



Engenharia de Computação
Inteligência Artificial
10 - Support Vector Machines (SVM)
Margem, Vetores de Suporte, Kernels e
PCA

Organização da aula

- Retomada: Árvore de Decisão, KNN e diferentes formas de construir uma fronteira de decisão.
- Separação linear, hiperplano e margem máxima.
- Vetores de suporte e sua influência na fronteira.
- Dados não perfeitamente separáveis: *soft margin* e parâmetro C .
- Separação não linear, visualização 2D $\rightarrow$ 3D e *kernel trick*.
- Kernel linear e RBF/Gaussiano; interpretação de γ .
- Padronização e implementação com *Scikit-learn*.
- PCA como extensão para redução de dimensionalidade e visualização.

Fio condutor

Se várias fronteiras conseguem separar duas classes, como escolher uma fronteira que generalize melhor para novos exemplos?

Retomada: três estratégias de classificação

Árvore de Decisão

Divide o espaço por regras hierárquicas.

KNN

Decide pela proximidade entre exemplos.

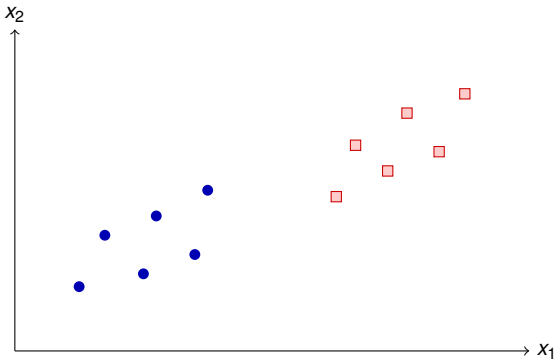
SVM

Aprende uma fronteira com grande margem entre as classes.

Continuidade

O problema continua sendo o mesmo: aprender uma função de classificação a partir de exemplos. O que muda é a hipótese utilizada por cada algoritmo.

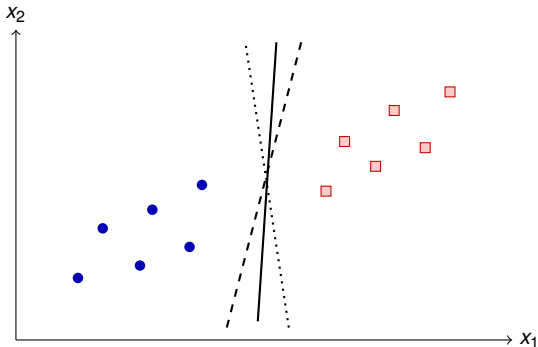
Um problema aparentemente simples



Pergunta

É possível traçar uma reta que separe as duas classes? Existe apenas uma reta possível?

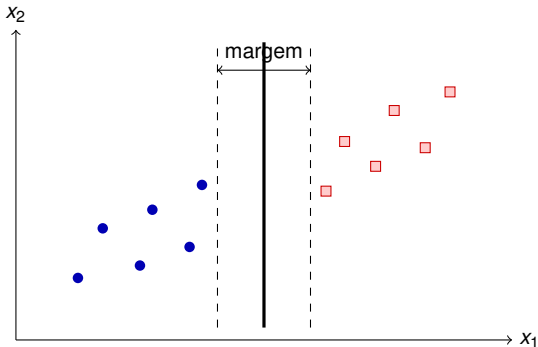
Existem várias fronteiras possíveis



Questão central

Se todas classificam corretamente os exemplos de treinamento, qual delas devemos preferir?

A ideia de margem



Ideia central

A SVM procura uma fronteira que maximize a distância até os exemplos mais próximos das duas classes.

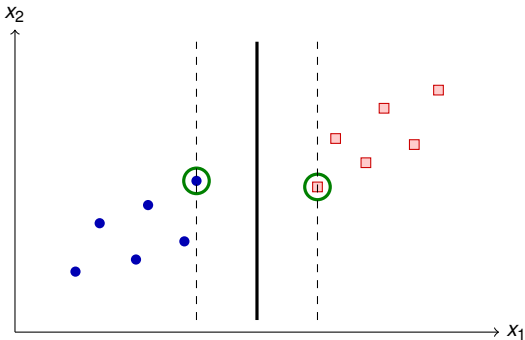
Por que maximizar a margem?

