Big Data Analytics

Conceitos de Big Data Analytics e demonstração de modelos subsimbólicos, redes neurais com Tensorflow e Python.

Prof. Fernando Cardoso Durier da Silva

1. Itens iniciais

Propósito

Compreender conceitos e processos de Big Data Analytics, assunto que permeia desde a análise de dados até a própria inteligência artificial que a instrumentaliza, e que tem ganhado popularidade com o avanço das tecnologias para tratamento de Big Data.

Preparação

Para a compreensão e reprodução dos exemplos, é imprescindível que você tenha instalado, em sua máquina de estudos, o Python na versão 3.8 ou superior, e as bibliotecas Pandas, Numpy, Sklearn e Plotly. Será necessária também uma ferramenta para execução de notebooks, como Jupyter, que pode ser instalado como uma biblioteca do Python. Alternativamente, você poderá também executar os códigos dos exemplos diretamente em um ambiente de execução como o Google Colab.

Objetivos

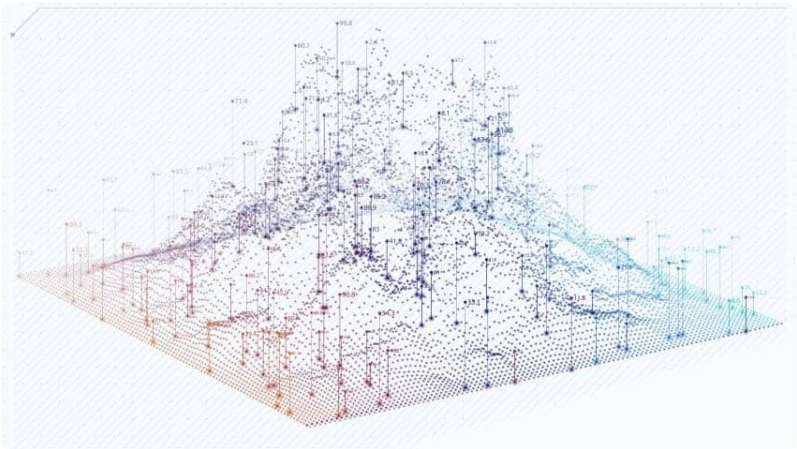
- Reconhecer o processo de KDD no contexto da inteligência artificial.
- Aplicar técnicas de aprendizado de máquina com Scikit-Learn.
- Aplicar técnicas de aprendizado profundo com TensorFlow.

Introdução

Atualmente, vivemos em um mundo imerso em dados, no qual os termos ciência de dados, business analytics, inteligência artificial e modelos de aprendizado de máquina são comuns e, muitas vezes, confundidos como sinônimos. Mas sabemos que não é bem assim!

Primeiro, temos que compreender o porquê de a área de dados estar tão aquecida e demandando tanto do mercado de Tecnologia de Informação e áreas afins. Assim, será possível entender como as "máquinas inteligentes" aprendem, qual o objetivo para tal, bem como os benefícios que podem trazer para a sociedade em suas diversas aplicações.

Devemos reconhecer que existem diferentes tipos de inteligência artificial e um enquadramento de métodos e técnicas compatíveis com cada tipo de problema específico. Desse modo, poderemos escolher melhor em qual estratégia apostar, bem como aprender a implementar alguns desses modelos usando bibliotecas existentes na linguagem de programação Python.



Orientação sobre unidade de medida

Em nosso material, unidades de medida e números são escritos juntos (ex.: 25km) por questões de tecnologia e didáticas. No entanto, o Inmetro estabelece que deve existir um espaço entre o número e a unidade (ex.: 25

km). Logo, os relatórios técnicos e demais materiais escritos por você devem seguir o padrão internacional de separação dos números e das unidades.

Motivação

Atualmente, a área de ciência de dados tem ganhado muita popularidade em virtude dos resultados de análise de dados que são entregues de forma rápida, eficiente e com custo reduzido. Mas, para entendermos melhor isso, precisamos voltar um pouco no tempo. Vamos juntos? Tudo começou com a internet...

A princípio, a internet surgiu como uma pesquisa científica na área de defesa, que seria utilizada para persistir nos meios de comunicação e na inteligência militar na **década de 1960**, ainda como ARPANET (*Advanced Research Projects Agency Network*), patrocinada pelo Departamento de Defesa do governo dos EUA.

Símbolo da World Wide Web.

Com a disseminação do seu uso acadêmico por universidades envolvidas na pesquisa, a partir da **década de 1970**, a internet começou a ganhar os contornos que conhecemos atualmente. Impulsionada pela popularização dos computadores pessoais nos **anos 1980**, a internet foi difundida na sociedade na **década de 1990**, com o advento da World Wide Web, ou simplesmente Web.

> Da década de 1990 até os dias atuais, a Web evoluiu e passou por várias fases. Vamos conhecê-las?

A Web 1.0 era caracterizada por ser a web do consumo de informações na rede, em que os detentores da informação eram institutos governamentais, universidades e empresas de tecnologia ligadas ao governo. Seu papel era prover informações relevantes sobre assuntos relacionados às suas áreas de interesse. Normalmente, os usuários consumiam os dados de fontes específicas de acesso, como bibliotecas, centros de pesquisa e gabinetes governamentais.

Em meados dos **anos 2000**, entramos na Web 2.0, denominada Web Semântica, em que o uso do computador pessoal já era uma realidade nas tarefas domésticas da sociedade moderna.

Big Data World Cloud.

Começou-se a observar um fenômeno interessante, com a colaboração cibernética e o surgimento das primeiras redes sociais, de modo bem rudimentar, inicialmente em forma de blogs, fóruns etc., assim como o crescimento de aplicações de comércio eletrônico com recursos de relacionamento com o cliente (CRM).

Dessa maneira, os usuários, que antes só consumiam informação, passaram a contribuir também para a produção de dados nesse novo mundo digital que ali estava se formando.

Foi então que, por volta do **ano de 2007**, ocorreu uma mudança na forma de acesso à internet, tão radical quanto a revolução da WWW nos anos 1990. A tecnologia de hardware, que vinha crescendo com recursos computacionais cada vez mais potentes, deu um salto com a evolução dos aparelhos telefônicos móveis para os smartphones. Isso propiciou a geração e o consumo de um grande volume de dados, com variedade de

formatos, propagando-se em alta velocidade. A esse grande fenômeno de produção e disseminação de dados atribuímos o nome de Big Data, com os seus três Vs: Volume, Variedade e Velocidade.

Saiba mais

Esses dados eram gerados pela colaboração humana com o ciberespaço, por meio da troca de informação entre os sistemas da Web, graças aos dados produzidos pelos novos dispositivos pessoais conectados à rede. Com Big Data e o avanço tecnológico, entramos na Web 3.0, da hiperpersonalização.

Na Web 3.0, destacam-se os algoritmos e motores de busca que tentam aproveitar essa massa de dados para extrair padrões e fazer recomendações aos usuários, ou então incorporá-las ao negócio de grandes organizações capazes de dominar essa tecnologia. Isso reaqueceu a área de inteligência artificial, fazendo surgir os conceitos de mineração de dados e ciência de dados.

Inteligência artificial.

Definição de inteligência artificial (IA)

Segundo Russel e Norvig (2013), a Inteligência Artificial é uma área de pesquisa da computação dedicada a buscar métodos ou dispositivos computacionais que possuam ou multipliquem a capacidade racional do ser humano de resolver problemas, pensar ou, de forma ampla, ser inteligente. A literatura também chama esse campo de pesquisa como **inteligência de agentes ou agentes inteligentes/cognitivos**.

