




MACHINE LEARNING

Lucas Claudino

Machine learning

1ª edição

Londrina
Editora e Distribuidora Educacional S.A.
2019



Autoria: Prof. Lucas dos Santos Araujo Claudino.
Revisão: Prof. Nathália dos Santos Silva Nolepa.
Como citar este documento: CLAUDINO, Lucas dos Santos Araujo. **Machine Learning**. Londrina-PR: Editora e Distribuidora Educacional S.A., 2019.

© 2019 por Editora e Distribuidora Educacional S.A.

Todos os direitos reservados. Nenhuma parte desta publicação poderá ser reproduzida ou transmitida de qualquer modo ou por qualquer outro meio, eletrônico ou mecânico, incluindo fotocópia, gravação ou qualquer outro tipo de sistema de armazenamento e transmissão de informação, sem prévia autorização, por escrito, da Editora e Distribuidora Educacional S.A.

Editora e Distribuidora Educacional S.A.
Avenida Paris, 675 – Parque Residencial João Piza
CEP: 86041-100 — Londrina — PR
e-mail: editora.educacional@kroton.com.br
Homepage: <http://www.kroton.com.br/>



SUMÁRIO

Apresentação da disciplina	5
Introdução ao <i>machine learning</i> : como funciona a aprendizagem de máquina	6
Algoritmos de <i>machine learning</i> : modelos preditivos	25
Regressão, <i>k-nearest neighbors</i> (kNN) e Naive Bayes	43
<i>Decision tree</i> , <i>Random Forest</i> e método <i>ensemble</i>	63
<i>Clustering</i> , <i>support vector machines</i> e processamento em linguagem natural	82
Redes neurais artificiais e introdução ao <i>deep learning</i>	103
Sistemas de recomendação	122



► Apresentação da disciplina

A disciplina *Machine Learning* apresenta conceitos essenciais para o estudo das técnicas computacionais capazes de aprimorar seu desempenho por meio da utilização de algoritmos que aprendem com sua própria experiência. O estudo do aprendizado de máquina trará diversas técnicas, como K vizinhos próximos, redes neurais, florestas aleatórias, regressão e máquinas de vetor de suporte.

Além disso, diversas aplicações atuais dos algoritmos de *Machine Learning* serão apresentadas, como reconhecimento de padrões, classificação multinível, detecção de fraudes e diversos outros casos que são resolvidos por meio da implementação de técnicas de inteligência computacional.

Cada uma das técnicas apresentadas será estudada em detalhes, partindo da explicação teórica dos princípios de funcionamento e chegando à apresentação de guias para implementação desses algoritmos.

O objetivo principal desta disciplina é apresentar os principais modelos de aprendizado de máquina, incluindo conceitos, teorias, aplicações sobre os seguintes algoritmos: modelagem preditiva, regressão, *K-nearest neighbors*, *naive bayes*, *decision tree*, *random forest*, método *ensemble*, *clustering*, *support vector machine*, processamento em linguagem natural, redes neurais artificiais, *deep learning* e sistemas de recomendação.



Introdução ao *machine learning*: como funciona a aprendizagem de máquina

Autor: Lucas Claudino

► Objetivos

- Aprender como funciona a dinâmica do aprendizado de máquina: se o algoritmo escolhido necessita ou não de uma etapa de treinamento.
- Classificar corretamente o aprendizado de máquina para então saber qual algoritmo é mais adequado para cada situação a ser resolvida.
- Estudar as principais topologias de problemas que utilizam a aprendizagem de máquina como ferramenta de resolução.

1. Introdução

O termo *machine learning*, em português, significa aprendizado de máquina. O principal objetivo desse aprendizado é buscar a maneira de entender a estrutura dos dados de interesse para poder ajustá-los a modelos já conhecidos, para então facilitar a interpretação desses dados, fornecendo meios para analisar, inferir e prever informações a partir desses modelos. Mesmo sendo um campo da ciência da computação, o aprendizado de máquina possui características bastante diferentes da computação tradicional.

Machine learning, como o próprio nome já diz, é capaz de aprender com os dados fornecidos. Os algoritmos aplicados nessa área proporcionam aos computadores técnicas para que eles sejam capazes de treinar sua inteligência por meio de dados de entrada, para então criar modelos que conseguem tomar decisões inteligentes baseadas nos resultados obtidos por meio do treinamento.

Neste estudo, serão abordados os principais fatores do aprendizado de máquina para mostrar seu funcionamento básico, quais vantagens ele pode trazer e suas principais aplicações.

PARA SABER MAIS

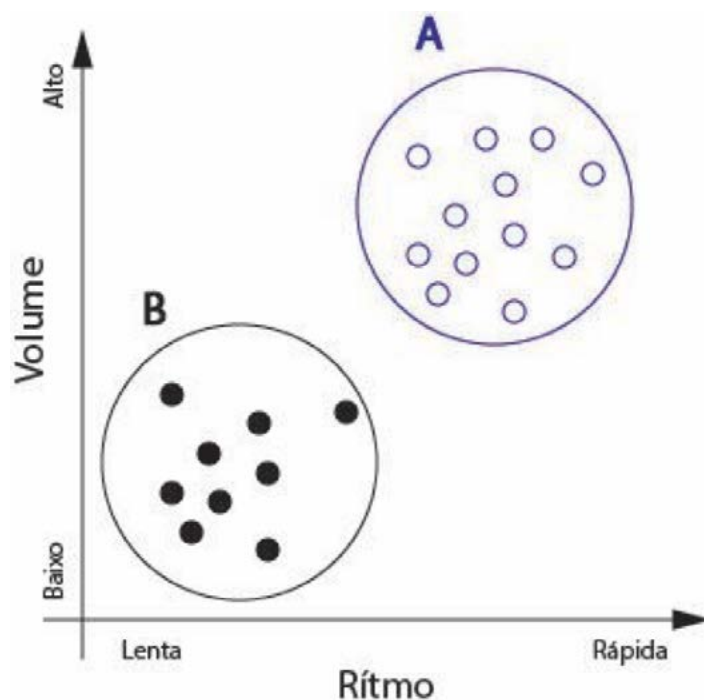
Antes mesmo de se ter o conceito de *machine learning* formulado, Alan Turing, em 1950, publicou um artigo intitulado "*Computing Machinery and Intelligence*". Nesse trabalho, Turing questionava, dentro de todas as limitações da época, se as máquinas podiam realmente pensar. Alan Turing propôs uma espécie de pensamento baseado na cópia de atividades humanas (TURING, 1950).

► 2. Entendendo a aplicação de *machine learning*

Segundo Sá (1995, p. 27): “[...] por meio da mesma ‘arte de conversação’ que abrange tão extensa e significativa parte da nossa existência cotidiana [...]”, o aprendizado de máquina (AM) é dito como sendo a capacidade que a máquina possui de melhorar seu desempenho na realização de algum processo por meio da experiência obtida com treinamentos (MITCHELL, 1997).

Para entender melhor como pode ser esse aprendizado e como ele pode ajudar na tomada de decisões, considere o caso de João, que é uma pessoa que gosta de músicas rápidas e com volume alto. Colocando essas informações em um gráfico, é possível identificar duas regiões: A é a região de músicas que João gosta e B é a região de músicas que ele não gosta, assim como mostra a Figura 1. Qualquer música nova reproduzida por João, se estiver dentro das regiões A ou B, poderá ser facilmente classificada de acordo com o gosto musical.

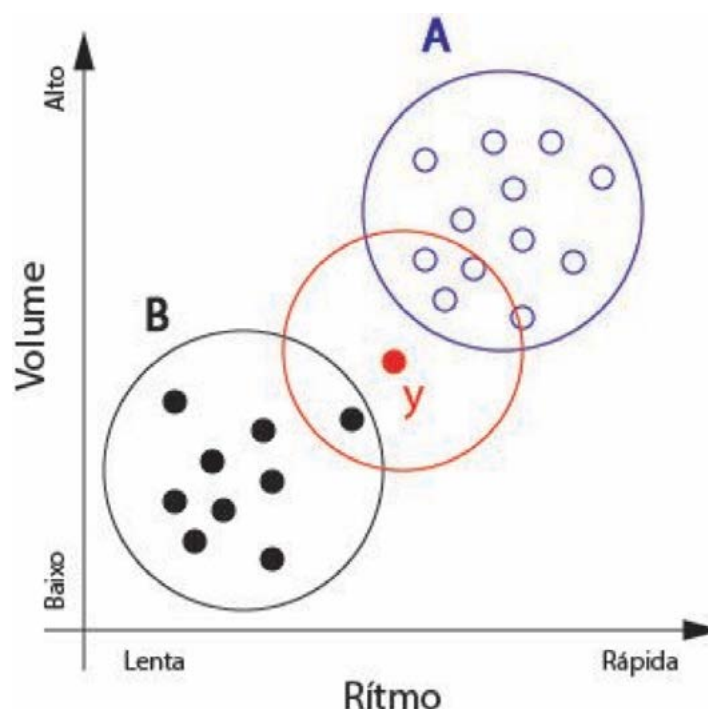
Figura 1 – Exemplo de músicas classificadas graficamente



Fonte: elaborada pelo autor.

O problema aparece quando João escutar uma música nova e ela não puder ser diretamente classificada em nenhuma das regiões. Um exemplo disso é o ponto **y** no gráfico da Figura 2. Para fazer a classificação correta da música **y**, é preciso traçar uma região ao redor do ponto, identificando que há mais músicas pertencentes a A do que a B perto do ponto **y**, assim como ilustra a Figura 2. Logo, pode-se afirmar que **y** é pertencente ao grupo A.

Figura 2 – Exemplo de músicas classificadas graficamente e uma música nova inserida no gráfico



Fonte: elaborada pelo autor.

Você observou que, para fazer essa classificação correta, primeiro houve um treinamento relacionando os grupos A e B. É dessa forma que os algoritmos de aprendizado de máquina funcionam. Mais especificamente, essa metodologia de classificação apresentada é como funciona o algoritmo KNN (*K nearest neighbors*), que identifica os K vizinhos mais próximos.

O exemplo das músicas de João fornece mais uma característica do funcionamento do aprendizado de máquina. O modelo a ser criado a partir do estudo da disposição gráfica das músicas tem o objetivo de

prever se uma nova música a ser escutada por João estará ou não de acordo com seu gosto musical. Sendo assim, você já sabe então a primeira classificação do AM quanto ao seu objetivo: ele pode ser **preditivo** ou **descritivo**. Sendo assim, estas e outras definições serão abordadas ao longo do texto.

2.1 Classificação do aprendizado de máquina

A partir do exemplo, você viu que o aprendizado de máquina pode ser classificado como preditivo ou descritivo. As tarefas de previsão usam os dados fornecidos durante a etapa de treinamento para criar funções capazes de prever um rótulo que caracterize o novo dado inserido (assim como o ato de prever a música no exemplo do João). Para que isso aconteça corretamente, cada dado fornecido durante o treinamento deve possuir rótulos de entrada e saída (FACELI; LORENA; GAMA, 2011). No exemplo, os rótulos de entrada eram o ritmo e o volume, e a saída era se João gostava ou não da música.

As tarefas de descrição, como o próprio nome já diz, tem como objetivo analisar um conjunto de dados e descrevê-lo de alguma maneira. Essa descrição pode ser, por exemplo, encontrar grupos de dados semelhantes, e então descrever os dados a partir desses grupos. Ou então, a tarefa descritiva pode ser também “encontrar regras de associação que relacionam um grupo de atributos a outro grupo de atributos” (FACELI; LORENA; GAMA, 2011, p. 6).

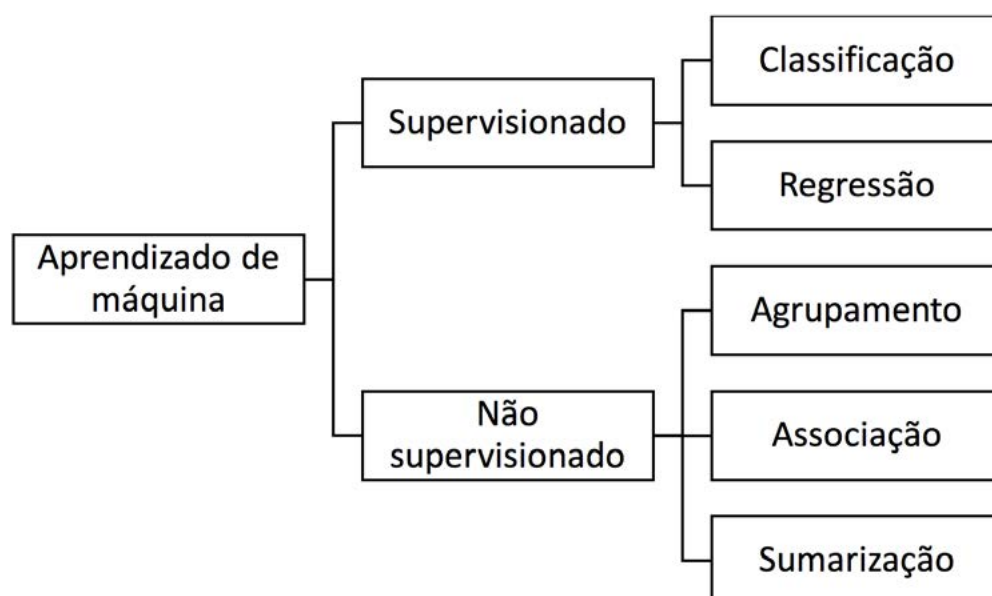
Além da classificação em preditivo e descritivo, pode-se também classificar o aprendizado de máquina como **supervisionado** e **não supervisionado**. O aprendizado supervisionado é aquele em que os dados utilizados para treinar o algoritmo contêm a resposta desejada. Ou seja, você fornece os dados já sabendo qual é o rótulo certo, então consegue treinar seu algoritmo para fornecer a resposta desejada. Isso é o que acontece no exemplo do João, em que você fornece todos os dados sobre as músicas da etapa de treinamento e sabe se ela

está de acordo ou não com o gosto musical dele. Dessa maneira, você supervisiona seu algoritmo durante o treinamento para que ele possa criar a função que será responsável por gerar a resposta desejada para qualquer dado novo que seja inserido.

Em diversos casos, não é possível obter dados já rotulados para fornecer à máquina, de maneira que ela possa criar funções relacionando entradas e saídas desejadas. Um exemplo clássico é um algoritmo criado para mostrar os perfis de compradores de uma determinada loja. Neste caso, o computador não tem uma classificação específica a fazer dos indivíduos, ele apenas vai separar os compradores de acordo com suas semelhanças. Uma opção é observar o histórico de compras e então verificar se existem padrões ou repetições que permitam a inferência do perfil dos clientes. Sendo assim, é possível identificar que o computador não será supervisionado, pois não há uma resposta esperada. O que acontece nos casos de aprendizado não supervisionado é que os resultados obtidos serão úteis para guiar a análise dos dados.

Para melhor entender a classificação do AM, observe a Figura 3.

Figura 3 – Classificação do aprendizado de máquina



Fonte: elaborada pelo autor.

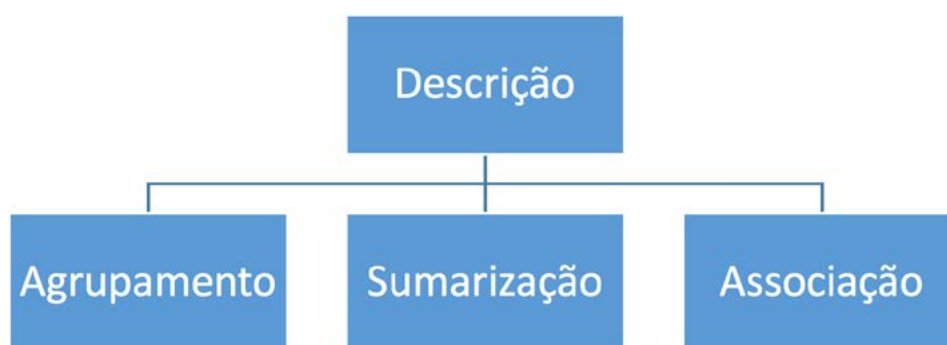
A partir da Figura 3, é possível observar que o AM supervisionado pode ser também subdividido. O AM de classificação irá classificar todo dado novo recebido, sendo que os critérios de classificação são fornecidos durante a etapa de treinamento. O trabalho da máquina é, a partir do treinamento, elaborar a maneira mais eficaz de fazer a classificação correta. Já o AM de regressão vai utilizar os dados do treinamento para gerar uma função responsável por ditar como uma variável se comporta, ao passo que outra variável é modificada.

O AM não supervisionado, assim como visto na Figura 3, pode ser ainda subdividido em agrupamento, associação e sumarização, sendo que essas três classificações significam:

As tarefas descritivas são genericamente divididas em: agrupamento, em que os dados são agrupados de acordo com sua similaridade; sumarização, cujo objetivo é encontrar uma descrição simples e compacta para um conjunto de dados; e associação, que consiste em encontrar padrões frequentes de associações entre os atributos de um conjunto de dados. (FACELI; LORENA; GAMA, 2011, p. 6)

Para melhor entender essa divisão, observe a Figura 4.

Figura 4 – Síntese da divisão do AM de descrição



Fonte: elaborada pelo autor.

Uma terceira abordagem para o aprendizado de máquina também pode ser estudada: a aprendizagem por **reforço**. Essa metodologia se baseia no princípio de que nenhum problema a ser resolvido é estático, pois

ele pode mudar ao longo do caminho. Nessa técnica, a máquina escolhe, dentro de um leque de possibilidades a ela fornecido, um caminho a tomar para a solução do problema proposto, e ao final observa qual foi o resultado obtido. Cada vez que o resultado não é satisfatório, a máquina busca modificar a estratégia de solução do problema.

Essa metodologia é baseada em estudos da psicologia, que utiliza uma estratégia de dar alguma forma de recompensa ou punição a alguém como consequência de seus atos. Um exemplo clássico disso é o adestramento de cães, em que o adestrador pede para o animal se sentar e, caso ele realmente sente, oferece uma recompensa (geralmente um biscoito).

Observe que essa metodologia não é tão simples quanto parece. Um grande desafio é fazer com que o algoritmo não pare em uma “zona de conforto”, onde ele sempre receberá uma recompensa, pois nesses casos não haverá uma exploração de estratégias mais eficientes para a resolução do problema. Isso é conhecido como o compromisso entre *exploration* e *exploitation*. *Exploration* refere-se à exploração, que é o estado de buscar estratégias mais eficientes; já *exploitation* refere-se ao estado de percorrer o caminho no qual se tem certeza do sucesso.

Essa estratégia é muito utilizada em algoritmos aplicados a instituições financeiras e/ou investidores, pois é um cenário no qual se pode optar por um investimento mais arriscado e com chances de gerar um lucro muito maior do que investimentos seguros com rentabilidade menor.

ASSIMILE

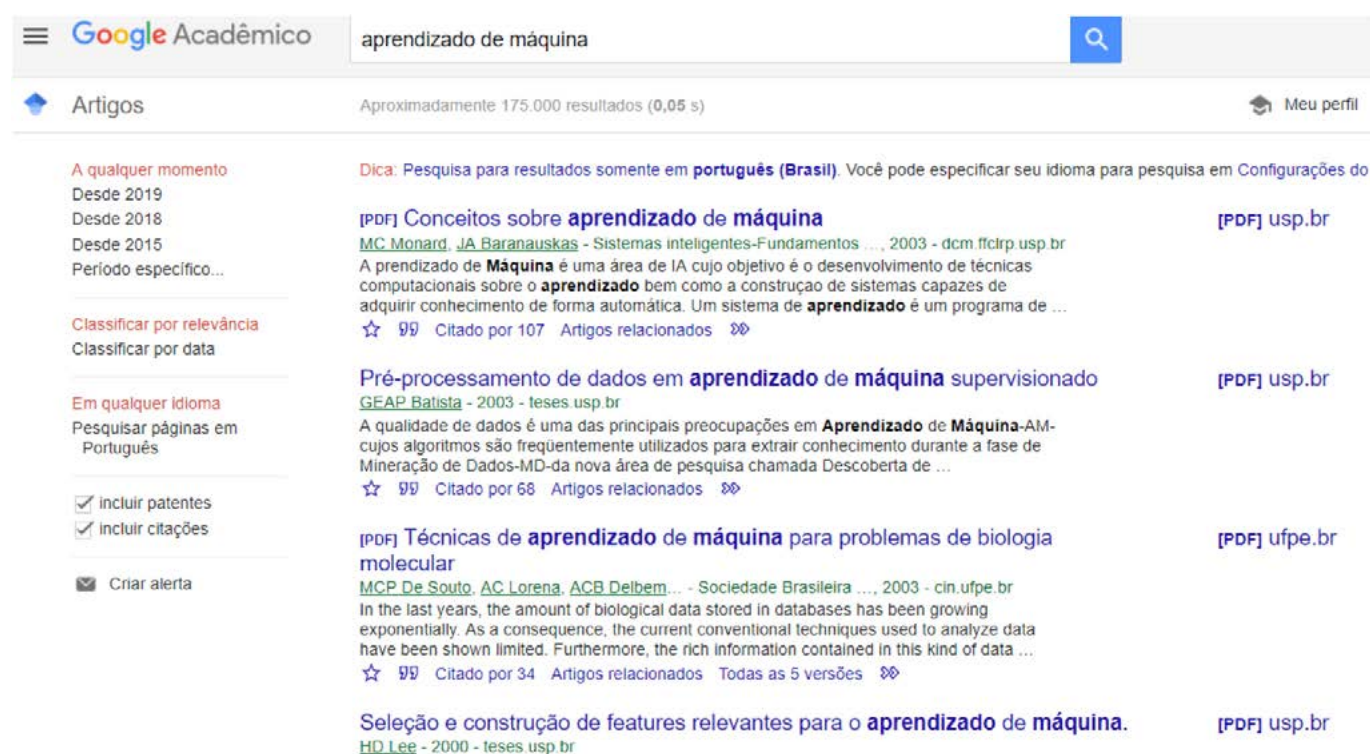


O aprendizado de máquina é muito amplo e com inúmeras aplicações, mas pode ser basicamente subdividido em três grandes categorias: aprendizado **supervisionado**, **não supervisionado** e de **reforço**.

2.2 Aplicações da aprendizagem de máquina


O aprendizado de máquina está intimamente ligado com diversas atividades de seu cotidiano, desde atividade simples, como uma pesquisa na internet, até tarefas complexas, como o reconhecimento da voz. Você, com certeza, já utilizou sites de pesquisa, como, por exemplo, o Google.com, para buscar algum assunto de seu interesse. Ao fazer esse tipo de pesquisa, você, com certeza, irá se deparar com um conceito chamado **ranqueamento de páginas**. Para fazer esse ranqueamento, a máquina recebe um dado de entrada, a palavra a ser pesquisada, e encontra as páginas mais relevantes ao assunto. A Figura 5 ilustra o ranqueamento que acontece ao digitar “aprendizado de máquina”.

Figura 5 – Ranqueamento para “aprendizado de máquina”



Fonte: https://scholar.google.com.br/scholar?hl=pt-BR&as_sdt=0%2C5&q=aprendizado+de+m%C3%A1quina&btnG=. Acesso em: 22 jul. 2019.

Para fazer o ranqueamento ilustrado na Figura 5, a máquina precisa, baseada no texto digitado, varrer o banco de dados, fazer uma seleção dos websites disponíveis e, a partir da popularidade de cada um deles, classificar por ordem de relevância.



Outro cenário de aplicação é a filtragem colaborativa. Muitas lojas virtuais, como Amazon e Netflix, usam essa técnica para influenciar seus clientes a comprarem produtos adicionais, além daquele principal que era alvo de interesse. Você, com certeza, já pesquisou algum produto na internet e, ao final da página de compra, viu uma janela indicando algo como “Quem comprou este item também comprou...”, e depois disso havia uma lista com produtos que você também gostaria de comprar.

Ao tratar de filtragem colaborativa, o problema a ser resolvido pela máquina é similar ao caso do ranqueamento, pois o objetivo é conseguir uma lista de produtos. A diferença principal é que, no caso da filtragem colaborativa, não há uma entrada de dados específica, como a linha de pesquisa do exemplo anterior. Nesse caso, a máquina deve usar estatísticas de outros compradores e, baseado em compras similares, fazer uma lista de sugestão de possíveis produtos adicionais a serem comprados.

Existe também outra aplicação muito importante para o aprendizado de máquina, o reconhecimento facial aplicado à solução de crimes. O número de câmeras de vigilância tem aumentado muito e isso é um grande auxílio para a identificação de criminosos. Mas do que adianta a aquisição de muitas imagens sem a tecnologia certa para fazer o reconhecimento de padrões?

Algoritmos de *deep learning* e redes neurais são muito recomendados para o trabalho com reconhecimento facial. As redes neurais são feitas para simular o funcionamento do cérebro humano e, por isso, indicadas para a identificação de padrões em imagens. Esses algoritmos conseguem digitalizar a imagem e retirar partes que não ajudam no reconhecimento. Os elementos mais importantes para identificar a identidade de uma pessoa são os olhos, nariz, boca e cicatrizes. Por meio da coleta de milhões de dados, podem-se identificar padrões repetidos em imagens diferentes, auxiliando na busca pela identidade. Um exemplo muito interessante de aplicação é o processo utilizado por diversos aplicativos de celular que deixam as pessoas com aparência mais velha; alguns desses aplicativos, como o FaceApp, utilizam redes neurais convolucionais para realizar essa tarefa.

Mas como isso pode ajudar na solução de crimes? A coleta de dados de crimes já resolvidos serve para treinamento dos algoritmos, facilitando a identificação de pessoas quando um novo fato ocorre. Devido à infinidade de câmeras coletando imagens de pessoas conhecidas e desconhecidas, as redes neurais podem ser muito bem treinadas, possuindo um enorme banco de dados para fazer o reconhecimento dos padrões (HURWITZ, 2018, p. 25).

► 3. Problemas clássicos no aprendizado de máquina

Claramente existem inúmeros problemas nos quais você pode aplicar o aprendizado de máquina, porém a grande maioria deles pode ser classificada de acordo com algumas topologias clássicas: classificação binária, classificação multiclasse, regressão, *clustering*, recomendação.

3.1 Classificação binária

Esse tipo de classificação é um dos problemas mais estudados em AM, e foi base teórica para o desenvolvimento de inúmeros algoritmos muito utilizados. Basicamente, pode-se reduzir o algoritmo a responder à seguinte questão: dado um padrão x contido em um domínio X , estime qual valor uma variável aleatória binária $y \in \{\pm 1\}$ irá assumir. Como exemplo, imagine um algoritmo que deseja falar se um novo e-mail chegou em sua caixa é *spam* ou não.

Mesmo sendo um problema aparentemente simples, observe como é possível criar situações mais complicadas:

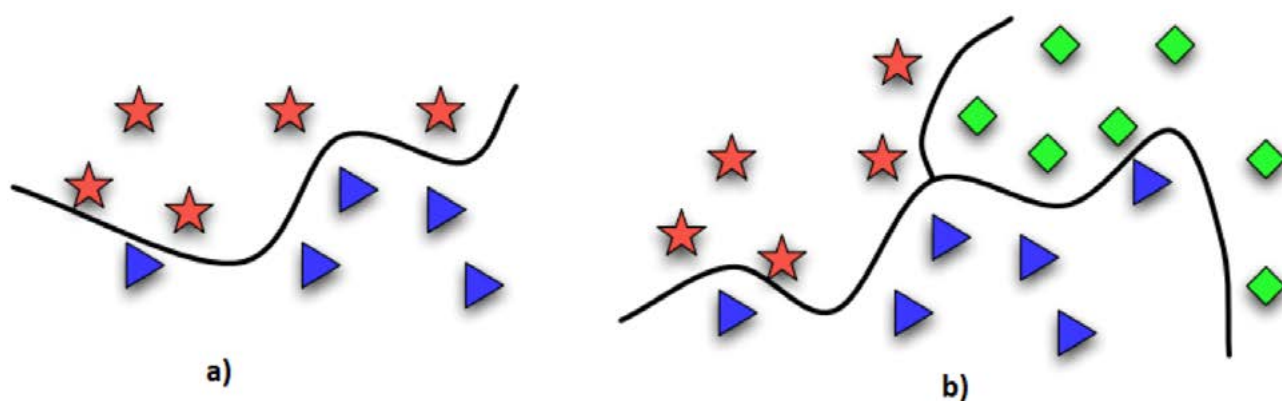
- Pode haver uma sequência de pares (x_i, y_i) a serem instantaneamente estimados. Isso é conhecido como estimativa on-line.
- Você pode observar uma coleção $X := \{x_1, \dots, x_m\}$ e $Y := \{y_1, \dots, y_m\}$ de pares (x_i, y_i) que serão utilizados para estimar y para um conjunto não conhecido $X' := \{x'_1, \dots, x'_m\}$. Isso pode ser conhecido como aprendizado em lote.

- Você pode ser permitido a escolher o conjunto X para construir o modelo. Isso é conhecido como aprendizado ativo.
- Pode haver casos em que não há informação completa sobre X , levando a um problema de estimativa com variáveis desconhecidas.
- Os conjuntos X e X' podem vir de fontes diferentes, levando a um problema de correção de deslocamento por covariância.

3.2 Classificação multiclasse

Pode ser entendida como a extensão da classificação binária. A diferença é que aqui o valor da variável y não assume mais valores binários. Agora se deve considerar que $y \in \{1, \dots, n\}$ assume n valores diferentes. Imagine que o algoritmo queira classificar um documento de acordo com a língua na qual ele foi escrito. Portanto, y pode assumir $y \in \{\text{inglês}, \text{português}, \text{espanhol}, \text{chinês} \dots\}$. Para entender melhor a diferença entre classificação binária e multiclasse, observe a Figura 6.

Figura 6 – Classificações a) binária e b) multiclasse



Fonte: Smola e Nishwanathan (2008, p. 10).

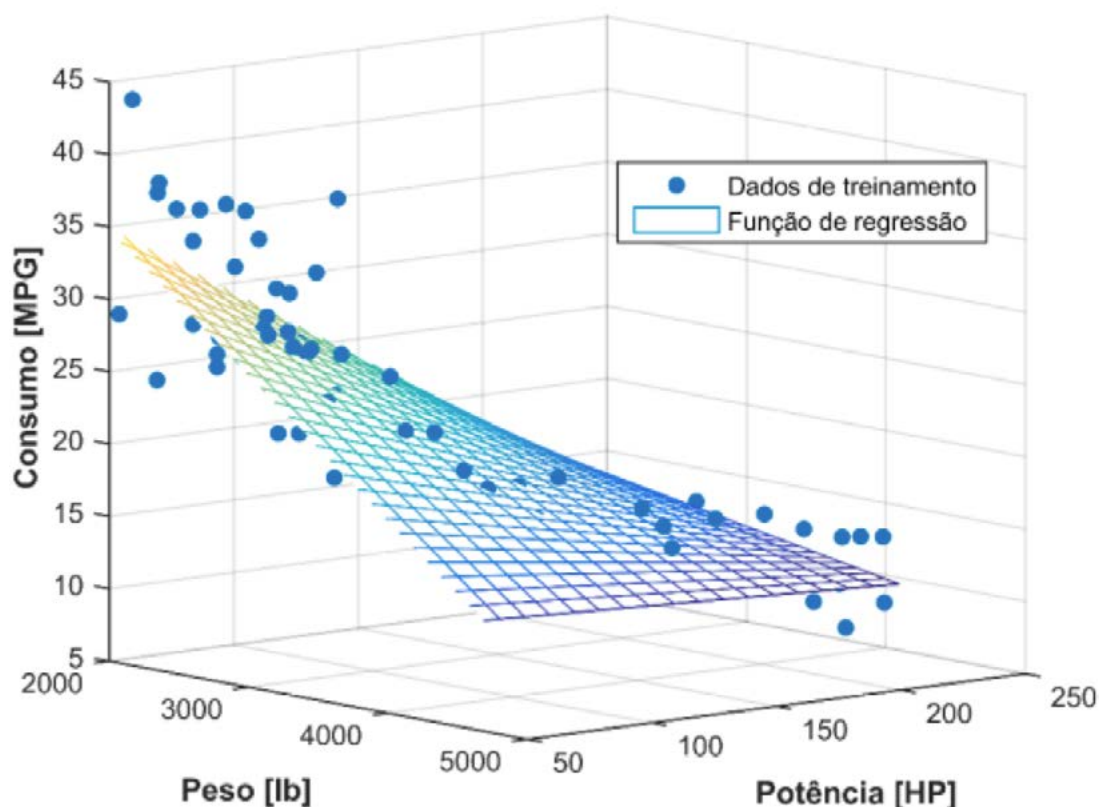
A Figura 6 ilustra o objetivo das classificações binária e multiclasse. No item a, você observa que o algoritmo busca identificar as regiões de separação entre duas classes; já no item b, o propósito é classificar o grupo de dado de acordo com o conjunto de três itens distintos apresentados.

