COM410 | Aprendizado de Máquinas - Revisão

Olá, estudante de Aprendizado de Máquinas! Este material de revisão serve como referência para você se preparar para a prova da disciplina. Ler este material é uma oportunidade de rever alguns dos conteúdos abordados durante todas as semanas da disciplina.

No entanto, é importante que você reveja o conteúdo de cada semana para ter uma preparação mais completa. Embora as questões de prova sejam mais relacionadas à parte teórica, também é importante fazer os exercícios de apoio, assim como exercitar o conteúdo das videoaulas tutoriais para que você esteja preparado(a) para usar essas técnicas de forma efetiva.

Introdução

É cada vez mais comum o uso de técnicas de Aprendizado de Máquinas nas diversas áreas do conhecimento. Hoje, essas técnicas são usadas para reconhecimento de imagens, classificação de conteúdo de texto, predizer classes às quais objetos de um conjunto de dados pertencem, ou ainda para identificar semelhanças e diferenças entre instâncias de um conjunto de dados.

Existem diversas técnicas de aprendizado de máquinas e saber quais são as possíveis técnicas mais adequadas a um conjunto de dados é fundamental para obter bons resultados na análise de dados. Assim, é essencial conhecermos quais são os pontos fortes e fracos das técnicas de aprendizado para as usarmos corretamente.

Em geral, quando fazemos a análise de dados usando aprendizado de máquina, usamos pacotes de algoritmos prontos, que estão disponíveis para diversas linguagens de programação, tais como Python, Java ou C#. Na maioria dos casos, os principais algoritmos já estão disponíveis. Em algumas situações específicas, pode ser necessário fazer alguma customização de um algoritmo, ou mesmo implementar algum que não exista. Independentemente de o algoritmo estar ou não disponível, é fundamental conhecer as técnicas existentes e ter uma boa ideia de como elas devem ser utilizadas para fazer um uso eficiente de aprendizado de máquinas.

Semana 1 - Conceitos básicos de Aprendizado de Máquina / Classificação usando Árvores de Decisão

Assunto 1: Conceitos Básicos de Aprendizado de Máquinas (AM)

Quando estamos criando um modelo de aprendizado de máquina não informamos ao computador os passos a seguir para que ele aprenda o que precisa, isso porque o conhecimento é adquirido, no geral, a partir de modelos estatístico/matemáticos que reconhecem padrões em dados, criando a possibilidade de que eles aprendam com seus erros e façam previsões em cima do que foi aprendido. Esses modelos podem ser definidos por diferentes formas de aprendizado, as quais serão abordadas a seguir.

Tipos de AM:

- Supervisionado
- Não Supervisionado

- Reforço

Aprendizado Supervisionado

O aprendizado supervisionado é um paradigma de aprendizado de máquina, que tem como objetivo adquirir informações de relacionamento entre entrada e saída de um sistema, baseado em um conjunto de amostras de treinamento.

Um algoritmo de aprendizado supervisionado analisa os dados de treino e produz uma função inferida que será utilizada para mapear novos exemplos.

Os dois principais problemas abordados pelo Aprendizado Supervisionado são: Classificação e Regressão.

Classificação

A classificação é o processo de categorizar um determinado conjunto de dados em classes. O caso particular em que temos apenas duas classes (exemplos positivos e exemplos negativos) chama-se de classificação binária. Um exemplo de classificação binária é a tarefa de identificar e-mails como spam, quando o modelo, através dos dados fornecidos, gera como resposta se o e-mail é spam ou não.

Alguns dos algoritmos mais famosos são:

- K-NN
- Naïve Bayes
- Logistic Regression
- Support Vector Machines
- Decision Trees

Regressão

Os modelos de regressão são utilizados quando queremos prever valores, por exemplo, prever o preço de uma casa ou o número de produtos que serão vendidos em determinado mês.

Aprendizado Não Supervisionado

O aprendizado não supervisionado consiste em treinar uma máquina a partir de dados que não estão rotulados e/ou classificados. Os algoritmos que fazem isso buscam descobrir padrões ocultos que agrupam as informações de acordo com semelhanças ou diferenças, por exemplo.

As técnicas de agrupamento e regras de associação são exemplos de implementações de aprendizado não supervisionado.

Agrupamento

A técnica de agrupamento consiste em agrupar dados não rotulados com base em suas semelhanças ou diferenças. Esses algoritmos de agrupamento ainda podem ser subdivididos em agrupamentos exclusivos, sobrepostos, hierárquicos e probabilísticos.

Regras de Associação

Ao usarmos as regras de associação, buscamos descobrir relações que descrevem grandes porções dos dados. A associação é muito utilizada em análises de cestas de compras, no qual a empresa pode tentar entender relações de preferências de compras entre os produtos.

Aprendizado por Reforço

O aprendizado por reforço (ou *Reinforcement Learning* – RL), conhecido como modelo de aprendizado semi-supervisionado em Machine Learning, é uma técnica para permitir que um agente tome ações e interaja com um ambiente, a fim de maximizar as recompensas totais. A cada resultado do modelo existe uma intervenção (auditoria) que valida ou não esse resultado e, a partir daí, o modelo "aprende" novos exemplos.

Assunto 2: Aprendizado para Classificação: Árvores de Decisão

Métodos Simbólicos

Neste paradigma um conceito é representado em uma estrutura simbólica, e o aprendizado é realizado através da apresentação de exemplos e contraexemplos desse conceito. Essas estruturas possibilitam uma interpretação mais direta por seres humanos. A vantagem principal desse tipo de método é uma maior compreensibilidade do processo decisório.

Árvores de Decisão e Regressão

Uma árvore de decisão é um algoritmo de aprendizado de máquina supervisionado que é utilizado para classificação e para regressão. Isto é, pode ser usado para prever categorias discretas (sim ou não, por exemplo) e para prever valores numéricos (o valor do lucro em reais).

Assim como um fluxograma, a árvore de decisão estabelece nós (*decision nodes*) que se relacionam entre si por uma hierarquia. Existe o nó-raiz (*root node*), que é o mais importante, e os nós-folha (*leaf nodes*), que são os resultados finais. No contexto de *machine learning*, o nó-raiz é um dos atributos da base de dados e o nó-folha é a classe ou o valor que será gerado como resposta.

Regras de Divisão por Classificação

O grande trabalho da árvore é justamente encontrar os nós que vão ser encaixados em cada posição. Quem será o nó-raiz? Depois, quem será o nó da esquerda? E o da direita?

Para isso, é preciso realizar alguns importantes cálculos. Uma abordagem comum é usar o ganho de informação e a entropia. Essas duas variáveis dizem respeito à desorganização e falta de uniformidade nos dados. Quanto mais alta a entropia, mais caóticos e misturados estão os dados. Quanto menor a entropia, mais uniforme e homogênea está a base.

