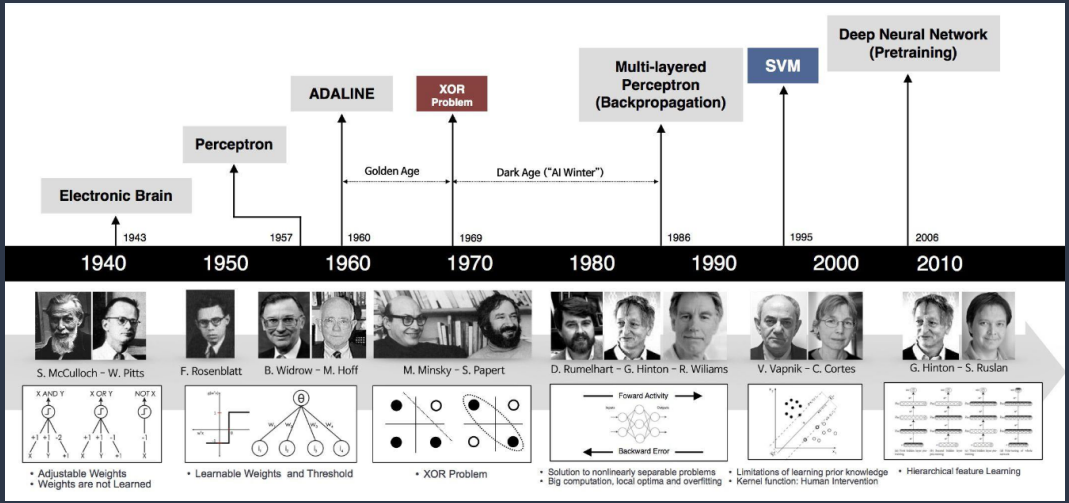
- Difficulty in model explainability;
- Validation strategy of results with high context dependency;
- Previous limitations of high-performance computing;

With the evolution of training techniques, hardware, optimization algorithms, and statistical strategies, it has become possible to use Machine Learning for various applications.

- Machine Learning guided by expert knowledge has been key to the development of these applications.

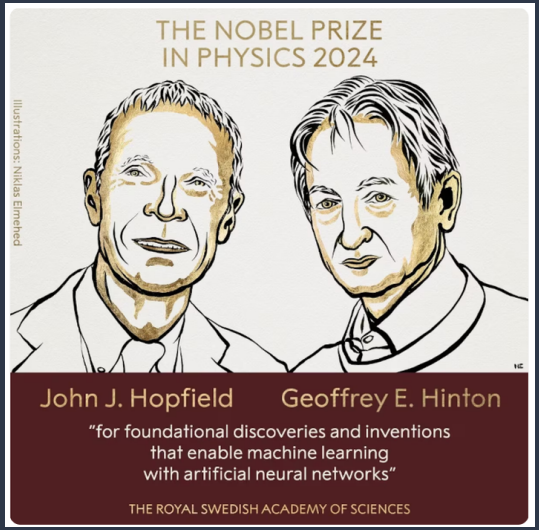
Introduction

Neural Networks Timeline:



Introduction

The 2024 Nobel Prize in Physics goes to two scientists with studies on machine learning:



Introduction

In High-Energy Physics, the area of pattern recognition finds distinct applications, such as:

- Signal/Noise Classification;
- Filtering Systems and Online Particle Identification (Trigger);
- Energy Calibration and Regression;
- Data/Monte Carlo difference estimation;
- Charged particle trajectory reconstruction;

MAGIC Telescope - γ/hadron Separation

The Trigger System of the MAGIC Telescope

R. Paoletti, R. Cecchi, D. Corti, F. Dazzi, M. Mariotti, R. Pegna, and N. Turini

***Abstract*—The MAGIC telescope aims at the detection of very low energy gamma rays ($E > 30$ GeV) through the atmospheric emission of Cherenkov light. The high background rate originating from the night sky background, muons, hadronic showers and bright stars sets a serious challenge to this goal. Application of topological selection cuts at trigger level can have a big impact on background reduction allowing the telescope to operate at lower thresholds and reducing the minimum detectable energy. The trigger of the MAGIC telescope is a two-level system following a pipeline philosophy, similar to those adopted in high energy physics experiments. Operative since October 2002, the trigger system has been a key point in the commissioning of the MAGIC telescope that is now taking data. The trigger hardware is described in detail.**

