INSTITUTO DE GESTÃO E TECNOLOGIA DA INFORMAÇÃO

Metodologias de Aprendizado

Ricardo Brito Alves

2021

Metodologias de Aprendizado

Ricardo Brito Alves

Sumário

Capítulo 1. Fundamentos de Aprendizado

O objetivo deste capítulo é abordar alguns temas importantes ao Aprendizado de Máquina. É importante descrever alguns conceitos como Big Data e análise de dados. Será de muita valia quando entrarmos mais a fundo no Aprendizado de Máquina.

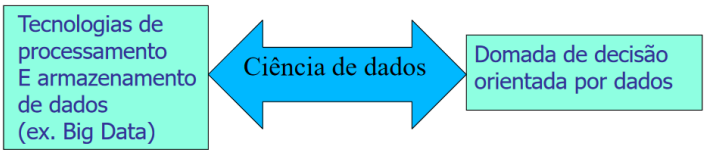
Big Data

O *Big Data* é um conceito que descreve o grande volume de dados estruturados e não estruturados, que são gerados de forma exponencial. Nos dias atuais, geramos um grande volume de dados com diversos dispositivos. Podemos citar as mídias sociais, que geram a todo tempo informações. Também é uma realidade a existência de carros, geladeiras e dispositivos vestíveis (*wearable devices*) conectados entre si e gerando ainda mais dados.

Ciência de Dados

Ciência de Dados ou *Data Science* é uma área interdisciplinar voltada para o estudo e a análise de dados, estruturados ou não, que visa a extração de conhecimento ou indícios de informações para possíveis tomadas de decisão, de maneira similar à mineração de dados. Ciência de Dados alia Big Data e Aprendizado de Máquina, além de técnicas de outras áreas interdisciplinares como estatística, economia, engenharia e outros subcampos da computação como: banco de dados e análise de agrupamentos (cluster analysis). A Ciência de Dados é um campo que já existe há 30 anos, porém ganhou mais destaque nos últimos anos devido a alguns fatores como: o surgimento e popularização do Big Data e o desenvolvimento de áreas como o Aprendizado de Máquina.

Figura 1 - Ciência de Dados.



Big Data e Data Mining

O diferencial do Big Data está justamente atrelado à possibilidade de cruzar dados por meio de diversas fontes para obtermos mais informações relevantes. Por exemplo, conseguimos obter informações de mercado por meio de nossos consumidores, insatisfações, satisfações, desejos, necessidades, entre outros. É aí que entra o *Data Mining,* ou mineração de dados. É o processo de explorar grandes quantidades de dados à procura de padrões consistentes, como regras de associação ou sequências temporais, para detectar relacionamentos sistemáticos entre variáveis, detectando assim novos subconjuntos de dados.

A mineração de dados é formada por um conjunto de ferramentas e técnicas que através do uso de algoritmos de aprendizagem, ou classificação, baseados em rede neural e estatística, são capazes de explorar um conjunto de dados, extraindo ou ajudando a evidenciar padrões nestes dados e auxiliando na descoberta de conhecimento. Esse conhecimento pode ser apresentado por essas ferramentas de diversas formas: agrupamentos, hipóteses, regras, árvores de decisão, grafos ou dendrogramas. O ser humano sempre aprendeu observando padrões, formulando hipóteses e testando-as para descobrir regras. A novidade da era do computador é o volume enorme de dados que não pode mais ser examinado à procura de padrões em um prazo de tempo razoável.

Data Mining e Business Intelligence

Diariamente as empresas acumulam grande volume de dados em seus aplicativos operacionais. São dados brutos que dizem quem comprou o quê, onde, quando e em que quantidade. É a informação vital para o dia a dia da empresa. Se fizermos estatística ao final do dia para repor estoques e detectar tendências de compra, praticaremos *Business Intelligence* (BI). Se analisarmos os dados com estatística de modo mais refinado, à procura de padrões de vinculações entre as variáveis registradas, então faremos mineração de dados (MD). Buscamos com a MD conhecer melhor os clientes, seus padrões de consumo e motivações. A mineração de dados resgata em grandes organizações o papel do dono, atendendo no balcão e conhecendo sua clientela. Esses dados agora podem agregar valor às decisões da empresa, sugerir tendências, desvendar particularidades dela e de seu meio ambiente e permitir ações mais bem informadas aos seus gestores.

Pode-se então diferenciar o *Business Intelligence* da mineração de dados como dois patamares distintos de atuação. O BI visa obter, a partir dos dados operativos brutos, informação útil para subsidiar a tomada de decisão nos escalões médios e altos da empresa. A MD busca subsidiar a empresa com conhecimento novo e útil acerca do seu meio ambiente. O primeiro funciona no plano tático, o segundo no estratégico.

Pré-processamento de dados

As técnicas de pré-processamento de dados geralmente se referem à adição, exclusão ou transformação dos dados do conjunto de treinamento. Embora se deva preocupar principalmente com técnicas de modelagem, a preparação de dados pode fazer ou quebrar um modelo de sua capacidade preditiva. Diferentes modelos têm diferentes sensibilidades em relação ao tipo de preditores no modelo, afinal a forma como os preditores entram também é importante.

A transformações dos dados para reduzir o impacto da assimetria de dados ou *outliers*, podem levar a melhorias significativas no desempenho. A extração de recursos é uma técnica empírica para se criar variáveis substitutas que são combinações de múltiplos preditores. Além disso, estratégias mais simples tais como remover preditores com base na sua falta de conteúdo de informação, pode também ser eficaz. A necessidade de pré-processamento de dados é determinada pelo tipo de modelo a ser utilizado. Alguns procedimentos, como modelos baseados em árvore, são notavelmente insensíveis às características dos dados do preditor. Outros, como a regressão linear, não são.

As transformações de dados atuam em grupos de preditores, que geralmente levam o conjunto inteiro sob consideração. Os métodos para resolver *outliers* e reduzir a dimensão dos dados são sua importância primordial.

Em geral, *outliers* são definidos como amostras excepcionalmente distantes da corrente principal (*mainstream*) dos dados. Sob certas hipóteses, existem estatísticas formais para definições de um *outlier*. Mesmo com uma compreensão completa dos dados, outliers podem ser difíceis de serem definidos ou detectados. No entanto, muitas vezes pode-se identificar um valor incomum olhando apenas para um gráfico. Quando uma ou mais amostras são suspeitas de serem *outliers*, o primeiro passo é garantir que os valores sejam cientificamente válidos, por exemplo uma pressão arterial que deve ser sempre positiva, e que nenhum erro ocorreu na gravação de dados. Deve-se tomar muito cuidado para remover ou alterar valores, especialmente se o tamanho da amostra é pequeno. Com amostras pequenas, aparente outliers podem ser o resultado de uma distribuição distorcida, onde ainda não há dados suficientes para se analisar a assimetria. Além disso, os dados periféricos podem ser uma indicação de uma parte especial da população em estudo, que está apenas começando a ser amostrada.

Dependendo de como os dados foram coletados, um cluster de pontos válidos que residem fora da corrente principal dos dados pode pertencer a uma diferente população em relação às outras amostras. Existem vários modelos preditivos que são resistentes aos outliers. Modelos de classificação baseados em árvores criam

divisões dos dados de treinamento e a equação preditiva é um conjunto de instruções lógicas, como se o preditor A fosse maior do que X prevê que a classe seja Y, então o outlier normalmente não tem uma influência excepcional no modelo. Além disso, a classificação através de máquinas de vetores de suporte geralmente desconsidera uma parte das amostras do conjunto de treinamento, criando uma equação de previsão. As amostras excluídas podem estar longe do limite de decisão e fora da corrente principal de dados. Se um modelo é considerado sensível a outliers, uma transformação de dados pode minimizar o problema. Esse procedimento projeta os valores dos preditores em uma esfera multidimensional e isto faz com que todas as amostras tenham mesma distância do centro da esfera.

Como o denominador é destinado a medir a distância ao quadrado do centro da distribuição do preditor, é importante centralizar e dimensionar os dados do preditor antes de usar essa transformação. Note que, ao contrário de centralização ou escalonamento, essa manipulação dos preditores os transforma como um grupo.

Técnicas de redução de dados é outra classe de transformações da previsão. A redução de dimensionalidade é uma das tarefas mais importantes para a análise multivariada e é especialmente crítica para regressões multivariadas. Esses métodos reduzem os dados gerando um conjunto menor de preditores, que buscam capturar a maioria das informações nas variáveis originais. Dessa forma, menos variáveis podem ser usadas para fornecer fidelidade razoável aos dados originais. Para a maioria das técnicas de redução de dados, os novos preditores são funções dos preditores originais, portanto, todos os preditores originais ainda são necessários para criar as variáveis substitutas. Essa classe de métodos é frequentemente chamada de extração de sinal ou técnicas de extração de recursos. Tem-se como redutor de dimensionalidade o *Principal Component Analysis* (PCA) ou *Partial Least Squares* (PLS). Os redutores de dimensionalidade podem ser aplicados sobre os vetores de características gerados pelos descritores, obtendo descrições de características mais compactas, porém, ainda assim acuradas. O PCA é uma técnica comumente usada.

Conhecidos como Componentes Principais (PCs), este método procura encontrar combinações lineares dos preditores que capturam a maior variação

possível. O primeiro PC é definido como a combinação linear dos preditores, que captura a maior variabilidade de todas as combinações lineares possíveis. Então, os PCs subsequentes são derivados de tal forma que essas combinações lineares capturem a maior variabilidade remanescente, enquanto também não são correlacionadas com todos os PCs anteriores.

Há vantagens em potencial para remover os preditores menos significativos antes da modelagem. Primeiro: menos preditores significam menor tempo computacional e complexidade. Segundo: se dois preditores são altamente correlacionados, isso implica que eles estão medindo a mesma informação subjacente. A remoção de um preditor não deve comprometer o desempenho do modelo e pode levar a uma análise mais parcimoniosa do modelo interpretável. Em terceiro: alguns modelos podem ser afetados por preditores com distribuições degeneradas. Nesses casos pode haver uma melhoria significativa no desempenho e/ou estabilidade do modelo sem as variáveis problemáticas.

Considere uma variável de previsão que tenha um único valor único. Refere-se a esse tipo de dados como um preditor de variância zero. Para alguns modelos, uma variável não informativa pode ter pouco efeito nos cálculos. Um modelo baseado em árvore é impermeável a esse tipo de preditor, uma vez que nunca seria usado em uma divisão. No entanto, um modelo como a regressão linear acharia esses dados problemáticos e provavelmente causaria um erro nos cálculos. Em ambos os casos esses dados não têm informações e podem ser facilmente descartados. Da mesma forma, alguns preditores podem ter apenas um punhado de valores únicos que ocorrem com frequências muito baixas. Esses preditores de variância próximos de zero podem ter um único valor para a grande maioria das amostras.

Seleção de Dados

No aprendizado de máquina e estatística, a seleção de atributos, também conhecida como seleção de variáveis ou seleção de subconjuntos de variáveis, é o processo de seleção de um subconjunto de recursos relevantes (variáveis e

preditores) para uso na construção de modelos. As técnicas de seleção de recursos são usadas por três razões:

- Simplificação de modelos para torná-los mais fáceis de interpretar por pesquisadores/usuários.
- Menores tempos de treinamento.
- Evitar a maldição da dimensionalidade, generalização melhorada reduzindo *overfitting* (formalmente, redução da variação).

A premissa central ao usar uma técnica de seleção de recursos é que os dados contêm alguns recursos que são redundantes ou irrelevantes e podem, portanto, serem removidos sem incorrer em muita perda de informações. Redundantes e irrelevantes são duas noções distintas, uma vez que uma característica relevante pode ser redundante na presença de outra característica relevante com a qual ela está fortemente correlacionada.

As técnicas de seleção de recursos devem ser diferenciadas da extração de recursos. A extração de recursos cria novos recursos a partir das funções dos recursos originais, enquanto a seleção de recursos retorna um subconjunto dos recursos. As técnicas de seleção de recursos costumam ser usadas em domínios em que há muitos recursos e comparativamente poucas amostras (ou pontos de dados).

A técnica de *Correlation Feature Selection* (CFS) é normalmente uma etapa importante na utilização de métodos de aprendizado de máquina e temos razões para tal. Conjuntos de dados são frequentemente descritos com muitas variáveis e algumas dessas variáveis são irrelevantes para a classificação. Obviamente, sua relevância não é conhecida de antemão. Existem várias desvantagens em lidar com grandes conjuntos de variáveis. Tecnicamente falando, lidar com grandes conjuntos de recursos retarda a performance de processamento dos algoritmos, pois exigem recursos e são simplesmente inconvenientes. Além disso, muitos algoritmos de aprendizado de máquina exibem uma diminuição de precisão quando o número de variáveis é significativamente superior ao ideal. Portanto, manter uma seleção do conjunto possivelmente mínimo possibilitará um melhor resultado de classificação.

Esse problema, conhecido como problema ótimo, tem sido intensamente estudado e existem muitos algoritmos para tal fim.

No entanto, manter uma seleção do conjunto possivelmente mínimo obscurece outro problema muito interessante, o da identificação de todos os atributos que são, em algumas circunstâncias, relevantes para a classificação, os chamados “problema relevante”. Encontrar todos os atributos relevantes, em vez de apenas os não redundantes, pode ser muito útil. Particularmente isso é necessário quando se está interessado em entender mecanismos relacionados ao assunto de interesse, em vez de simplesmente construir um modelo preditivo. Por exemplo, no contexto do câncer é necessário a identificação de todos os genes relacionados à doença para uma compreensão completa do processo, enquanto um conjunto de genes mínimo-ótimo pode ser mais útil como marcadores genéticos.

O problema mais relevante da seleção de recursos é mais difícil do que o usual. Uma razão é que não podemos confiar na precisão da classificação como critério para selecionar o recurso como importante ou como sem importância. A degradação da classificação na precisão, após a remoção do recurso do conjunto de recursos, é suficiente para declarar o recurso importante, mas a falta deste efeito não é suficiente para declará-lo sem importância. Portanto, é necessário outro critério para declarar variáveis importantes ou sem importância. Ademais, a falta de correlação direta entre um determinado recurso e a decisão não é uma prova de que esse recurso não é importante em conjunto. Em um método *wrapper* o classificador é usado como uma caixa preta, retornando uma classificação onde se pode usar qualquer classificador que forneça o ranking de recursos.

Dados ruidosos

O conjunto de dados da vida real é gerado usando sinais de uma variedade de sensores e alguns dispositivos de hardware. Quando os dados estão sendo coletados, pode haver "ruído" no conjunto de dados devido a alguns hardwares com limitações e alguns erros na calibração de hardware. Esses pontos de dados

indesejados são chamados de ruído e afetam a precisão de um modelo de aprendizado de máquina.

Por exemplo, quando você está gravando algum som, o ruído de fundo são alguns dados desnecessários que são coletados. Como alternativa, quando um algoritmo de classificação está tentando classificar qualquer fotografia na biblioteca usando algumas ferramentas de reconhecimento de imagem, os objetos desnecessários nessas fotografias precisam ser ignorados ou eliminados para garantir que a máquina possa identificar e classificar a fotografia com precisão. Vários métodos são usados para minimizar esse tipo de ruído, como agrupamento, binning, etc.

Um modelo de Binning consegue isso classificando as informações e suavizando-as com base em alguns pontos de dados vizinhos, e esse processo é chamado de processo de suavização de dados local. Um algoritmo de agrupamento é útil para eliminar os valores discrepantes. Vejamos o exemplo a seguir para entender o conceito de outliers.

Vamos supor que o modelo é uma pessoa francesa que está viajando para os Estados Unidos. O modelo precisará ser treinado para falar e entender inglês. Como Shakespeare é um dos maiores dramaturgos e poetas de todos os tempos, você pode ensinar a máquina tudo sobre inglês usando a poesia ou os dramas de Shakespeare. Você pode supor que esta é a melhor maneira de ensinar inglês ao modelo.

Quando o modelo chega a Nova York e testa Shakespeare em inglês, o modelo visualiza que esse idioma não é mais entendido. O modelo não pode responder aos habitantes locais, apesar de falarem inglês também. Isso significa que você alterou o modelo com dados relevantes e o modelo não é flexível o suficiente para se adaptar aos novos dados.

Este é um exemplo de um modelo de ajuste excessivo, com alta variação e baixo viés. Portanto, este modelo não ajuda a máquina a treinar para novos conjuntos de dados. Isso ocorre porque o conjunto de dados de treinamento é obra de

Shakespeare, enquanto o conjunto de dados de teste é o uso de inglês coloquial em Nova York. Se você testar o desempenho do modelo com base na aceitação social, aprenderá que os modelos falham em termos de generalização.

