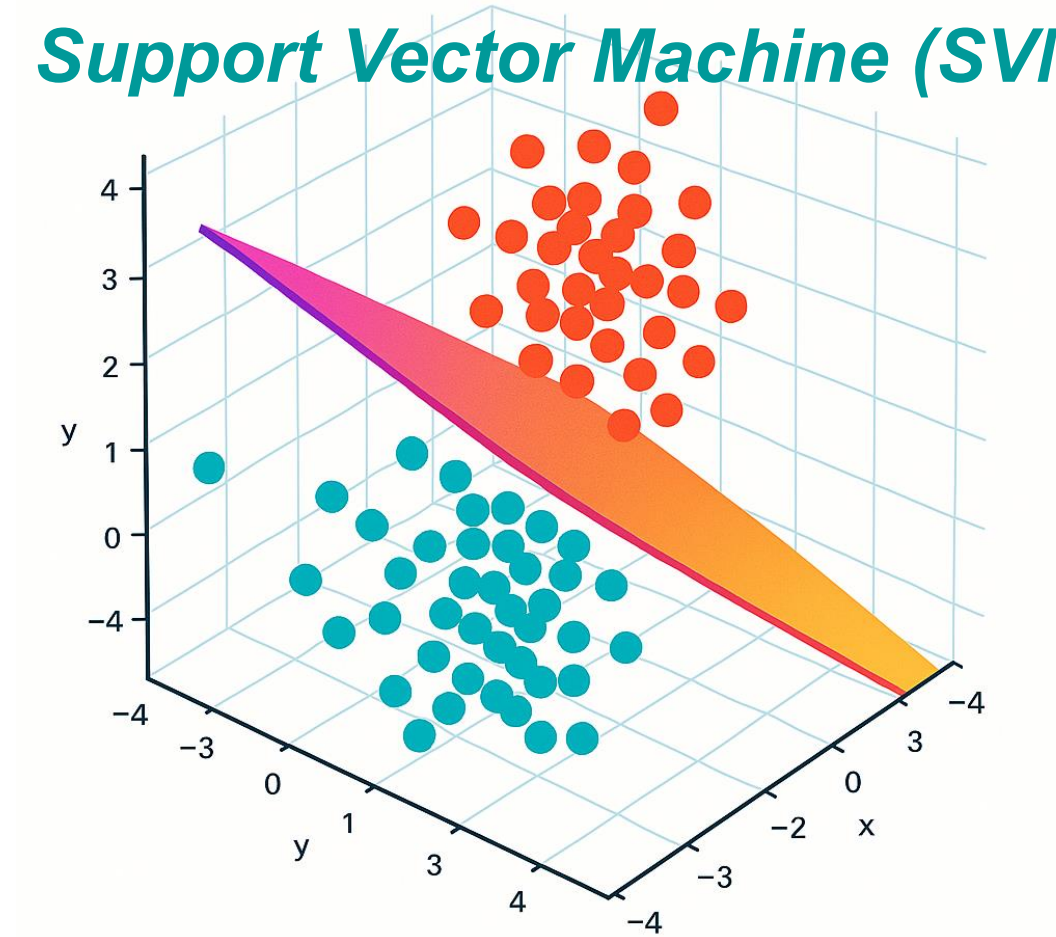


Introdução ao Aprendizado de Máquina:

Máquina de Vetores de Suporte (MVS)

Support Vector Machine (SVM)



Thales A. P. West, Ph.D.

Esclarecimentos...

Support Vector Machine (SVM) é um termo genérico que pode se referir tanto à classificação quanto à regressão:

- ❑ Quando o SVM é aplicado a problemas de classificação, ele é chamado de ***Support Vector Classifier (SVC)*** ou Classificador de Vetores de Suporte
- ❑ Quando o SVM é aplicado a problemas de regressão, ele é chamado de ***Support Vector Regression (SVR)*** ou Regressão de Vetores de Suporte

Support Vector Machine (SVM)

1. Support Vector Classifier (SVC)

Classificador de Vetores de Suporte

2. Support Vector Regression (SVR)

Regressão de Vetores de Suporte

Support Vector Machine (SVM)

1. Support Vector Classifier (SVC)

Classificador de Vetores de Suporte

2. Support Vector Regression (SVR)

Regressão de Vetores de Suporte

História...

Support-Vector Networks

CORINNA CORTES
VLADIMIR VAPNIK
AT&T Bell Labs., Holmdel, NJ 07733, USA

Editor: Lorenza Saitta

Abstract. The *support-vector network* is a new learning machine for two-group classification problems. The machine conceptually implements the following idea: input vectors are non-linearly mapped to a very high-dimension feature space. In this feature space a linear decision surface is constructed. Special properties of the decision surface ensures high generalization ability of the learning machine. The idea behind the support-vector network was previously implemented for the restricted case where the training data can be separated without errors. We here extend this result to non-separable training data.

High generalization ability of support-vector networks utilizing polynomial input transformations is demonstrated. We also compare the performance of the support-vector network to various classical learning algorithms that all took part in a benchmark study of Optical Character Recognition.

Keywords: pattern recognition, efficient learning algorithms, neural networks, radial basis function classifiers, polynomial classifiers.

Machine Learning, 20, 273–297 (1995)
© 1995 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.



Corinna Cortes

Google Research, NY
Verified email at google.com - [Homepage](#)
[Machine Learning](#) [Datamining](#)

FOLLOW

TITLE

CITED BY

YEAR

[Support-vector networks](#)

C Cortes, V Vapnik
Machine learning 20, 273-297

74157

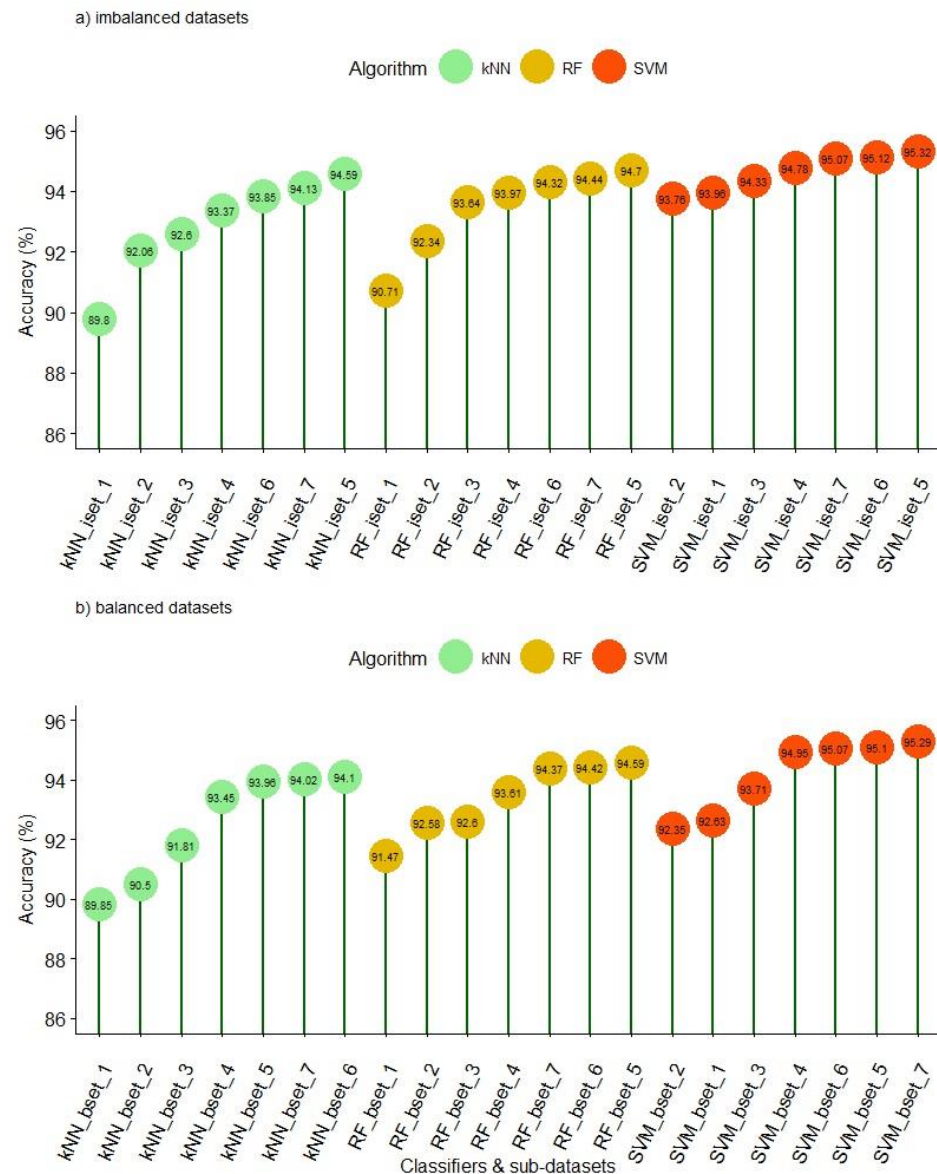
1995

Comparação de métodos...

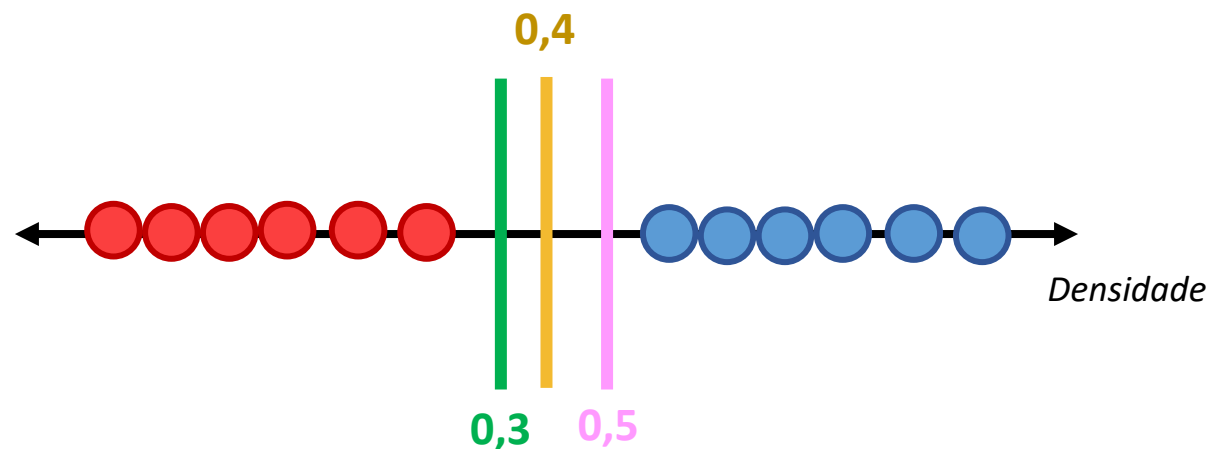
Thanh Hoi & Kappas (2017) compararam o desempenho de Random Forests, k-Nearest Neighbours e Support Vector Machines na classificação de imagens do Sentinel-2

No geral, o SVM apresentou o melhor desempenho, mas as diferenças não são extremas...

<https://www.mdpi.com/1424-8220/18/1/18>



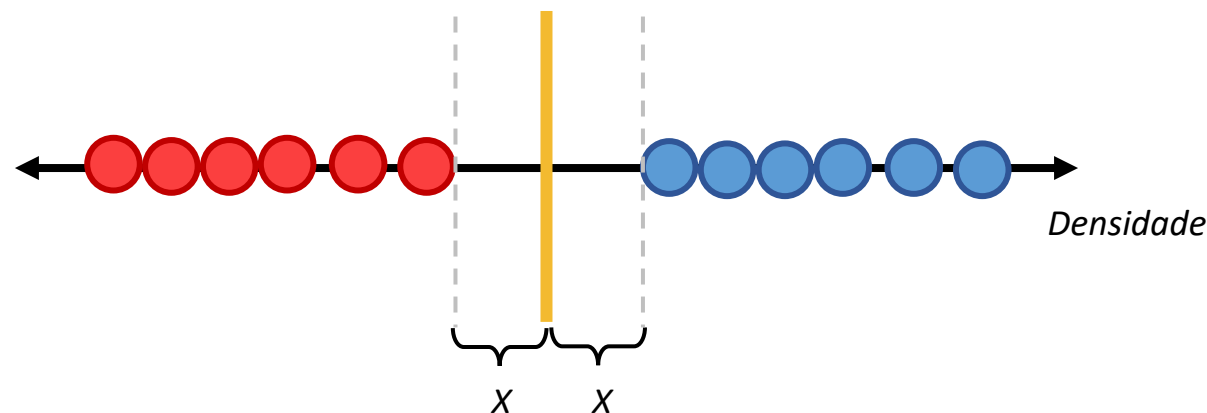
Classificação binária (em 2 grupos)



*Podemos traçar uma **linha (limite)** entre as observações e usa-la para classificações futuras...*

Onde traçar a linha (=estabelecer o limite) para a classificação?