***Index Terms*—Astroparticle physics, astrophysics, imaging air cherenkov telescope, trigger.**

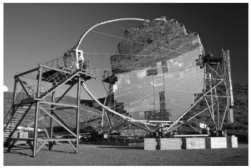
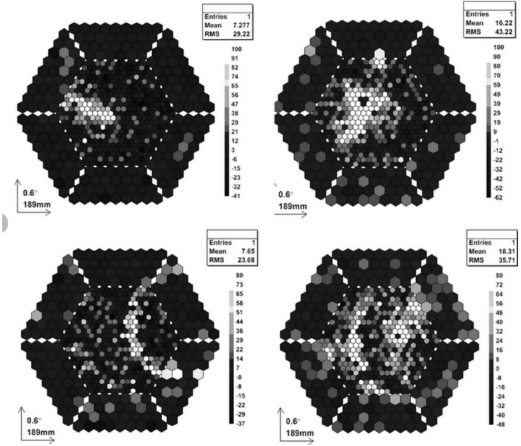


Fig. 1. The MAGIC telescope. On the left is the access tower, used to work on the camera.



Typical shower images in the MAGIC Telescope, corresponding to different candidate events: gamma (top-left), hadron (top-right), muon (bottom-left) and muon plus hadron (bottom-right).

MAGIC Telescope - γ/hadron Separation

The performance of the MAGIC telescopes using deep convolutional neural networks with CTLearn

T. Miener,[a,*] D. Nieto,[a] R. López-Coto,[b] J. L. Contreras,[a] J. G. Green,[c] D. Green[c] and E. Mariotti[d] on behalf of the MAGIC Collaboration

[a] *Instituto de Física de Partículas y del Cosmos and Departamento de EMFTEL, Universidad Complutense de Madrid, Spain*

[b] *Instituto de Astrofísica de Andalucía - CSIC, Granada, Spain*

[c] *Max-Planck-Institut für Physik, München, Germany*

[d] *Dipartimento di Fisica e Astronomia dell'Università and Sezione INFN, Padova, Italy*

E-mail: tmiener@ucm.es

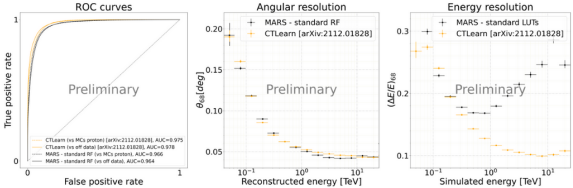


Figure 3: The validation of the performance is taken from [8]. *Left)* ROC curves with MC proton simulations and observational off data. *Center)* Angular resolution vs. reconstructed energy. *Right)* Energy resolution vs. simulated energy.

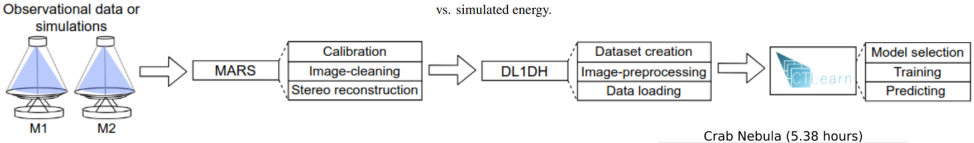
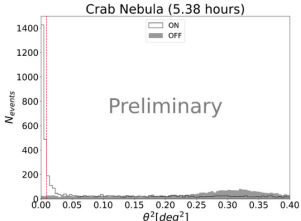


Figure 2: CTLearn's TRN model with channel-wise concatenation of the two stereoscopic images recorded by the MAGIC telescopes (M1 and M2).



Online Event Selection in ATLAS/LHC

Why an online event selection system in ATLAS?

