CT 720 Tópicos em Aprendizagem de Máquina e Classificação de Padrões

9-Aprendizagem de Máquina

Conteúdo

1. Introdução
2. Superioridade de classificadores
3. Polarização e variância
4. Reamostragem para estimar estatísticas
5. Reamostragem em projeto de classificadores
6. Estimando e comparando classificadores
7. Resumo

1-Introdução

- Algoritmos de aprendizagem
  - qual é o melhor algoritmo ?
  - mais simples ?
  - contínuos e suaves?
  - quais são os princípios fundamentais ?
- Este capítulo
  - mostra que nenhum classificador é inerentemente superior a outro
  - mecanismos para comparar algoritmos

2-Superioridade de classificadores

- Questões
  - sob o ponto de vista da capacidade de generalização existem razões para preferir um classificador do que outros ?
  - algum método de classificação é superior a todos os outros, independentemente do problema de classificação ?
  - é possível selecionar um algoritmo que completamente superior (ou inferior) a classificação feita aleatoriamente ?

- *No free lunch theorem*
  - respostas para as questões: *não*
  - se objetivo é obter um algoritmo com boa capacidade de generalização, independentemente do contexto da aplicação, não existem razões para favorecer um classificador sobre outros

- O que o teorema nos sugere
  - quando projetamos um classificador devemos concentrar em
  - informação *a priori*
  - distribuição dos dados
  - quantidade de dados de treinamento
  - função de custo/benefício
  - ceticismo quanto a estudos comparativos de algoritmos

- Comparação de algoritmos
  - *off-training set error*
  - erro para pontos que não estão no conjunto de treinamento

- Considerar o seguinte problema de classificação
  - duas categorias
  - padrões $\mathbf{x}^i$ associados com classes $y_i = \pm 1$, $i = 1,...,n$
  - $y_i = F(\mathbf{x}^i)$
  - F não é conhecida
  - atributos discretos

- Notação

$\mathcal{H}$ conjunto (discreto) de hipóteses

$h(\mathbf{x})$ hipótese particular de $\mathcal{H}$

$P(h)$ probabilidade que o algoritmo produza hipótese h
(não é a probabilidade que h é correta !)

$P(h|\mathcal{D})$ probabilidade que algoritmo produza hipótese h a partir de $\mathcal{D}$

$\mathcal{E}$ erro

– como avaliar a qualidade da capacidade de generalização de um algoritmo de aprendizagem se F é desconhecida ?

– valor esperado do erro, dado $\mathcal{D}$, somado para todas funções !

$$E[\mathcal{E} \mid \mathcal{D}] = \sum_{h,\, f} \sum_{x \notin \mathcal{D}} P(\mathbf{x})[1 - \delta(F(\mathbf{x}), h(\mathbf{x}))] P(h \mid \mathcal{D}) P(F \mid \mathcal{D})$$

δ = delta Kronecker

– erro de classificação quando função verdadeira é $F(\mathbf{x})$ e a probabilidade para o k-ésimo algoritmo de aprendizagem é $P_k(h(\mathbf{x})|\mathcal{D})$

$$E_k[\mathcal{E} \mid F, n] = \sum_{x \notin \mathcal{D}} P(\mathbf{x})[1 - \delta(F(\mathbf{x}), h(\mathbf{x}))] P_k(h(\mathbf{x}) \mid \mathcal{D})$$

- *No free lunch theorem*

For any two learning algorithms $P_1(h|\mathcal{D})$ *and* $P_2(h|\mathcal{D})$*, the following are true independent of the sampling distribution* $P(\mathbf{x})$ *and the number n of training points*:

1-*Uniformly averaged over all target functions,* $E_1(\mathcal{E}|F,n) - E_2(\mathcal{E}|F,n) = 0$

2-*For any fixed training set* $\mathcal{D}$*, uniformly averaged over F,*
$E_1(\mathcal{E}|F,\mathcal{D}) - E_2(\mathcal{E}|F,\mathcal{D}) = 0$

3-*Uniformly averaged over all priors* $P(F)$, $E_1(\mathcal{E}|n) - E_2(\mathcal{E}|n) = 0$

4-*For any fixed training set* $\mathcal{D}$*, uniformly averaged over* $P(F)$
$E_1(\mathcal{E}|\mathcal{D}) - E_2(\mathcal{E}|\mathcal{D}) = 0$

1- mostra que *off-training error* para todos algoritmos de aprendizagem é o mesmo, isto é:

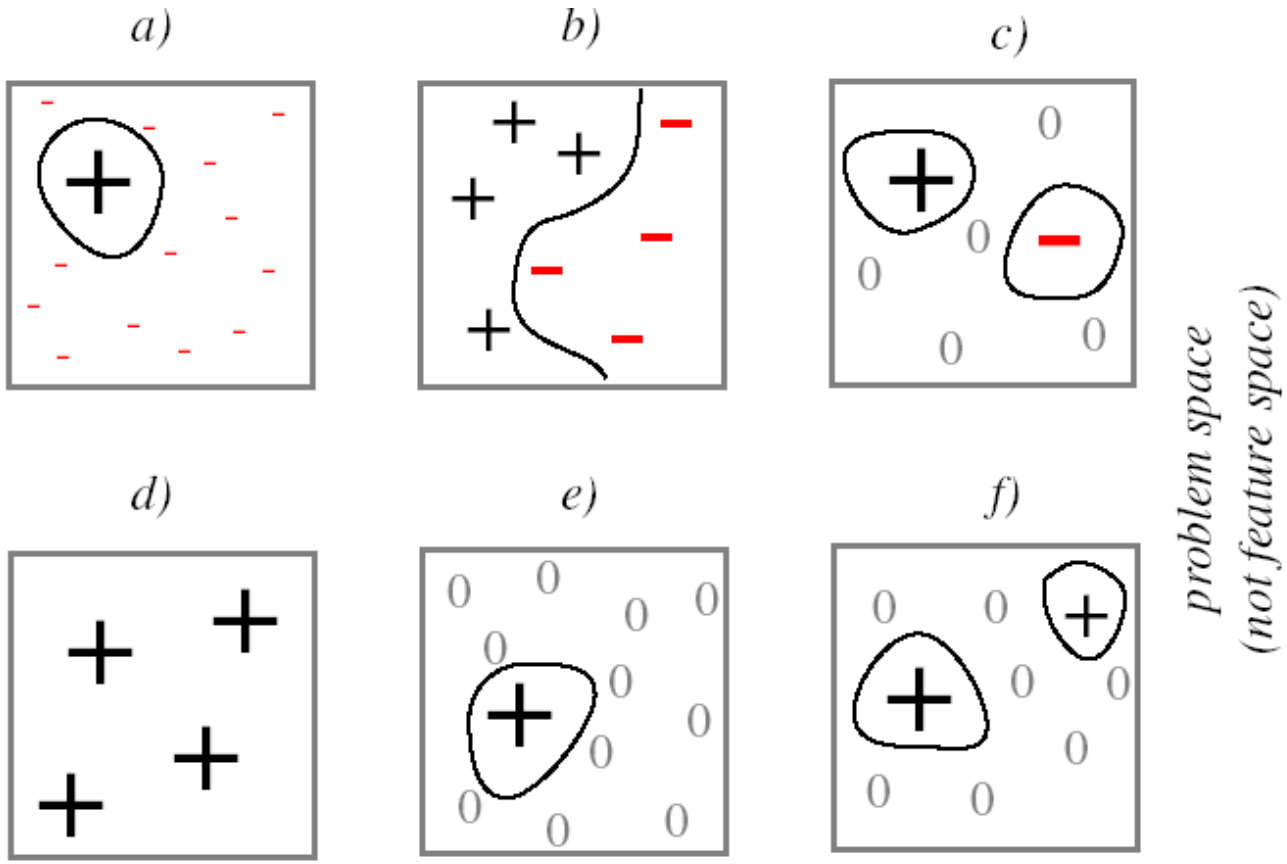
$$\sum_{F}\sum_{\mathcal{D}} P(\mathcal{D}\,|\,\mathcal{F})[E_1(\mathcal{E}\,|\,F,n) - E_2(\mathcal{E}\,|\,F,n)] = 0$$

