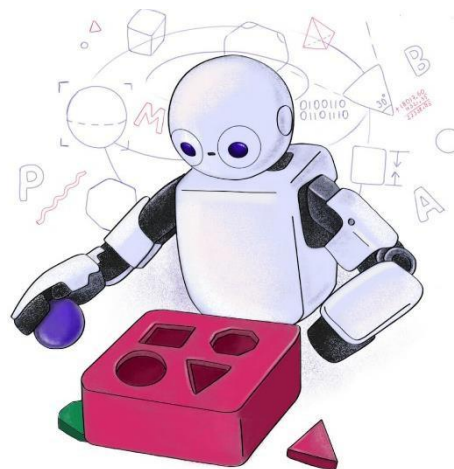


TP555 - Inteligência Artificial e Machine Learning: *Máquina de Vetores de Suporte*



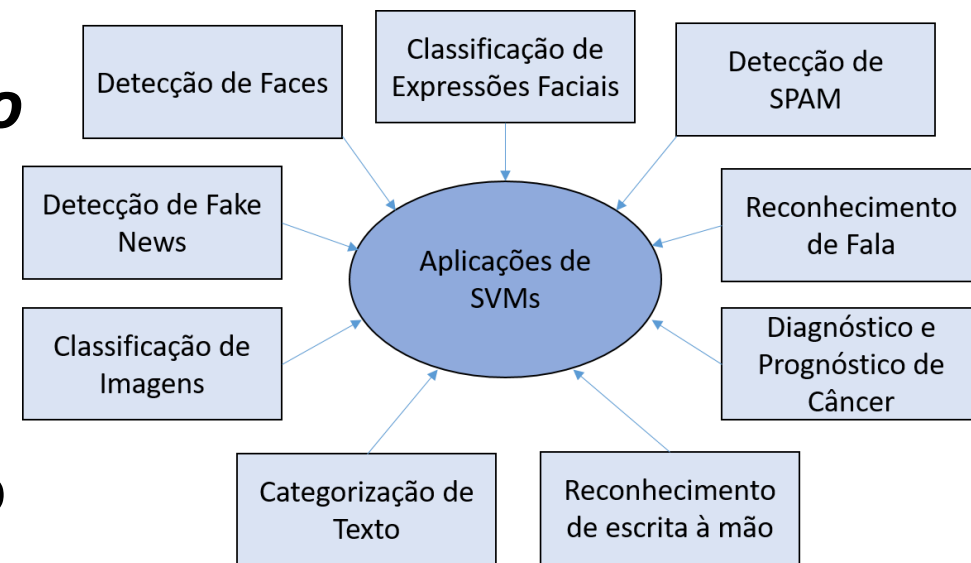
Esse material foi desenvolvido e gentilmente cedido pelo Prof. Dr. Felipe Augusto Pereira de Figueiredo, do Inatel. (felipe.figueiredo@inatel.br)

Inatel

Prof. Dr. Luiz Augusto Melo Pereira
luiz.melo@inatel.br

Máquinas de Vetores de Suporte

- Uma Máquina de Vetores de Suporte, do Inglês *Support Vector Machine* (SVM), é um modelo de aprendizado de máquina muito poderoso, versátil capaz de realizar ***classificação linear*** ou ***não-linear***, ***regressão*** e até ***detecção de outliers***.
- SVM é atualmente a abordagem mais popular para aprendizado supervisionado *pronto para uso*, ou seja, se você não tiver nenhum conhecimento prévio especializado sobre um problema (ou domínio), então SVM é um excelente método para se tentar inicialmente.



Propriedades das SVMs

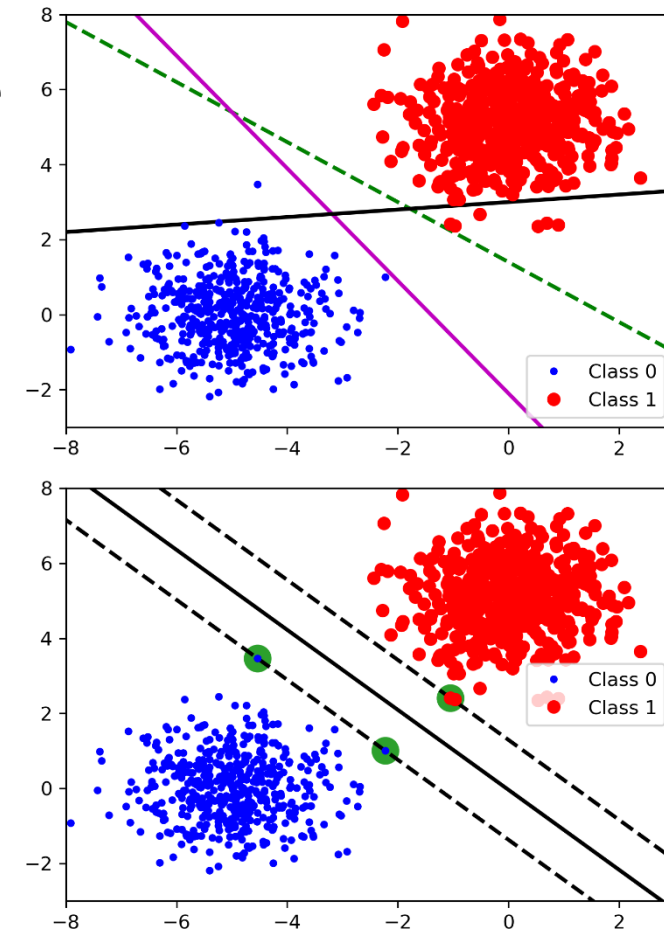
Existem 3 propriedades que tornam as SVM muito atrativas:

