Aula 7 - Classificação

UNIFESP

Universidade Federal de São Paulo

1933

AGENDA

- Problema de classificação
- Árvore de decisão
- Naive-bayes
- SVM
- MLP
- Florestas Aleatórias
- Avaliação dos classificadores
- K Fold Cross Validation

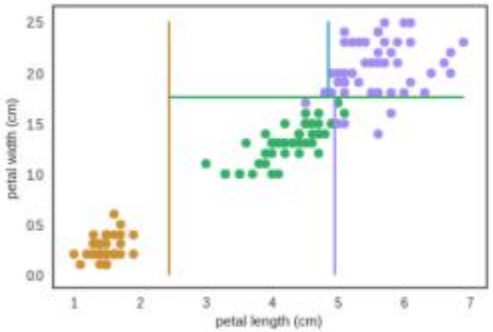
Problema de classificação

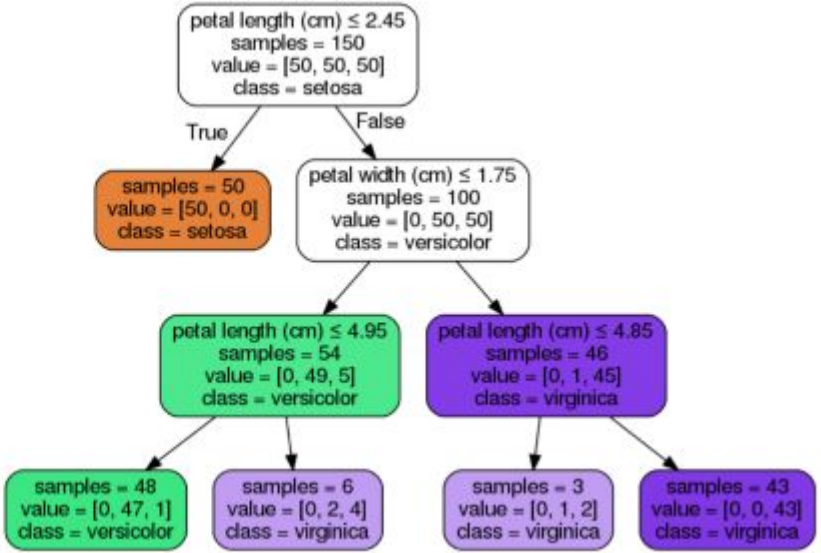
- Dado um conjunto de instâncias em que cada uma possui um rótulo de valor discreto que se deseja predizer tem-se então um problema de classificação.
- Cada instância desse conjunto possui atributos que podem ser avaliados para determinar sua classe.
- Busca-se então uma função f que possa por meio da análise dos atributos de uma instância estimar sua classe dentre um conjunto de classes C.
- Alguns algoritmos que são voltados para classificação: Árvore de decisão, SVM, MLP, Naive Bayes e Floresta Aleatória.

Árvore de decisão

- São baseadas na ideia computacional da estrutura de dados árvore binária.
- Consiste basicamente na divisão da classificação com base em um conjunto de escolhas com base nas características do dado analisado.
- Inicia-se na raiz da árvore e progride até suas folhas.
- Pode ser entendida como um conjunto de regras se-senão

Árvore de decisão





Fonte: [Árvores de Decisão. O objetivo desse post é explicar o que… | by Raphael Campos | Machine Learning Beyond Deep Learning | Medium]

Árvore de decisão - Implementação

- A Árvore de Decisão pode ser implementada utilizando a biblioteca ScikitLearn.
- O método DecisionTreeClassifier().fit(X_treino,Y_treino) retorna um classificador treinado sobre o conjunto especificado como parâmetro, sendo Y o conjunto classes alvo e X o conjunto de atributos

Naive Bayes

- O classificador Naive Bayes é um método probabilístico baseado no teorema de Bayes.
- Métodos probabilísticos bayesianos assumem que a probabilidade de um evento A, que pode ser uma classe, dado um evento B, que pode ser um conjunto de atributos de entrada não dependem unicamente da relação entre A e B mas também da probabilidade de observar A independentemente de observar B