2- mostra que, mesmo quando conhecemos $\mathcal{D}$, então *averaged over all target functions*, nenhum algoritmo de aprendizagem tem *off-training error* superior a outro, isto é:

$$\sum_{F}[E_1(\mathcal{E}\,|\,F,\mathcal{D}) - E_2(\mathcal{E}\,|\,F,\mathcal{D})] = 0$$

possible learning systems

impossible learning systems



problem space (not feature space)

- Exemplo

	x	F	h_1	h_2
$\mathcal{D}$	000	1	1	1
	001	-1	-1	-1
	010	1	1	1
	011	-1	1	-1
	100	1	1	-1
	101	-1	1	-1
	110	1	1	-1
	111	1	1	-1

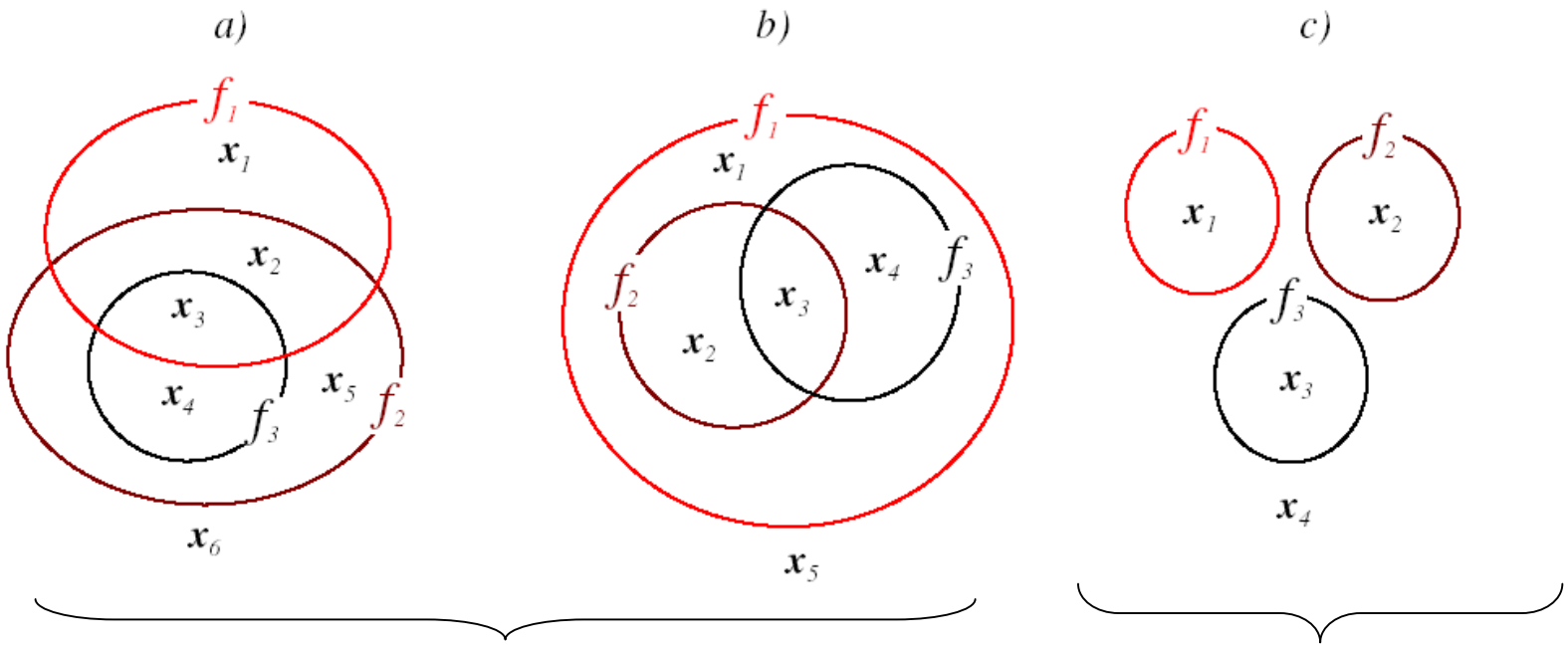
$$E_1(\mathcal{E}\,|F,\mathcal{D}) = 0.4$$

$$E_2(\mathcal{E}\,|F,\mathcal{D}) = 0.6$$

- *Ugly duckling theorem*
  - trata de características e padrões
  - na ausência de hipóteses, não existe a melhor representação de características/atributos e mesmo a noção de similaridade entre padrões depende implicitamente de hipóteses que podem ou não estarem corretas !
  - em representações discretas, podemos usar expressões lógicas (predicados) para descrever um padrão

    atributo binário: f_i

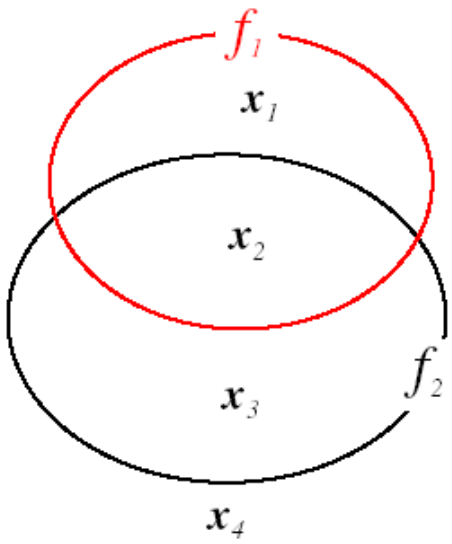
    descrição do padrão: f_1 *AND* f_2, *NOT* f_2, etc...



f_1 has_legs
f_2 has_righ_arm
f_3 has_righ_hand

f_1 brown_eyes
f_2 green_eyes
f_3 blue_eyes

– rank *r* de um predicado: número de elementos mais simples (indivisíveis) que o predicado contém



rank $r = 1$

$\mathbf{x}_1$	f_1 *AND NOT* f_2
$\mathbf{x}_2$	f_1 *AND* f_2
$\mathbf{x}_3$	f_2 *AND NOT* f_1
$\mathbf{x}_4$	*NOT* (f_1 *OR* f_2)

rank $r = 2$

$\mathbf{x}_1$ *OR* $\mathbf{x}_2$	f_1
$\mathbf{x}_1$ *OR* $\mathbf{x}_3$	f_1 *XOR* f_2
$\mathbf{x}_1$ *OR* $\mathbf{x}_4$	*NOT* f_2
$\mathbf{x}_2$ *OR* $\mathbf{x}_3$	f_2
$\mathbf{x}_2$ *OR* $\mathbf{x}_4$	*NOT* (f_1 *AND* f_2)
$\mathbf{x}_3$ *OR* $\mathbf{x}_4$	*NOT* f_1

nº total predicados = 2^n
(sem restrições)

rank $r = 3$

$\mathbf{x}_1$ *OR* $\mathbf{x}_2$ *OR* $\mathbf{x}_3$	f_1 *OR* f_2
$\mathbf{x}_1$ *OR* $\mathbf{x}_2$ *OR* $\mathbf{x}_4$	f_1 *OR NOT* f_2
$\mathbf{x}_1$ *OR* $\mathbf{x}_3$ *OR* $\mathbf{x}_4$	*NOT* (f_1 *AND* f_2)
$\mathbf{x}_2$ *OR* $\mathbf{x}_3$ *OR* $\mathbf{x}_4$	f_2 *OR NOT* f_1

– Questão: na ausência de informação a priori, existe algum princípio ou razão para julgar dois padrões distintos quaisquer mais ou menos similar que dois outros padrões distintos ?