A máquina simula o comportamento humano por meio de Processamento da Linguagem Natural, Visão Computacional ou Processamento de Imagens, Representação do Conhecimento e Raciocínio com Lógica (de predicados, modal, difusa etc.), e sistemas de agentes. Definindo esses sistemas como agentes, estes são capazes de aprender com o meio em que vivem, mediante observações feitas por intermédio de sensores e ações tomadas por atuadores, baseadas em processamentos lógicos que ocorrem graças ao aprendizado de máquina.

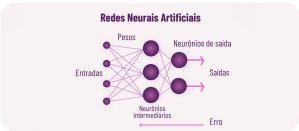
Na área de inteligência artificial, existem dois tipos principais: a **IA simbólica** e a **IA subsimbólica ou conexionista**. A seguir, vamos conhecer essas abordagens com mais detalhes.

IA simbólica

Segundo Wnek e Michalski (1992), o conhecimento representado por regras baseadas em lógica ou árvores de decisão é relativamente fácil de compreender e de fazer associações com o modelo de raciocínio lógico humano. Essa é a **abordagem simbólica**, na qual problemas se transformam em fórmulas e expressões lógicas, e o raciocínio ocorre a partir de operações lógicas feitas sobre símbolos, contra axiomas e regras já conhecidos.

IA subsimbólica ou conexionista

Para a abordagem subsimbólica, o conhecimento é representado a partir de modelos de sistemas classificadores ou redes neurais. A necessidade da IA subsimbólica ou conexionista surgiu da decepção com a IA simbólica, que não conseguia lidar bem com processamentos robustos e flexíveis, segundo Babbar *et al.* (2018), além do fato de ocultar o processamento de raciocínio de usuários não autorizados.



Aprendizado de máquina

O aprendizado de máquina é a forma como implementamos os processos cognitivos que a inteligência artificial tenta simular, pois assim como nós temos diferentes processos cognitivos para determinadas tarefas do dia a dia, a IA também tem para a resolução de classes de problemas similares.

Os métodos de aprendizado de máquina conhecidos são:

1

Aprendizado supervisionado

É o aprendizado no qual o modelo aprende a partir de exemplos positivos e negativos pré-rotulados, para que a máquina possa mapear o padrão de entrada e saída esperado.

2

Aprendizado não supervisionado

É o aprendizado autodidata dos modelos, ou seja, ao invés de o modelo ter um conjunto pré-mapeado/rotulado, é apresentado a ele um conjunto sem marcações prévias. E o modelo, por meio de análise de critérios de similaridade, encontra por si só rótulos e padrões automaticamente.

3

Aprendizado semissupervisionado

É o aprendizado no qual uma parte dos dados é rotulada previamente e outra é não rotulada. A maioria do conjunto é não rotulada, e o modelo de aprendizado de máquina tenta, a partir do subconjunto menor, generalizar padrões para o conjunto maior ainda não rotulado.

4

Aprendizado por reforço

É o aprendizado com o ambiente. O agente inteligente aprende políticas de ações com base em interações com o ambiente no qual ele é implementado, a partir de recompensas ou punições, dependendo de cada ação.

As técnicas de aprendizado de máquina mais populares são:

Classificação

Técnica do aprendizado de máquina supervisionado que faz com que o modelo consiga entender como categorizar observações do conjunto de dados com base em registros históricos, bem como suas características.

Regressão

Contraparte numérica da classificação categórica. Também é uma técnica de aprendizado supervisionado, em que o modelo aprende o mapeamento de entrada e saída para inferir um valor numérico ao invés de uma classe categórica.

Agrupamento

Técnica de aprendizado não supervisionado, em que o modelo, de forma autodidata, aprende a separar as observações seguindo critérios de similaridades predefinidos, com o intuito de formar grupos de observações similares.

Big Data e o Boom da IA

A IA não é uma área de pesquisa recente, e sua origem remonta aos anos 1950, antes mesmo do surgimento da internet como conhecemos. Mas, como mencionamos, no princípio, a IA era rudimentar e não trazia, aos olhos das grandes organizações, tantos benefícios que justificassem investimento.

> Graças ao advento da Web, em especial com a Web 2.0, e ao boom de tecnologia de 2007, começou-se a produzir um volume de dados variados e em uma velocidade nunca vista de disseminação.

Esses fatores impulsionaram a inteligência artificial de uma forma que não era possível quando foi concebida inicialmente, fazendo com que a área ganhasse força, popularidade e credibilidade nos tempos atuais.

Florescimento da IA.

Com mais pesquisas e avanços na tecnologia, e com o surgimento de novos problemas contemporâneos oriundos do cyberespaço e da nova sociedade conectada, as pesquisas em IA avançam cada vez mais, evoluindo e nos mostrando agentes mais capazes e inteligentes.

Dica

Não podemos esquecer a ciência de dados, que é a análise e resolução de um problema de negócio (organizacional, industrial e/ou acadêmico), por meio das técnicas de aprendizado de máquina e de processamento de dados previstas na mineração de dados.

Ou seja, a ciência de dados utiliza o instrumental do aprendizado de máquina para descobrir os padrões escondidos na grande massa de dados, e dali tirar conclusões para problemas ocorridos em um negócio. Isso faz com que o sistema se adapte ao ambiente da organização, por meio da observação dos dados e da identificação de soluções que maximizem o ganho e/ou minimizem o custo. Assim, a concretização da IA no mercado de trabalho praticamente vem se consolidando.

Descoberta de Conhecimento em Bases de Dados

> Você sabe o que significa mineração de dados e como se encaixa na IA?

Antes de falarmos de mineração de dados, vamos dar alguns passos para trás e revisitar os conceitos de Dados, Informação, Conhecimento e Sabedoria, bem como o de Sistemas de Informação.

Dado

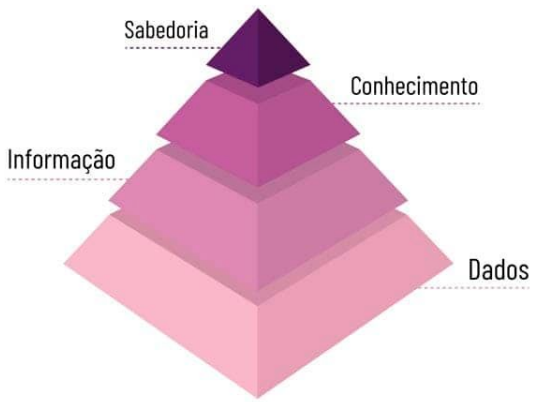
Um dado é um símbolo ou valor que isoladamente não significa muita coisa, por exemplo, o dado 38, isoladamente, não significa muito.

Informação

Informação é o dado processado, ou seja, o dado ao qual é feita uma atribuição. Por exemplo, se dissermos que 38 é a temperatura corporal de alguém, isso começa a fazer sentido, não é? E se tivermos a relação idade=50, temperatura=38, gênero=Masculino, peso=100kg, altura=1,70m? Você deve ter percebido que, aqui, temos as informações de um senhor de 50 anos, acima do peso, e com febre.

Estamos dando significado ao dado com esse processamento simples. Agora, se incluirmos uma variável de contexto, por exemplo, "esteve_em_contato_com_pessoas_com_COVID=sim", mediante análise de casos similares em uma base de dados, poderíamos inferir que esse senhor está com covid. Isso é **conhecimento**, ou seja, o processamento de informação e a descoberta de padrões.

Por fim, a resposta com prescrição médica relevante para supressão dos sintomas e recuperação é a **sabedoria**, ou seja, o conhecimento processado que resulta em uma ação prática para resolver o problema.



Dados – Informação – Conhecimento – Sabedoria .

> E o que é um sistema de informação?

Um sistema é um conjunto de partes, as quais individualmente podem não ser dotadas de muitas funcionalidades, mas, em um sistema, trabalham juntas para atingir um objetivo final, por exemplo, nossos sistemas vitais, ou um sistema que cadastra dados de pacientes e os salva em um banco de dados, para outro sistema explorar e descobrir padrões.