Para definir os posicionamentos, é preciso calcular a entropia das classes de saída e o ganho de informação dos atributos da base de dados. Quem tiver maior ganho de informação entre os atributos é o nó-raiz. Para calcular a esquerda e a direita, deve-se realizar novos cálculos de entropia e ganho com o conjunto de dados que atende à condição que leva à esquerda ou à direita.

Entropia

A entropia de um conjunto pode ser definida como sendo o grau de pureza desse conjunto. Esse conceito emprestado pela Teoria da Informação define a medida de "falta de informação", mais precisamente o número de bits necessários, em média, para representar a informação em falta, usando codificação ótima.

Dada uma coleção S contendo exemplos positivos (+) e negativos (-) de algum conceito-alvo, a entropia de S relativa a essa classificação booleana é:

$$Entropia\ (S) \equiv -p_+ \log_2 p_+ - p_- \log_2 p_-$$

Onde:

P+ é a proporção de exemplos positivos em S

P- é a proporção de exemplos negativos em S

Ganho de Informação

Ganho de Informação: redução esperada no valor da Entropia, devido à ordenação no conjunto de treino segundo os valores do atributo A.

Regras de Divisão por Regressão

As árvores de regressão são usadas quando a variável dependente é contínua, diferentemente das árvores de classificação que são usadas quando a variável dependente é categórica.

No caso da árvore de regressão, o valor obtido pelos nós de término nos dados de treinamento é o valor médio das suas observações. Assim, a uma nova observação de dados atribui-se o valor médio correspondente.

Ambas as árvores dividem o espaço preditor (variáveis independentes) em regiões distintas e não sobrepostas.

Para a realização das partições em uma árvore de regressão utiliza-se a métrica de redução da variância, definida pelas fórmulas abaixo:

$$sd(\mathbf{D}, \mathbf{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y})^2}$$

$$SDR\,(h_A) = sd(\mathbf{D}, \mathbf{y}) - \frac{n_L}{n} \times sd(\mathbf{D}_L, \mathbf{y}) - \frac{n_R}{n} \times sd(\mathbf{D}_R, \mathbf{y})$$

Regras de Decisão

Uma regra de decisão é uma implicação da forma: **se A então B**. A parte condicional A é uma conjunção de condições. Cada condição é definida por uma relação entre um atributo e os valores do domínio.

Vantagens de converter uma árvore de decisão em regras antes da poda:

- Permite distinguir entre os diferentes contextos em que os nós de decisão são utilizados.
- Remove a distinção entre atributos de testes que ocorrem próximos da raiz da árvore e aqueles que ocorrem próximos das folhas.
- Melhora a leitura humana.
- Regras são geralmente mais fáceis para pessoas entenderem.

Semana 2 - Aprendizado de Regras de Classificação

Assunto 1: Regras de Classificação

Uma regra de decisão é uma implicação da forma: **Se A então B**

Parte condicional A: conjunção de condições

Cada condição é relação entre um atributo e valores.

Relações: =, ≠, <, >, <=, >=, ∈, ∉

Ex. Atributoi = vi, Atributoi < vi, Atributoi ∈ Conjunto_valores etc.

Ex. Tempo = Ensolarado ^ Umidade <= 75 ⇒ Jogar = Sim

Parte Consequente B: rotula os exemplos

Tal como nas árvores de decisão, o conjunto de regras é disjunto (FND).

Regras de decisão e árvores de decisão são bastante similares em suas formas de representação para expressar generalizações dos exemplos. Ambas definem superfícies de decisão similares. As superfícies de decisão definidas pelas regras de decisão correspondem a hiper-retângulos no espaço definido pelos atributos.

Árvores de Decisão para Regras de Decisão

Qualquer árvore de decisão pode ser facilmente reescrita em um conjunto de regras de decisão. Cada regra corresponde a um percurso desde a raiz da árvore até uma folha. Existem tantas regras quanto as folhas da árvore de decisão. Esse processo gera um conjunto de regras com a mesma complexidade da árvore de decisão. Contudo, alguns antecedentes em regras consideradas individualmente podem conter condições irrelevantes. O algoritmo C4.5 usa um processo de otimização para simplificar o conjunto de regras, removendo condições irrelevantes.

O processo de otimização consiste em duas fases. Primeiro, cada regra é generalizada pela eliminação de condições que não contribuem para discriminar as classes. É utilizada uma procura gulosa, em que a cada passo a regra é avaliada removendo uma das condições. A

condição que produz um aumento menor da estimativa pessimista da taxa de erro é eliminada. Para obter a estimativa pessimista da taxa de erro, é utilizado um processo semelhante ao mecanismo de poda usado pelo C4.5. Após a generalização individual das regras, são removidas regras idênticas e as regras sem parte condicional. Em uma segunda fase, as regras são agrupadas pela classe que preveem. Para cada classe, o conjunto de regras é simplificado, eliminando as regras que não contribuem para a taxa de acerto do conjunto. O estudo experimental apresentado em Quinlan (1993) mostra que as regras de decisão são mais simples e com menor taxa de erro do que a árvore de decisão a partir da qual foram geradas.

Algoritmo de Cobertura

Tipicamente, o algoritmo procura regras da forma:

Se $\text{Atributo}_i = \text{valor}_j$ e $\text{Atributo}_l = \text{valor}_k$ ... Então Classe_z

Entrada: Um conjunto de treinamento $D = \{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$

Saída: Um conjunto de regras: **Regras**

```
1  Regras ← {};
2  Seja Y o conjunto das classes em D;
3  seja cada y_i ∈ Y faça
4     repita
5        Regra = Aprende_Uma_Regra(D,y_i);
6        Regras ← Regras ∪ {Regra}
7        D ← Remove exemplos cobertos pela Regra em D;
8     até não haver exemplos de y_i;
9  fim
10 Retorna: Regras;
```

Estratégias

Top-down: inicia a busca da regra mais geral, {} → Classe, e aplica operadores de especificação, acrescentando condições à parte condicional da regra (orientada pelo modelo).

Bottom-up: começa pela regra mais específica (é escolhido um dos exemplos aleatoriamente, o que implica restrições em todos os atributos) e aplica operadores de generalização, removendo restrições (orientada a dados).

Os modelos de decisão estudados neste capítulo, árvores e regras de decisão, estão entre os mais utilizados em aprendizado de máquina. Atualmente, existem algoritmos eficientes para indução de árvores de decisão ou conjuntos de regras e de aplicação eficiente, com um desempenho equivalente ao de outros modelos (como redes neurais e SVM), mas com maior grau de interpretabilidade. No entanto, as regras de decisão são de alguma forma mais expressivas e flexíveis para representar conhecimento.

Assunto 2: Algoritmos de Regras de Classificação

Sistemas Baseados em Regras

Os Sistemas Baseados em Conhecimento (SBCs) ou Sistemas Baseados em Regras têm como principais características: uma base de conhecimentos e um mecanismo de raciocínio para realizar inferências sobre essa base e extrair conclusões a partir desses conhecimentos.

