Introdução às Redes Neurais Artificiais

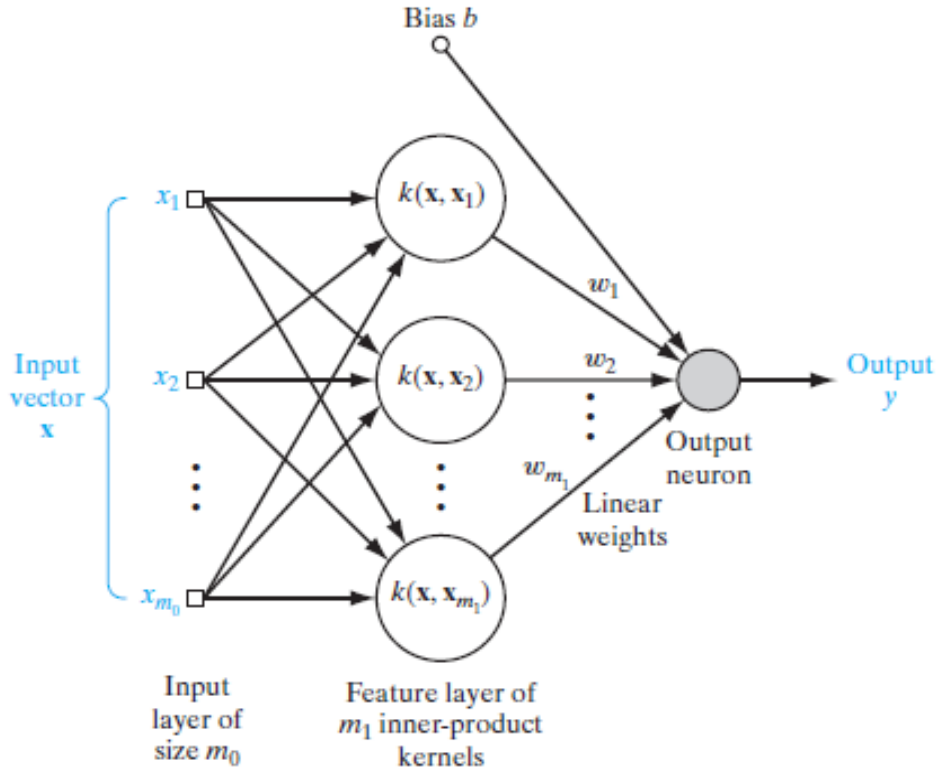
Máquinas de Vetores de Suporte

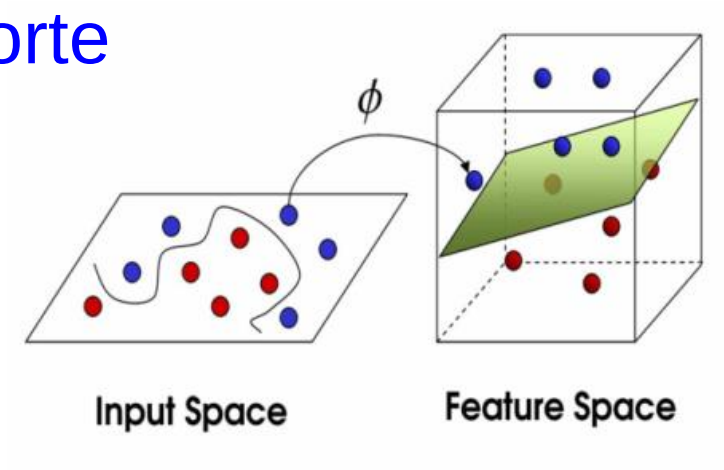
Prof. João Marcos Meirelles da Silva
jmarcos@id.uff.br

