Estratégia Concursos

Aula 00

SEFAZ-MG (Auditor Fiscal - Tecnologia da Informação) Passo de Ciências de Dados

Autor:

Thiago Rodrigues Cavalcanti

09 de Fevereiro de 2023

APRENDIZADO SUPERVISIONADO

Sumário

Análise Estatística 2
Roteiro de revisão e pontos do assunto que merecem destaque 3
Dicionário 3
Aprendizado de máquina 5
Tipos de Sistemas de Aprendizagem de Máquina 6
Aprendizagem supervisionada/não supervisionada 7
Aprendizagem supervisionada 7
Aprendizado não supervisionado 8
Aprendizado semisupervisionado 10
Aprendizado de reforço 11
Validação e avaliação de modelos preditivos 12
Treinamento, Validação e Teste 19
Underfitting, overfitting e técnicas de regularização 20
Parâmetros x hiperparâmetros 25
Classificação 38
Aposta estratégica 39
Questões estratégicas 45
Questionário de revisão e aperfeiçoamento 56
Perguntas 56
Perguntas com respostas 57

ANÁLISE ESTATÍSTICA

Começaremos nosso curso analisando a quantidade de questões disponíveis da FGV sobre o conteúdo de Ciência de Dados. Os tópicos mais relevantes estão presentes nessa aula, na minha opinião por dois motivos: (1) são os tópicos introdutórios (logo são os primeiros que a banca olha quando começa a fazer questões sobre o assunto e (2) são tópicos recorrentes (o conhecimento sobre eles será retomado em outros pontos do assunto).

Desta forma , convém destacar os percentuais de incidência de todos os assuntos previstos no nosso curso – quanto maior o percentual de cobrança de um dado assunto, maior sua importância:

Tópico	%
Aprendizado de máquina supervisionado (técnicas de classificação e técnicas de regressão)	6%
Aprendizado não supervisionado (técnicas de agrupamento, redução de dimensionalidade, técnicas de associação)	2%
Sistemas de recomendação.	0%
Processamento de linguagem natural (PLN).	4%
Visão computacional e Deep Learning.	2%
Big Data. Fundamentos. Tipos de dados: estruturados, semiestruturados e não estruturados. Conceito dos cinco Vs. Fluxo de big data: ingestão, processamento e disponibilização. Armazenamento de big data. Pipeline de dados.	2%
Conceito de data lake. ETL X ELT. Soluções de big data. Arquiteturas de big data.	2%
Tratamento de dados. Normalização numérica. Discretização. Tratamento de dados ausentes. Tratamento de outliers e agregações.	2%
Ingestão de dados. Conceito de ingestão de dados. Ingestão de dados estruturados, semiestruturados e não estruturados. Ingestão de dados em lote (batch). Ingestão de dados em streaming. Ingestão de dados full × incremental. Ingestão de dados CDC (change data capture).	2%
Processamento de dados. Conceitos de processamento massivo e paralelo. Processamento em lote (batch). Processamento em tempo real (real time). Processamento MapReduce.	2%
Linguagens de programação: R.	8%
Linguagens de programação: Python.	59%
Frameworks: Scala. Java. Spark (PySpark, Scala e Java). Pandas. Scikit-learn. TensorFlow. PyTorch. Keras. NLTK.	8%
Qualidade de dados. Conceitos e definições. Dimensões da qualidade de dados (visão DMBOK). Principais técnicas em qualidade de dados. Profiling. Matching. Deduplicação. Data cleansing. Enriquecimento. Boas práticas para adoção da qualidade de dados. Processos de qualidade para modelos de dados.	2%
	100%

ROTEIRO DE REVISÃO E PONTOS DO ASSUNTO QUE MERECEM DESTAQUE

A ideia desta seção é apresentar um roteiro para que você realize uma revisão completa do assunto e, ao mesmo tempo, destacar aspectos do conteúdo que merecem atenção.

Para revisar e ficar bem-preparado no assunto, você precisa, basicamente, seguir os seguintes tópicos:

Dicionário

Faremos uma lista de termos que são relevantes ao entendimento do assunto desta aula! Se durante sua leitura texto, você tenha alguma dúvida sobre conceitos básicos, esta parte da aula pode ajudar a esclarecer.

Aprendizado de Máquina

Agente - No aprendizado por reforço, a entidade que usa uma política para maximizar o retorno esperado obtido com a transição entre os estados do ambiente.

Árvore de decisão - é um algoritmo de aprendizado de máquina supervisionado que é utilizado para classificação e para regressão. Ela estabelece "nós" (*decision nodes*) que se relacionam entre si por uma hierarquia. Existe o nó-raiz (root node), que é o mais importante, e os nós-folha (leaf nodes), que são os resultados ou classes. No contexto de *machine learning*, o raiz é um dos atributos da base de dados e o nó-folha é a classe ou o valor que será gerado como resposta.

AUC (*Area under the ROC Curve*) - Uma métrica de avaliação que considera todos os limites de classificação possíveis. A área sob a curva ROC é a probabilidade de um classificador estar mais confiante de que um exemplo positivo escolhido aleatoriamente seja realmente positivo e de que um exemplo negativo escolhido aleatoriamente seja negativo.

Boosting - Uma técnica de aprendizado de máquina que combina iterativamente um conjunto de classificadores simples e não muito precisos (chamados de classificadores "fracos") em um classificador com alta precisão (um classificador "forte"), valorizando os exemplos que o modelo está classificando incorretamente.

Centroide - O centro de um cluster determinado por um algoritmo k-means ou k-median. Por exemplo, se k é 3, então o algoritmo k-means ou k-median encontra 3 centróides.

Classificador binário - Um tipo de tarefa de classificação que gera uma das duas classes mutuamente exclusivas. Por exemplo, um modelo de aprendizado de máquina que avalia mensagens de e-mail e gera "spam" ou "não spam" é um classificador binário.

Clustering - Agrupar exemplos relacionados, particularmente durante a aprendizagem não supervisionada. Uma vez que todos os exemplos são agrupados, um humano pode opcionalmente fornecer significado a cada cluster.

Cross-validation - Um mecanismo para estimar quão bem um modelo será generalizado para novos dados, testando o modelo em relação a um ou mais subconjuntos de dados não sobrepostos retidos do conjunto de treinamento.

Detecção de anomalia - O processo de identificação de outliers. Por exemplo, se a média de um determinado recurso for 100 com um desvio padrão de 10, a detecção de anomalias deverá sinalizar um valor de 200 como suspeito.

Dropout - Uma forma de regularização útil no treinamento de redes neurais. A regularização de dropout funciona removendo uma seleção aleatória de um número fixo de unidades em uma camada de rede para uma única etapa de gradiente.

Early stopping - Um método para regularização que envolve encerrar o treinamento do modelo antes que a perda de treinamento termine de diminuir. Na parada antecipada, você termina o treinamento do modelo quando a perda em um conjunto de dados de validação começa a aumentar, ou seja, quando o desempenho da generalização piora.

Função de ativação - Uma função (por exemplo, ReLU ou sigmoid) que recebe a soma ponderada de todas as entradas da camada anterior e, em seguida, gera e passa um valor de saída (normalmente não linear) para a próxima camada.

Função convexa - Uma função na qual a região acima do gráfico da função é um conjunto convexo. A função convexa típica tem a forma da letra U.

Gradiente - O vetor de derivadas parciais em relação a todas as variáveis independentes. No aprendizado de máquina, o gradiente é o vetor de derivadas parciais da função do modelo. O gradiente aponta na direção de subida/decida mais íngreme.

Linha de base - Um modelo usado como ponto de referência para comparar o desempenho de outro modelo (normalmente, um mais complexo). Por exemplo, um modelo de regressão logística pode servir como uma boa linha de base para um modelo de aprendizado profundo.

Matriz de confusão - Uma tabela NxN que resume o sucesso das previsões de um modelo de classificação; ou seja, a correlação entre o rótulo e a classificação do modelo. Um eixo de uma matriz de confusão é o rótulo que o modelo previu e o outro eixo é o rótulo real. N representa o número de classes. Em um problema de classificação binária, N=2.

Rede neural - Um modelo que, inspirando-se no cérebro, é composto por camadas (sendo pelo menos uma oculta) constituídas por unidades simples ou neurónios ligados seguidos de não linearidades.

Retropropagação - O algoritmo primário para executar gradiente descendente em redes neurais. Primeiro, os valores de saída de cada nó são calculados (e armazenados em cache) em uma

passagem direta. Em seguida, a derivada parcial do erro em relação a cada parâmetro é calculada em uma passagem para trás pela rede.

Viês (bias/math) - Uma interceptação ou deslocamento de uma origem. O viés (também conhecido como termo de viés) é chamado de b ou w0 em modelos de aprendizado de máquina.

Aprendizado de máquina

Vamos começar nossa revisão com uma simples pergunta ... você sabe a definição de aprendizado de máquina? ... **Machine Learning** é a ciência (e a arte) da programação de computadores para que eles possam *aprender com os dados*. Vejamos uma definição um pouco mais genérica:

[Machine Learning é o] campo de estudo que dá aos computadores a capacidade de aprender sem serem explicitamente programados.

Arthur Samuel, 1959

E um mais orientado para a engenharia:

Diz-se que um programa de computador aprende com a experiência E em relação a alguma tarefa T e alguma medida de desempenho P, se seu desempenho em T, medido por P, melhora com a experiência E.

Tom Mitchell, 1997

Falando de maneira mais prática ... um filtro de spam é um programa de Machine Learning que, dado exemplos de e-mails de spam (por exemplo, sinalizados pelos usuários) e exemplos de e-mails regulares (não-spam), podem aprender a sinalizar spam. Os exemplos que o sistema usa para aprender são chamados de ***conjunto de treinamento***. Cada exemplo de treinamento é chamado de ***instância de treinamento*** **(ou *amostra*)**. Neste caso, **a tarefa *T* é sinalizar spam** para novos e-mails, a ***experiência E*** são os ***dados de treinamento*** e a **medida *de desempenho P*** precisa ser definida; por exemplo, você pode usar a proporção de e-mails corretamente classificados. Esta medida de desempenho em particular é chamada ***de precisão***, e muitas vezes é usada em tarefas de classificação.

Neste sentido, podemos afirmar que o uso de Aprendizado de Máquina pode ser útil para diversos tipos de problemas. Dentre estes podemos destacar:

- Problemas para os quais as soluções existentes requerem um monte de ajustes finos ou longas listas de regras: um algoritmo de Machine Learning pode muitas vezes simplificar o código e ter um desempenho melhor do que a abordagem tradicional.

- Problemas complexos para os quais usar uma abordagem tradicional não produzem uma boa solução: as melhores técnicas de Machine Learning talvez possam encontrar uma solução.

- Ambientes fluidos: um sistema de Machine Learning pode se adaptar a novos dados.

- Obtendo insights sobre problemas complexos e grandes quantidades de dados.

Tipos de Sistemas de Aprendizagem de Máquina

Existem tantos tipos diferentes de sistemas de Machine Learning que é útil classificá-los em categorias amplas, com base nos seguintes critérios:

- Sejam eles treinados ou não exemplos já classificados (supervisionados, não supervisionados, semisupervisionados e aprendizado de reforço)
- Se eles podem ou não aprender incrementalmente (on-line versus aprendizado em lote)
- Se eles funcionam simplesmente comparando novos pontos de dados com pontos de dados conhecidos, ou, em vez disso, detectando padrões nos dados de treinamento e construindo um modelo preditivo, assim como os cientistas fazem (aprendizado baseado em instância versus baseado em modelo)

Esses critérios não são exclusivos; você pode combiná-los da maneira que quiser. Por exemplo, um filtro de spam de última geração pode aprender em tempo real usando um modelo de rede neural profunda treinado usando exemplos de spam e não spam; isso o torna um sistema de

aprendizagem on-line, baseado em modelos e supervisionado. Vamos olhar cada um desses critérios um pouco mais de perto.

Aprendizagem supervisionada/não supervisionada

Os sistemas de Machine Learning podem ser classificados de acordo com a quantidade e o tipo de supervisão que recebem durante o treinamento. São quatro categorias principais: aprendizagem supervisionada, aprendizagem não supervisionada, aprendizagem semisupervisionada e Aprendizado de Reforço.

Aprendizagem supervisionada

No *aprendizado supervisionado*, o conjunto de treinamento que você alimenta para o algoritmo inclui as soluções desejadas, *chamadas rótulos*.

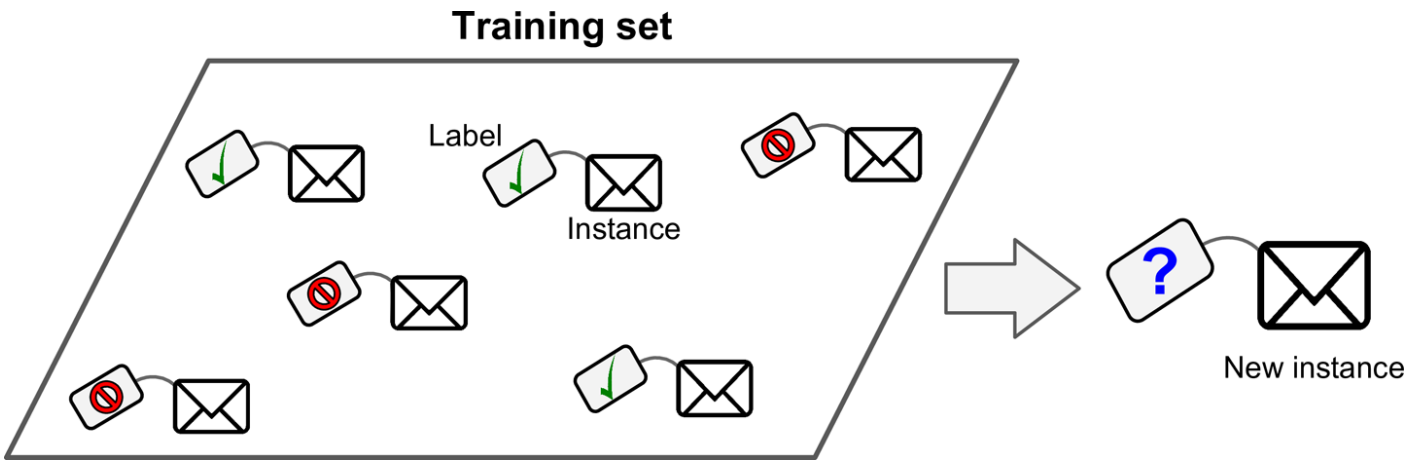


Figura 1 - Um conjunto de treinamento rotulado para classificação de spam (um exemplo de aprendizado supervisionado)

Uma tarefa típica de aprendizagem supervisionada é *a* ***classificação***. O filtro de spam é um bom exemplo disso: ele é treinado com muitos e-mails de exemplo junto com suas respectivas *classes* (spam ou *ham*), e deve aprender a classificar novos e-mails.

Outra tarefa típica é prever um valor numérico *de destino*, como o preço de um carro, dado um conjunto de *características* (quilometragem, idade, marca, etc.) chamados *preditores*. Esse tipo de tarefa é chamada *de* ***regressão***. Para treinar o sistema, você precisa dar-lhe muitos exemplos de carros, incluindo tanto seus preditores quanto suas etiquetas ou rótulos (ou seja, seus preços).

Observe que alguns algoritmos de regressão também podem ser usados para classificação, e vice-versa. Por exemplo, *a Regressão Logística* é comumente utilizada para classificação, pois pode produzir um valor que corresponde à probabilidade de pertencer a uma determinada classe (por exemplo, 20% de chance de ser spam).

Aqui estão alguns dos mais importantes algoritmos de aprendizagem supervisionados (abordados neste livro):

- k-Vizinhos mais próximos (KNN)
- Regressão Linear
- Regressão Logística
- Máquinas de vetores de suporte (SVMs)
- Árvores de decisão e Florestas Aleatórias
- Redes neurais

Aprendizado não supervisionado

Em *aprendizado não supervisionado*, como você pode imaginar, os dados de treinamento não são rotulados. O sistema tenta aprender sem um professor.

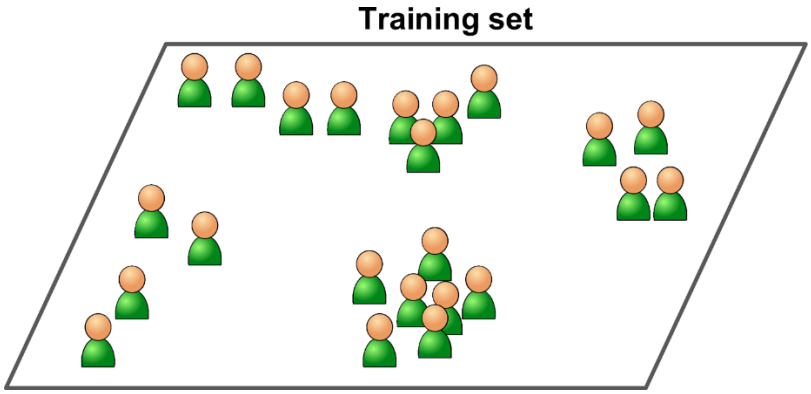


Figura 2 - Um conjunto de treinamento sem rótulo para aprendizado não supervisionado

Aqui estão alguns dos mais importantes algoritmos de aprendizagem não supervisionados:

- Clustering
  - K-Means
  - DBSCAN
  - Análise hierárquica de cluster (HCA)
- Detecção de anomalias e detecção de novidades
  - SVM de uma classe
  - Floresta de Isolamento
- Visualização e redução de dimensionalidade
  - Análise de componentes principais (PCA)
  - Kernel PCA
  - Incorporação linear local (LLE)
  - t-Distributed Stochastic Neighbor Embedding (t-SNE)
- Aprendizagem de regras da associação
  - Apriori
  - Eclat
  - Por amostragem
  - Árvore de Padrão-Frequente

Por exemplo, digamos que você tenha um monte de dados sobre os visitantes do seu blog. Você pode querer executar um algoritmo *de clustering* para tentar detectar grupos de visitantes semelhantes. Em nenhum momento você diz ao algoritmo a qual grupo um visitante pertence: ele encontra essas conexões sem a sua ajuda. Por exemplo, pode notar que 40% dos seus visitantes são homens que amam histórias em quadrinhos e geralmente leem seu blog à noite, enquanto 20% são jovens amantes de ficção científica que visitam durante os fins de semana. Se você usar um algoritmo *hierárquico de clustering*, ele também pode subdividir cada grupo em grupos menores. Isso pode ajudá-lo a direcionar seus posts para cada grupo.

