

exemplos (casos) de treinamento devem ser memorizados por um indutor *lazy* é muito importante. O ideal é reter apenas aqueles mais representativos do problema. Aha, Kibler, & Albert (1991) descrevem algumas estratégias para decidir quando um novo exemplo deve ser memorizado. *Nearest Neighbours* e Raciocínio Baseado em Casos (RBC) são, provavelmente, as técnicas mais conhecidas neste paradigma.

Conexionista Redes Neurais são construções matemáticas simplificadas inspiradas no modelo biológico do sistema nervoso. A representação de uma Rede Neural envolve unidades altamente interconectadas e, por esse motivo, o nome conexionismo é utilizado para descrever a área de estudo.

A metáfora biológica com as conexões neurais do sistema nervoso tem interessado muitos pesquisadores, e tem fornecido muitas discussões sobre os méritos e as limitações dessa abordagem de aprendizado. Em particular, as analogias com a biologia têm levado muitos pesquisadores a acreditar que as Redes Neurais possuem um grande potencial na resolução de problemas que requerem intenso processamento sensorial humano, tal como visão e reconhecimento de voz. Redes Neurais são tratadas com mais detalhes no Capítulo 6.

Evolutivo Este paradigma de aprendizado é derivado do modelo biológico de aprendizado (Goldberg 1989). Um classificador evolutivo consiste de uma população de elementos de classificação que competem para fazer a previsão. Elementos que possuem uma performance fraca são descartados, enquanto os elementos mais fortes proliferam, produzindo variações de si mesmos. Este paradigma possui uma analogia direta com a teoria de Darwin, na qual sobrevivem os mais bem adaptados ao ambiente. No Capítulo 9 este paradigma é tratado em mais detalhes.

4.4 Aprendizado Supervisionado: Conceitos e Definições

Alguns conceitos sobre a hierarquia de aprendizado foram introduzidos na seção anterior, fornecendo uma distinção entre paradigmas de aprendizado. Esta seção tem como objetivo descrever o problema de classificação no aprendizado supervisionado, assim como fornecer algumas definições de termos amplamente usados, tanto aqui como na literatura de AM.

Indutor Informalmente, o objetivo de um indutor (ou programa de aprendizado ou algoritmo de indução) consiste em extrair um bom classificador a partir de um conjunto de exemplos rotulados. A saída do indutor, o classificador, pode ser usada para classificar exemplos novos (ainda não rotulados) com a meta de predizer corretamente o rótulo de cada um. O classificador também pode ser avaliado considerando: precisão (Salzberg 1995; Dietterich 1997b), comprehensibilidade, grau de interesse (Freitas 1999), velocidade de aprendizado, requisitos de armazenamento, grau de compactação ou qualquer outra propriedade desejável que determine quão bom e apropriado ele é para a tarefa em questão. Embora este capítulo se concentre em classificação, vários aspectos dele se aplicam aos casos de regressão.

Exemplo Um exemplo, também denominado caso, registro ou dado na literatura, é uma tupla de valores de atributos (ou um vetor de valores de atributos). Um exemplo descreve o objeto de interesse: um paciente, dados médicos sobre uma determinada doença ou histórico de clientes de uma dada companhia.

Atributo Um atributo descreve uma característica ou um aspecto de um exemplo. Normalmente, há pelo menos dois tipos de atributos: nominal, quando não existe uma ordem entre os valores (por exemplo, cor: vermelho, verde, azul) e contínuo, quando existe uma ordem linear nos valores (por exemplo, peso $\in \mathbb{R}$, um número real).

Para qualquer tipo de atributo, usualmente existe também um símbolo importante que significa desconhecido, ou seja, a ausência de um valor para aquele atributo. Este símbolo especial é bem diferente, por exemplo, do valor zero (às vezes usado para números) ou de cadeias de caracteres vazias. Na maioria dos indutores disponíveis, este valor é representado por um ponto de interrogação '?'. Um outro símbolo especial, mesmo não sendo reconhecido por vários indutores, é o não-se-aplica. Por exemplo, para o atributo número de gestações, pode ser utilizado o símbolo não-se-aplica caso o paciente seja do sexo masculino. Normalmente, este símbolo é representado por um ponto de exclamação '!'.

Além disso, vários indutores assumem que os atributos originais que descrevem os exemplos são relevantes o suficiente para aprender a tarefa em questão. Entretanto, alguns atributos podem não ser diretamente relevantes e outros até irrelevantes. Um atributo é irrelevante se existe

uma descrição completa e consistente das classes a serem aprendidas que não usa aquele atributo (Michalski & Kaufman 1998).

Um ponto importante a ser considerado é a escolha de atributos com boa capacidade preditiva. Não importa qual método seja empregado, os conceitos que podem ser aprendidos estão à mercê dos dados e da qualidade dos atributos. Por exemplo, para a tarefa de determinar se uma pessoa está ou não com gripe, pode-se escolher atributos com baixo poder preditivo, tais como (cor-do-cabelo, cor-do-olho, modelo-do-carro, número-de-filhos) ou atributos com alto poder preditivo, tais como (temperatura, resistência-da-pele, exame-do-pulmão). Para esta tarefa específica, melhores previsões em exemplos não-rotulados provavelmente ocorrerão no segundo caso do que no primeiro conjunto de atributos.

Classe No aprendizado supervisionado, todo exemplo possui um atributo especial, denominado rótulo ou classe, que descreve o fenômeno de interesse, isto é, o conceito-meta que se deseja aprender para fazer previsões a respeito. Como já mencionado, os rótulos são tipicamente pertencentes a um conjunto discreto (nominal) de classes $\{C_1, C_2, \dots, C_k\}$ no caso de classificação, ou de valores reais no caso de regressão.

Conjunto de Exemplos Um conjunto de exemplos é composto por exemplos contendo valores de atributos bem como a classe associada. Na Tabela 4.1 é mostrado o formato padrão de um conjunto de exemplos T com n exemplos e m atributos. Nessa tabela, a linha i refere-se ao i -ésimo exemplo ($i = 1, 2, \dots, n$) e a entrada x_{ij} refere-se ao valor do j -ésimo ($j = 1, 2, \dots, m$) atributo X_j do exemplo i .