Um sistema de informação é isto: **uma coleção de partes que recebe dados como insumo, processa-os por meio de dinâmicas internas das partes e devolve informação relevante ao usuário final.**

Mineração de dados é um sistema de informação nesse sentido, mas o termo se confunde muito com o real nome do processo de descobrir esses padrões, que é **Descoberta de Conhecimento em Bases de Dados** (KDD, do inglês *Knowledge Discovery in Databases*).

KDD nada mais é do que um processo que tem uma definição similar à de sistema, ou seja, um conjunto de eventos ou partes, na maioria das vezes subsequentes, que recebem um insumo e devolvem um resultado processado dele.

O KDD pode ser dividido em seis etapas. São elas:

1ª etapa – Coleta de dados

Nesta etapa, são capturados dados de uma organização para a resolução de um problema. Aqui, temos Big Data, uma massa de dados que traz muita informação relevante, mas, com certeza, também muita informação irrelevante que pode até atrapalhar a descoberta de padrões.

2ª etapa – Seleção

Nesta etapa, começamos a separar o joio do trigo. Assim como a operação de projeção em bases de dados, aqui, escolhemos quais características dessa grande massa de dados devemos considerar, com base na sua relevância para o negócio.

3ª etapa – Pré-processamento

Projetados os dados, chegamos à etapa de pré-processamento. Normalmente, é a fase mais longa desse processo, em que se espera que um cientista de dados passe 70% do seu tempo de trabalho. Nessa etapa, ocorrem limpeza nos dados, remoção de dados faltantes e corrompidos, incorporação de dados de outras bases etc.

4ª etapa – Transformação

Na etapa de transformação de dados, o cientista regulariza a escala dos dados, reduz a dimensionalidade do conjunto e aplica outras técnicas de seleção de atributos automática. Isso é necessário para obter o conjunto de dados mais consistente e regular o possível para que os algoritmos possam lidar com o mínimo de viés e ruído.

5ª etapa – Mineração de dados ou descoberta de padrões

Aqui, o conjunto de dados é passado para um algoritmo de aprendizado de máquina compatível com o enquadramento do problema de dados, bem como da técnica em questão. Esta vai estudar uma grande parte do conjunto e medir seus conhecimentos em outra parte, quando falamos de aprendizado supervisionado, um dos métodos de aprendizado de máquina, por exemplo. Feito isso, teremos um modelo de solução dos nossos problemas.

6ª etapa – Avaliação e apresentação de resultados

Na última etapa do processo de KDD, vamos entender quais padrões o modelo do algoritmo escolhido conseguiu aprender e com qual confiança ele o fez, ou seja, vamos avaliar se o modelo é confiável e de fato aprendeu, ou apenas decorou, ou simplesmente não aprendeu nada. Fazemos isso por meio das métricas e visualizações, em que as métricas mostram o comportamento do modelo e as visualizações nos permitem ver o impacto mais concreto para o negócio.

Ao final do processo, obtemos o que precisávamos: **um padrão descoberto a partir de dados que provêm informações**, resultando em conhecimento, que poderá ser utilizado por outros sistemas para resolver problemas, tornando-se, assim, sabedoria.

Processo de KDD.

Processo CRISP-DM e a diferença para o KDD

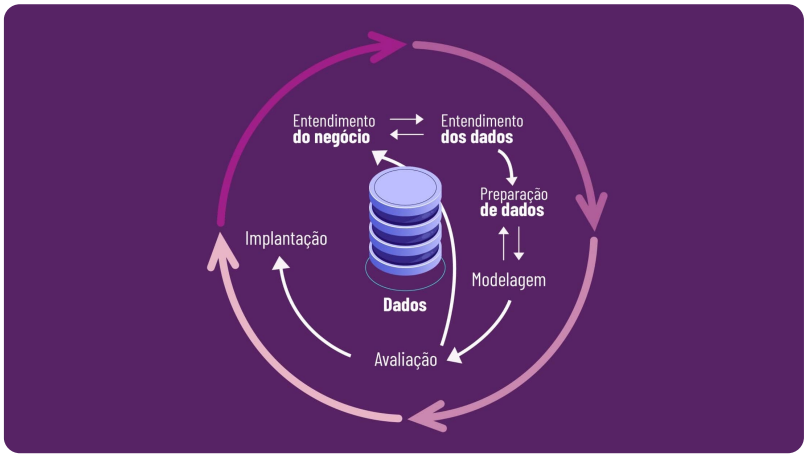
O processo do KDD tem um irmão, que na realidade muitos pesquisadores consideram um subconjunto dele, que é o CRISP-DM (do inglês *Cross Industry Standard Process for Data Mining*).

Igualmente, o processo CRISP-DM possui seis etapas, mas tem outro foco. São elas:

- **Entendimento do Negócio**
- **Entendimento dos Dados**

- **Preparação de Dados**
- **Modelagem**
- **Avaliação**
- **Implantação**

Como você pode observar, não só o número de etapas é bem parecido, mas também os nomes de suas fases. E isso não é coincidência! O processo do CRISP-DM é também conhecido como o KDD da indústria, pois tem um foco maior no negócio no começo e no fim.



Processo CRISP-DM.

Vejamos primeiro as semelhanças entre CRISP-DM e KDD, identificando as seguintes equivalências entre as etapas:

CRISP-DM	KDD
Preparação dos Dados	Seleção, Pré-Processamento e Transformação
Modelagem	Mineração de dados
Avaliação	Avaliação e Apresentação de Resultados

Comparação entre os processos CRISP-DM e KDD.

> Essas etapas têm as mesmas operações de suas contrapartes.

Agora, vamos às diferenças.

O modelo CRISP-DM traz o Entendimento do Negócio e Entendimento dos Dados no lugar da Coleta de Dados. Também traz a implantação (deployment) como atividade a seguir da avaliação, e isso é um diferencial importante.

O CRISP-DM, por ser um processo voltado para a indústria, tem como entrega de valor final o artefato do modelo de aprendizado de máquina. Esse modelo será capaz de resolver os problemas propostos pelo negócio na primeira etapa. Já o KDD não tem a obrigação de entregar um artefato, mas sim o conhecimento apenas, que já é suficiente.

Dica

Esta é a maior diferença entre os dois processos: a entrega de valor final.

Por isso, no fim das contas, a indústria prefere o CRISP-DM, enquanto a academia preza pelo KDD. Lembre-se de que esse é um comportamento esperado, mas não é uma obrigatoriedade, pois, se a organização se contenta com a descoberta de padrão para um nível estratégico, o uso do KDD é suficiente. Em contrapartida, ao aprender o KDD, aprende-se o fundamento, enquanto o CRISP-DM pode ser encarado como uma instanciação do KDD na indústria.

O processo CRISP-DM e o KDD

Veja agora um resumo do módulo, apresentando um panorama da Inteligência e do processo de KDD, usando o CRISP-DM como exemplo. Vamos lá!

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Vem que eu te explico!

Os vídeos a seguir abordam os assuntos mais relevantes do conteúdo que você acabou de estudar.

Aprendizado de máquina

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Big Data e o Boom da IA

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Verificando o aprendizado

Questão 1

Quando falamos do trinômio de dados, informação e conhecimento, estamos nos referindo ao conceito fundamental de sistemas de informação. No processo de KDD, qual dos componentes deste trinômio estamos tentando alcançar e em qual etapa o obtemos?

A

Dados; Seleção.

B

Conhecimento; Seleção.

C

Informação; Transformação.

D

Conhecimento; Interpretação e Avaliação.

E

Conhecimento; Pré-Processamento.

A alternativa D está correta.

Após a avaliação dos resultados de padrões descobertos, podemos consolidar o conhecimento, que nada mais é do que informação processada durante todo o KDD. As alternativas A e C apresentam aspectos anteriores ao conhecimento. E as alternativas B e E, embora abordem o aspecto correto, fazem referência a etapas do processo muito anteriores à geração de conhecimento.