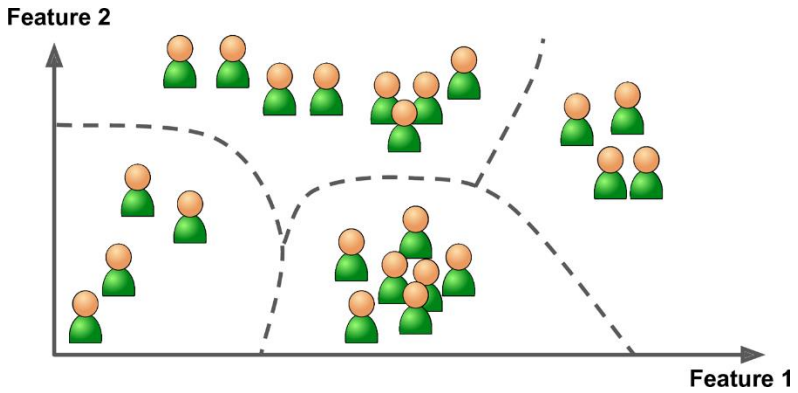


Figura 3 - Clustering

Algoritmos de visualização também são bons exemplos de algoritmos de aprendizagem não supervisionados: você os alimenta com muitos dados complexos e não rotulados, e eles fazem uma representação 2D ou 3D de seus dados que podem ser facilmente plotados. Esses algoritmos tentam preservar o máximo de estrutura possível (por exemplo, tentando evitar que os clusters separados no espaço de entrada se sobreponham na visualização) para que você possa entender como os dados são organizados e talvez identificar padrões insuspeitável.

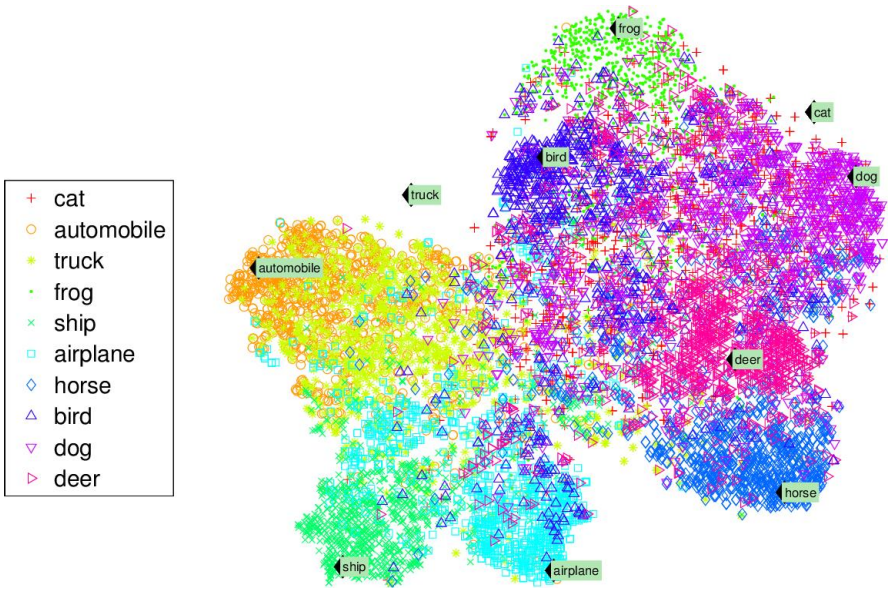


Figura 4 - Exemplo de visualização t-SNE destacando clusters semânticos

Uma tarefa relacionada é a *redução da dimensionalidade*, na qual o objetivo é simplificar os dados sem perder muitas informações. Uma maneira de fazer isso é fundir várias características correlacionadas em uma. Por exemplo, a quilometragem de um carro pode estar fortemente correlacionada com sua idade, de modo que o algoritmo de redução de dimensionalidade irá mesclá-los em uma característica que representa o desgaste do carro. Isso é chamado *de extração de recursos*.

Curiosidade: Muitas vezes é uma boa ideia tentar reduzir a dimensão de seus dados de treinamento usando um algoritmo de **redução de dimensionalidade** antes de alimentá-lo para outro algoritmo de Machine Learning (como um algoritmo de aprendizagem supervisionado). Ele será executado muito mais rápido, os dados ocuparão menos espaço em disco e memória, e em alguns casos também podem ter um desempenho melhor.

Outra tarefa não supervisionada comum é o *aprendizado de regras de associação*, no qual o objetivo é cavar grandes quantidades de dados e descobrir relações interessantes entre atributos. Por exemplo, suponha que você tenha um supermercado. Executar uma regra de associação em seus registros de vendas pode revelar que as pessoas que compram molho de churrasco e batatas fritas também tendem a comprar bife. Assim, você pode querer colocar esses itens perto um do outro.

Aprendizado semisupervisionado

Uma vez que rotular dados geralmente é demorado e caro, muitas vezes você terá muitas instâncias não rotuladas, e poucas instâncias rotuladas. Alguns algoritmos podem lidar com dados que são parcialmente rotulados. Isso é chamado *de aprendizagem semisupervisionados*.

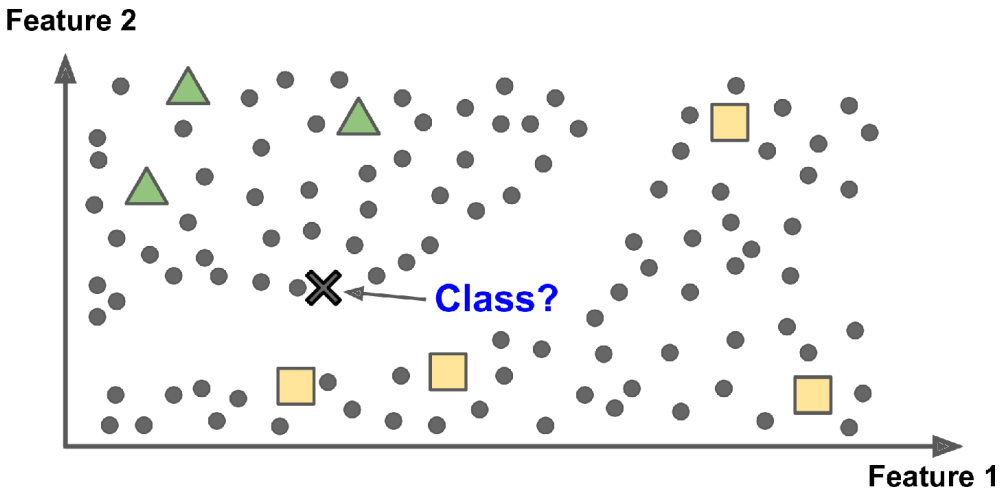


Figura 5 - Aprendizado semisupervisionado com duas classes (triângulos e quadrados): os exemplos não rotulados (círculos) ajudam a classificar uma nova instância (a cruz) para a classe triângulo em vez da classe quadrada, mesmo estando mais perto dos quadrados r

Alguns serviços de hospedagem de fotos, como o Google Fotos, são bons exemplos disso. Uma vez que você envia todas as fotos de sua família para o serviço, ele reconhece automaticamente que a mesma pessoa A aparece nas fotos 1, 5 e 11, enquanto outra pessoa B aparece nas fotos 2,

5 e 7. Esta é a parte não supervisionada do algoritmo (clustering). Agora tudo o que o sistema precisa é que você diga quem são essas pessoas. Basta adicionar um rótulo por pessoa e é capaz de nomear todos em cada foto, o que é útil para pesquisar fotos.

A maioria dos algoritmos de aprendizagem semisupervisionados são combinações de algoritmos não supervisionados e supervisionados. Por exemplo, *redes de crenças profundas* (DBNs) são baseadas em componentes não supervisionados chamados máquinas *Boltzmann restritas* (RBMs) empilhadas umas nas outras. Os RBMs são treinados sequencialmente de forma não supervisionada, e então todo o sistema é ajustado usando técnicas de aprendizagem supervisionadas.

Aprendizado de reforço

O Aprendizado de Reforço é usa um sistema diferente. O sistema de aprendizagem, chamado de ***agente*** neste contexto, pode observar o **ambiente**, selecionar e executar ações e receber ***recompensas*** em troca (ou ***penalidades*** sob a forma de recompensas negativas. Ele deve então aprender por si mesmo qual é a melhor estratégia, chamada *de **política***, para obter a maior recompensa ao longo do tempo. Uma política define qual ação o agente deve escolher quando está em uma determinada situação.

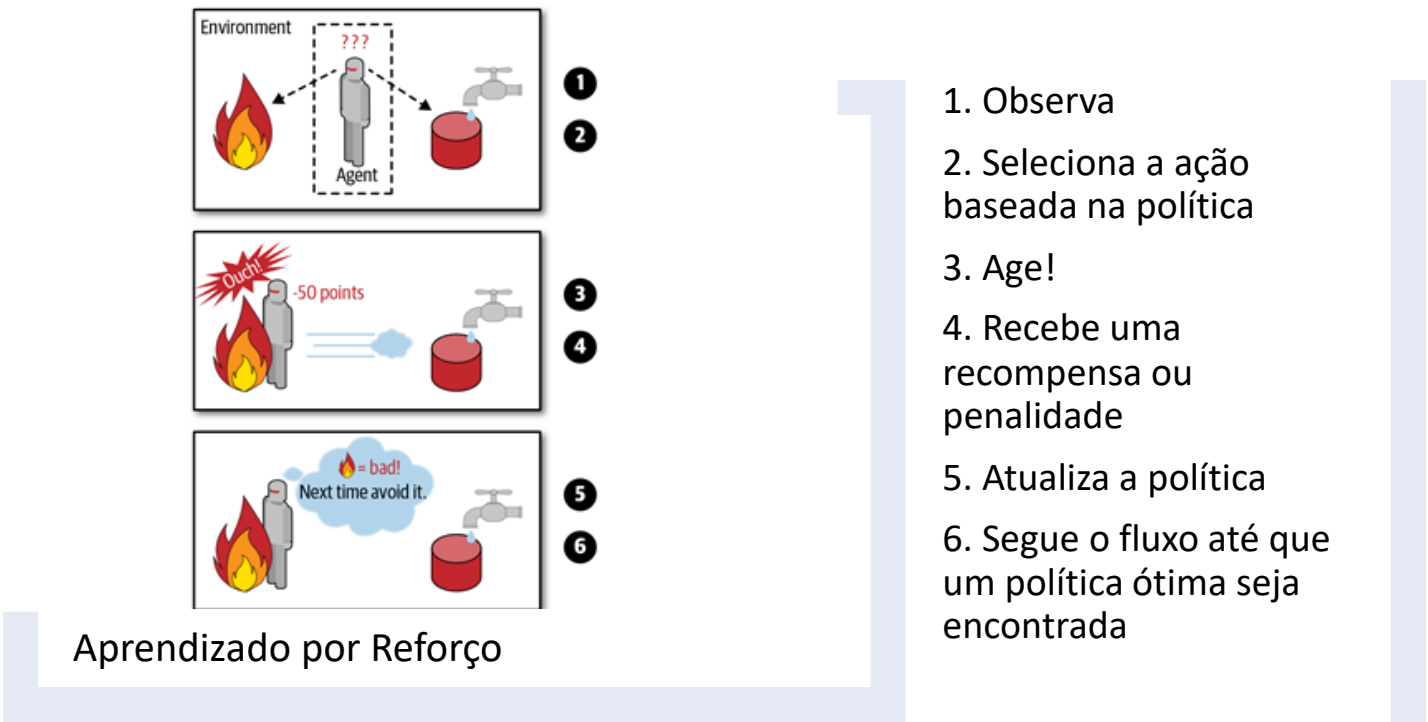


Figura 6 - Aprendizado de reforço

Por exemplo, muitos robôs implementam algoritmos de Aprendizagem de Reforço para aprender a andar. O programa AlphaGo do DeepMind também é um bom exemplo de Aprendizado de Reforço: ele fez as manchetes em maio de 2017, quando venceu o campeão mundial Ke Jie no jogo de Go. Aprendeu sua política vencedora analisando milhões de jogos e, em seguida, jogando muitos jogos

contra si mesmo. Note que o aprendizado foi desligado durante os jogos contra o campeão; AlphaGo estava apenas aplicando a política que tinha aprendido.

Resumindo os três tipos de aprendizado, podemos construir o seguinte esquema:

Validação e avaliação de modelos preditivos

Depois do treinamento do modelo, faz-se necessário avaliar se o resultado é adequado o suficiente para ser colocado em produção. Nas próximas linhas, iremos apresentá-lo à métodos de avaliação de modelo, onde avaliamos o desempenho de cada modelo que treinamos antes de decidir colocá-lo em produção. *Ao final desta seção, você será capaz de criar um conjunto de dados de avaliação. Você estará preparado para avaliar o desempenho dos modelos de regressão linear usando* o erro médio absoluto *e o* erro quadrático médio. Você também poderá avaliar o desempenho dos modelos de regressão logística ou classificadores binários usando as métricas acurácia, precisão, recall e F1 Score.

Avaliando o desempenho do modelo para modelos de regressão

Ao criar um modelo de regressão, você cria um modelo que prevê uma variável numérica contínua. Ao separar seu conjunto de dados de avaliação (teste), você tem algo que pode usar para comparar a qualidade do seu modelo.

O que você precisa fazer para avaliar a qualidade do seu modelo é comparar a qualidade da sua previsão com o que é chamado de verdade fundamental, que é o valor real observado que você está tentando prever. Dê uma olhada na tabela abaixo, na qual a primeira coluna contém a verdade fundamental (chamada de actuals) e a segunda coluna contém os valores previstos:

	Atual	*Valores Previstos*
0	4.891	4.132270
1	4.194	4.364320

2	4.984	4.440703
3	3.109	2.954363
4	5.115	4.987951

A *linha 0* na saída compara o valor real em nosso conjunto de dados de avaliação com o que nosso modelo previu. O valor real do nosso conjunto de dados de avaliação é 4,891. O valor que o modelo previu é 4,132270.

A linha 1 compara o valor real de 4,194 com o que o modelo previu, que é 4,364320.

Na prática, o conjunto de dados de avaliação conterá muitos registros, portanto, você não fará essa comparação visualmente. Em vez disso, você fará uso de algumas equações.

Você deve fazer essa comparação calculando a perda (loss). A perda é a diferença entre os valores reais e previstos da tabela anterior. Na mineração de dados, é chamada de medida de distância. Existem várias abordagens para calcular medidas de distância que dão origem a diferentes funções de perda. Duas delas são:

- Distância de manhattan
- Distância euclidiana

Existem várias funções de perda para regressão, veremos duas das funções de perda comumente usadas para regressão, que são:

- Erro médio absoluto (MAE - Mean absolute error) - é baseado na distância de Manhattan
- Erro quadrático médio (MSE - Mean squared error) - é baseado na distância euclidiana

O objetivo dessas funções é medir a utilidade de seus modelos, fornecendo a você um valor numérico que mostra quanto de desvio existe entre as verdades fundamentais e os valores previstos pelos seus modelos.

Sua missão é treinar novos modelos com erros consistentemente menores.

A pontuação R^2 (pronuncia-se "r quadrado") às vezes é chamada de "pontuação" e mede o coeficiente de determinação do modelo. Pense nisso como a capacidade do modelo de fazer previsões boas e confiáveis. Essa medida é acessada usando o método score() do modelo de regressão da biblioteca do Scikitlearn.

Seu objetivo é treinar modelos sucessivos com objetivo de obter a pontuação mais alta de R^2. Os valores de R^2 variam entre 0 e 1. Seu objetivo é tentar fazer com que o modelo tenha uma pontuação próxima a 1.

O erro médio absoluto (EMA) é uma métrica de avaliação para modelos de regressão que mede a distância absoluta entre suas previsões e a verdade fundamental. A distância absoluta é a distância independentemente do sinal, seja positivo ou negativo. Por exemplo, se a valor real for

6 e você predizer 5, a distância será 1. No entanto, se você predisser 7, a distância será -1. A distância absoluta, sem levar em consideração os sinais, é 1 em ambos os casos. Isso é chamado de magnitude. O EMA é calculado somando todas as magnitudes e dividindo pelo número de observações.

O EMA é calculado subtraindo todas as previsões da verdade fundamental, encontrando o valor absoluto, somando todos os valores absolutos e dividindo pelo número de observações. Esse tipo de medida de distância é chamado de distância de Manhattan na mineração de dados.

O erro quadrático médio (EQM) é calculado tomando os quadrados das diferenças entre os valores reais e as previsões, somando-as e dividindo pelo número de observações. O EQM é grande e, às vezes, a raiz quadrada deste valor é usada, que é a raiz do erro quadrático médio (REQM).

O erro logarítmico médio quadrático (ELMQ) introduz logaritmos na equação adicionando um ao valor real e à previsão antes de tomar os logaritmos, depois elevar ao quadrado as diferenças, somá-las e dividir pelo número de observações. O ELMQ tem a propriedade de ter um custo menor para previsões que estão acima do valor real do que para aquelas que estão abaixo dele.

Avaliando o desempenho do modelo para modelos de classificação

Os modelos de classificação são usados para prever em qual classe um grupo de recursos se enquadrará. Ao considerar um modelo de classificação, você pode começar a se perguntar o quão preciso é o modelo. Mas como você avalia a precisão? Você precisa criar um modelo de classificação antes de começar a avaliá-lo.

Como você já deve ter aprendido, avaliamos um modelo com base em seu desempenho em um conjunto de teste. Um conjunto de teste terá seus rótulos, que chamamos de verdade fundamental, e, usando o modelo, também geramos previsões para o conjunto de teste. A avaliação do desempenho do modelo envolve a comparação da verdade fundamental com as previsões. Vamos ver isso em ação com um conjunto de teste fictício:

Exemplos de teste	*Valores reais*	*Valores Previstos*	*Avaliação*
Exemplo 1	Sim	Sim	Correto
Exemplo 2	Sim	Não	Incorreto
Exemplo 3	Sim	Sim	Correto
Exemplo 4	Não	Não	Correto
Exemplo 5	Sim	Sim	Correto
Exemplo 6	Não	Sim	Incorreto
Exemplo 7	Sim	Não	Incorreto

A tabela anterior mostra um conjunto de dados fictício com sete exemplos. A segunda coluna é a verdade fundamental, que são os rótulos reais, e a terceira coluna contém os resultados de nossas previsões. A partir dos dados, podemos ver que quatro foram classificados corretamente e três foram classificados incorretamente.

