Documentação TP2 IIA
Victor Augusto de Lima Cruz
2020054854

1. Introdução

O trabalho consiste na implementação e utilização de 2 algoritmos de aprendizado, um supervisionado (KNN) e outro não supervisionado (K-means). O KNN utiliza os k vizinhos mais próximos para realizar a classificação de um elemento. O K-means cria k centróides que permitem a divisão dos dados em k clusters.

Os dados em que esses algoritmos serão utilizados são os dados de estatísticas de novos jogadores na NBA, são 1340 observações. São usados 19 atributos para realizar as classificações e agrupamentos e tem-se interesse em descobrir se um jogador ficou pelo menos 5 anos na liga, uma label presente nos dados.

2. Algoritmos implementados

2.1 KNN (K Nearest Neighbors)

O KNN implementado calcula a distância euclidiana do ponto sendo classificado para todos os pontos de treino. Após isso ele escolhe os k vizinhos mais próximos e classifica com base na maioria das classes deles, caso haja um empate o default é a classe 0.

As distâncias euclidianas de um ponto para todas foram calculadas de uma vez usando operações com array em broadcast da biblioteca numpy. Isso melhorou bastante o desempenho do código.

2.2 K-Means

O K-Means implementado escolhe k instâncias do dataset aleatoriamente para serem os primeiros centróides. Após isso entra em loop, associando cada instância no dataset ao centróide mais próximo (distância euclidiana), depois disso os centróides são recalculados com base na média dos atributos das instâncias associadas à eles. Esse processo é repetido até que a classificação das instâncias seja a mesma antes e depois do recálculo do centróide.

Os centróides foram mantidos em matrizes (k, d) onde k é o número de centróides e das dimensões deles. Para calcular as classificações foi usada uma matriz (k, n) onde n é o número de instâncias, essa matriz contém a distância de cada instância para cada centróide, a classificação de cada instância (coluna) é o índice do menor dos valores das linhas.

As distâncias euclidianas do centróide para todas foram calculadas de uma vez usando operações com array em broadcast da biblioteca numpy, assim como no KNN. Isso melhorou bastante o desempenho do código.

3. Apresentação e discussão dos resultados

3.1 KNN

Os experimentos foram feitos com os dados normalizados já que dessa forma todos os atributos tinham média 0 e desvio padrão 1. Isso foi feito para que os atributos tenham a mesma influência sobre o resultado já que as distâncias euclidianas podem considerar mais os atributos maiores em módulo.
As instâncias de teste foram classificadas usando todas as de treino, para os valores de k iguais a 2, 10, 50 e 80.

Tabela de comparativa

k	Acurácia	Precisão	Revocação	F1
2	0.5858	0.7568	0.5350	0.6269
10	0.6754	0.7396	0.6906	0.7143
50	0.6567	0.7111	0.7273	0.7191
80	0.6642	0.7216	0.7135	0.7175

Os resultados podem ser vistos na tabela acima. A maior precisão foi obtida com o k = 2, a maior acurácia foi obtida com o k = 10. Ambos a revocação e o F1 foram melhores com o k=50.
Os melhores resultados foram com k=10 e com k=50, cada um deles apresenta algumas das métricas melhores e devem ser escolhidos com base na prioridade da tarefa.
A Acurácia e Precisão melhores com k=10 significam que respectivamente, que classificou mais dados corretamente e que acertou mais dos que classificou como positivo.
A Revocação e F1 melhores com k=50 significam que respectivamente, que classificou mais positivos corretamente dentre todos deles e que a média entre revocação e precisão foi melhor nesse caso.

Matrizes de confusão

k=2 Actual / Predicted	1	0
1	84	27
0	84	73

k=10 Actual / Predicted	1	0
1	125	44
0	43	56

k=50 Actual Predicted	1	0
1	128	52
0	40	48

k=80 Actual Predicted	1	0
1	127	49
0	41	51

As matrizes de confusão mostram os verdadeiros e falsos positivos e negativos.
Caso o predito for 1 e o real for 1, é um verdadeiro positivo;
Caso o predito for 1 e o real for 0, é um falso positivo;
Caso o predito for 0 e o real for 1, é um falso negativo;
Caso o predito for 0 e o real for 0, é um verdadeiro negativo.
É possível ver que k=2 previu melhor as classes negativas. e que k=10, 50 e 80 previu de maneira similar as positivas.