Tabela 4.1: Conjunto de exemplos no formato atributo-valor

	X_1	X_2	\dots	X_m	Y
T_1	x_{11}	x_{12}	\dots	x_{1m}	y_1
T_2	x_{21}	x_{22}	\dots	x_{2m}	y_2
\vdots	\vdots	\vdots	\ddots	\vdots	\vdots
T_n	x_{n1}	x_{n2}	\dots	x_{nm}	y_n

Como pode ser notado, exemplos são tuplas $T_i = (x_{i1}, x_{i2}, \dots, x_{im}, y_i) = (\vec{x}_i, y_i)$, também denotados por (x_i, y_i) , onde fica subentendido o fato que

x_i é um vetor. A última coluna, Y , contém o atributo meta, também chamado de classe. Observa-se que cada x_i é um elemento do conjunto $\text{dom}(X_1) \times \text{dom}(X_2) \times \dots \times \text{dom}(X_m)$, onde $\text{dom}(X_j)$ é o domínio do atributo X_j e y_i pertence a uma das k classes, isto é, $y_i \in \{C_1, C_2, \dots, C_k\}$.

Usualmente, um conjunto de exemplos é dividido em dois subconjuntos disjuntos: o conjunto de treinamento usado para o aprendizado do conceito e o conjunto de teste usado para medir o grau de efetividade do conceito aprendido. Os subconjuntos são normalmente disjuntos para assegurar que as medidas obtidas, utilizando o conjunto de teste, sejam de um conjunto diferente do usado para realizar o aprendizado, tornando a medida estatisticamente válida.

Classificador ou Hipótese Dado um conjunto de exemplos de treinamento, um indutor gera como saída um classificador (também denominado hipótese ou descrição de conceito) de forma que, dado um novo exemplo, ele possa predizer com a maior precisão possível sua classe.

Formalmente, em classificação, um exemplo é um par $(x_i, f(x_i))$ onde x_i é a entrada e $f(x_i)$ é a saída. A tarefa de um indutor é, dado um conjunto de exemplos, induzir uma função h que aproxima f , normalmente desconhecida. Neste caso, h é chamada hipótese sobre a função objetivo f , ou seja, $h(x_i) \approx f(x_i)$.

Ruído É comum, no mundo real, trabalhar com dados imperfeitos. Eles podem ser derivados do próprio processo que gerou os dados, do processo de aquisição de dados, do processo de transformação ou mesmo devido a classes rotuladas incorretamente (por exemplo, exemplos com os mesmos valores de atributos mas com classes diferentes). Neste caso, diz-se que existe ruído nos dados.

Conhecimento do Domínio Em geral, o conhecimento prévio do domínio, ou *conhecimento do domínio*, inclui informação a respeito dos valores válidos dos atributos, um critério de preferência para a escolha entre possíveis atributos ou mesmo hipóteses. Nem todos os indutores são capazes de utilizar conhecimento do domínio quando aprendem conceitos, isto é, eles usam apenas os exemplos fornecidos. Observe que o número de unidades ocultas e conexões, assim como a topologia em uma Rede Neural (vide Capítulo 6), é uma forma de conhecimento do domínio.

Bias Qualquer preferência de uma hipótese sobre outra, além da simples consistência com os exemplos, é denominada *bias* de aprendizado. Devido ao fato de quase sempre existir um número grande de hipóteses consistentes, todos os indutores possuem alguma forma de *bias*. De fato, aprendizado sem *bias* é impossível (Mitchell 1982).

Modo de Aprendizado Sempre que todo o conjunto de treinamento esteja presente no aprendizado, o modo de aprendizado de um algoritmo é não-incremental, também conhecido como modo *batch*. Por outro lado, se o indutor não necessita construir a hipótese a partir do início, quando novos exemplos são adicionados ao conjunto de treinamento, o modo de aprendizado é incremental. Portanto, no modo incremental o indutor apenas tenta atualizar a hipótese antiga sempre que novos exemplos são adicionados ao conjunto de treinamento.

Espaço de Descrição Um exemplo $(x_{i1}, x_{i2}, \dots, x_{im}, y_i) = (\vec{x}_i, y_i)$ possui m atributos $x_{i1}, x_{i2}, \dots, x_{im}$ que podem ser interpretados como um vetor \vec{x}_i , no qual cada atributo x_{ij} corresponde a uma coordenada em um espaço m -dimensional (ou espaço de descrição), onde $1 \leq i \leq n$ e n é o número de exemplos. Além disso, cada ponto no espaço de descrição pode ser rotulado com a correspondente classe y_i associada.

Sob este ponto de vista, um classificador divide este espaço em regiões, cada região rotulada com uma classe. Um novo exemplo é classificado determinando-se a região onde o ponto correspondente se localiza e atribuindo ao exemplo a classe associada com aquela região.

Erro e Precisão Uma medida de desempenho comumente usada é a taxa de erro de um classificador h , também conhecida como taxa de classificação incorreta e denotada por $\text{err}(h)$. Usualmente, a taxa de erro é obtida utilizando a Equação (4.1), a qual compara a classe verdadeira de cada exemplo com o rótulo atribuído pelo classificador induzido. O operador $\| E \|$ retorna 1 se a expressão E for verdadeira, e zero caso contrário; n é o número de exemplos. O complemento da taxa de erro, a precisão do classificador, denotada por $\text{acc}(h)$ é dada pela Equação (4.2).

$$\text{err}(h) = \frac{1}{n} \sum_{i=1}^n \| y_i \neq h(x_i) \| \quad (4.1)$$

$$\text{acc}(h) = 1 - \text{err}(h) \quad (4.2)$$

Para problemas de regressão, o erro da hipótese (err) pode ser estimado calculando-se a distância entre o valor real com o atribuído pela hipótese induzida. Usualmente, duas medidas são comumente usadas: o erro quadrático médio (mse-err ou *mean squared error*) e a distância absoluta média (mad-err ou *mean absolute distance*), dadas pelas Equações (4.3) e (4.4), respectivamente.

$$\text{mse-err}(h) = \frac{1}{n} \sum_{i=1}^n (y_i - h(x_i))^2 \quad (4.3)$$

$$\text{mad-err}(h) = \frac{1}{n} \sum_{i=1}^n |y_i - h(x_i)| \quad (4.4)$$

Em geral, o erro calculado sobre o conjunto de exemplos de treinamento (erro aparente) é menor que o erro calculado sobre o conjunto de exemplos de teste (erro verdadeiro).

