Análise Forense de Documentos Digitais

Prof. Dr. Anderson Rocha

anderson.rocha@ic.unicamp.br

http://www.ic.unicamp.br/~rocha

Reasoning for Complex Data (RECOD) Lab.
Institute of Computing, Unicamp

Av. Albert Einstein, 1251 – Cidade Universitária
CEP 13083-970 • Campinas/SP – Brasil

Organização

Organização

- Validação e Métricas de Qualidade em Classificação de Padrões
- Classificadores de Padrões
  - KNN
  - LDA
  - Árvores
  - SVM
  - Coletâneas (*Bagging*)

Formas de Validação

Formas de Validação

- Ao utilizarmos aprendizado supervisionado para resolvermos um problema, temos três opções para validarmos nossos resultados [Friedman et al. 2001] [Bishop 2006]:
  - *Holdout validation*
  - *K-Fold Cross validation*
  - *Leave-one-Out validation*

Métricas de Qualidade

Métricas de Qualidade

- Problema de duas classes
- *False Negative Rate* (FNR)
- *True Positive Rate* (TPR)
- *False Positive Rate* (FPR)
- *True Negative Rate* (TNR)

Métricas de Qualidade

- Um bom classificador procura minimizar FNR e FPR e maximizar TNR e TPR
- Sabemos que FNR = 1 − TPR e FPR = 1 − TNR
- Acurácia pode ser definida como (TP + TN)/N

Classificadores de Padrões

(Algumas Abordagens)

Classificadores de Padrões

- Em Matemática e Estatística
  - Um classificador é um mapeamento a partir de um espaço de características X para um conjunto discreto de rótulos (*labels*) Y
    [Friedman et al. 2001] [Bishop 2006]

Classificadores de Padrões

- Em Inteligência Artificial
  - Um classificador é um tipo de motor de inferência que implementa estratégias eficientes para
    - computar relações de classificação entre pares de conceitos ou
    - para computar relações entre um conceito e um conjunto de instâncias [Duda et al. 2001]

Classificadores de Padrões

- K-Vizinhos mais Próximos (KNN)
- Discriminante de Fisher (LDA)
- Árvores de Classificação
- Máquinas de Vetores de Suporte (SVMs)
- Coletâneas

K-Vizinhos Mais Próximos (KNN)

KNN

- *K-Nearest Neighbors* (KNN)
- A estratégia do algoritmo K-Vizinhos é classificar um elemento de acordo com os seus vizinhos
- Noção de distância ou similaridade entre vizinhos
- Não exige treinamento explícito mas exige um conjunto de elementos marcados

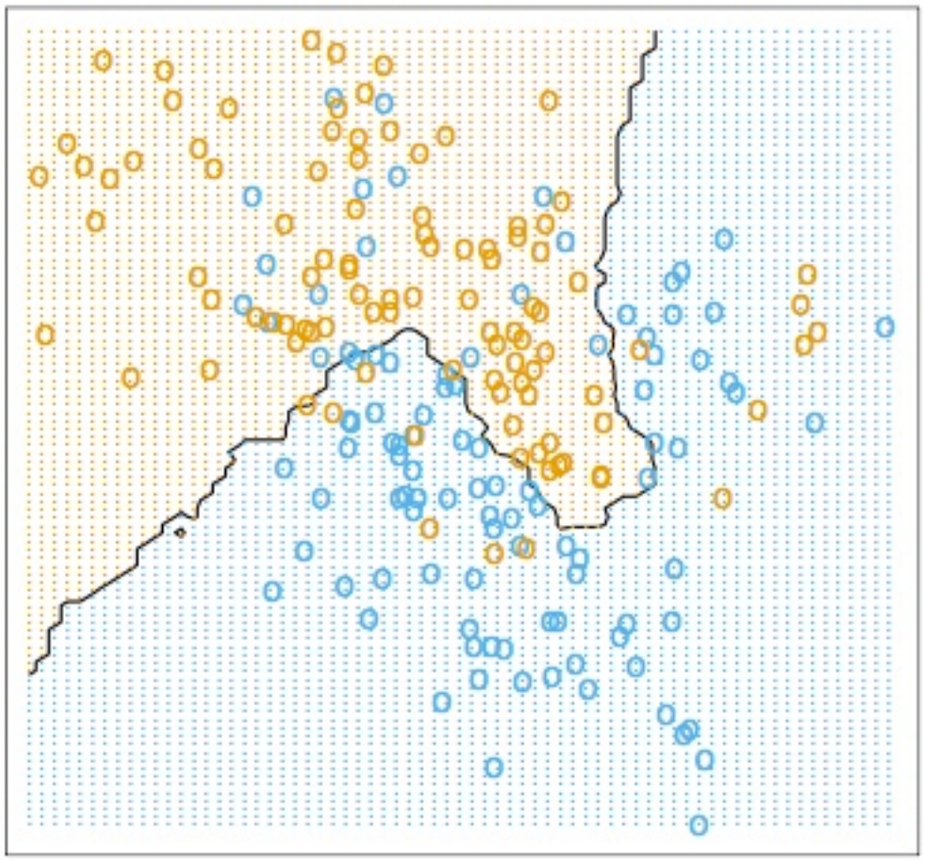
KNN

- Para classificar um elemento novo x, métodos baseados em K-Vizinhos usam observações em um conjunto de treinamento X_T próximos de x no espaço de características para formar a predição $f(x)$