- Collision frequency of 40 MHz. A collision event is $\approx$ 1.7 MB. The total data output rate (without trigger) would be $\approx$ 70 TB/s;
- Physical events of interest are rare, within an environment with intense background noise;
- Not everything can be stored: It must be efficient in selecting the physics of interest while rejecting the majority of irrelevant events;
- What is not selected by the trigger $\rightarrow$ **LOST FOREVER**.

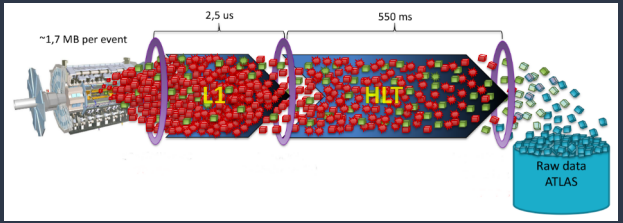


Illustration of the online event selection system employed in ATLAS.

NeuralRinger Algorithm - Online Electron Detection

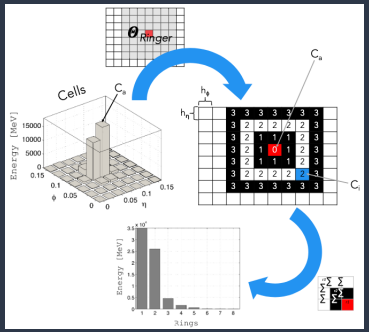


Illustration of ring extraction from cell information.

- The NeuralRinger algorithm operates in the fast stage of the HLT.
- Captures lateral and longitudinal particle shower information in an approximately conical geometry;
- Describes the particle's energy deposition profile in concentric rings around the hottest cell;
- A total of 100 rings are generated, distributed across each layer of the calorimetry system;
- A set of neural classifiers receives the ring signals as inputs and makes the e^-/j decision;

Introduction

Definition
Pattern recognition is formally defined as the process by which a received pattern/signal is assigned to a prescribed number of classes (Haykin, 2020).

- The design of learning machines for pattern recognition must be capable of providing a reasonable response for a given input and performing the "most likely" matching of a pattern;
- It is necessary to consider the statistical variation of the input attributes of the different classes.

Introduction

Pattern Recognition
Signal detection (or classification) consists of identifying, from observations subject to random variations, the type (or class) of signal that was originally sent.

- In the design of a detection system, two very important parameters can be mentioned: discrimination efficiency (maximizing correct hits and minimizing errors) and the execution time of the signal processing chain.
- Classification systems can be designed for two-class detection (binary detection) or multi-class problems (multi-class).

Introduction

A complete pattern recognition system can generally be composed of:

- Sensors that obtain observations to be classified or described;
- A feature extraction mechanism that computes numerical or symbolic information from the observations;
- And a classification scheme for the observations, which depends on the extracted features.

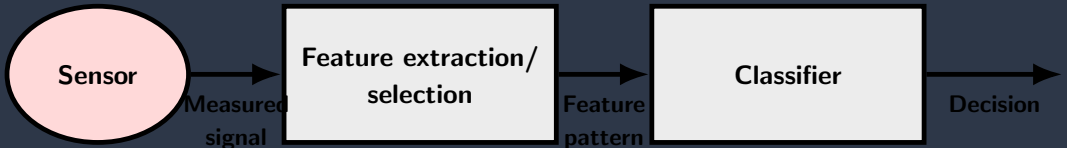
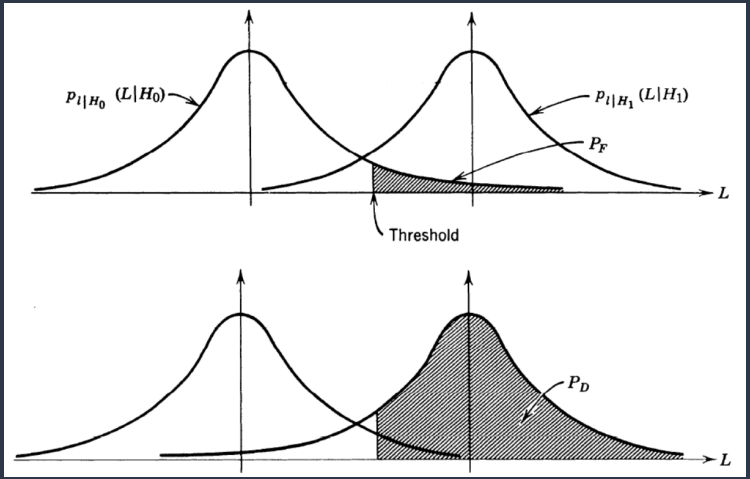


Figure 1: Complete pattern recognition system block diagram.