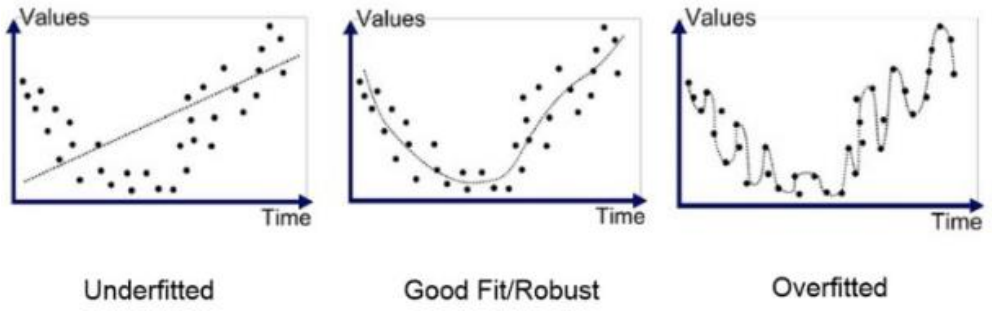
Usando o exemplo acima, você deve entender agora que a variação determinará a capacidade de um modelo responder a qualquer conjunto de dados fornecido a ele. O modelo acima tem um interesse de alta variação, pois depende exclusivamente do conjunto de dados de treinamento. Portanto, pode-se entender que um modelo com alta variação nunca funcionará bem com um novo conjunto de dados de teste.

O viés é uma medida da força das suposições que você faz sobre o ponto de conjunto de dados no exemplo acima, confiamos que o trabalho de Shakespeare será suficiente para treinar o modelo para aprender inglês. É o caso de um viés baixo. Deve-se sempre ser cético em relação ao conjunto de dados e nunca fazer suposições que não sejam copiadas pelo motivo.

O caso de um viés baixo em alta variância é denominado como um modelo super ajustado. Em palavras simples, um modelo sempre tem uma troca equilibrada entre o viés e a variação. A variação é uma medida do modelo, enquanto a base é uma medida da força do conjunto de dados.

Vamos agora ver o que acontecerá se a equação for revertida para um modelo de alto viés e variância. Esse tipo de modelo é chamado de modelo insuficiente. Esse tipo de modelo sempre prejudica o aprendizado, são as lições do conjunto de dados de treinamento e não entende a relação entre as variáveis de entrada e as variáveis de saída que levam a resultados imprecisos.

Figura 2 – Gráfico Underfitted x Overfitte.



Vamos tentar entender isso com o exemplo acima.

Vamos fazer algumas suposições sobre o conjunto de dados e mudar o conjunto de dados de treinamento de Shakespeare para uma série popular chamada Casa das Cartas para aprender inglês. De acordo com nossa experiência, solicitaremos ao modelo que escolha algumas frases em um formato específico. Por exemplo, você pode solicitar que o modelo escolha uma frase que comece com um artigo e ignore todas as outras frases. Agora que você alterou o modelo usando esse conjunto de dados aprimorado, pode esperar que o resultado seja um pouco melhor do que as ruas de Nova York. A conversa não será significativa, pois você não pediu ao modelo para entender a estrutura fundamental da linguagem. Isso ocorre devido ao alto viés do conjunto de dados.

Este é um exemplo simples dos conceitos de ajuste excessivo e insuficiente de um modelo e os problemas de polarização e variação que são abordados usando vários métodos, como validação cruzada. Nesse método, o conjunto de dados de treinamento é dividido em grupos menores e validado cruzadamente várias vezes.

Em suma:

- Ajuste excessivo: este é um modelo que depende muito do conjunto de dados de treinamento.
- Sob adaptação: este modelo não entende nenhuma relação de sublinhado entre as variáveis no conjunto de dados de treinamento.

- Viés alto: é o caso em que nossas suposições levam a ignorar os fatores sobre os dados de treinamento.
- Alta variação: este é um caso em que a resposta do modelo é significativamente afetada por uma pequena alteração no conjunto de dados de treinamento.
- Sobre ajuste e sub adequação resultarão em generalização ruim, portanto, um conjunto de validação sempre deve ser usado para ajustar o modelo para superar essas limitações.

Dados ausentes

Um segundo problema com conjuntos de dados após dados ruidosos é o caso de dados ausentes. Um conjunto de dados geralmente é alimentado no modelo usando tuplas, que são essencialmente um conjunto de recursos e atributos.

Se um registro específico tiver dados ausentes em relação a atributos, esses dados não serão úteis e o caso de dados ausentes. Uma maneira de resolver isso é varrer o conjunto de dados e ignorar todas as tuplas com atributos de dados incomuns ou ausentes. Porém, essa não é a maneira mais fácil e eficiente de resolver esse problema.

A alternativa é identificar as tuplas de dados ausentes e preencher manualmente os valores. Isso aumentará a precisão do conjunto de dados, mas não é uma solução viável se você estiver trabalhando com grandes volumes de dados. Este é um processo demorado e ineficiente. Uma solução mais sofisticada é examinar o conjunto de dados e prever quais podem ser os valores ausentes. Isso pode levar a um viés nos dados, pois as previsões nem sempre serão precisas, levando a algumas diferenças. Você sempre pode verificar e comparar os resultados obtidos, fazendo algumas alterações nos registros ausentes nas tuplas.

A última opção é fazer algumas inferências baseadas nos algoritmos. Você pode usar alguns algoritmos comuns para inferir ou prever qual pode ser um valor ausente. Os algoritmos comuns usados são árvores de decisão e fórmulas bayesianas.

Análise preditiva e descritiva com Aprendizado de Máquina

A Aprendizado de Máquina automatiza a criação de modelos analíticos. Modelos analíticos são utilizados para predições, simulações e análise de dados. Dentro da Inteligência Artificial, faz com que sistemas aprendam com dados, identifiquem padrões, de forma rápida e automática, e auxiliem na tomada de decisão. A Aprendizado de Máquina já existe há muito tempo, mas, recentemente, com a grande quantidade de dados digitais disponíveis, vem se tornando cada vez mais popular. Nesse contexto, a aprendizagem de máquina pode agir basicamente de duas formas, auxiliando na descrição dos dados ou até mesmo na predição deles.

A análise preditiva é indicada para tarefas em que é necessário prever algum comportamento ou resultado. Para tal, é feita uma análise de dados ao longo de um determinado período e a partir disso buscam-se padrões comportamentais, bem como variações em cada contexto, tais como dia, mês, temperatura, semestre, cotação do dólar, para prever como será o comportamento no futuro dado às condições atuais.

Este tipo de análise é útil para verificar tendências de consumo e flutuações econômicas. Por exemplo, é possível prever resultados de uma determinada ação de lançamento de um produto, prever o comportamento de clientes ou até mesmo a quantidade de insumos que é necessária para uma indústria nos próximos meses. Para isso, é feito um treinamento com dados históricos que servem de base para as predições.

Ela pode se dar de duas formas:

- Classificação: quando a análise busca aprender uma função que associa descrição de um exemplo a uma classe, ou seja, a definição de rótulos ou

etiquetas. Por exemplo, pode-se definir se um paciente vai ter uma doença ou não, mensurar a qualidade de um insumo, predizer se um cliente vai ser inadimplente ou qualquer outra coisa que envolva classificação em rótulos previamente definidos.

- Regressão: é utilizada quando se deseja prever valores reais. Por exemplo, quantidade de venda de um produto, lucros, quantidade de insumos necessários para um período de tempo, número de produtos com defeito, etc.

Já a análise descritiva busca a compreensão em tempo real dos acontecimentos. Ao invés de focar no futuro, busca uma fotografia do presente para tomadas de decisão de cunho imediato. Trabalha com análise de dados históricos e cruzamento de informações para gerar um panorama claro e preciso para o momento. Boa maneira de visualizar os dados e entender o presente. Na análise de crédito, por exemplo, analisa dados de pessoas e grupos sociais para definir o risco envolvido na concessão de um determinado crédito.

Uma das principais técnicas utilizadas para análise descritiva é a de agrupamento. O objetivo principal desta técnica é agrupar elementos utilizando medidas de similaridade. Esses elementos podem ser produtos, clientes, insumos ou qualquer coisa que se deseja analisar e que possuam características que permitam o agrupamento. Através disso, é possível isolar problemas e entender melhor a distribuição dos elementos, agrupar elementos é uma operação muito difícil para humanos quando se tem muitos elementos e características envolvidos.

Os dois tipos de análise podem ser utilizados em conjunto para fazer com que gestores tomem decisões tanto a longo prazo, com o uso da análise preditiva, ou até mesmo decisões em tempo real com a análise descritiva.

Objetivos do sistema de aprendizado de máquina

O sistema de aprendizado de máquina geralmente tem um dos seguintes objetivos.

1. Prever uma categoria

O modelo de aprendizado de máquina ajuda a analisar os dados de entrada e, em seguida, prevê uma categoria sob a qual a saída cairá. A previsão nesses casos geralmente é uma resposta binária baseada em "sim" ou "não". Por exemplo, ajuda com respostas como "choverá hoje ou não?" "Isso é uma fruta?" "Este e-mail é spam ou não?" e assim por diante. Isso é obtido referenciando um grupo de dados que indicará se um determinado e-mail se enquadra na categoria de spam ou não, com base em palavras-chave específicas. Esse processo é conhecido como classificação.

2. Prever uma quantidade

Esse sistema geralmente é usado para prever um valor como prever a precipitação de acordo com diferentes atributos do clima, como temperatura, porcentagem de umidade, pressão do ar e assim por diante. Esse tipo de previsão é chamado de regressão. O algoritmo de regressão possui várias subdivisões, como regressão linear, regressão múltipla, etc.

3. Detectar Anomalias

O objetivo de um modelo na detecção de anomalias é detectar quaisquer outliers no conjunto de dados fornecido. Esses aplicativos são usados em sistemas bancários e de comércio eletrônico, nos quais o sistema é construído para sinalizar qualquer transação incomum. Tudo isso ajuda a detectar transações fraudulentas.

4. Clusterizar

Essas formas de sistemas ainda estão nos estágios iniciais, mas seus aplicativos são numerosos e podem mudar drasticamente a maneira como os negócios são conduzidos. Nesse sistema, o usuário é classificado em diferentes grupos de acordo com vários fatores comportamentais, como faixa etária, região em que vive ou até o tipo de programa que gosta de assistir. De acordo com esse cluster, a empresa agora pode sugerir programas diferentes ou mostra que um usuário pode estar interessado em assistir de acordo com o cluster ao qual o usuário pertence durante a classificação.

Capítulo 2. Inteligência Artificial

A Inteligência Artificial é uma das áreas da informática em evidência e nos últimos anos tem mostrado a sua importância nos mais diversos domínios. A inteligência artificial (IA) ou em inglês Artificial Intelligence (AI), é a inteligência similar à humana exibida por mecanismos ou softwares, além de também ser um campo de estudo acadêmico. Os principais pesquisadores e livros didáticos definem o campo como “o estudo e projeto de agentes inteligentes”, onde um agente inteligente é um sistema que percebe seu ambiente e toma atitudes que maximizam suas chances de sucesso.

A inteligência artificial é definida por Andreas Kaplan e Michael Haenlein como uma capacidade do sistema para interpretar corretamente dados externos, aprender a partir desses dados e utilizar essas aprendizagens para atingir objetivos e tarefas específicos através de adaptação flexível. John McCarthy, quem cunhou o termo em 1956, a define como “a ciência e engenharia de produzir máquinas inteligentes”. É uma área de pesquisa da computação dedicada a buscar métodos ou dispositivos computacionais que possuam ou multipliquem a capacidade racional do ser humano de resolver problemas, pensar ou, de forma ampla, ser inteligente. Também pode ser definida como o ramo da ciência da computação que se ocupa do comportamento inteligente, ou ainda do estudo de como fazer os computadores realizarem coisas que, atualmente, os humanos fazem melhor.

O principal objetivo dos sistemas de IA é executar funções que, caso um ser humano fosse executar, seriam consideradas inteligentes. É um conceito amplo, e que recebe tantas definições quanto damos significados diferentes à palavra inteligência. Podemos pensar em algumas características básicas desses sistemas, como a capacidade de raciocínio (aplicar regras lógicas a um conjunto de dados disponíveis para chegar a uma conclusão), aprendizagem (aprender com os erros e acertos de forma que no futuro agirá de maneira mais eficaz), reconhecer padrões (tanto padrões visuais e sensoriais, como também padrões de comportamento) e inferência (capacidade de conseguir aplicar o raciocínio nas situações do nosso cotidiano).

O desenvolvimento da área começou logo após a Segunda Guerra Mundial, com o artigo “Computing Machinery and Intelligence” do matemático inglês Alan Turing, e o próprio nome foi cunhado em 1956. Seus principais idealizadores foram os cientistas Herbert Simon, Allen Newell, John McCarthy, Warren McCulloch, Walter Pitts e Marvin Minsky, entre outros. As construções de máquinas inteligentes interessam à humanidade há muito tempo, havendo na história tanto um registro significante de autômatos mecânicos (reais), quanto de personagens místicos (fictícios) construídos pelo homem com inteligência própria, tais como o Golem e o Frankenstein. Tais relatos, lendas e ficções demonstram expectativas contrastantes do homem, de fascínio e de medo, em relação à inteligência artificial.

Apenas recentemente, com o surgimento do computador moderno, é que a inteligência artificial ganhou meios e massa crítica para se estabelecer como ciência integral, com problemáticas e metodologias próprias. Desde então, seu desenvolvimento tem extrapolado os clássicos programas de xadrez ou de conversão e envolvido áreas como visão computacional, análise e síntese da voz, lógica difusa, redes neurais artificiais e muitas outras. Inicialmente a IA visava reproduzir o pensamento humano. A inteligência artificial abraçou a ideia de reproduzir faculdades humanas como criatividade, autoaperfeiçoamento e uso da linguagem. Porém, o conceito de inteligência artificial é bastante difícil de se definir. Por essa razão, inteligência artificial foi (e continua sendo) uma noção que dispõe de múltiplas interpretações, não raro conflitantes ou circulares.

A inteligência artificial é uma técnica usada para fazer com que as máquinas imitem qualquer comportamento humano. O objetivo é garantir que uma máquina possa imitar com precisão e eficiência qualquer comportamento humano.

O aprendizado de máquina, conforme definido acima, é o uso de modelos estatísticos e matemáticos para ajudar as máquinas a imitar o comportamento humano. Isso é feito usando dados passados.

Já o aprendizado profundo é um subconjunto de aprendizado de máquina e refere-se às funções e algoritmos que um engenheiro usa para ajudar uma máquina a se treinar. A máquina pode aprender a tomar a opção correta para obter uma saída.

As redes neurais e o processamento de linguagem natural fazem parte do ecossistema de aprendizado profundo.

Vantagens e desvantagens da utilização da Inteligência Artificial

Vantagens:

- Redução de erros: uma vez que se trata de máquinas, a inteligência artificial é mais resistente e tem maior capacidade de suportar ambientes hostis, reduzindo as chances de falharem em seus propósitos, tendo a possibilidade de alcançar um maior grau de precisão.
- Exploração: devido à programação dos robôs, eles podem realizar um trabalho mais laborioso e duro com maior responsabilidade. Assim, são capazes de serem utilizadas também em processos de exploração de minérios e de outros combustíveis, por exemplo no fundo do oceano, e, portanto, superar as limitações humanas.
- Aplicações diárias: inteligência artificial é amplamente empregada por instituições financeiras e instituições bancárias para organizar e gerenciar dados. A sua utilização está presente em vários mecanismos do nosso cotidiano, como o GPS (Global Positioning System), a correção nos erros de digitação na ortografia, entre outros.
- Sem pausas: as máquinas, ao contrário dos seres humanos, não precisam de intervalos frequentes. Elas conseguem exercer várias horas de trabalho sem ficarem cansadas, distraídas ou entediadas, apenas pela sua programação.

Desvantagens:

- Alto custo: o custo de produção das máquinas de IA é demasiado, o que se deve a complexidade e dificuldade de manutenção. O processo de recuperação de códigos perdidos, por exemplo, requer muito tempo e recursos.

- Falta de criatividade: a inteligência artificial não é desenvolvida ao ponto de atuar como o cérebro humano, de forma criativa. Além do mais, o cérebro humano ainda não é suficientemente compreendido para que um dia possa ser simulado fielmente em uma forma artificial. Portanto, a ideia de replicar funções do cérebro humano é intangível.

- Causa o desemprego: como são capazes de executar tarefas antes exclusivas aos humanos de maneira mais otimizada e eficiente, os mecanismos de inteligência artificial tendem a substituir a atividade humana em larga escala. O trabalho de uma máquina que possui inteligência artificial é, muitas vezes, mais viável que o trabalho humano, logo, a projeção de um crescimento no desemprego em função disso é coerente.

Principais pesquisadores de IA

Atualmente existem diversos pesquisadores de IA ao redor do mundo em várias instituições e companhias de pesquisa. Entre os muitos que fizeram contribuições significativas estão:

- Alan Turing (1912-1954):

Um dos homens de maior importância não só para seu tempo, como para a atualidade. Com estudos que não só foram base para a existência da inteligência artificial, mas de quase todos os aparelhos eletrônicos já feitos. Criou seu famoso teste, o “Teste de Turing”, usado até hoje para descobrir o nível de inteligência de um programa de inteligência artificial. Esse teste não foi criado para analisar a capacidade de um computador de pensar por si mesmo, já que as máquinas são completamente incapazes disso, mas sim de identificar o quão bem ele pode imitar o cérebro humano.

- John McCarthy (1927-2011):

Matemático, cientista, o criador do termo “Inteligência Artificial” e também o pai da linguagem de programação LISP. McCarthy foi considerado um dos primeiros

homens a trabalhar no desenvolvimento da inteligência artificial e sempre disse que ela deveria interagir com o homem. Nascido na cidade de Boston, ele trabalhou na Universidade de Stanford e no Massachusetts Institute of Technology (MIT), além de ter vencido o prêmio Turing em 1972 e a Medalha Nacional de Ciência em 1991. Já a programação LISP, uma das maiores conquistas de McCarthy, surgiu em 1958 e serviu para facilitar o desenvolvimento da inteligência artificial. A linguagem é das mais antigas ainda em uso e foi usada pela primeira vez ao colocar um computador para jogar xadrez contra um adversário humano.