- Uma fronteira muito próxima dos exemplos pode ser sensível a pequenas variações nos dados.
- Uma margem maior cria uma região de separação mais ampla entre as classes.
- A decisão não depende apenas de classificar corretamente os dados de treinamento, mas de escolher uma separação com boa capacidade de generalização.

Conexão com a Aula 08

Classificar perfeitamente todos os exemplos de treinamento garante bom desempenho em exemplos novos?

Vetores de suporte



Definição

Os exemplos que determinam as margens são os **vetores de suporte**. Eles são os pontos mais relevantes para posicionar a fronteira.

O que acontece se removermos pontos?

Ponto distante da margem

Remover um exemplo distante pode não alterar a fronteira aprendida.

Vetor de suporte

Alterar ou remover um vetor de suporte pode modificar a posição da fronteira.

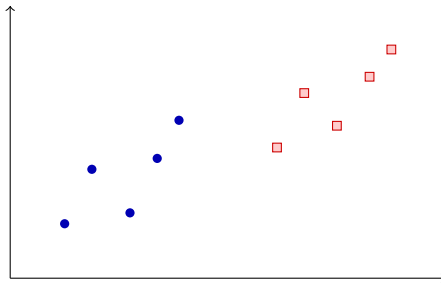
Pergunta

Isso significa que todos os exemplos de treinamento têm a mesma importância para a SVM?

Atividade 1 - identifique os vetores de suporte

Em duplas - 4 minutos

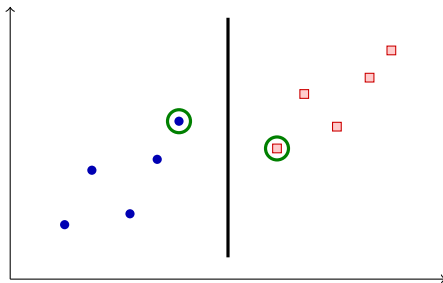
Observe o conjunto e indique quais exemplos provavelmente serão vetores de suporte para uma separação linear de margem máxima.



Justifique

Por que exemplos muito distantes da fronteira tendem a ter menor influência sobre sua posição?

Atividade 1 - resposta comentada



Leitura

Os pontos destacados são os candidatos naturais a vetores de suporte porque estão mais próximos da fronteira de separação.

Hard margin: um caso ideal

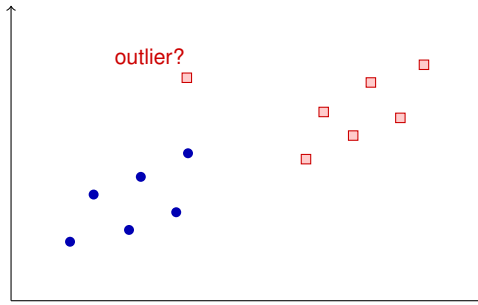
O caso mais simples assume que as classes podem ser separadas perfeitamente por uma fronteira linear.

- nenhum exemplo pode ficar do lado incorreto da margem;
- a margem deve ser maximizada;
- a hipótese é rígida e sensível a ruído e outliers.

Limitação

Dados reais raramente são perfeitamente separáveis. Precisamos permitir algumas violações da margem.

E quando existe um outlier?



Pergunta

Devemos deslocar completamente a fronteira para classificar corretamente esse único exemplo?

Soft margin

A SVM pode aceitar exemplos dentro da margem ou até do lado incorreto da fronteira.

- aumenta a tolerância a ruído e sobreposição entre classes;
- cria um compromisso entre margem larga e erros de treinamento;
- esse compromisso é controlado principalmente pelo parâmetro C .

Ideia

Em dados reais, uma fronteira ligeiramente imperfeita no treinamento pode generalizar melhor do que uma fronteira excessivamente ajustada.

O parâmetro C

C menor

- maior regularização;
- tolera mais violações;
- tende a produzir uma margem mais flexível.

C maior

- menor regularização;
- penaliza mais os erros;
- tenta ajustar mais os exemplos de treinamento.

Observação

No `SVC`, C é o parâmetro de regularização: a força da regularização é inversamente proporcional a C .

