Support Vector Machines

Bruno Coelho • 27/05/2020
bruno.gomes.coelho@usp.br

Support Vector Machines - SVM

- Modelo de classificação e regressão supervisionado
- Só funciona com dados numéricos
- Intuição geométrica bem interessante
- Muito estudado matematicamente
    - Garantias do “melhor” classificador possível!

SVM - Tópicos:

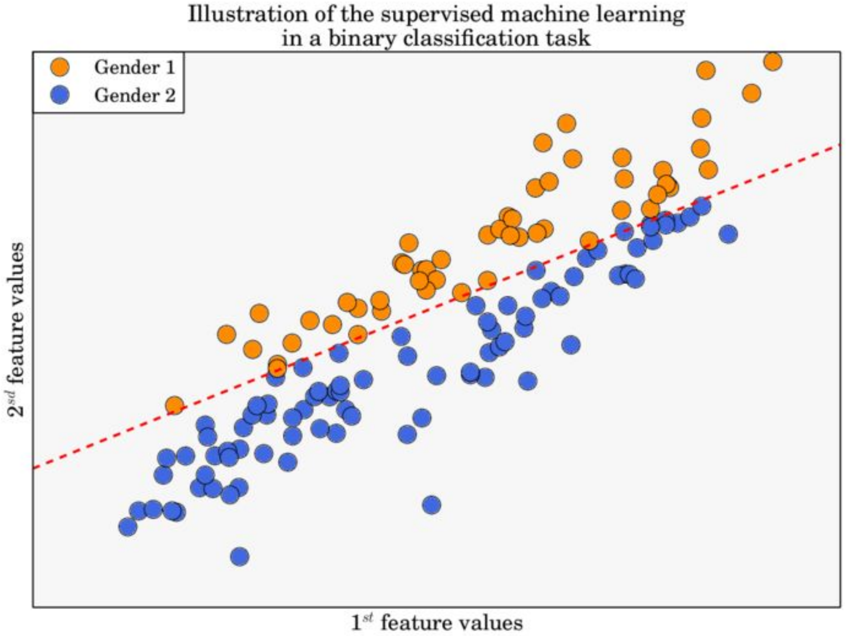
- Intuição
- Derivação simples original
- Extensões
- Aplicabilidade
- Tópicos avançados

Histórico das SVM

- A primeira versão foi:
  - SVM Linear - **Vladimir N. Vapnik**, Alexey Ya. Chervonenkis in 1963.
  - Vapnik é considerado o pai de toda uma área de ML (Teoria do Aprendizado Estatístico) e teve diversas contribuições para a área
  - Essa primeira versão era muito simples….
- Em 1992 usa se o "truque do kernel" para deixar o SVM melhor
- E em 1995 adiciona o "soft margin" para a versão que as bibliotecas usam hoje em dia

Intuição por trás

- Vamos pensar em um problema de classificação binária:



Intuição por trás

- Queremos separar linearmente o nosso conjunto de dados
  - Lembra algum algoritmo?
  - Como criamos a separação naquele caso?
  - Tente responder antes de passar pro próximo slide

Intuição por trás

- Queremos separar linearmente o nosso conjunto de dados
  - Lembra algum algoritmo?
    - A **regressão logística** faz isso!
  - Como criamos a separação naquele caso?
    - Utilizamos um **hiperplano** (em 2D uma reta, 3D um plano, etc)
    - Minimizamos uma **loss** (“erro”) utilizando **Gradient Descent** para ajustar nossos pesos

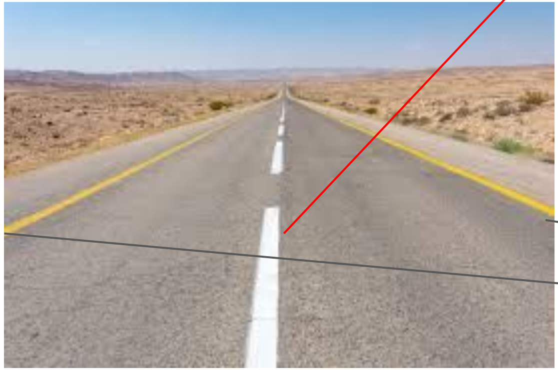
Voltando para SVM… Como calcular a separação?