Classificação binária (em 2 grupos)



*Faz sentido traçar a **linha** no meio...*

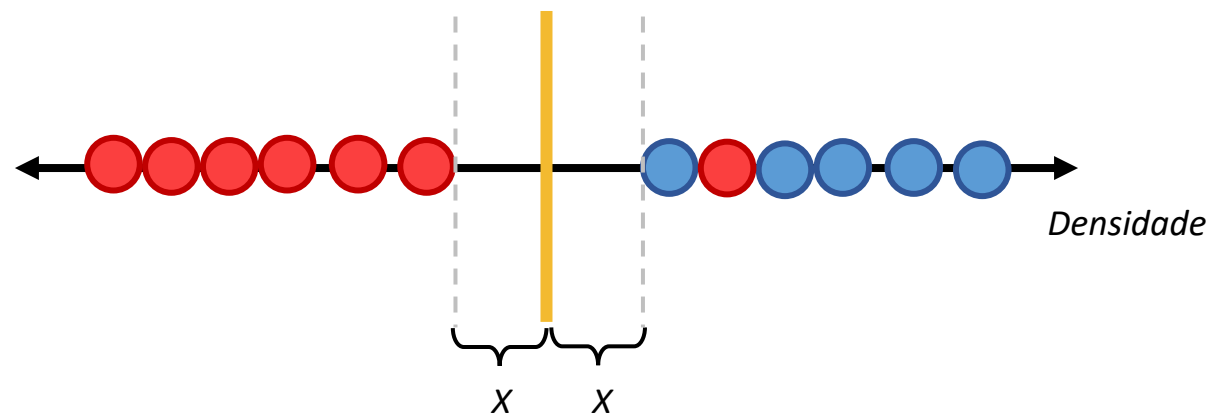
*Em outras palavras, selecionar o **limite** (“**threshold**”) que maximize as margens X entre as observações!*

Em SVM, isso é chamado de...

“Maximum Margin Classifier”

(ou “Hard Margin Classifier”)

Classificação binária (em 2 grupos)

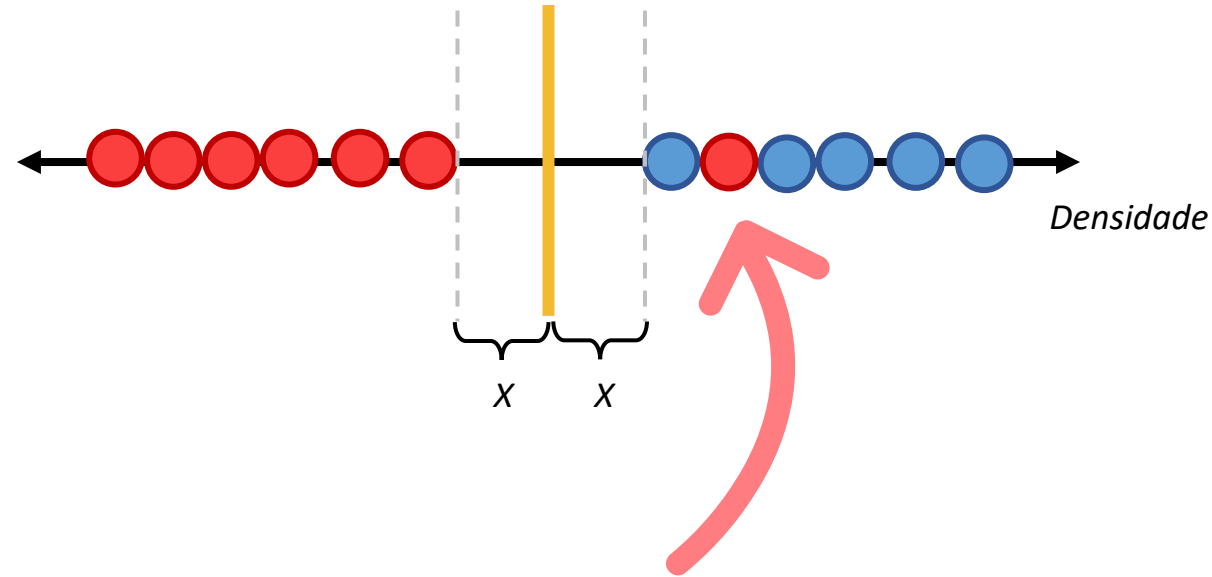


Mas e se os dados fossem assim...?

Maximum Margin Classifiers não conseguem lidar bem com valores discrepantes (“outliers”)...

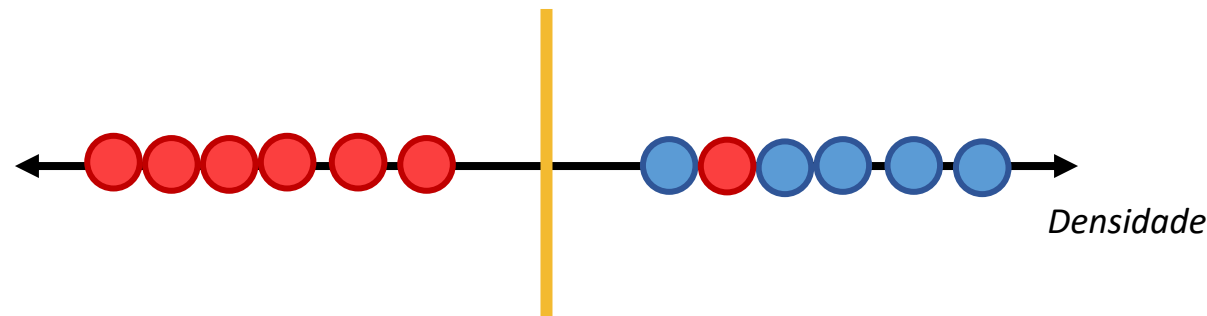
*Para selecionar um **limite** que consiga lidar com “outliers”, precisamos permitir algumas classificações erradas!*

Classificação binária (em 2 grupos)

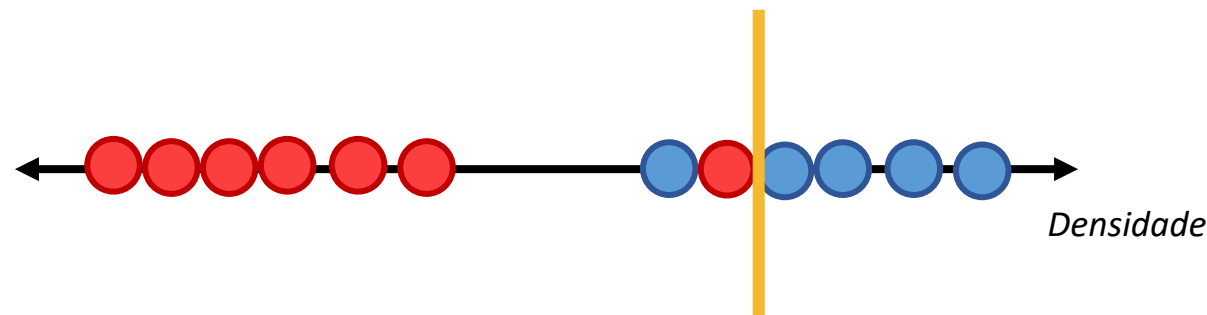


Nessa caso, temos uma classificação errada...

Classificação binária (em 2 grupos)

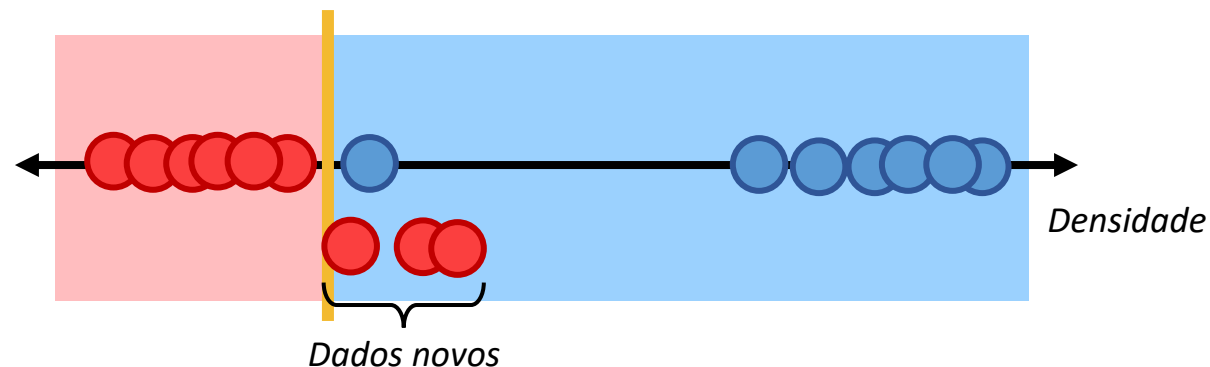


Mas ainda assim é melhor que qualquer outra opção, como:

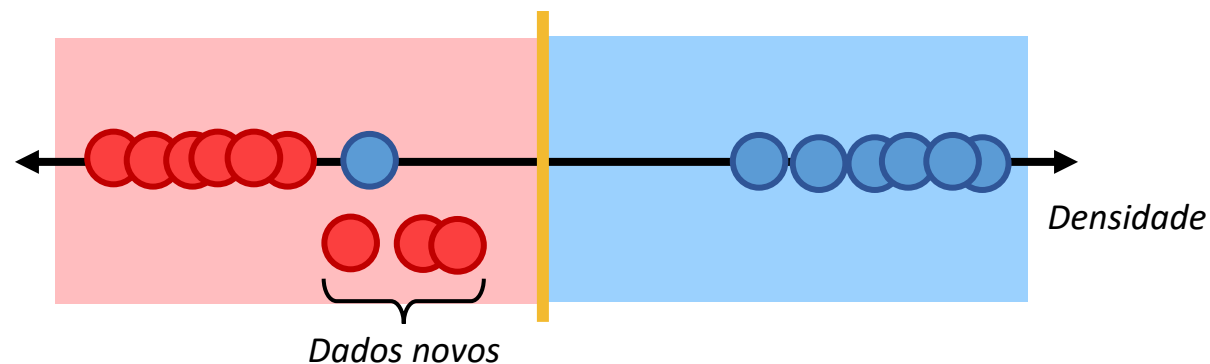


*Por sinal, a seleção do **limite** ideal é um exemplo do **trade-off de viés-variância**!*

Classificação binária (em 2 grupos)



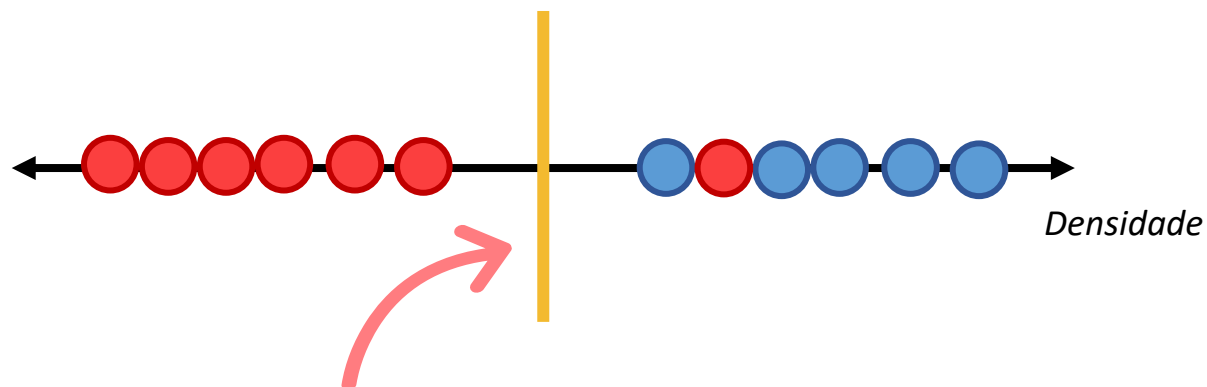
***Viés baixo, porém
variância alta...***



