Fundamentos

Bootcamp Analista de Machine Learning

João Paulo Barbosa Nascimento

2020

Fundamentos

Bootcamp Analista de Machine Learning

João Paulo Barbosa Nascimento

Sumário

Capítulo 1. Introdução ao Aprendizado de Máquina

A construção de máquinas capazes de executar tarefas consideradas inteligentes já é bastante antiga. Seu desenvolvimento é dado com hardwares disponíveis em cada época, como o caso do Ábaco e das primeiras calculadoras mecânicas. Esse capítulo tem o objetivo de apresentar a história da inteligência artificial, uma introdução ao Aprendizado de Máquina e às formas de aprendizado.

História da inteligência artificial

O primeiro trabalho agora reconhecido como inteligência artificial foi realizado por Warren McCulloch e Walter Pitts em (1943). Eles se basearam em três fontes: o conhecimento da fisiologia básica e da função dos neurônios no cérebro, uma análise formal da lógica proposicional criada por Russell e Whitehead e a teoria da computação de Turing. Esses dois pesquisadores propuseram um modelo de neurônios artificiais, no qual cada neurônio se caracteriza por estar ligado ou desligado, com a troca para o estado ligado ocorrendo em resposta a estimulação por um número suficiente de neurônios vizinhos.

O estado de um neurônio era considerado “equivalente em termos concretos a uma proposição que definia seu estímulo adequado”. Por exemplo, eles mostraram que qualquer função computável podia ser calculada por uma certa rede de neurônios conectados, e que todos os conectivos lógicos (e, ou, não etc.) podiam ser implementados por estruturas de redes simples. McCulloch e Pitts também sugeriram que redes definidas adequadamente seriam capazes de aprender. Donald Hebb (1949) demonstrou uma regra de atualização simples para modificar as intensidades da conexão entre neurônios. Sua regra, agora chamada aprendizagem de Hebb, continua sendo um modelo influente até hoje.

Dois alunos do departamento de matemática da Universidade de Princeton, Marvin Minsky e Dean Edmonds, construíram o primeiro computador em rede neural em 1951. O SNARC, como foi chamado, usava 3 mil válvulas eletrônicas e um

mecanismo de piloto automático retirado de um bombardeiro B-24 para simular uma rede de 40 neurônios. A banca examinadora da tese de doutorado de Minsky se mostrou cética sobre esse tipo de trabalho, sem saber se ele deveria ser classificado como um trabalho de matemática. Porém, segundo contam, Von Neumann teria dito: “Se não é agora, será algum dia”. Mais tarde, Minsky acabou provando teoremas importantes que mostravam as limitações da pesquisa em redes neurais.

Posteriormente, surgiram vários outros trabalhos que hoje podem ser caracterizados como IA, mas foi Alan Turing quem primeiro articulou uma visão completa da IA em seu artigo de 1950 intitulado “*Computing Machinery and Intelligency*”. Nesse artigo, ele apresentou o teste de Turing, aprendizagem de máquina, algoritmos genéticos e aprendizagem por reforço. Abaixo são listados algumas fases da inteligência artificial.

- **Gestação da IA (1943-1955):** nessa fase foi publicado o primeiro trabalho reconhecido como IA (Russel e Whitehead) e a teoria da computação de Alan Turing. Além disso, foi criado o primeiro computador de Rede Neural (SNRC).

- **Nascimento da IA (1956):** fase de John McCarthy (Universidade de Princeton). Criação da teoria dos autômatos, Redes Neurais e estudos da IA em geral. Criação do programa Logic Theorist (LT), programa de raciocínio não numérico.

- **Entusiasmo inicial e grandes expectativas (1952-1969)**: McCarthy referiu-se à essa época como a era do “Olha mamãe, sem as mãos”. Esperava-se uma grande evolução e visibilidade da IA. Criação do GPS (General Problem Solver) que foi projetado para imitar protocolos humanos de resolução de problemas. Nessa época foi criado o LISP, que foi uma linguagem dominante da IA.

- **Uma dose de realidade (1966-1973)**: Herbert Simon disse nessa época que em 10 anos um computador seria campeão de xadrez e também resolveria um teorema significativo. Essas previsões só vieram a ser efetivamente realizadas em 40 anos. O otimismo de Simon se devia aos promissores resultados iniciais

da IA, porém aplicada a problemas simples. Esses experimentos falharam para problemas mais extensos. A IA, nessa época, resolvia problemas experimentando diferentes combinações até encontrar a solução. A principal crítica sobre a IA era sua incapacidade em conviver com a explosão combinatória.

- **Sistemas baseados em conhecimento (1969- 1979)**: segundo Dendral, em 1969, para resolver um problema difícil é necessário saber respostas previamente. O problema estava em aumentar o número de objetos ou instâncias. Com isso, surgiu o primeiro sistema de conhecimento intensivo, pois era necessário capturar o conhecimento humano. Surgimento das linguagens de representação de conhecimento PROLOG e PLANNER.

- **A IA se torna indústria (1980 até hoje)**: primeiro sistema bem-sucedido (RI-1982) que auxiliava a configurar novos pedidos de computadores. Em 1986 ressurgem as Redes Neurais que foram outrora abandonadas nos anos 1970. Em 1980, houve a recriação do algoritmo de retropropagação, e em 1987 a IA se torna uma ciência com firmeza do método científico. Com isso, uma nova indústria surgiu e houve diversas revoluções, como no campo da robótica, visão computacional e representação do conhecimento.

Atualmente temos diversas aplicações bem-sucedidas de IA, como o Deep Blue da IBM que foi o primeiro programa a derrotar o campeão mundial de xadrez Kasparov. Temos a aplicação da IA em diversos outros campos, como automóveis autônomos (percorreu 4600 Km nos EUA e manteve o controle em 98% do tempo), diagnóstico de doenças, robótica e muito mais.

Outro ponto fundamental na evolução da IA foi a melhoria que tivemos no hardware. Máquinas mais poderosas e *clusters* que realizam o processamento de maneira paralela e distribuída estão atualmente disponíveis e com custos cada vez menores, auxiliando em uma popularização ainda maior das técnicas de IA.

Algumas perguntas são feitas há muitos anos: computadores são capazes de aprender? Aprender assim como os humanos?

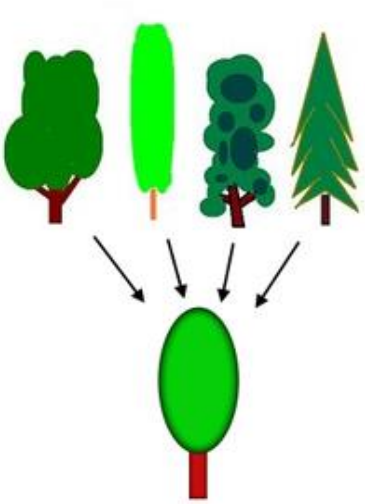
Podemos dizer que vários algoritmos de Aprendizado de Máquina estão alcançando bons resultados. Sabemos que os programadores não são capazes de antecipar todas as situações que podem acontecer em um programa. Isso seria inviável de ser programado. Imagine que precisássemos implementar todas as decisões possíveis que um carro autônomo pode tomar. Poderíamos ter um carro autônomo especificamente para andar em uma cidade. Se precisássemos que ele andasse em outra cidade, precisaríamos reprogramá-lo.

Outra pergunta que tem mexido com a cabeça de muita gente envolvida com técnicas e aplicações de Aprendizado de Máquina é: quando é necessário aplicar e utilizar Aprendizado de Máquina?

A melhor resposta que temos para isso é: sempre que não for possível escrever um algoritmo determinístico para resolver o problema. Nesse caso, para implementar a solução, buscamos conhecimento por meio de exemplos.

É importante destacar que aprender não é memorizar. Desejamos que nossos algoritmos generalizem um comportamento diante de uma nova situação, muitas vezes inesperada. Computadores memorizam facilmente, porém não generalizam com facilidade. A Figura 1.1 apresenta um exemplo de generalização. Ao treinarmos a construção de um algoritmo de Aprendizado de Máquinas, apresentamos a ele diversos tipos de árvores. Se esse algoritmo tem a capacidade de generalizar, iremos apresentar uma árvore diferente, que ele nunca recebeu em seus treinamentos, e baseando-se nos exemplos anteriores, essa máquina será capaz de generalizar e classificar o novo objeto com um elemento do conjunto árvores.

Figura 1.1 - Exemplo de generalização.



As Figuras 1.2 e 1.3 apresentam a generalização para dois outros tipos de objetos, prédio (apartamento) e automóveis, respectivamente.

Figura 1.2 – Exemplo de generalização.

Figura 1.3 – Exemplo de generalização.

Na Figura 1.2, o algoritmo deve ser capaz de generalizar ao receber como treinamento o apartamento físico e, posteriormente, ao receber a planta de um apartamento, ele deverá ser capaz de classificá-la também como um apartamento. Na Figura 1.3, o algoritmo deve ser capaz de classificar tanto os automóveis novos como os velhos.

Durante o aprendizado do algoritmo, algumas fases são seguidas:

- Treinamento:
  - São apresentados exemplos para o sistema.
  - O sistema aprende com os exemplos.
  - O sistema modifica os parâmetros ajustáveis.
  - A saída se aproxima do desejado.
- Utilização:
  - Novos exemplos “desconhecidos” são apresentados.
  - Espera-se que o sistema generalize e apresente resultados satisfatórios.

É importante destacar que não desejamos que um algoritmo aprenda por memorização, pois diante disso teremos respostas apenas para o que está

memorizado. O ideal é aprender com objetivo de generalizar posteriormente. Devemos analisar a estrutura dos dados e não somente aprender a "boa resposta".

Formas de Aprendizado (supervisionado, não supervisionado e por reforço)

Durante a fase de treinamento temos alguns tipos específicos de aprendizado que podem influenciar diretamente no sucesso de um projeto.

Na forma de aprendizado supervisionado, observamos alguns exemplos de entrada e saída, pois dada uma entrada, teremos disponível uma saída. Essa forma de aprendizado tem o objetivo de aprender uma função que mapeia as entradas em saídas. Para isso, é necessário que informemos ao sistema qual é a resposta (saída) correta para uma determinada entrada. Com isso, as classes são conhecidas previamente. Essa técnica trata de algo eficiente, pois o sistema trabalha com os resultados corretos. O aprendizado supervisionado é útil para tarefas que envolvem classificação, regressão e estimativa de probabilidade condicional.

O processo de aprendizagem supervisionada segue a linha de, primeiramente, apresentar exemplos ao sistema com respostas. O sistema irá aprender a partir desses exemplos. Em seguida, inicia-se a fase de testes com novos dados que jamais foram apresentados para o agente (sistema). O ideal é que o agente tenha um bom poder de generalização.

Dado um conjunto de exemplos de treinamento $\{(x1, y1), (x2, y2), ..., (xn, yn)\}$, queremos aprender um mapeamento $f: X \rightarrow y$, tal que: $y_i \approx f(x_i), i = 1, ..., n.$

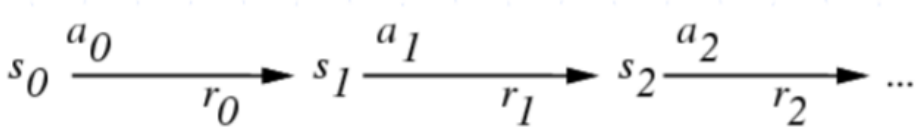
No aprendizado não supervisionado, o agente reconhece o padrão diretamente nos dados de entrada, mesmo sem conhecer nenhum feedback de saída, ou seja, mesmo sem conhecer nenhuma resposta prévia. Nesse tipo de aprendizado, as classes não são conhecidas e existe uma descoberta do conhecimento. Podemos dizer que os algoritmos que aprendem de maneira não supervisionada são algoritmos autodidatas. Esses algoritmos têm aplicação em reconhecimento de faces e

manuscritos, predição de ações e gerenciamento de riscos na área de finanças, além de predição de tráfego e jogos.

A aprendizagem por reforço lida com os conceitos de recompensas e penalidades. O objetivo dessa modalidade é escolher um conjunto de ações que maximize as recompensas.

A Figura 1.4 apresenta o fluxo da Aprendizagem por Reforço. A cada instante de tempo *t*, o agente está em um estado *s*, executa uma ação *a*, vai para o estado *s* e recebe uma recompensa *r*.

Figura 1.4 – Fluxo do aprendizado por reforço.



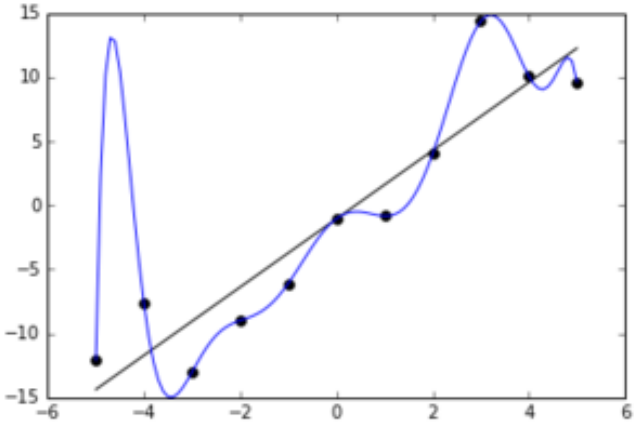
O agente fica sabendo de uma série de recompensas ou punições que podem ser recebidas durante a execução. Cabe ao agente tomar decisões que sempre maximizem as recompensas. Nesse tipo de aprendizagem, não damos a resposta para o sistema, ele mesmo faz uma hipótese e determina se ela foi boa ou ruim.

Generalização, overfitting e underfitting

Overfitting e underfitting são termos da Estatística, utilizados no Aprendizado de Máquina.

O overfitting (sobreajuste) ocorre quando o modelo estatístico se ajusta de maneira excessiva aos dados que foram fornecidos durante o treinamento. De certa forma, o algoritmo “presta muita atenção” nas particularidades dos dados de treinamento e não consegue generalizar de forma satisfatória. A Figura 1.5 apresenta um exemplo de overfitting.

Figura 1.5 – Exemplo de overfitting.