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Teorema de Bayes

$$P(y_i|x) \propto P(y_i) \prod_{j=1}^{d} P(x_j|y_i)$$

Equação que define o classificador

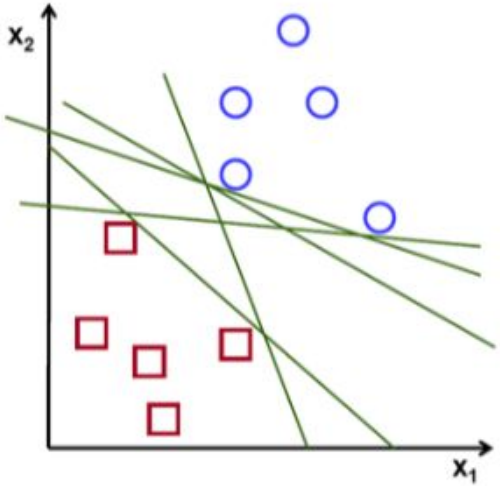
Naive Bayes - Implementação

- O classificador Naive Bayes pode ser implementado utilizando a biblioteca ScikitLearn.
- O método GaussianNB.fit(X_treino, Y_treino) retornada um classificador Naive Bayes Gaussiano treinado com base nos conjuntos X e Y de treino

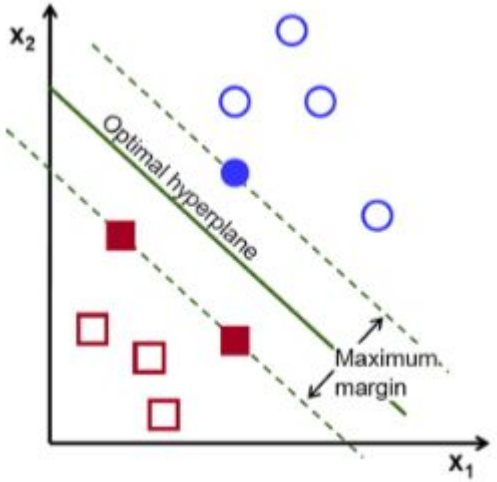
SVM - Support Vector Machine

- SVM é um classificador formalmente definido por um hiperplano de separação.
- Dado um conjunto de treinamento, o algoritmo irá gerar um hiperplano ideal que categoriza novos dados.
- No caso de um espaço bidimensional, o hiperplano é uma reta que separa um plano em duas partes, cada uma representando uma classe.
- O melhor hiperplano que divide o espaço é aquele que possui a maior distância mínima dos exemplos de treinamento.

SVM - Support Vector Machine



Linhas de separação



Melhor hiperplano

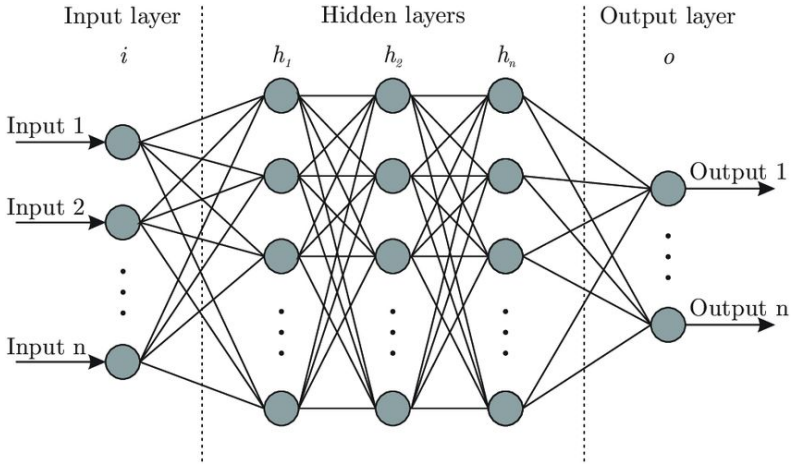
SVM - Implementação

- A SVM pode ser implementada utilizando a biblioteca ScikitLearn.
- O método SVC().fit(X_treino,Y_treino) retorna um classificador treinado sobre o conjunto especificado como parâmetro, sendo Y o conjunto classes alvo e X o conjunto de atributos

MLP - Multilayer Perceptron

- A MLP é uma rede neural artificial.
- As redes neurais são compostas por unidades básicas chamadas neurônios.
- Os neurônios da rede são dispostos em camadas altamente interligadas, sendo capaz, dessa forma, modelar complexas funções matemáticas.
- As conexões possuem pesos que são ajustados durante o processo de treino de modo a minimizar a taxa de erro.

