REDES NEURAIS E DEEP LEARNING

CLASSIFICAÇÃO

DIEGO RODRIGUES DSC

INFNET

Bloco	Matéria	Calendário	Avaliação
Treinamento Clássico	Introdução	06/10	
	Classificação	08, *13	
	Regressão	20, *22	
	Agrupamento	27, *29	
	Séries Temporais	03/11, *05	<Modelo Clássico>
Redes Profundas	Deep Feed Forward	10, *12	
	Visão Computacional	17, *19	
	Autoencoders	24, *26	<Modelo Profundo>
Treinamento Moderno	Transfer Learning	01/12, *03	
	Sequências	08, *10	
	Modelos Generativos	15, *17	<Modelo Avançado>

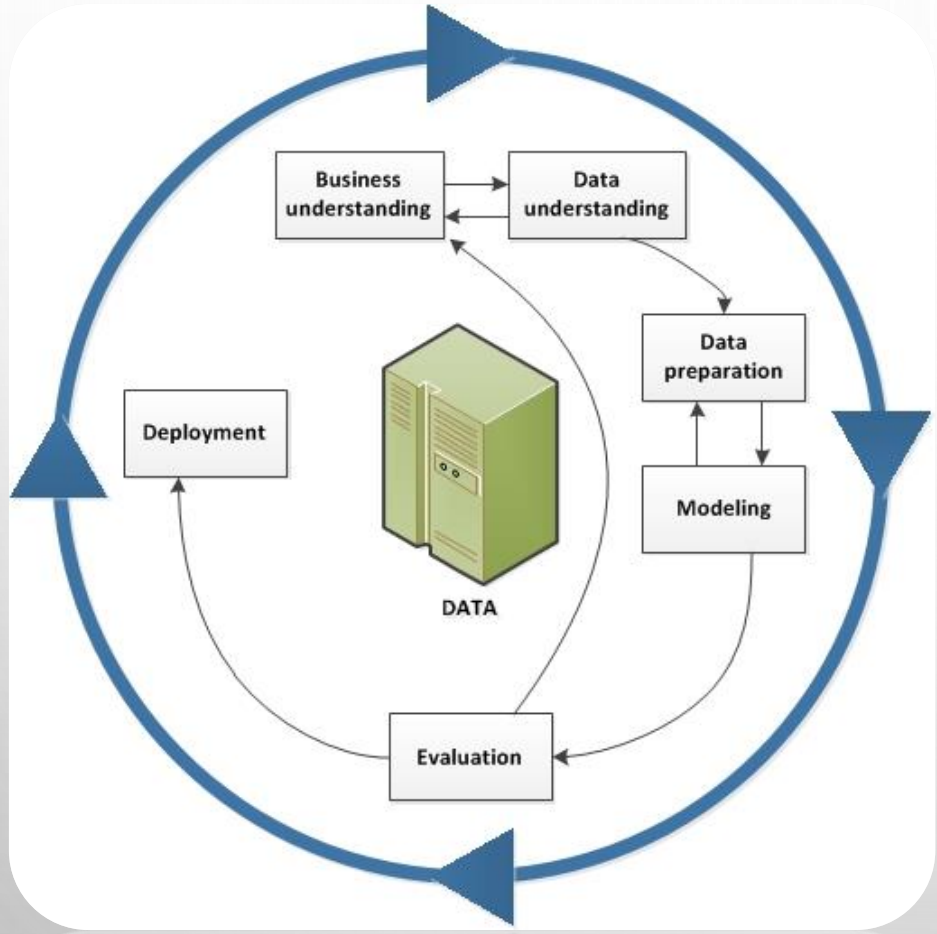
CLASSIFICAÇÃO

PARTE 1 : TEORIA

- CRISP
  - BUSINESS UNDERSTANDING
    - REPRODUTIBILIDADE
  - DATA UNDERSTANDING & PREPARATION
    - PREPARAÇÃO DOS DADOS
    - SELEÇÃO DE ATRIBUTOS
    - EXTRAÇÃO DE ATRIBUTOS

- PARTE 2 : PRÁTICA
  - PRIMEIROS EXPERIMENTOS
  - MODELING
    - REGULARIZAÇÃO
    - GRADIENTE DESCENDENTE
    - HIPERPARÂMETROS
    - PONTO DE OPERAÇÃO
  - EVALUATION
    - VALIDAÇÃO CRUZADA

PARTE 1 : TEORIA

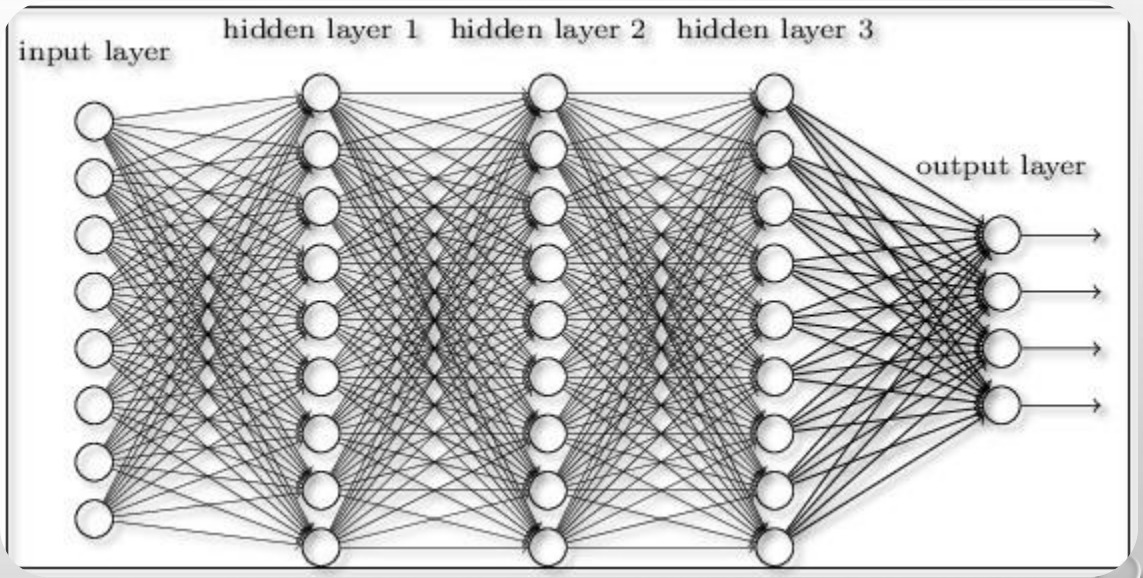


CROSS INDUSTRY PROCESS FOR DATA MINING (CRISP-DM)

BUSINESS UNDERSTANDING

REPRODUTIBILIDADE

- Reprodutibilidade do Experimento
  - Controlar SEED do Numpy & Keras.
  - Mitigar o efeito da inicialização dos Parâmetros



DATA UNDERSTANDING & PREPARATION

DATA PREPARATION

Quantificação dos Atributos

- Transformar todos os atributos em atributos numéricos.

Normalização

- Transformar todos os atributos para a mesma faixa dinâmica, de maneira a assegurar que todos tenham o mesmo "peso numérico" para o treinamento do modelo.

Seleção de Atributos

- Escolher os atributos que mais impactem no resultado do modelo.

Extração de Atributos

- Transformar o Espaço de Atributos para facilitar a resolução do problema.

ATRIBUTOS CATEGÓRICOS

One Hot Encoding

Gender
Female
Male
Male
Female

Gender
1
0
0
1

Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

DATAS

Componentes da Data

- Ano
- Mês
- Dia
- Dia do Ano
- Dia da Semana
- Hora
- Minuto
- Segundo

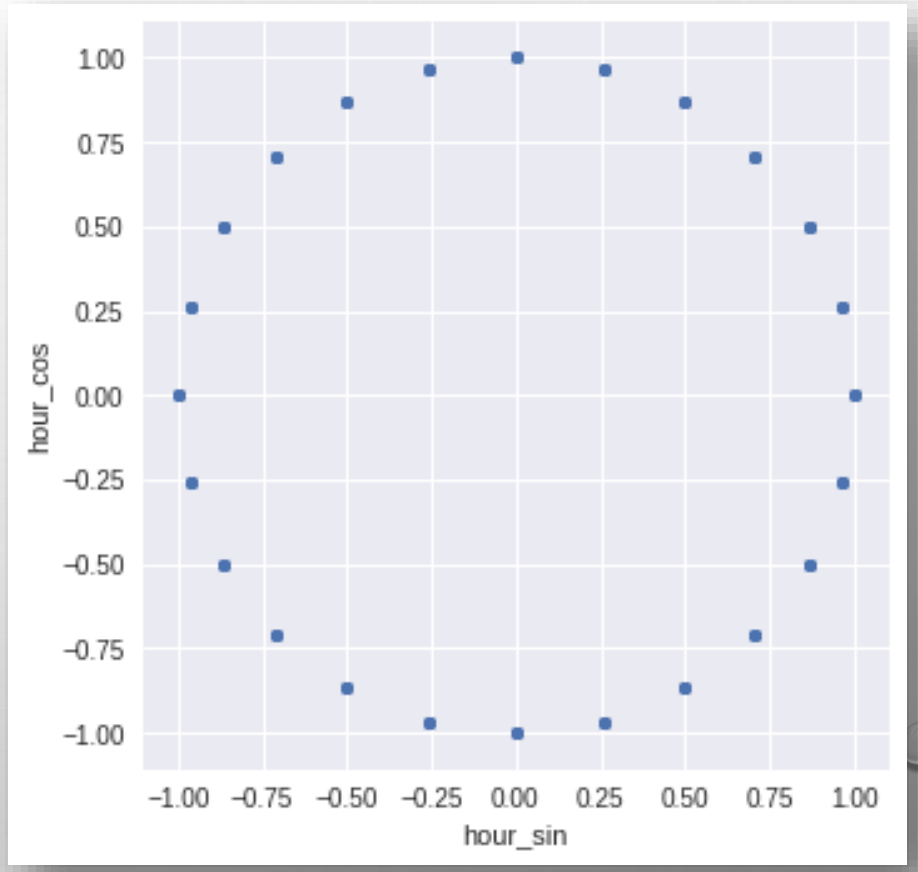
Flags

- É final de semana
- É feriado

Diferença entre Datas

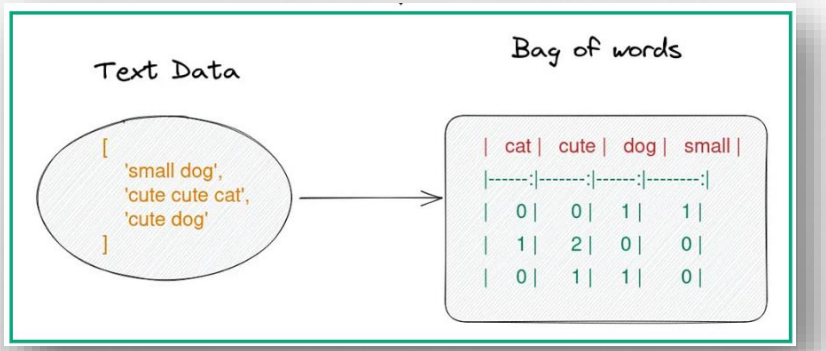
- Diferença em Dias
- Diferença em Horas
- Diferença em Meses

Encoding Cíclico



ATRIBUTOS TEXTUAIS

BAG OF WORDS



TF-IDF

Variants of term frequency (tf) weight

weighting scheme	tf weight
binary	$0, 1$
raw count	$f_{t,d}$
term frequency	$f_{t,d} \Big/ \sum_{t' \in d} f_{t',d}$
log normalization	$\log(1 + f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
double normalization K	$K + (1 - K)\frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