3.3 Regressão

A regressão é utilizada para criar uma metodologia capaz de prever algum tipo de rótulo do dado inserido. Algoritmos de regressão utilizam a etapa de treinamento para criar modelos que relacionam as características dos dados, tendo como resultado uma função matemática. A partir dessa função, você pode, então, inserir qualquer informação dos novos dados e prever o rótulo desejado.

Para entender melhor a regressão, observe a Figura 7, que mostra a relação entre peso, potência e consumo de automóveis. Os pontos representam os dados utilizados para treinamento e, a partir da análise dessas informações, o algoritmo criou uma função tridimensional capaz de estimar qual será o consumo de veículos automotivos.

Figura 7 – Regressão a partir de dados de entrada



Fonte: elaborada pelo autor.

3.4 Clustering

O *clustering*, ou clusterização, é uma tarefa do aprendizado de máquina não supervisionado e seu objetivo é agrupar os dados fornecidos em clusters que possuam características iguais ou semelhantes. Todos os objetos que estejam dentro de um cluster resultante possuem mais similaridade entre si do que com qualquer objeto que esteja fora desse cluster.

3.5 Recomendação

O algoritmo de recomendação é amplamente utilizado por lojas virtuais. O propósito dessa técnica é produzir uma lista de objetos recomendados a partir de algum tipo de entrada de dados, como, por exemplo, as palavras digitadas na busca por produtos a serem comprados.

Os algoritmos de recomendação buscam cruzar informações do perfil usuário com a lista de produtos disponíveis para compra, e assim conseguem criar uma lista de recomendações. Essa técnica não é utilizada somente em lojas. Você pode fazer proveito de uma lista de recomendações para estimular um usuário de revistas digitais a ler novas matérias. O algoritmo de recomendação só precisa elaborar a lista com temas que sejam de interesse do usuário. Um exemplo é o trabalho feito por aplicativos musicais, que criam *playlists* com músicas que provavelmente serão de interesse do usuário.

Nesta Leitura Fundamental, foram primeiramente abordadas as principais características do aprendizado de máquina, em que você foi apresentado aos termos e suas aplicações atuais. Com o exemplo de João e seu gosto musical, você observou como a máquina trabalha e como ela pode “aprender”. Além disso, foram apresentadas algumas das principais aplicações do AM, e foi nessa etapa que você foi também apresentado a algumas aplicações do AM e sua importância para a evolução de todos os sistemas tecnológicos. Finalmente, você estudou

as topologias clássicas de problemas resolvidos pelo aprendizado de máquina: classificação binária, classificação multiclasse, regressão, *clustering* e recomendação. Logicamente, o AM não se restringe a essas topologias. Existem inúmeros problemas nos quais você pode aplicar seus estudos, o seu trabalho é estudar todos os algoritmos e técnicas e utilizar o conhecimento adquirido para encontrar novas situações a serem resolvidas.

TEORIA EM PRÁTICA



Considere que você é um profissional que trabalha no setor de tecnologia de uma locadora online de filmes, a Filmeflix. A empresa implementou um sistema no qual os usuários podem dar notas de 1 a 10 para os filmes assistidos. Porém, as pessoas que estão buscando filmes para assistir acharam que notas com 10 níveis não são muito boas; elas preferem somente saber se o filme é “bom”, “médio” ou “ruim”. Como você é o responsável por fazer essa nova etapa do projeto, tem a liberdade para utilizar a tecnologia que julgar mais adequada.

Em seus estudos para essa tarefa, você viu que será possível criar um algoritmo de aprendizado de máquina capaz de separar os filmes com notas de 1 a 10. O seu algoritmo deverá realizar o seguinte agrupamento:

- Notas de 1 a 3: filme ruim.
- Notas de 4 a 7: filme médio.
- Notas de 8 a 10: filme bom.

O conteúdo estudado até agora já lhe proporcionou bastante conhecimento sobre as principais características e aplicações da aprendizagem de máquina. A partir dos tópicos estudados, descreva em qual problema clássico do AM a situação da Filmeflix se encaixa. Após identificar

o problema, descreva as principais características dessa modalidade de AM, pesquise quais os principais algoritmos já existentes são normalmente utilizados nesse caso e, finalmente, escolha um deles para ser utilizado pela empresa. É importante você descrever o algoritmo escolhido e citar suas principais vantagens.

VERIFICAÇÃO DE LEITURA



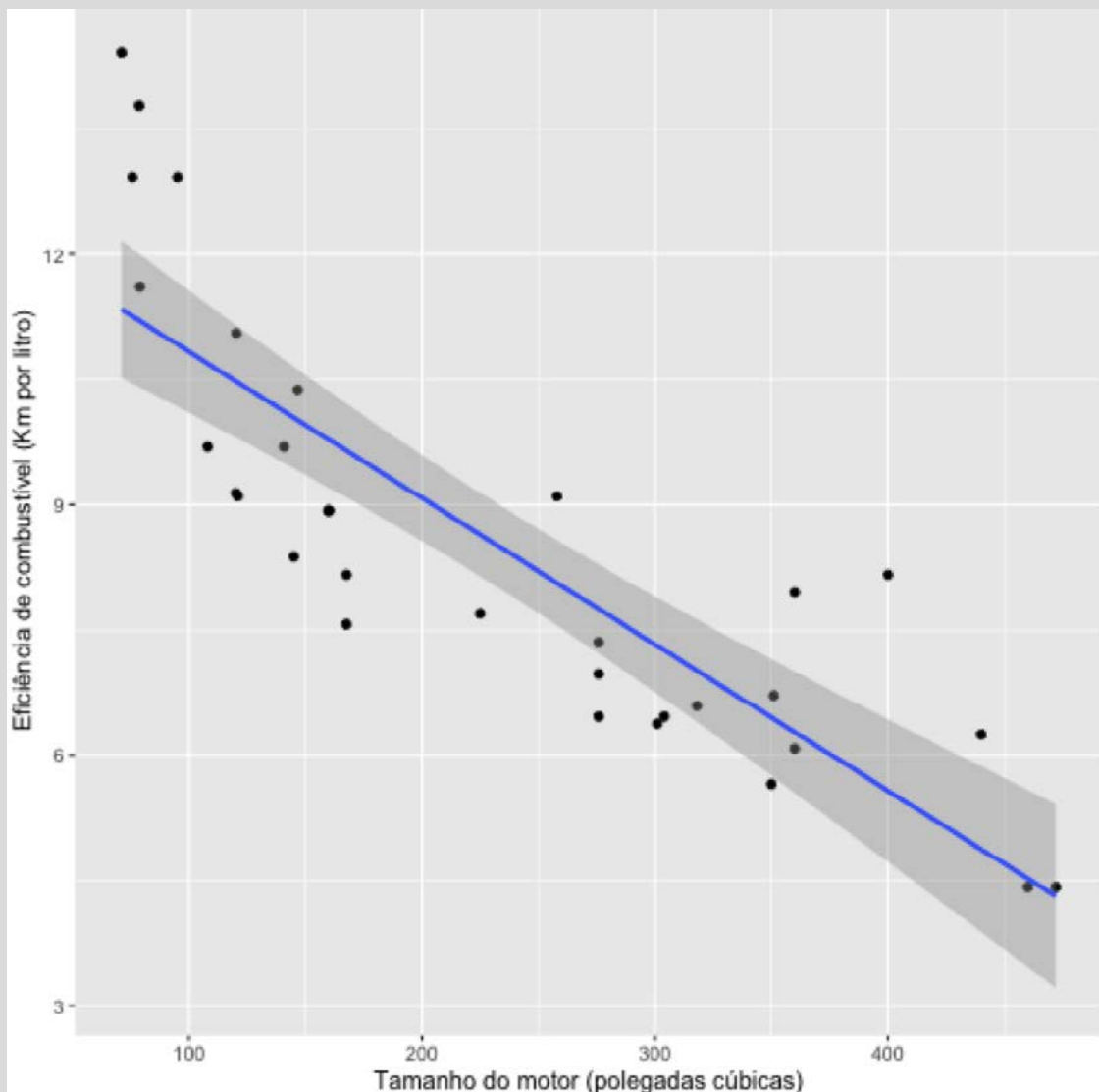
1. O aprendizado de máquina (AM) pode ser utilizado em inúmeros cenários. Considere as afirmações a seguir sobre possíveis aplicações do AM:
 - () É possível utilizar algoritmos de AM para detectar quais e-mails são *spam* e quais não.
 - () É possível utilizar AM para fazer identificação de imagens e classificar os objetos nela contidos.
 - () É possível utilizar algoritmos de AM para identificar o perfil de clientes de uma loja.
 - () Não é possível utilizar algoritmos de AM para fazer a identificação de emoções por meio da voz.

Qual das sequências a seguir indica, respectivamente, quais afirmações são verdadeiras (V) ou falsas (F)?

- a. F – F – V – V.
- b. V – V – F – F.
- c. V – V – V – F.
- d. V – F – V – V.
- e. V – V – V – V.

2. O aprendizado de máquina é amplamente utilizado na indústria automobilística para fazer a identificação de anomalias e proposição de otimizações. A seguir, é apresentado um gráfico que apresenta dados de vários carros e também um resultado obtido com aprendizado de máquina:

Figura 8 – Dados de aprendizado de máquina



Fonte: Prates (2019).

Qual das topologias a seguir corresponde corretamente à topologia de aprendizado de máquina ilustrada na figura?

- a. Classificação binária.
 - b. Classificação multiclasse.
 - c. Regressão.
 - d. Identificação de anomalias.
 - e. *Clustering*.
3. O aprendizado de máquina pode ser subdividido em duas grandes categorias: aprendizado supervisionado e não supervisionado. Assinale a alternativa que contém um algoritmo de aprendizado de máquina supervisionado.
- a. Regressão e classificação.
 - b. *Clustering* e *K-means*.
 - c. Regressão e *clustering*.
 - d. Classificação e *clustering*.
 - e. Regressão e decomposição em valores singulares.

Referências bibliográficas

BURGER, S. **Introduction to machine learning with R**: rigorous mathematical analysis. Sebastopol, CA: O'Reilly, 2018.

FACELI, K.; LORENA, A. C.; GAMA, J.; CARVALHO, A. C. P. L. F. **Inteligência artificial**: uma abordagem de aprendizado de máquina. São Paulo: LTC Editora, 2011.

HURWITZ, J.; KIRSCH, D. **Machine learning for dummies**. Hoboken, NJ: John Wiley & Sons, Inc., 2018. 70 p.

MITCHEL, M. B. **Machine Learning**. New York: McGraw-Hill Science/Engineering/Math, 1997. 432 p.

PRATES, W. R. **Aprendizado de máquina:** supervisionado e não supervisionado. 2018. Disponível em: https://www.wrprates.com/aprendizado-de-maquina-supervisionado-e-nao-supervisionado/#Modelo_supervisionado_1_Regressao. Acesso em: 25 jul. 2019.

SMOLA, A.; VISHWANATHAN, S. V. N. **Introduction to machine learning**. New York, NY: Cambridge University Press, 2008. 226 p.

TURING, A. M. Computing machinery and intelligence. **Mind: New Series**, Oxford, v. 59, n. 236, p. 433-460, out. 1950.

Gabarito

Questão 1 – Resposta C

O aprendizado de máquina pode ser aplicado em inúmeras situações. Inclusive no reconhecimento de emoções por meio da análise de áudios. Por meio da identificação do tom e da intensidade da fala, os algoritmos podem, sim, fazer esse tipo de reconhecimento. Por isso a sequência correta de afirmações é V – V – V – F.

Questão 2 – Resposta C

O resultado mostrado na figura é a utilização de um algoritmo de regressão, que cria uma função matemática baseada nos dados da etapa de treinamento.

Questão 3 – Resposta A

Dos algoritmos apresentados, somente a regressão e a classificação são supervisionadas. Logo, a alternativa A é a correta.



Algoritmos de *machine learning*: modelos preditivos

Autor: Lucas Claudino

► Objetivos

- Compreender a principal função dos algoritmos de *machine learning* preditivos, observando suas características básicas e finalidades.
- Conhecer os principais métodos de aprendizado de máquina preditivos e os algoritmos clássicos de cada um dos métodos.
- Conhecer as principais aplicações de *machine learning* preditivo e saber identificar qual técnica é a mais indicada para o problema a ser resolvido.

► 1. Introdução

O aprendizado de máquina pode ser classificado de diversas maneiras, e não existe uma delas que é considerada a forma mais correta de classificação. Você já estudou a divisão do AM em supervisionado, não supervisionado e de reforço. Outra forma de classificar o AM é a partir da finalidade do algoritmo utilizado. Para isso, a classificação mais comum é dividir o AM em modelos **preditivos** ou **descritivos**.

São aqueles que, a partir de um conjunto de dados rotulados fornecidos como entrada, conseguem elaborar estratégias para construir um estimador capaz de apresentar valores de saída de acordo com novos dados de entrada fornecidos. Já os modelos descritivos são aqueles capazes de avaliar os dados disponíveis e identificar informações relevantes e similares entre os objetos fornecidos para estudo. O resultado é alguma forma de classificação ou separação dos dados por meio dessas informações extraídas.

► 2. Introdução aos modelos preditivos

De acordo com Faceli *et al.* (2011), o algoritmo de aprendizado de máquina preditivo é aquele que, a partir de um conjunto de dados conhecidos, cria um estimador. Todo dado desse conjunto conhecido possui rótulos também conhecidos. Se os dados contidos nesse domínio conhecido forem finitos e tiverem valores nominais, o problema então é de classificação, e o estimador resultante do treinamento é o classificador. Se o domínio desse conjunto de dados for infinito, o problema é, então, de regressão, e o estimador é um regressor.

É possível também criar uma definição matemática formal para esse modelo de AM. Considere o conjunto de pares de dados como $D = \{(x_i, f(x_i)), i = 1, \dots, n\}$, sendo que $f(x_i)$ é uma função desconhecida. O algoritmo de AM preditivo analisa então esse conjunto para criar uma aproximação

\hat{f} de f . Com essa aproximação, o algoritmo consegue então estimar os rótulos para qualquer nova entrada x . O AM preditivo é, então, dividido entre classificação e regressão, de acordo com o domínio da função original $f(x_i)$ como:

- Classificação: $y_i = f(x_i) \in \{c_1, \dots, c_m\}$. Nesse caso, $f(x_i)$ assume somente valores discretos contidos em seu domínio.
- Regressão: $y_i = f(x_i) \in R$. Nesse caso, $f(x_i)$ assume valores contidos no domínio infinito R .

Para melhor entender essa distinção entre classificação e regressão, vamos considerar dois problemas conhecidos na área de aprendizado de máquina: o problema da Iris e o problema Swiss. O problema da Iris consiste em classificar flores em três espécies com base em alguns atributos físicos referentes a tamanho e largura das pétalas e sépalas das flores. Já o problema Swiss fornece dados para criar soluções capazes de relacionar a taxa de mortalidade a quatro parâmetros: fertilidade, agricultura, educação e renda.

Em ambos os casos, são consideradas como variáveis de entrada do problema as características que as diferenciam em suas classificações. Esses dados estão organizados no Quadro 1 e no Quadro 2.

Quadro 1 – Parte do conjunto de dados Iris

Tamanho (P)	Largura (P)	Tamanho (S)	Largura (S)	Espécie
5,1	3,5	1,4	0,2	Setosa
4,9	3,0	1,4	0,2	Setosa
7,0	3,2	4,7	1,4	Versicolor
6,4	3,2	4,5	1,5	Versicolor
6,3	3,3	6,0	2,5	Virgínica
5,8	2,7	5,1	1,9	Virgínica

Fonte: elaborada pelo autor.

Quadro 2 – Parte do conjunto de dados Swiss

Fertilidade	Agricultura	Educação	Renda	Mortalidade
80,2	17,0	12	9,9	22,2
83,1	45,1	9	84,8	22,2
92,5	39,7	5	83,4	20,2
85,8	36,5	7	33,7	20,3
76,9	43,5	15	5,2	20,6

Fonte: elaborada pelo autor.

Ambos os quadros, ou conjuntos de dados, cada linha é uma observação diferente. A última coluna é o rótulo de cada observação, ou atributo-alvo, que é o valor da função $f(x_i)$; ou seja, é a variável dependente, pois depende dos valores das outras colunas. Os outros dados são os atributos de entrada, ou variável independente.

No problema da Iris, parcialmente contido no Quadro 1, o conjunto de dados de diversas flores divididas em três grupos: setosa, versicolor e virgínica. Nesse caso, os atributos de entrada são características (tamanho e largura) das pétalas e sépalas das flores. A partir desses atributos, é possível separar as plantas em três rótulos, ou classes, que são as espécies da planta: setosa, versicolor e virgínica.

O problema da Iris é largamente utilizado para o ensino do AM de classificação devido à característica das três classes. A espécie setosa é bastante separada das outras duas; porém, as espécies versicolor e virgínica possuem características muito semelhantes, por isso é mais difícil separá-las. Você pode observar essa diferença por meio da largura da sépala, por exemplo.

Os dados contidos no Quadro 2, referentes ao problema Swiss, relacionam a taxa de mortalidade a diversas características da população. Nesse caso, as variáveis independentes são fertilidade, agricultura, educação e renda, enquanto a variável dependente é a taxa de mortalidade.

Em ambos os casos, o objetivo do algoritmo é aprender por meio desses dados de treinamento para criar uma função $\hat{f}(x_i)$ que seja capaz de mapear novos dados inseridos. No caso do problema Iris, a função terá que classificar novas plantas em alguma das três espécies. Já no caso do problema Swiss, a função $\hat{f}(x_i)$ deverá estimar qual o valor da taxa de mortalidade.

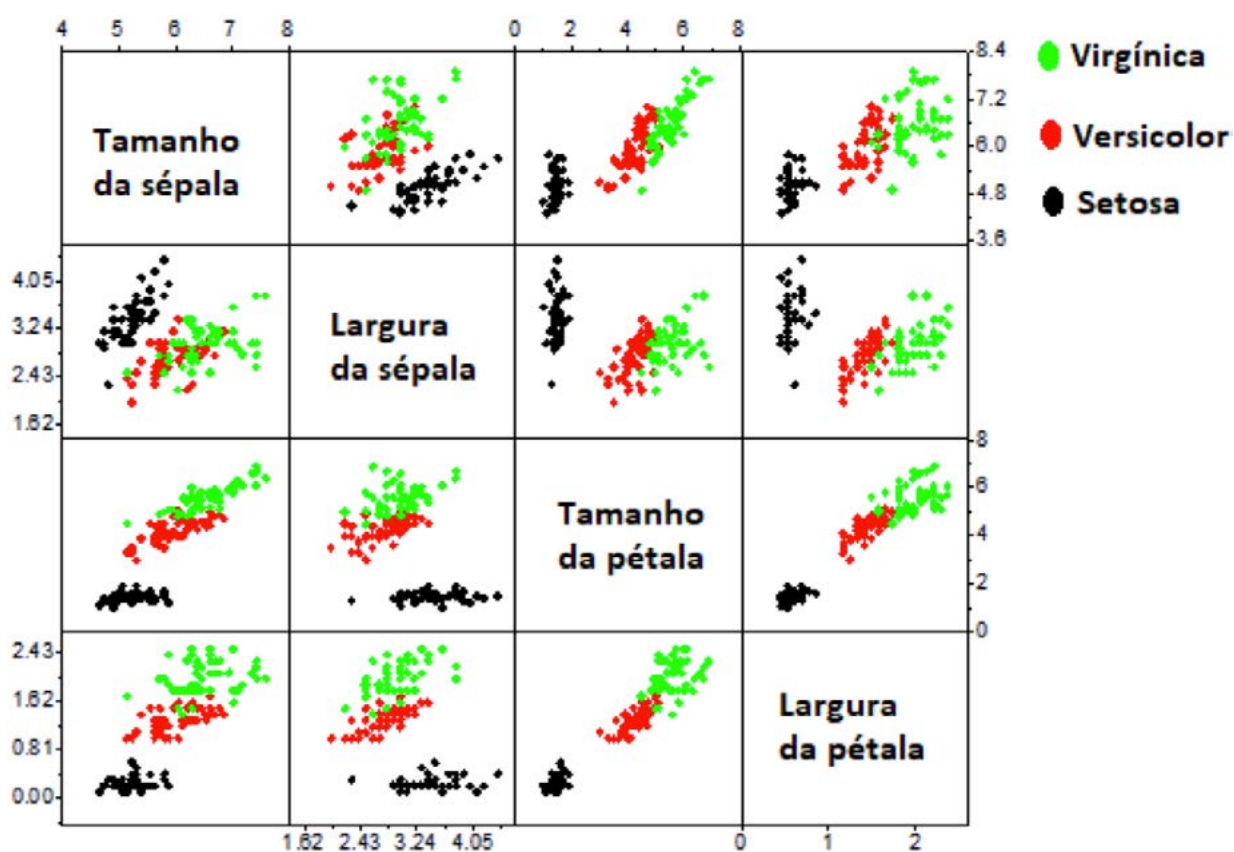
A função $\hat{f}(x_i)$ é uma estimativa, ou seja, mesmo sendo uma excelente aproximação, pode haver alguns casos em que $\hat{f}(x_i) \neq f(x_i)$. Por isso que o modelo tem que ser criado corretamente. Para medir a qualidade da estimativa, existem as chamadas funções custo, e para cada tipo de problema há uma função custo mais indicada. Para problemas de classificação, é utilizada a função binária 0 – 1: quando a estimativa for correta ($\hat{f}(x_i) = f(x_i)$), o custo é 0, já quando a estimativa é incorreta ($\hat{f}(x_i) \neq f(x_i)$), o custo é 1. Para os problemas do tipo regressão, costuma-se utilizar o custo chamado erro quadrático médio (MSE – *mean squared error*) $MSE = E [\hat{f}(x_i) - f(x_i)]^2$.

Os algoritmos de aprendizado de máquina preditivos podem ser subclassificados em métodos baseados em distância, otimização, procura e métodos probabilísticos. A seguir, você verá as principais características de cada um desses métodos e os algoritmos mais populares das subcategorias do AM preditivo.

2.1 Métodos baseados em distância

Segundo Faceli *et al.* (2011), o AM preditivo baseado em distância busca calcular a distância entre os pontos em análise. Esse método se baseia na hipótese de que dados semelhantes ficam dispostos de maneira concentrada no domínio do conjunto da função. Para melhor entender como funciona um método baseado em distâncias, observe a Figura 1, que é a representação gráfica dos dados obtidos com a base de dados Iris.

Figura 1 – Diagrama planificado da classificação do problema Iris



Fonte: elaborada pelo autor.

A Figura 1 possui diferentes gráficos, cada um relacionando duas das quatro variáveis independentes do Quadro 1. Em todos os gráficos, é possível observar que a classe setosa está geograficamente separada da versicolor e da virgínica. Então, o algoritmo consegue facilmente ser treinado a identificar qual é a região que representa a flor da espécie setosa. Porém, as classes virgínica e versicolor possuem vários elementos que se confundem, e o desafio maior do algoritmo é conseguir identificar pontos pertencentes a essas classes.

Os algoritmos baseados em distâncias irão, a cada novo ponto recebido para análise, calcular a distância entre ele e os pontos mais próximos. Com isso, saberão qual é a classe que está mais próxima ao ponto. Esses algoritmos são rotulados como “preguiçosos” (*lazy*), pois não aprendem ou criam um modelo baseados nos dados do treinamento, eles apenas memorizam os dados e criam um plano n-dimensional para possibilitar a análise das distâncias.

O algoritmo mais conhecido que se enquadra no método baseado em distâncias é o k-NN (*k nearest neighbors*). Basicamente, esse algoritmo identifica os k vizinhos mais próximos do ponto a ser analisado, para então verificar em qual classe o novo dado se enquadra.

► 3. Métodos probabilísticos

Métodos probabilísticos são muito utilizados quando não há informação completa sobre os dados utilizados. Nesses casos, as previsões são feitas com o auxílio do teorema de Bayes, que diz o seguinte:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Sendo $P(A|B)$ a probabilidade de o evento A acontecer, dada a situação B ; $P(B|A)$ é a verossimilhança entre A e B ; $P(B)$ e $P(A)$ são as probabilidades *a priori*.

O teorema de Bayes mostra que a probabilidade do evento B ocorrer, dado o conjunto ou evento A , depende da relação entre esses dois objetos, mas também depende da probabilidade das observações individuais de A e B .

Para encontrar as probabilidades *a priori*, você pode observar os dados e analisar com qual frequência os eventos A ou B ocorrem. Então, dado o conjunto total de observações, é possível extrair as probabilidades $P(A)$ e $P(B)$. Pode-se também observar a ocorrência de B para cada evento A que ocorreu. Mas como você vai observar a probabilidade de A ocorrer, para cada evento B , sendo que todos os eventos do tipo A já ocorreram? Para isso, você utiliza o Teorema de Bayes.

Assim como dito anteriormente, métodos probabilísticos podem ser utilizados, por exemplo, quando não há informação completa sobre os fatos, ou também quando os dados obtidos forem muito ruidosos.

Para entender melhor como esse método é utilizado, pense no caso de prever se uma pessoa terá algum problema cardíaco, sendo que os atributos preditivos mensuráveis disponíveis são: peso e frequência cardíaca. Nesse caso, informações muito importantes estão sendo ignoradas, como, por exemplo, a hereditariedade, consumo de bebidas alcoólicas ou o estresse. Por isso, utiliza-se, por exemplo, a probabilidade de pessoas que consomem bebidas alcoólicas terem problemas cardíacos. Essa seria a verossimilhança observada do Teorema de Bayes.

Algumas definições básicas de teoria das probabilidades são essenciais para a compreensão dos métodos probabilísticos. O espaço amostral Ω é a definição do conjunto composto por todos os resultados que podem acontecer em um dado experimento. A probabilidade de um evento E , pertencente ao espaço amostral Ω , ocorrer é $P(E)$ e satisfaz os conhecidos axiomas de Kolmogorof (PESTANA; VELOSA, 2002):

- $P(E) \leq 1$.
- Se o espaço amostral é Ω , então $P(\Omega) = 1$.
- Se F e G são eventos distintos, então a união entre eles é $P(F \cup G) = P(F) + P(G)$.

Esses axiomas são ferramentas necessárias para o cálculo da lei da probabilidade total: se $G_1, G_2, \dots, G_n \in \Omega$ então, para qualquer F , a seguinte probabilidade é verdadeira:

$$P(A) = \sum_{i=1}^n P(A|B_i) \times P(B_i)$$

Para encontrar o Teorema de Bayes, que é a base para os modelos probabilísticos, observe a probabilidade de ocorrência simultânea dos eventos A e B:

$$P(A \cap B) = P(A|B) \cdot P(B)$$

$$P(B \cap A) = P(B|A) \cdot P(A)$$

Igualando as equações:

$$P(A \cap B) = P(B \cap A) \therefore P(A|B) \cdot P(B) = P(B|A) \cdot P(A)$$

Portanto:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

O algoritmo mais popular que é baseado em métodos probabilístico que é o Naive Bayes, também conhecido como Bayes Ingênuo ou Bayes simples.

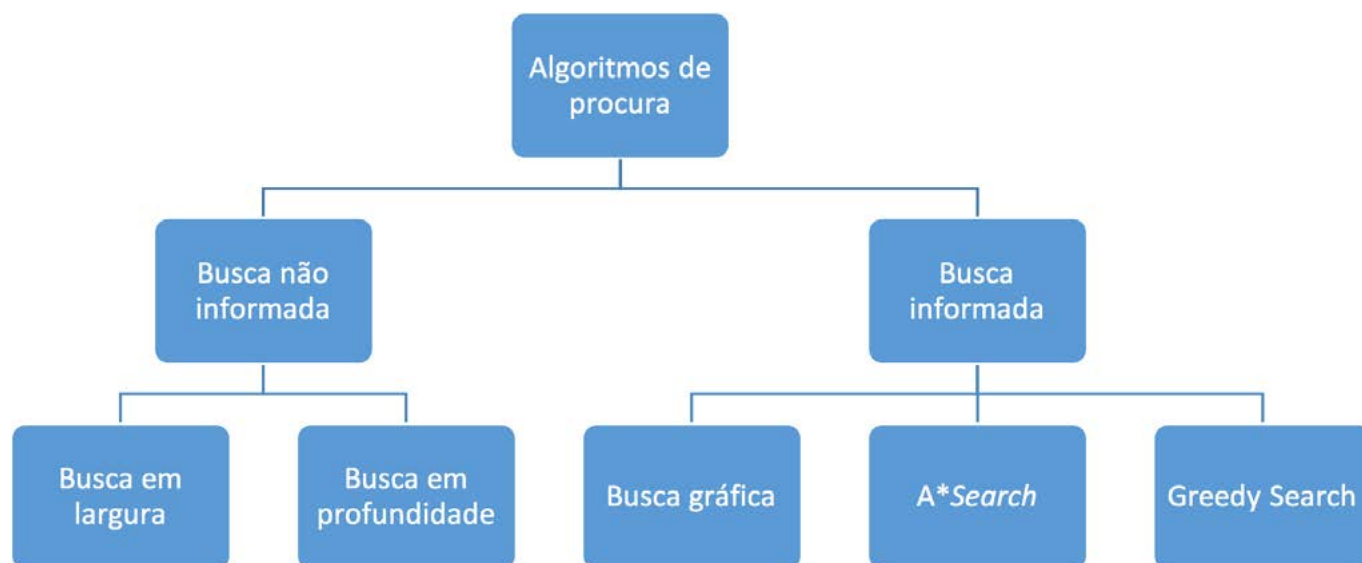
► 4. Métodos baseados em procura

Outra abordagem para o aprendizado de máquina é tratar o problema como uma procura no espaço de soluções disponíveis, até que a solução desejada seja alcançada. Segundo Russel e Norvig (2010), um problema de procura é basicamente constituído por:

- Espaço de estados: o conjunto com todos os estados possíveis nos quais o problema pode se encontrar.
- Um estado de início: o estado por onde a busca deve começar.
- Teste de objetivo: uma função capaz de observar o estado atual e retornar se é ou não o estado final.
- A solução para um problema de busca é uma sequência de atos, também chamada de plano, que transforma o estado inicial de tal maneira até chegar ao estado final.
- O plano é concretizado por meio do algoritmo de busca.

Existem diversos algoritmos de busca e cada um tem seus nichos de aplicação. Os métodos baseados em procura podem ser divididos da seguinte maneira:

Figura 2 – Divisão dos métodos baseados em procura



Fonte: elaborada pelo autor.

Os algoritmos de busca não informada não possuem informações adicionais sobre o estado final, eles só sabem se esse estado foi atingido ou não. O plano para chegar ao estado final a partir do estado inicial defere somente pela ordem e/ou tamanho das buscas tomadas por cada algoritmo. Devido a essas características, os algoritmos não informados são também conhecidos como algoritmos de “busca cega”, pois não enxergam onde estão, e só sabem se chegaram ou não ao destino (RUSSEL; NORVIG, 2010).

Observando a Figura 2, você pode identificar alguns exemplos de busca não informada. A busca em largura é um método que varre um ramo da árvore de procura até encontrar um nó em que não consiga mais prosseguir, mesmo que esse ramo esteja levando a busca para longe do estado final desejado.

Os algoritmos de busca informada utilizam conhecimento sobre o domínio do problema. Eles possuem informação sobre o estado final, o que ajuda na construção de buscas mais eficientes. Geralmente, esse tipo de algoritmo utiliza alguma função heurística para estimar quão perto a busca está do estado final. Essa heurística estima então o

custo necessário para passar de um estado ao outro. Caso o algoritmo detecte que está ficando mais distante da solução ótima, ele é capaz de retroceder a busca e procurar um caminho mais adequado (RUSSEL; NORVIG, 2010).

