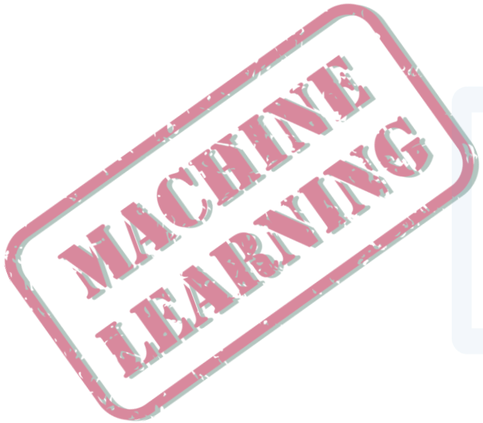
EPI5717: Machine learning para predições em saúde

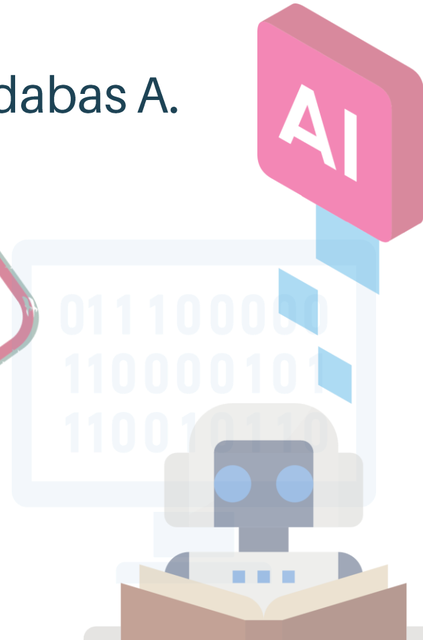
Aula 6

Prof. Dr. Alexandre Chiavegatto Filho

COMPARISON OF GRADIENT BOOSTING DECISION TREE ALGORITHMS FOR CPU PERFORMANCE

Authors: Al-Shari H, Saleh YA, Odabas A.





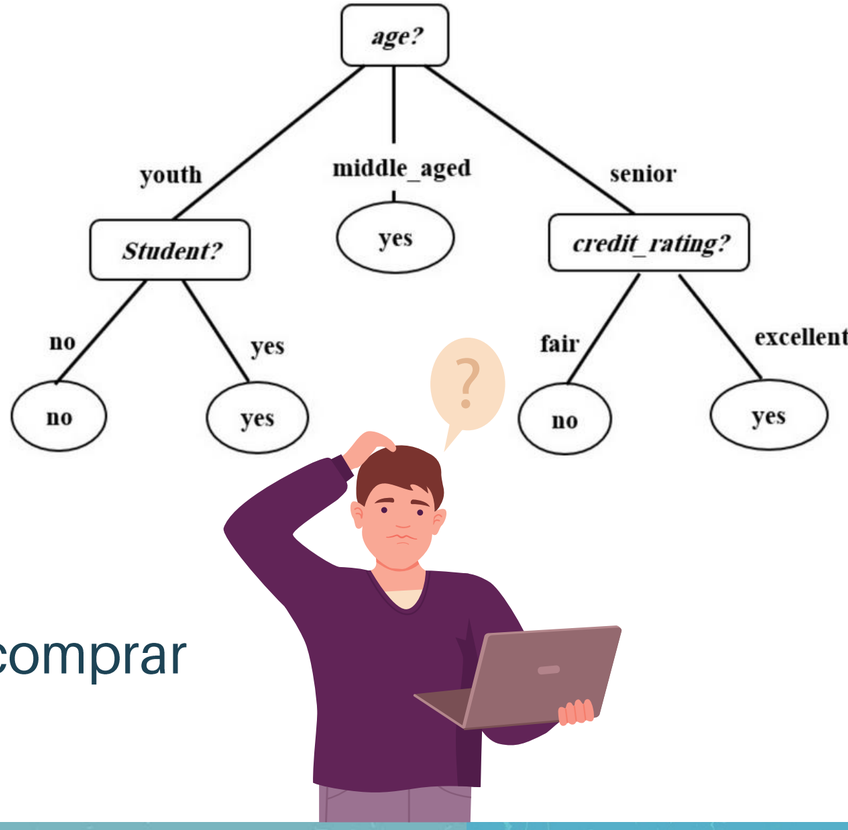
Resumo

- Algoritmos de Gradient Boosting Decision Trees (GBDT)
- XGBoost (mais popular)
- GBDT ganhou notoriedade em competições no Kaggle

- **Objetivo**: comparar a implementação dos 3 principais algoritmos de gradient boosting: XGboost, LightGBM e CatBoost em 5 problemas de classificação e regressão.

Como esses algoritmos funcionam?

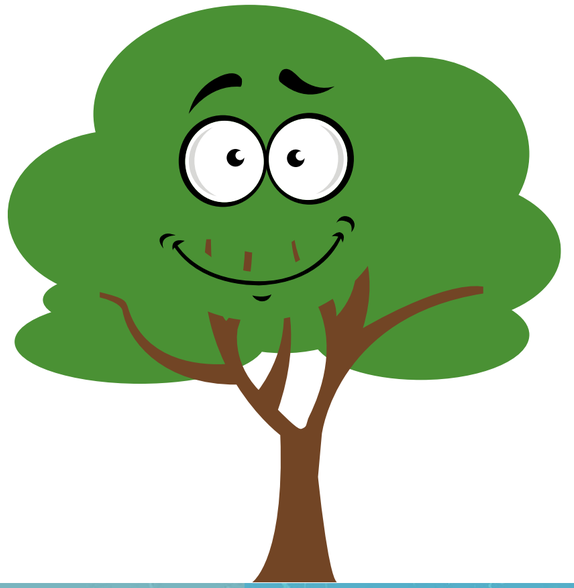
- Árvore de decisão é um método de ML que usa um gráfico semelhante a uma árvore
- O nó interno representa um teste em uma variável
- Cada ramo divide os dados em dois grupos
- As observações são atribuídas ao nó adequado
- O resultado da folha é a predição final



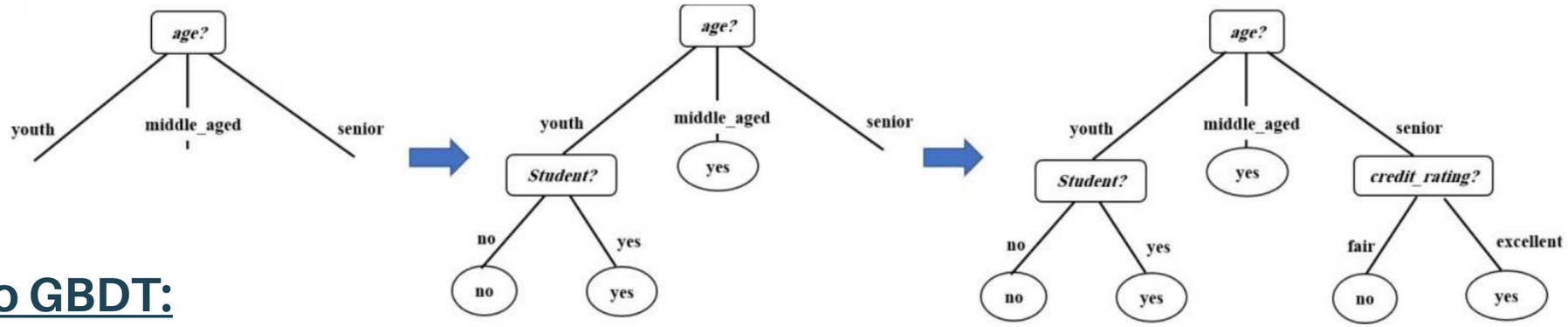
Exemplo: comprar um computador ou não

A predição do modelo mostra que um jovem estudante vai comprar

- Flexíveis e fáceis de interpretar
- Porém, usar apenas a árvore de decisão pode levar a sobreajuste do modelo
- Solução: Ensemble
- Ensemble = várias algoritmos combinados (método que generaliza bem)
- Existem formas diferentes de combinar as árvores
- GBDT é uma delas