$$f(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

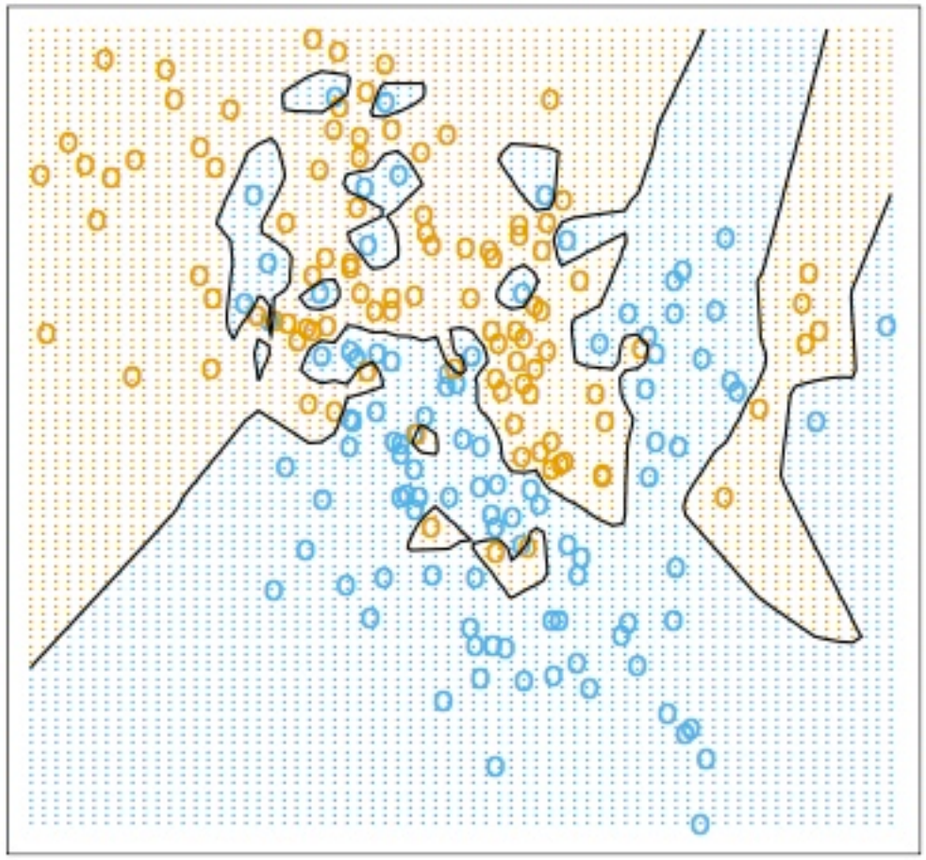
KNN

K = 15



© Friedman et al.

K = 1



© Friedman et al.

Comportamento no Treinamento!

Discriminante de Fisher (LDA)

LDA

- Linear Discriminant Analysis (LDA)
- Discriminante de Fisher
- Consiste em selecionar os componentes que maximizam a diferença entre as classes enquanto minimizam a variabilidade intra-classe [Friedman et al. 2001] [Bishop 2006]
- Por simplicidade, considere um problema de duas classes $Y(\text{-}1 = \text{a} \mid +1 = \text{b})$

LDA

- Seja X_a uma família de exemplares pertencentes à classe *a* e X_b uma família de exemplares pertencentes à classe *b* no conjunto de treinamento
- Considere $|X_a| = N_a$ e $|X_b| = N_b$
- Adicionalmente, supomos que ambas as classes seguem uma distribuição Gaussiana

LDA

- Médias intra-classe

$$\mu_a = \frac{1}{N_a} \sum_{x_i \in X_a} x_i \qquad \mu_b = \frac{1}{N_b} \sum_{x_j \in X_b} x_j.$$

- Média entre as classes (*a*,*b*)

$$\mu = \frac{1}{N_a + N_b} \left(\sum_{x \in X_a \cup X_b} x \right).$$

LDA

- Definimos uma matriz de dispersão intra-classe para medir o quanto cada classe varia

$$S_w = M_a M_a^T + M_b M_b^T$$

onde a i-ésima linha da matriz M_a contém a diferença $(x_i^a - \mu_a)$

- O mesmo procedimento se aplica a M_b

LDA

- Definimos agora a dispersão entre as classes

$$S_{bet} = N_a(\mu_a - \mu)(\mu_a - \mu)^T + N_b(\mu_b - \mu)(\mu_b - \mu)^T$$

- Para maximizarmos a diferença entre as classes e minimizarmos a variabilidade intra-classe em uma simples dimensão, é suficiente calcularmos o autovalor-autovetor generalizado $\vec{e}$ maximal de S_{bet} e S_w

