Redes neuraisartificiais (RNA)

Redes neurais artificiais (RNA), conceitos e aplicações.

Prof.º Fábio Daudt

1. Itens iniciais

Propósito

Conhecer as redes neurais artificiais é fundamental porque elas abrangem boa parte da evolução das tecnologias que experimentamos nos últimos anos, como reconhecimento de voz, visão computacional e muitas outras que utilizam na sua essência essa mesma tecnologia.

Preparação

Antes de iniciar seu estudo, verifique se possui acesso a algum editor de Python para executar o projeto proposto (**download disponível**), tal como Jupyter Notebook ou acesso ao Google Colab. Os dados a serem processados também estão incluídos no referido arquivo para download.

Objetivos

- Reconhecer os principais conceitos e características das redes neurais artificiais (RNA).
- Identificar as principais arquiteturas de redes neurais artificiais e suas respectivas características.
- Identificar os conceitos e as características do perceptron e perceptron multicamadas.
- Aplicar um projeto prático de redes neurais artificiais.

Introdução

Inicialmente, vamos analisar os principais conceitos e características de uma rede neural artificial como técnica computacional que apresenta um modelo matemático inspirado na estrutura neural de organismos inteligentes e que adquirem conhecimento por meio da experiência, podendo uma grande rede neural artificial ter centenas ou milhares de unidades de processamento.

Veremos também as principais arquiteturas de redes neurais artificiais, assim como suas respectivas utilidades, cabendo destacar que arquiteturas neurais são tipicamente organizadas em camadas, com unidades que podem estar conectadas às unidades da camada posterior.

Destacaremos as redes perceptron e perceptron multicamadas, sendo que o modelo perceptron permite uma compreensão clara de como funciona uma rede neural em termos matemáticos, sendo uma excelente introdução.

Ao final, vamos treinar/testar um modelo de machine learning com arquitetura de rede neural perceptron de multicamadas (multilayer perceptron) para detecção de diabetes, aplicando, assim, nossos conhecimentos.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Entendendo as redes neurais artificiais

Assista ao vídeo a seguir para aprender sobre Redes Neurais como Paradigma de programação.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Paradigma de programação

As RNAs, de acordo com Fleck (2016), tem a seguinte linha do tempo:

1943

Origem das RNAs

As RNAs surgiram com o modelo matemático do neurônio biológico proposto por Warren McCulloch e Walter Pitts. O modelo, denominado neurônio MCP (McCulloch-Pitts), é descrito por um conjunto de n entradas, as quais são multiplicadas por um determinado peso e, em seguida, os resultados são somados e comparados a um limiar.

1958

Proposta do Perceptron

Frank Rosenblatt propôs uma topologia de rede denominada de perceptron constituída por neurônios MLP (perceptrons de múltiplas camadas) e arranjada em forma de rede composta de duas camadas, a qual possibilitou um aumento de trabalhos relacionados a redes neurais até 1969.

1969

Limitações do MLP

A publicação de Minsky e Papert mostrou deficiências e limitações do modelo MLP, provocando um desinteresse pelos estudos relacionados às RNAs.

1982

Renascimento das RNAs

Somente a partir de 1982, com a publicação do trabalho de Hopfield, foi novamente despertado o interesse pelos estudos relacionados às redes neurais.

Nos dias atuais, a modelagem matemática aliada à simulação de cenários futuros merece especial destaque, tendo em vista as múltiplas finalidades às quais se destina, sendo que a teoria de RNAs representa uma alternativa aos algoritmos tradicionais empregando métodos determinísticos.

Em sua forma mais geral, uma rede neural é um sistema projetado para modelar a maneira como o cérebro realiza uma tarefa particular, sendo normalmente implementada utilizando-se componentes eletrônicos ou é simulada por propagação em um computador digital. Para alcançarem bom desempenho, as redes neurais empregam uma interligação maciça de células computacionais simples, denominadas de “neurônios” ou unidades de processamento.

> Um dos mais surpreendentes paradigmas de programação já criados são as redes neurais artificiais. No método tradicional de programação, instruímos o computador sobre o que fazer, dividindo grandes problemas em muitas tarefas gerenciáveis e cuidadosamente definidas. Em contraste, não instruímos uma rede neural sobre como resolver um problema. Em vez disso, ela obtém conhecimento de dados observacionais e resolve o problema por conta própria. Ou seja, é um método para criar programas de computador que podem aprender com dados.

Uma rede neural artificial é inspirada e baseada em como acreditamos que o cérebro humano funciona. Observe como acontece:

Primeiro

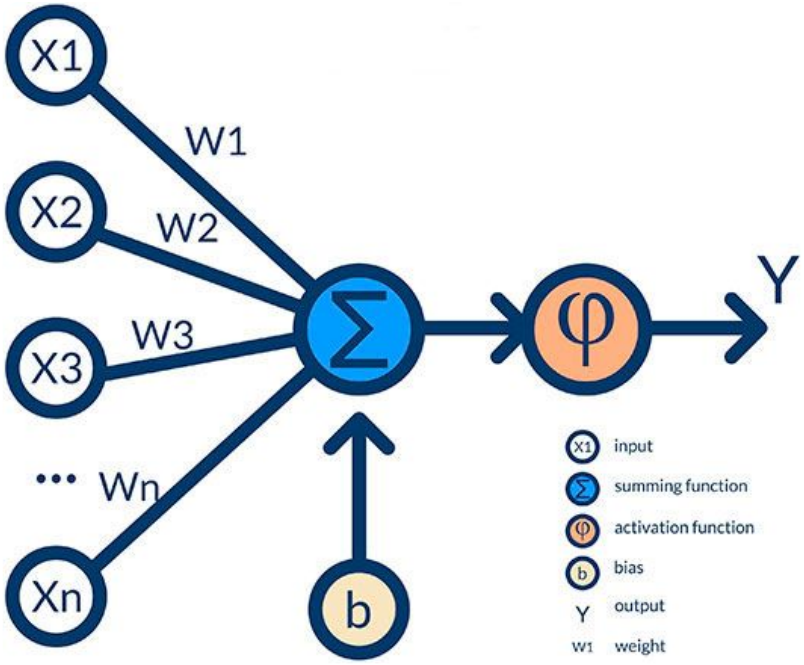
Uma rede de "neurônios artificiais" de software interconectados é construída e habilitada para que esses neurônios artificiais se comuniquem uns com os outros.

Segundo

A rede é então solicitada a resolver um problema, o que ela repetidamente tenta fazer enquanto amplia as conexões que levam ao sucesso e enfraquece as que levam ao fracasso.

