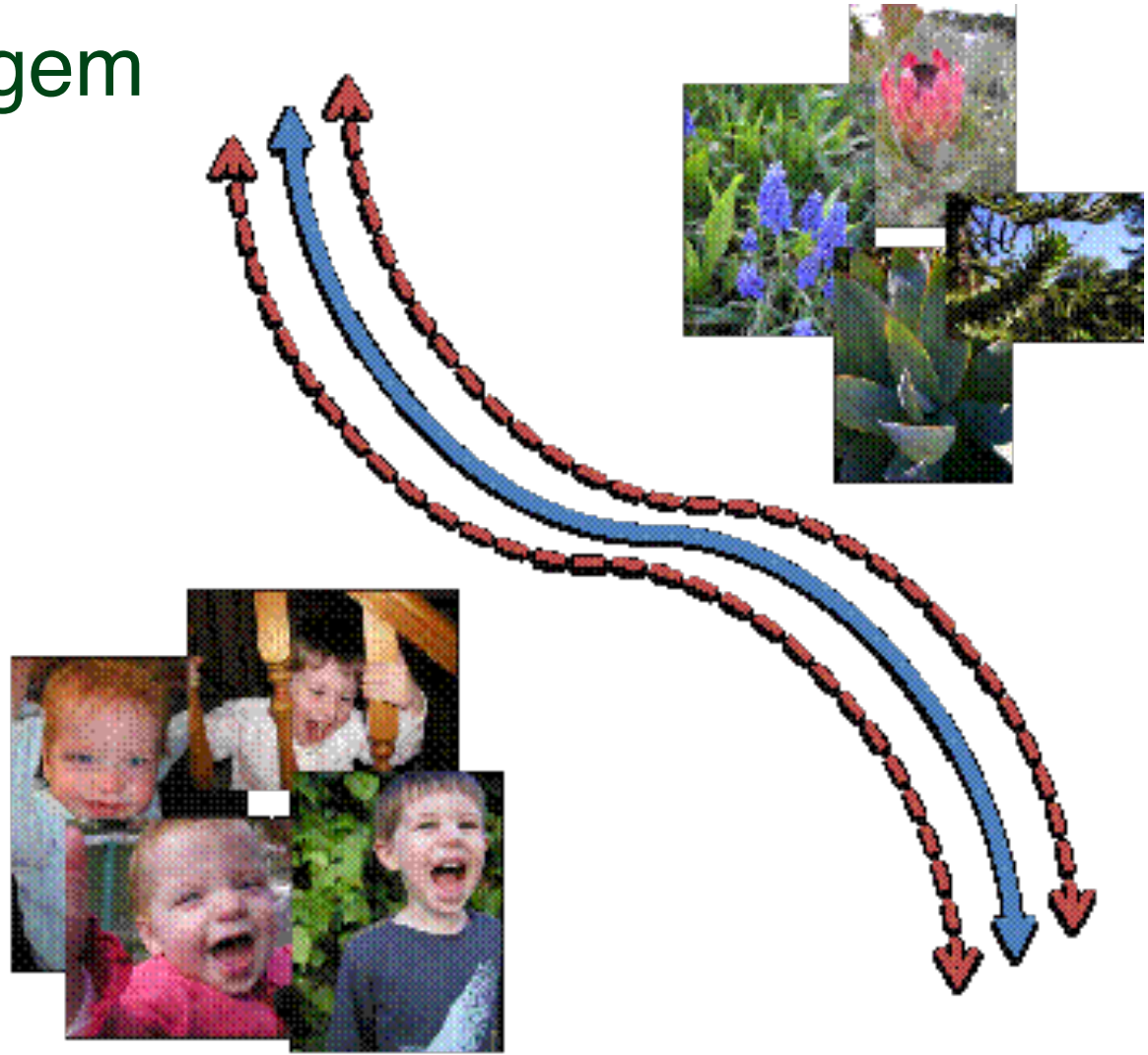
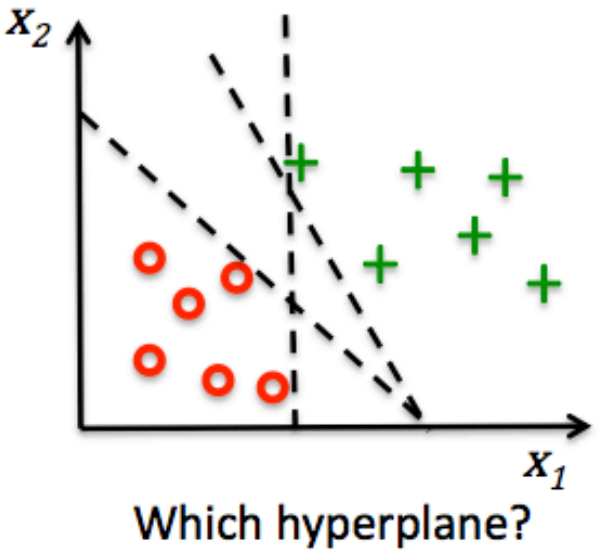
Máquina de Vetores de Suporte (SVM)