- Marvin Minsky (1927-2016):

Natural de Nova Iorque, o cientista recebeu diversos prêmios internacionais pelo seu trabalho pioneiro no campo da inteligência artificial, incluindo em 1969, o Prémio Turing, o maior prêmio em ciência informática. O cientista explorou a forma de dotar as máquinas de percepção e inteligência semelhantes à humana, criou mãos robóticas com capacidade para manipular objetos, desenvolveu novos marcos de programação e escreveu sobre assuntos filosóficos relacionados com a inteligência artificial. Minsky estava convencido de que o homem, um dia, desenvolveria máquinas que competiriam com a sua inteligência e via o cérebro como uma máquina cujo funcionamento pode ser estudado e reproduzido num computador, o que poderia ajudar a compreender melhor o cérebro humano e as funções mentais superiores.

- Raj Reddy (1937):

Informático indiano e naturalizado estadunidense, foi o primeiro asiático a vencer o Prêmio Turing. Entre suas contribuições para a IA estão a criação do Instituto de Robótica da CMU e demonstrações de diversos sistemas que usam alguma forma de IA. Entre esses sistemas, estão sistemas de: fala, controlados por voz, reconhecimento de voz, reconhecimento de voz independente do interlocutor, etc. Para Reddy, ao invés de substituir a humanidade, a tecnologia irá criar um novo tipo de humano que irá coexistir com seus antecessores enquanto se aproveita das vantagens de uma nova classe de ferramentas viabilizada pela tecnologia.

- Terry Winograd (1946):

Winograd é um cientista da computação estadunidense, professor da Universidade Stanford, e codiretor do grupo de interação humano-computador de Stanford. É conhecido nas áreas de filosofia da mente e inteligência artificial por seu trabalho sobre língua natural usando o programa SHRDLU. Para Terry, não restam dúvidas de que a tecnologia da informática, mais precisamente a área de inteligência artificial, transformará as sociedades, introduzindo modificações socioeconômicas irreversíveis. Esse especialista procura saber se os seres humanos seriam capazes de construir máquinas que poderiam compreendê-los, resolver seus problemas e dirigir suas vidas, além de buscar respostas sobre o que aconteceria se, algum dia, essas máquinas se tornassem mais inteligentes do que os próprios humanos que as criaram.

- Douglas Lenat (1950):

Nascido na Filadélfia, Pensilvânia, se formou na Universidade da Pensilvânia. Douglas Bruce Lenat é o Diretor Executivo do Cycorp e foi também um pesquisador proeminente em inteligência artificial, recebendo o prêmio bianual IJCAI Computers and Thought em 1976 pela criação do programa de aprendizado de máquinas. Ele também trabalhou em simulações militares e em numerosos projetos para organizações governamentais, militares, científicas e de inteligência dos EUA. A missão de Lenat, no longo ciclo do projeto Cyc iniciado em 1984, era de construir a base de uma inteligência artificial geral ao representar manualmente o conhecimento como axiomas lógicos contextualizados na linguagem formal, com base em extensões ao cálculo de predicados de primeira ordem; em seguida usar esse enorme motor de inferência de ontologia e a base de conhecimento contextualizada como um viés indutivo para automatizar e acelerar cada vez mais a educação contínua do próprio Cyc, via aprendizagem em máquina e compreensão da linguagem natural.

Capítulo 3. Machine Learning

De forma bem simples, Machine Learning é um conjunto de regras e procedimentos que permite que os computadores possam agir e tomar decisões baseados em dados ao invés de ser explicitamente programados para realizar uma determinada tarefa. Programas de Machine Learning também são projetados para aprender e melhorar ao longo do tempo quando expostos a novos dados. Machine Learning tem estado no centro de muitos avanços tecnológicos nos últimos anos, como carros autônomos, visão computacional e sistemas de reconhecimento de voz.

Treinamento Supervisionado

É o termo usado sempre que o programa é “treinado” sobre um conjunto de dados pré-definido. Baseado no treinamento com os dados pré-definidos, o programa pode tomar decisões precisas quando recebe novos dados. Quando tentamos prever uma variável dependente a partir de uma lista de variáveis independentes.

Exemplo: Pode-se usar um conjunto de dados de recursos humanos para treinamento da Aprendizado de Máquina, que tenha *tweets* marcados como positivos, negativos e neutros e assim treinar um classificador de análise de sentimento.

Outros exemplos:

Var. Independentes	Var. Dependentes
Anos de Carreira, Formação, Idade	Salário
Idade do Carro, Idade do Motorista	Risco de Acidente Automotivo
Texto de um livro	Escola Literária
Temperatura	Receita de venda de sorvete
Imagem da Rodovia	Ângulo da direção de um carro autônomo
Histórico escolar	Nota no ENEM

Note que a característica básica de sistemas de aprendizado supervisionado é que os dados que utilizamos para treiná-los contém a resposta desejada, isto é, contém a variável dependente resultante das variáveis independentes observadas.

Nesse caso, dizemos que os dados são anotados com as respostas ou classes a serem previstas.

Dentre as técnicas mais conhecidas para resolver problemas de aprendizado supervisionado estão regressão linear, regressão logística, redes neurais artificiais, Vector Machine - SVM, Decision Tree (árvores de decisão), K-Nearest e Naive-Bayes. Aprendizado de máquina supervisionado é a área que concentra a maioria das aplicações bem-sucedidas e onde a maioria dos problemas já estão bem definidos.

Este não é um passo essencial na maioria dos algoritmos de regressão e classificação. Isso ocorre porque eles podem facilmente ignorar os valores ausentes em um conjunto de dados ou contorná-lo.

Preparação de dados

Antes de usar o conjunto de dados em um modelo, você precisará seguir as etapas fornecidas abaixo.

1. Integração de dados

Na maioria dos cenários, o conjunto de dados é coletado de várias fontes. O processo necessário para integrar todos os dados em um formato consistente é chamado de integração de dados. Considere o seguinte exemplo: você está coletando dados imobiliários de diferentes partes do mundo. As unidades de medida podem ser jardas, metro quadrado e assim por diante. Para você poder usar todas essas medidas em um único sistema, elas precisarão ser alinhadas ou convertidas em um único formato. Uma combinação surge nesses casos é a redundância de dados, que você pode superar usando a análise de correlação. Isso resultará em dados de alta qualidade, o que garante que você possa treinar bem seu modelo.

2. Transformação de Dados

Às vezes, é eficiente usar dados em um formato específico. O processo de conversão dos dados disponíveis inseridos no formato desejável é chamado de

transformação de dados. O objetivo é converter os dados de uma unidade de formato para outra. Por exemplo, pode ser necessário converter o formato de um formato não linear para um formato linear para garantir que a análise dos dados seja mais significativa.

Algumas das técnicas comuns de transformação são:

Agregação: duas ou mais variáveis de um atributo semelhante serão agregadas em um valor. Este valor pode ser um valor resumido. Por exemplo, você pode combinar vários atributos de categorias em uma única variável.

Normalização: geralmente é usado quando você precisa representar um grupo de dados usando intervalos específicos. Por exemplo, você pode normalizar qualquer valor de sinal que caia em um intervalo e escala específicos. Algumas maneiras de fazer isso são normalização min-max, normalização por escala decimal e normalização do escore z. Escolher normalizar ou não depende dos dados com os quais estamos lidando.

Generalização: a agregação é o processo em que um resumo de algumas variáveis é calculado. Generalização é o processo de transformar os valores mais baixos em valores mais altos. Esse método também é chamado de escalada na hierarquia.

Construção de Atributos: é usada quando se deseja realizar qualquer análise simples. Você pode extrair um novo atributo dos atributos disponíveis anteriormente.

3. Redução de dados

Esse conceito está relacionado à preparação de um conjunto de dados para o treinamento de um modelo. O objetivo de qualquer análise é projetar uma saída significativa usando os dados de entrada. Isso significa que você deve poder usar valores numéricos, se necessário. Leva tempo para trabalhar com grandes conjuntos de dados e há momentos em que é inviável trabalhar com grandes conjuntos de dados. Nesses casos, você pode usar a técnica de redução de dados.

Você pode reduzir os dados de entrada usando alguns algoritmos comprovados para gerar uma representação efetiva do conjunto de dados sem afetar a precisão dos dados originais. A ideia por trás da redução de dados é facilitar o modelo de análise dos dados, removendo qualquer ruído, redundância nos dados e dados ausentes. Isso ajudará a melhorar a eficiência de um algoritmo de aprendizado. Esse método vem com seus prós e contras que qualquer desenvolvedor ou programador precisa conhecer.

Prós:

- A complexidade dos dados é reduzida.
- Apenas os atributos mais importantes estão em foco.
- A complexidade computacional dos sistemas é reduzida significativamente.
- Conjunto de dados reduzido torna a saída menos complexa e, portanto, fácil de entender e usar em aplicativos.

Contras:

- Uma das desvantagens da redução de dados é que alguns dos principais dados essenciais podem ser descartados incorretamente durante o processo de redução, impactando negativamente os resultados da saída.

Métodos de redução de dimensão

Os bancos de dados que usamos possuem milhões de milhares de registros e variáveis. É impossível identificar se essas variáveis dependem umas das outras ou não. É mais difícil entender se existe alguma correlação entre as diferentes variáveis no conjunto de dados. É importante que o usuário ou os engenheiros saibam sempre que haverá vários colineares no conjunto de dados. Essa é uma condição em que as variáveis preditoras no conjunto de dados são correlacionadas de alguma maneira.

Existe muita instabilidade nas soluções se houver várias colinearidades entre as variáveis no conjunto de dados. Isso levará a resultados inconsistentes e incoerentes. Por exemplo, se você observar alguns modelos de regressão múltiplos, descobrirá que existem várias correlações entre as diferentes variáveis preditivas. Isso resultará em uma solução tendenciosa na correlação entre as variáveis preditoras. Se isso acontecer, nenhuma dessas variáveis preditivas terá impacto no conjunto de soluções se usada de forma independente.

Se um usuário incluir variáveis com altos níveis de correlação entre eles, isso levará a uma ênfase excessiva em alguns componentes do modelo. Isso ocorre porque um componente específico está sendo contado ou considerado duas vezes.

Se muitas variáveis preditivas forem usadas, surgirão complicações desnecessárias onde você precisará identificar como modelar o relacionamento entre uma variável preditora e uma variável de resposta. Isso também complicará a interpretação e a análise, pois viola o princípio da parcimônia.

Validação cruzada usando a técnica K-Fold

Uma técnica comum de validação é a técnica de validação cruzada K-Fold, que um determinado conjunto de amostras é dividido em dois subconjuntos: conjunto de dados de treinamento e conjunto de dados de teste. O engenheiro pode decidir dividir os dados em partes iguais ou em partes k e k-1. Primeiro, preferiu esse método para validar o modelo, uma vez que é fácil de implementar, fácil de entender e os resultados desse modelo têm um viés baixo e são comparados a outros modelos.

Gerando conjuntos de dados de teste

Quando você trabalha em aplicativos de aprendizado de máquina, os usuários, especialmente os iniciantes, sempre usam os conjuntos de dados disponíveis na Internet, como o conjunto de dados Iris. Eles podem usar esse conjunto de dados para testar os modelos. Os cientistas de dados geralmente trabalham em grandes volumes de dados, têm seus próprios sensores e hardware para coletar dados em tempo real e usá-lo para treinar o modelo. Eles também permitem que o modelo faça inferências com base nesse conjunto de dados.

Um conjunto de dados de teste é um pequeno lote de dados que permitirá testar o modelo. A vantagem de usar esses conjuntos de dados é que eles têm propriedades bem definidas, como linearidade e não linearidade. Isso permitirá ao usuário explorar os diferentes cenários e entender como o algoritmo se comporta em cada cenário.

Técnicas para Aprendizado Supervisionado

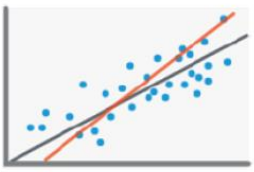
A tabela abaixo resume e fornece uma visão geral das diferenças entre o aprendizado de máquina supervisionado e não supervisionado.

Aprendizado Supervisionado	Aprendizado Não Supervisionado
Trabalha com dados rotulados	Trabalha com dados não rotulados
Também chamado de Modelo Preditivo	Também chamado de Modelo Descritivo
Algoritmos de Regressão e Classificação	Algoritmos de Associação e Clusterização
- Regressão Logística	- K-means
- Regressão Linear	- Regras de Associação
- Regressão Polinomial	- Fuzzy Means
- Gradiente Descendente	- Apriori
- Random Rorest	- Clusterização Hierárquica
- Arvore de Decisão	- PCA - Principal Component Analysis
- K-nearest	
- Naive Bayes	
- SVM - Support Vector Machines	

Regressão (Regression)

Outra subcategoria de aprendizagem supervisionada usada quando o valor que está sendo previsto difere de um “sim ou não” e que siga um espectro contínuo. Sistemas de regressão poderiam ser usados, por exemplo, para responder às perguntas: “Quanto custa?” ou “Quantos existem?”.

Figura 3 – Regressão.



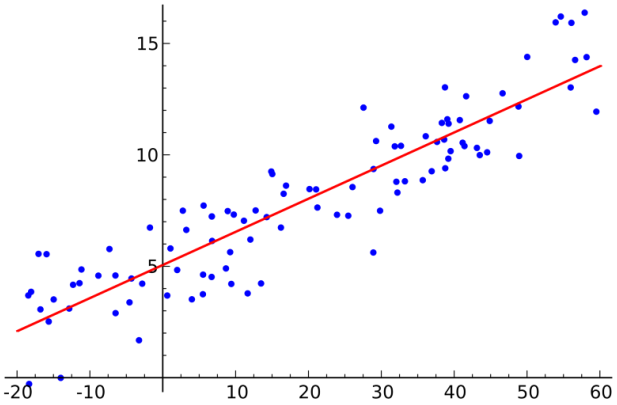
Regressão
(Supervisionado – preditivo)

Regressão Linear

Você pode pensar em regressão linear como a tarefa de encaixar uma linha reta através de um conjunto de pontos. Existem várias estratégias possíveis para isso e a de “mínimos quadrados comuns” é assim: você pode desenhar uma linha e, em seguida, para cada um dos pontos de dados, medir a distância vertical entre o ponto e a linha e somá-los. A linha ajustada seria aquela em que esta soma de distâncias é a menor possível.

Linear refere-se ao tipo de modelo que você está usando para ajustar os dados, enquanto mínimos quadrados refere-se ao tipo de métrica de erro que você está minimizando.

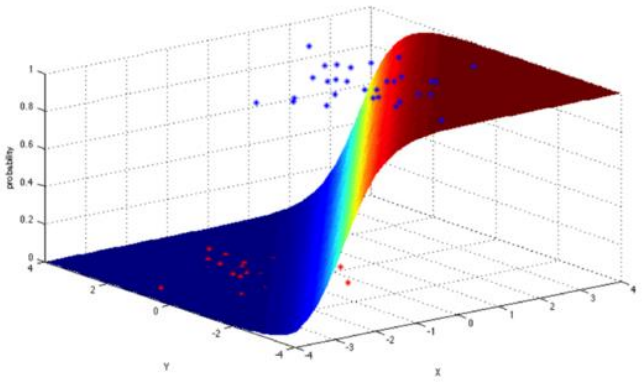
Figura 4 – Regressão Linear.



Regressão logística

A regressão logística é uma poderosa forma estatística de modelar um resultado binomial com uma ou mais variáveis explicativas. Ela mede a relação entre a variável dependente categórica e uma ou mais variáveis independentes, estimando as probabilidades usando uma função logística, que é a distribuição logística cumulativa.

Figura 5 – Regressão Logística.



E geral, as regressões podem ser usadas em aplicações reais, tais como:

– Pontuação de crédito.

– Medir as taxas de sucesso das campanhas de marketing.

– Prever as receitas de um determinado produto.

– Haverá um terremoto em um determinado dia?

Classificação (Classification)

A classificação é uma subcategoria de aprendizagem supervisionada. Classificação é o processo de tomar algum tipo de entrada e atribuir um rótulo a ela. Sistemas de classificação são usados geralmente quando as previsões são de natureza distinta, ou seja, um simples “sim ou não”. Exemplo: Mapeamento de uma imagem de uma pessoa e classificação como masculino ou feminino.

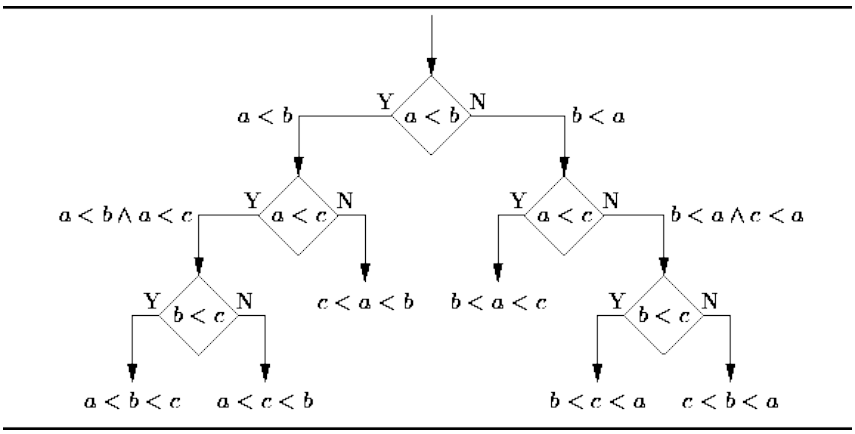
Figura 6 – Classificação.

