02042024_Card12

Neurônios artificiais (Perceptron)

O perceptron é um modelo matemático simples que representa o neurônio artificial básico, a unidade fundamental das redes neurais artificiais. A estrutura do neurônio é formada por:

- **Entradas**: O perceptron recebe N entradas que representam valores numéricos
- **Pesos:** As entradas são multiplicadas por pesos para que seja feita uma soma ponderada
- **Função de Ativação**: A soma ponderada é passada por uma função de ativação, que determina a saída final do perceptron. A função de ativação mais comum no perceptron é a função degrau, que retorna 1 se a soma ponderada for maior ou igual a um limiar, ou zero caso contrário.

A aprendizagem de uma rede neural é essencialmente o proccesso de descobrir os pesos ideais para cada entrada.

Multi-Layer Perceptron

O Perceptron Multicamada (MLP) é uma rede neural artificial composta por múltiplas camadas de perceptrons interconectados. Ao empilhar perceptrons em camadas, o MLP é capaz de aprender representações complexas e linearmente não separáveis

Estrutura:

A estrutura do perceptron multi camadas se difere do perceptron de camada única pela utilização de uma camada oculta, que recebe a saída dos perceptrons da camada anterior como entrada, aplica seus pesos e função de ativação para gerar uma nova saída. Os perceptrons da camada de saída recebem a saída da última camada oculta e geram a saída final da rede, quepode ser binária, para tarefas de classificaçã ou contínuas ,para tarefas de regressão.

Função Sigmoide(Função de ativação):

A função sigmoide, ou função logística, é uma função matemática utilizada em redes neurais artificiais como função de ativação. A função sigmoide possui uma curva em forma de "S", mapeando valores de entrada em um intervalo limitado entre 0 e 1. Essa característica permite que a função atue como um divisor probabilístico, convertendo entradas em probabilidades, e permite que os neurônios aprendam relações não lineares entre as features e a saída desejada.

A expressão da função sigmoide é a seguinte:

f(x) = (1 / (1 + exp(-k * (x - x0))))

Onde:

- `f(x)` =valor da função sigmoide
- `x` = entrada da função
- `k` = parâmetro de inclinação da curva
- `x0` = parâmetro de deslocamento da curva

Cálculo do erro

É através do calculo do erro que a rede consegue reajustar seus pesos para melhorar o seu desempenho na tarefa. Algumas funções utilizadas para calcular o erro:

Erro Quadrático Médio (MSE):

Mede a média dos quadrados das diferenças entre as saídas da rede e os valores reais.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Erro Absoluto Médio (MAE):

Mede a média dos valores absolutos das diferenças entre as saídas da rede e os valores reais. É mais robusta a outliers que o MSE, mas menos suave.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - p_i)^2}$$

Descida do Gradiente

A descida do gradiente é um algoritmo usado para encontrar o mínimo local de uma função. O algoritmo começa em um ponto inicial que pode ser aleatório, ou escolhido de acordo com algum critério pré-estabelecido. Em seguida, é calculado o gradiente da função no ponto atual, e o algoritmo se "move" na direção oposta do gradiente por um passo de tamanho pré estabelecido como taxa de aprendizado.

Vantagens:

- Simples de implementar
- Eficiente para encontrar mínimos locais
- Adaptável a diferentes tipos de funções

Desvantagens:

- Pode ser lento para convergir em algumas funções
- Pode encontrar mínimos locais em vez do mínimo global

Cálculo do parâmetro delta

Dentro de uma rede neural, o Delta representa a sensibilidade da função de erro em relação aos parâmetros da rede, é utilizado para ajustar os parâmetros na direção que minimiza o erro. Sua fórmula é expressa por:

```
Delta Saída = Erro × Derivada da Função de Ativação
```

Delta na camada oculta

O calculo do delta na camada oculta é semelhante ao cálculo do delta na camada de saída. A função de ativação utilizada para o calculo do delta na camada oculta é a **função Sigmoide**.

$$Delta\ Escondido = Peso \times Derivada\ da\ Função\ de\ Ativação \times Delta\ Saída$$

Backpropagation

O backpropagation, ou retropropagação, é um algoritmo que permite o treinamento da rede por meio do ajuste dos pesos das sinapses para minimizar o erro entre a saída prevista e a saída real.

Fórmula para a aplicação:

$$Peson+1 = (peson \times momento) + entrada \times delta \times taxa\ de\ aprendizagem$$

- Peso n+1 = Valor buscado pelo algoritmo.
- Peso n = Valor do peso atual
- Momento = "peso" que os pesos anteriores tem no processo de atualização

O ajuste do valor do momento define se o modelo terá um treinamento mais rápido, ou mais estável e preciso.