Um exemplo de busca informada é o algoritmo Greedy Search, que busca expandir o nó mais próximo da solução desejada, e para saber qual é esse nó, o algoritmo utiliza uma função heurística de estimação $h(x)$. Basicamente, $h(x)$ estima a distância entre o nó x e o estado final.

Outro exemplo é o algoritmo A* Search, que minimiza o custo total estimado para atingir a solução. O custo é também estimado por meio de funções heurísticas, e a estratégia de otimização varia para cada algoritmo utilizado, sendo ela otimização linear ou não.

PARA SABER MAIS



Algoritmos de busca são amplamente utilizados em problemas de localização de rotas ótimas, em que o aplicativo tem que fornecer a direção para o usuário chegar ao destino. Outra aplicação é a montagem automática de produtos, na qual o robô precisa estabelecer um plano de montagem que otimize alguma função custo, como, por exemplo, a energia gasta durante o processo.

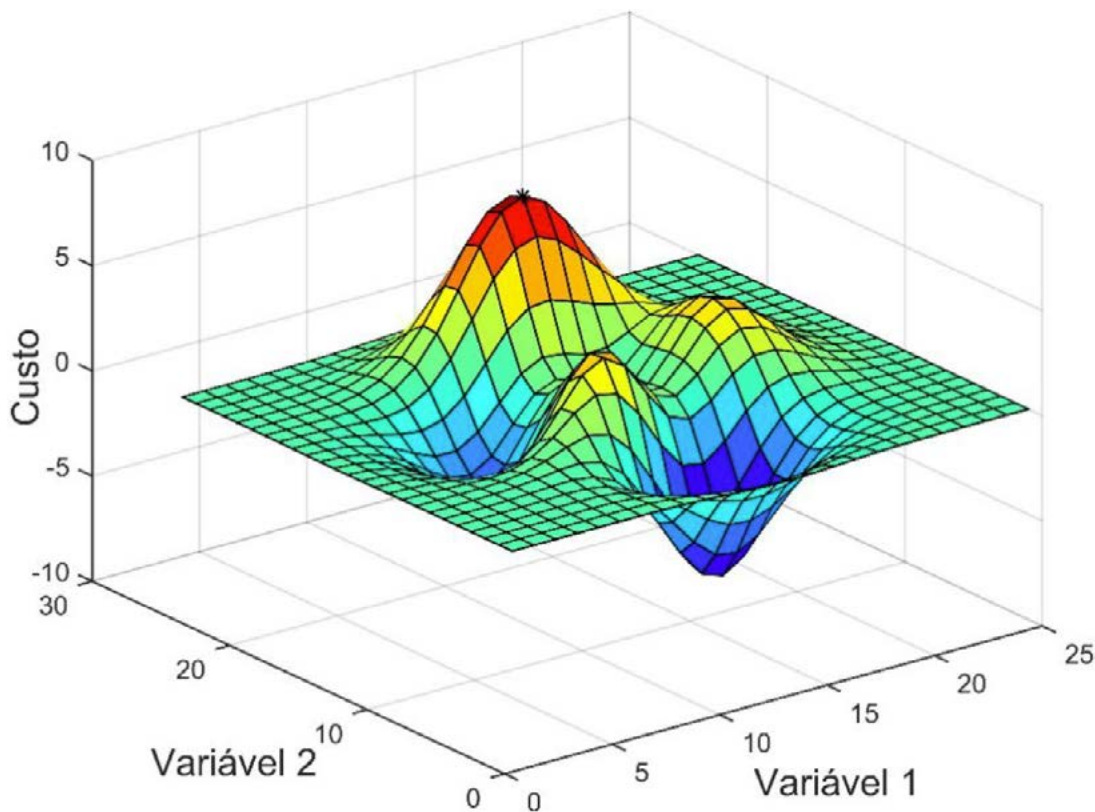
► 5. Métodos baseados em otimização

Existem estratégias de aprendizado de máquina que chegam à solução do problema por meio de estratégias de otimização de uma função. Nesses métodos, o problema é criado de tal forma que seu objetivo é otimizar (maximizar ou minimizar) uma função custo (ou função objetivo).

Pense no caso de otimização da produção, um problema bastante atual e no qual inúmeras indústrias estão prestando atenção. A otimização pode ser de diversos fatores, como redução de custos, redução de tempo, redução de poluição, aumento de produtividade. Você consegue perceber que todos esses problemas buscam minimizar ou reduzir alguma função? O objetivo é conseguir dados suficientes para descrever essas funções.

A Figura 3 ilustra uma função custo qualquer, que é dependente das variáveis 1 e 2. Claramente, é possível observar pontos de máximo e mínimo globais a partir da inspeção visual.

Figura 3 – Exemplo de função custo com duas variáveis independentes



Fonte: elaborada pelo autor.

Suponha que o custo ilustrado na Figura 3 é a quantidade relativa de itens produzidos por uma determinada indústria. Custo positivo significa que a produção está acima da meta e custo negativo significa que está abaixo do desejado. Com esses dados, um algoritmo de otimização tem que conseguir encontrar qual o ponto máximo da função custo, pois isso significa que a produção está em seu máximo.

Observe que a solução não é tão simples como parece, pois a função custo ilustrada não possui somente o ponto máximo global. Ela apresenta também os chamados máximos locais, que são aqueles pontos em que há um pico no valor da função, porém existem picos ainda maiores. São esses pontos que o algoritmo de otimização precisa evitar, pois eles são conhecidos com “falsos ótimos”.

ASSIMILE

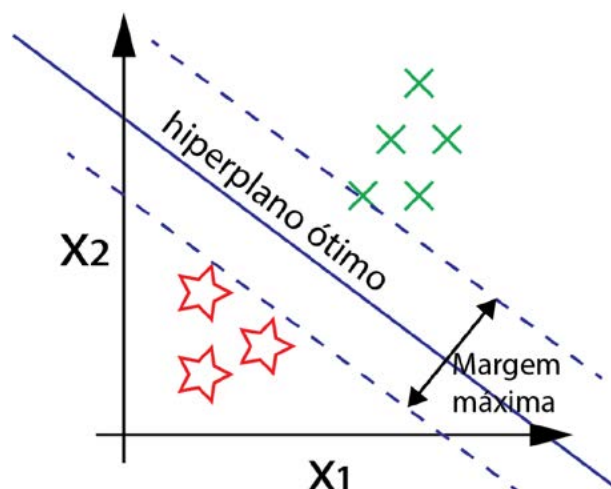


Para os algoritmos de otimização poderem maximizar (ou minimizar) uma função, ela precisa realmente apresentar pontos de máximo ou mínimo. Ou seja, a função custo precisa ser côncava ou convexa, pois isso significa que ela realmente apresentará os pontos ótimos desejados.

Duas técnicas bastante conhecidas de aprendizado de máquina baseado em métodos de otimização são as redes neurais artificiais (RNA) e as SVM (*support vector machine* ou máquinas de vetor de suporte) (BRAGA *et al.*, 2007; HAYKIN, 1999; CRISTIANINI; SHAW-TAYLOR, 2000). As RNAs são inspiradas no funcionamento das redes neurais biológicas que formam o cérebro humano. Esses sistemas são capazes de aprender para executar tarefas sem terem que ser programados para nenhuma tarefa específica. Esse tipo de algoritmo interpreta os dados por meio de um tipo de percepção de máquina, identificando, rotulando e classificando os dados disponíveis.

As SVM são algoritmos que buscam encontrar algum hiperplano em um espaço N-dimensional capaz de fazer a separação ótima entre os dados. Para entender melhor a ideia de hiperplano, observe a Figura 4, que ilustra um hiperplano em um sistema de duas variáveis. Nesse caso, o hiperplano é uma reta. Se fosse um sistema com três variáveis, o hiperplano seria uma superfície.

Figura 4 – Ilustração de um hiperplano 2D



Fonte: elaborada pelo autor.

Observe a Figura 4, que ilustra o hiperplano 2D. Só é possível considerar que esse hiperplano é ótimo pois ele está localizado exatamente no ponto que oferece a margem máxima de separação entre as duas classes dos dados.

Nesta Leitura Fundamental, você pôde compreender que o aprendizado de máquina possui uma abordagem bastante dinâmica e pode ser utilizado para a resolução de inúmeros problemas. Os modelos preditivos de AM podem ser classificados em métodos baseados em otimização, procura, distância e métodos probabilísticos. Cada uma dessas técnicas possui seus algoritmos mais conhecidos, e é importante que você conheça uma grande variedade deles, pois só assim será capaz de julgar qual algoritmo é mais indicado para o problema que for resolver.

TEORIA EM PRÁTICA



Um problema clássico no AM é o *8-puzzle*, que é uma espécie de jogo em uma placa de 3x3 peças, sendo que 8 peças são numeradas e uma posição está em branco.



VERIFICAÇÃO DE LEITURA

1. O problema Iris é uma situação clássica no estudo de *machine learning*. Nesse problema, o algoritmo deve ser capaz de distinguir os objetos entre três espécies de planta: versicolor, virgínica ou setosa.

Assinale a alternativa que afirma corretamente que tipo de problema de aprendizado de máquina preditivo é o problema Iris.

- a. Regressão.
 - b. Otimização.
 - c. Classificação.
 - d. Busca informada.
 - e. Busca não informada.
-
2. Assinale a alternativa que contém corretamente o nome do método de aprendizado de máquina que se baseia no princípio de que dados semelhantes ficam organizados de maneira concentrada no domínio do conjunto de dados do problema, para então poder classificar corretamente os dados.
- a. Regressão.
 - b. Método de busca.
 - c. Método de otimização.
 - d. Método das distâncias.
 - e. Método probabilístico.

3. Existe uma metodologia de aprendizado de máquina que é mais indicada para ser utilizada em casos nos quais não há informação completa sobre os fatos, ou também quando os dados obtidos forem muito ruidosos.

Assinale a alternativa que afirma corretamente qual método de *machine learning* é mais indicado para esse caso.

- a. Método probabilístico.
- b. Método baseado em busca.
- c. Método baseado em distância.
- d. Método baseado em otimização.
- e. Método baseado em aproximação linear.

Referências bibliográficas

FACELI, K.; LORENA, A. C.; GAMA, J.; CARVALHO, A. C. P. L. F. **Inteligência artificial: uma abordagem de aprendizado de máquina**. São Paulo: LTC Editora, 2011.

GANDHI, R. **Support vector machine: introduction to machine learning algorithms**. 2018. Disponível em: <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>. Acesso em: 31 jul. 2019.

GRIVA, I.; NASH, S. G.; SOFER, A. **Linear and nonlinear optimization**. 2. ed. Fairfax: George Mason University, 2009. 742 p.

PESTANA, D. D.; VELOSA, S. F. **Introdução à probabilidade e à estatística**. 4. ed. Lisboa: Fundação Calouste Gulbenkian, 2008. 1.164 p.

RUSSEL, S.; NORVIG, P. **Artificial Intelligence: a modern approach**. 3. ed. New Jersey: Pearson Education Inc., 2010. 1.132 p.

SMOLA, A.; VISHWANATHAN, S. V. N. **Introduction to machine learning**. New York, NY: Cambridge University Press, 2008. 226 p.



Gabarito

Questão 1 – Resposta C

O problema Iris é do tipo classificação, em que o algoritmo deve classificar corretamente qualquer nova planta entre as três classes já conhecidas.

Questão 2 – Resposta D

O método das distâncias se baseia na hipótese de que dados semelhantes ficam dispostos de maneira concentrada no domínio do conjunto da função.

Questão 3 – Resposta A

Métodos probabilísticos podem ser utilizados, por exemplo, quando não há informação completa sobre os fatos, ou também quando os dados obtidos forem muito ruidosos.



Regressão, *k-nearest neighbors* (kNN) e Naive Bayes

Autor: Lucas Claudino

► Objetivos

- Compreender as principais características e aplicabilidades dos algoritmos de regressão aplicados ao aprendizado de máquina.
- Entender o funcionamento dos algoritmos de aprendizado de máquina baseados em distância, mais especificamente, o algoritmo kNN.
- Estudar as propriedades básicas do algoritmo Naive Bayes, seus principais equacionamentos e implementação básica para a predição de rótulos.

► 1. Introdução

O aprendizado de máquina é utilizado em diversas aplicações, como, por exemplo, detecção de fraudes, filtragem de *spam* e análise de sentimentos. Cada aplicação precisa de uma técnica diferente de aprendizado. Neste estudo, você verá três técnicas muito utilizadas em diversos problemas: regressão, *k-nearest neighbors* (kNN) e Naive Bayes.

A regressão é uma técnica em que o estimador é um regressor. A kNN é um método baseado em distâncias e o Naive Bayes é um método probabilístico baseado na teoria de probabilidade de Bayes.

► 2. Regressão

Algoritmos de regressão consideram técnicas que você muito provavelmente já utilizou diversas vezes, mas não se deu conta de que se trata de uma forma de aprendizado de máquina. A essência da regressão é criar uma maneira de formular uma função matemática que, a partir da observação de dados fornecidos, seja capaz de estimar com precisão o valor de uma variável de interesse. Essa estimativa vale para qualquer plano N dimensional.

Por simplicidade, primeiro estude o caso em que uma variável y é dependente de x . Ou seja, você tem disponível para análise n observações da seguinte maneira:

$$y_i = f(x_i), i = 1, \dots, n$$

O algoritmo regressor vai então criar uma função $\hat{f}(x_i)$ que aproxime satisfatoriamente $f(x_i)$ para todo e qualquer valor de x_i . Matematicamente, pode-se definir então:

$$\hat{f}(x_i) \approx f(x_i)$$

Sendo que a aproximação criada só é tida como satisfatória caso apresente um erro δ menor ou igual a um valor predeterminado, ou seja:

$$[\hat{f}(x_i) - f(x_i)] < \delta$$

Para entender melhor como funciona a regressão, observe um exemplo do banco de dados *trees* (Quadro 1), que descreve o volume de uma árvore de cerejas a partir da sua altura e da circunferência do tronco. A circunferência e a altura são facilmente medidas; porém, o volume não é uma grandeza constatada facilmente, por isso o método de regressão é bastante utilizado.

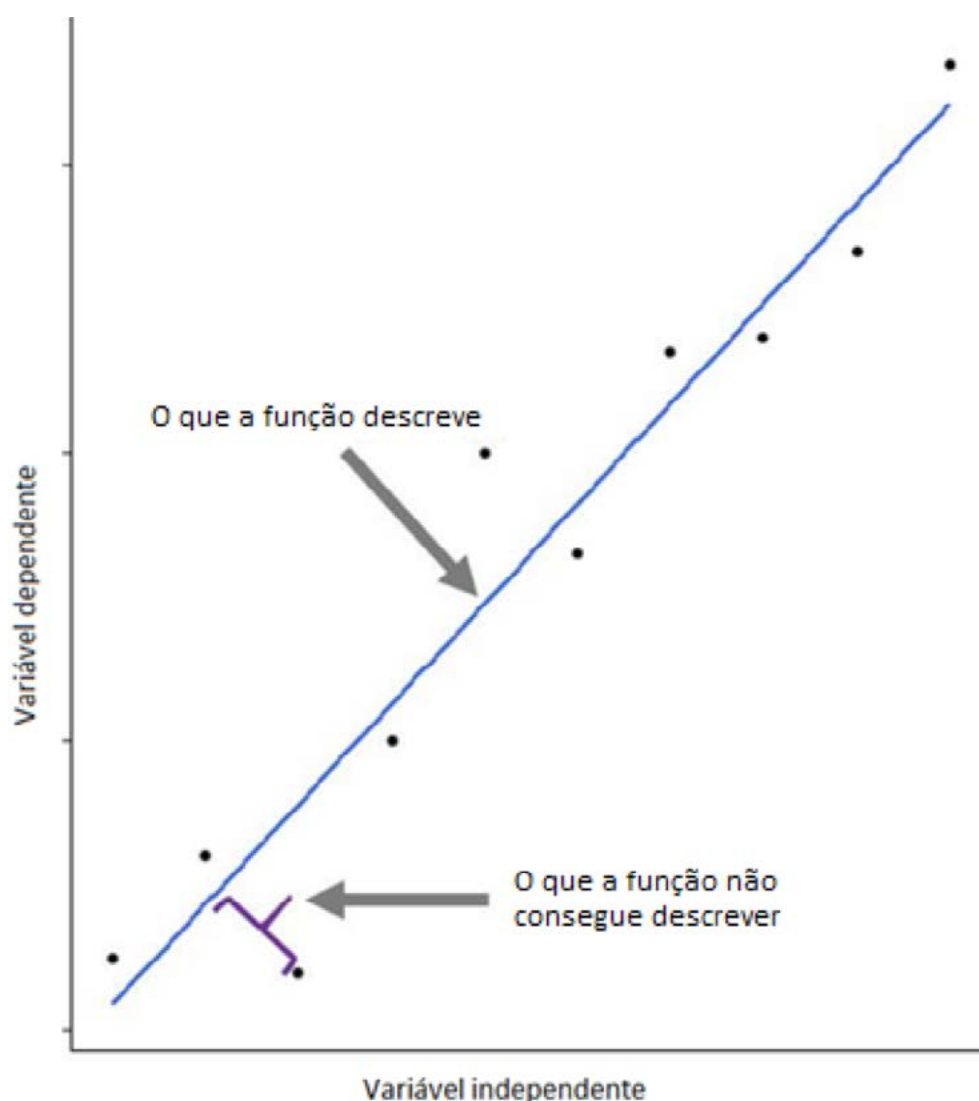
Quadro 1 – Subconjunto do banco de dados *trees*

Circunferência (in)	Altura (ft)	Volume (ft ³)
8,3	70	10,3
8,6	65	10,3
8,8	63	10,2
10,5	72	16,4
10,7	81	18,8
10,8	83	19,7

Fonte: elaborada pelo autor.

O algoritmo de regressão pode criar uma função $\hat{f}(x)$ que relaciona o volume da árvore com sua circunferência. Para esse caso, pode-se utilizar uma regressão linear, que é uma reta criada de tal forma a minimizar a diferença entre os pontos (amostras) e a linha (função), assim como ilustra a Figura 1.

Figura 1 – Regressão linear

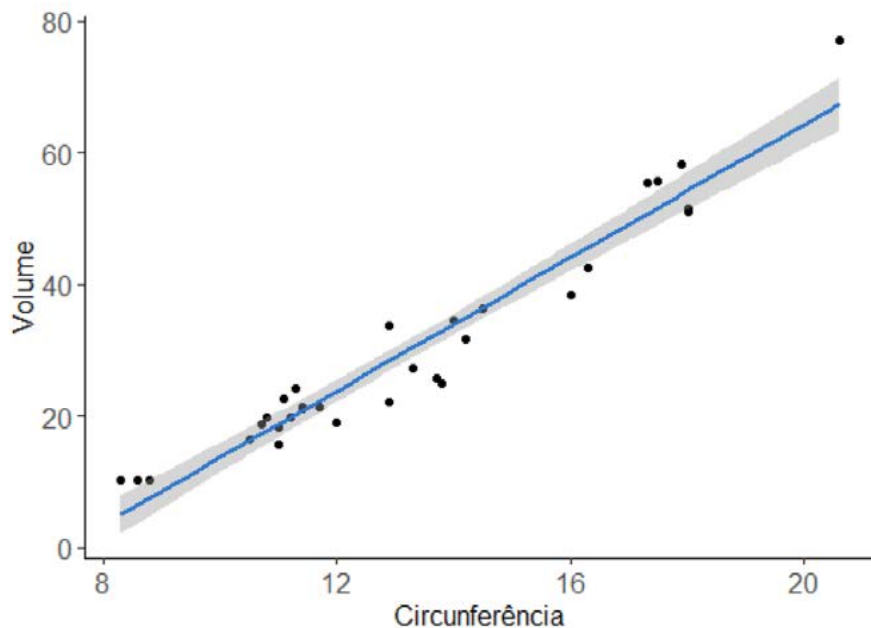


Fonte: elaborada pelo autor.

Por meio da Figura 1, também é possível observar que a função, mesmo sendo um método que busca minimizar a diferença entre o estimado e o observado, não consegue descrever com 100% de exatidão qual é o valor correto. Para isso, existe um parâmetro chamado **intervalo de confiança**, que representa o intervalo ao redor da função criada em que uma porcentagem dos dados observados estará contida.

A Figura 2 mostra uma regressão linear para o caso do banco de dados *trees*, com um intervalo de confiança de 95% evidenciado ao redor do regressor. Isso significa que 95% dos dados reais estarão contidos dentro desse intervalo.

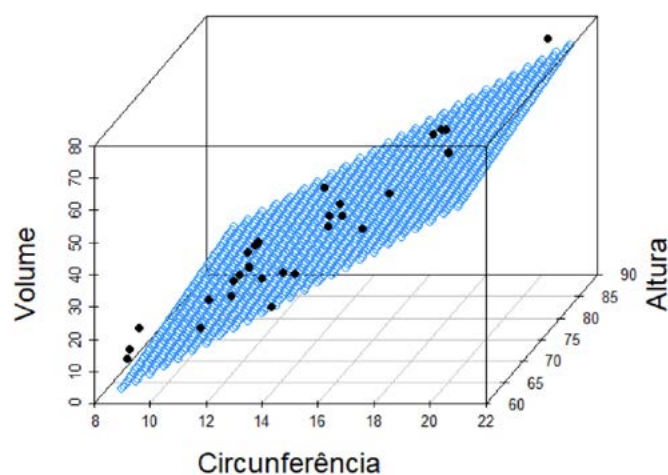
Figura 2 – Regressão linear com as observações e o intervalo de confiança



Fonte: elaborada pelo autor.

É interessante observar que o regressor utilizado continua tendo vários pontos fora do intervalo de confiança. Isso pode significar que o método escolhido (a regressão linear com uma variável) não é o mais indicado para estimar qual o volume da árvore. Para isso, você pode, por exemplo, utilizar uma regressão linear multivariável, que fará uso de duas variáveis independentes (circunferência e altura) para estimar qual o volume da árvore. Um exemplo do uso dessas duas variáveis independentes para o mesmo problema está ilustrado na Figura 3.

Figura 3 – Regressão multidimensional



Fonte: elaborada pelo autor.

Existem também diversos outros tipos de regressão, como, por exemplo, regressão polinomial, regressão quântica, ElasticNet, por mínimos quadrados parciais, regressão de Poisson, etc. Cabe a você estudar o problema de interesse e descobrir qual é a mais adequada para o caso.

► 3. *K-nearest neighbors* (kNN)

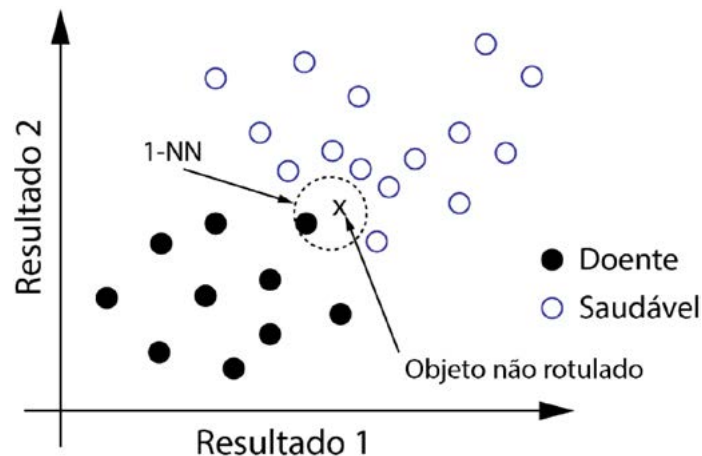
O algoritmo kNN (*k-nearest neighbors*) ou k-vizinhos mais próximos é uma técnica de aprendizado de máquina supervisionado baseado em distância. Ele utiliza a etapa de treinamento para memorizar o conjunto de treinamento e então utiliza essa memória para descobrir o rótulo de qualquer novo objeto inserido. O rótulo é descoberto por meio da distância entre o novo objeto e os objetos vizinhos. Para encontrar essa distância, existem algumas metodologias de cálculo utilizadas, como a distância de Manhattan, distância euclidiana, distância de Chebychev, distância de Mahalanobis, entre outras. A métrica mais usual para esse cálculo é a distância euclidiana, mostrada na equação a seguir:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{l=1}^d (x_i^l - x_j^l)^2}$$

A equação é o cálculo da distância euclidiana entre dois vetores \mathbf{x}_i e \mathbf{x}_j , sendo x_i^l e x_j^l elementos desse vetor, e l o l -ésimo atributo a ser analisado.

Para entender melhor como seria essa classificação a partir dos vizinhos mais próximos, considere um exemplo em que os resultados de dois exames classificam um objeto (indivíduo) como saudável ou doente. Um algoritmo kNN vai então memorizar os objetos já rotulados, sabendo seus respectivos rótulos e características. O classificador mais básico do tipo kNN é o 1-NN, que analisa somente qual é o vizinho mais próximo. Observe a Figura 4, em que um objeto não rotulado está sendo classificado como “doente”, pois o objeto rotulado mais próximo a ele é dessa classe.

Figura 4 – Classificação 1-NN

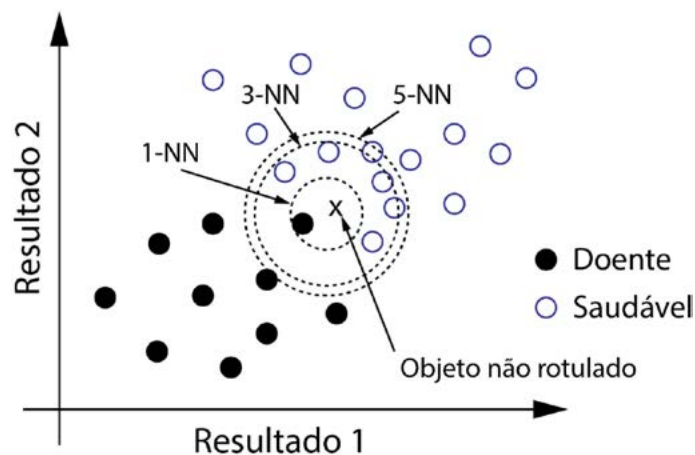


Fonte: elaborada pelo autor.

Porém, observe na Figura 4 que, apesar de o objeto mais próximo ao ponto "x" pertencer à classe "doente", há muitos outros objetos que podem ajudar nessa classificação e, inclusive, alterar o resultado. Por isso o algoritmo kNN é, geralmente, utilizado com análise de uma quantidade maior de vizinhos próximos, como, por exemplo, 3-NN ou 5-NN.

Observando a Figura 5, é possível observar que o uso de um algoritmo kNN mais elaborado é capaz de mudar totalmente o resultado. Por meio dos métodos 3-NN e 5-NN, o mesmo objeto não rotulado, que antes havia sido classificado como "doente", agora passa a ser classificado como "saudável".

Figura 5 – Classificação kNN com 1, 3 e 5 vizinhos próximos



Fonte: elaborada pelo autor.

A Figura 5 mostra como é importante a escolha correta do algoritmo a ser utilizado. Ao utilizar o 1-NN, o que houve foi que o novo objeto foi rotulado como “doente”, pois ele estava próximo a um *outlier*, que é um ponto ou observação que apresenta comportamento muito distinto daquele referente à classe à qual ele pertence.

O algoritmo kNN possui uma implementação bastante direta, tendo que seguir basicamente três passos (RASCHKA, 2015):

- a. Escolher o número de k e a métrica de distância.
- b. Encontrar os k vizinhos da amostra de interesse.
- c. Associar um rótulo a partir do “voto” da maioria, que é a maior quantidade de vizinhos próximos.

Para melhor entender como funciona a implementação da técnica kNN, observe a Figura 6, que apresenta dois pseudoalgoritmos da técnica kNN evidenciando as fases de treinamento e teste.

Figura 6 – Pseudoalgoritmo kNN

Algoritmo 1: *Treinamento kNN.*

Entrada: conjunto de treinamento $T = \{x_i, y_i\}_{i=1}^n$, valor de k e uma medida de distância $d(\cdot, \cdot)$

Saída : classificador kNN

- 1 armazenar o conjunto de treinamento e o valor de k ;
-

Algoritmo 2: *Teste kNN.*

Entrada: classificador kNN e um objeto x cuja classe é desconhecida

Saída : classe y atribuída a x

- 1 buscar pelos k objetos mais próximos a x no conjunto de dados de treinamento do classificador kNN informado;
 - 2 dentre os k vizinhos, determinar y como a classe mais frequente entre eles, resolvendo possíveis empates de maneira arbitrária;
-

Fonte: adaptada de Padilha (2017, p. 18).

Essa análise feita sobre a diferença entre a utilização de algoritmos 1-NN, 3-NN e 5-NN mostra como é importante o projeto do algoritmo a ser utilizado. Para a técnica kNN, você deve considerar o compromisso entre confiabilidade do resultado e complexidade computacional, uma vez que quanto mais vizinhos o algoritmo tiver que analisar, maior é a complexidade computacional do cálculo das distâncias. Para um k pequeno, a resposta, mesmo sendo entregue de forma rápida, não é a mais confiável, porém, caso você escolha um valor de k muito alto, o algoritmo pode nunca encontrar os k vizinhos próximos necessários para completar a rotina de código. Na verdade, a escolha do valor de k para um algoritmo kNN é um dos principais pontos, por isso vale a pena investir muito tempo no estudo do valor ótimo de k .

ASSIMILE

O algoritmo kNN memoriza os dados de treinamento e, durante a etapa de teste, encontra as k observações mais próximas do dado não rotulado para então julgar a qual classe esse objeto pertence. Esse é o motivo pelo qual o kNN é um algoritmo “preguiçoso”: ele não cria modelos, somente memoriza os dados de treinamento e utiliza suas características para fazer comparações.

► 4. Naive Bayes

Para entender o funcionamento do estimador Naive Bayes, é importante relembrar a equação que define o Teorema de Bayes:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

Essa regra possibilita o cálculo de um único termo $P(B|A)$ a partir de três outros termos: $P(A|B)$, $P(B)$ e $P(A)$. A regra de Bayes é bastante utilizada em casos nos quais existe uma boa estimativa desses três parâmetros, mas não do quarto. Um exemplo é a relação entre causa e efeito, como segue:

$$P(causa|efeito) = \frac{P(efeito|causa)P(causa)}{P(efeito)}$$

A probabilidade condicional de se ter um efeito dada uma determinada causa $P(efeito|causa)$ sinaliza a direção causal, ao passo que $P(causa|efeito)$ é a direção do diagnóstico, pois analisa a probabilidade de ser uma causa dado um efeito que já ocorreu.

Em tarefas de diagnósticos médicos, esse teorema pode ser bastante aplicado, pois os médicos sabem $P(sintoma|doença)$ (probabilidade de ter um sintoma dada uma determinada doença) e desejam diagnosticar a doença por meio de $P(doença|sintoma)$. Por exemplo, um médico pode saber que a meningite faz com que o paciente sofra de torcicolo em 70% das vezes. O médico também sabe alguns fatos indiscutíveis: a probabilidade de alguém ter meningite é 1/50.000, e a chance de qualquer paciente ter torcicolo é 1%. Considere t a proposição de o paciente ter torcicolo e m de ter meningite. Então:

$$\begin{aligned}P(t|m) &= 0,7 \\P(m) &= \frac{1}{50000} \\P(t) &= 0,01 \\P(m|t) &= \frac{P(t|m)P(m)}{P(t)} = \frac{0,7 \times 1/50000}{0,01} = 0,0014\end{aligned}$$

Isso significa que menos de 1 em cada 700 pacientes com torcicolo estão com meningite.

