

FIAP

NBA



Support Vector Machines

Dheny R. Fernandes

1. SVM

1. Intuição
2. Conceitos Básicos
3. Kernels:
 1. Polinomial
 2. Kernel trick
 3. RBF

2. Métricas de Avaliação

1. Matriz de confusão
2. Acurácia
3. Precision
4. Recall
5. F1-Score

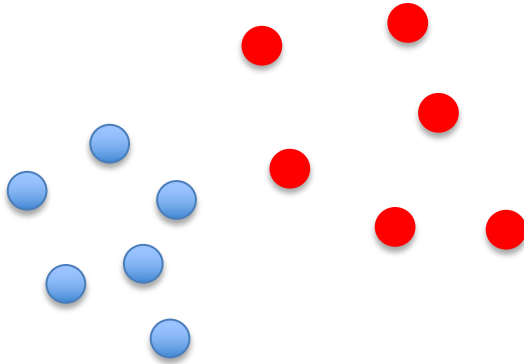
Support Vector Machines - SVM

Apresentado em 1979 por Vladimir Vapnik, é um dos algoritmos de classificação (apesar de haver sua variante para regressão) mais sólidos já criado.

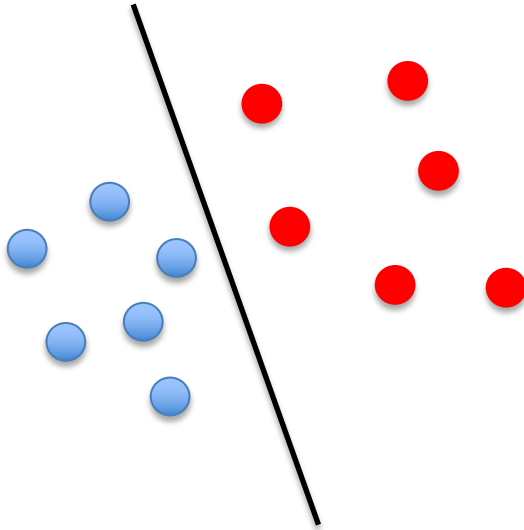
Com a existência de duas ou mais classes rotuladas de dados, ele age como um classificador discriminativo, formalmente definido por um hiperplano ideal que separa todas as classes.

Novos exemplos são classificados de acordo com a posição no espaço em que se encontram.

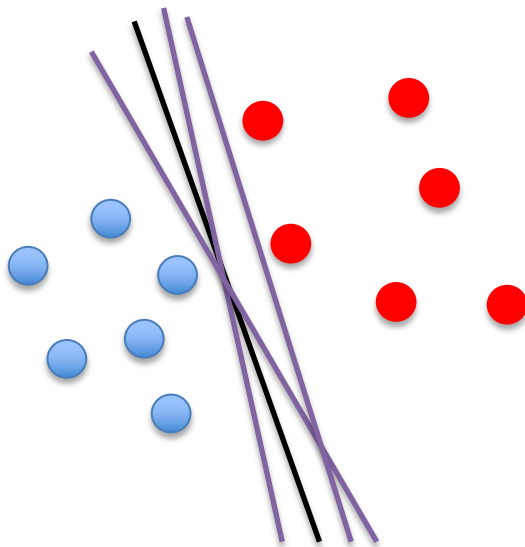
Suponha o seguinte conjunto de dados



A seguinte reta é capaz de separar os elementos de cada classe



Mas essas também são:



A questão é como encontrar uma
reta boa o suficiente de modo o
algoritmo generalize bem para
amostras ainda não vistas

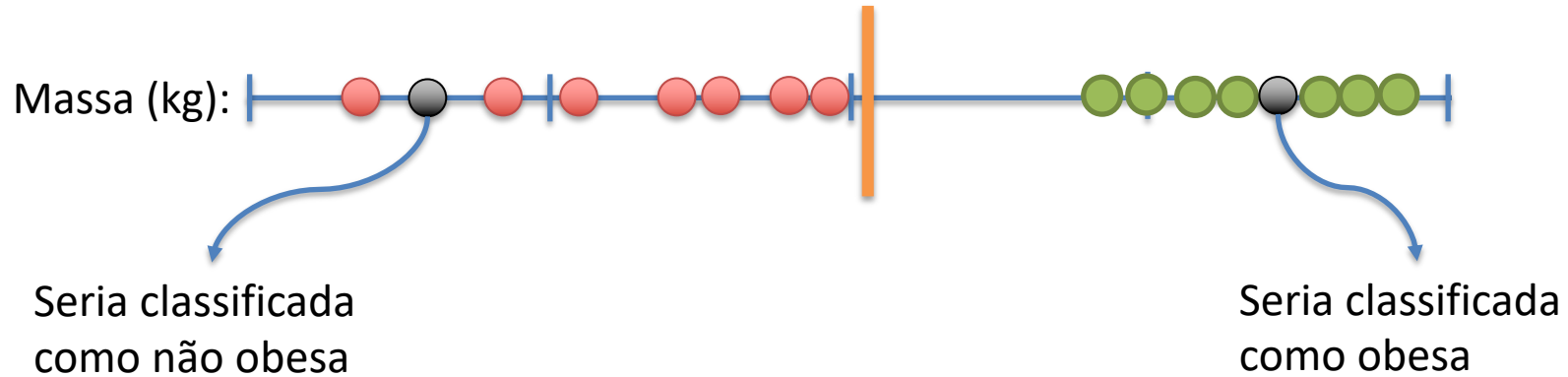
Vamos supor que mensuramos a massa de alguns cachorros:



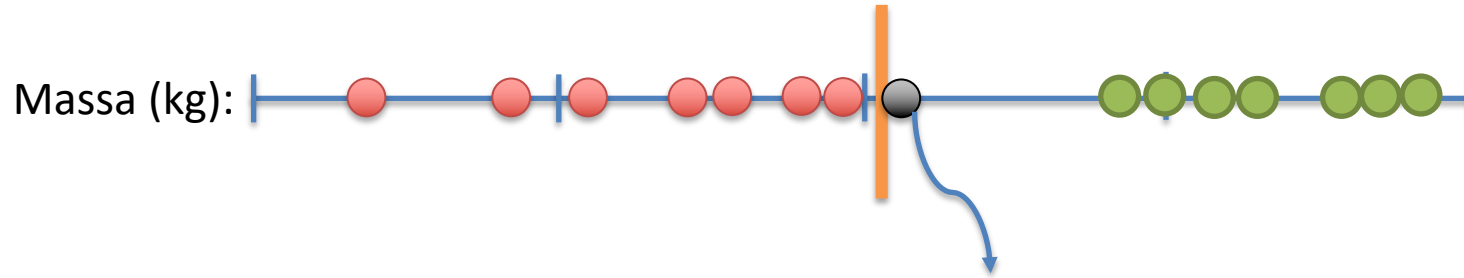
Os pontos em **vermelho** representam aqueles que não são obesos, enquanto os pontos em **verde** representam aqueles que são obesos:



Podemos, então, estabelecer um threshold e classificar novas amostras

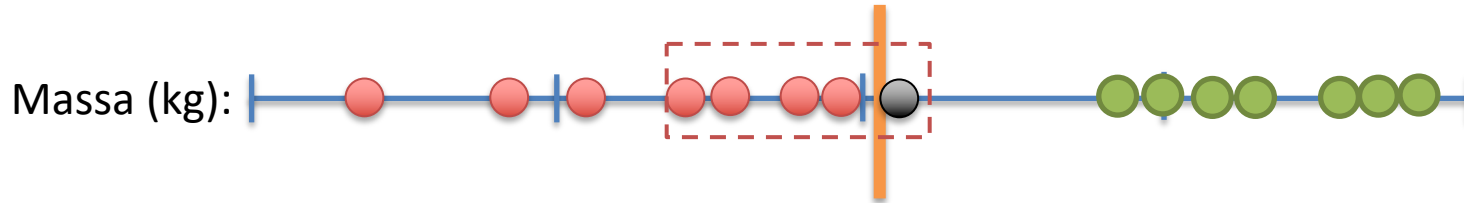


Entretanto, como essa amostra seria classificada?