Variants of inverse document frequency (idf) weight

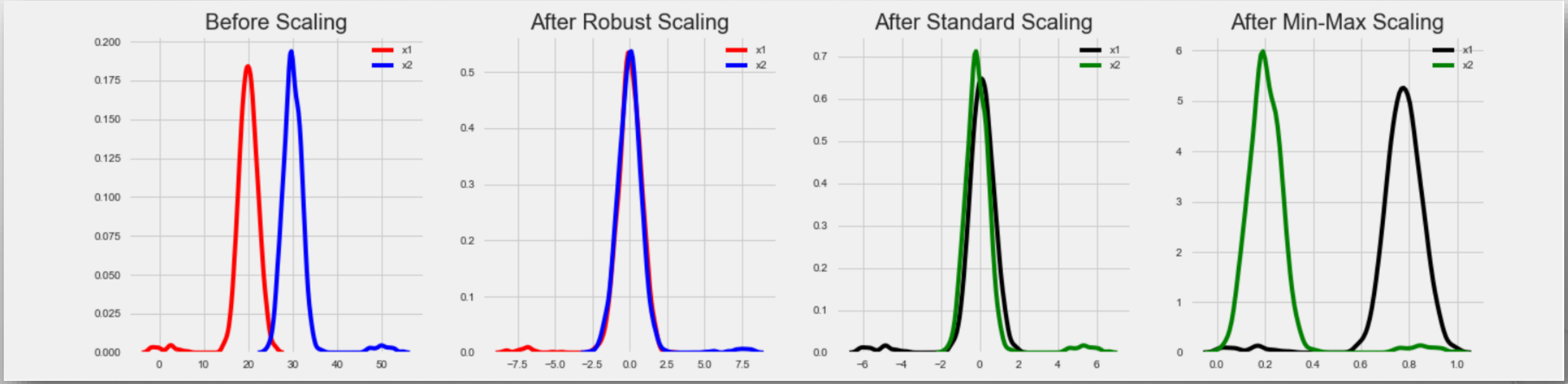
weighting scheme	idf weight ($n_t = \lvert\{d \in D : t \in d\}\rvert$)
unary	1
inverse document frequency	$\log \frac{N}{n_t} = -\log \frac{n_t}{N}$
inverse document frequency smooth	$\log\left(\frac{N}{1+n_t}\right) + 1$
inverse document frequency max	$\log\left(\frac{\max_{\{t' \in d\}} n_{t'}}{1+n_t}\right)$
probabilistic inverse document frequency	$\log \frac{N - n_t}{n_t}$

TF-IDF Calculation Example

Words	Count		Term Frequency (TF)		Inverse Document Frequency (IDF)	TF * IDF	
	Document 1	Document 2	Document 1	Document 2		Document 1	Document 2
read	1	1	0.17	0.17	0	0	0
svm	1	0	0.17	0	0.3	**0.05**	0
algorithm	1	1	0.17	0.17	0	0	0
article	1	1	0.17	0.17	0	0	0
dataaspirant	1	1	0.17	0.17	0	0	0
blog	1	1	0.17	0.17	0	0	0
randomforest	0	1	0	0.17	0.3	0	**0.05**

NORMALIZAÇÃO

- Garantir que as variáveis possuam a mesma escala
- Mesmo efeito numérico na otimização independente da escala.
- Transformar de outra distribuição para distribuição normal



TÉCNICAS DE SELEÇÃO DE ATRIBUTOS

Filtragem – mede a relação entre atributos ou atributos e classes, utilizando estatísticas, sem depender do modelo.

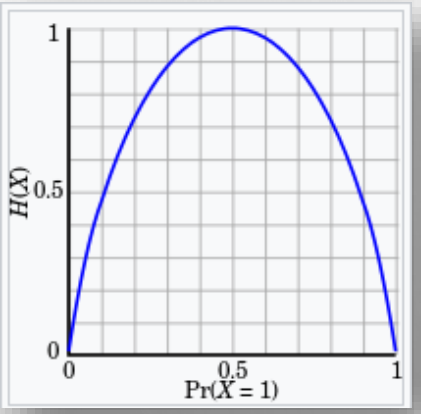
- **Coeficiente de Correlação de Pearson –** Estatística que mede a relação linear entre duas variáveis aleatórias.
- **Teste T de diferença de médias –** Informa se a média de um determinado atributo muda de acordo com uma categoria binária.
- **ANOVA –** O mesmo que o teste T, mas serve para múltiplas categoria.
- **Informação Mútua –** Estatística que mede relação não-linear entre duas variáveis aleatórias.

TÉCNICAS DE SELEÇÃO DE ATRIBUTOS

Wrapper – mede a relação entre atributos e classes, utilizando um modelo treinado.

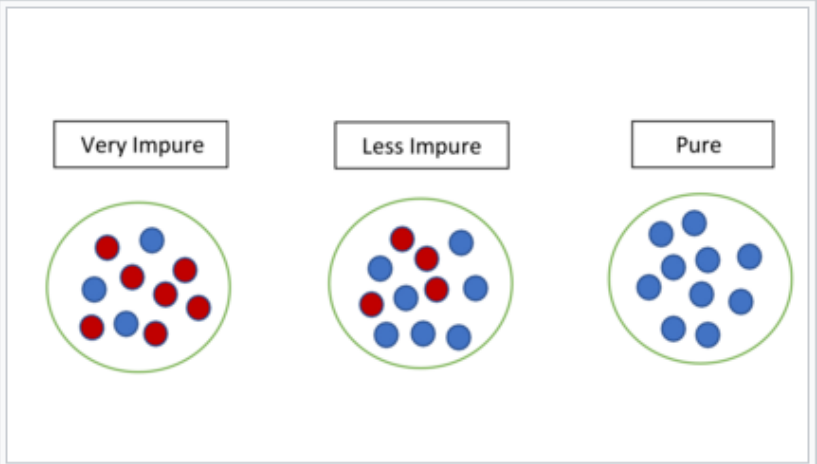
- **Gini –** Estatística que representa a importância de um atributo na divisão da base de dados por uma árvore de decisão.
- **Relevância** – Estatística que representa a variação causada na saída do modelo quando um atributo é substituído por sua média.

ENTROPIA E GANHO DE INFORMAÇÃO



Entropy $H(X)$ (i.e. the expected surprisal) of a coin flip, measured in bits, graphed versus the bias of the coin $\Pr(X = 1)$, where $X = 1$ represents a result of heads.[10]: 14–15

Here, the entropy is at most 1 bit, and to communicate the outcome of a coin flip (2 possible values) will require an average of at most 1 bit (exactly 1 bit for a fair coin). The result of a fair die (6 possible values) would have entropy $\log_2 6$ bits.



Entropy diagram[2]

Entropy [edit]

Entropy $\mathrm{H}(S)$ is a measure of the amount of uncertainty in the (data) set S (i.e. entropy characterizes the (data) set S).

$$\mathrm{H}(S) = \sum_{x \in X} -p(x) \log_2 p(x)$$

Where,

- S – The current dataset for which entropy is being calculated
  - This changes at each step of the ID3 algorithm, either to a subset of the previous set in the case of splitting on an attribute or to a "sibling" partition of the parent in case the recursion terminated previously.
- X – The set of classes in S
- $p(x)$ – The proportion of the number of elements in class x to the number of elements in set S

When $\mathrm{H}(S) = 0$, the set S is perfectly classified (i.e. all elements in S are of the same class).

In ID3, entropy is calculated for each remaining attribute. The attribute with the **smallest** entropy is used to split the set S on this iteration. Entropy in information theory measures how much information is expected to be gained upon measuring a random variable; as such, it can also be used to quantify the amount to which the distribution of the quantity's values is unknown. A constant quantity has zero entropy, as its distribution is perfectly known. In contrast, a uniformly distributed random variable (discretely or continuously uniform) maximizes entropy. Therefore, the greater the entropy at a node, the less information is known about the classification of data at this stage of the tree; and therefore, the greater the potential to improve the classification here.

As such, ID3 is a greedy heuristic performing a best-first search for locally optimal entropy values. Its accuracy can be improved by preprocessing the data.

Information gain [edit]

Information gain $IG(A)$ is the measure of the difference in entropy from before to after the set S is split on an attribute A. In other words, how much uncertainty in S was reduced after splitting set S on attribute A.

$$IG(S, A) = \mathrm{H}(S) - \sum_{t \in T} p(t)\mathrm{H}(t) = \mathrm{H}(S) - \mathrm{H}(S|A).$$

Where,

- $\mathrm{H}(S)$ – Entropy of set S
- T – The subsets created from splitting set S by attribute A such that $S = \bigcup_{t \in T} t$
- $p(t)$ – The proportion of the number of elements in t to the number of elements in set S
- $\mathrm{H}(t)$ – Entropy of subset t

In ID3, information gain can be calculated (instead of entropy) for each remaining attribute. The attribute with the **largest** information gain is used to split the set S on this iteration.

IMPUREZA DE GINI

$$Gini = 1 - \sum_j p_j^2$$

1.10.7.1. Classification criteria

If a target is a classification outcome taking on values 0,1,...,K-1, for node m, let

$$p_{mk} = \frac{1}{n_m} \sum_{y \in Q_m} I(y = k)$$

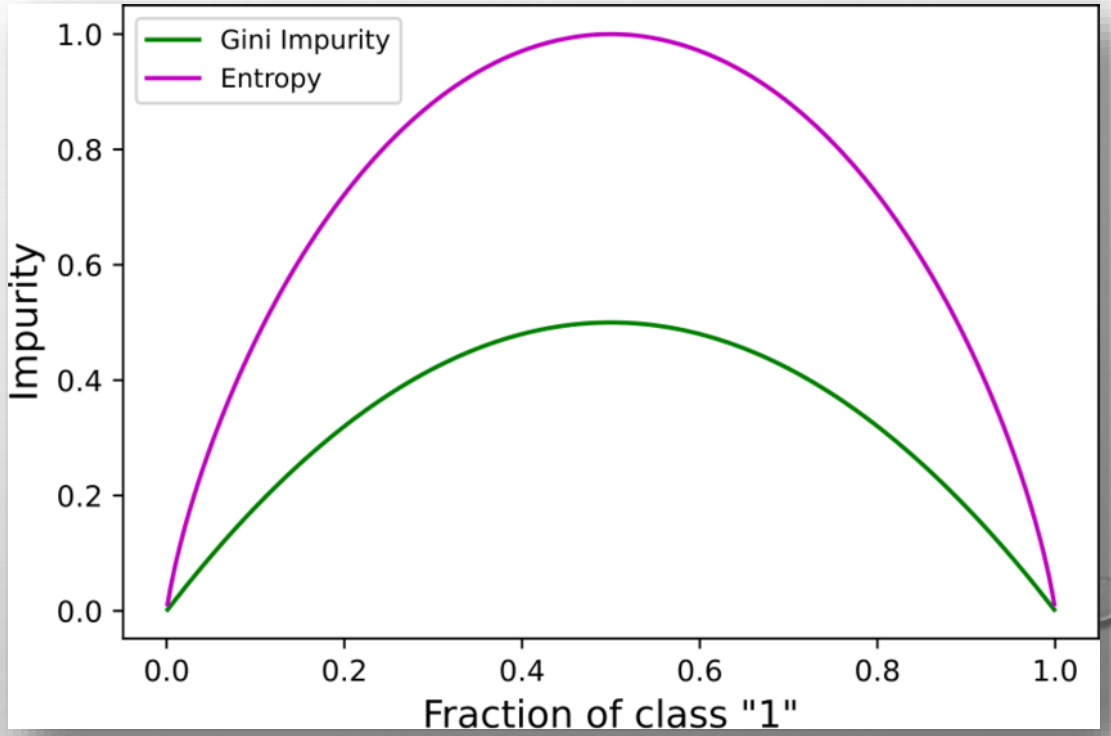
be the proportion of class k observations in node m. If m is a terminal node, `predict_proba` for this region is set to p_{mk}. Common measures of impurity are the following.