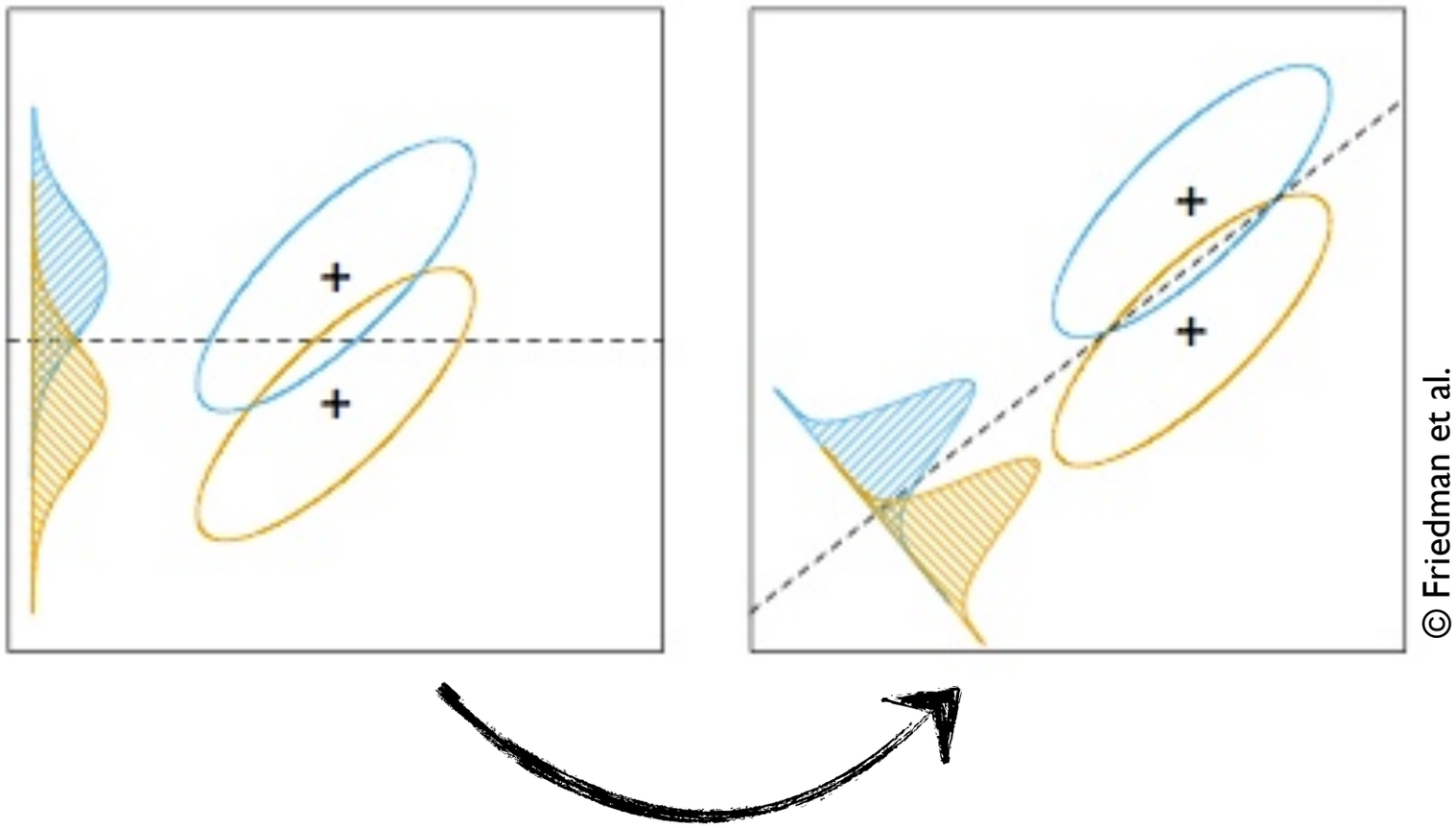
LDA

- Podemos calcular o autovalor-autovetor generalizado fazendo

$$S_{bet}\vec{e} = \lambda S_w \vec{e}$$

- Com o autovetor generalizado maximal, nós podemos projetar as amostras em um subespaço linear e aplicar um limiar para efetuarmos a classificação

LDA



© Friedman et al.

Árvores de Classificação (CTREES)

Árvores de Classificação

- *Classification Trees* (CTREES)
- Propriedades estatísticas da teoria da informação para proceder a classificação
- Alguns algoritmos conhecidos que aplicam esta abordagem são o ID3 e suas evoluções como as abordagens C4.5 e C5.0 [Mitchell 1997]

Árvores de Classificação

- Embora não descritas aqui, existem outras abordagens não baseadas na teoria da informação como a abordagem CART baseada em **regressão univariada** [Friedman et al. 2001]
- A maioria dos algoritmos baseados em árvores de classificação empregam técnicas gulosas *top-down* para analisar o espaço de possíveis soluções

Árvores de Classificação

- Em uma abordagem gulosa, em cada iteração, procuramos selecionar os atributos que são mais propícios a serem nós da árvore
- Para a seleção de atributos, aplicamos um teste estatístico para avaliarmos cada atributo (característica) de modo a analisarmos quão bem ele classifica, sozinho, os exemplos presentes no treinamento

Árvores de Classificação

- Considere um problema de duas classes

$$Y(\text{-}1 = \text{a} \mid +1 = \text{b})$$

- Caso tenhamos *N* exemplos no treinamento e cada exemplo possua *p* características
  - no primeiro passo do algoritmo selecionamos o atributo que melhor classifica os *N* exemplos presentes no treinamento
  - para isso, usamos o critério estatístico adotado

Árvores de Classificação

- O processo é repetido para cada nó folha descendente produzido até que
  - o número total de atributos seja avaliado ou
  - todos os exemplos no treinamento associado a este determinado nó folha pertençam à mesma classe (e.g., todos os exemplos pertençam à classe $Y = +1$)

Árvores de Classificação

- Ao utilizarmos árvores de classificação, temos que considerar dois pontos importantes
  - a seleção do atributo a ser testado em cada nó
  - discretização – seleção de limiares para atributos não-discretos

Seleção do Atributo

- Definimos uma propriedade estatística chamada ganho de informação (GI)
- Mede quão bem um dado atributo separa os exemplos de treinamento de acordo com a classificação esperada
- Para definir GI, definimos, primeiramente, Entropia

Seleção do Atributo

- Entropia caracteriza o grau de organização em um conjunto arbitrário de exemplos
- Dada uma coleção de exemplos *S* contendo exemplos pertencentes tanto à classe *a* quanto à nossa classe *b*, a entropia de *S* é dada por

$$E(S) = -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

onde $p_{\oplus}$ é a proporção de exemplos + em S

Seleção do Atributo

- Tendo E(*S*) definida, buscamos maximizar o ganho de informação GI para um determinado atributo A como

$$GI(S, A) = E(S) - \sum_{v \in V(A)} \frac{|Sv|}{|S|} E(S_v)$$

- onde V(A) é o conjunto de todos os valores possíveis para o atributo A e S_v é o subconjunto de S para o qual A tem o valor v