Distribuição de Classes Dado um conjunto de exemplos T , é possível calcular sua distribuição de classes no caso de classificação. Para cada classe C_j sua distribuição $\text{distr}(C_j)$ é calculada como sendo o número de exemplos em T que possuem classe C_j dividido pelo número total de exemplos n , ou seja, a proporção de exemplos em cada classe, dada pela Equação (4.5).

$$\text{distr}(C_j) = \frac{1}{n} \sum_{i=1}^n \|y_i = C_j\| \quad (4.5)$$

Por exemplo, se um conjunto com 100 exemplos possui 60 exemplos da classe C_1 , 15 exemplos da classe C_2 e 25 exemplos da classe C_3 então sua distribuição de classes é $\text{distr}(C_1, C_2, C_3) = (0,60, 0,15, 0,25) = (60\%, 15\%, 25\%)$. Nesse exemplo, a classe C_1 é a classe majoritária ou prevalente e a classe C_2 é a classe minoritária.

Erro Majoritário Uma vez calculada a distribuição de classes em um conjunto de exemplos T , pode-se calcular seu erro majoritário utilizando a Equação (4.6).

$$\text{maj-err}(T) = 1 - \max_{i=1,\dots,k} \text{distr}(C_i) \quad (4.6)$$

Dado o exemplo anterior, o erro majoritário é $\text{maj-err}(T) = 1 - 0,60 = 40\%$. Como pode ser observado, o erro majoritário de um conjunto de exemplos é independente do algoritmo de aprendizado. Ele fornece um limiar máximo abaixo do qual o erro de um classificador deve ficar.

Prevalência de Classe Um ponto muito importante em AM refere-se ao desbalanceamento de classes em um conjunto de exemplos. Por exemplo, suponha um conjunto de exemplos T com a seguinte distribuição de classes $\text{dist}(C_1, C_2, C_3) = (99\%, 0,25\%, 0,75\%)$, com prevalência da classe C_1 . Um classificador simples que classifique sempre novos exemplos como pertencentes à classe majoritária C_1 teria uma precisão de 99% ($\text{maj-err}(T) = 1\%$). Isto pode ser indesejável quando as classes minoritárias possuem uma informação muito importante, por exemplo, supondo C_1 : paciente normal, C_2 : paciente com doença A e C_3 : paciente com doença B.

Under e Overfitting Ao induzir, a partir dos exemplos disponíveis, é possível que a hipótese seja muito específica para o conjunto de treinamento utilizado. Como o conjunto de treinamento é apenas uma amostra de todos os exemplos do domínio, é possível induzir hipóteses que melhorem seu desempenho no conjunto de treinamento, enquanto pioram o desempenho em exemplos diferentes daqueles pertencentes ao conjunto de treinamento. Nesta situação, o erro (ou outra medida) em um conjunto de teste independente evidencia um desempenho ruim da hipótese. Neste caso, diz-se que a hipótese ajusta-se em excesso ao conjunto de treinamento ou que houve um *overfitting*.

Por outro lado, também é possível que poucos exemplos representativos sejam dados ao sistema de aprendizado, ou o usuário predefina o tamanho do classificador como muito pequeno (por exemplo, um número insuficiente de neurônios e conexões são definidos em uma Rede Neural ou um alto fator de poda é definido para uma árvore de decisão) ou uma

combinação de ambos. Portanto, é também possível induzir hipóteses que possuam uma melhora de desempenho muito pequena no conjunto de treinamento, assim como em um conjunto de teste. Neste caso, diz-se que a hipótese ajusta-se muito pouco ao conjunto de treinamento ou que houve um *underfitting*.

Overtuning O excesso de ajuste de um algoritmo, ou *overtuning*, pode ocorrer quando o desenvolvedor ajusta um algoritmo de aprendizado, ou os seus parâmetros, muito bem para otimizar seu desempenho em todos os exemplos disponíveis. Da mesma forma que *overfitting*, *overtuning* pode ser detectado e evitado utilizando-se apenas uma parte dos exemplos disponíveis para a execução do indutor e o restante dos exemplos para um teste final do classificador induzido. De fato, assim como o *overfitting*, se os exemplos não são separados em conjuntos de treinamento e teste, permitindo uma avaliação final independente, o desempenho do sistema não pode ser usado de forma confiável como uma estimativa do desempenho futuro do sistema.

Poda Poda é uma técnica de lidar com ruído e *overfitting*. A idéia geral consiste em lidar com o problema de *overfitting* por meio do aprendizado de uma hipótese bem genérica a partir do conjunto de treinamento melhorando o desempenho em exemplos não vistos. Há, basicamente, dois métodos de poda:

1. pré-poda: durante a geração da hipótese, alguns exemplos de treinamento são deliberadamente ignorados, de maneira que a hipótese final não classifique todos os exemplos de treinamento corretamente;
2. pós-poda: inicialmente, uma hipótese que explica os exemplos de treinamento é gerada. Após isso, a hipótese é generalizada por meio da eliminação de algumas partes, tais como o corte de alguns ramos em uma árvore de decisão, de algumas condições nas regras induzidas ou neurônios redundantes em uma Rede Neural.

Completude e Consistência Uma vez induzida, uma hipótese pode ser avaliada a respeito de sua completude, se ela classifica todos os exemplos, e de sua consistência, se ela classifica corretamente os exemplos. Portanto, dada uma hipótese, ela pode ser: (a) completa e consistente; (b) incompleta e consistente; (c) completa e inconsistente ou (d) incompleta e inconsistente. Na Figura 4.3 é dado um exemplo dos quatro casos e inconsistente.

mencionados, considerando dois atributos X_1 e X_2 , três classes (o , $+$, $*$), bem como as hipóteses induzidas para cada classe. Elas são representadas por duas regiões indicadas por linhas sólidas para a classe o , duas regiões indicadas por linhas pontilhadas para a classe $+$ e uma região indicada por linhas tracejadas para a classe $*$.