Classificação
(Supervisionado – preditivo)

Árvores de Decisão (Decision Trees)

Uma árvore de decisão é uma ferramenta de apoio à decisão que usa um gráfico de árvore ou modelo de decisões e suas possíveis consequências, incluindo resultados de eventos fortuitos, custos de recursos e utilidade. Uma árvore de decisão é também uma maneira de representar visualmente um algoritmo. Dê uma olhada na imagem para ter uma ideia de como ela se parece.

Figura 7 – Árvore de Decisão.



Do ponto de vista da decisão de negócios, uma árvore de decisão é o número mínimo de perguntas que devem ser respondidas para avaliar a probabilidade de tomar uma decisão correta, na maioria das vezes. Como um método, permite-lhe abordar o problema de uma forma estruturada e sistemática para chegar a uma conclusão lógica.

Modelo Gerador (Generative Model)

Em probabilidade e estatística, um Modelo Gerador é um modelo usado para gerar valores de dados quando alguns parâmetros são desconhecidos. Modelos geradores são usados em Machine Learning para qualquer modelagem de dados diretamente ou como um passo intermediário para a formação de uma função de densidade de probabilidade condicional. Em outras palavras, podemos modelar P (x, y), a fim de fazer previsões (que podem ser convertidos para p (x | y) aplicando a regra de Bayes), bem como para ser capaz de gerar prováveis pares (x, y), o que é amplamente utilizado na aprendizagem não supervisionada. Exemplos de Modelos Geradores incluem Naive Bayes, Latent Dirichlet Allocation e Gaussian Mixture Model.

Naive Bayes é um classificador probabilístico baseado no Teorema de Bayes, que mostra como determinar a probabilidade de um evento condicional através da probabilidade inversa. Para facilitar a computação, este classificador assume que a presença (ou ausência) de um atributo não tem relação alguma com qualquer outro atributo (por isto o nome Naive). Partindo do teorema de Bayes:

$$P(classe|atributos) = \frac{P(classe) \cdot P(atributos|classe)}{P(atributos)}$$

Como é assumido que os atributos $(a_1, .. a_n)$ são independentes:

$$P(classe|atributos) = \frac{P(classe) \cdot P(a_1|classe) \cdot \ldots \cdot P(a_n|classe)}{P(atributos)}$$

Em vez de calcular P(atributos) explicitamente, o algoritmo calcula apenas o denominador para cada classe e normaliza-os para que a soma seja 1. A este termo damos o nome de evidência.

$$P(classe|atributos) = \frac{P(classe) \cdot P(a_1|classe) \cdot \ldots \cdot P(a_n|classe)}{evidência}$$

Se estivermos interessados apenas em qual classe tem maior probabilidade, a evidência pode ser ignorada, pois é uma constante, determinada durante o treino. Para a classificação binária sabemos que a probabilidade das classes, para os

mesmos atributos, é complementar, assim precisamos apenas ter a probabilidade de o comentário ser rejeitado.

$$P(aprovado|atributos) = 1 - P(rejeitado|atributos)$$

Quando a probabilidade de rejeição for menor que 0.5 o comentário é classificado como aprovado. Quando for maior que 0.5 é classificado como rejeitado. Quando a probabilidade é igual a 0.5, definimos que ele será rejeitado.

Alguns exemplos reais são:

- Para marcar um e-mail como spam ou não spam.
- Classificar um artigo de notícias sobre tecnologia, política ou esportes.
- Verificar um pedaço de texto expressando emoções positivas ou negativas.
- Usado para software de reconhecimento facial.

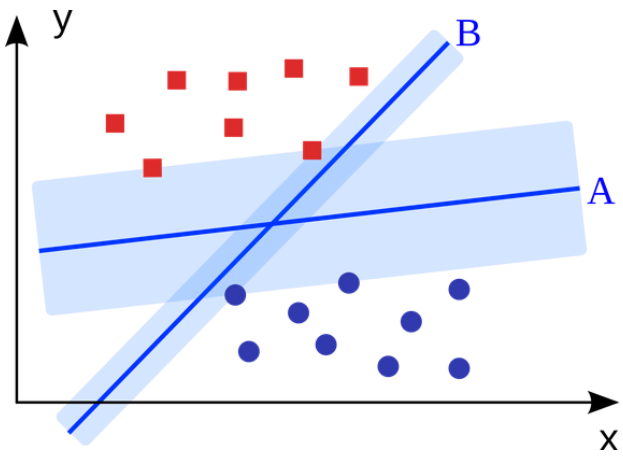
Modelos Condicionais (Discriminative Model)

Discriminative Model ou modelos condicionais, são uma classe de modelos usados em Machine Learning para modelar a dependência de uma variável y em uma variável x. Como esses modelos tentam calcular probabilidades condicionais, isto é, p (y | x) são frequentemente utilizados em aprendizado supervisionado. Exemplos incluem regressão logística, SVMs e Redes Neurais.

Support Vector Machine

SVM é um algoritmo binário da classificação. Dado um conjunto de pontos de 2 tipos em lugar N dimensional, SVM gera um hiperplano (N – 1) dimensional para separar esses pontos em 2 grupos. Digamos que você tem alguns pontos de 2 tipos em um papel que são linearmente separáveis. SVM encontrará uma linha reta que separa esses pontos em 2 tipos e situados o mais longe possível de todos esses pontos.

Figura 8 – Support Vector Machine.

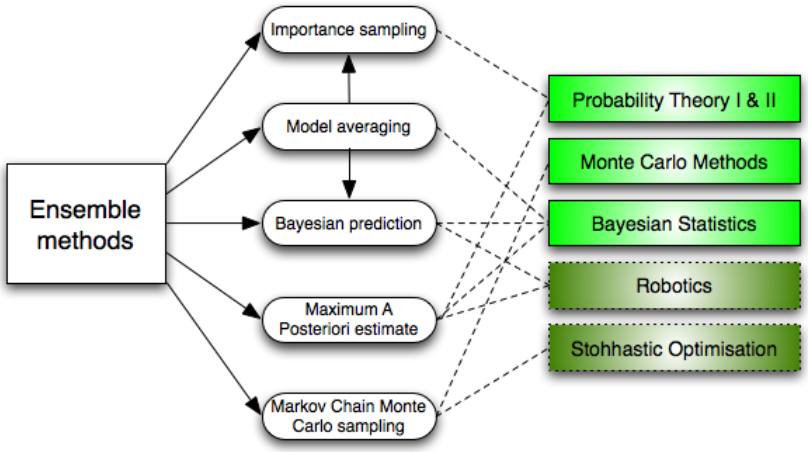


Em termos de escala, alguns dos maiores problemas que foram resolvidos usando SVMs (com implementações adequadamente modificadas) são publicidades em display, reconhecimento de site de splice humano, detecção de gênero baseada em imagem, classificação de imagem em grande escala, etc.

Ensemble Methods

São algoritmos de aprendizagem que constroem um conjunto de classificadores e, em seguida, classificam novos pontos de dados, tendo um ponderado voto de suas previsões. O método de conjunto original é a média bayesiana, mas os algoritmos mais recentes incluem codificação de saída, correção de erros, ***bagging*** e reforço.

Figura 9 – Ensemble Methods.



Então, como funcionam os métodos de conjunto e por que eles são superiores aos modelos individuais?

Eles reduzem a variância: A opinião agregada de um monte de modelos é menos barulhenta do que a opinião única de um dos modelos. Em finanças, isso é chamado diversificação – uma carteira mista de muitas ações será muito menos variável do que apenas um dos estoques sozinhos. É por isso que seus modelos serão melhores com mais pontos de dados do que menos.

É improvável que eles se sobrepõem: se você tem modelos individuais que não se sobrepõem e está combinando as previsões de cada modelo de uma maneira simples (média, média ponderada, regressão logística), então não há espaço para sobrecarga.

Aprendizado Semi-Supervisionado

Uma tarefa de aprendizado semi-supervisionado pode ser uma tarefa de agrupamento semi-supervisionado ou de classificação/regressão semi-supervisionada.

Agrupamento semi-supervisionado é utilizado quando os dados são não rotulados, mas são conhecidas algumas restrições sobre os dados, como que par de exemplos deve estar no mesmo grupo ou em grupos diferentes.

A classificação semi-supervisionada é utilizada em tarefas de classificação em que apenas parte dos exemplos de treinamento possui um rótulo de classe.

Raciocínio semelhante é utilizado para a regressão semi-supervisionada. Isso acontece porque rotular exemplos poder ter um custo elevado, pode ser uma tarefa difícil ou pode necessitar de dispositivos especiais.

O aprendizado semi-supervisionado procura aumentar o número de exemplos rotulados utilizando como conjunto de treinamento os exemplos atualmente rotulados. O modelo induzido com esses dados é utilizado para rotular exemplos não

rotulados. Para isso, o exemplo não rotulado deve ter sua classe predita com uma boa confiança. Esses exemplos são então adicionados ao conjunto de treinamento e o processo se repete até que os exemplos que satisfaçam uma dada condição sejam rotulados. Assim, esse tipo de aprendizado utiliza de forma passiva tanto os dados rotulados quando os não rotulados para induzir um modelo preditivo.

Capítulo 4. Deep Learning

Tecnicamente, o deep learning faz o “treinamento” de um modelo computacional, para que ele possa decifrar a linguagem natural, por exemplo. O modelo relaciona termos e palavras para inferir significado uma vez que é alimentado com grandes quantidades de dados. Em geral, as máquinas já são previamente “ensinadas” a ler os documentos e podem responder a questões colocadas sobre o seu conteúdo, mas as suas bases de conhecimento normalmente estão limitadas pelo tamanho dos documentos. Como a quantidade de algoritmos on-line não para de crescer, a abordagem deep learning vem para fazer com que os sistemas possam fazer uso de um maior número de linguagem natural, concedendo-lhe uma compreensão mais profunda de temas universais.

Tradicionalmente, a qualidade dos algoritmos depende muito da representação dos dados em certas características (as chamadas features). Assim, feature engineering e pré-processamento consomem grande parte dos esforços dos especialistas e são específicos para cada domínio. Na análise de imagens, por exemplo, é comum fazer o pré-processamento com algoritmos de detecção de fronteiras (os chamados edges) para facilitar a identificação de objetos.

Já os algoritmos do tipo deep learning têm uma abordagem inovadora, pois dispensam grande parte deste pré-processamento e geram automaticamente propriedades invariantes em suas camadas hierárquicas de representação. Ao usar métodos baseados em deep learning, cientistas de dados, engenheiros e desenvolvedores de softwares têm conseguido ótimos resultados em diferentes aplicações como em visão computacional, reconhecimento de voz e processamento de linguagem. Usando várias camadas de processamento de dados não lineares, eles conseguem obter uma representação complexa e abstrata dos dados de forma hierárquica.

Dados sensoriais (pixels de imagens, por exemplo) são alimentados a uma primeira camada, sendo que a saída de cada camada se torna a entrada da camada seguinte. Logo, o empilhamento de várias camadas destes “neurônios” é a ideia básica dos algoritmos do tipo deep learning.

Nos últimos anos, o deep learning ajudou a forjar avanços em áreas tão diversas como a percepção do objeto, a tradução automática e o reconhecimento de todos os tópicos de pesquisa de voz que têm sido por muito tempo uma barreira para os pesquisadores de inteligência artificial. A importância do deep learning é tamanha que o Google, por exemplo, disponibilizou um curso gratuito sobre o assunto para qualquer internauta interessado. Por meio da plataforma TensorFlow é possível conhecer o mecanismo e entender na prática como as máquinas podem “aprender e interpretar” mais profundamente.

A importância de entender deep learning se dá pelo intenso crescimento que esta abordagem tem passado nos últimos anos. Em geral, ela já está sendo utilizada dentro do conceito de aprendizado de máquina para as seguintes finalidades:

- Melhorar a experiência dos usuários em resultados de pesquisas on-line.
- Otimizar campanhas de anúncios on-line em tempo real (sites e aplicativos móveis).
- Analisar sentimentos por meio de textos (redes sociais, sobretudo).
- Melhorar as ofertas em e-commerces por meio de análise da navegação do cliente conectado.
- Prever falhas em equipamentos diversos.
- Melhorar a precificação através da análise do comportamento do consumidor nas lojas virtuais.
- Detectar fraudes.
- Detectar invasões de rede (inclusive aquelas orquestradas em massa).
- Reconhecer padrões e imagens.
- Filtrar spams nos e-mails.

Esta abordagem combina avanços no poder da computação com tipos especiais de redes neurais para que as máquinas “aprendam” padrões complicados em quantidades exponenciais de dados.

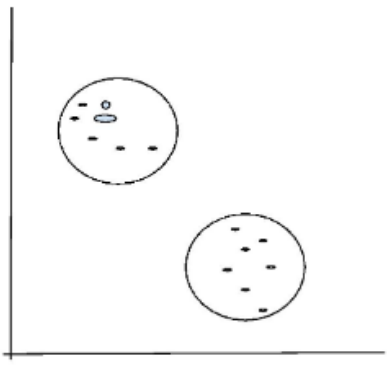
As grandes empresas de software estão investindo fortemente na construção de capacidades de aprendizagem profunda e incorporando-as em muitos dos seus produtos. O Facebook está lançando, de forma gratuita, os desenhos de um novo servidor com a intenção de melhor executar suas aplicações de inteligência artificial; a IBM abriu seu código de aprendizagem de máquina, o SystemML, entre inúmeros outros exemplos.

Aprendizado Não Supervisionado

Nesse tipo de modelo, o sistema é mais sofisticado no sentido de aprender a identificar padrões em dados não rotulados e produzir uma saída. Esse é um tipo de algoritmo usado para extrair qualquer inferência significativa de grandes conjuntos de dados. Esse modelo também é chamado de modelo descritivo, pois utiliza dados e resume esses dados para gerar uma descrição dos conjuntos de dados. Esse modelo é frequentemente utilizado em aplicativos de mineração de dados que envolvem grandes volumes de dados de entrada não estruturados.

Por exemplo, considerando um conjunto de dados de amostra para uma partida de futebol, o modelo de cluster pode agrupar os jogadores em titulares e reservas.

Figura 10 – Como os dados são plotados e agrupados.



Alguns algoritmos comuns que se enquadram no aprendizado de máquina não supervisionado incluem estimativa de densidade, clustering, redução de dados e compactação.

O algoritmo de clustering resume os dados e os apresenta de uma maneira diferente. Essa é uma técnica usada em aplicativos de mineração de dados. A estimativa de densidade é usada quando o objetivo é visualizar qualquer conjunto de dados grande e criar um resumo significativo. Isso nos trará o conceito de redução de dados e dimensionalidade. Esses conceitos explicam que a análise ou saída deve sempre entregar o resumo do conjunto de dados sem a perda de informações valiosas. Em palavras simples, esses conceitos dizem que a complexidade dos dados pode ser reduzida se a saída derivada for útil.

De uma forma geral, com aprendizado não supervisionado queremos achar uma representação mais informativa dos dados que temos. Geralmente, essa representação mais informativa é também mais simples, condensando a informação em pontos mais relevantes.

Portanto, o termo é usado quando um programa pode automaticamente encontrar padrões e relações em um conjunto de dados. Exemplo: Análise de um conjunto de dados de e-mails e agrupamento automático de e-mails relacionados ao tema, sem que o programa possua qualquer conhecimento prévio sobre os dados.

Mais exemplos:

Dados	Forma Representativa
Transações bancárias	Normalidade da transação
Registros de Compras	Associação entre produtos
Dados Multidimensionais	Dados com dimensão reduzida
Registros de Compras	Perfil dos consumidores
Palavras em um texto	Representação matemática das palavras

Outros exemplos de aplicações de aprendizado não supervisionados são sistemas de recomendação de filmes ou músicas, detecção de anomalias e visualização de dados. Dentre as técnicas mais conhecidas para resolver problemas

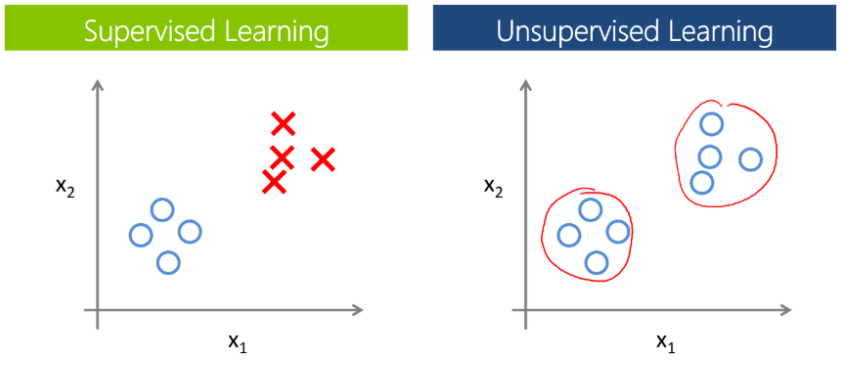
de aprendizado não supervisionado estão redes neurais artificiais, clusterização k-médias, Suport Vector Machine (SVM), Clusterização Hierárquica, PCA, Random Forest, máquinas de Boltzmann. Problemas de aprendizado não supervisionado são consideravelmente mais complicados do que problemas de aprendizado supervisionado, principalmente porque não temos a reposta rotulada nos dados. Como consequência, é extremamente complicado e controverso avaliar um modelo de aprendizado não supervisionado e esse tipo de modelo está na fronteira do conhecimento em aprendizado de máquina.