Relacionando C e generalização

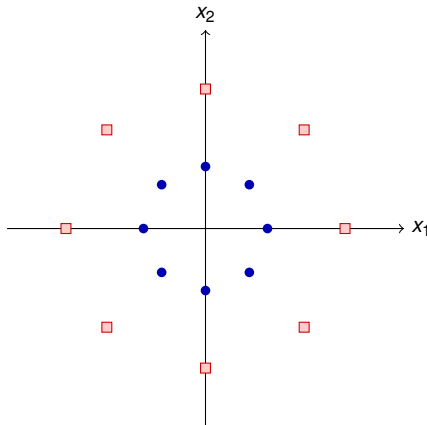
C muito baixo $\longleftarrow$ mais flexibilidade $\longrightarrow$ mais
ajuste $\longrightarrow C$ muito alto

- Não existe um valor universalmente melhor para C .
- O valor adequado depende dos dados e deve ser escolhido usando validação.
- O conjunto de teste não deve ser usado para escolher C .

Conexão com KNN

Que papel a validação cruzada tinha na escolha de k ? Podemos usar a mesma ideia para C ?

Nem todo problema é linearmente separável



Pergunta

Existe alguma reta capaz de separar perfeitamente a classe interna da classe externa?

Uma ideia: mudar a representação

Espaço original $\xrightarrow{\phi(x)}$ Espaço transformado

- No espaço original, a fronteira necessária pode ser não linear.
- Podemos representar os exemplos em um espaço de características diferente.
- Nesse novo espaço, uma separação linear pode se tornar possível.

Observação

O objetivo não é “deformar” arbitrariamente os dados, mas usar uma representação na qual relações relevantes possam ser separadas com uma fronteira simples.

Visualização: de 2D para 3D

- No espaço original, algumas classes não podem ser separadas por uma reta.
- Uma transformação pode representar os exemplos em um espaço de maior dimensão.
- Nesse novo espaço, uma separação por hiperplano pode se tornar possível.



Aponte a câmera para
abrir.

Durante o vídeo

Observe o que muda na representação dos dados e por que uma fronteira linear passa a ser suficiente no espaço transformado.

Abrir vídeo: Kernel Trick in SVM - visualização

Observação

A visualização em 3D é uma metáfora didática útil. O *kernel trick* evita a necessidade de construir explicitamente todas as coordenadas do espaço transformado.

O kernel trick

A SVM não precisa necessariamente construir explicitamente todas as novas coordenadas de $\phi(x)$.

$$K(x, z) = \phi(x)^T \phi(z)$$

- o kernel calcula produtos internos no espaço transformado;
- isso permite trabalhar implicitamente em espaços de dimensão elevada;
- diferentes kernels representam diferentes noções de similaridade.

Mensagem central

O kernel é uma etapa obrigatória de toda SVM ou apenas necessário quando queremos uma fronteira não linear?

Kernels mais comuns

Kernel	Ideia
linear	fronteira linear no espaço original
polinomial	combina interações polinomiais entre atributos
RBF	cria fronteiras flexíveis a partir de similaridade local
sigmoidal	forma relacionada à função tangente hiperbólica

Nesta aula

Vamos concentrar a implementação nos kernels **linear** e **RBF**.

Kernel RBF: a Gaussiana como similaridade

Uma forma comum do kernel RBF é:

$$K(x, z) = \exp(-\gamma \|x - z\|^2)$$

- $\|x - z\|$ mede a distância entre dois exemplos;
- se x e z estão muito próximos, $K(x, z)$ fica próximo de 1;
- conforme a distância aumenta, a similaridade cai rapidamente em direção a 0;
- a forma dessa queda é **Gaussiana**.

Interpretação

Por que uma medida de similaridade local pode ajudar a criar fronteiras de decisão não lineares?

C e γ atuam juntos

	γ menor	γ maior
C menor	mais regularização e fronteira suave	localidade maior, mas tolerância a erros
C maior	forte ajuste com fronteira ainda ampla	maior risco de fronteira muito complexa

Pergunta

Por que não faz sentido escolher C e γ olhando apenas para o desempenho no conjunto de teste?

Padronização com StandardScaler

A transformação por z-score é:

$$z = \frac{x - \mu}{\sigma}$$

- centraliza cada atributo em torno de zero;
- ajusta a escala para variância unitária;
- os parâmetros devem ser calculados somente no conjunto de treinamento.

