

Inteligência Artificial - *Machine Learning*

Produção e Fundamentos da Indústria 4.0

Carlos Diego Rodrigues

Federação das Indústrias do Estado do Ceará - FIEC

2 de junho de 2022

Agenda

Introdução

Naive Bayes

KNN

Árvores de decisão

Geração de regras

Métodos lineares

Support Vector Machines

Redes Neurais

Algoritmos de agrupamento

Tabela de entrada

Outlook	Temperature	Humidity	Wind	Play
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

ZeroR

- É o apelido do método sem regras: sem contribuições dos atributos.
- É o mais simples método de classificação.
- Ignora todos os atributos e avalia apenas a classe.
- Consiste em escolher apenas uma resposta para qualquer entrada.
- A partir dos dados de treinamento escolhe a resposta com a maior frequência.


OneR

- É o apelido do método com apenas uma regra.
- Simples, porém preciso.
- Para cada atributo ele vai criar uma regra preditiva para a classe.
- Escolhe o atributo cuja regra tem o menor erro.

Algoritmo OneR

- Para cada atributo
 - Para cada valor possível do atributo, faça uma regra da seguinte forma:
 - Conte quantas vezes cada classe aparece.
 - Escolha a classe mais frequente.
 - Crie uma regra associando o valor deste atributo à classe.
 - Calcule o erro total produzido pelas regras deste atributo
- Escolha o atributo com o menor erro total.

Algoritmo OneR

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3

Agenda

Introdução

Naive Bayes

KNN

Árvores de decisão

Geração de regras

Métodos lineares

Support Vector Machines

Redes Neurais

Algoritmos de agrupamento

Naive Bayes

- É um classificador baseado no Teorema de Bayes com suposição de independência entre os atributos.
- Fácil de construir, sem parametrização e adequado para grandes conjuntos de dados.
- Muito utilizado historicamente.
- Pode ser mais eficiente do que métodos mais sofisticados.

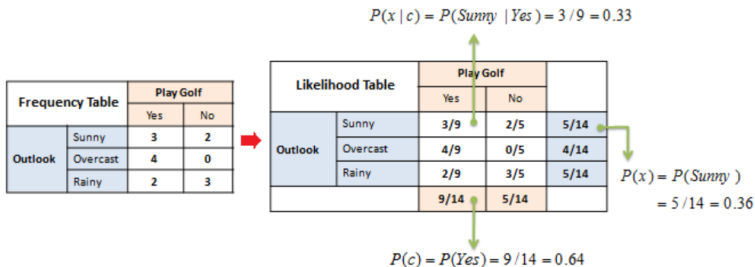
Teorema de Bayes

- Provê uma maneira de calcular probabilidades a posteriori $P(c|x)$ a partir de outras probabilidades $P(c)$, $P(x)$ e $P(x|c)$ que podem ser encontradas na tabela.

$$P(c|x) = \frac{P(x|c) \cdot P(c)}{P(x)}$$

- No algoritmo *OneR* nós escolhemos apenas um atributo. Já no *Naive Bayes* nós escolhemos todos os atributos.

Exemplo com um atributo



$$P(c|x) = P(\text{yes} | \text{Sunny}) = \frac{P(\text{Sunny} | \text{yes}) \cdot P(\text{yes})}{P(\text{Sunny})} = \frac{\frac{3}{9} \cdot \frac{9}{14}}{\frac{5}{14}} = \frac{3}{5} = 0.6$$

Fazendo um exemplo

Digamos que temos vários atributos x_1, x_2, \dots, x_n para prever o valor da classe. Aplicando o Teorema de Bayes, considerando a independência dos atributos temos:

$$P(c|x_1, x_2, \dots, x_n) = \frac{P(x_1|c) \cdot P(x_2|c) \cdots P(x_n|c) \cdot P(c)}{P(x_1) \cdot P(x_2) \cdots P(x_n)}$$

Para efeitos de comparação, não precisamos calcular o divisor, pois ele é o mesmo independente da classe. O valor do numerador é chamado de **verossimilhança** da classe.

Tabelas de frequência

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3



		Play Golf	
		Yes	No
Outlook	Sunny	3/9	2/5
	Overcast	4/9	0/5
	Rainy	2/9	3/5

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1



		Play Golf	
		Yes	No
Humidity	High	3/9	4/5
	Normal	6/9	1/5

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1



		Play Golf	
		Yes	No
Temp.	Hot	2/9	2/5
	Mild	4/9	2/5
	Cool	3/9	1/5

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3



		Play Golf	
		Yes	No
Windy	False	6/9	2/5
	True	3/9	3/5

Fazendo um exemplo

- Prever o exemplo: {Rainy, Cool, High, True}

$$\begin{aligned}P(y|RCHT) &= [P(R|y) \cdot P(C|y) \cdot P(H|y) \cdot P(T|y)] \cdot P(y) \\&= [(2/9) \cdot (3/9) \cdot (3/9) \cdot (3/9)] \cdot (9/14) \\&= 0,00529\end{aligned}$$

$$\begin{aligned}P(n|RCHT) &= [P(R|n) \cdot P(C|n) \cdot P(H|n) \cdot P(T|n)] \cdot P(n) \\&= [(3/5) \cdot (1/5) \cdot (4/5) \cdot (3/5)] \cdot (5/14) \\&= 0,02057\end{aligned}$$

Portanto a resposta do método é **não** para este exemplo.

Dois problemas

- O que fazer com os zeros na tabela?
 - O valor zero é problemático porque ele pode funcionar como um veto na natureza multiplicativa do *Naive Bayes*.
 - Basta adicionar o valor um a todas as frequências da tabela.
- O que fazer quando os dados são numéricos?
 - Dados numéricos não são imediatamente tratados pelo métodos, mas duas saídas são possíveis.
 - Construir uma distribuição de frequências, categorizar.
 - Adotar uma distribuição de probabilidade para os dados numéricos, tipicamente uma distribuição normal, calculando média e desvio padrão e então associando os valores numéricos à probabilidades à partir de uma função conhecida.

Agenda

Introdução

Naive Bayes

KNN

Árvores de decisão

Geração de regras

Métodos lineares

Support Vector Machines

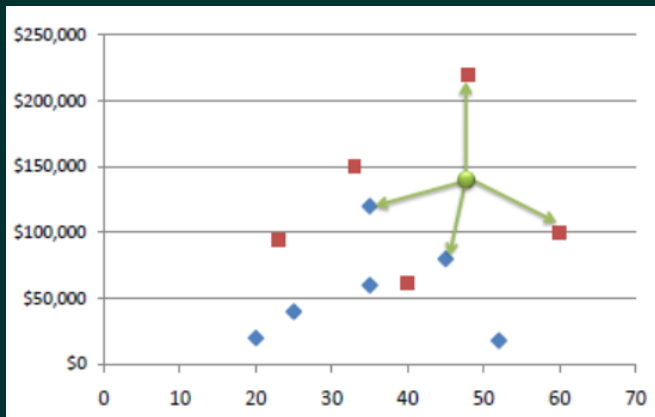
Redes Neurais

Algoritmos de agrupamento

K Vizinhos mais próximos (KNN)

- Método simples que utiliza todos os dados de treino.
- Classificador "lazy".
- Baseado em alguma medida de distância:
 - Linear: $\sum_{i=1}^k |x_i - y_i|$
 - Euclidiana: $\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$
 - Minkowski: $\left(\sum_{i=1}^k (|x_i - y_i|)^q \right)^{(\frac{1}{q})}$
- Em geral utiliza-se atributos padronizados

Exemplo: KNN



Agenda

Introdução

Naive Bayes

KNN

Árvores de decisão

Geração de regras

Métodos lineares

Support Vector Machines

Redes Neurais

Algoritmos de agrupamento

Por que árvores de decisão?

- Apresentação clara do algoritmo de decisão.
- Construção de informação sobre a importância de cada atributo.
- Estrutura altamente eficiente.
- Clareza sobre riscos e recompensas em relação às regras.

Algoritmo ID3

- O algoritmo ID3 constrói uma divisão iterativa dos dados onde em cada nó de decisão ele escolhe um único a ser ramificado.
- São criados ramos para cada valor possível do atributo escolhido.
- O atributo é escolhido a partir dos conceitos de entropia e ganho de informação.
- Os atributos são utilizados conforme possam ser úteis à classificação.

Pseudocódigo

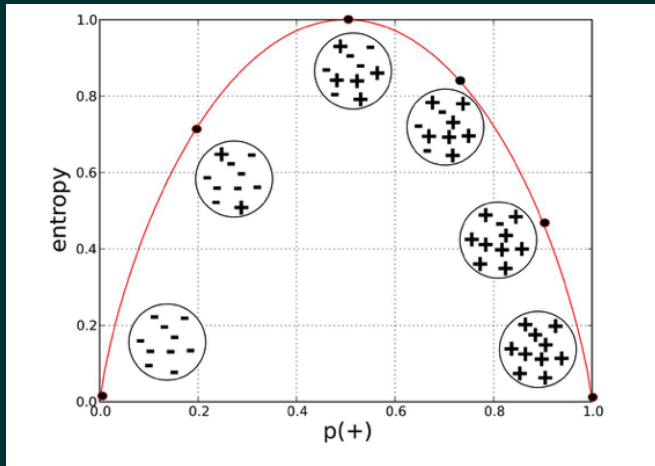
Algorithm 1 ID3

Entrada: A : conjunto de atributos, S : conjunto de instâncias

Saída: R : uma árvore

```
1: Criar um nó raiz  $R$ .
2: if  $\forall x \in S, Classe(x) = 1$  then
3:    $R \leftarrow 1$ 
4: else
5:   if  $\forall x \in S, Classe(x) = 0$  then
6:      $R \leftarrow 0$ 
7:   else
8:     if  $A = \emptyset$  then
9:        $R \leftarrow$  Resultado mais comum em  $S$ .
10:    else
11:      Seja  $D$  o atributo com maior ganho de informação.
12:      for each  $d_i \in D$  do
13:        Criar um novo ramo a partir de  $R$  com  $D = d_i$ .
14:        Adicionar a subárvore  $ID3(A - D, S[D = d_i])$ 
```

Entropia



Entropia

- A entropia em uma tabela de um único atributo é definida como:

$$H(S) = \sum_{v \in V(S)} -p_v \log_2 p_v$$

- Em uma tabela de dois atributos define-se como:

$$H(S, X) = \sum_{v \in V(X)} \frac{|X_v|}{|X|} H(X_v)$$

- Ganho de informação é definido como:

$$G(S, X) = H(S) - H(S, X)$$

Ganhos de informação da tabela original

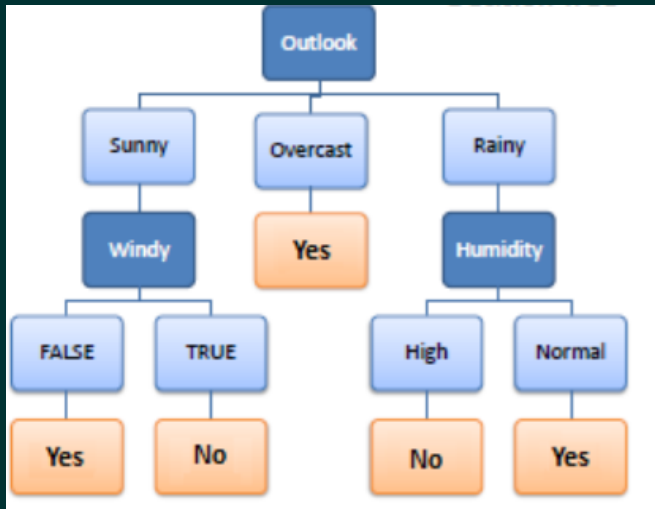
		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

Resultado do Algoritmo ID3



Algoritmo C4.5

- Melhorias sobre o algoritmo ID3:
 - Tratamento de valores numéricos: discretos e contínuos.
 - Tratamento para valores faltantes ou desconhecidos.
- Poda das árvores de decisão.
- Razão de ganho

Tratamento de valores numéricos.