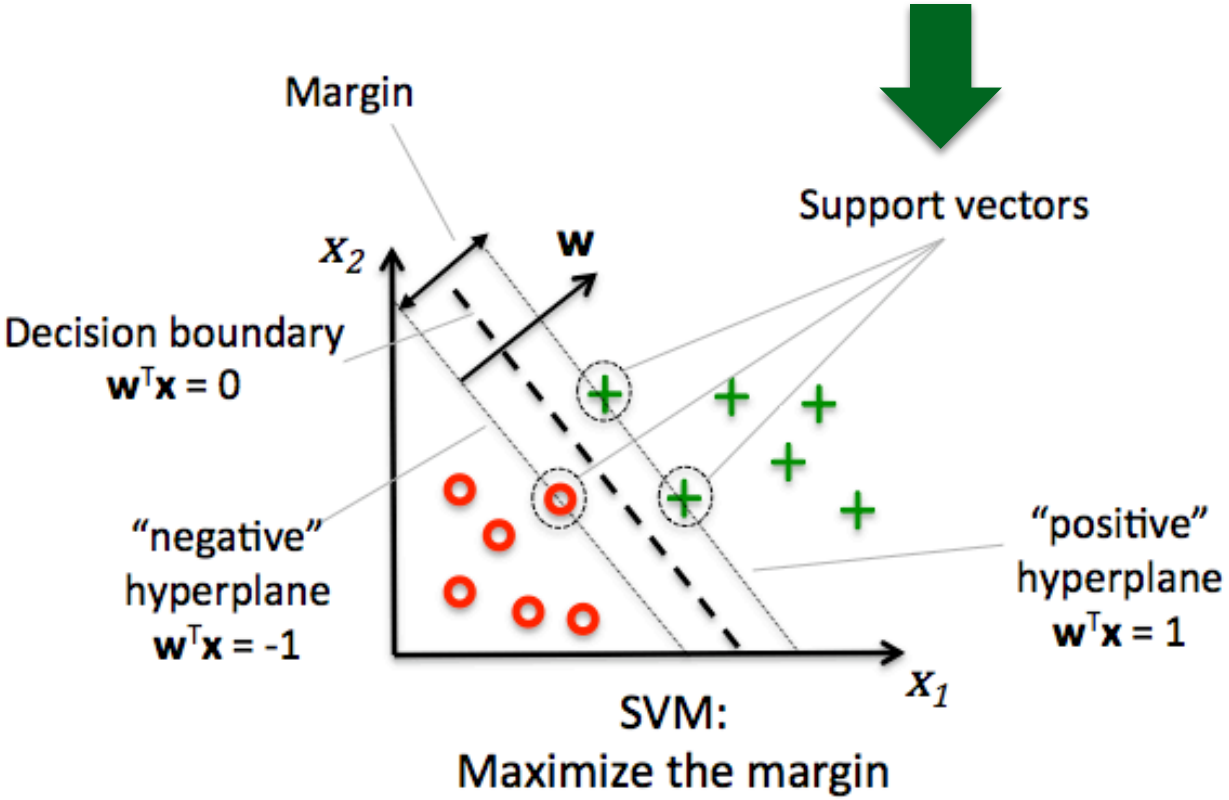
Maximizar a margem de separação



Minimizar erros *vs.* Maximizar a margem

(Perceptron) (SVM)





Fundamentação lógica

- Margens mais largas ➔ Maior poder de generalização ⇔ Menor chance de overfitting

$$w_0 + \mathbf{w}^T \mathbf{x}_{pos} = 1$$

$$w_0 + \mathbf{w}^T \mathbf{x}_{neg} = -1$$

$$\Rightarrow \mathbf{w}^T \left(\mathbf{x}_{pos} - \mathbf{x}_{neg} \right) = 2$$

$$\|\mathbf{w}\| = \sqrt{\sum_{j=1}^{m} w_j^2} \quad \Rightarrow \quad \frac{\mathbf{w}^T (\mathbf{x}_{pos} - \mathbf{x}_{neg})}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|} \quad \Leftrightarrow$$

Forma Primal

$$\text{minimize the reciprocal term } \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{com restrição } y^{(i)} \left(w_0 + \mathbf{w}^T \mathbf{x}^{(i)} \right) \geq 1 \quad \forall_i$$

$$w_0 + \mathbf{w}^T \mathbf{x}^{(i)} \geq 1 \text{ if } y^{(i)} = 1$$

$$w_0 + \mathbf{w}^T \mathbf{x}^{(i)} < -1 \text{ if } y^{(i)} = -1$$

Otimização quadrática via Função Lagrangiana

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b\right) - 1\right)$$

Maximizar

Minimizar

Derivando L (e igualando à 0) em relação à b...