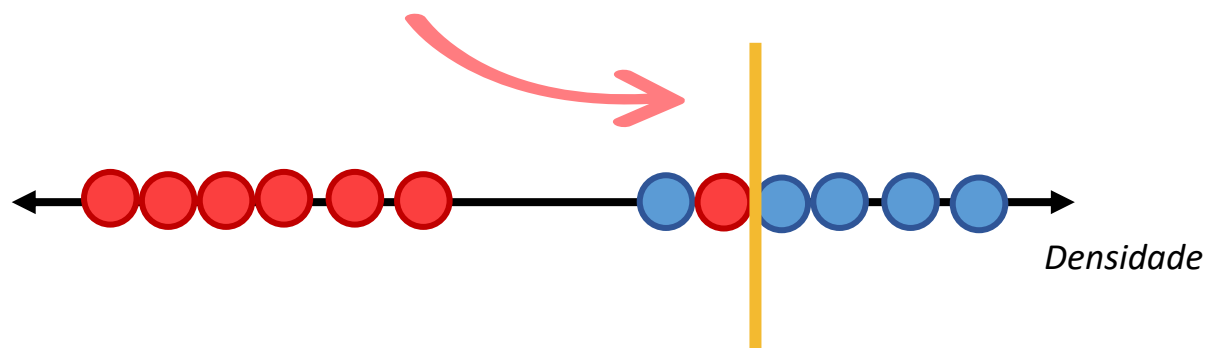
***Viés mais alto,
porém variância
bem mais baixa...***

*Por sinal, a seleção do **limite** ideal é um exemplo do **trade-off de viés-variância**!*

Classificação binária (em 2 grupos)



Mas como o **limite** é definido nesses casos?



Quando temos **limites** que incluem classificações erradas, usamos o termo **“Soft Margins”**

→ Temos então um **“Soft Margin Classifier”**

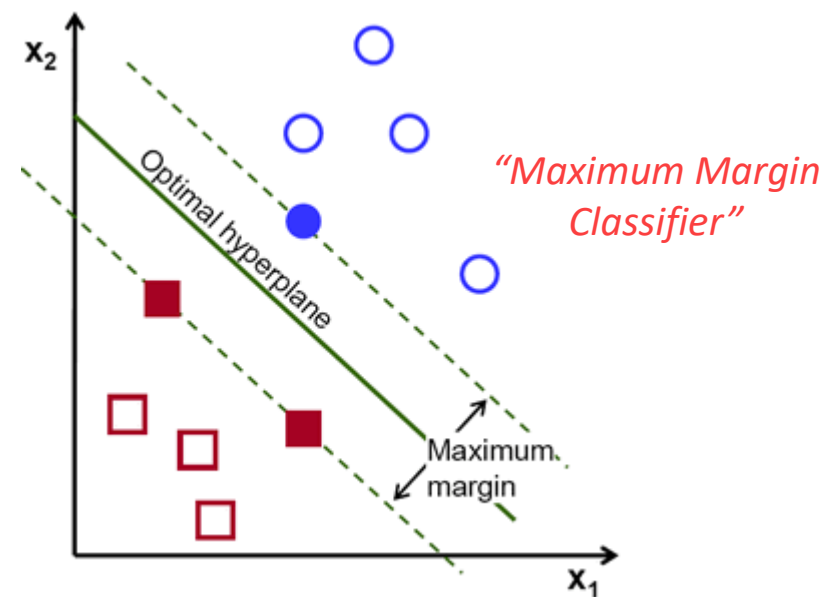
O **Soft Margin Classifier** introduz um parâmetro de penalidade que controla o equilíbrio entre maximizar a margem e minimizar os erros de classificação

→ **Validação cruzada (“cross-validation”)**

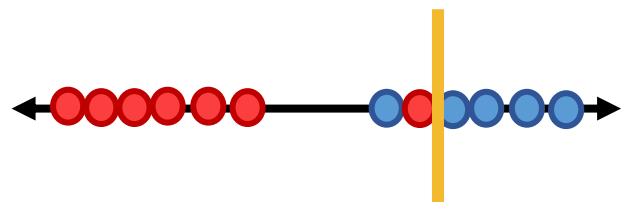
Vetores de Suporte (Support Vectors)

Os **vetores de suporte** são os pontos de dados que estão mais próximos do **hiperplano** de decisão (o que até então estavamos chamando de **limite**)

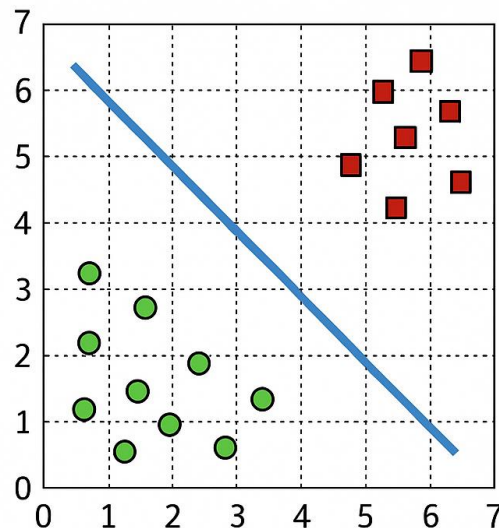
O objetivo é encontrar o **hiperplano** que maximiza a margem, que é a distância entre o **hiperplano** e os **vetores de suporte**



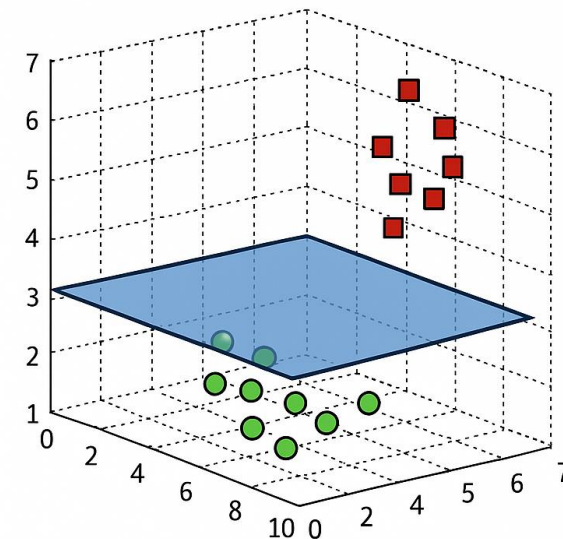
Hiperplanos (Hyperplanes)



A hyperplane in R^2 is



A hyperplane in R^3 is a plane



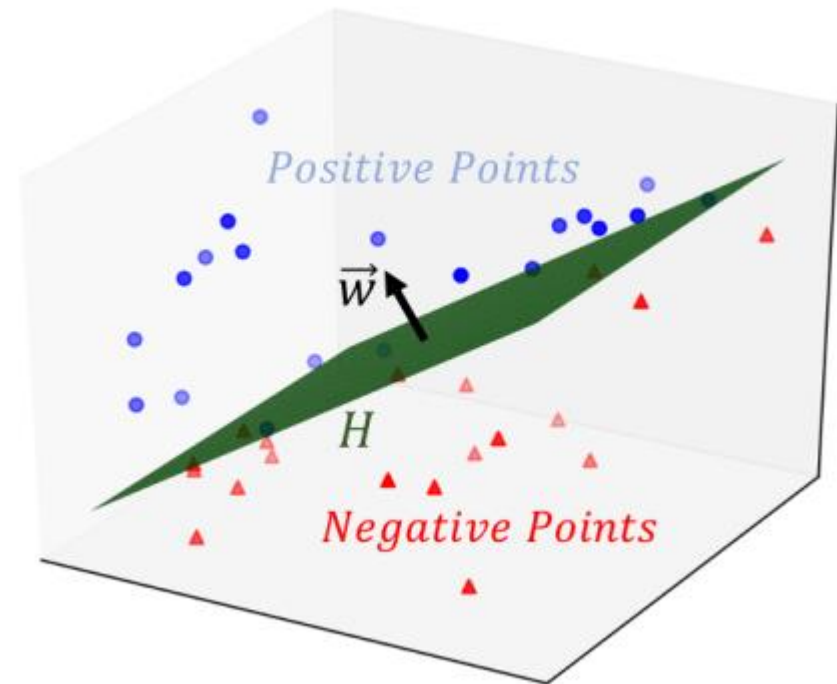
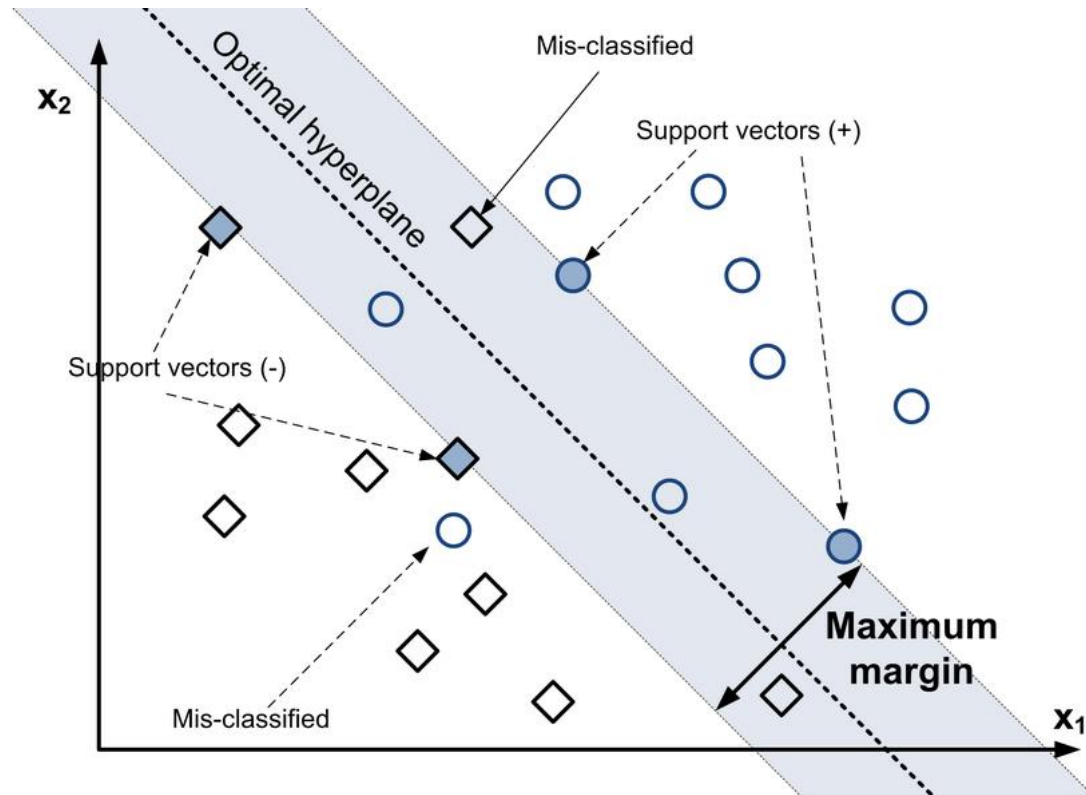
*Tecnicamente, alguns diriam que temos um **hiperplano** apenas em 4+ dimensões... → “flat affine subspace”*

*Em 1 dimensão (eixo), o **hiperplano** é apenas 1 ponto (1D) → “flat affine 0-Dimensional subspace”*