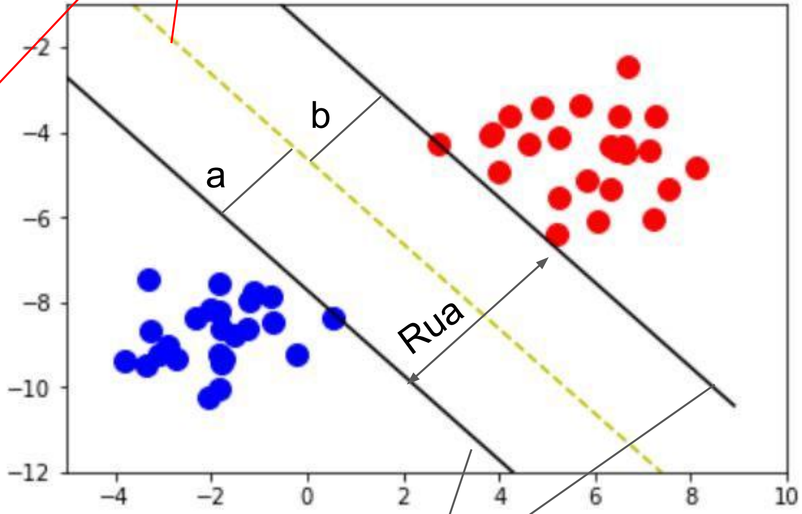
- Para mostrar o algoritmo, supor que **existe uma separação linear**
  - Em geral isso nunca é verdade mas suponha por enquanto...
- Em geral, temos mais de 1 possibilidade para o hiperplano
  - vamos escolher o que fica bem no **meio** entre as duas classes
- O hiperplano é o meio entre as margens de cada classe
- Vamos ver um exemplo:

Intuição por trás

- Volte para essa imagem sempre que ficar confuso :)
- Analogia com uma “rua”:

hiperplano separador





Margens ou calhas

[Fonte]

Intuição por trás

- A ideia é “*maximum street approach*” - maximizar o espaço da “rua”
    - As margens expandem o máximo que dá, até “bater” em um elemento de cada classe
- Somos bons arquitetos, então queremos os dois lados da rua iguais
    - Isto é, a faixa da esquerda e da direita têm o mesmo tamanho - a divisão vai no meio
    - ($a = b$) no desenho

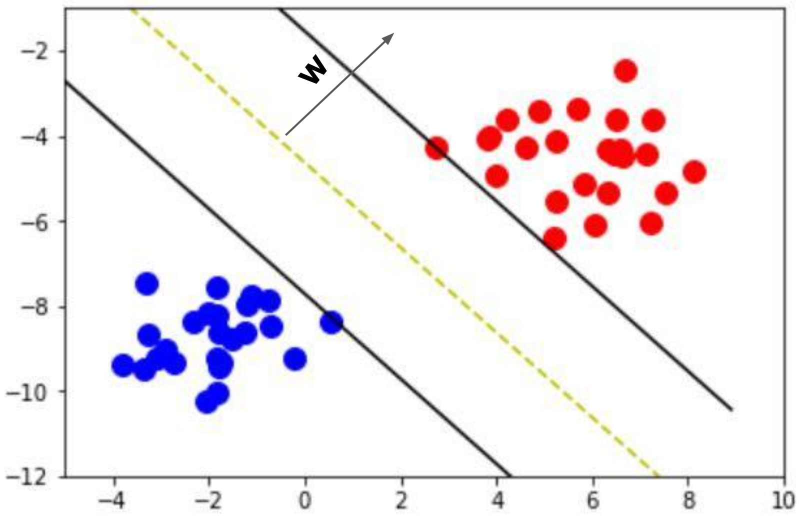
Classificar um dado

- Vamos por enquanto supor que temos o hiperplano e queremos classificar um novo dado.
- Como fazer?