Como ela está acima do threshold, a classificaríamos como obesa

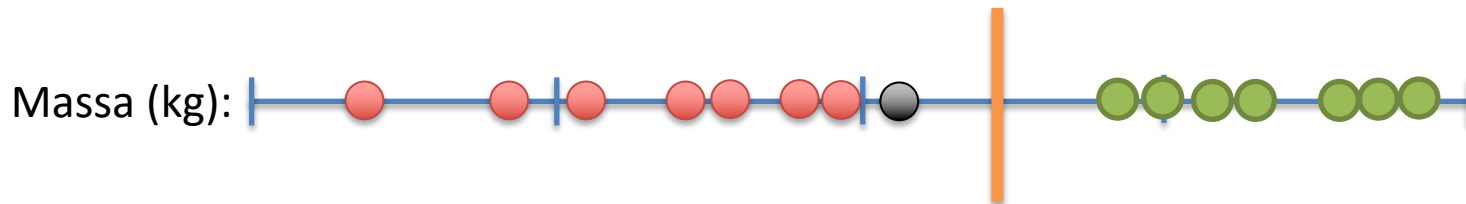
O que não faz sentido, visto que ela é muito mais próxima das amostras da outra classe



Assim, precisamos de uma maneira mais inteligente de definir o threshold.

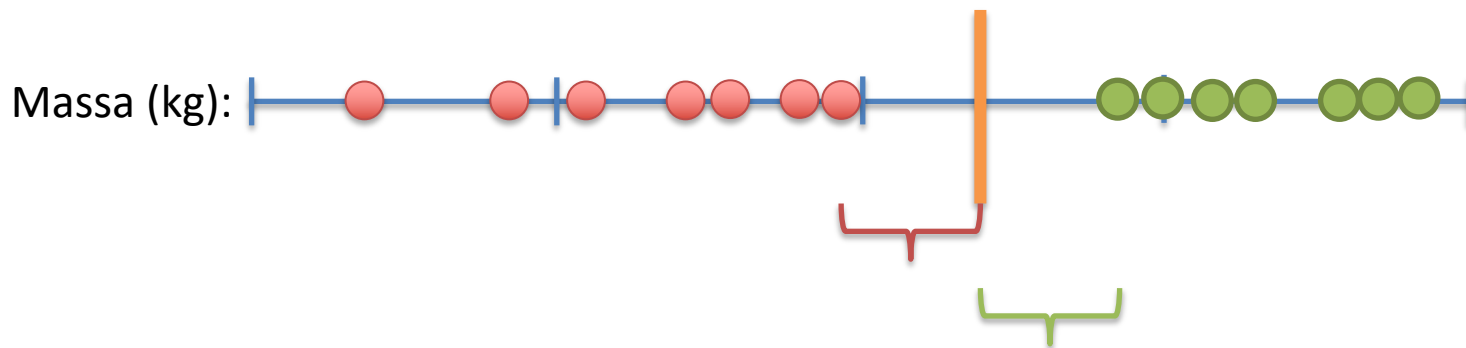
Sugestões?

Voltando aos dados originais, podemos escolher os pontos que estão na borda de cada grupo e usar o ponto médio entre eles como threshold:



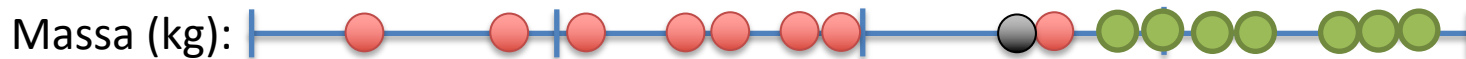
Agora, a nova amostra será posicionada a esquerda do threshold e será classificada como não obesa

O que fizemos ao escolher esse ponto como threshold foi definir uma **margem**, ou seja, a menor distância entre pontos de distintas classes e o threshold.



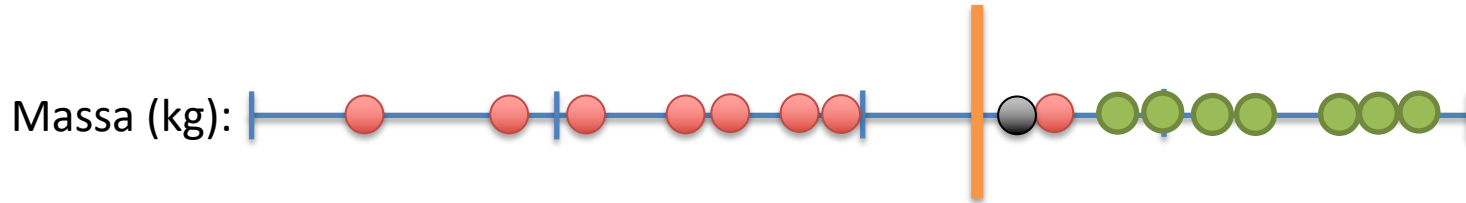
Quando usamos um threshold que nos fornece a maior margem para fazer as classificações estamos usando um ***Maximal Margin Classifier***.

Maximal Margin Classifier podem parecer a solução para nossos problemas, mas e se o conjunto de treino for assim?



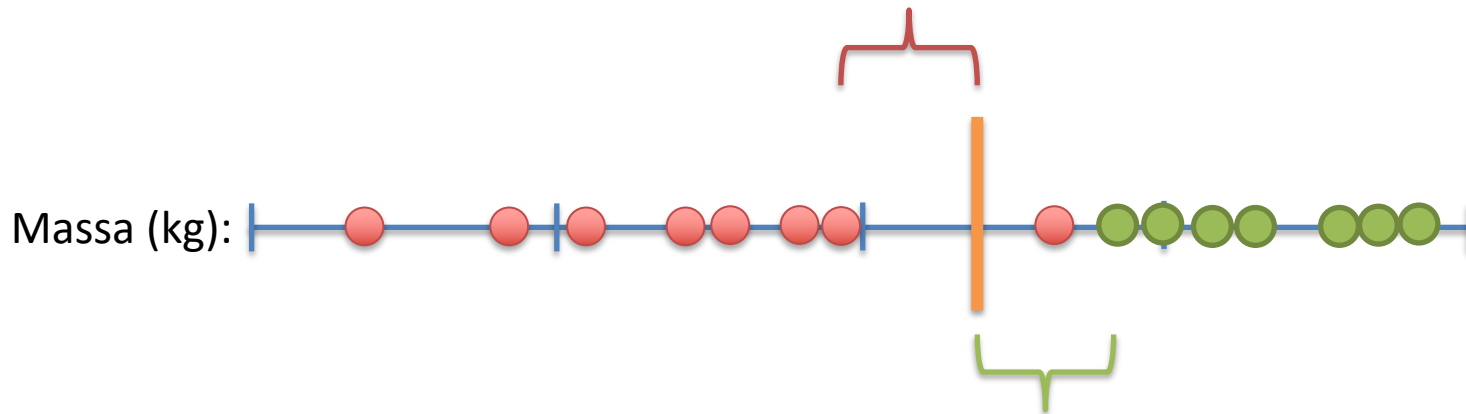
Dessa maneira, novas amostras seriam incorretamente classificadas. Isso torna ***Maximal Margin Classifier*** muito sensível a outliers.

Para resolver esse problema, precisamos estabelecer um threshold que não seja sensível a outliers e, para isso, devemos **permitir erros de classificação**.