Seleção do Atributo

- GI mede a redução esperada na entropia para um determinado atributo A
- Quanto maior o GI de um atributo, melhor este atributo separa os dados [Mitchell 1997]

Discretização de Valores Contínuos

- Quando trabalhamos com um conjunto *S* cujos atributos podem ter valores contínuos, temos outro problema a tratar:
  - selecionarmos um limiar que transforme cada variável contínua em uma variável discreta – discretização
- Para cada atributo, temos que selecionar o limiar *c* que produza o maior ganho de informação

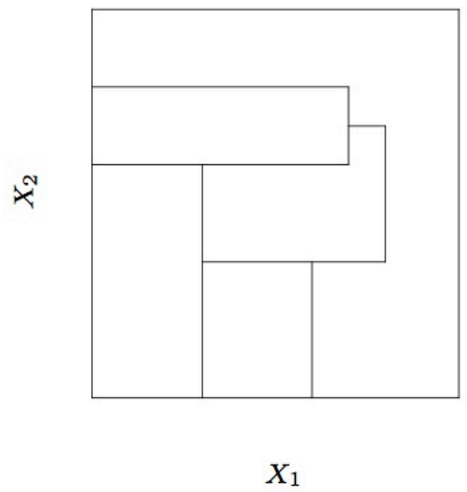
Discretização de Valores Contínuos

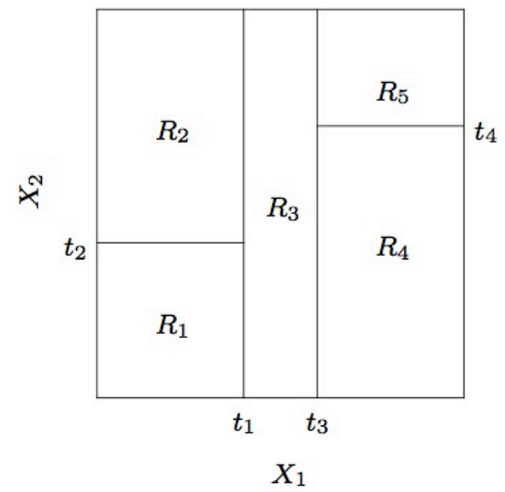
- Para isso, ordenamos nossos exemplos segundo o valor da variável contínua sendo analisada
- Geramos um conjunto de limiares candidatos selecionando exemplos adjacentes que tenham classificação diferente
- Para cada limiar candidato, dividimos o conjunto de treinamento segundo este limiar e calculamos o GI associado

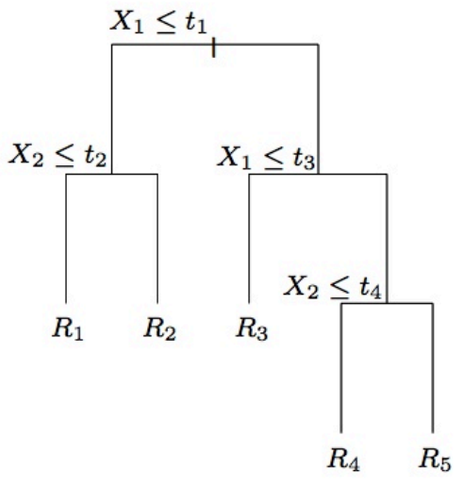
Discretização de Valores Contínuos

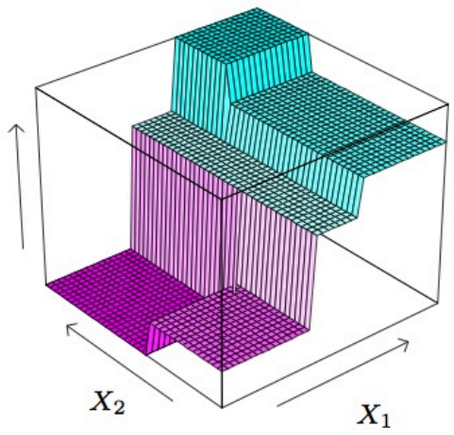
- Selecionamos o limiar que produz o maior GI
- Aplicamos o procedimento recursivamente até atingirmos o critério de parada

Árvores de Classificação









© Friedman et al.

Máquinas de Vetores de Suporte (SVM)