Bias, erro, descida do gradiente e mais parâmetros

BIAS:

O bias é um valor constante adicionado à entrada de um neurônio em uma rede neural. Ele induz a saída de um valor diferente mesmo quando todos os neurônios tem entrada com valor zero. O bias é atualizado durante o processo de treinamento junto com os pesos para minimizar a função de erro, e seu valor é inicializado em 1 por padrão das bibliotecas.

MSE:

Representa a média dos quadrados das diferenças entre os valores previstos pelo modelo e os valores reais. Utilizar esse cálculo faz com que o modelo penalize mais os resultados errados, pois o valor do erro é amplificado quando elevado ao quadrado.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

RMSE:

É a raiz quadrada do MSE.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - p_i)^2}$$

Stochastic Gradient Descent:

O SGD é um algoritmo de otimização é iterativo, e atualiza os parâmetros (pesos e bias) da rede na direção que minimiza a função de erro para cada registro, diferentemente da descida de gradientecomum, que atualiza os parametros para todos os registros.

Mini Batch Gradient Descent:

O Mini-batch GD é uma variante do SGD que atualiza os parâmetros com base em um pequeno conjunto de amostras (mini-batch) ao invés de todas as amostras do conjunto de treinamento

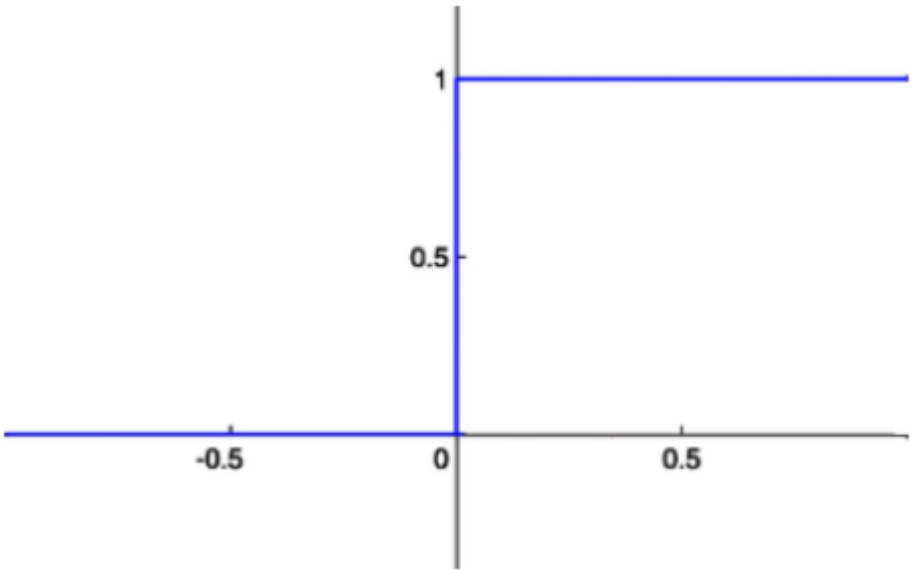
Parâmetros:

- Learning rate = taxa de atualização dos parâmetros
- Batch size = tamanho do conjunto de registros a ser atualizado
- Epochs = número de "épocas", iterações de treinamento.

Funções de ativação

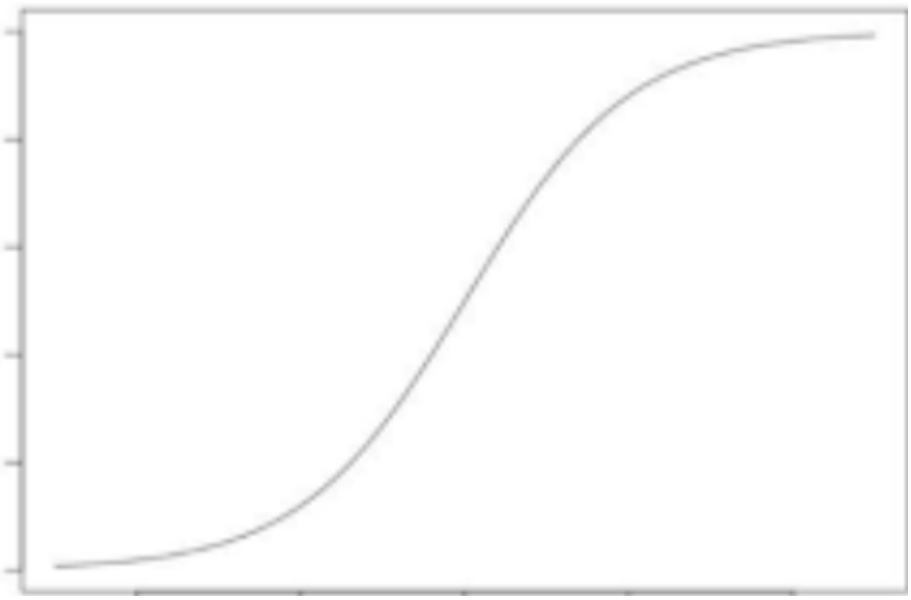
Step Function:

Retorna apenas os valores 0 ou 1, e serve apenas para problemas linearmente separáveis.