Tipos de Encadeamentos

Encadeamento para frente é um sistema baseado em regras que executa encadeamento para frente e utiliza assertivas dadas e inferidas para deduzir novas assertivas ou realizar uma ação. O processamento se dá partindo das assertivas para chegar a uma conclusão.

Encadeamento para trás é um sistema baseado em regras que executa encadeamento para trás e transforma uma assertiva que se quer provar ou uma ação que se quer tomar em uma hipótese. A partir daí, usa-se as regras retroativamente para analisar as assertivas que suportam a hipótese em questão.

Sistemas Especialistas

Sistemas que empregam o conhecimento humano para resolver problemas que requererem a presença de um especialista. Atualmente, consiste em uma das áreas de aplicação de mais sucesso da IA.

Um sistema especialista (SE) é desenvolvido a partir da necessidade de se processar informações não numéricas, é capaz de apresentar conclusões sobre um determinado tema, desde que devidamente orientado e "alimentado".

Um sistema especialista é uma forma de sistema baseado no conhecimento especialmente projetado para emular a especialização humana de algum domínio específico. Um SE irá possuir uma base de conhecimento formada de fatos e regras sobre o domínio, tal como um especialista humano faria, e deve ser capaz de oferecer sugestões e conselhos aos usuários.

Semana 3 - Algoritmos Probabilísticos

Assunto 1: Aprendizado Probabilístico

Paradigma Estatístico

No paradigma estatístico é utilizado um modelo estatístico que encontre uma hipótese que possua uma boa aproximação do conceito a ser induzido. O aprendizado consiste em encontrar os melhores parâmetros para o modelo. Esses modelos podem ser paramétricos (quando fazem alguma suposição sobre a distribuição dos dados, ou podem ser não paramétricos, quando não fazem suposição sobre a distribuição dos dados.

Dentre os modelos estatísticos utilizados em aprendizagem de máquina podemos destacar os modelos Bayesianos.

Aprendizagem Bayesiana

O pensamento bayesiano fornece uma abordagem probabilística para a aprendizagem. Está baseado na suposição de que as quantidades de interesse são reguladas por distribuições de probabilidades. Utiliza-se da estratégia de quantificar o custo/benefício entre diferentes decisões de classificação usando probabilidades e custos associados à classificação.

Teorema de Bayes

Mostra como alterar as probabilidades *a priori* tendo em conta novas evidências de forma a obter probabilidades *a posteriori*.

Basicamente, o teorema de Bayes mostra como rever as crenças sempre que novas evidências são coletadas. Ou seja, atualizar a probabilidade *a posteriori* utilizando para isso a probabilidade *a priori*, as verossimilhanças e as evidências.

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

$$P(A \mid B)P(B) = P(A \cap B) = P(B \mid A)P(A)$$

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

$P(A|B)$ é a probabilidade a posteriori
$P(A)$ é a probabilidade a priori
$P(B|A)$ são as verossimilhanças $(likelihood)$
$P(B)|$ são as evidências, dado por $\sum P(A_i) \times P(B|A_i)$

Classificador Naïve-Bayes

Um dos algoritmos de aprendizagem mais práticos e utilizados na literatura. Denominado Naïve (ingênuo) por assumir que os atributos são condicionalmente independentes, ou seja, a informação de um evento não é informativa sobre nenhum outro.

Apesar dessa premissa, o classificador reporta bom desempenho em diversas tarefas de classificação em que há dependência.

Aplicações bem-sucedidas:

– Diagnóstico médico

– Classificação de documentos textuais

Aplica-se em tarefas de aprendizagem em que cada instância **x** é descrita por uma conjunção de valores de atributos em que a função alvo, f(**x**), pode assumir qualquer valor de um conjunto. Um conjunto de exemplos de treinamento da função alvo é fornecido. E então uma nova instância é apresentada, descrita pela tupla de valores de atributos.

A tarefa é predizer o valor alvo (ou classificação) para esta nova instância.

O classificador é baseado na suposição de que os valores dos atributos são condicionalmente independentes dados o valor alvo.

Se usarmos Bayes para múltiplas evidências, temos:

$$P(H|E_1, E_2, \ldots, E_n) = \frac{P(E_1,E_2,\ldots,E_n|H)\times P(H)}{P(E_1,E_2,\ldots,E_n)}$$

Considerando a hipótese de independência, podemos reescrever o teorema de Bayes da seguinte forma:

$$P(H|E_1, E_2, \ldots, E_n) = \frac{P(E_1|H)\times,P(E_2|H)\times,\ldots,P(E_n|H)\times P(H)}{P(E_1,E_2,\ldots,E_n)}$$

O denominador pode ser ignorado por se tratar de um termo comum.

Assunto 2: Redes Bayesianas para Classificação

As redes bayesianas são grafos que representam relações de probabilidade condicional, ou seja, como que a ocorrência de certas variáveis depende do estado de outra. Elas foram desenvolvidas no início dos anos 1980 para facilitar a tarefa de predição e “abdução” em sistemas de inteligência artificial (AI).

Redes bayesianas são modelos de representação do conhecimento que trabalham com o conhecimento incerto e incompleto por meio do Teorema de Bayes, publicado pelo matemático Thomas Bayes em 1763.

Matematicamente, uma rede bayesiana é uma representação compacta de uma tabela de conjunção de probabilidades do universo do problema. Do ponto de vista de um especialista, redes bayesianas constituem um modelo gráfico que representa de forma simples as relações de causalidade das variáveis de um sistema.

<u>Método de Construção</u>

- Escolha um conjunto de variáveis Xi que descrevam o domínio.
- Escolha uma ordem para as variáveis.
- Enquanto existir variáveis:
    - Escolha uma variável Xi e adicione um nó na rede.
    - Determine os nós Pais(Xi) dentre os nós que já estejam na rede e que tenham influência direta em Xi.
    - Defina a tabela de probabilidades condicionais para Xi.

<u>Tipos de Conhecimento</u>

- Causal.

Refletem a direção conhecida de causalidade no mundo: para algumas propriedades do mundo percepções são geradas.

Exemplo: P(DorDeDente|Cárie), P(MariaLiga|Alarme)

- Diagnóstico.

Infere a presença de propriedades escondidas diretamente da percepção.

Produzem conclusões fracas.

Exemplo: P(Cárie|DorDeDente), P(Alarme|MariaLiga)

Aprendizagem

A aprendizagem bayesiana pode ser vista como uma forma de obter a representação interna da rede que define um dado domínio de modo a facilitar a extração do conhecimento.

O processo de aprendizagem é dividido em duas partes:

- aprendizagem da estrutura (e relações entre as variáveis);
- aprendizagem dos parâmetros numéricos (distribuição de probabilidade).

Formas de Aprendizagem

- Especialista;
- Indutiva.