REALIDADE							
PREDITOR 1							
PREDITOR 2							
PREDITOR 3							
PREDITOR 4							
PREDITOR 5							
COMITÊ							



Ideia do GBDT:

- Treina várias árvores sequenciais
- O resultado predito será a soma das predições de todas as árvores
- Treinamento iterativo = constrói as árvores uma por uma
- Objetivo do treinamento: minimizar uma função objetivo por meio da divisão repetitiva dos dados para maximizar algum critério até algum limite (ex.: profundidade da árvore)
- A árvore seguinte é construída de modo a minimizar a função de perda, e suas saídas são adicionadas a primeira árvore

XGboost, LightGBM e CatBoost são os principais representantes do GBDT pois:

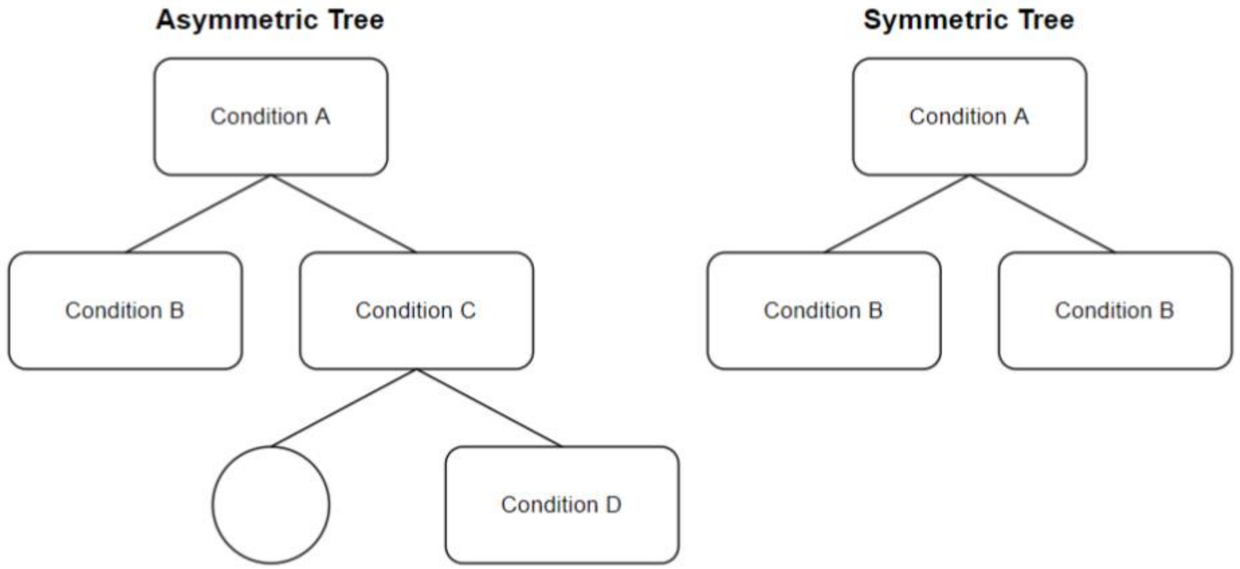
- Implementações fáceis (Windows, macOS e Linux)
- Bibliotecas de software de código aberto (C++, Python e R)
- Rápidos de treinar e resultados precisos

O desenvolvimento da árvore

	CatBoost	LightGBM	XGBoost
Developer	Yandex	Microsoft	DMLC
Release Year	2017	2016	2014
Tree Symmetry	Symmetric	Asymmetric Leaf-wise tree growth	Asymmetric Level-wise tree growth
Splitting Method	Greedy method	Gradient-based One-Side Sampling (GOSS)	Pre-sorted and histogram-based algorithm
Type of Boosting	Ordered	-	-
Numerical Columns	Support	Support	Support
Categorical Columns	Support Perform one-hot encoding (default) Transforming categorical to numerical columns by border, bucket, binarized target mean value, counter methods available	Support, but must use numerical columns Can interpret ordinal category	Supports, but must use numerical columns Cannot interpret ordinal category, users must convert to one-hot encoding, label encoding or mean encoding
Text Columns	Support Support Bag-of-Words, Naïve-Bayes or BM-25 to calculate numerical features from text data	Do not support	Do not support
Missing values	Handle missing value Interpret as NaN (default) Possible to interpret as error, or processed as minimum or maximum values	Handle missing value Interpret as NaN (default) or zero Assign missing values to side that reduces loss the most in each split	Handle missing value Interpret as NaN (tree booster) or zero (linear booster) Assign missing values to side that reduces loss the most in each split

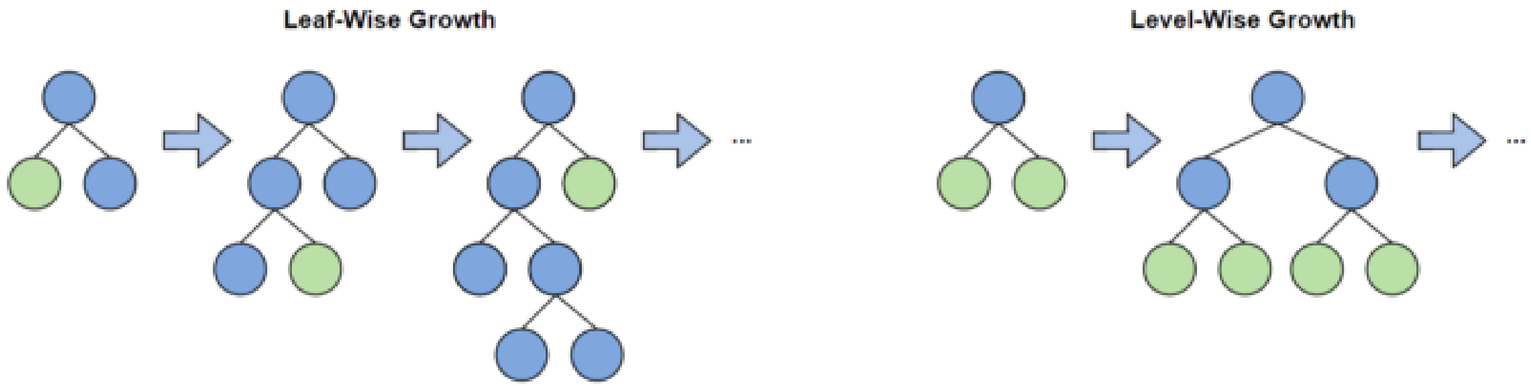
Fonte: https://towardsdatascience.com/catboost-vs-lightgbm-vs-xgboost-c80f40662924

O desenvolvimento da árvore



- XGBoost e LightGBM : árvores assimétricas.
- Catboost: árvores simétricas (no mesmo nível, todos os nós são iguais).