Uma matriz de confusão gera a comparação resultante entre a previsão e a verdade fundamental, conforme representado na tabela a seguir:

Valores reais↓	*Previsto Sim*	*Previsto Não*
Sim	Verdadeiro Positivo (TP) = 3	Falso Negativo (FN) = 2
Não	Falso Positivo (FP) = 1	Verdadeiro negativo (TN) = 1

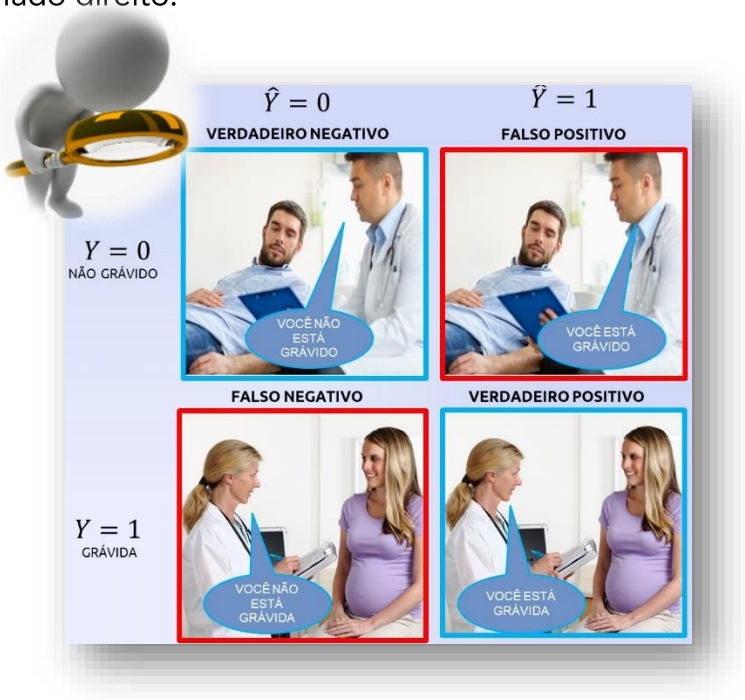
Figura 7 - Matriz de confusão

Como você pode ver na tabela, existem cinco exemplos cujos rótulos (verdade fundamental) são Sim e o dois exemplos que têm os rótulos Não.

A primeira linha da matriz de confusão é a avaliação do rótulo Sim. O verdadeiro positivo (TP) mostra aqueles exemplos cuja verdade fundamental e previsões são Sim (exemplos 1, 3 e 5). O falso negativo mostra aqueles exemplos cuja verdade fundamental é Sim e que foram erroneamente previstos como Não (exemplos 2 e 7).

Da mesma forma, a segunda linha da matriz de confusão avalia o desempenho do rótulo "Não". Falsos positivos são aqueles exemplos cuja verdade fundamental é "Não" e que foram erroneamente classificados como Sim (exemplo 6). Os verdadeiros exemplos negativos são aqueles cuja verdade fundamental e previsões são Não (exemplo 4).

Um exemplo bem-humorado da matriz de confusão pode ser visto na figura a seguir, perceba que neste exemplo e nos próximos esquemas o valor negativo fica no canto esquerdo, enquanto o valor positivo fica do lado direito.

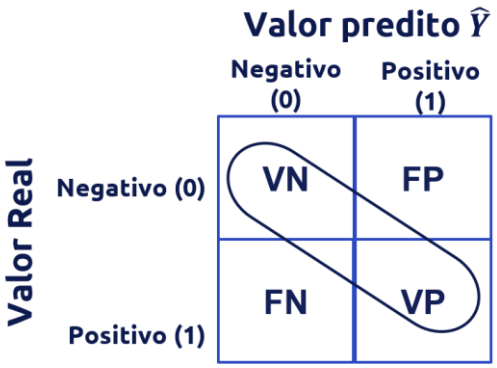


A geração de uma matriz de confusão é usada para calcular muitas das matrizes, como a acurácia e o relatório de classificação (composto pelos indicadores de precisão, recall e F1-score). Vamos trabalhar a definição destes indicadores nas próximas seções

Acurácia

Acurácia é a métrica mais simples, ela representa o número de previsões corretas do modelo. É uma ótima métrica para se utilizar quando os dados estão balanceados, vai dar uma visão geral do quanto o modelo está identificando as classes corretamente. Porém, não devemos utilizar a acurácia, quando temos classes desbalanceadas, pode causar uma falsa impressão de estamos obtendo um bom desempenho.

Por exemplo: considere um estudo em que apenas 5% da população apresenta uma determinada doença. Logo, temos um conjunto de dados desbalanceado. Se o modelo escolhido conseguir classificar corretamente todas as pessoas que não têm a doença e errar a classificação de todos os doentes, teremos uma acurácia de 95%, dando uma falsa impressão de que o modelo treinado tem uma ótima previsão. Porém, o modelo não consegue classificar corretamente a classe de interesse. A figura abaixo apresenta a fórmula para o cálculo da acurácia.



$$\text{Acurácia} = \frac{VP+VN}{VP+VN+FP+FN}$$

Valor Preditivo Negativo

Valor Preditivo Negativo (VPN) é a métrica que traz a informação da quantidade de observações classificadas como negativa (0) que realmente são negativas. Ou seja, entre todas as observações prevista como negativa (0), quantas foram identificadas corretamente. Por exemplo: entre os pacientes classificados como não doentes, quantos foram identificados corretamente.

		Valor predito $\hat{Y}$	
		Negativo (0)	Positivo (1)
Valor Real	Negativo (0)	VN	FP
	Positivo (1)	FN	VP

$$VPN = \frac{VN}{VN+FN}$$

Precisão (Precision)

Precision ou precisão, também conhecida como Valor Preditivo Positivo (VPP), é a métrica que traz a informação da quantidade de observações classificadas como positiva (1) que realmente são positivas. Ou seja, entre todas as observações identificadas como positivas (1), quantas foram identificadas corretamente. Por exemplo: entre os pacientes classificados como doentes, quantos foram identificados corretamente. A tabela abaixo apresenta a fórmula utilizada no cálculo da previsão.

		Valor predito $\hat{Y}$	
		Negativo (0)	Positivo (1)
Valor Real	Negativo (0)	VN	FP
	Positivo (1)	FN	VP

$$Precision = \frac{VP}{VP+FP}$$

Recall (Sensibilidade)

Recall ou Sensibilidade é a proporção dos Verdadeiros Positivos entre todas as observações que realmente são positivas no seu conjunto de dados. Ou seja, entre todas as observações que são positivas quantas o modelo conseguiu identificar como positiva. Representa a capacidade de um modelo em prever a classe positiva. Por exemplo: dentre todos os pacientes doentes, quantos pacientes o modelo conseguiu identificar corretamente.

		Valor predito $\hat{Y}$	
		Negativo (0)	Positivo (1)
Valor Real	Negativo (0)	VN	FP
	Positivo (1)	FN	VP

$$\text{Recall} = \frac{VP}{VP+FN}$$

Especificidade

Especificidade é a proporção dos Verdadeiros Negativos entre todas as observações que realmente são negativas no seu conjunto de dados. Ou seja, entre todas as observações que são negativas, quantas o modelo conseguiu prever como negativa. Representa a capacidade de um modelo em prever a classe negativa. Por exemplo: dentre todos os pacientes não doentes, quantos foram classificados corretamente.

		Valor predito $\hat{Y}$	
		Negativo (0)	Positivo (1)
Valor Real	Negativo (0)	VN	FP
	Positivo (1)	FN	VP

$$\text{Especificidade} = \frac{VN}{VN+FP}$$

F1- Score

F1-Score é a média harmônica entre o recall e a precisão (*precision*). Utilizada quando temos classes desbalanceada.

$$\text{F1-Score} = \frac{2*(precision*recall)}{precision+recall}$$

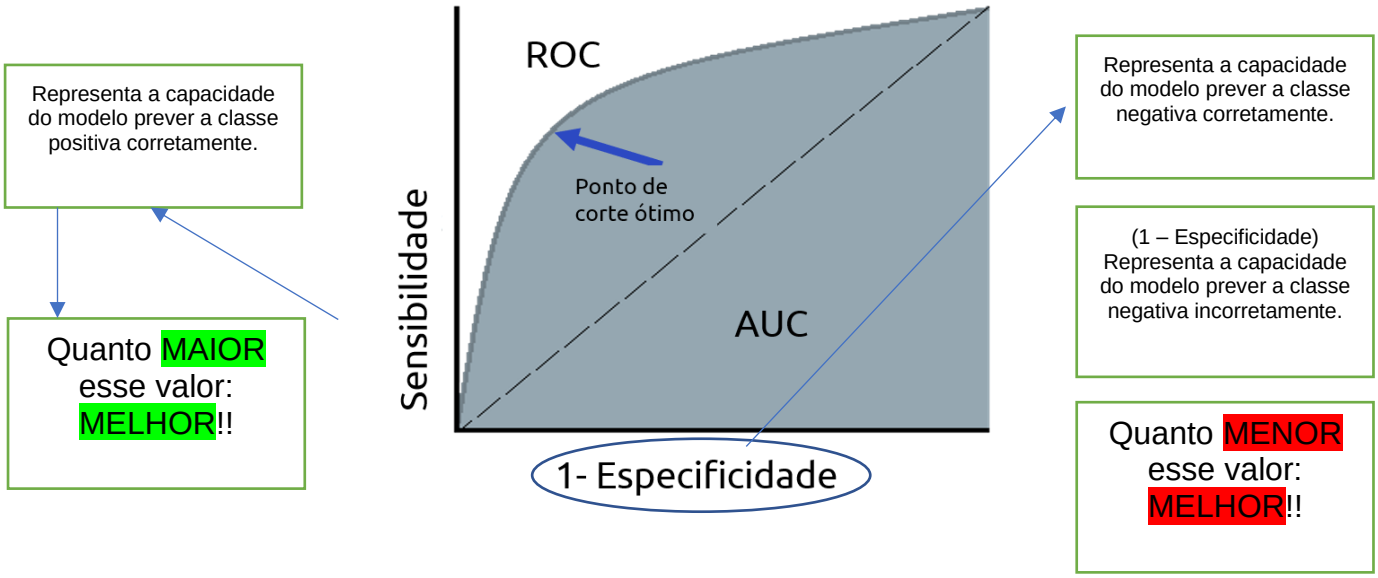
Área sob a Curva ROC

A curva ROC (*Receiver Operating Characteristic Curve*) é a curva gerada pela taxa de verdadeiros positivos (sensibilidade) e pela taxa de falsos positivos (1 – especificidade) para diferentes pontos de cortes (c). A curva ROC oferece uma visão geral de um classificador e pode ser utilizada para

encontrar pontos de corte ideias. O corte que deixa a curva mais próxima do vértice (0, 1) maximiza a sensibilidade conjuntamente com a especificidade.

Uma medida decorrente da curva ROC é o AUC (*Area Under the Curve*), que nada mais é que a área abaixo da curva. O AUC varia entre 0 e 1 e quanto maior o AUC melhor o modelo.

Utilizada quando temos classes desbalanceada e sua principal vantagem é poder escolher o melhor ponto de corte para otimizar o desempenho do modelo.



Todas essas métricas variam no intervalo [0,1] e quanto mais próximos de 1, melhor é o modelo.

Treinamento, Validação e Teste

Para fins de treinamento e teste de nosso modelo, devemos ter nossos dados divididos em três divisões distintas de conjuntos de dados: treinamento, validação e teste.

O conjunto de treinamento

É o conjunto de dados que é usado para treinar e fazer o modelo aprender os recursos/padrões ocultos nos dados. Em cada época, os mesmos dados de treinamento são alimentados repetidamente na rede neural e o modelo continua aprendendo os recursos dos dados. O conjunto de treinamento deve ter um conjunto diversificado de entradas para que o modelo seja treinado em todos os cenários e possa prever qualquer amostra de dados não vista que possa aparecer no futuro.

O conjunto de validação

O conjunto de validação é um conjunto de dados, separado do conjunto de treinamento, que é usado para validar o desempenho do nosso modelo durante o treinamento. Esse processo de validação fornece informações que nos ajudam a ajustar os hiperparâmetros e as configurações do modelo de acordo. É como um crítico nos dizendo se o treinamento está indo na direção certa ou não.

O modelo é treinado no conjunto de treinamento e, simultaneamente, a avaliação do modelo é realizada no conjunto de validação após cada época. A ideia principal de dividir o conjunto de dados em um conjunto de validação é evitar que nosso modelo seja superajustado, ou seja, o modelo se torna realmente bom em classificar as amostras no conjunto de treinamento, mas não pode generalizar e fazer classificações precisas nos dados que não viu antes.

O conjunto de teste

O conjunto de teste é um conjunto separado de dados usado para testar o modelo após a conclusão do treinamento. Ele fornece uma métrica de desempenho do modelo final imparcial em termos de exatidão, precisão, etc. Para simplificar, ele responde à pergunta " Qual é o desempenho do modelo? "

Underfitting, overfitting e técnicas de regularização

O overfitting ocorre quando o modelo é muito complexo em relação à quantidade e ao ruído dos dados de treinamento. Aqui estão as soluções possíveis:

- Simplifique o modelo selecionando um com menos parâmetros (por exemplo, um modelo linear em vez de um modelo polinomial de alto grau), reduzindo o número de atributos nos dados de treinamento ou restringindo o modelo.
- Reúna mais dados de treinamento.
- Reduza o ruído nos dados de treinamento (por exemplo, corrija erros de dados e remova outliers).

Como você pode imaginar, *underfitting* é o oposto de overfitting: ele ocorre quando seu modelo é muito simples para aprender a estrutura subjacente dos dados. Por exemplo, um modelo linear de satisfação com a vida tende a ser insuficiente; a realidade é mais complexa do que o modelo, portanto, suas previsões tendem a ser imprecisas, mesmo nos exemplos de treinamento. Aqui estão as principais opções para corrigir esse problema:

- Selecione um modelo mais poderoso, com mais parâmetros.
- Alimente melhores recursos para o algoritmo de aprendizagem (engenharia de recursos).
- Reduza as restrições no modelo (por exemplo, reduza o hiperparâmetro de regularização).

Em algum lugar entre overfitting e underfitting existe um ponto ideal onde temos a capacidade ideal de previsão; ou seja, os hiperparâmetros do modelo que são perfeitamente adequados para a tarefa e os dados - é isso que estamos buscando. O objetivo da regularização é evitar que nosso modelo se ajuste demais aos dados de treinamento. Agora que sabemos o propósito da regularização, vamos explorar algumas das muitas maneiras de regularizar nossas redes neurais.

Adicionar uma penalidade de norma de parâmetro à função objetivo é o mais clássico dos métodos de regularização. O que isso faz é limitar a capacidade do modelo. Esse método existe há várias décadas e antecede o advento do aprendizado profundo. Podemos escrever isso da seguinte forma:

$$\tilde{J}(\theta; X, y) = J(\theta; X, y) + \alpha\Omega(\theta)$$

Função de custo

Penalidade

Aqui, $\alpha \in [0, \infty]$. O valor α, na equação anterior, é um hiperparâmetro que determina o tamanho do efeito que o regularizador terá sobre a função de custo regularizada. Quanto maior o valor de α, mais regularização é aplicada, e quanto menor, menor o efeito da regularização na função de custo.

No caso de redes neurais, aplicamos apenas as penalidades de norma de parâmetro aos pesos, pois eles controlam a interação ou relacionamento entre dois nós em camadas sucessivas, e deixamos os vieses como estão. Existem algumas escolhas diferentes que podemos fazer quando se trata de que tipo de norma de parâmetro usar, e cada uma tem um efeito diferente na solução. Vejamos os dois principais métodos de regularização usados.

L2 regularization

O método de regularização L2 é muitas vezes referido como regressão Rigde (mais comumente conhecido como decaimento de peso). Ela força os pesos da rede na direção da origem através do seguinte termo de regularização para a função objetivo:

$$\Omega(\theta) = \frac{1}{2}\|\theta\|_2^2$$

Por simplicidade, vamos supor que θ = w e que todas as letras são matrizes. A função objetivo regularizada, neste caso, será a seguinte:

$$\tilde{J}(w; X, y) = J(w; X, y) + \frac{\alpha}{2} w^{\mathrm{T}} w$$

Se pegarmos seu gradiente, ele se torna o seguinte:

$$\nabla_w \tilde{J}(w; X, y) = \nabla_w J(w; X, y) + \alpha w$$

Usando o gradiente anterior, podemos calcular a atualização dos pesos em cada etapa do gradiente, como segue:

$$w \leftarrow w - \epsilon(\alpha w + \nabla_w J(w; X, y))$$

Podemos expandir e reescrever o lado direito da atualização anterior da seguinte maneira:

$$w \leftarrow (1 - \epsilon\alpha) w - \epsilon \nabla_w J(w; X, y)$$

A partir dessa equação, podemos ver claramente que a regra de aprendizado modificada faz com que nosso peso diminua $(1 - \epsilon\alpha)$ a cada passo, como no diagrama a seguir:

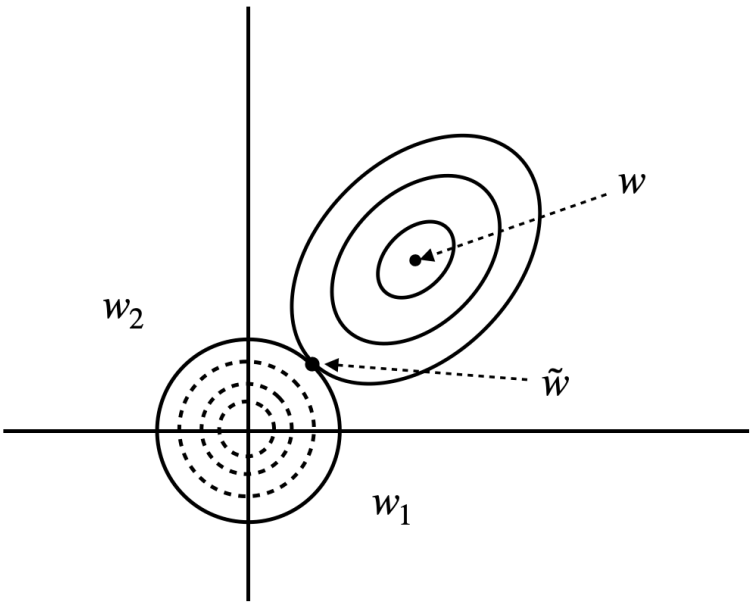


Figura 8 - Ridge Regularization - observe que o termo regularizador é um círculo.

No diagrama anterior, podemos ver o efeito que a regularização L2 tem em nossos pesos. Os círculos sólidos no lado superior direito representam contornos de igual valor da função de objetivo original, $J(w; X, y)$, à qual ainda não aplicamos nosso regularizador. Perceba que o w central aos círculos sólidos seria o ponto ótimo calculado usando gradiente descendente. Mas este

valor w é sobrestimado e precisamos alterar o valor dos parâmetros para convergir para outro ponto. Aí que entra o termo de regularização. Os círculos pontilhados, por outro lado, representam os contornos do termo regularizador, $\alpha w^{\mathrm{T}} w$. Finalmente, $\tilde{w}$, o ponto onde ambos os contornos se encontram, representa quando os objetivos concorrentes atingem o equilíbrio.

L1 regularization

Outra forma de penalidade de norma é usar a regularização L1, que às vezes é chamada de regressão de menor encolhimento absoluto e operador de seleção (LASSO). Neste caso, o prazo de regularização é o seguinte:

$$\Omega(\theta) = \|w\|_1 = \sum_i |w_i|$$

O que isso faz é somar os valores absolutos dos parâmetros. O efeito que isso tem é que introduz esparsidade (sparsity) em nosso modelo zerando alguns dos valores, nos dizendo que eles não são muito importantes. Isso pode ser pensado como uma forma de seleção de recursos.