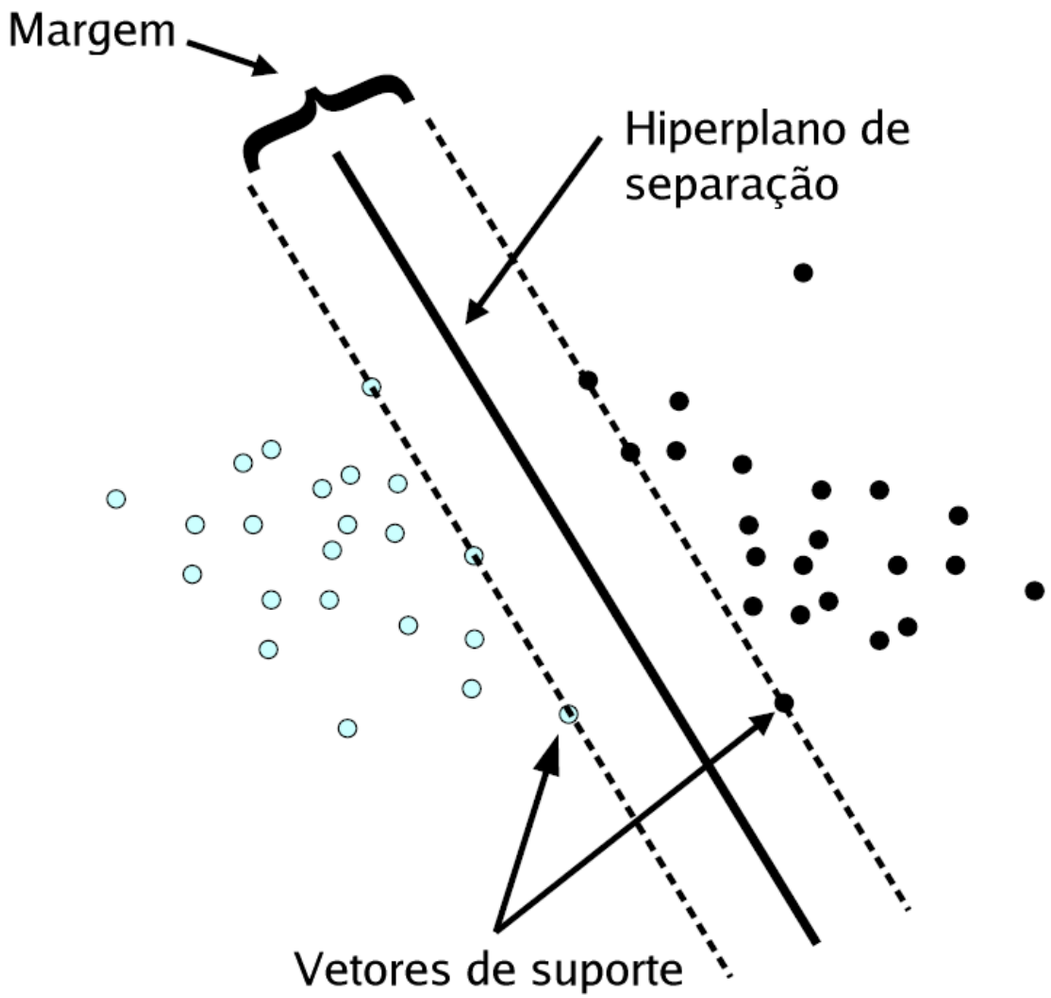
SVMs

- *Support Vector Machines* (SVMs)
- Tema de grande interesse na comunidade de aprendizado de máquina criando um entusiasmo similar ao que as redes neurais artificiais despertou anos atrás
- A teoria geral dos SVMs foi apresentada por [Cortes & Vapnik 1995] **para classificação binária (duas classes)**

SVMs

- Nessa abordagem, basicamente estamos procurando por um hiperplano de separação ótima entre duas classes [Schlkopf & Smola 2001] [Friedman et al. 2001] [Bishop 2006]
- Isto é feito através da maximização da margem
- Margem é a menor distância entre um ponto pertencente à classe *a* e um ponto pertencente à classe *b*

SVMs



© A. Rocha

SVMs

- Os pontos sobre as fronteiras são chamados vetores de suporte (*support vectors*)
- O meio da margem é o nosso hiperplano de separação ótima
- Para os pontos que estão do lado "errado" da margem discriminante, nós recalculamos os coeficientes de forma a reduzir sua influência (*soft margin methods*)

SVMs

- Quando não é possível achar um separador linear, os dados "são projetados" em um espaço de maior dimensão onde tornam-se linearmente separáveis
- Obviamente, a projeção não é feita de forma explícita
- A projeção é feita a partir de técnicas baseadas em *kernels* [Schlkopf & Smola 2001]

SVMs

- SVMs podem ser divididos em três categorias
  - linearmente separável
  - linearmente não-separável
  - não-linear

SVMs Linearmente Separáveis

- Suponha um problema de classificação de duas classes $Y(-1 = a \mid +1 = b)$
- Denote a tupla (x_i, y_i) com $i = 1, \ldots, N$ como sendo o nosso conjunto de treinamento
- O vetor coluna x_i contém as características que nós achamos suficientes para distinguir os dois possíveis grupos e $y_i = f(x_i)$

SVMs Linearmente Separáveis

- Com um SVM linearmente separável, queremos achar o hiperplano linear que separa as duas classes de nosso problema
- Vamos expressar a maximização da *margem* como uma função do vetor de pesos *w* e o viés (*bias*) *b* do hiperplano de separação

SVMs Linearmente Separáveis

- Assim, a distância entre um ponto *x* e um plano (*w*, *b*) é

$$\frac{w^T x + b}{||w||},$$

onde ||.|| é a norma Euclidiana.

- Hiperplano ótimo pode ter infinitas soluções devido às infinitas possibilidades de escalas de *w* e *b*

SVMs Linearmente Separáveis

- Assim, escolhemos a solução em que a função discriminante se torna 1 para os exemplos de treinamento próximos da fronteira
- Conhecido como hiperplano canônico