- A ideia é criar faixas de valores nos quais os valores podem ser inseridos.
- Temos duas maneiras distintas: **sem supervisão** ou **com supervisão**.
- Sem supervisão:
 - Faixas de mesmo tamanho.
 - Faixas de mesma frequência.
 - Quantis e outros métodos.
- Com supervisão:
 - Utilizando-se do conceito de entropia e ganho de informação, tentar estipular os pontos de corte da variável que podem melhor discernir sobre a classe a ser prevista.

Tratamento de valores faltantes.

- Valores faltantes são muito comuns em aplicações práticas.
- Políticas para valores faltantes:
 - Ignorar as instâncias com valores faltantes.
 - Dar um novo valor para indicar a falta de dados.
 - Preencher manualmente a partir de conhecimento prévio.
 - Utilizar uma média (quantitativo) ou moda (qualitativo).
 - Utilizar técnicas de ciência de dados para prever o valor faltante.
- Atualização do ganho de informação: ponderar pela proporção de instâncias em que o valor é conhecido.

$$G(S, X) = F \cdot (H(S) - H(S, X))$$

Poda das árvores de decisão.

- Árvores completas podem sofrer com o fenômeno do *overfitting*.
- Pré-poda: na construção da árvore são feitas avaliações se novos nós devem ser adicionados.
- Pós-poda: após a construção da árvore, nós são removidos para termos uma árvore mais eficiente.

Razão de ganho

- O ganho de informação pode esconder uma armadilha: o número de instâncias afetadas não é contabilizado.
- Então divisões em partes muito pequenas podem parecer boas, mas refletem na realidade apenas particularidades que podem não ter conexão com a generalidade.
- Por isso o autor dos métodos ID3 e C4.5 sugere a utilização de uma medida "padronizada", a taxa de ganho.
- Define-se portanto um fator de taxa chamado *split info* (valor intrínseco):

$$I(S, X) = \sum_{v \in V(X)} \frac{|X_v|}{|X|} \log_2 \frac{|X_v|}{|X|}$$

- E a taxa de ganho passa a ser:

$$GR(S, X) = \frac{G(S, X)}{I(S, X)}$$

Agenda

Introdução

Naive Bayes

KNN

Árvores de decisão

Geração de regras

Métodos lineares

Support Vector Machines

Redes Neurais

Algoritmos de agrupamento

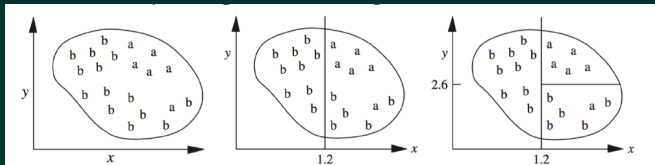
Regras de classificação

- Metodologia similar: dividir para conquistar.
- Algoritmos de cobertura
- Particularidades
 - Método de busca (guloso, busca em largura, etc.)
 - Teste do critério de seleção
 - Método de poda/redução de regras (MDL, validação, etc.)
 - Critério de parada (acurácia mínima, melhoramento mínimo, etc.)
 - Etapa de pós-processamento.
- Lista de decisão vs. Conjunto de regras por classe

Algoritmos de cobertura

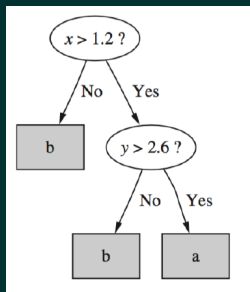
- Poderíamos gerar um conjunto de regras a partir de uma árvore de decisão: cada folha uma regra.
- Podemos gerar um conjunto de regras diretamente.
- Uma abordagem: para cada classe encontrar um conjunto de regras que cobre todas as instâncias nela.
- Abordagem de cobertura: a cada passo do algoritmo, identifica-se uma regra que cobre algumas instâncias.

Exemplo de cobertura



- Primeira regra: se $x \leq 1.2 \rightarrow \text{classe} = b$
- Segunda regra: se $x > 1.2 \ \& \ y > 2.6 \rightarrow \text{classe} = a$

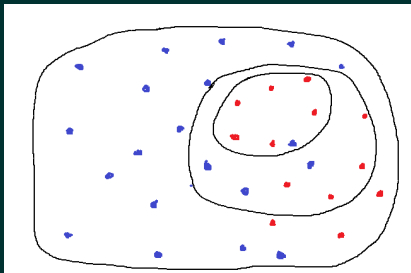
Árvore correspondente



- Regras e árvores podem gerar modelos equivalentes.
- É possível que a árvore seja muito complexa para que seja equivalente a um conjunto de regra simples.
- Por outro lado, as regras em geral tomam apenas uma classe em consideração por vez, enquanto as árvores consideram todas simultaneamente (vantagem especialmente no caso multiclasse).

Algoritmo de cobertura simples

- Ideia básica: gerar uma regra com testes que maximizam a acurácia dela.
- Similar ao problema na árvore de decisão sobre a ramificação.
- Cada novo teste especializa a regra e reduz abrangência da regra, em preferência à sua pureza.



Teste do critério de seleção

- Algoritmo de cobertura básico:
 - Continuar adicionando regras para melhorar a acurácia.
 - Adicionar a condição que aumenta ao máximo a acurácia.
- Medidas de acurácia
 - Considere
 - p : número de instâncias positivas coberto pela regra.
 - t : número total de instâncias coberto pela regra.
 - P : número de instâncias positivas coberto sem a regra.
 - T : número total de instâncias coberto sem a regra.
 - Medida 1: Abrangência de casos positivos: p/t
 - Medida 2: Ganho de informação: $p(\log(p/t)) - \log(P/T)$

Algoritmo PRISM

Algorithm 2 PRISM

Entrada: A : conjunto de atributos, S : conjunto de instâncias

Saída: R : um conjunto de regras

```
1:  $j \leftarrow 1$ 
2: for each classe  $C_i$  do
3:    $E \leftarrow S[Classe = C_i]$ 
4:   while  $E \neq \emptyset$  do
5:     Criar uma regra  $R_j$  que prevê a classe  $C_i$ 
6:     repeat
7:       Seja  $(D, d^*)$  o atributo e valor de máxima abrangência para a regra  $R_j$ 
8:        $R_j \leftarrow R_j \cup \{D = d^*\}$ 
9:     until  $R_j$  seja perfeita
10:    Remova de  $E$  as instâncias cobertas por  $R_j$ .
11:     $R \leftarrow R \cup \{R_j\}$ 
12:     $j \leftarrow j + 1$ 
```

Valores faltantes, atributos numéricos

- Metodologia mais comum: o valor faltante falha em qualquer teste.
- Consequências:
 - Outros testes precisam ser feitos para separar as classes.
 - Podem precisar de casos especiais para que sejam separados posteriormente no processo.
- Em alguns casos é melhor tratar como um valor especial, quando há um significado especial.
- Atributos numéricos são tratados da mesma forma que nas árvores de decisão, com pontos de separação (tipicamente separação binária).
 - Podem ser encontrados a partir de um problema de otimização, assim como nas árvores de decisão.

Regras de poda ou redução de regras

- Duas estratégias principais:
 - Poda incremental.
 - Poda global.
- Critérios de poda.
 - Erro no conjunto de validação.
 - Significância estatística da regra.
 - Princípio MDL.
- Pré-poda e pós-poda.

Conjunto de poda

- Divisão do conjunto de treinamento: conjunto de crescimento e conjunto de poda.
- Estratificação pode ser vantajosa para preservar as proporções das classes quando o conjunto for dividido.
- Poda de erro-reduzido: construir o conjunto de regras a partir do conjunto de crescimento e depois podar (reduzir).
- Poda de erro-reduzido incremental: simplificar as regras assim que ela é criada.

Algoritmo IREP

Algorithm 3 IREP

Entrada: A : conjunto de atributos, S : conjunto de instâncias

Saída: R : um conjunto de regras

```
1:  $E \leftarrow S$ 
2: while  $E \neq \emptyset$  do
3:   Divida  $E$  em um conjunto de crescimento e um de poda na razão 2 : 1.
4:   for each classe  $C_i$  do
5:     Use o método PRISM para gerar uma regra perfeita  $R_i$  para a classe  $C_i$ 
6:     Seja  $R_i^-$  a regra  $R_i$  sem a última cláusula.
7:     while  $\text{valor}(R_i) < \text{valor}(R_i^-)$  do
8:        $R_i \leftarrow R_i^-$ 
9:   Escolha a melhor regra entre todas as classes  $R_*$ 
10:  Remova de  $E$  as instâncias cobertas por  $R_*$ .
11:   $R \leftarrow R \cup \{R_*\}$ 
```

Regras utilizando otimização global

- RIPPER: *Repeated Incremental Pruning to Produce Error Reduction*
- Classes são processadas em ordem crescente de tamanho.
- Conjunto de regras inicial criado usando **IREP**.
- Um critério **MDL** é usado como critério de parada.
- Cada regra é reconsiderada e produzida na tentativa de obter um tamanho descritivo mínimo.
- Uma etapa final de limpeza considera cada regra em relação às demais para minimizar o tamanho descritivo.

PART: Regras baseadas em subárvores parciais

- Método híbrido: utiliza a exploração arborescente para a construção de regras.
- Não constrói toda a árvore, mas utiliza uma pré-poda para aumentar o desempenho.
- Expansão da árvore:
 - Similar à do C4.5
 - Escolhe atributo com maior ganho de informação.
 - Expande os nós e avalia se os filhos possuem menor **erro esperado**.
 - Poda caso o **erro esperado** seja maior que o do pai.
 - Escolhe uma folha com a maior abrangência para o armazenamento da regra.
- Atualiza o conjunto de instância, removendo aquelas cobertas pela regra escolhida.

Comparações com o C4.5 e RIPPER

- Ambos o C4.5 como o RIPPER utilizam um procedimento dito global de otimização.
- Eles produzem uma árvore sobre todas as instâncias ou regras sobre todas as instâncias e só depois fazem um refinamento para excluir nós da árvore ou regras.
- Os autores do PART defendem que o método não utiliza uma otimização global, uma vez que ele gera apenas uma regra por vez a partir de um método arborescente limitado (pré-poda).
- Porém: quanto menos poda o método fizer melhor será seu desempenho prático.
- No pior caso pode gerar uma exploração de mesma complexidade que o C4.5.