Técnicas para Aprendizado Não Supervisionado

Em tarefas de descrição, a meta é explorar ou descrever um conjunto de dados. Os algoritmos de Aprendizado de Máquina utilizados nessas tarefas não fazem uso de atributos de saída. Por isso, seguem o paradigma de aprendizado não supervisionado. Uma tarefa descritiva de agrupamento de dados, por exemplo, tem por meta encontrar grupos de objetos semelhantes no conjunto de dados. Outra tarefa descritiva é encontrar regras de associação que relacionem um grupo de atributos a outro grupo de atributos.

O objetivo do um modelo não supervisionado é organizar os dados de alguma forma ou descrever sua estrutura.

Figura 11 – Abordagens de Aprendizado.



Exemplo das diferentes abordagens de aprendizado.

Fonte: https://medium.com/opensanca/aprendizagem-de-maquina-supervisionada-ou-nao-supervisionada-7d01f78cd80a.

A tabela abaixo que resume e fornece uma visão geral das diferenças entre o aprendizado de máquina supervisionado e não supervisionado.

Aprendizado Supervisionado	Aprendizado Não Supervisionado
Trabalha com dados rotulados	Trabalha com dados não rotulados
Também chamado de Modelo Preditivo	Também chamado de Modelo Descritivo
Algoritmos de Regressão e Classificação	Algoritmos de Associação e Clusterização
- Regressão Logística	- K-means
- Regressão Linear	- Regras de Associação
- Regressão Polinomial	- Fuzzy Means
- Gradiente Descendente	- Apriori
- Random Rorest	- Clusterização Hierárquica
- Arvore de Decisão	- PCA - Principal Component Analysis
- K-nearest	
- Naive Bayes	
- SVM - Support Vector Machines	

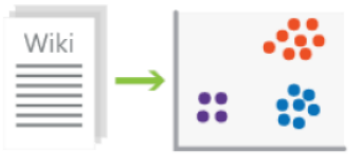
Associação

A tarefa de associação tem o intuito de identificar associações entre registros de dados que, de alguma maneira, estão ou devem estar relacionados. Sua premissa básica é encontrar elementos que implicam na presença de outros em uma mesma transação. Alguns algoritmos que utilizam os conceitos desta tarefa são as regras de associação e os padrões sequenciais.

Agrupamento

Nas seções anteriores, vimos como o algoritmo de aprendizado supervisionado, algoritmo de classificação e regressão, foram usados para desenvolver um modelo de aprendizado de máquina. Nesta seção, falaremos sobre o algoritmo de aprendizado de máquina não supervisionado em cluster.

Figura 12 – Agrupamento.



Agrupamento
(Não supervisionado – descritivo)

No aprendizado de máquina supervisionado, você está procurando a resposta com base nos dados de entrada x. No aprendizado não supervisionado, no entanto, procuramos descobrir o que um modelo preverá se x for fornecido.

No domínio da ciência de dados, você costuma pensar em como os dados disponíveis podem ser usados para fazer previsões de pontos de vista de dados. Há momentos em que você deseja categorizar as informações disponíveis em categorias de clusters e não apenas fazer previsões. O primeiro é um exemplo de aprendizado de máquina supervisionado e o último é um exemplo de aprendizado de máquina não supervisionado.

Vamos dar uma olhada no exemplo a seguir para entender isso. Vamos supor que você esteja trabalhando em uma loja de pizzas e tenha a tarefa de criar um recurso para gerenciar o pedido. Esse recurso deve prever o tempo de entrega do cliente. Você tem informações históricas das entregas anteriores, onde obteve informações sobre a distância percorrida para entregar as pizzas e outros parâmetros, como a hora do dia ou da semana. Usando essas informações, você pode prever o tempo de entrega futuro. Este é um exemplo de aprendizado de máquina supervisionado.

Vamos agora olhar para um requisito. Você trabalha na mesma lanchonete de pizza, mas tem a tarefa de identificar o segmento de seus clientes para executar uma campanha de cupom. Você tem acesso a alguns dados históricos, como o nome de seus clientes, a idade, a área e outras informações. Agora você pode classificar esses clientes em diferentes clusters com base em vários fatores. Este é um exemplo

de aprendizado de máquina não supervisionado, porque você não está fazendo uma previsão, quase categorizando seus clientes em vários grupos.

Algumas aplicações comuns do aprendizado de máquina não supervisionado incluem domínios com conjuntos de dados complexos e enormes, como a genômica.

Genômica é uma área em que o modelo forma um agrupamento de genes que possuem propriedades semelhantes. Para entender a importância do assunto, a Genômica é um ramo da genética que estuda o genoma completo de um organismo. Essa ciência pode se dedicar a determinar a sequência completa do DNA de organismos ou apenas o mapeamento de uma escala genética menor.

O agrupamento pode ser utilizado em astronomia para classificar diferentes estrelas e objetos celestes usando fatores diferentes, como tamanho, cor, distância, materiais, previsão de zonas de terremotos com base nos agrupamentos de epicentros. Também é usado pelas empresas para os seguintes fins:

- Ao direcionar anúncios segmentados.
- Nas recomendações da Netflix.
- Segmentação de mercado.
- Segmentação de imagem.
- Análise de mídia social.
- Segmentação de imagem.
- Detecção de imagens médicas.
- Detecção de anomalia.
- Motores de recomendação.

Algoritmos de Associação

A tarefa de associação tem o intuito de identificar associações entre registros de dados que, de alguma maneira, estão ou devem estar relacionados.

Algoritmo Apriori

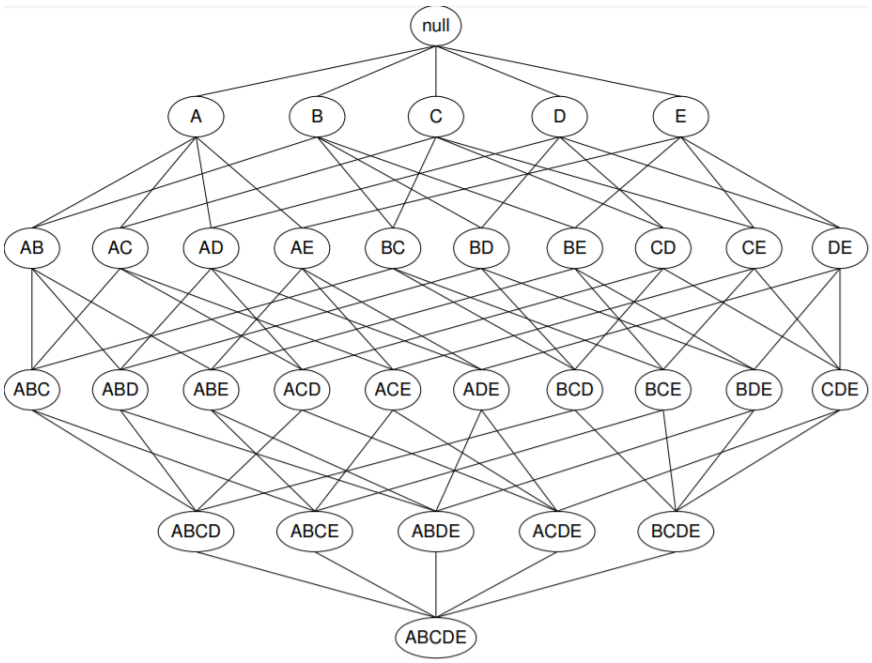
O algoritmo Apriori é um grande aliado no processo de datamining por regras de associação, sendo também o mais conhecido. Sua técnica se baseia no uso de vários subconjuntos oriundos de um grande conjunto de itens.

Primeira fase: consiste em descobrir um grande conjunto de itens que obedeçam a um suporte mínimo determinado em s.

Segunda fase: consiste em utilizar o conjunto com suporte mínimo para gerar as regras de associação para a base de dados.

Confiança = número de registros da tabela que contém todos os itens da regra / no de registros da tabela que contém o antecedente da regra.

Figura 13 – Algoritmo Apriori.



Algoritmos de Agrupamento (Clustering)

É a tarefa de agrupar um conjunto de objetos de tal forma que os do mesmo grupo (cluster) são mais semelhantes uns aos outros do que aqueles em outros grupos.

Cada algoritmo de agrupamento é diferente, e aqui estão alguns deles:

- Algoritmos baseados em Centroid.
- Algoritmos baseados em conectividade.
- Algoritmos baseados em densidade.
- Probabilístico.
- Redução da Dimensionalidade.

Redes Neurais/Aprendizagem Profunda

O clustering é uma das formas mais usadas de algoritmos de aprendizado de máquina não supervisionados. As duas maneiras pelas quais os algoritmos de cluster são classificados se baseiam em como os dados de entrada são classificados.

- Hard Clustering: são aqueles em que a chave mestra de entrada pertence a um dos clusters, no caso de classificação binária.
- Soft Clustering: esse tipo de algoritmo de clustering é a abordagem mais pragmática e pode ser usada em aplicativos de clustering em tempo real. Na maioria dos casos, os dados estão na área cinza e não precisam necessariamente pertencer a uma ou mais classes. Portanto, diferentemente do hard clustering, o ponto de dados não precisa pertencer a um cluster específico. Em vez disso, uma probabilidade é anexada à probabilidade como os dados de entrada pertencentes a um dos clusters existentes.

As duas classificações dos algoritmos de clustering incluem:

- Flat Clustering: onde o cientista de dados informará à máquina as “k” categorias de dados da classe os dados que devem ser divididos.

- Hierarchical Clustering: o cluster hierárquico é outro tipo de algoritmo de armazenamento em cluster em que o usuário não define o valor de k. A máquina decidirá quantos clusters deve criar com base em seus próprios algoritmos.

Outro tipo de classificação de algoritmos de clustering é baseado no modelo usado pelo aprendizado de máquina. Isso porque uma tarefa de agrupamento é subjetiva.

- Connectivity models: como o nome sugere, esses modelos são baseados no pressuposto de que os pontos de dados no espaço de dados estão mais próximos um do outro, exibindo propriedades semelhantes quando comparados aos pontos de dados mais distantes. Esses algoritmos seguirão as seguintes abordagens ao tentar agrupar os dados de entrada: primeiro, eles classificam os pontos de dados e depois agregam esses pontos à medida que a distância entre os pontos aumenta. Nesta etapa, os dados da partição são classificados conforme a distância entre eles aumentará. A função de distância é subjetiva e é por esse motivo que essa classe de algoritmos não é adequada para nenhuma função escalável. Ele não pode lidar com grandes conjuntos de dados. O cluster hierárquico e outras variantes se enquadram nessa categoria.

- Centroid models: todo algoritmo de cluster é baseado na identificação da distância entre um ponto de dados específico e o centroide do cluster no qual o modelo coloca o ponto de dados. O algoritmo k-means é um exemplo clássico desse tipo de algoritmo, pois é fortemente dependente do centroide. O número de clusters é definido pelo usuário, o que torna importante o conhecimento dos dados para que o modelo esteja usando. Pode ser uma limitação para alguns aplicativos que precisam responder a novos pontos de dados dinamicamente. Esses modelos serão executados iterativamente para encontrar o centroide ou o ponto central ideal.

- Disribuition models: essa classificação é baseada na probabilidade de dados específicos. Caindo dentro de um cluster. Também é baseado na probabilidade de que todos os pontos de dados no cluster sigam a distribuição gaussiana normal. Esses modelos sofrem com a questão do ponto de sobreajuste. Um algoritmo popular que se enquadra nessa categoria é o algoritmo de maximização de expectativa que usa a distribuição normal multivariada.

- Density models: os pontos de dados no conjunto de dados de entrada serão plotados no gráfico e variarão a intensidade no espaço do gráfico. Essa variação na densidade é a base dos principais algoritmos de cluster. Esses algoritmos funcionam isoladamente, identificando pontos de dados de amostra de sua região de densidade. Alguns modelos desta categoria são DBSCAN e OPTICS.

Algoritmo de agrupamento K-Means

Esse é um dos algoritmos de cluster mais populares, pois a chave no nome refere-se ao número de classes exclusivas que você deseja gerar a partir do conjunto de dados fornecido. O agrupamento também é chamado de classificação não supervisionada.

Como usuário, você pode definir o valor de k, e o modelo identificará essencial para cada classe para que seja formada. O centroide de cada cluster é o centro de todos os pontos que estão no cluster, e o número de centroides é igual ao número de clusters. Esse algoritmo funciona apenas com valores numéricos e ignora todos os outros valores simbólicos.

A principal vantagem e desvantagem desse algoritmo são as seguintes. A vantagem é que o algoritmo é muito popular, simples e fácil de entender. Esse algoritmo também é fácil de implementar. Isso ocorre porque os dados de entrada são atribuídos a um cluster imediatamente, o que significa que esse algoritmo é amigável para iniciantes, pois você não precisa se preocupar em identificar clusters.

Uma das desvantagens é que você precisará definir o número de clusters que o algoritmo pode criar. Isso significa que você não pode fazer ajustes dinâmicos, se necessário.

A produção também é significativamente influenciada pelos dados do mar que são alimentados no modelo. O algoritmo tenderá a se converter em mínimos locais e, portanto, é recomendável redefinir e executar novamente o algoritmo com sementes diferentes para garantir um erro mínimo ou inexistente.

Esse algoritmo não é eficiente se você estiver trabalhando com um grande conjunto de dados e não é escalável para aplicativos em tempo real. Pode ser mais fácil usar esse algoritmo em grandes conjuntos de dados, se você puder obter amostras do conjunto de dados. Esse algoritmo também é sensível a discrepantes no sentido de que um único discrepante pode afetar significativamente a precisão do cluster. Isso significa que o valor médio será inclinado se o conjunto de dados de entrada tiver algum valor externo. Uma solução para isso é usar o valor mediano dos clusters em vez da média.

K-Means Etapas do algoritmo

Escolha aleatoriamente e escolha o valor de k, que é o número de centros de clusters.

A próxima etapa é desenvolver um método para atribuir todos os pontos de dados a um cluster. Isso só pode ser feito usando técnicas como o cálculo da distância euclidiana para calcular o centro ou centroide mais próximo do cluster.

A posição do centroide do cluster só será atualizada quando todos os pontos tiverem sido atribuídos. O centroide é a média de todos os pontos no cluster.

Agora você deve repetir os segundo e terceiro passos até chegar a um ponto de convergência. Esse é o ponto que define que todos os clusters estão agora em um limite ideal. Quando esse ponto é alcançado, as próximas iterações não afetam as posições do cluster.

Os algoritmos de cluster são um exemplo de algoritmo de aprendizado de máquina não supervisionado; no entanto, eles podem ser usados como uma combinação com alguns algoritmos de aprendizado de máquina supervisionados para melhorar a precisão da saída gerada pelo modelo.

Modelo Gerador (Generative Model)

Em probabilidade e estatística, um Modelo Gerador é um modelo usado para gerar valores de dados quando alguns parâmetros são desconhecidos. Modelos geradores são usados em Aprendizado de Máquina para qualquer modelagem de dados diretamente ou como um passo intermediário para a formação de uma função de densidade de probabilidade condicional. Em outras palavras, podemos modelar P (x, y), a fim de fazer previsões (que podem ser convertidos para p (x | y) aplicando a regra de Bayes), bem como para ser capaz de gerar prováveis pares (x, y), o que é amplamente utilizado na aprendizagem não supervisionada. Exemplos de Modelos Geradores incluem Naive Bayes, Latent Dirichlet Allocation e Gaussian Mixture Model.

Modelos Condicionais (Discriminative Model)

Discriminative Model ou modelos condicionais são uma classe de modelos usados em Aprendizado de Máquina para modelar a dependência de uma variável y em uma variável x. Como esses modelos tentam calcular probabilidades condicionais, isto é, p (y | x) são frequentemente utilizados em aprendizado supervisionado. Exemplos incluem regressão logística, SVMs e Redes Neurais.