Aprendizagem dos parâmetros

1. Podem ser estimados através das frequências relativas, caso exista uma quantidade de dados significativa de uma amostra aleatória.

Aprendizagem de estrutura:

1. Métodos de Verossimilhança Máxima;
2. Métodos de Teste de Hipóteses;
3. Métodos de Verossimilhança Estendidos;
4. Métodos "*Minimum Information Complexity*";
5. Métodos "*Resampling*";
6. Métodos Bayesianos, destacando o clássico algoritmo K2 (COOPER; HERSKOVITS, 1992).

Considerações Finais:

- Redes bayesianas constituem uma forma natural para representação de informações condicionalmente independentes.

- Boa solução a problemas em que conclusões não podem ser obtidas apenas com o domínio do problema.
- Combina conhecimento *a priori* com dados observados.
- O impacto do conhecimento *a priori* (quando correto) é a redução da amostra de dados necessários.
- Inferências sobre redes bayesianas.
    - Podem ser executadas em tempo linear.
    - NP-hard para maioria dos casos.
    - Aplicação de técnicas.

Semana 4 - Aprendizado Baseado em Maximização de Margens

As máquinas de vetores de suporte (SVMs - *Support Vector Machines*) são um tipo de algoritmo de aprendizado de máquina que utiliza a estratégia de maximização de margens para separar o espaço entre dados com características distintas, como ilustrado na Figura 4.1. Para obter a maior margem entre duas classes, vários hiperplanos são gerados na fronteira entre os dados, e então os objetos mais próximos da fronteira desses hiperplanos são utilizados como referência para a identificação das margens. Portanto, esses objetos dão **suporte** ao hiperplano de separação. A escolha da melhor margem se dá por aquele cuja **fronteira de decisão** está mais distante desses vetores de suporte.

Maximizar as margens é uma forma de aumentar o espaço de decisão entre as classes. Uma vez que temos o modelo classificador treinado, novos dados que sejam classificados terão maior chance de serem classificados de forma correta.

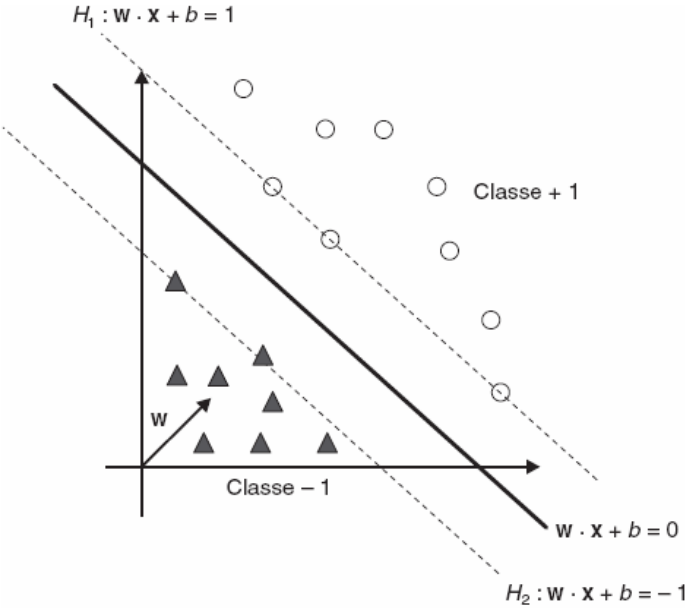


Figura 4.1: Hiperplanos de separação da SVM – Fonte: Faceli *et al*. (2020)

SVM são comumente usadas em tarefas de classificação, mas também podem ser usadas para regressão. Além disso, SVMs costumam ser usadas quando não temos um conhecimento aprofundado sobre os dados. SVMs funcionam bem quando temos uma alta dimensionalidade dos dados.

As SVMs podem ser **lineares** ou **não lineares**. As lineares assumem que os dados podem ser linearmente separáveis, e podem ter **margens rígidas** ou **suaves**. Quando os dados de classes diferentes estão perfeitamente separados, podemos usar as SVMs lineares de margens rígidas e obter uma classificação adequada dos dados. Acontece que, na maioria dos casos, os dados de classes diferentes não estão perfeitamente separados. Nesses casos, pode ser melhor usar SVMs com margens suaves, que vão permitir ajustar uma quantidade de objetos que possam ser classificados incorretamente, ou seja, permitem uma certa taxa de erro de classificação.

Pode haver outros casos em que os conjuntos de dados de classes distintas estejam distribuídos em espaços que não podem ser naturalmente separados por um hiperplano. Nesse caso, podemos usar SVMs não lineares. Uma característica importante desse tipo de SVM é o uso de **funções Kernel** para conseguir uma melhor separação dos dados. O uso dessas funções é conhecido como **truque de Kernel**. Em geral, o objetivo dessas funções é aumentar a dimensionalidade dos espaços de dados até que seja possível encontrar uma fronteira possível de divisão dos dados. Existem diferentes tipos de funções de Kernel: sigmoidal, RBF (função de base radial), polinomiais e cada uma possui parâmetros que devem ser alterados para obter resultados melhores de classificação. Na prática, ao usar SVM, é comum treinarmos usando as diferentes funções, testando diferentes valores dos parâmetros até obter um resultado satisfatório. A Figura 4.2 mostra um exemplo de aumento de dimensionalidade nos dados, de duas para três dimensões, para realizar a separação de dados não lineares.

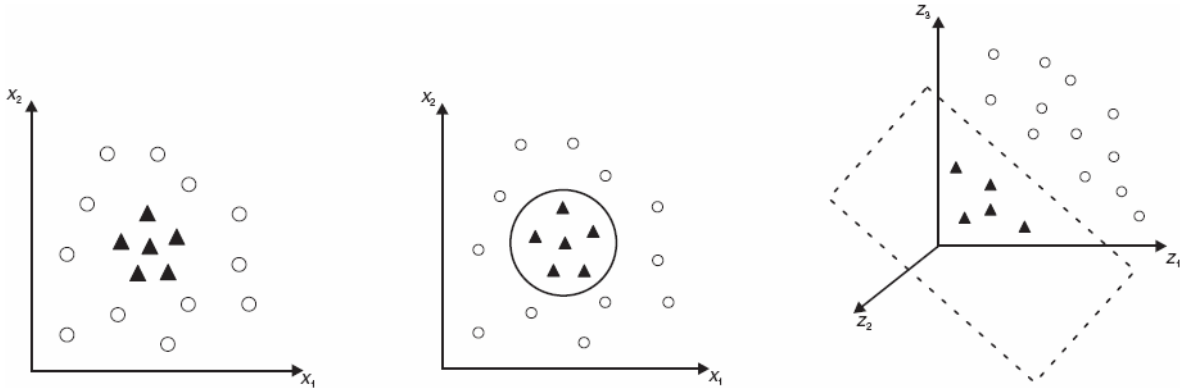


Figura 4.2: Transformação de duas para três dimensões – Fonte: Faceli *et al*. (2020)

Já quando lidamos com problemas de regressão, a fronteira de decisão e as margens são usadas para definir um limiar no qual os dados devem estar presentes, geralmente acompanhando a função linear que representa os dados. Portanto, as margens contêm os dados, ao contrário das SVMs de classificação, nas quais a região entre as margens e a fronteira de decisão costumam ser vazias. A Figura 4.3 exemplifica a estrutura de SVM para regressão, com as margens definidas entre [- ε, + ε].

