

Machine Learning



Cruzeiro do Sul Virtual
Educação a distância

Material Teórico



Introdução aos Modelos Preditivos

Responsável pelo Conteúdo:

Prof. Me. Orlando da Silva Junior

Revisão Textual:

Prof.^a Dr.^a Selma Aparecida Cesarin

UNIDADE

Introdução aos Modelos Preditivos



- Introdução;
- Métodos Baseados em Distâncias;
- Métodos Probabilísticos;
- Métodos Baseados em Busca;
- Métodos Baseados Em Otimização.



OBJETIVOS DE APRENDIZADO

- Estudar a aplicação geral dos modelos preditivos;
- Comparar os diferentes métodos de aprendizagem supervisionada.



Orientações de estudo

Para que o conteúdo desta Disciplina seja bem aproveitado e haja maior aplicabilidade na sua formação acadêmica e atuação profissional, siga algumas recomendações básicas:



Assim:

- ✓ Organize seus estudos de maneira que passem a fazer parte da sua rotina. Por exemplo, você poderá determinar um dia e horário fixos como seu “momento do estudo”;
- ✓ Procure se alimentar e se hidratar quando for estudar; lembre-se de que uma alimentação saudável pode proporcionar melhor aproveitamento do estudo;
- ✓ No material de cada Unidade, há leituras indicadas e, entre elas, artigos científicos, livros, vídeos e sites para aprofundar os conhecimentos adquiridos ao longo da Unidade. Além disso, você também encontrará sugestões de conteúdo extra no item **Material Complementar**, que ampliarão sua interpretação e auxiliarão no pleno entendimento dos temas abordados;
- ✓ Após o contato com o conteúdo proposto, participe dos debates mediados em fóruns de discussão, pois irão auxiliar a verificar o quanto você absorveu de conhecimento, além de propiciar o contato com seus colegas e tutores, o que se apresenta como rico espaço de troca de ideias e de aprendizagem.

Introdução

Um Algoritmo de Aprendizado de Máquina Preditivo é uma função que, dado um conjunto de exemplos rotulados, constrói um estimador. Nesse caso, o rótulo toma valores num domínio conhecido:

Se esse domínio for um conjunto de valores nominais, temos um problema de classificação, e o estimador gerado é um classificador;

Se o domínio for um conjunto infinito e ordenado de valores, temos um problema de regressão, e o estimador é um regressor.

Para exemplificar, observe o conjunto de dados da Tabela 1 e imagine dois problemas diferentes que podem ser solucionados a partir desses dados:

- Você deseja identificar o diagnóstico de um novo paciente. O rótulo desse problema é o atributo Diagnóstico, cujos valores são nominais (Doente ou Saudável). Nesse caso, você deverá trabalhar na construção de um classificador;
- Você deseja estimar a temperatura de um novo paciente a partir do diagnóstico dele e de outras informações. O rótulo Temperatura abrange valores de domínio do conjunto de números reais (por exemplo, 30,0°, 30,1°, 30,2°). Nesse caso, você trabalhará na construção de um estimador.

Tabela 1 – Conjunto de dados de pacientes para o diagnóstico de doenças

Nome	Idade	Sexo	Temperatura	Dores	Diagnóstico
Maria	54	F	39.0	Sim	Doente
João	33	M	38.7	Não	Saudável
José	29	M	35.4	Sim	Saudável
Carlos	48	M	36.0	Não	Doente
Ana	21	F	36.5	Sim	Doente

Métodos Baseados em Distâncias

Os métodos baseados em distâncias consideram a proximidade entre os dados para a construção do modelo. Esses métodos assumem que dados similares tendem a estar concentrados em uma mesma região no espaço.

O algoritmo dos vizinhos mais próximos é o método de *Machine Learning* baseado em distância mais utilizado.

Ele adota a ideia de que os exemplos relacionados ao mesmo conceito são semelhantes entre si. Dessa forma, o classificador relaciona um novo exemplo com base nos exemplos de treinamento mais próximos a ele.

Esse algoritmo tem diferentes variações, sendo a maior parte delas relacionada ao número de vizinhos.

Vamos conhecer duas variações desse algoritmo?

Algoritmo 1-NN

A primeira e mais simples variação do algoritmo dos vizinhos mais próximos é interpretado com apenas 1 vizinho, sendo chamado de 1-vizinho mais próximo (do inglês *1-nearest neighbour*, ou 1-NN).