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

...e à w

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

Forma Dual

$$\underset{\boldsymbol{\alpha}}{\text{Maximizar}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right)$$

Com as restrições:

$$\begin{cases} \alpha_i \geqslant 0, \quad \forall i = 1, \ldots, n \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases}$$

Soft-margin

- Restrições lineares precisam ser afrouxadas para permitir convergência sob situações onde **nem todos os exemplos estão perfeitamente** separados por hiperplanos

$$w_0 + \mathbf{w}^T \mathbf{x}^{(i)} \geq 1 \text{ if } y^{(i)} = 1$$

$$w_0 + \mathbf{w}^T \mathbf{x}^{(i)} < -1 \text{ if } y^{(i)} = -1$$

$\Rightarrow$

Variável slack

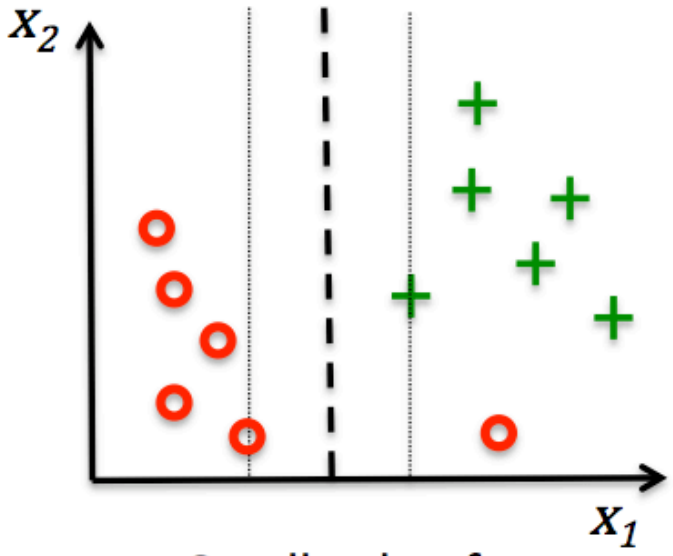
$$\mathbf{w}^T \mathbf{x}^{(i)} \geq 1 - \xi^{(i)} \text{ if } y^{(i)} = 1$$

$$\mathbf{w}^T \mathbf{x}^{(i)} < -1 + \xi^{(i)} \text{ if } y^{(i)} = -1$$

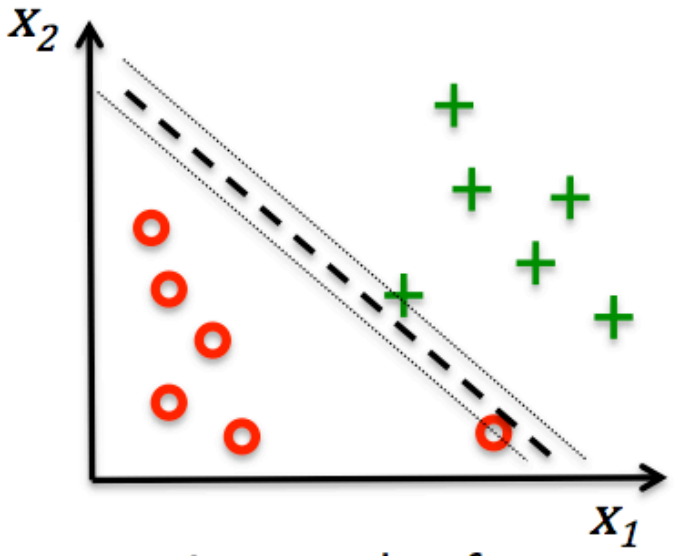
*C*ontrole da penalidade da classificação errada

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\left(\sum_i \xi^{(i)}\right)$$