Semelhante à regularização L2 anterior, na regularização L1, o hiperparâmetro α controla quanto efeito a regularização tem na função objetivo:

$$\tilde{J}(w; X, y) = J(w; X, y) + \alpha \|w\|_1$$

Isso é ilustrado a seguir:

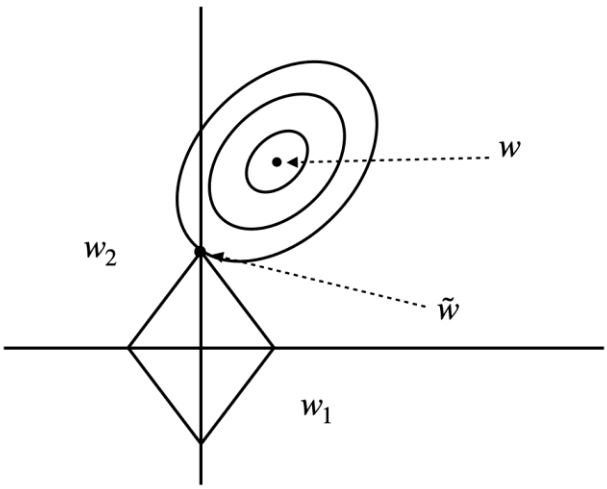


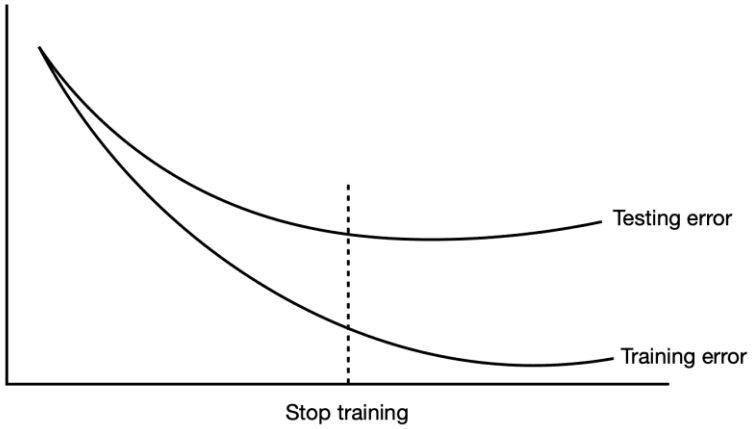
Figura 9 - Lasso regularization - perceba que a figura do termo de regularização é um losango.

Como você pode ver no diagrama anterior, os contornos da função objetivo agora se encontram nos eixos em vez de em um ponto distante dele que é de onde vem a esparsidade neste método.

Parada antecipada

Durante o treinamento, sabemos que nossas redes neurais (que têm capacidade suficiente para aprender os dados de treinamento) tendem a se ajustar demais aos dados de treinamento em muitas iterações e, portanto, são incapazes de generalizar o que aprenderam para ter um bom desempenho no conjunto de teste. Uma maneira de superar esse problema é plotar o erro nos conjuntos de treinamento e teste em cada iteração e avaliar analiticamente a iteração em que o erro dos conjuntos de treinamento e teste é o mais próximo. Em seguida, escolhemos esses parâmetros para o nosso modelo.

Outra vantagem deste método é que em nada altera a função objetivo, o que facilita o uso e não interfere na dinâmica de aprendizado da rede, que é mostrada no diagrama a seguir:



No entanto, essa abordagem não é perfeita – ela tem um lado negativo. É computacionalmente caro porque temos que treinar a rede por mais tempo do que o necessário e coletar mais dados para ela, e então observar o ponto em que o desempenho começou a degradar. Pense que, para construir o gráfico acima, você precisa estender o treinamento por mais tempo.

Dropout

Este método foi proposto como uma alternativa para evitar *overfitting* e permitir redes maiores explorarem mais regiões do espaço amostral. A ideia é bastante simples - durante cada etapa de treinamento, dada uma porcentagem predefinida n_d, uma camada de dropout seleciona aleatoriamente n_d*N unidades de entrada e as define para zero (a operação só está ativa durante a fase de treinamento, enquanto é completamente removida quando o modelo é empregado para novas previsões).

Esta operação pode ser interpretada de várias maneiras. Quanto mais camadas de dropout são empregadas, o resultado de sua seleção é uma sub-rede com capacidade reduzida que pode evitar o sobreajuste no conjunto de treinamento. A sobreposição de muitas sub-redes treinadas (cada uma com uma eliminação diferente de nós) compõe um conjunto implícito cuja previsão é uma média sobre todos os modelos. Se o dropout for aplicado em camadas de entrada, ele adiciona

um ruído aleatório às amostras. Ao mesmo tempo, o emprego de várias camadas de dropout permite explorar várias configurações potenciais que são continuamente combinadas e refinadas.

Essa estratégia é claramente probabilística, e o resultado pode ser afetado por muitos fatores impossíveis de prever; no entanto, vários testes confirmaram que o emprego de um dropout é uma boa escolha quando as redes são muito profundas, pois as sub-redes resultantes têm uma capacidade residual que lhes permite modelar uma grande parte das amostras, sem levar toda a rede a fixar sua configuração, superajustando ao conjunto de treinamento. Por outro lado, este método não é muito eficaz quando as redes são rasas ou contêm um pequeno número de neurônios (nestes casos, a regularização L2 é provavelmente a melhor escolha).

Perceba que foi introduzido um novo hiperparâmetro que especifica a probabilidade de eliminação das saídas da camada ou, inversamente, a probabilidade de retenção das saídas da camada. A interpretação é um detalhe de implementação que pode mudar de acordo com a biblioteca de código. Um valor comum é uma probabilidade de 0,5 para reter a saída de cada nó em uma camada oculta e um valor próximo a 1,0, como 0,8, para reter as entradas da camada visível. Abaixo temos uma figura que ilustra a aplicação de dropout sobre uma rede neural. Observe que, tanto dos neurônicos da camada de entrada quanto das camadas intermediárias podem ser excluídos.

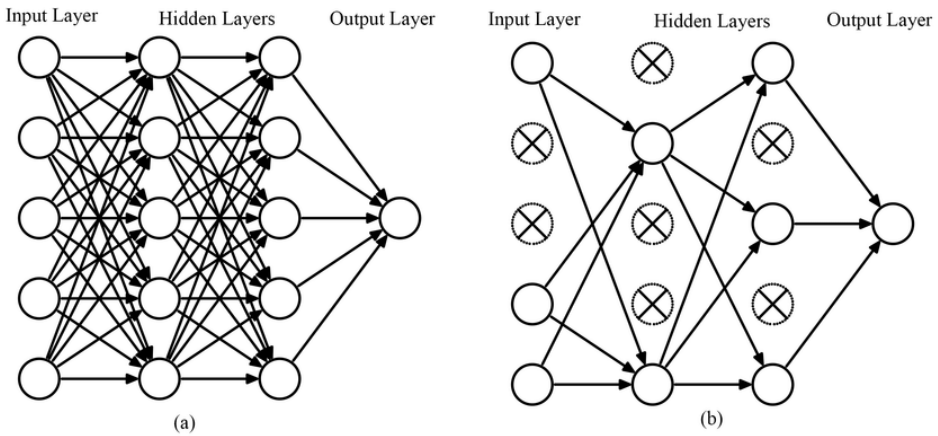
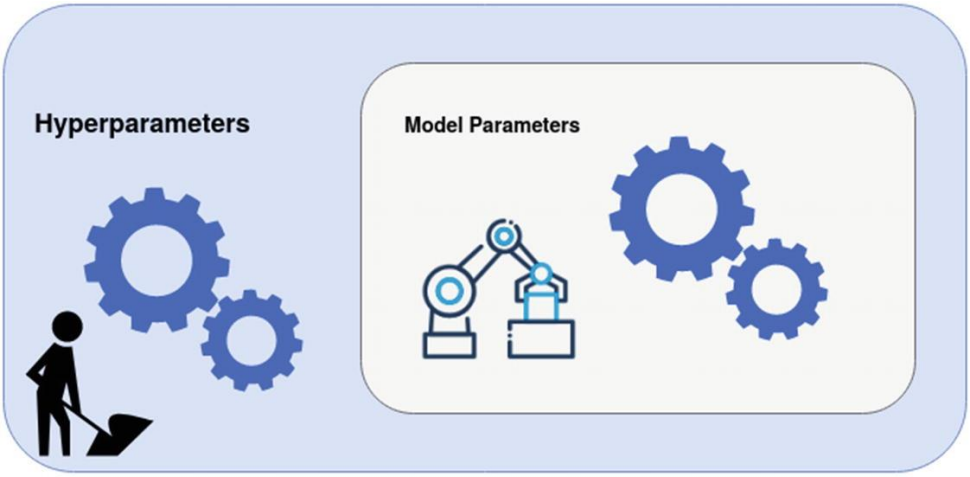


Figura 10 - (a) Rede Neural completa. (b) Rede Neural após o DROPOUT

Parâmetros x hiperparâmetros

Existem dois tipos de variáveis ao lidar com algoritmos de aprendizado de máquina, representados na figura abaixo:



- **Parâmetros**: esses são os parâmetros que **o algoritmo ajusta de acordo com o conjunto de dados** fornecido (você não tem uma palavra a dizer sobre esse ajuste)
- **Hiperparâmetros**: são os parâmetros de nível superior que **você define manualmente** antes de iniciar o treinamento, que se baseiam em propriedades como as características dos dados e a capacidade de aprendizado do algoritmo

Apresentarei um algoritmo de aprendizado de máquina como exemplo para mostrar a diferença entre um parâmetro e um hiperparâmetro. Vamos lembrar um algoritmo muito básico de regressão linear. A função de hipótese na regressão linear é a seguinte:

$$f(\Theta, \Theta_c) = \Theta.x + \Theta_c$$

Aqui, *x* e Θ são vetores, com *x* sendo um vetor de recursos e Θ sendo os pesos atribuídos a cada recurso, e Θ_c é uma tendência constante.

Vamos considerar como exemplo o problema clássico de **previsão do preço de uma casa**. O preço de uma casa depende de certos fatores, incluindo a metragem quadrada da casa, o número de quartos, o número de banheiros, a taxa de criminalidade na localidade, a distância do transporte público (estação rodoviária, aeroporto, estação ferroviária), distrito escolar (isso faz uma diferença enorme nos Estados Unidos), distância para o hospital mais próximo, e assim por diante.

Todos esses podem ser considerados como recursos; isto é, fazem parte do vetor x em nossa função de hipótese. O preço de uma casa aumenta, por exemplo, à medida que o número de quartos aumenta e a metragem quadrada aumenta; essas características teriam **peso positivo (Θ)**. O preço de uma casa diminui, por exemplo, quanto maior for a distância de escolas e hospitais e quanto maior for a taxa de criminalidade na vizinhança; eles teriam **Θ negativo**. Na Equação, f (Θ, Θ_c) fornece o preço da casa.

Podemos usar um algoritmo de otimização para **encontrar o melhor valor de Θ** para cada recurso com base nas observações anteriores. Assim, o vetor Θ é controlado e ajustado pelo algoritmo de otimização (por exemplo, gradiente descendente). **Esses pesos são *parâmetros***.

Agora, vamos discutir brevemente a descida do gradiente da função de otimização, que o ajudará a entender os hiperparâmetros.

Começaremos atribuindo alguns números aleatórios (ou seja, *pesos*) aos nossos parâmetros. Para uma observação, se tivermos o vetor *x* (com valores numéricos para cada característica) e o vetor Θ (valores numéricos aleatórios para cada peso), usando a Equação, obtemos o valor de f (Θ, Θ_c). Esta será a nossa previsão, que será algum valor aleatório (p_1') porque os pesos são aleatórios. E temos o valor verdadeiro do preço da casa (p_1).

Podemos calcular a diferença, C_1 (para a primeira observação), $| p_1 - p_1' |$. Este valor corresponde a uma perda que temos que reduzir. Da mesma forma, se calcularmos a média da soma da perda (C) para todas as observações:

$$C\,(\Theta, \Theta c) = (1/n) \sum | \, p_i - p_i' \, |$$

A Equação acima é denominada função de perda, o objetivo da função de otimização é reduzir o valor de C, para que possamos dar previsões mais precisas. A função de perda depende de pesos e tendências, conforme ilustrado na Figura abaixo.

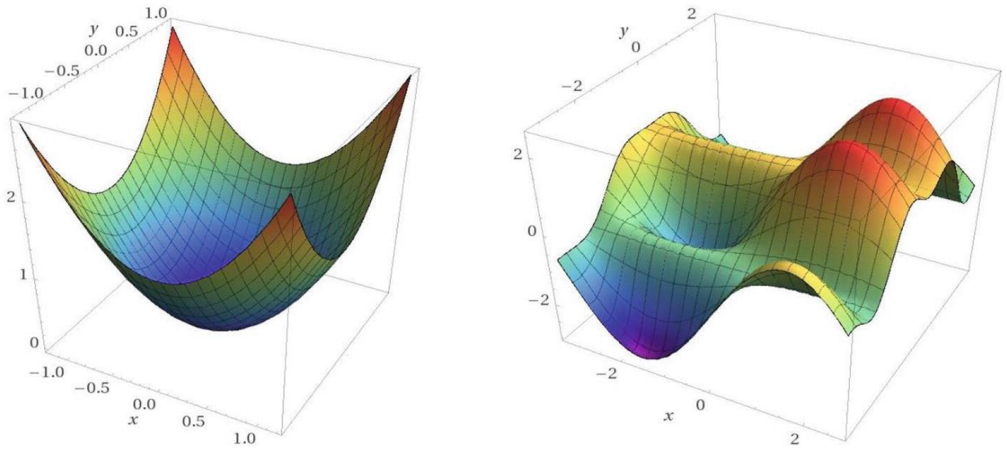


Figura 11 - Curvas de perda em três dimensões, com os eixos x e y sendo pesos e o eixo z sendo a perda

As curvas tridimensionais na Figura acima podem ser representações possíveis de uma função de perda. Lembre-se de que começamos nossos pesos e vieses com valores aleatórios; agora precisamos mudar esses valores de forma a alcançarmos a perda mínima. De acordo com o cálculo, C muda da seguinte forma:

$$\Delta C \cong \sum (\delta C/\delta \Theta_i) * \Delta \Theta_i$$

$i = \{0, n\}$, Θ_0 sendo Θ_c

Vamos representar [($\delta C/\delta\Theta_0$), ($\delta C/\delta\Theta_1$), ...] como vetor e [$\Delta\Theta_0$, $\Delta\Theta_1$ ...] como vetor $\Delta\Theta$; portanto:

$$\Delta C \cong \ddot{\nabla} C * \Delta \Theta$$

Mas suponha o seguinte:

$$\Delta \Theta = -\alpha \ddot{\nabla} C$$

Substituindo-o na Equação anterior, obtemos este resultado:

$$\Delta C \cong -\alpha * \left(\ddot{\nabla} C\right)^2$$

Aqui, sendo **α um número positivo**, a mudança na perda sempre será negativa e queremos que nossa perda seja sempre negativa. Portanto, a Equação acima permanece verdadeira. A partir dela, obtemos que

$$\left(\Theta_i' - \Theta_i\right) = -\alpha * \delta C/\delta \Theta_i$$

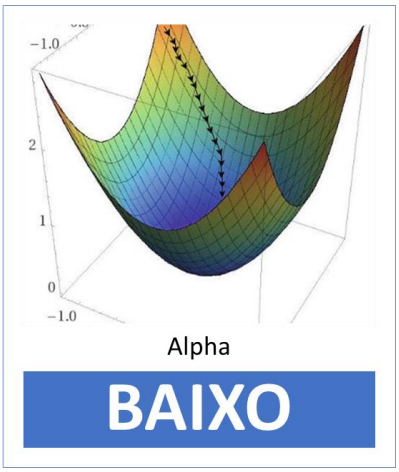
$$\therefore \Theta_i' = \Theta_i - \alpha * \delta C/\delta \Theta_i$$

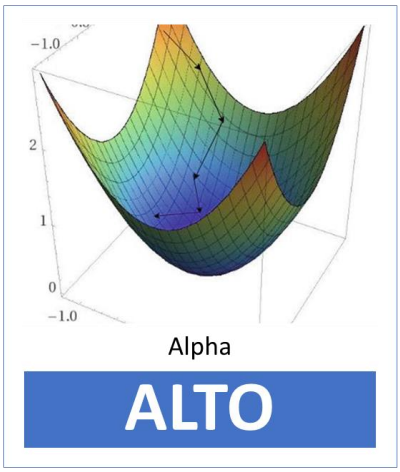
Onde Θ_i' é **o novo valor atualizado para o peso Θ_i**. Na Equação acima, o valor atualizado do peso Θ_i' depende do valor anterior do peso (Θ_i), do gradiente ($\delta C/\delta\Theta_i$) e de um número positivo α; ***α* aqui é um dos hiperparâmetros para a descida do gradiente**. Ele controla o desempenho do algoritmo. Para cada observação, executamos esta equação de atualização e diminuímos a perda enquanto alteramos os valores dos pesos, eventualmente atingindo os mínimos para a função de perda.

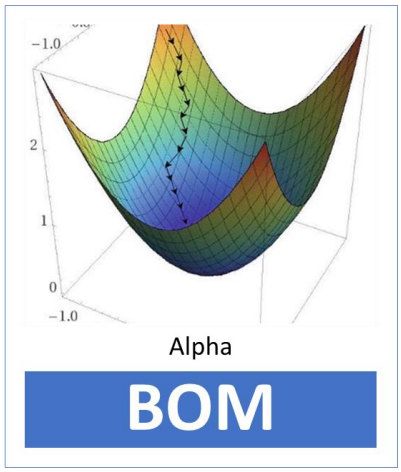
A necessidade de otimização de hiperparâmetros

Na seção anterior, usamos um número positivo α na Equação para controlar o algoritmo. Este α é chamado de **taxa de aprendizado no algoritmo de gradiente descendente**. Ele controla a taxa

pela qual a perda atinge seu mínimo. As Figuras abaixo demonstram como, conforme descrito nas figuras a seguir.







