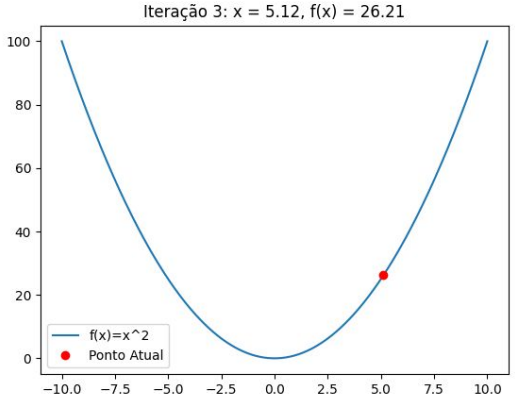
Support Vector Machine

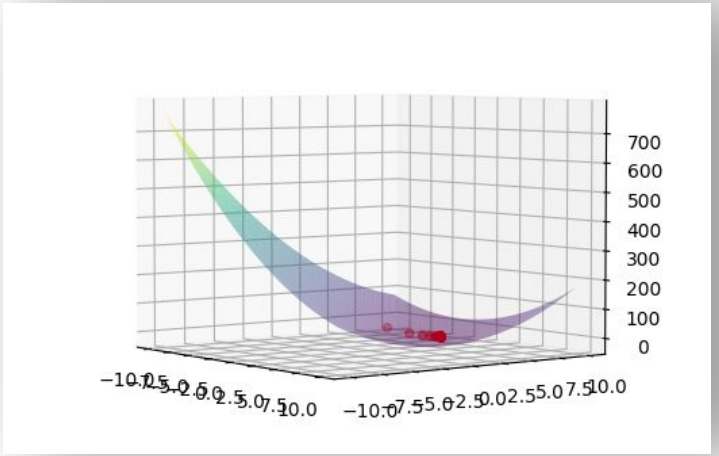
Solon Alves Peixoto

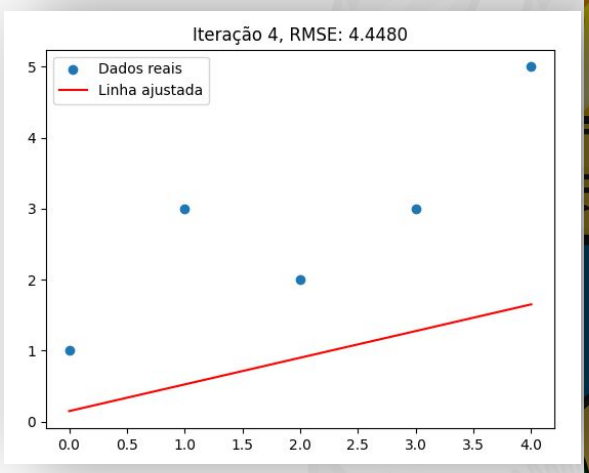
CENTRO DE TECNOLOGIA - UFC
DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA - UFC

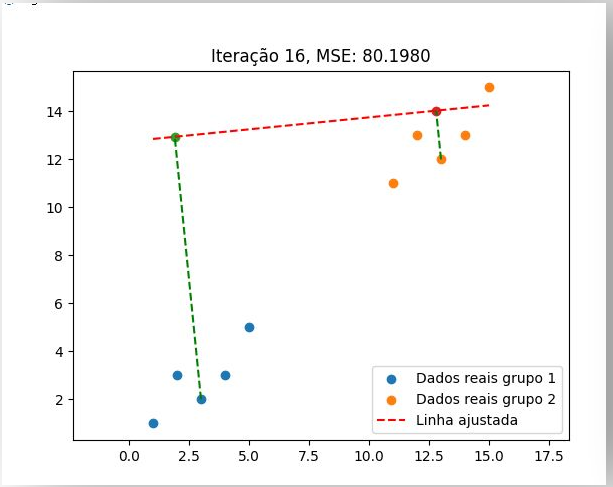
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE TELEINFORMÁTICA – PPGETI-UFC
LABORATÓRIO DE PROCESSAMENTO DE IMAGENS, SINAIS E COMPUTAÇÃO APLICADA – LAPISCO-IFCE

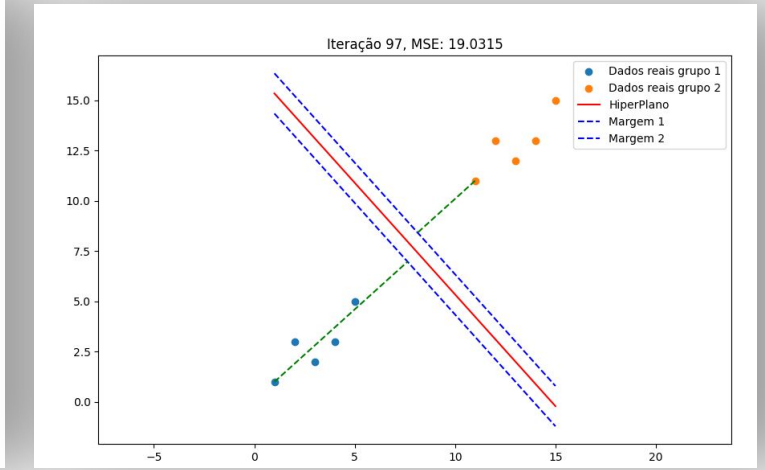
Visão Geral

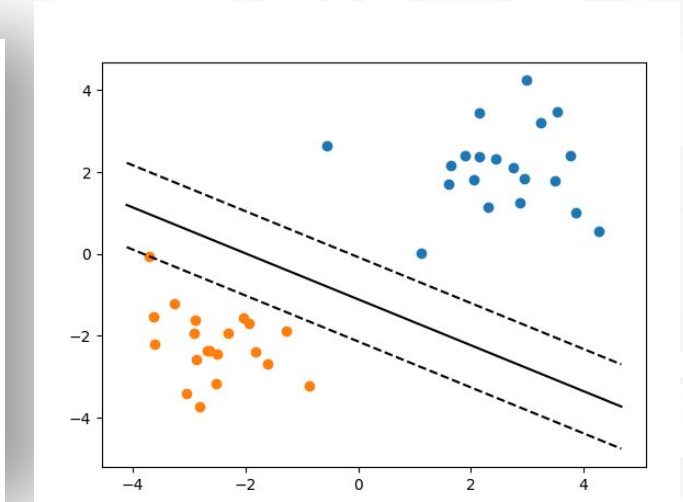

Iteração 3: x = 5.12, f(x) = 26.21
f(x)=x^2
Ponto Atual




Iteração 4, RMSE: 4.4480
Dados reais
Linha ajustada


Iteração 16, MSE: 80.1980
Dados reais grupo 1
Dados reais grupo 2
Linha ajustada


Iteração 97, MSE: 19.0315
Dados reais grupo 1
Dados reais grupo 2
HiperPlano
Margem 1
Margem 2