As probabilidades *a priori* e condicionais necessárias para a utilização do método Naive Bayes são obtidas a partir da análise dos dados durante a etapa de treinamento. A probabilidade *a priori* pode ser obtida por meio da contagem de cada objeto y_i pertencente a uma determinada classe. Já a probabilidade condicional é obtida com a observação de um determinado valor, tendo que esse objeto pertence a uma classe, porém é necessário haver a distinção entre valores (atributos) contínuos e nominais. Essa distinção é feita da seguinte forma:

No caso de atributos nominais, o conjunto de possíveis valores é um conjunto enumerável. Para calcular a probabilidade condicional, basta manter um contador para cada valor de atributo por classe. No caso de atributos contínuos, quando o número de possíveis valores é infinito, há duas possibilidades. A primeira é assumir uma distribuição particular para os valores do atributo, e geralmente é assumida a distribuição normal. A segunda alternativa é discretizar o atributo em uma fase de pré-processamento. Já foi mostrado que a primeira possibilidade produz piores resultados que a última. (FACELI; LORENA; GAMA, 2011, p. 6)

O algoritmo baseado na regra de Bayes é chamado de Bayes Ingênuo, pois parte do princípio que, para a observação de um vetor de objetos \mathbf{X} , todos os elementos $\{x_1, \dots, x_n\}$ são independentes, o que permite a seguinte manipulação com a equação do Teorema de Bayes:

$$P(Y = k|\mathbf{X}) = \frac{P(\mathbf{X}|Y = k)P(Y = k)}{P(\mathbf{X})}$$
$$P(Y = k|x_1, \dots, x_n) = \frac{P(x_1|Y = k)P(x_2|Y = k) \cdots P(x_n|Y = k)P(Y = k)}{P(x_1)P(x_2) \cdots P(x_n)}$$

3.1 Implementação do método Naive Bayes em R

Para melhor entender o funcionamento do algoritmo Naive Bayes, analise a implementação dele utilizando a linguagem R. Para isso, serão utilizados o banco de dados Iris e os pacotes "*el071*", "*klar*",

"*caret*" e "*bnlearn*". A implementação por meio desses pacotes se torna bastante simples, e o resultado mostra tanto a densidade de cada uma das três espécies quanto a quantidade de acertos obtidas com o uso do estimador Naive Bayes. A Figura 7 contém o algoritmo para implementação utilizando o R. Observe que o algoritmo contém uma das inúmeras formas possíveis de se implementar a técnica Naive Bayes. A implementação utilizada conta com o auxílio do pacote "*caret*", que facilita a descrição em linhas de código, e possibilita o foco na análise dos resultados.

Figura 7 – Algoritmo de implementação do Naive Bayes

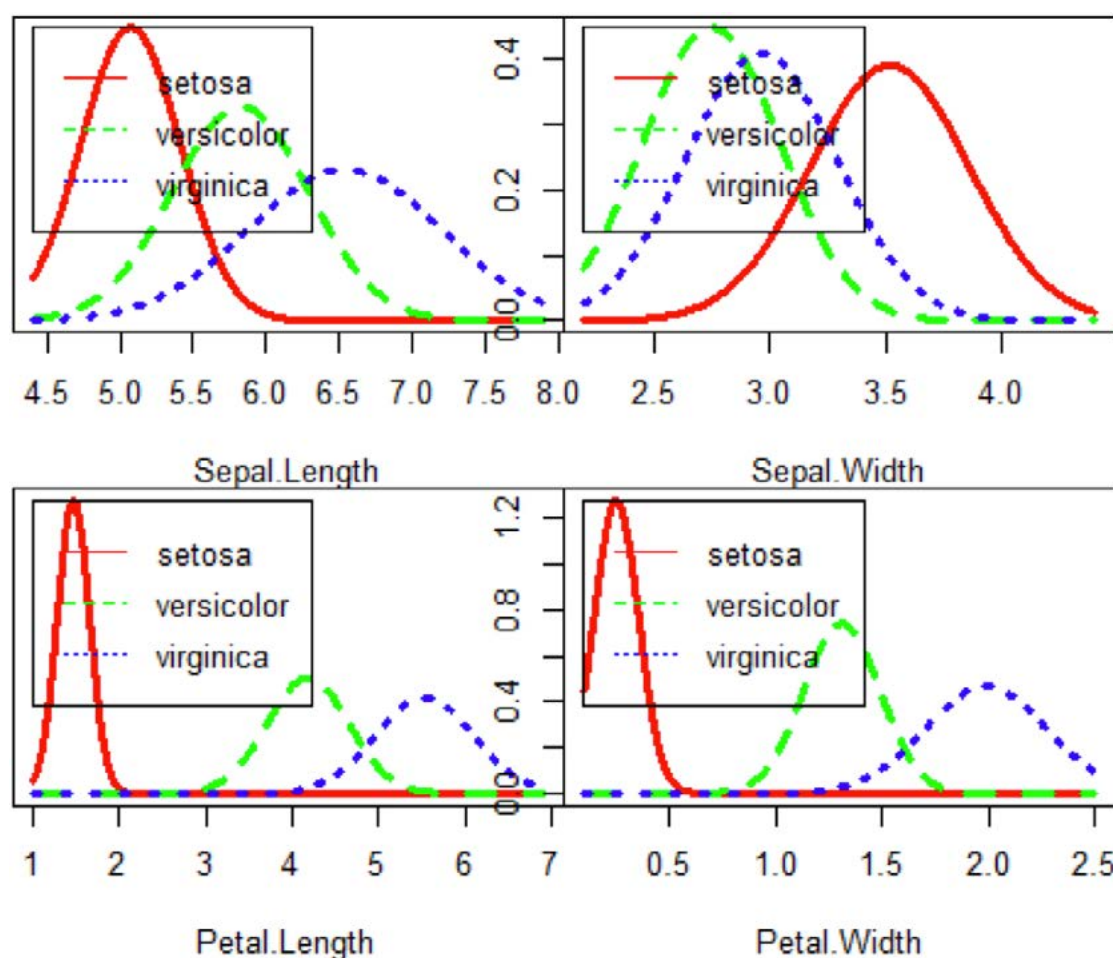
```
1  # Importando os dados
2  training <- read.csv('iris_train.csv')
3  test <- read.csv('iris_test.csv')
4  # Usando klaR para Naive Bayes
5  library(klaR)
6  nb_mod <- NaiveBayes(Species ~ ., data=training)
7  pred <- predict(nb_mod, test)
8  # Matriz de confusão
9  tab <- table(pred$class, test$Species)
10 caret::confusionMatrix(tab)
11
12 # Plotagem da densidade de cada características usando nb_mod
13 opar = par(mfrow=c(2, 2), mar=c(4,0,0,0))
14 plot(nb_mod, main="")
15 par(opar)
16
17 # Plotagem da matriz de confusão
18 library(ggplot2)
19 test$pred <- pred$class
20 ggplot(test, aes(Species, pred, color = Species)) +
21   geom_jitter(width = 0.2, height = 0.1, size=2) +
22   labs(title="Matriz de confusão",
23        subtitle="Predição vs. Observado a partir dos dados Iris",
24        y="Predição",
25        x="Realidade")
```

Fonte: elaborada pelo autor.

O primeiro resultado gerado com o algoritmo da Figura 7 é a densidade de cada espécie de acordo com cada uma das características. O resultado está mostrado na Figura 8. A análise dessa figura mostra

um fator bastante importante no problema de classificação do banco de dados Iris: a classe setosa possui características bastante marcantes e distanciadas das classes versicolor e virginica. Isso pode ser observado pelo fato de que o pico de densidade da classe setosa está, em todos os gráficos, localizado em um ponto afastado das outras classes.

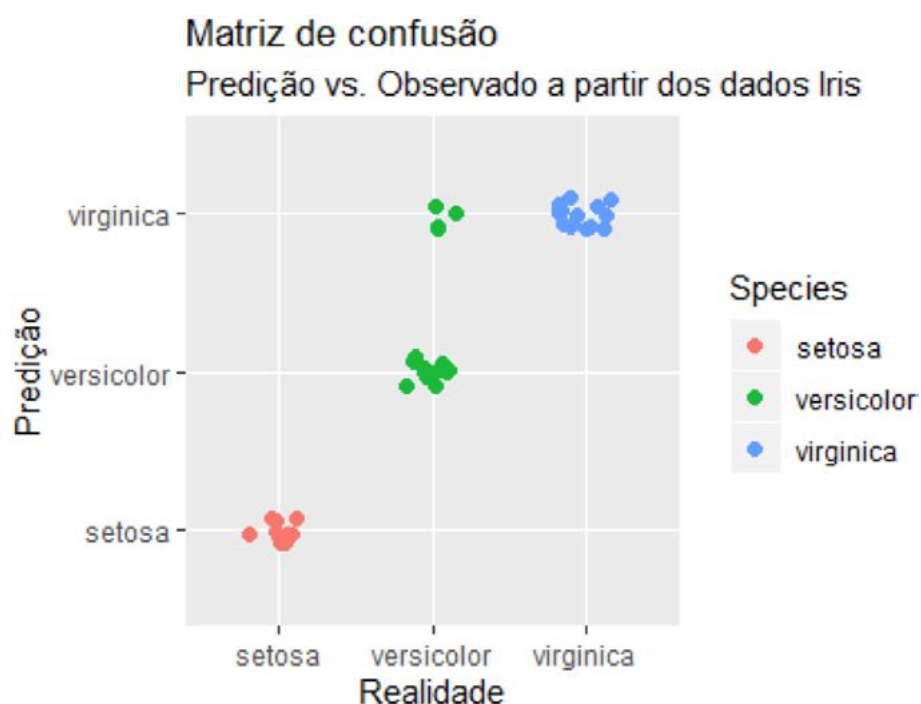
Figura 8 – Densidade das classes de acordo com as características



Fonte: elaborada pelo autor.

O segundo resultado obtido com o algoritmo Naive Bayes da Figura 7 é referente à matriz de confusão do estimador obtido. Essa matriz relaciona os valores obtidos com o estimador durante a etapa de teste com os objetos já rotulados fornecidos para a fase de treinamento. Observe esse resultado, que está contido na Figura 9.

Figura 9 – Relação entre objetos estimados e objetos com rótulo verdadeiro



Fonte: elaborada pelo autor.

A Figura 9 mostra o resultado obtido na etapa de teste para as três espécies de planta. Observe que todos os objetos pertencentes à classe setosa foram corretamente rotulados. Porém, alguns objetos que são da espécie versicolor foram rotulados como virgínica. Isso comprova a característica já vista com a densidade das classes, na Figura 8.

Com o conteúdo das seções 2, 3 e 4, você conseguiu aprender as principais características de três técnicas muito utilizadas na implementação de sistemas baseados em *machine learning*: regressão, classificação kNN e método Naive Bayes. Cada algoritmo possui suas aplicações ideais, sejam para prever, classificar ou outra função de interesse.

Além disso, você aprendeu também que cada técnica possui alguns pontos especiais a serem considerados durante a criação do algoritmo. A regressão exige um estudo de qual modelo regressor será utilizado, para evitar o uso de um grau muito elevado ou uma insuficiência

de estimativa. O algoritmo kNN existe uma atenção especial para a escolha do parâmetro k , que é fundamental para o compromisso entre complexidade e eficácia do algoritmo. O método Naive Bayes exige uma etapa de treinamento bem elaborada, pois é a partir dela que são adquiridas as probabilidades *a priori* e informações necessárias para a implementação do estimador.

TEORIA EM PRÁTICA



O problema de aprendizado de máquina baseado no banco de dados Iris é um exemplo clássico no AM e pode ser utilizado para aplicar diversas técnicas e algoritmos. Primeiro relembre do que se trata esse problema. Ele contém dados de diversas plantas, de três espécies: setosa, versicolor e virgínica. Cada uma das observações, ou objetos, possui medidas de largura e comprimento das pétalas e sépalas e também qual o seu rótulo (a qual classe a observação pertence). A seguir, você pode observar um exemplo de cada espécie dessa planta.

Figura 10 – Plantas setosa, versicolor e virgínica



Fonte: <https://mc.ai/visualization-and-understanding-iris-dataset/>. Acesso em: 14 ago. 2019.

Como você pôde observar na figura anterior, as plantas possuem diferenças em suas dimensões. É assim que os algoritmos de aprendizado de máquina podem, então, classificar plantas não rotuladas.

Durante o estudo, você já viu um algoritmo na linguagem *R* utilizado para resolver esse problema de classificação baseado no método Naive Bayes. Os estudos mostraram que a classe setosa pode ser facilmente distinguida das demais. Porém, as classes virgínica e versicolor possuem algumas observações que são erroneamente classificadas. Essa situação não acontece apenas para o classificador Naive Bayes.

O seu desafio é aplicar a teoria aprendida sobre os métodos estudados e implementar um algoritmo classificador kNN ou regressor para resolver o problema. Para isso, primeiro, escolha uma das técnicas aprendidas e descreva os motivos pelos quais você a escolheu. Para a implementação, escolha o software que mais lhe agrada, pois ambas as técnicas podem ser facilmente implementadas em diversas plataformas. Duas opções recomendadas são Python e *R*. Após implementar corretamente o algoritmo, experimente também mudar seus parâmetros principais. Caso utilize a regressão, mude o grau da função. Se escolher o kNN, mude o valor de k . Fazendo isso, você poderá verificar qual o impacto na qualidade do algoritmo.

VERIFICAÇÃO DE LEITURA



1. A regressão é um método de aprendizado de máquina que pode ser utilizado em diferentes cenários, inclusive em situações multivariáveis em que há correlação entre as variáveis independentes.

Sobre os algoritmos de regressão, assinale a alternativa correta.

- a. O algoritmo de regressão busca criar uma função $\hat{f}(x)$ tal que $\hat{f}(x) < f(x)$ para todo o domínio da função.
 - b. O algoritmo de regressão busca criar uma função $\hat{f}(x)$ tal que $\hat{f}(x) > f(x)$ para todo o domínio da função.
 - c. Os algoritmos de regressão são aplicados somente nos casos em que há uma dependência linear entre as variáveis independentes e dependentes.
 - d. Os algoritmos de regressão são aplicados somente nos casos em que há uma dependência linear entre as variáveis independentes e dependentes.
 - e. O algoritmo de regressão busca criar uma função $\hat{f}(x)$ tal que $\hat{f}(x) \approx f(x)$ para todo o domínio da função.
2. O algoritmo kNN (*k-nearest neighbors*) ou k-vizinhos mais próximos é uma técnica de aprendizado de máquina supervisionado baseado em distância.

Sobre a técnica kNN, encontre a afirmativa correta.

- a. O algoritmo kNN não utiliza etapa de treinamento. Ele apenas observa o dado a ser rotulado sem comparar com nenhum outro dado já observado.
- b. O algoritmo kNN precisa utilizar alguma métrica para calcular a distância entre pontos no domínio da função, como, por exemplo, a distância euclidiana.

- c. O algoritmo kNN é uma técnica de aprendizado de máquina que só pode ser utilizado em problemas com uma variável independente.
 - d. O valor do parâmetro k afeta apenas o peso computacional do algoritmo kNN.
 - e. O algoritmo kNN é um problema binário, sendo capaz de distinguir objetos apenas entre duas classes diferentes.
3. O algoritmo Naive Bayes é uma técnica de aprendizado de máquina que utiliza o Teorema de Bayes para classificar eventos e observações.

Sobre a técnica Naive Bayes, ou Bayes ingênuo, assinale a alternativa correta.

- a. O método Naive Bayes utiliza as probabilidades *a posteriori* das observações para poder, então, calcular qual é a probabilidade *a priori* dos eventos analisados.
- b. O método Naive Bayes é bastante indicado quando se tem informações completas sobre as observações, pois assim é mais fácil encontrar as probabilidades desejadas.
- c. Como o método Naive Bayes é uma técnica probabilística, ela pode utilizar o Teorema de Bayes para obter resultados que estarão sempre corretos, pois estarão baseados nos acontecimentos já ocorridos com os dados de treinamento utilizados.

- d. O algoritmo Naive Bayes é chamado de “Ingênuo”, pois parte do princípio que os dados são todos independentes, o que nem sempre é uma verdade absoluta.
- e. O algoritmo Naive Bayes é uma técnica de aprendizado de máquina descritivo, pois utiliza as probabilidades a priori e condicionais para obter uma descrição detalhada dos dados.

► Referências bibliográficas

- FACELI, K.; LORENA, A. C.; GAMA, J.; CARVALHO, A. C. P. L. F. **Inteligência artificial: uma abordagem de aprendizado de máquina**. São Paulo: LTC Editora, 2011.
- GANDHI, R. **Support vector machine: introduction to machine learning algorithms**. 2018. Disponível em: <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>. Acesso em: 31 jul. 2019.
- GRIVA, I.; NASH, S. G.; SOFER, A. **Linear and nonlinear optimization**. 2. ed. Fairfax: George Mason University, 2009. 742 p.
- PESTANA, D. D.; VELOSA, S. F. **Introdução à probabilidade e à estatística**. 4. ed. Lisboa: Fundação Calouste Gulbenkian, 2008. 1.164 p.
- RASCHKA, S. **Python machine learning**. Birmingham: Packt Publishing, 2015.
- RUSSEL, S.; NORVIG, P. **Artificial Intelligence: a modern approach**. 3. ed. New Jersey: Pearson Education Inc., 2010. 1.132 p.
- SMOLA, A.; VISHWANATHAN, S. V. N. **Introduction to machine learning**. New York, NY: Cambridge University Press, 2008. 226 p.

► Gabarito

Questão 1 – Resposta E

O algoritmo de regressão busca encontrar uma função aproximada $\hat{f}(x)$ que seja o mais próximo possível da função real, ou seja $\hat{f}(x) \approx f(x)$.

Questão 2 – Resposta B

Para o algoritmo kNN conseguir calcular a distância entre dois pontos, seja no espaço bidimensional ou n-dimensional, é preciso utilizar alguma técnica para cálculo de distância, como, por exemplo, a distância euclidiana.

Questão 3 – Resposta D

O método Naive Bayes, assim como outros métodos probabilísticos, é indicado quando não se tem informação completa sobre os dados. Ele possui esse nome, “Bayes Ingênuo”, pois é uma técnica preditiva que parte do princípio que os dados são todos independentes, o que nem sempre é uma verdade absoluta.



Decision tree, Random Forest e método ensemble

Autor: Lucas Claudino

Objetivos

- Estudar os conceitos para criação de árvores de decisão, bem como aprender seu funcionamento e técnicas eficazes para implementação computacional.
- Aprender a técnica de criação de florestas aleatórias, para poder criar estimadores robustos.
- Estudar as características principais dos métodos de agrupamento, que são ferramentas muito importantes para a criação de algoritmos eficazes de aprendizado de máquina.

► 1. Introdução

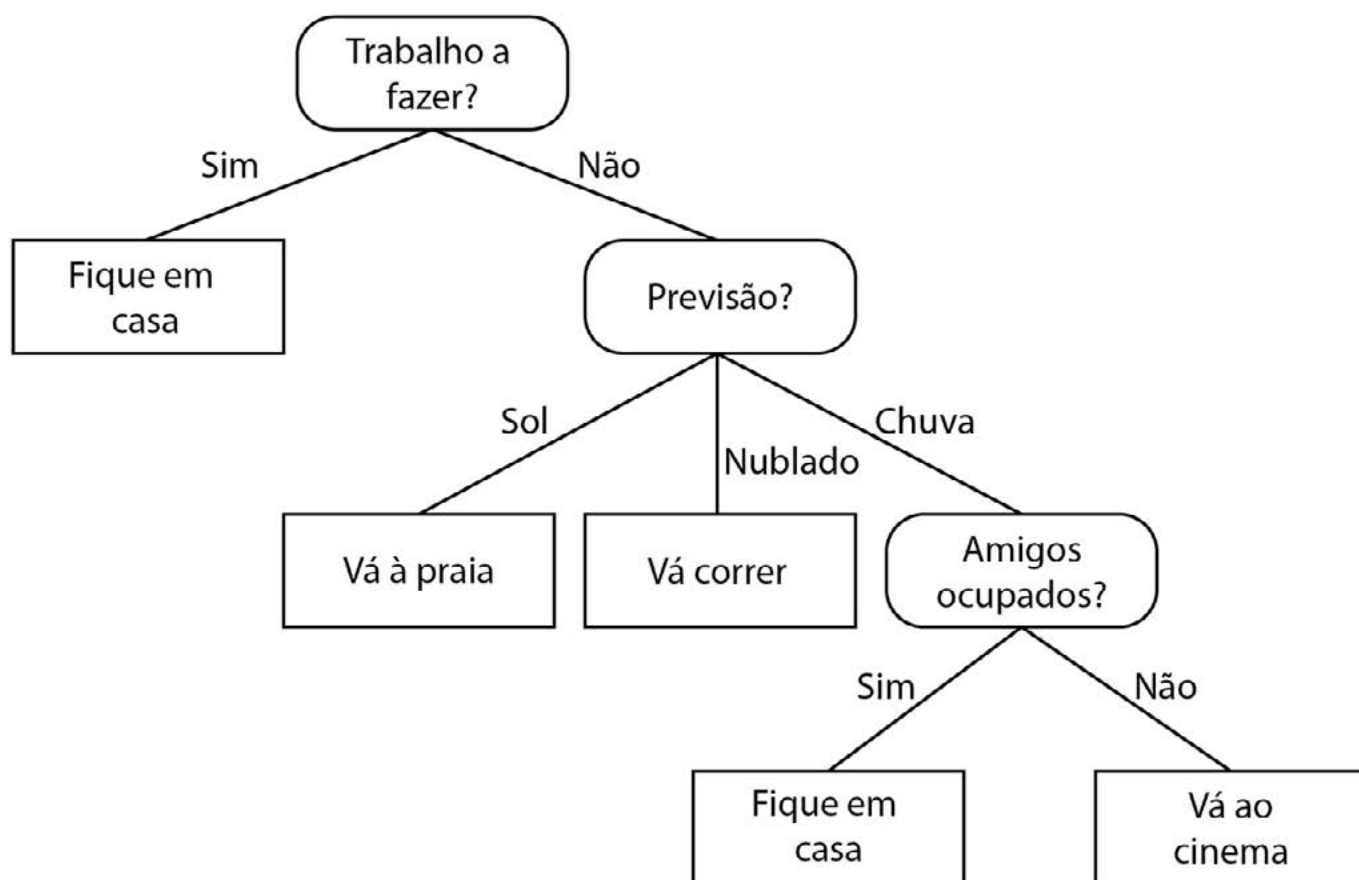
Machine learning é um conteúdo muito rico e importante para o desenvolvimento da tecnologia e análise de dados. Cada situação exige um algoritmo diferente, seja ele supervisionado ou não. Agora você irá estudar três técnicas essenciais: *decision tree*, *Random Forest* e método *ensemble*.

O método *decision tree* busca uma forma de particionar os dados por meio de uma série de perguntas e respostas que levam até um resultado. O algoritmo *Random Forest* pode ser basicamente entendido como uma montagem constituída por diversas árvores de decisão. Já o método *ensemble* é uma constituição feita por meio da combinação de diversas técnicas para produzir um único modelo preditivo tido como ótimo.

► 2. *Decision tree*

O aprendizado por *decision tree*, que em português significa “árvore de decisão”, é um dos métodos mais utilizados para fazer inferência a partir dos dados. É uma técnica capaz de aproximar funções discretas de uma maneira robusta ao ruído dos dados. A função aprendida é então representada por uma árvore de decisão, por isso esse nome dado ao método. A árvore é criada a partir de uma série de perguntas e respostas sobre os dados, que permitem, então, ao algoritmo a elaboração de uma decisão baseada nessa análise (RASCHKA, 2015). Para entender melhor, observe a figura a seguir:

Figura 1 – Estrutura de uma árvore de decisão



Fonte: elaborada pelo autor.

Baseado na estrutura e nas características dos dados, o algoritmo aprende uma árvore de decisão criada a partir de perguntas que ajudam na inferência dos rótulos das amostras. Embora a Figura 1 ilustre um problema de decisão baseado em variáveis categóricas, o mesmo conceito de árvore pode ser aplicado em variáveis numéricas. Por exemplo, caso você queira aplicar a árvore de decisão ao problema Iris, pode fazer uma pergunta baseada no tamanho da sépala: "A sépala é maior do que 2,5 cm?".

Utilizando o algoritmo de árvore de decisão, você deve começar pela raiz do problema e então dividir os dados segundo a característica que resulte no maior ganho de informação (o conceito de ganho de informação será explicado posteriormente). Em um processo iterativo, essa divisão pode acontecer em cada nó filho até que todas as folhas estejam puras. Isso significa que todas as amostras em cada nó

pertencem a uma mesma classe, pois vieram de uma divisão resultante de uma pergunta específica (RASCHKA,2015). Na prática, esse processo pode resultar em árvore com inúmeros nós, o que acaba levando o problema ao *overfitting*. Por isso, geralmente são utilizados métodos de poda que limitam a profundidade máxima da árvore.

Para poder dividir os nós em um ponto que forneça a maior quantidade possível de informação, é necessário definir uma função objetivo a ser otimizada a partir do algoritmo de árvore de decisão. Nesse caso, o objetivo é maximizar o ganho de informação em cada divisão. O ganho de informação (GI) é definido como (RASCHKA, 2015):

$$GI(D_p, f) = I(D_p) - \sum_{j=1}^m \frac{N_j}{N_p} I(D_j)$$

Em que f é o aspecto para proceder a divisão, D_p e D_j são os conjuntos de dados no nó pai e no j -ésimo nó filho, N_p é o número total de amostras no nó pai e N_j o número de amostras no j -ésimo nó filho. Segundo a equação, o GI é apenas a diferença entre a impureza do nó pai e a soma das impurezas dos nós filhos.

Segundo Mitchell (1997), existem vários critérios para medir a impureza dos dados, que é a medida utilizada para fazer a divisão dos dados. Um dos métodos mais utilizados é o critério de entropia $I_e(t)$, para classes não vazias, que é matematicamente definido como:

$$I_E(t) = - \sum_{i=1}^c p(i|t) \log_2 p(i|t)$$

Nesse caso, $p(i|t)$ é a proporção de amostras que pertencem a uma classe i para um dado nó t . A entropia é, então, igual a zero se todas as amostras nesse dado nó pertencem à mesma classe. A entropia é máxima se há uma distribuição uniforme das classes no nó.

A seguir, você verá um pseudoalgoritmo da implementação da técnica ID3, que é um tipo de algoritmo de árvore de decisão. O código é descrito para o caso de classes binárias, ou seja, o domínio X de dimensão d das classes é $X = \{0,1\}^d$. O algoritmo é chamado e recebe o conjunto de treinamento S e o subconjunto de classes $A \subseteq [d]$ e retorna a árvore de decisão (SHALEV-SHWARTZ; BEN-DAVID, 2014).

Figura 2 – Pseudocódigo de um algoritmo do tipo árvore de decisão

Algorithm 1 Árvore de decisão

Data: conjunto de treinamento S , subconjunto de classes $A \subseteq [d]$

Result: Árvore de decisão

if todas amostras em S são rotuladas como 1 **then**

 | **return** folha 1

end

if todas amostras em S são rotuladas como 0 **then**

 | **return** folha 0

end

if $A = \emptyset$ **then**

 | **return** folha cujo valor = maioria dos rótulos de S

else

 Faça $j = \operatorname{argmax}_{i \in A} \{I_E(S, i)\}$

if todos exemplos em S tem mesmo rótulo **then**

 | **return** folha cujo valor = maioria dos rótulos de S

else

 Faça T_1 a árvore retornada por ID3 ($\{(\mathbf{x}, y) \in S : x_j = 1\}, A/\{j\}$)

 Faça T_2 a árvore retornada por ID3 ($\{(\mathbf{x}, y) \in S : x_j = 0\}, A/\{j\}$)

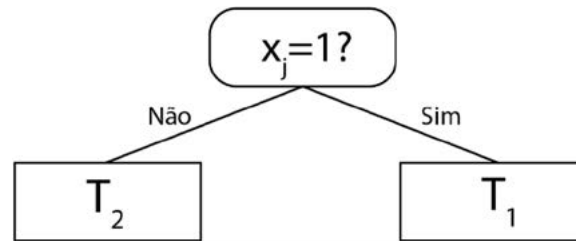
end

end

Fonte: elaborada pelo autor.

A Figura 2 contém um pseudocódigo que tem como objetivo a classificação de um conjunto de objetos. O algoritmo, primeiro, faz uma série de testes para verificar se o conjunto de dados não está vazio ou possui somente uma classe de objetos. Uma vez passados esses testes, o algoritmo classifica os objetos de forma binária, que são as classificações possíveis. A figura a seguir ilustra de forma gráfica como seria o comportamento desse algoritmo.

Figura 3 – Árvore de decisão referente ao algoritmo



Fonte: elaborada pelo autor.

O algoritmo ID3 descrito anteriormente ainda sofre de um grande problema: a árvore criada, geralmente, é muito grande ou profunda. Essas árvores possuem um baixo risco empírico, porém o risco real pode ser grande, uma vez que o tempo computacional pode ser elevado. Uma possível solução é limitar o número de iterações a serem realizadas, resultando em uma árvore com menor número de nós. Outra solução comum é o uso de técnicas de poda. Essas técnicas são aplicadas depois que a árvore já está pronta, buscando reduzir sua dimensão a uma árvore muito menor.

Essa poda é, normalmente, realizada de baixo para cima. Cada nó pode ser substituído por uma de suas subárvores ou folhas. Essa substituição é feita com base em uma estimativa de erro realizada para cada nó a ser substituído. Observe o pseudocódigo a seguir, que mostra o esquema de implementação da poda em uma árvore de decisão (SHALEV-SHWARTZ; BEN-DAVID, 2014).

Figura 4 – Pseudocódigo de poda de árvore de decisão

Algorithm 2 Poda em Árvore de decisão

Data: função $f(T, m)$ (estimativa para o erro de generalização, baseado no tamanho m da amostra)

Result: Substituição ou não do nó

```
for nó  $j$  em um caminho de baixo para cima (das folhas à raiz) do
    encontre  $T'$  que minimze  $f(T, m)$ , aonde  $T'$  é qualquer um dos:
        a árvore atual após substituir o nó  $j$  por uma folha 1
        a árvore atual após substituir o nó  $j$  por uma folha 0
        a árvore atual após substituir o nó  $j$  por sua sub-árvore esquerda
        a árvore atual após substituir o nó  $j$  por sua sub-árvore direita
```

end

Fonte: elaborada pelo autor.

► 3. *Random Forest*

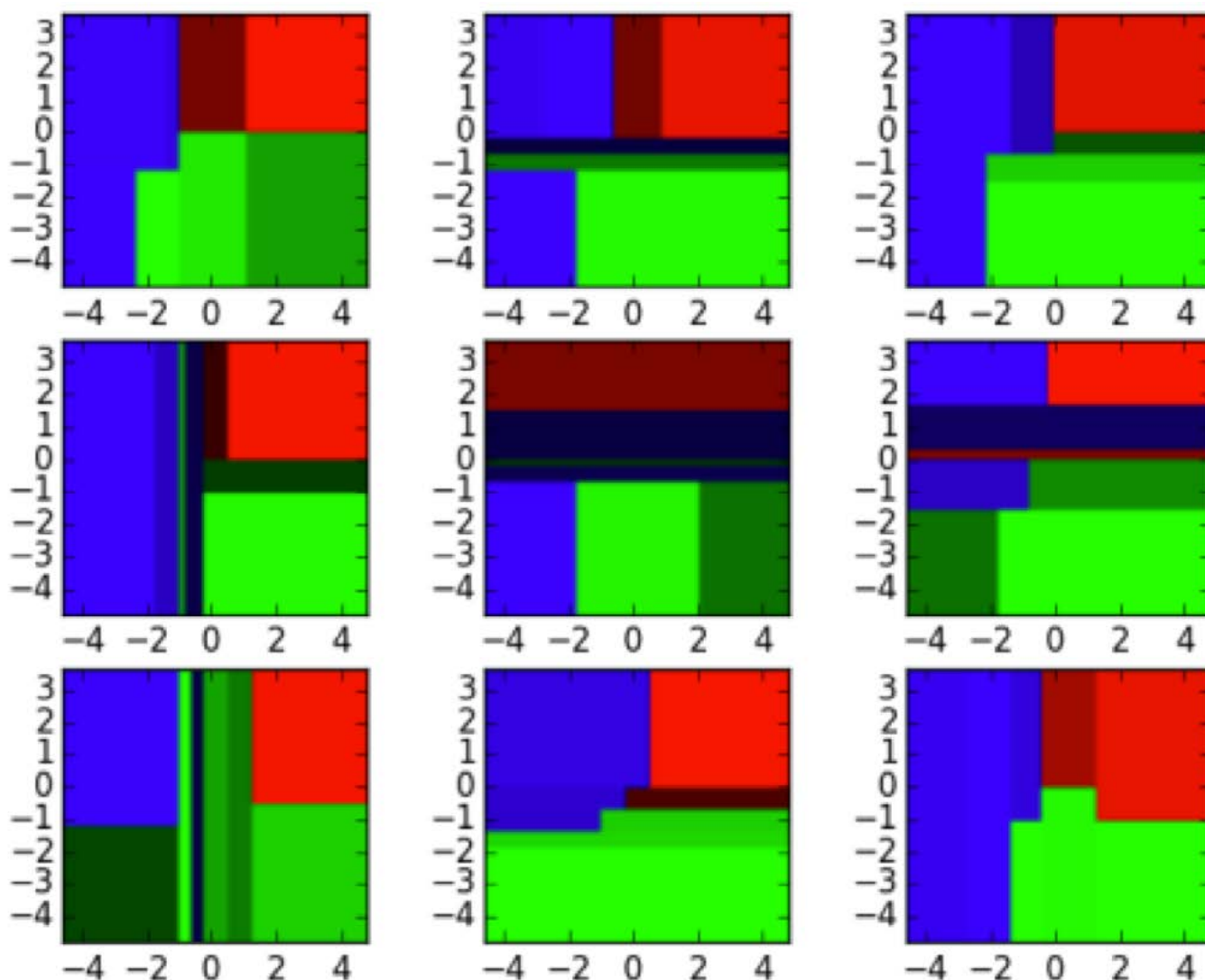
A técnica *Random Forest*, ou floresta aleatória, obteve uma grande popularidade em aplicações de aprendizado de máquina devido à sua boa performance de classificação, escalabilidade e facilidade de uso. A ideia por trás de uma floresta aleatória é combinar diversos estimadores fracos para construir um modelo robusto com menor erro de generalização e é menos susceptível a *overfitting* (RASCHKA, 2015).