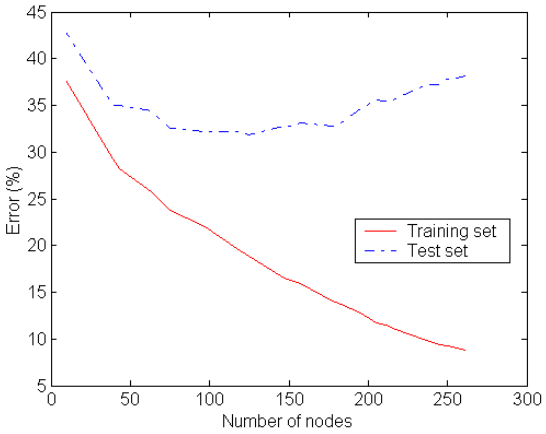
Podemos fazer uma analogia do overfitting com o fato de um estudante que memoriza toda a matéria antes de realizar uma prova, sem realmente entender o conteúdo. Aprender por memorização não permitirá a generalização em novos casos, assim o aluno não conseguirá resolver os problemas da prova caso o problema varie em comparação aos problemas resolvidos durante os estudos.

O overfitting apresenta baixa taxa de erros nos exemplos de aprendizagem (treinamento) e uma alta taxa para os dados de testes. Isso pode acontecer quando usamos um conjunto muito grande de exemplos com pequena variação entre as classes.

Para verificar o overfitting, divida o conjunto de dados em dois conjuntos, treino e testes. Treine e avalie o modelo com os dados de treinamento. Em seguida, avalie o modelo com os dados de teste. A diferença entre as duas avaliações irá mensurar a habilidade do sistema em generalizar.

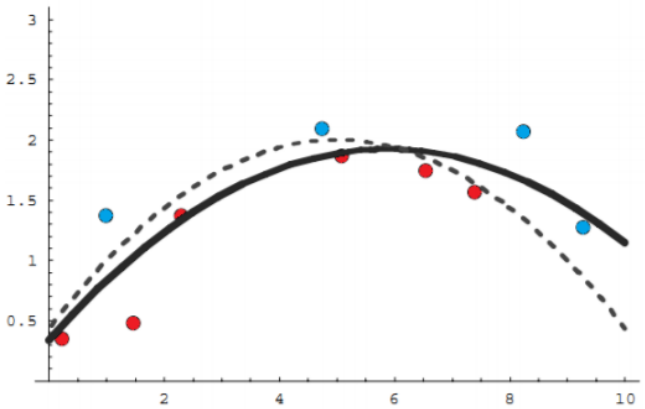
O underfitting (subajuste) é considerado um complemento ao problema do overfitting, pois apresenta problemas no próprio conjunto de dados de treinamento. Aqui temos uma alta taxa de erros na aprendizagem e no teste. A Figura 1.6 apresenta um comparativo entre o overfitting e o underfitting.

Figura 1.6 – Comparativo entre overfitting e underfitting.



Um bom modelo é aquele que é balanceado, isto é, que seja flexível o suficiente para capturar a forma da curva da função *f* e que não seja exatamente igual à função *f*. A Figura 1.7 apresenta um modelo balanceado. Observe que as duas curvas são próximas, mas não iguais.

Figura 1.7 – Modelo balanceado.



Dimensionalidade e a definição de problemas

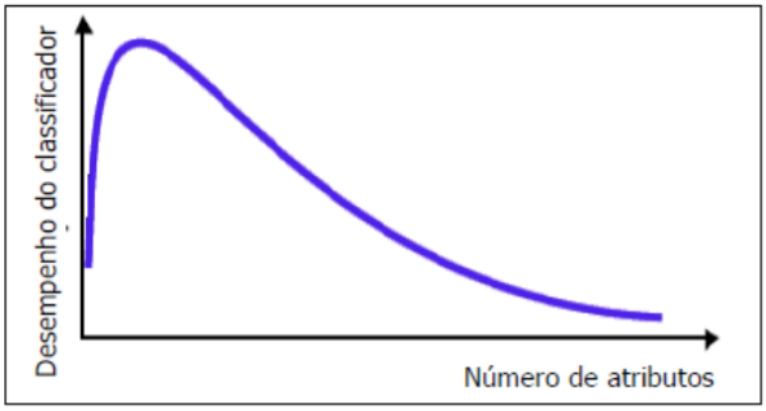
Suponha a seguinte situação: um conjunto de dados é descrito por meio de 20 atributos. Desses atributos, apenas dois são relevantes e os demais são ruins, podem ser descartados ou são correlacionados. O resultado da utilização de toda

essa quantidade de atributos será um desempenho ruim na classificação. O algoritmo KNN, por exemplo, é enganado quando a quantidade de atributos é muito grande.

A maldição da dimensionalidade é o termo que se refere a vários fenômenos que surgem na análise de dados em espaços com muitas dimensões (atributos). Muitas vezes podemos lidar com centenas ou milhares de dimensões. Um grande número de atributos tende a gerar informação redundante, e isso tende a prejudicar o desempenho do sistema.

O desempenho de um algoritmo tende a cair a medida que aumentamos a quantidade de atributos, mesmo que esses atributos sejam relevantes, conforme apresentado na Figura 1.8.

Figura 1.8 – Desempenho x Número de Atributos.



Cada atributo representa uma dimensão no espaço de características, e o volume cresce exponencialmente ao adicionar novos atributos. Se tivemos um atributo com dez valores possíveis, teremos dez objetos possíveis. Se tivermos cinco atributos com dez valores possíveis, teremos 10^5 objetos possíveis.

Nos espaços com muitas dimensões, as amostras se tornam esparsas e pouco similares, ficando distantes umas das outras. Por isso, é muito importante trabalhar a redução da dimensionalidade. Essa etapa é fundamental no projeto de um sistema de Machine Learning. Deve-se utilizar um número pequeno de atributos, fazendo a seleção adequada ou a composição de atributos. Pode-se ainda criar novos atributos via transformação de dados (agregação). Além disso, é necessário realizar

a seleção dos melhores atributos. Essa tarefa melhora a eficácia dos algoritmos, reduz o tamanho da amostra, simplifica o modelo e melhora a visualização dos resultados.

Por outro lado, problemas de aprendizado bem-definidos devem identificar a classe das tarefas, a medida de desempenho a ser melhorada e a fonte de conhecimento. Um programa de computador aprende a partir de uma experiência E, com uma classe de tarefas T e uma medida de desempenho P. Com isso, seu desempenho nas tarefas T, medido por P, melhora com a experiência E. Exemplo:

- Tarefa T: aprender a jogar xadrez.
- Medida de desempenho P: percentual de jogos ganhos contra adversários.
- Experiência de treinamento E: jogar contra si mesmo ou memorizar os principais movimentos de jogadores experientes.

Projetando um sistema de aprendizagem

Uma dúvida que sempre paira sobre os desenvolvedores de soluções que utilizam aprendizado de máquina: como projetar um sistema de aprendizagem? Iremos apresentar a resposta com as etapas:

1. Escolher uma experiência de treinamento para o sistema.
2. Escolher uma função objetivo.
3. Escolher uma representação da função objetivo.
4. Escolher um algoritmo para a função de aproximação.

Escolher o tipo de experiência de treinamento pelo qual o sistema aprenderá e o grau de controle sobre a sequência de treinamento (supervisionado ou não supervisionado). Essa etapa representa a distribuição de exemplos usados no treinamento.

A função objetivo responde questões como: que tipo de conhecimento será aprendido? Como esse conhecimento será usado pelo sistema? No caso de um jogo de xadrez, o sistema precisa aprender como escolher o melhor movimento dentre todos os possíveis.

Para a representação da função objetivo, devemos escolher o programa de aprendizagem que será usado: redes neurais, funções polinomiais, coleções de regras, etc.

Dentre os diversos algoritmos disponíveis, devemos escolher aquele que melhor se adequa ao nosso problema.

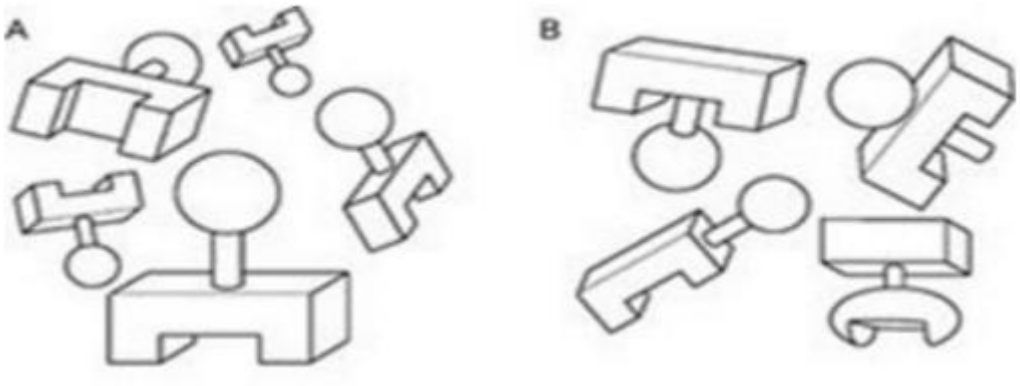
Pré-processamento e extração de características

A extração de características é uma tarefa fundamental no mundo do Aprendizado de Máquinas.

A característica é um atributo utilizado na descrição de um padrão, objeto ou situação. Exemplo: padrão pessoa (dados de uma instância) programadora, 1.60 cm de altura, feminina. Outro padrão: analista de testes, 1.73 de altura, masculino. A tarefa de extrair características é um processo que tem o objetivo de caracterizar objetos, verificando atributos que são similares para objetos da mesma classe e bastante diferentes dos objetos de outras classes.

As características devem ser discriminantes e invariantes, e os objetos devem ser reconhecidos independentemente de sua rotação, translação, escala etc. A Figura 1.9 apresenta um objeto que sofreu várias transformações, e mesmo assim ele precisará ser reconhecido como o mesmo objeto.

Figura 1.9 – Objeto que sofreu transformações.



Uma característica de um objeto é qualquer medida que se possa extrair desse objeto, seja simbólica (cor do objeto), numérica contínua (peso ou altura do objeto), numérica discreta (idade) ou numérica binária, que determina a presença ou ausência de uma característica. Exemplo: possui barba? Possui cabelos longos?

Uma outra tarefa importante é o pré-processamento, pois os dados podem estar incompletos, com ausência de algum valor importante, com ruídos (outliers) ou inconsistentes. Alguns algoritmos são mais indicados para determinado tipo de característica do que outros. O pré-processamento envolve a limpeza dos dados, que procura preencher dados ausentes, remover ruídos (valores absurdos) e resolver inconsistências. A integração e transformação de dados envolve integrar múltiplas bases e agregar dados externos, caso seja necessário.

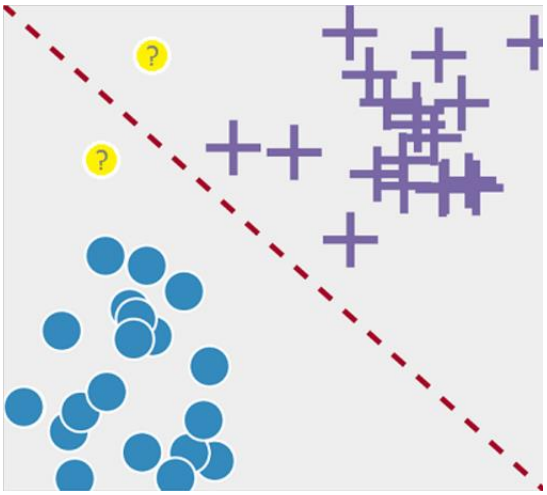
Capítulo 2. Técnicas de Aprendizado de Máquina

Ao longo da evolução da inteligência artificial, diversas técnicas foram criadas e aprimoradas, como classificação, regressão, agrupamento e outras. Esse capítulo tem o objetivo de apresentar e discutir as técnicas mais comuns utilizadas no Aprendizado de Máquina.

Classificação

Diante de uma base de dados que possui objetos que foram pré-classificados durante uma fase de treinamento, o objetivo da técnica de classificação é construir um modelo que seja capaz de classificar automaticamente novos objetos, baseado em suas características (atributos). Esse modelo é chamado de classificador. A Figura 2.1 apresenta uma visão dessa técnica. Observe que temos dois tipos de objetos, as bolinhas e as cruzinhas. Dados novos objetos, aqui representados por uma interrogação, o objetivo é analisar suas características e encaixá-los em um dos grupos.

Figura 2.1 – Representação de um classificador.



A maioria dos problemas com os quais nos deparamos em nosso cotidiano envolve algum tipo de classificação. Esse é o processo de identificar um modelo (ou função) que descreve e identificar a classe de um determinado objeto. O modelo

utilizado será derivado de uma base de dados de análise, chamada treinamento. Esse modelo é utilizado para predizer a classe de um conjunto de dados que ainda não foi classificado. Dessa forma, essa técnica utiliza um treinamento supervisionado.

A técnica de classificação é uma das mais utilizadas dentro do Aprendizado de Máquina, e suas aplicações são inúmeras. Algumas dessas aplicações envolvem a classificação de riscos (financeiros e outros), classificação de e-mails como SPAM, separação de imagens e diagnóstico de doenças baseado em sintomas.

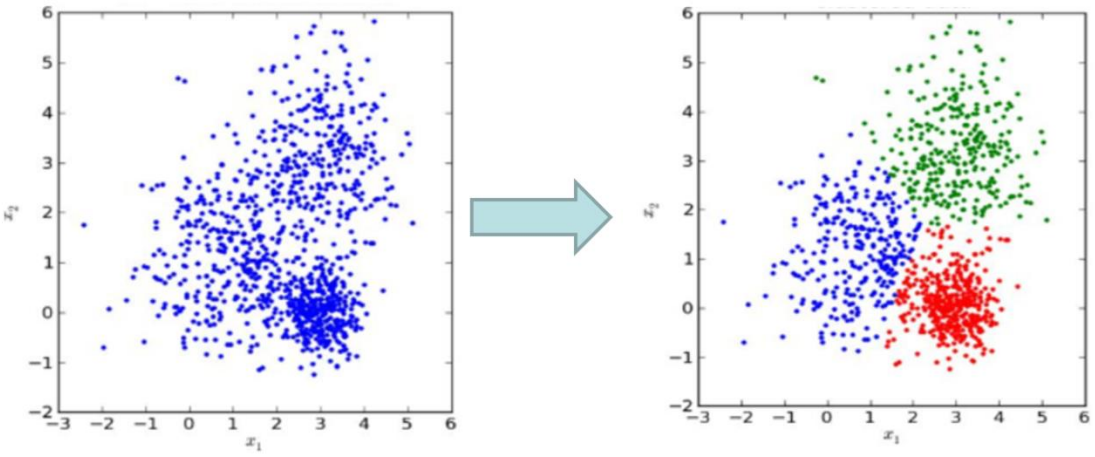
Diversos algoritmos podem ser aplicados nessa tarefa, tais como árvores de decisão, KNN, redes neurais, Naive Bayes e SVM.