Exemplo

f_1 `blind_righ_eye` $\mathbf{x}_1 = \{1,0\}$ (*blind only right eye*)
f_2 `blind_left_eye` $\mathbf{x}_2 = \{0,1\}$ (*blind only left eye*)

$\mathbf{x}_1$ maximamente diferente de $\mathbf{x}_2$
$\mathbf{x}_1$ mais similar a $\mathbf{x}_c = \{1,1\}$ e a $\mathbf{x}_p = \{0,0\}$

alternativamente: $\mathbf{x}_1$ e $\mathbf{x}_2$ poderiam ser consideradas similares
(e.g. ambas poderiam dirigir automóvel)

– existem maneiras distintas de representar vetores de atributos

Exemplo

f'_1 *blind_right_eye*

f'_2 *same_both_eyes*

as quatro pessoas poderiam ser representadas como:

	f_1	f_2	f'_1	f'_2
$\mathbf{x}_1$	0	0	0	0
$\mathbf{x}_2$	0	1	0	1
$\mathbf{x}_3$	1	0	1	0
$\mathbf{x}_4$	1	1	1	1

- Medida de similaridade entre dois padrões

  – número de predicados ao invés de número de atributos

$$\sum_{r-2}^{d} \binom{d-2}{r-2} = 2^{d-2} \qquad (6)$$

  – esta expressão é independente das escolhas de $\mathbf{x}_i$ e $\mathbf{x}_j$

  – número de predicados é constante e independente dos padrões !

- *Ugly duckling theorem*

Given that we use a finite set of predicates that enables us to distinguish any two patterns under consideration, the number of predicates shared by any two such patterns is constant and independent f the choice of those patterns. Furthermore, if pattern similarity is based on the total number of predicates shared by two patterns, then any two patterns are "equally similar."

– não existe o melhor conjunto de atributos independente do problema
– noção de similaridade é baseada em hipóteses implícitas sobre o problema
– teorema vale também para atributos contínuos

- Minimum description length (MDL)

  – classificadores mais “simples” são preferíveis aos “complexos”

  – complexidade de Kolmogorov: complexidade *string* binário

  – em comunicação:

    mensagem x transmissor

    mensagem y receptor

    x transmitida e decodificada por um método fixo L

    $y = L(x)$

    custo transmissão de x: tamanho mensagem transmitida $|\, y \,|$

    custo mínimo $\Rightarrow$ mensagem tamanho mínimo

$$L_{\min} = \min_{y:L(y)=x} |\, y \,|$$

- Complexidade algorítmica

  – definida por analogia à comunicação
  – programas executados em máquinas abstratas ao invés de método *L*
  – descrição da mensagem: universal, independente da implementação
  – complexidade de Kolmogorov de um string binário *x*, K(*x*)
  tamanho (em bits) do menor string binário *y* que define o programa que decodifica *x* quando executado por uma máquina abstrata e pára.

$$K(x) = \min_{y:U(y)=x} |y|$$

U máquina de Turing

- Princípio MDL

  – supor que queremos projetar classificador utilizando D

  – MDL: devemos minimizar a soma da complexidade algorítmica do modelo e a descrição dos dados de treinamento com relação ao modelo

$$K(h, \mathcal{D}) = K(h) + K(\mathcal{D} \text{ using } h)$$

$$h^* = \arg\min_h K(h, \mathcal{D})$$

  – classificadores projetados de acordo com MLD convergem para o ideal (modelo verdadeiro) no limite de mais e mais dados

  – não é necessariamente superior para conjunto finito de dados (violaria *no free lunch theorem*)

– Exemplo: classificador de Bayes (hipóteses e dados discretos)

$$P(h \mid \mathcal{D}) = \frac{P(h)P(\mathcal{D}|h)}{P(\mathcal{D})}$$

hipótese ótima h^* : aquela com a maior probabilidade *a posteriori*

$$h^* = \arg\max_h [P(h)P(\mathcal{D} \mid h] =$$
$$= \arg\max_h [\log_2 P(h) + \log_2 P(\mathcal{D} \mid h)$$

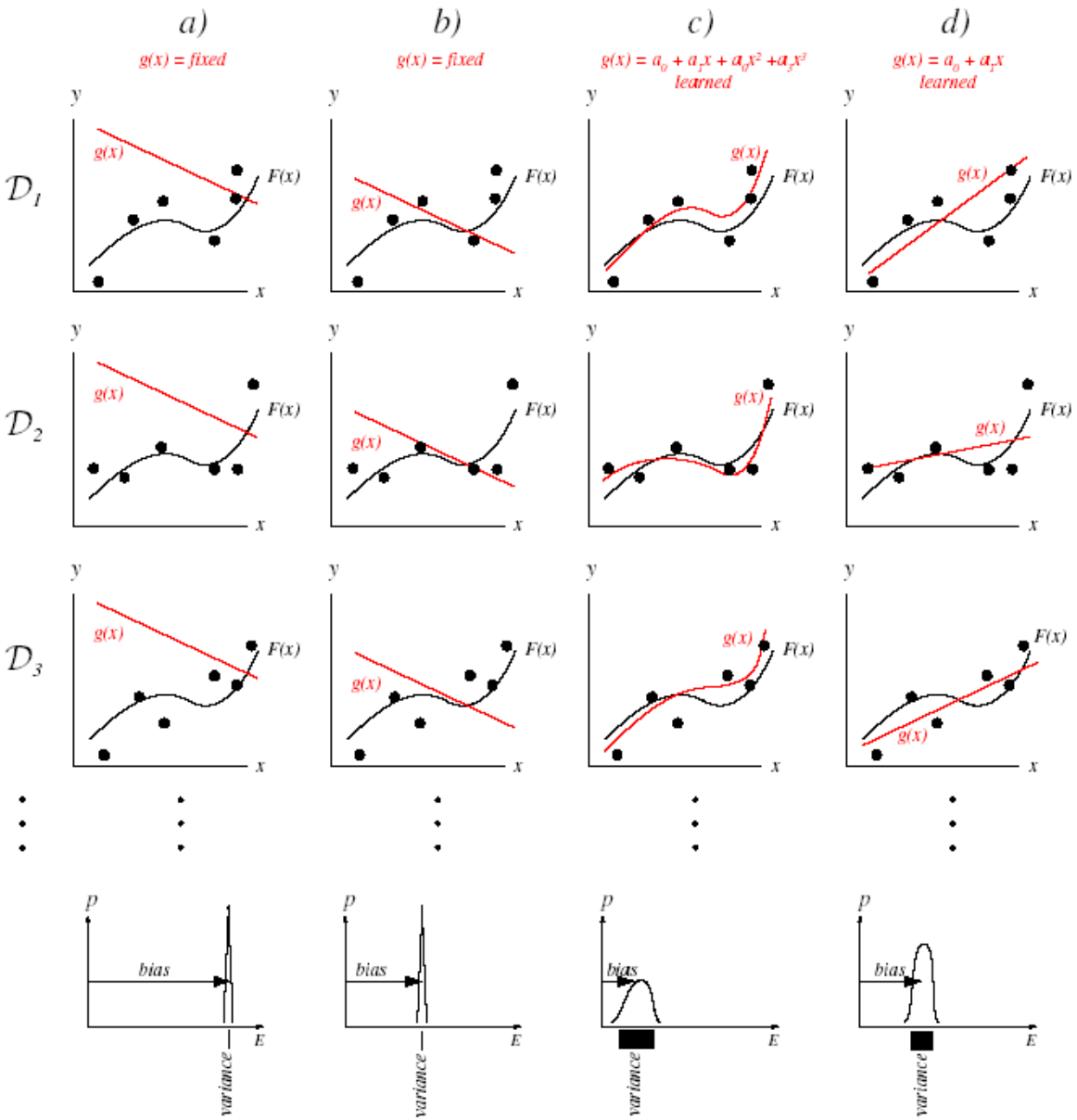
3-Bias e variância

– mais fácil de entender em regressão

função $F(\mathbf{x})$ (desconhecida) + ruído
deseja-se estimar $F(.)$ a partir de $\mathcal{D}$ gerado por F
função de regressão $g(\mathbf{x};\mathcal{D})$
objetivo: analisar como a aproximação $g(\mathbf{x};\mathcal{D})$ depende de $\mathcal{D}$

$$E_{\mathcal{D}}[(g(x;\mathcal{D})-F(x))^2]=$$

$$=(\underbrace{E_{\mathcal{D}}[g(x;\mathcal{D})-F(x)]^2}_{\text{bias}}+\underbrace{E_{\mathcal{D}}[(g(x;\mathcal{D})-E_{\mathcal{D}}[g(x;\mathcal{D})])^2]}_{\text{variância}}$$