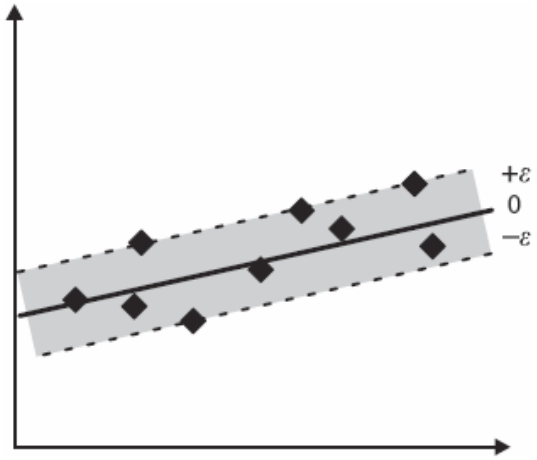


Figura 4.3: Exemplo de SVM para regressão – Fonte: Faceli *et al*. (2020)

Embora os algoritmos SVM sejam amplamente usados para diversas tarefas de classificação e regressão, uma desvantagem que esses algoritmos apresentam é a dificuldade em entender a estrutura dos modelos gerados, uma vez que não é possível visualizar a distribuição dos dados em alta dimensionalidade. Por isso, esses algoritmos são chamados de caixa-preta.

Semana 5 - Agrupamento de Dados/Aprendizado Baseado em Distância

O agrupamento de dados é um conjunto de técnicas de aprendizado de máquinas que nos permite agrupar dados de acordo com as suas características. Os algoritmos de agrupamento fazem parte dos **modelos descritivos**, cuja principal característica é a ausência de rótulos nos dados. Como não sabemos *a priori* quais são as possíveis categorias, ou grupos, às quais os dados pertencem, é necessário identificar semelhanças e diferenças nos valores dos atributos para agrupá-los com os seus semelhantes e, ao mesmo tempo, separá-los dos dados mais distintos.

Dependendo das características que são usadas para o agrupamento, os mesmos dados podem formar grupos totalmente distintos. Por exemplo, na Figura 5.1, os dados podem ser agrupados pela forma, pelo preenchimento, ou ainda pelo preenchimento e pela forma simultaneamente. Isso vai depender do objetivo do agrupamento, o que muitas vezes será definido por algum especialista no domínio dos dados.

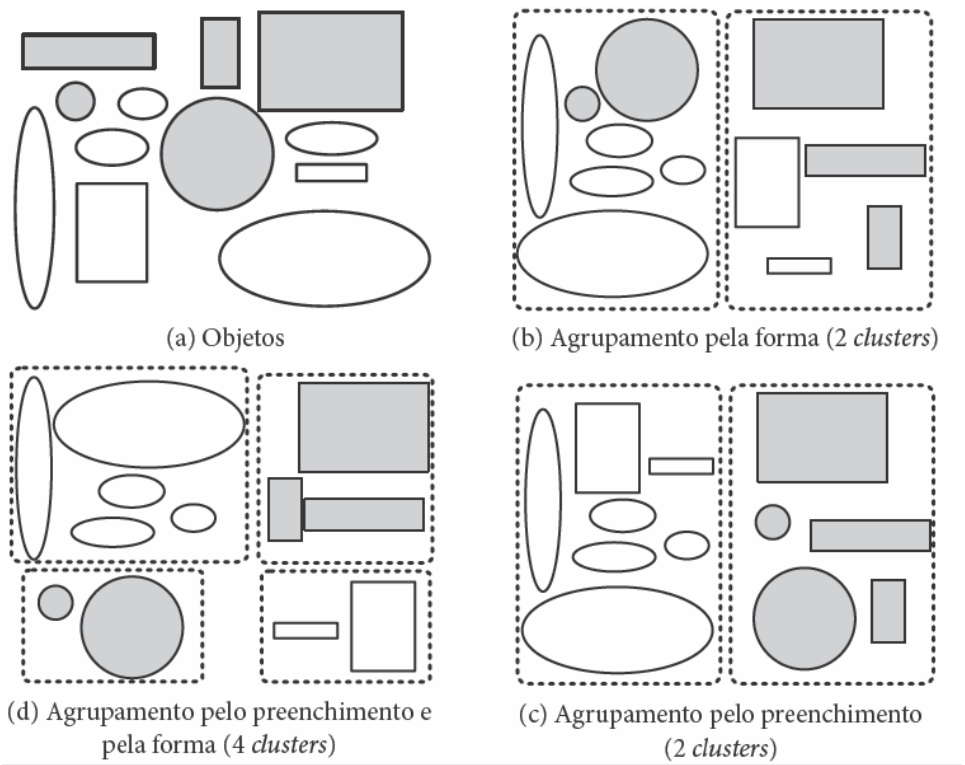


Figura 5.1: Diferentes agrupamentos para o mesmo conjunto de dados – Fonte: Faceli *et al*. (2020)

Muitas técnicas de agrupamento precisam que seja informado qual é o número de grupos que os dados formam. É comum usarmos o termo em inglês, *cluster*, para nos referirmos a esses grupos. Formalmente, um cluster é um conjunto de objetos que possuem uma similaridade que os une. Essa união é obtida através dos **critérios** utilizados para realizar o agrupamento (por exemplo, a seleção de determinados atributos de interesse).

Em alguns casos, o especialista do domínio pode saber quantos grupos existem. Quando não é possível saber a quantidade de grupos, o que geralmente fazemos é testar diferentes tamanhos de grupos, até achar algum que gere um agrupamento satisfatório, o que pode ser avaliado pelo especialista ou então usando índices de validação de agrupamento, como os vistos durante a Semana 7 desta disciplina.

Existem três categorias principais de critérios de agrupamento, que correspondem ao tipo de característica que representam:

- Compactação: usados quando há bastante proximidade entre os dados de um *cluster* e *clusters* com formato esférico.
- Encadeamento ou ligação: usados quando os clusters são formados por vizinhança, funcionando bem para clusters de formas arbitrárias.
- Separação espacial: usados em conjunto com os outros critérios e funciona bem quando há uma boa separação entre os diferentes *clusters*.