Comparações com o C4.5 e RIPPER

- A pré-poda proposta é simples e direta, afeta apenas a regra gerada, enquanto no C4.5 o processo é complexo, envolvendo todas as regras em uma "otimização global" das regras.
- A redução de regras é feita apenas sobre folhas ou nós tornados folhas. Isso reduz efeitos ruins, como a "generalização apressada" do método RIPPER.

Rule	Coverage			
	Training Set		Pruning Set	
	\oplus	\ominus	\oplus	\ominus
1: $a = \text{true} \Rightarrow \oplus$	90	8	30	5
2: $a = \text{false} \wedge b = \text{true} \Rightarrow \oplus$	200	18	66	6
3: $a = \text{false} \wedge b = \text{false} \Rightarrow \ominus$	1	10	0	3

PART

Algorithm 4 PART

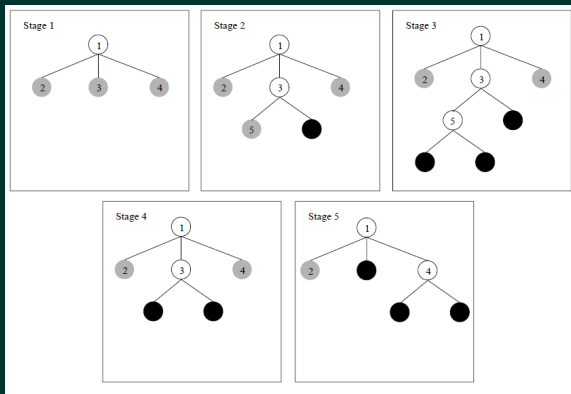
Entrada: A : conjunto de atributos, S : conjunto de instâncias

Saída: R : um conjunto de regras

- 1: Selecione um atributo A_i com maior ganho de informação.
 - 2: Calcular $\text{erro}(S)$
 - 3: Particione o conjunto de instâncias em subconjuntos $E \leftarrow \{S_1, S_2, \dots, S_m\}$.
 - 4: **while** $E \neq \emptyset$ **do**
 - 5: Escolha um S_j em E
 - 6: $R_j \leftarrow \text{PART}(A - \{A_i\}, S_j)$
 - 7: $E \leftarrow E - \{S_j\}$
 - 8: **if** $\forall S_k$ explorado é folha & $\forall S_k \text{ erro}(S) \leq \sum_k \frac{|S_k|}{|S|} \text{erro}(S_k)$ **then**
 - 9: Retraia o nó S em uma folha
 - 10: $R_0 \leftarrow \text{ZeroR}(S)$
 - 11: Retorne a regra R_j de maior abrangência.
-

Ilustração da árvore parcial

- Nos estágios 1, 2 e 3 a exploração arborescente ocorre como de costume. Entre o estágio 3 e 4 temos uma substituição da subárvore 5 por uma folha em vista que ela possui menor erro esperado. O mesmo ocorre para a subárvore 3 entre os estágios 4 e 5. No estágio 5 a subárvore 4 é explorada dando origem a duas folhas.



Regras de associação

- Ao invés de prever exclusivamente as relações dos atributos sobre as classes, podemos estar interessados nas relações dos atributos entre si.
- Problema clássico: associação entre os itens em um carrinho de supermercado.
- Sistemas de recomendação de conteúdo.
- Aplicações: compras online, sistemas de *streaming*, redes sociais.

Conceitos

- Vamos chamar de **item** um par **atributo: valor**.
- Nosso objetivo é encontrar uma regra do tipo $A \implies B$, onde A é um conjunto de itens premissa e B é um conjunto resultado.
- O **suporte** (*support*) de uma regra é a frequência de ocorrência simultânea daqueles itens no conjunto de dados.

$$\text{supp}(A \implies B) = \frac{\text{freq}(A \cap B)}{N}$$

- A **confiança** (*confidence*) de uma regra de associação é a frequência relativa de ocorrência simultânea dos itens em relação à ocorrência das premissas.

$$\text{conf}(A \implies b) = \frac{\text{freq}(A \cap B)}{\text{freq}(A)}$$

Princípio

- A **sustentação** (*lift*) de uma regra é o valor do suporte dividido pelo produto dos suportes das premissas e da conclusão (como se fossem independentes).

$$lift(A \implies B) = \frac{freq(A \cap B)}{freq(A) \cdot freq(B)}$$

O princípio das regras de associação é tentar encontrar um subconjunto de itens com um razoável *suporte* e com alta *confiança*, cuja relevância pode ser apreciada pela sua *sustentação*. Na prática, são estabelecidos valores mínimos de *suporte*, *confiança* e *sustentação* para os quais as regras devem ser mineradas e retidas.

Desafios para minerar as regras de associação.

- Já vimos algoritmos que procuram as relações (regras) entre os atributos e um atributo específico *classe*.
- Em geral eles levavam em consideração critérios de abrangência (que nesse caso se traduzem como *confiança*).
- Aqui temos um desafio de quais itens escolher, dentre todas as combinações de itens, quais deles fazem parte da regra. E depois quais deles formam as premissas e quais deles formam a conclusão.
- Mas todas as combinações importam? Quais delas são interessantes?
- Limtações são necessárias.

Como minerar as regras de associação?

- Primeiramente vamos escolher os itens que fazem parte da regra usando como critério um filtro sobre o suporte das regras.
- Escolhido um conjunto S de itens que fazem parte da regra, vamos escolher agora quais fazem parte das premissas ou da conclusão. Neste caso, vamos avaliar qual é a confiança de cada regra e retemos aquelas com uma confiança mínima.
- Finalmente depois de selecionadas as regras podemos fazer um refinamento por um critério de relevância (sustentação) que pode servir a ordenar e eliminar um número excessivo de regras geradas.

Exemplo

Outlook	Temperature	Humidity	Wind	Play
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes

Escolhendo uma regra

Considerando os itens:

$$\{humidity = normal, windy = false, play = yes\}$$

temos suporte: 4/14.

- $\{humidity = normal \text{ e } windy = false\} \implies play = yes$ 4/4
- $\{humidity = normal \text{ e } play = yes \implies windy = false$ 4/6
- $\{windy = false \text{ e } play = yes \implies humidity = normal$ 4/6
- $\{humidity = normal \implies windy = false \text{ e } play = yes$ 4/7
- $\{windy = false \implies humidity = normal \text{ e } play = yes$ 4/8
- $\{play = yes \implies humidity = normal \text{ e } windy = false$ 4/9
- $\{\} \implies humidity = normal \text{ e } windy = false \text{ e } play = yes$ 4/14

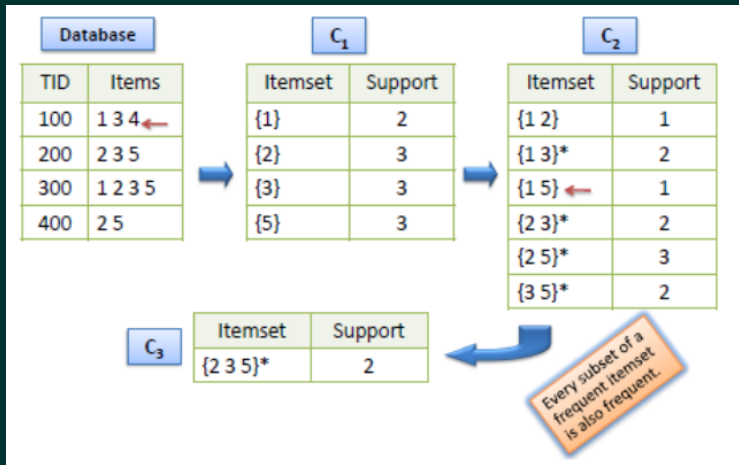
Melhores regras

	Association Rule			Coverage	Accuracy (%)
1	Humidity = normal windy = false	⇒	Play = yes	4	100
2	Temperature = cool	⇒	Humidity = normal	4	100
3	Outlook = overcast	⇒	Play = yes	4	100
4	Temperature = cool play = yes	⇒	Humidity = normal	3	100
5	Outlook = rainy windy = false	⇒	Play = yes	3	100
6	Outlook = rainy play = yes	⇒	Windy = false	3	100
7	Outlook = sunny humidity = high	⇒	Play = no	3	100
8	Outlook = sunny play = no	⇒	Humidity = high	3	100

Algoritmo APRIORI

- O algoritmo *APRIORI* enumera os conjuntos de itens iterativamente na cardinalidade do número de itens desejados, até uma cardinalidade máxima.
- A partir da tabela de dados original, enumera todos os conjuntos contendo 1 item. Todos os elementos com suporte insuficiente são eliminados.
- Então, a partir daí, o algoritmo considera a união de dois conjuntos com abrangências (suporte) suficientes. E com isso calcula a abrangência do conjunto união, fazendo o cruzamento do conjunto com ele mesmo. Novamente, todos os conjuntos com suporte insuficiente são eliminados.
- Utiliza-se do fato que se um conjunto é abrangente, todos os seus subconjuntos também o são.

Ilustração APRIORI



Agenda

Introdução

Naive Bayes

KNN

Árvores de decisão

Geração de regras

Métodos lineares

Support Vector Machines

Redes Neurais

Algoritmos de agrupamento

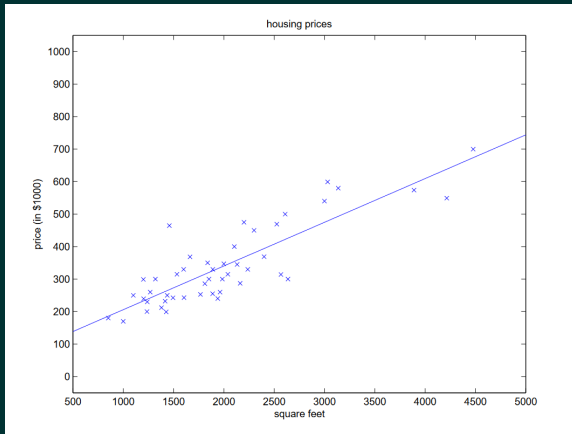
Métodos lineares

- Regressão: consiste em encontrar uma função que traduz as relações entre uma variável dependente (classe) e diversas variáveis independentes (atributos). Essa relação se traduz matematicamente como uma função e o tipo mais comum é a função **linear**.

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_1 x_1 + \cdots + \theta_n x_n = \sum_{j=0}^n \theta_j x_j = \theta^T x$$

- Método canônico na estatística
- Existem diversas formas de encontrar uma regressão linear, mas tipicamente utiliza-se o método de **mínimos quadrados**.