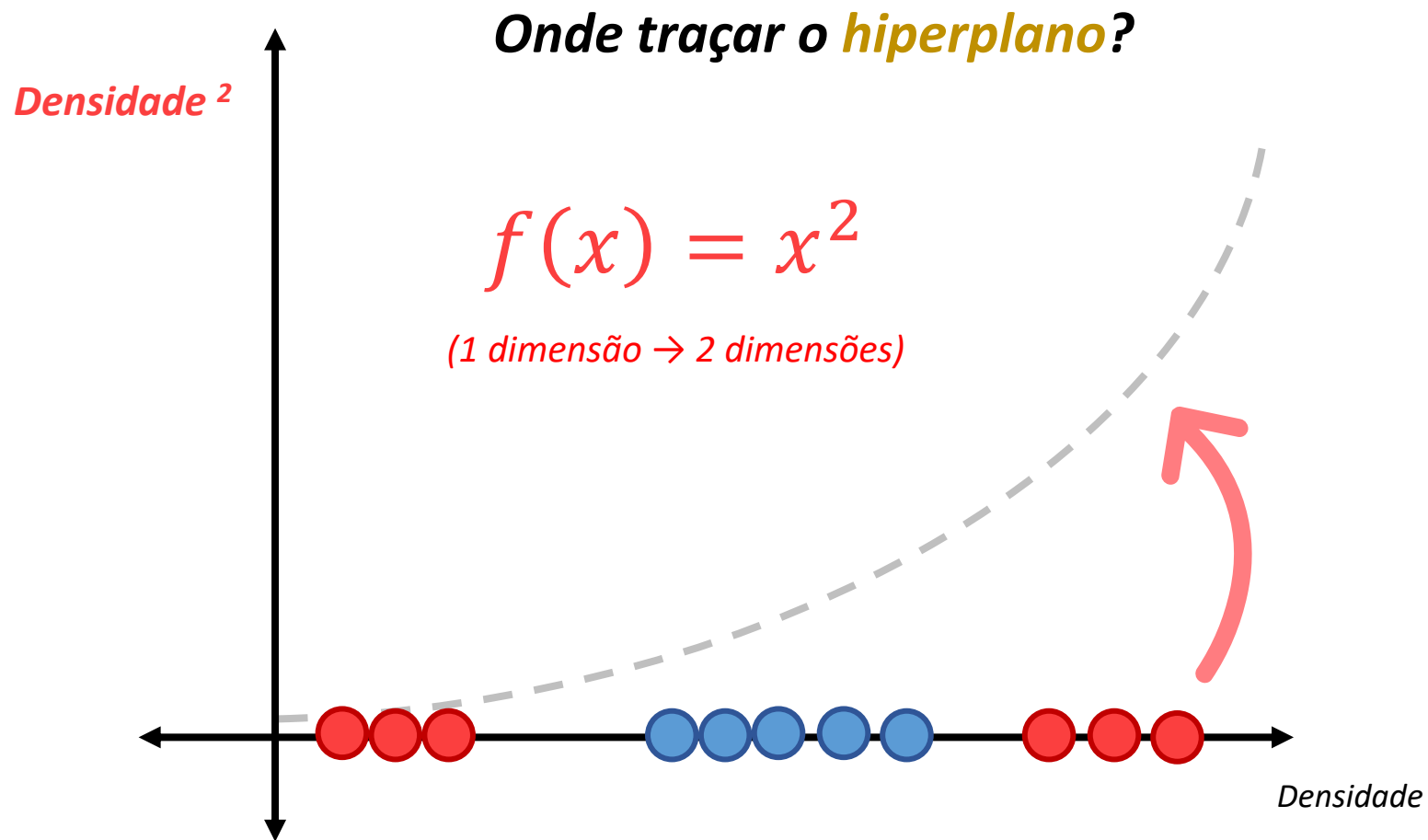
*Em 2 dimensões (eixos), o **hiperplano** é 1 linha (2D) → “flat affine 1-Dimensional subspace”*

*Em 3 dimensões (eixos), o **hiperplano** é 1 plano (3D) → “flat affine 2-Dimensional subspace”*

Hiperplanos (Hyperplanes)



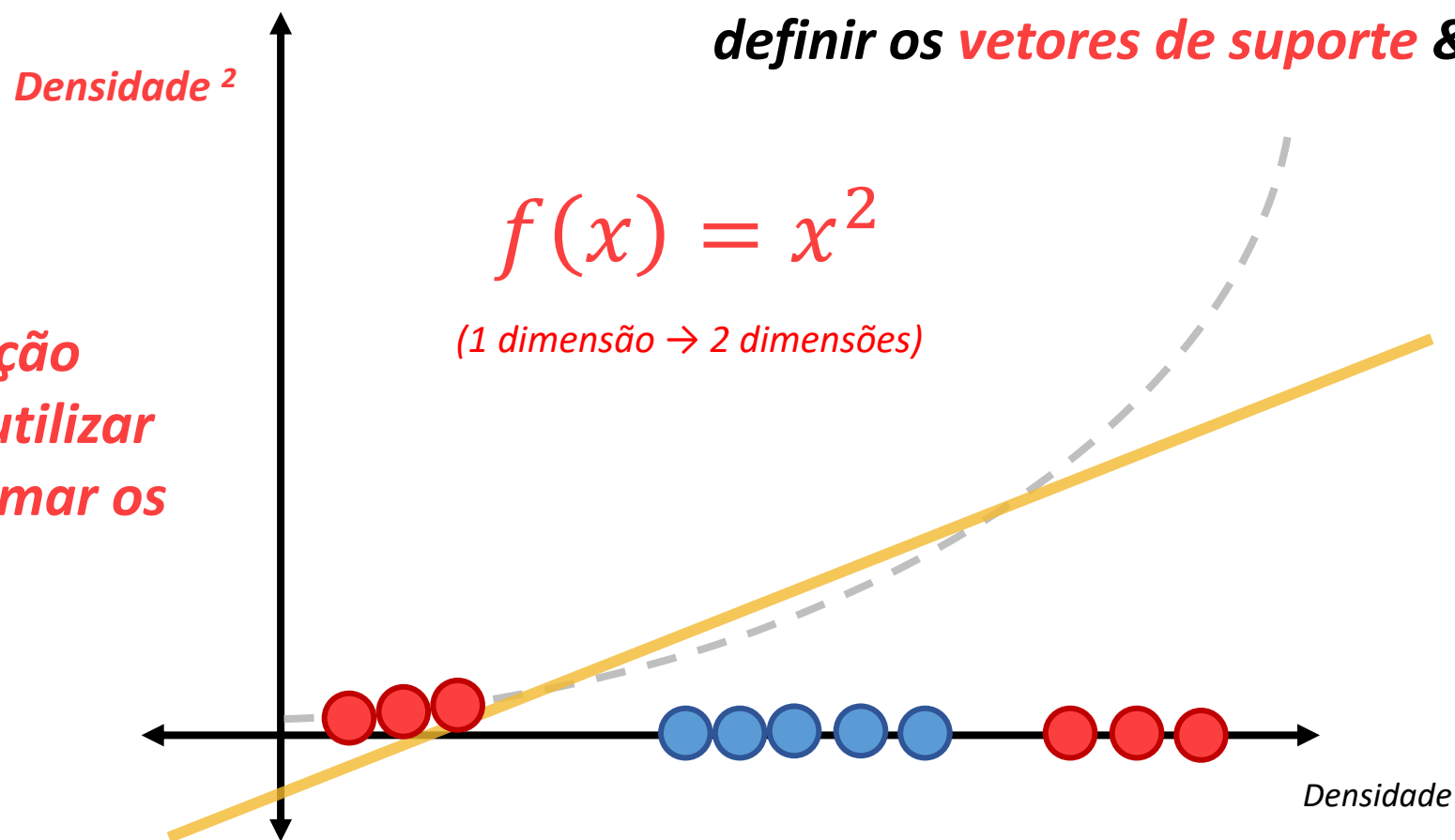
Support Vector Machine (SVM)



Support Vector Machine (SVM)

*SVMs usam funções matemáticas para mapear dados em dimensões mais altas para definir os **vetores de suporte** & o **hiperplano***

Mas qual função matemática utilizar para transformar os dados?!



Funções Kernel

SVMs usam funções chamadas kernel !!!

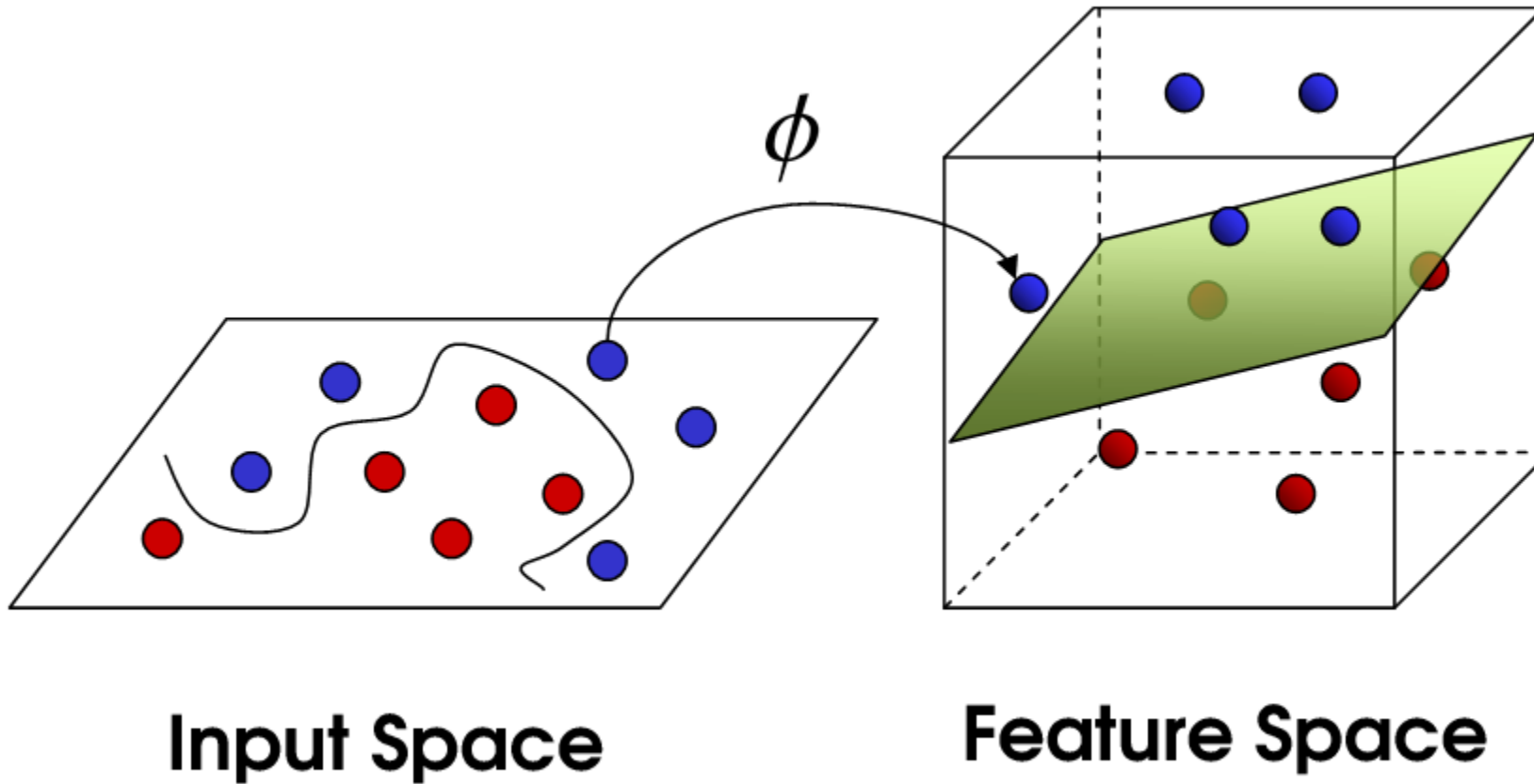
Por exemplo, $f(x) = x^2$ é “equivalente” a uma função kernel polinomial de segunda ordem

Em SVMs, uma função kernel calcula o produto escalar entre dois vetores em um espaço de características de alta dimensão, sem explicitamente mapear/calcular as coordenadas dos pontos de dados nesses espaços

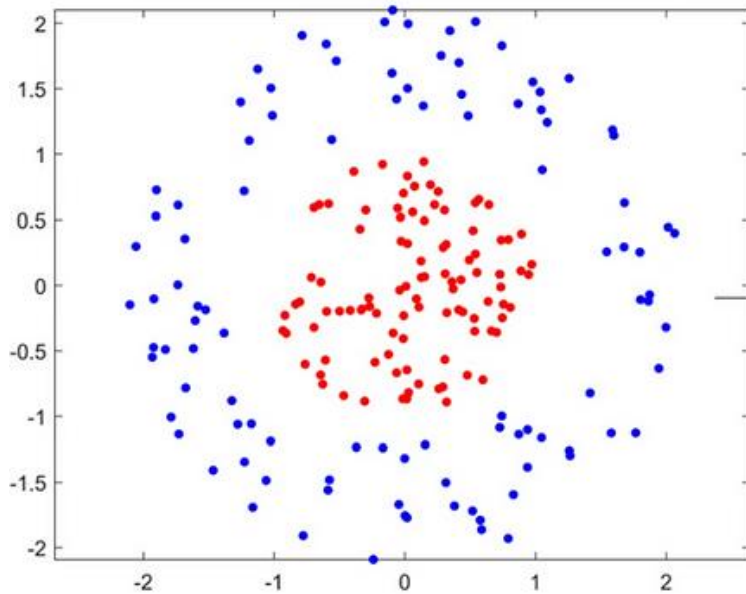
→ “Truque do Kernel” (“Kernel Trick”)