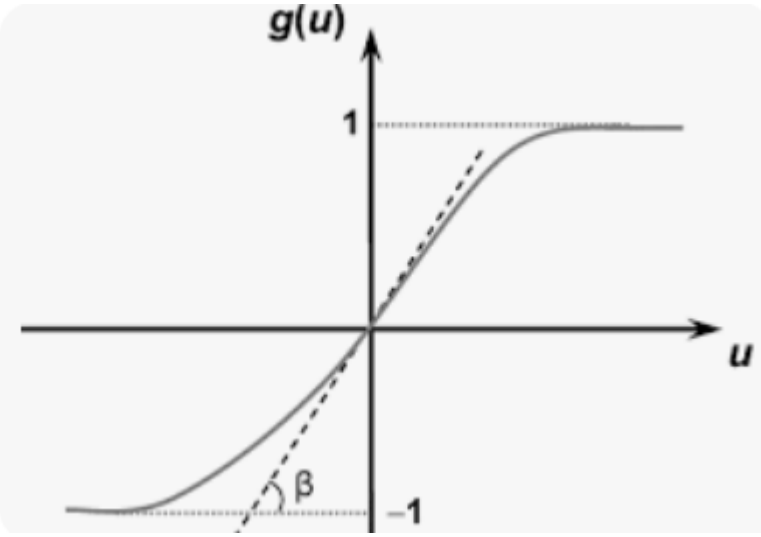
Função Sigmoide:

Retorna valores entre 0 e 1, que representam probabilidades dos eventos. É utilizada em problemas não linearmente separáveis.



Tangente Hiperbólica:

Retorna valores entre -1 e 1. Pode ser utilizada em problemas que tem apenas duas classes, e aceita etradas com valores negativos.



Classificação binária Breast Cancer Dataset

Estrutura da rede neural:

A rede possui 3 camadas:

- **Camada de entrada:** 30 neurônios (um para cada variável do dataset).
- **Camada oculta 1:** 16 neurônios com ativação ReLU.
- **Camada oculta 2:** 16 neurônios com ativação ReLU.
- **Camada de saída:** 1 neurônio com ativação sigmoide (para classificação binária).

Configuração da Rede Neural:

- **Otimizador:** Adam com taxa de aprendizado de 0.01, decaimento de 0.0001 e clipvalue de 0.5.
- **Função de perda:** Entropia cruzada binária.
- **Métrica:** Acurácia binária.
- **Treinamento:** 100 epochs com batch size de 10.

Validação Cruzada:

A validação cruzada divide o conjunto de dados em múltiplos subconjuntos e treina o modelo em cada um deles, alternando entre conjuntos de treinamento e validação. Isso é feito para evitar o overfitting, e avalia quais porções do dataset prepara melhor o modelo para a generalizr para dados novos.

Parâmetros utilizados na validação cruzada do Breast_Cancer dataset:

- **Tipo:** K-fold.
- **Ferramenta:** `cross_val_score` do scikit-learn.
- **Folds:** 10 (cv=10).
- **Métrica:** Acurácia (`scoring='accuracy'`).

Overfitting e Dropout:

Dropout é uma técnica de regularização usada para prevenir o overfitting. Ele desativa aleatoriamente uma fração dos neurônios em cada camada oculta durante o treinamento. Isso força a rede a aprender representações gerais das características dos dados. **Ajuste de parâmetros:**

O método GridSearchCV foi utilizado para testar combinações de parâmetros pré definidos em um dicionário, para verificar qual combinação terá a melhor performance.

Vantagens do GridSearchCV:

- Testa múltiplas configurações automaticamente.
- Evita overfitting ao avaliar em diferentes subconjuntos de dados com validação cruzada.
- Identifica a combinação de parâmetros que otimiza a performance.
- Elimina a necessidade de reescrever o código para testar diferentes parâmetros.

Classificação com 1 registro:

Foi feita a construção do modelo utilizando os melhores parâmetros encontrados pelo GridSearchCV, e utilizado o método predict para classificar se um novo registro na rede é maligno ou benigno.

Salvar a rede em .Json:

Salvamento do modelo para evitar passar pelo processo de parametrização toda vez que for classificar um novo registro. Para isso, usa-se Json(Javascript object notation), utilizando o método .to_jason para salvar toda a estrutura da rede neural em uma string, e o arquivo é salvo como um arquivo json no disco.Também são salvos os pesos em formato h5.

Carregando a rede:

A rede neural é carregada usando a fução model_from_json e load_weights. Isso não tira a necessidade de importar o modelo e base de dados.