Implementação do k-Nearest Neighbor (k-NN) para o problema de reconhecimento de dígitos (0-9)

Rafael RampimSoratto[1]

[1] Universidade Tecnológica Federal do Paraná - UTFPR// (Campus Campo Mourão) - Brasil - Campo Mourão - PR

sorattol@alunos.utfpr.edu.br

Abstract. *The objective of this work is to propose the implementation of an algorithm that classifies an element based on its k closest neighbors (k-NN). Data entry are classes of numbers that have quantitative characteristics that can be analyzed to check the proximity of one element to another. The methodology uses a configuration with 10 values for k, using the euclidean and manhattan distance between the elements and the normalization of data of the type min _max and zscore. The training data of the algorithm are reduced in percentages to analyze the importance of a larger number of training data in relation to the accuracy of the algorithm. Using 25 % of the data we obtain the highest accuracy of 90 %. Using 50 % of the training data, we obtain the best accuracy of 91 %. Using 100 % of the training set, the best accuracy result is 92.60 %. Therefore, the number of test elements significantly improves the accuracy of the k-NN classification algorithm.*

Resumo. *O objetivo deste trabalho é propor a implementação de um algoritmo que classifica um elemento com base nos seus k vizinhos mais próximos (k-NN). A entrada de dados são classes de números que possuem características quantitativas que foram extraídas de uma imagem contendo o número de alguma classe (de 0 a 9). Essas características que foram obtidas das imagens podem ser analisadas para verificar a proximidade de um elemento com outro, ou seja, reconhecer a imagem de um dígito a partir de uma parecida com ela. A metodologia utiliza uma configuração com 10 valores para k. Utilizando k's vizinhos próximos pode-se calcular a distância euclidiana e de manhattan entre os vizinhos e a normalização de dados do tipo min_max e zscore. Os dados de treinamento do algoritmo são reduzidos em porcentagens para analisar qual a importância de maior número de dados de treinamento em relação a acurácia do algoritmo. Utilizando 25% dos dados obtemos a maior acurácia de 90%. Utilizando 50% dos dados de treinamento obtemos a melhor acurácia de 91%. Utilizando 100% do conjunto de treinamento o melhor resultado de acurácia é 92.60%. Portanto a quantidade de elementos de teste melhora significativamente a acurácia do algoritmo de classificação k-NN no problema de reconhecimento de dígito de 0 a 9.*

1 Requisitos do trabalho

1. Seu algoritmo deve avaliar o desempenho para diferentes valores de k 1,3,5,7,9,11,13,15,17,19 ;
2. Gerar a matriz de confusão ;
3. Usar a distância Euclidiana e Manhattan ;
4. Normalizar os dados com Min-Max e Z-score ;
5. Separar o conjunto de treinamento (aleatoriamente) em 25%, 50% e 100% dos dados de treinamento.
6. Avaliar qual o impacto de usar mais e menos instâncias no conjunto de treinamento.

2 Introdução ao k-NN

O algoritmo k-NN(k-Nearest Neighbor ou k Vizinhos mais próximos) trata-se de um algoritmo de classificação de dados clássico e muito simples. Ele assume que todas as instâncias correspondem a pontos em um espaço n-dimensional. de Aprendizagem Supervisionada, onde se encontra a 'boa resposta' durante o treinamento.

As vantagens de se utilizar o algoritmo k-NN é que trata-se de uma técnica simples e de fácil implementação, que em alguns casos apresenta ótimos resultados. Pode ser aplicada a problemas complexos, como: Análise de Crédito, Diagnósticos Médicos, Detecção de Fraudes, entre outros.

A desvantagens são: tempo; e ruíidos nos dados ou características irrelevantes podem "enganar" o algoritmo.

Na aprendizagem supervisionada:

- E possível ajustar os pesos em função das respostas corretas;
- O desafio é capacitar o sistema a atuar de acordo com o padrão observado nos exemplos de entrada e saída ;

3 Funcionamento do k-NN

Protocolo para funcionamento do algoritmo.

3.1 Entradas do algoritmo

1. Um elemento x no qual deseja-se classificar;
2. Um conjunto para treinamento ;
3. Uma métrica para calcular a distância entre x e as demais amostras;
4. Definir um valor para k, ou seja, quantos vizinhos iremos considerar (1,3,5,7,9,11,13,15,17,19).