Conexão com KNN

KNN e SVM são dois exemplos em que a escala dos atributos pode alterar significativamente o comportamento do modelo.

Pipeline básico: SVM linear

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC

model_linear = make_pipeline(
    StandardScaler(),
    SVC(kernel="linear", C=1.0)
)

model_linear.fit(X_train, y_train)
y_pred = model_linear.predict(X_test)
```

Por que Pipeline?

O `StandardScaler` é ajustado apenas com os dados de treinamento e a mesma transformação é aplicada posteriormente aos exemplos de validação e teste.

Agora com kernel RBF

```
model_rbf = make_pipeline(
    StandardScaler(),
    SVC(
        kernel="rbf",
        C=1.0,
        gamma="scale"
    )
)

model_rbf.fit(X_train, y_train)
y_pred = model_rbf.predict(X_test)
```

Pergunta

O que mudou no código? O processo de treinamento e predição mudou ou apenas a hipótese usada pelo classificador?

Linear ou RBF?

Linear

- mais simples;
- interpretável geometricamente;
- adequado quando uma fronteira linear é suficiente.

RBF

- fronteira não linear;
- mais flexível;
- exige atenção especial a C e γ .

Observação

Mais flexível não significa automaticamente melhor. O desempenho deve ser avaliado em dados não usados no ajuste dos hiperparâmetros.

Classificação multiclasse

A formulação clássica da SVM é binária. Para mais de duas classes, o problema é decomposto em vários classificadores binários.

- **One-vs-Rest:** uma classe contra todas as demais;
- **One-vs-One:** classificadores para pares de classes.

Scikit-learn

O `SVC` utiliza internamente uma estratégia one-vs-one para treinamento multiclasse.

Como escolher C , kernel e γ ?

Assim como escolhemos k no KNN, precisamos evitar ajustar hiperparâmetros usando o conjunto de teste.

treino $\rightarrow$ validação cruzada $\rightarrow$ melhores hiperparâmetros $\rightarrow$ teste final

- avaliar diferentes valores de C ;
- comparar kernels quando fizer sentido;
- para RBF, avaliar também γ ;
- manter o teste reservado para a avaliação final.

GridSearchCV: busca sistemática

```
from sklearn.model_selection import GridSearchCV

param_grid = {
    "svc__C": [0.1, 1, 10, 100],
    "svc__gamma": ["scale", 0.01, 0.1, 1],
    "svc__kernel": ["rbf"]
}

grid = GridSearchCV(
    estimator=model_rbf,
    param_grid=param_grid,
    cv=5,
    scoring="accuracy"
)

grid.fit(X_train, y_train)
print(grid.best_params_)
```

O que o GridSearchCV está fazendo?

- 1 seleciona uma combinação de hiperparâmetros;
- 2 executa validação cruzada no conjunto de treinamento;
- 3 calcula o desempenho médio;
- 4 repete para todas as combinações;
- 5 seleciona a configuração com melhor resultado segundo a métrica definida.

Observação

Busca em grade não “descobre” magicamente o modelo ideal: ela compara apenas as opções que foram fornecidas.

PCA: por que introduzir redução de dimensionalidade?

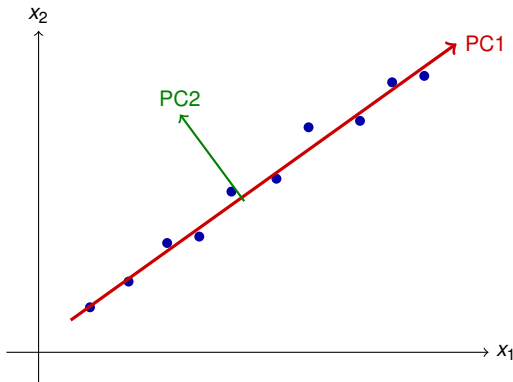
Em conjuntos com muitos atributos podemos desejar:

- reduzir o número de dimensões;
- remover parte da redundância entre atributos;
- visualizar os exemplos em duas ou três dimensões;
- reduzir custo de etapas posteriores.

Importante

PCA não é parte obrigatória da SVM. Ele é uma técnica separada de redução de dimensionalidade que pode ser usada antes do classificador.

Ideia intuitiva do PCA



Intuição

O PCA procura novas direções que representem grande parte da variabilidade dos dados. As componentes principais são combinações dos atributos originais.

PCA não é seleção de atributos

Seleção de atributos

Mantém alguns atributos originais e descarta outros.

PCA

Cria novos atributos como combinações lineares dos atributos originais.

Observação

Depois do PCA, “PC1” não corresponde diretamente a um único atributo original, o que pode reduzir a interpretabilidade.

Por que padronizar antes do PCA?

O PCA é sensível à variância dos atributos.

- atributos em escalas muito maiores podem dominar as primeiras componentes;
- a padronização coloca atributos em escalas comparáveis;
- em nosso pipeline, `StandardScaler` será aplicado antes de `PCA`.

dados $\rightarrow$ StandardScaler $\rightarrow$ PCA $\rightarrow$ SVM

Pipeline: StandardScaler + PCA + SVM