Gini:

$$H(Q_m) = \sum_k p_{mk}(1 - p_{mk})$$

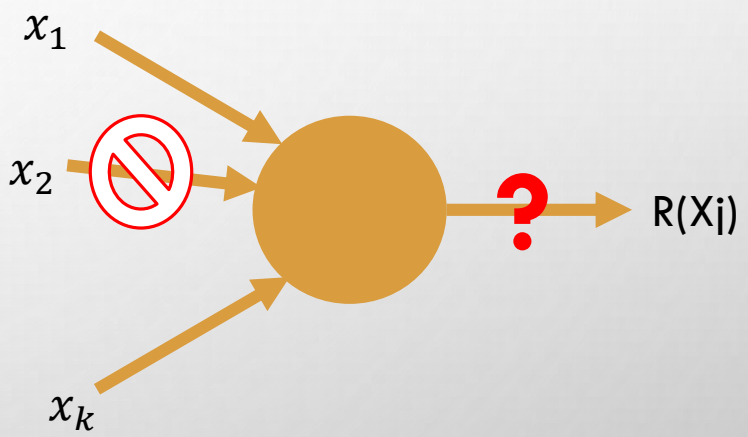
Log Loss or Entropy:

$$H(Q_m) = -\sum_k p_{mk} \log(p_{mk})$$



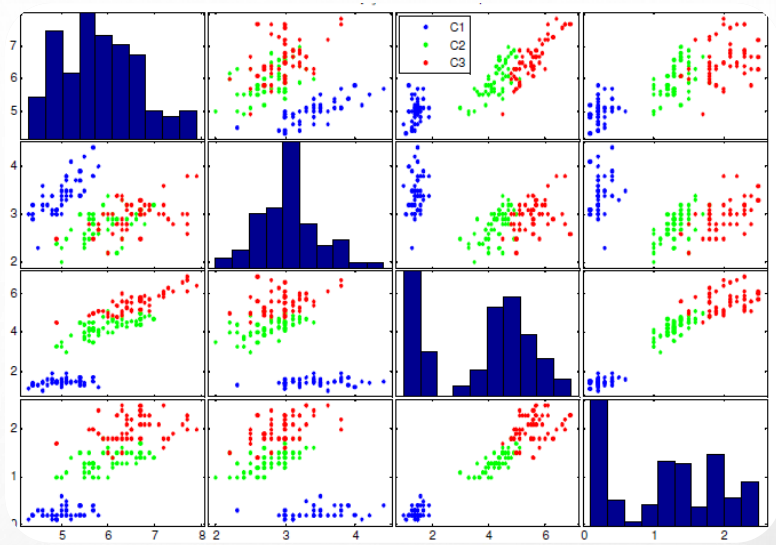
RELEVÂNCIA

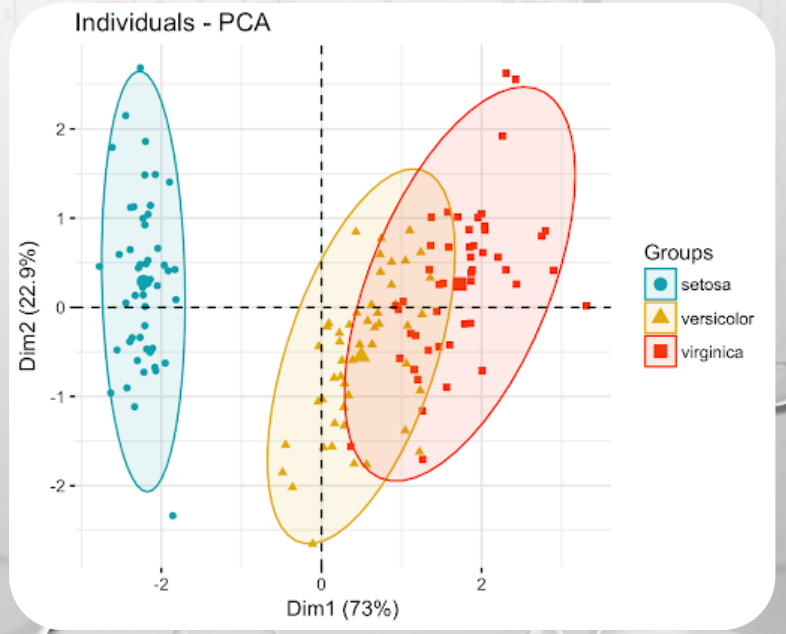
$$R(X_j) = \frac{\sum_{i=1}^{N} \left| \left| \hat{y}(\mathbf{x_i}) - \hat{y}(\mathbf{x_i}|_{x_{ij}=\bar{x}_j}) \right| \right|^2}{N}$$



EXTRAÇÃO DE ATRIBUTOS ANÁLISE DE COMPONENTES PRINCIPAIS (PCA)

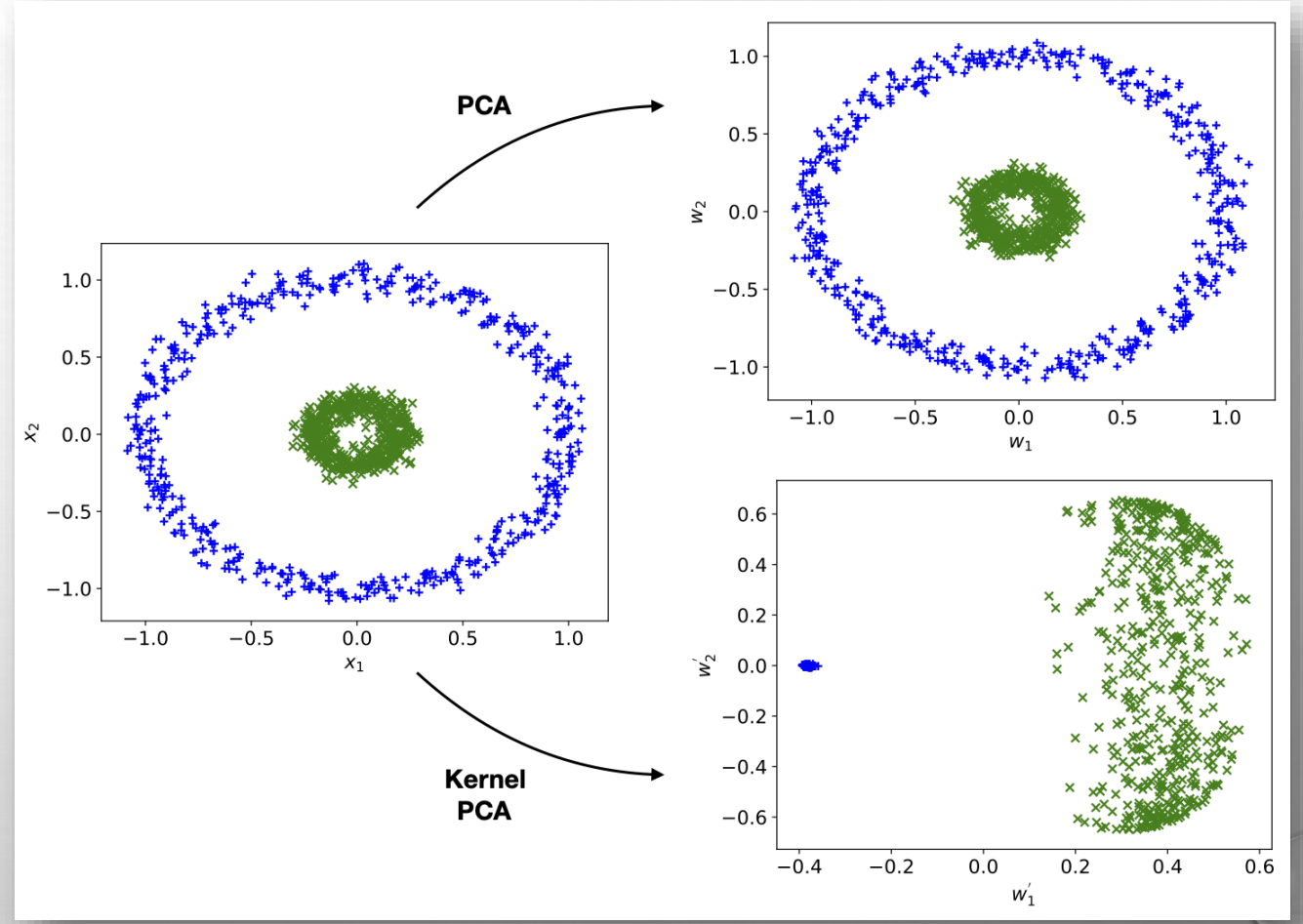
- Garantir que as variáveis independentes sejam descorrelacionadas.
- Identificar novas direções com maior concentração de energia / informação.
- Variáveis transformadas perdem o sentido físico.





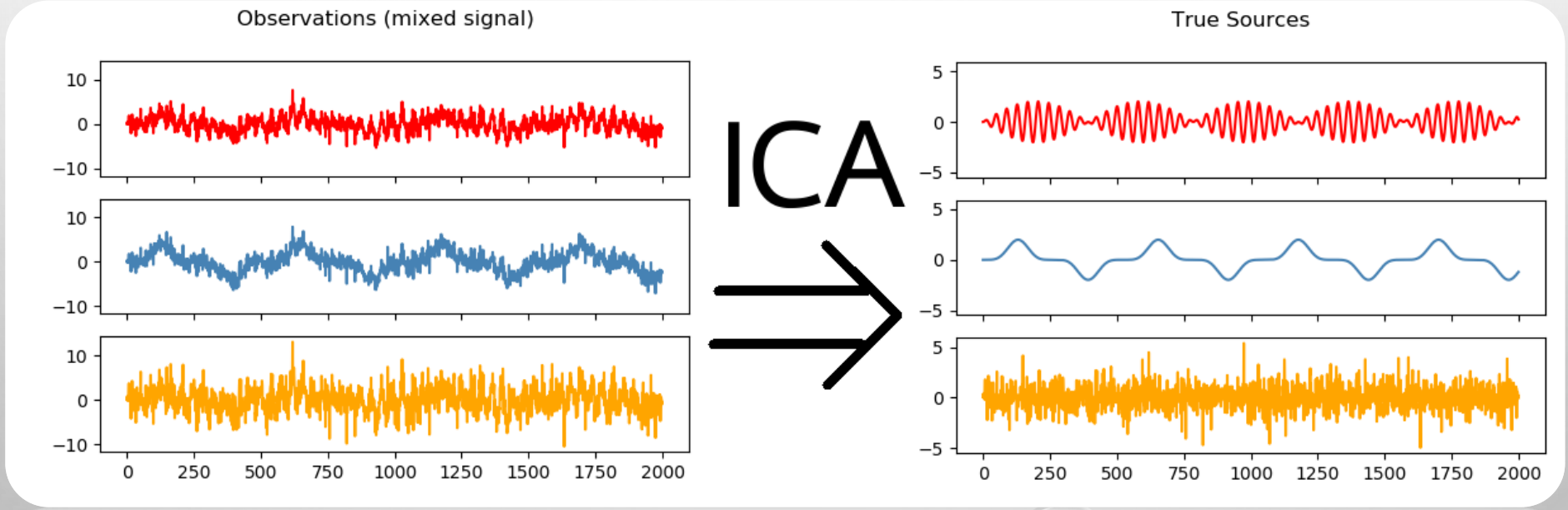
EXTRAÇÃO DE ATRIBUTOS – KERNEL PCA

- Identifica novo espaço que favoreça a modelagem.
- Como selecionar o Kernel Adequado?