Exemplo regressão



Mínimos quadrados

- Neste método queremos minimizar o erro quadrático do modelo gerado em relação às observações presentes no conjunto de treinamento.

$$\min \sum_{i=1}^m (h_{\theta}(x^i) - y^i)^2 = 2 \min \frac{1}{2} (h_{\theta}(x^i) - y^i)^2 = J(\theta)$$

- Isto pode ser feito através de um processo de otimização iterativo que atualiza um vetor tentativa θ iterativamente até convergir para aquele que corresponde ao mínimo da expressão acima, na direção da derivada da função a ser otimizada.

$$\theta_j \leftarrow \theta_j - \alpha \frac{d}{d\theta_j} J(\theta)$$

- O fator α é chamado taxa de aprendizado do método.

Derivando e concluindo

- Derivando temos:

$$\begin{aligned}\frac{d}{d\theta_j} J(\theta) &= \frac{d}{d\theta_j} (h_\theta(x) - y)^2 \\ &= 2 \cdot \frac{1}{2} (h_\theta(x) - y) \cdot \frac{d}{d\theta_j} (h_\theta(x) - y) \\ &= (h_\theta(x) - y) \cdot \frac{d}{d\theta_j} \left(\sum_{k=1}^n \theta_k x_k - y \right) \\ &= (h_\theta(x) - y) x_j\end{aligned}$$

- Portanto podemos atualizar o valor de θ fazendo

$$\theta_j \leftarrow \theta_j + \alpha (y^i - h_\theta(x^i)) x_j^i$$

- O algoritmo consiste portanto em repetir estas atualizações até que se obtenha a convergência desejada.

Algoritmo Descida do gradiente

Algorithm 5 Descida do gradiente

Entrada: X, Y : conjunto de atributos e classes

Saída: θ : um modelo de regressão

- 1: Encontre um valor inicial para θ
 - 2: **repeat**
 - 3: **for** $i \leftarrow 1 \cdots m$ **do**
 - 4: **for** $j \leftarrow 1 \cdots n$ **do**
 - 5: $\theta_j \leftarrow \theta_j + \alpha(y^i - h_{\theta}(x^i))x_j^i$
 - 6: **until** convergência
 - 7: Retorna θ
-

Explicação probabilística

- Podemos pensar que a resposta do modelo de regressão se relaciona aos dados originais através da expressão:

$$y^i = \theta^T x^i + \epsilon^i$$

onde os valores ϵ^i são independentes e normalmente distribuídos ($\epsilon^i \sim N(0, \sigma)$).

- Dessa forma, vamos considerar a distribuição de cada erro:

$$p(\epsilon^i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\epsilon^i)^2}{2\sigma^2}\right)$$

- Portanto:

$$p(y^i | x^i; \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^i - \theta^T x^i)^2}{2\sigma^2}\right)$$

Minimizando a verossimilhança

- Definindo a verossimilhança:

$$L(\theta) = \prod_{i=1}^m p(y^i | x^i; \theta)$$

- No entanto o produtório é uma função difícil de trabalhar. Podemos tomar, contudo, o logaritmo desta função, cujo máximo é dado pelo mesmo argumento.
- Assim temos:

$$\begin{aligned} l(\theta) &= \log L(\theta) \\ &= m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{i=1}^m (y^i - \theta^T x^i)^2 \end{aligned}$$

- A conclusão é que minimizar o erro sob estas hipóteses é equivalente a encontrar o melhor θ com o método de mínimos quadrados.

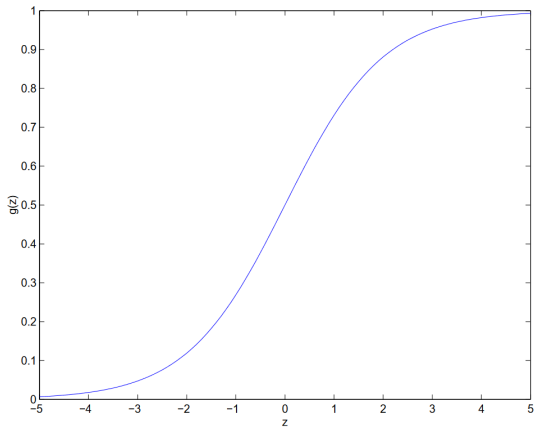
Regressão logística

- Se ignorarmos o fato que y é uma variável binária poderíamos usar a regressão para os problemas de classificação.
- No entanto facilmente encontraríamos exemplos onde esse método funciona muito, pois os valores intermediários são difíceis de distinguir para uma função linear.
- Isto pode ser corrigido se utilizarmos uma função logística.

$$h_{\theta}(x) = g(\theta^T x) = \frac{1}{1 + \exp(-\theta^T x)}$$

onde a função g é chamada função logística ou sigmoid.

Exemplo regressão



Derivando a função logística

- Para aplicarmos uma metodologia similar à anterior seria interessante conhecer a derivada da função g :

$$\begin{aligned}g'(z) &= \frac{d}{dz} \frac{1}{1 + e^{-z}} \\&= \frac{1}{(1 + e^{-z})^2} e^{-z} \\&= \frac{1}{1 + e^{-z}} \cdot \left(1 - \frac{1}{1 + e^{-z}}\right) \\&= g(z)(1 - g(z))\end{aligned}$$

- Vamos utilizar a mesma ideia da regressão linear para derivar um estimador interessante θ para a verossimilhança do modelo logístico.

Verossimilhança do modelo logístico

- Vamos supor que:

$$P(y = 1|x; \theta) = h_{\theta}(x)$$

$$P(y = 0|x; \theta) = 1 - h_{\theta}(x)$$

- Isto pode ser transformado em uma função de distribuição:

$$p(y|x; \theta) = (h_{\theta}(x))^y (1 - h_{\theta}(x))^{1-y}$$

- Definindo a função de verossimilhança:

$$\begin{aligned} L(\theta) &= \prod_{i=1}^m p(y^i|x^i; \theta) \\ &= \prod_{i=1}^m (h_{\theta}(x^i))^{y^i} (1 - h_{\theta}(x^i))^{1-y^i} \end{aligned}$$

Otimizando o logaritmo

- Definindo o logaritmo da verossimilhança

$$\begin{aligned}l(\theta) &= \log L(\theta) \\&= \sum_{i=1}^m y^i \log h_{\theta}(x^i) + (1 - y^i) \log(1 - h_{\theta}(x^i))\end{aligned}$$

- Derivando a função $l(\theta)$:

$$\begin{aligned}\frac{d}{d\theta_j} l(\theta) &= \left(y \frac{1}{g(\theta^T x)} - (1 - y) \frac{1}{1 - g(\theta^T x)} \right) \frac{d}{d\theta_j} g(\theta^T x) \\&= \left(y \frac{1}{g(\theta^T x)} - (1 - y) \frac{1}{1 - g(\theta^T x)} \right) g(\theta^T x)(1 - g(\theta^T x)) \frac{d}{d\theta_j} x \\&= (y(1 - g(\theta^T x)) - (1 - y)g(\theta^T x))x_j \\&= (y - h_{\theta}(x))x_j\end{aligned}$$

Descida do gradiente para Regressão logística

- Podemos otimizar o valor de $l(\theta)$ a partir do valor de sua derivada, fazendo

$$\theta \leftarrow \theta + \alpha \nabla_{\theta} l(\theta)$$

- Encontramos portanto que a minimização da verossimilhança (e portanto a definição de um modelo) para a regressão logística pode ser obtida de forma semelhante àquela obtida na regressão linear, através de sucessivas atualizações do modelo θ :

$$\theta_j \leftarrow \theta_j + \alpha(y^i - h_{\theta}(x^i))x_j$$

- Esta função é muito semelhante, porém não idêntica àquela encontrada para a regressão pois as funções $h_{\theta}(x)$ são distintas.

Agenda

Introdução

Naive Bayes

KNN

Árvores de decisão

Geração de regras

Métodos lineares

Support Vector Machines

Redes Neurais

Algoritmos de agrupamento

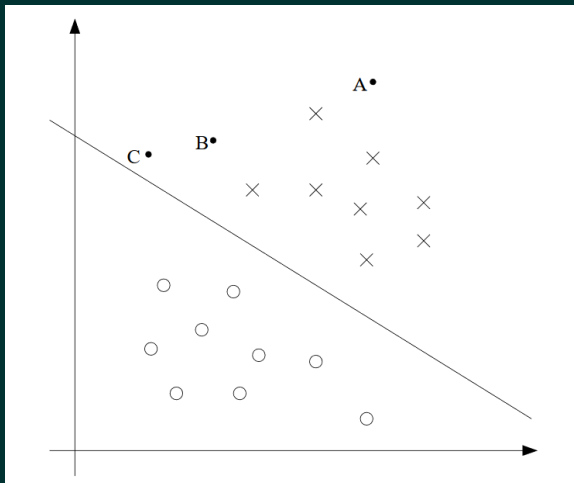
Máquinas de vetor de suporte - SVM

- Máquinas de vetor de suporte estão entre os métodos mais populares e mais eficientes para problemas de classificação.
- Consiste em encontrar uma borda (ou margem) de classificação que separa exemplos de classes distintas.
- Qual a melhor margem separadora? Vetores de suporte.
- A partir de uma borda linear é possível encontrar outras bordas de maior complexidade que satisfaçam algumas condições na definição da função de borda.
- *Kernell tricks*

Margens

- Quando fazemos uma regressão logística, temos como resultado um modelo θ de tal forma que ao aplicarmos a função sobre um exemplo x^i , o resultado $h_{\theta}(x^i) \geq 0,5 \leftrightarrow \theta^T x^i \geq 0$.
- Se $\theta^T x^i \gg 0$ então temos alta confiança que $y^i = 1$ enquanto se $\theta^T x^i \ll 0$ temos alta confiança que $y^i = 0$.
- Podemos dizer assim que o hiperplano $\theta^T x = 0$ é um hiperplano separador para as duas classes.