Questão 2

Em qual fase da Web nasce o tema de Big Data?

A

Web da Informação

B

Web Semântica

C

Web 1.0

D

Web 2.0

E

Web 3.0

A alternativa E está correta.

A partir de 2007, o termo Big Data surgiu devido à produção de um grande volume variado de dados que se propagou de forma veloz com o desenvolvimento dos smartphones e de novos dispositivos conectados em redes.

Motivação

Neste módulo, faremos demonstrações práticas de técnicas de aprendizado de máquina, que faz parte da etapa de mineração de dados ou descoberta de padrões do processo de KDD, correspondente à etapa de modelagem do CRISP-DM.

Para fins de demonstração, escolhemos o método de aprendizado de máquina supervisionado mais comum no mercado e mais intuitivo para compreensão, que é a classificação.

Dica

As técnicas de classificação empregadas nos exemplos são a máquina de vetor de suporte (do inglês Support Vector Machine – SVM) e a árvore de decisão.

Demonstração de classificação com SVM usando Scikit-Learn

Para implementar um classificador SVM em Python, é necessária a instalação das bibliotecas pandas, numpy, matplotlib e scikit-learn. Começaremos com a importação das bibliotecas, e para isso, devemos fazer o seguinte:

```
python

import pandas as pd
 import numpy as np
 from matplotlib import pyplot as plt
```

Em seguida, importaremos o dataset que será explorado em nossos estudos: o dataset Íris, considerado clássico no âmbito do Aprendizado de Máquina, classifica tipos de flores da espécie Íris a partir de características como comprimento de pétala, largura da pétala, comprimento da sépala e largura da sépala (todas em cm). Para isso, o adicionaremos ao nosso código, observando o destaque da coluna de classe (target) das características.

```
python

from sklearn.datasets import load_iris
 data = load_iris()
 iris = pd.DataFrame(data['data'], columns=data.feature_names)
 target = data.target
```

Para a instanciação simples do classificador, podemos adicionar o seguinte bloco de código, observando que o classificador SVM é identificado no scikit-learn a partir do nome SVC (C de Classifier):

```python
#Importando o algoritmo de SVM
 from sklearn.model_selection import cross_val_score
 from sklearn.svm import SVC
 svc = SVC(gamma='auto')
```

Para treinarmos o modelo e conhecermos a sua performance, adicionaremos o bloco de código a seguir. A validação cruzada é um dos modos mais comuns de se treinar nossos modelos, pois ela divide o conjunto de dados em (k-1)/k partições de treinamento e 1/k de teste de maneira circular e interativa, tendo assim todas as 1/k possíveis partições, podendo ser testadas contra o resto.

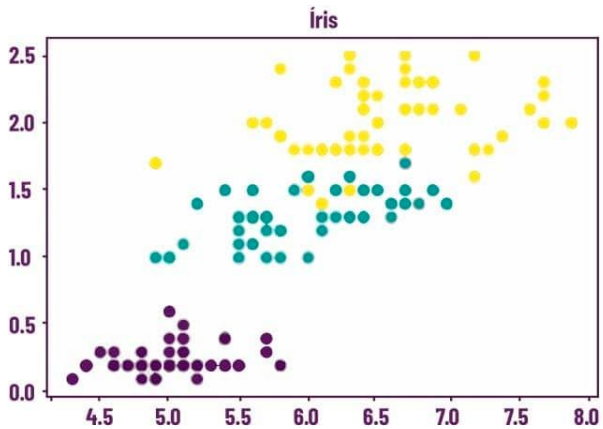
```python
#Testando o modelo 'svc' na nossa base 'iris'
 cv_result = cross_val_score(svc, iris, target, cv=10, scoring='accuracy')
 #Retorna a acurácia em porcentagem do nosso modelo
 print('Acurácia com cross validation:', cv_result.mean()*100)
```

Agora que fizemos o experimento de construir o classificador com os parâmetros padrões da biblioteca, podemos fazer predições. Para tal, vamos treinar nosso modelo com o dataset inteiro e tentar predizer um valor inédito.

```python
svc.fit(iris, target)
 #Prediz a que classe pertencerá a flor com sépala de comprimento 6.9 cm e de largura 2.8 cm, e com pétala de comprimento 6.1 cm e de largura 2.3 cm
 svc.predict([[6.9,2.8,6.1,2.3]])
```

Feito isso, vamos visualizar nossos dados e os hiperplanos definidos pelo modelo. Nossos dados têm o seguinte comportamento:

```python
plt.scatter(iris['sepal length (cm)'], iris['petal width (cm)'], c=target)
 plt.title('Iris')
```

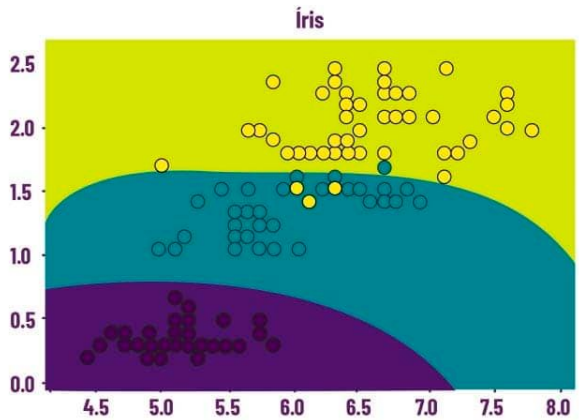


Dispersão dos dados íris.

Como podemos ver no gráfico, nosso problema parece ser linearmente separável, o que pode ser feito com este outro bloco de código:

```
python

#Provavelmente, criando 2 features novas no íris, o svm com 2 features terá mais sucesso,
mas por enquanto usei só
 #sepal length e petal width (os mais relevantes das 4 features já existentes)
 x0_min, x0_max = iris['sepal length (cm)'].min(), iris['sepal length (cm)'].max()
 x1_min, x1_max = iris['petal width (cm)'].min(), iris['petal width (cm)'].max()
 w = x0_max - x0_min
 h = x1_max - x1_min
 x0, x1 = np.meshgrid(np.linspace(x0_min-.1*w, x0_max+.1*w, 300),
                      np.linspace(x1_min-.1*h, x1_max+.1*h, 300))
 svc.fit(iris[['sepal length (cm)', 'petal width (cm)']], target)
 ypred = svc.predict(np.c_[x0.reshape(-1, 1), x1.reshape(-1, 1)])
 ypred = ypred.reshape(x0.shape)
 plt.contourf(x0, x1, ypred)
 plt.scatter(iris['sepal length (cm)'], iris['petal width (cm)'], c=target, s=64,
edgecolors='k')
 plt.title('Iris')
 plt.show()
```



Hiperplanos gerados pelo modelo SVM.

Podemos observar que, de fato, o SVM foi capaz de definir os hiperplanos e, se olharmos bem no hiperplano superior e no hiperplano central desse gráfico, vemos alguns outliers entre eles. No caso, o algoritmo SVM os ignorou estrategicamente para não superajustar sua classificação, conferindo, assim, certa generalização ao classificador.

Demonstração de classificação com árvore de decisão usando Scikit-Learn

Para implementarmos a árvore de decisão, precisaremos apenas das bibliotecas sklearn e matplotlib.

