Implementação do algoritmo k-NN

Alexandre A. Scrocaro Junior[1], Caio L. Araujo Miglioli[1]

[1]Bacharelado em Ciência da computação - Universidade Tecnológica Federal do Paraná (UTFPR); Campus Campo Mourão.

alexandre.2001@alunos.utfpr.edu.br, caiomiglioli@alunos.utfpr.edu.br

Resumo. *Foi implementado o algoritmo KNN utilizados conjuntos de treino e teste com manipulações das instâncias de entrada para buscar uma otimização dos dados utilizados*

1. Implementação

Com o intuito de facilitar o reuso do mesmo conjunto de treino em diferentes circunstâncias, o k-NN foi implementado em formato de classe, sendo seus principais aspectos o construtor, e os métodos testset() e classify().

1.1 Construtor

O método construtor é importante nessa implementação pois realiza todo o processo de tratamento do conjunto de treino, desde a captação dos dados e criação da tabela, seleção de um subconjunto dos dados e também a normalização dos dados. O método construtor executa o método privado __openFile() que generaliza todo esse processo.

No openFile, para captar os dados foi utilizado regex, que garante ao sistema abrir um arquivo com qualquer quantidade de atributos. É também permitido dizer em qual coluna contém a classe, para que o programa possa separar em um vetor essas informações, realizar a normalização das colunas utilizando Min-Max e por fim, concatenar novamente as classes a suas respectivas linhas, retornando uma matriz Numpy.

1.2 Classify

O método classify() é responsável por comparar um vetor de atributos (desde que contenha o mesmo número de atributos que o treino) com o conjunto de treino, e por fim retornando qual a classe que esse vetor pertence. É aqui que o kNN propriamente dito, é implementado.

Primeiro, é calculada a distância da entidade para cada item no conjunto de treino e guardado em um vetor auxiliar. Após o término, o vetor auxiliar é ordenado utilizando o método sort() nativo das listas do Python e selecionado as K primeiras instâncias. Então é utilizado o método value_counts() da classe Series da biblioteca Pandas para realizar a contagem de cada classe no subvetor resultante. Como o value_counts() retorna uma lista ordenada, o index 0 contém sempre o valor mais comum no subvetor, ou seja, a classe que desejamos encontrar.

1.3 Test Set

Por fim, temos o método para realizar o teste de um conjunto a fim de detectar a taxa de acerto do conjunto, este é o testSet(). O testSet() recebe o nome de um arquivo, o valor

de K, e o tipo de distância a ser utilizado, e retorna a taxa de acerto ao comparar o conjunto de teste com o conjunto de treino.

Sua implementação é simples, é utilizado o método __openFile() para gerar uma matriz numpy já normalizada do conjunto de teste, e então para cada instância do conjunto de teste, é chamado o método classify() para determinar a classe. Caso a classe determinada pelo algoritmo seja a mesma extraída do conjunto de teste, é detectado um acerto. Ao final, é feito a taxa de acertos pela quantidade de instâncias, e retornado a porcentagem.

2. Resultados

Para obter os resultados, os dados foram manipulados conforme foi explicado no tópico 1. Na Tabela 1 estão os resultados obtidos com um conjunto de características que consiste na quantidade de pixels brancos e pretos na imagem e ela é dividida em nove quadrantes; na Tabela 2 estão os resultados obtidos com um conjunto de características aleatórias da imagem.

k	25% do treino usado		50% do treino usado		100% do treino usado	
	distância euclidiana	distância manhattan	distância euclidiana	distância manhattan	distância euclidiana	distância manhattan
1	92.4 %	92.6 %	94.8 %	94.7 %	95.0 %	95.2 %
3	92.8 %	91.5 %	95.6 %	95.3 %	96.0 %	95.4 %
5	91.7 %	90.2 %	94.2 %	93.8 %	95.7 %	95.4 %
7	90.7 %	90.3 %	95.1 %	94.8 %	95.8 %	95.1 %
9	92.1 %	91.8 %	93.9 %	94.3 %	95.6 %	95.4 %
11	80.6 %	81.4 %	94.7 %	94.0 %	95.8 %	95.2 %
13	84.7 %	84.6 %	93.1 %	93.2 %	95.7 %	95.5 %
15	83.2 %	81.9 %	92.0 %	90.8 %	95.7 %	95.6 %
17	84.7 %	83.7 %	94.0 %	93.6 %	95.7 %	95.6 %
19	84.7 %	83.2 %	90.3 %	89.7 %	95.7 %	95.5 %

Tabela 1. Resultados do reconhecimento de dígitos utilizando KNN para o conjunto de features extraídas

k	25% do treino usado		50% do treino usado		100% do treino usado	
	distância euclidiana	distância manhattan	distância euclidiana	distância manhattan	distância euclidiana	distância manhattan
1	88.5 %	89.5 %	89.5 %	92.3 %	92.7 %	93.5 %
3	90.5 %	90.3 %	91.0 %	92.1 %	93.1 %	94.6 %
5	89.1 %	89.1 %	91.0 %	91.9 %	94.8 %	95.2 %
7	90.7 %	91.0 %	92.4 %	92.6 %	94.8 %	95.0 %
9	89.1 %	91.3 %	92.6 %	92.4 %	94.9 %	95.2 %
11	88.8 %	88.6 %	91.2 %	92.8 %	94.7 %	94.6 %
13	92.1 %	88.8 %	92.7 %	91.7 %	94.4 %	94.4 %

15	88.5 %	92.7 %	89.6 %	92.7 %	94.2 %	94.7 %
17	90.6 %	90.7 %	91.6 %	91.8 %	93.9 %	94.7 %
19	81.2 %	85.5 %	91.2 %	92.8 %	93.7 %	94.6 %

Tabela 2. Resultados do reconhecimento de dígitos utilizando KNN para o conjunto de features providas

4. Análise

Com base nos resultados das tabelas, pode-se inferir que aumentar a porcentagem de uso dos dados de treinamento tende a melhorar o desempenho do algoritmo, porém há um ponto no qual o ganho é ínfimo, logo deveria ser feita uma análise para encontrar uma fatia ideal para a quantidade de dados de treino. Por exemplo, entre os conjuntos de treino usados, o de 50% obteve resultados bem próximos ao de 100%. E a distância de Manhattan teve uma média melhor em relação à Euclidiana (x e y %, respectivamente). Já quando o k aumenta muito, os resultados tendem a diminuir. Além disso, como os conjuntos seguiram os mesmos padrões é possível afirmar que o KNN é um bom classificador de dígitos.

5. Conclusão

Em geral, os resultados mostram que usar mais instâncias no conjunto de treinamento tem um impacto positivo na taxa de acerto do algoritmo de classificação. À medida que a porcentagem de uso do conjunto de treinamento é aumentada, o modelo tem acesso a mais informações e pode aprender padrões melhores. Entretanto, é importante encontrar um ponto de equilíbrio, porque adicionar mais instâncias além deste ponto pode não significar uma taxa de acerto significativamente maior ou até mesmo levar a um *overfitting* dos dados de treinamento. Além de que com menos dados o modelo é otimizado.

References

Mineração de Dados. (s.d.). Distância Manhattan. Recuperado de https://mineracaodedados.wordpress.com/tag/distancia-manhattan/. Acesso em: 03 de maio 2023.