1. SVM constroem um ***separador de margem máxima***, ou seja, a ***fronteira de decisão*** tem a maior distância possível para os pontos mais próximos das classes. Esta propriedade ajuda as SVMs a generalizar muito bem.
2. SVMs criam um ***hiperplano de separação linear***, porém, elas têm a habilidade de projetar os dados de entrada em um espaço com dimensão mais alta, usando o chamado ***truque do kernel*** (do Inglês, *kernel trick*). Frequentemente, classes que não são linearmente separáveis no espaço de entrada original são facilmente separáveis em um espaço dimensional mais alto. O separador linear de alta dimensão é não-linear no espaço original. Isso significa que o espaço de hipóteses é expandido em relação aos métodos que usam representações estritamente lineares.
3. SVMs são abordagens não-paramétricas. Elas retêm exemplos de treinamento e, potencialmente, precisam armazená-los todos. Por outro lado, na prática, elas frequentemente precisam reter apenas uma pequena fração do número de exemplos; algumas vezes tão pequeno quanto uma pequena constante multiplicada pelo número de dimensões. Portanto, SVMs combinam as vantagens de modelos paramétricos e não-paramétricos: elas têm a flexibilidade para representar funções complexas, mas são resistentes ao ***sobreajuste***.

Ideia fundamental por trás das SVMs

- A figura ao lado mostra 2 classes que podem ser claramente separadas com uma reta (elas são linearmente separáveis).
- A figura superior mostra os limites de decisão de três classificadores lineares possíveis. O modelo cujo limite de decisão é representado pela linha preta é tão ruim que nem mesmo separa as classes adequadamente.
- Os outros dois modelos funcionam perfeitamente neste conjunto de treinamento, mas seus limites de decisão chegam tão perto dos exemplos que esses modelos provavelmente não generalizarão bem.
- Em contraste, a linha sólida na figura inferior representa o limite de decisão de um classificador SVM. Esta linha não apenas separa as duas classes, mas também fica o mais longe possível dos exemplos de treinamento mais próximos.

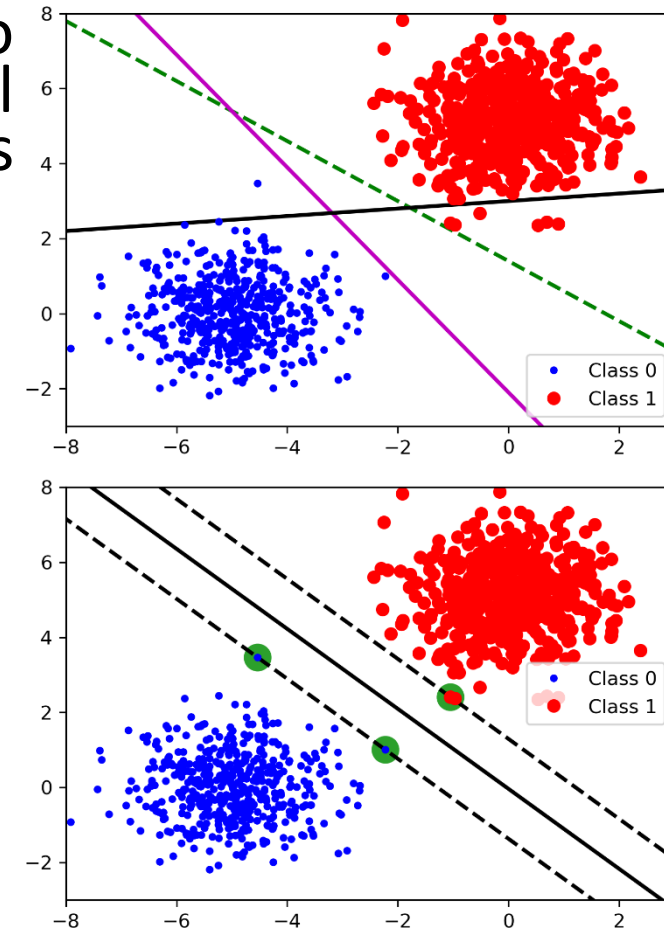
[Exemplo: svm_exemplo1.ipynb](#)