Aprendizagem Profunda (Deep Learning)

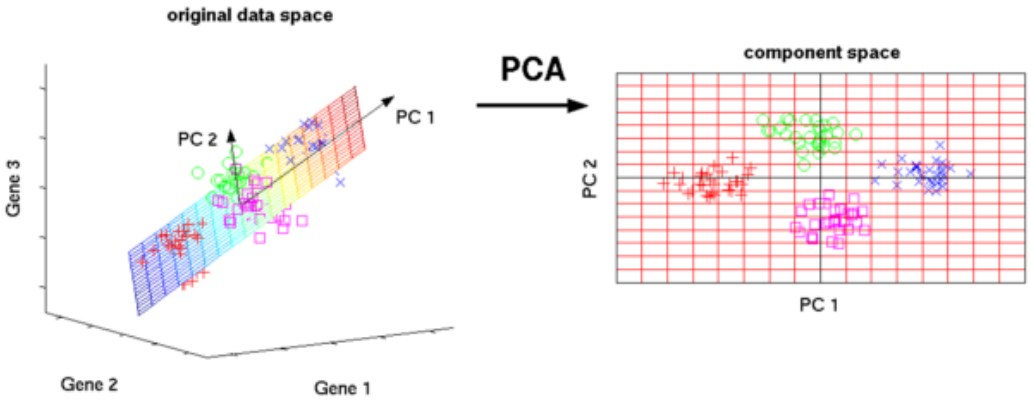
Este tem sido um tema muito discutido recentemente. Basicamente, a aprendizagem profunda refere-se a uma categoria de algoritmos de Aprendizado de Máquina, que muitas vezes usam redes neurais artificiais para gerar modelos. Técnicas de deep learning, por exemplo, foram muito bem-sucedidas na resolução de problemas de reconhecimento de imagem devido à sua capacidade de escolher as melhores características, bem como para expressar camadas de representação. Inspirado por redes neurais biológicas, redes neurais artificiais são uma rede de nós

interconectados que compõem um modelo. Eles podem ser definidos como modelos de aprendizagem estatística que são usados para calcular ou aproximar funções que dependem de um grande número de entradas. As redes neurais são normalmente utilizadas quando o volume de entrada é demasiado grande para as abordagens convencionais de aprendizagem automáticas previamente discutidas.

Análise de Componentes Principais - PCA

PCA é um procedimento estatístico que usa uma transformação ortogonal para converter um conjunto de observações de variáveis, possivelmente correlacionadas em um conjunto de valores de variáveis linearmente não correlacionadas, chamadas componentes principais.

Figura 14 – PCA.



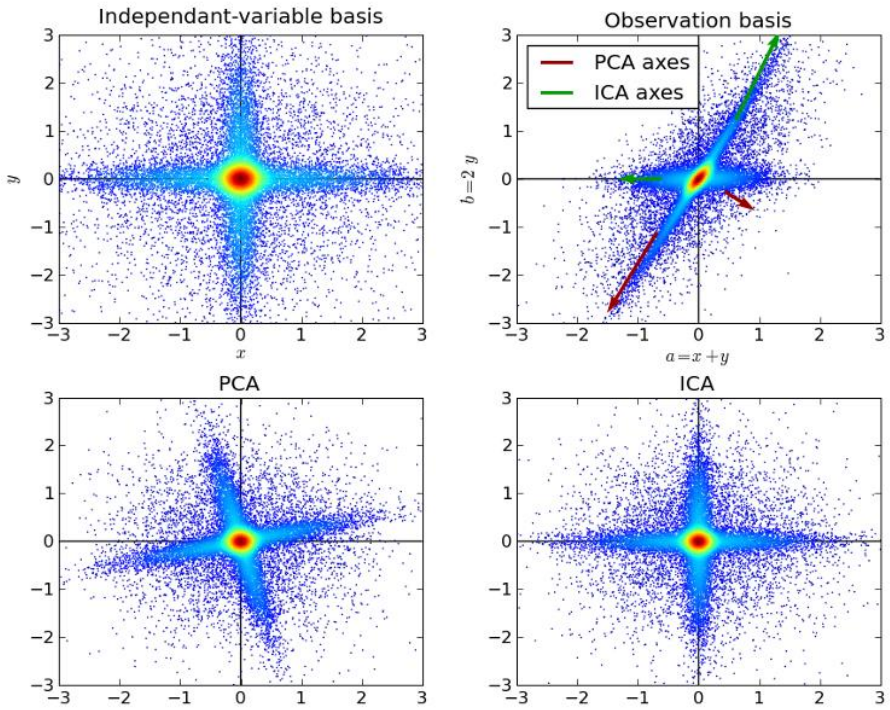
Algumas das aplicações do PCA incluem compressão, simplificação de dados para facilitar o aprendizado e visualização. Observe que o conhecimento do domínio é muito importante ao escolher se deseja avançar com PCA ou não. Não é adequado nos casos em que os dados são ruidosos (todos os componentes da PCA têm uma variação bastante alta).

Análise de componentes independentes - ICA

ICA é uma técnica estatística para revelar fatores ocultos que estão subjacentes a conjuntos de variáveis aleatórias, medições ou sinais. A ICA define um modelo generativo para os dados multivariados observados, que normalmente é dado

como um grande banco de dados de amostras. No modelo, as variáveis de dados são assumidas como misturas lineares de algumas variáveis latentes desconhecidas, e o sistema de mistura também é desconhecido. As variáveis latentes são consideradas não-gaussianas e mutuamente independentes, e são chamadas de componentes independentes dos dados observados.

Figura 15 – ICA.



A ICA está relacionada ao PCA, mas é uma técnica muito mais poderosa, capaz de encontrar os fatores subjacentes das fontes quando esses métodos clássicos falham completamente. Suas aplicações incluem imagens digitais, bancos de dados de documentos, indicadores econômicos e medições psicométricas. Agora siga em frente e use sua compreensão de algoritmos para criar aplicações de aprendizagem de máquina que possibilitam melhores experiências para as pessoas em todos os lugares.

Capítulo 5. Aprendizagem por reforço

A terceira abordagem de aprendizagem de máquinas é a chamada “aprendizagem por reforço”, em que a máquina tenta aprender qual é a melhor ação a ser tomada, dependendo das circunstâncias na qual essa ação será executada.

O futuro é uma variável aleatória: como não se sabe a priori o que irá acontecer, é desejável uma abordagem que leve em consideração essa incerteza, e consiga incorporar as eventuais mudanças no ambiente do processo de tomada da melhor decisão. Essa ideia de fato deriva do conceito de “aprendizagem por reforço” da psicologia, no qual uma recompensa ou punição é dada a um agente, dependendo da decisão tomada; com o tempo e a repetição dos experimentos, espera-se que o agente consiga associar as ações que geram maior recompensa para cada situação que o ambiente apresenta, e passe a evitar as ações que geram punição ou recompensa menor. Na psicologia, essa abordagem é chama de behaviorismo e tem B. F. Skinner como um dos principais expoentes, um famoso psicólogo que, dentre outros experimentos, usou a ideia de recompensas e punições para treinar pombos para conduzir mísseis na Segunda Guerra Mundial.

Essa mesma ideia é vista no aprendizado de máquinas: a máquina observa um “estado da natureza” dentre o conjunto de cenários futuros possíveis e, com base nisso, escolhe uma ação a se tomar e recebe a recompensa associada a essa ação específica e nesse estado específico obtendo, assim, a informação dessa combinação. O processo se repete até que a máquina seja capaz de escolher a melhor ação a se tomar para cada um dos cenários possíveis a serem observados no futuro.

Ilustrando com um exemplo bem simples: suponha que você queira adestrar seu cão a se sentar ao seu comando por essa abordagem. De primeira, dificilmente o animal executará o comando requerido, e você responde a isso dando um “reforço negativo” (punição), repreendendo-o verbalmente, com suas expressões faciais ou mesmo com uma pancada de jornal (ou algo mais hostil, a depender do seu temperamento). Quando o cão se aproxima do que deveria fazer, você pode dar “reforços positivos” como sinais de aprovação ou incentivo. Se o cão de fato se sentar

após o comando, você lhe dá uma recompensa - um biscoitinho, por exemplo. Com várias repetições desse mesmo experimento, espera-se que, com o tempo, o cão passe a associar a relação de “causa-efeito” entre o comando e a recompensa a ser recebida, e com isso “aprenda” a obedecer a esse comando. O famoso experimento do “cão de Pavlov” ilustra bem esse paradigma de aprendizagem. Ivan Pavlov foi um cientista russo notório por apresentar a ideia do “reflexo condicionado”, baseado no seguinte experimento: apresentando um pedaço de carne a um cão, o animal passa a salivar, desejando o alimento. Em vez de apresentar apenas a carne, Pavlov soava uma campainha sempre que isso acontecia; com a repetição, o cão passava a associar os dois “estímulos” (carne e campainha) e salivar assim que ouve a campainha.

Essa ideia é bastante versátil quando a transportamos para o âmbito da ciência de dados: em vez de adestrar cachorros, podemos por exemplo construir uma máquina que monta portfolios no mercado financeiro e que ajusta a combinação de ativos comprados/vendidos a depender da “recompensa” (retorno financeiro) da carteira anterior e da evolução (“estados”) do mercado. Ou ainda um automóvel que dirige “sozinho”, que toma decisões dependendo do cenário que observa ao redor, recebendo recompensas negativas quando colide com o ambiente ou com outros veículos, e com repetidas etapas, aos poucos “aprenda” a contornar os obstáculos.

Vamos mais além: voltando ao exemplo do adestramento de cães, suponha que, depois de tê-lo ensinado a sentar com sucesso, você queira fazer um teste de obediência - fazer o cão ficar sentado enquanto você anda para trás, até chegar a uma distância de cinco metros dele. Caso o cão se sente, você dá a ele o biscoitinho (assim como antes); mas caso ele não se levante até você ficar a cinco metros dele, você dá a ele uma recompensa maior (um pedaço de bife, por exemplo). Como o cão poderia ganhar a recompensa maior? Note que agora o cão tem uma “escolha difícil”: ele pode simplesmente sentar (já que ele já aprendeu essa tarefa) e ganhar sempre o biscoito, assim como pode “explorar” novas possibilidades - no caso, ficar parado mesmo com o dono longe - para ver se eventualmente não há uma recompensa maior ainda. O dono continuará fornecendo reforços positivos e negativos enquanto anda

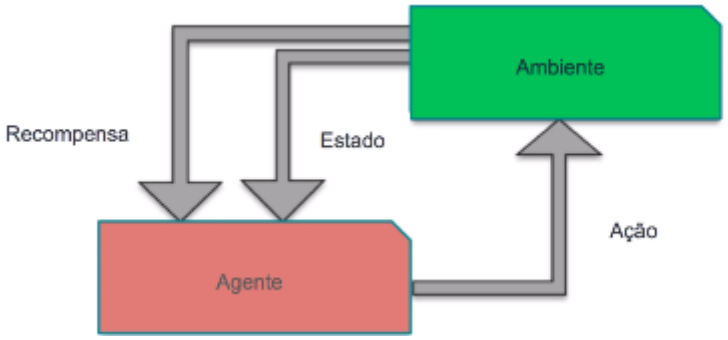
para trás, elogiando-o se ficar parado e repreendendo-o caso saia da posição; mas será que o animal está disposto a renunciar à recompensa que ele já tem “garantida”?

Essa situação ilustra um trade-off: o animal pode escolher por “testar” novas combinações de ações ótimas em uma sequência de “estados” não realizadas anteriormente em busca de uma recompensa maior, mas não imediata (o chamado “exploration”); ou simplesmente se ater à recompensa que se obtém através de uma ação já conhecida (o chamado “exploitation”). Veja como essa ideia pode se estender facilmente para outros contextos; em finanças, por exemplo, reflete bem o grau de aversão ao risco de um agente econômico - o quanto ele está disposto a correr mais riscos em busca de retornos maiores, ou está disposto a correr menos risco e obter um retorno menor, porém mais seguro.

Fazer a máquina ser capaz de encontrar o meio termo ótimo entre “exploration” e “exploitation” é um dos principais desafios da aprendizagem por reforço, e é bastante pertinente para aplicações mais complicadas, como ensinar uma máquina a jogar xadrez, por exemplo: como é bem sabido, uma estratégia vencedora frequentemente envolve abrir mão de vantagem imediata, ou até mesmo sacrifício de peças, visando ao sucesso a longo prazo - um bom jogador deve ser capaz de levar em consideração as consequências de sua jogada várias rodadas adiante, e sabendo que a resposta do oponente também estará visando a um benefício futuro, e assim por diante.

O aprendizado por reforço tem como meta reforçar ou recompensar uma ação considerada positiva e punir uma ação considerada negativa. Um exemplo de tarefa de reforço é a de ensinar um robô a encontrar a melhor trajetória entre dois pontos. Algoritmos de aprendizado utilizados nessa tarefa, em geral, punem a passagem por trechos pouco promissores e recompensam a passagem por trechos promissores.

Figura 16 – Aprendizado por Esforço.



Fonte: https://www.datascienceacademy.com.br.

No aprendizado por reforço nós temos:

– Agente.

– Ambiente.

O Agente faz uma ação no Ambiente.

A mudança de estado gera uma recompensa (positiva, negativa).

Um agente com aprendizado por reforço é sempre dirigido por objetivos.

Ao longo de seu processo de tomadas de decisão, em relação à política a ser adotada, seu objetivo é maximizar seu valor de recompensa. O valor de recompensa é o fator principal que orienta o agente em seu aprendizado. Por vezes, uma ação executada pelo agente pode afetar não apenas a recompensa imediata, mas todas as recompensas subsequentes, uma vez que sua ação altera o estado do ambiente.

A interação com ambientes de estados dinâmicos é um dos aspectos essenciais do aprendizado por reforço.

Dessa forma, o equilíbrio entre exploration e exploitation consiste em um dos maiores desafios para modelagem de um algoritmo de aprendizado por reforço.

Cada passo da interação de um agente com o ambiente pode ser descrito, de forma sucinta, como uma transição na forma de uma transição t, na forma:

t(s, a, r, s')

s → Estado atual do ambiente.

a → Ação tomada pelo agente em s.

r → Valor de recompensa.

s'→ Novo estado resultante de a em s.

Uma transição t(s, a, r, s') consiste no fator essencial da interação do agente com o ambiente, mas outros fatores principais precisam estar presentes neste processo:

- Uma política.
- Um valor de recompensa.
- Uma função de valor.
- Um modelo do ambiente (que pode ser tabular e exato ou aproximado).

Política

A política define a estratégia de seleção de ações do agente a cada momento, ou iteração ao longo do tempo, e em cada novo estado do ambiente.

Abaixo temos a política π definida para um estado s e uma ação a como uma distribuição de probabilidade:

$$\pi(s,a) = P(a_t = a | s_t = s)$$

$$\pi: S \rightarrow A$$

O que se deseja fazer é encontrar uma política que maximize a recompensa total esperada, todavia há muitas políticas possíveis.

A equação abaixo escreve a política como o argumento que maximiza a esperança da recompensa total, dado um estado inicial s0.

$$argmax_{\pi}E[r_{0+}r_{1+} \ldots + r_{T}|s_{0}]$$

De uma forma simplificada, podemos afirmar que uma política é um mapeamento dos estados do ambiente percebidos pelo agente para ações a serem possivelmente tomadas naquele estado.

A política pode ser desde uma função muito simples sobre uma tabela a um processo complexo e computacionalmente custoso.

Recompensa

A recompensa consiste em um valor de retorno enviado pelo ambiente ao agente a cada intervalo de tempo ou interação. O agente tem como objetivo maximizar a recompensa total que receberá em longo prazo.

Como o valor de recompensa define quão boa ou ruim foi uma dada ação de um agente em um dado estado, ao maximizar o valor total de recompensa, o agente está procurando selecionar as melhores ações para cada estado de modo a receber o maior valor de recompensa.

Em cenários mais simples, com um horizonte curto e finito, onde se tem tarefas episódicas, a recompensa total de um episódio de interação do agente pode ser dada simplesmente pela soma das recompensas recebidas a cada interação com o ambiente.

$$R_t = r_t + r_t + r_t + \cdots + r_T$$

Em cenários complexos, com um horizonte infinito e em tarefas contínuas, as recompensas a cada iteração sofrem um valor de desconto, comumente denotado por Υ.

O intuito é equilibrar a distribuição sobre o horizonte de ações, fazendo com que receber um dado valor de recompensa na iteração atual seja equivalente a receber um valor de recompensa maior em uma iteração futura.

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

O fator de desconto também garante que o algoritmo irá convergir em algum momento.

A recompensa é o principal elemento para que a política seja alterada, pois se uma ação selecionada pela política é seguida de uma recompensa baixa, o agente deve alterar a política para que seja escolhida outra ação, naquele estado, em um passo futuro.

Enquanto a recompensa indica no instante atual se a ação tomada foi boa ou ruim em um dado estado, a função de valor indica o quão bom será um conjunto de ações determinadas por uma política em longo prazo, com base na recompensa esperada para a política para cada estado.

$$V^{\pi}(S) = E_{\pi}[r_t + r_{t+1} + \cdots + r_T | s_t = s]$$

Equação de Bellman

Os conceitos da equação são:

- s – Estado (que o agente está).
- a – Ação (ir para cima, para baixo, etc.).
- R – Recompensa (+1 ou -1).
- Y - (gama) é o fator de desconto.

Richard Bellman criou esses conceitos de programação dinâmica em 1953.

Programação Dinâmica

A programação dinâmica é uma ferramenta bastante usada em problemas de otimização dinâmica e é bastante útil para o tratamento de incertezas.

Basicamente, a programação dinâmica divide a sequência de decisões em duas partes: uma decisão imediata e uma função de avaliação que engloba os resultados das decisões subsequentes. Se o horizonte de planejamento é finito, a última tomada de decisão pode ser realizada com base em algum método padrão de otimização estática. Essa solução permite obter uma função de avaliação para a penúltima decisão. Esse processo é repetido sucessivamente, até se chegar à decisão inicial (processo backward). Se o horizonte de planejamento é infinito, o que parece ser uma grande dificuldade para os cálculos, simplifica-se pelo fato de que cada decisão tomada leva a outro problema semelhante ao original. Essa característica do problema não só facilita a solução numérica, mas também permite dar uma conotação teórica à solução, levando algumas vezes a uma solução analítica do problema.