Para ele, cada exemplo de treinamento está situado no espaço de entrada, sendo possível calcular a distância entre dois exemplos.

Em geral, usamos a distância euclidiana para calcular a distância entre eles:

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

Na expressão acima, a distância euclidiana d entre dois exemplos de treinamento x_i e x_j é a raiz quadrada da somatória das diferenças entre os valores dos atributos l desses exemplos.

Para o algoritmo 1-NN, $d = 1$.

O algoritmo memoriza os exemplos durante o treinamento e , para classificar um novo exemplo, calcula a distância entre os valores dos atributos do novo exemplo com todos os exemplos memorizados. Assim, o novo exemplo será classificado de acordo com o rótulo do exemplo de treinamento mais próximo.

Um dos problemas dos métodos baseados em distância é que o desempenho preditivo pode ser afetado pela função de distância adotada. Neste caso, você deve considerar duas questões importantes ao selecionar algoritmos:

- **O tipo do atributo:** se o atributo é qualitativo, você deverá selecionar uma função que calcule a distância para valores não numéricos;
- **A escala dos valores:** em geral, valores numéricos devem ser normalizados antes de serem utilizados em algoritmos baseados em distância.

Algoritmo k-NN

O algoritmo k-NN é uma extensão do algoritmo dos vizinhos mais próximos que seleciona os k exemplos de treinamento mais próximos do novo exemplo a ser classificado.

Quando $k > 1$, o algoritmo recupera k vizinhos para cada nova classificação, fazendo com que cada vizinho escolha uma classe. Finalmente, a classificação é realizada conforme a seguinte regra:

- Se o rótulo é qualitativo, ou seja, o problema é um problema de classificação, o novo exemplo é classificado de acordo com a classe mais votada, por meio da moda estatística;
- Se o rótulo é quantitativo e estamos lidando com um problema de regressão, o novo exemplo será classificado por meio da média ou da mediana.

Entre as principais vantagens do k -NN estão a simplicidade do algoritmo de treinamento e a sua aplicabilidade em diferentes problemas, inclusive aqueles mais complexos.

Por outro lado, é importante estar atento aos pontos negativos, como a inexistência de um modelo que represente os dados (considerando que os dados são apenas memorizados durante o treinamento) e a dificuldades em conjuntos com muitos atributos, vez que a quantidade de atributos define o número de dimensões no espaço de busca do modelo.

Métodos Probabilísticos

Os métodos probabilísticos são uma alternativa interessante quando você possui um conjunto de dados com informações incompletas ou imprecisas. O Teorema de Bayes, base dos algoritmos probabilísticos, pode ajudar a resolver problemas com essas características.

Conforme o Teorema de Bayes, você pode calcular a probabilidade de um evento acontecer a partir da ocorrência de outro evento que já aconteceu. Nesse caso, você precisará das informações desse evento para estimar o que acontecerá com o evento que você deseja prever.

Comumente, o Teorema de Bayes é descrito pela fórmula:

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

Onde:

- $P(A)$ e $P(B)$ correspondem, respectivamente, à probabilidade dos eventos A e B acontecerem; e
- $P(A|B)$ corresponde à probabilidade do evento A acontecer dado que o evento B já aconteceu.

Para entender esse Teorema, imagine a ocorrência dos seguintes eventos:

- **A:** Um paciente apresentar uma determinada doença;
- **B:** Ter resultado positivo em um exame de raio X;

Como estimar a probabilidade de que um paciente apresente uma determinada doença dado que o resultado do exame de raio X é positivo?

Essa pergunta indica que queremos calcular $P(A|B)$. Para encontrarmos $P(A|B)$, teremos de conhecer antecipadamente $P(B)$.

A probabilidade de ocorrência do evento B, dada por $P(B)$, pode ser estimada por meio da observação da frequência com que esse evento ocorre.

Em outras palavras, $P(B)$ quer responder à seguinte pergunta: qual é a probabilidade de um resultado ser positivo em um exame de raio X?

Em *Machine Learning*, o Teorema de Bayes fornece uma maneira de calcular a probabilidade de um exemplo pertencer a uma classe. Os métodos probabilísticos assumem que $P(A|B)$ não depende apenas da relação entre A e B, mas também da probabilidade de observar A independentemente de observar B.