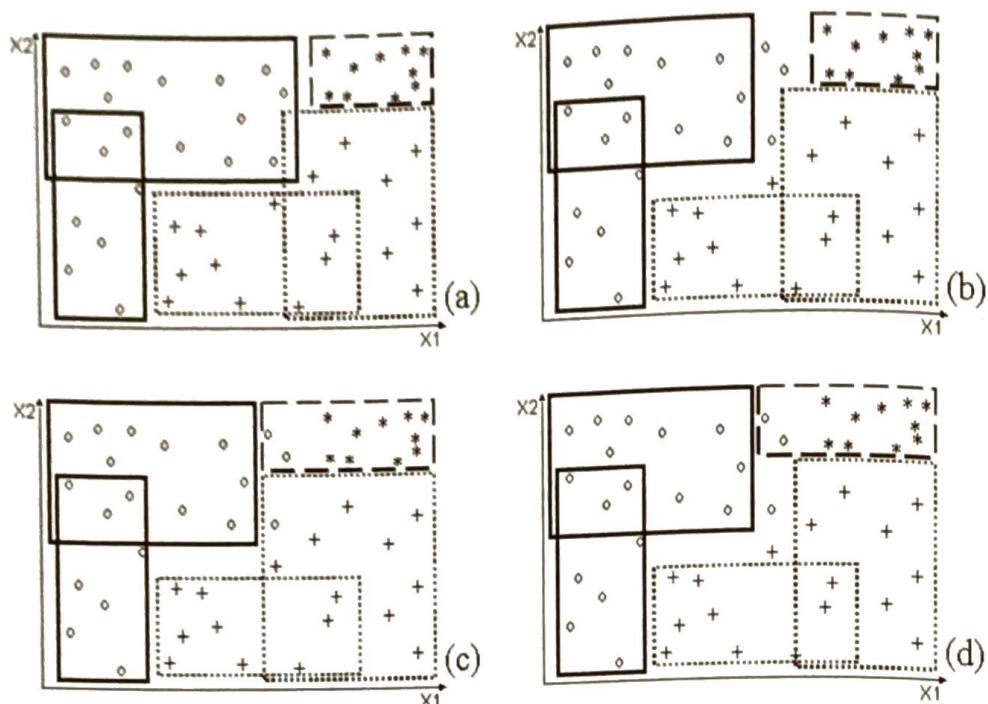


Figura 4.3: Completude e consistência de uma hipótese

Matriz de Confusão A matriz de confusão de uma hipótese h oferece uma medida efetiva do modelo de classificação, ao mostrar o número de classificações corretas *versus* as classificações preditas para cada classe, sobre um conjunto de exemplos T . Como mostrados na Tabela 4.2, os resultados são totalizados em duas dimensões: classes verdadeiras e classes preditas, para k classes diferentes $\{C_1, C_2, \dots, C_k\}$. Cada elemento $M(C_i, C_j)$ da matriz, $i, j = 1, 2, \dots, k$, calculado pela Equação (4.7), representa o número de exemplos de T que realmente pertencem à classe C_i , mas foram classificados como sendo da classe C_j .

Tabela 4.2: Matriz de confusão de um classificador

Classe	Predita C_1	Predita C_2	...	Predita C_k
verdadeira C_1	$M(C_1, C_1)$	$M(C_1, C_2)$...	$M(C_1, C_k)$
verdadeira C_2	$M(C_2, C_1)$	$M(C_2, C_2)$...	$M(C_2, C_k)$
:	:	:	..	:
verdadeira C_k	$M(C_k, C_1)$	$M(C_k, C_2)$...	$M(C_k, C_k)$

$$M(C_i, C_j) = \sum_{\{\forall(x,y) \in T : y=C_i\}} \| h(x) = C_j \| \quad (4.7)$$

O número de acertos, para cada classe, se localiza na diagonal principal $M(C_i, C_i)$ da matriz. Os demais elementos $M(C_i, C_j)$, para $i \neq j$, representam erros na classificação. A matriz de confusão de um classificador ideal possui todos esses elementos iguais a zero uma vez que ele não comete erros.

Por simplicidade, considere um problema de duas classes. Com apenas duas classes, usualmente rotuladas como “+” (positiva) e “-” (negativa), as escolhas estão estruturadas para predizer a ocorrência ou não de um evento simples. Neste caso, os dois erros possíveis são denominados falso positivo (F_P) e falso negativo (F_N). Na Tabela 4.3 é ilustrada a matriz de confusão para o problema com duas classes, onde T_P é o número de exemplos positivos classificados corretamente e T_N é o número de exemplos negativos classificados corretamente do total de $n = (T_P + F_N + F_P + T_N)$ exemplos.

Tabela 4.3: Matriz de confusão para a classificação com duas classes

Classe	Predita C_+	Predita C_-	Taxa de Erro da Classe	Taxa de Erro Total
verdadeira C_+	Verdeiros positivos T_P	Falsos negativos F_N	$\frac{F_N}{T_P+F_N}$	$\frac{F_P+F_N}{n}$
	Falsos positivos F_P	Verdeiros negativos T_N	$\frac{F_P}{F_P+T_N}$	
verdadeira C_-				

Outras medidas podem ser derivadas a partir da matriz de confusão, tais como confiabilidade positiva prel (positive reliability ou C_+ predictive value), confiabilidade negativa nrel (negative reliability ou C_- predictive value), suporte sup, sensitividade sens (sensitivity ou recall ou true C_+ value), especificidade spec (specificity ou true C_- rate), precisão total tacc rate), especificidade cov (coverage) calculadas a partir das Equações (4.8) à (4.14), respectivamente (Weiss & Kulikowski 1991).

$$\text{prel}(h) = \frac{T_P}{T_P + F_P} \quad (4.8)$$

$$\text{nrel}(h) = \frac{T_N}{T_N + F_N} \quad (4.9)$$

$$\text{sup}(h) = \frac{T_P}{n} \quad (4.10)$$

$$\text{sens}(h) = \frac{T_P}{T_P + F_N} \quad (4.11)$$

$$\text{spec}(h) = \frac{T_N}{T_N + F_P} \quad (4.12)$$

$$\text{tacc}(h) = \frac{T_P + T_N}{n} \quad (4.13)$$

$$\text{cov}(h) = \frac{T_P + F_P}{n} \quad (4.14)$$

Custos de Erro Medir adequadamente o desempenho de classificadores, por meio da taxa de erro (ou precisão) assume um papel importante em Aprendizado de Máquina, uma vez que o objetivo consiste em construir classificadores com baixa taxa de erro em novos exemplos (Weiss & Kulikowski 1991). Entretanto, ainda considerando o problema anterior contendo duas classes, se o custo de ter falsos positivos e falsos negativos não é o mesmo, então outras medidas de desempenho devem ser usadas. Uma alternativa natural, quando cada tipo de classificação incorreta possui um custo diferente ou mesmo quando existe prevalência de classes, consiste em associar um custo para cada tipo de erro.