ANÁLISE DE COMPONENTES INDEPENDENTES

- Garantir que as variáveis independentes sejam independentes
- Identificar principais direções de não-gaussianidade



MODELING

REDE NEURAL PROFUNDA

4.2.2 *Continuous inputs*

We now discuss the case of continuous input variables, again for units with threshold activation functions, and we consider the possible decision boundaries which can be produced by networks having various numbers of layers (Lippmann, 1987; Lonstaff and Cross, 1987). In Section 3.1 it was shown that a network with a single layer of weights, and a threshold output unit, has a decision boundary which is a hyperplane. This is illustrated for a two-dimensional input space in Figure 4.4 (a). Now consider networks with two layers of weights. Again, each hidden units divides the input space with a hyperplane, so that it has activation

$z = 1$ on one side of the hyperplane, and $z = 0$ on the other side. If there are M hidden units and the bias on the output unit is set to $-M$, then the output unit computes a logical AND of the outputs of the hidden units. In other words, the output unit has an output of 1 only if all of the hidden units have an output of 1. Such a network can generate decision boundaries which surround a single convex region of the input space, whose boundary consists of segments of hyperplanes, as illustrated in Figure 4.4 (b). A convex region is defined to be one for which any line joining two points on the boundary of the region passes only through points which lie inside the region. These are not, however, the most general regions which can be generated by a two-layer network of threshold units, as we shall see shortly.

Networks having three layers of weights can generate arbitrary decision regions, which may be non-convex and disjoint, as illustrated in Figure 4.4 (c). A simple demonstration of this last property can be given as follows (Lippmann, 1987). Consider a particular network architecture in which, instead of having full connectivity between adjacent layers as considered so far, the hidden units are arranged into groups of $2d$ units, where d denotes the number of inputs. The topology of the network is illustrated in Figure 4.5. The units in each group send their outputs to a unit in the second hidden layer associated with that group. Each second-layer unit then sends a connection to the output unit. Suppose the input space is divided into a fine grid of hypercubes, each of which is labelled as class C_1 or C_2. By making the input-space grid sufficiently fine we can approximate an arbitrarily shaped decision boundary as closely as we wish. One group of first-layer units is assigned to each hypercube which corresponds to class C_1,

Neural Networks for Pattern Recognition, Bishop

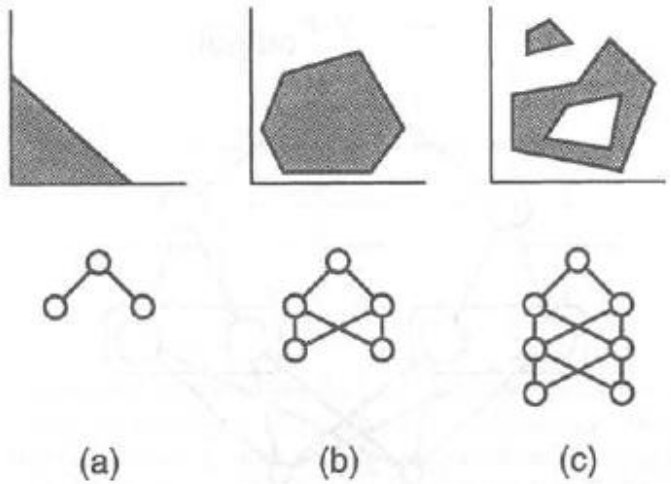
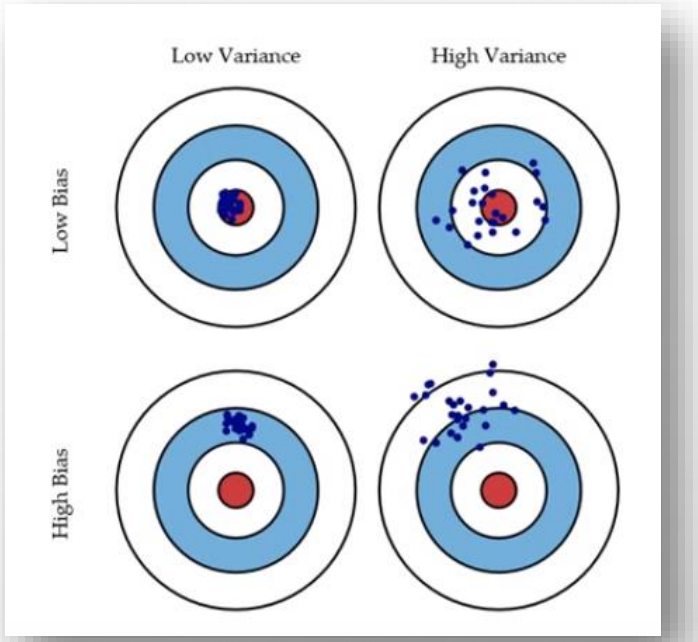
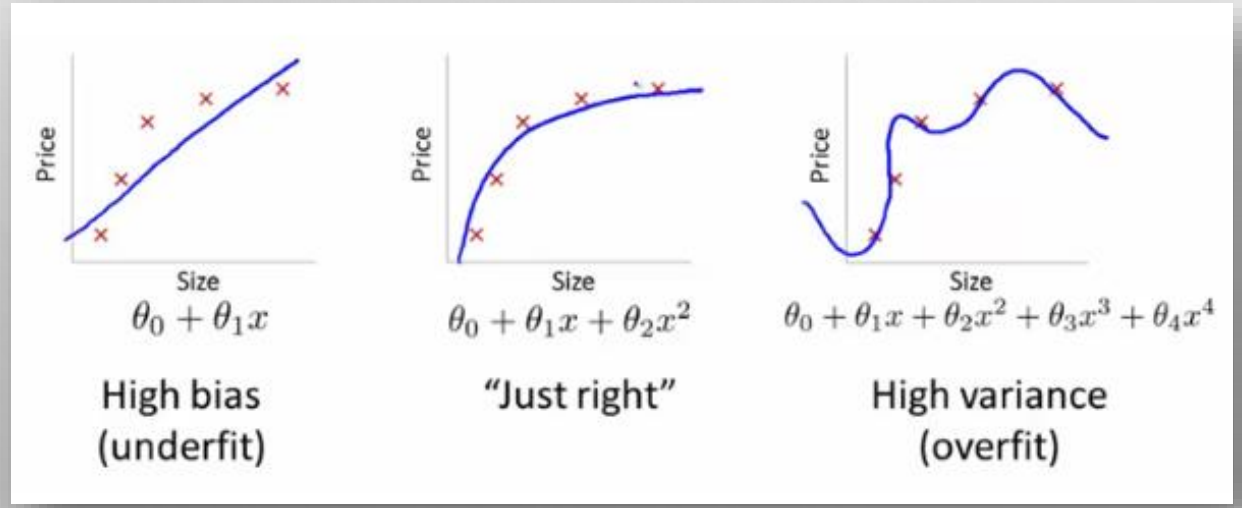
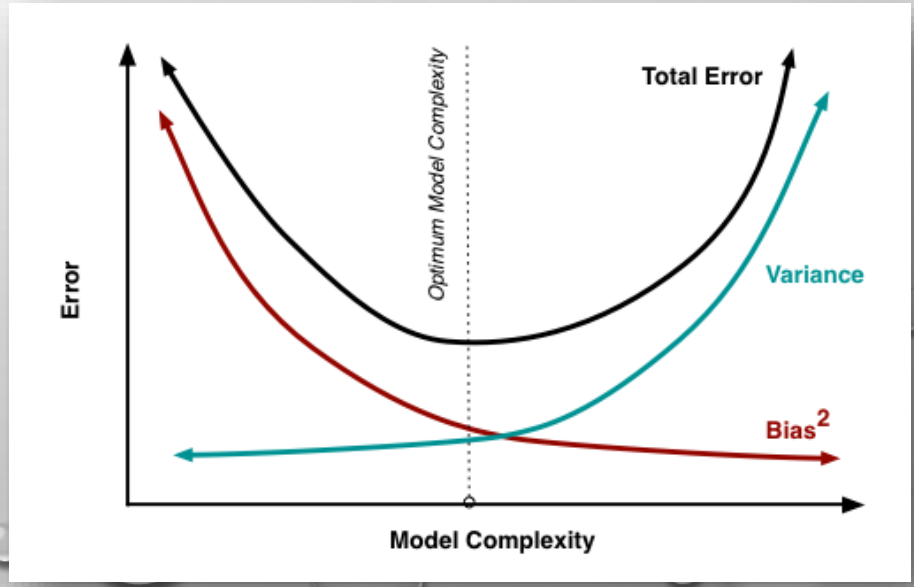


Figure 4.4. Illustration of some possible decision boundaries which can be generated by networks having threshold activation functions and various numbers of layers. Note that, for the two-layer network in (b), a single convex region of the form shown is not the most general possible.

BIAS x VARIANCE







REGULARIZAÇÃO

In mathematics, statistics, finance,[1] and computer science, particularly in machine learning and inverse problems, **regularization** is a process that changes the result answer to be "simpler". It is often used to obtain results for ill-posed problems or to prevent overfitting.[2]

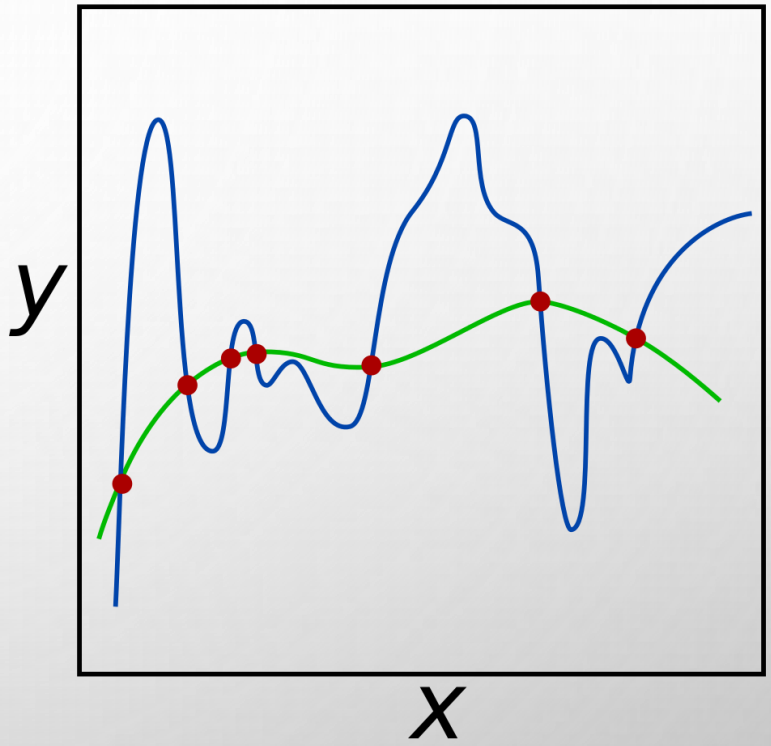
Although regularization procedures can be divided in many ways, the following delineation is particularly helpful:

- **Explicit regularization** is regularization whenever one explicitly adds a term to the optimization problem. These terms could be priors, penalties, or constraints. Explicit regularization is commonly employed with ill-posed optimization problems. The regularization term, or penalty, imposes a cost on the optimization function to make the optimal solution unique.
- **Implicit regularization** is all other forms of regularization. This includes, for example, early stopping, using a robust loss function, and discarding outliers. Implicit regularization is essentially ubiquitous in modern machine learning approaches, including stochastic gradient descent for training deep neural networks, and ensemble methods (such as random forests and gradient boosted trees).

In explicit regularization, independent of the problem or model, there is always a data term, that corresponds to a likelihood of the measurement and a regularization term that corresponds to a prior. By combining both using Bayesian statistics, one can compute a posterior, that includes both information sources and therefore stabilizes the estimation process. By trading off both objectives, one chooses to be more addictive to the data or to enforce generalization (to prevent overfitting). There is a whole research branch dealing with all possible regularizations. In practice, one usually tries a specific regularization and then figures out the probability density that corresponds to that regularization to justify the choice. It can also be physically motivated by common sense or intuition.