Binary Signal Detection

Binary Decision

Considering that the decision is based on the observation of the random variable **L**:



Binary Decision

True Positives (TP): Correctly classified positive class examples:

$$TPR = \frac{TP}{TP + FN} \tag{1}$$

True Negatives (TN): Correctly classified negative class examples:

$$TNR = \frac{TN}{TN + FP} \tag{2}$$

False Positives (FP): Events belonging to the negative class, incorrectly classified as positive class:

$$FPR = \frac{FP}{FP + TN} \tag{3}$$

S

False Negatives (FN): Examples belonging to the positive class, incorrectly classified as negative class:

$$FNR = \frac{FN}{TP + FN} \tag{4}$$

ROC Curve

The decision threshold of the classification system can be chosen through the analysis of the ROC (Receiver Operating Characteristics) curve:

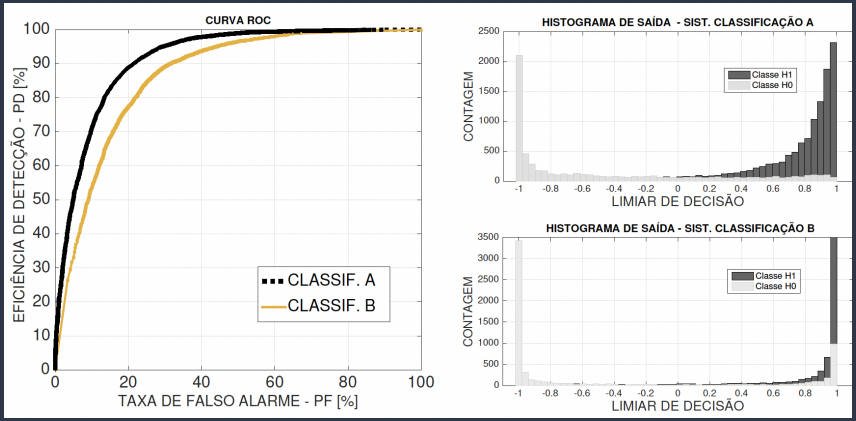
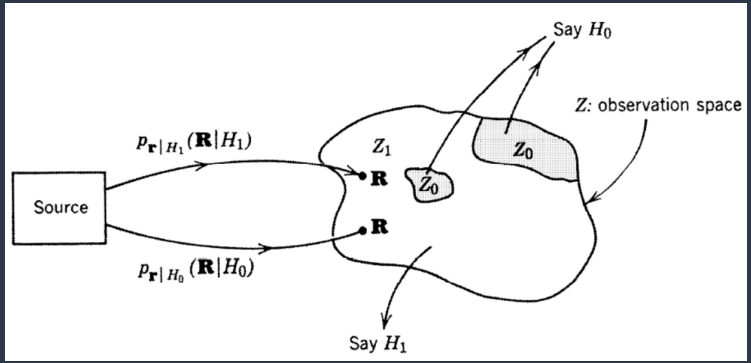


Figure 2: ROC curves of two distinct hypothetical binary classifiers and their respective output histograms

Binary Decision Based on a Single Observation

Considering that each hypothesis is associated with an output mapped to a region of an N-dimensional observation space, a point in this space can be represented by the vector $\mathbf{r}$.

- A decision rule (threshold) partitions the observation space into two regions, Z_1 and Z_0, associated with hypotheses H_1 and H_0 respectively, depending on the observation of $\mathbf{r}$.