Gradiente

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

$$f'(x) = 2 \cdot x^{(2-1)}$$

$$f'(x) = 2 \cdot x^1$$

- Inclinação
- Velocidade

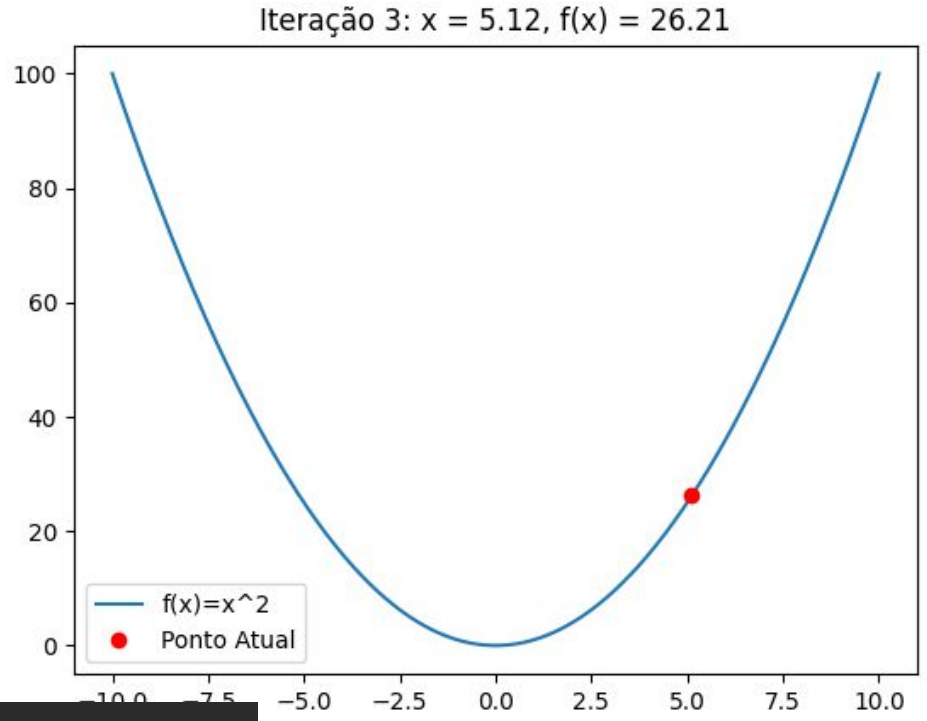
```
def function(x):
    return x ** 2

def gradient(x):
    return 2 * x
```

- O que isso faz aqui?

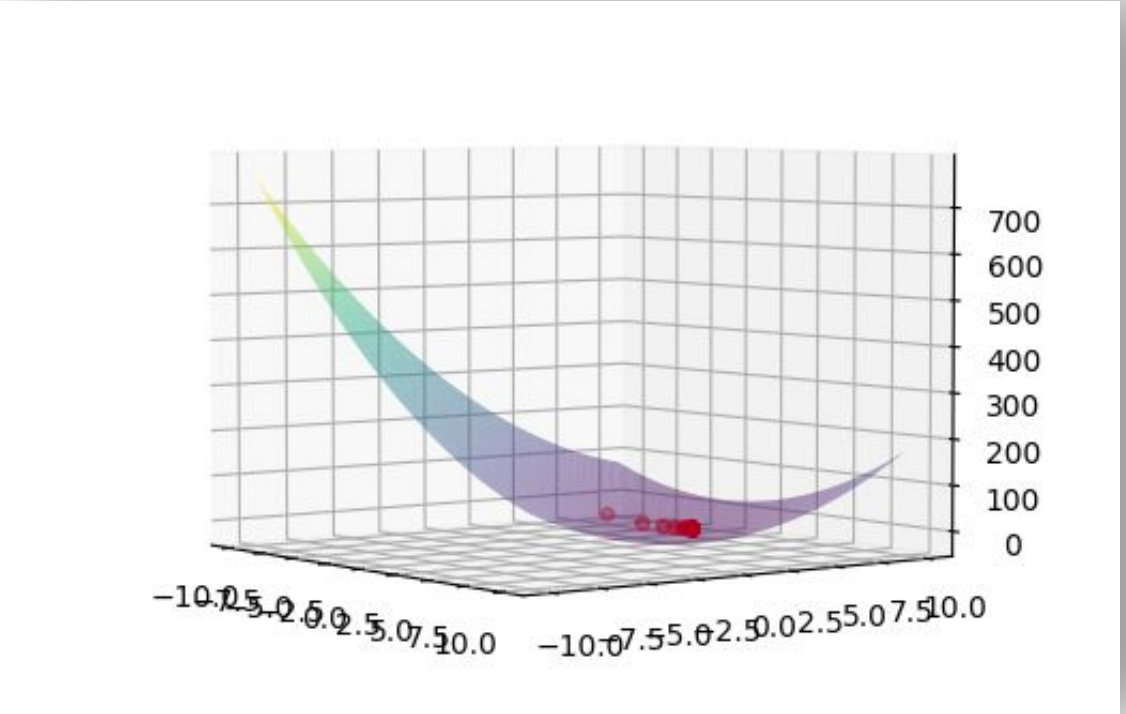
```
grad = gradient(x)

x_new = x - learning_rate * grad
y_new = function(x_new)
```



Gradiente

- O que fazer quando a função, que caracteriza os dados, for desconhecida?
- Aproximação
- Precisamos de alguma equação!



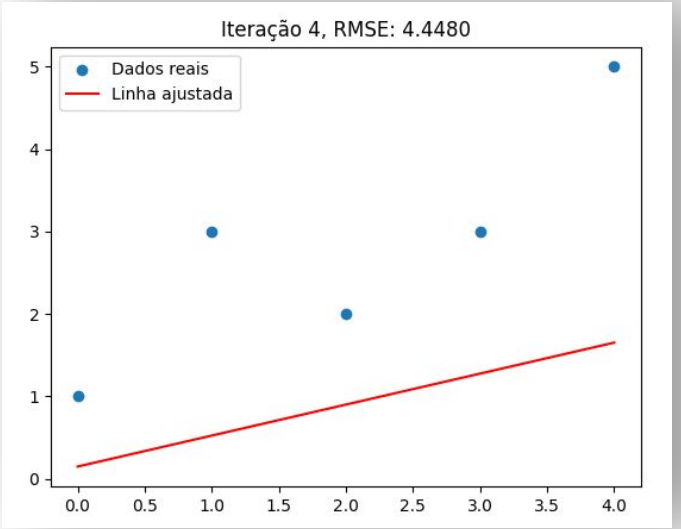
Mean Square Error (MSE)

- $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$

```
x = np.array([0, 1, 2, 3, 4])
y = np.array([1, 3, 2, 3, 5])
```