Agrupamento (Clusters)

Agrupamento ou *clustering* é o nome dado para o grupo de técnicas computacionais cujo propósito consiste em separar objetos em grupos, baseando-se nas características que estes objetos possuem. A ideia básica consiste em colocar em um mesmo grupo objetos que sejam similares de acordo com algum critério pré-determinado. Essa divisão acontece de acordo com um tipo de relacionamento de similaridade e, nesse caso, as classes podem não ser conhecidas.

Cada um dos grupos que é criado pode ser exclusivo, e uma determinada instância pertence a apenas um cluster. Por uma abordagem probabilística, cada instância possui a probabilidade de pertencer a cada grupo. O algoritmo mais comum de agrupamento é K-médias. A Figura 2.2 apresenta um agrupamento realizado.

Figura 2.2 – Representação de um agrupamento.

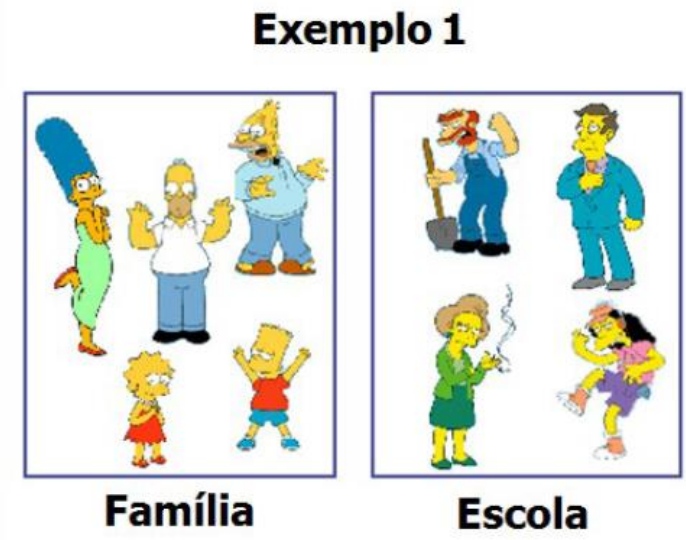


Dados não agrupados

Dados agrupados

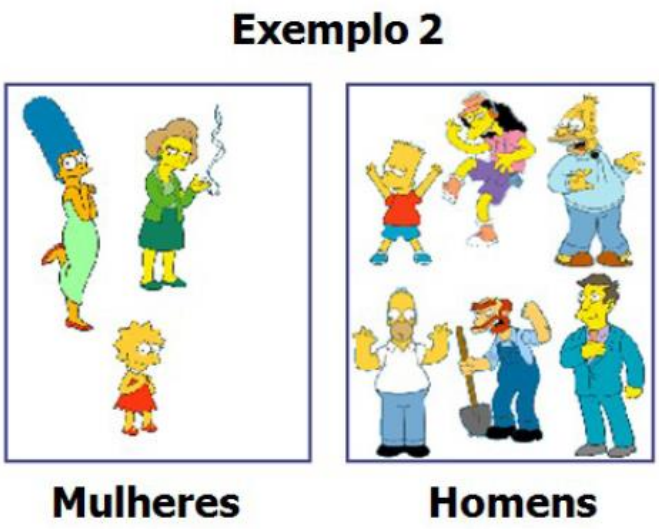
Dados os personagens do seriado The Simpsons representados na Figura 2.3, como poderíamos realizar um agrupamento com eles? A Figura 2.4 apresenta duas soluções possíveis, nas quais os personagens foram agrupados em dois grupos, família e escola.

Figura 2.3 – Primeiro agrupamento realizado.



Um outro exemplo que pode ser realizado com esses personagens é apresentado na Figura 2.4. Mais dois grupos foram criados, mulheres e homens.

Figura 2.4 – Segundo agrupamento realizado.



Diversos outros agrupamentos poderiam ser feitos, como por exemplo, crianças x adultos, inteligentes x não inteligentes, etc., bastando para isso que o algoritmo utilizado reconhecesse as similaridades entre os personagens e criasse um agrupamento para cada um deles.

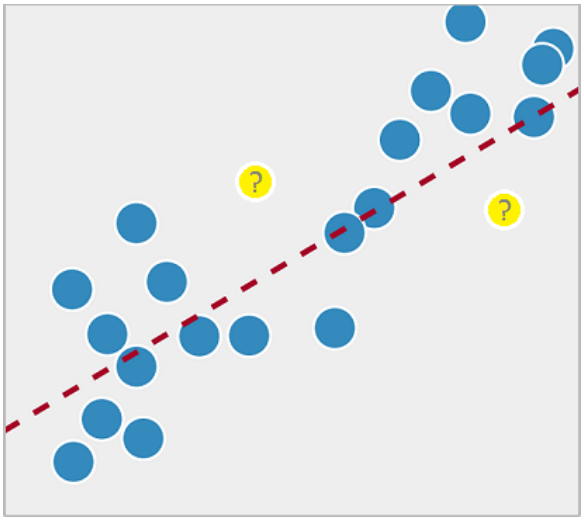
A técnica de agrupamento pode ser utilizada em diversas aplicação, tais como: identificação de clientes similares e ser mais assertivo no momento de oferecer ou recomendar um novo produto, agrupamento de pacientes com os mesmos sintomas, identificação de um público-alvo ou segmento de mercado para uma campanha de marketing, classificação de documentos e qualquer aplicação com grande quantidade de dados.

Regressão

A técnica de regressão é utilizada quando deseja-se criar um algoritmo de Aprendizado de Máquina que faça previsões de valores numéricos. Geralmente consiste em encontrar uma relação entre um conjunto de dados de entrada e uma saída numérica e os valores são contínuos.

Um exemplo de aplicação seria a necessidade de determinar o preço de venda de um automóvel. A Figura 2.5 apresenta esse exemplo. Para realizar essa tarefa, buscaríamos as características de vários carros, na Figura 2.5 representadas pelas bolinhas azuis, e os seus preços de venda, representados pela linha vermelha.

Figura 2.5 – Exemplo de regressão linear.



O objetivo é criarmos um algoritmo que seja capaz de estabelecer uma relação entre as características do carro e seu preço de venda. Ao final poderíamos determinar o preço de um novo carro (bolinhas amarelas na Figura 2.5) baseado em suas características.

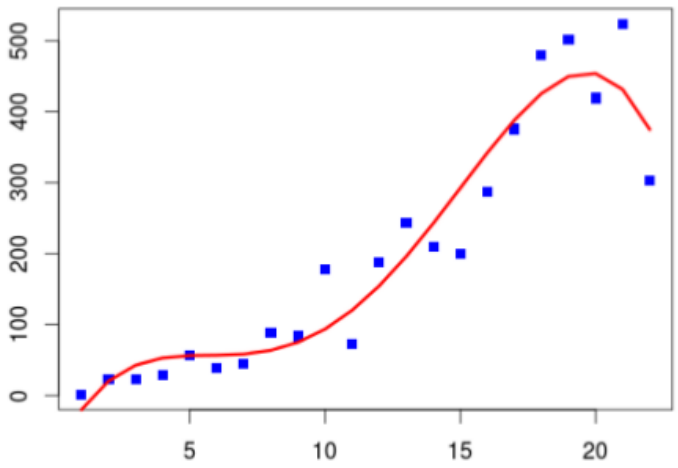
Muitas aplicações se beneficiam atualmente da técnica de regressão. Uma determinada rede de lojas de varejo pode desejar prever qual será a quantidade que certo produto irá vender num determinado mês com base nas vendas dos anos anteriores. Previsão do tempo e comportamento de bolsa de valores também são dois exemplos de utilização da regressão.

A regressão linear é uma equação usada para determinar o valor estimado de uma variável *y*, dados os valores de algumas variáveis *x*.

Na regressão não linear, os dados observacionais são modelados por uma função que é uma combinação não linear dos parâmetros do modelo e depende de uma ou mais variáveis independentes. Os dados são ajustados por um método que

realiza aproximações sucessivas. A Figura 2.6 apresenta um exemplo de regressão não linear.

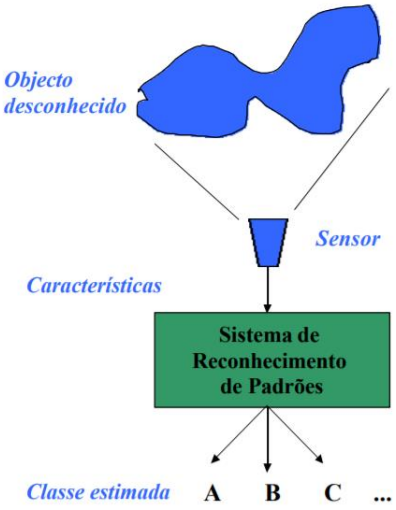
Figura 2.6 – Exemplo de regressão não linear.



Reconhecimento de padrões

O reconhecimento de padrões é uma área de pesquisa que tem o objetivo de classificar objetos (padrões) em um número de categorias ou classes. É o ato de observar os dados brutos e tomar uma ação baseada na categoria de um padrão. A Figura 2.7 apresenta o fluxo de trabalho (*workflow*) de uma aplicação da técnica de reconhecimento de padrões. Um objeto desconhecido é apresentado ao algoritmo, um sensor faz a extração de suas características e esse objeto é classificado em uma das classes possíveis.

Figura 2.7 – Exemplo do fluxo de trabalho de um sistema de reconhecimento de padrões.



Pode-se atribuir um padrão a um conjunto desconhecido de classes de padrões (clustering) ou identificar um padrão como membro de um conjunto de padrões conhecidos (classificação).

Atualmente temos diversas aplicações como identificação de suspeitos criminais por meio de impressão digital ou reconhecimento facial, reconhecimento de uma palavra construída em uma escrita cursiva, reconhecimento de caracteres, classificação de células em análises laboratoriais e contagem de partículas de matéria (células, bactérias, vírus, etc.). A Tabela 2.1 apresenta outras aplicações para a técnica de reconhecimento de padrões.

Tabela 2.1 – Outras aplicações utilizando a técnica de reconhecimento de padrões.

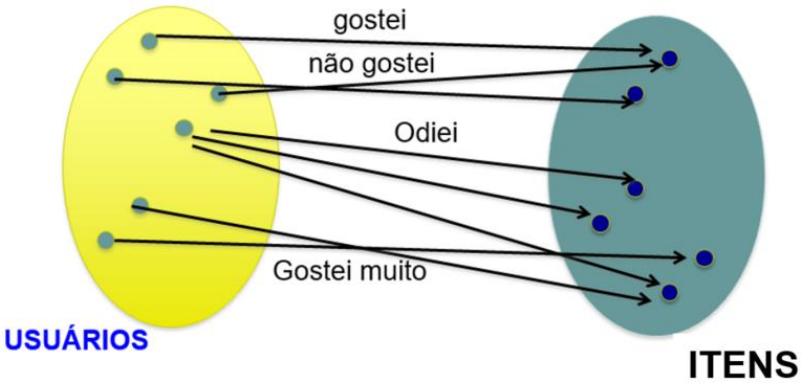
Aplicação	Padrão de Entrada	Classes (saída)
Reconhecimento óptico de caracteres	imagem de um documento	caracteres/palavras
Busca na internet	documento texto/imagem	categoria semântica
Filtro de e-mails	e-mail	spam/normal
Identificação de pessoas	face, iris, impressão digital	acesso de usuários credenciados
Diagnóstico auxiliado por computador	imagem microscópica	células saudáveis/doentes
Reconhecimento de alvos militares	imagem óptica ou infravermelho	tipo do alvo
Seleção automática de qualidade	imagem em esteira de produção	níveis de qualidade
Análise de sequências de DNA	sequência de DNA	gene conhecido/desconhecido

Sistemas de recomendação

Os sistemas de recomendação são aplicações que procuram auxiliar indivíduos a identificarem conteúdos de interesse em um conjunto de opções. São sistemas que procuram facilitar a atividade de busca por conteúdo interessante, auxiliando no processo de indicação já existente nas relações sociais. O processo de recomendação não é algo que existe somente no mundo digital. Esse processo já é feito muito antes da existência dos sistemas de informação.

A principal função desses sistemas é analisar os dados de um problema e extrair informações úteis para futuras predições. É muito utilizado em experiências de usuários. A Figura 2.8 apresenta a relação entre usuários e itens por meio de avaliações (gostei, não gostei etc.).

Figura 2.8 – Relação entre usuários e itens por meio de avaliações.



A rede de streaming Netflix faz recomendações com base na identificação de determinados parâmetros de produções já assistidas. Se você viu um filme de ação até o final, provavelmente você gosta dessa categoria de filmes. A recomendação baseada em um ator de uma produção já assistida pode não ser uma boa escolha, pois um ator pode atuar em diversos gêneros de filmes diferentes. O sistema de recomendação do Netflix utiliza técnicas de Deep Learning e leva em consideração o que o usuário faz, o tempo que passa vasculhando determinada categoria e quanto tempo o usuário passa na tela de descrição de um vídeo.

Um sistema de recomendação precisa de, basicamente, duas coisas para poder funcionar de maneira apropriada: (i) informações sobre as preferências dos usuários; (ii) um método para determinar se um dado item é interessante para um usuário.

Temos também dois tipos de sistemas de recomendação. Na recomendação explícita, o usuário indica espontaneamente o que é importante para ele, e na implícita o próprio sistema realiza ações baseadas em algumas características do usuário, como quando ele insere uma determinada página em seus favoritos, visualiza página por longo prazo, realiza movimentos e cliques com o mouse em determinadas áreas da página, usa a barra de rolagem e também por meio de análises de logs (pages visualizadas, frequência, localização física do usuário etc.).