Comparação com Scikit

	Acurácia		Precisão		Revocação		F1	
k	Implementado	Scikit Learn	Implementado	Scikit Learn	Implementado	Scikit Learn	Implementado	Scikit Learn
2	0.5858	0.5858	0.7568	0.7568	0.5350	0.5350	0.6269	0.6269
10	0.6754	0.6754	0.7396	0.7396	0.6906	0.6906	0.7143	0.7143
50	0.6567	0.6567	0.7111	0.7111	0.7273	0.7273	0.7191	0.7191
80	0.6642	0.6642	0.7216	0.7216	0.7135	0.7135	0.7175	0.7175

Como é possível ver na tabela, os resultados obtidos com o KNN foram os mesmos na implementação do trabalho do que na biblioteca scikit-learn. A biblioteca por default usa a distância de minkowski com p = 2 que equivale a distância euclidiana. Isso ajuda a ter uma ideia de que a implementação feita no trabalho está correta.

3.2 K-Means

Nos experimentos foram feitos testes com os dados normalizados, já que dessa forma todos os atributos teriam média 0 e desvio padrão 1, e com a remoção de outliers, algum dos atributos está fora da média por mais de 3 vezes o desvio padrão. A intuição para a normalização é a mesma do KNN, como o algoritmo usa a distância para classificar as instâncias no clusters, ter os dados com média 0 e desvio padrão 1 ajuda a considerar todos igualmente. A remoção dos outliers foi feita pois atributos com valores muito diferentes da média podem fazer com que o centróide representante deles se mova em sua direção prejudicando a clusterização.
Apesar dessa intuição os resultados foram melhores sem a normalização, apenas com a remoção de outliers, portanto os resultados a seguir se referem aos experimentos dessa forma.

Além disso, por se tratar de um algoritmo não determinístico, foi escolhida uma semente para garantir que os experimentos seriam reprodutíveis.
Como o algoritmo é de clusterização não há separação entre treino e teste então o dataset foi concatenado para esta etapa.
Para avaliar a qualidade do k-means foram analisados:
Quantos pontos são associados a cada centróide;
Qual a classe mais frequente dos pontos associados ao centróide;
Considerando a classe mais frequente qual a porcentagem que foi agrupada corretamente.
Além disso, considerando que cada ponto agrupado em cada centróide foi classificado pela classe mais frequente também foram calculada a Acurácia, Precisão, Revocação e F1.

k=2

Pontos associados a centróide	Classificação	% prevista corretamente
751	1	0.7696
589	0	0.5705

k=3

Pontos associados a centróide	Classificação	% prevista corretamente
510	1	0.7667
364	0	0.6236
466	1	0.6502

Tabela de comparativa

k	Acurácia	Precisão	Revocação	F1
2	0.6821	0.7696	0.6324	0.6943
3	0.6873	0.7111	0.7535	0.7317

Como é possível ver nas tabelas, apenas a precisão foi melhor com o k = 2. As demais métricas foram melhores com o k = 3. Dessa forma é possível dizer que o valor de k = 3, ou seja com 3 centróides, é melhor para clusterizar os dados da maneira desejada.

Comparação com Scikit
Como há muitas dimensões nos centróides, não é possível comparar os resultados com o scikit learn por elas. Para isso foi calculado as distâncias euclidianas entre os respectivos centróides.

k=2

Distâncias	
centróide 1	≈ 0
centróide 2	≈ 0

k=3

Distâncias	
centróide 1	0.0482
centróide 2	0.0554
centróide 3	≈ 0

Como é possível ver nas tabelas, os centróides encontrados pela implementação são muito próximos aos encontrados pela scikit learn. Há uma pequena diferença, mas como o algoritmo não é determinístico, a proximidade dos centróides encontrada é suficiente para dizer que são bem similares.

	Acurácia		Precisão		Revocação		F1	
k	Implementado	Scikit Learn	Implementado	Scikit Learn	Implementado	Scikit Learn	Implementado	Scikit Learn
2	0.6821	0.6821	0.7696	0.7696	0.6324	0.6324	0.6943	0.6943
3	0.6873	0.6881	0.7111	0.7118	0.7535	0.7527	0.7317	0.7317

Como visto nas distâncias os resultados com o k-means implementado e o da scikit learn são bem parecidos. Isso ajuda a ter uma ideia de que a implementação feita no trabalho está correta.

4. Conclusão

Durante o trabalho foi possível ver como pequenas mudanças de implementação como o cálculo das distâncias com operações de matrizes ao invés de uma por uma pode melhorar o desempenho do código. Além disso, foi possível entender melhor como as métricas de avaliação estudadas funcionam e como a avaliação dos algoritmos podem ser feitas. Também foi possível aprender a usar versões dos algoritmos disponíveis em bibliotecas que podem ser usados de maneira simples.
Por último foram implementados os algoritmos vistos em aula, tornando os tópicos estudados mais palpáveis e gerando uma maior compreensão dos temas abordados.