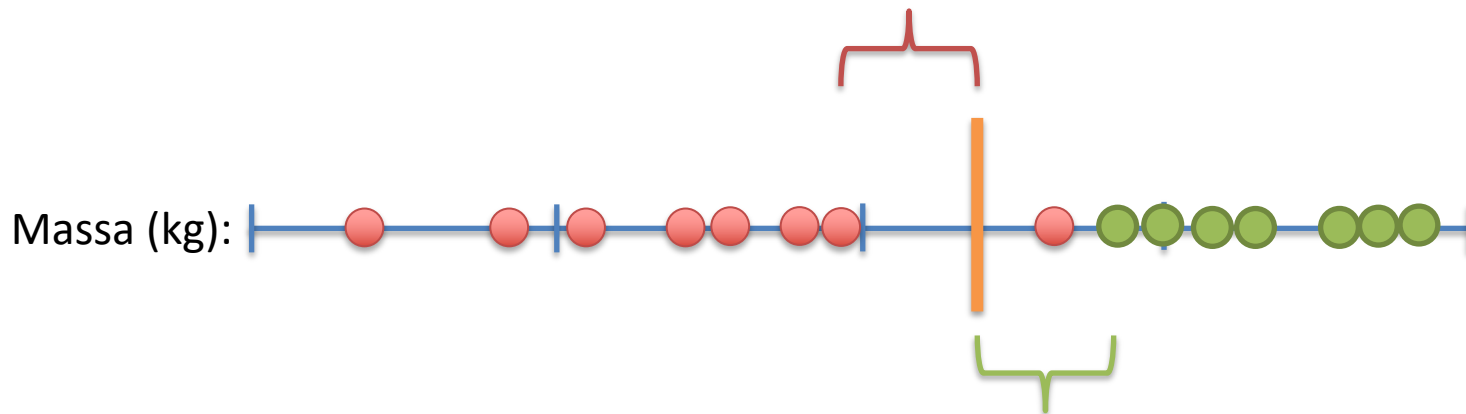


Se, por exemplo, usarmos o threshold acima, iremos errar um ponto (em vermelho), mas permitiria melhores predições.

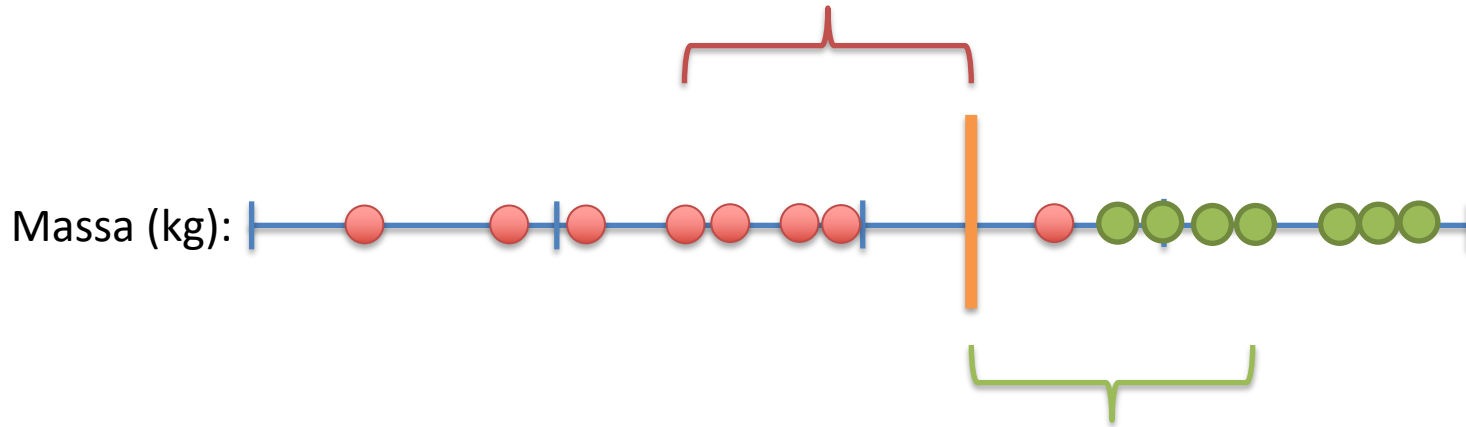
Quando permitimos erros de classificação, a distância entre os pontos e o threshold é chamado de ***Soft Margin***.



Agora a questão é: como eu sei que essa margem...

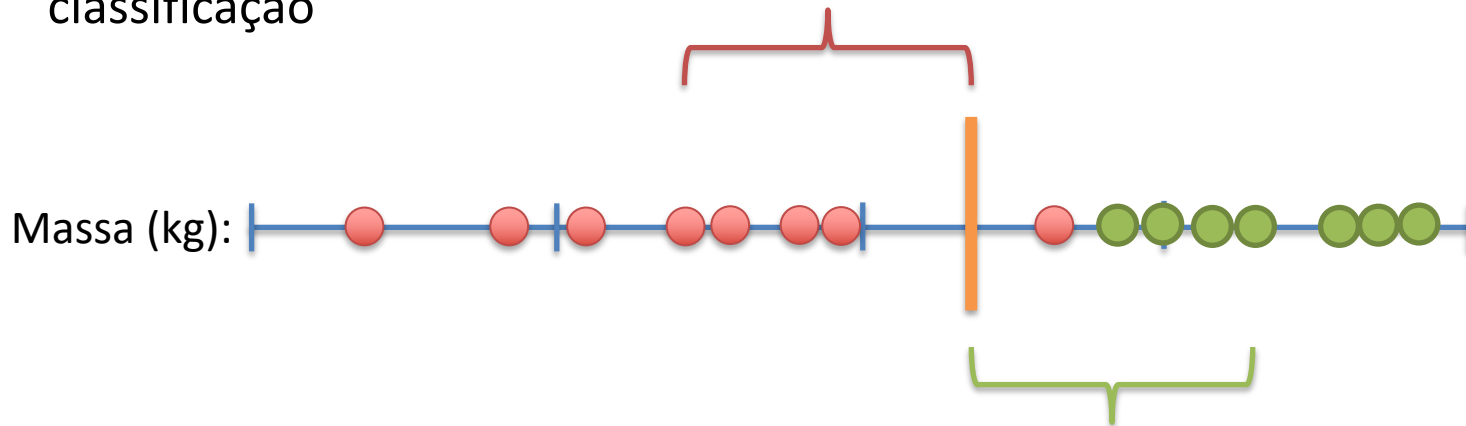


É melhor que essa?

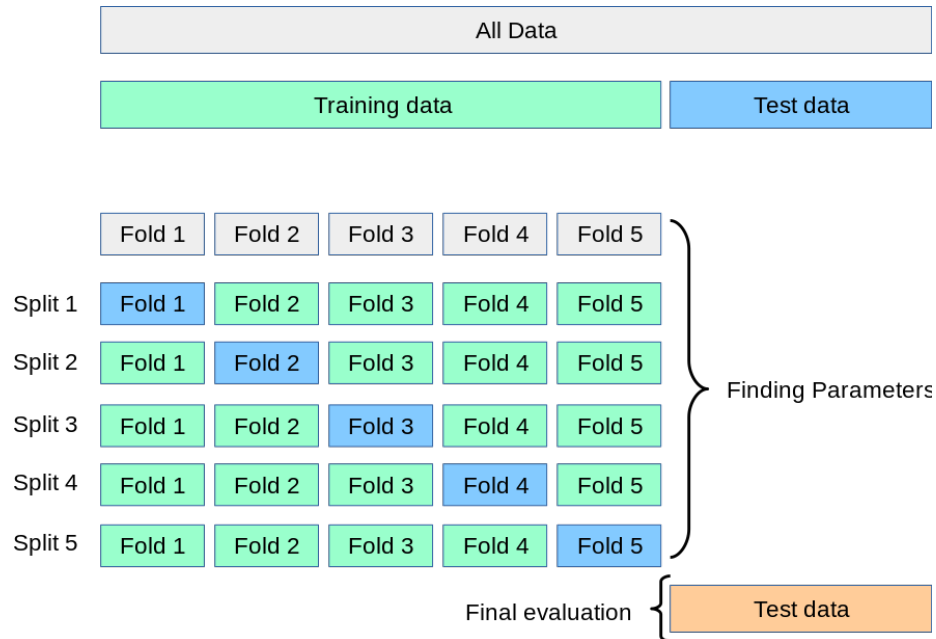


Ou melhor que qualquer outra?