A filtragem colaborativa é o processo de filtrar informação ou padrões usando técnicas que envolvem colaboração de múltiplos agentes, pontos de vista, fontes de

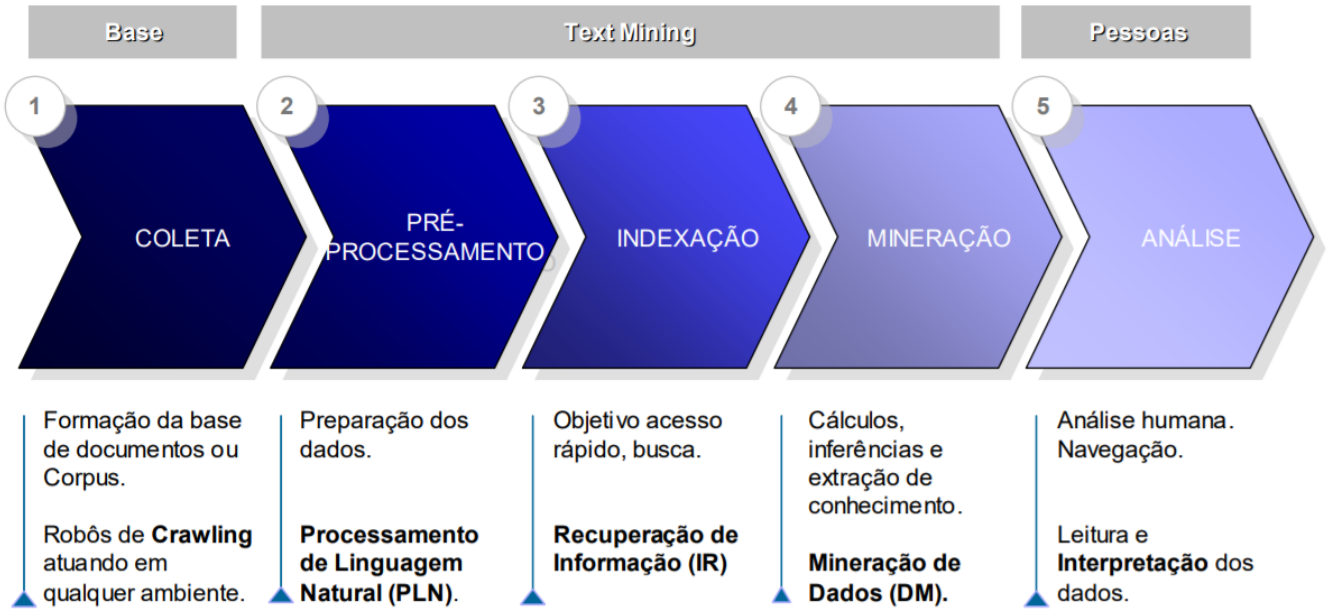
dados, etc. A técnica funciona a partir da construção de uma base de dados de preferências de usuários. Um novo usuário é comparado à essa base para encontrar seus vizinhos que possuem características similares.

Processamento e mineração de textos

O processamento de textos é uma evolução da área de recuperação de informação (R.I.). A mineração de textos (Text Mining) é um processo de descoberta do conhecimento que aplica algoritmos que processam textos e identificam informações úteis e implícitas. A técnica utiliza análise e extração de dados a partir de textos completos, frases e simples palavras.

Essas informações geralmente não podem ser recuperadas usando os métodos tradicionais de consultas disponíveis, pois geralmente são armazenadas no formato não estruturado. A técnica de processamento e mineração de textos contribui com busca específica em documentos, análise quantitativa e qualitativa de grandes volumes de dados e melhor compreensão de textos disponíveis em documentos. A Figura 2.9 apresenta o processo de mineração de textos com todas as suas etapas.

Figura 2.9 – Processo de mineração de textos.



Algumas aplicações que podem se beneficiar das técnicas de processamento e mineração de textos: coleta de dados da web (Web Crawler), análise de sentimento, inteligência competitiva, suporte e atendimento ao usuário, área do direito (textos jurídicos, relatórios policiais, depoimentos, etc.) e medicina (registro de prontuários não estruturados, pesquisa de doentes e doenças, etc.).

Análise de sentimentos

A técnica de análise de sentimentos também é conhecida como mineração de opinião. Diante do crescimento da quantidade de informações disponíveis na Web e do conteúdo disponibilizado pelas empresas, é necessária uma técnica que possa ser usada para entender atitudes, opiniões ou pensamentos expressos na web.

A análise de sentimentos visa identificar o sentimento que os usuários apresentam a respeito de uma entidade de interesse, como um produto, uma empresa, uma figura pública ou um lugar. O objetivo da técnica é permitir que um usuário obtenha um relatório contendo o que as pessoas andam dizendo sobre algo, de maneira compilada, ou seja, sem precisar encontrar e ler todas as informações. No final, é gerado um termômetro sobre determinado assunto, conforme apresentado na Figura 2.10.

Figura 2.10 – Termômetro gerado após a aplicação da técnica de análise de sentimento.

As empresas querem saber como anda a sua reputação perante seu cliente ou mesmo saber como está a aceitação de um novo produto que acabou de ser lançado. Outros interesses: bolsa de valores, esportes, política, artistas (audiência ou repercussão de algum fato) e doenças (epidemias).

Apesar da técnica de análise de sentimento estar amplamente difundida, alguns desafios ainda existem: São eles:

- Textos com erros. Sentenças sintaticamente mal formatadas.
- Distinguir se um texto é opinião ou fato. Fatos com opiniões embutidas.
- Textos com sarcasmos e ironias.
- Uso de pronomes para referenciar itens.
- Uso de termos informais da internet. Ex.: “blz”, “fds”.
- Uso de memes.
- Uso de emoticons.

O processo de análise de sentimento envolve algumas etapas: (i) coleta do conteúdo (obtenção dos dados); (ii) classificação (opinião positiva, negativa ou neutra); (iii) sumarização dos resultados.

Métodos ensembles

Em nossas vidas, quando tomamos uma decisão, procuramos uma segunda, terceira ou até uma quarta opinião, seja de parentes, amigos ou na própria internet. Essas decisões podem ser dos mais diversos assuntos, tais como: assuntos médicos, financeiros, sociais, investimentos, etc. Nesse processo de tomada de decisão, nós atribuímos inconscientemente pesos para cada uma das opiniões. Com uma combinação dessas opiniões e pesos, espera-se alcançar a opinião mais bem-informada de todas. Realizar a consulta de outros “especialistas” é algo presente

sempre no comportamento humano. Somente nos últimos tempos esse processo foi descoberto pela inteligência computacional, surgindo aí os métodos ensembles.

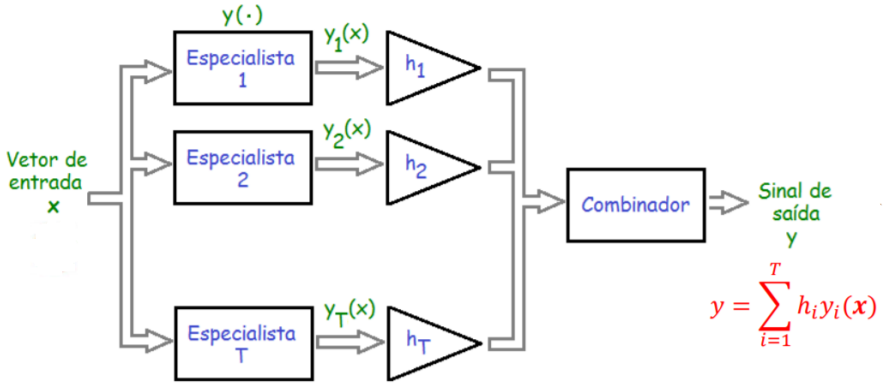
Essa técnica tem apresentado bons resultados ao comparar o resultado de seu processamento com o resultado do processamento de um único especialista. As razões disso estão aqui listadas:

- Um bom desempenho no conjunto de treinamento não garante um bom desempenho na generalização.
- Não seguir a recomendação de um único especialista.
- O volume de dados a ser analisado pode ser muito grande para um único classificador.

A ideia aqui é que o sistema adote a técnica de dividir para conquistar. O espaço de dados é dividido em porções menores e mais fáceis de serem aprendidas por diferentes classificadores. A linha base de fronteira de decisão pode ser aproximada por meio de uma diferente combinação de classificadores, alcançando uma fusão dos dados.

A Figura 2.11 apresenta o processo realizado pelos métodos Ensembles com T classificadores. Observe que dado um vetor de entrada, os dados deste são enviados para cada um dos classificadores. Em seguida, os dados de saída desses classificadores são enviados para um combinador que finalmente dará a classificação final baseada em um conjunto de pesos, que torna a opinião de um classificador mais influente que a de outro.

Figura 2.11 – Método ensemble.



O método ensembles é uma alternativa para a criação de classificadores mais precisos. Deve-se analisar cuidadosamente o desempenho da solução, pois se temos mais classificadores é possível que tenhamos um gasto maior de tempo que pode ser compensado por classificações mais precisas.

Capítulo 3. Redes neurais

Este capítulo tem o objetivo de fazer uma introdução às Redes Neurais Artificiais (RNA). O funcionamento dessas redes será discutido, assim como a estrutura e o comportamento de um neurônio artificial, que é a base para a construção dessas redes complexas. Além disso, iremos discutir os tipos de redes existentes, apresentando exemplos e comparando cada um deles.

Por fim, iremos apresentar o conceito de Deep Learning por meio de exemplos de funcionamento e utilização, além das principais aplicações que utilizam essa tecnologia.

Introdução às redes neurais artificiais

As redes neurais artificiais surgiram na década de 40, mais precisamente em 1943, quando o neurofisiologista Warren McCulloch e o matemático Walter Pitts, da Universidade de Illinois, fizeram uma analogia entre as células nervosas e o processo eletrônico num artigo publicado no *Bulletin of Mathematical Biophysics* com o título: *A Logical Calculus of the Ideas Immanent in Nervous Activity*. Em 1949, o biólogo e psicólogo Donald Hebb, que estudava o comportamento dos animais, escreveu um livro chamado *The Organization of Behavior*, que reforçava as teorias de que o condicionamento psicológico estava presente em qualquer parte dos animais, pelo fato de que esta é uma propriedade de neurônios individuais. As ideias de Hebb não eram pioneiras, mas ele propôs um princípio de aprendizado em sistemas nervosos complexos, ou seja, uma lei que descreve o funcionamento quantitativo da sinapse e do processo de treinamento humano. Desde então, vários outros pesquisadores, entusiasmados com as novas descobertas, voltaram-se para esta linha de pesquisa.

Em 1951, Marvin Minsky, cofundador do laboratório de inteligência artificial do MIT, construiu o SNARC, o primeiro simulador de cadeia neural. O SNARC trabalhava com êxito e podia ajustar seus pesos sinápticos automaticamente. Ele nunca chegou a executar alguma função de processamento de informação

interessante, servindo somente de fator motivador para ideias que surgiram posteriormente.

Em 1956, na Primeira Conferência Internacional de Inteligência Artificial, foi apresentado um modelo de rede neural artificial pelo pesquisador da IBM Nathaniel Rochester. Seu modelo consistia numa simulação de centenas de neurônios interconectados através de um sistema que verificaria como a rede responderia aos estímulos ambientais.

Já em 1959, Frank Rosenblatt, na Universidade de Cornell, criou uma rede de múltiplos neurônios do tipo discriminadores lineares e a batizou de rede Perceptron. Rosenblatt baseou-se nas linhas de pensamento de McCulloch para desenvolver o seu modelo matemático de sinapse humana. Devido às suas complexas pesquisas e inúmeras contribuições técnicas, muitos o consideram como fundador da neurocomputação.

No final da década de 50, Minsky e Seymour Papert lançaram, em uma obra chamada “Perceptron”, uma demonstração de que o modelo apresentado por Rosenblatt não era muito promissor, devido ao uso de técnicas empíricas, das grandes dificuldades da matemática envolvida e dos poucos recursos computacionais disponíveis na época. A publicação de Minsky e Papert acabou esfriando as pesquisas e praticamente todo o investimento financeiro nesta área foi cancelado.

Enquanto Rosenblatt trabalhava no Perceptron, Bernard Widrow, da Universidade de Stanford, com a ajuda de alguns estudantes, desenvolveu um novo modelo de processamento de redes neurais chamado de Adaline (ADAptive LINear Element), a qual se destacava pela sua poderosa lei de aprendizado. O princípio de treinamento para as redes Adalines ficou conhecido como a Regra Delta, que foi mais tarde generalizada para redes com modelos neurais mais sofisticados. Mais tarde, Widrow criou a Madaline, que era uma generalização multidimensional do Adaline.

Nos anos seguintes, muitos artigos foram publicados e várias previsões exageradas e pouco confiáveis para a época foram anunciadas. A maioria destas suposições falava de computadores com um poder de raciocínio e/ou processamento

igual ou superior ao do cérebro humano. Desta forma, a credibilidade de futuros estudos das RNAs foi fortemente comprometida.

No início da década de 80, muitos pesquisadores publicaram inúmeras propostas para a exploração de desenvolvimento e pesquisa em redes neurais. Foi quando o administrador de programas da DARPA (Defense Advanced Research Projects Agency), Ira Skurnick, resolveu dar atenção às proposições da neurocomputação, contrariando todos os preceitos e fundando, em 1983, as pesquisas em neurocomputação da DARPA. Este fato acabou abrindo novos horizontes para a neurocomputação.