- Resumindo

  – se o modelo tem muitos parâmetros (geralmente bias pequena) então ele se ajusta bem aos dados, mas com variância grande

  – se o modelo tem poucos parâmetros (geralmente bias grande) então ele pode não se ajustar bem aos dados, e este ajuste não se modifica significativamente para conjuntos distintos de dados (variância pequena)

- Bias e variância em classificação
  - não linear
  - discriminante de Bayes, $p(g(\mathbf{x};\mathcal{D})$ Gaussiano

$$\Pr[g(\mathbf{x};\mathcal{D}) \neq y_B] = \Phi\left[Sgn[F(\mathbf{x}) - 1/2] \frac{E_{\mathcal{D}}[g(\mathbf{x};\mathcal{D})] - 1/2}{\sqrt{Var[g(\mathbf{x};\mathcal{D})]}} \right] =$$

$$= \Phi\Big[\underbrace{Sgn[F(\mathbf{x}) - 1/2][E_{\mathcal{D}}[g(\mathbf{x};\mathcal{D}) - 1/2]}_{\text{boundary bias}} \underbrace{Var[g(\mathbf{x};\mathcal{D})]^{-1/2}}_{\text{variance}} \Big]$$

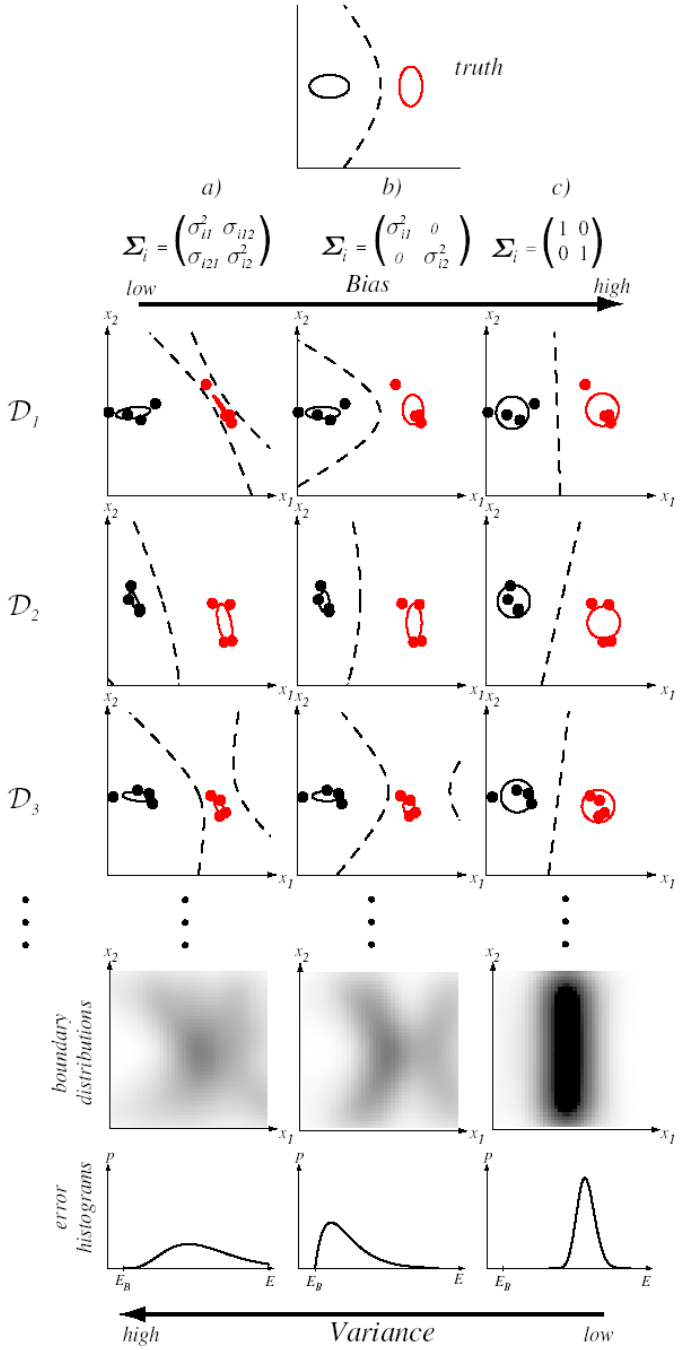
$$\Phi[t] = \frac{1}{\sqrt{2\pi}} \int_t^{\infty} e^{-1/2u^2 du} = \frac{1}{2}[1 - erf(t/\sqrt{2})]$$

- Resumindo

  – valor pequeno da variância geralmente é importante para uma boa precisão de classificação

  – geralmente, variância domina bias: em termos práticos, isso implica que não precisamos ficar particularmente preocupados se nossa estimativa é polarizada, desde que a variância seja pequena

  – aumentar a flexibilidade (e.g. número de parâmetros livres) de adaptação aos dados dos algoritmos de classificação tende a produzir bias pequenas e variâncias grandes.



$p(\mathbf{x}|\omega_i) \sim N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$
estimativa MV

4-Reamostragem para estimar estatísticas

- Questão

– como estimar bias e variância quando utilizamos um algoritmo de aprendizagem para resolver um novo problema de reconhecimento de padrões ?

– múltiplas amostras → reamostragem

– métodos: *jackknife* e *boosting*

- *Jackknife*

$\mathcal{D} = \{x_i, i = 1,...,n\}$ amostras de uma distribuição unidimensional

– estimativa da média

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

– estimativa da variância

$$\hat{\mu} = \frac{(n-1)}{n}\sum_{i=1}^{n}(x_i - \hat{\mu})^2$$

– como estimar a mediana, a moda, ou outra estatística qualquer?

– *leave-one-out mean*

$$\mu_{(i)} = \frac{1}{n-1}\sum_{j\neq i}^{n} x_j = \frac{n\bar{x} - x_i}{n-1} \qquad (24)$$

– estimativa *jackknife* da média

$$\mu_{(.)} = \frac{1}{n}\sum_{i=1}^{n}\mu_{(i)} \qquad (25)$$

– estimativa *jackknife* da variância da média

$$Var[\mu_{(.)}] = \frac{n-1}{n}\sum_{i=1}^{n}(\mu_{(i)} - \mu_{(.)})^2 \qquad (26)$$

– estimador geral (mediana, moda, etc.)

$$\hat{\theta}_{(i)} = \hat{\theta}(x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$$

– estimativa *jackknife* do parâmetro

$$\hat{\theta}_{jack} = \frac{1}{n}\sum_{i=1}^{n}\hat{\theta}_{(i)}$$

– estimativa *jackknife* da variância do parâmetro

$$Var_{jack}[\hat{\theta}_{jack}] = \frac{n-1}{n}\sum_{i=1}^{n}(\hat{\theta}_{(i)} - \hat{\theta}_{(.)})^2 \qquad (32)$$

– estimativa *jackknife* da bias

$$bias = \theta - E[\hat{\theta}]$$

$$bias_{jack} = (n-1)(\hat{\theta}_{(.)} - \hat{\theta})$$

$$\tilde{\theta} = \hat{\theta} - bias_{jack} = n\hat{\theta} - (n-1)\hat{\theta}_{(.)} \qquad (30)$$

estimativa não polarizada da bias

- Exemplo: estimativa da moda

$\mathcal{D} = \{0, 10, 10, 10, 20, 20\}$, $n = 6$

$$\hat{\theta}_{jack} = \hat{\theta}_{(.)} = \frac{1}{n}\sum_{i=1}^{n}\hat{\theta}_{(i)} = \frac{1}{6}[10+15+15+15+10+10] = 12.5 \quad \hat{\theta}_{(.)} > \hat{\theta}$$

$$bias_{jack} = (n-1)(\hat{\theta}_{(.)} - \hat{\theta}) = 5(12.5-10) = 12.5$$

$$Var_{jack}[\hat{\theta}_{jack}] = \frac{n-1}{n}\sum_{i=1}^{n}(\hat{\theta}_{(i)} - \hat{\theta}_{(.)})^2 =$$

$$= \frac{5}{6}[(10-12.5)^2 + 3(15-12.5)^2 + 2(10-12.5)^2]$$

$$= 31.25$$

- *Bootstrap*

– conjunto de dados é criado selecionando n pontos aleatoriamente de $\mathcal{D} = \{x_i,.., x_n\}$, com reposição