Defining the Decision Threshold Between Classes

- The conditional probability $P_{\mathbf{r}/H_i}(\mathbf{R}/H_i)$ is called the a posteriori probability, as it is estimated after observing $\mathbf{r}$.
- When the conditional probability densities are known (or can be estimated), the pattern recognition system design can be simplified.
- The **Maximum A Posteriori**, **Bayes**, **Minimax**, and **Neyman-Pearson** criteria are classical procedures used to choose the decision rule when conditional probabilities are known.

Artificial Neural Network Training

In supervised learning, the model encodes the knowledge of the pattern that best describes the classes within its internal variables, based on a statistically representative set of labeled samples:

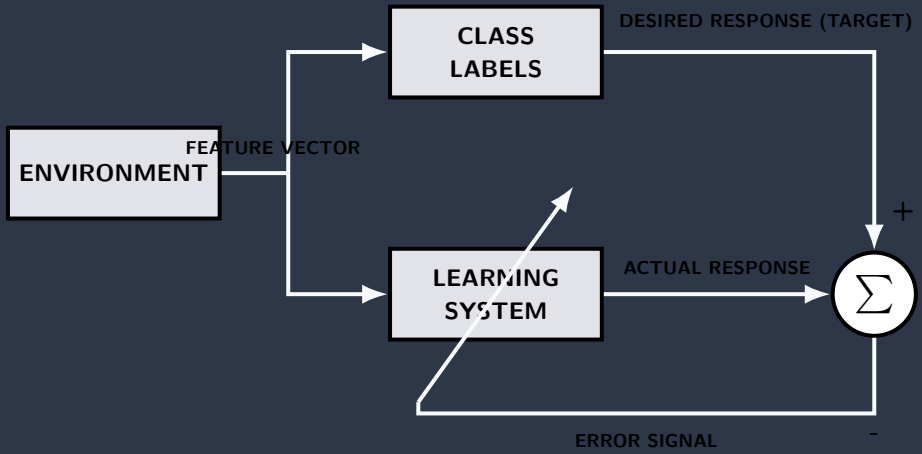


Figure 3: Supervised learning paradigm block diagram.

Introduction to Artificial Neural Networks

The Perceptron

The Perceptron: Historical Context

Proposed by Frank Rosenblatt in 1958, the Perceptron is the oldest and simplest form of an artificial neural network. It serves as the fundamental building block for modern deep learning architectures.

Biological Inspiration

It was designed to mimic the behavior of a biological neuron:

- **Dendrites** receive incoming signals (Inputs).
- The **Soma** processes and integrates these signals (Summation).
- The **Axon** transmits the output if a threshold is reached (Activation & Output).

Figure 4: Rosenblatt's Perceptron concept.

The Perceptron: Mathematical Model

The Perceptron is a binary classifier that maps an input vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ to a single binary output $y \in \{0, 1\}$ (or $y \in \{-1, 1\}$).

The Processing Steps

1. **Weighted Sum:** Compute the linear combination of inputs and weights, adding a bias term w_0 (or b):

$$u = \sum_{i=1}^{n} w_i x_i + w_0 = \mathbf{w}^T \mathbf{x} + w_0$$

2. **Activation Function:** Pass the sum through a step function (Heaviside):

$$y = \varphi(u) = \begin{cases} 1 & \text{if } u \geq 0 \\ 0 & \text{if } u < 0 \end{cases}$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta \mathbf{x}(n) e(n)$$

The Decision Boundary

Geometrically, the Perceptron partitions the input space using a linear decision boundary (a hyperplane).

- **Equation of the Boundary:**
$$\mathbf{w}^T\mathbf{x} + w_0 = 0$$
- In a 2D input space, this boundary is a straight line:
$$w_1 x_1 + w_2 x_2 + w_0 = 0 \implies x_2 = -\frac{w_1}{w_2} x_1 - \frac{w_0}{w_2}$$
- **The Linear Separability Constraint:** The standard Perceptron can only converge and correctly classify data that is **linearly separable** (i.e., a single straight line/hyperplane can perfectly divide the classes).
- It famously fails on non-linearly separable problems like the **XOR** gate (as proven by Minsky and Papert in 1969).