Aumenta *C*, aumenta a penalidade do erro



Small value for parameter *C*



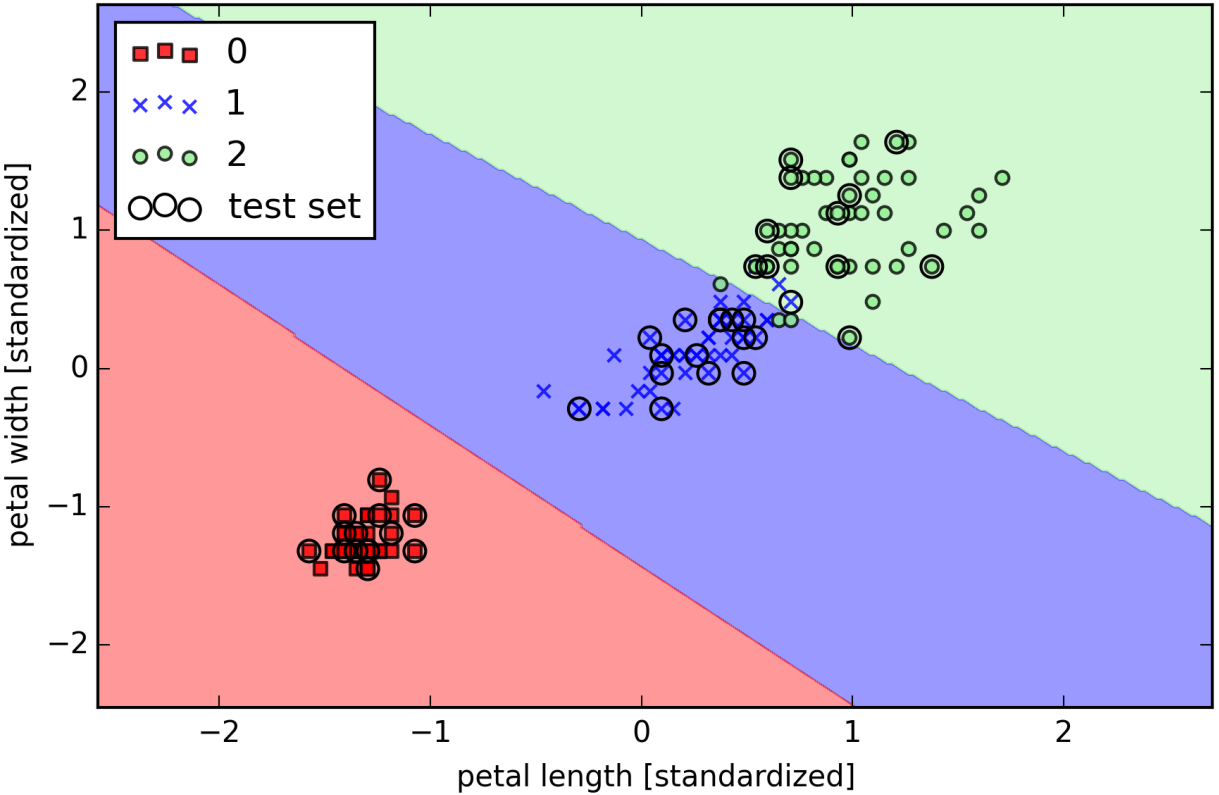
Large value for parameter *C*

Aumenta o viés e diminui a variância do modelo

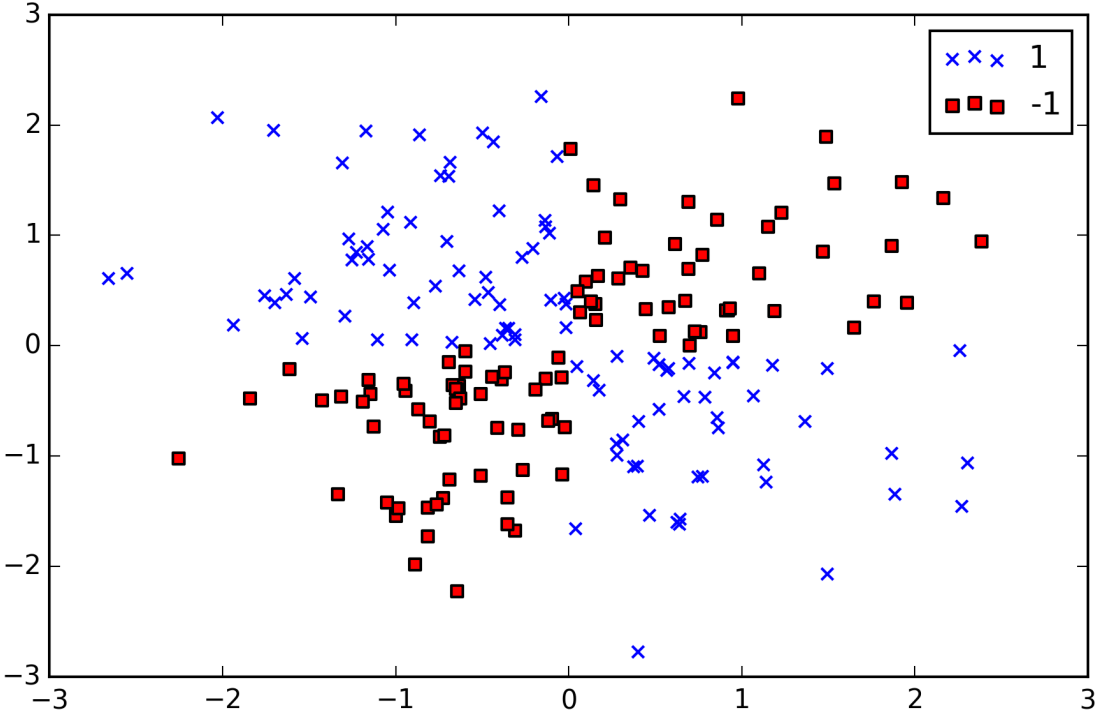
Treinamento de um SVM linear

```
from sklearn.svm import SVC

svm = SVC(kernel='linear', C=1.0, random_state=0)
svm.fit(X_train_std, y_train)

plot_decision_regions(X_combined_std, y_combined,
                      classifier=svm, test_idx=range(105, 150))
plt.xlabel('petal length [standardized]')
plt.ylabel('petal width [standardized]')
plt.legend(loc='upper left')
plt.tight_layout()
# plt.savefig('./figures/support_vector_machine_linear.png', dpi=300)
plt.show()
```