$$|w^T x_i + b| = 1.$$

SVMs Linearmente Separáveis

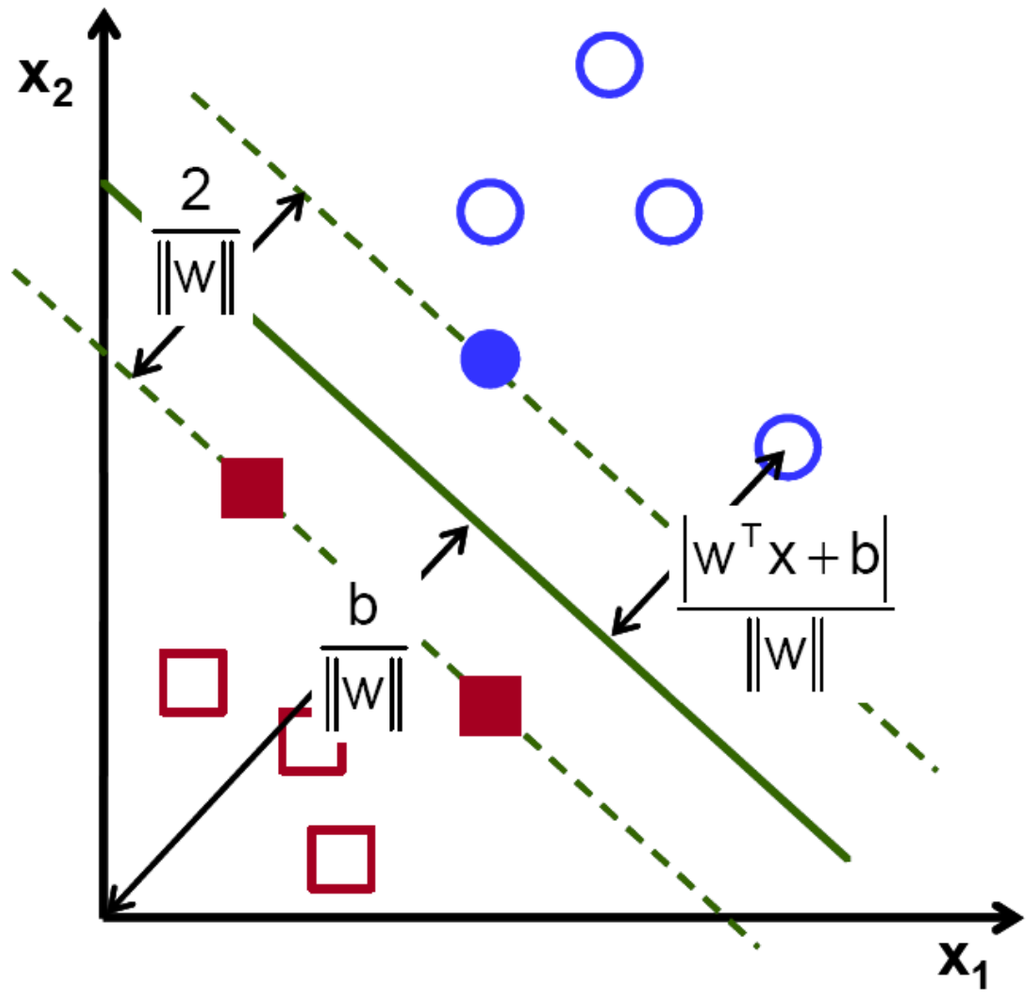
- Desta forma, a distância dos exemplos mais próximos à fronteira é

$$\frac{|w^T x + b|}{||w||} = \frac{1}{||w||}$$

- E a margem se torna

$$m = \frac{2}{||w||}.$$

SVMs Linearmente Separáveis



SVMs Linearmente Separáveis

- O problema de maximizar a margem é equivalente a

$$\min J(w) = \frac{1}{2}||w||^2 \text{ sujeito a } y_i(w^T x_i + b) \geq 1 \ \forall i.$$

- Por definição, $J(w)$ tem um mínimo global
- Para resolver esta função, podemos empregar técnicas de Otimização Lagrangiana

SVMs Linearmente Separáveis

- A introdução de multiplicadores de Lagrange para resolver este problema resulta em

$$L_P(w, b, \alpha_1, \ldots, \alpha_N) = \frac{1}{2}||w||^2 - \sum_{i=1}^{N} \alpha_i (w^t x_i + b) y_i + \sum_{i=1}^{N} \alpha_i,$$

- Esta função de erro precisa ser minimizada com relação a *w* e *b*

SVMs Linearmente Separáveis

- Problema de otimização quadrática
- Definimos a *margem* para qualquer hiperplano dado como a soma das distâncias do hiperplano aos exemplares mais próximos das duas classes.
- O hiperplano é escolhido de forma a maximizar a margem

SVMs Linearmente Separáveis

- A partir do hiperplano de separação (*w*, *b*), um novo exemplar *z* pode ser classificado simplesmente a partir do cálculo sobre qual lado do hiperplano ele se encontra
- Se o valor $w^t z + b \geq 0$ então o exemplar é classificado como +1, caso contrário como -1

SVMs Linearmente Não-Separáveis

- Quando os dados não estão uniformemente dispostos em algum dos lados de um hiperplano, um SVM linearmente separável não resultará em uma solução razoável
- Podemos tratar tal situação relaxando as restrições iniciais através da introdução de variáveis de folga (*slack variables*)

SVMs Linearmente Não-Separáveis

- Variáveis de folga relaxam as restrições do hyperplano canônico

$$y_i|w^T x_i + b| \geq 1 - \xi_i \; \forall i = 1 \ldots N$$

- Com a introdução de variáveis de folga, o objetivo agora é achar o hiperplano de separação que produza a menor taxa de erros no treinamento, $\sum_i \xi_i$ enquanto maximiza a *margem*