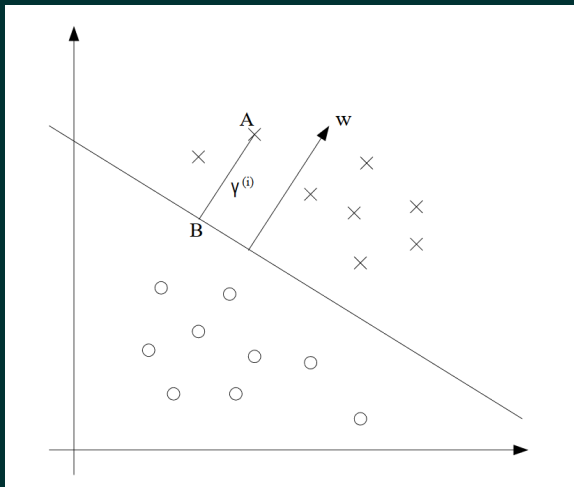
Margens



Margens geométricas

- Primeiro vamos considerar que $y^i \in \{-1, +1\}$.
- Dado um exemplo (x^i, y^i) , vamos definir uma margem funcional de (w, b) com relação ao exemplo o valor $\gamma^i = y^i(w^T x^i + b)$
- Se $y^i = 1$ então queremos que o valor da margem seja positivo, enquanto se $y^i = -1$ queremos o valor da margem negativo.
- Restrição: $y^i(w^T x^i + b) \geq 0$

Margens geométricas



Normalização da margem

- Normalização da margem (w, b):

$$w^T \left(x^i - \gamma^i \frac{w}{\|w\|} \right) + b = 0$$

- Resolvendo para γ^i temos:

$$\gamma^i = y^i \left(\left(\frac{w}{\|w\|} \right)^T x^i \right) + \frac{b}{\|w\|}$$

- Menor margem é a medida de qualidade do modelo:

$$\gamma = \min_{i=1, \dots, m} \gamma^i$$

Problema de otimização

- Queremos maximizar a margem para aumentar a confiança do modelo:

$$\begin{aligned} \max_{\gamma, w, b} \quad & \gamma \\ \text{s.a : } & y^i(w^T x^i + b) \geq \gamma \quad i = 1, \dots, m \\ & \|w\| = 1 \end{aligned}$$

- Podemos utilizar a relação entre γ e $\|w\|$ para reescrever o problema de uma outra forma:

$$\begin{aligned} \max_{\gamma, w, b} \quad & \frac{\hat{\gamma}}{\|w\|} \\ \text{s.a : } & y^i(w^T x^i + b) \geq \hat{\gamma} \quad i = 1, \dots, m \end{aligned}$$

Modelo final

- Fazendo $\hat{\gamma} = 1$ chegamos na minimização de $\|w\|$:

$$\min_{w,b} \frac{1}{2} \|w\|^2$$
$$y^i (w^T x^i + b) \geq 1 \quad i = 1, \dots, m$$

- Problema de otimização não linear (quadrático)
- Técnicas consolidadas para resolvê-lo, incluindo a decomposição lagrangeana.

Decomposição Lagrangeana

- Dado um problema de otimização qualquer:

$$\begin{aligned} \min_w \quad & f(w) \\ \text{s.a : } \quad & g_i(w) \leq 0 & i = 1, \dots, k \\ & h_i(w) = 0 & i = 1, \dots, l \end{aligned}$$

- Definimos o dual Lagrangeano como a função:

$$\mathcal{L}(w, \alpha, \beta) = f(w) + \sum_{i=1}^k \alpha_i g_i(w) + \sum_{i=1}^l \beta_i h_i(w)$$

onde α e β são chamados multiplicadores de Lagrange.

Decomposição Lagrangeana

- Considere, para um w fixo o valor:

$$\theta(w) = \max_{\alpha \geq 0, \beta} f(w) + \sum_{i=1}^k \alpha_i g_i(w) + \sum_{i=1}^l \beta_i h_i(w)$$

- Otimizando em w temos:

$$\min_w \theta(w) = \min_w \max_{\alpha \geq 0, \beta} \mathcal{L}(w, \alpha, \beta)$$

- Sob certas condições (funções g são convexas e funções h são afins), o problema acima é equivalente a:

$$\max_{\alpha \geq 0, \beta} \min_w \mathcal{L}(w, \alpha, \beta)$$

Condições de Karush-Khan-Tucker (KKT)

- Para encontrar um trio (w^*, α^*, β^*) ótimo é necessário e suficiente que satisfaça:

$$\frac{d}{dw_i} \mathcal{L}(w^*, \alpha^*, \beta^*) = 0 \quad i = 1, \dots, n \quad (1)$$

$$\frac{d}{d\beta_i} \mathcal{L}(w^*, \alpha^*, \beta^*) = 0, \quad i = 1, \dots, l \quad (2)$$

$$\alpha_i^* (g_i(w^*)) = 0, \quad i = 1, \dots, k \quad (3)$$

$$g_i(w^*) \leq 0, \quad i = 1, \dots, k \quad (4)$$

$$\alpha_i^* \geq 0, \quad i = 1, \dots, k \quad (5)$$

Aplicando no modelo SVM

- Relembrando o modelo:

$$\min_{w,b} \frac{1}{2} \|w\|^2$$
$$y^i(w^T x^i + b) \geq 1 \quad i = 1, \dots, m$$

- Portanto:

$$f(w, b) = \frac{1}{2} \|w\|^2$$
$$g_i(w, b) = 1 - y^i(w^T x^i + b)$$

- A função Lagrangeana é então:

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^m \alpha_i \cdot (y^i(w^T x^i + b) - 1)$$

Observando as condições de KKT

- Primeiramente derivando em relação a w :

$$\nabla_w \mathcal{L}(w, b, \alpha) = w - \sum_{i=1}^m \alpha_i y^i x^i = 0$$

- Portanto:

$$w = \sum_{i=1}^m \alpha_i y^i x^i$$

- Agora derivando em relação a b :

$$\frac{d}{db} \mathcal{L}(w, b, \alpha) = \sum_{i=1}^m \alpha_i y^i = 0$$

Utilizando as condições de KKT para voltar ao modelo

- Vamos inserir os resultados anteriores para eliminar os fatos w e b :

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^m \alpha_i \cdot \left(y^i (w^T x^i + b) - 1 \right)$$

- Usando o fato $w = \sum_{i=1}^m \alpha_i y^i x^i$:

$$\mathcal{L}(w, b, \alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m y^i y^j \alpha_i \alpha_j (x^i)^T x^j - b \sum_{i=1}^m \alpha_i y^i$$

- Agora sabendo que $\sum_{i=1}^m \alpha_i y^i = 0$:

$$\mathcal{L}(w, b, \alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m y^i y^j \alpha_i \alpha_j \langle x^i, x^j \rangle$$

Modelo dual

- Chegamos a um modelo que depende apenas da variável lagrangeana α :

$$\begin{aligned} \max_{\alpha} W(\alpha) &= \sum_{i=1}^m \sum_{j=1}^m y^i y^j \alpha_i \alpha_j \langle x^i, x^j \rangle \\ \text{s.a : } \alpha_i &\geq 0, \quad i = 1, \dots, m \end{aligned} \quad \sum \alpha_i y^i = 0$$

- A partir do valor ótimo de α para este modelo podemos reestabelecer os valores ótimos para w e b .

Algoritmo SMO e *Kernels*

- Chegamos a um modelo que depende apenas da variável lagrangeana α :

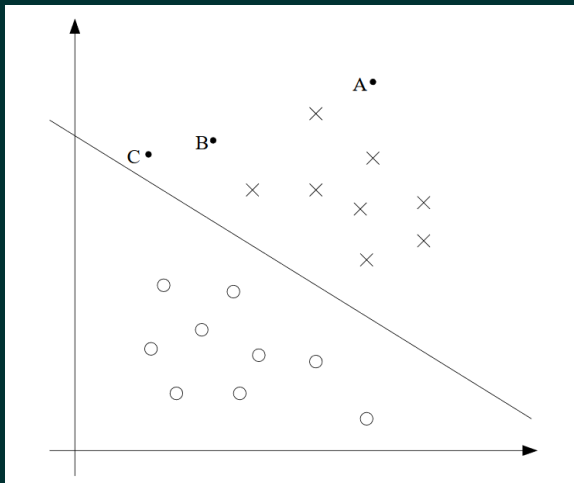
$$\max_{\alpha} W(\alpha) = \sum_{i=1}^m \sum_{j=1}^m y^i y^j \alpha_i \alpha_j \langle x^i, x^j \rangle$$

$$s.a : \sum \alpha_i y^i = 0$$

$$\alpha_i \geq 0, \quad i = 1, \dots, m$$

- Vamos elaborar uma estratégia de otimização iterativa observando propriedades particulares do modelo e suas condições de otimalidade.
- O que ocorre com os casos que não são linearmente separáveis?
- *Kernels*

Relembrando o que queremos fazer



Voltando ao modelo

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^m \sum_{j=1}^m y^i y^j \alpha_i \alpha_j \langle x^i, x^j \rangle$$

$$s.a : \sum \alpha_i y^i = 0$$

$$\alpha_i \geq 0, \quad i = 1, \dots, m$$

- Partindo de uma solução inicial (por exemplo $\alpha_i = 0$, $i = 1, \dots, m$), observe que não é possível modificar apenas um único coeficiente sem violar a restrição

$$\sum \alpha_i y^i = 0$$

- Portanto, se quisermos mudar iterativamente de solução devemos considerar pelo menos dois exemplos (x^a, y^a) e (x^b, y^b) de tal forma que $y^a + y^b = 0$.

Desenvolvendo em função do par a, b

- Considerando exatamente dois exemplos podemos manipular a restrição, isolando os termos de α_a e α_b :

$$\alpha_a y^a + \alpha_b y^b = - \sum_{i \notin \{a, b\}} \alpha_i y^i$$

- Fixando todos os valores que de α que não sejam aqueles de a e b temos:

$$\alpha_a y^a + \alpha_b y^b = \beta$$

- Por esta igualdade, podemos ainda escrever tudo em função de α_b :

$$\alpha_b = y^b (\beta - \alpha_a y^a)$$

Modelo simplificado em uma única variável

- Assim o valor de

$$W(\alpha) = W(\alpha_1, \dots, \alpha_a, \dots, \alpha_b, \dots, \alpha_m)$$

passa a

$$W(\alpha) = W(\alpha_1, \dots, \alpha_a, \dots, y^b(\beta - \alpha_a y^a), \dots, \alpha_m)$$

- Considerando fixos os demais α_i para $i \notin \{a, b\}$ temos um modelo que depende de uma única variável $\alpha_a \geq 0$ com uma simples função quadrática $A\alpha_a^2 + B\alpha_a + C$ a ser otimizada.
- Naturalmente o valor ótimo desta função ocorre onde

$$\alpha_a = \frac{-B}{2A}$$

quando $\frac{-B}{2A} \geq 0$.

Algoritmo SMO

Algorithm 6 SMO

Entrada: X, Y : conjunto de atributos e classes

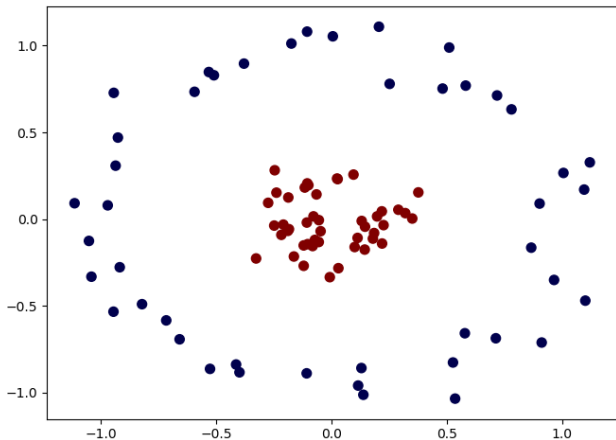
Saída: α : coeficientes lagrangeanos

- 1: Encontre um valor inicial para α
 - 2: **repeat**
 - 3: Escolha dos exemplos (x^a, y^a) e (x^b, y^b) .
 - 4: Faça $\alpha_b = y^b (\beta - \alpha_a y^a)$.
 - 5: Encontre e otimize a equação $A\alpha_a^2 + B\alpha_a + C$.
 - 6: Atualize os valores de α_a e α_b .
 - 7: **until** convergência
 - 8: Retorna α
-

Casos não linearmente separáveis

- Podemos ter casos onde uma separação linear seria inviável para a classificação.
- No caso do SVM isto indicaria um modelo inviável que não resultaria em qualquer solução.
- Classicamente existem duas propostas para lidar com estes casos:
 1. uma relaxação do modelo original, aplicando penalidades aos casos erroneamente classificados;
 2. uma transformação dos pontos originais para um espaço de dimensão ampliada, onde neste espaço eles sejam linearmente separáveis.
- É possível combinar ambos os métodos!