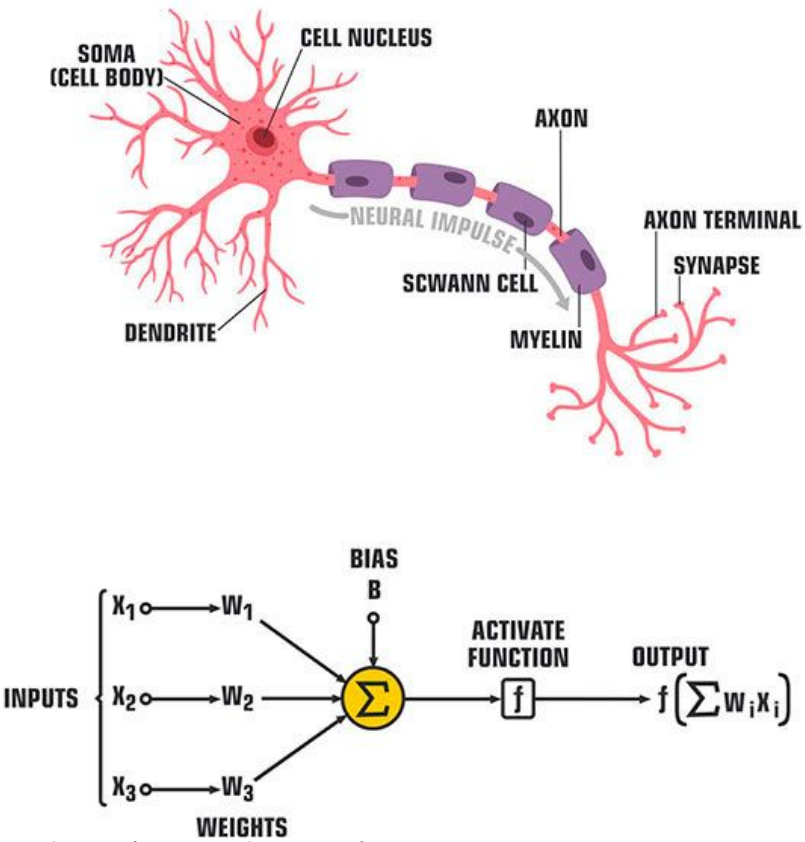
Um neurônio é o bloco de construção básico de perceptrons e redes neurais artificiais. Eles são inspirados por neurônios biológicos. Na programação, um neurônio é um dado e uma coleção de pesos entre esse neurônio e os neurônios conectados. Todas as conexões têm pesos associados para simular como a ativação de um neurônio afeta os demais neurônios conectados.

O modelo perceptron permite uma compreensão clara de como funciona uma rede neural em termos matemáticos. Basicamente, é um modelo matemático que recebe várias entradas, x1, x2, ... e produz uma única saída binária, como ilustra a imagem a seguir.



Contextualizando o neurônio artificial.

As redes neurais artificiais (RNA) foram desenvolvidas como resultado de pesquisas sobre o cérebro. Elas consistem em muitos nós interconectados, cada um dos quais executa uma operação matemática direta. Esse procedimento, juntamente com um conjunto de parâmetros exclusivos para cada nó, determina a saída de cada nó. Funções muito complexas podem ser aprendidas e calculadas combinando esses nós e especificando cuidadosamente seus parâmetros. Observe a imagem a seguir:



Neurônio biológico x neurônio matemático.

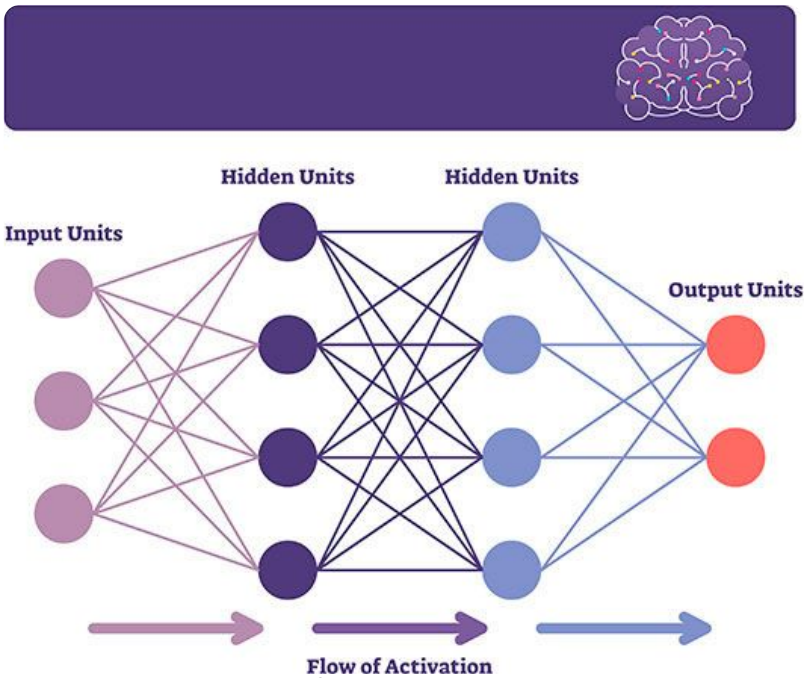
As redes neurais artificiais são comumente apresentadas como um grafo orientado, em que os vértices são os neurônios e as arestas, as sinapses. A direção das arestas informa o tipo de alimentação, ou seja, como os neurônios são alimentados (recebem sinais de entrada). As redes neurais derivam seu poder devido à sua estrutura massiva e paralela e a habilidade de aprender por experiência. Essa experiência é transmitida por meio de exemplos obtidos do mundo real, definidos como um conjunto de características formadas por dados de entrada e de saída.

Funcionamento de uma rede neural artificial

Assista ao vídeo a seguir para ver como funciona de uma rede neural artificial.

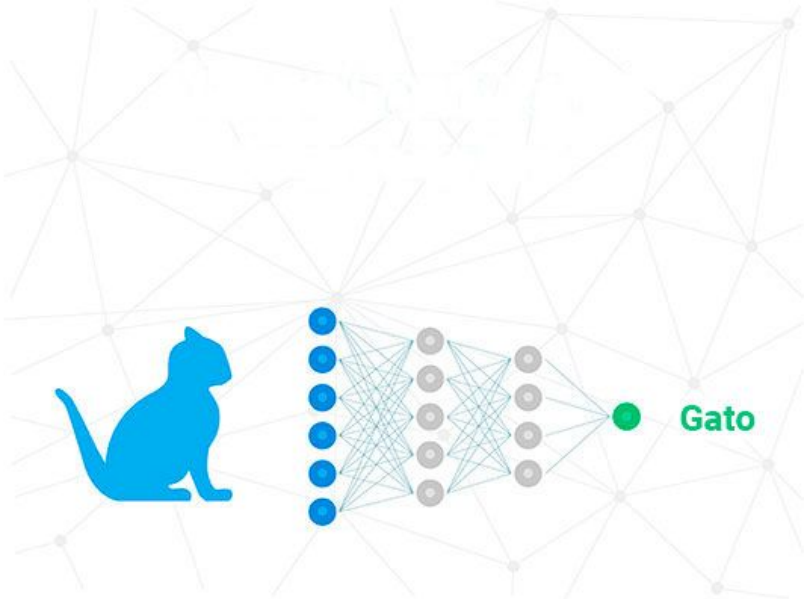
> Conteúdo interativo
>
> Acesse a versão digital para assistir ao vídeo.

Normalmente, a rede neural artificial é configurada em camadas. As camadas são compostas de vários "nós" interconectados, cada um com uma "função de ativação". Os seguintes componentes podem ser encontrados em uma rede neural artificial: camada de entrada (input layer), camada oculta (hidden layer), camada de saída (output layer), função de ativação (activate function), pesos (weights) e viés (bias).



Estrutura de uma rede neural artificial.

A imagem a seguir exemplifica o processamento de imagem, uma das aplicações das redes neurais.



Reconhecimento de imagem utilizando redes neurais.

Camada de entrada (input layer)

Os valores das características explicativas para cada observação devem ser inseridos como entrada na camada de entrada, sendo os inputs dos dados da imagem anterior. Em uma camada de entrada, normalmente existem tantos nós de entrada quanto variáveis explanatórias. A rede recebe os padrões da "camada de entrada" e os transmite para uma ou mais "camadas ocultas" via rede.

Os nós da camada de entrada são passivos, o que significa que não alteram os dados. Eles recebem um valor como entrada e o replicam em todas as suas saídas. Ele copia cada valor da camada de entrada e o envia para cada nó oculto.

Camada oculta (hidden layer)

Responsável pelo processamento dos dados, não tem contato com o mundo externo. A função dos neurônios ocultos é intervir entre a camada de entrada e saída. Por meio de uma ou mais camadas de neurônios ocultos, a rede se torna capaz de extrair estatísticas mais elaboradas.