Na primeira imagem, o valor de α é **pequeno**; o algoritmos alcançará o ponto de convergência, mas o $\Delta\Theta$ (ou seja, a mudança nos pesos) será tão pequeno que **um grande número de etapas seria necessário**, aumentando assim o tempo de execução. Um grande valor de taxa de aprendizado (α) mudará a perda drasticamente, portanto, ultrapassando o limite e levando à divergência, conforme mostrado na imagem do meio. No entanto, se encontrarmos **um valor ótimo de α**, seremos capazes de alcançar a convergência em menos tempo e sem ultrapassagem, conforme representado na figura a direita. E é por isso que **precisamos ajustar α para seu valor mais eficiente, e esse processo de otimização é chamado de ajuste de hiperparâmetros.**

Em variantes mais avançadas do algoritmo de descida do gradiente, começamos com etapas maiores (ou seja, um valor maior de taxa de aprendizagem) para economizar tempo e, à medida que alcançamos o ponto de convergência, diminuímos o valor para evitar ***overshooting***. Mas o fator pelo qual diminuímos α é agora outro hiperparâmetro. Então, agora você entende a importância de ajustar esses hiperparâmetros.

Para ajustar esses hiperparâmetros, você deve ter um bom conhecimento do algoritmo e de como esses hiperparâmetros estão afetando o desempenho. Mesmo que você planeje usar algoritmos de ajuste de hiperparâmetros, é muito importante definir um bom ponto de partida. Isso economizará muito tempo e aumentará o desempenho do seu algoritmo.

Algoritmos e seus hiperparâmetros

Nesta seção, discutirei alguns algoritmos básicos de aprendizado de máquina para ajudá-lo a entender como seus hiperparâmetros funcionam. Discutirei esses hiperparâmetros com as convenções do scikit-learn, mas, como são genéricos, você pode usá-los para outras implementações ou até mesmo para algoritmos auto implementados. Não vou me aprofundar na matemática, mas darei a você o suficiente para ter uma intuição de como eles afetam o algoritmo.

KNN - K-vizinhos mais próximos

O algoritmo K-vizinhos mais próximos (KNN) pode ser usado como um algoritmo de aprendizado de máquina supervisionado e pode ser aplicado a problemas de classificação, regressão e detecção de outlier. KNN assume que pontos semelhantes estão mais próximos, conforme ilustrado na figura abaixo.

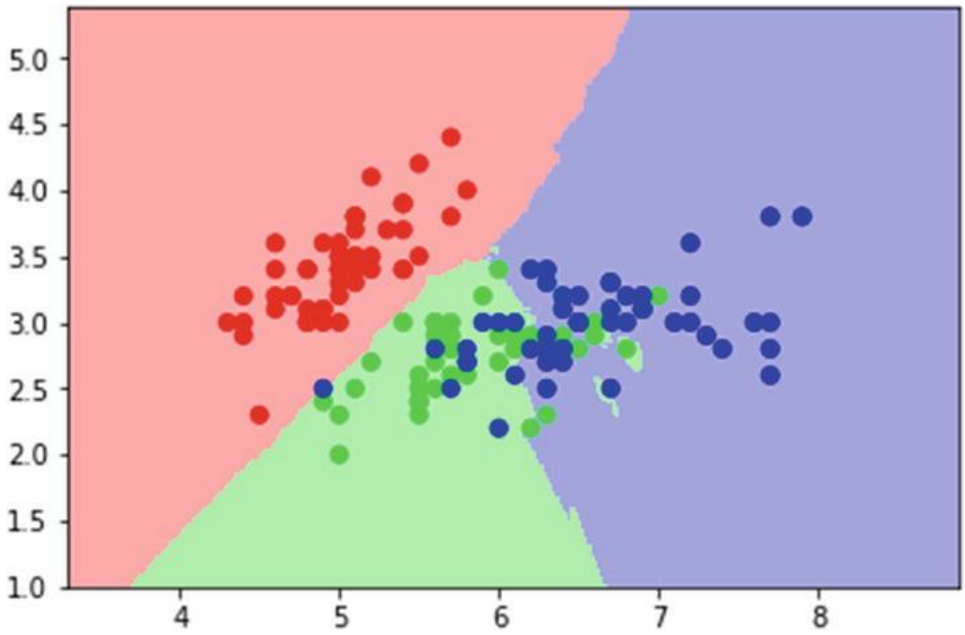


Figura 12 - Um conjunto de dados de classificação com duas dimensões quando usado com KNN mostra os limites de decisão.

K-vizinhos mais próximo encontra os K registros mais próximos do ponto que deve ser previsto. Este K pode ser definido pelo usuário. E a proximidade, portanto a distância, pode ser calculada por diferentes métricas, como distância euclidiana, distância de Manhattan e assim por diante. Para encontrar esses pontos mais próximos, algoritmos de indexação como *kd-tree* e *ball tree* são usados. Vamos discutir esses hiperparâmetros.

- ***Número K do vizinho mais próximo***: Definimos o valor de K, que é um número inteiro positivo que decide o número de amostras rotuladas do conjunto de dados de treinamento que devem ser consideradas para prever o novo ponto de dados. A figura abaixo mostra como aumentar o K pode resultar em limites mais suaves.

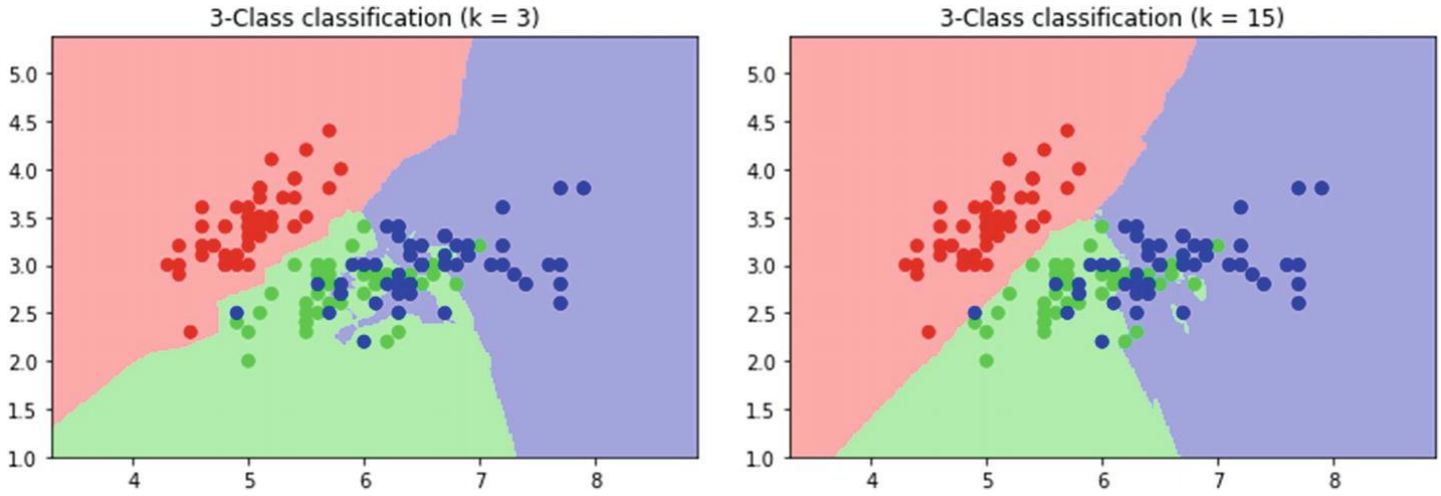


Figura 13 - Imagem a esquerda com k = 3 e imagem a direita com k = 15

- ***Pesos***: podemos dar aos nossos vizinhos mais próximos a mesma prioridade ou decidir seus pesos com base na distância do ponto de consulta; quanto mais longe o ponto, menor o peso.

- ***Algoritmo de indexação***: *algoritmos de* indexação são usados para mapear os pontos mais próximos. Se usarmos o método de força bruta resultaria no cálculo da distância de todos os pares de pontos de dados em um conjunto de dados. Sendo assim, usamos algoritmos de indexação baseados em árvore, como *kd-tree* e *ball tree*. Quando o número de dimensões é maior, o ball tree é mais eficiente do que o kd-tree.

- ***Métrica de distância***: uma métrica deve ser usada para calcular a distância entre os pontos. Pode ser a Euclidiana ou a Manhattan ou ordens superiores da métrica de Minkowski.

Máquina de vetores de suporte

Máquina de vetores de suporte (SVM) é um algoritmo poderoso que encontra um plano hiperdimensional que separa classes distintas. Um exemplo é mostrado na figura a seguir, na qual temos duas classes denotadas pelas cores vermelha e azul. A linha preta que os divide é o nosso **hiperplano** (uma linha neste caso, já que estamos visualizando em duas dimensões). O SVM encontra o hiperplano de forma que a margem (a distância entre as duas linhas pontilhadas) seja máxima.

Os pontos de dados próximos às linhas pontilhadas são chamados de ***vetores de suporte***. Eles são altamente responsáveis pela formação do hiperplano. Usamos o método de otimização de multiplicadores de Lagrange para encontrar este hiperplano.

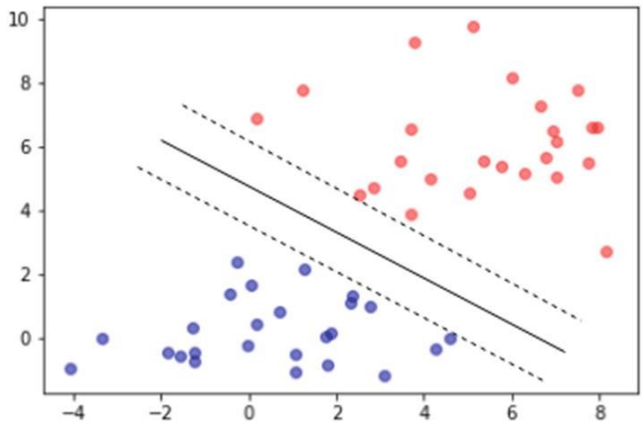


Figura 14 - Classes separadas por hiperplano

Mas esse era um problema **linearmente separável**. Na vida real, os conjuntos de dados não são linearmente separáveis. Portanto, vamos dar outro exemplo e ver como o SVM funcionaria:

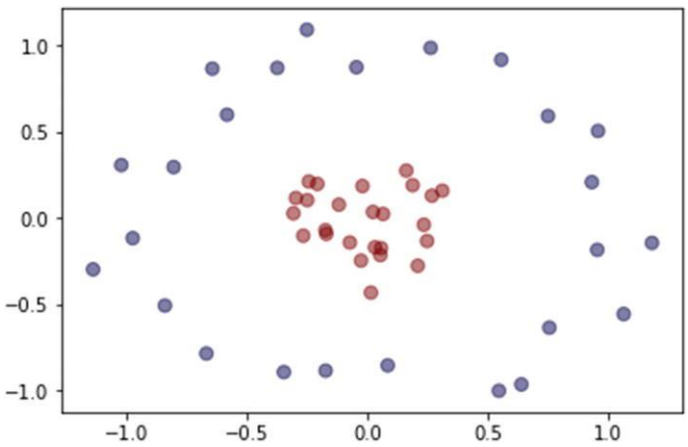


Figura 15 - Conjunto de dados com duas classes, azul e vermelho

A figura acima não é linearmente separável. Portanto, é necessário projetá-la para uma dimensão superior (três dimensões, neste caso), como mostrado na figura a seguir e agora podemos aplicar SVM e encontrar o plano separando-o.

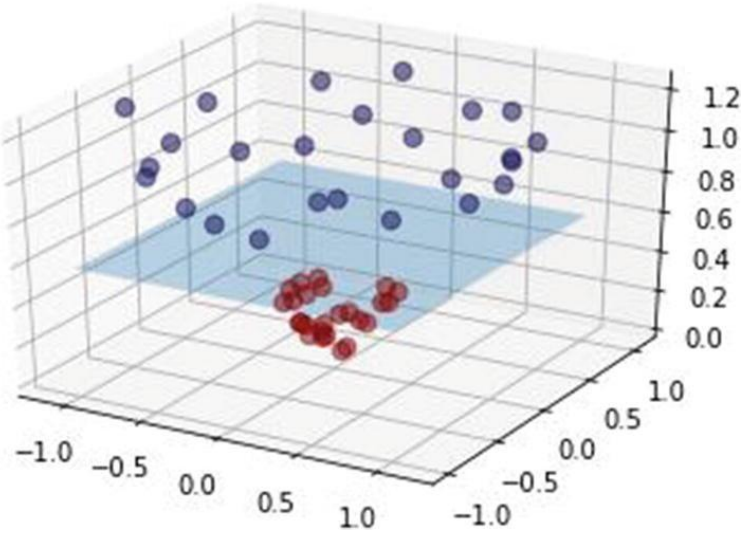


Figura 16 - Dados projetados em dimensões superiores e separados por um plano

Precisamos encontrar a função de mapeamento correta para projetar dados em dimensões superiores. É aí que diferentes **kernels** entram em jogo. As funções de mapeamento corretas podem ser obtidas usando as funções de kernel corretas, que é um dos hiperparâmetros mais importantes em SVMs. Vamos agora discutir os diferentes hiperparâmetros:

- **Kernel**: conforme descrito anteriormente, um kernel ajuda a obter a função de mapeamento correta, que é essencial para o desempenho eficiente do SVM. Encontrar apenas o kernel reduz a complexidade de encontrar a função de mapeamento; há uma relação matemática direta entre a função de mapeamento e a função de kernel. A Figura acima é um exemplo de problema que pode ser resolvido usando o kernel da função de base radial (RBF). Alguns dos kernels amplamente usados são o polinomial, Gaussian, sigmoid e, claro, o RBF, a maioria deles estão definidos na implementação SVM no scikit-learn (o sklearn também permite que você defina seu próprio kernel).

- **C**: C é um parâmetro de regularização. Ele cria um trade off entre a precisão do treinamento e a largura da margem. Uma diminuição em C resulta em margens maiores e menor precisão de treinamento e vice-versa.

- **Gama**: Gama (γ) define a influência dos pontos de treinamento.

- **Sigma** = 1/gama

- **Grau**: Este hiperparâmetro é usado apenas em kernels polinomiais; um grau mais alto significa um limite de decisão mais flexível. O grau 1 resultaria em um kernel linear.

Árvore de Decisão

A árvore de decisão é semelhante a um monte de instruções *if-else*, um algoritmo simples, mas elegante, com uma visualização muito intuitiva. É realmente fácil entender o que está acontecendo, ao contrário das redes neurais. Além disso, pouco ou nenhum pré-processamento de dados é necessário.

Como o nome sugere, é uma árvore, portanto, começa com **um nó raiz**, que é **um dos recursos**. Com base no valor desse recurso para nosso ponto de dados, selecionamos o próximo nó da árvore. Isso continua até **chegarmos à folha** e, portanto, **ao valor de previsão**.

Criar esta árvore é um pouco complicado; vários algoritmos diferentes são usados para selecionar qual recurso vai para o topo, qual vai para o segundo e assim por diante. Alguns dos algoritmos que calculam a importância dos recursos são **o índice de Gini**, o de **ganho de informação** e o **qui-quadrado**. A seleção desse algoritmo pode ser considerada um dos hiperparâmetros importantes no algoritmo da árvore de decisão.

Vejamos o exemplo de um conjunto de dados Iris clássico. Aqui, o objetivo é classificar as três espécies de flores de íris, Setosa, Versicolor e Virginica, com base em quatro características, comprimento da sépala, largura da sépala, comprimento da pétala e largura da pétala.

Como eu disse antes, a visualização de uma árvore de decisão é muito fácil; sklearn fornece uma função, tree.plot_tree() , onde você apenas tem que inserir seu classificador treinado e ele irá plotar a árvore (veja a figura abaixo).

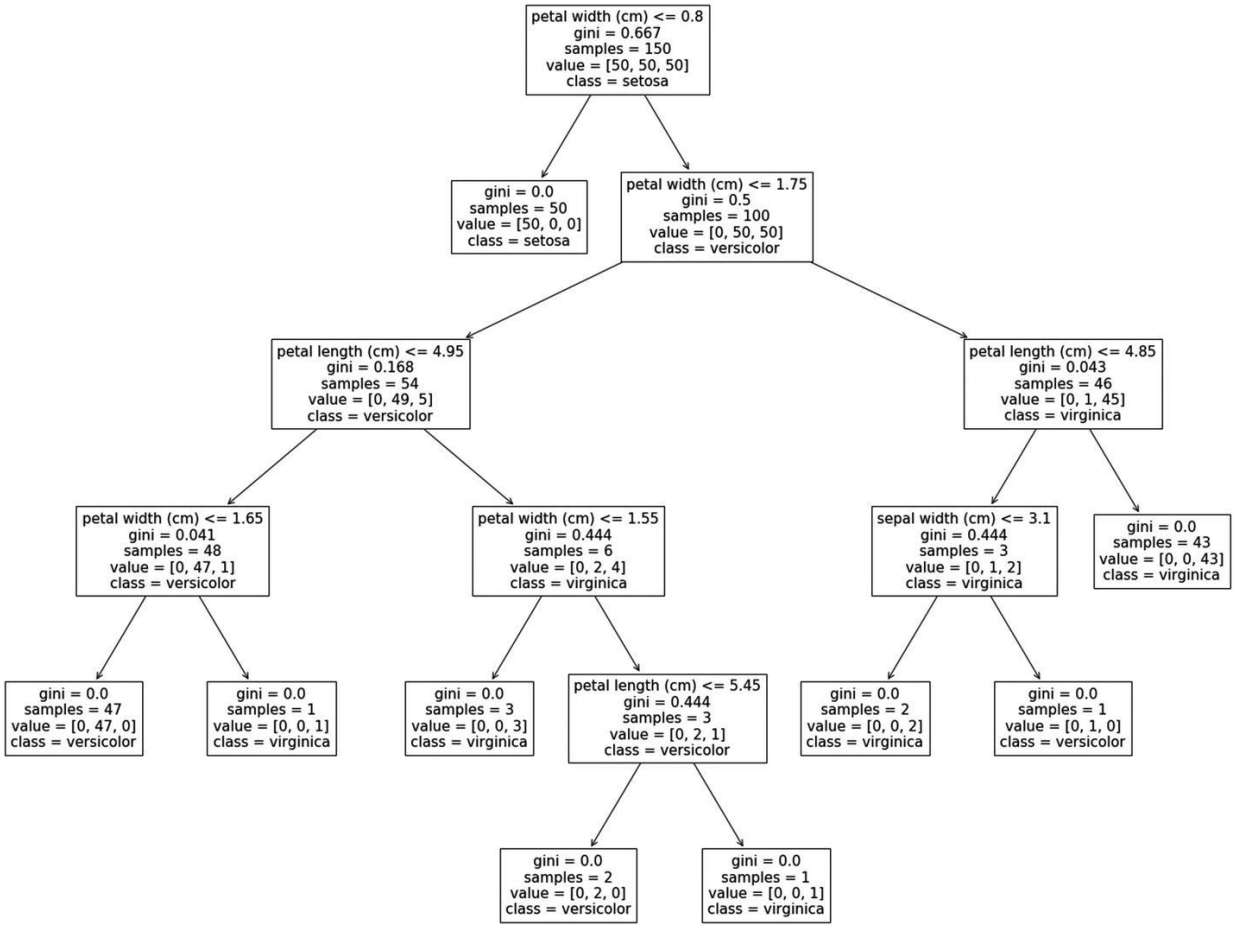


Figura 17 - Um classificador de árvore de decisão treinado no conjunto de dados Iris

Como você pode ver na figura acima, existem índices de gini para todos os nós, com base em quais recursos são colocados na árvore. Conforme descemos na árvore, o valor do índice de gini diminui. No topo dos nós, uma condição é especificada; se for verdadeiro, o ponto de dados vai para o filho da direita e, se for falso, o ponto de dados vai para o filho esquerdo. O valor das amostras nos diz o número de instâncias que se encontram na condição verdadeira e falsa do nó pai.