Exemplo não-separável



Penalidades

- Retomando o modelo original:

$$\min_{w,b} \frac{1}{2} ||w||^2$$
$$y^i(w^T x^i + b) \geq 1 \quad i = 1, \dots, m$$

- Vamos adicionar fatores que permitam, para cada exemplo, que a sua restrição seja violada, mas estes fatores vão ter um custo na função objetivo.

$$\min_{w,b} \frac{1}{2} ||w||^2 + C \cdot \sum_{i=1}^m \epsilon_i$$
$$y^i(w^T x^i + b) \geq 1 - \epsilon_i \quad i = 1, \dots, m$$
$$\epsilon_i \geq 0 \quad i = 1, \dots, m$$

Aplicando a decomposição Lagrangeana

- O modelo dualizado é:

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^m \sum_{j=1}^m y^i y^j \alpha_i \alpha_j \langle x^i, x^j \rangle$$

$$s.a : \sum \alpha_i y^i = 0$$

$$0 \leq \alpha_i \leq C, \quad i = 1, \dots, m$$

- Toda a ideia do algoritmo SMO permanece.
- Temos que averiguar a restrição $\alpha_i \leq C$ quando vamos resolver para uma única variável.

Kernels

- O espaço vetorial dos atributos originais das instâncias pode não oferecer a possibilidade de uma separação linear.
- Podemos construir um mapeamento do espaço dos atributos para um espaço expandido, chamado espaço de características. Isto é feito aplicando-se uma função ϕ a cada instância.
- Como vimos anteriormente, nossos modelos e algoritmos podem ser escritos dependendo quase que exclusivamente do produto interno dos atributos $\langle x^i, x^j \rangle$.
- Definimos a função **Kernel** como o produto interno das características:

$$K(x^i, x^j) = \langle \phi(x^i), \phi(x^j) \rangle$$

Efeitos do *Kernel*

- Portanto a qualquer momento em nosso desenvolvimento se trocarmos $\langle x^i, x^j \rangle$ por $K(x^i, x^j)$ estaremos aplicando o SVM sobre um outro espaço, mais adequado à separação linear.
- Isto também é compatível com a ideia das penalidades, agora no espaço de características.
- O chamado truque dos *Kernels* acontece quando a função $K(x^i, x^j)$ é fácil de calcular, porém as transformações $\phi(x^i)$ e $\phi(x^j)$ são complexas (e desnecessárias!)
- Podemos portanto trabalhar apenas com o produto vetorial $K(x^i, x^j)$ como uma medida de similaridade entre os objetos x^i e x^j sem necessariamente expandi-los.

Condições para uma função *Kernel*

- Como saber se existe um mapeamento de características ϕ para uma determinada função candidata $K(x^i, x^j)$ a um Kernel?
- **Teorema de Mercer:** Dada uma função $K : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$, dizemos que K é um *Kernel* se, e somente se, para quaisquer pontos x^1, \dots, x^m , com $m < \infty$ a matriz de kernel

$$\begin{bmatrix} K(x^1, x^1) & K(x^1, x^2) & \cdots & K(x^1, x^m) \\ K(x^2, x^1) & K(x^2, x^2) & \cdots & K(x^2, x^m) \\ \vdots & \vdots & \ddots & \vdots \\ K(x^m, x^1) & K(x^m, x^2) & \cdots & K(x^m, x^m) \end{bmatrix}$$

é semidefinida positiva ($x^T K x \geq 0, \forall x \in \mathbb{R}^m$).

Exemplos de *Kernels*

- *Kernel* polinomial:

$$K(x^i, x^j) = (\langle x^i, x^j \rangle + c)^d$$

(espaço de características com $\binom{n+d}{d}$ dimensões)

- *Kernel* gaussiano:

$$K(x^i, x^j) = \exp \left(-\frac{\langle x^i - x^j, x^i - x^j \rangle}{2\sigma^2} \right)$$

(espaço de características infinito!)

Agenda

Introdução

Naive Bayes

KNN

Árvores de decisão

Geração de regras

Métodos lineares

Support Vector Machines

Redes Neurais

Algoritmos de agrupamento

Redes Neurais

- Motivação: *Deep Learning*.
- Por que usar mais de uma camada?
- Descida de gradiente estocástica.
- Retropropagação do erro.
 - Rede de duas camadas
 - Generalização matricial

Deep Learning

- Redes neurais causaram grande impacto em anos recentes em aplicações como visão computacional e reconhecimento de linguagem natural.
- Um dos motivos para isto é a capacidade dos dispositivos lidarem com grande quantidade de dados.
- Estudos mostram que uma estrutura expandida, com diversas camadas de computação, também traz benefícios significativos na qualidade do algoritmo (em comparação com a aplicação de uma função diretamente: regressão).
- Os modelos *deep learning* são flexíveis e conseguem explorar mais combinações de informação enterradas nos conjuntos de instâncias com maior eficiência.
- Surgimento de software e hardware próprios para redes neurais: *tensors* e *GPUs*.

Deep Learning

1. Aprendizado de máquina clássico: fazer previsões diretamente de um conjunto de atributos pré-especificados pelo usuário.
2. Aprendizado com técnicas de representação: transformar os atributos em uma representação antes de fazer uma previsão.
3. Aprendizado profundo (*deep learning*): uma técnica de aprendizado com representação que pode utilizar múltiplos passos (camadas) de representação para criar características complexas.

MNIST - Reconhecimento de números escritos

Classifier	Test Error Rate (%)	References
Linear classifier (1-layer neural net)	12.0	LeCun et al. (1998)
K-nearest-neighbors, Euclidean (L2)	5.0	LeCun et al. (1998)
2-Layer neural net, 300 hidden units, mean square error	4.7	LeCun et al. (1998)
Support vector machine, Gaussian kernel	1.4	MNIST Website
Convolutional net, LeNet-5 (no distortions)	0.95	LeCun et al. (1998)
Methods using distortions		
Virtual support vector machine, deg-9 polynomial, (2-pixel jittered and deskewing)	0.56	DeCoste and Scholkopf (2002)
Convolutional neural net (elastic distortions)	0.4	Simard, Steinkraus, and Platt (2003)
6-Layer feedforward neural net (on GPU) (elastic distortions)	0.35	Ciresan, Meier, Gambardella, and Schmidhuber (2010)
Large/deep convolutional neural net (elastic distortions)	0.35	Ciresan, Meier, Masci, Maria Gambardella, and Schmidhuber (2011)
Committee of 35 convolutional networks (elastic distortions)	0.23	Ciresan, Meier, and Schmidhuber (2012)

Uma primeira rede neural: regressão

- Uma função de regressão (ou regressão logística) pode ser vista como uma primeira rede neural simples.
- Nesta rede temos um conjunto de entradas igual ao número de atributos da instância original e uma única saída que é calculada diretamente a partir das entradas:

$$h_w(x, b) = \sigma(w^T x + b)$$

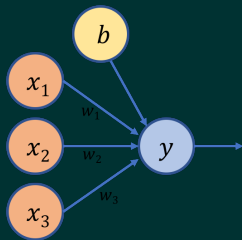
- Este modelo pode ser avaliado para cada exemplo, pelo erro quadrático proporcionado:

$$J^i(w) = \frac{1}{2}(h_w(x^i, b^i) - y^i)^2 \forall i = 1, \dots, m$$

- Podemos adotar como critério para um modelo ideal a minimização do erro médio sobre todas as instâncias:

$$w^* = \arg \min_w \left\{ \frac{1}{m} \sum_{i=1}^m J^i(w) \right\}$$

Rede neural de regressão

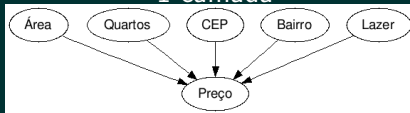


Mais camadas

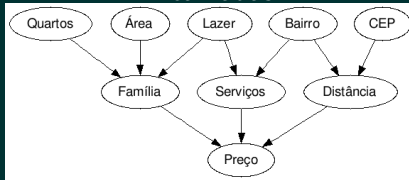
- Digamos que temos um problema de prever o preço de um imóvel.
- Alguns atributos interessantes podem ser:
 - Área do imóvel
 - Número de quartos
 - CEP
 - Bairro
 - Área de lazer
- Estas características estão relacionadas às características que fazem de fato com que as pessoas paguem mais por um imóvel:
 - Tamanho da família
 - Acesso a bons serviços
 - Proximidade com o centro da cidade
- Poderíamos então associar os atributos a essa camada escondida.

Rede neural com duas camadas

1 camada



2 camadas



Relações entre camadas

- Podemos observar que a saída agora não depende mais diretamente da entrada através da relação:
$$h_w(x, b) = \sigma(w^T x + b).$$
- De maneira geral poderíamos dizer que cada nó além da camada de entrada pode ser escrito através de alguma relação dessa forma.
- O mais comum é considerarmos que todos os nós de uma camada estão conectados a todos os nós da camada seguinte e deixar que os coeficientes emergjam no processo de resolução.
- Portanto em cada camada k podemos considerar uma matriz $W^{[k]}$, onde cada coluna $W_j^{[k]}$ corresponde ao j -ésimo nó daquela camada e o valor $W_{ij}^{[k]}$ é o peso da sua ligação com o i -ésimo elemento da camada seguinte.