O algoritmo SVM só precisa dos valores do kernel (os produtos escalares no espaço de alta dimensão) para a classificação

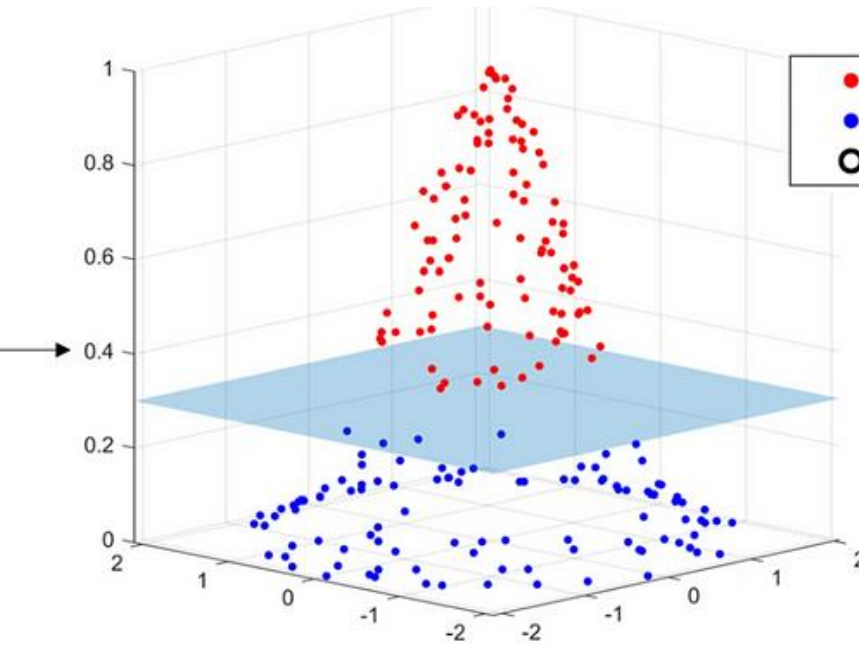
Funções Kernel: Exemplo



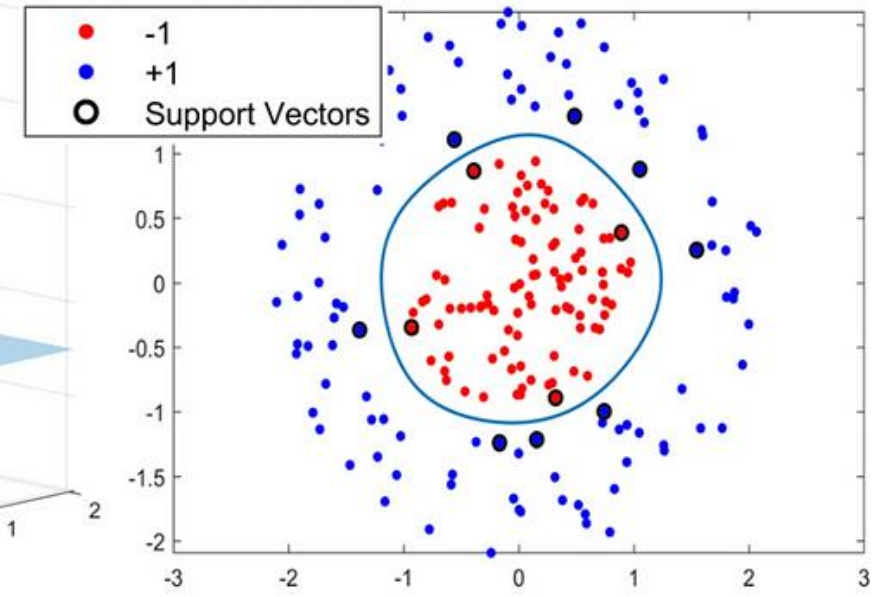
Funções Kernel: Exemplo



(a) 2D non-linear training data



(b) 3D mapped data points using the Gaussian kernel and separating hyperplane



(c) non-linear SVM with the Gaussian kernel training result

Produto Escalar

Amostra	Grupo	Densidade	Porosidade	Granularidade
1	A (+)	0.5	200	7
2	B (-)	0.4	150	4
3	A (+)	0.6	170	6

Vetores:

Amostra 1: (0.5, 200, 7)

Amostra 2: (0.4, 150, 4)

Amostra 3: (0.6, 170, 6)

Cálculos do Produto Escalar:

- ➔ **Produto escalar entre Amostra 1 e Amostra 2:** $(0.5 * 0.4) + (200 * 150) + (7 * 4) = 0.2 + 30000 + 28 = \mathbf{30028.2}$
- ➔ **Produto escalar entre Amostra 1 e Amostra 3:** $(0.5 * 0.6) + (200 * 170) + (7 * 6) = 0.3 + 34000 + 42 = \mathbf{34042.3}$
- ➔ **Produto escalar entre Amostra 2 e Amostra 3:** $(0.4 * 0.6) + (150 * 170) + (4 * 6) = 0.24 + 25500 + 24 = \mathbf{25524.24}$
- ➔ **Produto escalar entre Amostra 1 e Amostra 1:** $(0.5 * 0.5) + (200 * 200) + (7 * 7) = 0.25 + 40000 + 49 = \mathbf{40049.25}$
- ➔ **Produto escalar entre Amostra 2 e Amostra 2:** $(0.4 * 0.4) + (150 * 150) + (4 * 4) = 0.16 + 22500 + 16 = \mathbf{22516.16}$
- ➔ **Produto escalar entre Amostra 3 e Amostra 3:** $(0.6 * 0.6) + (170 * 170) + (6 * 6) = 0.36 + 28900 + 36 = \mathbf{28936.36}$

Os produtos escalares que calculamos são a base para o cálculo da *função kernel* ...

Kernel

- Produto escalar entre Amostra 1 e Amostra 2 = 30028.2
- Produto escalar entre Amostra 1 e Amostra 3 = 34042.3
- Produto escalar entre Amostra 2 e Amostra 3 = 25524.24
- Produto escalar entre Amostra 1 e Amostra 1 = 40049.25
- Produto escalar entre Amostra 2 e Amostra 2 = 22516.16
- Produto escalar entre Amostra 3 e Amostra 3 = 28936.36

$$f(x) = x^2$$

a e b são os vetores de entrada (amostras)

Produto escalar

c é uma constante,
geralmente 0 ou 1
(vamos usar $c = 1$)

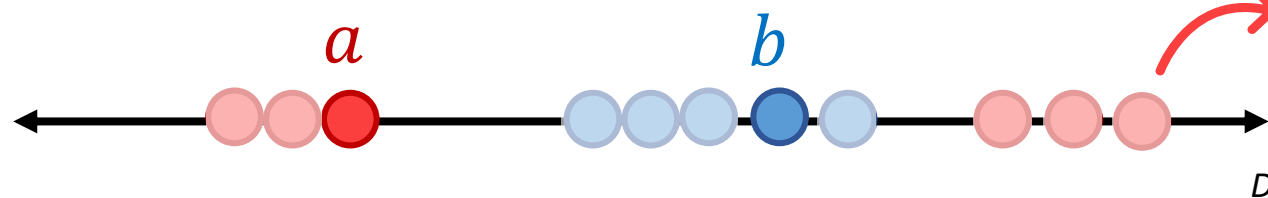
A **função kernel** polinomial de segunda ordem é: $K(a, b) = (a * b + c)^2$

Cálculos do **Kernel Polinomial**:

- Kernel entre Amostra 1 e Amostra 2: $K(A1, A2) = (30028.2 + 1)^2 = 901752885.64$
- Kernel entre Amostra 1 e Amostra 3: $K(A1, A3) = (34042.3 + 1)^2 = 1158946244.89$
- Kernel entre Amostra 2 e Amostra 3: $K(A2, A3) = (25524.24 + 1)^2 = 651540955.78$
- Kernel entre Amostra 1 e Amostra 1: $K(A1, A1) = (40049.25 + 1)^2 = 1604022006.25$
- Kernel entre Amostra 2 e Amostra 2: $K(A2, A2) = (22516.16 + 1)^2 = 506992529.57$
- Kernel entre Amostra 3 e Amostra 3: $K(A3, A3) = (28936.36 + 1)^2 = 837370335.7$

Esses resultados são equivalentes aos produtos escalares dos vetores mapeados no espaço de alta dimensão! Ou seja, não precisamos calcular explicitamente a transformação dos dados no espaço de alta dimensão → O “truque do kernel” permite que o SVM opere no espaço de alta dimensão apenas com os valores da **função kernel**

Exemplo anterior: Truque do Kernel



Apenas 1 variável, o seja o vetor de cada ponto tem apenas um valor contido nele

$$K(a, b) = (a * b + 1)^2$$

$$K(a, b) = (a * b + 1) * (a * b + 1)$$

$$K(a, b) = 2ab + a^2b^2 + 1$$

Essa função kernel polinomial é igual à esse produto escalar!

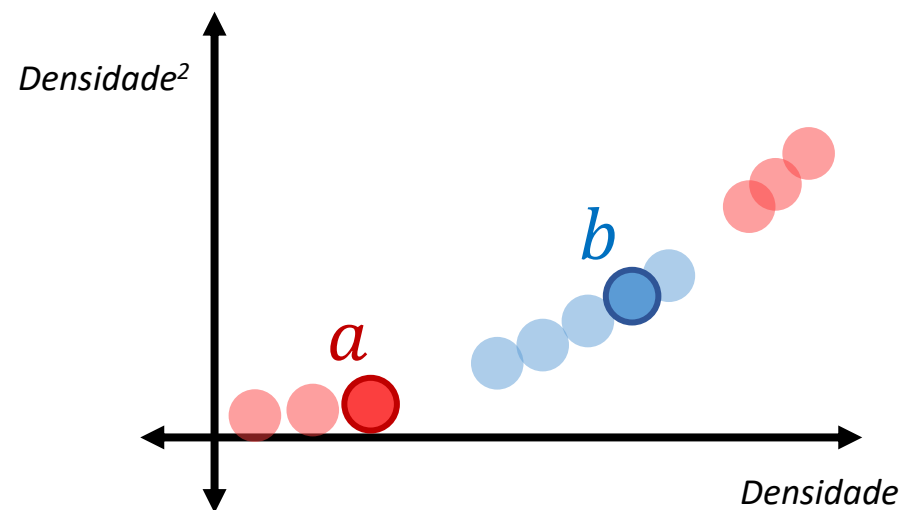
$$K(a, b) = (\sqrt{2}a, a^2, 1) * (\sqrt{2}b, b^2, 1)$$

Essas são as coordenadas dos dados na alta dimensão:

→ $\sqrt{2}a$ e $\sqrt{2}b$ são as coordenadas do eixo x (=Densidade)

→ a^2 e b^2 são as coordenadas do eixo y (=Densidade²)

→ 1 e 1 são as coordenadas do eixo z (=não temos...!)



Problema de otimização (com kernel não linear)

- Kernel entre Amostra 1 e Amostra 2: $K(A1, A2) = (30028.2 + 1)^2 = 901752885.64$
- Kernel entre Amostra 1 e Amostra 3: $K(A1, A3) = (34042.3 + 1)^2 = 1158946244.89$
- Kernel entre Amostra 2 e Amostra 3: $K(A2, A3) = (25524.24 + 1)^2 = 651540955.78$
- Kernel entre Amostra 1 e Amostra 1: $K(A1, A1) = (40049.25 + 1)^2 = 1604022006.25$
- Kernel entre Amostra 2 e Amostra 2: $K(A2, A2) = (22516.16 + 1)^2 = 506992529.57$
- Kernel entre Amostra 3 e Amostra 3: $K(A3, A3) = (28936.36 + 1)^2 = 837370335.7$

$$\underset{\alpha}{\text{maximize}} \quad \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j K(x_i, x_j) - \sum_{i=1}^N \alpha_i$$

Onde:

- α_i são multiplicadores Lagrange associados à amostra i
- y_i é o rótulo (+1 ou -1) da amostra i
- $K(x_i, x_j)$ é a função kernel entre x_i e x_j

A otimização envolve maximizar os **multiplicadores de Lagrange** associados aos vetores de suporte

Essa transformação permite resolver a otimização do SVM usando funções de kernel para classificação não linear

A formulação otimiza os multiplicadores de Lagrange α_i
→ Os vetores de suporte são aquelas amostras de treinamento onde $\alpha_i > 0$

Mais informações...