Aprendizado Bayesiano

Suponha que a probabilidade de você encontrar um paciente com uma determinada doença é de 10%. Existe um teste para detectar a existência dessa doença que pode ser aplicado ao paciente.

Usando a linguagem da probabilidade, temos:

$$P(\text{doença} = \text{presente}) = 10\%$$

Como todo teste clínico, esse teste está sujeito a erros: um paciente doente pode ser detectado como não tendo a doença, assim como um paciente saudável pode ser detectado como portador da doença.

Sabemos que, em 80% dos casos, o resultado do teste foi positivo, a doença foi confirmada. Ou seja:

$$P(\text{teste} = \text{positivo} \mid \text{doença} = \text{presente}) = 80\%$$

Sabemos também que, em 98%, o teste deu negativo e o paciente não tinha a doença. Ou seja:

$$P(\text{teste} = \text{negativo} \mid \text{doença} = \text{ausente}) = 98\%$$

Podemos resumir todas essas informações probabilísticas por meio da Tabela 2:

Tabela 2 – Informações probabilísticas a respeito do problema médico

Teste	Doença	
	Presente	Ausente
positivo	80%	2%
negativo	20%	98%

Em *Machine Learning*, chamamos de **sensibilidade** à probabilidade dos verdadeiros positivos. Observando a Tabela, a taxa de **verdadeiros positivos** corresponde a 80%. Essa taxa indica o número de exemplos da classe positiva classificados corretamente.

Também chamamos de **especificidade** à probabilidade dos verdadeiros negativos. Neste caso, a taxa de **verdadeiros negativos** corresponde a 98%, indicando a quantidade de acertos do teste para a classe negativa.

Sabemos que o valor do atributo Doença influencia no valor do atributo Teste, mas o contrário não acontece. Mesmo assim, desejamos saber de que forma o atributo Teste é capaz de identificar o valor da Doença.

Nesse caso, consideramos o atributo Teste como um atributo de entrada, e o atributo Doença como o nosso atributo de saída (ou atributo alvo).

Algoritmo Naïve Bayes

Considerando que os valores dos atributos são independentes de sua classe, podemos decompor a probabilidade em um produto de probabilidades:

$$P(x | y_i) = P(x_1, x_2, \dots, x_n | y_i) = \prod_{j=1}^n P(x_j, y_i)$$

Ao aceitarmos isso, estamos dizendo que os atributos do exemplo x , representados por x_1, x_2, \dots, x_n , não estão correlacionados entre si.

Assim, a probabilidade de um exemplo x pertencer à classe y_i é proporcional (representada pelo símbolo \propto) à expressão:

$$P(y_i | x) \propto P(y_i) \cdot \prod_{j=1}^n P(x_j, y_i)$$

Sabendo que o algoritmo pode classificar incorretamente um novo exemplo, como realizar a melhor classificação?

Para encontrar a melhor classificação, o algoritmo deverá maximizar a probabilidade $P(y_k | x)$ de associar corretamente o exemplo x à classe y_i .

Assim, a classe que deve ser associada ao exemplo x é:

$$y_{\text{MAP}} = \text{argmax}_i P(y_i | x)$$

Onde:

- argmax_i retorna a classe y_i com maior probabilidade de estar associada a x .

O termo *naïve* (do inglês, ingênuo) vem da hipótese de que os valores dos atributos de um exemplo são independentes de sua classe.

Durante a implementação desse algoritmo, o algoritmo deverá calcular todas as probabilidades necessárias para a obtenção do classificador. Ele deverá computar essas informações a partir dos dados de treinamento seguindo estas regras:

- Para calcular a **probabilidade a priori** $P(y_i)$ de observar a classe y_i , é necessário manter um contador para cada classe;
- Para calcular a **probabilidade condicional** de observar um valor de um atributo dado que o exemplo pertence à classe, é necessário distinguir entre atributos nominais e atributos contínuos.

Entre as vantagens do algoritmo *naïve* Bayes, estão: a possibilidade de todas as probabilidades poderem ser calculadas em uma única varredura, o bom desempenho em domínios em que há claras dependências entre os atributos e a robustez do modelo na presença de ruídos e atributos irrelevantes.

Redes Bayesianas

Embora o classificador *naïve* Bayes tenha suas vantagens, ele é incapaz de lidar com a independência dos atributos.

Alternativamente usamos Redes Bayesianas para incluir o conceito de independência condicional, fazendo com que exista um equilíbrio entre o número de parâmetros e a representação de dependências entre os atributos.