O custo, denotado por $\text{cost}(C_i, C_j)$, é um número que representa uma penalidade aplicada quando o classificador faz um erro ao rotular exemplos, cuja classe verdadeira é C_i , como pertencentes à classe C_j , onde $i, j = 1, 2, \dots, k$ e k é o número de classes. Assim, $\text{cost}(C_i, C_i) = 0$, uma vez que não constitui um erro e $\text{cost}(C_i, C_j) > 0$, $i \neq j$. Em geral, os induutores assumem que $\text{cost}(C_i, C_j) = 1$, $i \neq j$, caso esses valores não sejam definidos explicitamente.

No cálculo utilizando custos, os erros são convertidos em custos pela multiplicação do erro pelo custo correspondente, calculados a partir da

Equação (4.15), onde n representa o número de exemplos. É também possível obter os custos por meio da matriz de confusão utilizando-se a Equação (4.16). Assim, em vez de projetar um algoritmo que minimize a taxa de erro, o objetivo poderia ser minimizar custos de classificação incorreta (Domingos 1999).

$$\text{err-cost}(h) = \frac{1}{n} \sum_{i=1}^n \| y_i \neq h(x_i) \| \times \text{cost}(y_i, h(x_i)) \quad (4.15)$$

$$\text{err-cost}(h) = \frac{1}{n} \sum_{i=1}^k \sum_{j=1}^k M(C_i, C_j) \times \text{cost}(C_i, C_j) \quad (4.16)$$

Complexo É uma disjunção de conjunções de testes de atributos na forma: $X_i \text{ op Valor}$, onde X_i é um atributo, op é um operador pertencente ao conjunto $\{=, \neq, <, \leq, >, \geq\}$ e Valor é um valor constante válido para o atributo X_i . Para atributos contínuos, é também possível ter uma combinação linear de atributos na forma: $a_1 \times X_1 + a_2 \times X_2 + \dots + a_m \times X_m \text{ op Valor}$, onde a_i é uma constante, X_i é um atributo contínuo (inteiro ou real), op é um operador pertencente ao conjunto $\{<, \leq, >, \geq\}$ e Valor é uma constante.

Regra Assume a forma if L then R ou na forma simbólica $L \rightarrow R$ que também pode ser representada pelas formas equivalentes $R \leftarrow L$ ou $R :- L$. Normalmente, as partes esquerda L e direita R da regra são complexos, sem atributos comuns entre eles, ou seja, $\text{atributos}(L) \cap \text{atributos}(R) = \emptyset$. A parte esquerda L é denominada condição, premissa, cauda ou corpo da regra, e a parte direita R é denominada conclusão ou cabeça da regra.

Regra de Classificação Uma regra de classificação assume a forma restrita de uma regra if L then classe = C_i , onde C_i pertence ao conjunto de k valores de classe $\{C_1, C_2, \dots, C_k\}$.

Regra de Associação Uma regra de associação assume que não existe uma definição explícita de classe e qualquer atributo (ou atributos) pode ser usado como parte da conclusão da regra. Exemplo: if $X_3 = 'S'$ and $X_5 > 2$ then $X_1 = 'N'$ and $X_2 < 1$.

Cobertura Considerando uma regra $L \rightarrow R$, os exemplos que satisfazem a parte L da regra constituem o seu conjunto de cobertura, ou seja, os

exemplos que são cobertos pela regra ou que a regra dispara para esses exemplos. Exemplos que satisfazem tanto a condição L como a conclusão R são cobertos corretamente pela regra (verdadeiros positivos). Já os exemplos que satisfazem a condição L mas não a conclusão R são cobertos incorretamente (falsos positivos) pela regra. Por outro lado, os exemplos que não satisfazem a condição L não são cobertos pela regra.

Matriz de Contingência Como descrito anteriormente, a matriz de confusão é aplicada ao classificador visto como uma caixa-preta, ou seja, o classificador pode ser simbólico ou não para se calcular essa matriz. Já a matriz de contingência, uma generalização da matriz de confusão binária, é calculada para cada regra, exigindo, desta maneira, que o classificador seja simbólico.

Linguagens de Representação Ao solucionar problemas com o uso de computadores é importante definir como traduzi-los em termos computacionais. Especificamente, em AM isto significa como representar exemplos, hipóteses e conhecimento do domínio. Para descrevê-los, as seguintes linguagens de representação (ou linguagens de descrição) são usadas:

- Linguagem de Representação de Exemplos (LRE);
- Linguagem de Representação de Hipóteses (LRH);
- Linguagem de Representação de Conhecimento do Domínio (LRC).

Na Tabela 4.4 são consideradas as linguagens de representação de alguns indutores, freqüentemente encontrados na literatura.

Tabela 4.4: Linguagens de representação de alguns indutores

Indutor	LRE	LRH	LRC
C4.5	atributo-valor	atributo-valor	
CART	atributo-valor	atributo-valor	
CN2	atributo-valor	atributo-valor	
RIPPER	atributo-valor	atributo-valor	atributo-valor
FOIL	atributo-valor	primeira ordem	primeira ordem
PROGOL	atributo-valor	primeira ordem	primeira ordem
rede neural	atributo-valor	funções matemáticas	

No Capítulo 5 são descritas algumas linguagens de representação freqüentemente utilizadas em AM simbólico em ordem crescente de com-

plexidade e força expressiva. Funções matemáticas podem ser usadas para descrever hipóteses, por exemplo para Redes Neurais, tratadas no Capítulo 6. Neste caso, o espaço de representação é dividido em regiões complexas por meio da combinação de várias funções matemáticas.

Na Tabela 4.5 estão resumidas algumas características gerais dos sistemas de Aprendizado de Máquina discutidos neste capítulo.