Universidade Federal Fluminense

www.latelco.uff.br
www.professores.uff.br/jmarcos

Máquinas de Vetores de Suporte

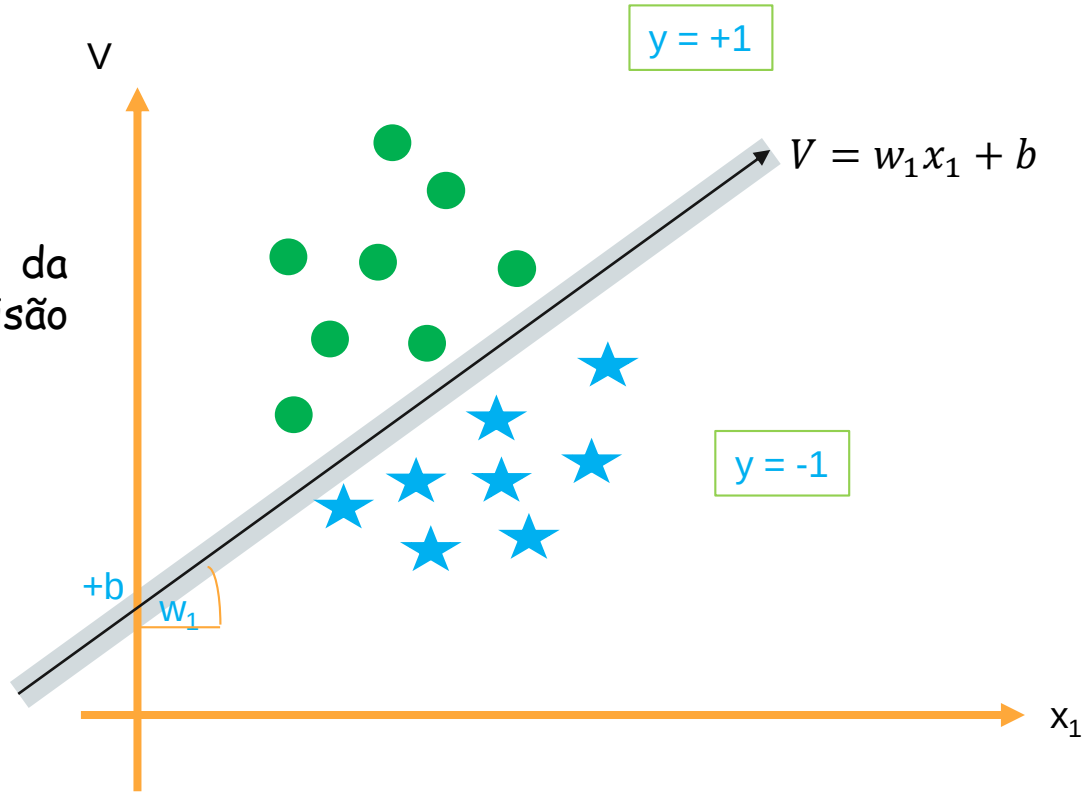




Tipo de rede neural artificial usada para tarefas de classificação, regressão e detecção de anomalias.