Treinamento de um SVM linear



Reconsidere o problema da porta lógica XOR



Não dá pra separar via hiperplano linear!

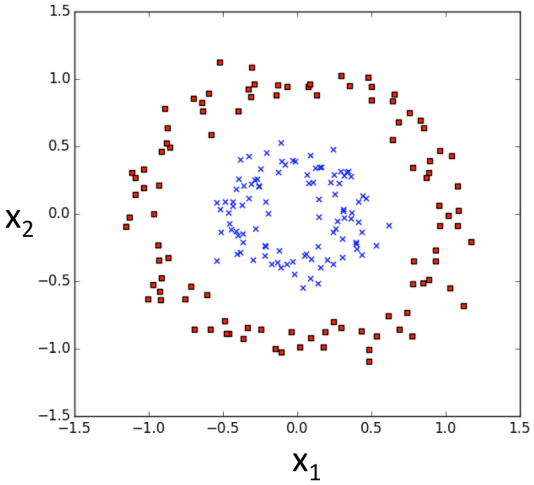
Como lidar com a não separatividade linear?

- Criar combinações não lineares das características originais e projetá-las num espaço de maior dimensionalidade

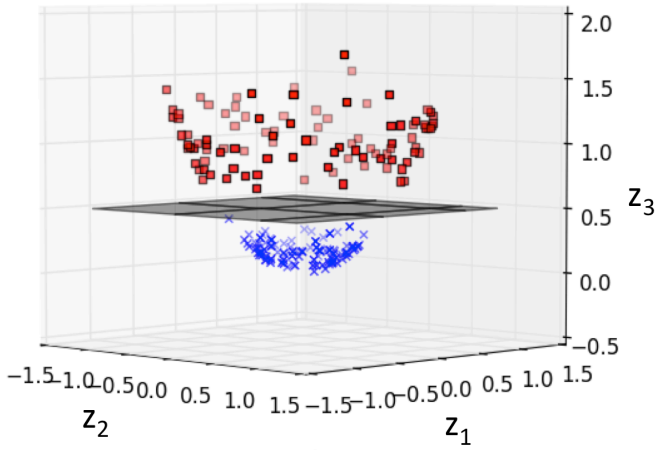
$$\phi(x_1, x_2) = (z_1, z_2, z_3) = (x_1, x_2, x_1^2 + x_2^2)$$

Função de mapeamento

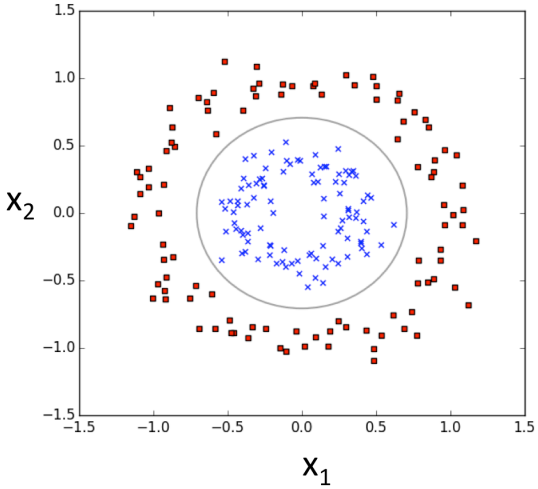
$$\phi\left(\mathbf{x}^{(i)}\right)$$


1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5 −1.0 −0.5 0.0 0.5 1.0 1.5
x_2
x_1

ϕ


2.0
1.5
1.0
0.5
0.0
−0.5
z_3
−1.5 −1.0 −0.5 0.0 0.5 1.0 1.5
z_2
−1.5 −1.0 −0.5 0.0 0.5 1.0 1.5
z_1

ϕ^{-1}


1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5 −1.0 −0.5 0.0 0.5 1.0 1.5
x_2
x_1

$$\underset{\boldsymbol{\alpha}}{\text{Maximizar}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right)$$

$\mathbf{x}^{(i)\,T}\mathbf{x}^{(j)}$

Bem custoso!

grande sacada!!!