A independência condicional $P(A|B,C)$ diz que dois eventos A e B são condicionalmente independentes se, ocorrendo um terceiro evento C, a distribuição de probabilidades de A é independente do valor de B, conhecendo-se o valor de C. Por exemplo, o som do trovão é condicionalmente independente da chuva sabendo que o relâmpago aconteceu.

Uma Rede Bayesiana corresponde à distribuição de probabilidade conjunta para um conjunto de atributos. Em geral, ela é representada por um gráfico acíclico direcionado, no qual cada nó é declarado ser condicionalmente independente de seus nós dependentes dado a seu predecessor imediato.

Uma pergunta-chave para a aprendizagem em redes bayesianas é: como inferir as probabilidades dos valores de um ou mais atributos a partir dos valores observados de outros atributos?

Embora esse seja um problema difícil, as redes bayesianas possuem toda a informação necessária para realizar a inferência, vez que o grafo é inteiramente conectado. Além disso, se apenas um atributo tiver valores desconhecidos, a tarefa de inferência é trivial.

A principal vantagem dessa forma de aprendizado é que o algoritmo procura combinar conhecimento prévio (*a priori*) com os dados observados, permitindo que a complexidade da amostra seja menor.

Métodos Baseados em Busca

Um problema de *Machine Learning* pode ser formulado como um problema de busca em um espaço de possíveis soluções, necessitando apenas de:

- Uma linguagem para representar as generalizações dos exemplos; e
- Uma função de avaliação de modelos (hipóteses).

As árvores de decisão, as árvores de regressão e as regras de decisão são os principais modelos baseados em busca.

Árvores de Decisão

Uma árvore é um grafo acíclico direcionado em que cada nó é um nó de divisão ou um nó folha, seguindo a regra:

- **Nó folha:** é rotulado como uma função em que os valores do atributo alvo nos exemplos que chegam a um nó folha;
- **Nó de divisão:** possui dois ou mais sucessores, sendo um teste condicional baseado nos valores do atributo. Em geral, os testes envolvem um único atributo e os valores no domínio desse atributo.

A Figura 1 (GRABUSTS; BORISOV; ALEKSEJEVA, 2015) ilustra a árvore de decisão construída para o famoso conjunto de dados Iris, que possui 150 exemplos, 4 atributos de entrada (*petalwidth*, *petallength*, *sepalwidth* e *sepallength*) e 1 atributo de saída com 3 classes (**setosa**, **versicolor** e **virginica**).

Entre os diversos modelos que o algoritmo pode induzir, a figura apresenta uma árvore construída com apenas dois atributos de entrada (*petalwidth* e *petallength*), representados pelas formas ovais.

Nas formas retangulares, encontramos o nome da classe e a quantidade de exemplos que a divisão discriminou naquele nó. A conexão entre as formas, representada pelas setas, informa o valor do atributo que ocasionou a divisão.

Para induzir árvores de decisão, você precisará construir um algoritmo de treinamento que receba como entrada o conjunto de dados de treinamento e devolva como saída a árvore de decisão com os nós de divisão e nós folha. Para transformar a entrada na saída, o seu algoritmo deverá dividir a entrada em subconjuntos cada vez menores e utilizar esses subconjuntos para construir incrementalmente a árvore desejada.

Uma pergunta-chave para a construção de árvores de decisão é: como escolher um atributo para servir como nó de divisão?