Nas tarefas de agrupamento, as **medidas de distância ou similaridade** são fundamentais para a identificação e formação dos clusters. As distâncias podem ser calculadas entre objetos, entre *clusters* ou ainda entre objetos e *clusters*. Existem medidas para diferentes tipos de dados: nominais, binários, discretos ou contínuos. Entre as medidas mais utilizadas estão a distância euclidiana e a distância de Manhattan, para dados contínuos, e a distância de Hamming, para dados nominais.

Há diversos algoritmos de agrupamento, que são categorizados de acordo com algum critério de agrupamento, o que serve como um indicativo dos tipos de estrutura mais adequados para seu uso. A seguir, são apresentadas as principais categorias, com uma breve descrição de cada uma e alguns dos seus principais algoritmos.

- Algoritmos hierárquicos: formam clusters de forma aglomerativa ou divisiva, usando métricas de integração. Funcionam bem com distribuições de dados heterogêneas.
    - Ligação simples, ligação completa
- Algoritmos particionais baseados em erro quadrático: geram clusters de forma iterativa, criando uma partição inicial de dados que é atualizada a cada novo objeto avaliado. São computacionalmente eficientes e com bom desempenho para *clusters* compactos.
    - k-médias, k-medoide, CLARA, PAM
- Algoritmos baseados em densidade: identificam *clusters* como sendo regiões de alta densidade de objetos. Por consequência, regiões de baixa densidade são áreas que dividem *clusters*.
    - DENCLUE, DBSCAN

- Algoritmos baseados em grafo: usa grafos para representar a distância entre objetos. Os nós representam os objetos e as arestas representam a distância entre os nós.
    - HSC, CLICK
- Algoritmos baseados em redes neurais: são redes neurais auto-organizáveis, nas quais os pesos dos neurônios se adaptam aos dados para gerar modelos de agrupamento.
    - SOM, GCS, SOTA
- Algoritmos baseados em grid: definem uma grade para o espaço de dados, adaptando o tamanho das células da grade de acordo com a distribuição dos dados. Apresentam bom desempenho para dados volumosos.
    - CLIQUE, MAFIA

Semana 6 - Algoritmos Genéticos

Assunto 1: Conceitos Básicos de Algoritmos Genéticos (AGs)

Os problemas de otimização são baseados em três pontos principais: a codificação do problema, a função objetivo que se deseja maximizar ou minimizar e o espaço de soluções associado.

Os algoritmos genéticos são uma família de modelos computacionais inspirados na evolução, que incorporam uma solução potencial para um problema específico numa estrutura semelhante à de um cromossomo e aplicam operadores de seleção e "*crossover*" a essas estruturas de forma a preservar informações críticas relativas à solução do problema. Normalmente, os AGs são vistos como otimizadores de funções, embora a quantidade de problemas para o qual os AGs se aplicam seja bastante abrangente.

Uma das vantagens de um algoritmo genético é a simplificação que eles permitem na formulação e solução de problemas de otimização. AGs simples normalmente trabalham com descrições de entrada formadas por cadeias de bits de tamanho fixo. O AG é indicado para a solução de problemas de otimização complexos, NP-Completos, como o "caixeiro viajante", que envolvem um grande número de variáveis e, consequentemente, espaços de soluções de dimensões elevadas. Além disso, em muitos casos em que outras estratégias de otimização falham na busca de uma solução, os AGs convergem. Os AGs são numericamente robustos, ou seja, não são sensíveis a erros de arredondamento no que se refere aos seus resultados finais.

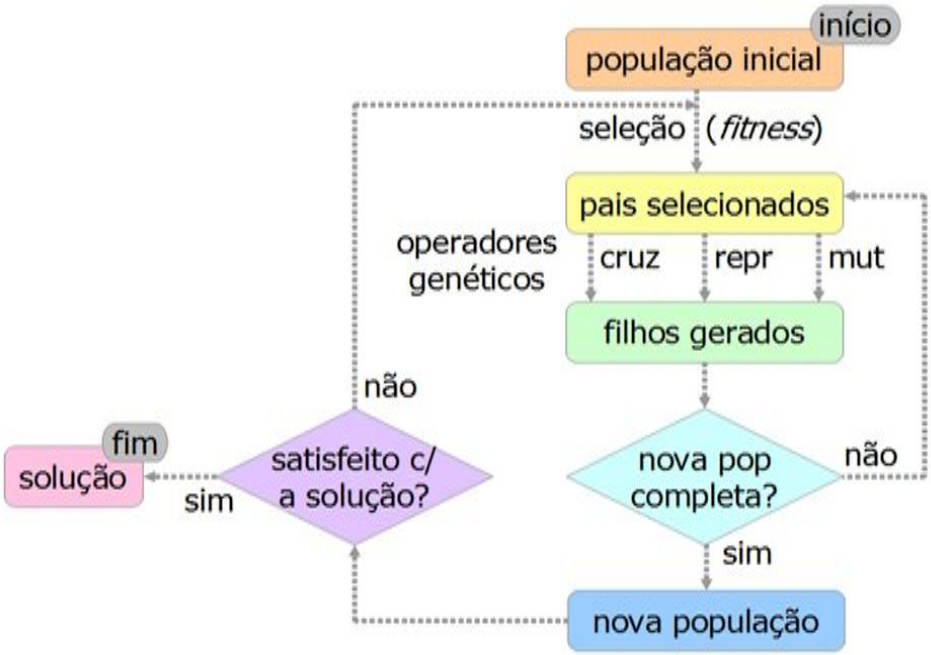
Existem três tipos de representação possíveis para os cromossomos: binária, inteira ou real. A essa representação se dá o nome de alfabeto do AG. De acordo com a classe de problema que se deseja resolver, pode-se usar qualquer um dos três tipos.

Uma implementação de um algoritmo genético começa com uma população aleatória de cromossomos. Essas estruturas são, então, avaliadas e associadas a uma probabilidade de reprodução de tal forma que as maiores probabilidades são associadas aos cromossomos que representam uma melhor solução para o problema de otimização do que àqueles que representam uma solução pior. A aptidão da solução é tipicamente definida com relação à população corrente.

A função objetivo de um problema de otimização é construída a partir dos parâmetros envolvidos no problema. Ela fornece uma medida da proximidade da solução em relação a um conjunto de parâmetros. Os parâmetros podem ser conflitantes, ou seja, quando um aumenta o outro diminui.

O objetivo é encontrar o ponto ótimo. A função objetivo permite o cálculo da aptidão bruta de cada indivíduo, que fornecerá o valor a ser usado para o cálculo de sua probabilidade de ser selecionado para reprodução.

Estrutura Básica de um AG



INICIALIZAÇÃO

Uma população de n indivíduos é gerada aleatoriamente. Cada um dos indivíduos da população representa uma possível solução para o problema, ou seja, um ponto no espaço de soluções.

CÁLCULO DA APTIDÃO

Geralmente a aptidão do indivíduo é determinada através do cálculo da função objetivo, que depende das especificações de projeto. Neste trabalho, cada indivíduo é uma entrada para uma ferramenta de análise de desempenho cuja saída fornece medidas que permitem ao algoritmo genético o cálculo da aptidão do indivíduo. Ainda nesta fase os indivíduos são ordenados conforme a sua aptidão.