Podemos usar **Cross Validation** para determinar quantos erros de classificação iremos permitir dentro da Soft Margin para obter a melhor classificação

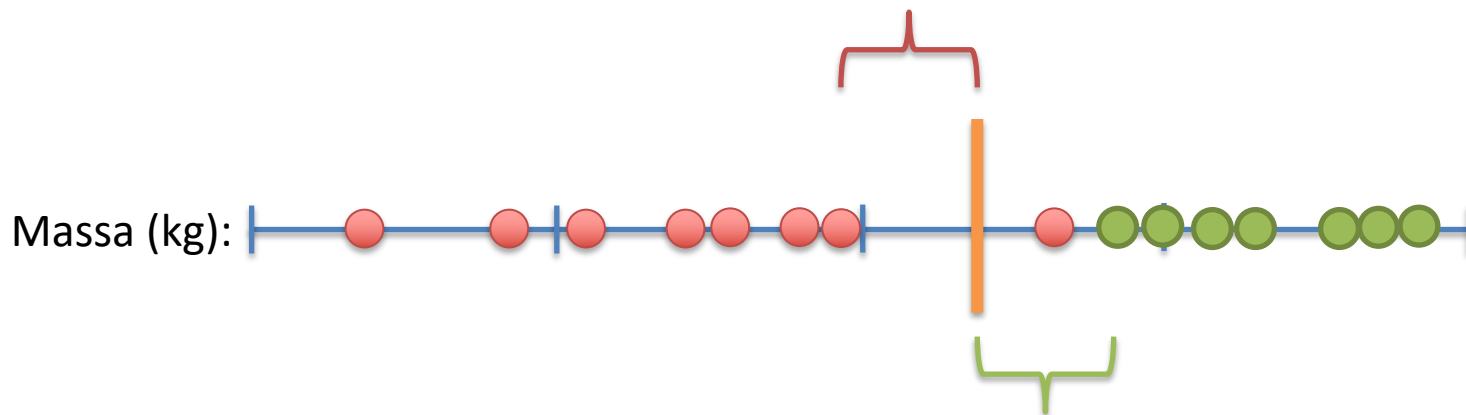


Cross Validation é o processo de dividir um conjunto de treino em N conjuntos de treino e validação e escolher aquele que retorna a menor quantidade de erros de classificação

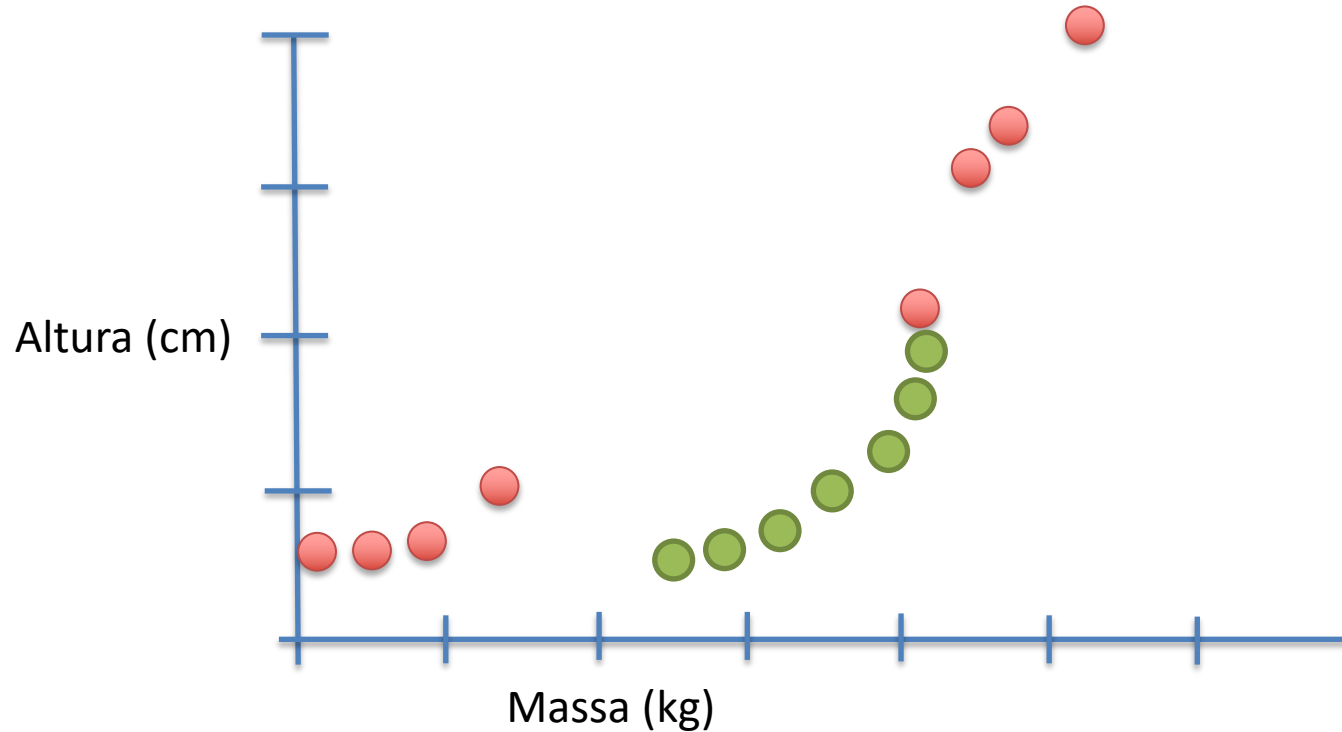


Assim, quando usamos Soft Margin para determinar a localização de um threshold, estamos usando um **Support Vector Classifier** para classificar observações.

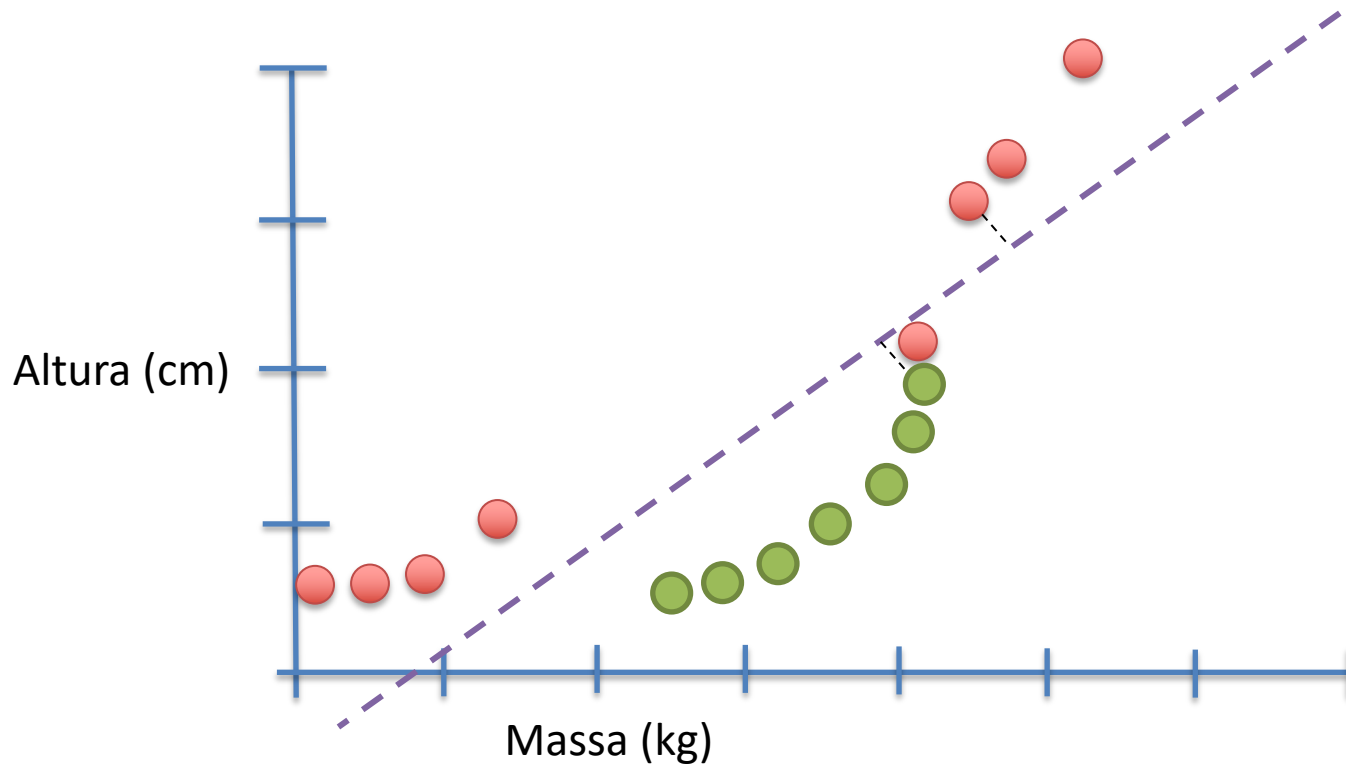
Esse nome vem do fato que observações na borda e entre a Soft Margin são denominadas Support Vector.