- y = mx + b

```
y_pred = m * x + b
```



- $$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - (mx_i + b))^2$$

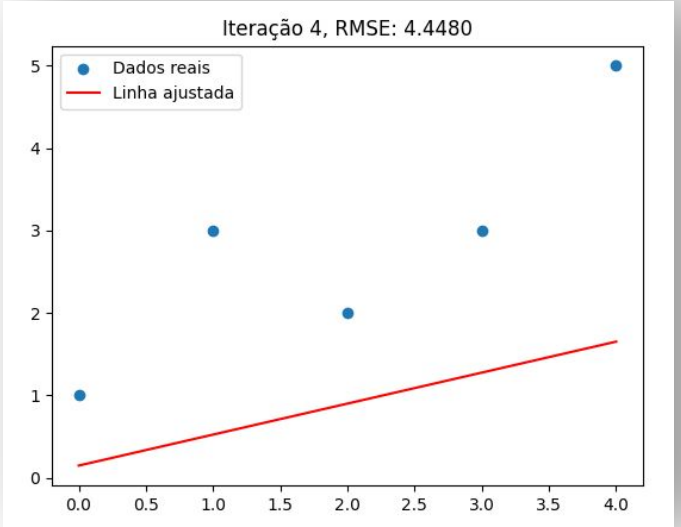
Mean Square Error (MSE)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - (mx_i + b))^2$$

$$\frac{\partial \text{MSE}}{\partial m} = \frac{\partial}{\partial m}\left(\frac{1}{n}\sum_{i=1}^{n}(y_i - mx_i - b)^2\right)$$

$$= \frac{1}{n}\sum_{i=1}^{n} 2(y_i - mx_i - b)(-x_i)$$

$$= -\frac{2}{n}\sum_{i=1}^{n} x_i(y_i - mx_i - b)$$




```
dm = - (2 / n) * np.sum((y - y_pred) * x)
```

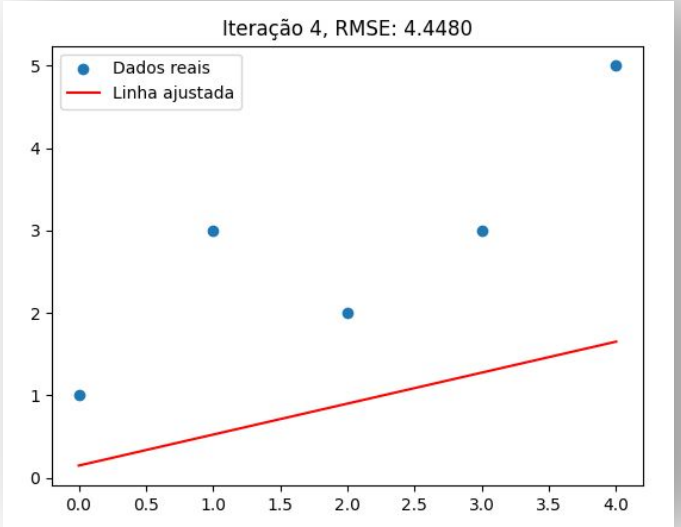
Mean Square Error (MSE)

- $$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - (mx_i + b))^2$$

$$\frac{\partial \text{MSE}}{\partial b} = \frac{\partial}{\partial b}\left(\frac{1}{n}\sum_{i=1}^{n}(y_i - mx_i - b)^2\right)$$

$$= \frac{1}{n}\sum_{i=1}^{n} 2(y_i - mx_i - b)(-1)$$

$$= -\frac{2}{n}\sum_{i=1}^{n}(y_i - mx_i - b)$$

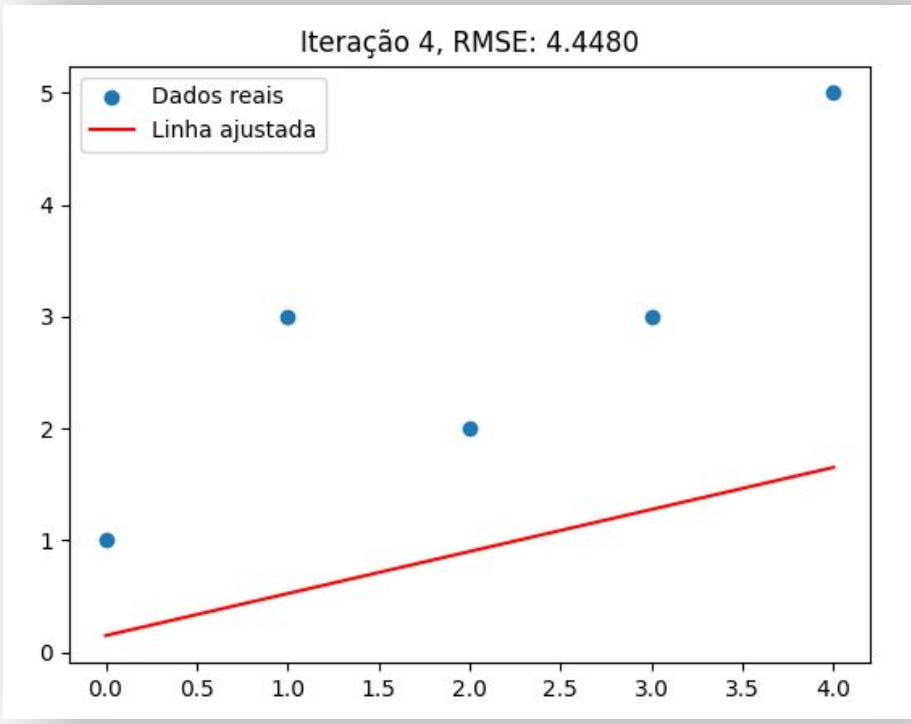



```
db = -  (2 / n) * np.sum(y - y_pred)
```

Mean Square Error (MSE)

- $$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - (mx_i + b))^2$$

- Faça para todas as **variáveis** do seu modelo



Mean Square Error (MSE) para dados "separáveis"

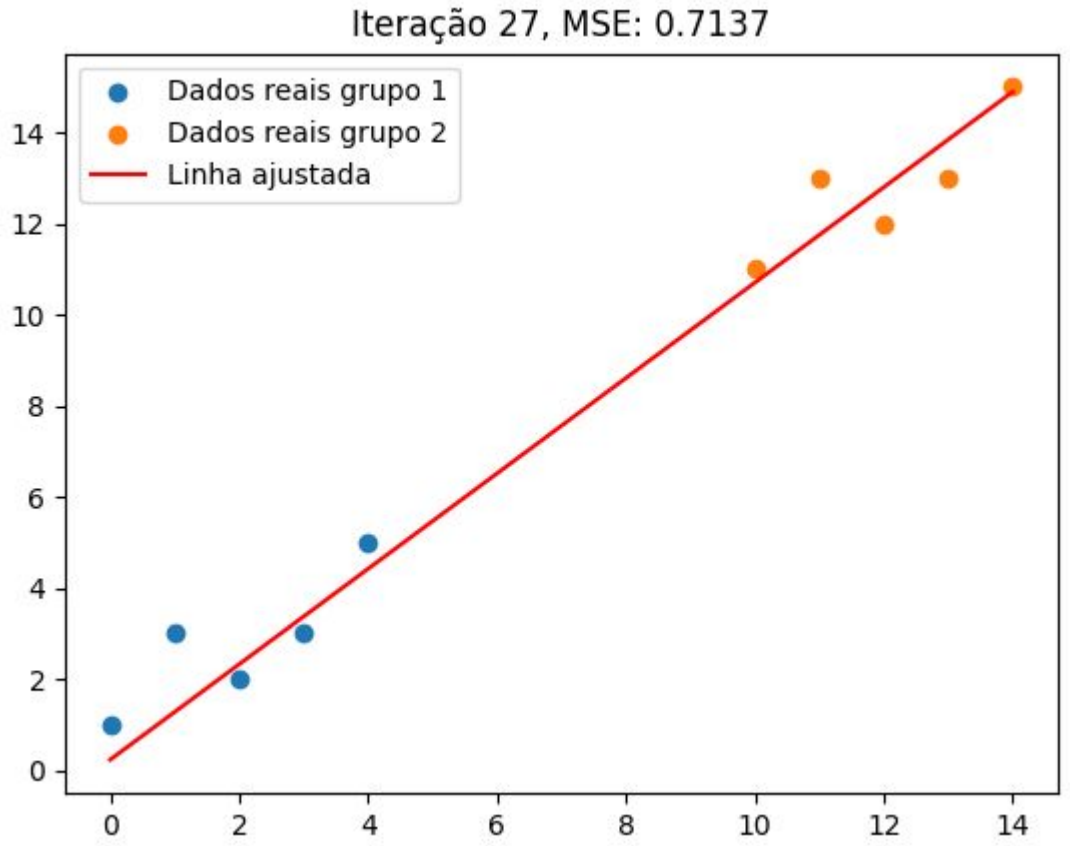
- $$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - (mx_i + b))^2$$