Modelos do tipo *ensemble* são compostos de muitos outros modelos, e geralmente constituem uma estratégia de competição. Uma única árvore de decisão pode tomar várias decisões erradas, uma vez que possui julgamentos binários. O método *Random Forest* é um metaestimador que agrega diversas árvores de decisão. Porém, adiciona algumas pequenas, mas fundamentais, modificações (MAINI, 2017):

1. A quantidade de características que podem ser divididas em cada nó é limitada a um determinado percentual do total de características.
2. Cada árvore extrai uma amostra aleatória dos dados originais ao gerar uma divisão. Isso agrega uma aleatoriedade extra ao algoritmo e previne o *overfitting*.

Para ilustrar a combinação feita com o algoritmo *Random Forest*, observe a Figura 5, que contém nove classificadores do tipo árvore de decisão. Observe que cada uma das árvores cria um classificador diferente, mesmo tendo sido criadas a partir do mesmo banco de dados.

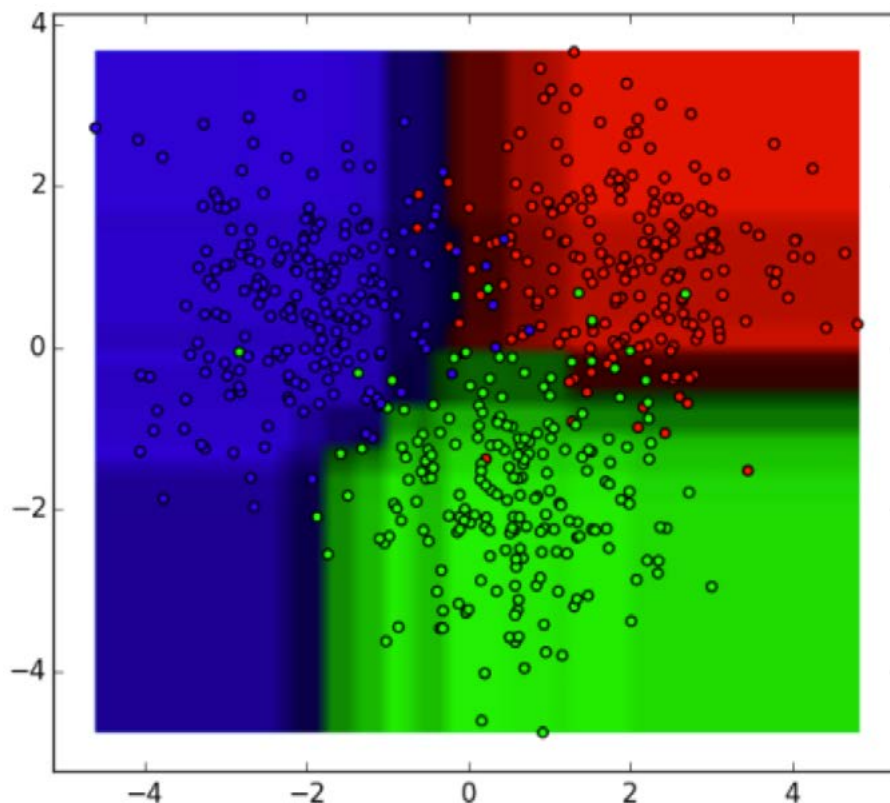
Figura 5 – Árvores de decisão



Fonte: Maini (2017, p. 53).

Como dito anteriormente, o algoritmo *Random Forest* faz a utilização de diversas árvores de decisão para criar um estimador robusto e sem viés. As árvores da Figura 5 podem ser agrupadas em uma floresta aleatória que combina todas as entradas, com o objetivo de obter uma resposta mais precisa do que cada uma das árvores individualmente. Utilizando a estratégia do maior voto a partir das nove árvores, é possível verificar qual característica teve maior voto, criando, então, um classificador que divide os dados entre “verde”, “azul” e “vermelho”. Observe a Figura 6, que é o resultado do *ensemble* com as nove árvores de decisão da Figura 5.

Figura 6 – Resultado da combinação das nove árvores de decisão



Fonte: Maini (2017, p. 53).

A implementação do algoritmo *Random Forest* pode ser feita da seguinte maneira:

Figura 7 – Pseudocódigo para implementação do algoritmo *Random Forest*

Algorithm 3 Implementação de floresta aleatória

Data: conjunto de dados S

Result: classificador *random forest*

for $j = 1$ até k **do**

 remova aleatoriamente n amostras

 crie uma árvore aleatória a partir das n amostras

 em cada nó:

for cada nó **do**

 selecione aleatoriamente d características, sem reposição

 divida o nó da maneira que maximize o ganho de informação

end

end

Agrupe as predições de cada árvore para associar um rótulo (ou classe) de acordo com o voto da maioria

Fonte: elaborada pelo autor.

ASSIMILE

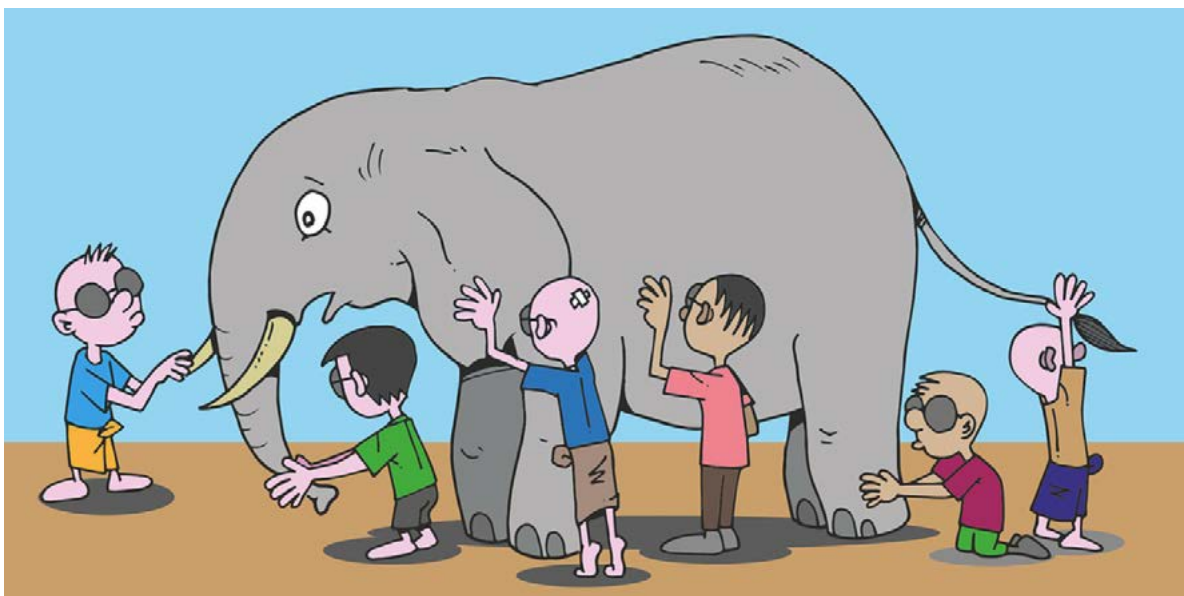
Na prática, o único parâmetro que precisa de cuidado especial é a quantidade k de árvores utilizadas para a implementação do algoritmo. Normalmente, quanto mais árvores utilizadas, melhor é o desempenho da floresta aleatória; porém, grandes valores de k trazem um elevado custo computacional (RASCHKA, 2015).

► 4. Método *ensemble*

Método *ensemble*, também chamado de método de agrupamento, é um modelo que, em *machine learning*, combina decisões de múltiplos modelos para poder melhorar a performance geral do sistema. Esses tipos de AM tentam lidar, e mitigar, as três maiores causas de erros: ruído, viés e variância.

Para você entender como esse agrupamento pode ajudar a mitigar esses erros, observe atentamente a figura a seguir.

Figura 8 – Observações individuais e enviesadas



Fonte: <https://towardsdatascience.com/simple-guide-for-ensemble-learning-methods-d87cc68705a2>. Acesso em: 27 ago. 2019.

A Figura 8 contém seis crianças com os olhos tapados brincando do jogo “Pegue e Fale”. Elas precisam tocar o objeto totalmente desconhecido, que neste caso é um elefante, e então descrevê-lo a partir do toque. Cada criança terá uma versão diferente do elefante, pois estão enviesadas pela posição na qual se encontram. Ao final, a tarefa é submeter um relatório descrevendo o que sentiram. Os relatórios individuais serão muito diferentes, porém, o agrupamento de todos poderá fornecer uma descrição detalhada e precisa da aparência do elefante.

O método *ensemble* funciona de maneira similar, pois combina resultados vindos de vários modelos para fornecer uma resposta que, na maioria das vezes, fornece uma predição mais precisa do que todos os resultados individuais.

Existem técnicas de agrupamento que são bastante simples, e você, provavelmente, já é familiarizado com elas, mas não no conceito de *machine learning*. Essas técnicas são: moda e média.

Há também técnicas avançadas de *ensemble*, que são *Bagging*, *Boosting* e *Random Forest*. A seguir, você estudará tanto as técnicas básicas como as avançadas.

ASSIMILE



O método *Random Forest* é também um tipo de *ensemble*, pois utiliza um conjunto de árvores de decisão para criar um estimador que leve em consideração os votos individuais fornecidos por cada uma das árvores.

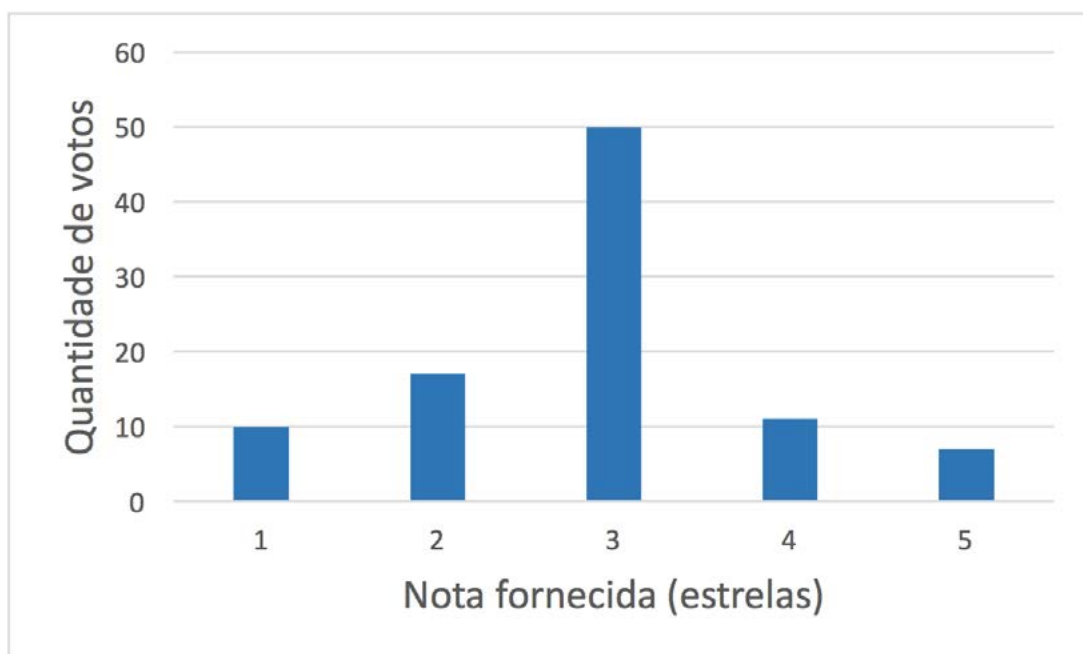
4.1 Aplicando a moda aos resultados

Moda é uma medida estatística que se refere à ocorrência mais frequente em um determinado conjunto de dados (BUSSAB; MORETTIN, 2017). O aprendizado de máquina, ao aplicar a técnica de avaliar a

moda dos resultados, utiliza vários modelos para fazer previsões sobre os dados. Cada previsão individual é considerada como um voto separado, e aquela opinião que for a mais votada será utilizada como classificação final.

Para melhor entender o funcionamento dessa técnica, observe o gráfico a seguir, que contém a avaliação feita por 95 usuários sobre um determinado aplicativo de celular. Cada usuário forneceu uma nota de 1 a 5 estrelas para o aplicativo.

Figura 9 – Avaliações feitas para um determinado aplicativo de celular



Fonte: elaborada pelo autor.

Neste caso, um algoritmo *ensemble* baseado na moda dos dados iria apenas verificar que 50 pessoas votaram com nota “3 estrelas” e então classificaria o aplicativo com esse mesmo rótulo.

4.2 Aplicando a média aos resultados

Esta técnica vai também utilizar o resultado de cada um dos modelos. Porém, utiliza a média aritmética para decidir qual classificação obteve resultado mais significativo.

Considerando o mesmo exemplo utilizado na Figura 32, a média dos votos seria obtida pela seguinte equação:

$$\text{Média} = \frac{\sum(\text{nota} \times \text{votos})}{\text{total de votos}} = \frac{1 \times 10 + 2 \times 17 + 3 \times 50 + 4 \times 11 + 5 \times 7}{95} = 2,87$$

Como a nota não pode ser exatamente igual a 2,87 estrelas, é preciso aproximar para o valor mais próximo contido no domínio da função. Portanto, a média dos resultados será igual a 3. Com isso, o algoritmo pode finalmente fornecer a classificação do aplicativo como “3 estrelas”.

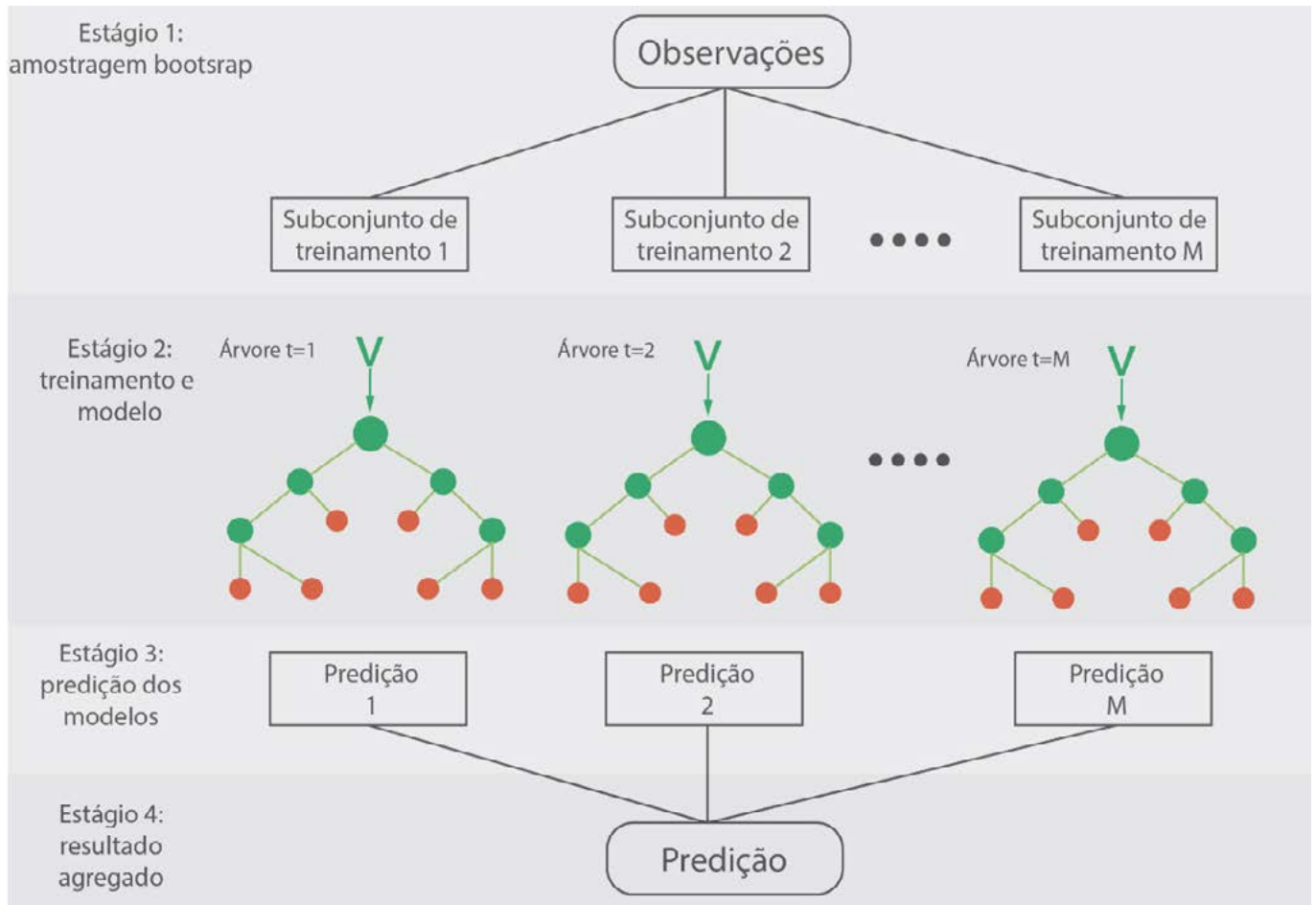
► 5. *Bagging*

O método de *Bagging* é que combina técnicas de *bootstrap* e *aggregating*, por isso recebe esse nome. Primeiramente, uma amostra aleatória é tirada do conjunto de treinamento. Depois, um modelo (classificador ou árvore de decisão) é criado para cada conjunto de amostras retiradas. Finalmente, os resultados desses múltiplos modelos são combinados utilizando técnicas como média ou maioria de votos.

Cada modelo é exposto a diferentes subconjuntos de dados, e o algoritmo usa a saída coletiva ao final do processo. Isso garante que o problema de *overfitting* não ocorra, pois os subconjuntos de treinamentos são retirados aleatoriamente (*bootstrap*). Esse processo também ajuda a reduzir o erro de variância.

Para melhor entender como funciona o método *ensemble* do tipo *Bagging*, analise a Figura 10, que contém as principais etapas do método. Observe que a primeira etapa é o *bootstrapping*, em que os subconjuntos de amostras são aleatoriamente retirados do total de observações. Após isso, ocorre a criação de um modelo para cada subconjunto, aqui ilustrado por uma árvore de decisão. O terceiro estágio é a predição individual de cada um dos modelos. Finalmente, no estágio 4, essas predições são agrupadas de alguma forma.

Figura 10 – Estrutura de um modelo *Bagging*



Fonte: elaborada pelo autor.

ASSIMILE

Combinação de múltiplos modelos ajuda a diminuir a variância dos resultados, especialmente no caso de modelos instáveis, e também pode produzir uma predição mais confiável do que um modelo único.

TEORIA EM PRÁTICA

A teoria do aprendizado de máquina é muito ampla e mostra que todo problema pode ser resolvido de diversas maneiras. Nesta Leitura Fundamental, você estudou três

técnicas muito importantes para o desenvolvimento do *machine learning*: árvores de decisão, *Random Forest* e método *ensemble*. Você já estudou diversos problemas clássicos utilizados para o aprendizado de técnicas *machine learning*, como, por exemplo, o problema Iris.

A sua tarefa agora é aplicar a teoria aprendida sobre os métodos estudados nesta Leitura Fundamental para analisar mais um problema clássico utilizado no ensino de aprendizado de máquina: as medidas da qualidade do ar da cidade de Nova York. Nele você verá como a qualidade do ar e a presença do gás ozônio estão relacionados a diversos fatores, como, por exemplo, a temperatura média da cidade.

Por se tratar de um problema clássico, seus dados já estão disponíveis nos principais softwares de simulação utilizados até agora. Ao utilizar o programa “R”, basta executar o comando `require(data.table)` e ler os dados com o comando `bagging_data=data.table(airquality)`. Assim você já terá todos os dados com as seguintes variáveis:

- Ozone: média de ozônio em partes por bilhão.
- Solar.R: radiação solar, em Langleys.
- Wind: velocidade média do vento, em milhas por hora.
- Temp: temperatura diária máxima, em graus Fahrenheit.

O seu trabalho agora é utilizar esse banco de dados disponível e pesquisar como fazer a implementação do método *ensemble* com árvore de decisão para relacionar a quantidade de ozônio e a velocidade do vento.

Se você utilizar a linguagem “R”, terá várias funções para auxiliá-lo nessa grande tarefa, basta pesquisar um pouco e conseguirá encontrar inúmeros exemplos.

Ao implementar o método *ensemble Bagging*, preste atenção em alguns pontos importantes:

- A relação entre porcentagem dos dados que está sendo utilizada para o treinamento e para o teste.
- A quantidade de árvores de decisão a serem utilizadas para o agrupamento.

Após implementar corretamente, verifique como esses dois fatores alteram o resultado final. Você verá que a porcentagem de dados utilizada para teste/treinamento é muito importante. Além disso, pode observar também que a quantidade de árvores influencia muito na complexidade computacional e na acurácia do resultado, mas pode levar a um *overfitting* dos dados.

VERIFICAÇÃO DE LEITURA



1. A árvore de decisão é um método clássico de *machine learning*, que pode ser utilizado individualmente ou até agrupado, formando *ensembles* ou florestas aleatórias.

Sobre os algoritmos do tipo árvore de decisão, assinale a alternativa correta.

- a. É uma técnica criada especificamente para aproximar funções numéricas contínuas.
- b. O resultado do algoritmo é uma árvore criada a partir de uma série de perguntas e respostas sobre os dados.

- c. Baseado na estrutura da árvore fornecida para o treinamento, o algoritmo cria respostas para resolver a árvore sem decisões tomadas.
 - d. O algoritmo deve começar pela raiz do problema e então dividir os dados segundo a característica que resulte no menor ganho de informação.
 - e. O algoritmo deve começar pelas folhas, ir em direção à raiz do problema e então dividir os dados segundo a característica que resulte no maior ganho de informação.
2. O método *Random Forest*, ou árvore aleatória, é uma técnica de aprendizado de máquina capaz de trazer um grande ganho de performance e qualidade para modelos preditivos e classificativos.
- Sobre essa técnica de aprendizado de máquina, assinale a alternativa correta.
- a. A quantidade de características que podem ser divididas em cada nó é limitada a um determinado percentual do total de características.
 - b. Cada árvore extrai a mesma amostra dos dados originais ao gerar uma divisão.
 - c. O algoritmo *Random Forest* faz a utilização de diversas árvores de decisão para criar um estimador robusto e enviesado.
 - d. A quantidade de árvores criadas não influencia na qualidade do resultado.
 - e. A quantidade de árvores criadas, apesar de influenciar na qualidade do resultado, não afeta o desempenho computacional.

3. Sobre os métodos *ensemble*, quando aplicados ao aprendizado de máquina, assinale a alternativa correta.
- a. Esse tipo de AM pode lidar com ruído e viés nos dados, porém não consegue mitigar a variância.
 - b. Esse tipo de AM pode lidar com ruído e variância nos dados, porém não consegue mitigar o viés.
 - c. O método *Random Forest*, apesar de ser uma combinação de árvores de decisão, não pode ser considerado um método *ensemble*.
 - d. No método *Bagging*, cada submodelo é exposto a diferentes subconjuntos de dados extraídos do conjunto principal.
 - e. Esse tipo de AM pode lidar com variância e viés nos dados, porém não consegue mitigar o ruído.

Referências bibliográficas

BUSSAB, W. O.; MORETTIN, P. A. **Estatística básica**. 9. ed. São Paulo: Saraiva, 2017. 568 p.

FACELI, K.; LORENA, A. C.; GAMA, J.; CARVALHO, A. C. P. L. F. **Inteligência artificial: uma abordagem de aprendizado de máquina**. São Paulo: LTC Editora, 2011.

MAINI, V.; SABRI, S. **Machine learning for humans**. 2017. Disponível em: <https://everythingcomputerscience.com/books/Machine%20Learning%20for%20Humans.pdf>. Acesso em: 26 ago. 2019.

MITCHEL, M. B. **Machine learning**. New York: McGraw-Hill Science/Engineering/Math, 1997. 432 p.

RAMZAI, J. **Simple guide for ensemble learning methods**. 2019. Disponível em: <https://towardsdatascience.com/simple-guide-for-ensemble-learning-methods-d87cc68705a2>. Acesso em: 27 ago. 2019.

RASCHKA, S. **Python machine learning**. Birmingham: Packt Publishing, 2015.

RUSSEL, S.; NORVIG, P. **Artificial Intelligence: a modern approach**. 3. ed. New Jersey: Pearson Education Inc., 2010. 1.132 p.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. **Understanding machine learning: from theory to algorithms**. New York: Cambridge University Press, 2014.

SMOLA, A.; VISHWANATHAN, S. V. N. **Introduction to machine learning**. New York, NY: Cambridge University Press, 2008. 226 p.

Gabarito

Questão 1 – Resposta B

O algoritmo de árvore de decisão é uma técnica capaz de aproximar funções discretas de uma maneira robusta ao ruído dos dados. Você deve começar a rastrear o problema pela raiz e então dividir os dados segundo a característica que resulte no maior ganho de informação.

Questão 2 – Resposta A

Resolução:

- Cada árvore extrai uma amostra aleatória dos dados originais ao gerar uma divisão.
- O algoritmo *Random Forest* faz a utilização de diversas árvores de decisão para criar um estimador robusto e sem viés.
- A quantidade de árvores criadas influencia na qualidade do resultado e no desempenho computacional.

Questão 3 – Resposta D

Esse tipo de AM pode lidar com ruído, viés e variância nos dados. O método *Random Forest* é uma combinação de árvores de decisão e pode ser considerado um método *ensemble*.



***Clustering, support vector machines* e processamento em linguagem natural**

Autor: Lucas Claudino

Objetivos

- Analisar as principais características dos algoritmos de *clustering*, verificando as melhores metodologias de implementação e cenários de aplicação.
- Estudar a metodologia de aprendizado de máquina baseado em máquinas de vetor de suporte e suas técnicas de implementação.
- Conhecer os métodos de processamento em linguagem natural e como os algoritmos podem reconhecer padrões em linguagens naturais.

► 1. Introdução

O aprendizado de máquina é uma área muito ampla e capaz de ser aplicada a inúmeros problemas. Duas aplicações bastante comuns são a mineração de dados, em que o algoritmo explora uma grande quantidade de dados para buscar padrões consistentes, e a interação entre homem e máquina. No cenário de mineração de dados, é muito comum a utilização de algoritmos de agrupamento (*clustering*) e também máquinas de vetor de suporte (SVM, *support vector machine*). Para aplicações de interação entre homem e máquina, normalmente utilizam-se algoritmos de processamento de linguagem natural, como, por exemplo, o N-gram.

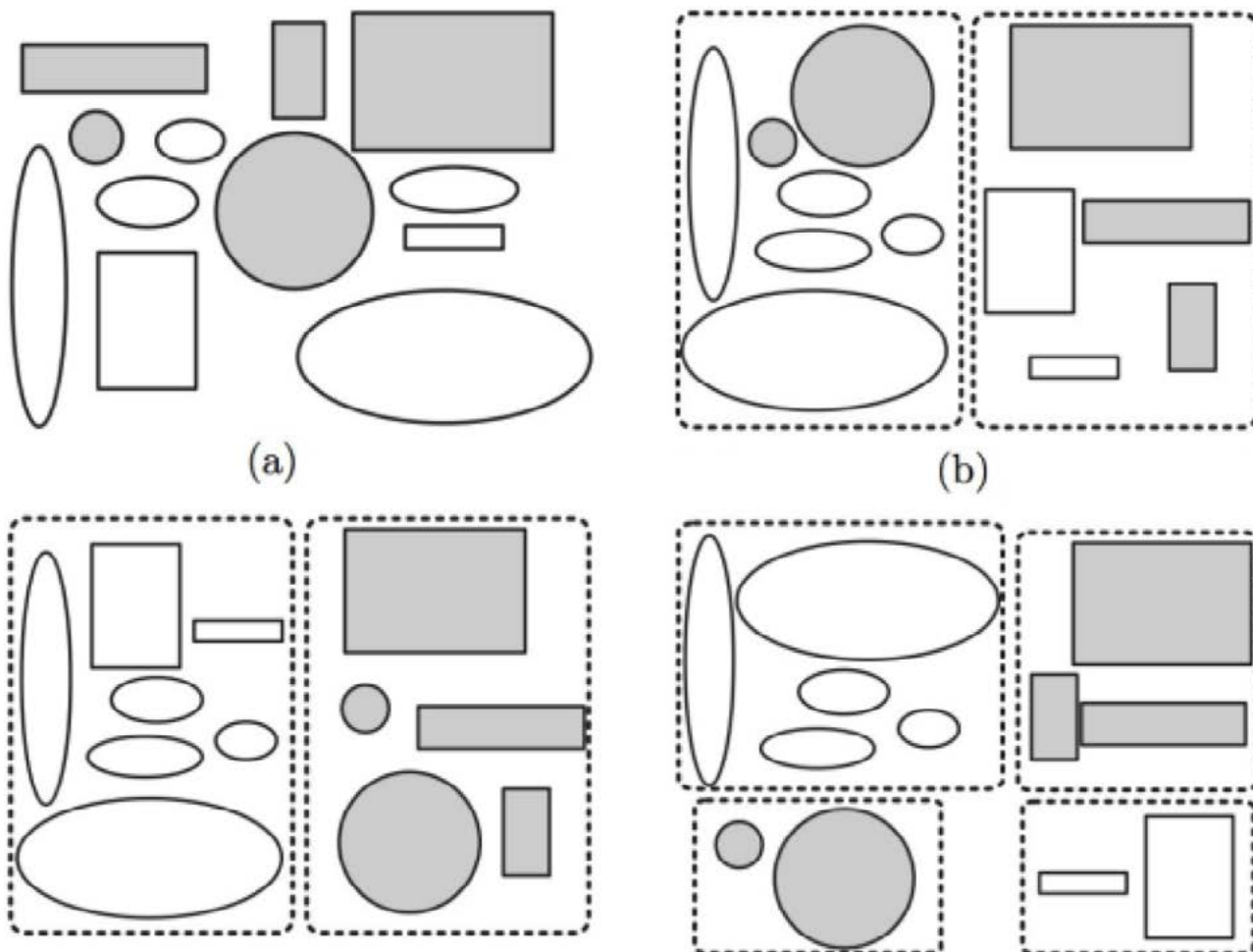
Tendo em vista essas aplicações principais e inúmeras outras que necessitam do auxílio de técnicas de processamento computacional e aprendizado de máquina, esta Leitura Fundamental irá tratar sobre três metodologias fundamentais para o *machine learning*: técnicas de *clustering*, *support vector machines* e processamento em linguagem natural.