SELEÇÃO

Nesta fase os indivíduos mais aptos da geração atual são selecionados. Esses indivíduos são utilizados para gerar uma nova população por cruzamento. Cada indivíduo tem uma probabilidade de ser selecionado proporcional à sua aptidão. Para visualizar este método, considere um círculo dividido em n regiões (tamanho da população), a área de cada região é proporcional à aptidão do indivíduo. Coloca-se sobre esse círculo uma "roleta" com n cursores, igualmente espaçados. Após um giro da roleta, a posição dos cursores indica os indivíduos selecionados. Este método é denominado amostragem universal estocástica. Evidentemente, os indivíduos cujas regiões possuem maior área terão maior probabilidade de serem selecionados

várias vezes. Como consequência, a seleção de indivíduos pode conter várias cópias de um mesmo indivíduo enquanto outros podem desaparecer.

CRUZAMENTO (CROSSOVER)

Os indivíduos selecionados na etapa anterior são cruzados da seguinte forma: a lista de indivíduos selecionados é embaralhada aleatoriamente, criando, dessa forma, uma segunda lista, chamada de lista de parceiros. Cada indivíduo selecionado é então cruzado com o indivíduo que ocupa a mesma posição na lista de parceiros. Os cromossomos de cada par de indivíduos a serem cruzados são particionados em um ponto, chamado de ponto de corte, sorteado aleatoriamente. Um novo cromossomo é gerado permutando-se a metade inicial de um cromossomo coma metade final do outro.

MUTAÇÃO

A operação de mutação é utilizada para garantir uma maior varredura do espaço de estados e evitar que o algoritmo genético convirja muito cedo para mínimos locais. A mutação é efetuada alterando-se o valor de um gene de um indivíduo sorteado aleatoriamente com uma determinada probabilidade, denominada probabilidade de mutação, ou seja, vários indivíduos da nova população podem ter um de seus genes alterado aleatoriamente.

Escolha dos Parâmetros do AG

Além da forma como o cromossomo é codificado, existem vários parâmetros do algoritmo genético que podem ser escolhidos para melhorar o seu desempenho, adaptando-o às características particulares de determinadas classes de problemas. Entre eles os mais importantes são: o tamanho da população, o número de gerações, a probabilidade de *crossover* e a probabilidade de mutação. A influência de cada parâmetro no desempenho do algoritmo depende da classe de problemas que se está tratando. Assim, a determinação de um conjunto de valores otimizado para esses parâmetros dependerá da realização de um grande número de experimentos e testes. Na maioria da literatura os valores encontrados estão na faixa de 60 a 65% para a probabilidade de *crossover* e entre 0,1 e 5% para a probabilidade de mutação. O tamanho da população e o número de gerações dependem da complexidade do problema de otimização e devem ser determinados experimentalmente. No entanto, deve ser observado que o tamanho da população e o número de gerações definem diretamente o tamanho do espaço de busca a ser coberto.

Aplicações

Os AGs possuem uma larga aplicação em muitas áreas científicas, entre as quais podem ser destacadas:

- Síntese de circuitos analógicos: para uma certa entrada e uma saída desejada, por exemplo tensão, o AG gera a topologia, o tipo e o valor dos componentes do circuito.
- Síntese de protocolos: determinação de quais funções do protocolo devem ser implementadas em hardware e quais devem ser implementadas em software para que um certo desempenho seja alcançado.

- Programação genética: gera a listagem de um programa, numa determinada linguagem especificada, para que um determinado conjunto de dados de entrada forneça uma saída desejada.
- Gerenciamento de redes: supervisão do tráfego nos links e das filas nos buffers de roteadores para descobrir rotas ótimas e para reconfigurar as rotas existentes no caso de falha de algum link.
- Computação evolutiva: gera programas que se adaptam a mudanças no sistema ao longo do tempo.
- Otimização evolutiva multicritério: otimização de funções com múltiplos objetivos que sejam conflitantes.
- Problemas de otimização complexos: problemas com muitas variáveis e espaços de soluções de dimensões elevadas. Ex.: problema do caixeiro viajante, gerenciamento de carteiras de fundos de investimento.
- Ciências biológicas: modela processos biológicos para o entendimento do comportamento de estruturas genéticas.
- Autômatos autoprogramáveis.

Conclusões

Os AGs são apropriados para problemas de otimização complexos, que envolvem muitas variáveis e um espaço de soluções de dimensão elevada. Abrangem um grande número de aplicações. O controle sobre os parâmetros do algoritmo é de fundamental importância para uma convergência rápida. Para problemas específicos é aconselhável a utilização de algoritmos híbridos, que misturam as técnicas dos AGs com os métodos de otimização tradicionais.

Devido ao grande número de variáveis que um AG trata e às populações elevadas e alto número de gerações para a cobertura do espaço de soluções, os AGs possuem um custo computacional elevado.

Assunto 2: Aplicação de AG no Problema do Caixeiro Viajante

O problema do caixeiro é um clássico exemplo de problema de otimização combinatória. A primeira coisa que podemos pensar para resolver esse tipo de problema é reduzi-lo a um problema de enumeração: achamos todas as rotas possíveis e, usando um computador, calculamos o comprimento de cada uma delas e então vemos qual a menor (É claro que se acharmos todas as rotas estaremos contando-as, daí podermos dizer que estamos reduzindo o problema de otimização a um de enumeração).

Para acharmos o número R(n) de rotas para o caso de n cidades, basta fazer um raciocínio combinatório simples e clássico. Por exemplo, no caso de n = 4 cidades, a primeira e última posição são fixas, de modo que elas não afetam o cálculo; na segunda posição podemos colocar qualquer uma das 3 cidades restantes B, C e D, e uma vez escolhida uma delas, podemos colocar qualquer uma das 2 restantes na terceira posição; na quarta posição não teríamos nenhuma escolha, pois sobrou apenas uma cidade; consequentemente, o número de rotas é 3 x 2 x 1 = 6, resultado que tínhamos obtido antes contando diretamente a lista de rotas acima.

De modo semelhante, para o caso de n cidades, como a primeira é fixa, o leitor não terá nenhuma dificuldade em ver que o número total de escolhas que podemos fazer é (n-1) x (n-2) x ... x 2 x 1. De modo que, usando a notação de fatorial: R(n) = (n - 1)!.

Solução com Algoritmos Genéticos

Indivíduos:

O cromossomo será constituído de um vetor de valores (de 1 a n), que corresponde ao número de cada cidade.

Função de avaliação: minimizar o valor

f(x) = soma das distâncias entre as cidades

A seleção da população será realizada por meio do fitness.