O desenvolvimento da árvore



- O LightGBM cresce as árvores por folhas (horizontais).
- O XGBoost emprega uma estratégia por nível (verticais).

O desenvolvimento da árvore

Estratégia baseada em níveis

Vantagens:

- Mantém a árvore balanceada
- Qualquer árvore construída com essa estratégia pode ser construída com uma estratégia baseada em folhas (o inverso não é verdadeiro).

Estratégia baseada em folhas

Vantagens:

- Melhor para grandes conjuntos de dados
- Estratégia mais flexível
- Pode ter maior profundidade

Semelhança de Hiperparâmetros

Table 1. Table shows the hyperparameters similarities between the three algorithms.

Function	XGBoost	CatBoost	LightGBM
Important arameters which control overtting	**learning_rate or eta:** Optimal values lie between 0.01-0.2. **max_depth** **min_child_weight:** similar to min child leaf; default is 1.	**learning_rate** **depth:** Value can be any integer up to 16. Recommended [1 to 10]. No such feature like min child weight. **L2-leaf-reg:** L2 regularization coefficient is used for the calculation of leaf value (positive integer)	**learning_rate** **max_depth:** default is 20. Important to note that tree still grows leaf-wise. Hence it is important to tune. **num_leaves:** (Number of leaves in tree) which should be smaller than 2^(max depth) it is very important parameter. **min_data_in_leaf:** default = 20.
Parameters for categorical values	Not Available	**Cat_features:** It represents the index of the categorical features. **One_hot_max_size:** Use to specifiy the maximum number of different values while performing one-hot encoding for all features (max - 255)	**categorical-feature:** Use to specify which of the features is categorical during training the model.
Parameters for controlling speed	**colsample_bytree:** subsample ratio of columns. **subsample:** subsample ratio of the training instance. **n_estimators:** maximum number of decision trees; high value can lead to over fitting	**rsm:** Random subspace method. The percentage of features to use at each split selection. No such parameter to subset data. **iterations:** specify the upper limit of the trees that can be built; high value can lead to overfitting.	**feature_fraction:** fraction of features to be taken for each iteration. **bagging_fraction:** specify the data fraction which will be used in each iteration and it is used to avoid overfitting and speed up the training. **Num_iterations:** specify how many boosting iterations to be performed; default = 100.

Conjuntos de dados

- **The US Adult Census Dataset (Classificação)**

EUA 1994 | 48.000 observações | 15 variáveis | Prever a renda do cidadão

- **WHO Life Expectancy Dataset (Regressão)**

193 países | Dados econômicos | 2.938 observações | 22 variáveis

- **Data Hackathon 3.x (Classificação)**

Dados de clientes de bancos da Índia | 87.020 observações | 26 variáveis

- **Bank Customer Churn (Classificação)**

Dados de clientes de bancos da Europa| 10.000 observações | 14 variáveis

- Framingham (**Classificação**)

EUA 1948 | 4.238 observações | 15 variáveis

Critérios para avaliação do desempenho

Acurácia

Velocidade

Confiabilidade

Facilidade de uso

- **Algoritmos testados com hiperparâmetros padrão e ajustados**
- **Python**

Resultados

Table 3. The US Adult Census Dataset's accuracy and speed results of the four algorithms.

	XGBoost		**XGBoost - hist**	
	Default	**Tuned**	**Default**	**Tuned**
Speed	01.104954 Sec	13.84216 Sec	00.367410 Sec	01.940795 Sec
Accuracy	86.794423	87.599042	86.788281	87.777164
	LightGBM		**CatBoost**	
	Default	Tuned	Default	Tuned
Speed	00.361316 Sec	00.653143 Sec	94.158556 Sec	27.683200 Sec
Accuracy	87.341073	87.660463	87.482341	87.580615

Resultados

Table 4. WHO Life Expectancy Dataset's accuracy and speed results of the four algorithms.

	XGBoost		**XGBoost - hist**	
	Default	**Tuned**	**Default**	**Tuned**
Speed	00.943869 Sec	08.87374 Sec	00.351246 Sec	06.529343 Sec
Accuracy	94.248202	96.579688	94.199483	96.559307
	LightGBM		**CatBoost**	
	Default	**Tuned**	**Default**	**Tuned**
Speed	00.311644 Sec	01.374218 Sec	36.196142 Sec	17.62043 Sec
Accuracy	94.359751	95.877220	92.017064	96.132491

Resultados

Table 5. Data Hackathon 3.x dataset's accuracy and speed results of the four algorithms.

	XGBoost		**XGBoost - hist**	
	Default	**Tuned**	**Default**	**Tuned**
Speed	01.782270 Sec	00.864689 Sec	00.465752 Sec	00.403920 Sec
Accuracy	98.501494	98.501494	98.501494	98.501494
	LightGBM		**CatBoost**	
	Default	**Tuned**	**Default**	**Tuned**
Speed	00.813822 Sec	00.395931 Sec	68.196164 Sec	33.320850 Sec
Accuracy	98.483107	98.501494	98.501494	98.501494

Resultados

Table 6. Bank Customer Churn Dataset's accuracy and speed results of the four algorithms.

	XGBoost		**XGBoost - hist**	
	Default	**Tuned**	**Default**	**Tuned**
Speed	00.75184 Sec	00.71689 Sec	00.300096 Sec	00.727746 Sec
Accuracy	87.15	87.15	86.90	86.95
	LightGBM		**CatBoost**	
	Default	**Tuned**	**Default**	**Tuned**
Speed	00.194973 Sec	00.113583 Sec	33.449706 Sec	09.558820 Sec
Accuracy	86.00	86.85	86.65	87.20

Resultados

Table 7. Framingham Heart Study Dataset's accuracy and speed results of the four algorithms.

	XGBoost		**XGBoost - hist**	
	Default	**Tuned**	**Default**	**Tuned**
Speed	00.173532 Sec	00.122078 Sec	00.150508 Sec	00.98946 Sec
Accuracy	85.519126	85.792350	85.382514	85.519126
	LightGBM		**CatBoost**	
	Default	**Tuned**	**Default**	**Tuned**
Speed	00.146186 Sec	00.150189 Sec	52.861558 Sec	10.478870 Sec
Accuracy	84.562842	85.519126	85.245902	85.792350

Conclusão

- LightGBM tem desempenho mais equilibrado
- Ótima velocidade (mais rápido em 3 conjuntos de dados)
- Teve um resultado consistente em todos os conjuntos de dados
- CatBoost: mais fácil de usar quando se tem dados categóricos
- Mas é o algoritmo mais lento comparado aos demais

Referência:

ALSHARI, Haithm Haithm; SALEH, Abdulrazak Yahya; ODABAŞ, Alper.
Comparison of Gradient Boosting Decision Tree Algorithms for CPU Performance.