Ou veja esse vídeo

Classificar um dado

- Como fazer?
  - Basta ver para qual lado do hiperplano ele está!
- Criamos um vetor auxiliar, **W**, perpendicular ao hiperplano.
- Fazemos **W** ter norma 1



Ou veja [esse vídeo]

Classificar um dado

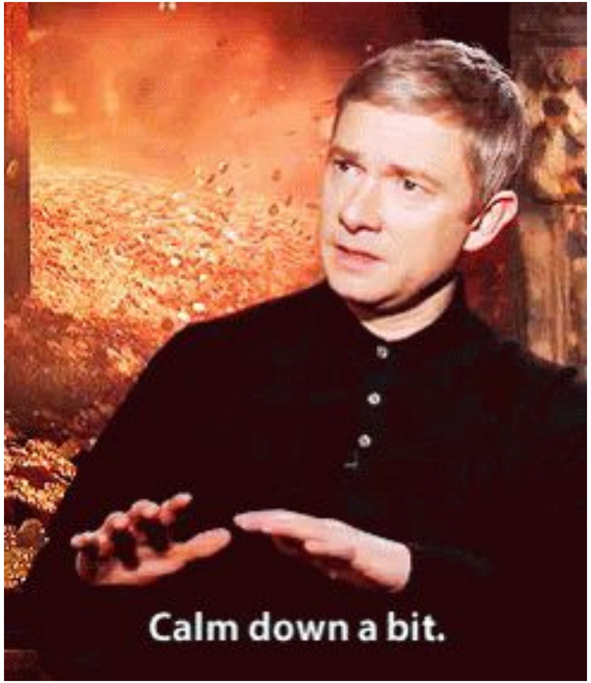
- Entendemos um ponto x1 que queremos classificar como um vetor u1
    - É simplesmente o vetor que liga a origem ao ponto
- Projetamos u1 na direção do hiperplano utilizando w: projeção = w*b
- Dado b = distância da origem para o hiperplano,
- Verificamos se a projeção está acima ou abaixo de b:

if $w^T u1+b > 0$ then (+) class
if $w^T u1+b < 0$ then (-) class

if $w^T u1+b=0$ then its on decision boundary

Ou veja [esse vídeo]

Mas como obtenho o hiperplano?



Como obter o hiperplano?

- Vamos adicionar mais restrições/complicações ao problema. Porque?
  - Vai nos ajudar a chegar numa fórmula fechada!
  - Diz a lenda que Vapnik bebeu 7 litros de café ao longo de 3 meses antes de chegar nessas restrições [1]
- Adicionamos algumas restrições quanto:
  - Ao nosso conjunto de treino
  - Aos elementos que estão na “margem”

[1]

Restrições

- Nosso y será -1 para a classe negativa e +1 para classe positiva
  - Para a derivação da SVM, não funciona ter y 0 ou 1, precisamos da classe negativa sendo -1.
- X+ são todos os exemplos da classe positiva, X- da classe negativa
- Forçamos:

$$w \cdot x_{+} + b \geq +1$$
$$w \cdot x_{-} + b \leq -1$$

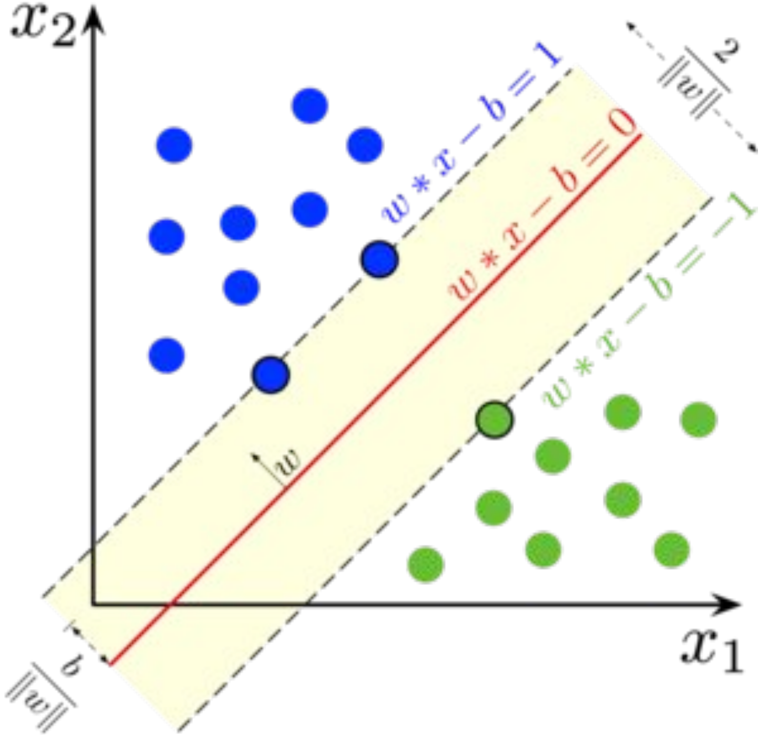
- Perceba que antes tinhamos 0 em vez de 1 devido devido ao fato que estamos supondo que o hiperplano existe