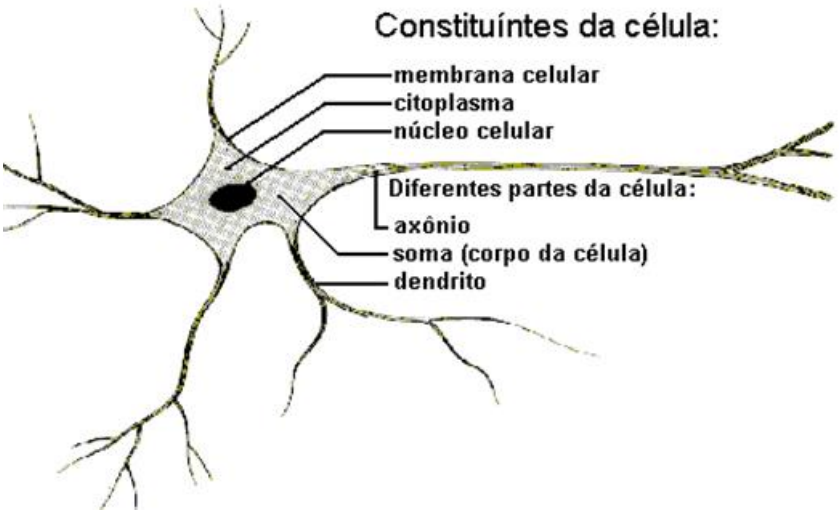
O físico e biólogo de reputação mundial John Hopfield também se interessou pela neurocomputação e escreveu vários artigos em 1982 que levaram muitos cientistas a se unirem nesta nova área emergente. Hopfield reaqueceu as pesquisas em neurocomputação, criticando fortemente as teorias apresentadas por Minsky e Papert na década de 50.

Este campo de pesquisa explodiu mesmo em 1986, quando o professor de psicologia da Universidade de Stanford, David E. Rumelhart, e seu colega James L. McClelland, professor de psicologia da Universidade de CarnegieMellon, publicaram o livro *Parallel Distributed Processing: Explorations in the Microstructure of Cognition (vol.1: Foundations, vol.2: Psychological and Biological Models)*. Nesse livro, eles apresentam um modelo matemático e computacional que propicia o treinamento supervisionado dos neurônios artificiais. Surgia, então, o *Backpropagation*, um algoritmo de otimização global sem restrições.

Em 1987 ocorreu a Primeira Conferência de Redes Neurais. Também foi formada a Sociedade Internacional de Redes Neurais (*International Neural Networks Society - INNS*) juntamente com o *INNS Journal* em 1989, do Neural Computation e do IEEE Transactions on Neural Networks em 1990. A partir desses acontecimentos, muitas instituições formaram institutos de pesquisa e programas de educação em neurocomputação.

O cérebro de um ser humano é constituído por bilhões de neurônios, e um neurônio caracteriza-se por uma célula formada por três partes específicas e complementares: corpo, dendritos e axônio. Os dendritos capturam os estímulos recebidos em determinado período de tempo e os transmitem ao corpo do neurônio. No momento em que os estímulos atingirem um limite determinado, o corpo da célula envia um novo impulso que irá se propagar pelo axônio e será transmitido às células vizinhas por meio de outra função, que são as sinapses. O processo descrito poderá ser repetido por diversas camadas de neurônios, e como resultado, a informação de entrada será processada, podendo obter como resultado a condução do cérebro a comandar reações físicas. A Figura 3.1 ilustra a divisão de um neurônio de forma simplificada.

Figura 3.1 – Representação simplificada de um neurônio.



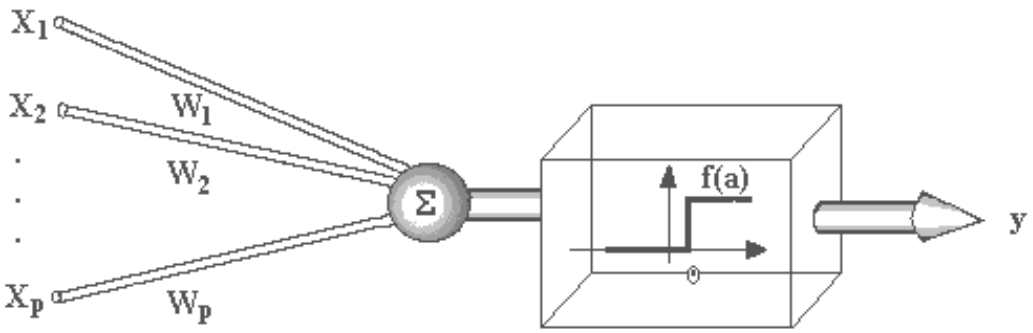
A habilidade que um ser humano tem de realizar funções complexas e de aprender surge da forma com que o seu cérebro realiza o processamento, ou seja, de forma paralela e distribuída. Na camada externa do cérebro temos o córtex, e os neurônios que lá estão são os responsáveis pelo processamento cognitivo. A partir de uma nova experiência pessoal ou mesmo de um novo conhecimento, podemos apresentar alterações estruturais em nosso cérebro. Tais alterações são efetivadas por meio de um reposicionamento das redes de neurônios, e isso pode inibir ou mesmo reforçar algumas sinapses.

As redes neurais artificiais (RNAs) são um conjunto de técnicas que tentam simular, em meio computacional, o funcionamento do cérebro humano. Elas são capazes de reconhecer padrões, extrair regularidades e detectar relações subjacentes em um conjunto de dados aparentemente desconexos.

As redes neurais artificiais (RNAs) são modelos matemáticos que se assemelham às estruturas neurais biológicas e que têm capacidade computacional adquirida por meio de aprendizado e generalização. O aprendizado em RNAs está normalmente associado à capacidade que elas possuem de adaptarem seus parâmetros como consequência de sua interação com o meio externo. Os critérios de desempenho que podem determinar a qualidade do modelo neural e o ponto de parada dos treinamentos são determinados por meio dos parâmetros utilizados no treinamento da rede neural. A generalização de uma rede está relacionada à capacidade que essa rede tem de promover respostas coerentes para dados utilizados como entrada em seu treinamento.

As informações em RNAs fluem através de estruturas neurais artificiais, onde o processamento da informação é feito de forma paralela e distribuída. Cada elemento que realiza esse processamento é representado por uma estrutura denominada neurônio artificial. A Figura 3.2 apresenta um esquema do neurônio artificial.

Figura 3.2 – Representação de um neurônio artificial.



As entradas do neurônio artificial correspondem ao vetor de entrada denominado $X = [x_1, x_2, \ldots, x_p]^T$ de dimensão p. Para cada uma das entradas x_i, há um peso correspondente w_i na entrada do neurônio. A soma dessas entradas x_i

ponderadas pelos pesos correspondentes w_i é chamada de saída linear u, onde $u = P_i w_i x_i$. A saída y do neurônio é denominada saída de ativação e é obtida por meio da aplicação da função de ativação $f(:::)$ à saída linear u, $y = f(u)$. A função de ativação $f(:::)$ pode assumir várias formas geralmente não lineares. Um exemplo típico de função de ativação é a função de grau unipolar, também chamada de limiar, aqui representada na Figura 3.3.

Figura 3.3 – A função de ativação de um neurônio.

$$f(u) = \begin{cases} 0 & u < \theta \\ 1 & u \geq \theta \end{cases}$$

Uma rede neural artificial é, em suma, constituída por elementos processadores, e cada elemento processador executa uma função relativamente simples, porém a rede neural artificial como um todo tem capacidade computacional para a resolução de problemas complexos

O aprendizado é definido como o processo pelo qual os parâmetros de uma rede neural são ajustados, por meio de uma sequência ordenada de estímulos aplicados no ambiente no qual a rede está sendo aplicada. As redes neurais artificiais têm a capacidade de aprendizado por meio de exemplos e também por meio de extrapolações e interpolações em cima do que elas aprenderam. O algoritmo de aprendizagem de uma determinada rede neural trata do conjunto de procedimentos definidos para adaptação dos parâmetros de uma RNA, para que ela tenha a capacidade de aprender uma função específica.

Atualmente podemos encontrar na literatura inúmeros métodos para treinamento de Redes Neurais Artificiais, sendo estes classificados como aprendizado supervisionado e aprendizado não supervisionado.

Tipos de redes

Um dos tipos de redes neurais mais simples que temos é a Perceptron de uma camada. Essa rede de única camada é utilizada para dividir duas classes linearmente separáveis (Figura 3.4), apresentando erro para classes não linearmente separáveis (Figura 3.5), onde se recomenda usar a rede Perceptron multicamadas, que falaremos a respeito posteriormente.

Figura 3.4 – Sistema linearmente separável.

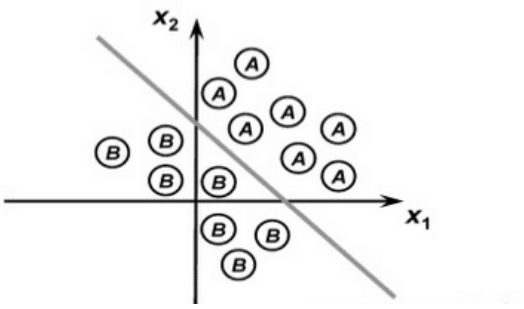
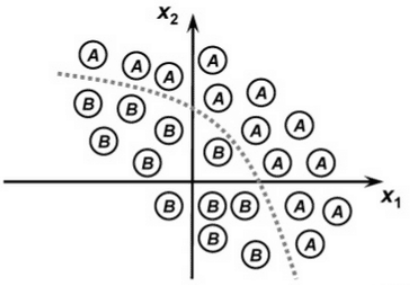


Figura 3.5 – Sistema não linearmente separável.



O funcionamento da rede Perceptron é muito simples, as entradas (X_i) representam as informações do processo que desejamos mapear, sendo que cada uma das entradas terá um peso sináptico ponderado (W_i) que representa a importância de cada entrada em relação ao valor de saída desejado (y). O resultado da somatória das entradas ponderadas será somado ao limiar de ativação (θ) e então repassado como argumento da função de ativação g(.), a qual terá como resultado a saída desejada. Normalmente, a função de ativação costuma ser do tipo função degrau ou degrau bipolar. Representando matematicamente teremos:

$$y = g(u)$$

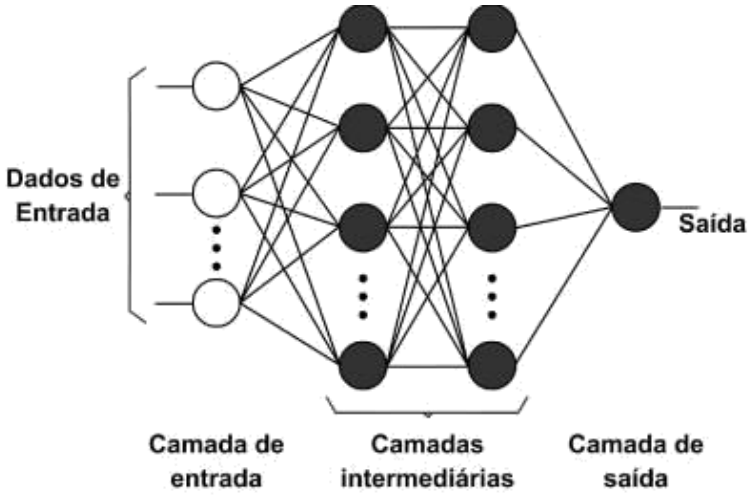
$$u = \sum_{i=1}^{n} wi.Xi - \theta$$

Onde:

- Xi: Entradas: valor real ou binário.
- Wi: Pesos sinápticos: valor real aleatório.
- Θ: Limiar de ativação: valor real aleatório.
- y: Saída: valor binário.
- g(.): Função de ativação.

As redes de múltiplas camadas representam uma importante classe das redes neurais artificiais. Basicamente, as MLPs consistem de um conjunto de unidades sensoriais (nós fonte) que representam a camada de entrada da rede. Além disso, essas redes possuem uma ou mais camadas de nós computacionais (camadas ocultas) e uma camada de saída. O sinal de entrada se propaga para frente dentro da rede, camada por camada. Essas redes representam uma generalização do Perceptron de camada única. A Figura 3.6 mostra o grafo arquitetural de um Perceptron de múltiplas camadas com duas camadas ocultas e uma camada de saída, sendo dada como exemplo uma rede totalmente conectada. Isso significa que um neurônio em qualquer camada da rede está conectado a todos os outros neurônios da camada anterior. Assim, o sinal da rede avança de trás para frente (esquerda para direita) e de camada em camada.

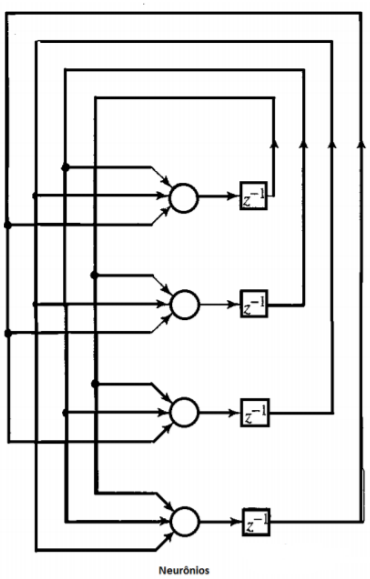
Figura 3.6 – Grafo arquitetural de um Perceptron de múltiplas camadas com duas camadas ocultas.



As redes do tipo MLP (Multilayer Perceptron) possuem uma capacidade computacional maior do que as redes que não possuem camadas intermediárias, pois elas podem tratar dados que não são linearmente separáveis. Os problemas não linearmente separáveis podem ser resolvidos por meio das redes com uma ou mais camadas ocultas. A alteração da arquitetura dessa rede, como a inclusão de mais neurônios ou de mais camadas ocultas, não parece ser um problema a princípio, pois o principal agravante passa a ser o algoritmo para treinamento. Essa questão causou uma queda na quantidade de pesquisas realizadas com MLPs na década de 1970. A determinação da quantidade de camadas a ser utilizada pode influenciar drasticamente no aprendizado da rede. O uso de uma grande quantidade de camadas ocultas não é recomendado, visto que o erro ocorrido em uma camada é propagado a outras camadas da rede. A quantidade de neurônios que pertence a camadas ocultas é definida de forma empírica e normalmente depende da distribuição dos padrões de treinamento e validação da rede. Um uso excessivo de neurônios pode levar a rede a decorar o conjunto de treinamento, ao invés de atuar na generalização (extração de características gerais). Uma quantidade relativamente pequena de neurônios poderá levar a rede a aumentar o tempo de treinamento, e com isso dificultar a ótima representação do problema proposto.