Ideia fundamental por trás das SVMs

- Portanto, podemos pensar em um classificador SVM como se ele estivesse criando uma rua, a mais larga possível (representada pelas linhas tracejadas paralelas), entre as classes.
- A linha sólida é chamada de **separador de margem máxima** e fica no ponto médio das margens (área entre as linhas tracejadas). Essa SVM também é conhecida como **classificador de margem máxima**.
- Observem que adicionar mais exemplos de treinamento “fora da rua” não afetará de forma alguma o limiar de decisão: ele é totalmente determinado (ou “suportado”) pelos exemplos localizados na beira da rua.
- Esses exemplos são chamados de **vetores de suporte** (círculos grandes verdes) e são os exemplos mais próximos do separador.

[Exemplo: svm_exemplo1.ipynb](#)



Separador de Margem Máxima

- Em vez de minimizar a ***perda empírica*** nos dados de treinamento, SVMs tentam minimizar a ***perda de generalização***.
- Como nós não sabemos onde os pontos ainda não vistos podem cair, mas sob a suposição probabilística de que eles são extraídos da mesma distribuição que os exemplos anteriores, existem alguns argumentos vindos da ***teoria de aprendizagem computacional*** sugerindo que se minimize a ***perda de generalização*** escolhendo-se o separador que esteja o mais longe dos exemplos vistos até agora.
- Nós chamamos esse separador, mostrado na figura anterior de ***separador de margem máxima***.
- A margem é a largura da área delimitada pelas linhas tracejadas na figura, ou seja, duas vezes a distância do ***separador*** até o ponto do exemplo mais próximo.
- Agora fica a pergunta, como nós podemos encontrar esse ***separador de margem máxima***?

Separador de Margem Máxima

- Antes de vermos como se encontra o separador, precisamos definir algumas notações usadas pelas SVMs.
- Tradicionalmente, as SVMs usam a convenção de que os rótulos de classe são $+1$ e -1 , em vez de $+1$ e 0 que usamos até agora.
- Além disso, diferentemente do que fazíamos anteriormente onde colocávamos o ponto de interseção no vetor de pesos \mathbf{w} (e um valor 1 correspondente ao atributo x_0), as SVMs não fazem isso, elas mantêm o ponto de interseção como um parâmetro separado, b .
- Com isso em mente, o separador é definido como o conjunto de pontos $\{\mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = 0\}$.
- Nós poderíamos pesquisar o espaço de \mathbf{w} e b com o gradiente descendente para encontrar os parâmetros que maximizam a margem enquanto classificamos corretamente todos os exemplos.

Separador de Margem Máxima

- No entanto, existe outra abordagem para resolver esse problema.
- Nós não veremos os detalhes, mas apenas diremos que existe uma representação alternativa chamada **representação dual**, na qual a solução ótima é encontrada resolvendo-se

$$\arg \max_{\alpha} \sum_j \alpha_j - \frac{1}{2} \sum_{j,k} \alpha_j \alpha_k y_j y_k (\mathbf{x}_j \cdot \mathbf{x}_k),$$

sujeito às restrições $\alpha_j \geq 0$ e $\sum_j \alpha_j y_j = 0$.

- Este é um problema de otimização de **programação quadrática**, para o qual existem boas bibliotecas e soluções de software (e.g., APMonitor, CPLEX, Matlab, Mathematica, etc.).
- Uma vez encontrado o vetor α , podemos voltar a \mathbf{w} com a equação $\mathbf{w} = \sum_j \alpha_j \mathbf{x}_j$, ou podemos permanecer com a representação dual.

Separador de Margem Máxima

- A equação anterior possui três propriedades importantes.
- Primeiro, a expressão é convexa, ou seja, ela tem um único máximo global que pode ser encontrado com eficiência.
- Em segundo lugar, ***os valores entram na expressão apenas na forma de produtos escalares de pares de pontos***. Essa segunda propriedade também é verdadeira para a equação do próprio separador, uma vez que o valor α_j ideal tenha sido calculado, ou seja

$$h(\mathbf{x}) = \text{sign} \left(\sum_j \alpha_j y_j (\mathbf{x} \cdot \mathbf{x}_j) - b \right),$$

onde $\text{sign}(\cdot)$ é a ***função sinal***, $\mathbf{x}_j, \forall j$ são os vetores de suporte e $\alpha_j, \forall j$ são os ***pesos*** ou ***coeficientes de Lagrange***.

- Uma propriedade importante final é que os pesos α_j associados a cada ponto de dados são iguais a zero, exceto para os ***vetores de suporte***, ou seja, os pontos mais próximos do separador.
- Eles são chamados de vetores de ***suporte*** porque ***sustentam*** o plano de separação.
- Como geralmente há muito menos vetores de suporte do que exemplos, as SVMs ganham algumas das vantagens dos modelos paramétricos.