Journal of Institue Of Science and Technology, v. 37, n. 1, 2021.

ÁRVORES DE DECISÃO

Maior problema das árvores de decisão simples são a sua forte tendência ao sobreajuste (sensíveis a pequenas variações nos dados).

- Uma solução é o uso de bagging (bootstrap aggregation).

Bagging: selecionar amostras aleatórias dos dados de treino.

Cada observação tem a mesma probabilidade de ser sorteada (com reposição).

- Mesmo tamanho dos dados originais, então cada observação tem 63,2% de probabilidade de ser sorteada em cada rodada.

ÁRVORES DE DECISÃO

- Cada amostra de bootstrap é utilizada para definir a sua árvore.
- Árvores são definidas sem regularização (complexas).
- No caso de regressão, a predição final de uma observação é a sua média de todas as árvores.
- No caso de classificação, a predição final de uma observação é a sua categoria mais predita em todas as árvores (probabilidade é a proporção de votos).
- É considerada uma técnica de ensemble por agregar os resultados de vários algoritmos (de árvore).

ÁRVORES DE DECISÃO

- O problema do uso de bagging é o custo computacional (porém como as árvores são independentes é fácil paralelizar a análise).

- Pode envolver centenas ou milhares de árvores.

- Uma outra limitação importante é que a facilidade de interpretação, um dos principais pontos fortes das árvores de decisão, foi perdida.

ÁRVORES DE DECISÃO

Apesar do ganho de variância ao não utilizar determinadas observações em algumas árvores, todos os preditores são analisados em todos os nós:

- Alguns preditores podem tem efeito desproporcional e indesejado nos resultados (utilizados muitas vezes em todas as árvores).
  - Maior risco de sobreajuste e semelhança entre as árvores.
- Alto tempo de treinamento.

Algoritmos de

MACHINE LEARNING

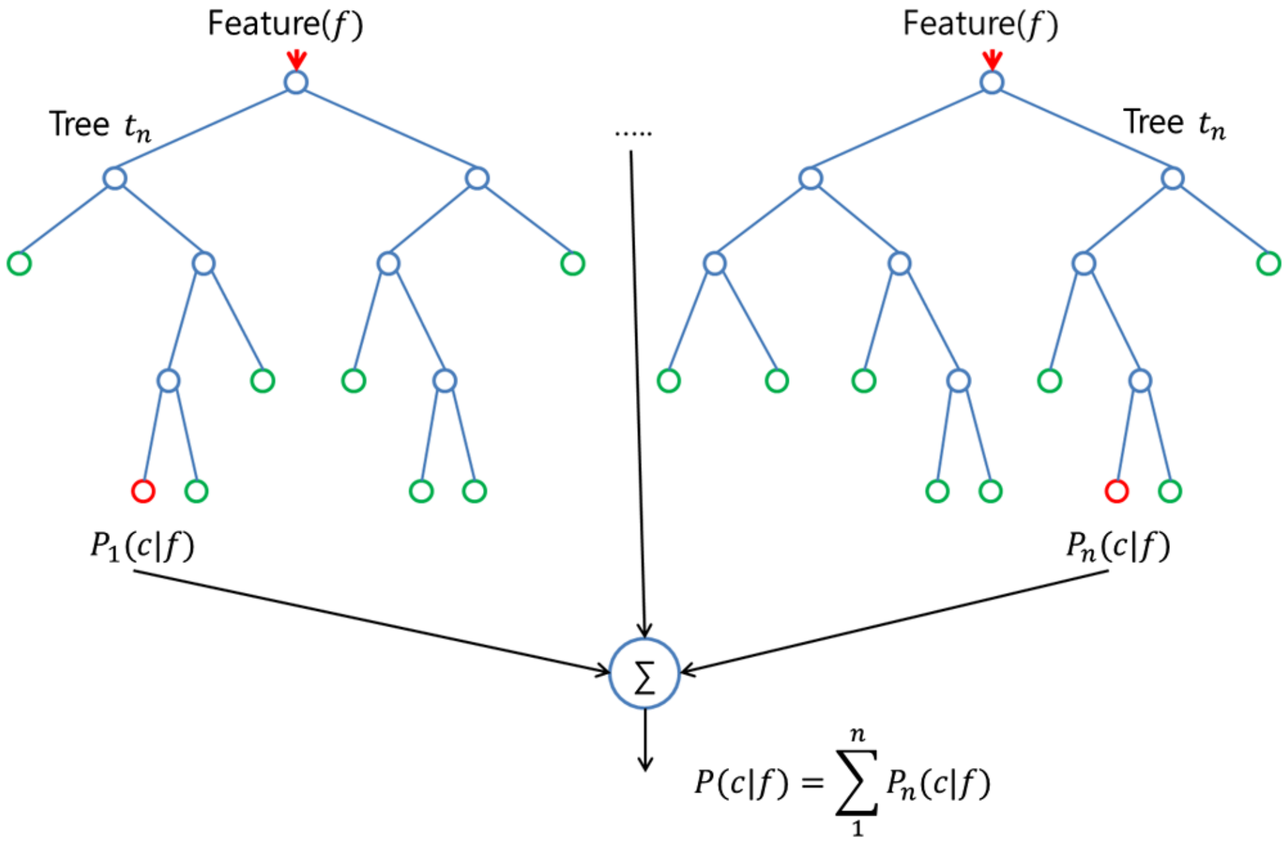
RANDOM FORESTS

- Redução da correlação entre as árvores por meio do uso de um subgrupo aleatório das variáveis preditoras.
- Bagging + seleção de preditores em cada nós = random forests.
- O principal regularizador das random forests é o número de preditores analisados em cada nó (em geral, a raiz quadrada do total de preditores).

Algoritmos de

MACHINE LEARNING

RANDOM FORESTS



REGRESSÕES PENALIZADAS

Tanto a regressão linear (para desfecho contínuo) quanto a regressão logística (para desfecho categórico) são também utilizadas em machine learning.

São mais comuns em estudos de inferência, mas também geram uma predição.

Predição de índice glicêmico.

$$Y_i = \beta_0 + \beta_1 * X_{1i} + \beta_2 * X_{2i}$$

$$Y_{glicemia} = 1{,}23 + 1{,}35 * X_{idade} + 0{,}91 * X_{dieta}$$

Algoritmos de

MACHINE LEARNING

REGRESSÕES PENALIZADAS

Modelos facilmente interpretáveis.

Problema: em geral esses modelos têm sobreajuste quando há muitas variáveis preditoras, principalmente se forem colineares (baixo viés e alta variância).

Solução: adicionar hiperparâmetros regularizadores.

- Penalização contra a complexidade dos modelos.
- Forçar o aumento do viés.
- Pode ajudar a diminuir a variância e o erro de teste

Algoritmos de

MACHINE LEARNING

REGRESSÕES PENALIZADAS

- Adicionar uma penalização se os parâmetros (β) ficarem muito altos.

- Na regressão linear, o objetivo é encontrar os β que minimizem a soma dos erros quadráticos.

$$\text{SSE} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

REGRESSÕES PENALIZADAS

- É possível controlar (regularizar) o tamanho dos coeficientes pela adição de uma penalização à formula anterior.