Função kernel

$$k\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right) = \phi\left(\mathbf{x}^{(i)}\right)^T \phi\left(\mathbf{x}^{(j)}\right)$$

Produto escalar de dois pontos no novo espaço

Simplicidade do cálculo

Radial Basis Function (RBF) kernel

$$k\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right) = \phi\left(\mathbf{x}^{(i)}\right)^T \phi\left(\mathbf{x}^{(j)}\right)$$

$$k\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right) = \exp\left(- \frac{\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2}{2\sigma^2} \right)$$

$$k\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right) = \exp\left(- \gamma \, \|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2 \right)$$

$$\gamma = \frac{1}{2\sigma^2}$$

Classes de funções kernel

Tipo de kernel	**Função** $K(\mathbf{x}_i, \mathbf{x}_j)$	**Parâmetros**
Polinomial	$(\delta(\mathbf{x}_i \cdot \mathbf{x}_j) + \kappa)^d$	δ, κ e d
RBF	$\exp\left(-\gamma \lVert \mathbf{x}_i - \mathbf{x}_j \rVert^2\right)$	γ
Sigmoidal	$\tanh(\delta(\mathbf{x}_i \cdot \mathbf{x}_j) + \kappa)$	δ e κ

Q: Que tipo de kernel teríamos se δ, κ e d

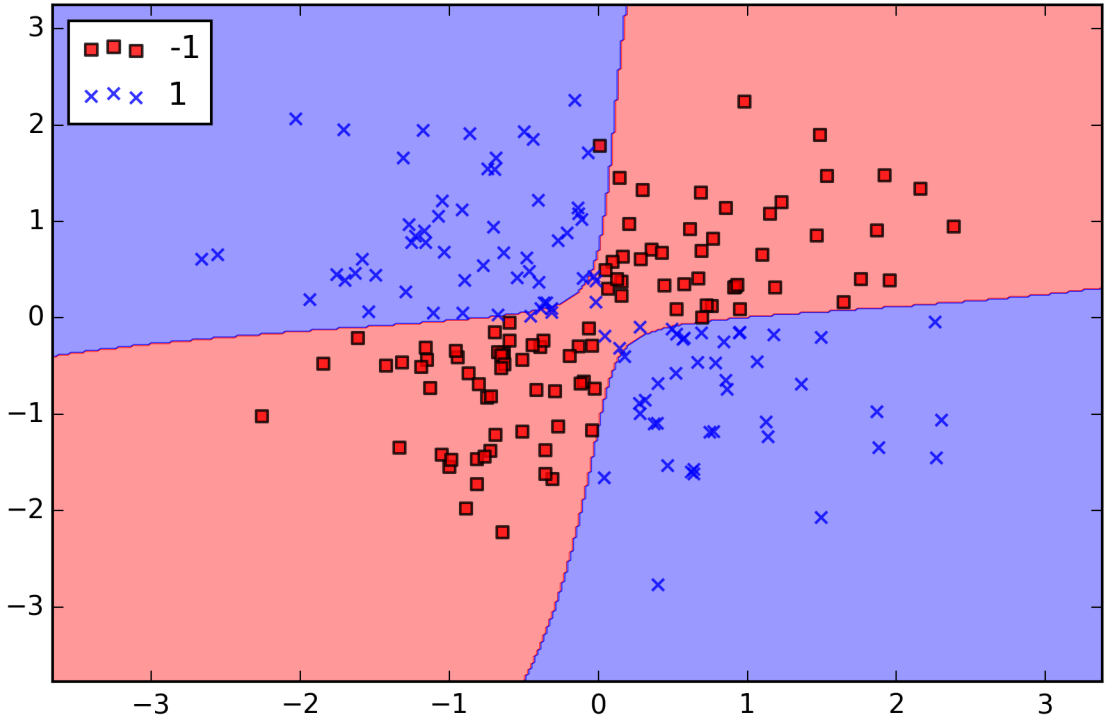
1 0 1

em:

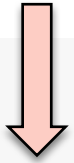
Polinomial	$(\delta(\mathbf{x}_i \cdot \mathbf{x}_j) + \kappa)^d$	δ, κ e d