Os valores de entrada dentro da rede estão sujeitos às modificações aplicadas pelas camadas ocultas. Isso envolve arcos de entrada que vêm de nós de entrada conectados a cada nó ou de outros nós ocultos.

Ele se conecta a nós de saída ou outros nós ocultos usando arcos que estão saindo do sistema. O processamento real é realizado por meio de um sistema de "conexões" ponderadas em uma camada oculta. Uma ou mais camadas ocultas podem existir.

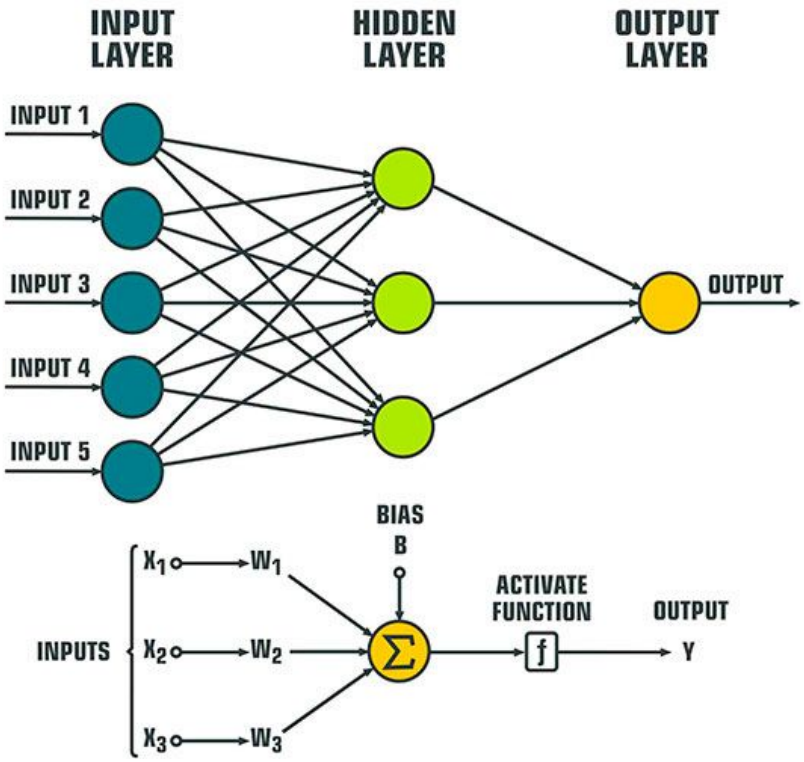
Camada de saída (output layer)

Camada onde ocorre o resultado final do processamento de uma rede neural. As camadas ocultas são vinculadas a uma “camada de saída”. A camada de saída recebe conexões das camadas ocultas ou da camada de entrada. Ela retorna um valor de saída que corresponde à previsão da variável de resposta. Em problemas de classificação, geralmente há apenas um nó de saída. Os nós ativos da camada de saída combinam e alteram os dados para produzir os valores de saída.

Pesos (weights)

Os pesos, que correspondem às variáveis W_1, W_2 e W_3 na imagem a seguir, são uma coleção de números predeterminados contidos no programa que são multiplicados pelos valores inseridos em um nó oculto. Um único número é então criado adicionando as entradas ponderadas.

Para que haja o processamento, a rede precisa de uma função de ativação, detalhada a seguir, que é a função matemática utilizada para definir os pesos associados a cada rede.



Estrutura de uma rede neural artificial.

Viés (bias)

O viés (bias) permite mover valores em uma direção ou outra. As redes neurais não sabem antecipadamente quais os valores a escolher para o bias. O bias também pode ser atualizado e alterado pela rede neural durante o treinamento, assim como os pesos.

O bias é uma variável incluída ao somatório da função de ativação, com o intuito de aumentar o grau de liberdade dessa função e, consequentemente, a capacidade de aproximação da rede. O valor do bias é ajustado da mesma forma que os pesos sinápticos.

Função de ativação (activate function)

A função de ativação é uma função que mapeia corretamente a camada anterior para as restrições da próxima (geralmente 0/1).

A função de ativação é uma mudança não linear nos valores antes de enviar os valores de resultado. Por que precisamos de uma função de ativação? Se usarmos apenas cálculo linear sem funções de ativação, não podemos dar nenhum “efeito de camada oculta’ ao nosso modelo. Ou seja, para "ativar" o poder real das redes neurais artificiais, precisamos aplicar uma "função de ativação". As principais funções de ativação são:

Sigmoid

Contínua entre 0 e 1.

Tanh

Contínuo entre -1 e 1.

Threshold (com limite)

Define a saída 1 ou 0 de acordo com um limite estabelecido.

ReLu

Torna qualquer valor negativo igual a 0, caso contrário não faz nada.

Linear

Multiplica a saída por uma constante.

Softmax

Converte um conjunto de números em probabilidades, usado para classificação.

Vejamos algumas representações a seguir:

Função de ativação.

Aplicando as redes neurais artificiais

Assista ao vídeo a seguir para aprender a aplicar as redes neurais artificiais.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Vantagens e desvantagens de uma rede neural artificial

Existem algumas vantagens e desvantagens de uma rede neural artificial para aprendizado de máquina.

Vamos conhecê-las:

Vantagens

- Funciona bem com dados lineares e não lineares.
- Arquitetura facilmente adaptável a vários domínios de problemas.
- As redes neurais funcionam mesmo que uma ou algumas unidades não respondam à rede.
- A rede neural aprende com os dados analisados e não requer reprogramação.

Desvantagens

- Requer muitos dados de treinamento limpos – precisam de uma ampla variedade de dados de treinamento para funcionar no mundo real, sendo esse o caso, porque qualquer máquina de aprendizado precisa de um conjunto grande o suficiente de instâncias representativas para entender completamente a estrutura subjacente e generalizar para novas situações.
- Requer alto poder computacional – são necessários muitos recursos de processamento e armazenamento para criar redes neurais de software grandes e eficientes. Até mesmo a forma mais simples de uma rede neural usando a tecnologia de Von Neumann pode exigir que um projetista de rede neural preencha milhões de linhas de banco de dados com conexões (dados), o que pode consumir muitos recursos computacionais (memória, armazenamento e processamento).
- Dificuldade na interpretação dos resultados – são chamados de modelos de caixa preta e fornecem pouquíssimo conhecimento sobre o que esses modelos realmente fazem.

Aplicações das redes neurais artificiais

As redes neurais têm uma ampla gama de aplicações e são capazes de analisar uma variedade de entradas, incluindo arquivos, bancos de dados, fotos, vídeos e muito mais. Além disso, eles não precisam de programação explícita para interpretar as informações nessas entradas.

Quase não há restrições nos campos em que as redes neurais podem ser usadas devido à abordagem generalista para a solução de problemas que elas fornecem. Hoje, as redes neurais são aplicadas de várias maneiras e seu uso está se espalhando rapidamente. Algumas aplicações comuns para redes neurais atualmente incluem: reconhecimento de imagem/padrão, previsão de trajetória de veículo autônomo, identificação facial, mineração de dados, filtragem de spam de e-mail, diagnóstico médico e pesquisa de câncer.

Sistema de reconhecimento de imagem.

Verificando o aprendizado

Questão 1

Redes neurais artificiais são técnicas computacionais que apresentam um modelo matemático inspirado na estrutura neural de organismos inteligentes e que adquirem conhecimento por meio da experiência. Como é chamada a primeira camada de uma rede neural artificial?

A

Camada de entrada (input layer).

B

Feedback layer (camada de retorno).

C

Camada de saída (output layer).

D

Activation layer (camada de ativação).

E

Camada de pesos (weight layer).

A alternativa A está correta.

A primeira camada de uma rede neural artificial é a camada de entrada (input layer).