Vamos começar nosso código importando as bibliotecas e funções necessárias:

```python
import matplotlib.pyplot as plt
 from sklearn.datasets import load_iris
 from sklearn.model_selection import cross_val_score
 from sklearn.tree import DecisionTreeClassifier, plot_tree
```

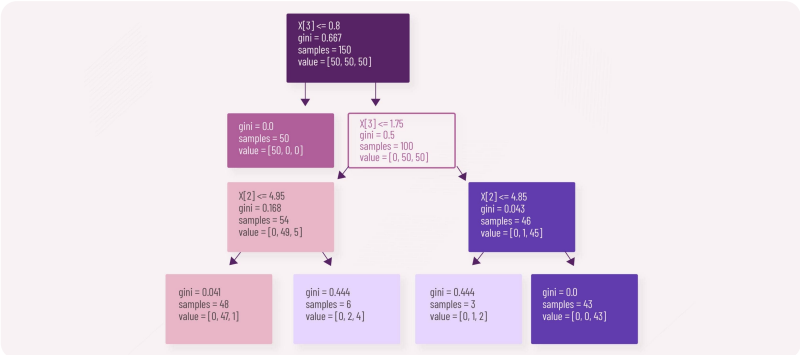
Agora, vamos ao processo de experimentação por meio do treinamento feito com validação cruzada. Basta adicionarmos este bloco de código:

```python
clf = DecisionTreeClassifier(max_depth=3, random_state=0)
 iris = load_iris()
 cross_val_score(clf, iris.data, iris.target, cv=10)
```

Com os dados carregados e o experimento executado para checarmos a possível média de acurácia do modelo treinado com o conjunto, vamos ao treinamento propriamente dito e à visualização do resultado.

```python
clf.fit(iris.data, iris.target)
 plot_tree(clf, filled=True)
 plt.show()
```

Pronto! Com poucas linhas de código, pudemos treinar um classificador de árvore de decisão e visualizar sua árvore resultante.



Árvore de decisão resultante da implementação em Python.

A biblioteca Scikit-Learn no Python

Acompanhe agora uma apresentação genérica da biblioteca Scikit-Learn, destacando as aplicações de aprendizado de máquina para as quais ela é empregada. Vamos lá!

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Vem que eu te explico!

Os vídeos a seguir abordam os assuntos mais relevantes do conteúdo que você acabou de estudar.

Demonstração de Classificação com SVM usando Scikit-Learn

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Demonstração de Classificação com Árvore de Decisão usando Scikit-Learn

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Verificando o aprendizado

Questão 1

Qual a métrica adequada para o problema de aprendizado supervisionado cuja técnica é classificação?

A

Coeficiente de Silhueta

B

Altura

C

Comprimento

D

Acurácia

E

Erro Médio Absoluto

A alternativa D está correta.

A acurácia mede quantos acertos de categorias o modelo teve ao treinar com um conjunto de dados, o que é possível apenas pelo fato de termos um ground truth, que justamente são os rótulos preestabelecidos.

Questão 2

Qual é o método da biblioteca do Scikit-Learn utilizado para resolver um problema de classificação com o algoritmo de árvore de decisão?

A

GaussianNaiveBayes

B

SVM

C

plot_tree

D

DecisionTreeRegressor

E

DecisionTreeClassifier

A alternativa E está correta.

A árvore de decisão é um dos algoritmos mais poderosos do aprendizado de máquina, não só por sua construção e lógica, mas por ser um dos poucos explicados por meio da estrutura que lhe dá nome. Essa estrutura serve para resolver tanto problemas de regressão quanto de classificação. Na hora de instanciar o modelo, é importante usar o DecisionTreeClassifier para resolver problemas de classificação (que é o caso da questão) e DecisionTreeRegression para problemas de regressão. As alternativas A e B se referem a outros modelos, e a alternativa C apresenta a função de imprimir a estrutura de dados do algoritmo (onde está sua explicabilidade).

Definindo Aprendizado Profundo

Como um ramo do aprendizado de máquina, o aprendizado profundo (do inglês *Deep Learning*) é uma das técnicas que podem ser usadas na etapa de modelagem do CRISP-DM, a qual corresponde à etapa de mineração de dados ou descoberta de padrões, que é o coração do processo de KDD.

O aprendizado profundo tenta modelar abstrações de alto nível de dados, usando um grafo com várias camadas de processamento, compostas de várias transformações lineares e não lineares. Em outras palavras, trata-se de um modelo de aprendizado de máquina que decompõe dados complexos em representações mais bem compreendidas pela máquina, como vetores, matrizes e séries temporais.

Saiba mais

Uma das maiores vantagens desse modelo é incorporar em suas camadas escondidas e profundas todo o pré-processamento e a geração de características, que, em modelos clássicos, são feitos em etapas anteriores externas ao modelo. Devido a esse aspecto de múltiplas transformações lineares e aprendizado conexionista, os grafos empregados nessa área são as redes neurais artificiais.

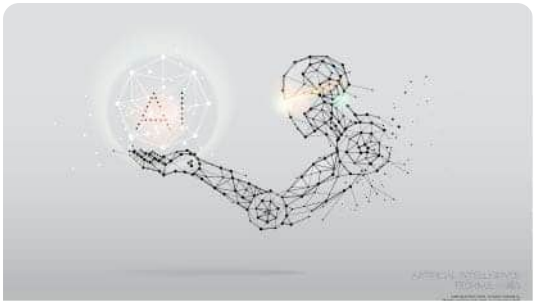
Assim como as contrapartes biológicas (neurônios), o modelo artificial tenta, por meio do ajuste iterativo de pesos das camadas da rede bem como das operações extras entre transições de camadas, representar os dados recebidos, da maneira como nossos neurônios e nossa rede nervosa representam nossas observações, sensações e reações no mundo real.

> Cada camada da rede neural é um aspecto da observação, uma abstração do todo.

Exemplo

Em redes que reconhecem imagens, cada camada pode representar um aspecto da imagem: a primeira camada pode detectar linhas; a segunda, detectar círculos; e a terceira, identificar composições entre linhas e círculos.

Redes neurais artificiais



Como sabemos, na área de inteligência artificial, temos dois grandes grupos: um chamado de IA simbólica e o outro de IA subsimbólica. Basicamente, o que os difere é o processo de aprendizado e, ainda que ambos se enquadrem em aprendizado supervisionado e não supervisionado, o tipo de processo varia.

Nos modelos simbólicos, temos o aprendizado baseado em regras, propriedades, lógica, enquanto nos modelos subsimbólicos, temos o aprendizado direto dos dados ou da experiência. Isso ocorre porque esses tipos de modelos não precisam de etapas de pré-processamento para funcionar, pois isso já está "embutido" em seu funcionamento. As redes neurais pertencem ao grupo de IA subsimbólica.

Um dos primeiros modelos de IA subsimbólica foi o perceptron, criado por **Frank Rosenblatt**, no laboratório aeronáutico de Cornell, em 1958. Esse modelo simulava um neurônio e, assim como a sua contraparte biológica, tinha entradas e saídas por onde os estímulos passavam, e uma função de ativação que levava entradas a um domínio desejado (segundo o conjunto de dados). Individualmente, cada perceptron pode ser encarado como um classificador linear, e o conjunto desses neurônios artificiais é uma rede neural.

Frank Rosenblatt

Psicólogo americano, conhecido no campo da inteligência artificial como criador do perceptron. Por vezes, é chamado de pai do aprendizado profundo, juntamente com outros pesquisadores de renome.

> Existem vários tipos de redes neurais, com arquiteturas e poderes computacionais diferentes.

As **redes neurais convolucionais** (do inglês *Convolutional Neural Network* – CNN), por exemplo, são muito utilizadas em processamento de imagens. Sua arquitetura é composta de **camadas de entrada, de saída e camadas escondidas ou ocultas**, diferenciadas dos demais tipos por contarem com filtros (normalmente matrizes que “deslizam” sobre a entrada de dados).

Neurônio.

As camadas ocultas mapeiam de forma equivariante os pesos e os filtros para as camadas subsequentes e, por isso, são invariantes ao espaço e ao deslocamento. **A maior vantagem desse tipo de rede neural é sua maior autonomia**, pois, para processamentos de imagem, por exemplo, a rede não precisa ter filtros preestabelecidos pelo cientista de dados, uma vez que ela consegue apreender o melhor núcleo de filtragem (matriz de filtragem).