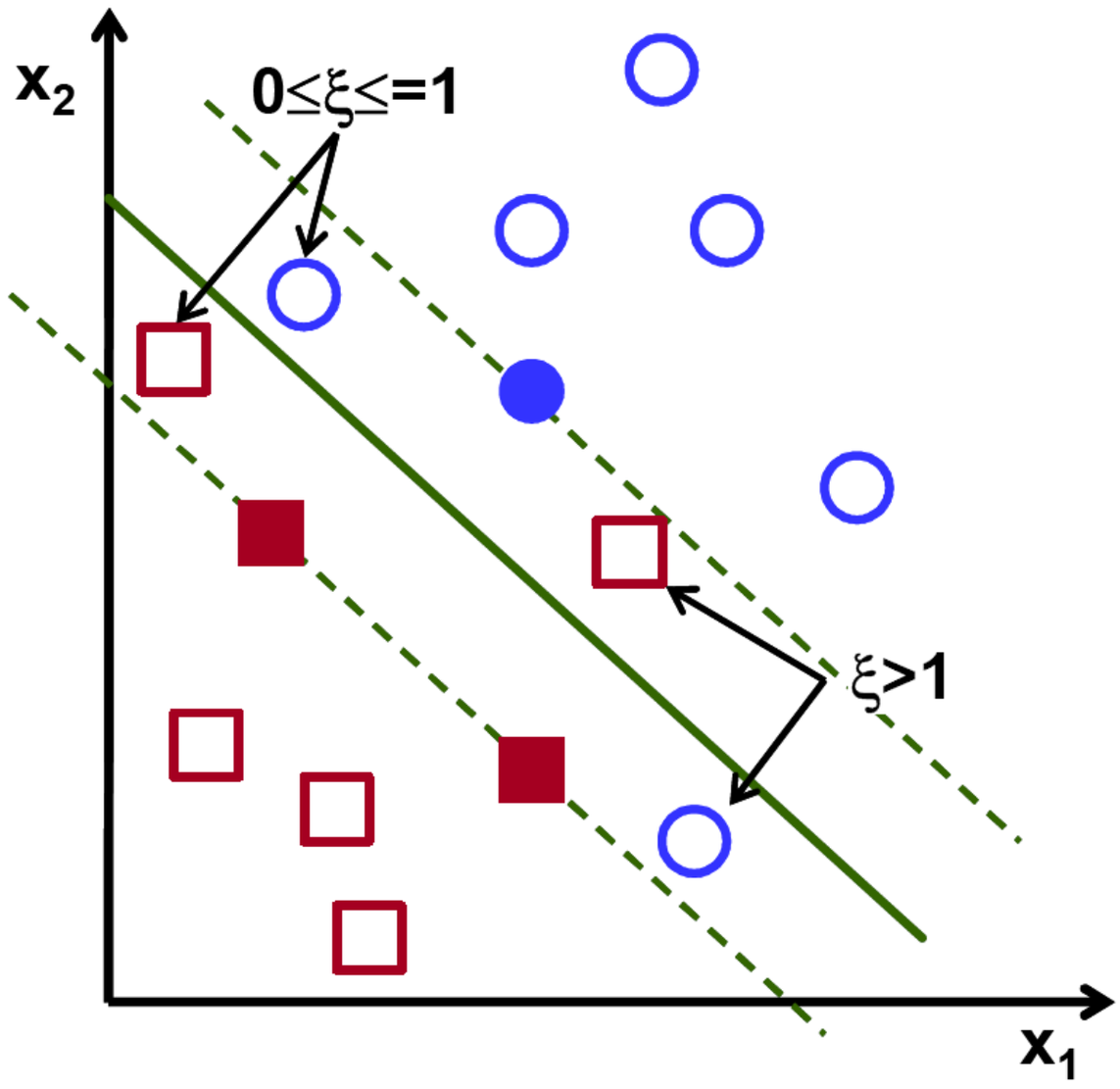
SVMs Linearmente Não-Separáveis

- Função de erro a ser minimizada

$$\frac{||w||^2}{2} + C \sum_{i=1}^{N} \xi_i$$

- Valores altos de *C* implicam poucos erros de classificação e maior complexidade no classificador
- Valores baixos para *C* implicam preferência por soluções de baixa complexidade e com maior quantidade de erros na classificação

SVMs Linearmente Não-Separáveis



SVMs Não Lineares

- Uma característica dos SVMs vistos até o momento é que eles estão restritos a hiperplano linear
- Há casos em que um hiperplano linear não é capaz de garantir a classificação
- Para isso, usamos SVMs não lineares

SVMs Não Lineares

- Para atingir nosso objetivo, mapeamos os exemplos de treinamento num espaço Euclidiano de mais dimensões
- Mapeamento é implícito e não explícito

$$\phi : \mathcal{L} \rightarrow \mathcal{H}$$

- O erro a ser minimizado depende apenas de produtos internos sobre exemplares do conjunto de treinamento $x_i^t x_j$

SVMs Não Lineares

- Definimos uma função *kernel K* capaz de fazer este mapeamento sob demanda (*on-the-fly*)

$$K(x_i, x_j) = \phi(x_i)^t \phi(x_j)$$

- Há várias escolhas possíveis para *K* dentre elas
  - funções radiais base (*Radial Basis Functions* – RBF)
  - polinomiais etc.

SVMs Não Lineares

- Um novo exemplar z, é classificado pela determinação sobre qual lado do hiperplano de separação (w, b) ele se encontra
- Se o valor $w^t\phi(z) + b \geq 0$ ele é classificado como da classe +1
- Caso contrário, ele é classificado como -1

SVMs Não Lineares

- Teste precisa ser feito no espaço de maior dimensão usando a mesma função *kernel*

$$w^t \phi(z) + b = \sum_{i=1}^{N} \alpha_i K(x_i, z) y_i + b$$

- Como o SVM funciona no cenário de mais de duas classes?

Coletâneas

Coletâneas – *Bagging*

- *Bagging* ou *Bootstrap Aggregation*
- Avaliamos as predições sobre uma coleção de amostras (*bootstrap samples*)
- *Bootstrap samples* são amostras com reposição e mesmo tamanho feitas a partir de um conjunto de dados

Coletâneas – *Bagging*

- A amostragem serve para reduzir sua variância e consequentemente o erro na predição
- Criamos uma coletânea (*ensemble*) a partir do treinamento individual de classificadores nas amostras feitas do conjunto de dados

Coletâneas – *Bagging*

- Seja X o conjunto de dados de entrada e

  $Z^i, i = 1, 2, \ldots, B$ uma amostra de X

- Para fazermos a classificação em cada Z^i selecionamos um classificador (LDA, CTREE, etc.)