As redes neurais de Hopfield (HNN, do inglês Hopfield Neural Networks) são redes neurais artificiais que apresentam realimentação e foram desenvolvidas por John Hopfield. O trabalho de Hopfield – publicado em 1982 – ajudou no ressurgimento do interesse pela área de redes neurais que passava por um período de crise após a publicação de Minsky e Papert na qual mostrava que o Perceptron só resolvia problemas linearmente separáveis. As redes de Hopfield podem ser utilizadas como memórias associativas, nas quais a rede é capaz de armazenar informações baseadas em alguns de seus estados. Além disso, elas podem resolver problemas de otimização combinatória. Neste caso, a rede de Hopfield possui uma função de energia que provê uma medida de desempenho para o problema de otimização que possui um conjunto grande, mas finito, de possíveis soluções. A Figura 3.7 apresenta a topologia de uma rede de Hopfield.

Figura 3.7 – Topologia de uma rede de Hopfield.



O modelo de Hopfield consiste em um conjunto de neurônios e um conjunto correspondente de atrasos unitários, formando um sistema realimentado de múltiplos laços. Como ilustrado na Figura 3.7, a saída de cada neurônio é realimentada, através de um elemento de atraso unitário, para cada um dos outros neurônios da rede (não há autorrealimentação).

Existem diversos outros tipos de Redes Neurais e também a combinação de uso entre mais de um tipo.

Deep Learning

Entre as diversas variações de redes neurais artificiais, uma das que mais ganhou destaque é a rede neural convolucional, que é o conjunto de múltiplas tarefas de aprendizado de máquina que lida com diferentes tipos de abstração, sendo que sua característica mais marcante e a resolução de problemas a partir de estruturas de representação dos dados de diferentes maneiras.

As redes neurais convolucionais (RNCs) foram inspiradas no sistema de visão humana. Elas recebem como entrada pequenas porções de imagens, na camada mais baixa da estrutura hierárquica. A arquitetura mais complexa permite estabilidade mesmo com alterações em relação a escala, ou rotação. A rede é composta basicamente de neurônios, que possuem peso e inclinações treináveis, e acesso a recursos básicos, como por exemplo, bordas.

Aprendizado de máquina é uma área de inteligência artificial, cujo objetivo é o desenvolvimento de técnicas computacionais sobre o aprendizado, construindo sistemas capazes de adquirir conhecimento de forma automática. Através das experiências acumuladas, esses sistemas de aprendizado conseguem encontrar soluções mais eficientes para determinados problemas. A área de aprendizado de máquina desenvolveu-se rapidamente através dos estudos de reconhecimento de padrões, transformando-se na base da análise preditiva. Assim, sistemas de aprendizado de máquinas são utilizados para identificar objetos em imagens, classificar pessoas em redes sociais, transcrever voz em texto, recomendar produtos a usuários em sites de comércio eletrônico, entre várias outras aplicações. Cada vez mais, estas aplicações fazem uso de um conjunto de técnicas chamado de Deep Learning.

A técnica de Deep Learning é também conhecida como aprendizado profundo ou aprendizado hierárquico e é o ramo do Machine Learning baseado em um conjunto

de algoritmos que tentam modelar abstrações de alto nível de dados usando um grafo profundo com várias camadas de processamento. Além disso, essa técnica permite lidar com problemas que envolvem transformações lineares e não lineares.

Dentre as técnicas mais implementadas de Deep Learning, vale mencionar uma: convolutional neural networks. Esse método teve um grande aumento em sua popularidade, em especial para processamento de imagens, devido à qualidade dos resultados que têm sido publicados com essa técnica. Para detecção de caracteres escritos à mão e leitura de texto, por exemplo, essa é a tecnologia *state-of-the-art*. As técnicas de Deep Learning são aplicadas em diversas áreas, tais como: reconhecimento facial, classificação de doenças, carros autônomos, visão computacional, processamento de linguagem natural, dentre outras.

Capítulo 4. Algoritmos de Machine Learning

Existem atualmente na literatura diversos algoritmos que visam auxiliar na implementação das principais técnicas de Machine Learning, tais como: regressão, classificação, agrupamentos, etc. O objetivo deste capítulo é apresentar o funcionamento e a aplicação dos principais algoritmos de Aprendizado de Máquina.

Árvores de decisão

Grande parte das aplicações de relevância prática em inteligência artificial está baseada na concepção de modelos computacionais do conhecimento empregado por um especialista humano. Na síntese de modelos de classificação, a associação entre as classes e o conjunto de atributos que caracterizam os objetos a serem classificados pode se dar de formas variadas, empregando processamento simbólico e/ou numérico. A construção dos modelos computacionais de classificação geralmente emprega um dentre dois paradigmas alternativos.

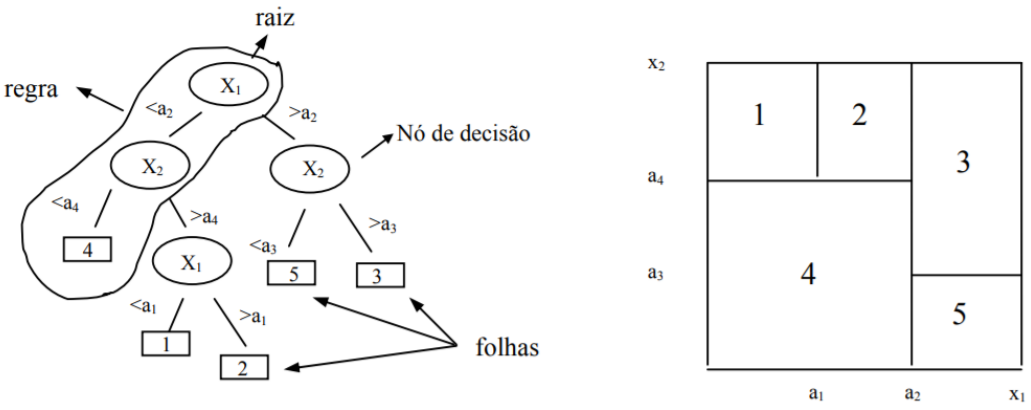
- **Top-down:** obtenção do modelo de classificação a partir de informações fornecidas por especialistas;
- **Bottom-up:** obtenção do modelo de classificação pela identificação de relacionamentos entre variáveis dependentes e independentes em bases de dados rotuladas. O classificador é induzido por mecanismos de generalização fundamentados em exemplos específicos (conjunto finito de objetos rotulados). Existem propostas também para dados não rotulados.

Árvores de decisão são modelos estatísticos que utilizam um treinamento supervisionado para a classificação e previsão de dados. Em outras palavras, em sua construção é utilizado um conjunto de treinamento formado por entradas e saídas. Estas últimas são as classes. Estes modelos utilizam a estratégia de dividir para conquistar: um problema complexo é decomposto em subproblemas mais simples e recursivamente esta técnica é aplicada a cada subproblema. As árvores de decisão estão entre os mais populares algoritmos de inferência e tem sido aplicadas em várias

áreas como, por exemplo, diagnóstico médico e risco de crédito, e delas pode-se extrair regras do tipo “se-então” que são facilmente compreendidas. A capacidade de discriminação de uma árvore vem da divisão do espaço definido pelos atributos em subespaços e a cada subespaço é associada uma classe.

A Figura 4.1 abaixo representa uma árvore de decisão onde cada nó de decisão contém um teste para algum atributo, cada ramo descendente corresponde a um possível valor deste atributo, o conjunto de ramos é distinto, cada folha está associada a uma classe, e cada percurso da árvore, da raiz à folha, corresponde a uma regra de classificação. No espaço definido pelos atributos, cada folha corresponde a um hiper-retângulo onde a interseção destes é vazia e a união é todo o espaço.

Figura 4.1 – Representação de uma árvore de decisão no espaço.



Entropia é o cálculo do ganho de informação baseado em uma medida utilizada na teoria da informação. A entropia caracteriza a (im)pureza dos dados: em um conjunto de dados, é uma medida da falta de homogeneidade dos dados de entrada em relação a sua classificação. Por exemplo, a entropia é máxima (igual a 1) quando o conjunto de dados é heterogêneo. Dado um conjunto de entrada (S) que pode ter c classes distintas, a entropia de S será dada por:

$$\text{Entropia}(S) = - p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

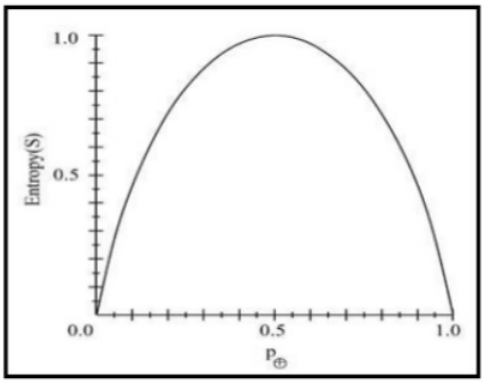
Onde:

P+: proporção de exemplos positivos em S.

P-: proporção de exemplos negativos em S.

A função de Entropia com proporção P+ de exemplos positivos varia de 0 até 1. A Figura 4.2 apresenta uma função de Entropia.

Figura 4.2 – Função Entropia.



Um exemplo de aplicação do algoritmo de árvore de decisão é a decisão por realizar ou não um jogo de tênis. A árvore criada irá auxiliar na decisão de jogar ou não a partida. A Figura 4.3 apresenta o conjunto de atributos selecionados:

Figura 4.3 – Conjunto de atributos selecionados.

Conjunto de Atributos
Tempo
Temperatura
Umidade
Vento

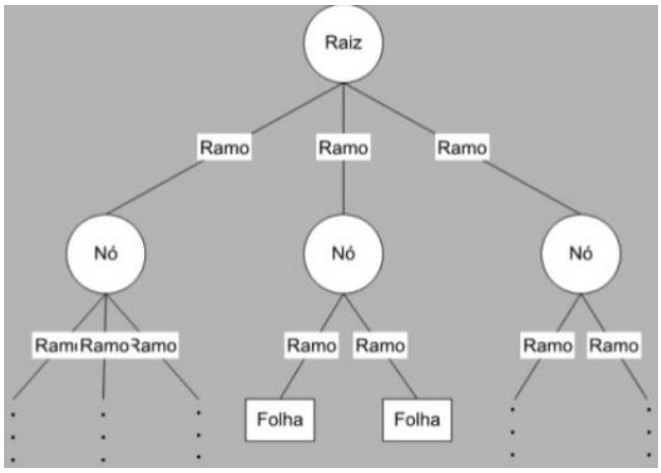
A Figura 4.4 apresenta a base de treino para o algoritmo árvore de decisão. A base contempla os conjuntos de atributos e a decisão pelo jogo de tênis (sim ou não).

Figura 4.4 – Base de treino para o algoritmo de árvore de decisão.

Tempo	Temperatura	Umidade	Vento	Joga
Sol	Alta	Media	Não	Não
Sol	Alta	Alta	Sim	Não
Nublado	Alta	Alta	Não	Sim
Chuva	Baixa	Alta	Não	Sim
Chuva	Baixa	Media	Não	Sim
Chuva	Baixa	Baixa	Sim	Não
Nublado	Baixa	Baixa	Sim	Sim
Sol	Media	Alta	Não	Não
Sol	Baixa	Baixa	Não	Sim
Chuva	Media	Media	Não	Sim
Sol	Media	Baixa	Sim	Sim
Nublado	Media	Alta	Sim	Sim
Nublado	Alta	Baixa	Não	Sim
Chuva	Baixa	Alta	Sim	Não

Em uma árvore de decisão, cada nó contém um teste em um atributo, cada ramo corresponde a um possível valor desse atributo, cada nó-folha está associado a uma classe e cada percurso na árvore (raiz até a folha) corresponde a uma regra de classificação. A Figura 4.5 apresenta uma árvore de decisão genérica.

Figura 4.5 – Árvore de decisão genérica.



As árvores de decisão são aplicadas em diversos tipos de problemas, principalmente naqueles que são muito bem realizados por um ser humano especialista, como por exemplo análise de crédito, decisões de compra e venda, diagnóstico de problemas em equipamentos e diagnóstico médico.

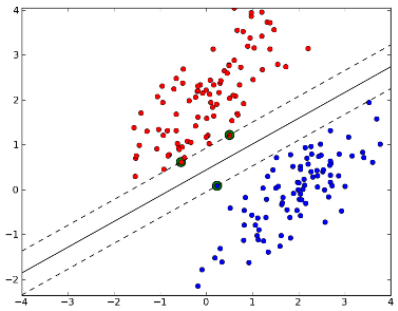
Os principais algoritmos de árvores de decisão são: ID3 (recursivo e baseado em busca exaustiva) e C4.5 (usa a técnica dividir para conquistar).

SVM – Support Vector Machine

O algoritmo SVM (Support Vector Machine) foi proposto pelo cientista russo Vladimir Vapnik. Trata-se de um método de aprendizado que tenta encontrar a maior margem para separar duas classes diferentes de dados em um espaço de amostragem. O SVM pertence à classe dos algoritmos de aprendizado supervisionado e, devido à divisão dos dados em duas classes, é conhecido como uma classificação binária.

A Figura 4.6 apresenta um exemplo de comportamento do algoritmo SVM na classificação dos dados em duas classes.

Figura 4.6 – Separação em duas classes do algoritmo SVM.



Uma das aplicações do algoritmo SVM é na tarefa de reconhecimento de padrões, que vem sendo utilizada com sucesso em problemas de categorização de textos. É uma abordagem geométrica para o problema de classificação onde cada contexto do conjunto de treinamento pode ser visto como um ponto x_i em um espaço R^M. O aprendizado consiste em dividir os elementos positivos dos negativos nesse espaço Euclidiano.