<https://www.geeksforgeeks.org/machine-learning/support-vector-machine-algorithm/>

The screenshot shows the top portion of a web browser displaying the GeeksforGeeks website. The address bar shows the URL: `geeksforgeeks.org/machine-learning/support-vector-machine-algorithm/`. The website's navigation bar includes links for 'Tutorials' and 'Courses', a search bar, and a list of categories: 'Data Science', 'Data Science Projects', 'Data Analysis', 'Data Visualization', 'Machine Learning', 'ML Projects', 'Deep Learning', 'NLP', and 'C'. The main content area features the article title 'Support Vector Machine (SVM) Algorithm' with a 'Last Updated' date of '28 May, 2025'. To the left, a 'Next Article' link points to 'Classifying data using Support Vector...'. The article text begins by defining SVM as a supervised machine learning algorithm for classification and regression, aiming to find a hyperplane that separates data classes. It mentions the goal of maximizing the margin between classes. At the bottom, a diagram titled 'Support Vectors & Hyperplane' is partially visible, showing a 'Hyperplane' and a 'Maximum Margin / Positive HyperPlane'.

← → ↻ 🌐 [geeksforgeeks.org/machine-learning/support-vector-machine-algorithm/](https://www.geeksforgeeks.org/machine-learning/support-vector-machine-algorithm/)

Tutorials ▾ Courses ▾  Search...

Data Science Data Science Projects Data Analysis Data Visualization Machine Learning ML Projects Deep Learning NLP C

Next Article:
[Classifying data using Support Vector...](#) →

Support Vector Machine (SVM) Algorithm

Last Updated : 28 May, 2025

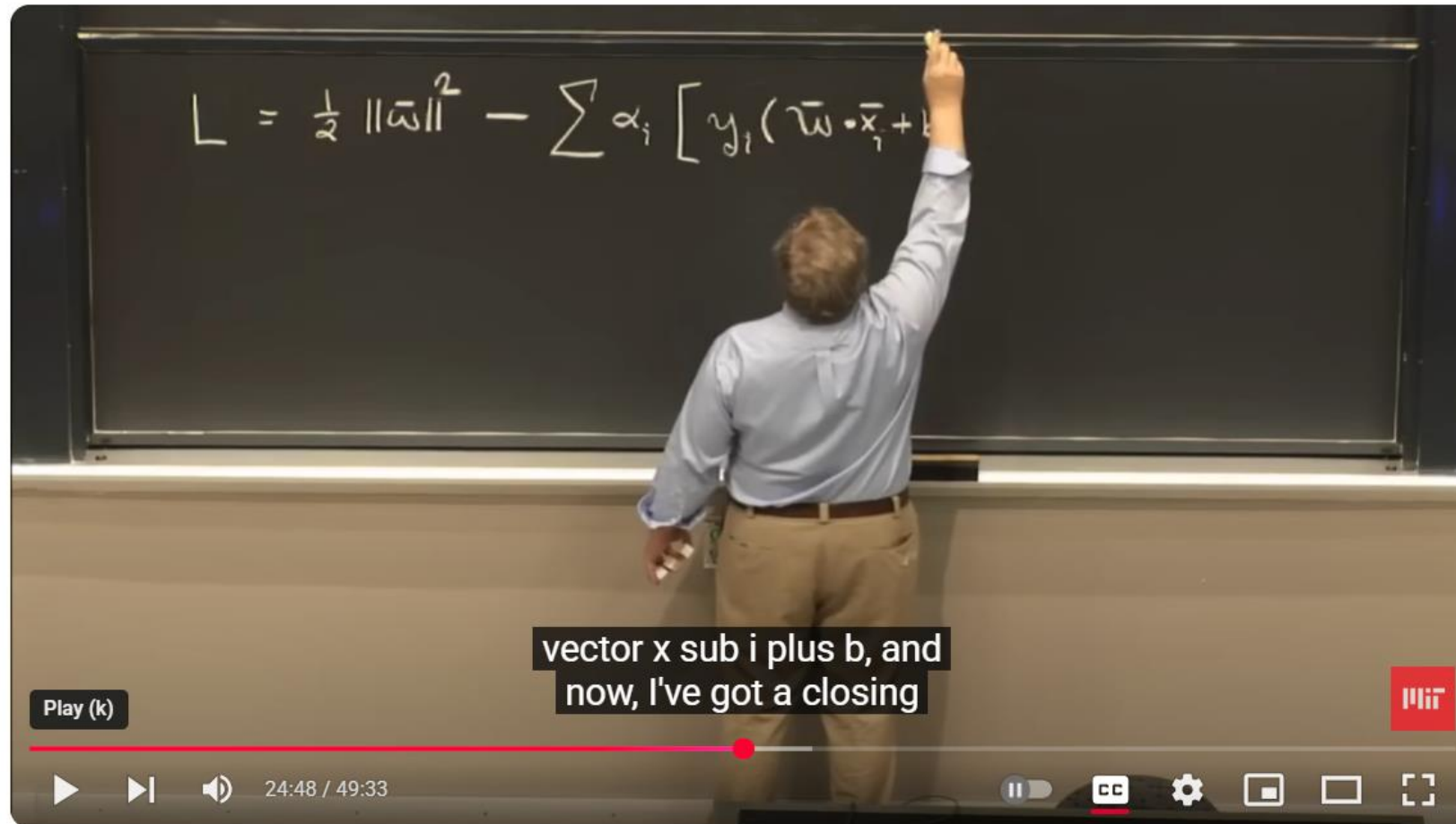
Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. It tries to find the best boundary known as hyperplane that separates different classes in the data. It is useful when you want to do binary classification like spam vs. not spam or cat vs. dog.

The main goal of SVM is to maximize the margin between the two classes. The larger the margin the better the model performs on new and unseen data.

Support Vectors & Hyperplane

Hyperplane: ↑ Maximum Margin / Positive HyperPlane

Mais informações...



16. Learning: Support Vector Machines



MIT OpenCourseWare
5.74M subscribers



Subscribed



25K



Share



Download



Algumas funções Kernel...

<https://www.geeksforgeeks.org/machine-learning/how-to-choose-the-best-kernel-function-for-svms/>

Nome	Equação
Kernel polinomial	$K(x, y) = (x * y + c)^d$
Kernel Gaussiana	$K(x, y) = \exp\left(\frac{\ x - y\ ^2}{2\sigma^2}\right)$
Função de Base Radial Gaussiana (RBF)	$K(x, y) = \exp(-\gamma\ x - y\ ^2)$
Laplace RBF kernel	$K(x, y) = \exp\left(\frac{\ x - y\ }{\sigma}\right)$
Kernel sigmoidal	$K(x, y) = \tanh(\alpha x^T * y + c)$

Não existe uma função kernel universalmente melhor, pois a melhor escolha depende das características dos seus dados → validação cruzada (“cross-validation”)

E quanto a classificações com mais de 2 grupos?

*SVM pode ser utilizado para classificações com mais de 2 classes utilizando a abordagem **One-Versus-One***

Como funciona essa abordagem em SVMs?

Criação de Classificadores Binários: Para um problema com K classes, a abordagem *One-Versus-One* constrói *um classificador binário separado para cada possível par de classes*. Por exemplo, se você tem 3 classes (A, B, C), serão treinados os seguintes classificadores binários:

- Um classificador para distinguir entre a classe A e a classe B
- Um classificador para distinguir entre a classe A e a classe C
- Um classificador para distinguir entre a classe B e a classe C

Em geral, para K classes, serão treinados $2K(K-1)$ classificadores binários

Treinamento dos Classificadores: Cada um desses classificadores binários é treinado apenas com os dados pertencentes às duas classes que ele deve distinguir. Os rótulos das outras classes são ignorados durante o treinamento desse classificador específico. No exemplo de 3 classes:

- O classificador A vs. B é treinado apenas com as amostras rotuladas como A e B
- O classificador A vs. C é treinado apenas com as amostras rotuladas como A e C
- O classificador B vs. C é treinado apenas com as amostras rotuladas como B e C

Classificação de Novas Amostras: Quando uma nova amostra precisa ser classificada, ela é apresentada a *todos* os classificadores binários treinados.

Votação: Cada classificador binário emite um "voto" para a classe que ele considera ser a mais provável para a nova amostra.

Determinação da Classe Final: A classe que receber o *maior número de votos* é atribuída à nova amostra. Em caso de empate, estratégias de desempate podem ser utilizadas (por exemplo, escolher aleatoriamente entre as classes com o maior número de votos).

Resumo: Support Vector Classifier (SVC)

- *SVC classifica dados em duas categorias (mas lembre-se do 1-vs.-1)*
- *SVC pode utilizar **funções kernel** para “mapear” dados em altas dimensões e conseguir separar dados linearmente inseparáveis*
- *Em prática, o **Truque do Kernel** permite pular a etapa de mapeamento de dados em altas dimensões...*
- *A classificação é definida pelo **hiperplano**, que por sua vez é definido pelos **vetores de suporte***

Support Vector Machine (SVM)

1. Support Vector Classifier (SVC)

Classificador de Vetores de Suporte

2. Support Vector Regression (SVR)

Regressão de Vetores de Suporte

História...

Support-Vector Networks

CORINNA CORTES
VLADIMIR VAPNIK
AT&T Bell Labs., Holmdel, NJ 07733, USA

Editor: Lorenza Saitta

Abstract. The *support-vector network* is a new learning machine for two-group classification problems. The machine conceptually implements the following idea: input vectors are non-linearly mapped to a very high-dimension feature space. In this feature space a linear decision surface is constructed. Special properties of the decision surface ensures high generalization ability of the learning machine. The idea behind the support-vector network was previously implemented for the restricted case where the training data can be separated without errors. We here extend this result to non-separable training data.

High generalization ability of support-vector networks utilizing polynomial input transformations is demonstrated. We also compare the performance of the support-vector network to various classical learning algorithms that all took part in a benchmark study of Optical Character Recognition.

Keywords: pattern recognition, efficient learning algorithms, neural networks, radial basis function classifiers, polynomial classifiers.

Machine Learning, 20, 273–297 (1995)
© 1995 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.