Semana 7 - Estimação de Acurácia e Comparação de Modelos

Para entender qual é o desempenho dos algoritmos de aprendizado de máquina, é fundamental avaliarmos os resultados obtidos pelo modelo de classificação ou de agrupamento. Isso nos permite verificar a necessidade de fazer ajustes nos parâmetros de um algoritmo quando o desempenho não é o esperado, ou ainda trocar o algoritmo. Existem várias métricas e índices que podem ser usados para avaliar e comparar modelos a partir dos resultados obtidos.

Nas tarefas preditivas, de classificação e regressão, o desempenho pode ser avaliado comparando os resultados de classificação obtidos com os rótulos reais do conjunto de dados de teste. Entre as principais medidas de classificação, temos:

- **Erro**: proporção de objetos classificados incorretamente.
- **Acurácia**: é o complemento do erro, indicando a proporção de objetos classificados corretamente.

As métricas de regressão medem a distância entre os objetos conhecidos e os valores preditos. Entre as métricas de regressão, temos:

- **Erro quadrático médio (MSE)**: soma do quadrado das distâncias entre os objetos conhecidos e preditos.
- **Distância absoluta média (MAD)**: soma das distâncias entre os objetos conhecidos e preditos.

Quando a classificação é binária, podemos usar a notação da matriz de confusão para extrair outras métricas importantes para a avaliação de desempenho. A matriz de confusão para um dado algoritmos indica a quantidade de verdadeiros positivos e negativos (VP e VN, respectivamente) e a quantidade falsos positivos e negativos (FP, FN) considerando todos os objetos avaliados. Com esses valores, podemos calcular as métricas de acurácia e erro, descritas acima, e ainda outras métricas relacionadas:

- **Precisão**: proporção de objetos positivos classificados corretamente em relação a todos objetos positivos existentes.
- **Revocação**: proporção de acertos na classe positiva.
- **Especificidade**: proporção de acertos na classe negativa.
- **Métrica f1**: média harmônica entre precisão e revocação.

Outra medida usada para avaliação de classificadores em problemas binários é a **análise ROC**, que usa as taxas de verdadeiros positivos e de verdadeiros negativos para exibir graficamente o desempenho de classificadores. A Figura 7.1 indica o espaço ROC e o desempenho de alguns algoritmos. Valores mais próximos do céu ROC são os de melhor desempenho.

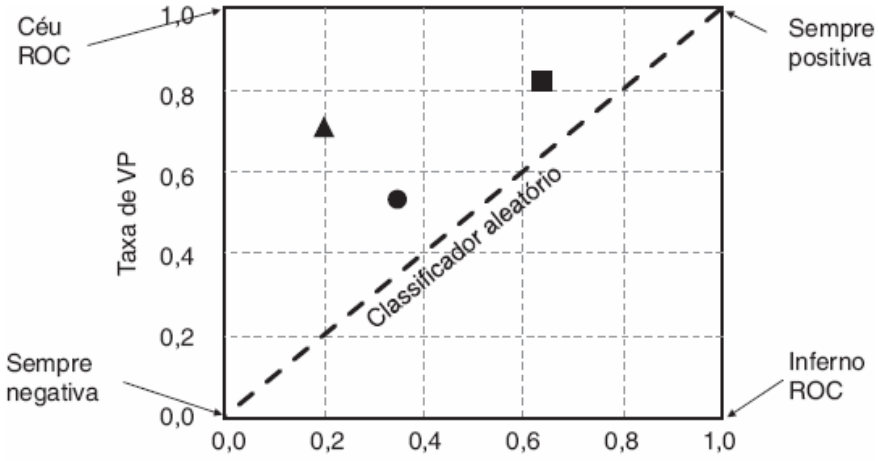


Figura 5.1: espaço ROC – Fonte: Faceli *et al*. (2020)

Outra possibilidade de avaliação e comparação entre algoritmos são os **testes de hipótese**. Nesses testes estatísticos, partimos da hipótese nula de que o desempenho dos algoritmos é similar, dado em geral pela taxa de erro médio, trabalhando com um nível de significância de 5%. Se o teste mostrar que, para tal margem de confiança, há diferenças no desempenho entre os algoritmos, então a hipótese nula é rejeitada e sabemos que um dos algoritmos possui uma taxa de erro menor que a do outro. Trabalhar com os testes de hipótese é uma forma robusta de comparar algoritmos, e um dos testes mais usados é o teste de Wilcoxon signed-rank (ranqueamento de sinais).

Nas **tarefas descritivas**, não temos a possibilidade de contar com rótulos dos objetos para avaliar o desempenho dos algoritmos. Nesses casos, trabalhamos com índices baseados em três tipos de **critérios de avaliação**: relativos, internos e externos.

Os critérios relativos comparam os resultados de agrupamento com alguma medida relacionada, como o número de clusters ou características das relações entre os objetos. Entre esses índices, temos:

- **Variância intracluster**: mede a variação existente entre todos os objetos de um cluster e o seu centroide.
- **Conectividade**: mede o grau com que objetos vizinhos pertencem ao mesmo cluster.
- **Família de índice de Dunn**: mede a razão entre intracluster e intercluster dos objetos.

Os critérios internos dependem de alguma estrutura presente nos dados. A **estatística Gap** é um desses índices, que compara a estrutura obtida pelo agrupamento com a esperança para uma amostra com base em uma distribuição de referência.

Já os critérios externos podem usar várias medidas de similaridade para a avaliação de um agrupamento. No entanto, não é comum que se saiba qual é a estrutura que um agrupamento deveria ter, o que limita o uso desse tipo de critério em muitos casos. Podemos destacar os seguintes índices:

- **Índice Rand**: mede a probabilidade de dois objetos pertencerem ao mesmo cluster ou pertencerem a clusters distintos.
- **Índice Jaccard**: mede a probabilidade de dois objetos que pertencem ao mesmo cluster tanto no modelo gerado quanto no modelo de referência.
- **Índice Fowlkes e Mallows**: indicam a semelhança entre o modelo gerado e o modelo de referência.
- **Índice Variação de Informação**: mede a quantidade de informação perdida ou ganhada entre os dados resultantes do agrupamento e do modelo de referência do agrupamento.

Concluindo

Durante as sete semanas de Aprendizado de Máquinas, estudamos as principais técnicas dos diferentes tipos de aprendizagem existentes. A aplicação dessas técnicas depende das características dos dados e exige que o cientista de dados entenda quais são as técnicas que podem ser mais adequadas. Cada algoritmo de aprendizado de máquina tem seus parâmetros de ajuste, e muitas vezes é necessário avaliar os resultados obtidos pelos algoritmos para um determinado conjunto de dados, de forma a avaliar seu desempenho para classificar ou agrupar tais dados.

Esperamos que os conhecimentos vistos nesta disciplina sejam úteis para o sucesso da sua carreira e, lembre-se, é importante praticar e exercitar as técnicas para se aperfeiçoar cada vez mais.