Algumas das principais características do algoritmo SVM que torna atrativo o seu uso:

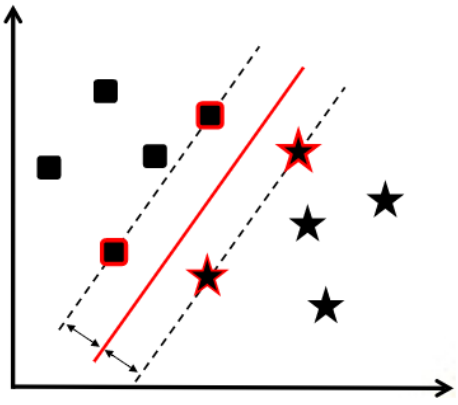
- **Boa capacidade de generalização:** os classificadores gerados por uma SVM em geral alcançam bons resultados de generalização. A capacidade de generalização de um classificador é medida por sua eficiência na classificação de dados que não pertençam ao conjunto utilizado em seu treinamento. Na geração de preditores por SVMs, portanto, é evitado o overfitting, situação na qual o preditor se torna muito especializado no conjunto de treinamento, obtendo baixo desempenho quando confrontado com novos padrões.

- **Robustez em grandes dimensões**: as SVMs são robustas diante de objetos de grandes dimensões, como, por exemplo, imagens. Comumente há ocorrência de overfitting nos classificadores gerados por outros métodos inteligentes sobre esses tipos de dados.

Entre as características citadas, o destaque das SVMs está em sua capacidade de generalização. Estes resultados foram apresentados por Vapnik e Chernovenkis através da Teoria de Aprendizado Estatístico, proposta por estes autores nas décadas de 60 e 70. As SVMs surgiram como resultado direto do emprego dos princípios apresentados nestes estudos. Apesar de sua teoria ser relativamente antiga, as primeiras aplicações práticas das SVMs são recentes e datam da década de 90.

Durante a execução do algoritmo SVM, diversas retas podem ser traçadas para a separação dos dados, conforme apresentado na Figura 4.7. O hiperplano ótimo é a reta mais distante dos objetos das duas classes.

Figura 4.7 – Separação entre classes feita pelos algoritmos SVM.



Um método puramente linear para classificar um conjunto pode sofrer dois problemas comuns: outliers e exemplos rotulados erroneamente.

O SVM foi originalmente concebido para lidar com classificação binária, entretanto a maioria dos problemas reais requerem múltiplas classes. Para utilizar o SVM em múltiplas classes é preciso transformar o problema em vários problemas de classes binárias.

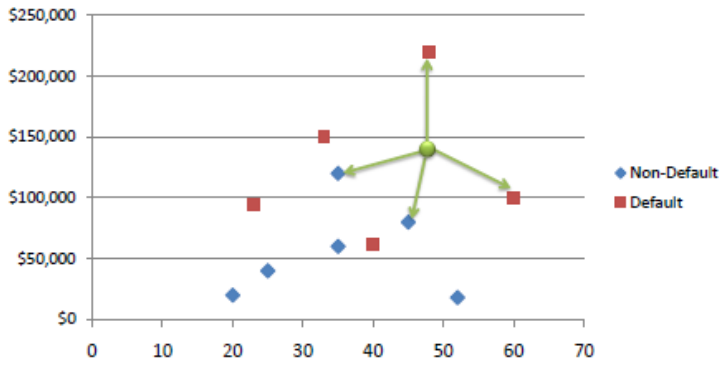
KNN – K-Nearest Neighbours

O algoritmo KNN (K-Nearest Neighbor) é um algoritmo que utiliza aprendizado supervisionado e é considerado um dos algoritmos de classificação mais simples. Como o algoritmo utiliza o aprendizado supervisionado, usamos para classificar objetos com base em exemplos de treinamento.

Para a utilização do KNN é necessário um conjunto de exemplos de treinamento, a definição de uma métrica para calcular a distância entre os exemplos de treinamento e a definição de K que é o número de vizinhos mais próximos que serão considerados durante a classificação de um novo dado.

O KNN realiza a classificação de um novo objeto baseado no vizinho mais próximo e é uma técnica muito utilizada para o reconhecimento de padrões. O núcleo de seu funcionamento está em descobrir os K vizinhos mais próximos de uma determinada nova instância, conforme apresentado pela Figura 4.8.

Figura 4.8 – Apresentação do algoritmo KNN.



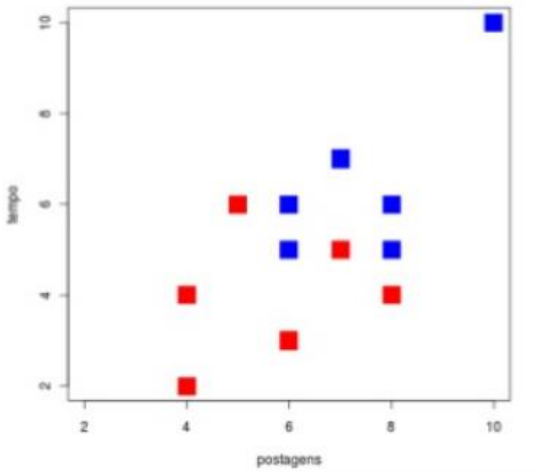
Como exemplo de funcionamento do KN, vamos considerar a Tabela 4.1. Nessa tabela temos alguns dados referentes ao comportamento de um aluno em um curso a distância. Os atributos considerados são três: tempo de utilização do fórum (X1), número de postagens (X2) e se o aluno passou na disciplina (y).

Tabela 4.1 – Dados usados para treinamento do algoritmo KNN.

x_1 : Tempo de utilização	x_2 : Número postagens	y : Passou na disciplina
2	4	Não
3	6	Não
4	8	Não
4	4	Não
5	7	Não
6	5	Não
6	6	Sim
6	5	Sim
7	7	Sim
8	5	Sim
8	6	Sim
10	10	Sim

A Figura 4.9 apresenta a distribuição dos dados da Tabela 1 em um gráfico. Os alunos que foram aprovados na disciplina estão destacados em azul e os que não foram estão destacados em vermelho. O eixo X representa o número de postagens e o eixo Y o tempo que cada aluno permaneceu no fórum.

Figura 4.9 – Representação gráfica dos dados de treinamento.



Para o cálculo das distâncias, foi utilizada a fórmula abaixo (euclidiana quadrada):

$$d(A,B) = \sum_{i=1}^{n}(x_i^A - x_i^B)^2.$$

A Tabela 4.2 apresenta o resultado do cálculo das distâncias ao incluirmos uma nova instância (um novo aluno). Esse novo aluno possui seis postagens no fórum e tempo de permanência igual a 7.

Tabela 4.2 – Cálculo das distâncias para uma nova instância.

x_1 : utilização	x_2 : postagens	Distância para o $(6, 7)$
2	4	$(2-6)^2 + (4-7)^2 = 25$
3	6	10
4	8	5
4	4	13
5	7	1
6	5	4
6	6	1
6	5	4
7	7	1
8	5	8
8	6	5
10	10	25

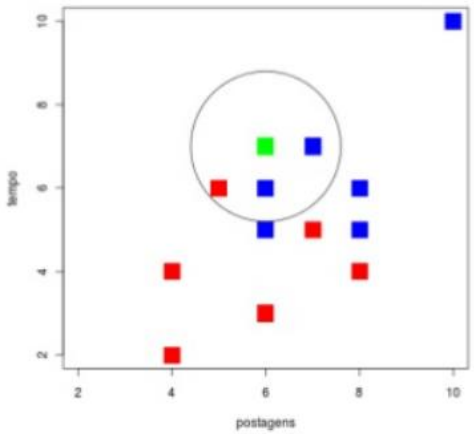
Por último, a Tabela 4.3 identifica os K registros de referência que apresentam a menor distância em relação ao novo registro. Para esse exemplo, definimos o valor de K como 3.

Tabela 4.3 – Cálculo das distâncias para um novo registro.

x_1 : utilização	x_2 : postagens	Distância para o $(6, 7)$	KNN
2	4	25	
3	6	10	
4	8	5	
4	4	13	
5	7	1	✓
6	5	4	
6	6	1	✓
6	5	4	
7	7	1	✓
8	5	8	
8	6	5	
10	10	25	

A Figura 4.10 apresenta a representação gráfica da inserção do novo aluno. Observe que, dos três vizinhos mais próximos, dois são dos alunos aprovados e um dos reprovados. Como a maioria dos vizinhos é da classe dos aprovados, o aluno vai ser classificado como aprovado.

Figura 4.11 – Classificação de um novo aluno baseado nos três vizinhos mais próximos.



Naive Bayes

O algoritmo Naive Bayes é um classificador probabilístico baseado no "Teorema de Bayes", o qual foi criado por Thomas Bayes (1701 - 1761) para tentar provar a existência de Deus. Atualmente, o algoritmo se tornou popular na área de

Aprendizado de Máquina para categorizar textos baseados na frequência das palavras usadas, e assim pode ser usado para identificar se determinado e-mail é um SPAM ou sobre qual assunto se refere determinado texto, por exemplo.

Por ser muito simples e rápido, possui um desempenho relativamente maior do que outros classificadores. Além disso, o Naive Bayes só precisa de um pequeno número de dados de teste para concluir classificações com uma boa precisão. A principal característica do algoritmo, e também o motivo de receber “naive” (ingênuo) no nome, é que ele desconsidera completamente a correlação entre as variáveis (*features*). Ou seja, se determinada fruta é considerada uma “maçã” se ela for “vermelha”, “redonda” e possui “aproximadamente 10 cm de diâmetro”, o algoritmo não vai levar em consideração a correlação entre esses fatores, tratando cada um de forma independente.

O teorema de Bayes é apresentado abaixo:

$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$$

Onde:

- **P(c | x):** probabilidade da classe c, dado o vetor de características x.
- **P(x | c):** probabilidade do vetor x dada a classe c.
- **P(c):** probabilidade a priori da classe c.
- **P(x):** probabilidade a priori do vetor de treinamento x.

Podemos utilizar o exemplo do jogo de tênis. O objetivo é predizer o valor alvo (sim/não) que decide se o jogo será realizado ou não.

Devemos estimar duas probabilidades a posteriori:

- $P(c_j = yes \mid x_t)$, $P(c_j = no \mid x_t)$

Os dados usados no treinamento são apresentados na Figura 4.12.

Figura 4.12 – Dados utilizados no treinamento do algoritmo Naive Bayes.

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Podemos calcular as classes P(yes) e P(no) da forma apresentada abaixo, lembrando que o total de instâncias (14) será levado em consideração no cálculo.

- P(yes) = 9 / 14 = 0,643.
- P(no) = 5/14 = 0,357.

Por último, um vetor de combinações com todas as possibilidades do treinamento é montado. Durante a montagem desse vetor, devemos estimar todas as possibilidades condicionais, considerando todas as classes possíveis. Diante do exemplo apresentado, totalizaremos 72 probabilidades condicionais.

O Naive Bayes é um dos métodos mais práticos usados principalmente quando a disponibilidade de um conjunto de treinamento é grande ou moderada e os atributos descrevem a classe de maneira independente.

Capítulo 5. Atualidades sobre Machine Learning

Este capítulo tem o objetivo de discutir as principais linguagens que são atualmente utilizadas para a implementação de algoritmos de Machine Learning, visando a solução dos mais diversos tipos de problemas nas inúmeras áreas do conhecimento. Além disso, esse capítulo irá apresentar as principais bibliotecas e *frameworks* disponíveis atualmente para facilitar o trabalho do desenvolvedor. Por fim, aplicações modernas que utilizam Machine Learning em conjunto com IoT (Internet das Coisas) e Big Data serão discutidas.

Principais linguagens e frameworks para Machine Learning

Atualmente, temos disponíveis no mercado diversas linguagens de programação que visam, cada vez mais, facilitar o trabalho de desenvolvimento de software. As mais modernas dessas linguagens prezam pelo reuso e o desenvolvimento ágil de código-fonte, permitindo que o desenvolvedor se preocupe cada vez menos com a tecnologia a ser aplicada e possa se dedicar mais à solução do problema.

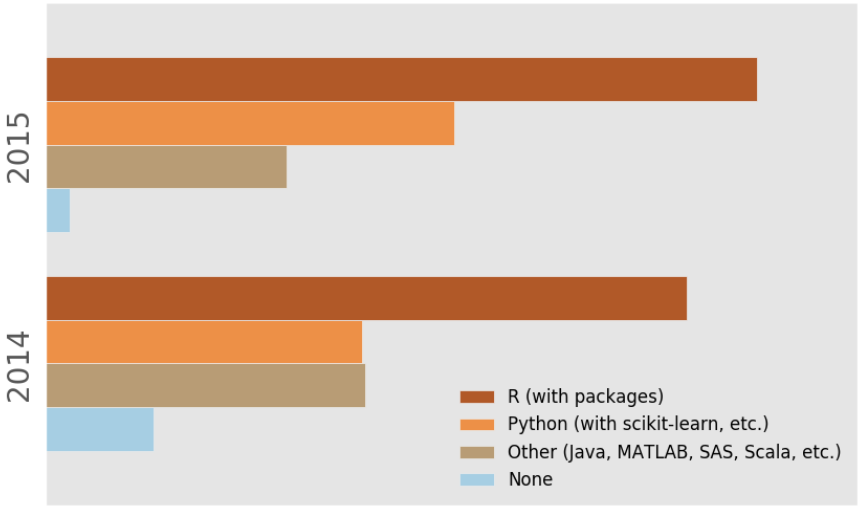
Muitas vezes encontramos discussões acaloradas em fóruns, blogs e afins sobre qual linguagem é a mais adequada para determinada solução que envolve algoritmos de aprendizado de máquina. Sempre percebemos desenvolvedores defendendo aquelas tecnologias com as quais mais simpatizam. Essa defesa é natural, não somente no mundo do aprendizado de máquina, mas também no contexto geral do desenvolvimento de software. Não é difícil encontrarmos simpatizantes e verdadeiros defensores de determinadas tecnologias, seja dos sistemas operacionais (Windows x Linux), dos Sistemas Gerenciadores de Bancos de Dados – SGBDs (Oracle x MSSql), das metodologias (Metodologia Ágil x Tradicional) e das linguagens de programação (Java x C#, PHP x ASP.Net, Python x outra linguagem qualquer).

Diante dessa situação, os iniciantes no mundo do Machine Learning sempre se pegam fazendo as mesmas perguntas por diversas vezes: por qual linguagem devo começar meus estudos? Qual linguagem é mais comercial para aplicações de Machine Learning? O mercado está procurando por um profissional que domina qual(is) linguagem(ns)? Qual linguagem aumentará a minha empregabilidade? Realmente essas e outras perguntas correlatas são difíceis de serem respondidas, principalmente porque as tecnologias vêm e vão com o tempo. O que é muito utilizado hoje pode estar completamente depreciado daqui um ano. O que ninguém conhece hoje pode ser o que vai realmente ser mais usado no futuro próximo.