Separador de Margem Máxima

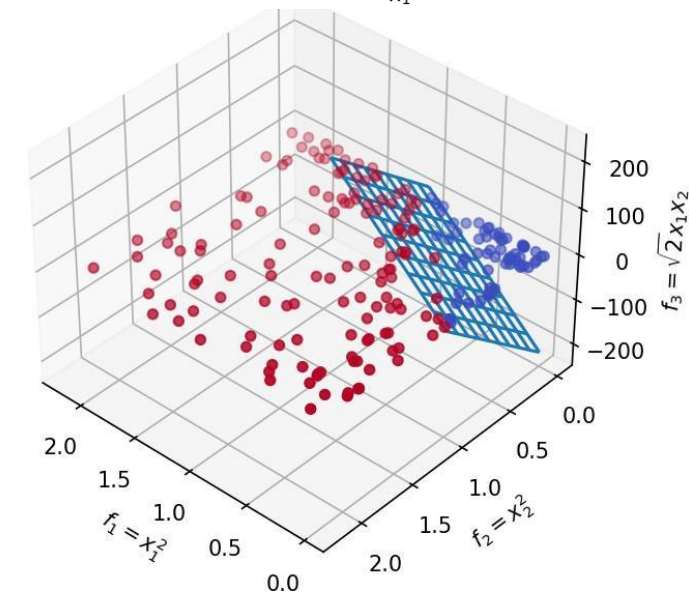
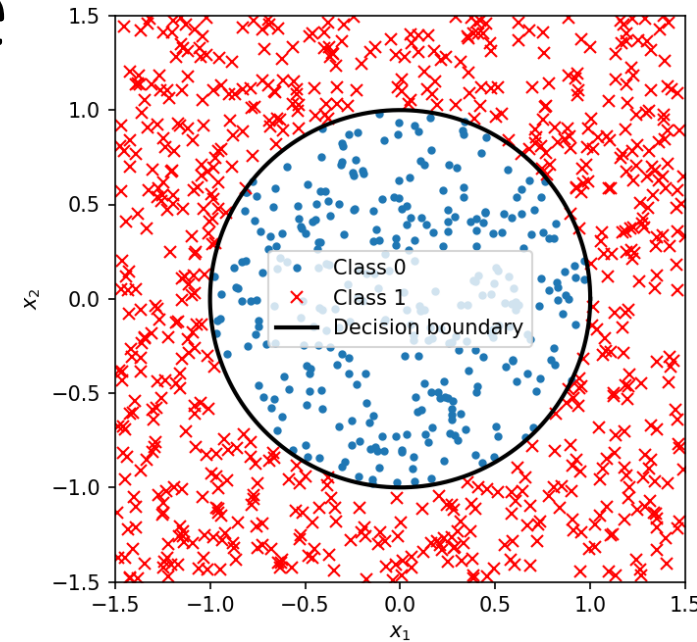
- O ***classificador de margem máxima***, quando aplicado a dados não separáveis linearmente, não encontra a solução desejada.
- Isso é evidenciado pela equação ***da representação dual*** que, aplicada a dados não linearmente separáveis, cresce arbitrariamente.
- O principal problema desse classificador é que ele sempre constrói hipóteses que se baseiam na ***inexistência de erros de treinamento***.
- Entretanto, para dados com ruídos, que geralmente implica em separação não linear, a maximização da equação ***da representação dual*** não pode ser calculado dessa forma, pois pode causar ***overfitting***.
- Essas desvantagens motivaram o desenvolvimento de técnicas que permitem o tratamento de problemas não linearmente separáveis via SVMs.
- Então o que fazer se as classes não forem linearmente separáveis?

Classes não separáveis linearmente

- A figura superior ao lado mostra um espaço de entrada definido pelos atributos $\mathbf{x} = (x_1, x_2)$, com exemplos positivos ($y = +1$) dentro do círculo e exemplos negativos ($y = -1$) fora.
- Claramente, não há separador linear para esse problema.
- Agora, suponha que reexpressemos os dados de entrada, ou seja, mapeamos cada vetor de entrada \mathbf{x} para um novo vetor de valores de atributos, $F(\mathbf{x})$.
- Em particular, vamos usar os três atributos

$$f_1 = x_1^2, \quad f_2 = x_2^2, \quad f_3 = \sqrt{2}x_1x_2$$

[Exemplo: svm exemplo2.ipynb](#)



Classes não separáveis linearmente

- Nós veremos em breve de onde esses 3 atributos vieram, mas por enquanto, basta vermos o que acontece.
- A figura inferior mostra os dados no novo espaço tridimensional definido pelos três novos atributos.
- Vejam que agora os dados são linearmente separáveis neste espaço.
- Este fenômeno é na verdade bastante geral: se os dados são mapeados em um espaço de dimensão suficientemente alta, então eles quase sempre serão linearmente separáveis.