Um dos problemas que enfrentamos com as árvores de decisão é que, quando a árvore fica complexa, há uma grande chance de que o modelo se ajuste demais aos dados de treinamento. Alguns dos hiperparâmetros podem ajudar a reduzir essa complexidade. Para resolver este problema, podemos podar a árvore, usando hiperparâmetros como **profundidade máxima** da árvore e **número mínimo de amostras no nó folha**: Aqui estão os hiperparâmetros:

- **Algoritmo**: Como mencionado anteriormente, este algoritmo decide a prioridade dos recursos e, portanto, sua ordem na estrutura em árvore.

- **Profundidade da árvore**: define a profundidade máxima da árvore. Isso certamente pode afetar a complexidade estrutural e o tempo de processamento da árvore. Podemos remover nós sem importância e reduzir a profundidade.

- **Divisão de amostra mínima**: este é um valor inteiro que define o número mínimo de amostras necessárias para dividir um nó interno. Na figura acima, se tivéssemos escolhido 101, a árvore teria parado após a segunda camada.

- **Quantidade de amostra mínima na folha**: define o número mínimo de amostras na folha. Este hiperparâmetro pode ajudar a reduzir o sobreajuste, reduzindo a profundidade da árvore.

Redes neurais

Uma *rede neural* básica é composta de nós e camadas de nós, e esses nós nada mais são do que a saída da camada anterior multiplicadas pelos pesos. Chamamos pesos e vieses, neste contexto, de *parâmetros* (uma vez que são decididos por um algoritmo de modelagem baseado no conjunto de dados) e chamamos o **número de nós**, **número de camadas** e assim por diante de *hiperparâmetros* (já que intervimos na sua definição).

Definir a arquitetura de uma rede neural é uma das tarefas mais desafiadoras enfrentadas pelos cientistas de dados em uma tarefa de aprendizado de máquina. A arquitetura não pode ser descoberta por força bruta (tentar todos os modelos possíveis) porque a complexidade associada ao tempo de processamento das redes neurais é muito alta e não é possível experimentar todas as combinações de hiperparâmetros. Portanto, criar uma arquitetura de rede neural é mais uma arte, contando com lógica e algoritmos de ajuste de hiperparâmetros mais avançados.

Um grande número de hiperparâmetros diferentes existem no contexto das redes neurais, então vamos discutir alguns deles aqui:

- **Número de camadas**: adicionar camadas aumenta a profundidade da rede neural e a capacidade de aprender recursos mais complexos.

- **Número de nós**: O número de nós varia de acordo com as camadas, mas o número de nós na primeira camada oculta e na última camada oculta deve ser igual ao número de recursos de entrada e de classes a serem previstas, respectivamente. Para a camada oculta, por convenção, usamos o número de nós em expoentes de 2, o que significa 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024 e assim por diante. Isso ocorre porque o hardware tem um desempenho mais eficiente quando os números são armazenados em potências de dois, embora não haja prova de que essa seja a maneira mais eficiente de selecionar esse tipo de hiperparâmetros.

- **Tamanho de lote**: Se tomarmos a uma subamostra do conjunto de dados, ela deve representar as propriedades de todo o conjunto de dados. Este lote pode ser usado para

calcular o gradiente e atualizar os pesos. E iteramos em todas as subamostras até cobrir todo o conjunto de dados. A ideia é economizar espaço na memória. Mas você precisa escolher o valor ideal do tamanho do subconjunto, porque um tamanho de lote menor causaria mais flutuações ao atingir os valores mínimos, e um valor maior pode causar erros de memória.

- **Função de ativação**: as funções de ativação são usadas para introduzir uma não linearidade em cada nó. Poucas coisas que precisamos ter certeza ao decidir as funções de ativação são, elas devem ser usadas em milhares e milhões de nós, e a retropropagação usa suas derivadas, portanto, a função e sua derivada devem ser pouco complexas computacionalmente. Algumas das ativações amplamente utilizadas são ReLU, Sigmoid e Leaky ReLU. Falaremos mais sobre elas no contexto de redes neurais, ainda nesta aula

- **Função de perda (loss)**: a função de perda é escolhida com base na saída, seja uma classificação binária, classificação multiclasse, regressão e assim por diante. Existem também outros fatores. Por exemplo, usar a ativação sigmóide na última camada e a função de perda quadrática pode resultar na lentidão do aprendizado. Portanto, coisas como essas precisam ser observadas. Também existem hiperparâmetros internos para a função de perda que podem ser ajustados.

- **Otimizador**: anteriormente, discutimos um método de otimização, o de gradiente descendente. Existem outros métodos de otimização mais avançados, como Adagrad, Adam Optimizer e assim por diante, e esses otimizadores também contêm vários hiperparâmetros que afetam a otimização geral.

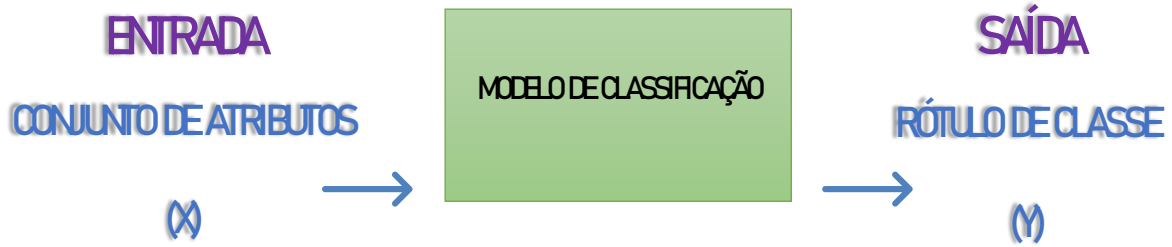
Existem muitos outros hiperparâmetros em redes neurais, como normalização em lote (batch normalization), eliminação (dropout) e assim por diante. E a cada poucos dias essas variáveis estão aumentando com o avanço da tecnologia. Enfim, não vamos conseguir esgotar a quantidade de hiperparâmetros possíveis. Esta seção teve como objetivo dar uma ideia do que são hiperparâmetros e como funcionam.

Classificação

Parece ser um imperativo humano. A fim de compreender e comunicar sobre o mundo que estamos constantemente a classificar, categorizar e classificar. Dividimos as coisas vivas em filos, espécies e gênero; matéria em elementos; cães em raças, as pessoas em raças. Os objetos a serem classificados são geralmente representados por registros em um banco de dados ou um arquivo, e o ato de classificação consiste em adicionar uma nova coluna com um código de classe de algum tipo.

Uma das tarefas mais comuns dentro de mineração de dados consiste em examinar as características de um objeto recém-apresentado e atribuí-lo a um dos conjuntos predefinidos de classes. A tarefa de classificação é caracterizada por uma definição das classes (1), e conjunto dados para aprendizado (2) pré-classificados.

Uma definição mais formal para a classificação é a tarefa de aprendizado de uma função alvo f que mapeia cada atributo de um conjunto x para um rótulo de classe predefinido y. Essa descrição foi dada por Tan em seu livro de mineração e pode ser observada na figura abaixo:



O modelo construído baseia-se na análise prévia de um conjunto de dados de amostragem ou dados de treinamento, contendo objetos corretamente classificados. Por exemplo, suponha que o gerente do supermercado está interessado em descobrir que tipo de características classificam seus clientes em "bom comprador" ou "mau comprador". Um modelo de classificação poderia incluir a seguinte regra: "Clientes da faixa econômica B, com idade entre 50 e 60 são maus compradores".

Na classificação, o objetivo é a construção de um modelo que possa ser aplicado a dados não classificados e classificá-los. São exemplos de tarefas de classificação que foram abordados por meio de técnicas de mineração de dados: classificação de pedido de crédito como baixo, médio ou alto risco, escolher conteúdo a ser exibido em uma página Web, determinar quais os números de telefone correspondem a máquinas de fax, descobrir sinistros fraudulentos e atribuir códigos da indústria e denominações de emprego com base nas descrições de texto livre.

Em todos os exemplos, há um número limitado de classes, e espera-se ser capaz de atribuir qualquer registo em um ou outra. As árvores de decisão e técnicas semelhantes são bem adaptadas para a classificação. Rede neural e análise de links também são úteis para a classificação de certas circunstâncias. Vejam na figura a seguir um fluxo que mostra o funcionamento de um algoritmo de classificação:

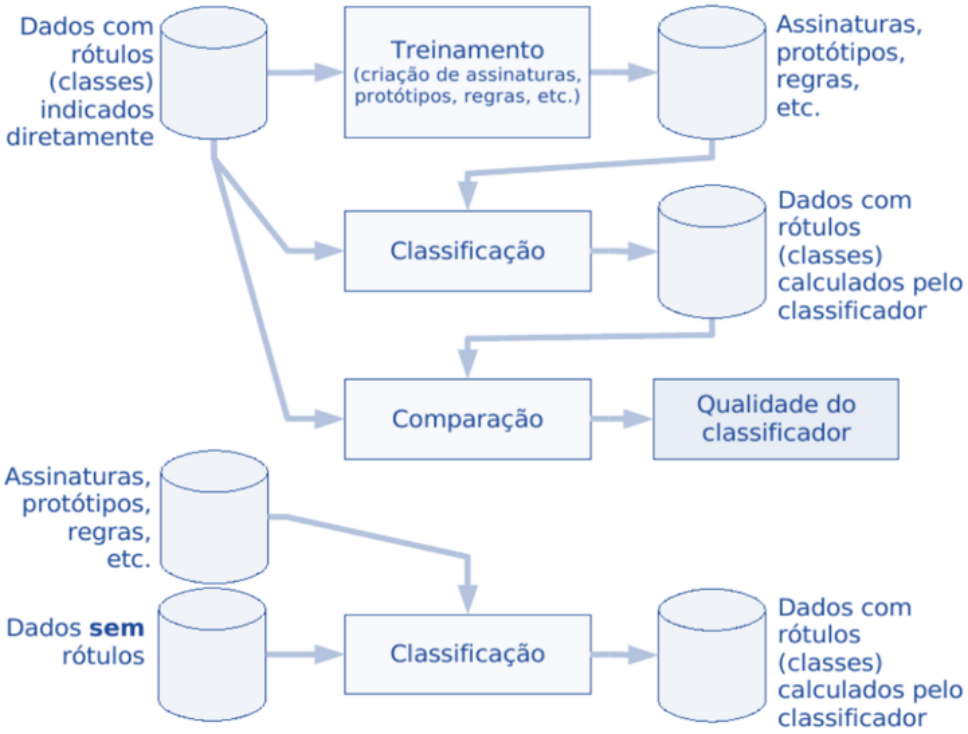


Figura 18 - Processo de construção de um classificador

APOSTA ESTRATÉGICA

A ideia desta seção é apresentar os pontos do conteúdo que mais possuem chances de serem cobrados em prova, considerando o histórico de questões da banca em provas de nível semelhante à nossa, bem como as inovações no conteúdo, na legislação e nos entendimentos doutrinários e jurisprudenciais[1].

Algoritmos de aprendizagem de máquina são programas que podem aprender com dados e melhorar a partir da experiência, sem intervenção humana. Tarefas de aprendizagem podem incluir aprender a função que mapeia a entrada para a saída, aprender a estrutura oculta em dados não rotulados; ou "aprendizagem baseada em instâncias", onde um rótulo de classe é produzido para uma nova instância, comparando a nova instância (linha) com instâncias dos dados de treinamento, que foram armazenados na memória. O 'aprendizado baseado em instâncias' não cria uma abstração a partir de instâncias específicas.

[1] Vale deixar claro que nem sempre será possível realizar uma aposta estratégica para um determinado assunto, considerando que às vezes não é viável identificar os pontos mais prováveis de serem cobrados a partir de critérios objetivos ou minimamente razoáveis.

Tipos de algoritmos de aprendizagem de máquina

Existem 3 tipos de algoritmos de aprendizado de máquina (ML):

Algoritmos de aprendizagem supervisionados: O aprendizado supervisionado usa dados de treinamento rotulados para aprender a função de mapeamento que transforma as variáveis de entrada (X) na variável de saída (Y). Em outras palavras, ele resolve para f na seguinte equação:

$$Y = f(X)$$

Isso nos permite gerar saídas com precisão quando dadas novas entradas.

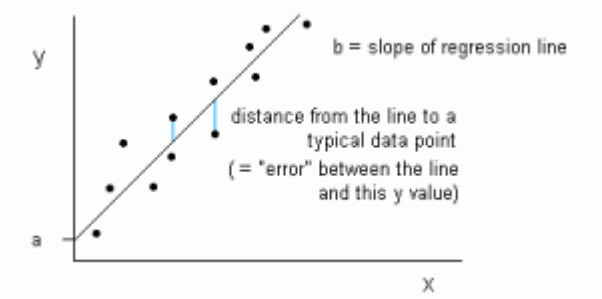
Algoritmos de aprendizagem não supervisionados: Modelos de aprendizagem não supervisionados são usados quando temos apenas as variáveis de entrada (X) e nenhuma variável de saída correspondente. Eles usam dados de treinamento não rotulados para modelar a estrutura subjacente dos dados.

Aprendizado por reforço: O aprendizado por reforço é um tipo de algoritmo de aprendizado de máquina que permite que um agente decida a melhor próxima ação com base em seu estado atual (política), aprendendo comportamentos que maximizarão uma recompensa.

Algoritmos de aprendizado de máquina supervisionado

1. Regressão Linear

No aprendizado de máquina, temos um conjunto de variáveis de entrada (x) que são usadas para determinar uma variável de saída (y). Existe uma relação entre as variáveis de entrada e a variável de saída. O objetivo da ML é quantificar essa relação.



A Regressão Linear é representada como uma linha na forma de y = a + bx.

Na Regressão Linear, a relação entre as variáveis de entrada (x) e a variável de saída (y) é expressa como uma equação da forma y = a + bx. Assim, o objetivo da regressão linear é descobrir os valores dos coeficientes a e b. Aqui, a é a interceptação e b é a inclinação da linha.

A Figura acima mostra os valores de x e y plotados para um conjunto de dados. O objetivo é encaixar uma linha mais próxima da maioria dos pontos. Isso reduziria a distância ('erro') entre o valor y de um ponto de dados e a linha.

2. Regressão Logística

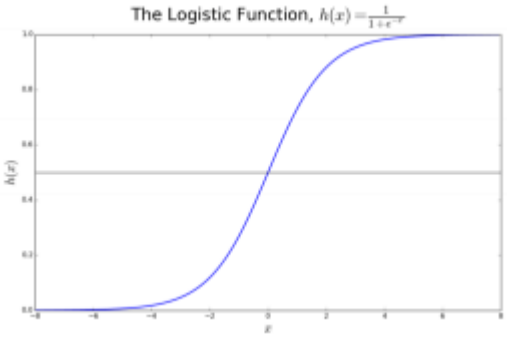
As previsões de regressão linear são valores contínuos (ou seja, chuvas em cm), as previsões de regressão logística são valores discretos (ou seja, se um aluno passou/falhou) após a aplicação de uma função de transformação.

A regressão logística é mais adequada para classificação binária: conjuntos de dados onde y = 0 ou 1, onde 1 denota a classe padrão. Por exemplo, ao prever se um evento ocorrerá ou não, há apenas duas possibilidades: que ocorra (o que denotamos como 1) ou que não (0). Então, se estivéssemos prevendo se um paciente estava doente, rotularíamos pacientes doentes usando o valor do nosso conjunto de dados.

A regressão logística é nomeada após a função de transformação que utiliza, que é chamada de função logística $h(x)= 1/ (1 + e^x)$. Isso forma uma curva em forma de S.

Na regressão logística, a saída assume a forma de probabilidades da classe padrão (ao contrário da regressão linear, onde a saída é produzida diretamente). Como é uma probabilidade, a saída está na faixa de 0-1. Então, por exemplo, se estamos tentando prever se os pacientes estão doentes, já sabemos que pacientes doentes são denotados como 1, então se nosso algoritmo atribui a pontuação de 0,98 a um paciente, ele acha que o paciente é bastante provável que esteja doente.

Esta saída (valor y) é gerada por log transformando o valor x, utilizando a função logística h(x)= 1/ (1 + e^{-x}). Um limiar é então aplicado para forçar essa probabilidade em uma classificação binária.



Regressão Logística para determinar se um tumor é maligno ou benigno. Classificado como maligno se a probabilidade h(x)>= 0,5.

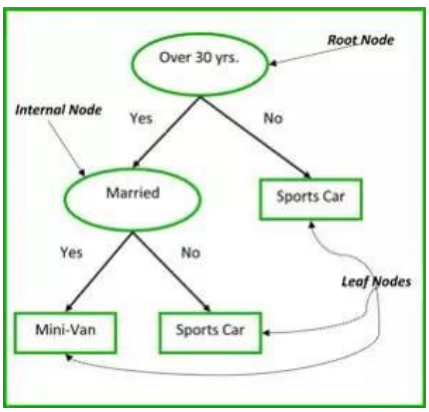
Na Figura anterior, para determinar se um tumor é maligno ou não, a variável padrão é y = 1 (tumor = maligno). A variável x pode ser uma medida do tumor, como o tamanho do tumor. Como mostrado na figura, a função logística transforma o valor x das várias instâncias do conjunto de dados, na faixa de 0 a 1. Se a probabilidade cruzar o limiar de 0,5 (mostrado pela linha horizontal), o tumor é classificado como maligno.

A equação de regressão logística $P(x) = e^{(b_0 + b_1x)}/1 + e(b_0 + b_1x))$ pode ser transformada em $ln(p(x)/1-p(x)) = b_0 + b_1x$.

O objetivo da regressão logística é utilizar os dados de treinamento para encontrar os valores dos coeficientes b_0 e b_1 de forma que minimize o erro entre o desfecho previsto e o resultado real.

3. CART

As Árvores de Classificação e Regressão (CART) são uma implementação das Árvores de Decisão.



Os nós não terminais de Árvores de Classificação e Regressão são o nó raiz e o nó interno. Os nós terminais são os nós de folhas. Cada nó não terminal representa uma única variável de entrada (x) e um ponto de divisão nessa variável; os nódulos de folha representam a variável de saída (y). O modelo é usado da seguinte forma para fazer previsões: caminhe as rachaduras da árvore para chegar a um nó de folha e produzir o valor presente no nó da folha.