- E se parte dos dados tiverem rótulos diferentes?

```
x1 = np.array([0, 1, 2, 3, 4])
y1 = np.array([1, 3, 2, 3, 5])

x2 = x1 + 10
y2 = y1 + 10
```

- Como Separá-los?

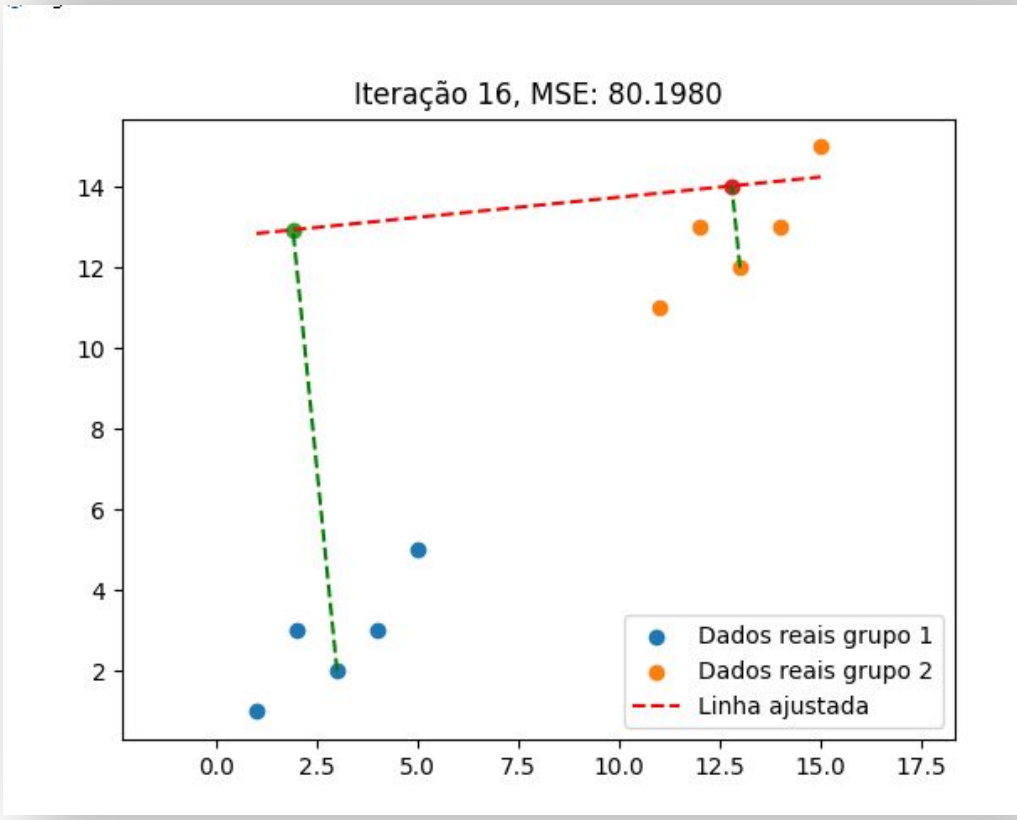


Mudando a interpretação da reta

- O que acontece, em relação aos dados, caso ocorra um ajuste na reta de forma a separar estes mesmos dados?

```
m1 = -(1 / m)
```