Tamanho da rua

- Podemos calcular o tamanho da rua utilizando: (X+ - X-) * W
- Simplificando, chegamos em:
  - Tamanho da rua = 2/ ||W||
- Num problema de minimização:
  - As constantes não vão importantar
  - Maximizar a 1/X é a mesma coisa que minimizar X
  - Minimizar X é a mesma coisa que minimizar X^2
- Então queremos: min ($||W^2||$)

Novamente, esse vídeo esse vídeo faz a derivação passo a passo

Visualmente



Lembrando...

$$min(\frac{1}{2}\left\|W^2\right\|)$$

Adicionamos o ½ para a derivada ser mais fácil. Sujeito a:

$$y_i * (w \cdot x + b) \geq 0$$

$$y_i * (w \cdot x + b) = 0, \forall x_i \in \text{“}calha\text{”}$$

Como realizar essa otimização para obter o W?

- Entregamos o problema para matemáticos e eles nos dão a resposta:
    - Multiplicadores de Lagrange

Multiplicadores de Lagrange

- Método de otimização:
  - Encontrar o mínimo/máximo de uma função
  - Funciona com restrições de igualdade
  - Nos mostra que a otimização depende apenas **do produto escalar**
  - Nos dá uma resposta **exata** nesse caso pois o espaço é convexo
- Com ele temos uma resposta exata para o nosso vetor **w**
- Por agora não nos importa os detalhes

Revisando

Temos um método para dado um conjunto de treinamento binário:

- Tenta encontrar a melhor “rua” que separa os dados **linearmente**
- Nos dá uma resposta exata para esses dados
- Nos fornece uma regra de classificação para dados novos

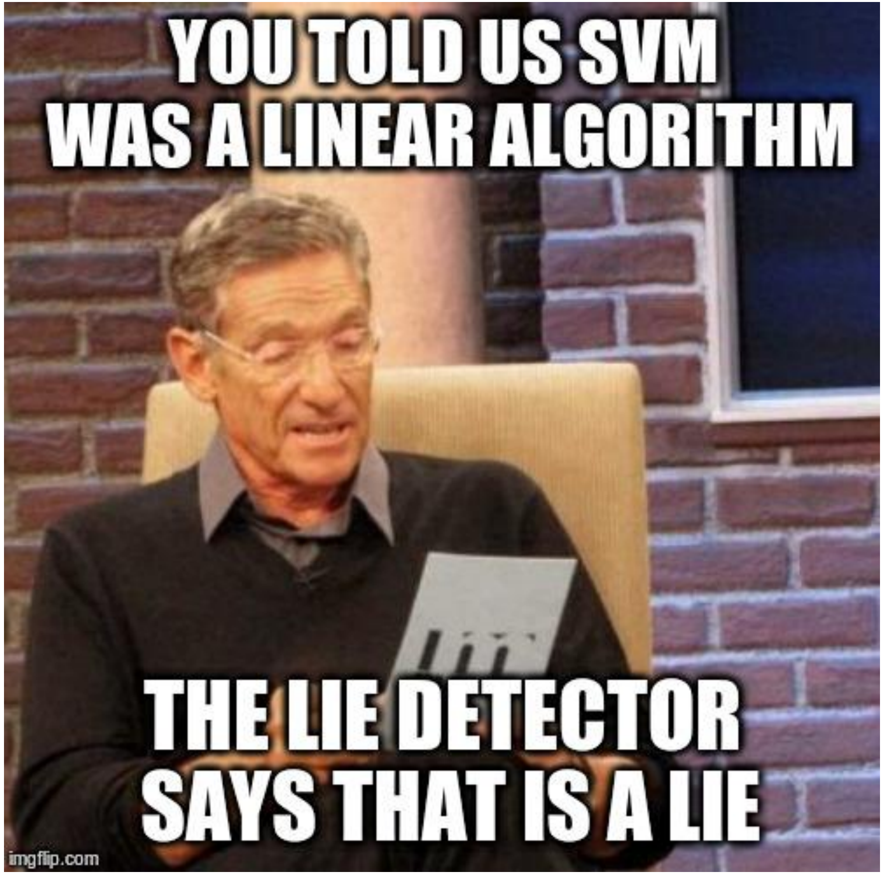
Queremos:

Maior divisão linear possível :)

Problema:

Muitas vezes uma divisão linear não é o suficiente!