A árvore de decisão na Figura acima classifica se uma pessoa comprará um carro esportivo ou uma minivan dependendo de sua idade e estado civil. Se a pessoa tem mais de 30 anos e não é casada, caminhamos na árvore da seguinte forma: 'mais de 30 anos?' -> sim -> 'casados?' -> não. Assim, o modelo produz um carro esportivo.

4. Bayes ingênuo

Para calcular a probabilidade de ocorrer um evento, dado que outro evento já ocorreu, usamos o Teorema de Bayes. Para calcular a probabilidade de que a hipótese(h) seja verdadeira, dado nosso conhecimento prévio(d), usamos o Teorema de Bayes da seguinte forma:

$$P(h|d)= (P(d|h)\ P(h)) / P(d)$$

onde:

- P(h|d) = Probabilidade posterior.
- P(d|h) = Probabilidade.
- P(h) = Probabilidade prévia da classe.
- P(d) = Probabilidade prévia do preditor. Probabilidade dos dados (independentemente da hipótese)

Este algoritmo é chamado de "ingênuo" porque assume que todas as variáveis são independentes umas das outras, o que é uma suposição ingênua de se fazer em exemplos do mundo real.

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Usando Baias Ingive para prever o status de 'jogar'/'play' usando a variável 'weather'.

Usando a tabela como exemplo, qual é o resultado se o tempo = 'ensolarado'?

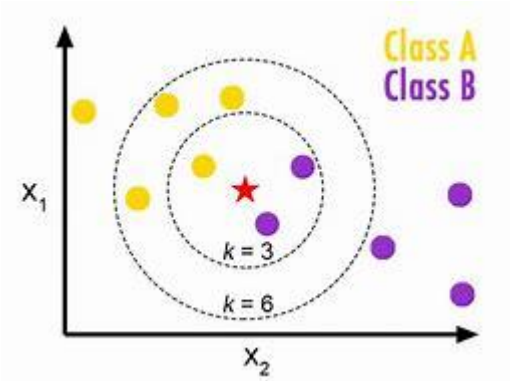
Para determinar o desfecho de jogar = 'sim' ou 'não' dado o valor do tempo variável = 'ensolarado', calcule P(yes|sunny) e P(no|sunny) e escolha o resultado com maior probabilidade.

->P(yes|sunny)= (P(sunny|yes) * P(yes)) / P(sunny) = (3/9 * 9/14) / (5/14) = 0,60
-> P(no|sunny)= (P(sunny|no) * P(no)) / P(sunny) = (2/5 * 5/14) / (5/14) = 0,40

Assim, se o tempo = 'ensolarado', o resultado é jogo = 'sim'.

5. KNN

O algoritmo K-Mais Próximo dos Vizinhos usa todo o conjunto de dados como conjunto de treinamento, em vez de dividir os dados definidos em um conjunto de treinamento e conjunto de testes.



Quando um resultado é necessário para uma nova instância de dados, o algoritmo KNN passa por todo o conjunto de dados para encontrar as instâncias k mais próximas da nova instância, ou o número k de instâncias mais semelhante ao novo registro, e então produz a média dos resultados (para um problema de regressão) ou o modo (classe mais frequente) para um problema de classificação. O valor de k é especificado pelo usuário.
A semelhança entre as instâncias é calculada utilizando medidas como distância euclidiana e distância de Hamming.

Imprima o capítulo Aposta Estratégica separadamente e dedique um tempo para absolver tudo o que está destacado nessas duas páginas. Caso tenha alguma dúvida, volte ao Roteiro de Revisão e Pontos do Assunto que Merecem Destaque. Se ainda assim restar alguma dúvida, não hesite em me perguntar no fórum.

QUESTÕES ESTRATÉGICAS

Nesta seção, apresentamos e comentamos uma amostra de questões objetivas selecionadas estrategicamente: são questões com nível de dificuldade semelhante ao que você deve esperar para a sua prova e que, em conjunto, abordam os principais pontos do assunto.

A ideia, aqui, não é que você fixe o conteúdo por meio de uma bateria extensa de questões, mas que você faça uma boa revisão global do assunto a partir de, relativamente, poucas questões.

1. (FGV - Analista Judiciário (TJ RO)/Analista de Sistemas/2021)

A Inteligência Artificial (IA) apoia o desenvolvimento de soluções tecnológicas capazes de realizar atividades similares às capacidades cognitivas humanas. Como exemplo, a plataforma

Sinapses, desenvolvida pelo Tribunal de Justiça do Estado de Rondônia (TJRO) e adaptada para uso nacional, gerencia o treinamento supervisionado de modelos de IA.

Em soluções de IA, a tecnologia que possui a capacidade de melhorar o desempenho na realização de alguma tarefa por meio da experiência usando dados de treinamento, podendo ser supervisionado ou não, é o(a):

A Motor de Inferência (Inference Engine) de Sistemas Especialistas (Expert Systems);

B Raciocínio Automatizado (Automated Reasoning);

C Compreensão de Linguagem Natural (Natural-Language Understanding);

D Representação do Conhecimento (Knowledge Representation) usando Lógica de Primeira Ordem (First Logic Order);

E Aprendizado de Máquina (Machine Learning).

Comentário:

Gabarito: E

Comentários

Veja que tratamos de desse conceito de melhoria de eficiência por meio da aprendizagem de máquina no início desta seção, logo, temos a resposta na alternativa E. As outras assertivas descrevem outras áreas/conceitos da Inteligência Artificial:

a) Um motor de inferência é uma ferramenta informatizada "caixa preta", também utilizada em Sistema Especialista (Inteligência Artificial), que após ser estimulada com solicitações predeterminadas, oferece as soluções possíveis. Este é o núcleo da inteligência artificial de um sistema especialista, onde a capacidade do motor de inferência é baseada numa combinação de procedimentos de raciocínios de forma regressiva (partindo de uma conclusão, feita pelo usuário ou pelo sistema, é feita uma pesquisa por meio do conhecimento acumulado para se provar a afirmação inicial) e progressiva (respostas fornecidas pelo usuário desencadeando um processo de busca até que se encontre a solução ótima). Ela é provocada por requisições e tem que ter base de dados de conhecimento. Maior parte baseado em lógica de primeira ordem.

b) Raciocínio Automatizado (Automated Reasoning): cria formas de simular raciocínio lógico

c) Compreensão de Linguagem Natural (Natural-Language Understanding); meios para Compreensão da linguagem humana a partir de texto, áudio ou vídeo.

d) Representação do Conhecimento (Knowledge Representation) usando Lógica de Primeira Ordem (First Logic Order): descreve objetos e predicados relacionando objetos. Admite quantificadores (∀ e ∃). Representação por lógica matemática

Veja que aprendizado de máquina vai além desses conceitos porque propõe maior independência na execução dos sistemas, com base em aprendizado contínuo como se fosse um neurônio.

Gabarito: E.

2. (FGV – Auditor de Finanças e Controle (CGU) – 2022)

Considere uma matriz de confusão de um modelo de classificação binária de relatórios financeiros. O modelo classifica os relatórios em fraudulentos ou não fraudulentos. Se essa matriz apresenta 200 verdadeiros positivos, 100 verdadeiros negativos, 40 erros do "tipo 1" e 20 erros do "tipo 2", podem-se calcular as métricas de desempenho aproximadas como:

(A) Precision = 0.71. Recall = 0.83;

(B) Precision = 0.83. Recall = 0.71;

(C) Precision = 0.83. Recall = 0.90;

(D) Precision = 0.90. Recall = 0.71;

(E) Precision = 0.90. Recall = 0.83.

Comentários

Veja que essa questão é uma aplicação direta das fórmulas de Precisão e Recall. Vamos usar a tabela abaixo para relembrar quais os termos a serem usados. Só esclarecendo que os erros do tipo 1 são os falsos positivos e os erros do tipo 2 são os falso negativos.

ACURÁCIA	PRECISÃO	RECALL
$\frac{VP + VN}{VP + FP + VN + FN}$	$\frac{VP}{VP + FP}$	$\frac{VP}{VP + FN}$
F-MEASURE	**ESPECIFICIDADE**	**TAXA DE FALSO POSITIVO**
$2 \text{ X } \frac{\text{PRECISÃO x RECALL}}{\text{PRECISÃO + RECALL}}$	$\frac{VN}{VN + FP}$	$\frac{FP}{FP + VN}$

Desta forma temos que:

$$Precisão = \frac{VP}{VP + FP} = \frac{200}{200 + 40} = 0{,}8333 \ldots$$

$$Recall = \frac{VP}{VP + FN} = \frac{200}{200 + 20} = 0{,}909090 \ldots$$

Desta forma, podemos encontrar nosso resultado na alternativa C.

Gabarito: C.

3. (FGV – Auditor de Finanças e Controle (CGU) – 2022)

Um time de ciência de dados utilizou um modelo linear para resolver uma tarefa de análise de dados financeiros provenientes de diferentes unidades de uma organização. Um membro do time, que não participou da modelagem, testa o modelo e verifica que ele apresenta um péssimo resultado. Preocupado, ele busca os resultados apresentados no treino e pode concluir que ocorreu:

(A) underfitting, se o resultado do treino foi ótimo. Uma possível solução é a utilização de um modelo mais complexo e a redução do tempo de treinamento;

(B) underfitting, se o resultado do treino também foi péssimo. Uma possível solução é a utilização de um modelo menos complexo e métodos de validação cruzada;

(C) overfitting, se o resultado do treino também foi péssimo. Uma possível solução é a utilização de técnicas de regularização e métodos de validação cruzada;

(D) overfitting, se o resultado do treino foi ótimo. Uma possível solução é a utilização de um modelo menos complexo e métodos de validação cruzada;

(E) overfitting, se o resultado do treino foi ótimo. Uma possível solução é a utilização de um modelo mais complexo e o aumento do tempo de treinamento.

Comentários

Observe que o erro só foi percebido sobre o conjunto de dados de teste. Ou seja, o modelo consegue ter um desempenho bom sobre o conjunto de dados de treinamento, entretanto, não vai conseguir generalizar bem, esse é um problema de overfitting que pode ser mitigado usando métodos de regularização (que permitem construir um modelo menos complexo). Uma das técnicas de regularização é a validação cruzada.

Para descobrir até qual complexidade devemos treinar o modelo ou ajustar os parâmetros do modelo, devemos usar a **validação cruzada**. Na validação cruzada, dividimos o conjunto de dados D em duas partições, ou seja, conjunto de treinamento denotado por T e conjunto de teste denotado por R onde a união desses dois subconjuntos é todo o conjunto de dados e a interseção deles é o conjunto vazio:

$$T \cup R = D$$

$$T \cap R = \emptyset$$

O T é usado para treinar o modelo. Depois que o modelo é treinado, o R é usado para testar o desempenho do modelo. Temos diferentes métodos para validação cruzada. Dois dos métodos mais conhecidos para validação cruzada são o K-fold e o Leave-One-Out (LOOCV).

Gabarito: D.

---

4. (FGV – Auditor de Finanças e Controle (CGU) – 2022)

Dois colegas de um time de ciência de dados discutem o novo projeto do time: avaliar um grupo de unidades de negócio e tentar, através de algumas características compartilhadas, separá-las em grupos. O objetivo é migrar de um cenário em que são elaborados contratos individuais para um cenário em que possam ser elaborados contratos por grupo. Alice acha

que deve ser usado um método supervisionado. Ela escolhe o K-means Clustering e propõe ajustar os hiperparâmetros C e sigma para alcançar um resultado adequado. Bob prefere métodos não supervisionados, já que a base de dados não possui rótulos, e está em dúvida entre utilizar Naive Bayes (em razão de a base de dados ser pequena) ou Decision Trees (por talvez ser necessário ter um modelo explicável). Analisando as posições de Alice e Bob sobre esse projeto, pode-se afirmar que:

(A) Alice e Bob estão corretos. Entretanto, não é possível realizar uma análise prévia dos algoritmos – avaliam-se apenas modelos e suas métricas de desempenho;

(B) Alice e Bob estão errados. K-means Clustering é um método não supervisionado e não possui os parâmetros C e sigma. Naive Bayes e Decision Trees são métodos supervisionados;

(C) Alice está correta, mas a sugestão de Bob de utilizar Naive Bayes é fraca, pois esse algoritmo não apresenta bom desempenho com pequenos conjuntos de dados;

(D) Bob está correto e Alice está errada. K-means Clustering é um algoritmo não supervisionado;

(E) Bob está errado e Alice está correta. Modelos baseados em Decision Trees não são explicáveis.

Comentários

Vamos comentar cada uma das alternativas:

A. ERRADO. Alice e Bob estão incorretos, eles estão confundindo os conceitos de aprendizado supervisionado e não supervisionado.

B. CERTO. K-means é um algoritmo de aprendizado não supervisionado para clusterização. Os parâmetros C e sigma são usados no SVM. Já Naive Bayes e Decision Trees são métodos de aprendizados supervisionados que podem ser usados para regressão ou classificação.

C. ERRADO. Alice e Bob estão errados. O algoritmo de Naive Bayes apresenta um bom desempenho em pequenas amostras.

D. ERRADO. Alice e Bob estão errados.

E. ERRADO. Alice e Bob estão errados

Gabarito: B.

---

5. (FGV - Auditor Federal de Controle Externo (TCU)/Controle Externo/Auditoria Governamental/2022)

Um analista do TCU recebe o conjunto de dados com covariáveis e a classe a que cada amostra pertence na tabela a seguir.

X_1	X_2	Classe
0	1	A
0	2	B
1	0	A
1	-1	B
2	2	B
1	2	A
-1	1	B
2	3	A

Esse analista gostaria de prever a classe dos pontos (1,1), (0,0) e (-1,2) usando o algoritmo de k-vizinhos mais próximos com k=3 e usando a distância euclidiana usual.

Suas classes previstas são, respectivamente:

A) A, B, A;

B) B, A, A;

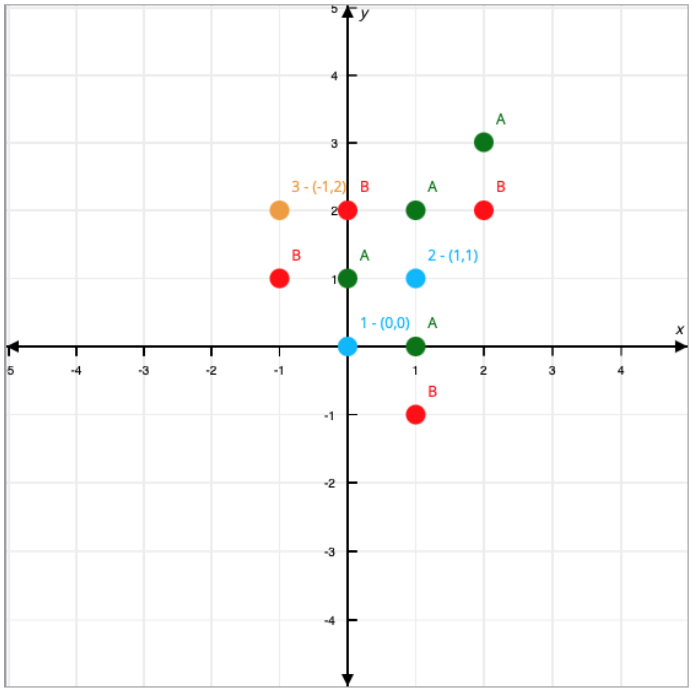
C) A, B, B;

D) A, A, B;

E) A, A, A.

Comentários

O examinador quer que se use o algoritmo k-vizinhos (ou k-neighboor). Esse algoritmo classifica uma determinada instância dos dados, comparando com os elementos mais próximos a ele. A medida de distância é um hiperparâmetro do modelo. Neste caso, o problema usa a distância euclidiana que é distância em linha reta entre dois pontos. Vamos colocar os pontos em um sistema de coordenadas cartesianas para podermos visualizar a posição e a respectiva classe. Na figura, a cor azul os pontos mais próximos de A e a laranja o ponto mais próximos de B:



O examinador é claro em dizer que estamos usando o algoritmo de k-vizinhos mais próximos com k=3, K é a quantidade de vizinhos mais próximos que vão influenciar na escolha da classe do novo elemento. Assim, podemos observar as classes de cada um dos pontos requisitados.

- **(1,1) = A**
- **(0,0) = A**
- **(-1,2) = B**

Logo, gabarito da questão **encontra-se na alternativa.**

Gabarito: D.

---

6. (FGV - Auditor Federal de Controle Externo (TCU)/Controle Externo/Auditoria Governamental/2022)

Em um problema de classificação é entregue ao cientista de dados um par de covariáveis, (x1, x2), para cada uma das quatro observações a seguir: (6,4), (2,8), (10,6) e (5,2). A variável resposta observada nessa amostra foi “Sim”, “Não”, “Sim”, “Não”, respectivamente.

A partição que apresenta o menor erro de classificação quando feita na raiz (primeiro nível) de uma árvore de decisão é:

A) $x_1 > 2$ (“Sim”) e $x1 \leq 2$ (“Não”);

B) $x_1 > 5$ (“Sim”) e $x1 \leq 5$ (“Não”);

C) $x_2 > 3$ (“Sim”) e $x2 \leq 3$ (“Não”);

D) $x_2 > 6$ (“Sim”) e $x2 \leq 6$ (“Não”);

E) $x_1 > 1$ ("Sim") e $x1 \leq 1$ ("Não").

Comentários

A questão pede para você construir uma árvore de decisão que tenha apenas um nível e a capacidade de prever com maior precisão as classe para o conjunto de informações apresentadas. Se observarmos todas as alternativas, a única que apresenta 100% de acerto para todas as instâncias ou observações apresentadas é a alternativa B. Vejamos:

Temos 4 observações: (6,4), (2,8), (10,6) e (5,2), o primeiro valor representa x1 o segundo x2.

Assim temos,

No caso (6,4), como x1 > 5, o rótulo predito é Sim

No caso (2,8), como x1 <= 5, o rótulo predito é Não

No caso (10,6), como x1 > 5, o rótulo predito é Sim

No caso (5,2), como x1 <= 5, o rótulo predito é Não

Logo, usando essa regra no nó raiz temos 100% de acerto em nossas previsões.

Gabarito: B.

---

7. (FGV - Auditor Federal de Controle Externo (TCU)/Controle Externo/Auditoria Governamental/2022)

Um analista de dados deseja criar um modelo para classificação de documentos em duas categorias: sigilosos e públicos. À sua disposição, existe um conjunto de dados com N documentos, dos quais uma fração α deles é sigilosa. O analista quer escolher uma fração β dos N documentos para pertencer ao conjunto de teste. O objetivo é garantir que cada uma das classes (documentos sigilosos e públicos) seja responsável, em média, por ao menos 10% do total de documentos. Essa restrição precisa ser válida tanto no conjunto de treino quanto no conjunto de teste.

Um par (α,β) que satisfaz as restrições do analista é:

A α = 20% e β = 70%;

B α = 30% e β = 80%;

C α = 40% e β = 60%;

D α = 60% e β = 80%;

E α = 70% e β = 20%.