- Nesse caso é uma penalização L_2 , ou seja, quadrática nos parâmetros.
- A consequência é que agora estamos tentando minimizar o erro e o tamanho dos parâmetros.

$$\mathrm{SSE}_{L_2} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{P} \beta_j^2$$

REGRESSÕES PENALIZADAS

- Nesse caso é uma penalização L_2 , ou seja, quadrática nos parâmetros

- A consequência é que agora estamos tentando minimizar o erro e o tamanho dos parâmetros

$$\mathrm{SSE}_{L_2} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{P} \beta_j^2$$

Algoritmos de

MACHINE LEARNING

REGRESSÕES PENALIZADAS

$$\text{SSE}_{L_2} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{P} \beta_j^2$$

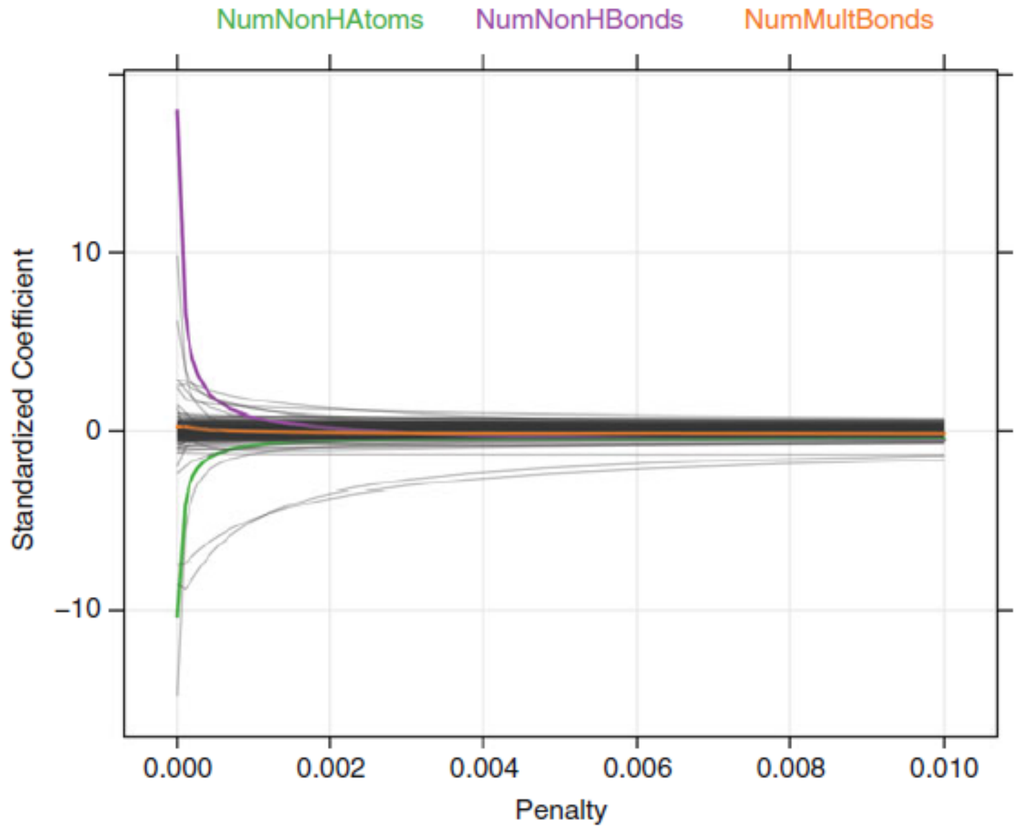
- A quantidade de regularização é controlada pelo parâmetro λ, quanto maior, maior a penalização.
- Ocorre um *encolhimento* dos parâmetros.
- Se λ = 0 não há penalização, regressão comum.
- Não tem encolhimento no $\beta_{0,}$ queremos diminuir os efeitos das variáveis individuais e não do intercepto (média quando todas as variáveis são 0).

Algoritmos de

MACHINE LEARNING

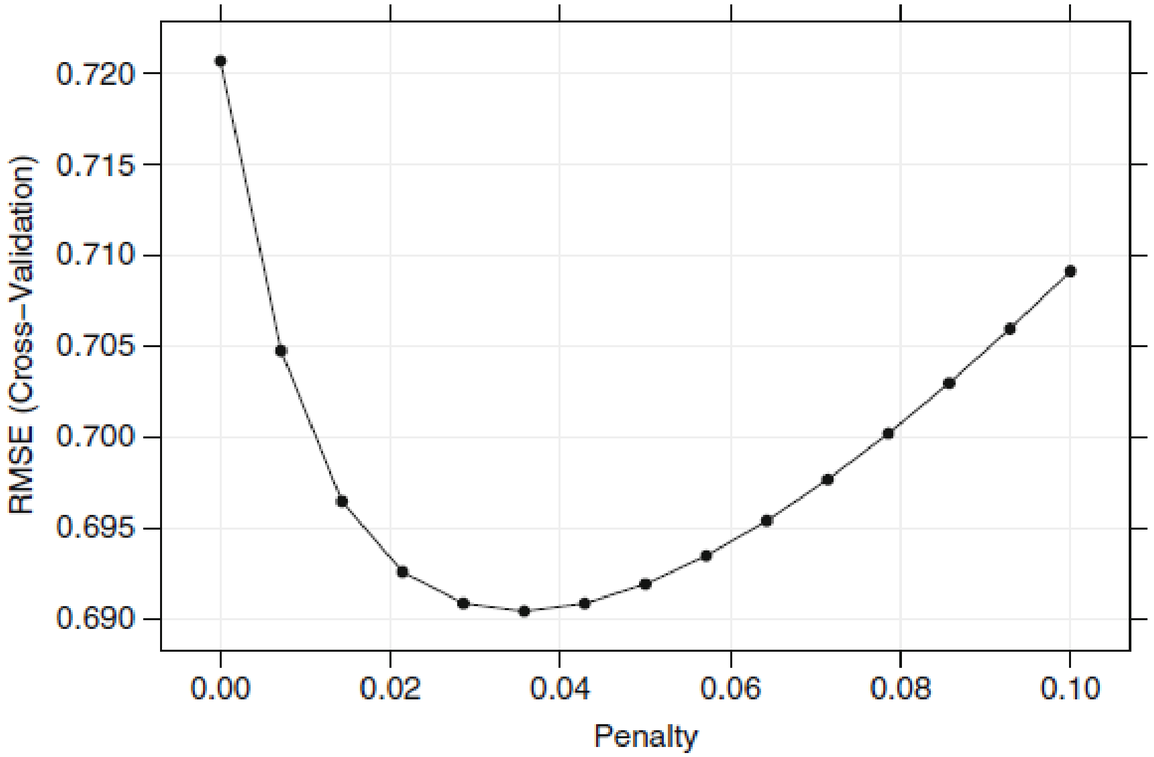
REGRESSÕES PENALIZADAS

A penalização L_2 é também conhecida como penalização ridge



REGRESSÕES PENALIZADAS

O valor do valor de λ (penalização) é escolhido por validação cruzada. No caso: 0,036.