- Pré-classificação baseada na comparação do dado em relação a reta

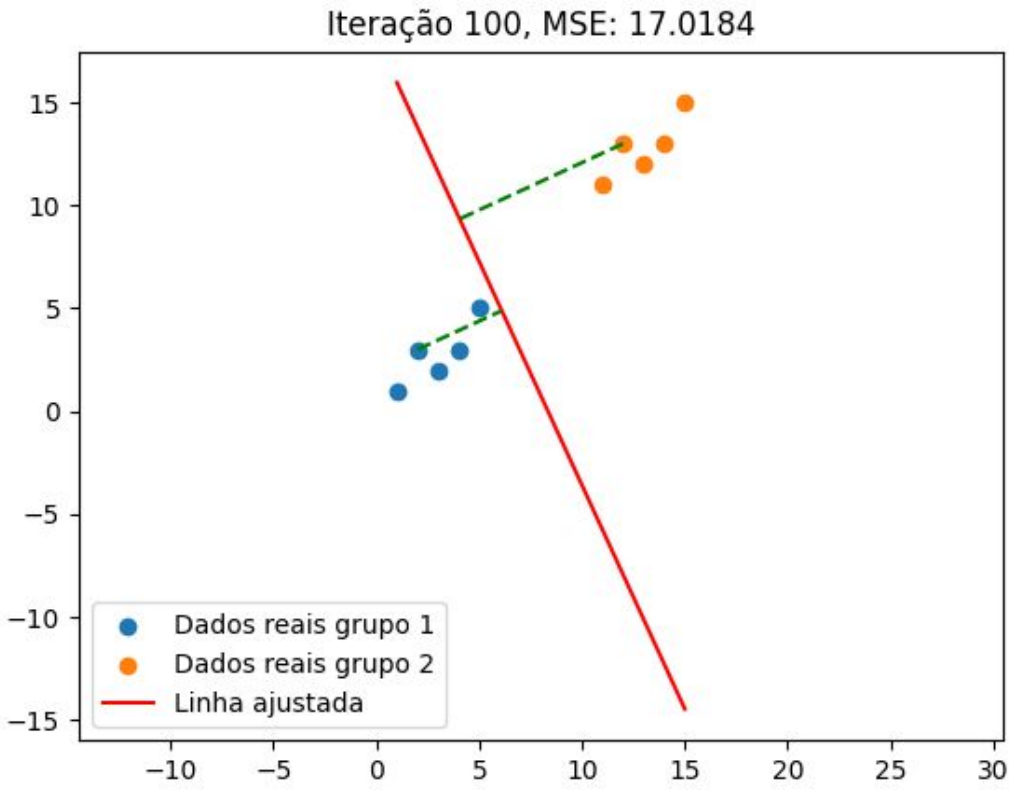


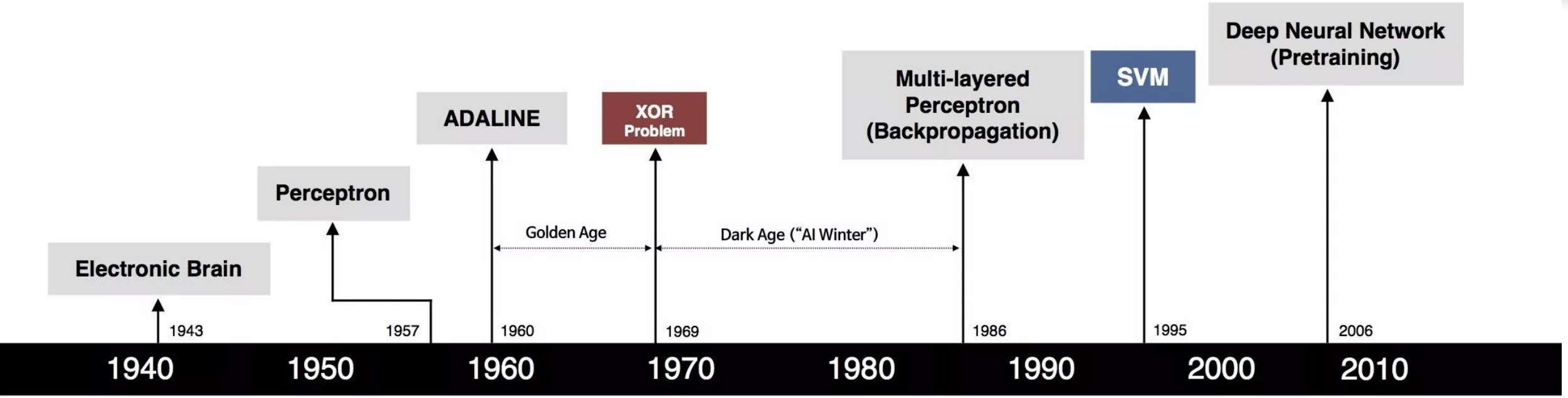
Mudando a interpretação da reta

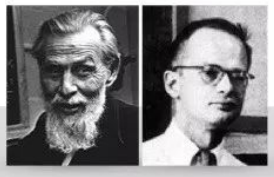
- O que acontece, em relação aos dados, caso ocorra um ajuste na reta de forma a separar estes mesmos dados?

- O que ocorre se subtrairmos esses valores?

Primeiro Grupo	
15.07	1
14.07	2
13.06	3
12.06	4
11.06	5
Segundo Grupo	
5.04	11
4.03	12
3.03	13
2.03	14
1.02	15







S. McCulloch – W. Pitts

F. Rosenblatt

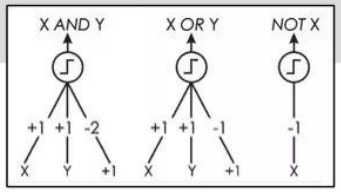
B. Widrow – M. Hoff

M. Minsky – S. Papert

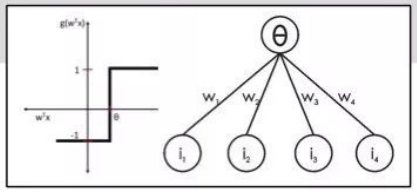
D. Rumelhart – G. Hinton – R. Wiliams

V. Vapnik – C. Cortes

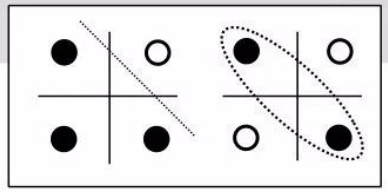
G. Hinton – S. Ruslan



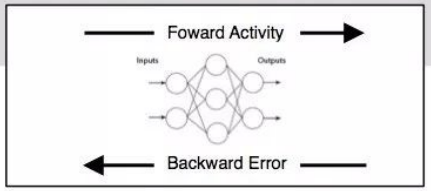
- Adjustable Weights
- Weights are not Learned



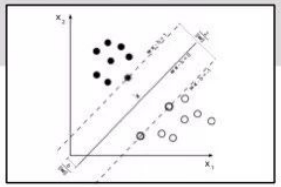
- Learnable Weights and Threshold



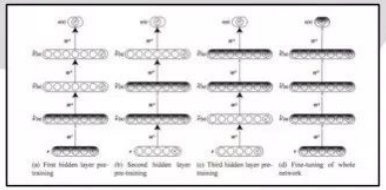
- XOR Problem



- Solution to nonlinearly separable problems
- Big computation, local optima and overfitting



- Limitations of learning prior knowledge
- Kernel function: Human Intervention



- Hierarchical feature Learning

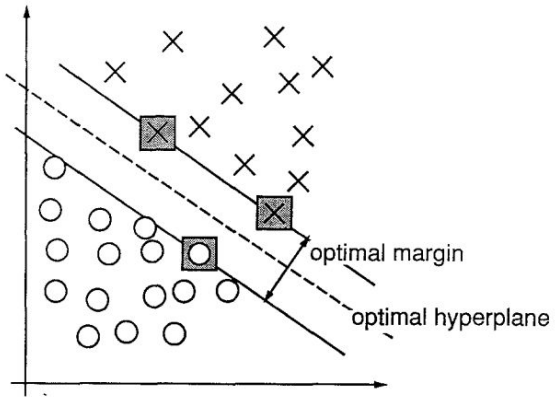


Figure 2. An example of a separable problem in a 2 dimensional space. The support vectors, marked with grey squares, define the margin of largest separation between the two classes.

Machine Learning, 20, 273–297 (1995)

Support-Vector Networks

CORINNA CORTES corinna@neural.att.com
VLADIMIR VAPNIK vlad@neural.att.com
AT&T Bell Labs., Holmdel, NJ 07733, USA

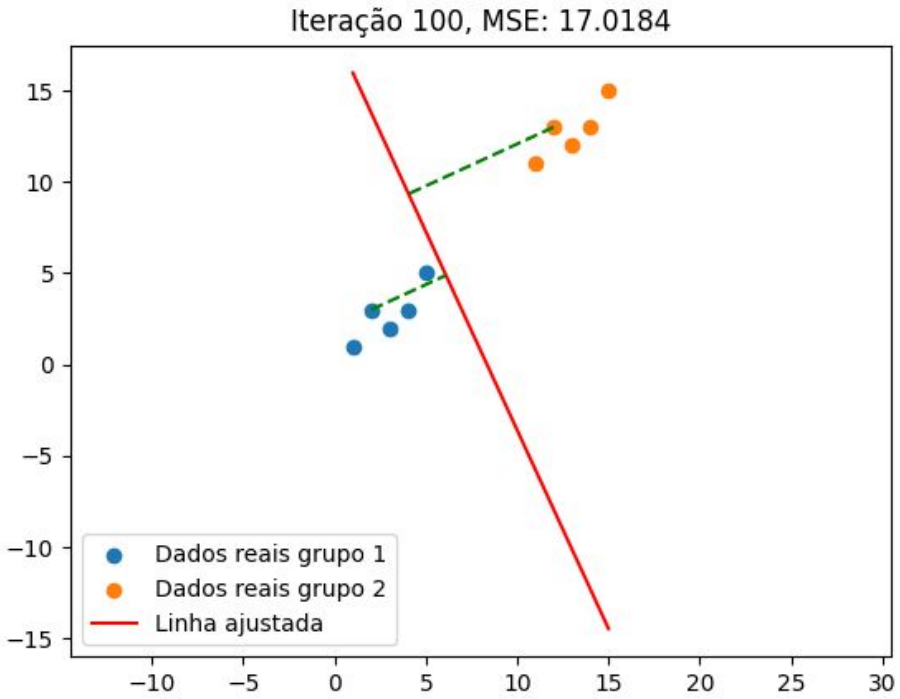
Editor: Lorenza Saitta

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.