Classes não separáveis linearmente

- Nós normalmente não esperaríamos encontrar um separador linear no espaço de entrada \mathbf{x} , mas podemos encontrar separadores lineares no espaço de atributos de alta dimensão $F(\mathbf{x})$ simplesmente substituindo $\mathbf{x}_j \cdot \mathbf{x}_k$ na equação de otimização de **programação quadrática** por $F(\mathbf{x}_j) \cdot F(\mathbf{x}_k)$.
- Isso por si só não é algo notável. Substituir \mathbf{x} por $F(\mathbf{x})$ em *qualquer* algoritmo de aprendizado de máquina tem o efeito necessário, mas o produto escalar tem algumas propriedades especiais.
- Acontece que $F(\mathbf{x}_j) \cdot F(\mathbf{x}_k)$ muitas vezes pode ser calculado sem primeiro calcular $F(\cdot)$ para cada ponto.
- Em nosso espaço de atributos tridimensionais definido pelas equações $f_1 = x_1^2$, $f_2 = x_2^2$ e $f_3 = \sqrt{2}x_1x_2$ um pouco de álgebra mostra que

$$F(\mathbf{x}_j) \cdot F(\mathbf{x}_k) = (\mathbf{x}_j \cdot \mathbf{x}_k)^2$$

- Por esse motivo temos $\sqrt{2}$ em f_3
- Percebam que o produto escalar dos vetores transformados é igual ao quadrado do produto escalar dos vetores originais.

Função de Kernel

- A expressão $(\mathbf{x}_j \cdot \mathbf{x}_k)^2$ é chamada de **função de kernel**, e geralmente é escrita como $K(\mathbf{x}_j, \mathbf{x}_k)$.
- A **função de kernel** pode ser aplicada a pares de dados de entrada para avaliar produtos escalares em algum espaço de atributos correspondente.
- Portanto, podemos encontrar separadores lineares no espaço de atributos de alta dimensão $F(\mathbf{x})$ simplesmente substituindo $\mathbf{x}_j \cdot \mathbf{x}_k$ na equação de otimização de **programação quadrática** por uma **função de kernel** $K(\mathbf{x}_j, \mathbf{x}_k)$.
- Desta forma, podemos aprender no espaço de alta dimensão, mas calculamos apenas as **funções de kernel** em vez da lista completa de atributos para cada ponto de dados.

Função de Kernel

- A próxima etapa é verificar que não há nada de especial sobre o kernel $K(\mathbf{x}_j, \mathbf{x}_k) = (\mathbf{x}_j \cdot \mathbf{x}_k)^2$.
- O kernel corresponde a um determinado espaço de atributos de alta dimensão, mas outras **funções de kernel** correspondem a outros espaços de atributos.
- Um resultado muito importante em matemática, o **teorema de Mercer**, diz que qualquer **função de kernel** $K(\mathbf{x}_j, \mathbf{x}_k)$ que respeite a condição matemática chamada de **condição de Mercer** corresponde a algum espaço de atributos.
- Uma **função de kernel** que seja **positiva-definida**, i.e, $\sum_j \sum_k c_j K(\mathbf{x}_j, \mathbf{x}_k) c_k \geq 0$, satisfaz a condição.
- Em outras palavras, se $K(\mathbf{x}_j, \mathbf{x}_k)$ for **positiva-definida**, então existe uma função $F(\cdot)$ que mapeia \mathbf{x}_j e \mathbf{x}_k em outro espaço (possivelmente com dimensões muito mais altas) tal que $K(\mathbf{x}_j, \mathbf{x}_k) = F(\mathbf{x}_j) \cdot F(\mathbf{x}_k)$.
- Esses espaços de atributos podem ser muito grandes, mesmo para kernels de aparência simples.
- Por exemplo, o kernel polinomial, $K(\mathbf{x}_j, \mathbf{x}_k) = (1 + \mathbf{x}_j \cdot \mathbf{x}_k)^d$, corresponde a um espaço de atributos cuja dimensão é exponencial em d

Truque do Kernel

- Insight principal aqui é que se aplicarmos a transformação $F(\cdot)$ a todas os exemplos de treinamento, então o problema dual conterá o produto escalar $F(\mathbf{x}_j) \cdot F(\mathbf{x}_k)$, mas se $F(\cdot)$ é uma transformação polinomial de 2º grau, então podemos substituir o produto escalar dos vetores transformados simplesmente por $(\mathbf{x}_j \cdot \mathbf{x}_k)^2$.
- O resultado seria o mesmo se tivéssemos tido o trabalho de transformar todo o conjunto de treinamento e, em seguida, treinando uma SVM.
- Este é o **truque do kernel**: conectando esses kernels na equação de otimização de **programação quadrática**, *separadores lineares ideais podem ser encontrados com eficiência computacional em espaços de atributos com bilhões de (ou, em alguns casos, infinitas) dimensões.*
- Os separadores lineares resultantes, quando mapeados de volta para o espaço de entrada original, podem corresponder a limiares de decisão não lineares arbitrariamente tortuosos entre os exemplos das classes positiva e negativa.