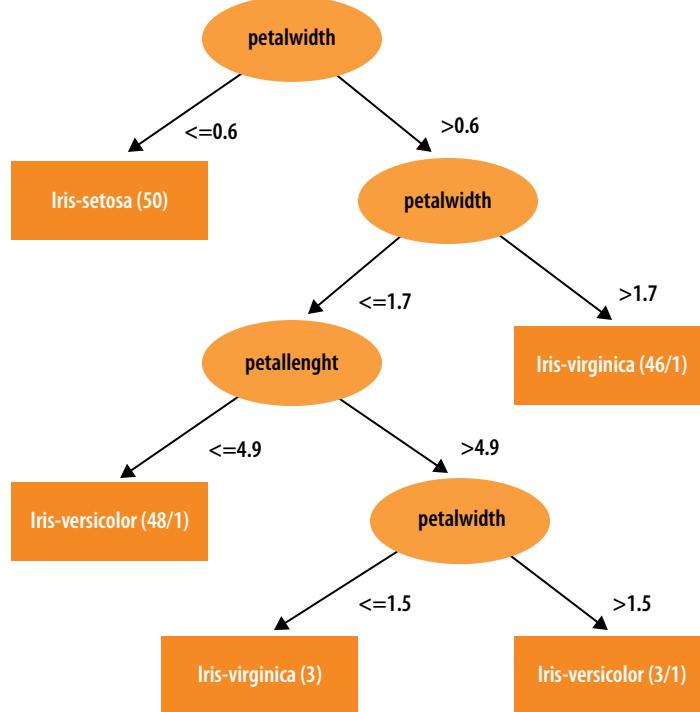


Figura 1 – Exemplo de árvore de decisão para o conjunto de dados Iris

De maneira geral, o algoritmo deve escolher um atributo que maximiza o critério de divisão no conjunto de treinamento. Nas tarefas de classificação, o atributo escolhido deve discriminar as classes, permitindo que o novo exemplo seja atribuído a uma das classes do problema.

Uma estratégia utilizada por alguns algoritmos é construir a árvore visando a reduzir a aleatoriedade do atributo alvo, ou seja, a dificuldade em predizê-lo.

Para isso, os algoritmos medem a entropia do atributo para cada nó de decisão que será gerado para a árvore. Como os valores dos atributos definem partições no conjunto de treinamento, o algoritmo calcula o ganho de informação para cada atributo.

Os algoritmos ID3, C4.5 e CART são os mais populares da literatura de *Machine Learning*.

Entre as principais vantagens desses algoritmos, podemos destacar:

- A robustez da formulação da árvore às transformações de variáveis de entrada. Por exemplo, o algoritmo produz a mesma árvore se o atributo for x , $\log x$ ou e^x ;
- A flexibilidade do algoritmo em não assumir uma distribuição para os dados, sendo, assim, categorizado como um método não paramétrico;
- A interpretabilidade do modelo, que é apresentado em forma de uma árvore de decisão, estrutura bastante comum em outras áreas do conhecimento; e
- A seleção automática de bons atributos, que colabora para a produção de modelos mais robustos.

Apesar de todos esses benefícios, você precisa estar ciente das situações em que usar um algoritmo de indução de árvores de decisão pode não ser uma boa ideia:

- Árvores não lidam bem com valores ausentes;
- Atributos numérico contínuos podem ser um problema e precisam ser tratados; e
- O modelo da árvore pode variar conforme o conjunto de treinamento.

Regras de Decisão

A indução de regras de decisão é bastante semelhante ao processo que você acabou de aprender para as árvores de decisão.

Ambas utilizarão a mesma forma de representação de generalizações, definindo superfícies de decisão similares. Enquanto os modelos de árvores induzem árvores de decisão com nós e conexões, as regras de decisão induzem regras na forma **SE X ENTÃO Y**, sendo X um conjunto de condições em que cada condição é definida pelo atributo e seus possíveis valores.

Uma vantagem das regras sobre as árvores é que elas, as regras, podem ser interpretadas isoladamente, enquanto que um nó, geralmente, possui nós predecessores que exigem a sua interpretação prévia.

O algoritmo OneR é um dos algoritmos de regras de decisão. A ideia fundamental desse algoritmo é gerar uma árvore de decisão com apenas um nível, criando diversas regras para testar apenas um atributo. Na existência de mais atributos, o algoritmo cria uma regra para cada um deles.

Métodos Baseados em Otimização

Existem métodos de *Machine Learning* que induzem modelos formulando o problema de aprendizagem como um problema de otimização. Esses métodos recorrem a funções de otimização com o propósito de maximizar ou minimizar uma função objetiva.

Os métodos baseados em otimização mais populares são as redes neurais artificiais e as SVMs.

Redes Neurais Artificiais

O cérebro humano possui entre 10 e 500 bilhões de neurônios, organizados em aproximadamente 1000 módulos, sendo cada um com 500 redes neurais.

Cada neurônio pode estar conectado a centenas ou milhares de outros neurônios. As redes biológicas trabalham de forma paralela e fornecem grande rapidez de processamento.

A ideia central das Redes Neurais Artificiais (ou RNAs) é utilizar os neurônios biológicos do Sistema Nervoso humano como inspiração para o desenvolvimento da estrutura e do funcionamento dos neurônios artificiais. Trata-se de um Sistema de Processamento de Informações desenvolvido a partir de modelos matemáticos.