► 2. Clustering

As técnicas de aprendizado de máquina do tipo *clustering* buscam encontrar *clusters* (grupos) de dados que possuem algum tipo de característica similar ou comum. Para a grande maioria dos conjuntos de dados, não existe somente uma forma de realizar o agrupamento e nem com um número específico de *clusters*.

Para melhor entender esse conceito de agrupamento, observe a Figura 1, que contém quatro possíveis divisões para o mesmo banco de dados.

Figura 1 – (a) Domínio de objetos; (b) objetos agrupados pela forma (2 *clusters*); (c) objetos agrupados pelo preenchimento (2 *clusters*); (d) objetos agrupados pelo preenchimento e pela forma (4 *clusters*)



Fonte: adaptada de Faceli (2011, p. 191).

A Figura 1a contém uma série de objetos, cada um com uma característica específica. Observando a Figura 1b, você consegue verificar que é possível organizar esses objetos de acordo com as suas formas, e na Figura 1c, eles são organizados pelos seus preenchimentos. Dessas duas maneiras, foram feitos agrupamentos de duas maneiras diferentes, por isso você pode falar que eles foram divididos em **dois clusters**. Porém, pode-se aumentar o número de divisões, assim como ilustra a Figura 1d.

Segundo Faceli (2011), um cluster pode ser considerado como sendo uma coleção ou um agrupamento de objetos próximos. Para o algoritmo criar algum tipo de cluster, ele precisa seguir alguns critérios (HANDL *et al.*, 2005):

- Compactação: também conhecida como homogeneidade, e está relacionada à variação intracluster. Os algoritmos de AM que priorizam ou otimizam esse critério, geralmente são bem aplicados para descobrir algum tipo de cluster esférico e/ou bem separado.
- Encadeamento ou ligação: este conceito está relacionado à ideia de que objetos vizinhos compartilham o mesmo cluster. Algoritmos baseados neste critério são indicados para a organização de *clusters* de forma arbitrária, porém não possui bom desempenho na identificação de casos em que *clusters* diferentes estão muito próximos.
- Separação espacial: este critério leva em conta a separação entre os *clusters*. Sendo assim, como não analisa profundamente os objetos dentro do *clusters*, este método pode resultar em uma solução trivial. Por isso, este critério é geralmente aplicado somente quando associado a outros.

A seguir, você estudará alguns dos principais algoritmos de agrupamento e eles estarão divididos em três classes: algoritmos baseados em densidade, algoritmos particionais baseados em erro quadrático e algoritmos hierárquicos. Porém, as classificações dos algoritmos de agrupamento não estão limitadas a essas três citadas, e um mesmo algoritmo pode ser classificado de diversas maneiras diferentes.

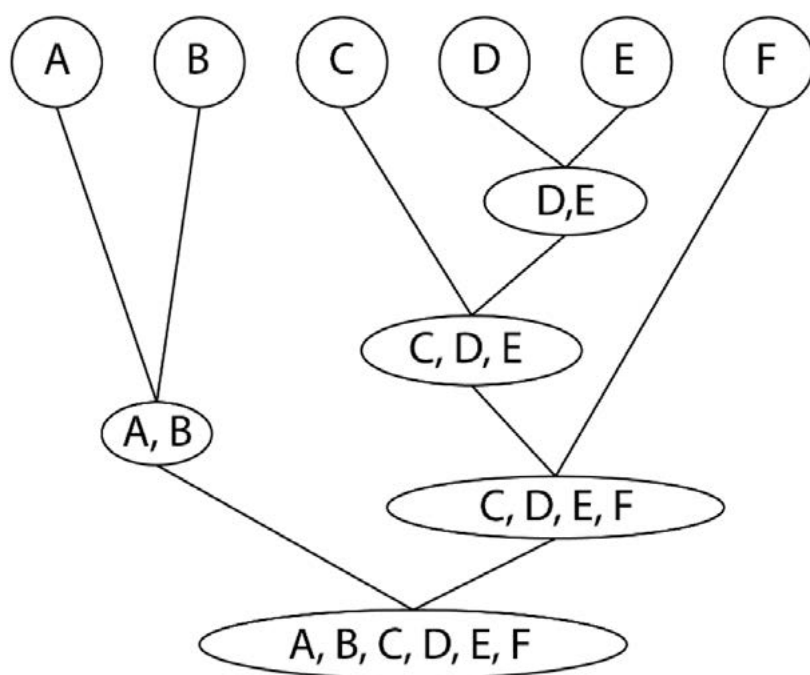
2.1 Algoritmos hierárquicos

Clustering do tipo hierárquico pode ainda ser subdividido de duas maneiras: aglomerativo e divisivo. Na técnica de **agrupamento hierárquico aglomerativo**, cada dado é inicialmente considerado como sendo um cluster individual. A cada iteração, os *clusters* similares são então **aglomerados** até que sobre somente um *cluster*. O método de trabalho desse tipo de algoritmo é bastante simples (REDDY, 2018):

- Calcule a matriz de proximidade.
- Considere cada ponto como um *cluster*.
- Repita: aglomere dois *clusters* e calcule novamente a matriz de proximidade.
- Faça até sobrar somente 1 *cluster*.

Observe a Figura 2 e depois analise os possíveis passos que o algoritmo *clustering* hierárquico aglomerativo realizaria para dividir os dados.

Figura 2 – Agrupamento do tipo aglomerativo



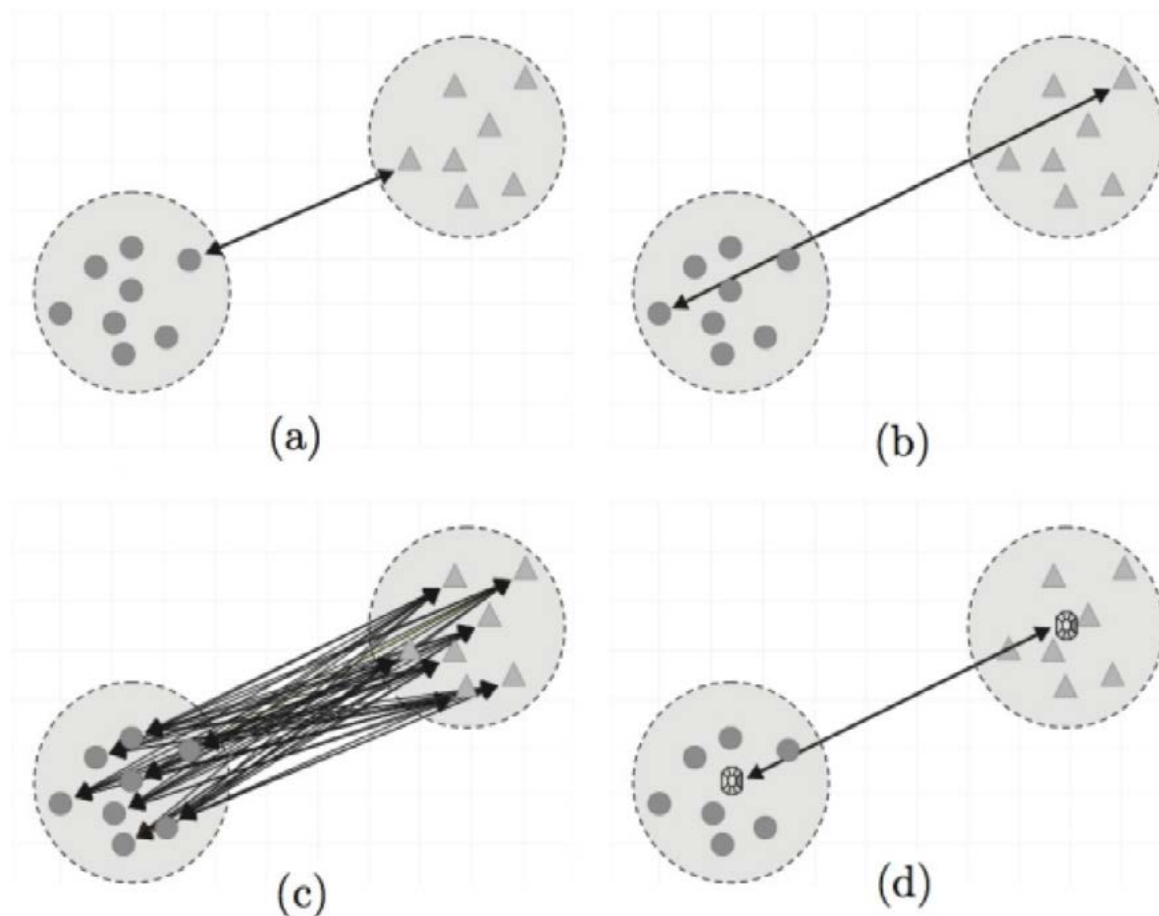
Fonte: elaborada pelo autor.

- **Passo 1:** calcula-se a proximidade dos pontos individuais, considerando-os então como sendo 6 *clusters*.
- **Passo 2:** *clusters* similares são agrupados. Neste caso, foram formados os grupos "D, E" e "A, B".
- **Passo 3:** calcula-se de novo a matriz de proximidades e agrupa os elementos similares. Isso resulta no agrupamento que possui os seguintes *clusters*: "C, D, E", "A, B" e "F".

- **Passo 4:** calcula-se novamente as distâncias e agrupam-se os *clusters* similares. Com isso, agrupa-se o cluster “F” ao “C, D, E”.
- **Passo 5:** finalmente, após o cálculo e agrupamento, sobra somente um cluster composto pelos elementos “A, B, C, D, E, F”.

A similaridade entre *clusters* pode ser então calculada de algumas maneiras diferentes: distância mínima, máxima, distância média ou distância entre centroides. Cada uma dessas distâncias é mais indicada para um tipo de dados, e é conveniente ter uma ideia de qual será a possível organização dos objetos para então escolher o cálculo de distâncias mais oportuno para a sua situação. Observe a Figura 3, que contém uma ilustração das quatro técnicas de medida de similaridades citadas acima.

Figura 3 – Distâncias (a) mínima; (b) máxima; (c) média; (d) entre centroides



Fonte: adaptada de Faceli (2011, p. 211).

Na prática, a técnica de **agrupamento hierárquico divisivo** não é muito utilizada, mas é conveniente explicá-la mesmo assim. Basicamente, a técnica divisiva faz o processo oposto à aglomerativa. Os dados são inicialmente considerados como todos pertencentes a um grande cluster, e o algoritmo deve, então, a cada iteração, separar os objetos que não possuem características similares ao cluster no qual estão inseridos. Ao final do processo, restarão apenas K clusters.

► 3. Algoritmos particionais baseados em erro quadrático

Esse tipo de algoritmo parte da criação de uma partição inicial do conjunto de dados fornecido. Após isso, é necessário mover iterativamente os objetos entre os *clusters* para então encontrar a disposição que otimize o critério de agrupamento, que é o erro quadrático. Logo, deve-se conseguir minimizar o erro quadrático.

O objetivo é conseguir dividir os dados em K clusters, de tal forma a minimizar a distância entre os objetos \mathbf{x}_i e o centroide $\bar{\mathbf{x}}^{(j)}$ do cluster. A soma das distâncias é o erro quadrático, definido da seguinte maneira (DUDA *et al.*, 2001):

$$E = \sum_{j=1}^k \sum_{\mathbf{x}_i \in \mathcal{C}_j} d(\mathbf{x}_i, \bar{\mathbf{x}}^{(j)})^2$$

O algoritmo mais conhecido pertencente a esta categoria é o *k-means*, ou também *k-médias*. Esta técnica parte da escolha de k centroides, que geralmente é uma escolha aleatória de objetos do conjunto. Em seguida, cada um dos objetos restantes é associado ao cluster cujo centroide está mais próximo. Após essa associação, o algoritmo recalcula os centroides e esse processo é então repetido até que não haja mais variação de centroide entre iterações subsequentes (ou até que a variação seja menor do que um limiar predeterminado). Portanto, a implementação do algoritmo *k-means* pode ser simplificada em passos:

- **Passo 1:** especificar o número k de *clusters* a serem criados.
- **Passo 2:** selecionar aleatoriamente k objetos do domínio para serem os centroides iniciais.
- **Passo 3:** associar cada observação ao cluster mais próximo.
- **Passo 4:** para cada um dos k *clusters*, calcule o novo centroide, fazendo a média de todos os dados de cada i -ésimo cluster.
- **Passo 5:** iterativamente, minimize o erro quadrático. Isso é, repita os passos 3 e 4 até o centroide estabilizar ou até o número máximo de iterações ser atingido.

ASSIMILE



Uma das grandes desvantagens desse método é que ele é intimamente dependente da estimativa inicial do centroide dos *clusters*. Isso pode levar o algoritmo a convergir para um ótimo local, em vez de realmente encontrar o ótimo global do problema.

► 4. Algoritmos baseados em densidade

O algoritmo *k-means*, mesmo sendo um dos mais populares, não tem a capacidade de identificar *outliers*, então todos os pontos são associados a algum cluster, mesmo não pertencendo realmente a nenhum.

Em problemas com dados anômalos, esses pontos acabam sendo classificados como normais e, conseqüentemente, puxam o centroide para perto deles, o que pode mudar bastante a solução do problema.

Os algoritmos baseados em densidade, como, por exemplo, o DBSCAN (*density based spatial clustering of applications with noise*), conseguem lidar bem com esse tipo de problema e também são capazes de descobrir

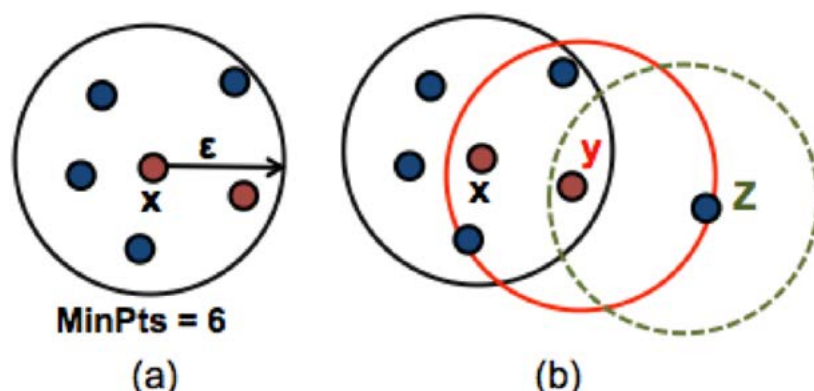
cluster com formatos aleatórios. Além disso, não requerem um número-alvo de *clusters* como parâmetro inicial. O DBSCAN precisa basicamente de dois parâmetros para poder encontrar os agrupamentos:

- ϵ : o raio de vizinhança ao redor de qualquer ponto p a ser analisado.
- minPTs: o número mínimo de pontos necessários para que uma vizinhança seja definida como cluster.

Com esses parâmetros, o algoritmo consegue dividir os dados em **pontos principais**, **pontos de fronteira** e **outliers**. Um ponto é considerado principal se ao seu redor existem pelo menos minPts pontos. Porém, se a vizinhança de um ponto q^* contém menos do que minPts, mas esse ponto é alcançável a partir de um ponto principal, então ele é considerado um ponto de fronteira. Alternativamente, se o ponto não se encaixa em nenhuma dessas classificações, ele é tido como um *outlier*.

Observe a Figura 4. Utilizando minPts=6, o ponto **x** é principal, pois dentro do raio ϵ partindo de **x**, existem 6 pontos. Já o ponto **y** é uma fronteira, pois em sua vizinhança existem 5 pontos, mas ele está dentro do raio de alcance de um ponto central. Porém, o ponto **z** não é atingível por nenhum ponto principal e em sua vizinhança não há minPTs.

Figura 4 – Disposição de pontos e classificação via DBSCAN

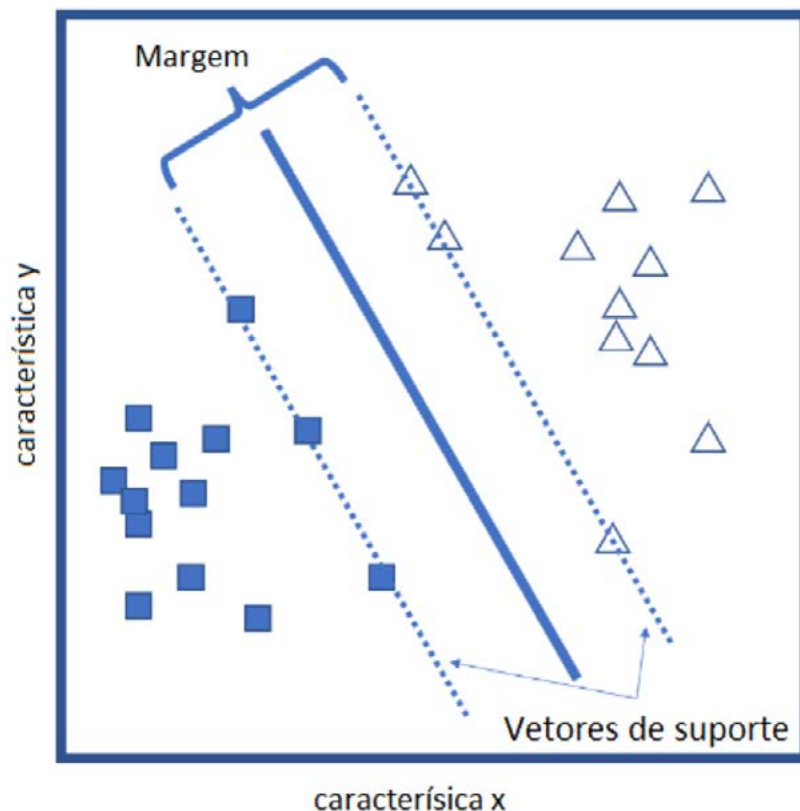


Fonte: Ester *et al.* (1996).

► 5. Support vector machine

As máquinas de vetor de suporte (SVM, *support vector machine*) são modelos de aprendizado de máquina que utilizam hiperplanos para separar os dados do problema. Para particionar os dados, é necessário encontrar algum plano (no caso de dados bidimensionais, encontra-se uma linha) que separa os dados e utiliza vetores de suporte para maximizar essa separação. A Figura 5 ilustra essa separação utilizando os vetores de suporte.

Figura 5 – Gráfico de uma SVM simples aplicada a dados aleatórios

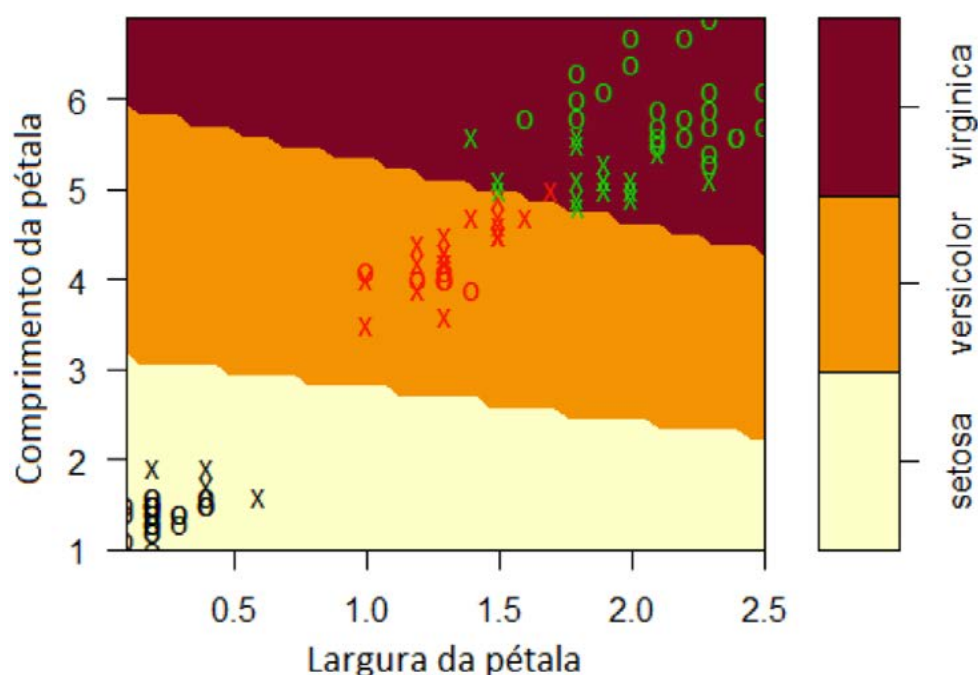


Fonte: adaptada de Burger (2018).

Observe que o algoritmo precisa escolher algum vetor para dar suporte ao critério de otimização da separação entre vetores de dados referentes a classes diferentes. No caso da Figura 5, fez-se necessário escolher dois vetores (um contendo quadrados e outro contendo triângulos) para poder criar uma linha que mantivesse a maior margem de separação possível entre esses dados.

Para entender melhor como ficaria essa separação em um problema real de aprendizado de máquina, observe a Figura 6, que contém dados do problema de classificação Iris. Neste gráfico, todos os objetos denotados por "X" são os vetores de suporte, e aqueles descritos como "O" são os elementos restantes. A observação da figura deixa claro, mais uma vez, que os dados da classe "setosa" podem ser facilmente diferenciados, mas as classes "virgínica" e "versicolor" precisam de mais cuidado ao serem analisadas.

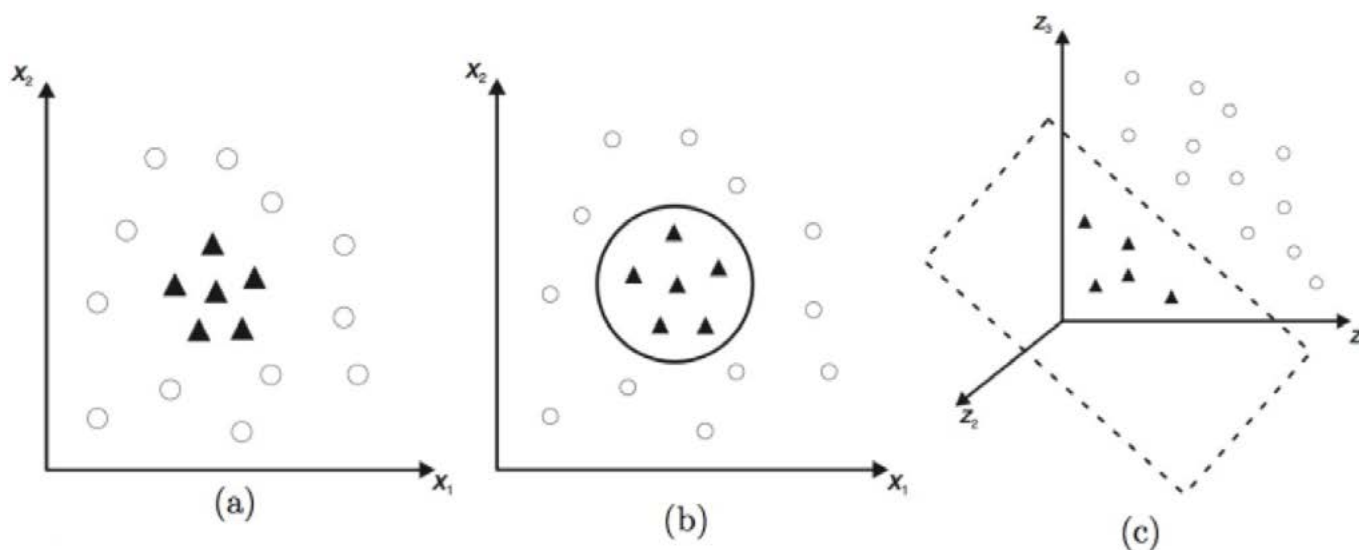
Figura 6 – Resolução do problema Iris utilizando SVM



Fonte: elaborada pelo autor.

Os exemplos apresentados até agora foram resolvidos com uma fronteira linear. Porém, o problema aparece quando essas barreiras precisam ser não lineares. Para esses casos, é aconselhável utilizar uma SVM, pois ela consegue aplicar uma técnica chamada *kernel trick*, que, primeiramente, aumenta a dimensão do problema para poder então encontrar uma solução linear no domínio aumentado.

Figura 7 – (a) Conjunto de dados; (b) fronteira bidimensional não linear; (c) fronteira tridimensional linear



Fonte: adaptada de Faceli (2015, p.131).

Considere então os dados com domínio em \mathbb{R}^2 apresentados na Figura 7a, cuja fronteira teria que ser não linear, assim como mostra a Figura 7b. Porém, caso você aumente o domínio do problema para \mathbb{R}^3 , poderá utilizar um hiperplano linear para separar os dados, assim como mostra a Figura 7c.

► 6. Processamento em linguagem natural

As linguagens formais, como, por exemplo, as linguagens de programação, possuem modelos e regras bem definidas e finitas. Você sabe que a *string* “printf(2 + 2)” é um comando legal, ao passo que “printf)2 + 2(” não é. As linguagens naturais, como português e inglês, não podem ser caracterizadas da mesma maneira. Qualquer pessoa que ler a frase “Aluga-se apartamentos” consegue saber qual o sentido dela; porém alguns podem falar que ela está errada e o correto seria “Alugam-se apartamentos”.

Além disso, a linguagem natural pode ser ambígua, ou seja, a mesma frase pode ter dois significados. Um exemplo é a frase “Lucas foi atrás do táxi correndo”, que pode significar que Lucas saiu correndo para

alcançar o táxi, ou então que Lucas saiu atrás do táxi que já estava em movimento. Isso mostra por que o processamento em linguagem natural não pode focar em um único sentido, mas sim em uma distribuição de probabilidade sobre vários significados possíveis.

Devido a essas características citadas, pode-se dizer que é difícil trabalhar com as linguagens naturais, pois elas são muito abrangentes, sem definição e estão em constante mudança. Portanto, os modelos de linguagem são, em seus melhores casos, uma aproximação. Para isso, você aprenderá uma aproximação clássica utilizada nesse contexto: modelo **N-gram de caracteres** (RUSSEL, 2010).

Todo texto escrito é composto de caracteres (letras, pontos, espaço, dígitos, etc.). Sendo assim, uma das formas mais simples de criar um modelo da linguagem natural é encontrar uma distribuição de probabilidade sobre uma sequência de caracteres. Para seguir com os estudos, considere $P(c_{1:N})$ como sendo a probabilidade de uma sequência de N caracteres, c_1 até c_N . Sendo assim, pode-se escrever, por exemplo, $P(\text{"isto"}) = 0,027$ e $P(\text{"zhp"}) = 0,000000002$.

Segundo Russel (2010), qualquer sequência de N símbolos é então um N-gram, e os casos especiais para sequência de um, dois e três caracteres são, respectivamente, monograma, bigrama, trigrama. Além disso, um N-gram pode ser definido como uma cadeia de Markov, o modelo estocástico que descreve uma sequência de possíveis eventos, sendo que a probabilidade de cada evento depende exclusivamente do estado do evento anterior (LAWLER, 2006). Com essa definição, é possível escrever a probabilidade de uma sequência de caracteres $P(c_{1:N})$ utilizando a ideia dos eventos da cadeia de Markov:

$$P(c_{1:N}) \prod_{i=1}^N P(c_i | c_{1:i-1})$$

Uma tarefa na qual o modelo N-gram é bastante utilizado é para a identificação da língua de textos. Esse é um problema relativamente fácil, e até com textos pequenos como “Olá mundo” ou “Hello world”.

Para essa identificação, uma possibilidade é criar um modelo trigrama de cada língua candidata, $P(c_i | c_{i-2:i-1}, l)$, em que l é referente a cada uma das línguas. O objetivo, então, é encontrar a língua l^* que seja ótima, ou seja, a mais provável. Para isso, pode-se utilizar a regra de Bayes e a característica da cadeia de Markov para escrever o problema como o argumento que maximiza a probabilidade $P(\text{texto} | \text{língua})$, da seguinte maneira (RUSSEL, 2010):

$$\begin{aligned} l^* &= \underset{l}{\operatorname{argmax}} P(l | c_{1:N}) \\ &= \underset{l}{\operatorname{argmax}} P(l) P(c_{1:N} | l) \\ &= \underset{l}{\operatorname{argmax}} P(l) \prod_{i=1}^N P(c_i | c_{i-2:i-1}, l) \end{aligned}$$

Observe que a probabilidade a *posteriori* pode ser facilmente calculada a partir dos dados. E como você saberá a probabilidade a *priori* $P(l)$? Esse parâmetro deve ser selecionado de acordo com o cenário de aplicação. Por exemplo, caso o texto a ser analisado seja de um website, a probabilidade de ele estar em inglês é muito maior do que a chance de ser escrito em vietnamita.

Existem também outras aplicações para essa técnica, como, por exemplo, correção de erros ortográficos, classificação de gênero ou reconhecimento de entidades. Classificação de gênero significa reconhecer qual é o gênero do texto (história, romance, artigo científico, jornal, etc.). Reconhecimento de entidade é a tarefa de encontrar nomes de entidades de documentos e decidir a que classe elas pertencem.

TEORIA EM PRÁTICA



O aprendizado de máquina é um conteúdo muito amplo, e como visto durante todos os conteúdos, um mesmo problema pode ser resolvido de inúmeras maneiras diferentes. Com certeza, existem técnicas mais eficientes

do que outras, ou até técnicas que não funcionam para a resolução de todo tipo de problema. Um fator muito importante na escolha do algoritmo a ser utilizado é a estrutura do domínio dos dados.

O estudo de cada algoritmo apresentado só é realmente aprendido quando você tentar implementar a técnica estudada, pois a implementação exige muito conhecimento.

As plataformas de simulação, como o Python ou o R, fornecem diversas ferramentas para auxiliar na implementação e criação do algoritmo escolhido. Tendo em vista que você já está bastante familiarizado com pelo menos uma dessas plataformas, seu desafio agora será implementar o algoritmo *K-means* para a resolução do clássico problema Iris.

Uma atividade interessante a ser feita é também a comparação entre o seu algoritmo *K-means* e o algoritmo já existente no software R. Para utilizar a função que realiza a classificação *K-means* em R, você pode utilizar as seguintes linhas de código:

```
kmeans(x, centers, iter.max = 10, nstart = 1,  
algorithm = c("Hartigan-Wong", "Lloyd", "Forgy","MacQueen"),  
race=FALSE)
```

As entradas da função são:

- x: matriz de dados.
- centers: pode ser o número de *clusters*, *k*, ou os centroides iniciais. Se especificar o valor de *k*, os centroides serão escolhidos aleatoriamente.
- iter.max: o número máximo de iterações.
- nstart: número de conjuntos iniciais a serem escolhidos.
- algorithm: algoritmo a ser utilizado.
- x: matriz de dados;

Para implementar o *k-means* manualmente, lembre-se da sequência de passos necessários:

Passo 1: especificar o número k de *clusters* a serem criados.

Passo 2: selecionar aleatoriamente k objetos do domínio para serem os centroides iniciais.

Passo 3: associar cada observação ao cluster mais próximo.

Passo 4: para cada um dos k *clusters*, calcule o novo centroide, fazendo a média de todos os dados de cada i -ésimo cluster.

Passo 5: iterativamente, minimize o erro quadrático. Isso é, repita os passos 3 e 4 até o centroide estabilizar ou até o número máximo de iterações ser atingido.

Agora é com você!

Utilize o conhecimento adquirido e faça pesquisas em materiais e nas referências bibliográficas.

Para testar seu algoritmo e a função já fornecida pelo software R, você deverá, mais uma vez, utilizar o banco de dados Iris, que é uma coleção de informações sobre a classificação de plantas em três grupos: setosa, virgínica e versicolor por meio da largura e do comprimento de suas pétalas e sépalas.

Bom trabalho.

VERIFICAÇÃO DE LEITURA



1. Um algoritmo de aprendizado de máquina baseado em agrupamento (*clustering*) precisa seguir alguns critérios básicos.

Sobre esses critérios, analise as afirmativas a seguir:


- I. A compactação é um desses critérios e está relacionada à ideia de que objetos que estão muito próximo pertencem ao mesmo cluster.
- II. O encadeamento, ou ligação, é um desses critérios e está relacionada à ideia de que objetos que estão muito próximo pertencem ao mesmo cluster.
- III. A separação espacial é um desses critérios e ele leva em consideração a separação entre os *clusters*.

Assinale a alternativa que julga corretamente os itens I a III.

- a. Somente II está correto.
 - b. Somente II e III estão corretos.
 - c. Somente I e III estão corretos.
 - d. Somente III está correto.
 - e. Somente I está correto.
2. As máquinas de vetor de suporte são algoritmos de aprendizado de máquina supervisionado muito eficientes que analisam os dados para fazer tanto classificação como regressão.

