Inteligência Artificial

Aprendizado de Máquina

PUC
RIO

Bianca Faria Dutra Fragoso

Pedro Felipe Magalhães

Introdução

Neste trabalho, usamos alguns algoritmos de aprendizado de máquina aprendidos em sala de aula, como o Support Vector Machine (SVM) e o K-Nearest Neighboors (KNN).

Utilizamos bibliotecas próprias para aprendizado de máquina que nos ajudaram a gerar os algoritmos de aprendizado, como a SkLearn.

Para usar os algoritmos, escolhemos instâncias encontradas no DataSet *"Car Evaluation"* que foi retirado do repositório da UCI.

Esse DataSet possui 6 atributos para classificação de carros : Preço de compra, preço de manutenção, número de portas, número de pessoas, tamanho da mala e segurança. Todos os atributos têm valores discretos.

Existem 4 classificações para os carros: Não aceitável, aceitável, bom e muito bom.

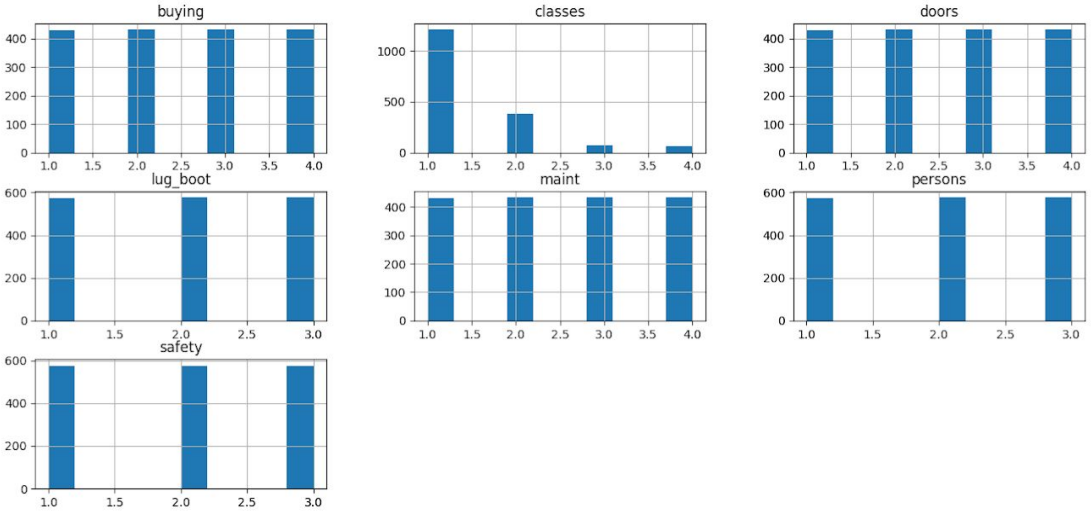
No arquivo de DataSet existem 1728 instâncias e esse grupo de instâncias cobre todo o espaço de possibilidades de combinações de atributos.

1. Descrição da modelagem dos exemplos de treinamento

1.1 Atributos selecionados para descrever os exemplos

Todos os atributos foram utilizados nos dois algoritmos implementados:

- buying
- maint
- doors
- persons
- lug_boot
- safety



1.2 Justificativa para a escolha dos atributos

Inicialmente fizemos a avaliação de quanto de informação cada atributo provê para a classificação. Utilizamos o Weka para ver mais facilmente esse ranking de atributos.

Verificamos que o atributo número de portas contribuía muito pouco. No entanto, ao retirarmos esse atributo e rodarmos nossos algoritmos (KNN e SVM), os resultados foram piores do que os com todos os atributos. Assim, optamos por manter todos os atributos no nosso treinamento.

2. Descrição dos experimentos realizados

2.1 Variações no conjunto treinamento e testes

Separamos o conjunto de teste do conjunto de treinamento randomicamente, usando uma proporção de 80% para treinamento e 20% para testes.

Variamos essa proporção para os dois algoritmos (KNN e SVM), no entanto, nos dois casos, quando colocávamos a proporção de testes maior, os resultados pioravam um pouco, na faixa de 1-2%.

