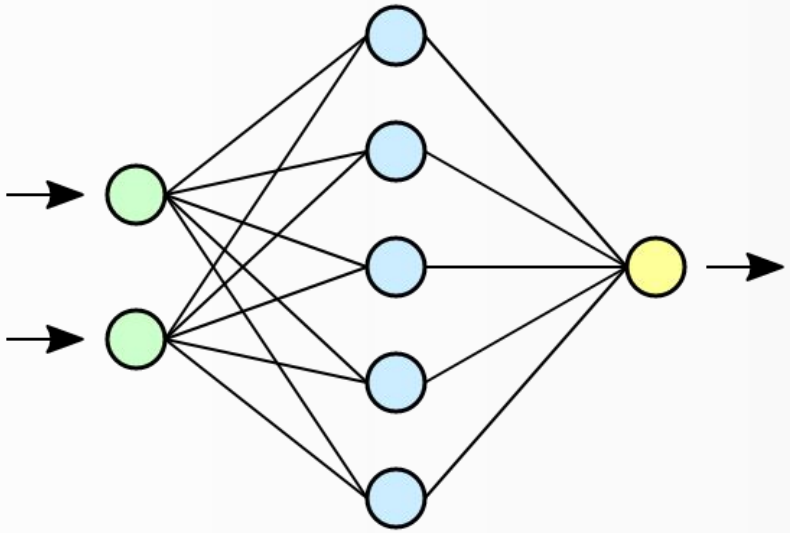
Modelos de Redes Neurais

João Marcos Alves Modesto Ramos - 3506

O que são redes neurais?

“Redes neurais são sistemas de computação com nós interconectados que funcionam como os neurônios do cérebro humano. Usando algoritmos, elas podem reconhecer padrões escondidos e correlações em dados brutos, agrupá-los e classificá-los, e – com o tempo – aprender e melhorar continuamente.”



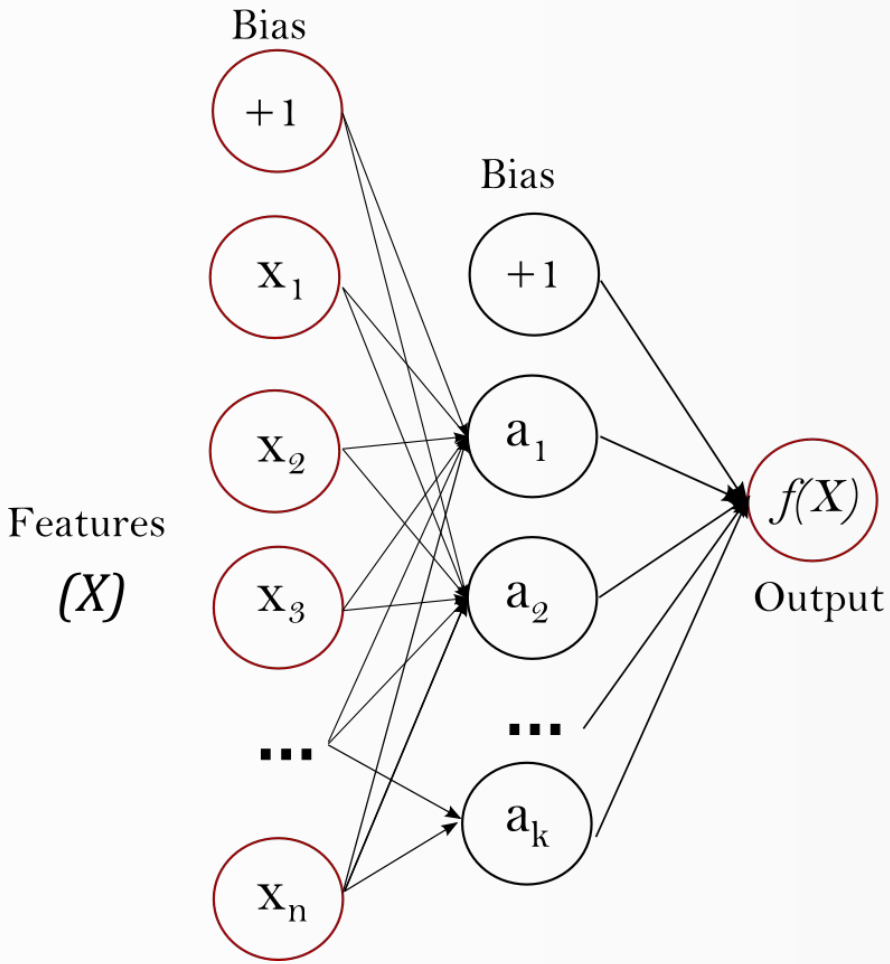
Multi-layer Perceptron (MLP)

Multi-layer Perceptron (MLP) é um algoritmo de aprendizado supervisionado que aprende uma função $f(\cdot) : R^m \rightarrow R^o$ treinando em um conjunto de dados, onde **m** é o número de dimensões para entrada e **o** é o número de dimensões para saída. Dado um conjunto de características e um target , ele pode aprender um aproximador de função não linear para classificação ou regressão.

É diferente da regressão logística, pois entre a camada de entrada e a camada de saída pode haver uma ou mais camadas não lineares, chamadas de camadas ocultas.