Tabela 4.5: Características gerais de sistemas de Aprendizado de Máquina

Modos de Aprendizado	Paradigmas de Aprendizado	Linguagens de Descrição	Formas de Aprendizado
Supervisionado	Simbólico	LRE - Exemplos ou Objetos	Incremental
Não-Supervisionado	Estatístico	LRH - Hipóteses	Não-Incremental
	Baseado em Instâncias	LRC - Conhecimento do Domínio	
	Conexionista		
	Evolutivo		

4.5 Avaliação de Algoritmos

Aprendizado de Máquina é uma ferramenta poderosa, mas não existe um único algoritmo que apresente o melhor desempenho para todos os problemas (Dietterich 1997a; Kohavi, Sommerfield, & Dougherty 1996; Schaffer 1994). Dessa forma, é importante compreender o poder e a limitação dos diferentes algoritmos utilizando alguma metodologia de avaliação que permita comparar algoritmos. Para tanto, nesta seção é descrita uma metodologia de avaliação, freqüentemente utilizada pela comunidade de AM, para comparar dois algoritmos, baseando-se na idéia de amostragem (*resampling*), explicada na próxima seção.

Métodos de Amostragem

Dados um conjunto de exemplos de tamanho finito e um indutor, é importante estimar o desempenho futuro do classificador induzido utilizando o conjunto de exemplos. Todos os métodos não-paramétricos descritos a seguir, exceto o

método de resubstituição, estão baseados na idéia de amostragem, ilustrada na Figura 4.4. O mundo real apresenta uma distribuição de exemplos D em um dado domínio, a qual é desconhecida. Ao extrair exemplos do mundo real, formando assim um conjunto de exemplos, obtém-se uma distribuição de exemplos D' , a qual é supostamente similar à distribuição D . De modo a estimar uma medida, geralmente a precisão ou o erro, de indutores treinados com base na distribuição D' , extraem-se amostras a partir de D' , treina-se um indutor com essas amostras e testa-se seu desempenho em exemplos de D' (normalmente com exemplos fora da amostra utilizada para treinamento). Desta maneira, simula-se o processo de amostragem que ocorre no mundo real, assumindo que D' representa o mundo real.

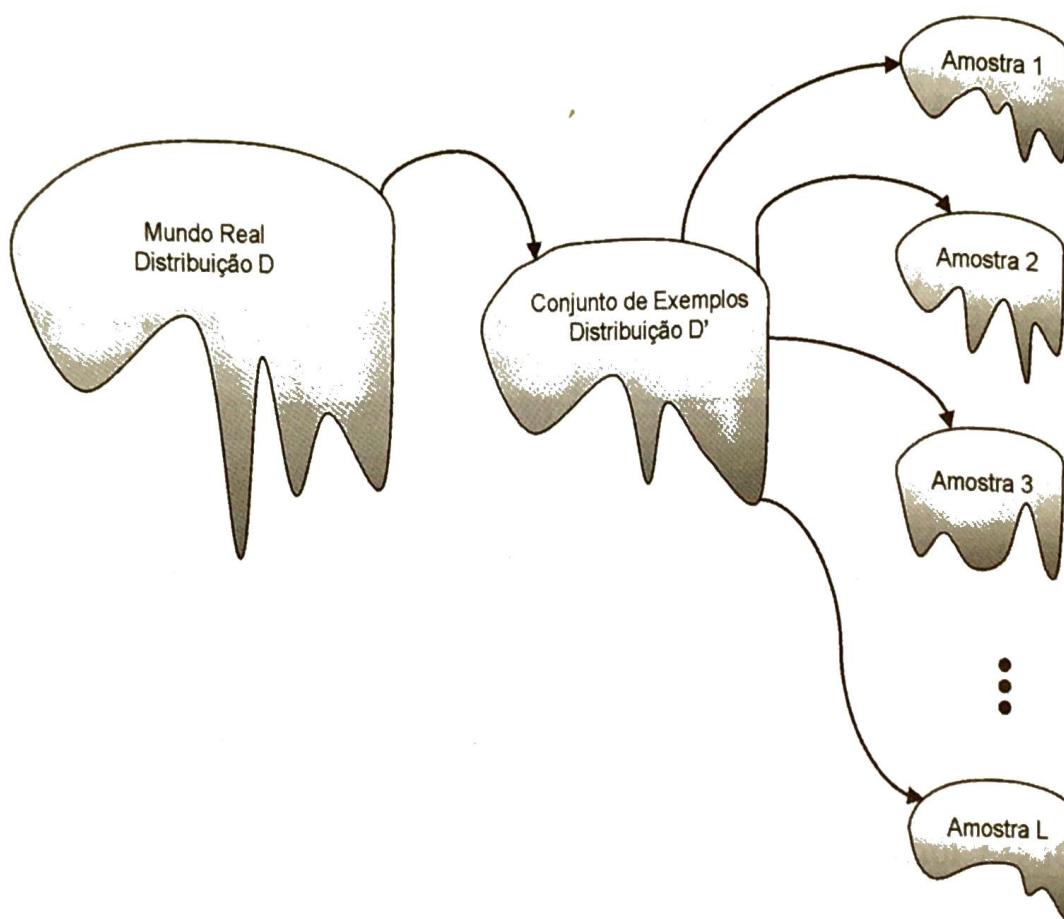


Figura 4.4: Idéia de amostragem

É importante, ao estimar uma medida verdadeira (por exemplo, o erro verdadeiro, conforme definição na seção anterior, Erro e Precisão), que a amostra seja aleatória, isto é, os exemplos não devem ser pré-selecionados. Para problemas reais, normalmente é tomada uma amostra de tamanho n e o objetivo consiste em estimar uma medida para aquela população em particular (não para todas as populações). Existem vários métodos para estimar uma medida verdadeira, os quais são descritos a seguir e resumidos na Tabela 4.6.

Tabela 4.6: Alguns parâmetros típicos de estimadores, onde n representa o número de exemplos, r o número de folds (partições), p um número tal que $0 < p < 1$, t um número tal que $0 < t < n$ e L o número de hipóteses induzidas

	Holdout	Aleatória	Leave-One-Out	r-Fold CV	r-Fold-Strat CV	Bootstrap
Treinamento	pn	t	$n - 1$	$n(r - 1)/r$		
Teste	$(1 - p)n$	$n - t$	1		$n(r - 1)/r$	n
Iterações	1	$L \ll n$	n	n/r	n/r	$n - t$
Reposição	não	não	não	r	r	≈ 200
Prevalência de Classe	não	não	não	não	não	sim
				n	sim	sim/não

Resubstituição: o método de resubstituição consiste em construir o classificador e testar seu desempenho no mesmo conjunto de exemplos, ou seja, o conjunto de teste é idêntico ao conjunto de treinamento. Este estimador fornece uma medida aparente, possuindo uma estimativa altamente otimista da precisão, pois o processo de classificação tenta maximizá-la. Para muitos algoritmos de indução que classificam corretamente todos os exemplos, como 1-Nearest Neighbors ou Árvores de Decisão sem poda, esta estimativa é muito otimista: se não houver exemplos conflitantes, a estimativa de precisão atinge 100%. Assim, o desempenho calculado com este método possui um *bias* otimista, ou seja, o bom desempenho no conjunto de treinamento em geral não se estende a conjuntos independentes de teste.