Kernels


YOU TOLD US SVM
WAS A LINEAR ALGORITHM
THE LIE DETECTOR
SAYS THAT IS A LIE
imgflip.com

Queremos:

Não linear! -> Kernels

Queremos:

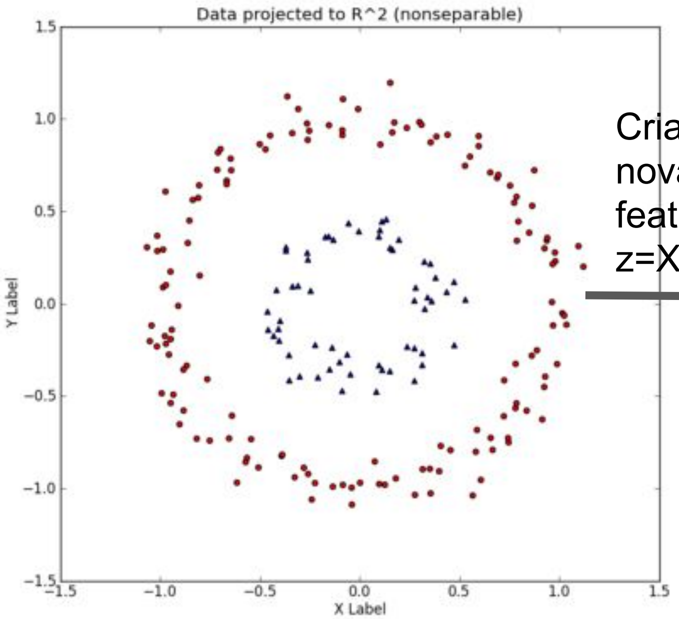
Não linear! -> Kernels

Kernels: Motivação

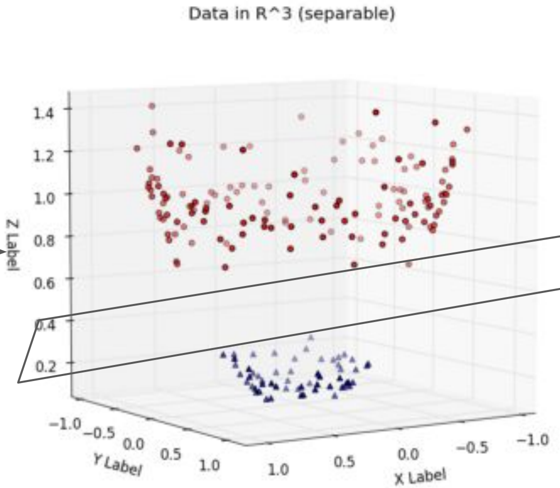
- Se tivermos um conjunto de dados onde um círculo de raio r1 é a classe + e outro de raio r2 é a classe -, não existe como separar os dados com uma reta!
  - Imagem 1 próximo slide
- Mas se adicionarmos uma dimensão a mais, $z = x^2 + y^2$, podemos dividir o espaço tri-dimensional com 1 hiperplano!
  - Imagem 2 próximo slide

Kernels: Motivação

Hiperplano separador
após a nova feature



Criando
nova
feature
$z=X^2+y^2$



[Fonte](#)

Kernels

- Nada a ver com linux
- Vamos fazer uma mudança no espaço dos nossos dados
- Como só precisamos do produto escalar entre 2 vetores (multiplicadores de lagrange), teoricamente só precisamos disso no novo espaço
- Kernels nos fornecem esse **produto escalar sem ter que mudar o espaço**

Kernels: Exemplo polinomial 2

- **U**, **V** são vetores - K é o nosso kernel - Phi é a nossa transformação no espaço
- Por exemplo se mapearmos de 2D para 3D:

$$u, v \in R^2$$

$$K(u, v) = \Phi(u) \cdot \Phi(v)$$

$$\Phi : R^2 \rightarrow R^3$$

$$\Phi(x) = \Phi((x_1, x_2)) = (x_1^2, \sqrt{2} * x_1 * x_1, x_2^2)$$

Kernels: Exemplo polinomial 2

- A “mágica” de kernels é que em vez de transformarmos os dados e daí calcular o produto vetorial nesse espaço de mais alta dimensionalidade…
- … Podemos chegar numa fórmula que nos dê o produto vetorial direto!
- Isso é muito mais eficiente computacionalmente e permite usar transformações de grau maior.