UFV

Multi-layer Perceptron (MLP)



Multi-layer Perceptron (MLP)

Vantagens do MLP:

- Capacidade de aprender modelos não-lineares
- Capacidade de aprender modelos em tempo real usando *partial_fit*

Desvantagens do MLP:

- MLP com camadas ocultas tem uma função de perda não convexa onde existe mais de um mínimo local
- Requer o ajuste de vários hiperparâmetros, como o número de neurônios ocultos, camadas e iterações.
- O MLP é sensível ao dimensionamento de recursos.

UFV

MLPClassifier

O termo "perceptron multicamadas" não se refere a um único perceptron que possui várias camadas. Em vez disso, ele contém muitos perceptrons que são organizados em camadas. Uma alternativa é a "rede perceptron multicamadas".

Além disso, os "perceptrons" do MLP não são perceptrons no sentido mais estrito possível. Os perceptrons verdadeiros são formalmente um caso especial de neurônios artificiais que usam uma função de ativação de limiar. Os perceptrons MLP podem empregar funções de ativação arbitrárias. Um verdadeiro perceptron realiza classificação binária, um neurônio MLP é livre para realizar classificação ou regressão, dependendo de sua função de ativação

MLPClassifier - Backpropagation

Backpropagation,que consiste em duas fases:

1. O passo para frente (forward pass), onde nossas entradas são passadas através da rede e as previsões de saída obtidas (essa etapa também é conhecida como fase de propagação).

2. O passo para trás (backward pass), onde calculamos o gradiente da função de perda na camada final (ou seja, camada de previsão) da rede e usamos esse gradiente para aplicar recursivamente a regra da cadeia (chain rule) para atualizar os pesos em nossa rede (etapa também conhecida como fase de atualização de pesos ou retro-propagação).

UFV

MLPClassifier - Características

- O MLPClassifier treina iterativamente, pois a cada passo de tempo as derivadas parciais da função de perda em relação aos parâmetros do modelo são calculadas para atualizar os parâmetros.
- Ele também pode ter um termo de regularização adicionado à função de perda que reduz os parâmetros do modelo para evitar o overfitting.

MLPClassifier - Entradas

O MLP treina em dois arrays: array X de tamanho (n_amostras, n_features), que contém as amostras de treinamento representadas como vetores de recursos de ponto flutuante; e array y de tamanho (n_amostras), que contém os valores de destino (rótulos de classe) para as amostras de treinamento:

```
X = [[0., 0.], [1., 1.]]
y = [0, 1]
```

MLPClassifier - Exemplo

```
>>> from sklearn.neural_network import MLPClassifier
>>> X = [[0., 0.], [1., 1.]]
>>> y = [0, 1]
>>> clf = MLPClassifier(solver='lbfgs', alpha=1e-5,
...                     hidden_layer_sizes=(5, 2), random_state=1)
...
>>> clf.fit(X, y)
MLPClassifier(alpha=1e-05, hidden_layer_sizes=(5, 2), random_state=1,
              solver='lbfgs')
```

UFV

MLPClassifier - Parâmetros

- hidden_layer_sizes : tuple, length = n_layers - 2, default=(100,)
- activation : {‘identity’, ‘logistic’, ‘tanh’, ‘relu’}, default=’relu’
- solver : {‘lbfgs’, ‘sgd’, ‘adam’}, default=’adam’
- alpha : float, default=0.0001
- batch_size : int, default=’auto’
- learning_rate : {‘constant’, ‘invscaling’, ‘adaptive’}, default=’constant’
- learning_rate_init : float, default=0.001
- power_t : float, default=0.5
- max_iter : int, default=200

UFV

MLPClassifier - Parâmetros

- shuffle : bool, default=True
- random_state : int, RandomState instance, default=None
- tol : float, default=1e-4
- verbose : bool, default=False
- warm_start : bool, default=False
- momentum : float, default=0.9
- nesterovs_momentum : bool, default=True
- early_stopping : bool, default=False
- validation_fraction : float, default=0.1

MLPClassifier - Parâmetros

- beta_1 : float, default=0.9
- beta_2 : float, default=0.999
- epsilon : float, default=1e-8
- n_iter_no_change : int, default=10
- max_fun : int, default=15000

MLPClassifier - Muti-label classification

```
>>> X = [[0., 0.], [1., 1.]]
>>> y = [[0, 1], [1, 1]]
>>> clf = MLPClassifier(solver='lbfgs', alpha=1e-5,
...                     hidden_layer_sizes=(15,), random_state=1)
...
>>> clf.fit(X, y)
MLPClassifier(alpha=1e-05, hidden_layer_sizes=(15,), random_state=1,
              solver='lbfgs')
>>> clf.predict([[1., 2.]])
array([[1, 1]])
>>> clf.predict([[0., 0.]])
array([[0, 1]])
```

MLPClassifier - Quando usar?

Os MLPs são adequados para problemas de previsão de classificação em que as entradas são atribuídas a uma classe ou rótulo. Eles também são adequados para problemas de previsão de regressão em que uma quantidade de valor real é prevista a partir de um conjunto de entradas. Os dados geralmente são fornecidos em um formato tabular, como você veria em um arquivo CSV ou em uma planilha. Ex:

- Datasets Tabulares
- Problemas de predição de classificação
- Problemas de predição de regressão

UFV

Obrigado pela atenção!

Dúvidas?