How the Perceptron Learns

The Perceptron uses a supervised, iterative learning rule to adjust its weights based on classification errors.

For each training sample $(\mathbf{x}, d)$, where $d \in \{0, 1\}$ is the desired (target) class:

1. Compute the actual model output: $y = \varphi(\mathbf{w}^T \mathbf{x} + w_0)$.
2. Compute the error signal:
$$e = d - y$$
3. Update the weights and bias using the learning rate $\eta \in (0, 1]$:
$$\mathbf{w} \leftarrow \mathbf{w} + \eta\, e\, \mathbf{x}$$
$$w_0 \leftarrow w_0 + \eta\, e$$

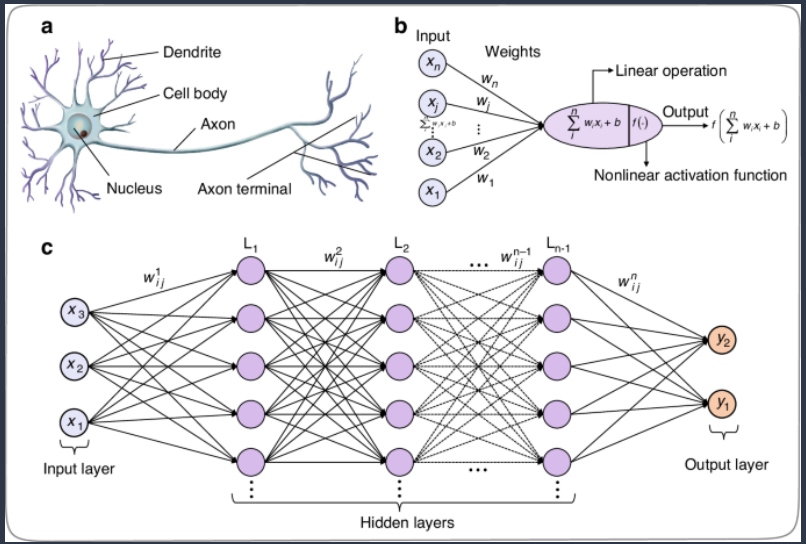
The Convergence Theorem

If the training dataset is linearly separable, the Perceptron learning algorithm is guaranteed to converge to a correct solution in a finite number of steps (Novikoff, 1962).

Artificial Neural Networks

- Multi-layer feed-forward networks are composed of sequential connections of two or more layers of neurons.
- These networks are usually called Multi-layer Perceptrons (MLP) as they are a generalization of the perceptron.
- The output of one layer is used as the input for the next. Since they have no feedback loops (feed-forward networks), they are structurally stable.
- In a pattern recognition process, the hidden layer is responsible for extracting higher-order statistical features by applying a non-linear transformation to the input data.

Artificial Neural Networks



Backpropagation: The Core Concept

At its heart, training a neural network is an optimization problem: we want to minimize a Loss Function $\mathcal{L}$ by adjusting the weights W and biases b.

What is Backpropagation?

It is a highly efficient method to calculate the gradient of the loss function with respect to every single parameter in the network:

$$\nabla \mathcal{L} = \left(\frac{\partial \mathcal{L}}{\partial W}, \frac{\partial \mathcal{L}}{\partial b} \right)$$

- **The Chain Rule:** It systematically applies the calculus chain rule from the output layer backward to the input layer.
- **Dynamic Programming:** It avoids redundant calculations by caching intermediate gradients at each layer, running in $\mathcal{O}(N)$ time.

Step 1: The Forward Pass

Before computing gradients, we must feed the input forward through the network to compute all intermediate activations and the final loss.

For any layer l (from $l = 1$ to L):

1. **Pre-activation (linear combination):**
$$z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$$
2. **Post-activation (non-linear transformation):**
$$a^{[l]} = g^{[l]}(z^{[l]})$$

Where:

- $a^{[0]} = x$ (the input features).
- $g^{[l]}(\cdot)$ is the activation function of layer l.
- $a^{[L]} = \hat{y}$ (the model's final prediction).