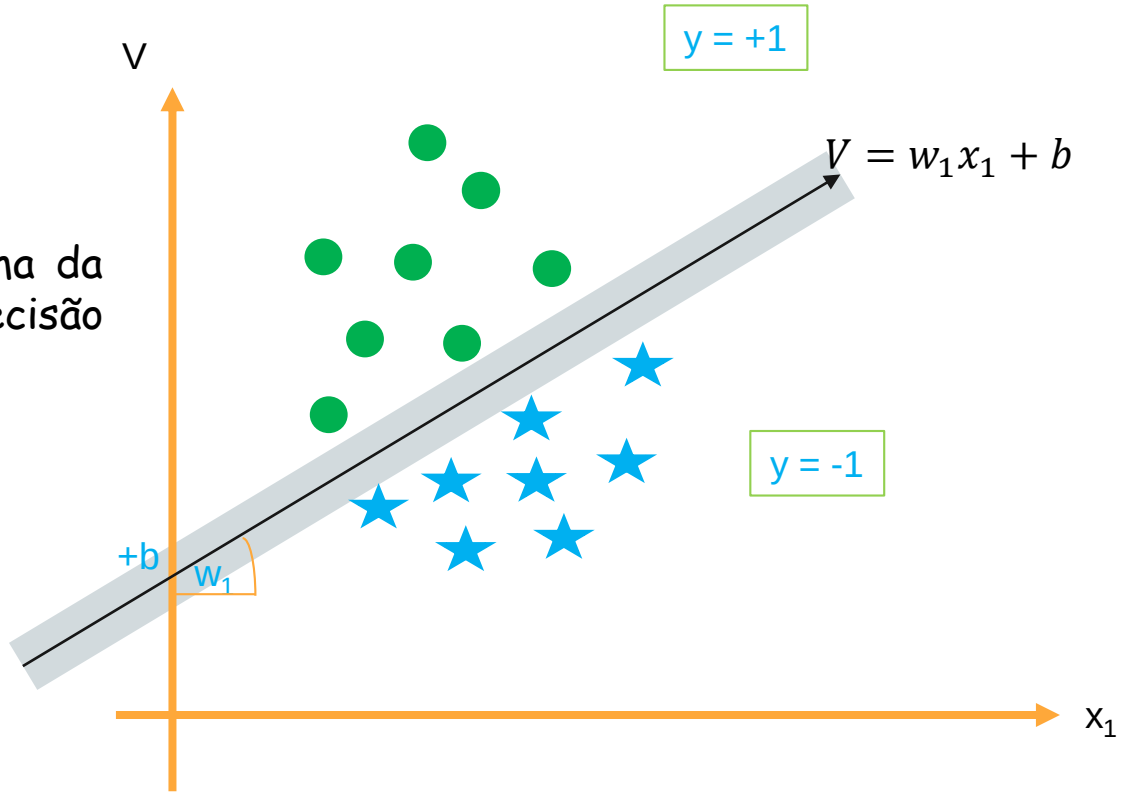
Margem do Classificador

Definida como a largura máxima da área em torno da reta de decisão antes de tocar um dado.



Margem do Classificador

Definida como a largura máxima da área em torno da reta de decisão antes de tocar um dado.



Pros e Cons

PROS:

- Efetivo em espaços de altas dimensões
- Efetivo quando o número de dimensões é maior que o número de amostras
- Utiliza apenas poucos pontos do conjunto de treinamento (chamados de vetores de suporte) $\Rightarrow$ Eficiência no uso de memória!
- Versatilidade: Possibilidade de usar diferentes funções de ativação (mais conhecidas como *funções de Kernel*)

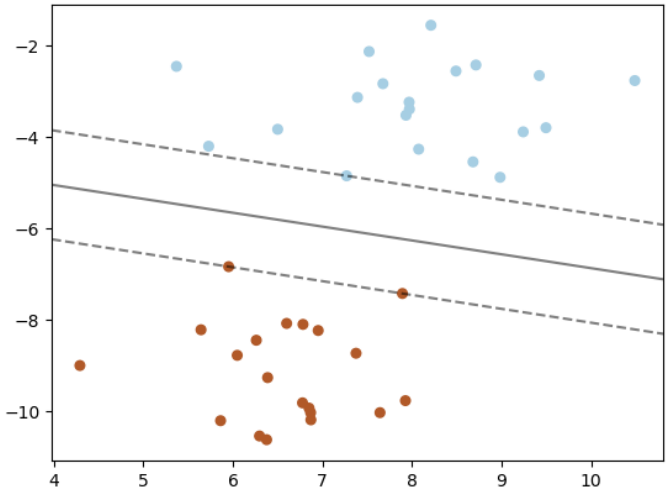
Pros e Cons

CONS:

- Escolha do tipo de função Kernel e parâmetro de regularização são cruciais quando o número de atributos for maior que o número de amostras
- SVMs não provêem estimativas de probabilidades, necessitando de algoritmos auxiliares para isso.

Formulação Matemática

Uma rede SVM constrói um hiperplano, ou um conjunto de hiperplanos, em um espaço dimensional grande, que pode ser usada para problemas de classificação, regressão (predição de séries), detecção de *outliers* e clustering.



Intuitivamente, uma boa separação é obtida pelo hiperplano que maximiza a distância aos pontos de dados de treinamento mais próximos de qualquer classe (máxima margem).