Classificador de Margem Suave

- No caso de dados inerentemente ruidosos, podemos não querer um separador linear em algum espaço de alta dimensão.
- Em vez disso, podemos querer uma superfície de decisão em um espaço de dimensão inferior que não separe as classes de maneira clara, mas reflita a realidade dos dados ruidosos.
- Isso é possível com o ***classificador de margem suave***, que permite que os exemplos caiam do lado errado do limiar de decisão, mas atribui a eles uma penalidade proporcional à distância necessária para movê-los de volta para o lado correto.

Kernelização

- O ***truque do kernel*** pode ser aplicado não apenas a algoritmos de aprendizagem que encontram separadores lineares ideais, mas também com qualquer outro algoritmo que possa ser reformulado para funcionar apenas com produtos escalares de pares de pontos de dados.
- Uma vez feito isso, o produto escalar é substituído por uma ***função de kernel*** e temos uma versão kernelizada do algoritmo.
- Isso pode ser feito facilmente para algoritmos como o k-vizinhos mais próximos (do Inglês, k-Nearest Neighbours) e perceptron, entre outros.

Outras funções de Kernel

- Alguns kernels comuns incluem:
 - **Linear:** $K(\mathbf{x}_j, \mathbf{x}_k) = \mathbf{x}_j \cdot \mathbf{x}_k$.
 - **Polinomial (homogêneo):** $K(\mathbf{x}_j, \mathbf{x}_k) = (\mathbf{x}_j \cdot \mathbf{x}_k)^d$
 - **Polinomial (não homogêneo):** $K(\mathbf{x}_j, \mathbf{x}_k) = (1 + \mathbf{x}_j \cdot \mathbf{x}_k)^d$
 - **Função de base radial gaussiana:** $K(\mathbf{x}_j, \mathbf{x}_k) = e^{-\gamma \|\mathbf{x}_j - \mathbf{x}_k\|^2}$ para $\gamma > 0$
 - **Tangente hiperbólica (sigmoide):** $K(\mathbf{x}_j, \mathbf{x}_k) = \tanh(\kappa \mathbf{x}_j \cdot \mathbf{x}_k + c)$ para $\kappa > 0$ e $c < 0$
 - E várias outras...

Vantagens das SVMs

- Além das propriedades mencionadas anteriormente, as SVM também apresentam as seguintes propriedades e vantagens:
 - Elas têm a capacidade de lidar com grandes espaços de atributos.
 - As SVMs são muito boas quando não se tem ideia sobre os dados.
 - O truque do kernel é a verdadeira força por trás das SVMs. Com uma *função de kernel* apropriada, podemos resolver qualquer problema complexo.
 - Elas escalonam relativamente bem para dados com altas dimensões.
 - Em geral, SVMs generalizam bem, portanto, o risco delas sobreajustarem aos dados de treinamento é menor, porém, elas podem sobreajustar se o número de atributos for muito maior do que o número de amostras.
 - SVMs são relativamente eficientes em termos de uso de memória.
 - SVMs são definidas por um problema de otimização convexa (sem mínimos locais) para o qual existem métodos eficientes e com otimização ótima (i.e., mínimo global) garantida.

Limitações das SVMs

- Algumas das limitações/desvantagens das SVMs são:
 - São sensíveis:
 - ✓ às escalas de atributos ([Exemplo: svm sensitivity to feature scales.ipynb](#)).
 - ✓ à outliers, i.e., valores atípicos, discrepantes ([Exemplo: svm sensitivity to outliers.ipynb](#)).
 - Escolher uma ***função de kernel*** “boa” nem sempre é fácil.
 - Longo tempo de treinamento para grandes conjuntos de dados.
 - Difícil de entender, visualizar e interpretar o modelo final.
 - Já que o modelo final não é tão fácil de ser visualizado e interpretado, fica difícil fazer pequenos ajustes nos parâmetros do modelo.
 - As SVMs não fornecem estimativas de probabilidade, o que é desejável na maioria dos problemas de classificação.