Step 2: The Backward Pass (The Chain Rule)

Let us define the "error" (or sensitivity to change) of layer l as:

$$\delta^{[l]} = \frac{\partial \mathcal{L}}{\partial z^{[l]}}$$

Using the chain rule, we compute this error recursively from the last layer L backward:

- **For the Output Layer (L):**

  $$\delta^{[L]} = \nabla_{a^{[L]}} \mathcal{L} \odot {g^{[L]}}'(z^{[L]})$$

  How much the loss changes with the output $\times$ how much the output changes with the pre-activation.

- **For any Hidden Layer ($l = L - 1$ down to 1):**

  $$\delta^{[l]} = \left((W^{[l+1]})^T \delta^{[l+1]} \right) \odot {g^{[l]}}'(z^{[l]})$$

  We backpropagate the error $\delta^{[l+1]}$ from the next layer through the weight matrix $W^{[l+1]}$.

Here, $\odot$ represents the element-wise (Hadamard) product.

Step 3: Parameter Gradients & Optimization

Once we have calculated the error $\delta^{[l]}$ for each layer, computing the partial derivatives with respect to our actual weights and biases is straightforward:

Weight Gradients

$$\frac{\partial \mathcal{L}}{\partial W^{[l]}} = \delta^{[l]} (a^{[l-1]})^T$$

Bias Gradients

$$\frac{\partial \mathcal{L}}{\partial b^{[l]}} = \delta^{[l]}$$

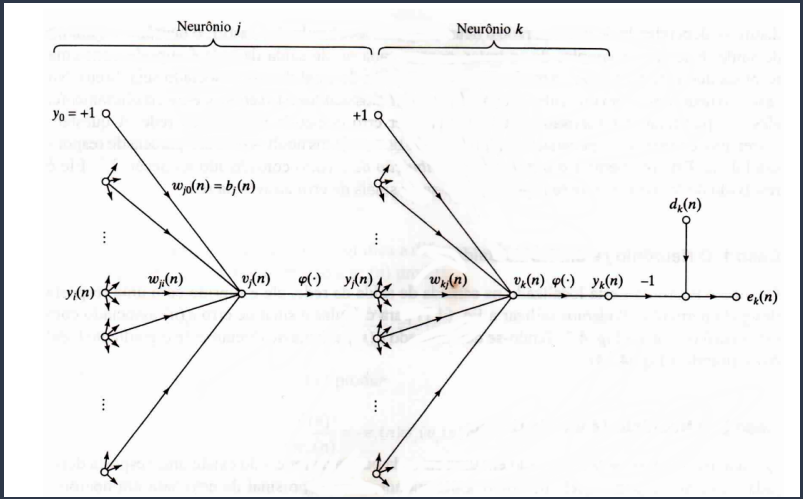
The Weight Update (Gradient Descent): Using a learning rate α, we update the parameters to step in the direction of the steepest descent:

$$W^{[l]} \leftarrow W^{[l]} - \alpha \frac{\partial \mathcal{L}}{\partial W^{[l]}}$$

$$b^{[l]} \leftarrow b^{[l]} - \alpha \frac{\partial \mathcal{L}}{\partial b^{[l]}}$$

Artificial Neural Network Training

Error Backpropagation Algorithm:



Error Backpropagation Algorithm:

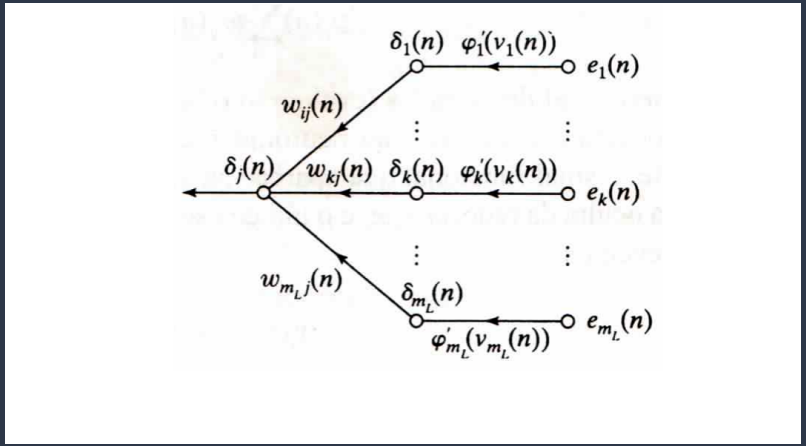


Figure 6: Error backpropagation (backpropagation).