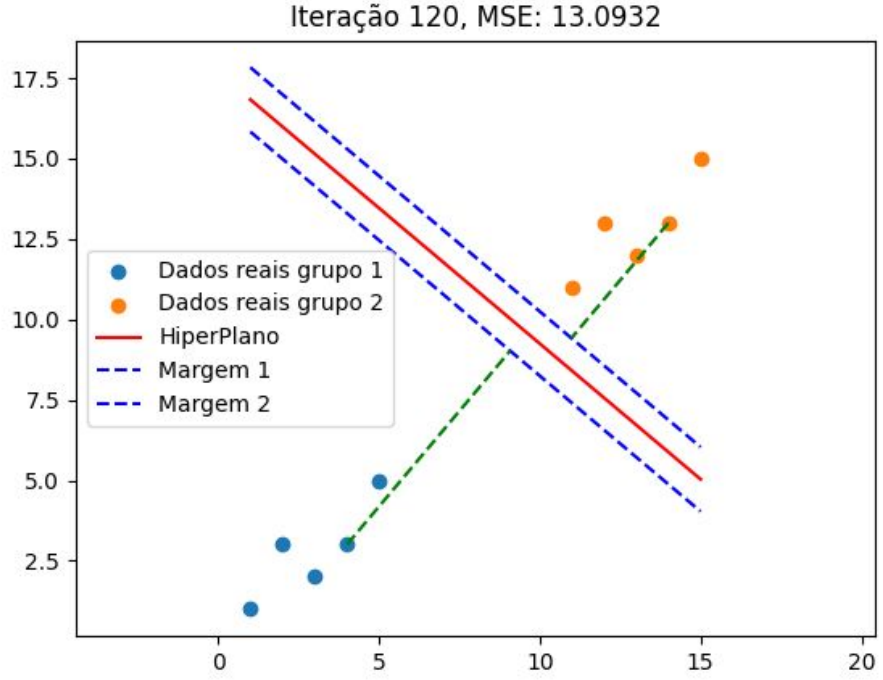
High generalization ability of support-vector networks utilizing polynomial input transformations is demonstrated. We also compare the performance of the support-vector network to various classical learning algorithms that all took part in a benchmark study of Optical Character Recognition.

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

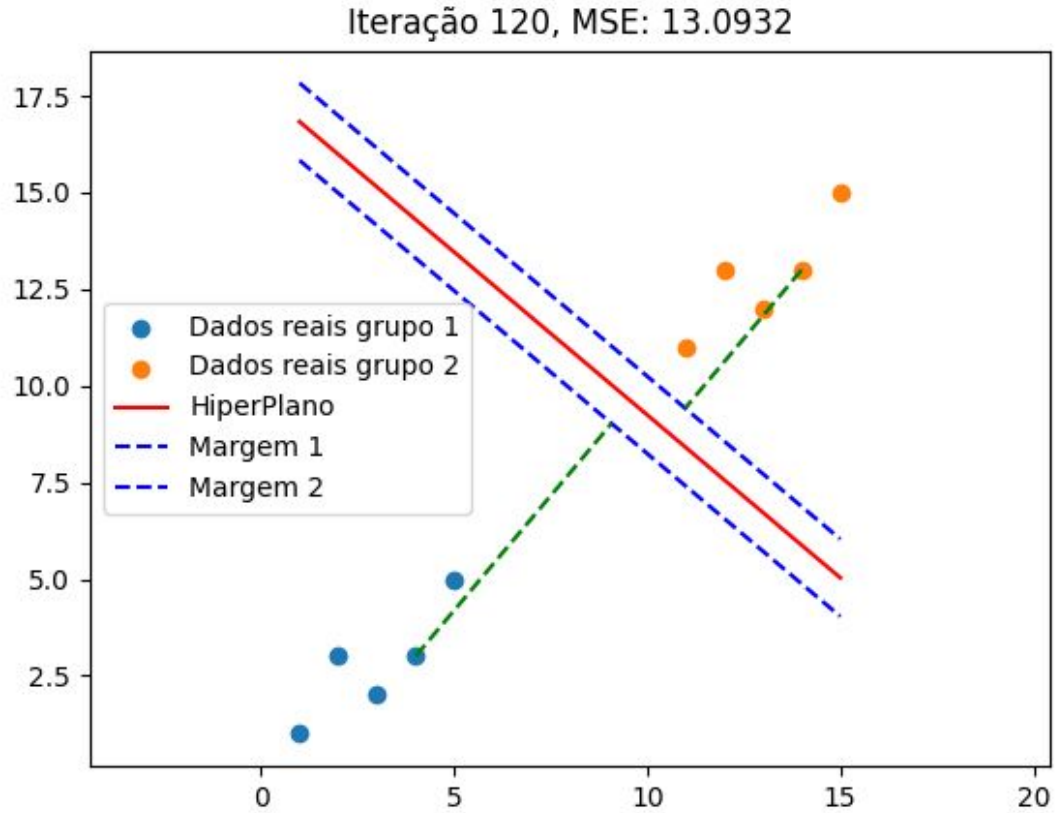
Conceito de Margem

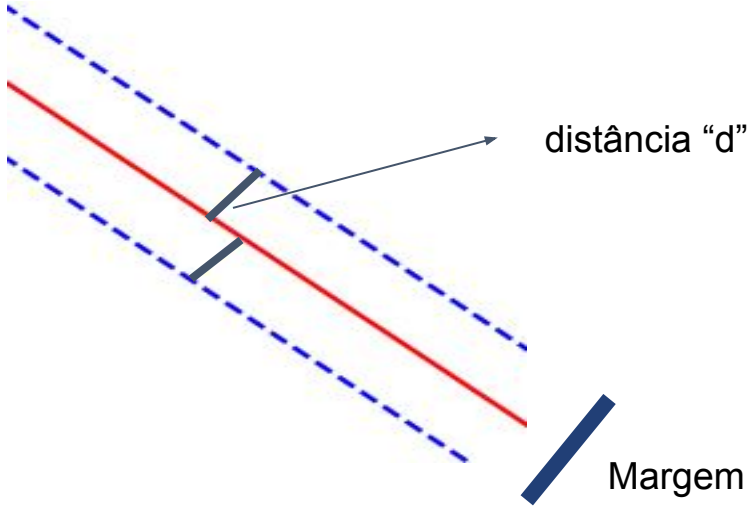


- O que ocorre se adicionarmos uma margem?