Agora, imagine que nossos dados possuem esse comportamento:

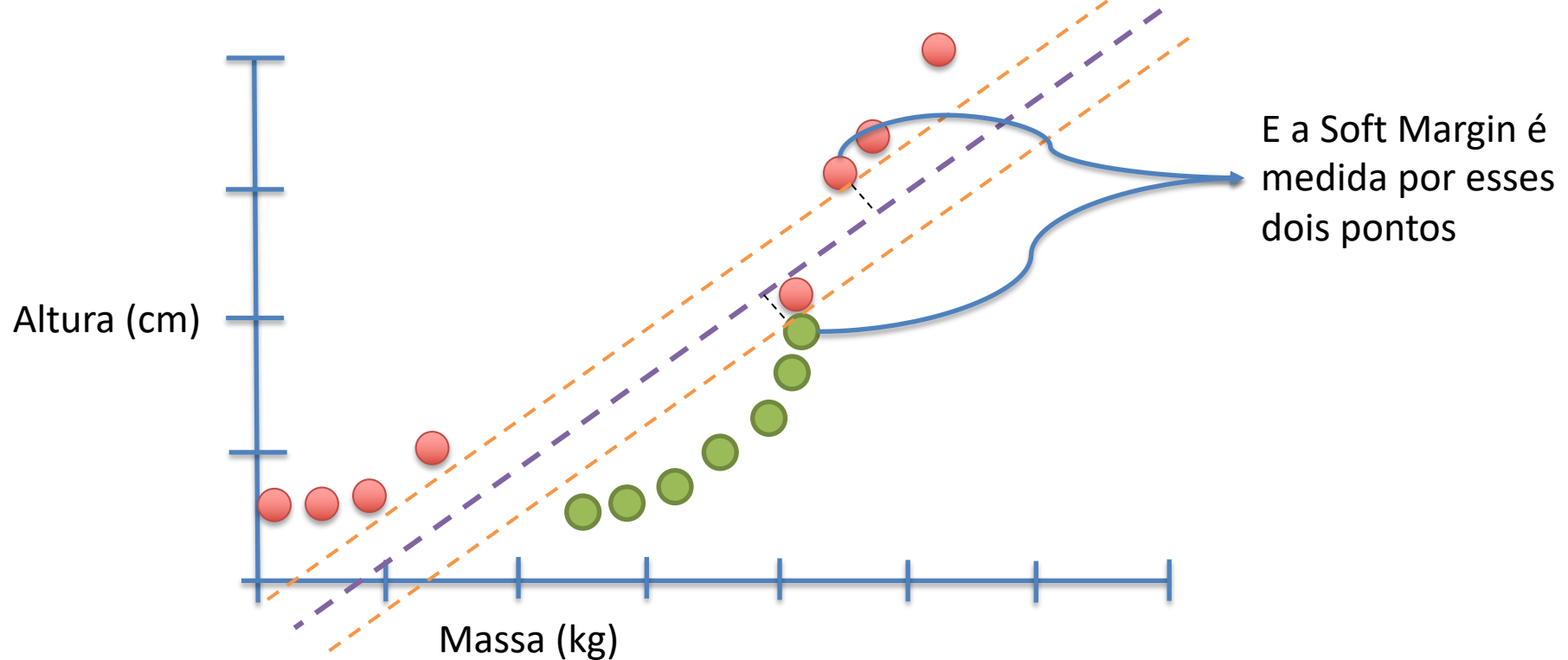


Agora, imagine que nossos dados possuem esse comportamento:

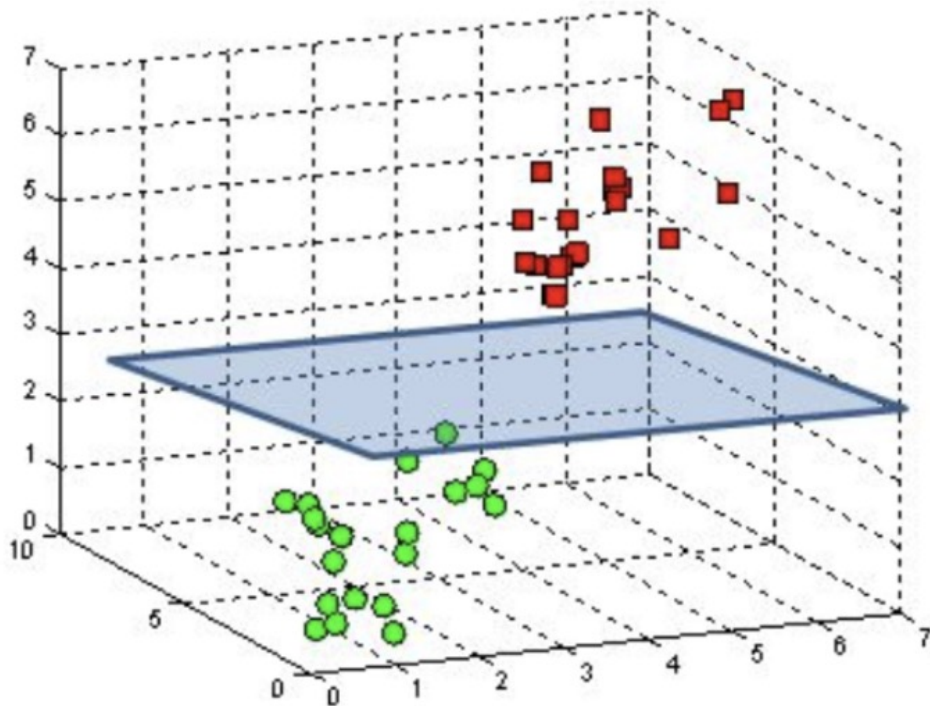


Nesse caso, o
Support Vector
Classifier é uma
linha

Agora, imagine que nossos dados possuem esse comportamento:



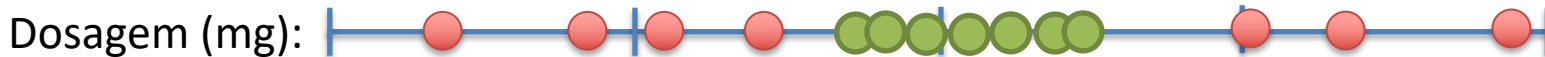
Quando os dados possuem 3 dimensões, temos plano como Support Vector Classifier



Para 4 ou mais dimensões, temos um hiperplano (e não tem como representar isso graficamente).

Support Vector Classifiers podem ser interessantes pois eles lidam bem com outliers e, por permitirem erros de classificações na etapa de treino, podem lidar com amostras mais difíceis na etapa de teste.

Entretanto, e se esse for nosso conjunto de treino?