As RNAs são Sistemas Computacionais distribuídos compostos de Unidades de PROCESSAMENTO SIMPLES e densamente interconectados, chamados de neurônios artificiais.

Eles executam fórmulas matemáticas simulando as sinapses biológicas. Cada fórmula possui um conjunto de pesos associados, que podem ser positivos ou negativos, dependendo do comportamento da conexão (excitatório ou inibitório).

O valor do peso é ajustado durante o Processo de Aprendizagem para codificar o conhecimento da rede.

A Figura 2 apresenta um modelo de neurônio artificial. Os terminais de entrada recebem valores, que são combinados com pesos e combinados em uma função f_a .

A saída corresponde à resposta do neurônio:

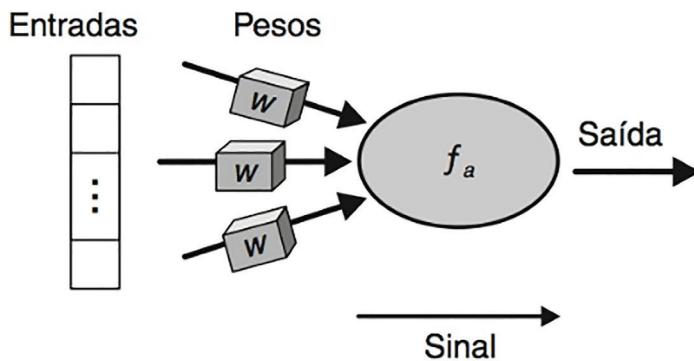


Figura 2 – Estrutura do neurônio artificial

Fonte: Adaptado de Faceli, Katti et al., 2011

Em geral, a combinação das entradas de um vetor \mathbf{x} de d atributos com o vetor de pesos \mathbf{w} pode ser representada por:

$$u = \sum_{i=1}^d \mathbf{x}_i \cdot \mathbf{w}_i$$

Diferentes funções matemáticas podem calcular a função $f_a(u)$. Entre as mais conhecidas, estão:

- **Função linear:** $f_a(u) = u$
- **Função limiar:** $f_a(u) = \begin{cases} 1, & \text{se } u \geq 0 \\ 0, & \text{se } u < 0 \end{cases}$
- **Função sigmoide:** $f_a(u) = \frac{1}{1 + e^{-\lambda u}}$

Um aspecto muito importante no uso de RNAs é a arquitetura da Rede. A combinação dos neurônios artificiais pode fazer com que a Rede esteja disposta em uma ou mais camadas.

Quando duas ou mais camadas são utilizadas, um neurônio pode receber em seus terminais de entrada valores de saída de neurônios da camada anterior e/ou enviar seu valor de saída para terminais de entrada de neurônios da próxima camada.

A topologia de uma rede é formada pelos seguintes elementos:

- O número de camadas;
- O número de neurônios em cada camada;
- O grau de conectividade; e
- A presença ou não de conexões de retropropagação.

O ajuste dos pesos (também conhecidos como parâmetros) de cada neurônio da rede neural, sendo ela de apenas uma camada ou possuindo múltiplas camadas, é o segredo do processo de aprendizagem.

Atualmente, existem muitos algoritmos que realizam o ajuste dos pesos e podem ser categorizados da seguinte forma:

- **Algoritmos de correção de erro:** ajustam os pesos de forma a reduzir os erros cometidos pela Rede;
- **Algoritmos hebbianos:** usam a regra de Hebb (“se dois neurônios estão simultaneamente ativos, a conexão entre eles deve ser reforçada”);
- **Algoritmos competitivos:** promovem a competição entre os neurônios, atualizando os pesos daqueles que respondem mais fortemente ao estímulo apresentado;
- **Algoritmos com base termodinâmica (Boltzmann):** algoritmos estocásticos quem utilizam princípios da metalurgia.

As RNAs fazem parte dos métodos de *Machine Learning* mais estudados nos dias de hoje em razão de suas inúmeras características e áreas de aplicação.

SVMs

As Máquinas de Vetores de Suporte (ou *Support Vector Machines*, SVMs) são baseadas na Teoria do Aprendizado Estatístico (TAE), que estabelece princípios a serem seguidos para a obtenção de classificadores com boa generalização.

Elas foram inicialmente formuladas pelo matemático russo Vladimir Vapnik, em 1995, como um método de aprendizado que tenta encontrar a maior margem para separar diferentes classes de dados.