```
from sklearn.decomposition import PCA

model_pca = make_pipeline(
    StandardScaler(),
    PCA(n_components=2),
    SVC(kernel="linear", C=1.0)
)

model_pca.fit(X_train, y_train)
y_pred = model_pca.predict(X_test)
```

Objetivo didático

Com duas componentes podemos visualizar os exemplos e a fronteira em um plano. Isso não significa que duas componentes sejam sempre a melhor escolha para desempenho preditivo.

Variância explicada

Cada componente principal preserva uma parcela da variabilidade observada nos dados.

pca.explained_variance_ratio_

- permite verificar quanto de informação, no sentido de variância, foi preservado;
- duas componentes podem ser boas para visualizar, mas insuficientes para manter toda a estrutura relevante para classificação.

Pergunta

Se duas componentes explicam 70% da variância, isso garante que elas preservam 70% da informação necessária para classificar corretamente?

PCA melhorou o classificador?

Não podemos concluir isso apenas porque o gráfico ficou mais simples.

- PCA pode remover ruído ou redundância;
- PCA também pode descartar informação útil para a classe;
- reduzir dimensões pode acelerar algumas etapas;
- o efeito sobre desempenho deve ser medido experimentalmente.

Pergunta para a turma

O objetivo do PCA neste exemplo é melhorar a acurácia ou facilitar a representação e a análise dos dados?

Erros conceituais frequentes

- acreditar que SVM significa sempre uma fronteira linear;
- interpretar todos os exemplos como igualmente importantes para determinar a fronteira;
- aumentar C indefinidamente para “melhorar” o treinamento;
- usar kernel RBF sem considerar a escala dos atributos;
- ajustar C e γ usando o conjunto de teste;
- assumir que PCA é uma etapa obrigatória da SVM;
- interpretar PC1 e PC2 como atributos originais.

Até aqui: núcleo da Aula 10

margem $\rightarrow$ vetores de suporte $\rightarrow C \rightarrow$ kernel $\rightarrow$
 $\gamma \rightarrow$ padronização $\rightarrow$ SVC $\rightarrow$ PCA

Ponto de parada para o encontro de 100 minutos

Ao final do encontro, o estudante deve conseguir explicar a intuição da SVM, distinguir kernel linear e RBF, interpretar C e γ e montar um pipeline básico de classificação.

Referências

- Russell, S.; Norvig, P. *Artificial Intelligence: A Modern Approach*. Aprendizagem a partir de exemplos.
- Duda, R.; Hart, P.; Stork, D. *Pattern Classification*. Wiley.
- Cristianini, N.; Shawe-Taylor, J. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.
- Scikit-learn. Documentação de SVC, StandardScaler, GridSearchCV e PCA.
- When Maths Meet Coding. *Kernel Trick in Support Vector Machine (SVM) - explained with visualization*. Vídeo de apoio.
- Material-base anterior da disciplina, revisado e reorganizado para a sequência atual.

Próxima aula: atividade integradora

Como comparar Árvore de Decisão, KNN e SVM de forma justa no mesmo problema?

Continuidade

A próxima atividade consolida preparação de dados, treinamento, validação e comparação dos classificadores, preparando a execução do Trabalho 2.

Dúvidas



Dúvida é o começo da sabedoria.