Quando o *bias* do estimador de resubstituição foi descoberto, diversos métodos de reamostragem foram propostos, os quais são descritos a seguir. Todos eles estão baseados no mesmo princípio: não deve haver exemplos em comum entre o conjunto de treinamento (ou aprendizado) e o conjunto de teste. Alguns desses métodos de reamostragem são:

Holdout: O estimador *holdout* divide os exemplos em uma porcentagem fixa de exemplos p para treinamento e $(1-p)$ para teste, considerando normalmente $p > 1/2$. Valores típicos são $p = 2/3$ e $(1 - p) = 1/3$, embora não existam fundamentos teóricos sobre estes valores.

Amostragem Aleatória: Na amostragem aleatória, $L \ll n$, são induzidas a partir de cada conjunto de treinamento. O erro final é a média dos erros de todas as hipóteses induzidas calculados em conjuntos de teste independentes e extraídos aleatoriamente. Amostragem aleatória pode produzir melhores estimativas de erro que o estimador *holdout*.

Cross-Validation: Este estimador é um meio termo entre os estimadores *holdout* e *leave-one-out*, descrito posteriormente. Em *r-fold cross-validation* — CV — os exemplos são aleatoriamente divididos em *r* partições mutuamente exclusivas (*folds*) de tamanho aproximadamente igual a n/r exemplos. Os exemplos nos $(r - 1)$ *folds* são usados para treinamento e a hipótese induzida é testada no *fold* remanescente. Este processo é repetido *r* vezes, cada vez considerando um *fold* diferente para teste. O erro na *cross-validation* é a média dos erros calculados em cada um dos *r folds*. Na Figura 4.5 é mostrada a porcentagem de exemplos compartilhados ao usar o *cross-validation*.

Stratified Cross-Validation: O estimador *stratified cross-validation* é similar à *cross-validation*, mas ao gerar os *folds* mutuamente exclusivos, a distribuição de classe — a proporção de exemplos em cada uma das classes — é considerada durante a amostragem. Isto significa, por exemplo, que se o conjunto original de exemplos possui duas classes com distribuição de 20% e 80%, cada *fold* também terá esta proporção de classes.

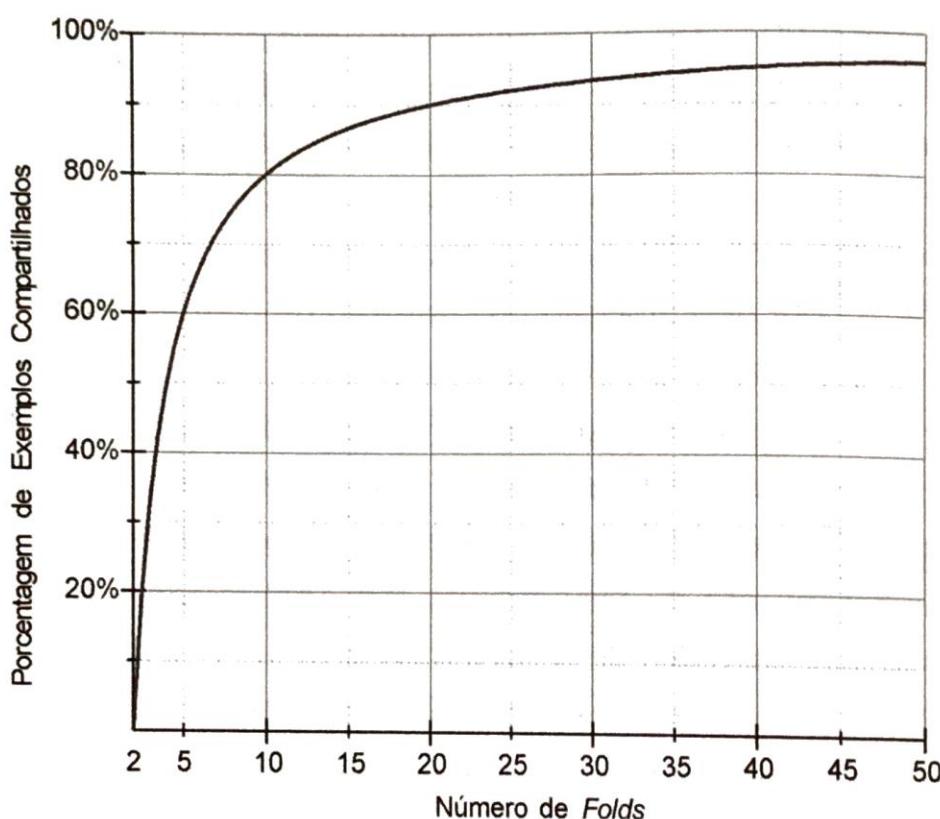


Figura 4.5: Número de *folds* versus porcentagem de exemplos compartilhados em *cross-validation*

Leave-One-Out: O estimador *leave-one-out* é um caso especial de *cross-validation*. É computacionalmente dispendioso e freqüentemente usado em amostras pequenas. Para uma amostra de tamanho n uma hipótese é indu-

zida utilizando $(n - 1)$ exemplos; a hipótese é então testada no único exemplo remanescente. Este processo é repetido n vezes, cada vez induzindo uma hipótese deixando de considerar um único exemplo. O erro é a soma dos erros em cada teste dividido por n .

Bootstrap: No estimador *bootstrap*, a idéia básica consiste em repetir o processo de classificação diversas vezes (Efron & Tibshirani 1993). Estima-se então valores, como o erro ou *bias*, a partir dos experimentos replicados; cada experimento sendo conduzido com base em um novo conjunto de treinamento obtido por amostragem com reposição do conjunto original de exemplos.