Formulação Matemática

A função mais emprega para otimização é a “*Hinge Loss*”:

$$c(x, y, f(x)) = (1 - y * f(x))_+$$

onde c é a função de erro, x é a amostra, y é a classe correta e f(x) é a classe predita.

Isto significa que:

$$c(x, y, f(x)) = \begin{cases} 0, & se\ y * f(x) \geq 1 \\ 1 - y * f(x), & caso\ contrário \end{cases}$$

Formulação Matemática

A função objetivo é dada por:

$$min_w \, \lambda \|w\|^2 + \sum_{i=1}^{n} (1 - y_i \langle x_i, w \rangle)_+$$

onde c é a função de erro, x é a amostra, y é a classe correta e f(x) é a classe predita.

Como otimizar a função objetivo (isto,é: aprender)?

Temos que derivar a F.O. para obter os gradientes!

Formulação Matemática

$$\frac{\partial}{\partial w_k}\lambda\|w\|^2 = 2\lambda w_k$$

$$\frac{\partial}{\partial w_k}(1 - y_i\langle x_i, w\rangle)_+ = \begin{cases} 0, & se\ y_i\langle x_i, w\rangle \geq 1 \\ -y_i x_{ik}, & caso\ contário \end{cases}$$

Atualização de pesos:

$y_i\langle x_i, w\rangle < 1$ ⇨ $w = w + \eta(y_i x_i - 2\lambda w)$ (erro de classificação)

$y_i\langle x_i, w\rangle \geq 1$ ⇨ $w = w + \eta(-2\lambda w)$ (acerto de classificação)

Formulação Matemática

O termo de regularização controla o balanço entre um pequeno erro de treinamento e um pequeno erro de teste, o que corresponde a sua capacidade de generalização:

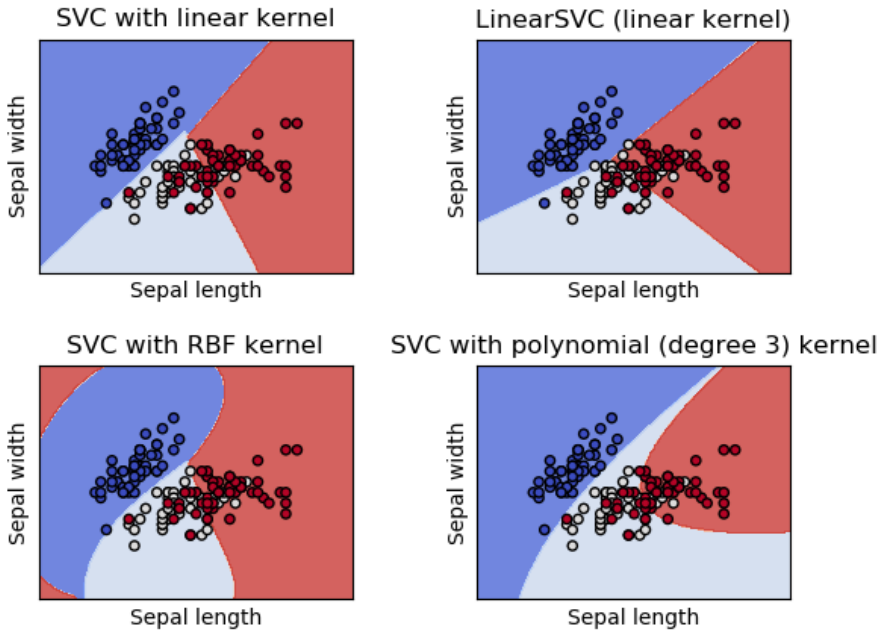
- Regularização muito alta ? *Overfiting* (grande erro no teste)
- Regularização muito baixa? *Underfitting* (grande erro de treinamento)

Podemos fazer $\lambda = 1/Epoca$, assim esse parâmetro diminui conforme o contador Epoca aumenta.

Scikit Learn - Classificador SVM

Classe sklearn.svm é composta pelos seguintes classificadores:

1. SVC
2. NuSVC
3. LinearSVC



Classificador SVM

Classe sklearn.SVM

Implementa redes de máquinas de vetores de suporte.