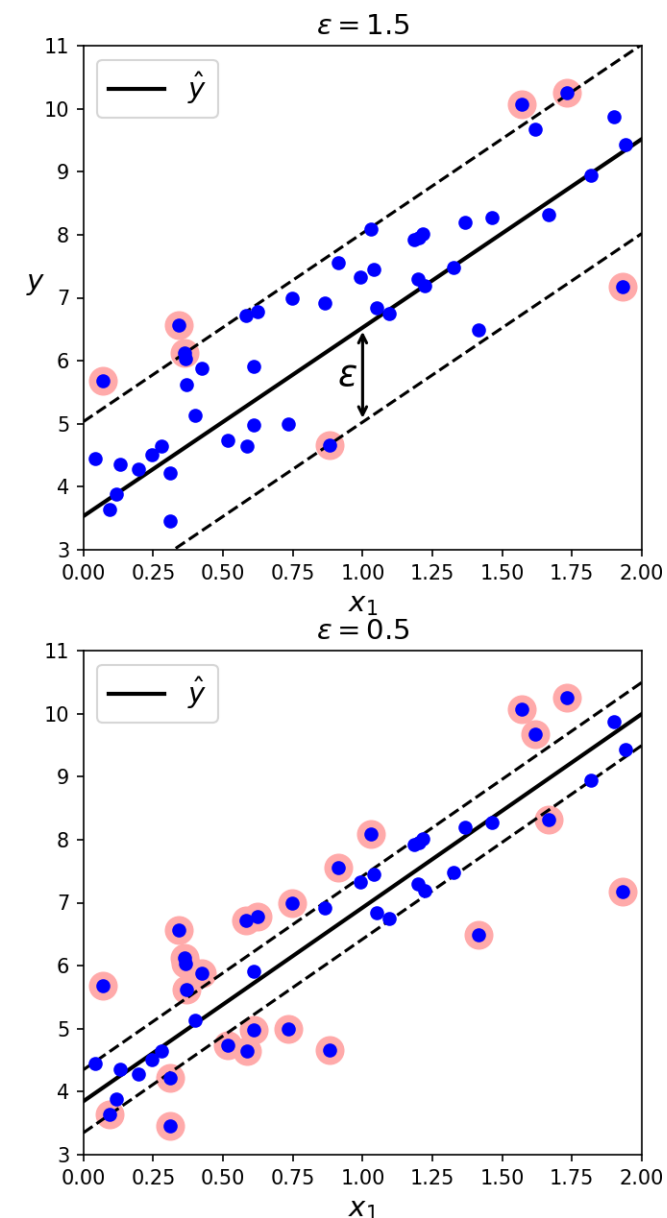
SVMs para problemas com múltiplas classes

- Embora as SVMs separem os dados linearmente em **duas classes**, a classificação de mais do que duas classes é possível utilizando a estratégia de decomposição do problema multiclass em subproblemas binários.
- Existem algumas técnicas que são utilizadas para resolver o problema com múltiplas classes, as mais conhecidas são a **Um-Contra-Todos** e **Um-Contra-Um**.
- Seja Q o número de classes, a técnica **Um-Contra-Todos** consiste em separar uma classe A e agrupar as $Q - 1$ classes restantes em uma classe B , a partir da separação encontraremos o hiperplano que separa a classe A da classe B . O processo de separação da classe é realizado Q vezes para cada classe pertencente ao conjunto de classes, logo, encontraremos Q hiperplanos que separam as classes.
- Seja Q o número de classes, a técnica **Um-Contra-Um** consiste em separar duas classes A e B do conjunto de classes e encontrar um hiperplano que separe esse par de classes. O processo de separação é realizado para cada par de classes pertencente ao conjunto de classes, logo encontraremos Q hiperplanos que separam as classes.

Regressão linear com SVMs

- SVMs são bastante versáteis: não apenas oferecem suporte à classificação linear e não linear, mas também à regressão linear e não linear.
- O truque é inverter o objetivo: em vez de tentar encontrar a maior rua possível entre duas classes, enquanto limita as violações de margem (ou seja, exemplos entre as margens), a regressão com SVMs tenta encaixar tantos exemplos quanto possíveis na rua enquanto limita violações de margem (ou seja, exemplos fora da rua).
- A largura da rua é controlada por um hiperparâmetro ϵ .
- A figura ao lado mostra dois modelos de regressão SVM linear treinados em alguns dados lineares aleatórios, um com uma grande margem ($\epsilon = 1.5$) e o outro com uma pequena margem ($\epsilon = 0.5$).
- Adicionar mais exemplos de treinamento dentro da margem não afeta as previsões do modelo, portanto, o modelo é dito ser insensível a variações de ϵ .