Existem também as **redes neurais recorrentes** (do inglês *Recurrent Neural Networks* – RNN), usadas em problemas sequenciais ou temporais. Tendo a estrutura de um grafo direcionado com ciclos, essa rede é capaz de recorrentemente fazer laço(s) sobre as entradas, permitindo a compreensão de sequenciamento dos dados. A arquitetura mais conhecida de todas dessa categoria é a LSTM (Long Short-Term Memory), inventada por **Hochreiter e Schmidhuber**, em 1997.

Hochreiter e Schmidhuber

Josef Hochreiter é um cientista de computação alemão reconhecido na área de Machine Learning. Com seu professor e renomado pesquisador alemão, Jürgen Schmidhuber, tem contribuído com numerosas publicações na área de aprendizado profundo, com destaque para a rede neural recorrente denominada long short-term memory (LSTM).

De modo superficial, **o que diferencia essas redes de seus concorrentes é o fato de terem portas (gates) de esquecimento**, fazendo com que a rede não exagere nos pesos, nem para mais, nem para menos. Isso ocorre porque, em redes RNN comuns, o gradiente (base do aprendizado desse tipo de modelo) pode acabar sumindo à medida que se propaga para trás ou aumentando indiscriminadamente à medida que se propaga para frente.

Saiba mais

Para evitar esse problema computacional de exageros nos gradientes, esse tipo de rede conta com as suas unidades de memória de curto prazo (short term memory) que servem para se lembrar dos estados anteriores, evitando que o gradiente suma, e podem ser reguladas com pequenos episódios de perda de memória desejada, para não exacerbar os gradientes.

A área de redes neurais artificiais é muito vasta e sempre está caminhando para novas evoluções e variações de modelos, dependendo apenas de quais problemas vamos tentar solucionar com elas.

Demonstrações de redes neurais

Agora, vamos praticar o desenvolvimento de redes neurais. Utilizaremos Python como nossa linguagem de programação e Jupyter Notebook como ambiente de desenvolvimento. Para praticarmos os conhecimentos de redes neurais, especificamente, vamos usar o TensorFlow, que é uma biblioteca de código aberto para aprendizado de máquina e de cálculos em dados descentralizados.

Essa descentralização deve-se ao fato de o Tensorflow poder ser rodado em apenas um único processador ou em múltiplos, seja do tipo CPU normal ou GPU com capacidades computacionais incríveis, uma vez que esse tipo de processador gráfico foi criado e aprimorado ao longo dos anos para lidar com multiplicações de matrizes e subproblemas derivados.

Como reflexo da evolução da tecnologia e do poder computacional, desde 2017, a Google disponibiliza uma versão lite do TensorFlow, para rodar em dispositivos mobile, chamada de TensorFlow Lite.

> Primeiro, vamos instalar o TensorFlow:

```
python

pip install --upgrade tensorflow
```

Dica

Lembre-se de que o TensorFlow pode apresentar restrições por sistema operacional. Então, tome cuidado e preste atenção nos logs do terminal, pois eles podem indicar instruções alternativas de instalação. Em último caso, basta criar um notebook compartilhado no Google Drive e fazer a experimentação pelo Google Colab.

> Feito isso, vamos instalar as bibliotecas de ciência de dados que darão suporte a métricas e gráficos, como o Scikit-Learn, Numpy e o matplotlib:

```
python

pip install sklearn numpy matplotlib
```

> Agora, para finalizar a preparação, vamos instalar o Keras; para isso basta digitar o seguinte código:

```
python

pip install keras
```

O Keras é uma biblioteca para desenvolvimento de redes neurais, projetada para permitir experimentação rápida. Ela oferece um conjunto de abstrações que serve de interface para outros frameworks de aprendizado de máquina, podendo rodar junto ao TensorFlow, Microsoft Cognitive Toolkit, Theano, PlaidML etc.

O interessante do Keras são seus recursos, contando com métodos de construção de camadas, métodos de funções de ativação das mais variadas, funções de perda/custo diferentes, otimizadores de gradiente etc. Por ser uma biblioteca de código aberto, basta procurar sua documentação no Google ou entrar em seu repositório do Github, acompanhar as trilhas de discussão em caso de dúvidas e entrar em contato com os desenvolvedores.

Saiba mais

De modo geral, o mundo da computação é colaborativo. É comum, quando encontramos erros no TensorFlow, Keras, Scikit, copiarmos o log de erro e procurarmos em qualquer motor de busca. Provavelmente, alguma resposta será retornada em algum fórum de discussão, como StackOverflow, 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.

> Assim, estamos preparados e vamos importar as bibliotecas necessárias à demonstração:

```
python

from sklearn.preprocessing import LabelBinarizer
 from sklearn.metrics import classification_report
 from tensorflow.keras.models import Sequential
 from tensorflow.keras.layers import Dense
 from tensorflow.keras.optimizers import SGD
 from tensorflow.keras.datasets import mnist
 from tensorflow.keras import backend as K
 import matplotlib.pyplot as plt
 import numpy as np
```

Na importação de bibliotecas e seus métodos, você deve ter percebido que:

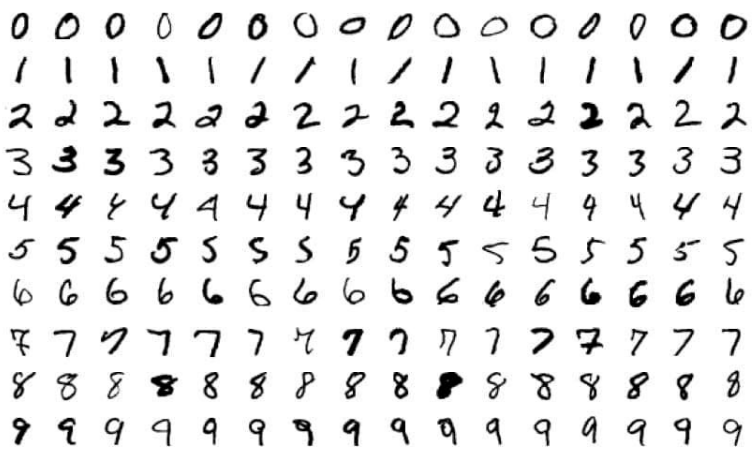
- Trouxemos o Label Binarizer para regularizar as classes do nosso modelo.
- Importamos o Sequential, que é a base da construção de camadas.
- Importamos o Dense, que é o método de construção de camadas totalmente conectadas.
- Importamos o SGD (*Stochastic Gradient Descent*), que é o otimizador de gradiente descendente tradicional.

Além desses métodos essenciais, cabe ressaltar que:

- Importamos o classification report, método que nos mostrará o resultado de nossa demonstração em detalhes (muito recomendado para o uso no dia a dia com problemas de classificação de um modo geral).
- Importamos o MNIST, que é o nosso conjunto de dados do qual falaremos mais adiante.
- Importamos o pyplot para visualizarmos a evolução do aprendizado no nosso modelo, bem como o numpy para regularizar o dimensionamento dos nossos dados.

> Agora, vamos baixar nosso conjunto de dados para experimentação.

Utilizaremos um dos conjuntos de dados mais conhecidos e usados na área de redes neurais: **os conjuntos de dígitos escritos à mão do MNIST** (*Modified National Institute of Standards and Technology*). A imagem a seguir mostra um dos 70.000 datasets de dígitos.



Dataset do MNIST.