Conceito de Margem



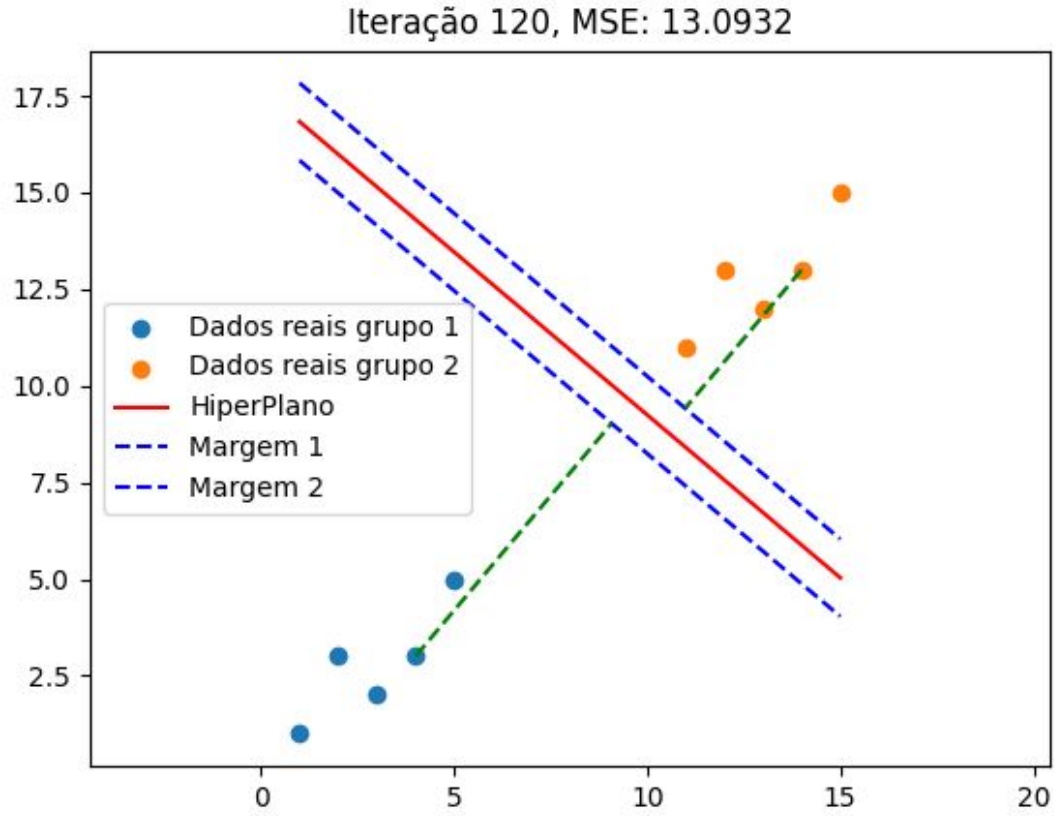


- Fixando o valor para margem. 2 **por convenção**

$$M = 2 / d$$

- Aplicando o conceito de limites, quando que a margem é maior?

Restrições



- Minimização de $\frac{1}{2}\|w\|^2$
- Em relação à $\begin{cases} y_i - \langle w, x_i \rangle - b \leq \varepsilon \\ \langle w, x_i \rangle + b - y_i \leq \varepsilon \end{cases}$

Mais Restrições!

3. The Soft Margin Hyperplane

Consider the case where the training data cannot be separated without error. In this case one may want to separate the training set with a minimal number of errors. To express this formally let us introduce some non-negative variables $\xi_i \geq 0,\ i = 1, \ldots, \ell$.

We can now minimize the functional

$$\Phi(\xi) = \sum_{i=1}^{\ell} \xi_i^{\sigma} \tag{21}$$

for small $\sigma > 0$, subject to the constraints

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \qquad i = 1, \ldots, \ell, \tag{22}$$

$$\xi_i \geq 0, \qquad i = 1, \ldots, \ell. \tag{23}$$

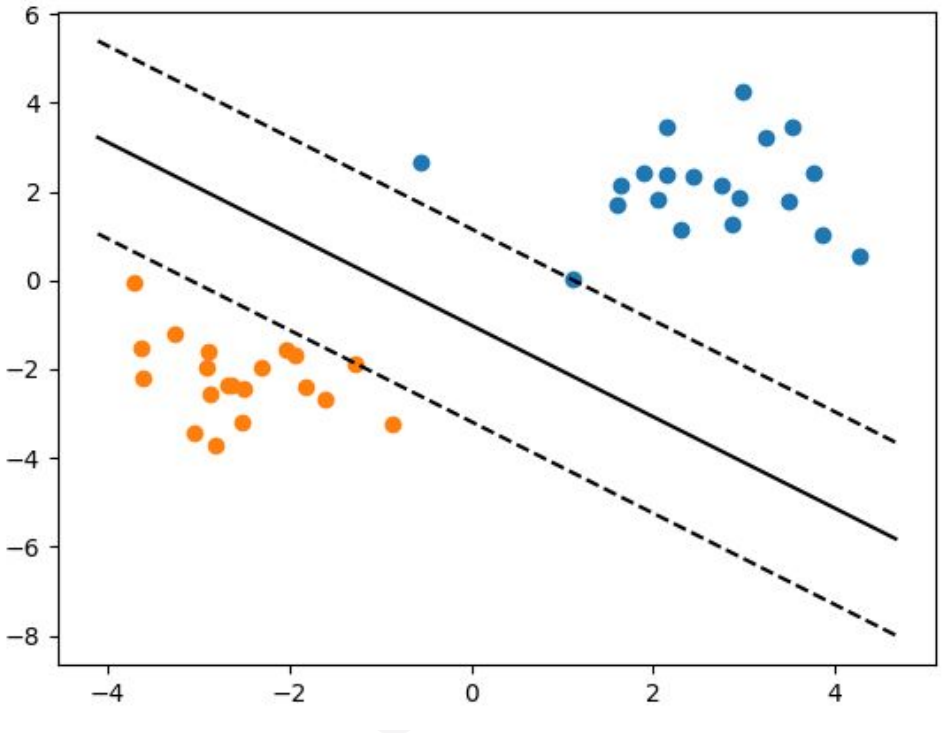
For sufficiently small σ the functional (21) describes the number of the training errors[5].

Minimizing (21) one finds some minimal subset of training errors:

$$(y_{i_1}, \mathbf{x}_{i_1}), \ldots, (y_{i_k}, \mathbf{x}_{i_k}).$$

Mais Restrições!

A constante C > 0 determina a penalidade para os quais ε são tolerados.