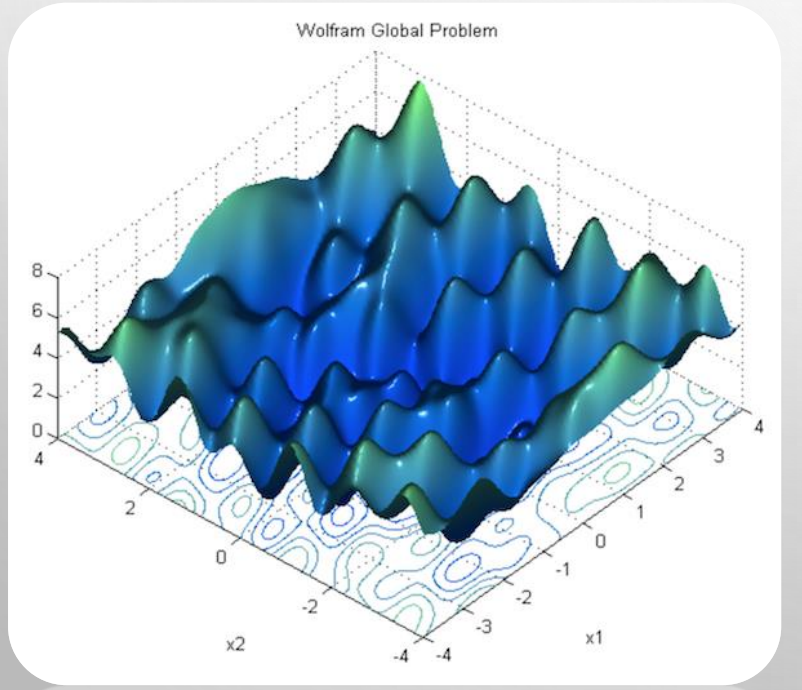
In machine learning, the data term corresponds to the training data and the regularization is either the choice of the model or modifications to the algorithm. It is always intended to reduce the generalization error, i.e. the error score with the trained model on the evaluation set and not the training data.[3]

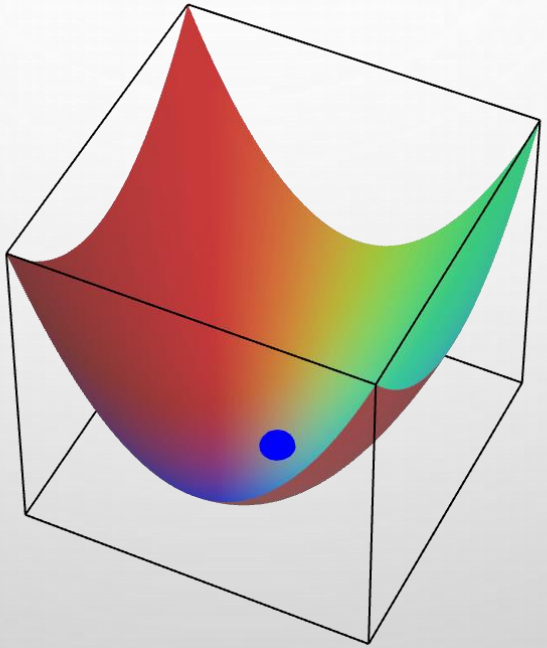
One of the earliest uses of regularization is Tikhonov regularization (ridge regression), related to the method of least squares.

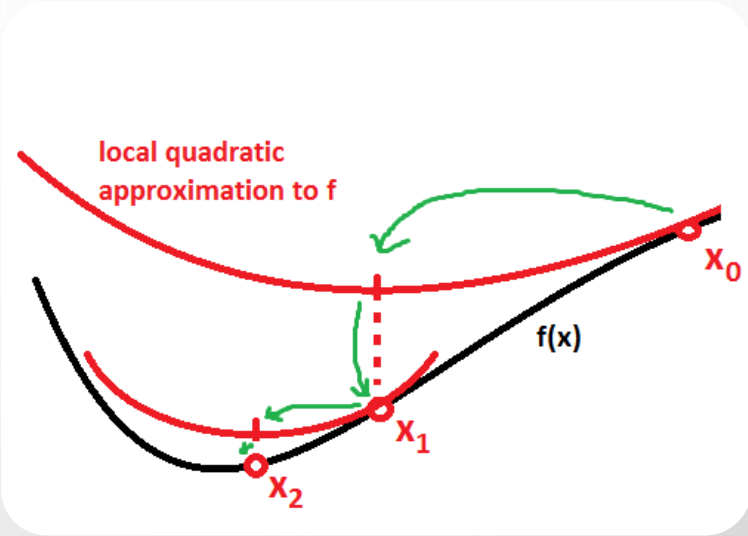


SUPERFÍCIE DO ERRO MÉDIO QUADRÁTICO

$$E = \frac{1}{N}\sum (y - f(w, x))^2$$







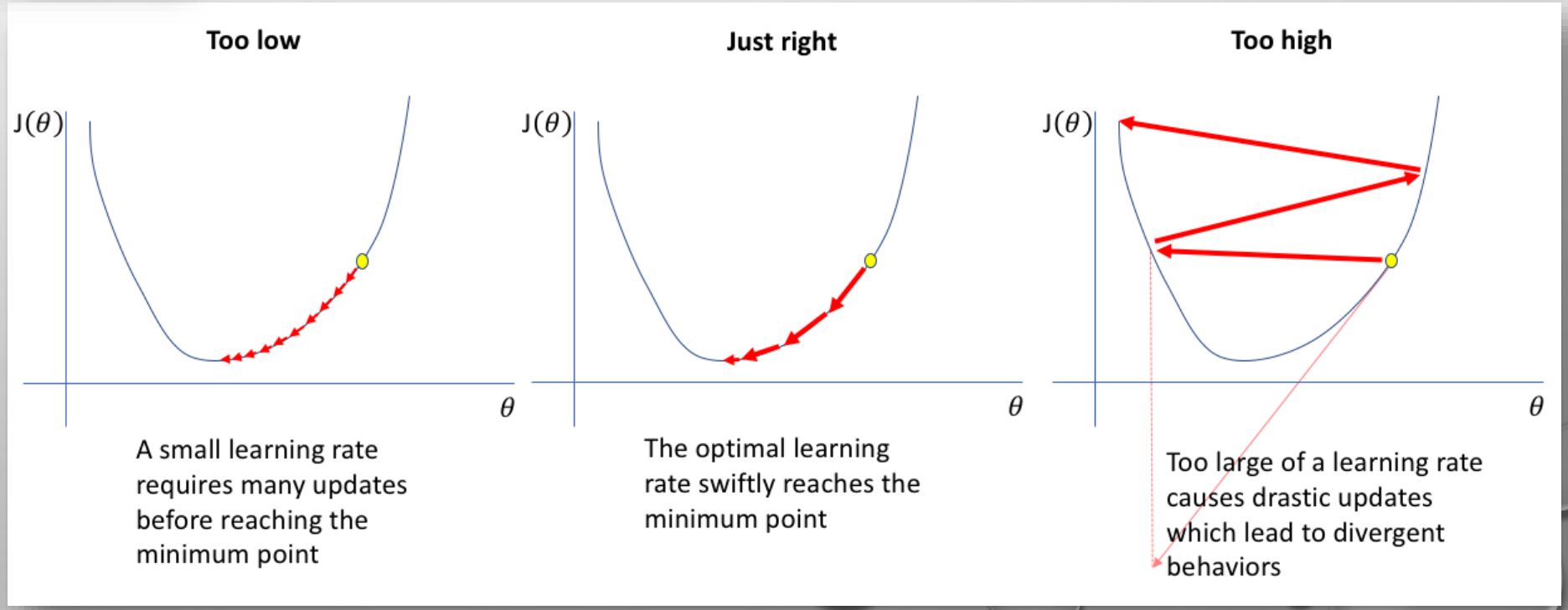
ALGORITMO DO GRADIENTE DESCENDENTE

$$\Delta w_{ij} = (\eta * \frac{\partial E}{\partial w_{ij}})$$

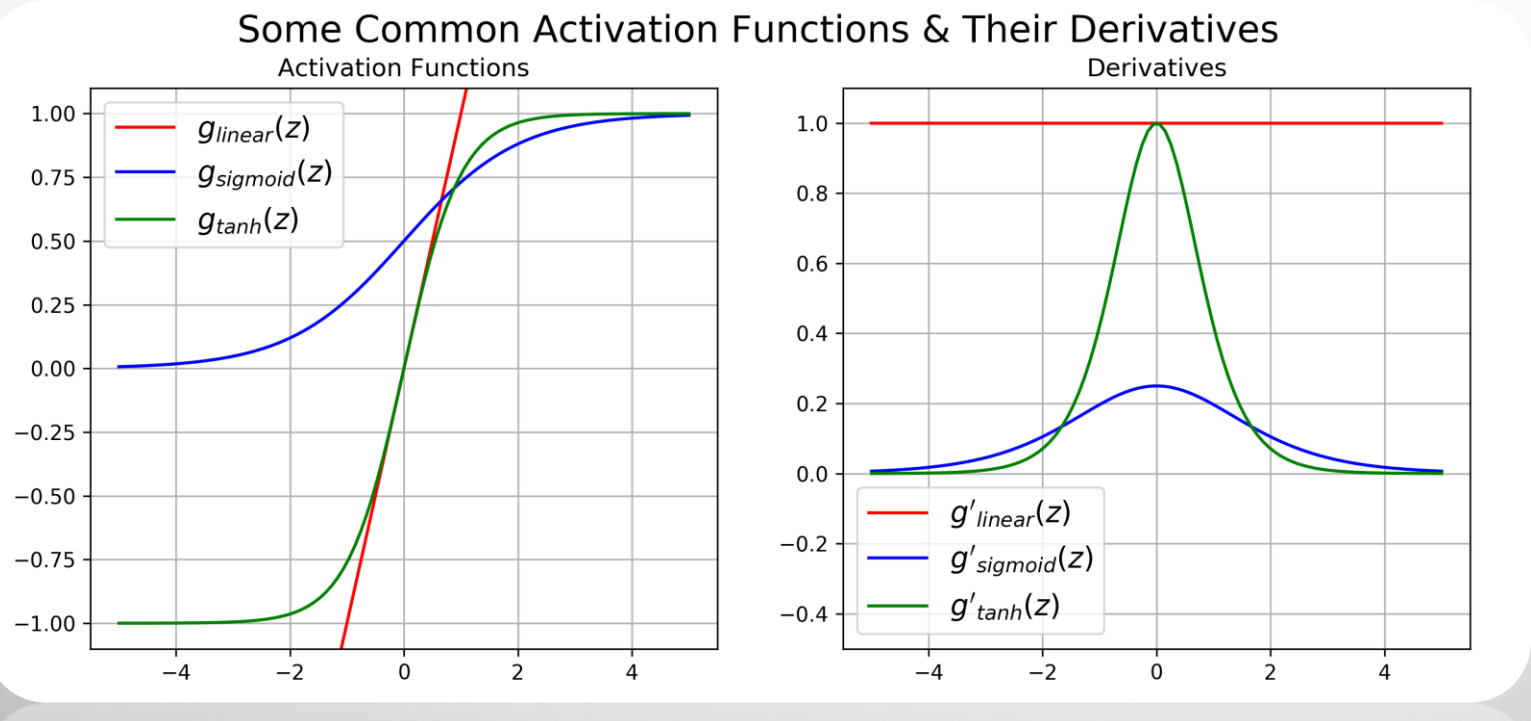
weight increment — learning rate — weight gradient