> Basicamente, nossa rede neural vai aprender a reconhecer o padrão de escrita de números. Para conseguirmos esse conjunto, vamos utilizar o seguinte bloco de código:

```
python

print('[INFO] accessing MNIST...')
 ((trainX, trainY), (testX, testY)) = mnist.load_data()
```

Agora baixado, o conjunto já vem no formato de treinamento e teste, como podemos ver na declaração do bloco de código anterior, mas vamos precisar rearrumar o conjunto. Cada imagem do MNIST tem dimensões 28×28×1, mas, para a rede neural, vamos precisar chapar a imagem em 28×28=784 pixels.

> Em seguida, normalizaremos os dados para que fiquem entre 0 e 1, e faremos isso dividindo o conjunto por 255 (valor máximo de um pixel). Para isso, utilizaremos o código a seguir:

```
python

trainX = trainX.reshape((trainX.shape[0], 28 * 28 * 1))
 testX = testX.reshape((testX.shape[0], 28 * 28 * 1))
 trainX = trainX.astype('float32') / 255.0
 testX = testX.astype('float32') / 255.0
```

> Agora, para adequar a última camada, a de saída, vamos binarizar a classe da seguinte forma:

```
python

lb = LabelBinarizer()
 trainY = lb.fit_transform(trainY)
 testY = lb.transform(testY)
```

O LabelBinarizer faz com o que o resultado da classe se torne binário, ou seja, ao invés de lidarmos com a classe de valor 5, passaremos a lidar com 0000100000. Isso é importante, pois a camada final deve ter tamanho proporcional às possibilidades de resultados esperados. Vejamos outro exemplo: Se a rede classificasse animais em Gato, Cão ou Peixe, teríamos 001, 010, 100, respectivamente.

Essa prática é muito comum em problemas de classificação multiclasse, mas existe uma categoria que é a de multirótulos, em que os resultados podem ser de mais de uma classe ao mesmo tempo. Por exemplo, categorias de autores de livros (Terror, Ficção Científica, Romance) admitem que um livro seja de Terror e Ficção Científica, tendo a binarização 110. Para cada caso como esse, existe uma função de custo diferente.

> Agora, vamos definir a arquitetura da nossa rede neural.

Com a ajuda do Keras, isso pode ser feito de maneira simples, adicionando uma camada atrás da outra em sequência, conforme vemos a seguir:

```
python

model = Sequential()
 model.add(Dense(256, input_shape=(784,), activation='sigmoid'))
 model.add(Dense(128, activation='sigmoid'))
 model.add(Dense(10, activation='softmax'))
```

Como podemos ver, a arquitetura seguirá este formato:

- Uma camada de entrada de 784 nós, um para cada pixel da imagem em questão, que se conectará a uma camada oculta densa de 256 nós pela função de ativação da sigmoide.
- Depois, a primeira camada oculta se conectará à segunda, de 128 nós, também por sigmoide.
- Esta se conectará à última camada de predição com 10 nós conectados a partir da Softmax. São 10 nós, porque temos 10 possíveis dígitos.

Para treinar o modelo, vamos usar como otimizador do gradiente o SGD, aquele baseado no gradiente descendente, e com taxa de aprendizado 0.01. Também faremos uso da métrica de acurácia para acompanhar o modelo.

Para calcular a perda ou função de custo, vamos usar a entropia cruzada categórica (categorical_crossentropy), que é a mais utilizada em problemas de classificação.

> Agora, digitaremos o seguinte código:

Antes disso, entenda que para as épocas da nossa rede, vamos escolher 100 épocas, ou seja, a rede tem 100 iterações para convergir e apreender, e vamos apresentar lotes de 128 imagens cada por iteração. O código a seguir será para isso.

```
python

sgd = SGD(0.01)
 model.compile(loss='categorical_crossentropy', optimizer=sgd,metrics=['accuracy'])
 H = model.fit(trainX, trainY, validation_data=(testX, testY),epochs=100, batch_size=128)
```

> Neste ponto, basta digitar o seguinte bloco de código:

Isso porque agora chegou o momento de ver como nossa rede se saiu. Para tal, vamos utilizar a classification_report, uma função do sklearn que compara os valores preditos com os reais, passados como argumentos.

```
python

predictions = model.predict(testX, batch_size=128)
 print(classification_report(testY.argmax(axis=1),
     predictions.argmax(axis=1),
     target_names=[str(x) for x in lb.classes_]))
```

O resultado disso será um relatório de classificação. Como esse problema é multiclasse, além de mostrar a acurácia geral da classificação, o relatório apresentará o resultado de cada classe possível, assemelhando-se à imagem a seguir.

	precision	recall	f1-score	support
0	0.94	0.98	0.96	980
1	0.97	0.97	0.97	1135
2	0.93	0.90	0.91	1032
3	0.91	0.91	0.91	1010
4	0.92	0.93	0.93	982
5	0.91	0.87	0.89	892
6	0.93	0.95	0.94	958
7	0.93	0.93	0.93	1028
8	0.89	0.89	0.89	974
9	0.92	0.90	0.91	1009
accuracy			0.92	10000
macro avg	0.92	0.92	0.92	10000
weighted avg	0.92	0.92	0.92	10000

Relatório resultante do código anterior.

Para interpretarmos esse relatório, tenha em mente que cada linha da matriz principal é uma possível classe, cada coluna é uma métrica de acompanhamento (no caso, precisão, recall e medida F1), finalizando com suporte (ou cobertura, ou seja, quantos casos foram cobertos pelas métricas escolhidas).

Nas linhas de baixo, temos:

Acurácia geral

Média de acertos do modelo ao tentar prever o alvo do problema. É classificada como geral, pois contabiliza o número de acertos sem levar em consideração a ponderação entre possíveis classes.

Média macro de acurácia

Comparação a nível macro de acurácia para cada classe, feita sem considerar a distribuição da classe em relação às demais.

Média ponderada de acurácia

Métrica que leva em consideração a distribuição das observações por classe em relação às demais.

As métricas macro, micro e ponderada são importantíssimas para conjuntos desbalanceados, pois, se tomarmos a métrica de modo generalista, isso pode dissuadir nossa avaliação de modelo, porque pode ser que o modelo tenha acertado tudo de algumas classes e pouco de outras, e essas métricas diferenciadas nos ajudam a encontrar esses detalhes.

micro

A microacurácia é uma métrica de acertos para problemas multiclasses, que agrega as contribuições de cada possível classe.

Em nossa análise, podemos perceber que o modelo tem mais facilidade em encontrar o dígito 1 do que outros dígitos e mais dificuldades de identificar os dígitos 8 e 5. De modo geral, a acurácia de 92% não é ruim, significando que, em cerca de 9 a cada 10 tentativas, a rede acerta. Se analisarmos as métricas diferenciadas, podemos ver que, de maneira geral, os acertos entre classes estão bem equilibrados.

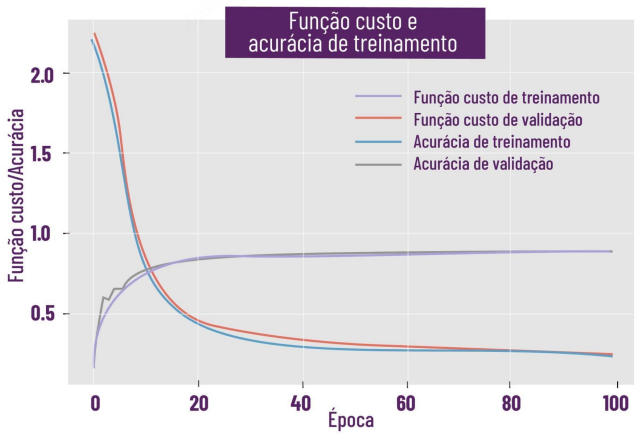
> Para finalizar, vamos utilizar o seguinte bloco de código:

Saiba que com o código, a seguir, podemos ver como a rede evoluiu até chegar a essas métricas, ou seja, como a função de custo foi sendo otimizada e a acurácia foi subindo.

```
python

plt.style.use('ggplot')
 plt.figure()
 plt.plot(np.arange(0, 100), H.history['loss'], label='train_loss')
 plt.plot(np.arange(0, 100), H.history['val_loss'], label='val_loss')
 plt.plot(np.arange(0, 100), H.history['accuracy'], label='train_acc')
 plt.plot(np.arange(0, 100), H.history['val_accuracy'], label='val_acc')
 plt.title('Training Loss and Accuracy')
 plt.xlabel('Epoch #')
 plt.ylabel('Loss/Accuracy')
 plt.legend()
```

Esse código resultará no gráfico a seguir. Como podemos ver, o resultado da função de custo diminui à medida que as épocas passam, e a acurácia aumenta, o que é bastante intuitivo, pois a rede está aprendendo com o passar das épocas (iterações).