– seleção dos n pontos é repetida B vezes independentemente

– B dados *bootstrap* são tratados como conjuntos independentes

– estimativa *boostrap* de uma estatística θ é a média das B estimativas dos conjuntos *boostrap* individuais

– estimativa *bootstrap* do parâmetro

$$\hat{\theta}^{*(.)} = \frac{1}{B}\sum_{b=1}^{B}\hat{\theta}^{*(b)} \qquad (33)$$

– estimativa *bootstrap* da bias

$$bias_{boot} = \frac{1}{B}\sum_{b=1}^{B}\hat{\theta}^{*(b)} - \hat{\theta} = \hat{\theta}^{*(.)} - \hat{\theta} \qquad (34)$$

– estimativa *bootstrap* da variância do parâmetro

$$Var_{boot}[\theta] = \frac{1}{B}\sum_{i=1}^{n}\left[\hat{\theta}^{*(b)} - \hat{\theta}^{*(.)}\right]^2 \qquad (35)$$

5-Reamostragem em projeto de classificadores

- Idéia
  - utilizar técnicas de reamostragem para treinamento de classificadores
  - reamostragem pode ser usada com vários métodos de treinamento
  - retreinamento também está relacionado com comparação de classificadores

- *Bagging*

– *arcing*: *adaptive reweighting and combining*

– *arcing*: reuso/seleção de dados para melhorar classificação

– *boosting*: *bootstrap aggregation*

  - múltiplas versões de dados de treinamento
  - cada versão con $n' < n$ amostras de $\mathcal{D}$, com reposição
  - cada conjunto de dados é usado para treinar um classificador
  - decisão final classificação: voto de cada classificador\
  - classificador: tem a mesma forma (e.g. redes neurais, etc.)

• *bagging*: é uma forma de multiclassificador

- *Boosting*

– objetivo: melhorar precisão de qualquer algoritmo de aprendizagem

– primeiro cria classificador com precisão maior que a média e acrescenta novos classificadores para forma um conjunto cuja regra de decisão conjunta possui maior precisão para dados de treinamento

– visão geral
  - treina uma sucessão de classificadores com um subconjunto de $\mathcal{D}$
  - treinamento considera dados de treinamento "mais informativo"

- Exemplo

– três classificadores, duas classes

  - selecionar $n_1 < n$ padrões de $\mathcal{D}$ (sem reposição): $\mathcal{D}_1$
  - treinar classificador C_1 com $\mathcal{D}_1$
  - treinar classificador C_2 com $\mathcal{D}_2$ mais informativo, dado C_1
    - 50% padrões de $\mathcal{D}_2$ são corretamente classificados por C_1
    - 50% padrões de $\mathcal{D}_2$ são incorretamente classificados por C_1

– construção de $\mathcal{D}_2$

- jogar moeda
- se cara: selecionamos padrões restantes de $\mathcal{D}$ e apresentamos a C_1, um a um, até que C_1 classifica incorretamente o padrão; acrescentamos este padrão a $\mathcal{D}_2$ 2.
- se coroa: encontramos um padrão que C_1 classifica corretamente
- continuar até que nenhum padrão possa ser acrescentado

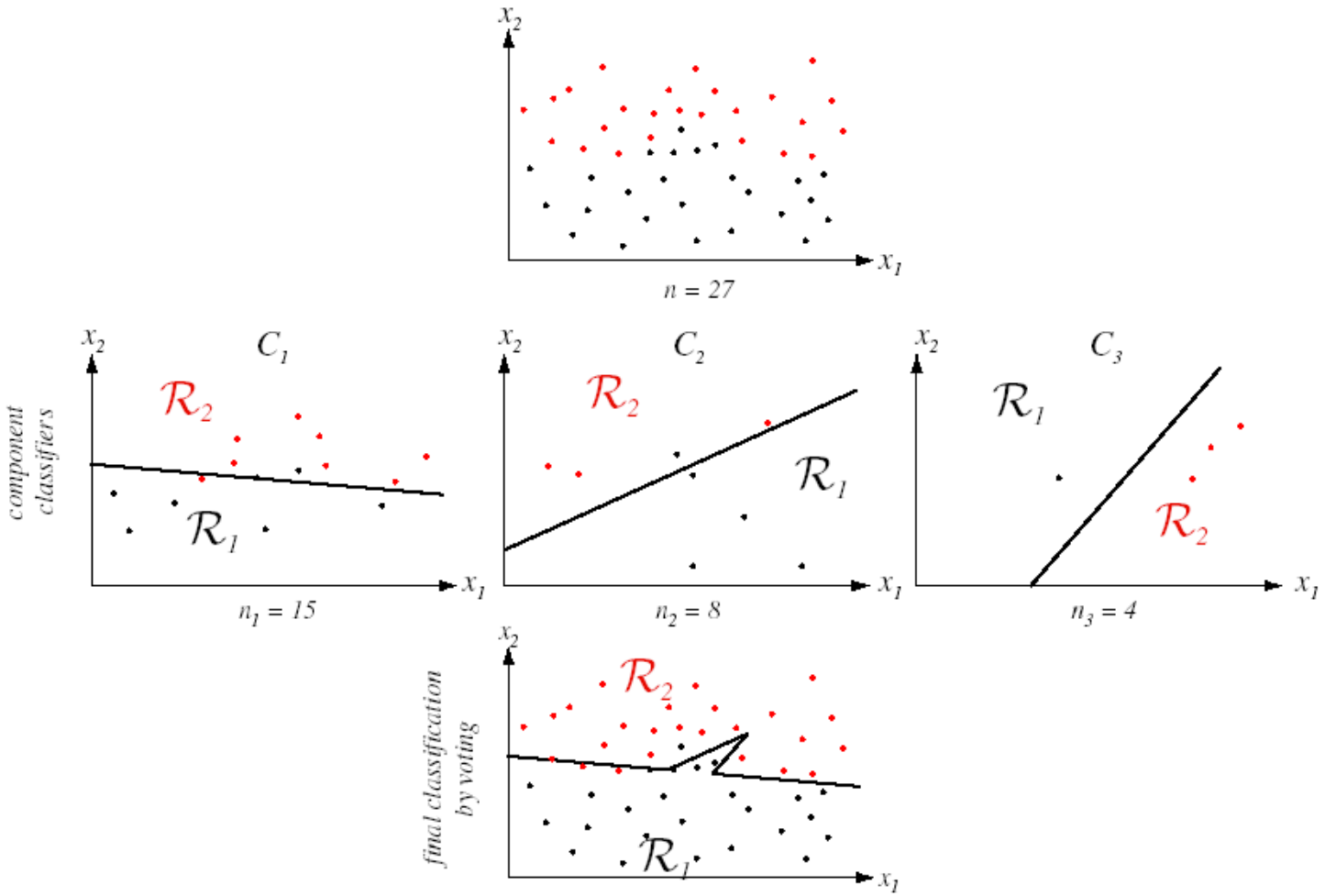
– construção de $\mathcal{D}_3$

- dados que não são corretamente classificadores por C_1 e C_2
- selecionar padrão aleatoriamente de $\mathcal{D}$ e classificar com C_1 e C_2
- se C_1 e C_2 discordam, acrescentar este padrão à $\mathcal{D}_3$
- caso contrário ignorar o padrão