Treinamento de um SVM RBF

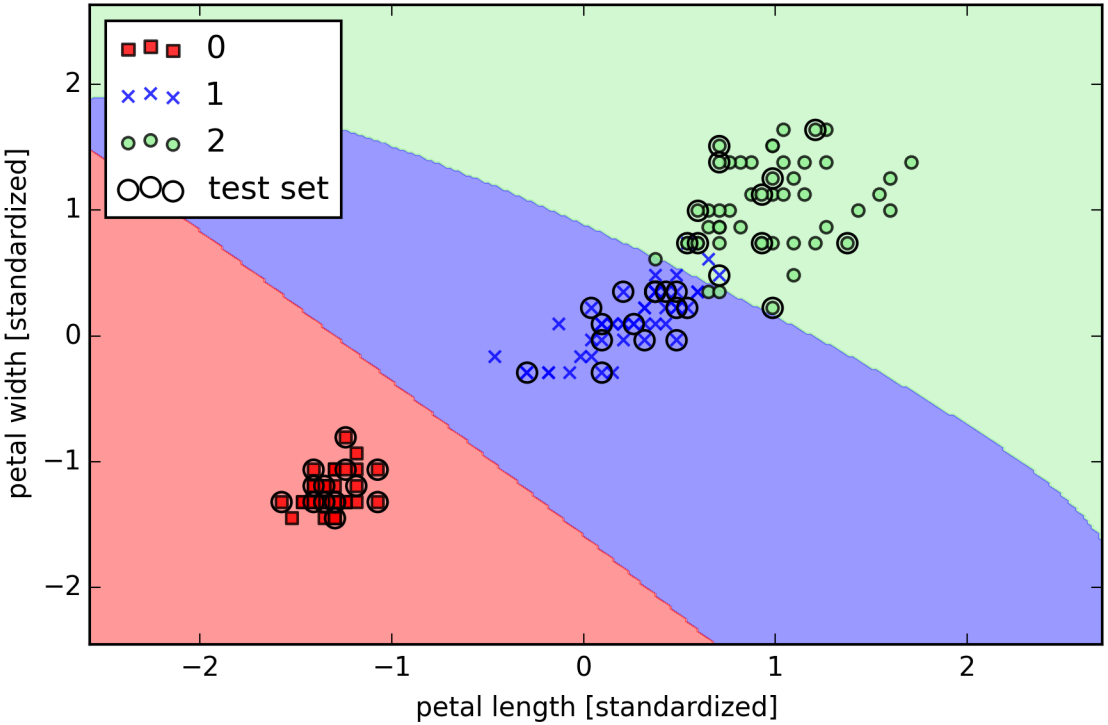
```
svm = SVC(kernel='rbf', random_state=0, gamma=0.10, C=10.0)
svm.fit(X_xor, y_xor)
plot_decision_regions(X_xor, y_xor,
                      classifier=svm)

plt.legend(loc='upper left')
plt.tight_layout()
# plt.savefig('./figures/support_vector_machine_rbf_xor.png', dpi=300)
plt.show()
```


-1
1
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3

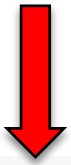
SVM RBF para dataset Íris



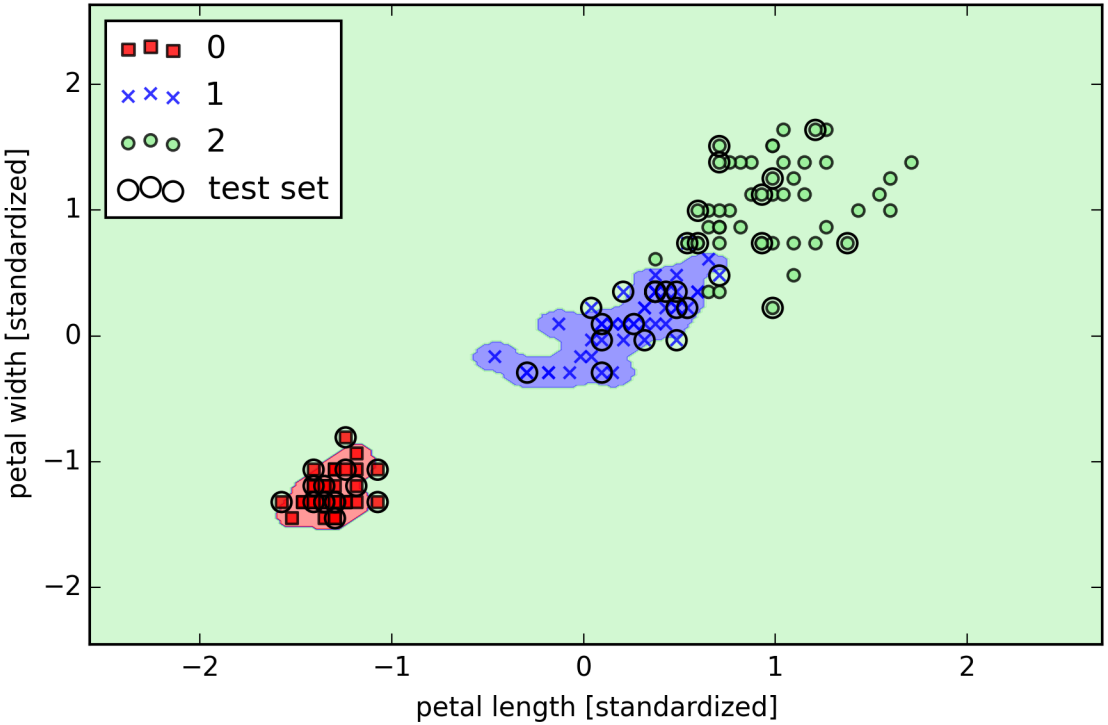
```
from sklearn.svm import SVC

svm = SVC(kernel='rbf', random_state=0, gamma=0.2, C=1.0)
svm.fit(X_train_std, y_train)

plot_decision_regions(X_combined_std, y_combined,
                      classifier=svm, test_idx=range(105,150))
plt.xlabel('petal length [standardized]')
plt.ylabel('petal width [standardized]')
plt.legend(loc='upper left')
plt.tight_layout()
# plt.savefig('./figures/support_vector_machine_rbf_iris_1.png', dpi=300)
plt.show()
```