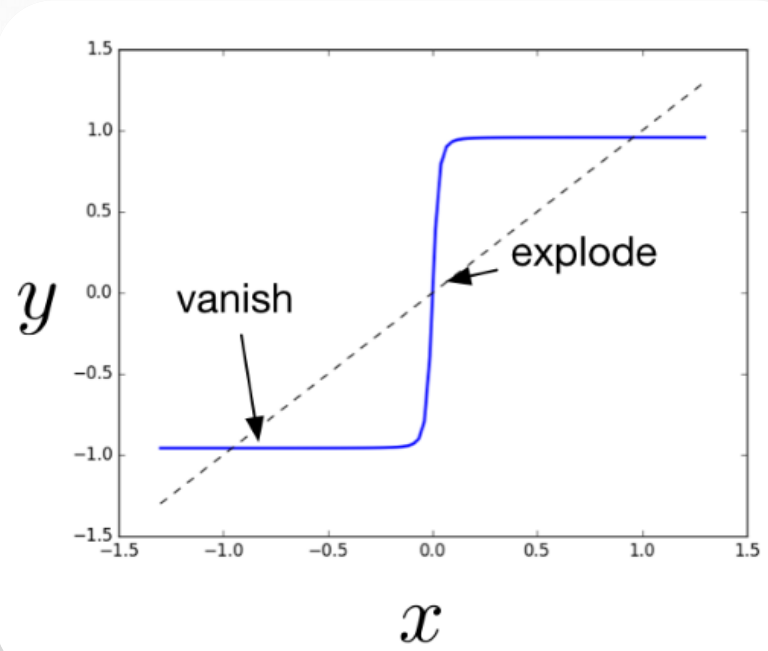
$$\Delta w_{ij} = (\eta * \frac{\partial E}{\partial w_{ij}}) + (\gamma * \Delta w_{ij}^{t-1})$$

momentum factor — weight increment, previous iteration



O PROBLEMA DA DISSIPAÇÃO DO GRADIENTE





OTIMIZADORES (REGULARIZADOS)

$$v_t^w = v_{t-1}^w + (\nabla w_t)^2$$
$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t^w + \epsilon}} * \nabla w_t$$

$$v_t^b = v_{t-1}^b + (\nabla b_t)^2$$
$$b_{t+1} = b_t - \frac{\eta}{\sqrt{v_t^b + \epsilon}} * \nabla b_t$$

ADAGRAD

$$v_t^w = \beta * v_{t-1}^w + (1-\beta)(\nabla w_t)^2$$
$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t^w + \epsilon}} * \nabla w_t$$

$$v_t^b = \beta * v_{t-1}^b + (1-\beta)(\nabla b_t)^2$$
$$b_{t+1} = b_t - \frac{\eta}{\sqrt{v_t^b + \epsilon}} * \nabla b_t$$

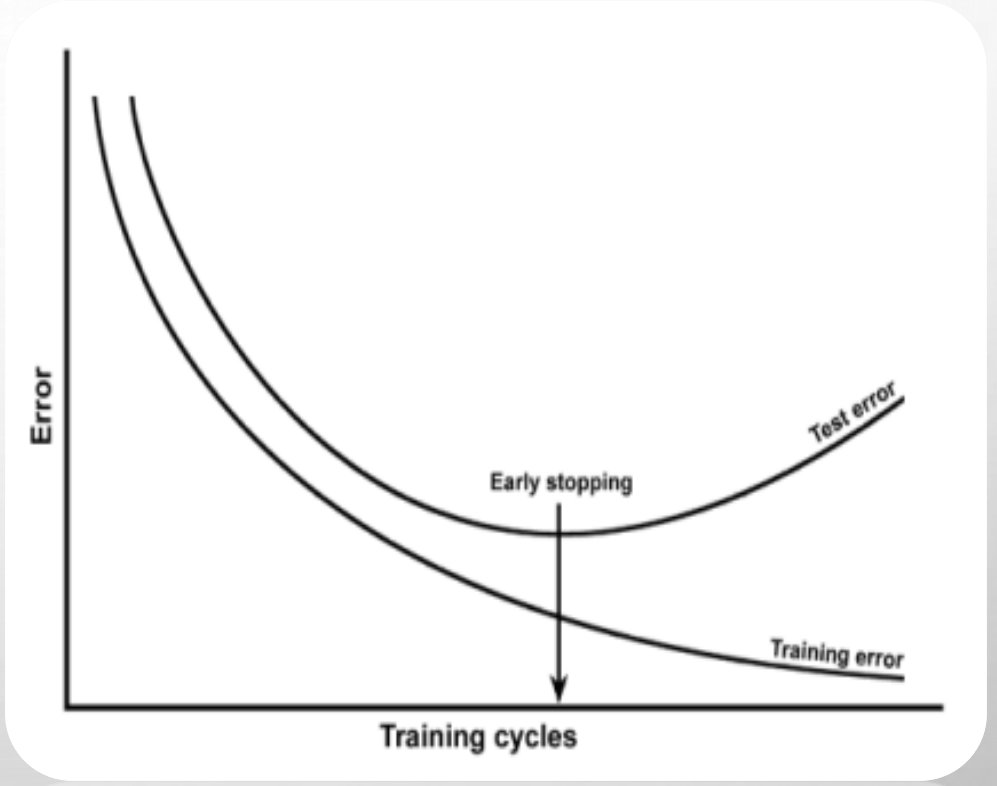
RMSPROP

$$m_t = \beta_1 * m_{t-1} + (1-\beta_1) * \nabla w_t$$
$$v_t = \beta_2 * v_{t-1} + (1-\beta_2) * (\nabla w_t)^2$$
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \qquad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$
$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} * \hat{m}_t$$

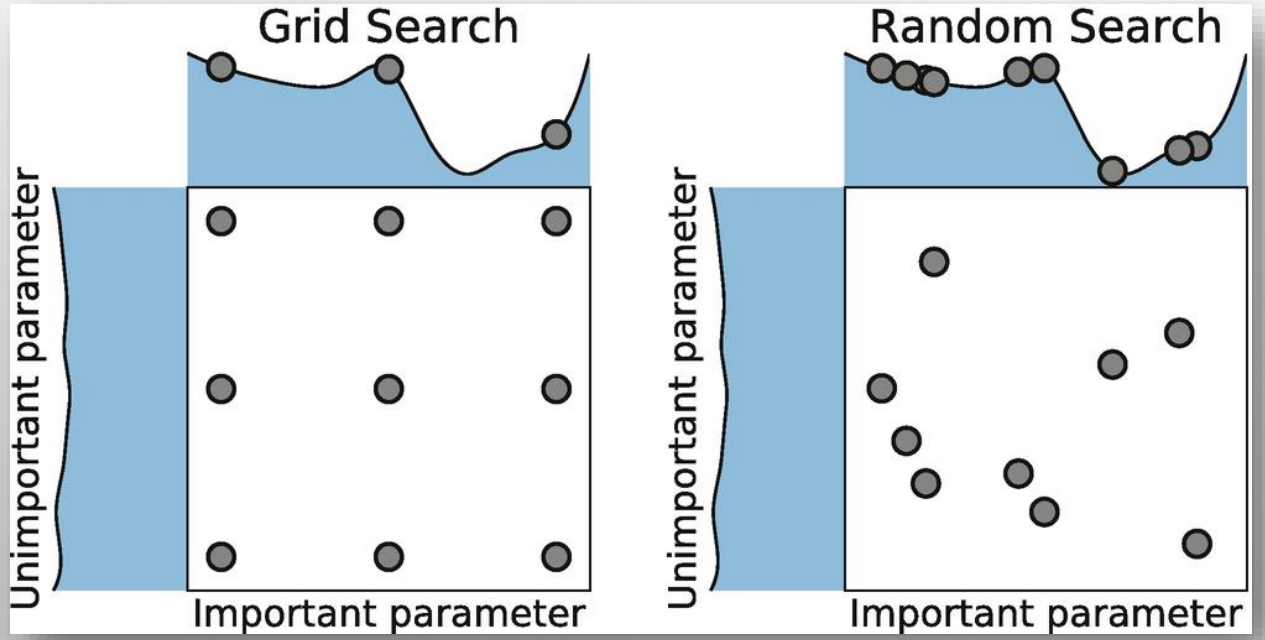
ADAM

PARADA PREMATURA DO TREINAMENTO

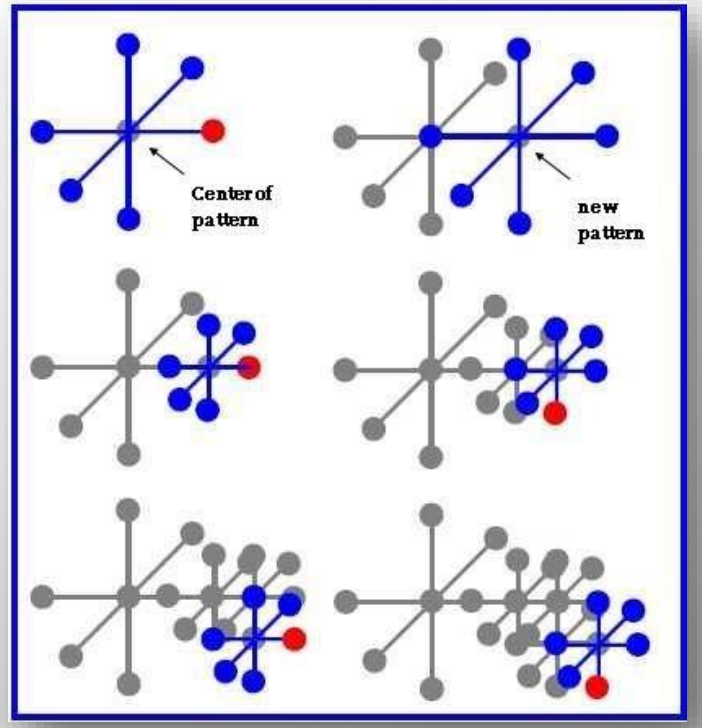
- Aumento no Erro de Validação (Teste)
- Estabilidade da Figura de Mérito no Treino



OTIMIZAÇÃO DE HIPERPARÂMETROS



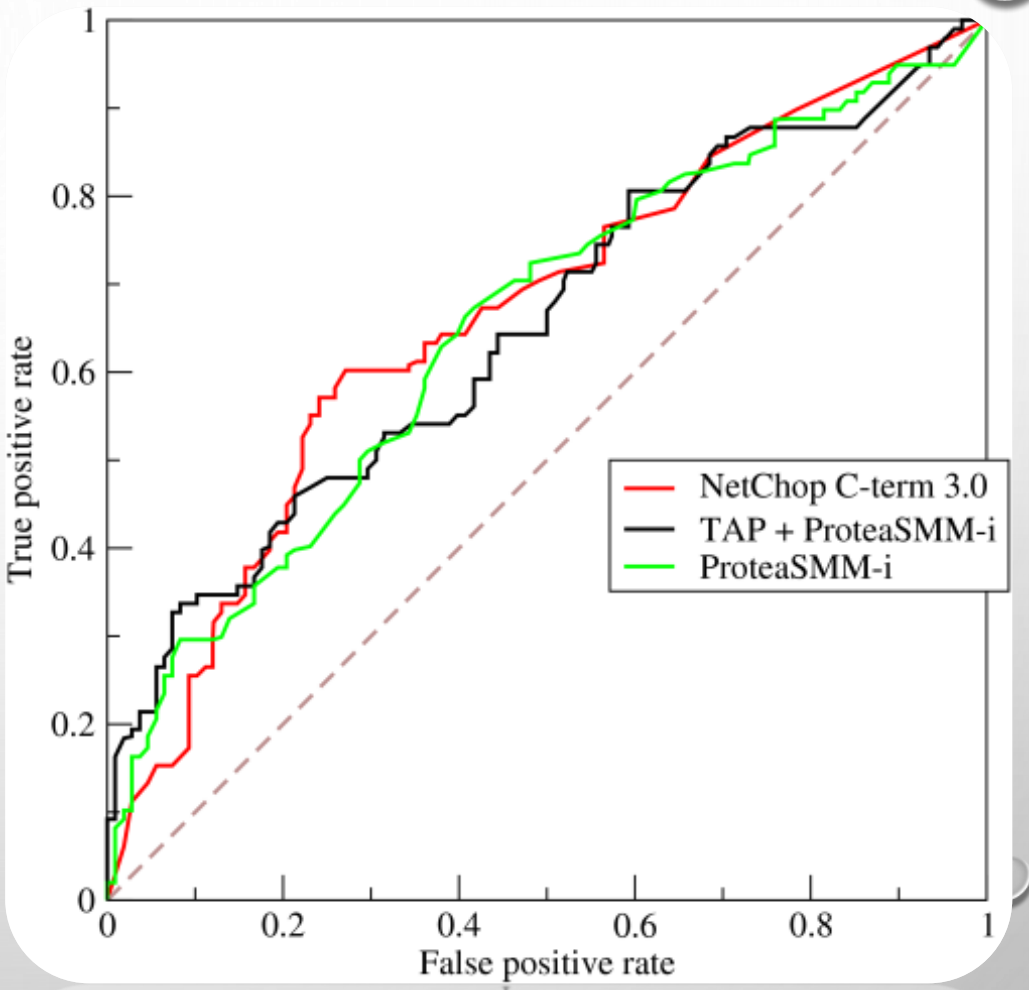
GRID SEARCH



PATTERN SEARCH

PONTO DE OPERAÇÃO

- **Curva ROC**
  - Calibra a saída do modelo, ajudando a configurar o ponto de operação entre Precisão / Recall / Acurácia.