A respeito do AM do tipo máquina de vetor de suporte (SVM), assinale a alternativa correta.

- a. As SVM são algoritmos capazes de lidar com fronteiras lineares para fazer a separação dos dados, mas não conseguem manipular o domínio da função para resolver problemas que exijam uma separação não linear.

- 
- b. As SVM recebem esse nome porque esse tipo de algoritmo gera como resultado um vetor de dados, que é criteriosamente selecionado para poder proporcionar o suporte necessário para o usuário resolver o problema.
 - c. A técnica de *kernel trick* é utilizada pelas SVM para manipular o domínio do problema, aumentar sua dimensão e então encontrar uma solução linear no novo domínio.
 - d. As SVM criam hiperplanos para classificar os dados e por isso elas não são capazes de lidar com *outliers*.
 - e. A técnica de *kernel trick* é utilizada pelas SVM para manipular o domínio do problema, diminuir sua dimensão e então encontrar uma solução linear no novo domínio.
3. Ao falar de aplicações de aprendizado de máquina e mineração de dados, dois algoritmos devem ser sempre lembrados: modelos hierárquicos e *K-means*. Sobre esses dois algoritmos, assinale a alternativa correta.
- a. O método hierárquico é mais subjetivo que o *K-means*, e o analista que o utiliza precisa saber exatamente como os dados do problema são relacionados.
 - b. O método hierárquico exige elevados recursos computacionais para trabalhos que possuam muitas amostras.
 - c. O método hierárquico analisa o relacionamento entre variáveis, ou objetos, dependentes e independentes.

d. No modelo hierárquico, é imprescindível que o número de *clusters* seja definido *a priori*.

e. O modelo hierárquico executa a análise dos dados no mesmo sentido que o *K-means*.

Referências bibliográficas

BURGER, S. **Introduction to machine learning with R: rigorous mathematical analysis**. Sebastopol, CA: O'Reilly, 2018.

DUDA, R. O.; HART, P. E.; STORK, D. G. **Pattern classification**. 2. ed. Nova Jersey: Wiley-interscience, 2001. 688 p.

ESTER, M.; KRIEGEL, H.-P.; SANDER, J. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. *In*: INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 2., 1996, Oregon. **Proceedings [...]**. Oregon, 4 ago. 1996. p. 226-231.

FACELI, K.; LORENA, A. C.; GAMA, J.; CARVALHO, A. C. P. L. F. **Inteligência artificial: uma abordagem de aprendizado de máquina**. São Paulo: LTC Editora, 2011.

HANDL, J.; KNOWLES, J.; KELL, D. Computational cluster validation in post-genomic data analysis. **Bioinformatics**, [s. l.], v. 21, n. 5, p. 3.201-3.212, 1 ago. 2005.

LAWLER, G. F. **Introduction to stochastic processes**. 2. ed. Boca Raton: Chapman & Hall/crc, 2006. 234 p.

MAINI, V.; SABRI, S. **Machine learning for humans**. 2017. Disponível em: <https://everythingcomputerscience.com/books/Machine%20Learning%20for%20Humans.pdf>. Acesso em: 26 ago. 2019.

MITCHEL, M. B. **Machine learning**. New York: McGraw-Hill Science/Engineering/Math, 1997. 432 p.

RASCHKA, S. **Python machine learning**. Birmingham: Packt Publishing, 2015.

REDDY, C. **Understanding the concept of hierarchical clustering technique**. 2018. Disponível em: <https://towardsdatascience.com/understanding-the-concept-of-hierarchical-clustering-technique-c6e8243758ec>. Acesso em: 2 set. 2019.

RUSSEL, S.; NORVIG, P. **Artificial Intelligence: a modern approach**. 3. ed. New Jersey: Pearson Education Inc., 2010. 1.132 p.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. **Understanding machine learning: from theory to algorithms**. New York: Cambridge University Press, 2014.

SMOLA, A.; VISHWANATHAN, S. V. N. **Introduction to machine learning**. New York, NY: Cambridge University Press, 2008. 226 p.

► Gabarito

Questão 1 – Resposta B

Resolução:

- Compactação: também conhecida como homogeneidade, e está relacionada à variação intracluster. Os algoritmos de AM que priorizam ou otimizam este critério geralmente são bem aplicados para descobrir algum tipo de cluster esférico e/ou bem separado.
- Encadeamento ou ligação: este conceito está relacionado à ideia de que objetos vizinhos compartilham o mesmo cluster. Algoritmos baseados neste critério são indicados para a organização de *clusters* de forma arbitrária, porém não possui bom desempenho na identificação de casos em que *clusters* diferentes estão muito próximos.
- Separação espacial: este critério leva em conta a separação entre os *clusters*. Sendo assim, como não analisa profundamente os objetos dentro dos *clusters*, este método pode resultar em uma solução trivial. Por isso, este critério é geralmente aplicado somente quando associado a outros.

Questão 2 – Resposta C

Resolução:

- As SVM são algoritmos capazes de lidar com fronteiras lineares para fazer a separação dos dados, mas **conseguem** manipular o domínio da função para resolver problemas que exijam uma separação não linear.
- As SVM recebem esse nome porque esse tipo de algoritmo seleciona vetores de suporte para poder resolver o problema.
- As SVM criam hiperplanos para classificar os dados. As SVM são bastante indicadas para problemas que possuem *outliers*.

- A técnica de *kernel trick* é utilizada pelas SVM para manipular o domínio do problema, **aumentar** sua dimensão e então encontrar uma solução linear no novo domínio.

Questão 3 – Resposta B

Resolução:

- O analista que utiliza o método hierárquico **não** precisa saber exatamente como os dados do problema são relacionados.
- O método hierárquico analisa a similaridade entre as variáveis que, inicialmente, são todas definidas como *clusters* diferentes.
- No modelo hierárquico, **não é necessária a definição** do número de *clusters*.
- O modelo hierárquico executa a análise dos dados em **sentido oposto** do *K-means*.



Redes neurais artificiais e introdução ao *deep learning*

Autor: Lucas Claudino

► Objetivos

- Estudar os principais conceitos das redes neurais artificiais, seu funcionamento e arquiteturas básicas.
- Estudar a metodologia de treinamento e síntese de resultados das redes neurais artificiais.
- Conhecer o princípio de funcionamento e as características básicas dos algoritmos de *deep learning*.

► 1. Introdução

Redes neurais artificiais e *deep learning*, atualmente, fornecem grande parte das melhores soluções para vários problemas em reconhecimento de imagem e voz e processamento de linguagem natural, além de inúmeras outras aplicações.

Tendo em vista essas aplicações principais e inúmeras outras que necessitam do auxílio de técnicas de processamento computacional e aprendizado de máquina, esta Leitura Fundamental irá tratar sobre três metodologias fundamentais para o *machine learning*: técnicas de *clustering*, *support vector machines* e processamento em linguagem natural.

► 2. Redes neurais artificiais

Uma rede neural artificial (RNA), ou *neural network*, assim como o próprio nome sugere, tem sua construção baseada na forma como os neurônios trabalham em um sistema nervoso animal (MCCULLOH *et al.*, 1943). Basicamente, para uma determinada lista de entrada, uma rede neural processa um determinado número de passos antes de retornar uma saída ou resposta. A complexidade de uma RNA advém da quantia de passos necessários e quão complexo é cada um desses passos.

Um exemplo bastante simples para você entender o princípio de funcionamento de uma RNA é o uso de portas lógicas. Considere uma porta lógica do tipo AND, cuja saída é considerada verdadeira (nível lógico 1) apenas se ambas as entradas forem verdadeiras. O quadro-verdade para essa porta lógica relaciona suas entradas (A e B) e a saída, e pode ser observada no Quadro 1.

Quadro 1 – Quadro-verdade de uma porta lógica AND com duas entradas

Entrada A	Entrada B	Saída
Falso (nível 0)	Falso (nível 0)	Falso (nível 0)
Falso (nível 0)	Verdadeiro (nível 1)	Falso (nível 0)
Verdadeiro (nível 1)	Falso (nível 0)	Falso (nível 0)
Verdadeiro (nível 1)	Verdadeiro (nível 1)	Verdadeiro (nível 1)

Fonte: elaborada pelo autor.

É possível, então, definir uma rede neural simples tão simples quanto uma que recebe as entradas, calcula a função AND e gera o resultado. Isso pode ser representado graficamente por meio de camadas (*layers*) e nós. A elaboração matemática desse modelo, em grande parte das vezes, irá requerer a utilização de uma variável de *bias*, que é simplesmente uma constante a ser adicionada e possui seu próprio nó.

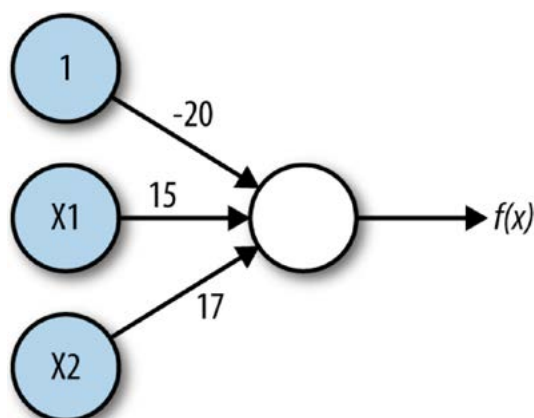
Para o caso da porta lógica AND, é necessária uma função que gere o valor 1 para representar a opção “Verdadeiro” e valor 0 para representar “Falso”. Isso pode ser feito por meio da função sigmoide

$$f(x) = \frac{1}{1 + e^{-x}}$$

Com x variando de $-\infty$ a $+\infty$.

Se houver uma definição dos pesos para cada nó, uma rede neural para este caso pode ser descrita da seguinte maneira:

Figura 1 – Exemplo de rede neural artificial para uma porta lógica AND



Fonte: Burger (2018).

A primeira camada contém as entradas $X1$ e $X2$ e a variável de *bias*, que é uma constante de valor 1. Cada nó possui um peso associado. O nó representado pelo círculo branco é o responsável por fazer o cálculo relacionando todas as entradas, seus respectivos pesos, e levar o resultado para a função de ativação. A saída da função sigmoide será, então, o resultado da RNA.

As variáveis $X1$ e $X2$ podem assumir dois valores: verdadeiro ou falso. Sendo assim, considere, a seguir, o cálculo detalhado para cada uma das opções de entrada:

$$X1 = 1; X2 = 1 \rightarrow x = -1 \cdot -20 + X1 \cdot 15 + X2 \cdot 17 = 12 \rightarrow f(x) = f(12) \approx 1$$

$$X1 = 1; X2 = 0 \rightarrow x = -1 \cdot -20 + 1 \cdot 15 + 0 \cdot 17 = -5 \rightarrow f(x) = f(-5) \approx 0$$

$$X1 = 0; X2 = 1 \rightarrow x = -1 \cdot -20 + 0 \cdot 15 + 1 \cdot 17 = -3 \rightarrow f(x) = f(-3) \approx 0$$

$$X1 = 0; X2 = 0 \rightarrow x = -1 \cdot -20 + 0 \cdot 15 + 0 \cdot 17 = -20 \rightarrow f(x) = f(-20) \approx 0$$

Portanto, o resultado da RNA, que é a saída de $f(x)$, representa corretamente a porta lógica AND. Observe que, neste exemplo, todos os nós já possuíam seus pesos predeterminados. Porém, a etapa de treinamento é a responsável por calcular esses pesos.

De forma simples, o que aconteceu nessa rede neural artificial exemplificada foi o seguinte: iniciou-se com um *layer* de variáveis de entrada já com seus pesos definidos. Passou-se, então, por uma camada de processamento (nesse caso, uma função sigmoide) e assim o resultado foi obtido. Esse exemplo foi simples, porém, se houvesse várias camadas de processamento ou muitas variáveis de entrada, o problema seria mais complexo.

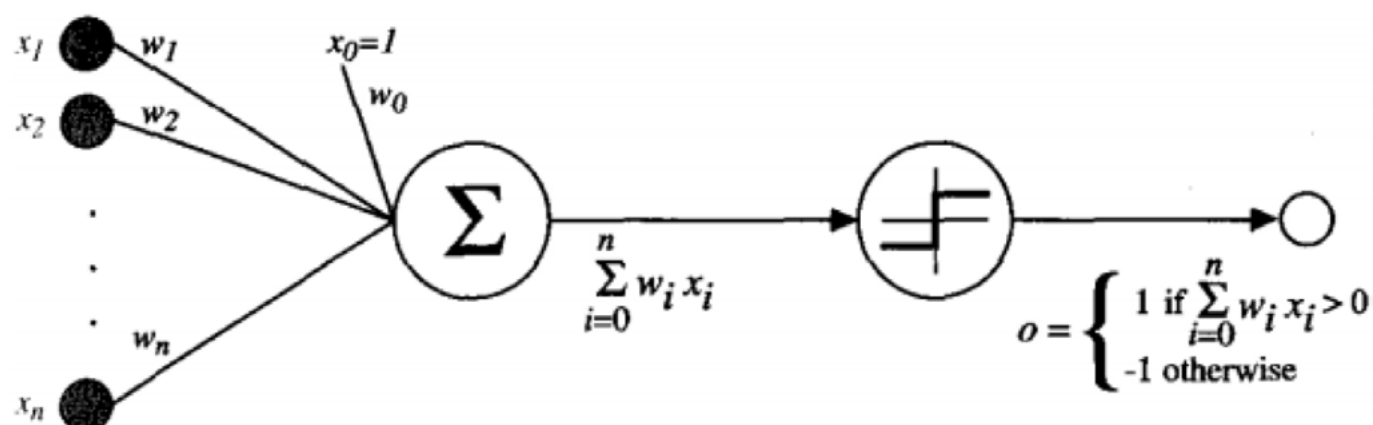
Com esse exemplo, você pôde lidar com as três camadas básicas de toda rede neural artificial:

- Camada de entrada: é composta pelos neurônios de entrada e traz os dados iniciais para dentro do sistema, para que assim possam ser posteriormente processados.

- Camada oculta: é a camada de processamento. É o *layer* que calcula uma determinada função a partir de cada uma das características. Esta camada pode ser tão simples quanto somente um nó, ou então possuir várias subcamadas com diversos nós.
- Camada de saída: é o nó final de processamento, que pode ser uma função única.

Um tipo clássico de rede neural é a RNA *perceptron*, que é muito semelhante à rede neural utilizada no exemplo, porém utiliza outra função para realizar a decisão e gerar a resposta do sistema. A Figura 2 contém o diagrama de uma *perceptron* de n características de entrada:

Figura 2 – Rede neural artificial do tipo *perceptron*



Fonte: Mitchel (1997, p. 87).

Essa RNA considera um vetor de entradas reais, calcula a combinação linear entre esses objetos de entrada e gera um resultado 1 se a combinação é maior do que um limiar predeterminado, caso contrário, a saída é -1. Matematicamente, a saída pode ser, então, descrita por:

$$o(x_1, \dots, x_n) = \begin{cases} 1, & \text{se } w_0 + w_1x_1 + \dots + w_nx_n > 0 \\ -1, & \text{caso contrário} \end{cases}$$

Sendo que $\{x_1, \dots, x_n\}$ são as entradas, w_i são as constantes que pesam cada um dos objetos de entrada. Observe que a combinação $\sum_{i=1}^n w_i x_i$ precisa ser maior do que $-w_0$ para que a saída seja igual a 1.

Para simplificar a notação, pode-se considerar $x_0 = 1$, o que permite escrever a inequação $\sum_{i=1}^n w_i x_i$ por meio do produto escalar $\vec{w} \cdot \vec{x} > 0$. Logo, a função de saída pode ser descrita por meio da função *senal* aplicada ao vetor de entrada \vec{x} :

$$o(\vec{x}) = \text{sgn}(\vec{w} \cdot \vec{x})$$

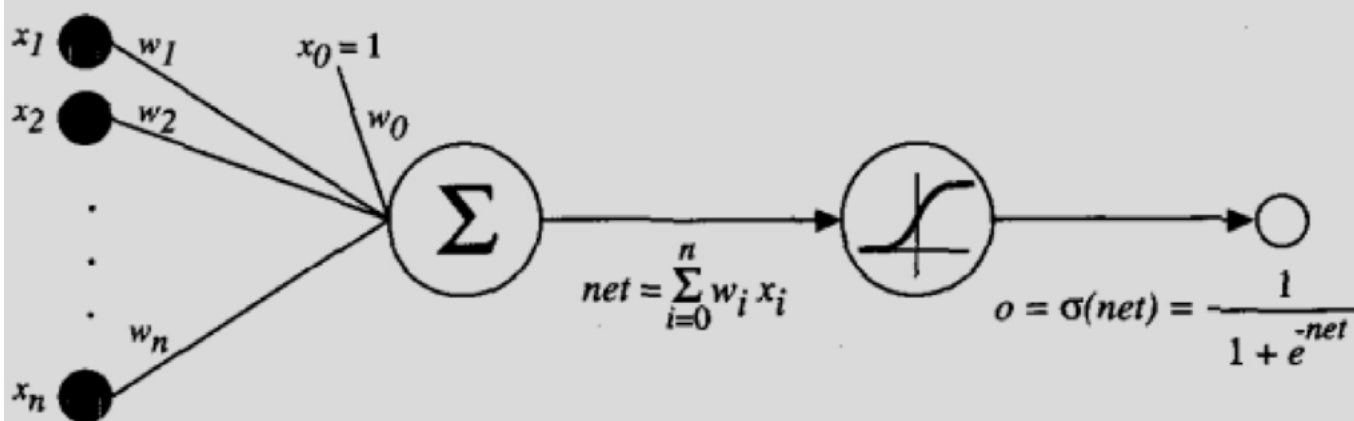
Em que:

$$\text{sgn}(y) = \begin{cases} 1, & \text{se } y > 0 \\ -1, & \text{caso contrário} \end{cases}$$

ASSIMILE

Observe que a RNA utilizada no exemplo pode ser facilmente descrita matematicamente de forma similar à RNA *perceptron*, basta apenas trocar a função sinal pela função sigmoide, assim como mostra a Figura 3.

Figura 3 – Rede neural artificial com uso de cálculo de limiar por meio da função sigmoide



Fonte: Mitchel, (1997, p. 87).

2.1 Treinamento em redes *perceptron*

Uma etapa muito importante para a rede neural artificial é a fase em que ela faz o cálculo dos pesos associados a cada nó. O problema aqui é determinar um vetor de pesos que leva a *perceptron* a gerar a saída correta (± 1) para cada um dos objetos de treinamento.

Duas metodologias podem ser consideradas as principais para este treinamento: regra *perceptron* e regra delta. Essas duas metodologias têm a garantia de convergir para resultados diferentes, mas ambos geram resultados apropriados (MITCHEL, 1997).

Uma maneira é associar um peso aleatório a cada nó e, a cada iteração, modificar os pesos sempre que houver um erro de classificação. Esse processo é então repetido até que a *perceptron* classifique corretamente os objetos de treinamento. Os pesos são modificados a cada passo, de acordo com a regra de treinamento da *perceptron*, que revisa cada peso w_i associado a uma entrada x_i da seguinte maneira:

$$w_i \leftarrow w_i + \Delta w_i$$

Sendo que o valor do incremento é dado por:

$$\Delta w_i = \eta(t - o) x_i$$

Em que t é o valor-alvo, o é a saída gerada pela *perceptron* e η é uma constante positiva chamada de taxa de treinamento. A função da constante η é regular a velocidade com que os pesos são modificados a cada iteração. Geralmente, η é um valor pequeno, menor do que 1, e pode ser feito de maneira a diminuir conforme o número de iterações realizadas aumenta.

Considere um exemplo em que $x_i = 0,8$, $\eta = 0,1$, $t = 1$ e $o = -1$, ou seja, a saída $o = -1$ está errada para o valor esperado $t=1$. Nesse caso, o incremento será:

$$\Delta w_i = \eta(t - o) x_i = 0,1(1 - (-1))0,8 = 0,16$$

E o novo peso então pode ser calculado como:

$$w_i = w_i + 0,16$$

Ou seja, o peso w_i associado a x_i irá aumentar, levando então a saída o na iteração seguinte a se aproximar do valor esperado $t = 1$.

PARA SABER MAIS




Uma das formas mais utilizadas para fazer o correto treinamento de uma RNA e calcular corretamente todos os pesos é por meio do chamado *feed-forward*, que busca calcular o erro da RNA atual e utiliza a técnica de *backpropagation* para alterar os pesos e recalcular o erro. Para entender melhor essa dinâmica de implementação de uma RNA, pesquise sobre esses termos: *backpropagation* e *feed-forward*.

► 3. Introdução ao *deep-learning*

O crescente aumento na quantidade de dados disponíveis, como, por exemplo, dados provindos da implementação de redes de sensores industriais que captam informações sobre toda a produção de uma indústria, traz consigo a necessidade da criação de modelos complexos para descrever esses dados, e esses modelos não podem ser computacionalmente ineficientes.

É nesse contexto que as técnicas de *deep learning* (ou aprendizado profundo) são aplicadas. *Deep learning* é uma forma de extrair padrões úteis a partir dos dados, de uma maneira automatizada e com o menor trabalho possível. (SCHMIDHUBER, 2015). Essa técnica é um ramo do aprendizado de máquina baseado em um conjunto de algoritmos que buscam modelar abstrações em alto nível a partir da leitura de grandes quantidades de dados.



Por meio de uma simples exemplificação, considere dois conjuntos de neurônios em uma RNA: aqueles que recebem um sinal de entrada e os que enviam um sinal de saída. Quando a camada de entrada recebe um sinal, ela transmite uma versão modificada dessa entrada para a próxima camada. Em uma rede neural densa (ou profunda), existem muitas camadas entre a entrada e a saída, possibilitando que o algoritmo use vários *layers* de processamento compostos de transformações lineares e não lineares (DENG, 2014).

Segundo Deng (2014), as redes de *deep learning* podem ser classificadas de diversas maneiras; porém, a mais comum é a utilização de três classes, baseadas na forma como as redes ou tecnologias são utilizadas, como, por exemplo, a utilização para síntese/geração ou então para reconhecimento/classificação. Com isso, as técnicas de *deep learning* podem ser divididas em **redes para aprendizado não supervisionado ou de geração, redes para aprendizado supervisionado e redes híbridas**.

As redes de *deep learning* para aprendizado não supervisionado têm o principal objetivo de identificar alguma correlação dos dados observados ou visíveis para possibilitar a análise de padrões quando não se tem informação sobre os rótulos desses dados. Quando utilizado para o aprendizado de geração, pode também ter o objetivo de caracterizar distribuições estatísticas conjuntas dos dados visíveis e suas classes associadas quando disponíveis e puderem ser tratadas como parte dos dados.

Quando os algoritmos de *deep learning* são utilizados para trabalhar com aprendizado supervisionado, o propósito direto é fornecer métodos discriminativos para auxiliar na classificação de padrões, geralmente pela caracterização posterior da distribuição das classes, sendo que essa caracterização é condicionada aos dados visíveis disponíveis. Para tal aprendizado supervisionado, os rótulos estão sempre disponíveis, seja de forma direta ou indireta.

Arquiteturas de *deep learning* também podem ser utilizadas de maneira híbrida. Nesse contexto de aprendizado de máquina, o termo “híbrido” se refere a topologias que tanto utilizam metodologias generativas quanto discriminativas. Nesse caso, o componente generativo é mais explorado para discriminação, que é o objetivo da arquitetura híbrida.

A seguir, você terá a chance de estudar algumas aplicações de *deep learning*.

3.1 Coloração de imagens preto e brancas

Deep learning pode ser aplicado para utilizar os objetos e seu contexto dentro de uma fotografia para colorir a imagem, similarmente como um ser humano faria para resolver o problema. Esse cenário de utilização requer o uso de redes neurais convolucionais bastante densas e camadas supervisionadas que recriam a imagem com a adição de cores.

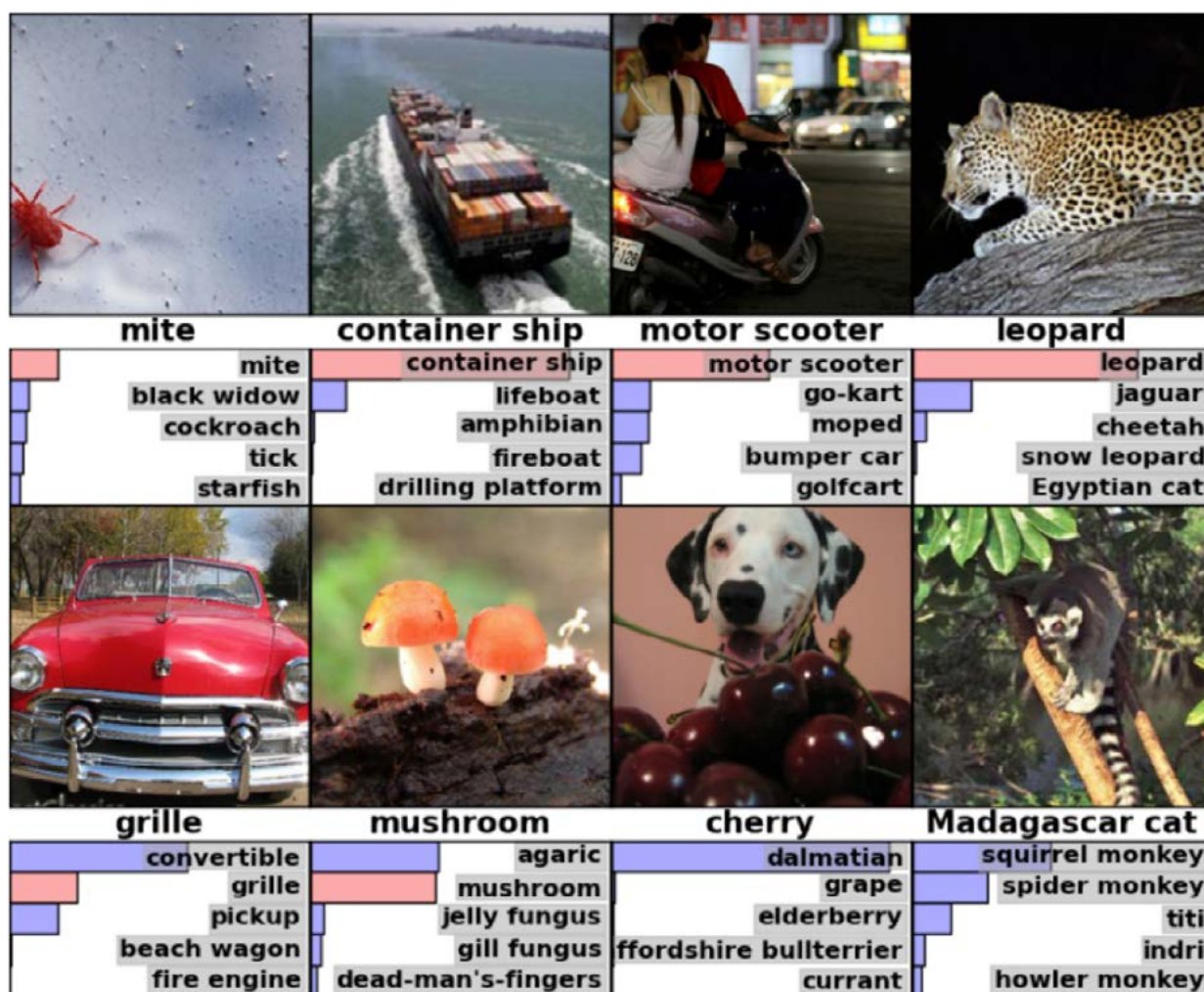
3.2 Tradução automática

Tradução de textos pode ser performada sem a utilização de pré-processamento da sequência de caracteres, permitindo que o algoritmo aprenda a dependência entre as palavras e seus respectivos mapeamentos para a nova linguagem.

3.3 Classificação e detecção de objetos em fotografias

Esta tarefa requer que objetos contidos em fotografias sejam classificados de acordo com conjunto de objetos previamente conhecidos. Os resultados significativos nessa área foram obtidos com o uso de grandes redes neurais convolucionais. Um exemplo muito conhecido é o trabalho de Krizhevsky *et al.* (2012), que treinou uma rede neural convolucional com 60 milhões de parâmetros e 650 mil neurônios distribuídos por cinco camadas convolucionais. O resultado foi um algoritmo capaz de classificar mais de um milhão de imagens em alta resolução do banco ImageNet em mil classes diferentes. Observe na Figura 4 como é o resultado desse algoritmo de *deep learning*.

Figura 4 – Oito imagens de teste aplicadas ao algoritmo de classificação de objetos em fotografias



Fonte: Krizhevsky *et al.* (2012, p. 8).

A Figura 4 mostra oito imagens classificadas pelo algoritmo de Krizhevsky *et al.* (2012). Abaixo de cada imagem está a classificação correta e também as cinco classes prováveis geradas como resultado. Cada classe contém sua probabilidade de acerto gerada e a probabilidade de cor vermelha é referente à classe que descreve corretamente o objeto da imagem.

3.4 Jogadores automáticos para jogos computacionais

Este é o desafio em que o algoritmo precisa aprender a jogar baseado apenas na análise dos pixels da tela. Essa difícil tarefa é executada por algoritmos de *deep learning* de reforço. A vantagem desses algoritmos

é que eles não necessitam de uma solução para servir de meta a ser atingida. A função do método de aprendizado computacional aqui é gerar um mapa completo dos pixels contidos na tela e, baseado em seu conhecimento adquirido, tomar decisões, como, por exemplo, a melhor rota a percorrer.

Para ilustrar esse fato, pense em um jogo de corrida. O algoritmo precisa analisar a tela, os obstáculos próximos, a pista de corrida e os competidores próximos para então decidir se pode continuar em seu caminho, se há necessidade de algum desvio ou se pode acelerar para tentar alcançar seu adversário que está à frente.

PARA SABER MAIS



A criação de jogadores automáticos, também conhecidos como *bots*, é bastante utilizada. Para saber mais sobre essa técnica, leia o artigo "*Playing Atari with deep reinforcement learning*" (Mnih *et al.*, 2013), que utiliza *deep reinforcement learning* para criar um jogador automático de Atari.

TEORIA EM PRÁTICA



Uma rede neural artificial é um modelo/algoritmo computacional baseado no sistema nervoso dos seres humanos e busca imitar a forma com que os neurônios transportam informação. Você estudou o funcionamento e a metodologia necessária para implementar uma rede neural do tipo *perceptron*, atualizando os valores dos pesos associados a cada nó para então fazer com que a saída da RNA seja igual à saída esperada.

A sua tarefa agora será implementar uma rede neural de camada única. Você poderá utilizar o software com o qual se sinta mais confortável. Para essa implementação, você deverá seguir alguns passos:

- Definir a matriz de dados a ser utilizada.
- Iniciar os pesos com zero ou um valor aleatório.
- Escolher uma taxa de aprendizagem. Pode utilizar a taxa $\eta = \left(\frac{n_1 \cdot n_2}{n_1 + n_2}\right)^{-1}$, em que n_1 e n_2 são, respectivamente, a quantidade de linhas e colunas da matriz de amostras.
- Calcular a saída a partir dos pesos atuais, utilizando a função sinal como função de ativação.
- Enquanto ainda houver diferença entre a saída obtida e a saída desejada, atualizar os pesos com a regra já estudada:

$$w_{i+1} = w_i + \Delta w_i = w_i + \eta(t - o) x_i$$

A primeira matriz de dados que você deverá utilizar é constituída de duas entradas e uma saída, assim como mostra o Quadro 2:

Quadro 2 – Amostras para o treinamento da rede neural *perceptron*

x1	x2	t
0	0	-1
0	1	-1
1	0	-1
1	1	1

Fonte: elaborada pelo autor.

Para testar seu algoritmo, você deverá utilizar os mesmos dados do Quadro 1, porém não precisará da saída “t”, pois ela servirá apenas para comparar seus resultados obtidos após o treinamento com os esperados.

Você deverá verificar que a rede neural de uma única camada, se implementada corretamente, deverá acertar todas as tentativas. Porém, isso não é verdade para o caso em que se tem mais do que duas entradas, pois a rede neural de camada única não é capaz de criar uma superfície de decisão adequada para essa tarefa.