Expostas essas questões sobre as linguagens, podemos dizer que o esforço que realmente vale a pena para os iniciantes é o de entender os algoritmos, suas estruturas e, principalmente, suas aplicações dentro do contexto do Machine Learning. A grande preocupação dos desenvolvedores deve estar concentrada em conhecer os algoritmos, técnicas e metodologias de Machine Learning e saber o melhor momento de aplicar cada uma delas. Um determinado algoritmo pode ser desenvolvido utilizando quase todas as linguagens de alto-nível disponíveis no mercado, daí o motivo para não nos preocuparmos somente com qual linguagem de programação devemos nos especializar.

É comum encontrarmos projetos que implementam algoritmos de Aprendizado de Máquina que utilizam linguagens como: Matlab, R, Python, Java e a família C (C, C++ e C#). É indiscutível também que temos algumas linguagens que, aparentemente, parecem ser mais usadas que outras. Nesse conjunto podemos destacar o Python, o R e o Java. Isso se deve ao fato da grande popularidade dessas linguagens, não somente para aplicações que envolvem técnicas de Machine Learning, mas para aplicações em geral.

Figura 5.1 – Principais linguagens de programação para Análise de Dados e Machine Learning – Kdnuggets (2015).



Na Figura 5.1 podemos perceber que em pesquisas realizadas entre os anos de 2014 e 2015, a linguagem R dominou a preferência dos desenvolvedores, seguida pela linguagem Python. O que chama a atenção entre as duas pesquisas é que a linguagem R cresceu no número de adeptos, e a linguagem Python superou todas as outras linguagens (Java, MATLAB, Scala, etc.) juntas.

Outro ponto importante a se destacar no desenvolvimento de algoritmos que envolvem Aprendizado de Máquina é o uso de bibliotecas e frameworks. Muitas ferramentas disponíveis no mercado e na comunidade de desenvolvedores já trazem implementados e prontos os principais algoritmos de Aprendizado de Máquina. Isso facilita muito a vida do desenvolvedor, pois a maioria desses algoritmos já foi amplamente testada e desenvolvida utilizando as melhores técnicas e padrões de projeto. A variedade dessas ferramentas é muito grande e praticamente existe um *framework* para cada uma das principais linguagens de programação. Abaixo estão listados os principais *frameworks* e bibliotecas disponíveis, classificados por linguagens de programação:

- C++: MultiBoost, Shogun e LibSVN.

- JavaScript: Math.js, Brain.js (especialista em Redes Neurais), Natural (processamento de linguagem natural) e Webdnn (Deep Learning).

- Scala: Epic, ScalaNLP e PredictionIO.

Alguns outros *frameworks* são também muito utilizados e possuem grande variedade de algoritmo já prontos. São eles:

- Scikit-learn (http://scikit-learn.org/stable/): é um *framework* de código aberto e desenvolvido para a utilização em conjunto com a linguagem de programação Python. Traz implementados diversos algoritmos de classificação, regressão e agrupamento, além de uma vasta documentação e uma grande comunidade de desenvolvedores e usuários. A ferramenta também é bastante utilizada por empresas.

- TensorFlow (https://www.tensorflow.org/): é um *framework* de código fonte aberto e desenvolvido pela Google com o objetivo de ser um padrão aberto para a inteligência artificial. A base do Tensorflow são as redes neurais e diversas ferramentas do Google foram em partes desenvolvidas por essa ferramenta, como o Google Translator, Photos e até o próprio buscador.

- Shogun (http://www.shogun-toolbox.org/): é um dos *frameworks* mais antigos e veneráveis para Machine Learning. Nasceu em 1999 e foi escrito utilizando C++, entretanto não está mais limitado à essa linguagem de programação. Atualmente pode ser usado em ambientes Java, Python, C#, Ruby, MATLAB e outros.

- Apache Mahout (http://mahout.apache.org/): o Mahout está diretamente alinhado à outra tecnologia que é o Apache Hadoop (http://hadoop.apache.org/). O Hadoop é um *framework* para processamento de grandes massas de dados de maneira distribuída. Ao utilizá-lo, o desenvolvedor não precisa se preocupar com a distribuição do processamento, pois isso fica a cargo do próprio *framework*. O Mahout disponibiliza algoritmos de Machine Learning que podem ser executados ou não na plataforma Hadoop.

- Apache Spark MLlib (https://spark.apache.org/mllib/): o MLlib está alinhado à tecnologia de processamento distribuído Spark (https://spark.apache.org/). O Apache Spark se diferencia do Apache Hadoop por priorizar o processamento em memória, ao invés do disco. Com isso, o Spark consegue desempenhos melhores que o Hadoop em diversos tipos de testes. O Java é a linguagem primária para a utilização dessa tecnologia, porém pode-se utilizar também o Python e o Scala. O Spark MLlib traz implementadas diversas técnicas de Machine Learning, tais como algoritmos para classificação, regressão e agrupamento.

- WEKA (https://www.cs.waikato.ac.nz/ml/weka/): o Weka é outra opção de *framework* para Machine Learning criado pela universidade de Waikato (Nova Zelândia). Possui um conjunto de algoritmos que utilizam diversas técnicas, tais como: classificação, agrupamento e regressão. Além disso, o Weka disponibiliza interface gráfica e possibilidade de apresentação dos resultados em diversos formatos de gráfico. Nas últimas versões, o Weka já traz integrações com ferramentas de processamento distribuído, como Apache Spark e Hadoop.

Outro ponto que é importante destacar é que diversos serviços em nuvem já trazem os principais algoritmos de Machine Learning implementados. Serviços como Microsoft Azure, Amazon AWS e Google Cloud já têm embutido essas tecnologias em seus serviços, cabendo ao desenvolvedor escolher as opções de algoritmos que mais se adequam ao problema que deverá ser resolvido. As Figuras 5.2 e 5.3 apresentam as interfaces do Google Cloud e Microsoft Azure, respectivamente.

Figura 5.2 – Google Cloud Machine Learning.

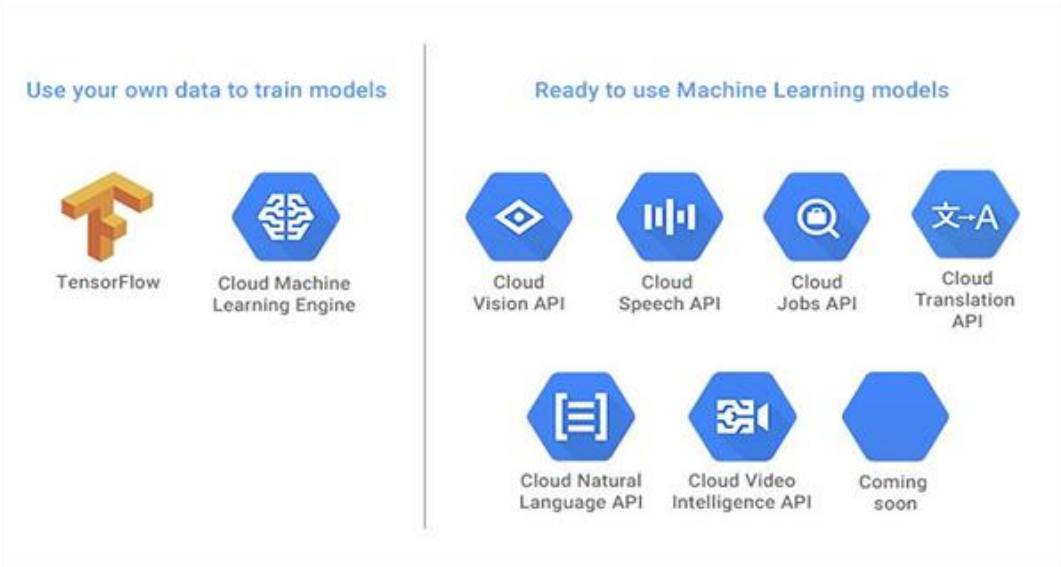
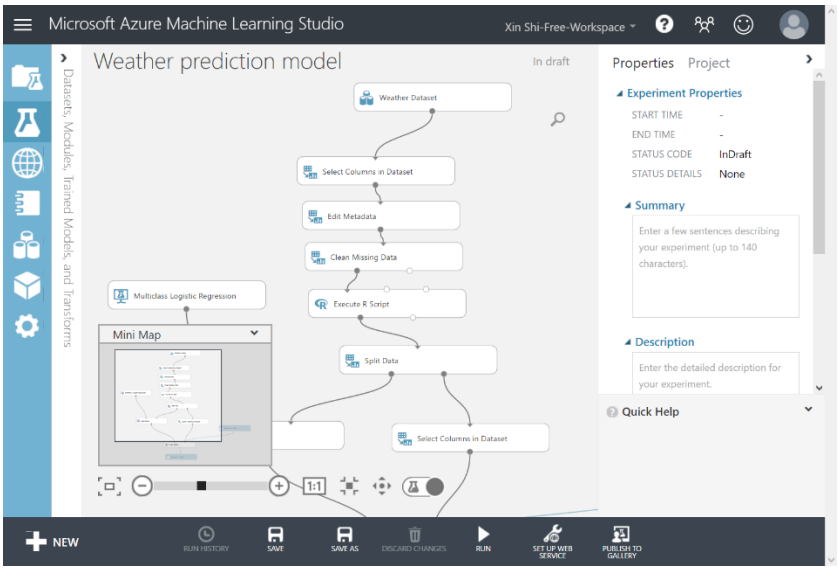


Figura 5.2 – Microsoft Azure Machine Learning.



Aplicações atuais de Machine Learning (IOT, Big Data, etc.)

Atualmente, podemos dizer que vivemos a grande retomada das aplicações de Machine Learning. Desde as primeiras ideias e modelos da inteligência artificial dos anos 60 até hoje, passamos por vários altos e baixos. Muitos problemas foram

enfrentados e resolvidos, o hardware evoluiu, assim como os modelos e também os algoritmos.

Podemos dizer que a grande quantidade de dados que temos hoje e a evolução do poder computacional contribuíram diretamente para a popularização da utilização dos algoritmos de Machine Learning nos nossos problemas cotidianos e contemporâneos. Diante dessa situação, temos três pilares que, nos dias atuais, estão sustentando a inovação na computação. Esses pilares são: o aprendizado de máquina, as grandes massas de dados (Big Data) e a Internet das Coisas (IOT).

Como esses três pilares se relacionam? A Internet das Coisas vem permitindo que cada vez mais dispositivos possam se conectar na rede (Internet) e produzirem dados. Atualmente temos automóveis, geladeiras, televisores, smartphones, relógios e uma grande variedade de dispositivos interligados e compartilhando informações. Todos esses dados que são produzidos acabam gerando uma grande massa, em uma velocidade e variedade nunca vistas antes. Essa variedade acontece devido aos diversos formatos de dados que são gerados, tais como: fotos, vídeos, músicas, textos, logs de máquinas e dados de experimentos científicos. Por último, precisamos de alguma maneira tirar vantagem dessa grande quantidade de dados que temos disponível. Para isso, utilizamos também os algoritmos de Machine Learning, pois toda essa base pode ser útil, por exemplo, no treinamento dos algoritmos.

Mesmo que não percebamos, atualmente utilizamos constantemente recursos tecnológicos que mesmo em pequena escala implementam algum algoritmo inteligente. Diariamente acessamos sistemas que nos recomendam produtos e serviços, fazemos alguma solicitação em instituições financeiras que realizam análises de crédito utilizando algoritmos inteligentes e acessamos nossas redes sociais e muitas vezes temos nossos passos monitorados e analisados.

Até no esporte podemos encontrar o uso desses algoritmos em larga escala. Equipes de futebol já estudam constantemente o comportamento dos seus adversários a partir de dados do passado e tentam realizar predições, como por exemplo, o lado que um goleiro mais pula no momento do pênalti ou o lado que determinado batedor mais escolhe. Além disso, temos diversos varejistas, como

supermercados, utilizando a análise de folhetos de promoções da concorrência, por meio de reconhecimento de imagem, para precificar seus produtos.

Outras aplicações já têm grande impacto na sociedade como um todo, como análise de tendências políticas, diagnósticos médicos para auxílio na tomada de decisão, tendências de epidemias e outros serviços como os Chatbots (assistentes digitais).

Na área de aplicações preditivas (que tentam prever o futuro baseado em dados passados), temos identificação de tendência de clientes, de bolsa de valores e do mercado financeiro como um todo. Esses sistemas tentam prever o comportamento de um agente para melhorar o desempenho dos negócios.

Na área da automação já é comum o uso de robôs inteligentes em indústrias e também nos lares. Os eletrodomésticos inteligentes, que tentam entender nossos comportamentos e preferências, são o sonho de consumo de muita gente. A Internet das Coisas já permite a integração de vários dispositivos em uma casa, como cortinas inteligentes, acendimento de luzes automático, garagens que “sabem” quando o carro da casa se aproxima, sensores que detectam diversas anomalias no lar, como vazamento de gás, falta de energia ou presença de fumaça. Além disso, os drones já cumprem tarefas que são árduas para o ser humano e vão a lugares onde a presença humana é arriscada.

Um ponto que é preciso destacar nessa história toda é a legislação. É importante estarmos preparados para alterações em nossas leis para que tudo esteja adequado para receber essas inovações, que são as máquinas inteligentes. É preciso definir os limites e as atribuições que serão permitidas à essas máquinas. Até onde vai a responsabilidade do dono da máquina, a partir do momento que ela consegue tomar uma decisão sozinha? Como ficarão algumas questões e valores fundamentais como moral e ética? É preciso abrir um amplo debate sobre esse tema, antes que toda a evolução que a Inteligência Artificial poderá trazer fique engessada.

Referências

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.

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

KELLEHER, D.J. et al. *Fundamentals of Machine Learning for Predictive Data Analytics*. 1. ed. MIT Press, 2015.

PENTREATH, N. *Machine Learning with Spark* – Tackle Big Data with Powerful Spark Machine Learning. 1. ed. Packt Publishing, 2015.