– treinar C_3 com $\mathcal{D}_3$

– classificação de de um padrão **x**

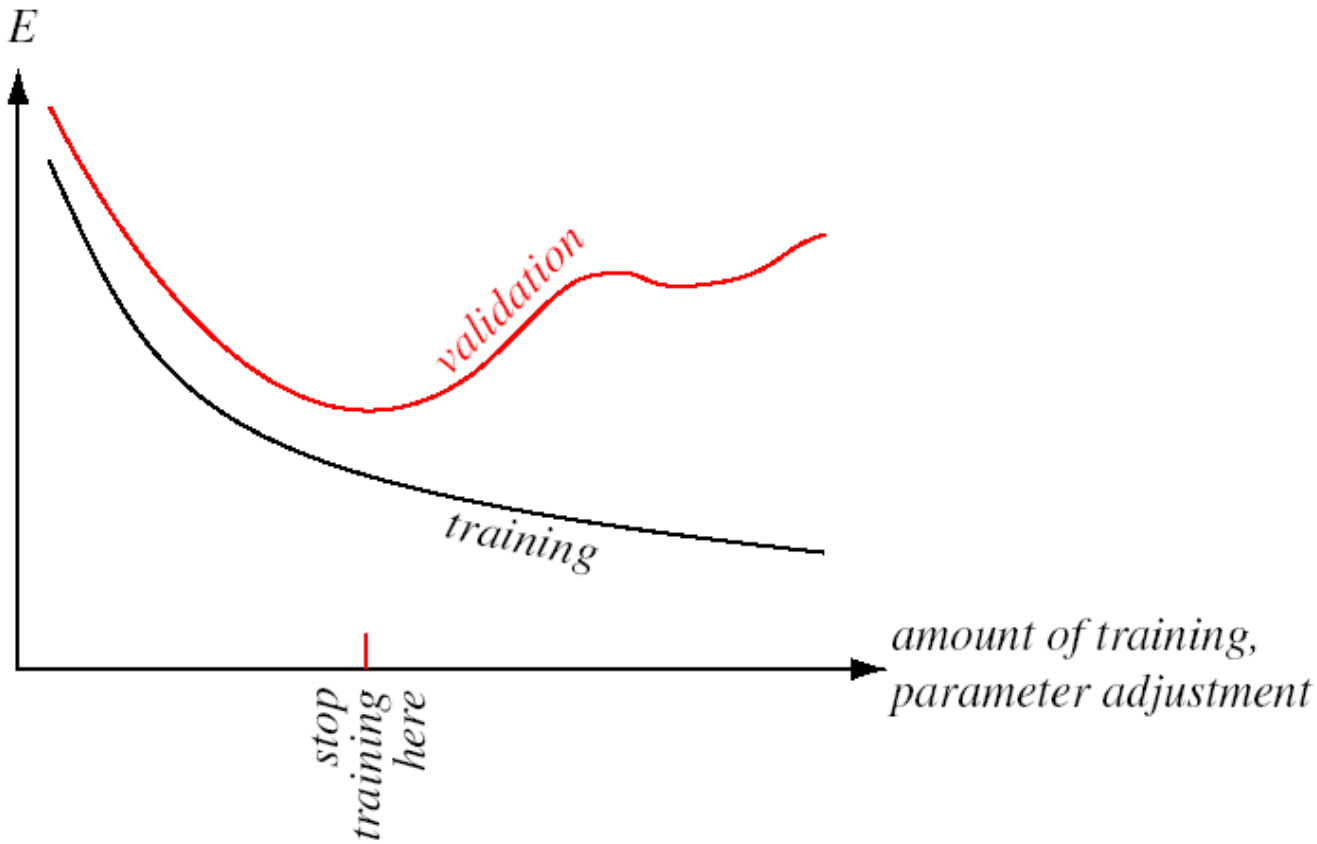
- se C_1 e C_2 concordam, aceita-se esta classe como correta
- se C_1 e C_2 discordam, então adota-se a classe estabelecida por C_3



6-Estimando e comparando classificadores

- Motivação
  - verificar se classificador tem desempenho satisfatório capacidade de generalização
  - comparar classificadores
  - métodos são heurísticos (*no free lunch theorem*)

- Validação cruzada
  - dividir $\mathcal{D}$ em duas partes
    - conjunto treinamento (e.g. ajuste parâmetros)
    - conjunto validação (estimar generalização)
  - treinar classificador até atingir erro validação mínimo
  - conjunto validação não pode ter padrões do conjunto treinamento
  - generalização: *m-fold cross-validation*
    - m conjuntos disjuntos de mesma cardinalidade n/m ($n = |\mathcal{D}|$)
    - classificador treinado m vezes
    - desempenho estimado: média dos m erros de classificação



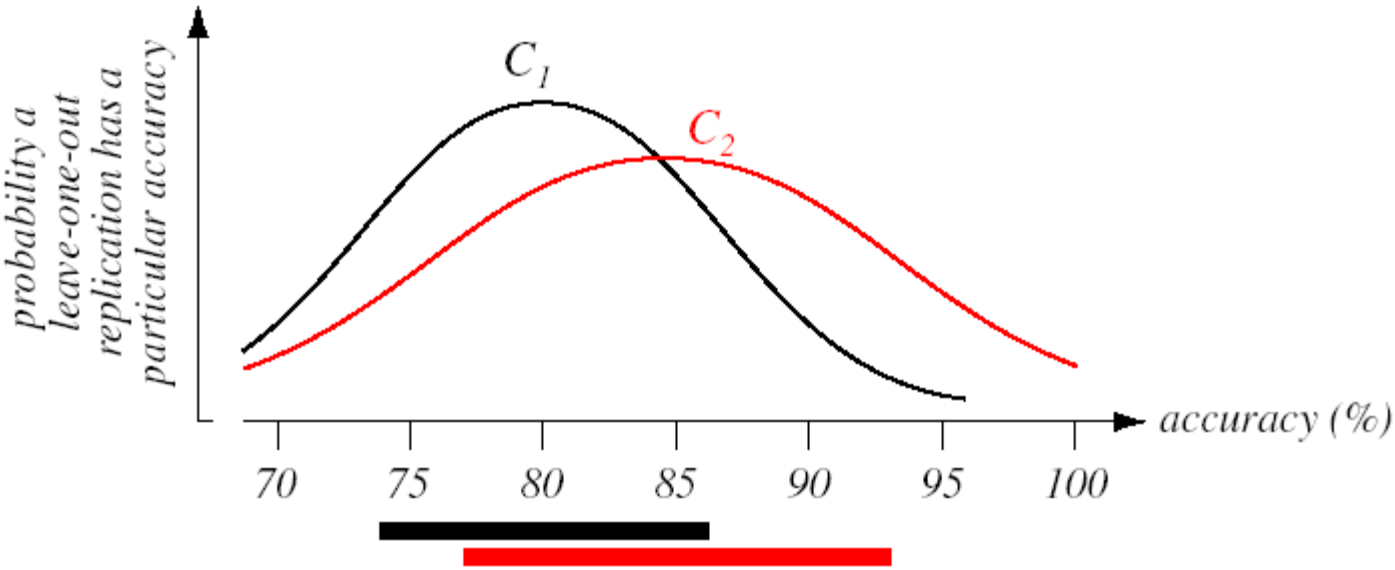
- *Jackknife* e *bootstrap*

  – jackknife
    - conjunto treinamento: $\mathcal{D}$ deletando um de seus padrões
    - treinar classificador n vezes
    - desenpenho: média destes n treinamentos

  – fornece boas estimativas, mas computacionalmente caro

  – estima variância usando generalização da equação (32)

  – exemplo: estimador C_1 com 80% precisão e C_2 (*jackknife*) 85%
    - qual deles é o melhor ?
    - determinarmos as variâncias da precisão de classificação
    - usamos teste de hipótese