Para comprovar esse fato, após ter implementado corretamente a RNA de camada única de duas entradas, faça o mesmo para um problema com três entradas. Os dados a serem utilizados são os contidos no Quadro 3.

Quadro 3 – Amostras para o treinamento da rede neural perceptron

X1	X2	X3	t
0,1	0,4	0,7	1
0,3	0,7	0,2	-1
0,6	0,9	0,8	-1
0,5	0,7	0,1	1

Fonte: elaborada pelo autor.

De maneira similar ao que foi feito na tarefa anterior, agora você também deverá utilizar seu algoritmo para proceder as iterações necessária e calcular os pesos que gerem a saída esperada para a RNA. Após essa etapa, utilize os mesmos dados X1, X2 e X3 para testar sua RNA e verificar se a saída obtida está de acordo com o esperado.

Agora é com você!

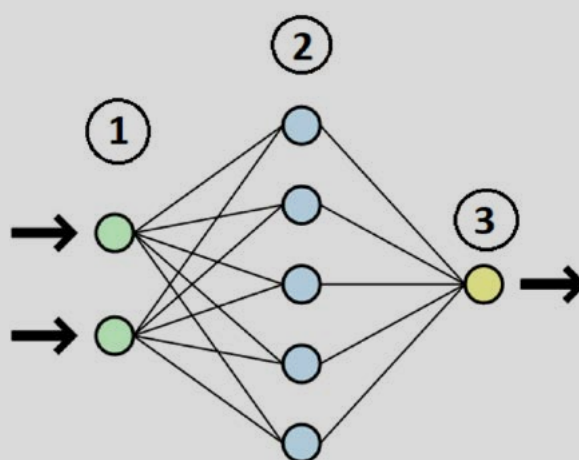
Utilize o conhecimento adquirido e faça pesquisas em materiais e nas referências bibliográficas.

Bom trabalho.

VERIFICAÇÃO DE LEITURA

1. As redes neurais artificiais podem ser constituídas tanto por estruturas simples como por sistemas complexos com diversas camadas. A Figura 5 ilustra uma estrutura genérica de uma RNA:

Figura 5 – Estrutura básica de uma rede neural artificial



Fonte: elaborada pelo autor.

Assinale a alternativa que contém corretamente o nome das camadas 1, 2 e 3, respectivamente, ilustradas na Figura 5.

- a. Subcamada; camada intermediária; camada de saída.
- b. Camada principal; camada intermediária; camada de saída.

- c. Camada de entrada; camada oculta; camada de saída.
- d. Camada principal; camada oculta; camada de saída.
- e. e) Camada de entrada; camada oculta; camada final.

2. As redes neurais artificiais (RNA) recebem este nome pois têm seu funcionamento baseado na arquitetura das redes neurais biológicas. Elas são constituídas de diversas camadas, sendo que cada uma delas pode conter diversos neurônios, ou nós.

Sobre as redes neurais artificiais, seu funcionamento e sua arquitetura, assinale a alternativa que contém uma afirmação correta.

- a. Um exemplo clássico de rede neural artificial é a rede do tipo *perceptron*, que utiliza a função sigmoide como função de decisão para gerar a resposta do sistema.
- b. Uma rede neural artificial precisa associar pesos referentes a cada uma das entradas do algoritmo, e uma vez estabelecidos quais os valores desses pesos, não é possível modificá-los.
- c. Uma rede neural artificial não possui uma arquitetura dinâmica, e por isso ela não é indicada para a utilização em sistemas em que os dados a serem analisados estão em constante modificação.
- d. Uma rede neural funciona de maneira similar ao cérebro humano. Um neurônio em uma RNA é uma função matemática que coleta e classifica informações de acordo com sua arquitetura específica.

e. A etapa de treinamento é fundamental para o cálculo dos pesos associados aos neurônios, porém, ela só funciona de maneira correta caso os pesos iniciais sejam associados com um erro menor do que 10% em relação ao seu valor ótimo.

3. Segundo Deng (2014), as redes de *deep learning* podem ser classificadas de diversas maneiras; porém, a mais comum é a utilização de três classes, baseadas na forma como as redes ou tecnologias são utilizadas, como, por exemplo, a utilização para síntese/geração ou então para reconhecimento/classificação.

Assinale a alternativa que contém essas três classificações possíveis das técnicas de *deep learning*.

- a. Redes para aprendizado não supervisionado ou generativo; redes para aprendizado supervisionado; redes híbridas.
- b. Redes para aprendizado não supervisionado ou generativo; redes para aprendizado supervisionado; redes neurais.
- c. Redes simples; redes complexas; redes densas.
- d. Redes densas; redes supervisionadas; redes não supervisionadas.
- e. Redes de regressão; redes de classificação e redes híbridas.

► Referências bibliográficas

- BURGER, S. **Introduction to machine learning with R: rigorous mathematical analysis**. Sebastopol, CA: O'Reilly, 2018.
- DENG, L. Deep learning: methods and applications. **Foundations and Trends® in Signal Processing**, [s.l.]: Now Publishers, v. 7, n. 3-4, p. 197-387, 2014. Disponível em: <http://dx.doi.org/10.1561/20000000039>. Acesso em: 13 jan. 2020.
- FACELI, K.; LORENA, A. C.; GAMA, J.; CARVALHO, A. C. P. L. F. **Inteligência artificial: uma abordagem de aprendizado de máquina**. São Paulo: LTC Editora, 2011.
- KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet classification with deep convolutional neural networks. *In*: INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS. Nevada, 3 jun. 2012. p. 1-9.
- MAINI, V.; SABRI, S. **Machine learning for humans**. 2017. Disponível em: <https://everythingcomputerscience.com/books/Machine%20Learning%20for%20Humans.pdf>. Acesso em: 26 ago. 2019.
- MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The Bulletin of Mathematical Biophysics**, [s.l.]: Springer Science and Business Media LLC, v. 5, n. 4, p. 115-133, dez. 1943. Disponível em: <http://dx.doi.org/10.1007/bf02478259>. Acesso em: 13 jan. 2020.
- MITCHEL, M. B. **Machine learning**. New York: McGraw-Hill Science/Engineering/Math, 1997. 432 p.
- MNIH, V. *et al.* Playing Atari with deep reinforcement learning. *In*: CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS. Denver, 1 jan. 2013. p. 1-9.
- RASCHKA, S. **Python machine learning**. Birmingham: Packt Publishing, 2015.
- RUSSEL, S.; NORVIG, P. **Artificial Intelligence: a modern approach**. 3. ed. New Jersey: Pearson Education Inc., 2010. 1.132 p.
- SCHMIDHUBER, J. Deep learning in neural networks: an overview. **Neural Networks**, [s.l.]: Elsevier BV, v. 61, p. 85-117, jan. 2015. Disponível em: <http://dx.doi.org/10.1016/j.neunet.2014.09.003>. Acesso em: 13 jan. 2020.
- SMOLA, A.; VISHWANATHAN, S. V. N. **Introduction to machine learning**. New York, NY: Cambridge University Press, 2008. 226 p.

► Gabarito

Questão 1 – Resposta C

As camadas principais ilustradas na Figura 5 são, respectivamente, camada de entrada, camada oculta e camada de saída.

Questão 2 – Resposta D

Resolução:

- Um exemplo clássico de rede neural artificial é a rede do tipo *perceptron*, que utiliza a função **sinal** como função de decisão para gerar a resposta do sistema.
- Uma rede neural artificial precisa associar pesos referentes a cada uma das entradas do algoritmo, e esses pesos podem ser modificados para otimizar o resultado da RNA.
- Uma rede neural artificial **possui** uma arquitetura dinâmica, e por isso ela **é** indicada para a utilização em sistemas em que os dados a serem analisados estão em constante modificação.
- Uma rede neural funciona de maneira similar ao cérebro humano. Um neurônio em uma RNA é uma função matemática que coleta e classifica informações de acordo com sua arquitetura específica.
- A etapa de treinamento é fundamental para o cálculo dos pesos associados aos neurônios.

Questão 3 – Resposta A

Segundo Deng (2014), as redes de *deep learning* podem ser classificadas de diversas maneiras, porém, a mais comum é a utilização de três classes, baseadas na forma como as redes ou tecnologias são utilizadas, como, por exemplo, a utilização para síntese/geração ou então para reconhecimento/classificação. Com isso, as técnicas de *deep learning* podem ser divididas em **redes para aprendizado não supervisionado ou de geração, redes para aprendizado supervisionado e redes híbridas**.



Sistemas de recomendação

Autor: Lucas Claudino

► Objetivos

- Estudar as metodologias utilizadas em sistemas capazes de fazer recomendações a usuários.
- Compreender como os sistemas de recomendação criam algoritmos e fornecem a sugestão adequada.
- Estudar as aplicações básicas de sistemas de recomendação.

► 1. Introdução

A crescente necessidade de novas informações e dados é um cenário extremamente importante para a aplicação de algoritmos de *machine learning*. A partir disso, empresas, websites e usuários precisam utilizar metodologias capazes de entender as características e interações usuário-objeto para criar sugestões que estimulem ainda mais a busca por novos conteúdos.

Os sistemas de recomendação são baseados nessa necessidade. Eles buscam criar arquiteturas capazes de fazer essas recomendações com precisão, sabendo que o usuário a receber a sugestão irá, muito provavelmente, fazer uso dela. A seguir, você estudará as metodologias de criação de recomendações confiáveis e suas principais (e atuais) aplicações.

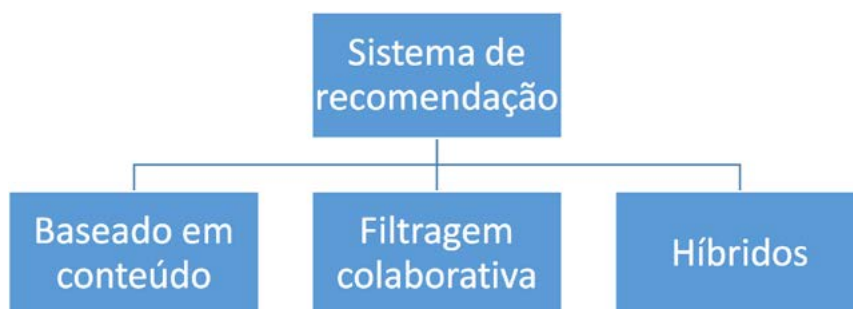
► 2. Sistemas de recomendação

O principal objetivo de um sistema de recomendação é sugerir itens relevantes aos usuários. Esses itens não precisam necessariamente ser físicos, podem ser uma recomendação de palavra a ser utilizada, música para ouvir, caminho a seguir e muitas outras situações possíveis.

Para realizar essa tarefa de recomendação, existem basicamente três metodologias: filtragem colaborativa, filtragem baseada em conteúdo e métodos híbridos. Os métodos colaborativos para sistemas de recomendação são baseados somente nas interações passadas do usuário com o sistema. Métodos baseados em conteúdo utilizam informações adicionais sobre os usuários e os itens passíveis de serem recomendados. A metodologia híbrida surge a partir de uma mistura das duas arquiteturas anteriores.

Observe a Figura 1 para assimilar os três métodos de sistemas de recomendação:

Figura 1 – Métodos utilizados em sistemas de recomendação



Fonte: elaborada pelo autor.

A seguir, você estudará as características básicas de cada uma dessas metodologias e como elas são aplicadas para a resolução de problemas.

2.1 Filtragem colaborativa

Os algoritmos de filtragem colaborativa fazem a recomendação baseados no histórico de preferências de cada usuário pelos itens disponíveis. Uma maneira de representar a preferência, ou afinidade, dos usuários pelos itens é por meio de uma matriz. Observe, a seguir, uma matriz usuário-item que relaciona cinco usuários e quatro itens, sendo que cada elemento p_{ij} é a preferência do usuário i pelo objeto j (DENG, 2019):

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \\ p_{51} & p_{52} & p_{53} & p_{54} \end{bmatrix}$$

As entradas da matriz podem ser tanto numéricas (a nota dada para uma comida pelo aplicativo de celular) como binárias (se o cliente comprou determinado produto ou clicou em algum link). A representação de P mostrada anteriormente possui dimensão 5×4 , porém, na realidade, essa matriz pode possuir milhares de linhas e colunas. Um exemplo seria o caso de uma grande empresa, como a Amazon, que necessita criar inúmeras matrizes relacionando seus usuários e os produtos disponíveis no estoque.

Para realizar a tarefa de recomendar o item mais provável, os algoritmos precisam ser capazes de fazer o tratamento das matrizes que relacionam usuários e itens. A seguir, você estudará três abordagens para essa tarefa: vizinho-próximo, fatoração de matrizes e *deep learning*.

2.1.1 Nearest-neighbor

O método do vizinho mais próximo, assim como na classificação *nearest-neighbor*, é baseado na similaridade entre pares de objetos ou usuários. Essa similaridade é medida por meio do uso da função cosseno:

$$\text{Similaridade}(x, y) = \cos(x, y) = \frac{x \cdot y}{\|x\| \|y\|}$$

Retomando-se, então, o conceito da matriz de proximidade, é possível escrevê-la em termos de itens (P_I) ou usuários (P_U):

$$P_I = [I_1 \quad I_2 \quad I_3 \quad I_4]$$

$$P_U = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \end{bmatrix}$$

Com isso, a similaridade entre dois itens é calculada como $\cos(I_1, I_2)$, e entre dois usuários, o cálculo é feito na forma $\cos(U_1, U_2)$.

As técnicas de aproximação entre itens (item-item) são comumente utilizadas por lojas virtuais, como a Amazon. Nesse caso, quando um usuário compra ou demonstra interesse por um produto, o algoritmo consegue rapidamente varrer o domínio de produtos similares e verificar qual é o item mais próximo àquele comprado. Com o resultado desse cálculo de preferência, a loja consegue indicar um produto que seja de interesse do usuário (LINDEN; SMITH; YORK, 2003).

Em alguns casos, a lista de itens similares pode ser muito pequena. Para essas situações, podem-se utilizar os “similares dos similares”, que são aqueles objetos que apresentam alguma similaridade com os inicialmente indicados pelo algoritmo.

Além disso, alguns algoritmos, como os utilizados pelo YouTube, executam um pós-processamento para ranquear a recomendação de acordo com algumas características, como qualidade do vídeo, diversidade e especificidades do usuário. Essa é uma técnica que pode dar mais força aos objetos a serem recomendados para o usuário, fazendo com que ele fique ainda mais propenso a assistir, comprar ou gostar da indicação (DAVIDSON *et al.*, 2010).

2.1.2 Fatoração de matrizes

A fatoração matricial consiste em uma técnica que trabalha a matriz de proximidade de tal forma a decompô-la em submatrizes de menor dimensão e mais fáceis de serem manipuladas. Essa técnica também possibilita que matrizes de dimensão $m \times n$, sendo m o número de usuários e n o número de itens, sejam decompostas em matrizes somente de usuários e outras somente com itens.

Uma técnica utilizada para fazer essa fatoração é a decomposição SVD (*singular value decomposition*), que decompõe a matriz de preferências $P_{(m \times n)}$ da seguinte maneira:

$$P_{m \times n} = U_{m \times m} \Sigma_{m \times n} I_{n \times n}$$

Sendo U e I matrizes unitárias e Σ uma matriz diagonal, em que suas entradas diagonais são os conhecidos valores singulares da matriz P (ALTER; BROWN; BOTSTEIN, 2000).

A decomposição matricial terá a seguinte forma:

$$P = \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} & u_{15} \\ u_{21} & u_{22} & u_{23} & u_{24} & u_{25} \\ u_{31} & u_{32} & u_{33} & u_{34} & u_{35} \\ u_{41} & u_{42} & u_{43} & u_{44} & u_{45} \\ u_{51} & u_{52} & u_{53} & u_{54} & u_{55} \end{bmatrix} \cdot \begin{bmatrix} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 \\ 0 & 0 & 0 & \sigma_4 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} i_{11} & i_{12} & i_{13} & i_{14} \\ i_{21} & i_{22} & i_{23} & i_{24} \\ i_{31} & i_{32} & i_{33} & i_{34} \\ i_{41} & i_{42} & i_{43} & i_{44} \end{bmatrix}$$

Sendo que os valores singulares são ordenados da seguinte forma: $\sigma_1 > \sigma_2 > \sigma_3 > \sigma_4$. E a preferência para o primeiro usuário é escrita da seguinte forma:

$$p_{11} = \sigma_1 u_{11} i_{11} + \sigma_2 u_{12} i_{21} + \sigma_3 u_{13} i_{31} + \sigma_4 u_{14} i_{41}$$

Portanto, como os valores singulares foram ordenados de forma crescente, o algoritmo poderá fazer uma aproximação da preferência para o primeiro usuário, utilizando somente σ_1 e σ_2 :

$$p_{11} = \sigma_1 u_{11} i_{11} + \sigma_2 u_{12} i_{21}$$


PARA SABER MAIS



As matrizes U e I podem ser facilmente calculadas quando se têm todos, ou quase todos, os elementos de P definidos. Porém, isso não é verdade para o caso de P ser uma matriz esparsa (pouco definida). Para esse caso, existe a técnica SVD de Simon Funk. Pesquise sobre ela para aprender como ela pode auxiliar na resolução de problemas com muitas incógnitas na matriz P .

2.1.3 Recomendação baseada em aprendizado profundo

O *deep learning* está sendo aplicado com sucesso a diversos problemas, como visão computacional e reconhecimento de voz, e tanto a indústria como as pesquisas educacionais estão comprovando a eficácia dessa técnica até mesmo para a solução de problemas complexos (COVINGTON; ADAMS; SARGIN, 2016). O aprendizado profundo conseguiu também revolucionar as arquiteturas de recomendação, pois consegue ultrapassar algumas barreiras dos modelos convencionais e, conseqüentemente, fornecer uma recomendação de melhor qualidade.



Segundo Zhang *et al.* (2019), os algoritmos de *deep learning*, sendo eles utilizados para sistemas de recomendação ou não, aprendem representações profundas sobre os dados, que são arquiteturas de várias camadas capazes de representar informações extraídas das observações. As redes neurais artificiais (RNAs) são as arquiteturas utilizadas para as tarefas de aprendizado profundo, e os algoritmos mais comuns são *Multilayer Perceptron*, *Autoencoder*, *Convolutional Neural Network*, *Adversarial Networks* e *Deep Reinforcement Learning*. Essas RNAs podem ser basicamente definidas da seguinte maneira:

- *Multilayer Perceptron* (MLP) é uma RNA retroalimentada com múltiplas camadas ocultas entre as entradas e saídas.
- *Autoencoder* (AE) é um modelo não supervisionado que busca reconstruir na camada de saída os dados que foram inseridos na entrada.
- *Convolutional Neural Network* (CNN) é um caso especial de RNA retroalimentada, com camadas ocultas convolucionais. Ela pode capturar traços locais e/ou globais e aumentar significativamente a eficiência e acurácia do resultado.
- *Adversarial Network* (AN) é uma rede generativa que consiste em uma rede geradora e uma discriminadora. As duas redes são treinadas simultaneamente e competem entre si em um jogo do tipo minimax¹.
- *Deep Reinforcement Learning* (DRL): uma RNA de reforço, ou seja, opera em um paradigma do tipo tentativa e erro, em que o reforço tem a finalidade de dizer se o resultado é o esperado.

¹ Definição: minimax, ou minmax, é uma regra de decisão muito utilizada em algoritmos de inteligência artificial, estatística e teoria da decisão, que tem como objetivo principal minimizar a perda possível para o pior caso. Ou seja, o algoritmo minmax busca minimizar a perda que inevitavelmente terá que acontecer no cenário em que for aplicado.



ASSIMILE

Os sistemas de recomendação baseados em arquiteturas de aprendizado profundo utilizam esses algoritmos para criar redes neurais que relacionam objetos e usuários e então poder recomendar os itens mais prováveis ao usuário de interesse. Essas redes neurais criadas substituem a matriz de preferências dos sistemas de recomendação baseados em decomposição matricial ou vizinho mais próximo.

► 3. Recomendação baseada em conteúdo

O site Pandora, que faz a disponibilização on-line de músicas para seus usuários, possui um time que rotula cada música com mais de 400 atributos. Com isso, quando um usuário seleciona uma estação musical, arquivos que se assemelhem à estação selecionada serão adicionados à *playlist*.

A recomendação baseada em conteúdo é isso. O algoritmo se baseia em alguma característica do objeto, ou usuário, para vasculhar seu banco de dados e poder, então, fazer a recomendação adequada. O exemplo das músicas do Pandora necessita que algum funcionário insira manualmente rótulos para cada uma das músicas. Porém, existem casos em que isso não é necessário, como, por exemplo, o sistema do site de empregos LinkedIn, em que os próprios usuários inserem suas características.

Uma maneira direta de relacionar usuários e objetos é a procura por palavras iguais ou compatíveis. Imagine o caso em que um empresário está procurando novos funcionários e cria, então, uma lista de pré-requisitos para cada uma das vagas ofertadas. O algoritmo de recomendação pode buscar nos currículos disponíveis palavras compatíveis com cada uma das qualificações necessárias e então recomendar aqueles profissionais que tiverem mais semelhança com a vaga.

Os algoritmos baseados em conteúdo, em geral, são rápidos e com resultados de fácil interpretação, além de poderem ser facilmente adaptados para novos itens ou usuários. Contudo, algumas características podem não ser passíveis de fácil captura ou interpretação.

A ideia central por trás de sistemas de recomendação baseados em conteúdo é criar um modelo a partir das características disponíveis para então explicar a interação entre usuário e objetos. A partir desse modelo, qualquer nova entrada pode receber uma recomendação baseada na análise de seus aspectos. Por exemplo: após criado um modelo que relaciona características de adolescentes e seus filmes favoritos, qualquer novo adolescente usuário que acessar o Netflix poderá receber uma recomendação de programação baseada em fatores como idade, sexo, país, etc.

ASSIMILE



Os métodos baseados em filtragem colaborativa fornecem recomendações a partir da análise da **interação** entre usuários e itens, ao passo que as arquiteturas baseadas em conteúdos criam as sugestões levando em consideração as **características** de usuários e objetos.

► 4. Metodologias híbridas

Métodos híbridos combinam informações de interações usuário-item e características de cada um. De acordo com WU *et al.* (2014), para determinar por qual empresa o usuário deve se interessar, o site LinkedIn utiliza um classificador regressor logístico criado a partir de um conjunto de atributos. A informação da filtragem colaborativa é inserida em uma propriedade, indicando se a companhia é similar àquelas as

quais você já segue. A informação do conteúdo inclui afirmações como “a localização bate com o desejado”, “a área de atuação da empresa é similar à sua” etc.

Modelos de aprendizado profundo podem ser bastante úteis para combinar informações providas de filtragens colaborativas e métodos baseados em conteúdo. O sistema de recomendação do YouTube construiu modelos de aprendizado profundo para prever vídeos a serem visualizados por usuários a partir de suas atividades anteriores e algumas características estáticas (gênero, idade, localidade etc.) Dado que redes neurais geralmente utilizam conjuntos de entrada de tamanho fixo, os vídeos e/ou *playlists*, assistidos pelos usuários são reduzidos e concatenados com outras características relevantes (COVINGTON; ADAMS; SARGIN, 2016).

ASSIMILE



Métodos híbridos podem se tornar dependentes em recomendações baseadas em conteúdo quando os usuários não possuem nenhum tipo de atividade (ou são muito pouco interativos).

TEORIA EM PRÁTICA



O aprendizado de máquina é uma técnica muito versátil e pode ser aplicada a inúmeros problemas. Você pode utilizar o AM para resolução desde uma simples regressão linear até uma complexa recomendação em tempo real de conteúdos e/ou produtos a serem utilizados por milhares de usuários.

Existem algumas maneiras clássicas de adicionar um sistema de recomendação ao seu software ou website e, geralmente, eles precisam implementar três tarefas básicas:

- Projetar e avaliar um modelo para recomendação.
- Agendar atualizações do sistema e direcionar dados, no sentido: usuário → modelo.
- Integrar o modelo de recomendação com o sistema comercial da empresa.

A Figura 2 contém um fluxograma que ilustra o processo e os blocos fundamentais para um sistema de recomendação poder ser implementado corretamente dentro de um cenário qualquer.

Figura 2 – Fluxograma de um sistema de recomendação



Fonte: elaborada pelo autor.

Você, com certeza, percebeu que estão faltando quatro nomes para os blocos contidos na Figura 2. A sua primeira tarefa é descobrir quais são esses blocos. Para isso, você deverá analisar a lista completa de elementos, que está desordenadamente escrita a seguir:

- a. Extrai atributos dos itens.
- b. Modelo de recomendação.
- c. Usuários.

- d. Interface de recomendação.
- e. Itens.
- f. Coleta de informações de comportamento de usuários e *ranking* de itens.
- g. Atributos dos itens e palavras-chave.
- h. Dados de usuários e ranking.
- i. Atualizações de modelos de recomendação.
- j. Avaliação de desempenho.

Após criado o modelo de recomendação e a metodologia do sistema, é necessário que o algoritmo consiga coletar constantemente dados sobre usuários, itens e a interação entre eles. Porém, neste momento, surge uma das principais complicações de um sistema de recomendação: **como coletar corretamente dados de inúmeros usuários para realizar recomendações precisas?**

Existem duas alternativas: a primeira é pedir para os usuários avaliarem cada item com o qual eles interagem, mas quase nunca o usuário deseja realmente fazer essa avaliação, e isso pode incomodá-lo. A segunda opção é extrair conclusões a partir do comportamento do usuário, mas com isso surge o problema "*cold start*", que é quando o algoritmo não possui dados confiáveis para começar as recomendações.

O seu trabalho agora é pesquisar e fornecer algumas possíveis soluções para esses problemas. Pois somente solucionando-os é que o sistema de recomendação poderá gerar sugestões efetivas para os usuários.

Bom trabalho.



VERIFICAÇÃO DE LEITURA

1. Os sistemas de recomendação possuem como objetivo principal elaborar uma recomendação, ou seja, sugerir itens relevantes aos usuários. Esses itens não precisam, necessariamente, ser físicos, eles podem ser algum tipo de recomendação de palavra a ser utilizada em uma mensagem de texto, música para ouvir, melhor caminho a seguir e muitas outras situações possíveis.

Assinale a alternativa que contém as três principais metodologias utilizadas em sistemas de recomendação.

- a. Direta; indireta; cruzada.
 - b. Direta; indireta; híbrida.
 - c. Filtragem passa banda; baseada em conteúdo; híbrida.
 - d. Filtragem colaborativa; baseada em conteúdo; híbrida.
 - e. Filtragem colaborativa; baseada em interação; híbrida.
2. Os métodos híbridos para sistemas de recomendação são constituídos por algoritmos capazes de relacionar interações e características, por isso eles recebem esse nome.

Assinale a alternativa que apresenta uma desvantagem do método híbrido para quando o usuário possui nenhuma ou pouca interação com o sistema de recomendação.

- a. Nesse caso, o método híbrido pode se tornar dependente de recomendações baseadas em conteúdo.

- b. Nesse caso, o método híbrido pode se tornar dependente de recomendações baseadas em interações.
 - c. Nesse caso, o método híbrido pode apresentar uma recomendação enviesada pela última interação feita pelo usuário.
 - d. Nesse caso, o método híbrido não conseguirá fazer uma recomendação para o usuário, pois não há nenhuma interação registrada.
 - e. Nesse caso, o sistema de recomendação precisa fazer sua primeira sugestão para o usuário como sendo uma recomendação aleatória, pois só assim será possível garantir a imparcialidade do algoritmo.
3. Existem diversas metodologias utilizadas para a criação de sistemas de recomendação. Um método bastante utilizado são os sistemas de filtragem colaborativa que utilizam arquiteturas de aprendizado profundo (*deep learning*). Sobre esse método, analise o trecho a seguir:

“Os sistemas de recomendação baseados em arquiteturas de aprendizado profundo utilizam esses algoritmos para criar redes neurais que relacionam objetos e usuários e então poder recomendar os itens mais prováveis ao usuário de interesse. Essas redes neurais criadas substituem _____ dos sistemas de recomendação baseados em decomposição matricial ou vizinho mais próximo”.

Assinale a alternativa que preenche corretamente a lacuna.

- a. Os usuários.
- b. Os objetos.
- c. A matriz de preferências.
- d. A matriz de autovalores.
- e. A matriz de autovetores.

Referências bibliográficas

ALTER, O.; BROWN, P. O.; BOTSTEIN, D. Singular value decomposition for genome-wide expression data processing and modeling. **Proceedings of the National Academy of Sciences**, [s.l.], v. 97, n. 18, p. 10.101-10.106, 29 ago. 2000. Disponível em: <http://dx.doi.org/10.1073/pnas.97.18.10101>. Acesso em: 13 jan. 2020.

COVINGTON, P.; ADAMS, J.; SARGIN, E. Deep neural networks for YouTube recommendations. **Proceedings of the 10th Acm Conference on Recommender Systems – Recsys '16**, [s.l.]: ACM Press, p. 191-198, 2016. Disponível em: <http://dx.doi.org/10.1145/2959100.2959190>. Acesso em: 13 jan. 2020.

DAVIDSON, J. *et al.* The YouTube video recommendation system. **Proceedings of the Fourth Acm Conference on Recommender Systems – Recsys '10**, [s.l.]: ACM Press, p. 293-296, 2010. Disponível em: <http://dx.doi.org/10.1145/1864708.1864770>. Acesso em: 13 jan. 2020.

DENG, H. **Recommender Systems in Practice**: how companies make product recommendations. 2019. Disponível em: <https://towardsdatascience.com/recommender-systems-in-practice-cef9033bb23a>. Acesso em: 22 set. 2019.

FACELI, K.; LORENA, A. C.; GAMA, J.; CARVALHO, A. C. P. L. F. **Inteligência artificial**: uma abordagem de aprendizado de máquina. São Paulo: LTC Editora, 2011.

GORMLEY, M. **Matrix factorization and collaborative filtering**. 2017. Disponível em: <https://www.cs.cmu.edu/~mgormley/courses/10601-s17/slides/lecture25-mf.pdf>. Acesso em: 26 set. 2019.

LINDEN, G.; SMITH, B.; YORK, J. Amazon.com recommendations: item-to-item collaborative filtering. **IEEE Internet Computing**, [s.l.]: Institute of Electrical and Electronics Engineers (IEEE), v. 7, n. 1, p. 76-80, jan. 2003. Disponível em: <http://dx.doi.org/10.1109/mic.2003.1167344>. Acesso em: 13 jan. 2020.

RASCHKA, S. **Python machine learning**. Birmingham: Packt Publishing, 2015.

SMOLA, A.; VISHWANATHAN, S. V. N. **Introduction to machine learning**. New York, NY: Cambridge University Press, 2008. 226 p.

WU, L. *et al.* The browsermaps: collaborative filtering at LinkedIn. **Proceedings of the 6th Workshop on Recommender Systems and the Social Web**, Foster City, p. 1-8, 2014.

ZHANG, S. *et al.* Deep Learning Based Recommender System. **Acm Computing Surveys**, [s.l.]: Association for Computing Machinery (ACM), v. 52, n. 1, p. 1-38, 25 fev. 2019. Disponível em: <http://dx.doi.org/10.1145/3285029>. Acesso em: 13 jan. 2020.

Gabarito

Questão 1 – Resposta D

Para realizar essa tarefa de recomendação, existem basicamente três metodologias: filtragem colaborativa, filtragem baseada em conteúdo e métodos híbridos. Os métodos colaborativos para sistemas de recomendação são baseados somente nas interações passadas do usuário com o sistema. Métodos baseados em conteúdo utilizam informações adicionais sobre os usuários e os itens passíveis de serem recomendados. A metodologia híbrida surge a partir de uma mistura das duas arquiteturas anteriores.

Questão 2 – Resposta A

Métodos híbridos podem se tornar dependentes em recomendações baseadas em conteúdo quando os usuários não possuem nenhum tipo de atividade (ou são muito pouco interativos).

Questão 3 – Resposta C

Os sistemas de recomendação baseados em arquiteturas de aprendizado profundo utilizam esses algoritmos para criar redes neurais que relacionam objetos e usuários e então poder recomendar os itens mais prováveis ao usuário de interesse. Essas redes neurais criadas substituem **a matriz de preferências** dos sistemas de recomendação baseados em decomposição matricial ou vizinho mais próximo.



Bons estudos!