Algoritmos de

MACHINE LEARNING

REGRESSÕES PENALIZADAS

- A penalização ridge encolhe os parâmetros, mas não reduz nenhum a 0, ou seja, não faz seleção de variáveis.
- Para isso, existe a penalização lasso (L_1).
- Lasso faz regularização e seleção de variáveis preditoras, pela penalização dos valores absolutos dos parâmetros.

$$\mathrm{SSE}_{L_1} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{P} |\beta_j|.$$

REGRESSÕES PENALIZADAS

Regressão logística

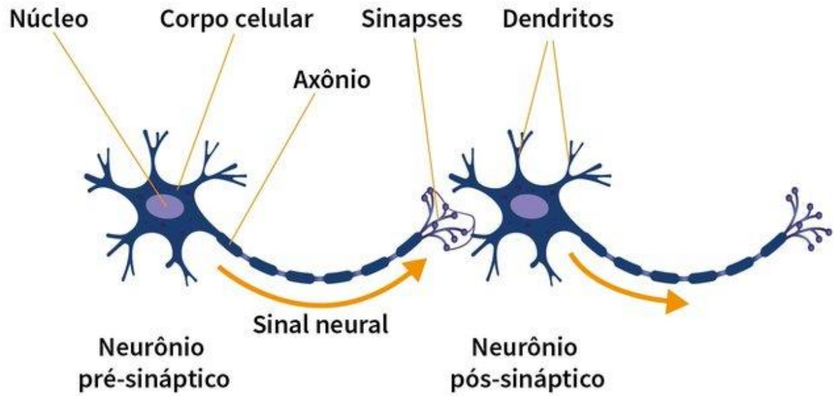
- Utilizada quando o desfecho a ser predito é categórico (problema de classificação).
- Também é possível incluir penalizações de ridge, lasso e redes elásticas.
- Mesmo sistema (só que os parâmetros da regressão logística são estabelecidos pelos métodos de máxima verossimilhança).

Algoritmos de

MACHINE LEARNING

REDES NEURAIS

- Inspiradas pelo funcionamento do cérebro.
  - Neurônios conectados por axônios e dendritos.
  - Região de contato: sinapses.
    - Sinais são propagados sequencialmente.
    - Força das conexões sinápticas dependem de estímulos externos.



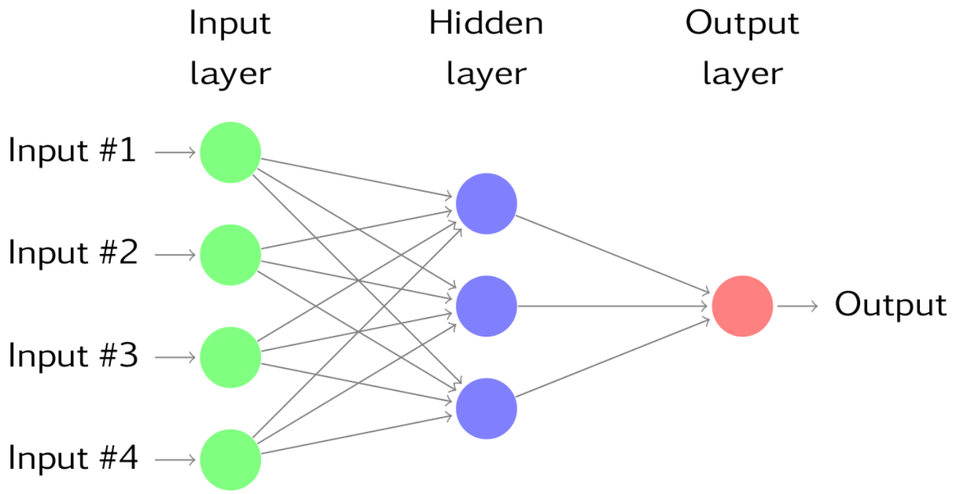
Algoritmos de

MACHINE LEARNING

REDES NEURAIS

Problemas de regressão: apenas um neurônio de desfecho (*output*), Y_1.

Problemas de classificação: número de neurônios presentes no output são iguais ao número total de categorias a serem preditas.



REDES NEURAIS

- O modelo com apenas uma camada de unidades latentes (“ocultas”) representa a estrutura mais simples de uma rede neural.

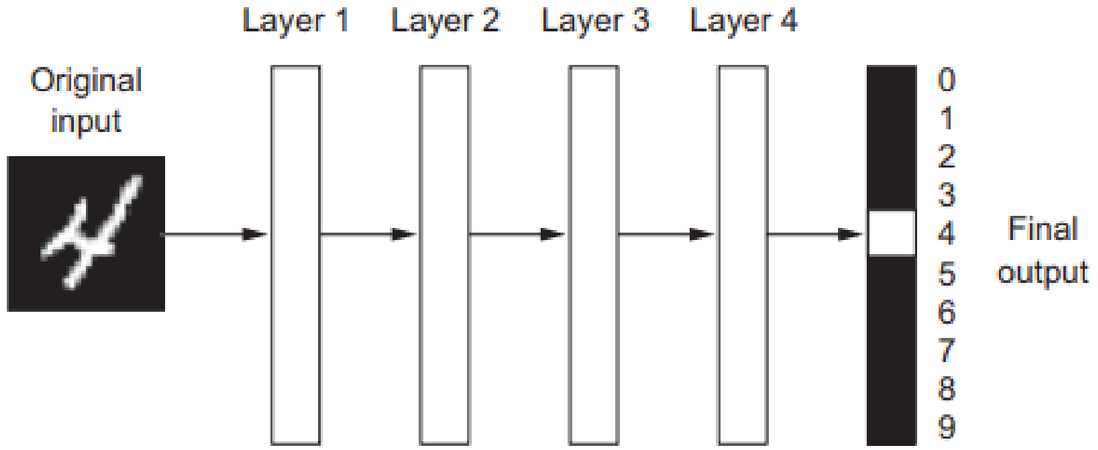
- Outros modelos, em que mais camadas são utilizadas como passo intermediário entre os preditores e a resposta: deep learning.

Algoritmos de

MACHINE LEARNING

DEEP LEARNING

Redes neurais profundas (mais do que uma camada oculta), em que cada camada encontra novas representações para os dados.



Algoritmos de

MACHINE LEARNING

DEEP LEARNING

Os dados são transformados em cada neurônio por meio de pesos (parâmetros) e vieses (interceptos). O objetivo é encontrar os pesos e vieses que levem ao desfecho correto.

DEEP LEARNING

Como medir o erro da predição para calibrar os pesos e vieses?