A essência da programação dinâmica pode ser representada pela seguinte equação:

$$F_t(x_t) = \max_{u_t} \left\{ \pi_t(x_t, u_t) + \frac{1}{1+\rho} E_t[F_{t+1}(x_{t+1})] \right\}, \text{ onde:}$$

- x_t = variável de estado no instante t (preço do petróleo, por exemplo);
- u_t = variável de decisão no instante t (investir ou esperar, por exemplo);
- ρ = taxa de desconto (exógena ao projeto);
- $F_t(x_t)$ = valor da oportunidade de investimento no instante t;
- $\Pi_t(x_t, u_t)$ = lucro imediato no instante t;
- $E_t[F_{t+1}(x_{t+1})]$ = valor esperado, na data t, dos fluxos de caixa futuros a partir do instante t + 1. Esta parcela é chamada de valor de continuação.

Processos Estocásticos

Um dos aspectos mais importantes na avaliação de opções reais é determinar de que forma serão tratadas as incertezas do projeto. Na maioria dos estudos desenvolvidos, assume-se que estes fatores de incerteza seguem um processo estocástico, ou seja, sua evolução no tempo tem uma parcela de aleatoriedade.

Quanto às suas propriedades estatísticas (média e variância, principalmente), um processo estocástico pode ser classificado como estacionário, quando mantém as propriedades ao longo do tempo, ou não-estacionário, quando as propriedades mudam ao longo do tempo. Com relação a variável tempo, podemos classificar um processo estocástico como contínuo ou discreto.

Processos de Decisão de Markov (MDP)

A teoria de aprendizagem por reforço é teoricamente restrita a processos de decisão Markovianos, apesar de as ideias e métodos poderem ser aplicados de forma mais genérica.

Um ambiente satisfaz a propriedade de Markov se o seu estado resume o passado de forma compacta, sem perder a habilidade de prever o futuro. Isto é, pode-se predizer qual será o próximo estado e próxima recompensa esperada dado o estado e ação atuais. Um processo de Markov é uma sequência de estados, com a propriedade de que qualquer predição de valor de estado futuro dependerá apenas do estado e ação atuais, e não da sequência de estados passados.

Um processo de aprendizagem por reforço que satisfaz a propriedade de Markov é chamado de processo de decisão Markoviano (MDP - Markov Decision Process). Se o espaço de estados e ações for finito, então ele é chamado de processo de decisão Markoviano finito, base para a teoria de aprendizagem por reforço, que assume o ambiente como sendo deste tipo.

Formalmente, um processo de decisão Markoviano é definido por um conjunto (S, A, P, R), onde temos: um conjunto finito de estados S do sistema, um conjunto finito de ações A, um modelo de transição de estados P, que mapeia os pares estado-ação em uma distribuição de probabilidades sobre o conjunto de estados, e, finalmente, uma função de recompensa R, que especifica o reforço que o agente recebe por escolher uma determinada ação a Î A no estado s Î S. O estado s e a ação a atuais determinam a) o próximo estado s' de acordo com a probabilidade P(s'|s,a), e b) a recompensa r(s,a) associada.

Se o modelo de transição de estados for conhecido, técnicas de Controle Ótimo baseadas em Programação Dinâmica podem - ao menos em tese - ser utilizados para determinar uma política ótima de ações a ser seguida pelo agente. Alternativamente, aprendizagem por reforço é utilizada quando o modelo não está disponível (aprendizagem autônoma) ou quando existe apenas um modelo de simulação, que não permite a formulação analítica necessária para algoritmos de Programação Dinâmica.

A **Propriedade de Markov** enuncia o seguinte:

“Dado o presente, o futuro independe do passado.”

Vamos meditar um pouco sobre isso. Basicamente, um estado seguirá essa propriedade caso todos os estados anteriores a ele não influenciarem na decisão do próximo estado, apenas o estado atual. Não são todos os estados que obedecem a **Propriedade de Markov**, mas caso obedeçam, o problema torna-se mais fácil.

Passando essa relação à matemática obtemos a seguinte expressão:

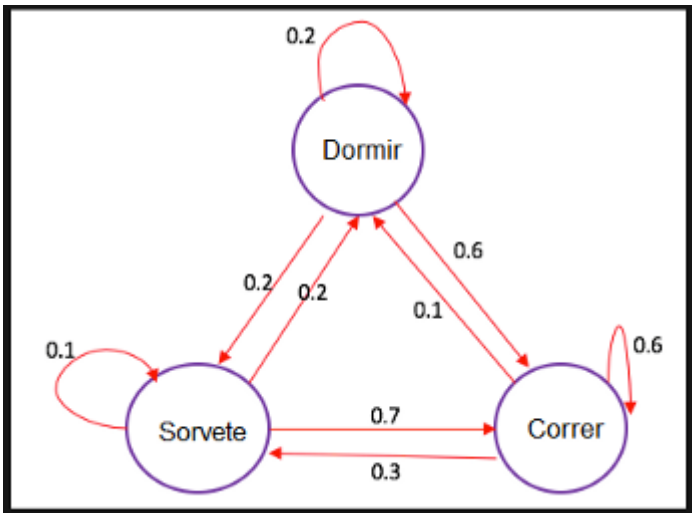
$$P(S_{t+1}|S_t) = P(S_{t+1}|S_1, S_2, ..., S_t)$$

Propriedade de Markov

Esse bando de letras significa algo muito simples, mas primeiro precisamos explicar o que esses símbolos significam. P(A | B, C) significa “probabilidade de A ocorrer dado que B e C ocorreram”. A probabilidade de eu passar para o estado St+1

dado que eu estou no estado St (parte esquerda da equação) é igual a probabilidade de eu passar para o estado St+1 dado todos estados em que estive em meu histórico. Significando que todos os estados antes do atual não são importantes para a passagem de um estado novo.

Figura 17 – Propriedade de Markov.



Se o meu estado atual é correr, não importa quantas vezes eu já dormi ou se antes eu estava tomando sorvete, a probabilidade de eu ir tomar sorvete ainda é **0.3**, a de dormir é **0.1** e a de continuar correndo é de **0.6**.

Função de Transição (Matriz de Probabilidade de Transição de Estados)

Todas essas probabilidades de mudança de estado podem ser colocadas em uma matriz chamada de Matriz de Probabilidade de Transição, com seu modelo geral dado abaixo. As linhas da matriz representam o estado atual, com os seus elementos sendo as probabilidades de mudar para os respectivos estados, tendo eles que somar 1. Vale notar que estas mudanças de estado registradas pela matriz acontecem naturalmente e não são decisões do agente sendo dessa forma também parte do ambiente.

$$P = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,n} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n,1} & p_{n,2} & \cdots & p_{n,n} \end{pmatrix}$$

Matriz de probabilidade de transição genérica

A matriz criada para o exemplo anterior, com os estados Dormir, Sorvete e Correr sendo os estados 1, 2 e 3, respectivamente, seria a seguinte:

$$P = \begin{pmatrix} 0.2 & 0.2 & 0.6 \\ 0.2 & 0.1 & 0.8 \\ 0.1 & 0.3 & 0.6 \end{pmatrix}$$

Matriz de Transição Dormir, Sorvete e Correr

Portando o Processo de Markov consiste em um espaço de estados S associado a uma função de transição P onde todos os estados seguem a ***propriedade de Markov***. No exemplo anterior, S seriam os três estados possíveis e P a sua matriz de probabilidade de transição.

Com processos de Markov (e as variações que veremos abaixo) podemos simular nosso ambiente em um problema de aprendizado por reforço.

Ambientes Completamente e Parcialmente Observáveis

Ambientes Completamente Observáveis são ambientes em que todas as informações estão disponíveis para o agente. Ex.: jogos de informação perfeita como Xadrez ou Go.

Figura 18 – Ambientes completamente observáveis.

Fonte: Imagem por Mitchell Johnson em Unsplash.

Ambientes Parcialmente Observáveis: nem todas as informações sobre o ambiente estarão disponíveis para o agente. Exemplo: jogos de informação imperfeita como jogos de cartas.

Figura 19 – Ambientes parcialmente observáveis.

Fonte: Imagem por Inês Ferreira em Unsplash.

Tarefas Episódicas e Contínuas

Tarefas episódicas: São aqueles que chegam ao fim alcançando um estado terminal. Podemos dizer que eles têm estados finitos. Exemplo: Uma partida de Go.

Tarefas contínuas: São tarefas infinitas, sem um estado terminal. Exemplo: simular o comportamento Humano.

Recompensa e Retorno

Para se definir **Retorno** deve-se primeiro definir **Recompensa (R)**, um certo valor que é dado para cada estado para indicar se ele é desejável ou não, como por exemplo, correr pode ter uma recompensa positiva e tomar sorvete pode ter uma recompensa negativa.

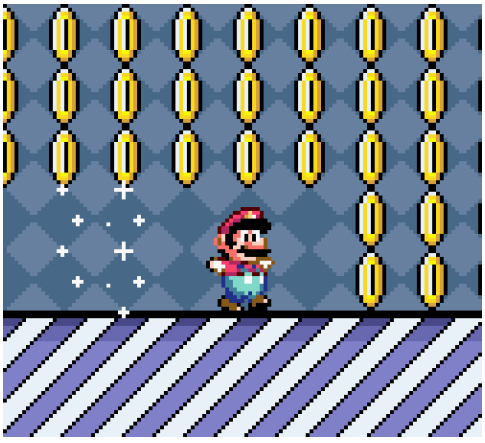
No Aprendizado por Reforço, a métrica que devemos maximizar trata-se da soma cumulativa de todas as recompensas que o agente recebe e não apenas a recompensa imediata do estado atual. A essa soma, damos o nome de **Retorno (Gt)**.

$$G_t = R_t + R_2 + ... + R_t$$

Retorno

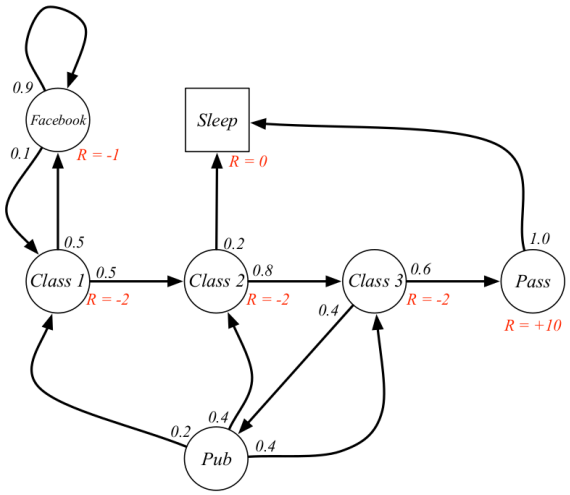
Podemos comparar a relação de recompensa e retorno com pontos em videogames. Dessa forma, toda vez que recebemos pontos é como se estivéssemos recebendo recompensas e nossa pontuação final seria nosso retorno.

Figura 20 – Retorno.



Exemplo: Considere o seguinte Processo de Recompensa de Markov (que explicaremos do que se trata mais a frente):

Figura 21 – Processo de Recompensa de Markov.



Se percorrermos os estados da seguinte forma partindo de "***Class 2***" : "***Class 2***" -> "***Class 3***" -> "***Pass***" -> "***Sleep***". Teremos acumulado as recompensas gerando o seguinte retorno: Gt = -2 + -2 + 10 + 0 = 6.

Para tarefas episódicas, o retorno é fácil de ser calculado, pois será a soma de todas as recompensas obtidas pelo agente. Mas para tarefas contínuas, como a atividade não tem fim e não podemos somar até o infinito, há a necessidade da inserção de um **fator de desconto** (***γ***).

O fator de desconto é um **hiperparâmetro** (definido pelo programador) e consiste em um número entre 0 e 1 que define a importância das recompensas futuras em relação à atual.

Valores mais próximos ao 0 dão mais importância a recompensas imediatas enquanto os mais próximos de 1 tentarão manter a importância de recompensas futuras.

Na prática, um algoritmo com fator de desconto 0 nunca aprenderá uma vez que irá considerar apenas as recompensas imediatas e com 1 continuará somando recompensas futuras até o infinito. Portanto, um valor ótimo de um fator de desconto está entre **0.2** e **0.8**.

Utilizando o fator de desconto, podemos escrever o retorno da seguinte forma:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Retorno com desconto

Exemplo: Utilizando a mesma sequência do exemplo anterior, com um fator de desconto de 0.5, temos: Gt = -2 + 0.5*-2 + 0.25*10 + 0.125*0 = -0.5.

Função de Valor

A função de valor basicamente determina o quão bom é estar em um estado específico, para o agente.

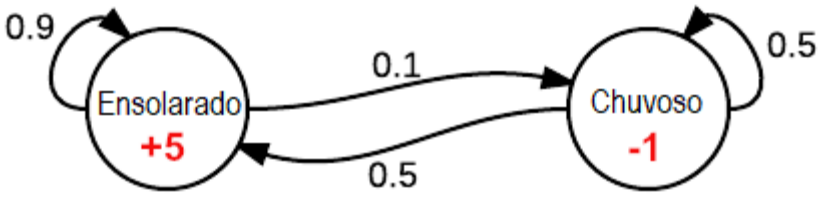
A função de valor V(s) é o **retorno esperado** iniciando no estado **s** e seguindo a política pelos próximos estados até encontrar um estado terminal (e o valor da recompensa deste será 0). A função (também chamada de função de estado-valor) é definida por:

$$V(s) = E[G_t | S_t = s]$$

Processo de Recompensa de Markov (MRP)

Juntando todos os conceitos abordados até aqui, podemos definir o **Processo de Recompensa de Markov**, que é basicamente um Processo de Markov onde adicionamos também a função de recompensa que nos dá a próxima recompensa esperada para cada estado e o fator de desconto (S, P, R, ***γ***).

Exemplo:



Com todas as informações armazenadas por um **MRP**, é possível calcular a função de valor para cada estado.

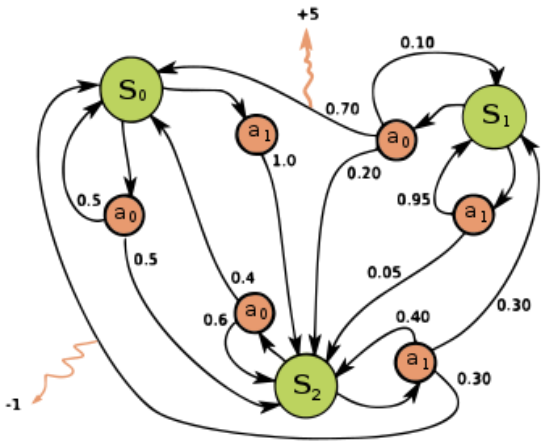
Se utilizarmos operações com matrizes para o cálculo, obtemos:

$$V(s) = \underbrace{R(s)}_{\text{recompensa imediata}} + \underbrace{\gamma \sum_{s' \in S} P(s'|s)V(s')}_{\text{soma descontada das recompensas futuras}}$$

$$\begin{pmatrix} V(s_1) \\ \vdots \\ V(s_N) \end{pmatrix} = \begin{pmatrix} R(s_1) \\ \vdots \\ R(s_N) \end{pmatrix} + \gamma \begin{pmatrix} P(s_1|s_1) & \cdots & P(s_N|s_1) \\ P(s_1|s_2) & \cdots & P(s_N|s_2) \\ \vdots & \ddots & \vdots \\ P(s_1|s_N) & \cdots & P(s_N|s_N) \end{pmatrix} \begin{pmatrix} V(s_1) \\ \vdots \\ V(s_N) \end{pmatrix}$$

Finalmente, o **Processo de Decisão de Markov** trata-se de um Processo de Recompensa de Markov com as ações que podem ser tomadas (S, A, P, R, ***γ***), onde A é o espaço de todas ações possíveis em todos os estados existentes.

Figura 22 – Processo de Decisão de Markov.



Para um **Processo de Decisão de Markov**, além da função de valor V(s), há também o retorno esperado da tomada de uma ação**,** chamado de **Q valor** ou **função de valor da ação.**

$$Q(s,a) = R(s,a) + \gamma V(s')$$

Função ação valor

Basicamente é uma função de valor, mas para uma ação. Ela determina a esperança da recompensa que uma ação tomada pode devolver. Lê-se “Retorno esperado (ou simplesmente valor) da ação ‘a’ no estado ‘s’ é igual à recompensa da ação ‘a’ no estado ‘s’ mais o valor descontado do estado seguinte”.

Q-Learning

Tudo o que é necessário para que essa convergência ocorra é que todos pares estado-ação continuem sendo atualizados. Sob essa suposição e com uma variação das condições usuais de aproximação estocástica, que gera uma sequência de variações do valor da variável α, é provado que Q converge para a política de ação-valor ótima q*.

A independência de política significa que o Q-Learning é capaz de convergir, ao longo de seu processo de aprendizado, para uma política ótima mesmo que não esteja agindo de forma ótima a todo o momento.

Deep Reinforcement Learning Algorithm

Antes de entendemos o significado do “deep”, vamos relembrar o Q-Learning.

Q-Learning é um algoritmo que utiliza uma tabela (Q-Table) que mapeia qual o valor (ou qualidade) de uma ação em determinado estado. Assim o nosso agente precisa apenas consultar a tabela para escolher uma ação.