Corinna Cortes

Google Research, NY
Verified email at google.com - [Homepage](#)
[Machine Learning](#) [Datamining](#)

FOLLOW

TITLE

CITED BY

YEAR

[Support-vector networks](#)

C Cortes, V Vapnik
Machine learning 20, 273-297

74157

1995

História...



vapnik

Professor of Columbia, Fellow of [NEC Labs America](#),
Verified email at nec-labs.com

[machine learning](#) [statistics](#) [computer science](#)

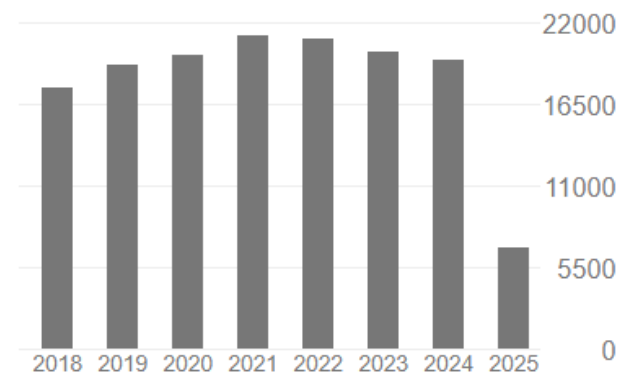
FOLLOW

TITLE	CITED BY	YEAR
The Nature of Statistical Learning Theory V Vapnik Data mining and knowledge discovery	110234 *	1995
Support-vector networks C Cortes, V Vapnik Machine learning 20, 273-297	74698	1995
Backpropagation applied to handwritten zip code recognition Y LeCun, B Boser, JS Denker, D Henderson, RE Howard, W Hubbard, ... Neural computation 1 (4), 541-551	18803	1989

Cited by

[VIEW ALL](#)

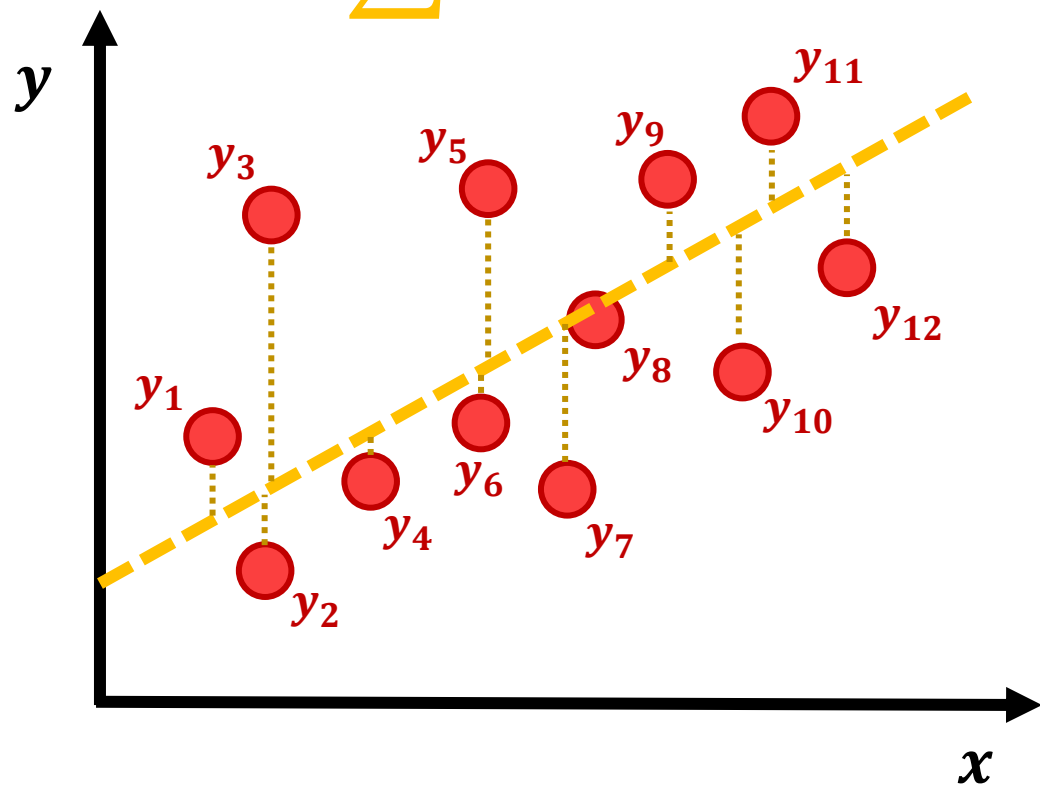
	All	Since 2020
Citations	326828	108294
h-index	106	73
i10-index	292	181



Support Vector Regression (SVR)

Mínimos quadrados

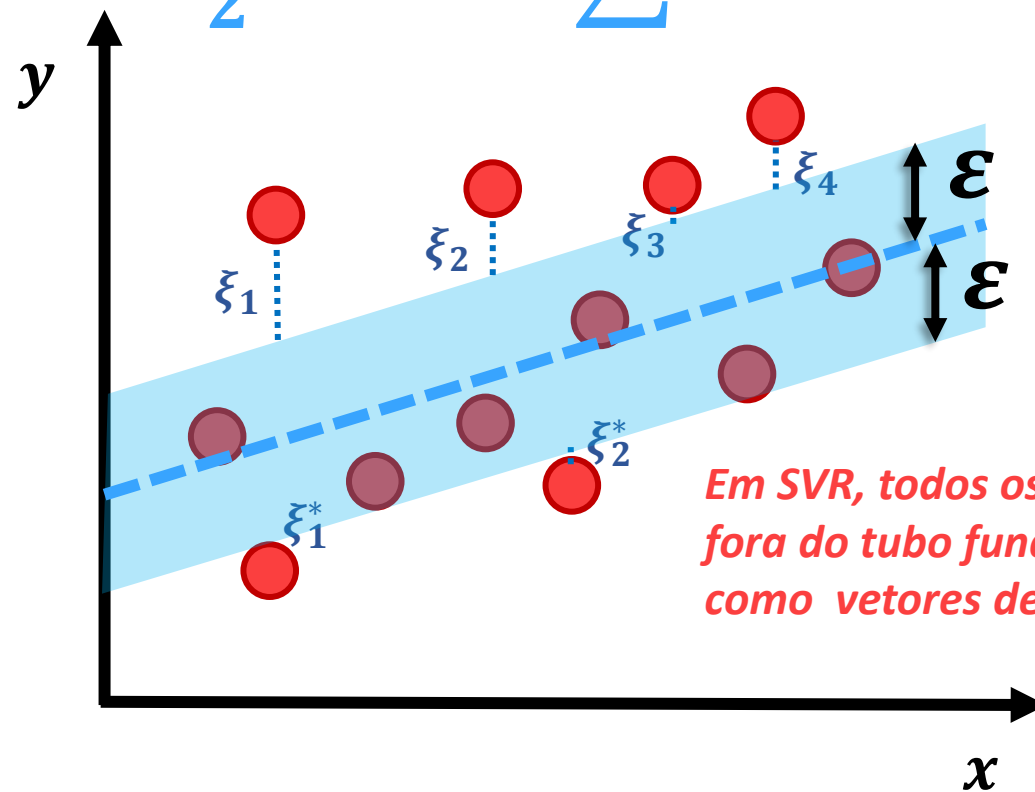
$$\sum (y_i - \hat{y}_i)^2$$



Tubo ε -insensível

(ε -insensitive tube; função custo)

$$\frac{1}{2} \|w\|^2 + c \sum (\xi_i - \xi_i^*)^2$$



*Por exemplo,
poderíamos ter
1000 pontos de
dados e apenas
100 vetores de
suporte*

*Em SVR, todos os pontos
fora do tubo funcionam
como vetores de suporte*

Support Vector Regression (SVR)

Por sinal, ξ_i e ξ_i^* são chamados *variáveis de folga* (“slak variables”)

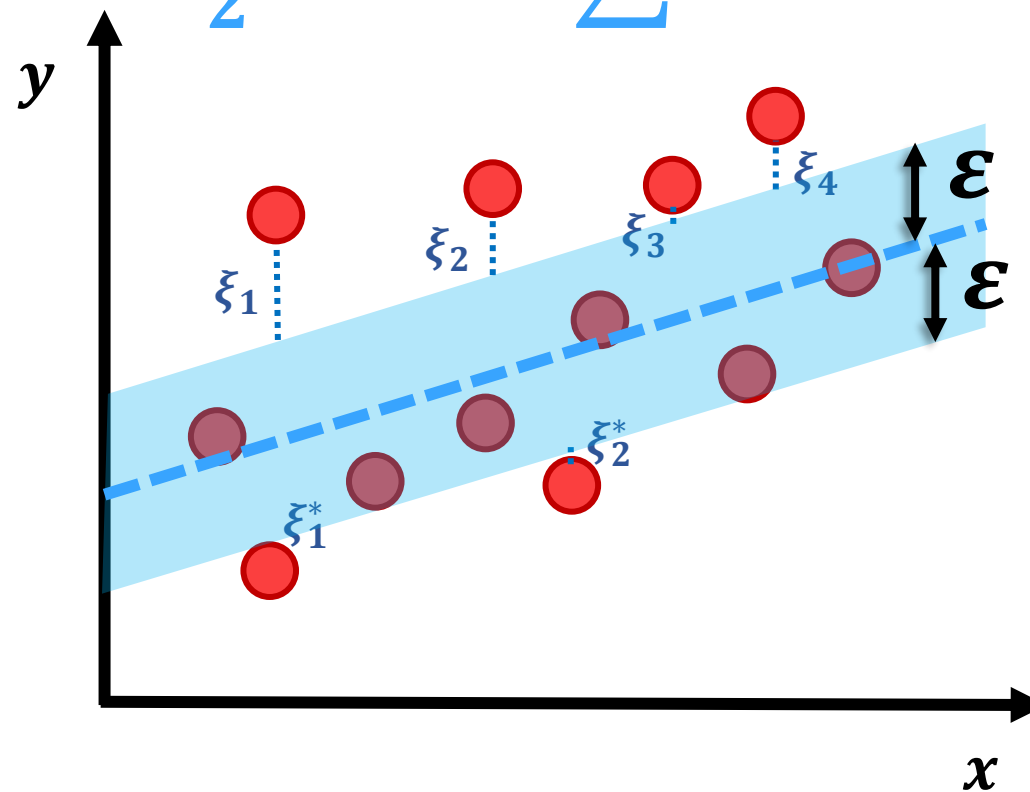
Em **SVR**, o objetivo é encontrar uma função que se ajuste aos dados, permitindo um certo nível de erro definido pela **margem ε** ou **tubo ε -insensível** (=nível de tolerância)

Ou seja, a **margem ε** é análoga à “**soft margin**” em um **SVC**

Tubo ε -insensível

(ε -insensitive tube; função custo)

$$\frac{1}{2} \|w\|^2 + c \sum (\xi_i - \xi_i^*)^2$$



Support Vector Regression (SVR)

Note que a inclinação do **hiperplano** é definida pelos pontos associados à ξ_i ou ξ_i^*

Se todos os **pontos** estiverem dentro do **tubo ε -insensível**, o **hiperplano** seria praticamente uma linha horizontal ($\approx$ “*intercept-only model*”)

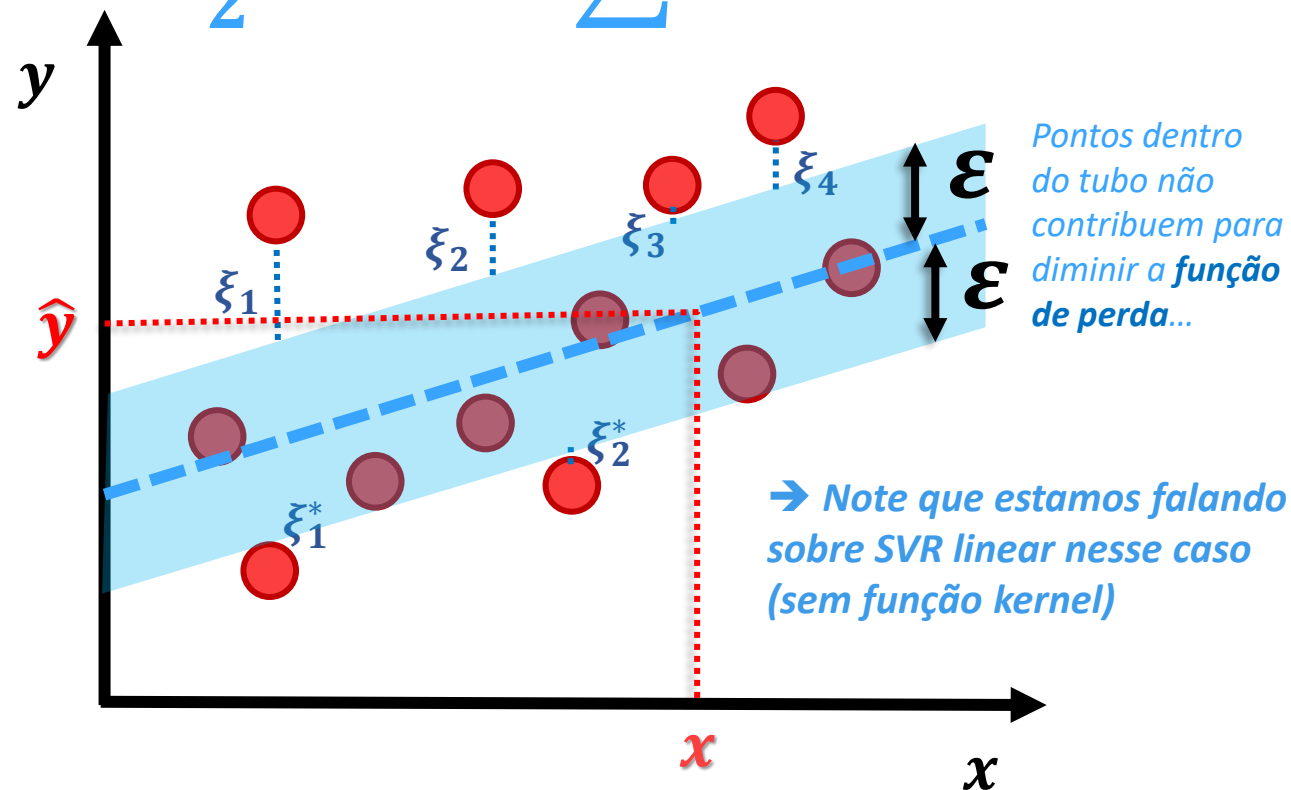
Ou seja, a predição do modelo seria aproximadamente a média de y (*constante*)