Curva resultante do código anterior.

Retomando os conhecimentos teóricos sobre redes neurais, a cada época, a rede tenta um palpite nos seus pesos, de forma a diminuir a **função custo** e a distância entre a realidade e o predito.

Função custo

Na maioria das bibliotecas e na literatura da área, é conhecida como função loss, que mede a distância entre a realidade e a predição.

Assim, reduz-se a função loss e aumenta-se um pouco a acurácia. Depois, na próxima rodada, o processo se repete, o otimizador do gradiente descendente gera um novo palpite, de modo a reduzir a função de custo até que ele seja mínimo ou que o número de épocas acabe. Por fim, teremos nosso modelo treinado e pronto para uso.

Dica

No dia a dia, para checar se o modelo está overfitting ou underfitting, recomenda-se apresentar um novo conjunto de dados ao modelo treinado e deixar que ele itere sobre o conjunto. No fim, pelo relatório de classificação, mede-se o quanto o modelo acertou.

Se o modelo tiver errado muito, saberemos que ou ele se superajustou ou se desajustou. Para tirar a dúvida de qual dos dois ocorreu, é necessário fazer uma rodada com os dados antigos: se ele acertar todos e não errar nenhum ou quase nenhum, é um forte indício de overfitting; se ele gerar valores aleatórios e errar bastante (muito mais da metade), então, está desajustado.

De modo geral, redes neurais sofrem mais de superajuste do que de desajuste, uma vez que a condição de parada principal delas é a minimização da função custo. Para que você possa ter os mesmos resultados em casa, os códigos podem ser copiados e executados em um ambiente como o Jupyter Notebook ou o Google Colab.

A biblioteca TensorFlow no Python

Veja agora uma apresentação genérica da biblioteca TensorFlow, destacando as aplicações de aprendizado de máquina para as quais ela é empregada. Vamos lá!

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Vem que eu te explico!

Os vídeos a seguir abordam os assuntos mais relevantes do conteúdo que você acabou de estudar.

Redes Neurais Artificiais (aspectos conceituais)

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Demonstrações de Redes Neurais (explicação da função de custo x acurácia)

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Verificando o aprendizado

Questão 1

A biblioteca Keras serve para

A

abstrair a construção de camadas de redes neurais.

B

construir o relatório de métricas.

C

embaralhar os dados.

D

organizar os dados.

E

pré-processar os dados.

A alternativa A está correta.

A biblioteca Keras foi criada com o propósito de implementar as funções matemáticas e de álgebra linear que as redes neurais precisam para trabalhar (principalmente multiplicação de matrizes), de forma que o desenvolvedor possa focar apenas a estruturação da arquitetura da rede. A alternativa B é tarefa do módulo Classification_Report. As demais alternativas tratam de atividades de preparação dos dados.

Questão 2

O que as camadas escondidas representam em um modelo de aprendizado profundo?

A

Operações

B

Relações

C

Números

D

Abstrações

E

Categorias

A alternativa D está correta.

Os modelos de aprendizado profundo tentam incorporar o pré-processamento de dados, para que, justamente no cerne dessa estrutura (as camadas ocultas), o modelo seja capaz de capturar diferentes aspectos dos dados complexos ou abstrações.

Considerações finais

A IA é a área que tenta simular os comportamentos humanos de raciocínio automático e representação do conhecimento, para que os algoritmos possam se adaptar a problemas novos, nunca vistos pelo agente inteligente, mas que sejam similares aos que ele já tratou.

Os avanços dos sistemas inteligentes atuais, e o fato de a própria IA ter saído do seu inverno de desuso, devem-se ao grande volume de dados que emergiu de forma veloz e variada após o surgimento de smartphones e dispositivos periféricos conectados, aumentando ainda mais a colaboração na rede e a produção desenfreada de dados.

Existem várias abordagens de como implementar em máquina os processos cognitivos humanos, cada uma resolvendo um tipo de problema específico, assim como nosso cérebro, que cria estratégias específicas para aprender um novo conceito ou agrupar um conjunto de novas observações por similaridades.

Vimos que as redes neurais são poderosos modelos de aprendizado de máquina, capazes de guardar o conhecimento por elas descoberto em suas camadas ocultas internas e criar diferentes níveis de abstração para dados complexos, eliminando até a necessidade de uma preparação de dados prévia.

Para obter sucesso em extrair padrões de grandes conjuntos de dados, é necessário seguir um processo bem definido do começo ao fim, como é o caso do processo de KDD, que contém seis etapas para coletar, tratar e transformar dados, de forma que um modelo de aprendizado de máquina seja capaz de analisar e extrair automaticamente os padrões existentes.

Podcast

Para encerrar, apresentaremos conceitos e processos em Big Data Analytics, respondendo a algumas perguntas. Vamos ouvir!

Conteúdo interativo

Acesse a versão digital para ouvir o áudio.

Explore +

Uma excelente introdução (em inglês) sobre Data Mining e Machine Learning pode ser lida no livro on-line **Data Mining and Machine Learning: fundamental concepts and algorithms**, publicado pelos professores Mohammed Zaki e Wagner Meira Jr.

Saiba mais sobre redes neurais artificiais no post do professor André Ponce de Leon de Carvalho, no site do ICMC/USP.

Conheça outras funcionalidades da biblioteca Scikit-Learn no site do próprio projeto.

Referências

AMARAL, F. **Aprenda mineração de dados**: teoria e prática. Rio de Janeiro: Alta Books, 2016. v. 1.

AZEVEDO, A. I. R. L; SANTOS, M. F. **KDD, SEMMA and CRISP-DM**: a parallel overview. IADS-DM, 2008.

BABBAR, P. *et al.* **Connectionist model in Artificial Intelligence**. International Journal of Applied Engineering Research, v. 13, n. 7, p. 5154-5159, 2018.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. **The KDD process for extracting useful knowledge from volumes of data**. Communications of the ACM, v. 39, n. 11, p. 27-34, 1996.

KABIR, S. M. S. **Methods of data collection**. *In:* KABIR, S. M. S. Basic guidelines for research – an introductory approach for all disciplines. 1. ed. Chatigão: Book Zone, 2016, p. 201-275. (cap. 9).

RUSSEL, S.; NORVIG, P. **Inteligência artificial**. 3. ed. São Paulo: GEN LTC, 2013.

SILVA, F. C. D.; GARCIA, A. C. B. **Judice Verum**: a methodology for automatically classify fake, sarcastic and true portuguese news. Publicado em: dez. 2019.

WIRTH, R.; HIPP, J. **CRISP-DM**: towards a standard process model for data mining. *In*: INTERNATIONAL CONFERENCE ON THE PRACTICAL APPLICATIONS OF KNOWLEDGE DISCOVERY AND DATA MINING, 4., 2000, London. Proceedings [...]. London: Springer-Verlag, 2000.

WNEK, J.; MICHALSKI, R. S. **Comparing symbolic and subsymbolic learning:** three studies. Fairfax: George Mason University, 1992.