Há muitos estimadores *bootstrap*, sendo o mais comum denominado *bootstrap e0*. Um conjunto de treinamento *bootstrap* consiste de n exemplos (ou seja, com o mesmo tamanho do conjunto original de exemplos) amostrados com reposição a partir do conjunto original de exemplos. Isto significa que alguns exemplos T_i podem não aparecer no conjunto de treinamento *bootstrap* e alguns T_i podem aparecer mais de uma vez. Os exemplos remanescentes (aqueles que não aparecem no conjunto de treinamento *bootstrap*) são usados como conjunto de teste. Para uma dada amostra *bootstrap*, um exemplo de treinamento tem probabilidade $1 - (1 - 1/n)^n$ de ser selecionado pelo menos uma vez em cada uma das n vezes nas quais os exemplos são aleatoriamente selecionados a partir do conjunto original de exemplos. Para um número alto de n resultaria em aproximadamente $1 - 1/e = 0,632$.

Portanto, para esta técnica, a fração média de exemplos não-repetidos é 63,2% no conjunto de treinamento e 36,8% no conjunto de teste. Geralmente, o processo de *bootstrap* é repetido um número de vezes, sendo o erro estimado como a média dos erros sobre o número de iterações.

4.6 Desempenho de Algoritmos

Esta seção descreve uma metodologia para a avaliação de algoritmos que é comumente utilizada em AM. Existem muitos outros testes estatísticos para medir se a diferença entre quaisquer dois algoritmos é significativa ou não, além do descrito nesta seção (Freedman, Pisani, & Purves 1998). Uma boa revisão sobre comparação de algoritmos pode ser encontrada em (Salzberg 1995; Dietterich 1997b).

4.6.1 Calculando Média e Desvio Padrão Utilizando Amostragem

Antes de comparar dois algoritmos, algumas definições adicionais são necessárias. Para tanto, assume-se o emprego de *cross-validation*, uma vez que é um método comumente utilizado pela comunidade de AM. Entretanto, qualquer outro método de amostragem (exceto resubstituição) pode ser utilizado no lugar de *cross-validation* para calcular a média e o desvio do padrão do erro de um algoritmo.

Dado um algoritmo A e um conjunto de exemplos T , assume-se que T seja dividido em r partições. Para cada partição i , é induzida a hipótese h_i e o erro denotado por $\text{err}(h_i)$, $i = 1, 2, \dots, r$, é calculado. Em seguida, a média, a variância e o desvio padrão para todas as partições são calculados utilizando-se as Equações (4.17), (4.18) e (4.19), respectivamente.

$$\text{mean}(A) = \frac{1}{r} \sum_{i=1}^r \text{err}(h_i) \quad (4.17)$$

$$\text{var}(A) = \frac{1}{r} \left[\frac{1}{r-1} \sum_{i=1}^r (\text{err}(h_i) - \text{mean}(A))^2 \right] \quad (4.18)$$

$$\text{sd}(A) = \sqrt{\text{var}(A)} \quad (4.19)$$

Assumindo $\text{mean}(A)$ como tendo uma distribuição normal (é importante lembrar que cada $\text{err}(h_i)$ é, por si só, uma média), observa-se que o termo $1/(r-1)$ na Equação (4.18) tem origem na definição do estimador não-viciado para variância, enquanto o termo $1/r$ é originário do Teorema do Limite Central para estimar a variância de médias (Moses 1986, Capítulo 4).

É possível denotar $\text{mean}(A)$ como $\text{mean}(A, T)$, quando a intenção é tornar evidente que o erro médio do algoritmo A foi calculado sobre o conjunto de exemplos T . Alternativamente, $\text{mean}(A)$ pode ser denotado por $\text{mean}(T)$, quando deseja-se evidenciar que o erro médio foi calculado utilizando o conjunto de exemplos T , assumindo o algoritmo A fixo para um dado experimento. Analogamente, essa notação se estende para $\text{var}(A)$, $\text{sd}(A)$ ou outros valores que possam ser derivados a partir destes (por exemplo, (4.22)).

4.6.2 Comparando Algoritmos

Comparando dois algoritmos observando apenas valores, por exemplo, a taxa de erro em problemas de classificação ou o erro em problemas de regressão, não é fácil perceber se um algoritmo é melhor do que o outro. Em várias situações, para comparar o erro (média e desvio padrão) obtido, *r-fold stratified cross-validation* é utilizada (para manter a distribuição de classes). De fato, a maioria dos trabalhos na área reportam erros utilizando 10-fold cross-validation ou stratified cross-validation.

Ao comparar dois indutores no mesmo domínio T , o desvio padrão pode ser visto como uma imagem da robustez do algoritmo: se os erros provenientes de hipóteses induzidas, calculados sobre diferentes conjuntos de teste e utilizando diferentes conjuntos de treinamento, são muito diferentes de um experimento para outro, então o indutor não é robusto a mudanças no conjunto de treinamento proveniente de uma mesma distribuição.

Por outro lado, suponha que deseja-se comparar dois algoritmos com taxas de erro iguais a 9 ± 1 e $7,50 \pm 0,80$. Para decidir qual deles é melhor (com grau de confiança de 95%) basta assumir o caso geral para determinar se a diferença entre dois algoritmos — A_S e A_P — é significante ou não, tendo-se uma distribuição normal (Weiss & Indurkhya 1998). Em geral, a comparação é feita de forma que A_P é o algoritmo proposto e A_S o algoritmo padrão. Para isso, a média e o desvio padrão combinados são calculados de acordo com as Equações (4.20) e (4.21), respectivamente. A diferença absoluta em desvios padrões, é calculada pela Equação (4.22).

$$\text{mean}(A_S - A_P) = \text{mean}(A_S) - \text{mean}(A_P) \quad (4.20)$$

$$\text{sd}(A_S - A_P) = \sqrt{\frac{\text{sd}(A_S)^2 + \text{sd}(A_P)^2}{2}} \quad (4.21)$$

$$\text{ad}(A_S - A_P) = \frac{\text{mean}(A_S - A_P)}{\text{sd}(A_S - A_P)} \quad (4.22)$$

Se $\text{ad}(A_S - A_P) > 0$ então A_P supera A_S . Porém, se $\text{ad}(A_S - A_P) \geq 2$ desvios padrões então A_P supera A_S com grau de confiança de 95%. Por outro lado, se $\text{ad}(A_S - A_P) \leq 0$ então A_S supera A_P e se $\text{ad}(A_S - A_P) \leq -2$ então A_S supera A_P com grau de confiança de 95%.