Relações entre camadas

- Dessa forma, para o caso de duas camadas, é possível obter um vetor intermediário $a = \sigma(W^{[1]}x + b^{[1]})$ e então calcular $h_W(x, b) = \sigma(W^{[2]}a + b^{[2]})$.
- Generalizando para r camadas temos:

$$a^{[1]} = \sigma^{[1]}(W^{[1]}x + b^{[1]})$$

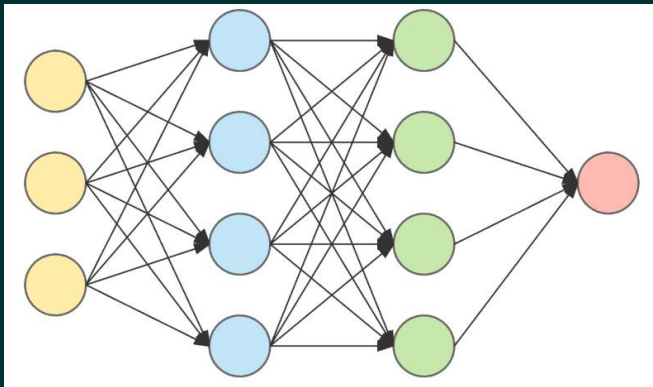
$$a^{[2]} = \sigma^{[2]}(W^{[2]}a^{[1]} + b^{[2]})$$

...

$$a^{[r-1]} = \sigma^{[r-1]}(W^{[r-1]}a^{[r-2]} + b^{[r-1]})$$

$$h_W(x) = \sigma^{[r]}(W^{[r]}a^{[r-1]} + b^{[r]})$$

Esquema de uma rede neural com 2 camadas internas



Funções de ativação

- A função de ativação pode mudar, a depender do interesse do desenvolvedor da rede.
- Observe que se a função de ativação for uma identidade, é possível reduzir aquela camada. Suponha que temos duas camadas, onde $a^{[1]} = W^{[1]}x + b^{[1]}$.

$$h_W(x) = \sigma(W^{[2]}a^{[1]} + b^{[2]})$$

$$h_W(x) = \sigma(W^{[2]}(W^{[1]}x + b^{[1]}) + b^{[2]})$$

$$h_W(x) = \sigma(W^{[2]}W^{[1]}x + (W^{[2]}b^{[1]} + b^{[2]}))$$

$$h_W(x) = \sigma(\tilde{W}x + \tilde{b})$$

- Funções de ativação definem portanto o objetivo e necessidade da camada.

Ativações típicas

- Função logística: $f(x) = \frac{1}{1+e^{-x}}$
- Tangente hiperbólica: $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- *Softmax*: $f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$
- *Softsign*: $f(x) = \frac{x}{|x|+1}$
- *Rectified Linear Unit*: $f(x) = \max\{0, x\} = [\max\{0, x_i\}]$
- *Softplus*: $f(x) = \log(1 + e^x)$
- Outras ...

Como resolver com mais camadas

- O processo de regressão nos elucida um caminho de resolução para uma rede neural onde entradas e saídas estão diretamente conectadas.
- A descida de gradiente consiste na atualização iterativa dos valores (w, b) de forma que:

$$w_j \leftarrow w_j + \alpha(y^i - h_w(x^i))x_j$$

- De forma mais geral poderíamos pensar no seguinte esquema:

$$w_j \leftarrow w_j + \alpha \frac{d}{dw_j} h_w(x)$$

- Dois desafios surgem no caso das redes neurais:
 1. Uma grande quantidade de entradas, neurons, exemplos ...
 2. Como atualizar os valores para mais de uma camada.

Descida de gradiente estocástica

- Consiste em aplicar exatamente a mesma metodologia do método de descida de gradiente.
- Contudo nem todos os exemplos são considerados para fazer uma atualização dos valores.
- Divide-se o conjunto de instâncias em amostras chamadas "*mini-batches*".
- Esta amostra aleatória não conduz ao valor ideal de descida de gradiente, mas permite uma atualização no sentido da otimização. Após uma nova avaliação, um outro "*mini-batch*" é usado e o método segue.

Algoritmo Descida de gradiente estocástica

Algorithm 7 Descida do gradiente estocástica

Entrada: X, Y : conjunto de atributos e classes

Saída: (w, b) : coeficientes aproximados

- 1: Encontre um valor inicial para (w, b)
 - 2: **for** $k \leftarrow 1 \cdots N_{iter}$ **do**
 - 3: Escolha uma amostra (X', Y') de (X, Y)
 - 4: **for** $(x', y') \in (X', Y')$ **do**
 - 5: **for** $j \leftarrow 1 \cdots n$ **do**
 - 6: $w_j \leftarrow w_j + \alpha \frac{d}{dw_j} h_w(x)$
 - 7: $b \leftarrow b + \alpha \frac{d}{db} h_w(x)$
 - 8: Retorna (w, b)
-

Retropropagação do erro

- A técnica para lidar com as várias camadas da rede neural é chamada de retropropagação do erro.
- Similarmente à teoria vista para regressão e regressão logística ela consiste em utilizar a margem de erro obtida por determinados candidatos (W, B) da rede para melhorá-los iterativamente com o auxílio da descida de gradiente (estocástica).
- **Teorema :** *Suponha que uma rede de tamanho N calcula uma função real $f : \mathbb{R}^I \rightarrow \mathbb{R}$. Então o gradiente ∇f pode ser calculado por um circuito de tamanho $O(N)$ em tempo $O(N)$.*
- Auto-diferenciação.

Retropropagação em uma rede de duas camadas

- Vamos voltar ao caso da rede de duas camadas explicitando cada passo do cálculo:

$$z = W^{[1]}x + b^{[1]}$$

$$a = \sigma(z)$$

$$o = W^{[2]}a + b^{[2]}$$

$$J = \frac{1}{2}(y - o)^2$$

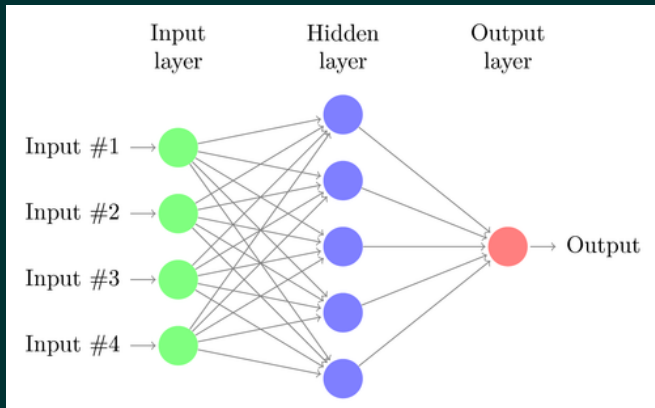
- Vamos utilizar ao longo desta seção a regra da cadeia:

$$g_j = g_j(x_1, x_2, \dots, x_n)$$

$$J = J(g_1, g_2, \dots, g_k)$$

$$\frac{dJ}{dx_i} = \sum_{j=1}^k \frac{dJ}{dg_j} \frac{dg_j}{dx_i}$$

MNIST - Reconhecimento de números escritos



Retropropagação em uma rede de duas camadas

- Nosso objetivo é calcular as derivadas de cada camada para propor atualizações para $W^{[1]}$, $W^{[2]}$, $b^{[1]}$ e $b^{[2]}$.
- Calculando $\frac{dJ}{dW^{[2]}}$.

$$\begin{aligned}\frac{dJ}{dW_i^{[2]}} &= \frac{dJ}{do} \cdot \frac{do}{dW_i^{[2]}} \\ &= (o - y) \cdot \frac{do}{dW_i^{[2]}} \\ &= (o - y) \cdot a_i\end{aligned}$$

- Calculando $\frac{dJ}{db^{[2]}}$

$$\begin{aligned}\frac{dJ}{db^{[2]}} &= \frac{dJ}{do} \cdot \frac{do}{db^{[2]}} \\ &= (o - y) \cdot \frac{do}{db} \\ &= (o - y)\end{aligned}$$

Retropropagação em uma rede de duas camadas

- Calculando $\frac{dJ}{dW_{ij}^{[1]}}$.

$$\begin{aligned}\frac{dJ}{dW_{ij}^{[1]}} &= \frac{dJ}{dz_i} \cdot \frac{dz_i}{dW_{ij}^{[1]}} \\ &= \frac{dJ}{dz_i} \cdot x_j\end{aligned}$$

- Calculando $\frac{dJ}{dz_i}$

$$\begin{aligned}\frac{dJ}{dz_i} &= \frac{dJ}{da_i} \cdot \frac{da_i}{dz_i} \\ &= \frac{dJ}{da_i} \cdot \sigma'(z_i)\end{aligned}$$

- Calculando $\frac{dJ}{da_i}$

$$\frac{dJ}{da_i} = \frac{dJ}{do} \cdot \frac{do}{da_i}$$

$$\left(\frac{dJ}{do} \right) \cdot w_{ij}^{[2]}$$

Retropropagação em uma rede de duas camadas

- Portanto, podemos concluir o processo para a primeira camada:

$$\begin{aligned}\frac{dJ}{dW_{ij}^{[1]}} &= \frac{dJ}{do} \cdot \frac{do}{da_i} \cdot \frac{da_i}{dz_i} \cdot \frac{dz_i}{dW_{ij}^{[1]}} \\ &= (o - y) \cdot W_i^{[2]} \cdot \sigma'(z_i) \cdot x_j\end{aligned}$$

- Em argumento similar podemos concluir que:

$$\begin{aligned}\frac{dJ}{db_i^{[1]}} &= \frac{dJ}{do} \cdot \frac{do}{da_i} \cdot \frac{da_i}{dz_i} \cdot \frac{dz_i}{db_i^{[1]}} \\ &= (o - y) \cdot W_i^{[2]} \cdot \sigma'(z_i)\end{aligned}$$

Retropropagação em uma rede de várias camadas

- Adotando a sistemática (pensando $x = a^{[0]}$):

$$a^{[1]} = \sigma^{[1]}(W^{[1]}a^{[0]} + b^{[1]})$$

$$a^{[2]} = \sigma^{[2]}(W^{[2]}a^{[1]} + b^{[2]})$$

...

$$a^{[r-1]} = \sigma^{[r-1]}(W^{[r-1]}a^{[r-2]} + b^{[r-1]})$$

$$a^{[r]} = \sigma^{[r]}(W^{[r]}a^{[r-1]} + b^{[r]})$$

$$J = \frac{1}{2}(a^{[r]} - y)^2$$

- Podemos adotar desenvolvimento similar àquele apresentado para a primeira camada da rede de duas camadas.

Algoritmo de Retropropagação em uma rede neural

Algorithm 8 Retropropagação

Entrada: X, Y : conjunto de atributos e classes

Saída: (W, b) : coeficientes da rede

```
1: repeat
2:   Escolher uma amostra  $(X', Y')$ 
3:    $a^{[0]} \leftarrow X'$ 
4:   for  $k = 1 \dots r$  do
5:     Calcular  $a^{[k]} = \sigma^{[k]}(W^{[k]}a^{[k-1]} + b^{[k]})$ 
6:   for  $k \leftarrow r \dots 1$  do
7:     if  $k = r$  then
8:       Calcular  $\delta^{[r]} = \frac{dJ}{dz^{[r]}} = (W^{[r]} \cdot (a^{[r]} - Y')) \cdot \sigma'^{[r]}(z^{[r]})$ 
9:     else
10:      Calcular  $\delta^{[k]} = \frac{dJ}{dz^{[k]}} = (W^{[k+1]} \cdot \delta^{[k+1]}) \cdot \sigma'^{[k]}(z^{[k]})$ 
11:      Calcular  $\frac{dJ}{dW^{[k]}} = \delta^{[k]} a^{[k]T}$ 
12:      Calcular  $\frac{dJ}{db^{[k]}} = \delta^{[k]}$ 
13:      Atualizar  $(W, b)$  de acordo com a descida de gradiente.
14: until convergência
15: Retorna  $(W, b)$ 
```

Agenda

Introdução

Naive Bayes

KNN

Árvores de decisão

Geração de regras

Métodos lineares

Support Vector Machines

Redes Neurais

Algoritmos de agrupamento

Agrupamento

- Definição: um grupo (*cluster*) é um subconjunto de dados que possui similaridades.
- Agrupamento, também chamado aprendizado não-supervisionado, é o processo de dividir um conjunto de dados em grupos cujos elementos em cada grupo sejam tão **similares** quanto possível e cujos elementos entre grupos sejam tão **diferentes** quanto possível.
- Este processo pode revelar relações imprevistas que podem auxiliar no aprendizado supervisionado.
- Porém diversos problemas reais são caracterizados como problemas de agrupamento.