MLP - Multilayer Perceptron



Fonte: Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks

MLP - Implementação

- A MLP pode ser implementada utilizando a biblioteca ScikitLearn.
- O método MLPClassifier(solver='lbgs').fit(X_treino,Y_treino) retorna um classificador treinado sobre o conjunto especificado como parâmetro, sendo Y o conjunto classes alvo e X o conjunto de atributos.
- O parâmetro *solver* corresponde ao otimizador que será utilizado para aprimoramento dos pesos durante o treino da rede.
    - lbgs - é um otimizador que tem uma velocidade de conversão alta, indicado para *datasets* pequenos.
    - adam - é indicado para datasets com grande quantidade de instâncias

Floresta Aleatória

- O algoritmo consiste no uso de um conjunto de árvores de decisão para realizar uma predição.
- Utiliza-se a técnica de *bagging* para gerar múltiplos conjuntos de dados de maneira aleatória.
- Cada conjunto gerado é utilizado para treinar uma árvore de decisão. Parte das árvores geradas irão compor modelo final, sendo feita a escolha com base no erro médio de cada uma

Floresta Aleatória - Implementação

- A Floresta Aleatória pode ser implementada utilizando a biblioteca ScikitLearn.
- RandomForestClassifier(n_estimators=100,max_depth=2).fit(X_treino,Y_treino).
- Parâmetros:
  - n_estimators - número máximo de árvores de decisão que irão compor o modelo.
  - max_depth - profundidade máxima das árvores, se não especificado as árvores serão expandidas ao máximo possível.

Métricas para avaliação

- Acurácia - Número de predições corretas em relação ao número total de instâncias.
- Recall - Porcentagem de instâncias pertencentes à classe A que foram classificadas corretamente
- Precision - Porcentagem de instâncias classificadas como pertencentes à classe A corretamente em relação a quantidade total de instâncias classificadas como A.
- F1 - Score - Média harmônica entre Precision e Recall

Métricas para avaliação

- Acurácia - Número de predições corretas em relação ao número total de instâncias. A = Nc/N
- Recall - Porcentagem de instâncias pertencentes à classe A que foram classificadas corretamente. R = VA/(VA+ FB)
- Precision - Porcentagem de instâncias classificadas como pertencentes à classe A corretamente em relação a quantidade total de instâncias classificadas como A. P = VA /(VA+ FA)
- F1 - Score - Média harmônica entre Precision e Recall. F1 = 2(P x R)/(P + R)

Holdout

- Para realizar o processo de treinamento e avaliação dos classificadores é necessário fazer a divisão do conjunto de dados em treino e teste.
- Uma abordagem para fazer essa divisão é a técnica de Holdout.
- Nessa abordagem o dataset é dividido em duas porções, com uma porção maior reservada para treino e a restante para teste.
- Em geral utiliza-se 80% ou 75% das amostras para treino. Contudo essa porcentagem pode variar para cada problema sendo indicado a realização de experimentos para determinar a melhor divisão.
- A porção de teste é utilizada para realizar o cálculo das métricas de avaliação

K Fold Cross Validation

- Outra abordagem para realizar a divisão do dataset é a K Fold Cross Validation.
- Nessa abordagem o dataset é dividido em K porções de tamanho igual.
- Desta forma o classificador é treinado com K-1 porções e a restante é utilizada para teste.
- Esse processo é repetido K vezes e ao final é calculada a média das métricas obtidas em cada teste.
- Esse método garante que o classificador seja avaliado com todos as instâncias do conjunto.

GRATIDÃO!

UNIFESP

UNIVERSIDADE FEDERAL DE SÃO PAULO

1933