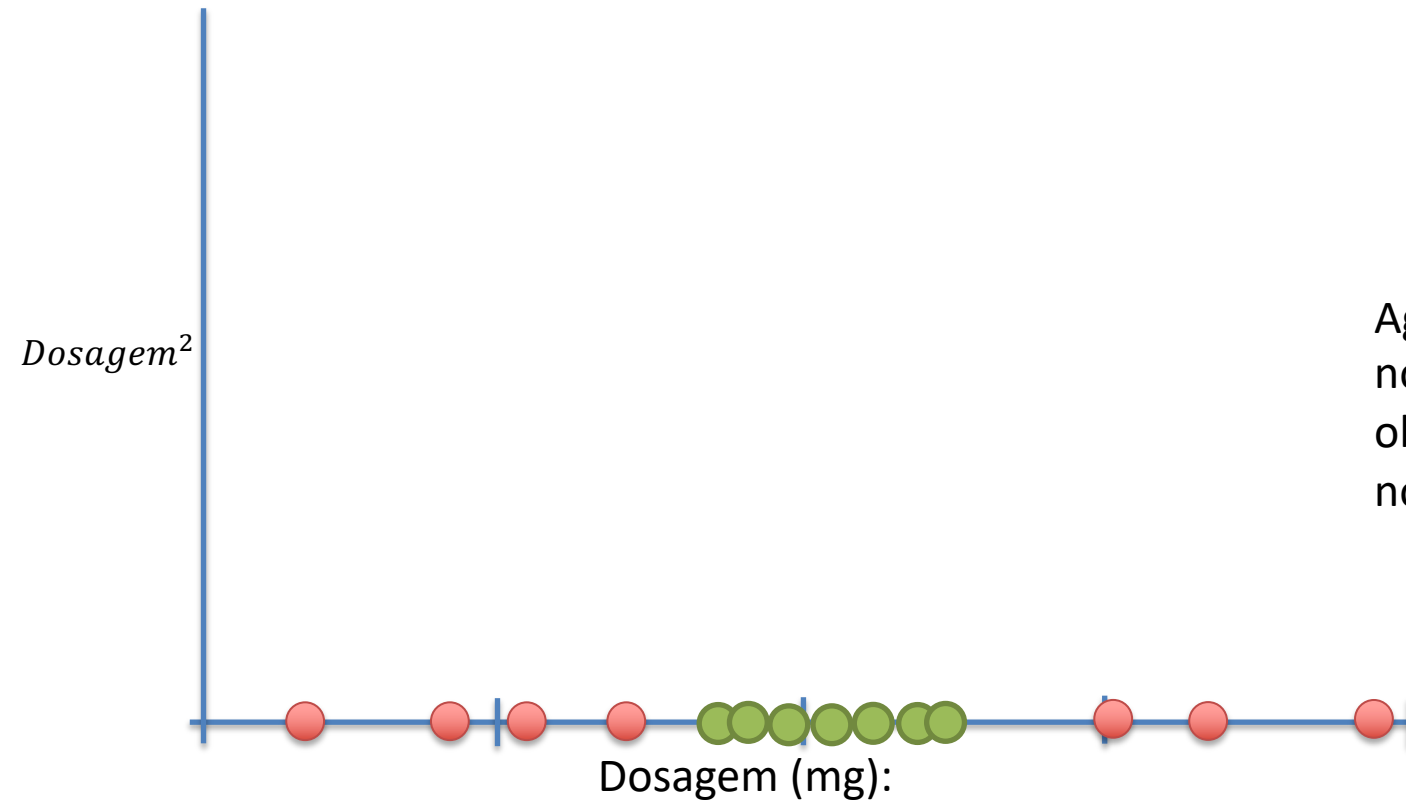


Nesse contexto é que entram Support Vector Machines

Vamos começar entendendo a principal ideia por trás do SVM. Para isso, adicionamos um eixo y para poder desenhar um gráfico:

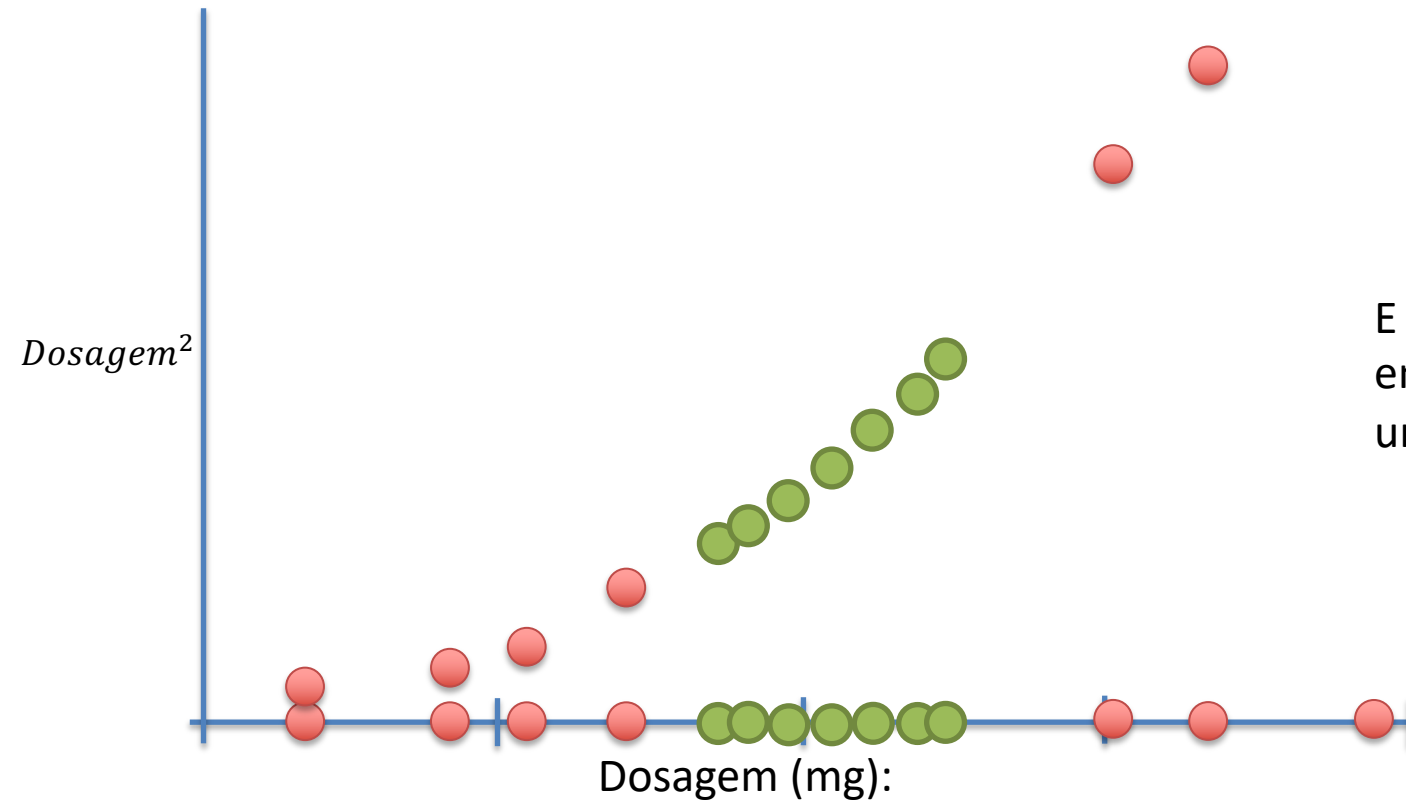


Vamos começar entendendo a principal ideia por trás do SVM. Para isso, adicionamos um eixo y para poder desenhar um gráfico:



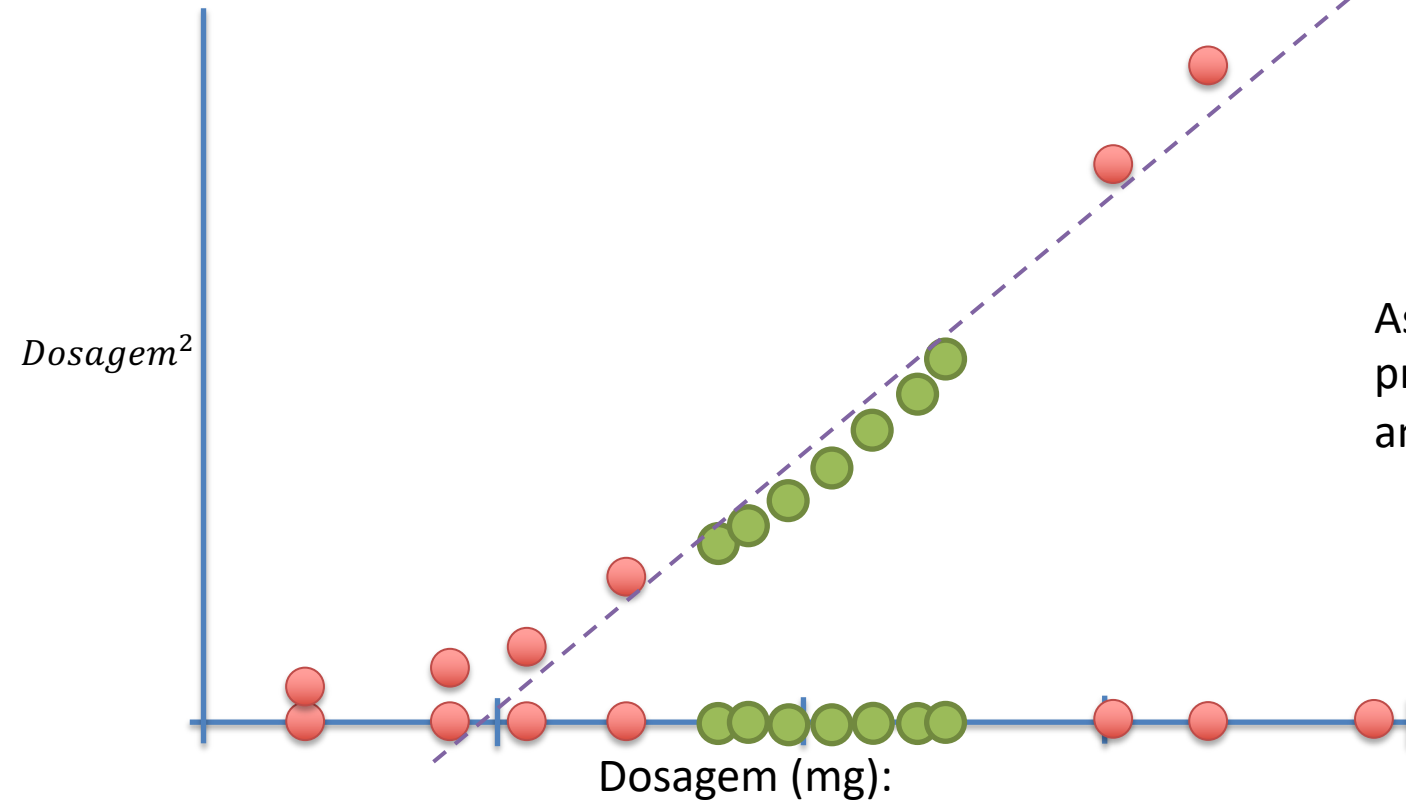
Agora, elevamos cada valor no eixo x ao quadrado para obter seu correspondente no eixo y

Vamos começar entendendo a principal ideia por trás do SVM. Para isso, adicionamos um eixo y para poder desenhar um gráfico:



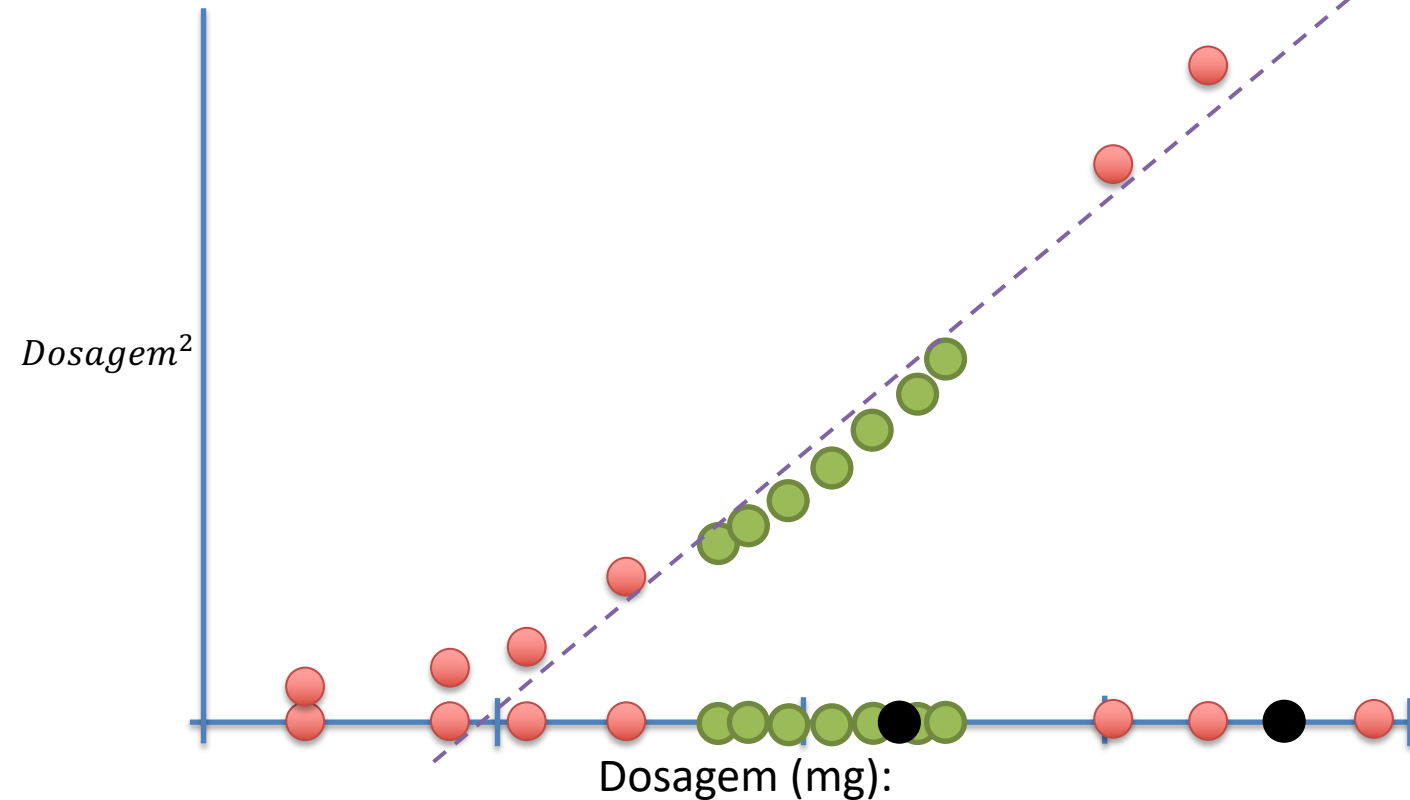
E agora que os dados estão em 2D, podemos desenhar um SVC

Vamos começar entendendo a principal ideia por trás do SVM. Para isso, adicionamos um eixo y para poder desenhar um gráfico:



Assim, é possível fazer previsões de novas amostras. Por exemplo:

Vamos começar entendendo a principal ideia por trás do SVM. Para isso, adicionamos um eixo y para poder desenhar um gráfico:



Assim, podemos listar os passos que compõem a principal ideia por trás do SVM:

- 1. Comece com os dados numa dimensão relativamente baixa***
- 2. Mova os dados para uma dimensão superior***
- 3. Encontre um SVC que separe os dados em grupos distintos***

A questão que permanece, entretanto, é: como decidimos a maneira que vamos transformar os dados?

O SVM usa as chamadas funções Kernel para sistematicamente encontrar SVCs em dimensões maiores.

Aqui, vou tratar dos dois principais Kernels: polinomial e RBF (Radial Basis Function)

O Kernel polinomial é definido da seguinte maneira:

$$k(x, y) = (x^T y + c)^d, \text{ em que:}$$

x e y são vetores de características que representam amostras de treino

c é um parâmetro de regularização, que diz ao SVM o quanto você evitar errar a Classificação de cada exemplo de treinamento. Grandes valores de c geram uma margem menor. Já menores valores de c geram uma margem maior, mesmo o hiperplano errando a classificação de mais pontos.

d é o grau do polinômio que vamos usar

Os valores de c e d são obtidos através de *Cross Validation*. (para efeitos de facilidade de cálculo, vou usar $x^T = x$)

Vamos supor que encontramos os seguintes os valores de c e d : $\frac{1}{2}$ e 2 .