Artificial Neural Network Training

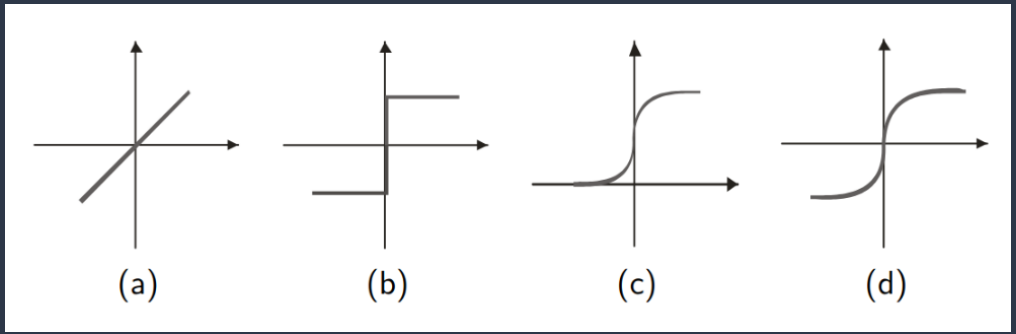
For the neural classifier design, the available input-output pairs are generally divided into the following sets:

- Training: used for adjusting synaptic weights.
- Validation: used to stop training, preventing the occurrence of overfitting.
- Testing: used to evaluate the obtained result and verify the generalization of the model, since this set was not used in weight adjustment or algorithm stopping.

Neural Network Architecture and Tuning Parameters

Activation Functions: Must be studied carefully depending on the problem and the selected optimization method.

- Examples of activation functions $\varphi(\cdot)$: (a) linear; (b) step; (c) sigmoid and (d) hyperbolic tangent.



Neural Network Architecture and Tuning Parameters

Hyperparameter Configuration:

- Training Epochs: The number of epochs is the total number of times the neural network passes through the entire training set during training. Each epoch consists of a pass through the complete training set.
- Stopping Criterion: The criterion adopted to stop training. E.g., maximum number of epochs, failure to improve the objective function, etc.
- Optimization Algorithm: Iterative algorithm selected to perform the optimization procedure for the network weights. E.g., Newton, quasi-Newton, Conjugate Gradient, ADAM, etc.
- Batch Size: The batch size generally corresponds to the number of patterns shown to the network before weights are updated.
- Number of Neurons and Hidden Layers: Must be carefully selected, evaluating network performance in terms of overfitting, computational cost, and optimization algorithm convergence.
- Objective Function (loss): Performance evaluation measure used to optimize network weights.

Final Remarks

Computational Exercise 01

1. Explore the Python notebook with the example of neural systems for classification, using MAGIC Telescope data.
2. Evaluate the impact of changing hyperparameters, such as: Number of Epochs, Stopping Criterion, Number of Hidden Neurons, Optimization Algorithm, and Loss Function.

References

- HAYKIN, S., Neural Networks: Principles and Practice. v. 3. Prentice Hall, 2008.
- KUNCHEVA, L. I., Combining Pattern Classifiers: Methods and Algorithms. v. 2. John Wiley and Sons, 2014.
- DA FONSECA PINTO, J. V., Ring-shaped Calorimetry Information for a Neural EGamma Identification with ATLAS Detector, Tech. rep., CERN, Geneva, Mar 2016.
- BETHAPUDI, S., DESAI, S., "Separation of pulsar signals from noise using supervised machine learning algorithms", Astronomy and Computing, v. 23, pp. 15–26, 2018.
- Lecture Notes: Prof. Eduardo F. Simas Filho, Course: Computational Intelligence (ENGA74), Graduate Program in Electrical Engineering - PPGEE/UFBA.