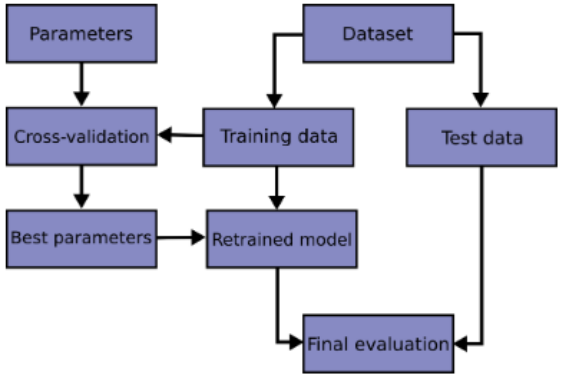


EXEMPLO DE CICLO DE TREINAMENTO

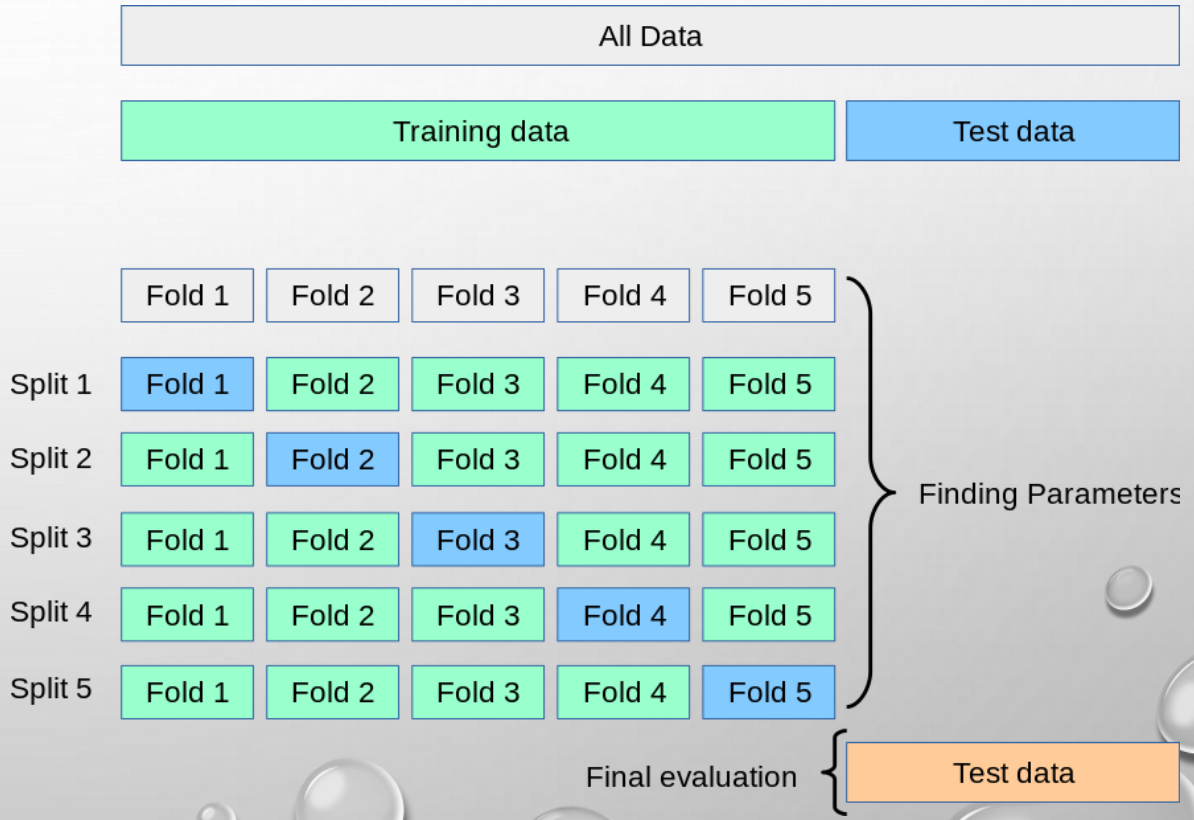
EVALUATION

3.1. Cross-validation: evaluating estimator performance

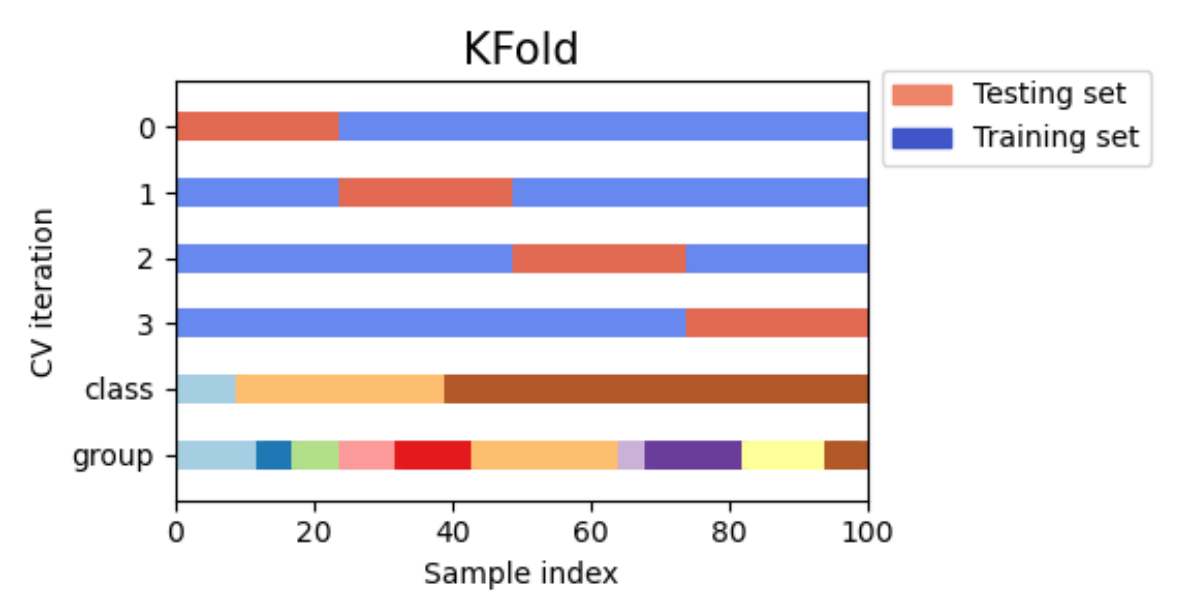
Learning the parameters of a prediction function and testing it on the same data is a methodological mistake: a model that would just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on yet-unseen data. This situation is called **overfitting**. To avoid it, it is common practice when performing a (supervised) machine learning experiment to hold out part of the available data as a **test set** `X_test, y_test`. Note that the word "experiment" is not intended to denote academic use only, because even in commercial settings machine learning usually starts out experimentally. Here is a flowchart of typical cross validation workflow in model training. The best parameters can be determined by grid search techniques.