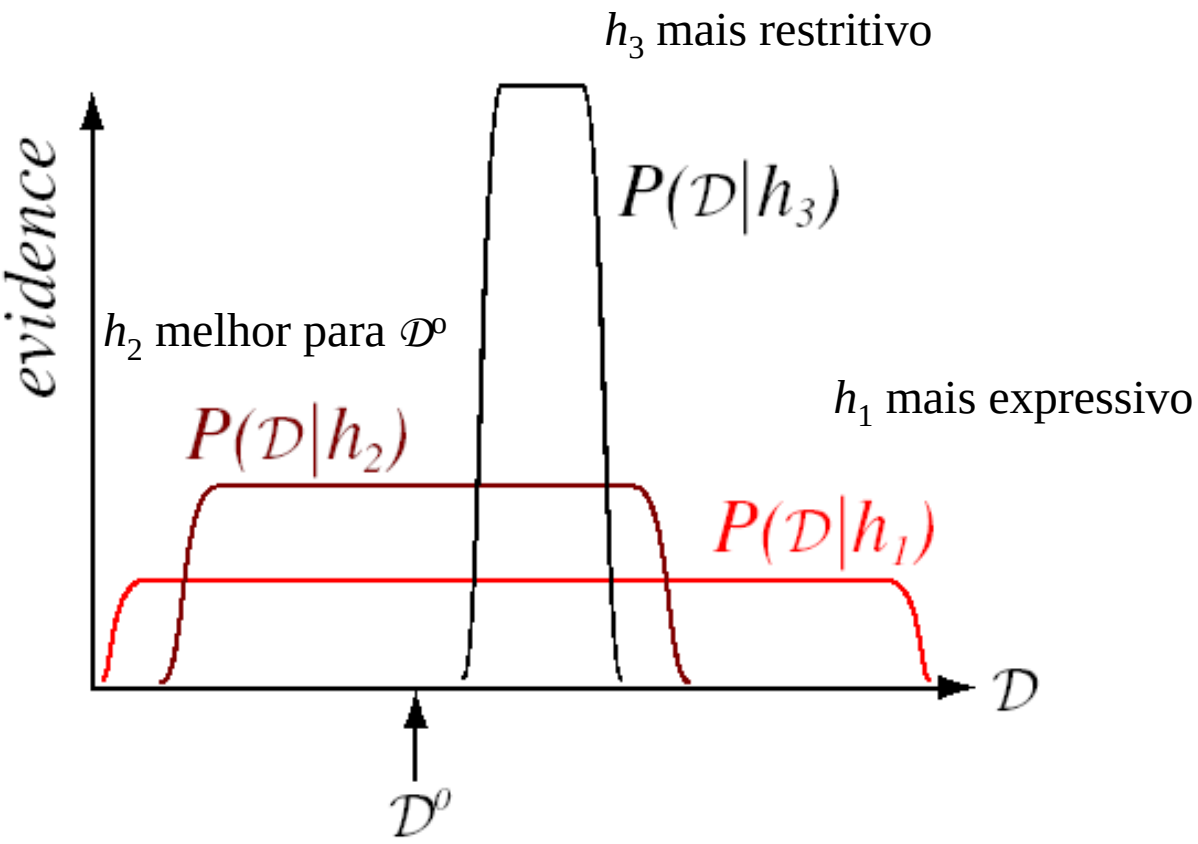
- Comparação por máxima verosimilhança
  - objetivo
    - selecionar o modelo que melhor caracteriza os dados
    - utiliza técnica apresentada no capítulo 3
    - determinar $\hat{\theta} = \arg\max_{\theta} p(\mathcal{D} \mid \theta)$
  - $h \in \mathcal{H}$, hipótese do modelo e $\mathcal{D}$ dados de treinamento
  - regra de Bayes para densidade *a posteriori* para qualquer modelo

$$P(h_i \mid \mathcal{D}) = \frac{P(\mathcal{D} \mid h_i)P(h_i)}{P(\mathcal{D})} \propto P(\mathcal{D} \mid h_i)P(h_i)$$

– considera-se evidência $P(\mathcal{D}|h_i)$ somente

– comparação por máxima verosimilhança

- determinar parâmetros de máxima verosimilhança para cada modelo
- calcular as respectivas verosimilhanças
- selecionar modelo com maio verosimilhança

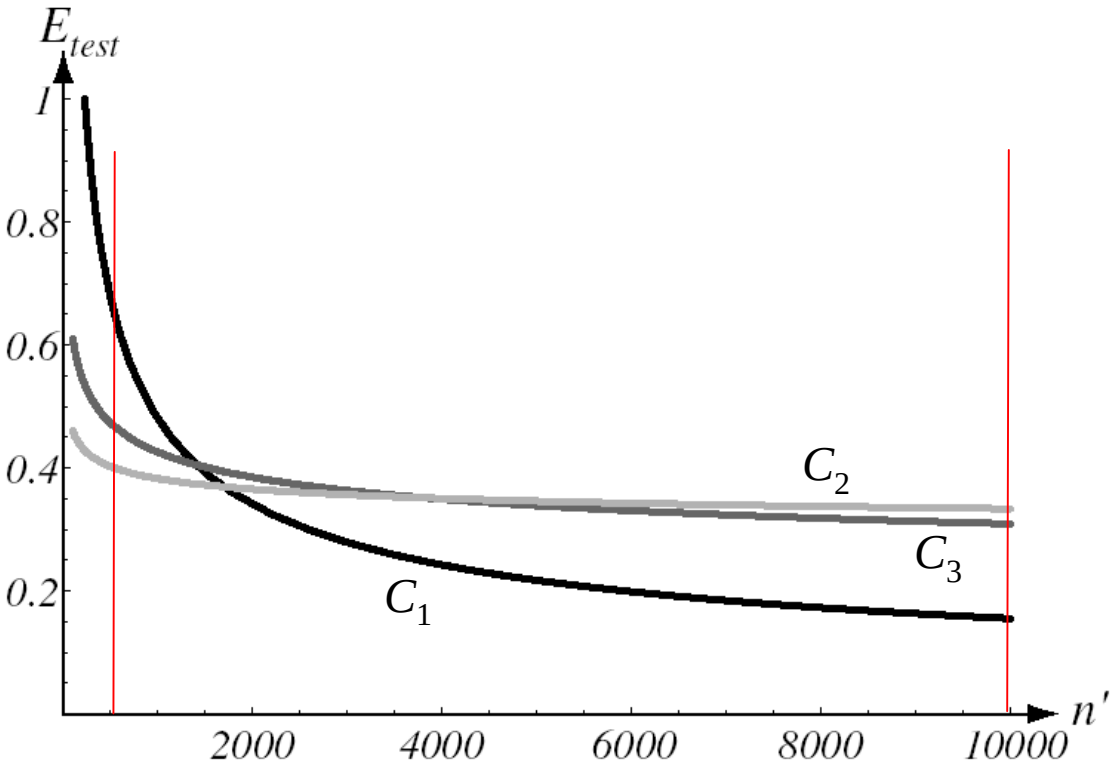


- Estimando desempenho com curvas de aprendizagem

  – objetivo
    - conjunto de dados muito grande requer treinamento intensivo
    - alto custo (computacional + esforço)
    - comparar classificadores sem treina-los com todos os dados
    - determinar o mais promissor e treina-lo plenamente

  – idéia: treinar com conjunto relativamente pequeno de dados
    prever desempenho com conjunto maior de dados

  – desempenho: caracterizado por uma curva de aprendizagem



– curva de aprendizagem (teste)

