Instituto Federal de Santa Catarina
Campus Florianópolis

Redes Neurais Artificiais

Prof. Glauco Cardozo

glauco.cardozo@ifsc.edu.br

Rede Neural Artificial

Redes neurais artificiais (Artificial Neural Network - ANN) são modelos computacionais inspirados na estrutura cerebral de animais.

- São formados por grupos de neurônios, que por sua vez, são modelos matemáticos baseados no funcionamento de uma célula nervosa.
- Proposto pelo neurofisiologista americano Warren S. McCulloch e pelo lógico americano H. Pitts Jr em 1943.

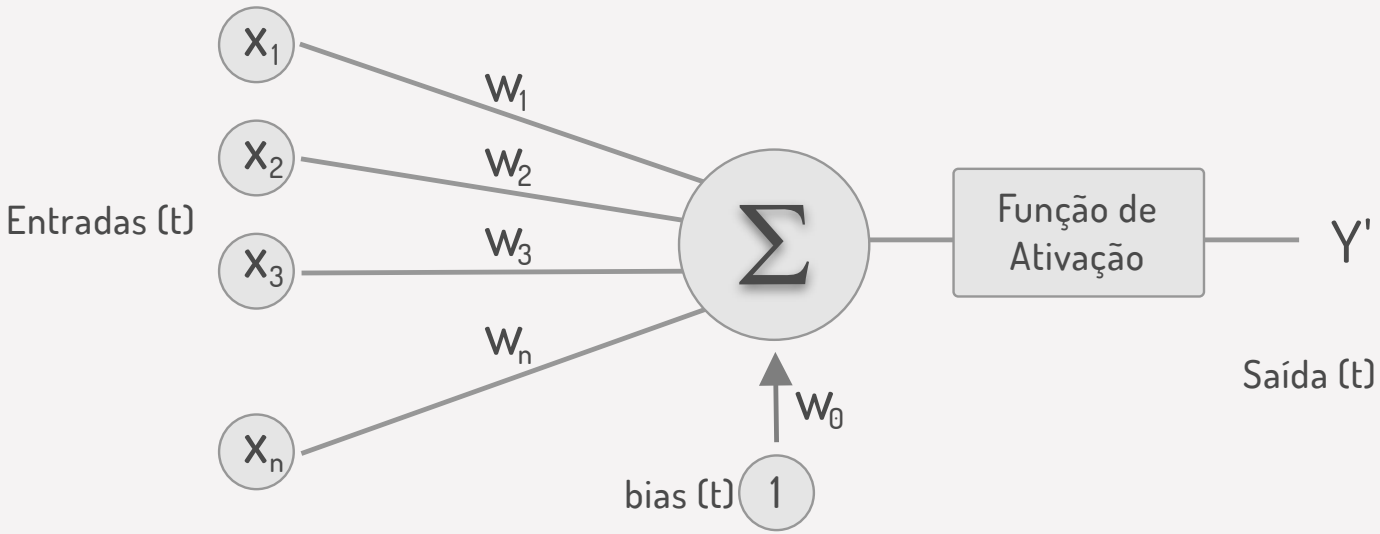
Rede Neural Artificial

Em 1957, o psicólogo americano Frank Rosenblatt, inspirado pela teoria hebbiana que defende a adaptação plástica do neurônio durante um processo de aprendizagem, propôs um novo modelo de neurônio chamado de **Perceptron**, sendo este aperfeiçoado por Marvin Minsky e Seymour Papert em 1969.

Com algumas melhorias em relação ao neurônio de Pitts e fazendo uso de um método de aprendizado supervisionado e diferentes funções de ativação, o Perceptron é capaz aprender e classificar padrões linearmente separáveis.

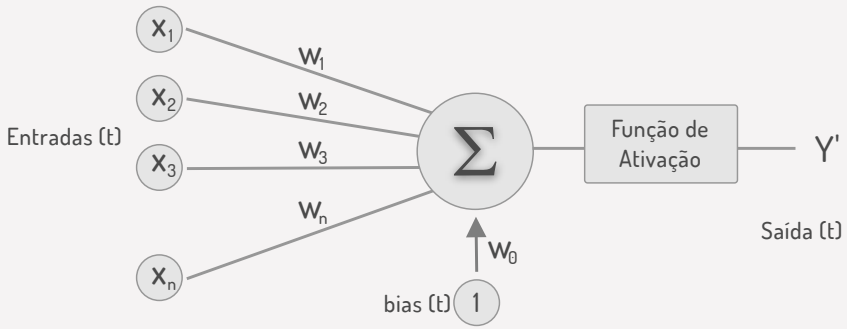
Rede Neural Artificial

Perceptron



Rede Neural Artificial

Perceptron



Possui uma entrada extra chamada de ***bias***.