- Em cada passo, nós construímos um conjunto de treinamento fazendo amostragens com reposição a partir do conjunto de dados original

Coletâneas – *Bagging*

- Cada classificador é treinado em média sobre 63.2% dos exemplos do conjunto de treinamento
- Para cada elemento *x*, armazenamos a predição $\vec{f}^{i}(x)$
- Definimos a estimativa da coletânea para *x* como

$$\vec{f}_{bag}(x) = \frac{1}{B}\sum_{i=1}^{B}\vec{f}^{i}(x)$$

Coletâneas – *Bagging*

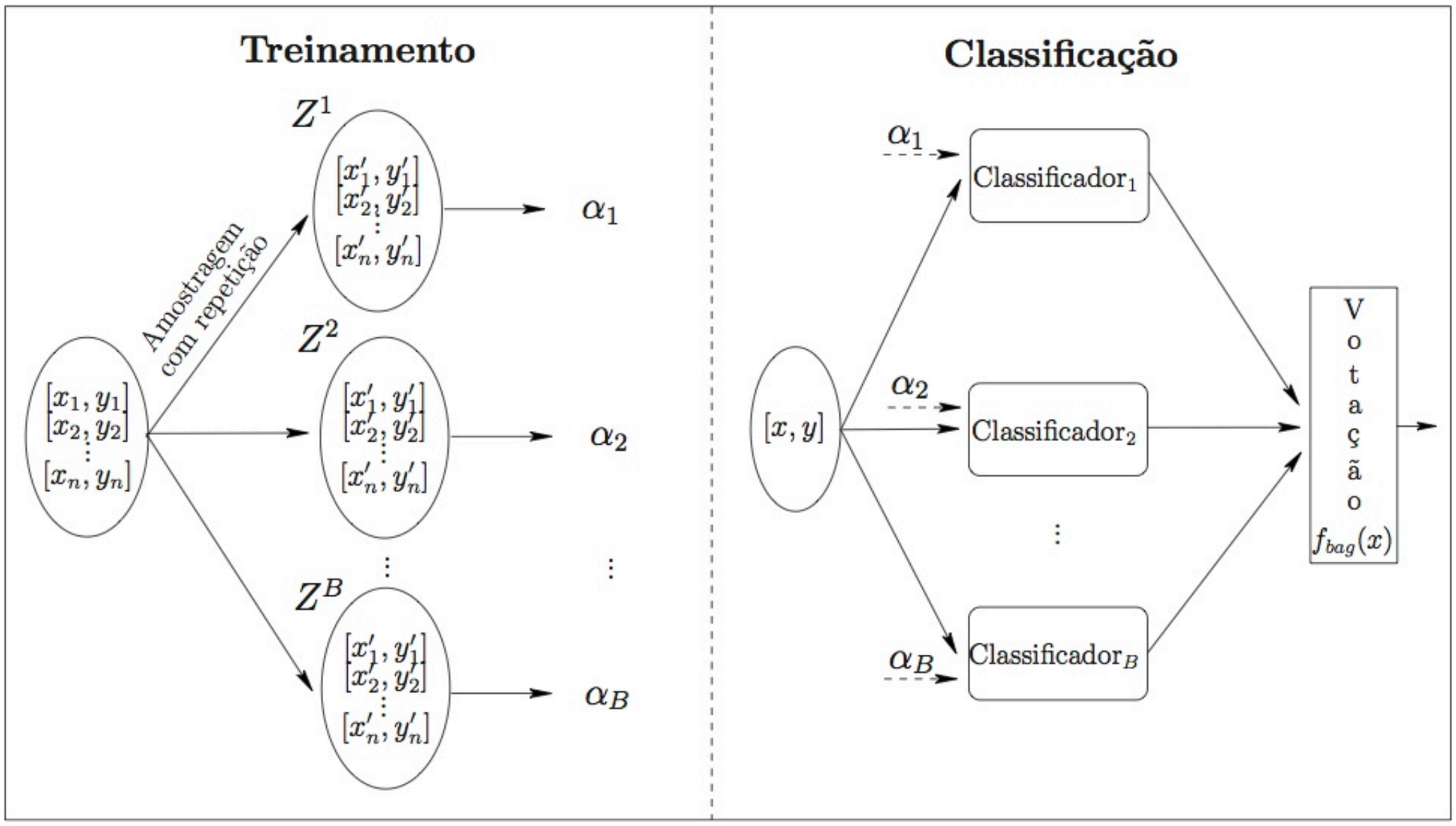
- A equação

$$\vec{f}_{bag}(x) = \frac{1}{B}\sum_{i=1}^{B}\vec{f^i}(x)$$

  funciona como uma votação dentro da coletânea

- Ao avaliarmos todas as classificações feitas para *x*, nós o classificamos como pertencente à classe para a qual ele foi classificado mais vezes

Coletâneas – *Bagging*



Coletâneas – *Bagging*

- Armazenamos os coeficientes relativos a cada classificador aplicado (α)
- Ao recebermos um exemplar *z* a ser testado, submetemos este exemplar aos *B* classificadores e procedemos a votação

Referências

Referências

1. [Bishop, 2006] **C. M. Bishop**. *Pattern Recognition and Machine Learning*. Springer, 1 edition, 2006.
2. [Cortes & Vapnik 1995] **C. Cortes and V. Vapnik** (1996). *Support-vector networks*. Machine Learning (ML), 20(3): 273–297, March 1995.
3. [Duda et al. 2001] **R. O. Duda, P. E. HART and D. G. STORK**. *Pattern Classification*. Wiley-Interscience, 2, 2000.
4. [Friedman et al. 2001] **J. Friedman, T. Hastie, and R. Tibshirani**. *The Elements of Statistical Learning*. Springer, 1 edition, 2001.
5. [Mitchell 1997] **T. M. Mitchell**. *Machine Learning*. McGraw-Hill, 1 edition, 1997.
6. [Schlkopf & Smola 2001] **Bernhard Schlkopf and Alexander J. Smola**. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, Boston, MA, USA, first edition, 2001.

Obrigado!