Assim, temos a seguinte equação:

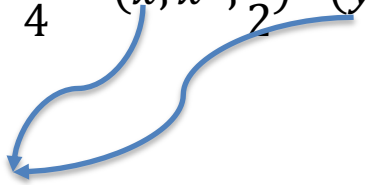
$$(x \cdot y + \frac{1}{2})^2 = \left(x \cdot y + \frac{1}{2}\right) \left(x \cdot y + \frac{1}{2}\right) = \left(x^2 y^2 + \frac{1}{2}xy + \frac{1}{2}xy + \frac{1}{4}\right) = xy + x^2 y^2 + \frac{1}{4}$$

$$\left(x \cdot y + \frac{1}{2}\right)^2 = \left(x \cdot y + \frac{1}{2}\right)\left(x \cdot y + \frac{1}{2}\right) = \left(x^2 y^2 + \frac{1}{2}xy + \frac{1}{2}xy + \frac{1}{4}\right) = xy + x^2 y^2 + \frac{1}{4}$$

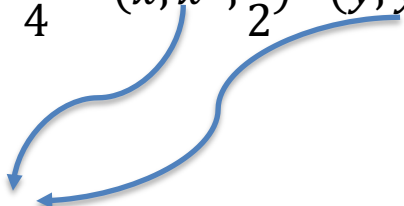
Este polinômio, por fim, é igual a esse ***dot product***:

$$xy + x^2 y^2 + \frac{1}{4} = \left(x, x^2, \frac{1}{2}\right) \cdot \left(y, y^2, \frac{1}{2}\right)$$

Esse dot product fornece as coordenadas dos dados numa alta dimensão.

$$xy + x^2y^2 + \frac{1}{4} = (x, x^2, \frac{1}{2}) \cdot (y, y^2, \frac{1}{2})$$


Representam as
coordenadas do eixo
x de duas amostras

$$xy + x^2y^2 + \frac{1}{4} = (x, x^2, \frac{1}{2}) \cdot (y, y^2, \frac{1}{2})$$


Representam as
coordenadas do eixo
y de duas amostras

$\frac{1}{2}$ e $\frac{1}{2}$ representam as
coordenadas no eixo z, mas
como estamos usando um
polinômio de grau 2,
podemos ignorá-las.

Se quisermos saber a relação entre duas amostras numa alta dimensão (definida pelo kernel da equação), basta substituir seus valores no lugar de x e y .

Observe que, ao fazer isso, os dados **não são** de fato transformados para uma outra dimensão. Essa equação que nos permite encontrar as relações entre amostras em outras dimensões sem de fato transformar os dados é chamado de kernel trick.

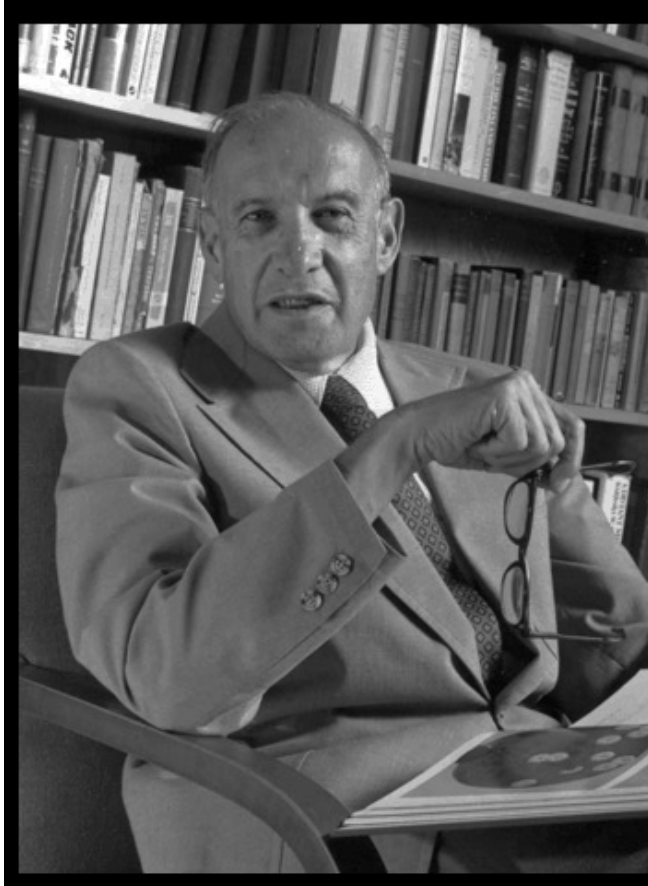
O Kernel *Radial Basis Function* (RBF) é definido da seguinte maneira:

$$e^{-\gamma \|x_i - x_j\|^2}, \text{ em que:}$$

e representa o número de Euler = 2,71828

No jupyter notebook eu dou mais detalhes do funcionamento da função.

Métricas de Avaliação



You can't
manage what
you can't
measure

-Peter Drucker

Precisamos de maneiras de mensurar a qualidade de predição de nosso algoritmo. Em classificação, as métricas mais comuns, e as mais simples, são **Acurácia**, **Precision**, **Recall** e **F1 Score**. Elas são obtidas a partir da [Matriz de Confusão](#), apresentada abaixo:

		True Class	
		Positive	Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN

A acurácia é dada pela seguinte função:

$$acc = \frac{tp + tn}{(tp + tn + fp + fn)}$$

Precision (acurácia das predições positivas) pode ser calculado da seguinte maneira:

$$P = \frac{tp}{(tp + fp)}$$

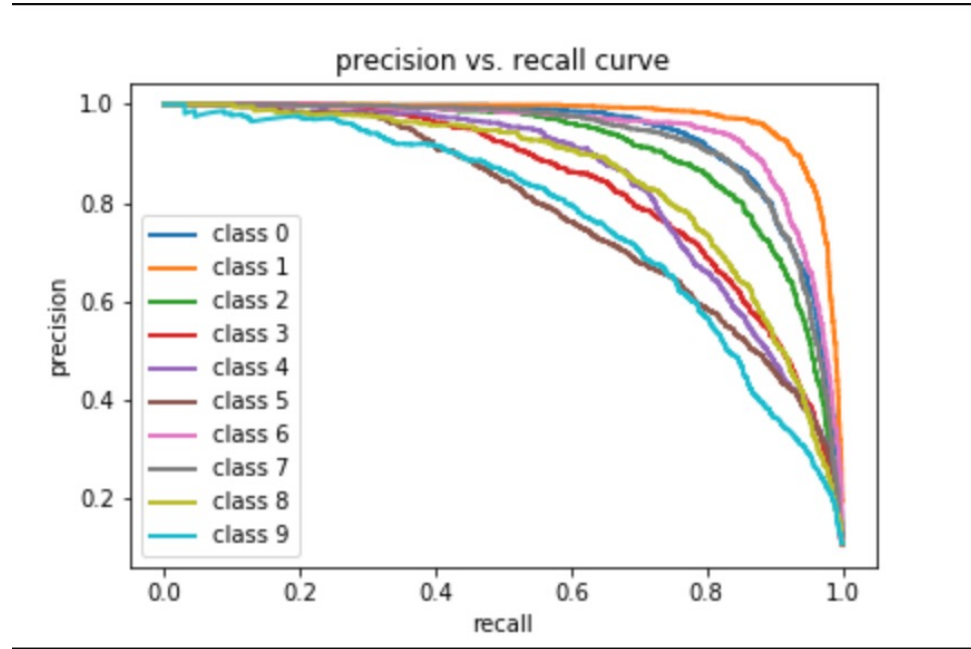
Já o Recall (taxa de amostras positivas corretamente encontradas pelo classificador) é calculado da seguinte forma:

$$R = \frac{tp}{(tp + fn)}$$

Precision e **Recall** tendem a ser inversamente proporcionais, como apresentado no gráfico ao lado.

Para minimizar esse problema, existe a métrica **F1-Score**, que leva em consideração tanto Precision quanto Recall.

$$F1 = \frac{2PR}{P + R}$$



Implementação




Exercícios



Obrigado!

profdheny.fernandes@fiap.com.br

 /dhenyfernandes

FIAP MBA⁺

Copyright © 2022 | Professor Dheny R. Fernandes

Todos os direitos reservados. Reprodução ou divulgação total ou parcial deste documento, é expressamente proibido sem consentimento formal, por escrito, do professor/autor.

FIAP