Entrada: Array $X_{(n\_amostras,\ n\_atributos)}$

Saída: Array $Y_{(n\_amostras)}$ → Rótulos de classes

Classificador SVM

```
>>> from sklearn import svm
>>> X = [[0, 0], [1, 1]]
>>> y = [0, 1]
>>> clf = svm.SVC()
>>> clf.fit(X, y)
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

Classificador SVM

Após o treinamento, a rede pode ser utilizada para predizer novos valores:

```
>>> clf.predict([[2., 2.]])
array([1])
```

Classificador SVM

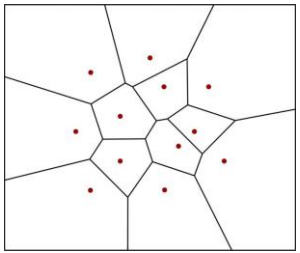
Algumas propriedades dos vetores de suporte podem ser analisados pelas funções membro da classe: *support_vectors_*, *support_* e *n_support*:

```
>>> # get support vectors
>>> clf.support_vectors_
array([[ 0.,  0.],
       [ 1.,  1.]])
>>> # get indices of support vectors
>>> clf.support_
array([0, 1]...)
>>> # get number of support vectors for each class
>>> clf.n_support_
array([1, 1]...)
```

Classificador SVM Multiclasses

SVC e NuSVC implementam a estratégia "1 contra 1" para uma classificação multiclasses. Se n_class for o número de classes do problema, então (n_class * (n_class-1))/2 máquinas de vetores de suporte são construídas.

Todas podem ser agregadas de forma conveniente utilizando-se a função membro: *decision_function_shape*.



⇒ Tecelagem de Voronoi

Classificador SVM Multiclasses

```
>>> X = [[0], [1], [2], [3]]
>>> Y = [0, 1, 2, 3]
>>> clf = svm.SVC(decision_function_shape='ovo')
>>> clf.fit(X, Y)
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovo', degree=3, gamma='auto', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
>>> dec = clf.decision_function([[1]])
>>> dec.shape[1] # 4 classes: 4*3/2 = 6
6
>>> clf.decision_function_shape = "ovr"
>>> dec = clf.decision_function([[1]])
>>> dec.shape[1] # 4 classes
4
```

Classificador SVM Multiclasses

Por outro lado, LinearSVC implementa a estratégia "1 contra todos". Se n_classes for o número de classes do problema, então n_classes máquinas de vetores de suporte serão construídas.

Se n_classes = 2, então apenas uma única máquina de vetor de suporte será construída.

```
>>> lin_clf = svm.LinearSVC()
>>> lin_clf.fit(X, Y)
LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
     intercept_scaling=1, loss='squared_hinge', max_iter=1000,
     multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
     verbose=0)
>>> dec = lin_clf.decision_function([[1]])
>>> dec.shape[1]
4
```

PRÁTICA

1) Visitar o site:

   www.scikit-learn.org

2) Clicar em "SVM"

3) Clicar no item "1.4.1.3 – Unbalanced Problems"

4) Em "Examples", clicar em "Plot different SVM Classifiers in the Iris Dataset"

5) Faça o download de plot_iris.py e execute no Spyder.

PRÁTICA

6) Identifique a instrução que seleciona os dois primeiros atributos (comprimento da sépala, largura da sépala), troque-os pelos dois últimos e explique o que acontece.

PRÁTICA

1) Baixe também e execute:

- SVM: Maximum margin separating hyperplane
- Non-linear SVM
- SVM: Weighted samples

PRÁTICA

Vá à seção 1.4, leia, entenda e execute o exemplo:

- Support Vector Regression (SVR) using linear and non-linear kernels

PRÁTICA

Vá à seção 1.4.3, leia, entenda e execute o exemplo:

- One-class SVM with non-linear kernel (RBF)

Referências

1. *Neural Networks and Learning Machines*, 3rd. Edition, Simon Haykin

2. *Fundamental of Neural Networks - Architectures, Algorithms and Applications*, Laurene Fausett

3. *Pattern Classification*, Richard O. Duda, Peter E. Hart, David G. Stork