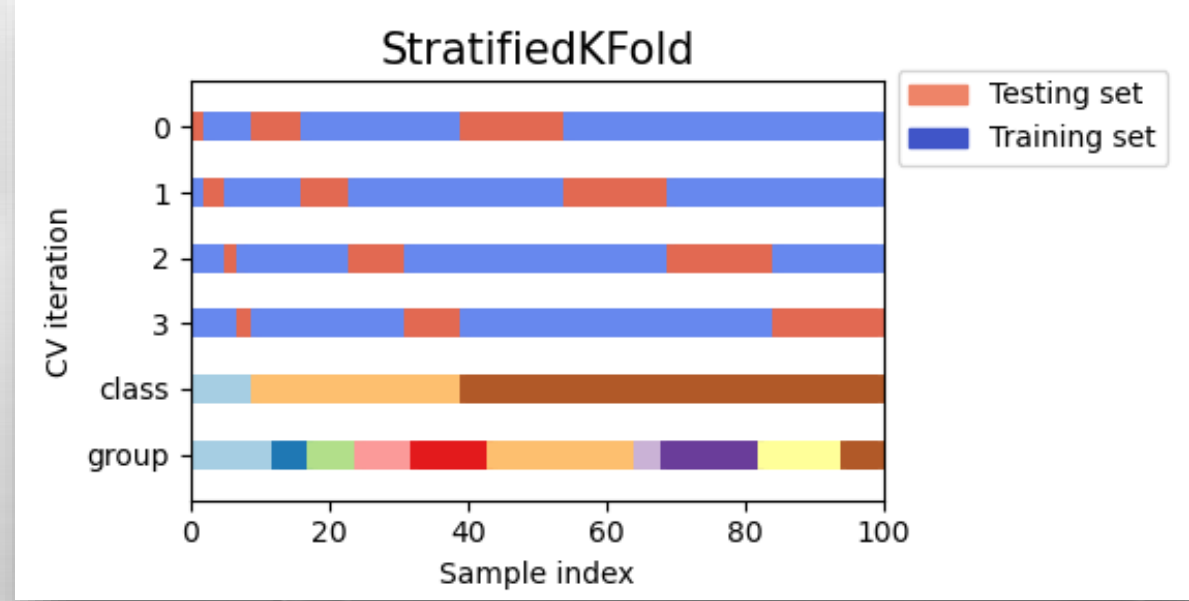


VALIDAÇÃO CRUZADA



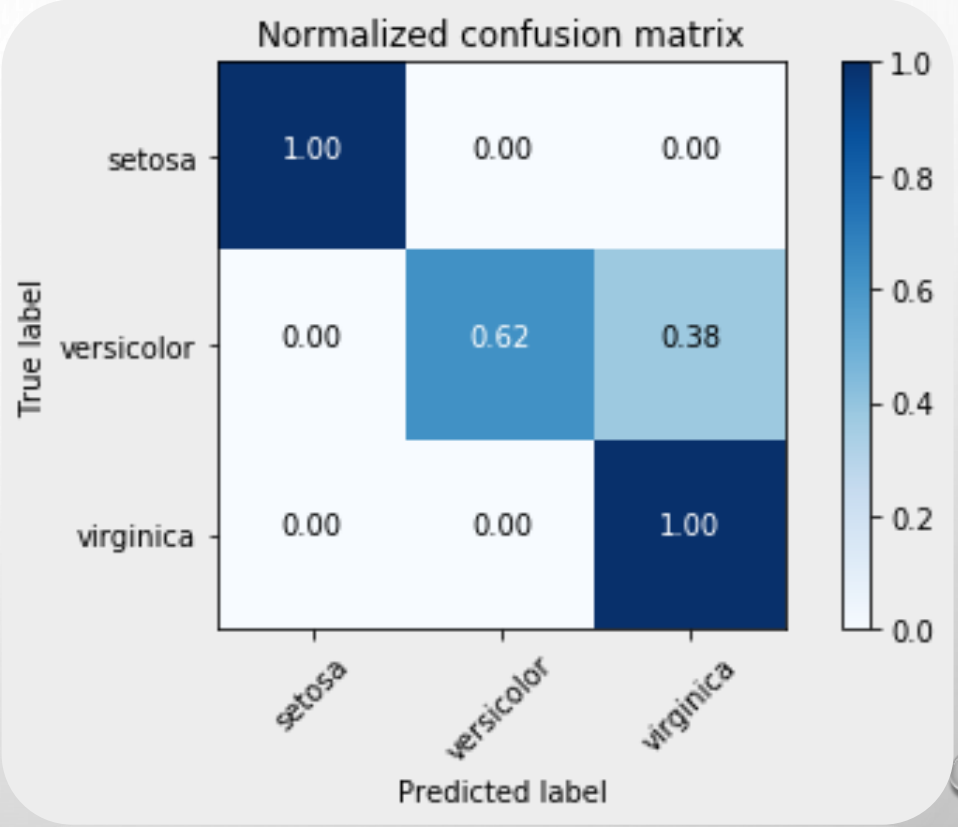
K-FOLDS & K-FOLDS ESTRATIFICADO



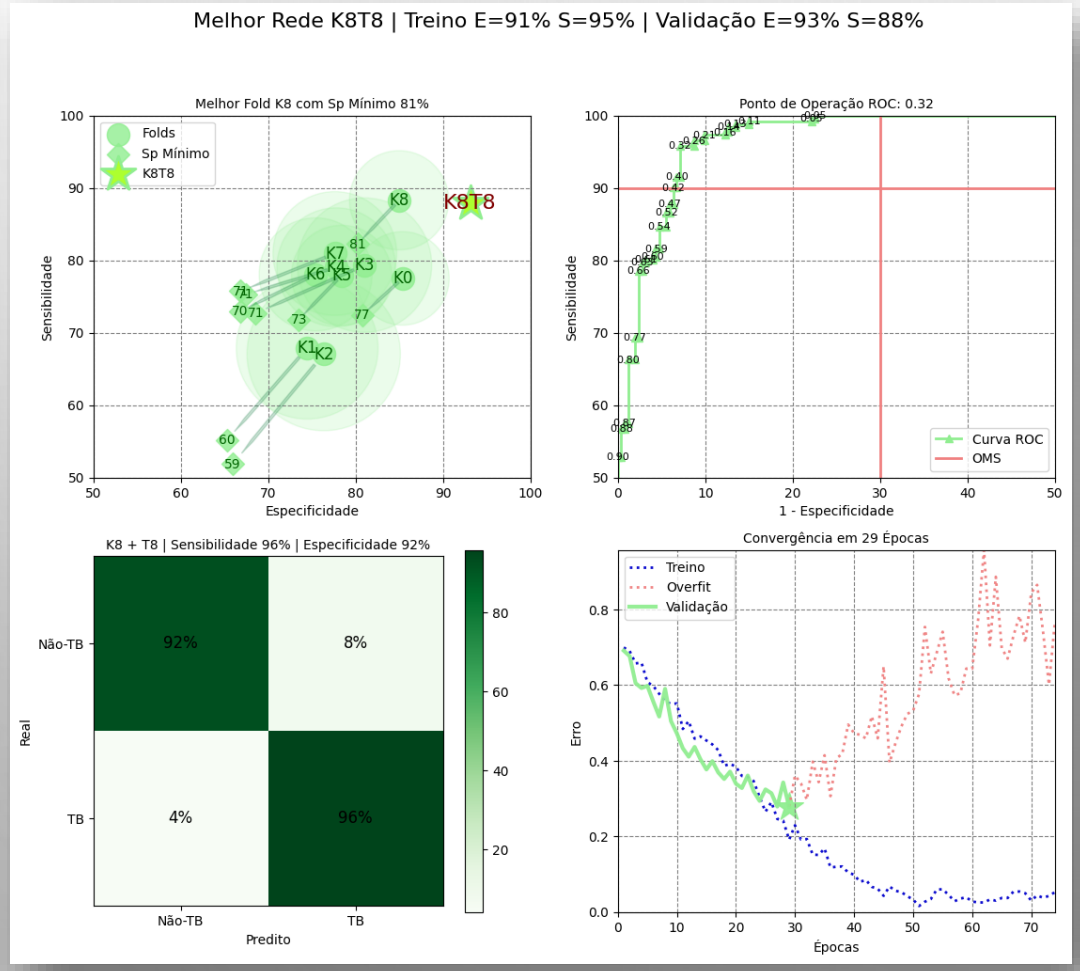


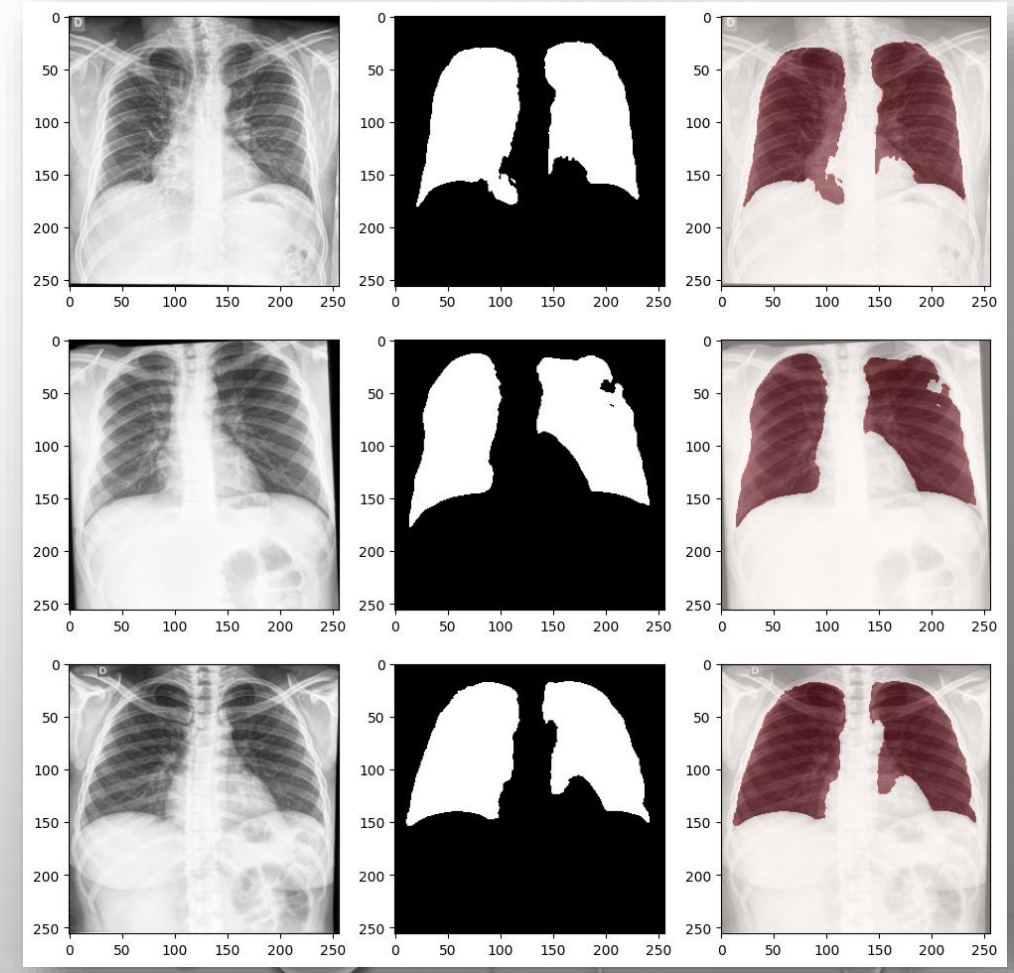
MATRIZ DE CONFUSÃO

Comparação entre o resultado do classificador para as diferentes classes.



VALIDAÇÃO DO TREINAMENTO : SEGMENTADOR DE PULMÕES





PARTE 2 : PRÁTICA

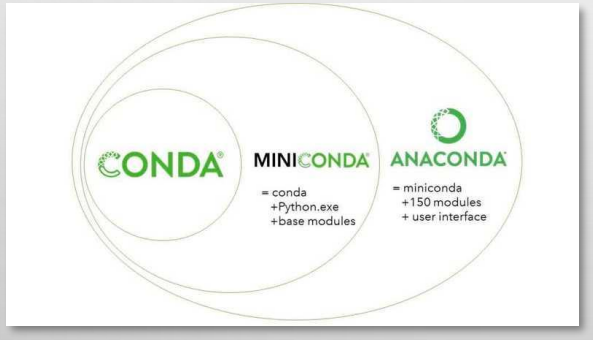
AMBIENTE PYTHON

5. Visualização

6. Machine Learning

4. Variáveis Aleatórias

1. Editor de Código



2. Gestor de Ambiente

3. Ambiente Python do Projeto

3. Notebook Dinâmico

PROBLEMA DE NEGÓCIO

Características das flores

Largura & comprimento da pétala

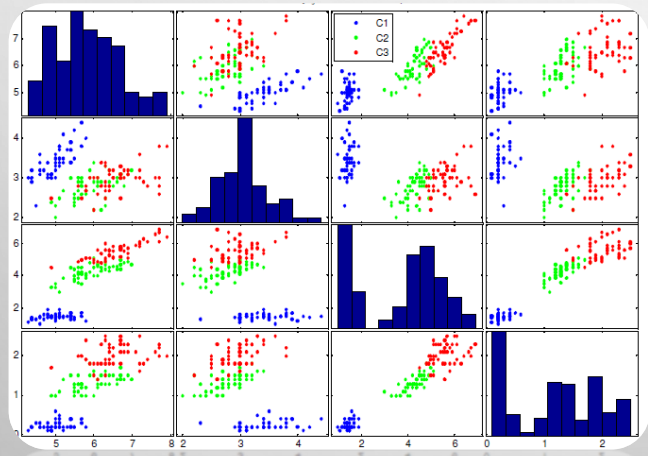
Largura & comprimento da sépala

Iris Setosa

Iris Versicolor

Iris Virginica

REPRESENTAÇÃO



http://archive.ics.uci.edu/ml/datasets/Iris

Características das flores

Largura & comprimento da pétala

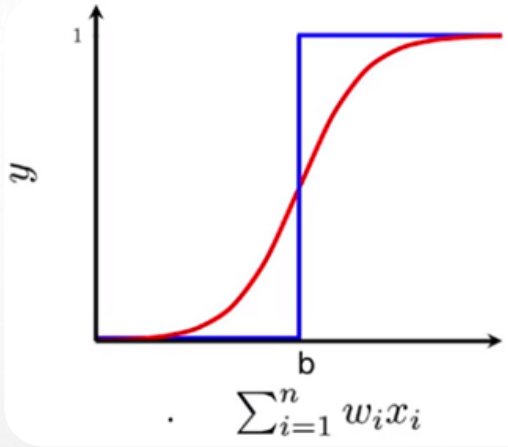
Largura & comprimento da sépala

Espaço de atributos com **4 dimensões!**

Iris Setosa

Iris Versicolor

Iris Virginica



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

CLASSIFICADOR IRIS

PRIMEIROS EXPERIMENTOS

Algoritmo	Representação	Preparação	Modelagem	Validação
• Reta 2 Pontos • NN 10% VAL • NN 5 Folds • Pattern Search 5 Folds	• 2D • 2D • 2D • 4D / 3 Classes	• Nenhuma • Nenhuma • Scale • Scale	• Reta 2 Pontos • NN Básica • NN Hidden • NN Hidden	• Nenhuma • Precisão/Recall • Precisão/Recall • Acurácia

- Garantir estabilidade no treinamento
- Chegar a mesma solução independente do experimento
- Garantir Generalização

- Modelo Multiclasse
- Busca nos hiperparâmetros ótimos (# funções de ativação)
- Identificar as variáveis mais relevantes

PRÓXIMA AULA: WORKSHOP CLASSIFICAÇÃO