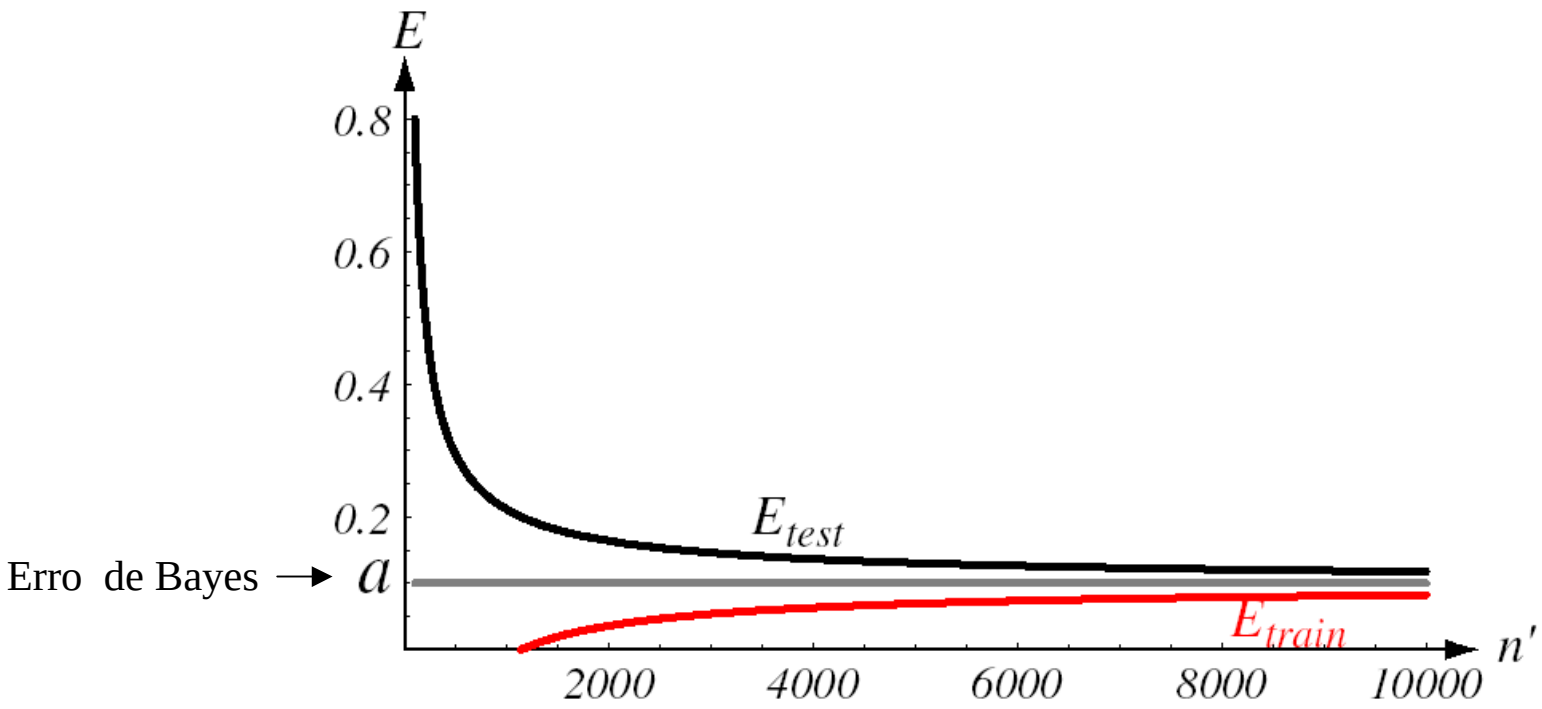
- decresce monotônicamente com $n' \leq n$
- não apresenta *overtraining* (como na curva validação cruzada)
- na prática podem ser aproximadas por funções da forma:

$$E_{test} = a + \frac{b}{n'^{\alpha}} \quad a, b, \alpha \geq 1 \qquad (49)$$

– curva de aprendizagem (treinamento)

- decresce monotônicamente com $n' \leq n$
- mesmo comportamento assintótico que para caso teste
- aproximadas por funções da forma:

$$E_{train} = a - \frac{c}{n'^{\beta}} \qquad a,\ c,\ \beta \geq 1 \qquad (50)$$


E
0.8
0.6
0.4
0.2
a
Erro de Bayes
E_{test}
E_{train}
n'
2000
4000
6000
8000
10000

– estimação dos parâmetros a, b, c, α, β

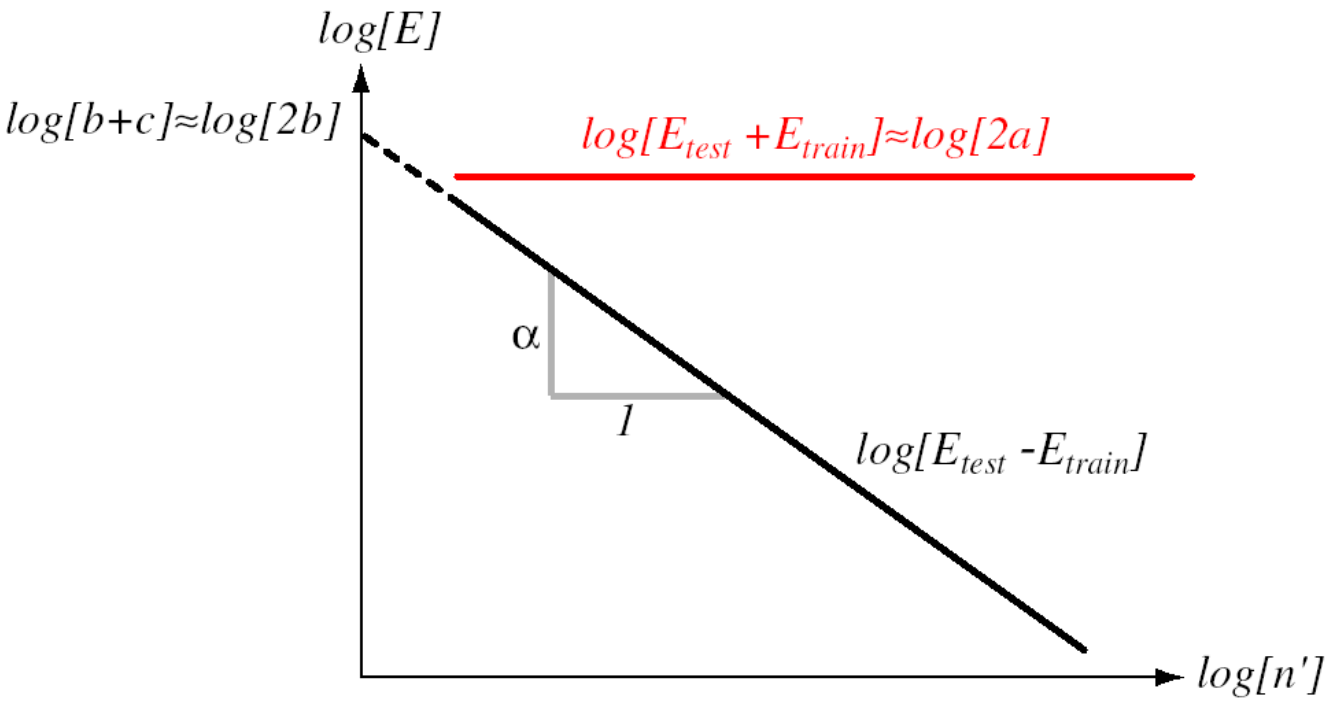
$$E_{test} + E_{train} = 2a + \frac{b}{n'^{\alpha}} - \frac{c}{n'^{\beta}} \qquad (51)$$

$$E_{test} - E_{train} = \frac{b}{n'^{\alpha}} - \frac{c}{n'^{\beta}}$$

– se assumimos $b = c$ e $\alpha = \beta$ então

$$E_{test} + E_{train} = 2a \qquad (52)$$

$$E_{test} - E_{train} = \frac{2b}{n'\alpha}$$


log[E]
log[b+c]≈log[2b]
log[E_test +E_train]≈log[2a]
α
1
log[E_test -E_train]
log[n']

– resumo:

- estimar parâmetro a para cada classificador candidato
- escolher o classificador com o menor valor de a
- treinar este classificador com conjunto pleno de dados $\mathcal{D}$

9-Resumo

- *No free lunch theorem*: na ausência de informação a priori sobre o problema, não existem razões para preferir um algoritmo de aprendizagem sobre outro

- *Ugly duckling theorem*: número de predicados compartilhado por quaisquer dois padrões diferentes é constante e não depende da escolha dos dois objetos

- *Bias* e variância caracterizam compatibilidade problema $\times$ classificador
  - *bias*: qualidade/acurácia (menor *bias*, maior compatibilidade)
  - variância: precisão/especificidade (variância grande, compatibilidade pequena)

- Reamostragem: múltiplos subconjuntos de $\mathcal{D}$ permitem determinar o valor de estatísticas arbitrárias

- Métodos para estimar precisão de classificadores e compará-los
  - validação cruzada
  - paramétrico
  - máxima verosimilhança
  - curvas de aprendizagem

Observação

Este material refere-se às notas de aula do curso CT 720 Tópicos Especiais em Aprendizagem de Máquina e Classificação de Padrões da Faculdade de Engenharia Elétrica e de Computação da Unicamp e do Centro Federal de Educação Tecnológica do Estado de Minas Gerais. Não substitui o livro texto, as referências recomendadas e nem as aulas expositivas. Este material não pode ser reproduzido sem autorização prévia dos autores. Quando autorizado, seu uso é exclusivo para atividades de ensino e pesquisa em instituições sem fins lucrativos.