Os valores da Q-Table são calculados realizando uma fase de exploração dos estados. Nessa etapa, o agente escolhe ações aleatórias e com base nas recompensas recebidas a tabela é atualizada seguindo a equação:

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma max_{a'} Q(s',a') - Q(s,a)]$$

- Q(s, a) é a Q-Function.
- s é o estado.
- a é ação.
- α é a taxa de aprendizagem.
- r é a recompensa recebida.
- γ é um desconto aplicado aos valores das ações do novo estado, já que essas ações não têm garantia de acontecerem.

Com isso em mente, foi proposto o chamado Deep Q-Learning, que utiliza Redes Neurais como substitutas para as Q-Tables e permite entradas dimensionalmente maiores.

O princípio é que a rede neural receba como entrada um estado e suas saídas sejam os valores associados à cada uma das ações possíveis, assim a de maior valor será escolhida.

O que nós queremos com a rede neural é minimizar o erro entre o valor da Q-Function para o estado atual e o próximo estado.

$$y_j = \begin{cases} r_j, \text{se o estado final} \\ r_j + \gamma max_{a'}(estado_{j+1}, a', \theta), \text{caso contrário} \end{cases}$$

$$Loss = (y_j - Q(estado_j, a_j, \theta))^2$$

Uma otimização sugerida é chamada de Experience Replay, que basicamente é um buffer que grava os estados, ações, recompensas e próximos estados cada vez que o agente realiza uma ação.

Para o treinamento são escolhidos conjuntos aleatórios desse buffer que servirão para atualizar os pesos da rede. Isso evita que a rede "esqueça" comportamentos mais antigos.

Referências

ABE, Naoki; ZADROZNY, Bianca; LANGFORD, John. Outlier detection by active learning. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006. p. 504-509.

ABRAHAM, Bovas; BOX, George E. P. Bayesian Analysis of Some Outlier Problems in Time Series. In: *Biometrika*, v. 66. n. 2, 1979. p. 229-236.

ABRAHAM, Bovas; CHUANG, Alice. Outlier Detection and Time Series Modeling. In: *Technometrics*, v. 31. n. 2, 1989. p. 241-248.

AGGARWAL, Charu C. *Outliers Analysis*. Springer. 1. ed. 2013.

ALVES, Ricardo Brito. *Redução da influência de confundidores em modelos de aprendizado de máquina*. Dissertação (Mestrado Acadêmico em Engenharia Elétrica) - Pontifícia Universidade Católica de Minas Gerais, Belo Horizonte. 2018.

AMER, Mennatallah; GOLDSTEIN, Markus. Nearest-Neighbor and Clustering based Anomaly Detection Algorithms for RapidMiner. In: *Proc. Of the 3rd RapidMiner Community Meeting and Conferernce*, 2012.

ANSCOMBE, F.J., GUTTMAN I. Rejection of Outliers. In: *Technometrics,* v. 2, 1960. p. 123-147.

APRENDIS. *Detecção de anomalias*. Disponível em: <http://aprendis.gim.med.up.pt/index.php/Detecção_de_anomalias>. Detecção de anomalias. Acesso em: 12 jan. 2021.

APRENDIZADO DE MÁQUINA. In: *WIKIPÉDIA, a enciclopédia livre*. Flórida: Wikimedia Foundation, 2020. Disponível em: <https://pt.wikipedia.org/w/index.php?title=Aprendizado_de_m%C3%A1quina&oldid=57895923>. Acesso em: 12 jan. 2021.

ASSIS, Pablo de. *O que são Redes Neurais?* TecMundo, 2009. Disponível em: <http://www.tecmundo.com.br/programacao/2754-o-que-sao-redes-neurais-.htm>. Acesso em: 12 jan. 2021.

BARBARÁ, Daniel; WU, Ningning; JAJODIA, Sushil. Detecting novel network intrusions using bayes estimators. In: *First SIAM Conf Data Min,* 2001. p. 1-17.

BECKMAN R.J.; COOK R.D. OUTLIERS. In: *Technometrics*, v. 25, 1983. p. 119- 149.

BENEVIDES, Thiago. *Os algoritmos de machine learning*. GeekHunter, 2019. Disponível em: <https://blog.geekhunter.com.br/aprendizado-de-maquina-e-seus-algoritmos/>. Acesso em: 12 jan. 2021.

BENIGER, J.R; BARNETT, V. LEWIS, T. Outliers in Statistical Data. In: *Contemp Sociol*, v. 9, 1980. p. 560.

BHUYAN, M. H.; BHATTACHARYYA, D. K.; KALITA, J. K. An effective unsupervised network anomaly detection method. In: *Proc. of the Conference on Advances in Computing, Communications, and Informatics*. p.533-539, 2012.

BIANCO, Ana; GARCIA BEN, Marta; MARTÍNEZ, E.; YOHAI, Victor. Outlier Detection in Regression Models with ARIMA Errors using Robust Estimates. In: *Journal of Forecasting*, v. 20, 2001. p. 565-579.

BRAGA, A. P.; LUDERMIR, T.B.; CARVALHO, A.C.P.L.F. *Redes Neurais Artificiais: Teoria e Aplicações*. Livros Técnicos e Científicos S.A., 2000.

BREUNIG, M.M.; KRIEGEL, H.P.; N.G., R.T.; SANDER, J. LOF: Identifying Density-Based Local Outliers. In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data - SIGMOD '00*. Nova Iorque, EUA: ACM Press, 2000. p. 93-104.

BRONSTEIN A.; et al. SELF-AWARE SERVICES: using Bayesian networks for detecting anomalies in Internet-based services. In: *2001 IEEE/IFIP International Symposium on Integrated Network Management Proceedings. Integrated Network*

Management VII. Integrated Management Strategies for the New Millennium, 2001. p. 623-638.

CHANDOLA, V.; BANERJEE, A.; KUMAR, V. Anomaly detection. In: *ACM Comput Surv*, v. 41, 2009. p. 1-58.

CHANDOLA, V; BANERJEE, A.; KUMAR, V. **Outlier detection:** A survey. ACM Computing Surveys, 2007.

CHE, D.; SHAO, X.; HU, B.; SU. Q. Simultaneous wavelength selection and outlier detection in multivariate regression of near-infrared spectra. In: *Anal Sci*, v. 21, 2005. p. 161-166.

CHOLLET, F. *Deep Learning with Python*. 1. ed. Manning. 2017.

CLAYTONET. *Mineração de dados*. Disponível em: <http://claytonet.net/blog/mineracao-de-dados/>. Acesso em: 12 jan. 2021.

DATA SCIENCE BRIGADE. *A Diferença Entre Inteligência Artificial, Machine Learning e Deep Learning*. 2016. Disponível em: <https://medium.com/data-science-brigade/a-diferença-entre-inteligência-artificial-machine-learning-e-deep-learning-930b5cc2aa42>. Acesso em: 12 jan. 2021.

DATA SCIENCE BRIGADE. *Turn your data into decisions*. 2017. Disponível em: <https://medium.com/data-science-brigade/hire-experts-516d69c1c47c>. Acesso em: 12 jan. 2021.

DE STEFANO, C.; SANSONE, C.; VENTO, M. To reject or not to reject: that is the question - an answer in case of neural classifiers. In: *IEEE Trans Syst Man Cybern Part C Appl Rev*, v. 20, 2000. p. 84-94.

DESFORGES, M.J.; JACOB, P.J.; COOPER, J.E. Applications of probability density estimation to the detection of abnormal conditions in engineering. In*: Proc Inst Mech Eng Part C J Mech Eng Sci*, v. 212, 1998. p. 687-703.

ENDERLEIN, G.; HAWKINS, D. M. *Identification of Outliers*. Londres: Chapman and Hall, v. 29, n. 2, 1987.

FILHO, Cléuzio Fonseca. H*istória da Computação - O caminho do pensamento e da tecnologia*. Porto Alegre: EDIPUCRS - PUCRS (Domínio Público), 2007.

FOX, A.J. Outliers in time series. In: *J R Stat Soc Ser B,* v. 34, 1972. p. 350–363.

FUJIMAKI, R.; YAIRI, T.; MACHIDA, K. An approach to spacecraft anomaly detection problem using kernel feature space. In: *Proceeding Elev ACM SIGKDD Int Conf Knowl Discov data Min - KDD '05*, 2005. p. 401.

FUKUSHIMA, William. *Aprendizado por Reforço #2 | Processo de Decisão de Markov — Parte 1*. 2020. Disponível em: <https://medium.com/turing-talks/aprendizado-por-reforço-2-processo-de-decisão-de-markov-mdp-parte-1-84e69e05f007>. Acesso em: 12 jan. 2021.

GAEA CONSULTING. *Afinal, o que é Deep Learning?*. Disponível em: <https://gaea.com.br/afinal-o-que-e-deep-learning>. Acesso em: 12 jan. 2021.

GALEANO, P.; PEÑA, D.; TSAY, R. Outlier detection in multivariate time series by projection pursuit. In: *J Am Stat Assoc*, v. 101, 2006. p. 654-669.

GARCIA, Debora. *Você sabe o que é deep learning? Esse artigo vai te ajudar a entender os benefícios de uma aprendizagem imersiva*. Geekie, 2018. Disponível em: <http://info.geekie.com.br/deep-learning/>. Acesso em: 12 jan. 2021.

GOLDBERGER, Ary. et al. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. In: *Circulation*, v. 101, 2000. p. 215–220.

GOMES, Thales de Oliveira. *Redes Neurais - O que é e características*. 2017. Disponível em: <https://thalesgomes.escavador.com/artigos/1611/redes-neurais-o-que-e-e-caracteristicas>. Acesso em: 12 jan. 2021.

GOODFELLOW, Ian; et. al. *Deep Learning*. 1. ed. MIT Press, 2016.

GRUBBS, F.E. Procedures for Detecting Outlying Observations in Samples. In: Technometrics, v. 11, 1969. p. 1-21.

HAWKINS, D. *Identification of Outliers*. Londres: Chapman and Hall, 1980.

HAWKINS, Simon; HE, Hongxing; BAXTER, Rohan. Outlier Detection Using Replicator Neural Networks. In: *Proceedings of the 4th International Conference on Data Warehousing and Knowledge Discovery*, 2002. p. 170–180.

HAYKIN, S. *Redes Neurais – Princípios e prática*. 2. ed. Porto Alegre, RS: Bookman, 2001.

HE, Zengyou; XU, Xiaofei; DENG, Shengchun. Discovering cluster-based local outliers. In: Pattern Recognition Letters, v. 24, 2003. p. 1641-1650.

HORN, P.S.; FENG, L.; LI, Y.; PESCE, A.J. Effect of outliers and nonhealthy individuals on reference interval estimation. In: Clin Chem, v. 47, 2001. p. 2137-2145.

INOVAÇÃO TECNOLÓGICA. *Chip neural: rumo ao computador neuromórfico*. 2011. Disponível em: <http://www.inovacaotecnologica.com.br/noticias/noticia.php?artigo=chip-neural-computador-neuromorfico#.WDCYaX2uYZM>. Acesso em: 12 jan. 2021.

INTELIGÊNCIA ARTIFICIAL. In: *WIKIPÉDIA, a enciclopédia livre*. Flórida: Wikimedia Foundation, 2020. Disponível em: <https://pt.m.wikipedia.org/wiki/Inteligência_artificial>. Acesso em: 12 jan. 2021.

JACOBS, Paige. *Comprehensive Beginner's Guide to Aprendizado de Máquina in Python with Exercises and Case Studies*. Independently Published, 2019.

KNORR, E.; NG R. *A unified approach for mining outliers*. In: PROCEEDINGS KNOWLEDGE DISCOVERY KDD, p. 219-222, 1997.

LAURIKKALA J.; et al. Informal identification of outliers in medical data. In: *Fifth International Workshop On Intelligent Data Analysis In Medicine And Pharmacology*, p. 20–24, 2000.

LE, James. *10 Algoritmos de Machine Learning que você precisa conhecer*. 2016. Disponível em: <https://semantix.com.br/10-algoritmos-de-machine-learning-que-voce-precisa-conhecer/>. Acesso em: 12 jan. 2021.

LECUN, Y.; BENGIO, Y.; HINTON, G. *Deep learning, Nature*. v.521(7553), p.436-444, 2015.

LIN, J.; KEOGH, E.; FU, A.; VAN HERLE, H. *Approximations to Magic*: Finding Unusual Medical Time Series. In: 18th IEEE Symp Comput Med Syst p.329–334, 2005.

MATOS, David. *Conceitos Fundamentais de Machine Learning*. Ciência e Dados, 2020. Disponível em: <http://www.cienciaedados.com/conceitos-fundamentais-de-machine-learning/>. Acesso em: 12 jan. 2021.

MITCHELL, T. *Aprendizado de Máquina*. McGraw-Hill, 1997.

MOTULSKY, H. *Intuitive Biostatistics*: Choosing a Statistical Test. v.17, 1995.

PANG-NING, T.; STEINBACH, M.; KUMAR, V. *Introduction to Data Mining*. 2006.

PETENATE, Marcelo. *BoxPlot: Saiba tudo sobre o Diagrama de caixa e como interpretar esse gráfico*. 2019. Disponível em: <https://www.escolaedti.com.br/o-que-e-um-box-plot>. Acesso em: 12 jan. 2021.

PETENATE, Marcelo. *Por que é importante entender correlação entre variáveis*. 2016. Disponível em: <https://www.escolaedti.com.br/entender-correlacao-entre-variaveis>. Acesso em: 12 jan. 2021.

RAMASWAMY, S.; RASTOGI, R.; SHIM, K. *Efficient algorithms for mining outliers from large data sets*. ACM SIGMOD Rec. v.29 p.427–438. 2000

RAVINDRA, Savaram. *Como as Redes Neurais Convolucionais realizam o reconhecimento de imagem*. 2017. Disponível em: <https://www.infoq.com/br/articles/redes-neurais-convolucionais/>. Acesso em: 12 jan. 2021.

referência do prof: Proc 24th VLDB Conf 1998, New York C:194–205.

ROTH, V. *Kernel Fisher Discriminants for Outlier Detection*. Neural Comput, v.18. p.942–960. 2006.

ROTH, V. *Outlier detection with one-class kernel Fisher discriminants*. Adv Neural Inf Process Syst. p.1169–1176. 2005.

SALLES, Elis. *Redes Neurais*. Disponível em: <https://www.elissalles.com.br/blog-conteudo/3481/redes-neurais>. Acesso em: 12 jan. 2021.

SCHÖLKOPF, B.; PLATT, JC.; SHAWE-TAYLOR, J.; SMOLA, A. J.; WILLIAMSON, R. C. *Estimating the support of a high-dimensional distribution*. Neural Comput. v.13. p.1443–1471. 2001.

SOLBERG, HE.; LAHTI, A. *Detection of outliers in reference distributions*: Performance of horn's algorithm. Clin Chem. v.51. p.2326–2332. 2005.

STEFANSKY, W. *Rejecting outliers in factorial designs*. Technometrics. v.14. p.469–479. 1972.

STEINWART, I.; GOV, D.; SCOVEL, C.; GOV, J. *A Classification Framework for Anomaly Detection Don Hush*. J Mach Learn Res. v.6. p.211–232. 2005.

SUMARES, Gustavo. *Saiba como as redes neurais deixam os computadores 'inteligentes'*. Olhar Digital, 2016. Disponível em: <http://olhardigital.uol.com.br/pro/noticia/saiba-como-as-redes-neurais-deixam-os-computadores-inteligentes/60876>. Acesso em: 25 juun. 2020.

TEAM, R. C. R. *A language and environment for statistical computing***.** Vienna: R Foundation for Statistical Computing, 2013. [S.l.]: ISBN 3-900051-07-0, 2014. 21, 51

THATTE, G.; MITRA, U.; HEIDEMANN, J. Parametric methods for anomaly detection in aggregate traffic. In: *IEEE/ACM Transactions on Networking*, v.19, 2011. p. 512-525.

THEILER, J.P.; CAI, D.M. Resampling approach for anomaly detection in multispectral images. In: *Proc SPIE*, n. 5093, set. 2003, p. 230–240.

TORR, P.H.S; MURRAY, D.W. Outlier Detection and Motion Segmentation. In: *Sens Fusion VI*, 1993. p. 432–443.

UBC. *Data Science for Social Good*. The University of British Columbia – Data Science Institute. Disponível em: <https://dsi.ubc.ca/data-science-social-good>. Acesso em: 12 jan. 2021.

VINICIUS, Anderson. *Introdução ao Aprendizado de Máquina*. Disponível em: <https://medium.com/@avinicius.adorno/introdução-a-aprendizado-de-máquina-e39ec5ef459b>. Acesso em: 12 jan. 2021.

WAINER, H. “Book Reviews: Robust Regression & Outlier Detection: Peter J. Rousseeuw and Annick M. Leroy New York: Wiley, 1987. xiv + 329 pp,” J. Educ. Behav. Stat., vol. 13, no. 4, pp. 358–364, Dec. 1988.

WALLER, M. A.; FAWCETT, S. E. Data science, predictive analytics, and big data: a revolution that will transform supply chain design and management. In: *Journal of Business Logistics, Wiley Online Library*, v. 34, n. 2, p. 77–84, 2013.

WAZLAWICK, Raul Sidne. História da Computação In: *Série Didática da Sociedade Brasileira de Computação*. Rio de Janeiro: Elsevier, 2016.

WEBER. R.; SCHEK H.J.; BLOTT, S. A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. In: *VLDB Conference*, n. 24, 1998, Nova York.