Tipos de kernels usados

Kernel	Equation
Linear	$K(x, y) = x \cdot y$
Sigmoid	$K(x, y) = \tanh(ax.y + b)$
Polynomial	$K(x, y) = (1 + x \cdot y)^d$
KMOD	$K(x, y) = a\left[\exp\left(\frac{\gamma}{\lVert x-y \rVert^2+\sigma^2}\right) - 1\right]$
RBF	$K(x, y) = \exp(-a\lVert x - y \rVert^2)$
Exponential RBF	$K(x, y) = \exp(-a\lVert x - y \rVert)$

Fonte

Queremos:

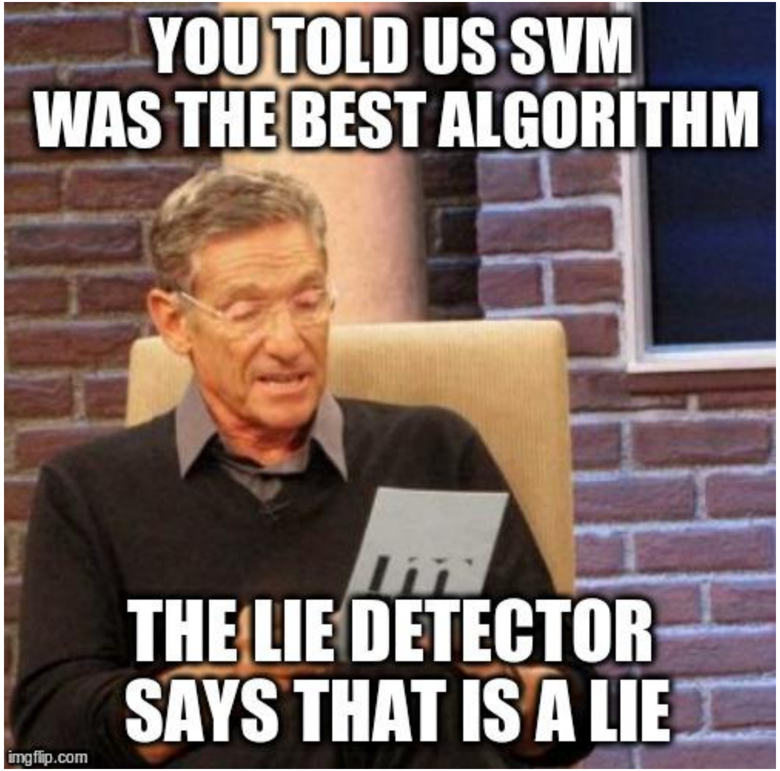
Melhor divisão “linear”* possível :)

*No espaço do kernel, a divisão ainda é linear

Problema:

Mesmo usando kernels, ainda pode ser impossível acertar todos os exemplos

Soft Margin


YOU TOLD US SVM
WAS THE BEST ALGORITHM
THE LIE DETECTOR
SAYS THAT IS A LIE
imgflip.com

Soft Margin

- SVM é o melhor algoritmo possível se conseguirmos achar o kernel certo
- Infelizmente é custoso demais testar todos os kernels possíveis (infinitos), ou até testar o suficiente para estarmos próximos o suficiente
- Não vamos mais supor que conseguimos dividir “linearmente” os dados
- Introduzimos um custo adicional para exemplo que erramos

Queremos:

Aceitar divisões não tão rígidas - *soft margin*

Soft Margin

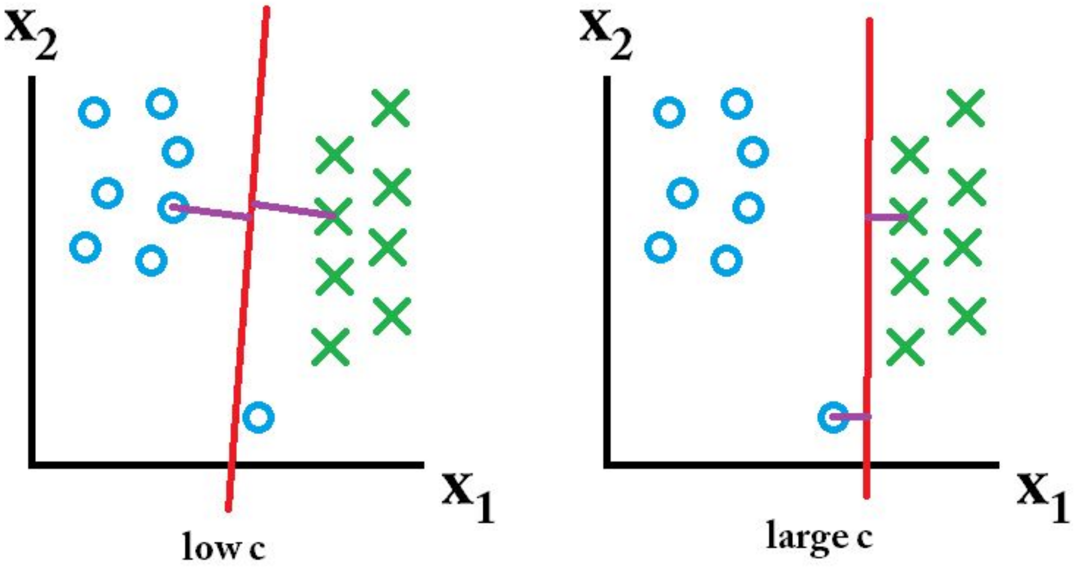
- "**Ei**" é o erro de cada exemplo
- "**C**" é um hiperparâmetro

$$min(\frac{1}{2}\left\|W^2\right\| + C * \sum E_i)$$

Influência do C

- Quanto menor o C, menor o "custo" de classificar errado um exemplo
  - Logo, mais exemplos o seu modelo pode errar → Gera um modelo mais simples
  - Maior viés, baixa variância - "underfitting"
  - Margem grande
- Ao contrário, se o C é grande, o custo de errar um exemplo é altíssimo
  - Logo vamos aprender um modelo mais complexo - "overfitting"
  - O viés é pequeno, mas estamos suscetíveis a variância dos dados
  - Margem pequena

Influência do C



Fonte

Influência do C

- Se C = infinito, temos o problema “hard margin” original
- Mesmo com a adição dessa regularização, ainda existe um mínimo único que os multiplicadores de lagrange são capazes de encontrar!

Tópicos extras

Gradient Descent

- Podemos ao invés de utilizar multiplicadores de lagrange, utilizar gradient descent para encontrar o **W**
- Precisamos mapear as restrições de igualdade da SVM
- Nessa formulação, é bem parecido com a regressão logística, apenas trocando a Loss function
- Mais informação: curso do Andrew NG

Extensão para regressão

- Encontrar uma função F(x) que no máximo está **ε** de distância do valor esperado
- Ao mesmo tempo, queremos F(x) o mais “flat” possível

Extensão para regressão

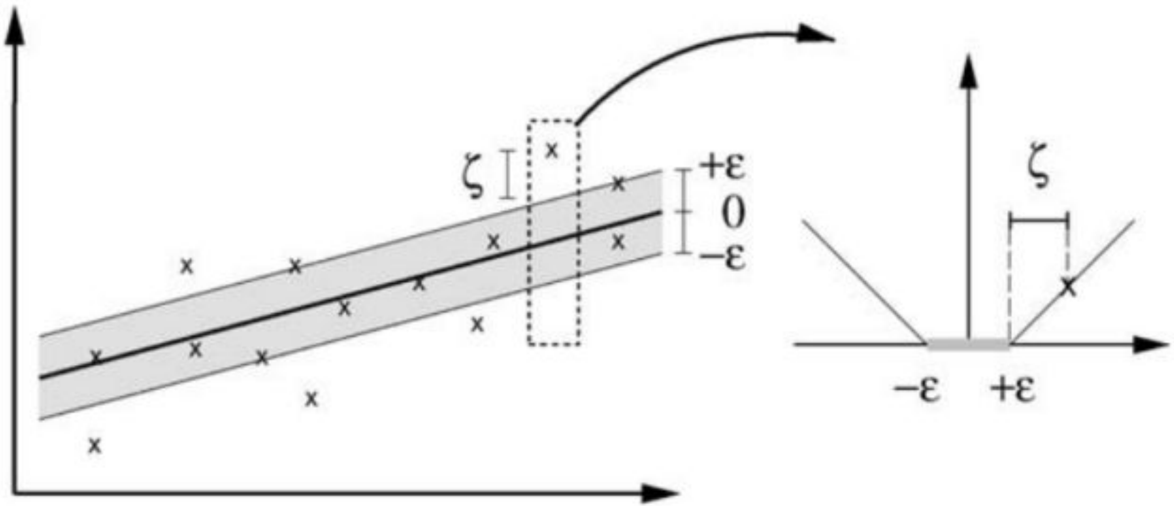
- No caso linear:

$$f(x) = \langle w, x \rangle + b \text{ with } w \in \mathcal{X}, b \in \mathbb{R}$$

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|w\|^2 \\ \text{subject to} & \begin{cases} y_i - \langle w, x_i \rangle - b \leq \varepsilon \\ \langle w, x_i \rangle + b - y_i \leq \varepsilon \end{cases} \end{array}$$

Fonte

Extensão para regressão



Fonte

Extensão para regressão

- Novamente incluímos um parâmetro de regularização que nos permita errar:

$$
\begin{aligned}
&\text{minimize} && \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*) \\
&\text{subject to} && \begin{cases} y_i - \langle w, x_i \rangle - b \le \varepsilon + \xi_i \\ \langle w, x_i \rangle + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \qquad\qquad \ge 0 \end{cases}
\end{aligned}
$$

$$
|\xi|_\varepsilon := \begin{cases} 0 & \text{if } |\xi| \le \varepsilon \\ |\xi| - \varepsilon & \text{otherwise.} \end{cases}
$$

Fonte

Extensão para regressão

- Novamente incluímos a questões dos kernels, etc….
- Quem quiser aprender mais:
  - Paper resumido legal
  - Papers originais para do Vapnik

Fonte

Discussões

Vantagens de SVM

- Tente responder antes de mudar o slide

Vantagens de SVM

- Uma vez treinado, sua predição é extremamente rápida e usa pouca memória.
- Como só é afetado por pontos próximo às margens, funciona bem mesmo para dados com alta dimensionalidade.
- Parâmetro de regularização por padrão → evita overfitting
- Importância do “truque” do Kernel → utilizado em diversas outras áreas
- Garantias estatísticas sobre ser o melhor modelo

Desvantagens de SVM

- Tente responder antes de mudar o slide

Desvantagens de SVM

- Tempo - Complexidade $\sim N^2$ ou N^3 dependendo da implementação
- Encontrar parâmetros ideais - tem que testar valores diferentes
- Não nos fornece probabilidades - existem truques para lidar com isso
- Interpretabilidade - Após aplicar kernels, fica difícil interpretar
- Multiclasse - Na prática, utilizamos a abordagem “One vs All”

Na prática - dados

- Escala das features faz diferença, devido principalmente a questões dos kernels
    - **Temos que fazer Feature scaling**!
- Complexidade entre N^2 e N^3 - na ordem de centenas de milhares começa a ficar tenso

Um guia prático de SVM - Como usar?

- Como sempre, SK-Learn tem quase tudo
- Se N << M, isto é, a quantidade de features é muito maior que a de exemplos:
  - Kernel Linear!
- Se N é grande:
  - Kernel Linear
- Caso contrário:
  - Kernel RBG/Gaussiano!
    - Fazer um grid search em C e gamma

A Practical Guide to Support Vector Classification

Observações avançadas e material extra

SVM Multiclasse

- Formulação de Wetson e Watkins - SVm for Multi-Class Pattern Recognition
- Formulação Stuctured SVMs
- Ambos criticados ou com maior tempo de execução
  - Rikin et al. 2004 in In Defense of One-Vs-All Classification
- NA PRÁTICA:
  - One vs Rest
  - One vs All

A Practical Guide to Support Vector Classification

Observações avançadas

- Material LINDO do CS 229 pelo Andrew NG
- O mesmo vídeo mencionado durante a aula.
- Existem mais maneiras de tratar o multiclasse que não abordamos
- Existe o NU-SVM - Nos garante uma quantidade máxima de erros no treino
- Existem provas de otimalidade de SVM
- SVM Transdutiva - utilizamos treino e teste juntos - muito louco
- SVM para One-Class - quando só temos exemplo de 1 classe - muito louco
- SV Clustering - aprendizado não supervisionado