De qualquer forma, com uma **SVR** ajustada podemos plugar um valor de x e obter uma estimativa de y

Tubo ε -insensível

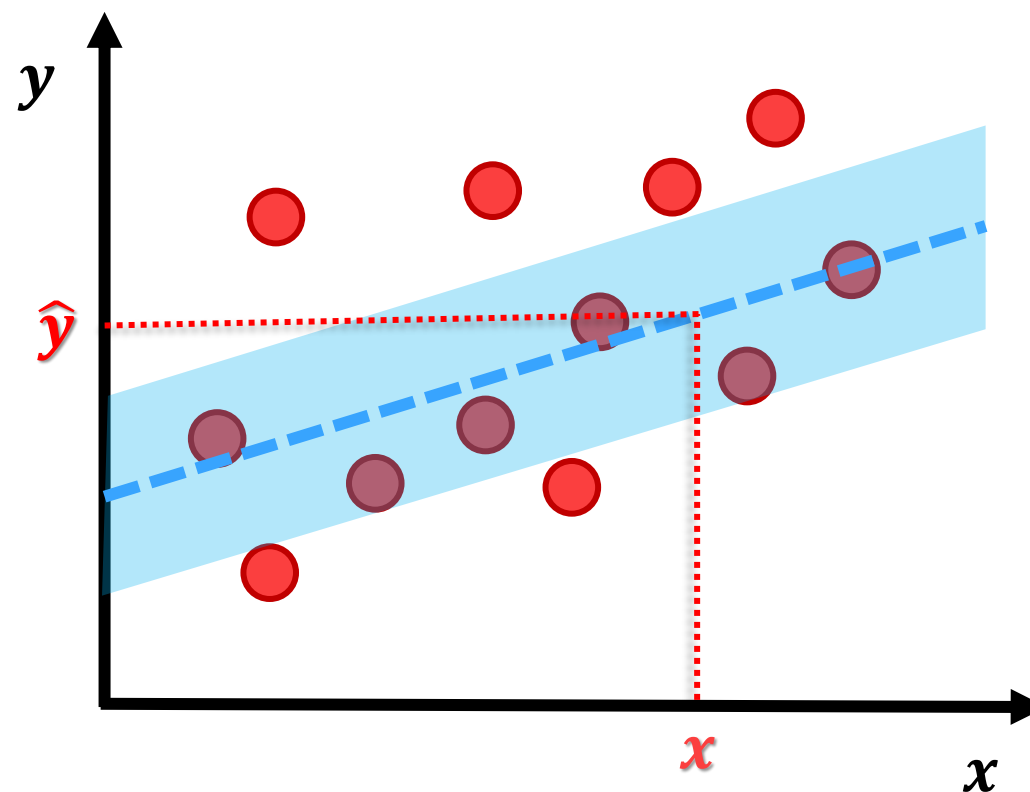
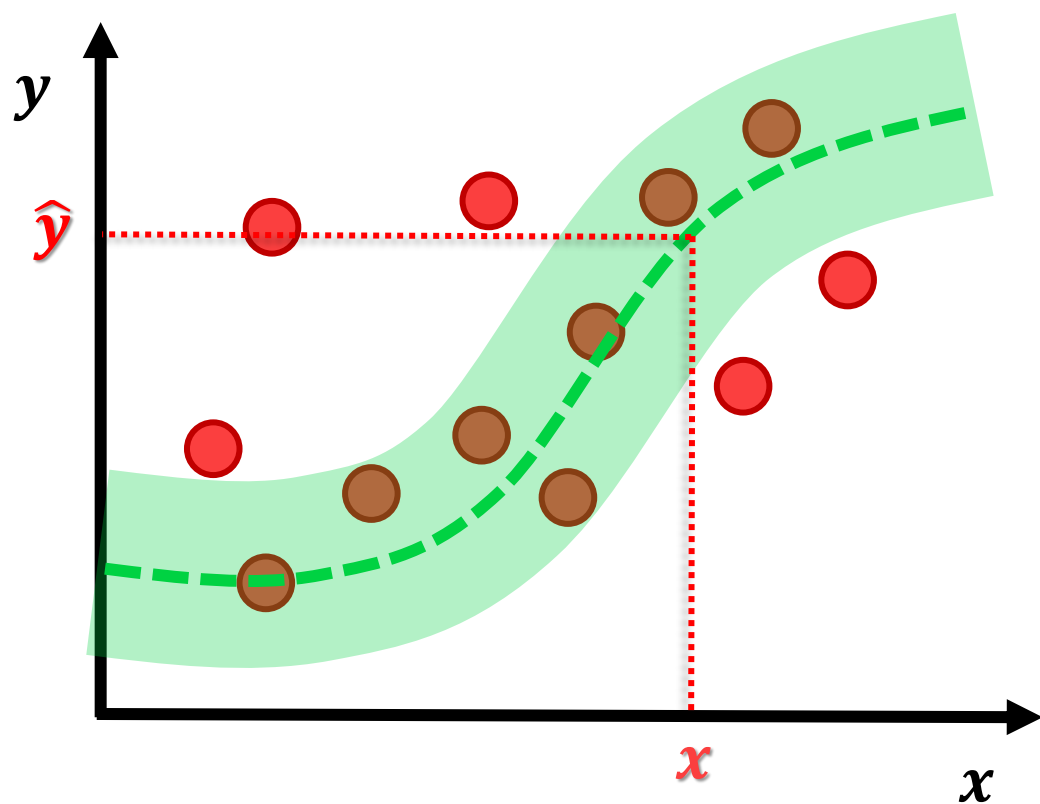
(ε -insensitive tube; função custo)

$$\frac{1}{2} \|w\|^2 + c \sum (\xi_i - \xi_i^*)^2$$



Support Vector Regression (SVR)

Mesma idéia para uma **SVR de kernel não linear**: plugamos um valor de x e obter uma estimativa de y



Support Vector Regression (SVR)

Parâmetro	Hiperparâmetro ou estimado?	Interpretação
C (Custo)	Hiperparâmetro	Controla o equilíbrio entre a complexidade do modelo e a penalização por erros maiores que ϵ . Valores altos permitem menos erro, mas podem levar a “overfitting”
ϵ (Épsilon)	Hiperparâmetro	Define a margem de tolerância (tubo ϵ) onde os erros não são penalizados. Controla a sensibilidade do modelo aos pequenos desvios
γ (Gama)	Hiperparâmetro	Especifica a influência de um ponto de treino no modelo, especialmente em kernels como RBF. Gamas altos $\rightarrow$ modelo mais complexo/localizado
α_i, α_i^*	Aprendido a partir dos dados	Coeficientes associados aos vetores de suporte. Determinam quanto cada ponto de treino fora do tubo ϵ contribui para a predição
b (viés)	Aprendido a partir dos dados	Termo de interceptação que ajusta verticalmente a função de regressão. Calculado com base nos vetores de suporte
w (vetor de pesos)	Aprendido a partir dos dados (somente para kernel linear)	Define a inclinação da reta de regressão no espaço original dos dados. Calculado a partir dos α_i, α_i^* e dos dados de entrada

Support Vector Regression (SVR)

Se o **tubo ϵ** for muito largo...

- Nenhum ponto viola a margem, ou seja, todos os erros são considerados aceitáveis
- Nenhuma penalidade será aplicada, e nenhum vetor de suporte é usado
- O modelo pode aprender uma reta muito simples (até mesmo uma constante), ignorando completamente os padrões nos dados

Ou seja, **ϵ** define a zona de erro aceitável e **C** define quanto nos importamos com erros que ultrapassam **ϵ** :

- Se **C** é muito alto, o modelo vai tentar incluir o máximo de pontos possível dentro do **tubo**
- Se **C** é muito baixo, o modelo pode deixar muitos pontos fora do **tubo**

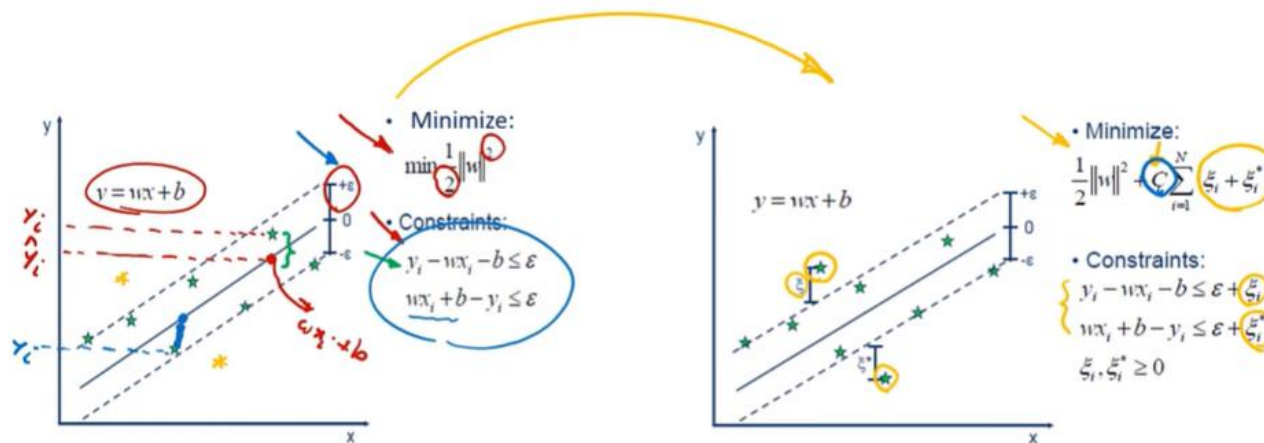
Moral da história:

- Um **tubo ϵ** muito largo pode aprender pouco...
- Um **tubo ϵ** muito estrito pode aprender demais (“overfitting”)...
- O segredo da SVR está em encontrar o equilíbrio certo entre **ϵ** e **C**, de acordo com a natureza dos seus dados ➔ **Validação cruzada (“cross-validation”)**

Support Vector Regression (SVR)

Característica	Regressão Linear	Support Vector Regression
Objetivo	Minimizar o erro quadrático entre previsões e reais (resíduo)	Minimizar o erro dentro de uma margem de tolerância (ϵ)
Função de custo	Soma dos erros quadráticos	Função de perda ϵ -insensível + penalização por slack
Sensibilidade a outliers	Alta	Menor, pois ignora erros dentro de ϵ
Tipo de função ajustada	Linear	Pode ser linear ou não-linear (com kernel)
Capacidade de modelar não-linearidades	Limitada (sem transformação)	Alta, com uso de kernels não-linear
Interpretação dos coeficientes	Direta e fácil de interpretar	Mais difícil, especialmente com kernel
Número de “suportes” (vetores)	Todos os pontos afetam a regressão	Apenas os pontos fora da margem ϵ afetam a regressão
Eficiência computacional	Alta, treinamento rápido	Pode ser mais lento, especialmente com kernels
Aplicações típicas	Problemas simples, dados lineares	Problemas complexos ou com ruído

Mais informações...



Pause (k)

Source: https://www.saedsayad.com/support_vector_machine_reg.htm



We had to pause your ad blocker. Toggle on to enable experimental ad blocking.



Part 26-Support Vector Machines Regression



Pedram Jahangiry
11.4K subscribers

Subscribe

643



Share

Download



Resumo: Support Vector Regression (SVR)

- *SVR funciona como uma regressão (dados numéricos)*
- *SVR utiliza o tubo ϵ -insensível e variáveis de folga*
- *SVR pode utilizar funções kernel*
- *Também emprega o Truque do Kernel*
- *Estimativas são baseadas no hiperplano, computado com base em vetores de suporte (i.e., todos os pontos fora do tubo ϵ -insensível)*