Questão 2

As redes neurais artificiais são muito utilizadas para encontrar soluções e escalar processos para facilitar a tomada de decisões. Indique uma vantagem das redes neurais artificiais.

A

Requer muitos dados de treinamento limpos.

B

Ter arquitetura facilmente adaptável a vários domínios de problemas.

C

Requer alto poder computacional.

D

Apresentar dificuldade de interpretação dos resultados.

E

Pouca quantidade de dados disponíveis atualmente.

A alternativa B está correta.

A única opção que representa uma vantagem das redes neurais artificiais é o fato de a arquitetura ser facilmente adaptável a vários domínios de problemas.

Feed forward (redes diretas)

Assista ao vídeo para aprender sobre a rede neural feed forward (redes diretas).

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

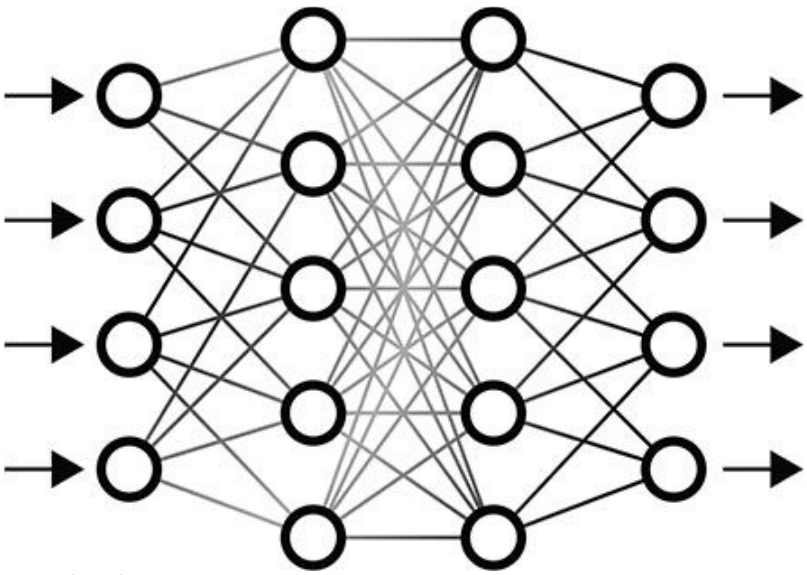
Arquitetura feed forward

Na arquitetura feed forward, os neurônios são dispostos em conjuntos distintos e ordenados sequencialmente, ou seja, são organizados em camadas (layers). A camada inicial recebe os sinais de entrada enquanto a camada final obtém as saídas. As camadas intermediárias são chamadas de camadas ocultas.

Uma rede neural feed forward é um tipo de rede neural artificial em que não há ciclo nas conexões entre os neurônios de uma mesma camada, ou seja, cada neurônio de uma camada é conectado com todos os neurônios da camada seguinte.

Como a entrada é processada apenas em uma direção, o modelo feed forward é o tipo mais simples de rede neural. Nessa arquitetura, embora os dados possam fluir através de vários nós, eles sempre avançam e nunca retrocedem. O fluxo de informação é sempre da camada de entrada para a camada de saída.

As redes feed forward podem ser redes de camada única ou redes de múltiplas camadas, apenas diferenciando o número de camadas, mas o conceito de alimentação adiante ou direta é o mesmo.



Redes feed forward.

Como funciona uma rede neural feed forward?

Uma rede neural feed forward é comumente vista em sua forma mais simples como um perceptron de camada única. Nesse modelo, uma série de entradas entra na camada e elas são multiplicadas pelos pesos. Cada valor é então somado para obter uma soma dos valores de entrada ponderados. Se a soma dos valores estiver acima de um limite específico, geralmente definido como zero, o valor produzido geralmente é 1, enquanto se a soma cair abaixo do limite, o valor de saída é -1.

1 Perceptron de camada única

É um importante modelo de redes neurais feed forward e é frequentemente usado em tarefas de classificação. Além disso, perceptrons de camada única podem incorporar aspectos de aprendizado de máquina. Usando uma propriedade conhecida como regra delta, a rede neural pode comparar as saídas de seus nós com os valores pretendidos, permitindo assim que a rede ajuste seus pesos por meio de treinamento para produzir valores de saída mais precisos. Esse processo de treinamento e aprendizado produz uma espécie de gradiente descendente.

2 Perceptron multicamadas (multilayer perceptrons)

O processo de atualização de pesos é semelhante, porém o processo é definido mais especificamente como retropropagação (backpropagation). Nesses casos, cada camada oculta dentro da rede é ajustada de acordo com os valores de saída produzidos pela camada final, ou seja, o objetivo do backpropagation é otimizar os pesos para que a rede neural possa aprender a mapear corretamente as entradas para as saídas.

Embora as redes neurais feed forward sejam extremamente simples, elas fornecem um benefício em alguns aplicativos de aprendizado de máquina devido à sua arquitetura simplificada. Usando um intermediário modesto para moderação, pode-se, por exemplo, colocar várias redes neurais feed forward com o objetivo de executá-las separadamente umas das outras. Semelhante ao cérebro humano, esse processo utiliza muitos neurônios individuais para lidar e compreender tarefas mais complexas. As descobertas de cada rede trabalhando independentemente podem ser integradas no final para criar uma saída sintetizada e coesa.

Feed backward networks (redes recorrentes)

Assista ao vídeo a seguir para aprender a usar as feed backward networks.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Arquitetura feed backward networks (redes recorrentes)

Nas redes feed backward networks (redes recorrentes), a saída de um neurônio é aplicada como entrada no próprio neurônio e/ou em outros neurônios de camadas anteriores, ou seja, há ocorrência de realimentação (ciclo no grafo).

Uma rede neural recorrente, na qual determinados caminhos são percorridos, é o inverso de uma rede neural feed forward (alimentação direta).

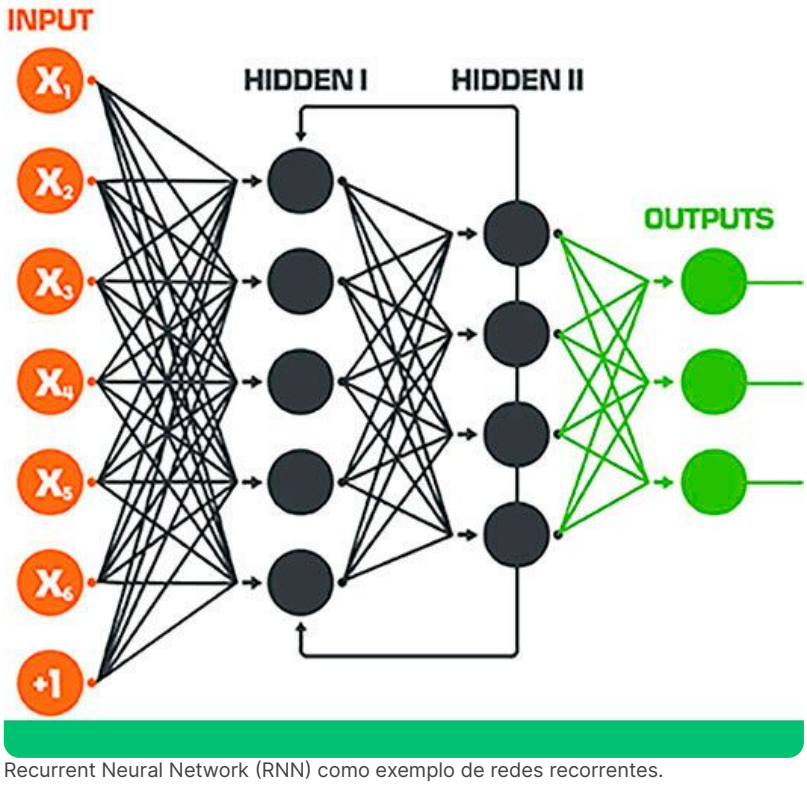
As redes recorrentes são identificadas por seus loops de feedback. Esses algoritmos de aprendizado são aproveitados principalmente ao usar dados de séries temporais para fazer previsões sobre resultados futuros (previsões do mercado de ações ou previsão de vendas), assistente de voz, previsão de tempo e geração de música.