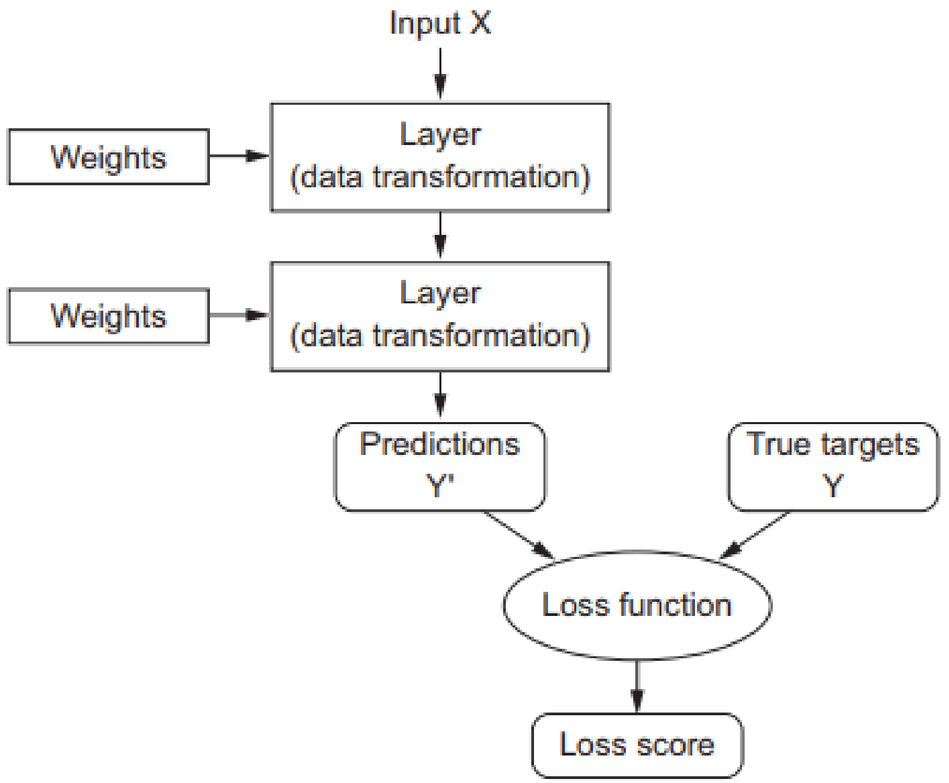
- Por meio da função de perda (ou função objetivo).
  - Regressão: erro quadrático médio.
  - Classificação: entropia cruzada.

Os pesos e vieses são ajustados de trás para frente por meio de backpropagation.

Algoritmos de

MACHINE LEARNING

DEEP LEARNING



DEEP LEARNING

Os pesos e vieses são iniciados randomicamente com valores baixos.

– Nesses casos, o erro da função de perda será alto.

– Mas os erros são corrigidos pelo ajustes dos pesos de trás para frente.

– O processo é repetido diversas vezes com diferentes lotes (menos custoso computacionalmente que todos os dados) de observações sorteadas.

DEEP LEARNING

Como corrigir os erros por meio do ajuste dos pesos?

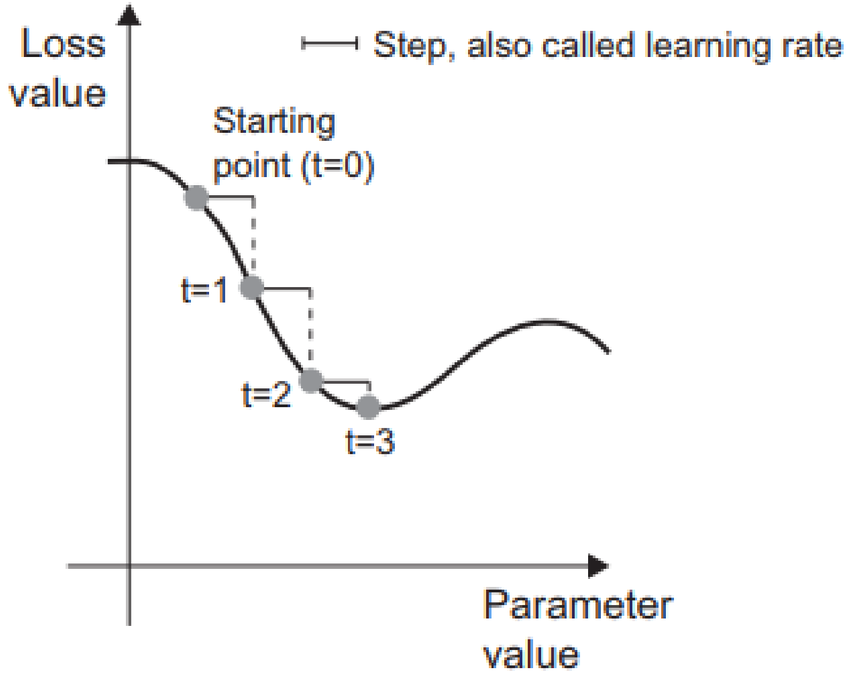
- Solução simples: deixar todos os pesos fixos menos um.
- Porém seria ineficiente, já que um lote de dados teria de passar pelo algoritmo para cada peso atualizado.
- Solução: usar gradient descent.
  - Calcular o gradiente da função de perda em relação aos pesos e mover o valor dos pesos na direção oposta do gradiente, diminuindo as perdas.

Algoritmos de

MACHINE LEARNING

DEEP LEARNING

Mini-batch stochastic gradient descent.



DEEP LEARNING

– Mini-batch stochastic gradient descent.

  - Sorteio de lotes que vão sendo utilizados para calibrar os pesos.

– Figura anterior foi para um só peso.

– É possível realizar o mesmo em várias dimensões, uma para cada peso a ser estimado.

Função de ativação

Sem uma função de ativação, os neurônios são apenas a soma dos pesos multiplicados pelos valores dos neurônios anteriores.

Isso limita a rede a aprender transformações lineares.

É preciso uma função de ativação não-linear: mais comum é relu (elimina dados negativos e permite ganhos de velocidade de computação).

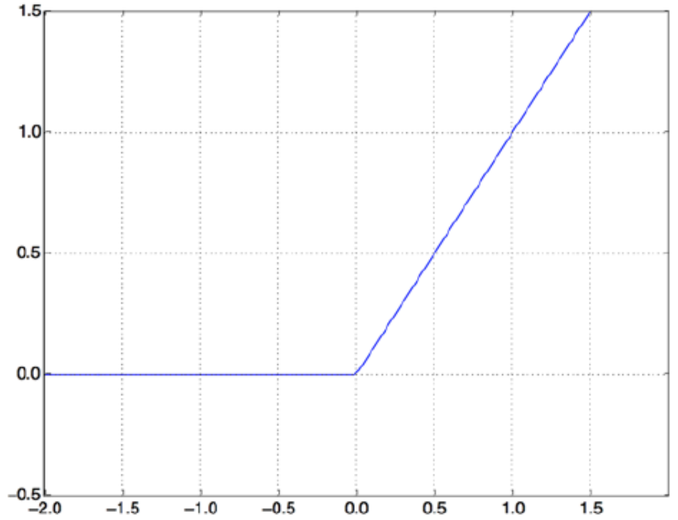


Figure 3.4 The rectified linear unit function

Algoritmos de

MACHINE LEARNING

DEEP LEARNING

Deep learning é composto por:

- Mais do que uma camada profunda, muitas vezes totalmente conectadas (densas).
- Função de ativação: como o sinal é modificado e passado entre neurônios.
- Lotes dos dados de treino que são passados de cada vez para calibrar os pesos.
- Função de perda, que orienta o aprendizado:
  - Entropia cruzada para desfecho binário (mede a distancia entre a realidade e a probabilidade predita).
  - Erro quadrático médio para problemas de regressão.