A ideia geral das SVMs é construir um modelo com um hiperplano ótimo de modo que seja possível separar diferentes classes de dados com a maior margem possível.

Observe a Figura 3 para compreender melhor como as SVMs funcionam.

Na figura, duas classes (quadrados e estrelas) estão representadas em um quadrante gráfico.

Entre as diversas retas (linhas tracejadas) separando os dados que podem ser utilizadas para discriminar as classes, o algoritmo identifica os vetores de suporte, representados na Figura pelos 4 objetos com linhas tracejadas sobre eles, para encontrar a melhor fronteira de decisão (linha vermelha).

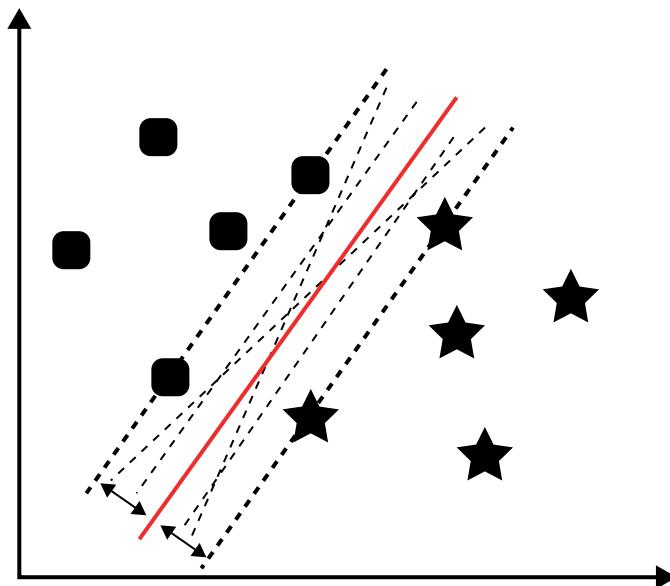


Figura 3 – Modelo SVM

Os vetores de suporte são encontrados durante o Processo de Aprendizagem e definem qual será o hiperplano. O controle dos hiperparâmetros C e γ do método permitem que o algoritmo lide com problemas comuns da aprendizagem supervisionada, como outliers e classificações incorretas.

Embora o tempo de treinamento possa ser bastante demorado dependendo do número de exemplos e da dimensionalidade dos dados, o desempenho preditivo é, em geral, superior aos demais métodos e bastante equivalente às RNAs.

Material Complementar

Indicações para saber mais sobre os assuntos abordados nesta Unidade:



Livros

Inteligência Artificial: uma abordagem de aprendizado de máquina

FACELI, K. *et al.* **Inteligência Artificial: uma abordagem de aprendizado de máquina**. Rio de Janeiro: LTC, 2011.

Redes neurais: princípios e prática

HAYKIN, S. **Redes neurais: princípios e prática**. Bookman, 2007.

Relatório Técnico do Instituto de Ciências Matemáticas e de Computação

LORENA, A. C.; DE CARVALHO, A. CPLF. Introdução às máquinas de vetores suporte. **Relatório Técnico do Instituto de Ciências Matemáticas e de Computação**, USP/São Carlos, v. 192, p. 11, 2003.

Sistemas Inteligentes: Fundamentos e Aplicações

MONARD, M. C.; BARANAUSKAS, J. A. Indução de regras e árvores de decisão. **Sistemas Inteligentes: Fundamentos e Aplicações**, São Paulo, v. 1, p. 115-139, 2003.

Referências

- FACELI, K. et al. **Inteligência Artificial**: Uma abordagem de Aprendizado de Máquina. Rio de Janeiro: LTC, 2011.
- GRABUSTS, P.; BORISOV, A.; ALEKSEJEVA, L. *Ontology-based classification system development methodology*. **Information Technology and Management Science**, Riga, v. 18, n. 1, p. 129-134, 2015.
- MITCHELL, T. M. **Machine learning**. New York: McGraw-Hill, 1997.
- PROVOST, F.; FAWCETT, T. **Data Science para negócios**. Tradução de Marina Boscatto. Rio de Janeiro: Alta Books, 2016.
- RUSSEL, S. J.; NORVIG, P. **Inteligência Artificial**: uma abordagem moderna. 2.ed. Rio de Janeiro: Elsevier, 2004.



Cruzeiro do Sul
Educacional