Como funcionam as feed backward networks?

Assim como as redes neurais de feed forward, as redes neurais recorrentes utilizam dados de treinamento para o aprendizado, ou seja, são considerados algoritmos de aprendizagem supervisionada. Eles se distinguem por sua “memória”, pois obtêm informações de entradas anteriores para influenciar a entrada e a saída atuais.

Algumas redes recorrentes derivadas da rede perceptron multicamadas (multilayer perceptron) são: Rede de Hopfield, Rede de Jordan e Rede de Elman. Essas redes inspiraram os modelos recorrentes mais atuais, como as redes neurais recorrentes (Recurrent Neural Network) e LSTM (Long Short Term Memory).

Enquanto as redes neurais profundas tradicionais assumem que as entradas e saídas são independentes umas das outras, a saída das redes neurais recorrentes depende dos elementos anteriores dentro da sequência. Embora eventos futuros também sejam úteis para determinar a saída de uma determinada sequência, as redes neurais recorrentes unidirecionais não podem explicar esses eventos em suas previsões.



Recurrent Neural Network (RNN) como exemplo de redes recorrentes.

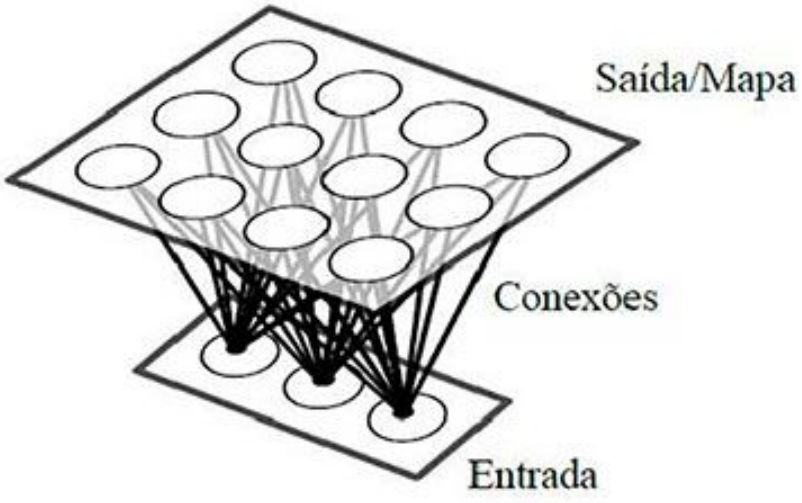
Rede competitiva

Assista ao vídeo a seguir para entender como criar uma rede neural competitiva.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Esse tipo de rede neural é composto por duas camadas, a camada de entrada, também conhecida como "fontes", e a camada de saída, também conhecida como "grau". Apenas o neurônio vencedor é disseminado (ativado) ao final de cada ciclo, o que é determinado pelo grau de similaridade entre os requisitos de entrada e o grau de neurônios. Essa classe de representações utiliza algoritmos para aprendizado competitivo. O "Mapa Auto-organizável", ou rede de Kohonen, é a rede mais conhecida desse tipo, ilustrada na imagem a seguir.



Rede de Kohonen.

Nessa arquitetura, as conexões entre as camadas podem gerar n números de estruturas diferentes.

A rede é chamada de amplamente conectada ou rede direta quando ela possui todas as saídas dos neurônios de uma camada conectadas com todos os neurônios da camada seguinte. Quando o sinal de saída de um neurônio servir como sinal de entrada para um ou mais neurônios na mesma camada ou em alguma camada anterior, a rede possui uma característica denominada realimentação (feedback). A presença desses laços de realimentação tem grande impacto na capacidade de aprendizagem da rede.

Verificando o aprendizado

Questão 1

Redes perceptron de multicamadas são redes diretas que possuem uma ou mais camadas de neurônios entre as camadas de entrada e saída, chamada de camada oculta. Essa camada adiciona um poder maior em relação às redes perceptron de camada única, que classificam apenas padrões linearmente separáveis, sendo os neurônios ocultos responsáveis por capturar a não linearidade dos dados. Essa descrição corresponde à(ao)

A

função de ativação.

B

feed backward networks (redes recorrentes).

C

feed forward.

D

backpropagation.

E

rede competitiva.

A alternativa C está correta.

A arquitetura feed forward são redes diretas, sendo a mesma usada nas redes perceptron de multicamadas.

Questão 2

Em quais redes a saída de um neurônio é aplicada como entrada no próprio neurônio e/ou em outros neurônios de camadas anteriores, ou seja, há ocorrência de realimentação? Estas são identificadas por seus loops de feedback?

A

Feed forward (redes diretas).

B

Rede competitiva.

C

Backpropagation.

D

Feed backward networks (redes recorrentes).

E

Função de ativação.

A alternativa D está correta.

Nas redes feed backward networks, a saída de um neurônio é aplicada como entrada no próprio neurônio ou em outros neurônios de camadas anteriores.

Rede perceptron

Assista ao vídeo a seguir para aprender sobre o funcionamento da rede perceptron.

Conteúdo interativo

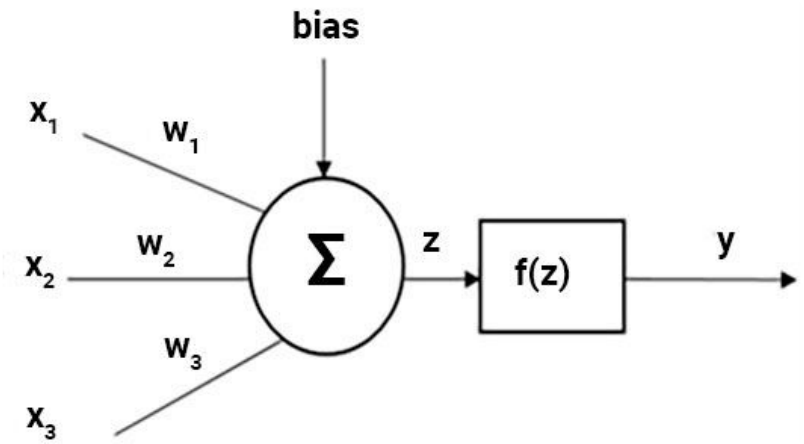
Acesse a versão digital para assistir ao vídeo.

Definição

O perceptron é um tipo de neurônio artificial. Ele foi desenvolvido no fim da década de 1950. Atualmente, o perceptron é muito pouco utilizado, mas esse modelo foi a base para o surgimento de outros modelos mais avançados.

O perceptron, ou perceptron de camada única, é o modelo mais simples de redes neurais artificiais. Um perceptron funciona de maneira equivalente a um neurônio. Esse tipo de rede neural apresenta um conjunto de neurônios de entrada e um conjunto de neurônios de saída, não apresentando nenhuma camada intermediária.

Na imagem a seguir, mostramos um exemplo da rede perceptron.



Exemplo de rede perceptron.

Funcionamento

Um perceptron é um algoritmo usado para aprendizado supervisionado de classificadores binários. Os classificadores binários decidem se uma entrada, geralmente representada por uma série de vetores, pertence a uma classe específica. O aprendizado supervisionado é baseado na regressão básica e classificação.