Comentários

Para resolver essa questão você precisa ter em mente que existem duas variáveis alpha e beta. Beta representa a fração dos documentos que pertencem ao conjunto de teste e alpha representa os documentos sigilosos. Logo, 1 - alpha representa o percentual de documentos públicos e 1 - beta representa o conjunto de elementos de treinamento. A questão pede para que cada grupo tenha, pelo menos, 10% da amostra ... assim:

alpha * beta > 10%

alpha * (1 - beta) > 10%

(1 - alpha)* beta > 10%

(1 - alpha)*(1 - beta) > 10%

A única alternativa onde os percentuais satisfazem essa regra é a letra C onde:

alpha * beta = 24%

alpha * (1 - beta) = 16%

(1 - alpha) * beta = 36%

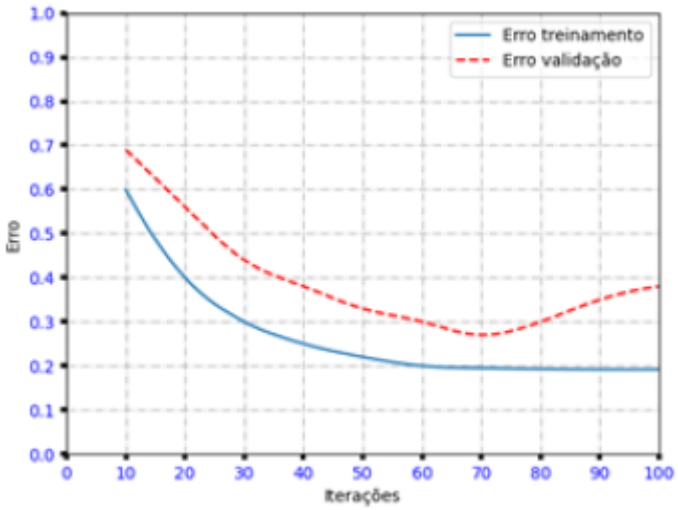
(1 - alpha) * (1 - beta) = 24%

Gabarito: C.

---

8. (FGV - Auditor Federal de Controle Externo (TCU)/Controle Externo/Auditoria Governamental/2022)

Durante o treinamento de uma rede neural artificial para classificação de imagens, foi observado o comportamento descrito pelo gráfico abaixo, que mostra a evolução do erro conforme o número de iterações.



O classificador em questão foi treinado em um conjunto de dados particionado (holdout) em 60%/30%/10% (treinamento/validação/teste). Entretanto, os especialistas envolvidos consideraram o modelo obtido insatisfatório após analisarem o gráfico.

Considerando essas informações, duas técnicas que poderiam ser utilizadas para contornar o problema encontrado são:

A Parada precoce, Minimização de Entropia Cruzada;

B Validação cruzada, Dropout;

C Sobreamostragem, Gradiente Descendente Estocástico;

D Dropout, Parada em convergência;

E Minimização de Entropia Cruzada, Validação cruzada.

Comentários

Primeiramente precisamos entender o gráfico. Este mostra duas curvas, a primeira representa o erro de treinamento e a segunda o erro de validação. Percebe-se que o erro de treinamento segue diminuindo depois 70ª interação, entretanto o erro de validação começa a aumentar. Esse é um típico problema de overfitting e acontece quando o modelo de ajusta demais aos dados treinamento, mas não generaliza tão bem.

Para resolver o problema de Overfitting podemos utilizar técnicas de regularização, aumentar a quantidade de dados de treinamento, utilizar parada antecipada do treinamento, dentre outras.

Dentre as técnicas de regularização para redes neurais podemos utilizar o Dropout. Essa técnica elimina, durante o treinamento, alguns itens que ficam ocultos. Isso faz com que os ajuste de parâmetros dos outros neurônios seja mais significativo.

Uma outra ferramenta que pode ser utilizada para essa mesma missão é a Validação Cruzada (embora seja mais conhecida por sua utilização na seleção de modelos). Nesse caso, pode-se utilizar essa técnica para avaliar os modelos por meio de treinamentos em subconjuntos. Ela pega o conjunto de treinamento e o divide em pedacinhos menores chamados de folds.

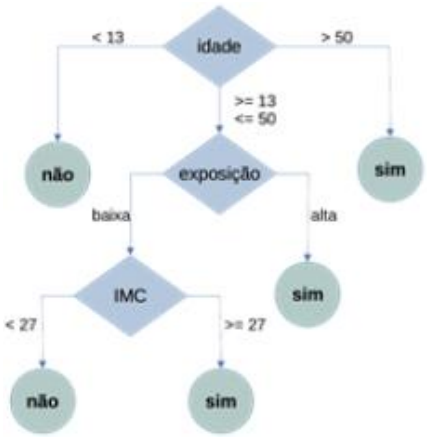
Segundo a própria FGV, " O enunciado da questão direciona à escolha de duas técnicas que possibilitam contornar o problema de overfitting apresentado no gráfico. Na ausência de informações detalhadas sobre a arquitetura da rede e sobre a qualidade dos dados apresentados a esta, a resposta deve trazer apenas técnicas que tenham como objetivo contornar o problema de overfitting, não cabendo a interpretação de que essencialmente qualquer alteração na estrutura da rede (e.g., funções de ativação, função objetivo) pode vir a mitigar o problema. A alternativa “Validação cruzada; Dropout” é a única que contém apenas técnicas utilizadas especificamente para minimizar o impacto do overfitting, no treinamento e avaliação do modelo, respectivamente."

Gabarito: B.

---

9. (Questão - FGV - Analista Judiciário (TJDFT)/Apoio Especializado/Análise de Dados/2022)

Baseado nos dados de infecções coletados durante semanas anteriores, a Secretaria de Saúde de uma pequena cidade solicitou a criação de um mecanismo para decidir sobre a priorização de vacinas na sua população.



Como resultado, foi criado o modelo de árvore de decisão ilustrado a seguir.

Uma evidência de que esse modelo foi construído usando o algoritmo C4.5 ou uma de suas variantes é:

A a diferença entre a profundidade mínima e máxima da árvore;

B os valores dos atributos “idade” e “IMC”;

C a árvore não ser binária;

D o tipo das folhas;

E a quantidade de atributos.

Comentários

C4.5 constrói árvores de decisão a partir de um conjunto de dados de treinamento da mesma forma que o algoritmo *ID3*, utilizando o conceito de ***Entropia***. O conjunto de dados de treinamento é um conjunto de amostras já classificadas. Cada amostra consiste de um vetor p-dimensional, com valores de atributos ou características da amostra, assim como a categoria ou a classe a qual pertence.

Em cada nó da árvore, o algoritmo C4.5 **escolhe o atributo dos dados que mais efetivamente particiona o seu conjunto de amostras em subconjuntos tendendo a uma categoria ou a outra**. O critério de particionamento é o **ganho de informação normalizado** (diferença em *entropia*). O atributo com maior ganho de informação normalizado é escolhido para tomar a decisão. O algoritmo C4.5 então repete a etapa anterior nas partições menores.

Este algoritmo possui alguns ações básicas:

- Quando todas as amostras do conjunto pertencem a uma mesma categoria, o algoritmo simplesmente cria um nó folha para a árvore de decisão e escolhe a categoria em questão;
- Quando nenhuma das características fornece ganho de informação, o algoritmo C4.5 cria um nó de decisão árvore acima usando o valor esperado;

- Quanto as instâncias previamente não vistas ou visitadas. Novamente, o algoritmo C4.5 cria um nó de decisão árvore acima usando o valor esperado.

O algoritmo de mineração de dados C4.5 foi desenvolvido por Ross Quinlan. O C4.5 gera Árvores de Decisão (DT), que podem ser utilizadas para classificação do conjunto de dados. C4.5 estende o algoritmo ID3 (que focava mais em atributos discretos) para trabalhar com mais naturalidade com atributos contínuos e discretos. C4.5 também lida com valores perdidos e poda de árvores após a construção.

Gabarito: B.

QUESTIONÁRIO DE REVISÃO E APERFEIÇOAMENTO

A ideia do questionário é elevar o nível da sua compreensão no assunto e, ao mesmo tempo, proporcionar uma outra forma de revisão de pontos importantes do conteúdo, a partir de perguntas que exigem respostas subjetivas.

São questões um pouco mais desafiadoras, porque a redação de seu enunciado não ajuda na sua resolução, como ocorre nas clássicas questões objetivas.

O objetivo é que você realize uma autoexplicação mental de alguns pontos do conteúdo, para consolidar melhor o que aprendeu ;)

Além disso, as questões objetivas, em regra, abordam pontos isolados de um dado assunto. Assim, ao resolver várias questões objetivas, o candidato acaba memorizando pontos isolados do conteúdo, mas muitas vezes acaba não entendendo como esses pontos se conectam.

Assim, no questionário, buscaremos trazer também situações que ajudem você a conectar melhor os diversos pontos do conteúdo, na medida do possível.

É importante frisar que não estamos adentrando em um nível de profundidade maior que o exigido na sua prova, mas apenas permitindo que você compreenda melhor o assunto de modo a facilitar a resolução de questões objetivas típicas de concursos, ok?

Nosso compromisso é proporcionar a você uma revisão de alto nível!

Vamos ao nosso questionário:

Perguntas

1) Como você definiria machine learning?
2) Você pode citar quatro tipos de problemas onde podemos usar ML?
3) O que é um conjunto de treinamento rotulado?
4) Quais são as duas tarefas supervisionadas mais comuns?
5) Você pode nomear quatro tarefas não supervisionadas comuns?

6) **Que tipo de algoritmo de Machine Learning você usaria para permitir que um robô andasse em vários terrenos desconhecidos?**
7) **Que tipo de algoritmo você usaria para segmentar seus clientes em vários grupos?**
8) **Você enquadraria o problema da detecção de spam como um problema de aprendizagem supervisionado ou um problema de aprendizagem não supervisionado?**
9) **O que é um sistema de aprendizagem online?**
10) **O que é aprendizado fora do núcleo?**
11) **Que tipo de algoritmo de aprendizagem depende de uma medida de similaridade para fazer previsões?**
12) **Qual é a diferença entre um parâmetro modelo e o hiperparâmetro de um algoritmo de aprendizagem?**
13) **O que os algoritmos de aprendizagem baseados em modelos buscam? Qual é a estratégia mais comum que eles usam para ter sucesso? Como eles fazem previsões?**
14) **Você pode citar quatro dos principais desafios do Machine Learning?**
15) **Se o seu modelo se sai bem nos dados de treinamento, mas se generaliza mal para novas instâncias, o que está acontecendo? Você pode nomear três soluções possíveis?**
16) **O que é um conjunto de testes, e por que você quer usá-lo?**
17) **Qual é o propósito de um conjunto de validação?**
18) **O que pode dar errado se você sintonizar hiperparâmetros usando o conjunto de testes?**

Perguntas com respostas

1) **Como você definiria machine learning?**

Machine Learning é sobre construir sistemas que possam aprender com dados. Aprender significa melhorar em alguma tarefa, dada alguma medida de desempenho.

2) **Você pode citar quatro tipos de problemas onde podemos usar ML?**

O Machine Learning é ótimo para problemas complexos para os quais não temos solução algorítmica, para substituir longas listas de regras afinadas à mão, para construir sistemas que se adaptem a ambientes flutuantes e, finalmente, para ajudar os humanos a aprender (por exemplo, a mineração de dados).

3) **O que é um conjunto de treinamento rotulado?**

Um conjunto de treinamento rotulado é um conjunto de treinamento que contém a solução desejada (também conhecida como um rótulo) para cada instância.

4) **Quais são as duas tarefas supervisionadas mais comuns?**

As duas tarefas supervisionadas mais comuns são regressão e classificação.

5) **Você pode nomear quatro tarefas não supervisionadas comuns?**

Tarefas não supervisionadas comuns incluem agrupamento, visualização, redução de dimensionalidade e aprendizado de regras de associação.

6) Que tipo de algoritmo de Machine Learning você usaria para permitir que um robô andasse em vários terrenos desconhecidos?

O Aprendizado de Reforço provavelmente será melhor se quisermos que um robô aprenda a andar em vários terrenos desconhecidos, já que este é tipicamente o tipo de problema que o Aprendizado de Reforço enfrenta. Pode ser possível expressar o problema como um problema de aprendizagem supervisionado ou semi-supervisionado, mas seria menos natural.

7) Que tipo de algoritmo você usaria para segmentar seus clientes em vários grupos?

Se você não sabe como definir os grupos, então você pode usar um algoritmo de clustering (aprendizado não supervisionado) para segmentar seus clientes em clusters de clientes semelhantes. No entanto, se você sabe quais grupos você gostaria de ter, então você pode alimentar muitos exemplos de cada grupo para um algoritmo de classificação (aprendizado supervisionado), e classificará todos os seus clientes nesses grupos.

8) Você enquadraria o problema da detecção de spam como um problema de aprendizagem supervisionado ou um problema de aprendizagem não supervisionado?

A detecção de spam é um típico problema de aprendizagem supervisionada: o algoritmo é alimentado com muitos e-mails junto com seus rótulos (spam ou não spam).

9) O que é um sistema de aprendizagem online?

Um sistema de aprendizagem on-line pode aprender incrementalmente, em oposição a um sistema de aprendizagem em lote. Isso o torna capaz de se adaptar rapidamente tanto à mudança de dados quanto aos sistemas autônomos, e ao treinamento em grandes quantidades de dados.

10) O que é aprendizado fora do núcleo?

Algoritmos fora do núcleo podem lidar com grandes quantidades de dados que não podem se encaixar na memória principal de um computador. Um algoritmo de aprendizagem fora do núcleo corta os dados em mini-lotes e usa técnicas de aprendizagem on-line para aprender com esses mini-lotes.

11) Que tipo de algoritmo de aprendizagem depende de uma medida de similaridade para fazer previsões?

Um sistema de aprendizagem baseado em instâncias aprende os dados de treinamento por semelhanças; então, quando uma nova instância é informada, ele usa uma medida de similaridade para encontrar as instâncias aprendidas mais semelhantes e as usa para fazer previsões.

12) Qual é a diferença entre um parâmetro modelo e o hiperparâmetro de um algoritmo de aprendizagem?

Um modelo tem um ou mais parâmetros de modelo que determinam o que ele vai prever dada uma nova instância (por exemplo, a inclinação de um modelo linear). Um algoritmo de aprendizagem tenta encontrar valores ideais para esses parâmetros de tal forma que o modelo generaliza bem para novas instâncias. Um hiperparâmetro é um parâmetro do algoritmo de aprendizagem em si, não do modelo (por exemplo, a quantidade de regularização para aplicar).

13) O que os algoritmos de aprendizagem baseados em modelos buscam? Qual é a estratégia mais comum que eles usam para ter sucesso? Como eles fazem previsões?

Algoritmos de aprendizagem baseados em modelos buscam um valor ideal para os parâmetros do modelo, de tal forma que o modelo generalize bem para novas instâncias. Geralmente treinamos esses sistemas minimizando uma função de custo que mede o quão ruim o sistema é em fazer previsões sobre os dados de treinamento, além de uma penalidade pela complexidade do modelo se o modelo for regularizado. Para fazer previsões, alimentamos as características da nova instância na função de previsão do modelo, e calculamos usando os valores dos parâmetros encontrados pelo algoritmo de aprendizagem.

14) Você pode citar quatro dos principais desafios do Machine Learning?

Alguns dos principais desafios no Machine Learning são a falta de dados, má qualidade dos dados, dados não representativos, características não informativas, modelos excessivamente simples que subestimam os dados de treinamento e modelos excessivamente complexos que superpõem os dados.

15) Se o seu modelo se sai bem nos dados de treinamento, mas se generaliza mal para novas instâncias, o que está acontecendo? Você pode nomear três soluções possíveis?

Se um modelo se sai bem nos dados de treinamento, mas se generaliza mal para novas instâncias, o modelo provavelmente está sobreajustado (overfitting) os dados de treinamento. Possíveis soluções para o sobreajuste são obter mais dados, simplificar o modelo (selecionar um algoritmo mais simples, reduzir o número de parâmetros ou recursos utilizados ou regularizar o modelo) ou reduzir o ruído nos dados de treinamento.

16) O que é um conjunto de testes, e por que você quer usá-lo?

Um conjunto de testes é usado para estimar o erro de generalização que um modelo fará em novas instâncias, antes do modelo ser lançado em produção.

17) Qual é o propósito de um conjunto de validação?

Um conjunto de validação é usado para comparar modelos. Torna possível selecionar o melhor modelo e sintonizar os hiperparâmetros.

18) O que pode dar errado se você sintonizar hiperparâmetros usando o conjunto de testes?

Se você sintonizar hiperparâmetros usando o conjunto de testes, você corre o risco de se adaptar demais ao conjunto de testes, e o erro de generalização que você mede será otimista (você pode lançar um modelo que tenha um desempenho pior do que você espera).

Forte abraço e bons estudos.

"Hoje, o 'Eu não sei', se tornou o 'Eu ainda não sei'"

(Bill Gates)

Thiago Cavalcanti

Face: www.facebook.com/profthiagocavalcanti
Insta: www.instagram.com/prof.thiago.cavalcanti
YouTube: youtube.com/profthiagocavalcanti

ESSA LEI TODO MUNDO CONHECE:
PIRATARIA É CRIME.

Mas é sempre bom revisar o porquê e como você pode ser prejudicado com essa prática.

1. Professor investe seu tempo para elaborar os cursos e o site os coloca à venda.

2. Pirata divulga ilicitamente (grupos de rateio), utilizando-se do anonimato, nomes falsos ou laranjas (geralmente o pirata se anuncia como formador de "grupos solidários" de rateio que não visam lucro).

3. Pirata cria alunos fake praticando falsidade ideológica, comprando cursos do site em nome de pessoas aleatórias (usando nome, CPF, endereço e telefone de terceiros sem autorização).

4. Pirata compra, muitas vezes, clonando cartões de crédito (por vezes o sistema anti-fraude não consegue identificar o golpe a tempo).

5. Pirata fere os Termos de Uso, adultera as aulas e retira a identificação dos arquivos PDF (justamente porque a atividade é ilegal e ele não quer que seus fakes sejam identificados).

6. Pirata revende as aulas protegidas por direitos autorais, praticando concorrência desleal e em flagrante desrespeito à Lei de Direitos Autorais (Lei 9.610/98).

7. Concurseiro(a) desinformado participa de rateio, achando que nada disso está acontecendo e esperando se tornar servidor público para exigir o cumprimento das leis.

8. O professor que elaborou o curso não ganha nada, o site não recebe nada, e a pessoa que praticou todos os ilícitos anteriores (pirata) fica com o lucro.

Deixando de lado esse mar de sujeira, aproveitamos para agradecer a todos que adquirem os cursos honestamente e permitem que o site continue existindo.