Cada entrada também recebe um peso sináptico **w**, permitindo que algumas entradas tenham maior influência do que outras

No aprendizado supervisionado a rede faz uso de respostas corretas a fim de calcular o erro diante das respostas encontradas pelo modelo. Por meio de um método de **retro propagação**, os pesos dos neurônios são ajustados com o objetivo de encontrar a saída esperada de acordo com cada entrada de dados

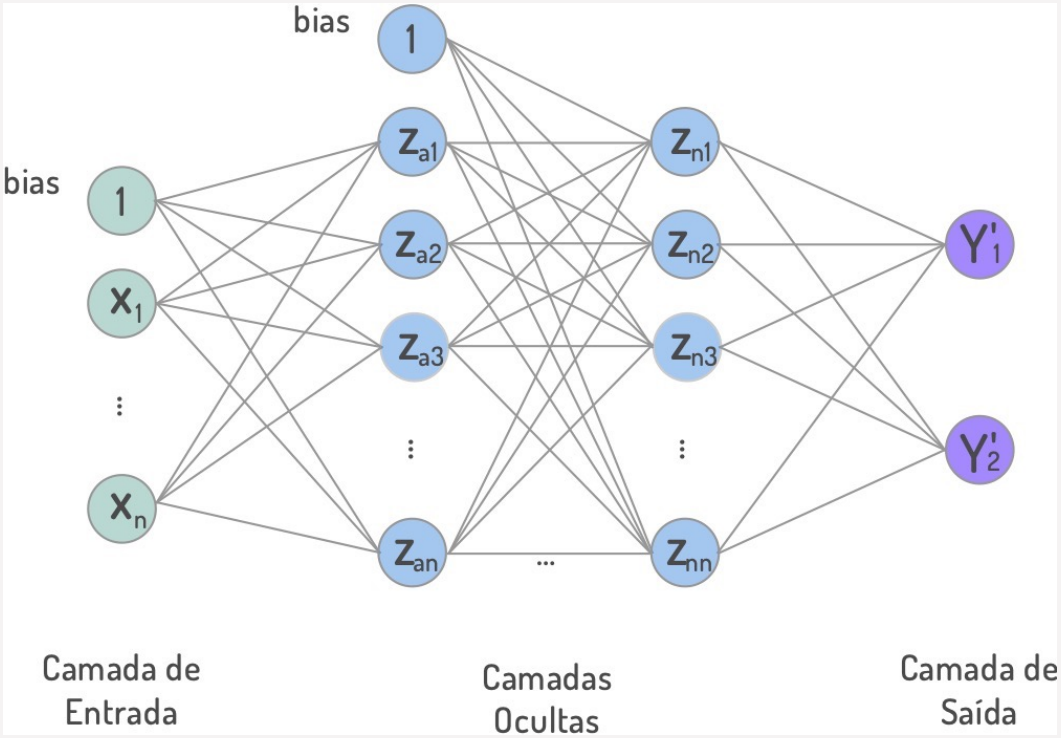
Rede Neural Artificial

Perceptron Multi Camadas

Adicionando mais camadas de neurônios, é possível resolver problemas linearmente não-separáveis. Este tipo de rede é conhecido como Perceptron Multicamadas (*Multi-layer Perceptron* – MLP).

Rede Neural Artificial

Perceptron Multi Camadas



Rede Neural Artificial

Perceptron Multi Camadas

A biblioteca *scikit-learn* possui um algoritmo **MLP** com métodos tanto para classificação quanto para regressão

A classe ***MLPRegressor*** implementa um MLP, também com treinamento por retropropagação, mas sem função de ativação na camada de saída, o que se compara ao uso da função identidade como função de ativação.

A classe usa o erro quadrado como função de perda e método do gradiente; e a saída é um conjunto de valores contínuos.

MLPRegressor também suporta regressão de várias saídas.

Rede Neural Artificial

Perceptron Multi Camadas - Otimização em mini-batches

- Cada iteração do método de otimização (cálculo da função custo e gradiente) é realizada usando um subconjunto (**mini-batch**) de **B < m** amostras de treinamento
- Cada passagem por todo o conjunto de treinamento (m/B iterações) é chamada de **época**
- Menor custo computacional e insensível a redundância nos dados
- Maior capacidade de escapar de mínimos locais

Rede Neural Artificial

Perceptron Multi Camadas - Parâmetros

hidden_layer_sizes ***: tuple, length = n_layers - 2, default=(100,)***
O i-ésimo elemento representa o número de neurônios na i-ésima camada oculta.

activation*{'identity', 'logistic', 'tanh', 'relu'}, default='relu'*
Função de ativação para a camada oculta

solver*{'lbfgs', 'sgd', 'adam'}, default='adam'*
O método para otimização de peso.

Alpha ***float, default=0.0001***
Parâmetro de penalidade L2 (prazo de regularização).

Rede Neural Artificial

Perceptron Multi Camadas - Parâmetros

max_iter ***int, default=200***
Número máximo de iterações.

Tol ***float, default=1e-4***
Tolerância para a otimização.

n_iter_no_change ***int, default=10***
Número máximo de épocas que não atende a melhora da tol.

Verbose ***bool, default=False***
Se as mensagens de progresso devem ser impressas.

Rede Neural Artificial

Perceptron Multi Camadas - Exemplo

```
model = MLPRegressor(hidden_layer_sizes=[30, 20, 10],
                     activation='logistic', solver='lbfgs', alpha=0.01,
                     max_iter=10000)
model.fit(X,y)
plot_predict(model,X,y)
print('R2 score:',model.score(X,y))
```