Exemplo

O humano fornece um banco de dados e ensina a máquina a reconhecer o que é uma bicicleta, por exemplo, entre padrões e semelhanças. A cor e tamanho podem variar, mas a máquina aprende que uma bicicleta possui pedais, duas rodas, guidão e outros elementos-chave.

Em outras palavras, um perceptron é uma rede neural de camada única que consiste em quatro partes principais, incluindo valores de entrada, pesos e viés, soma de pesos e uma função de ativação.

O processo começa da seguinte forma:

- Pegam-se todos os valores de entrada (input) em um perceptron e multiplicando-os por seus pesos que são atribuídos a essas entradas.
- Em seguida, todos esses valores multiplicados são somados para criar a soma ponderada. Esses pesos começam como valores aleatórios e, à medida que a rede neural aprende mais sobre o tipo de dados de entrada, a rede ajusta os pesos com base em qualquer erro na categorização resultante dos pesos anteriores.
- A soma ponderada é então aplicada à função de ativação, produzindo a saída do perceptron.

A função de ativação desempenha o papel integral de garantir que a saída seja mapeada entre os valores necessários, como (0,1) ou (-1,1). O peso (weight) de uma entrada é indicativo da força de um nó.

O valor de viés (bias) de uma entrada oferece a capacidade de deslocar a curva da função de ativação (function activate) para cima ou para baixo.

Como uma forma simplificada de uma rede neural, especificamente uma rede neural de camada única, os perceptrons desempenham um papel importante na classificação binária. Isso significa que o perceptron é usado para classificar os dados em duas partes, portanto, binários. Por esse motivo são também chamados de classificadores binários lineares.

Vejamos algumas limitações dessa rede:

Um único perceptron consegue resolver somente funções linearmente separáveis.

Em funções não linearmente separáveis, o perceptron não consegue gerar um hiperplano para separar os dados.

Rede perceptron multicamadas (multilayer perceptrons)

Assista ao vídeo a seguir para aprender sobre a rede perceptron multicamadas (multilayer perceptrons).

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Definição

Redes perceptron de multicamadas (multilayer perceptron) são redes diretas (feed forward) que possuem uma ou mais camadas de neurônios entre as camadas de entrada e camada de saída. Essas camadas intermediárias são chamadas de camada oculta e são responsáveis por adicionar um maior poder em relação às redes perceptron de camada única.

> As camadas ocultas são responsáveis por capturar a não linearidade dos dados, resolvendo assim uma importante limitação da rede perceptron de camada única, que classifica apenas padrões linearmente separáveis.

A rede perceptron de multicamadas é uma generalização das redes perceptron, e assim como esta, são treinadas de forma supervisionada, por meio da regra de aprendizagem que minimiza o erro.

Veja, a seguir, o que ocorre nesses dois tipos de aprendizagem:

1

No aprendizado supervisionado

Um modelo é treinado com dados de um conjunto de dados rotulado, que consiste em um conjunto de recursos (atributos) e um rótulo. Normalmente, é uma tabela com várias colunas representando recursos/atributos e uma coluna final para o rótulo. O modelo então aprende a prever o rótulo para exemplos não vistos. Os algoritmos mais simples de aprendizagem de máquina são os de aprendizado supervisionado.

2

No aprendizado não supervisionado

Um conjunto de dados é fornecido sem rótulos e um modelo aprende propriedades úteis da estrutura do conjunto de dados. O modelo encontra padrões e aufere conclusões dos dados não rotulados, ou seja, não é indicado ao modelo o que ele deve aprender. Os algoritmos no aprendizado não supervisionado são mais difíceis do que no aprendizado supervisionado, pois temos pouca ou nenhuma informação sobre os dados. Tarefas de aprendizagem não supervisionadas geralmente envolvem o agrupamento de exemplos semelhantes (clustering), redução de dimensionalidade e estimativa de densidade.

Funcionamento

No perceptron de multicamadas, as entradas são combinadas com os pesos iniciais em uma soma ponderada e submetidas à função de ativação e, assim, cada combinação linear é propagada para a próxima camada.

Cada camada alimenta a próxima com o resultado de sua computação, percorrendo todas as camadas ocultas (hidden layers) até a camada de saída (output layer). Considere as seguintes situações:

- Se o algoritmo computasse apenas as somas ponderadas em cada neurônio, propagasse os resultados para a camada de saída e parasse ali, não seria capaz de aprender os pesos que minimizam a função de custo (cost function).

- Se o algoritmo computasse apenas uma iteração, não haveria aprendizado real. Nesse contexto entra o conceito de backpropagation (retropropagação).

Backpropagation é o mecanismo de aprendizado que permite as redes perceptron de multicamadas ajustar iterativamente os pesos na rede, com o objetivo de minimizar a função de custo. O princípio por trás disso é que a mudança de peso sináptico acontece em consideração ao gradiente local da função de erro.

Dessa forma, cada neurônio pode definir a contribuição do seu peso para o erro cumulativo da rede. A regra de aprendizagem mais simples é o método descent gradiente (descida do gradiente), que é a mudança de peso sináptico proporcionalmente à sua contribuição para o erro cumulativo. Gradient descent é normalmente a função de otimização usada na rede perceptron de multicamadas (multilayer perceptron).

Embora exista uma série de algoritmos que abordam os diferentes problemas de aprendizagem, o backpropagation é um dos algoritmos mais eficientes.

Assista ao vídeo a seguir para ver os principais aspectos sobre os mecanismo de aprendizado backpropagation.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Verificando o aprendizado

Questão 1

Dentre os tipos de redes neurais existentes é a arquitetura mais simples, apresentando apenas um conjunto de neurônios de entrada e um conjunto de neurônios de saída, sem haver nenhuma camada de neurônio intermediária. Qual opção de rede atende a essas características?

A

Perceptron de múltiplas camadas.

B

Aprendizagem de máquina (machine learning).

C

Perceptron.

D

Inteligência Artificial.

E

Aprendizagem supervisionada.

A alternativa C está correta.

O perceptron é a arquitetura mais simples, não apresentando nenhuma camada de neurônios intermediária.

Questão 2

As redes perceptrons de múltiplas camadas (multilayer perceptron) são uma generalização da rede perceptron e, assim como esta, é treinada de que forma?

A

De pesos e sinapses.

B

Supervisionada.

C

Com a função de ativação.

D

Não supervisionada.

E

Camada oculta (hidden layer).

A alternativa B está correta.

As redes perceptrons de múltiplas camadas são treinadas de forma supervisionada, ou seja, aprendem uma função que mapeia uma entrada para uma saída com base em pares de entrada-saída de exemplo. Ele infere uma função a partir de dados de treinamento rotulados consistindo em um conjunto de exemplos de treinamento.

Preparação do ambiente

Assista ao vídeo a seguir para aprender o funcionamento da preparação do ambiente.

> Conteúdo interativo
>
> Acesse a versão digital para assistir ao vídeo.

Instalando os pacotes

A fim de que você possa executar o código a seguir, utilize o Jupyter Notebook ou Google Colab. De modo geral, o mundo da computação é colaborativo, sendo comum, quando encontramos erros nas bibliotecas utilizadas, copiarmos o log de erro e procurarmos em qualquer motor de busca. Provavelmente, alguma resposta será retornada em algum fórum de discussão, como Stack Overflow, GitHub, Reddit, entre outros. Não só isso é uma prática comum na comunidade de desenvolvimento de software e computação, como também nos possibilita aprender cada vez mais.

O nome do projeto que vamos criar é “Prevendo a Ocorrência de Diabetes”, vamos realizar a avaliação de um conjunto de dados a fim de verificar sua eficácia.

Inicialmente, você deverá atualizar os pacotes necessários à execução do exemplo prático a ser implementado, então execute o comando a seguir no terminal ou prompt de comando:

```
python
pip install -U nome_pacote
```

Para instalar a versão exata de um pacote, execute o comando abaixo no terminal ou prompt de comando:

```
python
pip install nome pacote==versão desejada
```

Depois de instalar ou atualizar o pacote, reinicie o Jupyter Notebook.