0
1
2
test set
petal width [standardized]
petal length [standardized]
2
1
0
−1
−2
−2
−1
0
1
2

SVM RBF para dataset Íris



```
svm = SVC(kernel='rbf', random_state=0, gamma=100.0, C=1.0)
svm.fit(X_train_std, y_train)

plot_decision_regions(X_combined_std, y_combined,
                      classifier=svm, test_idx=range(105, 150))
plt.xlabel('petal length [standardized]')
plt.ylabel('petal width [standardized]')
plt.legend(loc='upper left')
plt.tight_layout()
# plt.savefig('./figures/support_vector_machine_rbf_iris_2.png', dpi=300)
plt.show()
```


0
1
2
test set
petal width [standardized]
petal length [standardized]
2
1
0
−1
−2
−2
−1
0
1
2

Q: Fator γ contribui para solução do problema do overfitting/underfitting?

a. Sim
b. Não

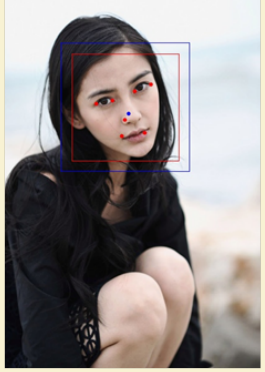
Q: De que forma o fator γ contribui para solução do problema do overfitting/underfitting?

a. Se aumentarmos seu valor, aumentamos a variância do modelo
b. Se aumentarmos seu valor, aumentamos o viés do modelo
c. Se diminuirmos seu valor, aumentamos o viés do modelo
d. Se diminuirmos seu valor, aumentamos a variância do modelo
e. Letras b e d estão corretas
f. Letras a e c estão corretas
g. N.d.a

Unconstrained face recognition

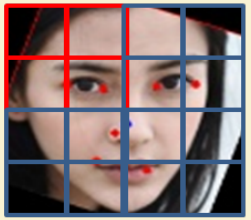
https://github.com/t0nyren/welfare/wiki

Unconstrained photo

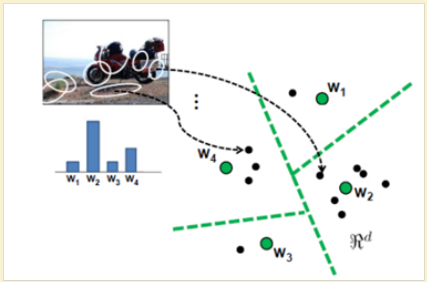


Face & Landmark detection

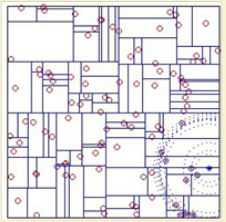
Alignment



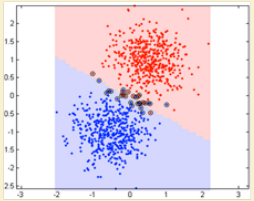
Representation



Encoding



Approximate KNN



SVM Classification

Divulgação científica

saense

http://www.saense.com.br
https://www.facebook.com/saense/

Hendrik Macedo
Escreve sobre Inteligência Artificial no Saense.
http://www.saense.com.br/autores/artigos-publicados-por-hendrik-macedo/