Não utilizamos uma proporção inferior a 20% para testes, pois o conjunto de testes ficaria muito pequeno e a confiabilidade da acurácia seria prejudicada.

Além disso, fixamos a nossa Seed de randomização para que os resultados fossem reproduzíveis e pudéssemos alterar nossos algoritmos, sem que a acurácia mudasse por causa da randomização e sim pela a eficiência de cada algoritmo.

2.2 Variação nos parâmetros dos algoritmos

- **KNN:**

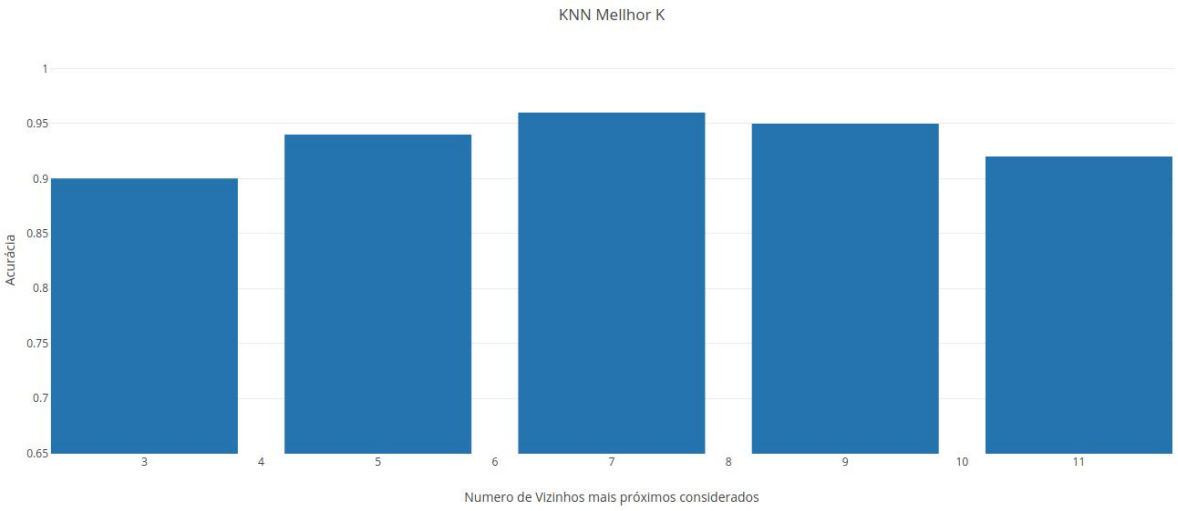
  No algoritmo de KNN, alteramos o nosso K para ver qual opção dava o melhor resultado.

  Ao final, chegamos a conclusão de que o K = 7 dava o melhor resultado.

  Abaixo, os testes com cada K testado:

  pfsmagalhaes@ncc17154:~/Documents/Puc/IA/IA-T2$ python knn.py

  score 0.9050925925925926 k 3
  score 0.9444444444444444 k 5
  score 0.9606481481481481 k 7
  score 0.9467592592592593 k 9
  score 0.9259259259259259 k 11
  melhor resultado para a Knn:
  Score = 0.9606481481481481, Numero de vizinhos = 7



- **SVM:**

  Na biblioteca que utilizamos para o SVM, tínhamos que escolher o kernel, o C (variável de folga) e o gama(usado no kernel gaussiano).

  O kernel é a função que projeta o espaço de entrada no espaço de características.

  Nos nossos testes variando os parâmetros, usamos o Kernel linear primeiro, depois o polinomial e por último o gaussiano. Os melhores resultados foram obtidos com o gaussiano.