Vamos à instalação do pacote watermark, sendo esse pacote usado para gravar as versões de outros pacotes usados nesse Jupyter Notebook.

```
python
pip install -q -U watermark
```

Os seguintes pacotes serão também necessários:

Matplotlib

É uma biblioteca Python de plotagem 2d, que auxilia a biblioteca matemática NumPy.pip install -q -U matplotlib==3.2.1

OpenCV

É uma biblioteca de código aberto voltada para as áreas de visão computacional e aprendizado de máquina.pip install -q -U cv2==4.2.0

NumPy

É uma biblioteca para a linguagem de programação Python, que suporta o processamento de grandes, multidimensionais arranjos e matrizes, juntamente com uma grande coleção de funções matemáticas de alto nível para operar sobre essas matrizes.pip install -q -U numpy==1.18.4

Importando os pacotes

Os pacotes são uma maneira de estruturar o “espaço de nomes” dos módulos Python, usando “nomes de módulo com pontos”.

Exemplo

O nome do módulo A.B designa um submódulo chamado B , em um pacote chamado A.

Para importar um módulo utilizamos o import. Vejamos o código a ser implementado no notebook:

```
python

# Importando os módulos
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline

# Versões dos pacotes usados nesse jupyter notebook
%reload_ext watermark
%watermark –iversions
```

Carregando o conjunto de dados

Assista ao vídeo a seguir para aprender a carregar o conjunto de dados.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Vamos agora carregar as imagens disponibilizadas neste tema, a fim de que possamos utilizá-la no processamento de redes neurais.

```
python

#Conjunto de Dados do Repositório de Machine Learning da UCI / Kaggle
# https://www.kaggle.com/uciml/pima-indians-diabetes-database/data
# Carregando o dataset
print("===Carregando o dataset===")
df = pd.read_csv("pima-data.csv")
```

Distribuição dos dados

Depois de carregar os dados, podemos visualizar a sua distribuição. A seguir, mostramos o código a ser implementado no notebook:

```
python

# Definindo as classes
diabetes_map = {True : 1, False : 0}
# Aplicando o mapeamento ao dataset
df['diabetes'] = df['diabetes'].map(diabetes_map)
# Verificando como os dados estão distribuídos
print()
print("===Verificando como os dados estão distribuídos===")
num_true = len(df.loc[df['diabetes'] == True])
num_false = len(df.loc[df['diabetes'] == False])
print("Número de Casos Verdadeiros: {0} ({1:2.2f}%)".format(num_true, (num_true/
(num_true + num_false)) * 100))
print("Número de Casos Falsos : {0} ({1:2.2f}%)".format(num_false, (num_false/ (num_true
+ num_false)) * 100))
```

Vejamos a distribuição dos dados:

```
===Verificando como os dados estão distribuídos===
Número de Casos Verdadeiros: 268 (34.90%)
Número de Casos Falsos     : 500 (65.10%)
```

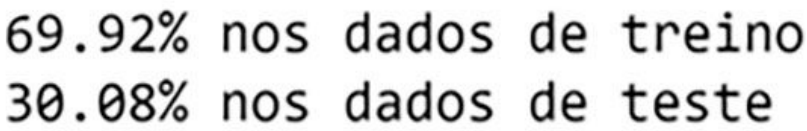
Tela do resultado da distribuição dos dados no Jupyter Notebook.

Pré-processamento dos dados

Nesse pré-processamento, vamos dividir os dados em treino e teste, para treinar o modelo e depois testar sua performance. A seguir, mostramos o código a ser implementado no notebook:

```
python

# Divisão em treino/teste com proporção 70/30

from sklearn.model_selection import train_test_split

# Seleção de variáveis preditoras (Feature Selection)
atributos = ['num_preg', 'glucose_conc', 'diastolic_bp', 'thickness', 'insulin', 'bmi',
'diab_pred', 'age']

# Variável a ser prevista
atrib_prev = ['diabetes']

# Criando objetos
X = df[atributos].values
Y = df[atrib_prev].values

# Definindo a taxa de split
split_test_size = 0.30

# Criando dados de treino e de teste
X_treino, X_teste, Y_treino, Y_teste = train_test_split(X, Y, test_size =
split_test_size, random_state = 42)

# Imprimindo os resultados
print("{0:0.2f}% nos dados de treino".format((len(X_treino)/len(df.index)) * 100))
print("{0:0.2f}% nos dados de teste".format((len(X_teste)/len(df.index)) * 100))
```

Vejamos o resultado:



Tela dos percentuais de treino e teste no Jupyter Notebook.

Implementação do modelo

Assista ao vídeo a seguir para aprender sobre a implementação do modelo, incluindo sua seleção, seu treinamento, e sua avaliação.

> **Conteúdo interativo**
>
> Acesse a versão digital para assistir ao vídeo.

Seleção do modelo

Há uma infinidade de modelos de machine learning disponíveis, cada um voltado ao cumprimento de uma determinada função. Portanto, a escolha do modelo mais adequado deve ser feita de acordo com o objetivo proposto inicialmente. Nesse caso, foi escolhida uma rede de perceptron multicamadas (multilayer perceptron – MLP). A seguir, mostramos o código a ser implementado no notebook:

```
python

#Versão do modelo usando Neural Networks MLPClassifier
from sklearn.neural_network import MLPClassifier
modelo_v7 = MLPClassifier(random_state=42)
```

Treinamento do modelo

A etapa do treinamento é fundamental não apenas para preparar a máquina, mas para aprimorar constantemente suas habilidades de previsão. Dessa forma, a máquina efetivamente aprende com seus erros e torna-se cada vez mais aperfeiçoada. O treinamento pode ser considerado o principal pilar do machine learning. A seguir, mostramos o código a ser implementado no notebook:

```
python

# Treina o modelo
modelo_v7.fit(X_treino, Y_treino.ravel())
mlp_predict_train = modelo_v7.predict(X_treino)
```

Avaliação do modelo

Uma maneira simples de observar o quão bom é um modelo de classificação é usando a acurácia. A acurácia indica uma performance geral do modelo. A acurácia mede o total de acertos considerando o total de observações. Vamos verificar a performance e o equilíbrio na previsão das classes do modelo. A seguir, mostramos o código a ser implementado no notebook:

```
python

mlp_predict_test = modelo_v7.predict(X_teste)

print("=========================================================")
print("Exatidão (Accuracy) com os dados de treino do modelo MLPClassifier :
{0:.4f}".format(metrics.accuracy_score(Y_treino, mlp_predict_train)))
print("Exatidão (Accuracy) com os dados de teste do modelo MLPClassifier:
{0:.4f}".format(metrics.accuracy_score(Y_teste, mlp_predict_test)))
print()
print("Confusion Matrix - Modelo MLPClassifier")
print("{0}".format(metrics.confusion_matrix(Y_teste, mlp_predict_test, labels = [1, 0])))
print()
print("Classification Report MLPClassifier")
print(metrics.classification_report(Y_teste, mlp_predict_test, labels = [1, 0]))
print("=========================================================")
```

Vejamos a saída:

```
================================================================================
Exatidão (Accuracy) com os dados de treino do modelo MLPClassifier : 0.7914
Exatidão (Accuracy) com os dados de teste do modelo MLPClassifier: 0.7229

Confusion Matrix - Modelo MLPClassifier
[[ 44  36]
 [ 28 123]]

Classification Report MLPClassifier
              precision    recall  f1-score   support

           1       0.61      0.55      0.58        80
           0       0.77      0.81      0.79       151

    accuracy                           0.72       231
   macro avg       0.69      0.68      0.69       231
weighted avg       0.72      0.72      0.72       231

================================================================================
```

Tela do relatório de classificação no Jupyter Notebook.