Metodologias

Dois grupos principais de algoritmos de agrupamento podem ser identificados:

- Agrupamento hierárquico
 - Aglomerativo
 - Divisivo
- Agrupamento particional
 - *K-means*
 - EM
 - Mapa auto-organizado

Características de um bom agrupamento

Um bom algoritmo de agrupamento deve possuir as seguintes características:

- Descobrir alguns ou todos os grupos escondidos
- Similaridades dentro do grupo, dissimilaridades fora do grupo.
- Lidar com vários tipos de atributos.
- Lidar com ruído, falta de dados e *outliers*.
- Lidar com alta dimensionalidade.
- Escalabilidade, interpretabilidade e usabilidade.

Agrupamento hierárquico

- Dois tipos:
 - Aglomerativo
 - Divisivo
- O agrupamento divisivo é mais raro e de difícil aplicação.
- Em geral envolve métodos próprios para determinados tipos de problema.
- Deve ser capaz de traçar as características fundamentais que separam os grupos.

Agrupamento hierárquico aglomerativo

- Baseia-se na ideia de aglutinação de grupos.
- A qualquer momento do algoritmo vamos avaliar quais são os dois grupos mais **próximos** entre si e realizar uma operação de fusão de grupos.
- Com a fusão de dois grupos, o número de grupos é reduzido em uma unidade e o algoritmo continua até que reste apenas um grupo.
- Gera como saída n agrupamentos, cada um com $i = 1, \dots, n$ grupos.

Funções de distância

- Atributos numéricos:

- Linear: $\sum_{i=1}^k |x_i - y_i|$

- Euclidiana: $\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$

- Minkowski: $\left(\sum_{i=1}^k (|x_i - y_i|)^q \right)^{(\frac{1}{q})}$

- Atributos nominais:

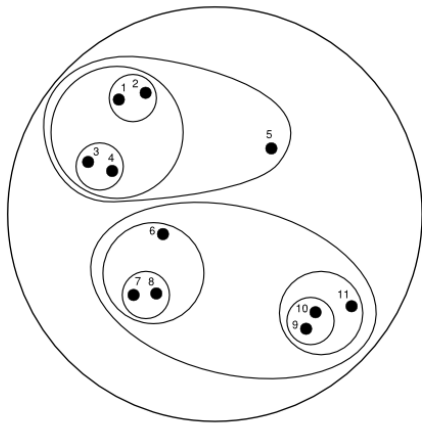
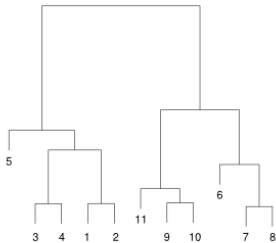
- Contagem de coincidências.

- Distância entre categorias (ordinais).

Distância entre grupos

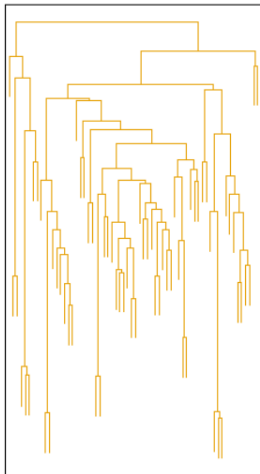
- Porém precisamos também medir a distância não apenas entre elemento, mas também a distância entre grupos.
- Algumas medidas típicas são:
 - Menor distância entre dois elementos (ligação simples).
 - Maior distância entre dois elementos (ligação completa).
 - Média das distâncias entre os elementos (ligação média).
 - Distância entre os **centroides** de cada grupo.

Exemplo

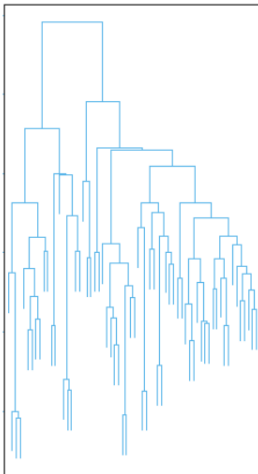


Exemplo

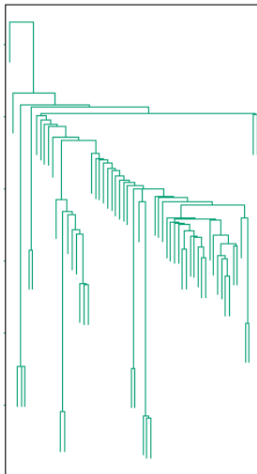
Average



Farthest



Nearest



Agrupamento particional: K-means

- O algoritmo *K-means* divide um conjunto de n instâncias passadas como parâmetro em k grupos, onde este valor é uma das entradas do algoritmo.
- Uma questão que acompanha este algoritmo é: qual o melhor valor de k de forma a minimizar a distância entre elementos de grupos diferentes?
- Não é conhecida uma resposta a priori, mas pode ser avaliada a posteriori conforme a execução do algoritmo para diferentes valores k .
- O algoritmo funciona através da criação de centroides artificiais para os grupos, que se deslocam de forma a minimizar a distância dos elementos associados àquele grupo.

K-means: definições

- Vamos chamar de c_r o centroide associado ao grupo r .
- Vamos também utilizar uma função de distância, assim como no agrupamento hierárquico, chamemos $D(x_i, x_j)$.
- Considere a variável α_{ri} indicando que a instância i foi associada ao grupo r . A função objetivo do algoritmo K-means é:

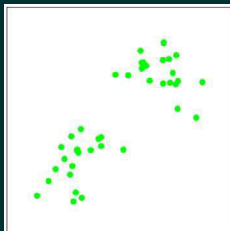
$$J(\alpha, c) = \sum_{r=1}^k \sum_{i=1}^n \alpha_{ri} \cdot D(x_i, c_r)$$

- Este problema é difícil de ser resolvido por um processo de otimização e o algoritmo K-means faz uma aproximação convergente a um mínimo local deste problema.

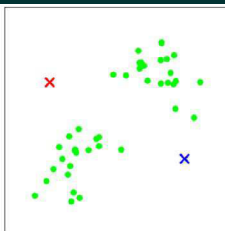
K-means: funcionamento

1. Partindo de um conjunto de centroides iniciais c_r , $r = 1, \dots, k$ (possivelmente aleatórios).
2. Calcula-se qual é a melhor associação α com c fixo, de maneira a minimizar a função $J(\alpha, c)$. Ou seja, para cada instância, qual centroide mais próximo dela.
3. Calcula-se então um valor de c com α fixo, de maneira a minimizar a função $J(\alpha, c)$. Ou seja, um reposicionamento dos centroides com um agrupamento fixo.
4. Repete-se os passos 2 e 3 até que não sejam mais observadas mudanças no valor de α (o que acarreta o fim das mudanças em c igualmente e portanto a convergência do processo).

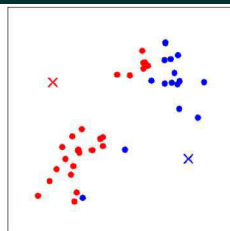
Exemplo K-means



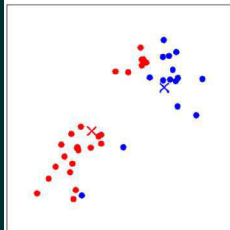
(a)



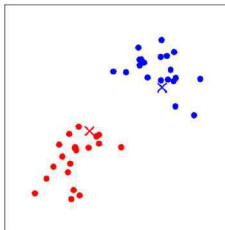
(b)



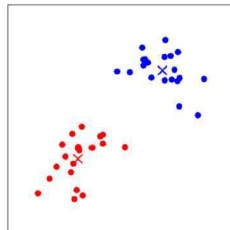
(c)



(d)



(e)



(f)

Aplicação compressão de imagens

K=2



K=3



K=10



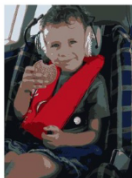
Original



4%



8%



17%



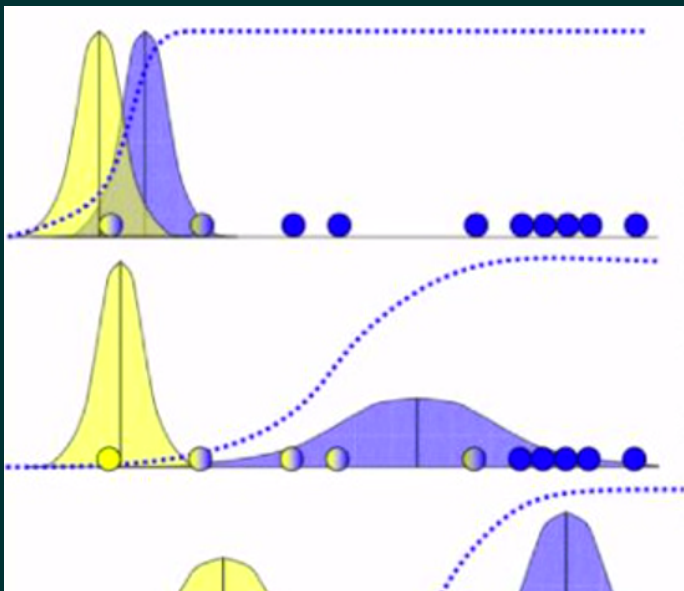
EM: Expectation Minimization

- Similarmente ao método K-means, o método EM consiste em descobrir modelos de dispersão para cada um dos grupos que desejamos encontrar.
- Também recebe como entrada um número fixo de grupos a serem formados.
- No entanto no método EM parte da premissa que os grupos possuem uma distribuição gaussiana dentro do espaço de características.
- Queremos portanto encontrar uma decomposição de máxima verossimilhança das instâncias em k distribuições gaussianas.

K-means: funcionamento

1. Partimos de distribuições gaussianas iniciais com parâmetros de média, variância e matriz de covariância possivelmente aleatórios.
2. Calcula-se qual é a probabilidade de cada instância para cada uma das distribuições consideradas (a priori).
3. A partir de uma redução bayesiana, calcula-se então qual é a probabilidade de cada ponto pertencer a cada uma das distribuições consideradas, dado que os pontos são explicados por alguma delas (a posteriori).
4. Recalcula-se os parâmetros de cada uma das distribuições consideradas, ponderando-se a média, a variância e matriz de covariância de acordo com as probabilidades (a posteriori) encontradas no passo anterior.
5. Repete-se os passos 2, 3 e 4 até que não sejam mais observadas mudanças significativas no valores das médias, variâncias e matrizes de covariância, o que acarreta a

Exemplo EM unidimensional



Exemplo EM bidimensional