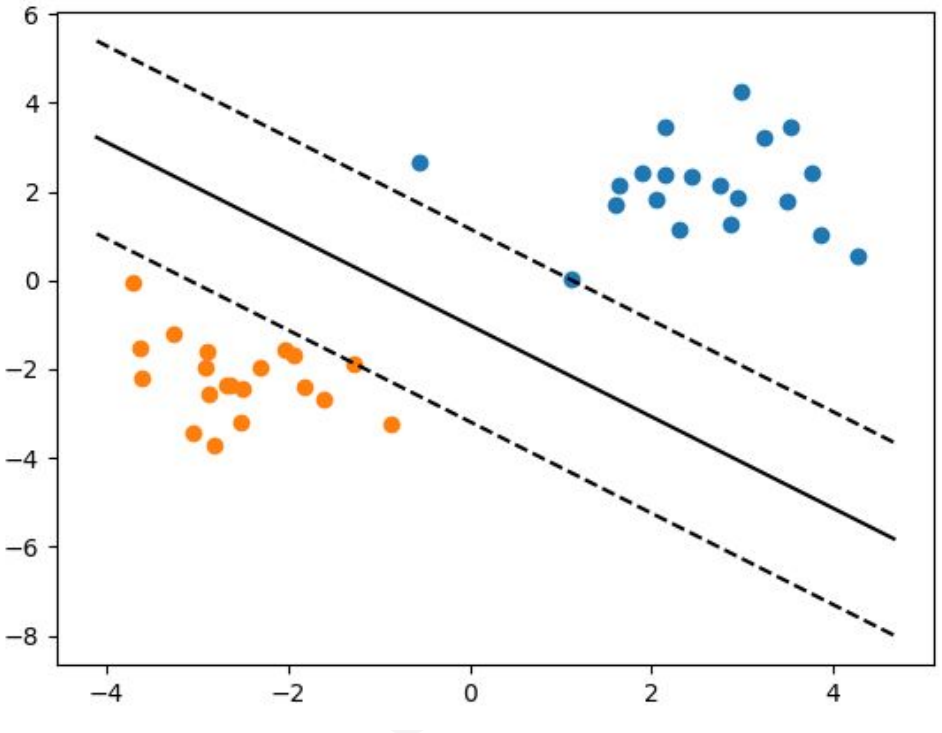


$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*)$$

$$\begin{cases} y_i - \langle w, x_i\rangle - b \le \varepsilon + \xi_i \\ \langle w, x_i\rangle + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0 \end{cases}$$

Ainda **Mais** Restrições! Problema da convexidade

$$L(w, b, \alpha, r) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i[y_i(w \cdot x_i + b) - 1 + \xi_i] - \sum_{i=1}^{n} r_i\xi_i$$



$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = 0$$

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = C - \alpha_i - \beta_i = 0$$

- Mas e a variável “C” ?

Regularização

An Introduction to Active Shape Models *

Tim Cootes

1 Introduction

Biomedical images usually contain complex objects, which will vary in appearance significantly from one image to another. Attempting to measure or detect the presence of particular structures in such images can be a daunting task. The inherent variability will thwart naive schemes. However, by using models which can cope with the variability it is possible to successfully analyse complex images.

Here we will consider a number of methods where the model represents the expected shape and local grey-level structure of a target object in an image.

Model-based methods make use of a prior model of what is expected in the image, and typically attempt to find the best match of the model to the data in a new image. Having matched the model, one can then make measurements or test whether the target is actually present.

This approach is a 'top-down' strategy, and differs significantly from 'bottom-up' methods. In the latter the image data is examined at a low level, looking for local structures such as edges or regions, which are assembled into groups in an attempt to identify objects of interest. Without a global model of what to expect, this approach is difficult and prone to failure.

A wide variety of model based approaches have been explored (see the review below). This chapter will concentrate on a statistical approach, in which a model is built from analysing the appearance of a set of labelled examples. Where

1. Examine a region of the image around each point $\mathbf{X}_i$ to find the best nearby match for the point $\mathbf{X}'_i$
2. Update the parameters $(X_t, Y_t, s, \theta, \mathbf{b})$ to best fit the new found points $\mathbf{X}$
3. Apply constraints to the parameters, $\mathbf{b}$, to ensure plausible shapes (eg limit so $|b_i| < 3\sqrt{\lambda_i}$).
4. Repeat until convergence.

In practice we look along profiles normal to the model boundary through each model point (Figure 9). If we expect the model boundary to correspond to an edge, we can simply locate the strongest edge (including orientation if known) along the profile. The position of this gives the new suggested location for the model point.

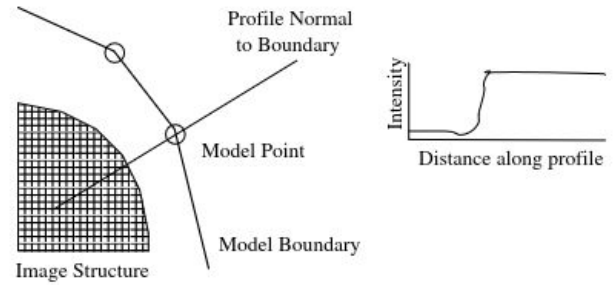
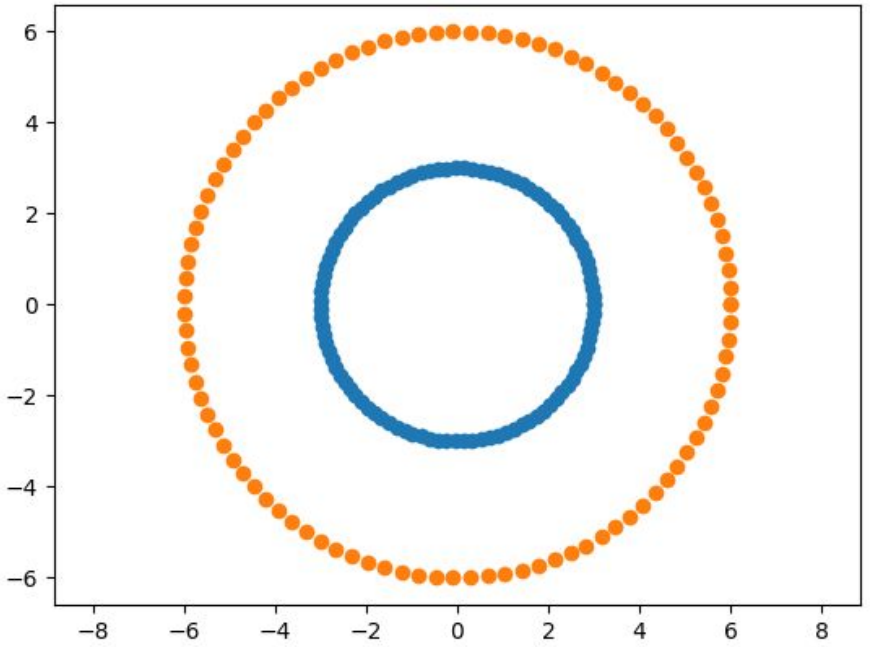


Figure 9: At each model point sample along a profile normal to the boundary

Kernels



1.4.6. Kernel functions

The *kernel function* can be any of the following:

- linear: $\langle x, x' \rangle$.
- polynomial: $(\gamma \langle x, x' \rangle + r)^d$, where d is specified by parameter `degree`, r by `coef0`.
- rbf: $\exp(-\gamma \|x - x'\|^2)$, where γ is specified by parameter `gamma`, must be greater than 0.
- sigmoid $\tanh(\gamma \langle x, x' \rangle + r)$, where r is specified by `coef0`.

Obrigado pela atenção!