Vamos entender o resultado.

Matriz de confusão

Uma matriz de confusão (matrix confusion) é uma tabela que mostra as frequências de classificação para cada classe do modelo. Pegando o nosso exemplo, ela vai nos mostrar as frequências de diabetes e não diabetes:

1

Verdadeiro positivo (true positive — TP)

Ocorre quando no conjunto real a classe que estamos buscando foi prevista corretamente. Por exemplo, quando a pessoa está com diabetes e o modelo previu corretamente que a pessoa está com diabetes.

2

Falso positivo (false positive — FP)

Ocorre quando no conjunto real a classe que estamos buscando prever foi prevista incorretamente. Exemplo: a pessoa não está com diabetes, mas o modelo disse que está.

3

Falso verdadeiro (true negative — TN)

Ocorre quando no conjunto real a classe que não estamos buscando prever foi prevista corretamente. Exemplo: a pessoa não estava com diabetes, e o modelo previu corretamente que ela não está.

4

Falso negativo (false negative — FN)

Ocorre quando no conjunto real, a classe que não estamos buscando prever foi prevista incorretamente. Por exemplo, quando a pessoa está com diabetes e o modelo previu incorretamente que ela não está com diabetes.

Matriz de confusão mostrando as seguintes categorias:

	Diabetes	Não diabetes
Diabetes	TP	FP

	Diabetes	Não diabetes
Não diabetes	FN	TN

Matriz de confusão com as categorias Fábio Daudt

Matriz de confusão mostrando os dados reais:

	Diabetes	Não diabetes
Diabetes	44	36
Não diabetes	28	123

Matriz de confusão com as categorias Fábio Daudt

Assim, nosso modelo:

- Previu diabetes 44 vezes corretamente.
- Previu não diabetes 123 vezes corretamente.
- Previu diabetes 36 vezes incorretamente.
- Previu não diabetes 28 vezes incorretamente.

Vejamos alguns conceitos das métricas.

Acurácia

Diz quanto o modelo acertou das previsões possíveis. No contexto visto, nosso modelo teve uma acurácia de 72%, pois acertou 72 de cada 100 previsões. E a razão entre o somatório das previsões corretas (verdadeiros positivos com verdadeiros negativos) sobre o somatório das previsões. Sua fórmula é:

$$\text{accurary} = \frac{TP+TN}{TP+FP+TN+FN} = \frac{\text{predições corretas}}{\text{todas as predições}}$$

Recall

O recall responde a seguinte pergunta: qual proporção de positivos foi identificada corretamente? Em outras palavras, quão bom meu modelo é para prever positivos, sendo positivo entendido como a classe que se quer prever, no nosso contexto, se a pessoa está com diabetes. É definido como a razão entre verdadeiros positivos sobre a soma de verdadeiros positivos com negativos falsos. Sua fórmula é:

$$\text{recall} = \frac{TP}{TP+FN}$$

Precisão

A precisão responde a seguinte pergunta: qual proporção de identificações positivas foi realmente correta? Em outras palavras, o quão bem meu modelo trabalhou. Sua fórmula é:

$$\text{precision} = \frac{TP}{TP+FP}$$

F-score

O f-score nos mostra o balanço entre a precisão e o recall de nosso modelo. Sua fórmula é:

$$2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

Verificando o aprendizado

Questão 1

Um modelo de rede neural é a equação final gerada por meio de um algoritmo de machine learning, que será utilizada para definir os valores de saída a partir de novos dados apresentados, sendo esses dados inseridos em uma equação e retornando o resultado do modelo. Assim, podemos ver que o que determina o desempenho de um modelo é o desenvolvimento dessa equação, que acontece por meio do algoritmo utilizado. Em qual etapa da implementação de uma solução utilizando a tecnologia de redes neurais é definido o referido algoritmo?

A

Teste do modelo

B

Avaliação o modelo

C

Seleção do modelo

D

Treinamento do modelo

E

Relatório de classificação

A alternativa C está correta.

Na etapa de seleção do modelo, o cientista de dados seleciona o modelo a fim de treinar e testar os dados a serem utilizados, variando os respectivos parâmetros e buscando aqueles que apresentam o melhor desempenho.

Questão 2

As redes neurais são compostas por neurônios que calculam funções matemáticas, sendo um método para ser usado quando não há uma forma determinística de se resolver um problema em qualquer área do conhecimento. Um projeto de redes neurais possui determinadas etapas, analise as afirmativas a seguir.

i. No pré-processamento, vamos dividir os dados em treino e teste, para treinar o modelo e depois testar sua performance.

II. A escolha do modelo mais adequado deve ser feita de acordo com o objetivo proposto inicialmente.

II. A etapa do teste é fundamental não apenas para preparar a máquina, mas para aprimorar constantemente suas habilidades de previsão.

Assinale a seguir a alternativa correta cujas afirmativas relacionam-se com computação nas nuvens.

A

As opções I, II e III.

B

Apenas as opções II e III.

C

Apenas as opções I e II.

D

Apenas as opções I e III.

E

Apenas a opção III.

A alternativa C está correta.

O treinamento da rede considera que, seguindo o algoritmo de treinamento escolhido, serão ajustados os pesos das conexões. É importante considerar, nessa fase, alguns aspectos, tais como a inicialização da rede, o modo de treinamento e o tempo de treinamento.

Considerações finais

Como vimos, o primeiro algoritmo de rede neural artificial era muito simples, comparado ao estado da arte atual. O perceptron é uma rede neural com apenas um neurônio e só pode entender relações lineares entre os dados de entrada e saída fornecidos.

Porém, com as redes perceptron multicamadas (multilayer perceptron), os horizontes são ampliados e agora essa rede neural artificial pode ter muitas camadas de neurônios, e pronta para aprender padrões mais complexos.

Por fim, apresentamos um projeto passo a passo de uma rede perceptron multicamadas para previsão de diabetes.

Podcast

Podemos considerar as redes neurais como um paradigma de programação? Como funciona uma rede neural artificial? Quais as principais aplicações das redes neurais artificiais? Para saber as respostas dessas perguntas e muitas mais, ouça o podcast.

Conteúdo interativo

Acesse a versão digital para ouvir o áudio.

Explore +

Para complementar seus estudos, sugerimos que pesquise os seguintes capítulos dentro do Deep Learning Book:

- Capítulo 4 – O Neurônio, Biológico e Matemático.
- Capítulo 6 – O Perceptron – Parte 1.
- Capítulo 7 – O Perceptron – Parte 2.
- Capítulo 8 – Função de Ativação.
- Capítulo 10 – As 10 Principais Arquiteturas de Redes Neurais.

Referências

CHRISTOFIDIS, C. **Awesome Deep Learning**. GitHub, 14 nov. 2022. Consultado na internet em: 05 jan. 2023.

DEEP Learning Book. **Learning Book**. Consultado na internet em: 05 jan. 2023.

FLECK, L. *et al*. **Redes Neurais Artificiais**: princípios básicos. Revista Eletrônica Científica Inovação e Tecnologia, UTFPR, 2016.

GOOGLE. **Classification**: Precision and Recall. Machine Learning Crash Course. Google Developers. Consultado na Internet em: 05 jan. 2023.

NIELSEN, M. A. **Neural Networks and Deep Learning**. Publicado em: dez. 2019. Consultado na Internet em: 05 jan. 2023.

RUSSEL, S.; NORVIG, P. **Artificial Intelligence**: a modern approach. 4. ed. [S.l.]: Prentice Hall, 2018.

TRASK, A. W. **Grokking deep learning**. Shelter Island, NY: Manning, 2019.