  Abaixo, os testes com todos os tipos de kernel:

  linear:
  tempo de treino: 0.034101009368896484s
  tempo de classificação dos casos de teste: 0.0020453929901123047s
  Misclassified samples using SVM are: 38
  Classification Accuracy of SVM is 0.8901734104046243

  polinomial:
  tempo de treino: 0.5070457458496094s
  tempo de classificação dos casos de teste: 0.0015845298767089844s
  Misclassified samples using SVM are: 21
  Classification Accuracy of SVM is 0.9393063583815029

  gaussiano:
  tempo de treino: 0.036444664001464844s

tempo de classificação dos casos de teste:
0.004890918731689453s
Misclassified samples using SVM are: 1
Classification Accuracy of SVM is 0.9971098265895953

Quanto ao C, testamos com valores de 0.5 à 10 e o melhor encontrado foi 7.

Com relação à gama testamos valores de 0.1 até 2.0 e o melhor encontrado foi 0.4.

Importante ressaltar, que variamos o C e o gama juntos, tentando encontrar a melhor combinação e a melhor encontrada foi C = 7 e gama = 0.4.

3. Comparação dos algoritmos analisados

3.1 Descrição de como os algoritmos escolhidos foram implementados

Inicialmente, pegamos os dados do DataSet de um arquivo .csv e geramos um DataFrame (estrutura do python).

Desse DataFrame, separamos os casos de teste dos de treino utilizando uma biblioteca que separa os dados randomicamente de acordo com uma porcentagem desejada. No nosso caso, utilizamos 80% para treino e 20% para teste.

Todos os algoritmos utilizam essa estrutura.

- **KNN**

Para implementar o algoritmo KNN, usamos a biblioteca SkLearn, que tem uma classe própria para esse algoritmo.

O construtor da classe recebe um K (quantidade de vizinhos mais próximos) e com esse objeto, fazemos o treinamento utilizando a função “fit” do objeto, passando os casos de treino já separados anteriormente.

Em seguida, fazemos a classificação dos casos de teste usando o classificador KNN treinado.

A biblioteca SkLearn provê também uma função para testarmos a acurácia do classificador. Logo, usamos essa função passando o resultado da classificação dos testes com o resultado esperado.

Fizemos um loop que testa vários K’s diferentes para que pudéssemos escolher o K que gerasse o melhor resultado (maior acurácia).

- **SVM**

  Primeiramente, fizemos a standartização dos atributos, para que eles ficassem de 0 a 1.

  Depois utilizamos a classe do SkLearn própria para o SVM. No contrutor da classe, passamos o kernel, o C e o gama.

  No kernel, utilizamos o “rbf” que é o gaussiano, com um C igual 7 e gama igual a 0.4, pois com esses parâmetros atingimos melhores resultados.

  Em seguida usamos a função “fit” da classe para fazer o treinamento do classificador SVM.

  No final, fazemos a classificação dos casos de teste e utilizamos a função de acurácia para comparar com os resultados esperados.

3.2 Acurácia com os melhores parâmetros

KNN (%)	**SVM (%)**
96,06	99,71

3.3 Tempo gasto no processo de treinamento

KNN (s)	**SVM (s)**
0.0007	0.04

3.4 Tempo gasto no processo de classificação de todos os casos de teste

KNN (s)	**SVM (s)**
0.0002	0.005

4. Comparação dos algoritmos analisados

4.1 Escolha dos classificadores

Escolhemos o KNN pois foi explicado em aula e achamos interessante e mais fácil de entender o funcionamento do método.

Já o SVM escolhemos por ser aparentemente mais poderoso o que foi percebido nos resultados encontrados, apesar de ser um método de mais difícil entendimento.

Para o nosso DataSet e com nossos testes, o melhor classificador foi o SVM com uma precisão de 99%, errando apenas 1 classificação nos casos de teste. Para obter esse resultado, utilizamos todos os atributos do DataSet.

4.2 Conclusões finais

Todos os algoritmos são muito dependentes das características de cada problema. Dependendo do problema, um algoritmo pode ser melhor que o outro.

A acurácia variou bastante na variação dos parâmetros de cada algoritmo. No caso do KNN isso pôde ser visto quando alteramos o número de vizinhos K e no caso do SVM quando alteramos o kernel,C e gama.

Para o nosso DataSet, o SVM obteve os melhores resultados errando apenas 1 classificação.