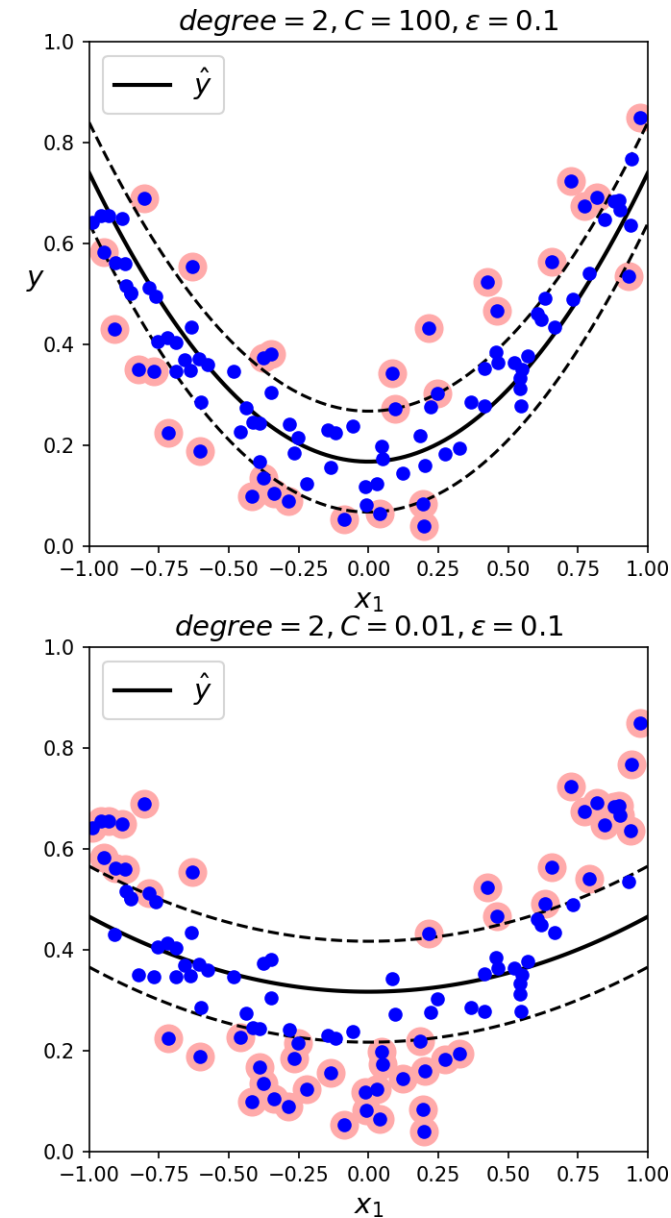
Exemplo: [svm_regression1.ipynb](#)



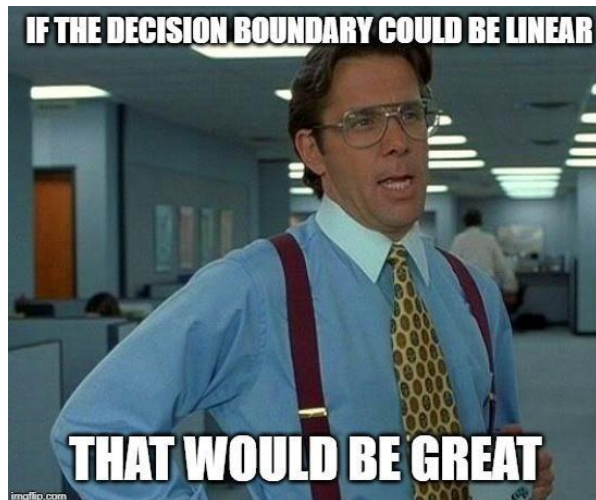
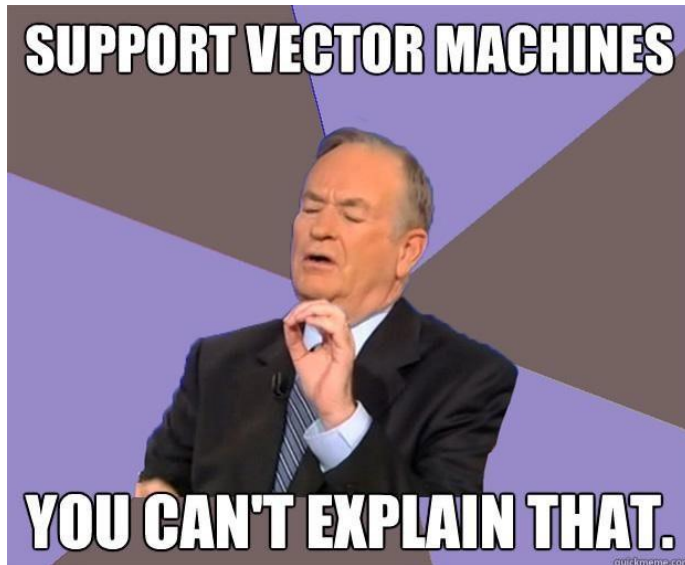
Regressão não linear com SVMs

- Para lidar com tarefas de regressão não linear, podemos usar um modelo SVM kernelizado.
- Por exemplo, a figura ao lado mostra a regressão com SVM em um conjunto de treinamento quadrático aleatório, usando um kernel polinomial de 2º grau.
- Conforme podemos ver, há pouca regularização na figura superior (ou seja, um grande valor C) e muito mais regularização na figura inferior (ou seja, um pequeno valor C).

Exemplo: [svm_regression2.ipynb](#)



Obrigado!



Coding
the SVM
algorithm in numpy

from sklearn
import svm



Figuras