Algoritmos de

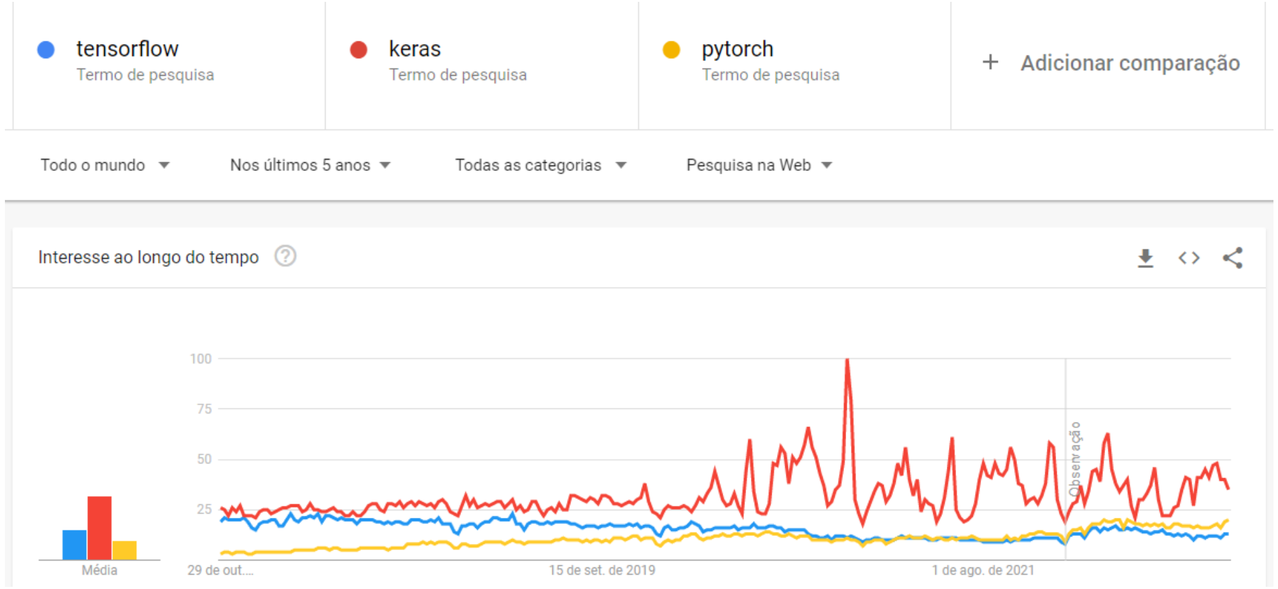
MACHINE LEARNING

DEEP LEARNING

- Keras é uma biblioteca relativamente simples e com linguagem amigável para rodar modelos de deep learning.

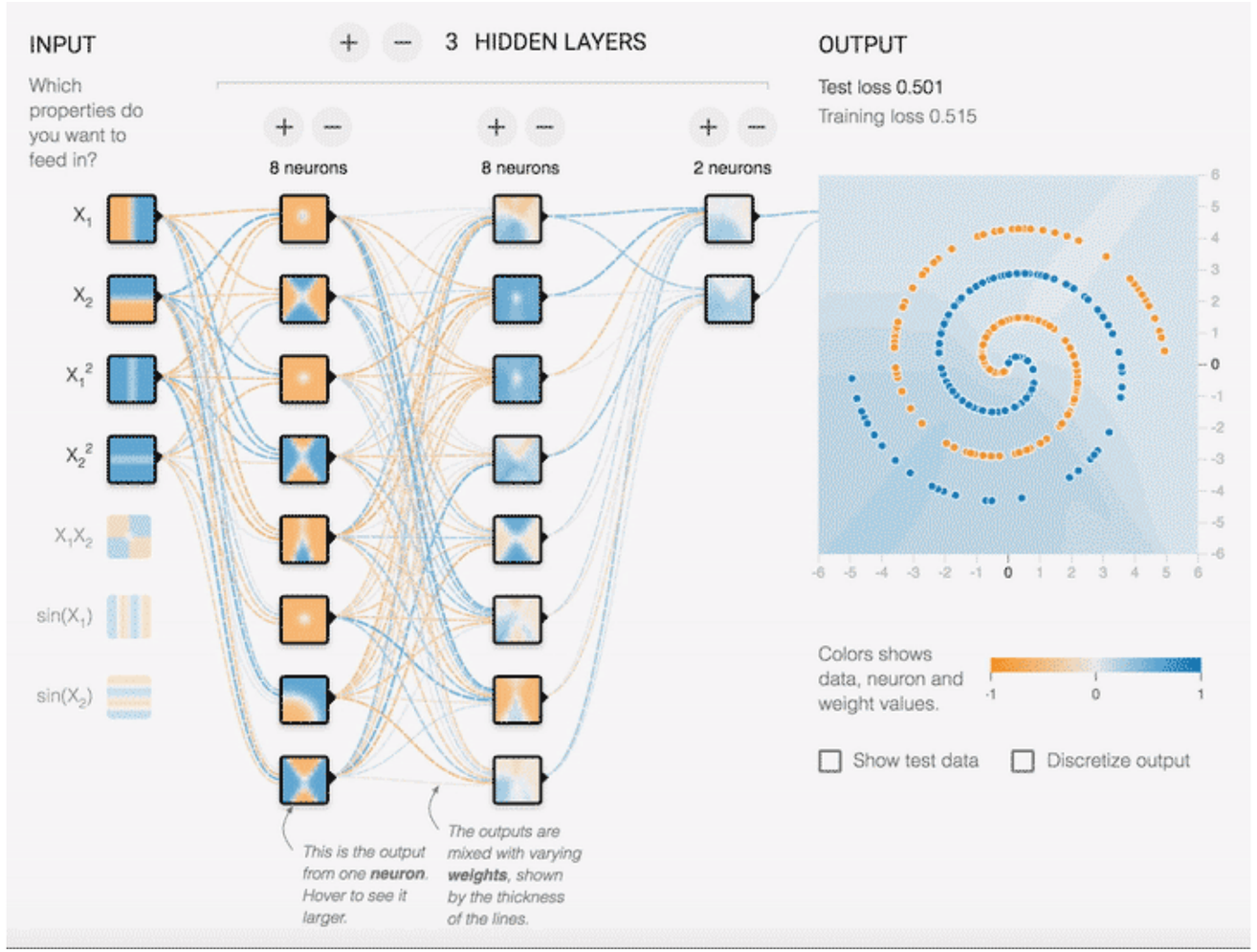
- Bastante prática, só não é recomendada se você quiser desenvolver novos modelos ainda não existentes.

- Utiliza tensorflow.



Algoritmos de

MACHINE LEARNING



Dica: Playground Tensorflow
https://playground.tensorflow.org

LABDAPS

LABORATÓRIO DE BIG DATA E ANÁLISE PREDITIVA EM SAÚDE

Obrigado!

Alexandre Chiavegatto Filho

http://labdaps.fsp.usp.br

@SaudenoBR

@labdaps

alexdiasporto@usp.br