3.2 Funcionamento do algoritmo

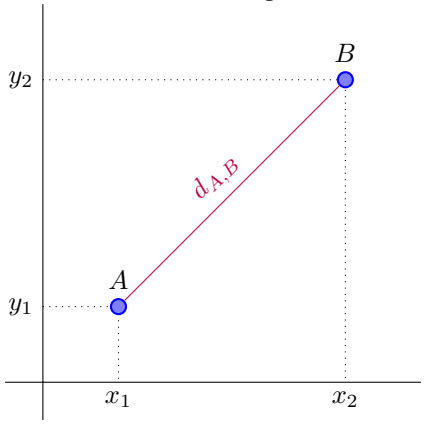
1. Inicialmente, calcula-se a distância entre o exemplo desconhecido x e todos os exemplos do conjunto de treinamento ;
2. Identifica-se os k vizinhos mais próximos;

3. A classificação é feita associando o exemplo desconhecido x à classe que for mais frequente, entre os k exemplos mais próximos de x;

   Utiliza o voto majoritário para definir a classe mais frequente.

3.3 Distância euclidiana

A distância euclidiana pode ser definida pelo gráfico:



$$d(A,B) = \sqrt{\sum_{i=1}^{n} (A_i - B_i)^2} \tag{1}$$

3.4 Distancia de manhattan

The manhattan distance between two points is defined as:

$$d(A,B) \equiv |A_x - B_x| + |A_y - B_y| \tag{2}$$

4 Normalizacao dos dados

Os termos padronizar e normalizar são usados indistintamente no pré-processamento de dados, embora nas estatísticas, o último termo também tem outras conotações.

A normalização dos dados tenta proporcionar a todos os atributos um peso igual. Normalização é particularmente útil para algoritmos de classificação envolvendo redes neurais ou medições de distância, como classificação de vizinho mais próximo (k-NN) e ”clustering”.

Para métodos baseados em distância, a normalização ajuda a prevenir atributos com intervalos inicialmente grandes de superação de atributos com intervalos inicialmente menores (por exemplo, atributos binários). Também é útil quando não é fornecido conhecimento dos dados.

Existem diversos métodos de normalização, neste trabalho serão utilizados os métodos Min-Max e Z-score. Para isto, utilizaremos um vetor com n elementos de $V_1 ... V_n$.

4.1 Normalização Min-Max

Normalização que executa uma transformação linear nos dados originais. Cada elemento do vetor é normalizado utilizando o valor máximo e mínimo do vetor. De acordo com a fórmula para definir cada elemento de um vetor A normalizado dentro de um intervalo $[0.0, 1.0]$:

$$v_i = \frac{v_i - min_a}{max_a - min_a} * 1 \tag{3}$$

4.2 Normalização z-score

Considera a média e o desvio padrão durante a normalização de acordo com a formula.

$$v_i = \frac{v_i - \overline{A}}{\sigma_A} \tag{4}$$

sendo $\overline{A}$ a média e σ_A o desvio padrão.

5 Matriz de confusão

Tabela que mostra as frequências de classificação para cada classe do modelo, neste caso serão as classes nomeadas de 0 até 9. A matriz de confusão pode ser gerada recebendo como parâmetros um array de valores reais $R[]$ e outro array de predições $P[]$. O resultado é a frequência de:

- **verdadeiros positivos (TP)** $(P[i] == 1 \&\& R[i] == 1)$,
- . **falsos positivos (FP)** $(P[i] == 1 \&\& R[i] == 0)$,
- **falso verdadeiro (TN)** $(P[i] == 0 \&\& R[i] == 0)$;
- **falso negativo (FN)**. $(P[i] == 0 \&\& R[i] == 1)$;

5.1 Acurácia

É o resultado da matriz de confusão que diz a percentagem de acerto das predições.

$$acurracy = \frac{TP + TN}{TP + FN + TN + FN} \tag{5}$$

6 Implementação

A implementação deste trabalho foi realizada utilizando a linguagem Python. A entrada do algoritmo são dois arquivos de dados de teste e de treino. De acordo com esses dados é possível criar instantes de teste para utilizar a seguinte configuração de nos dados de treino:

```
import dataframes as dataframes
import middlewares as middlewares
from scores import z_score, min_max

if __name__ == '__main__':
    middlewares.check_params()
    test_matrix = middlewares.load_test()
    training_matrix = middlewares.load_training
    ()
    instances_config = {
        'percents' : [0.25,0.5,1],
        'normalizations' : [min_max,z_score],
        'distances' : ['euclidean','manhattan'],
        'k' : [1,3,5,7,9,11,13,15,17,19]
    }
```

```
    instances =  dataframes.test_matrices(
    test_matrix, training_matrix,
    instances_config)
    middlewares.save_response(instances)
    print('Finish!')
```

6.1 Entrada de dados

Temos os conjuntos de dados nos arquivos de teste e treinamento, onde cada linha possui uma classe de dígito e os valores tabelados em relação as suas características.

6.2 Normalização e redução

Com base nesses valores de características em formato matricial é realizada a normalização dos dados e redução dos dados de acordo com a porcentagem para testar as matrizes e gerar a matriz de confusão de contém a acurácia do teste.

7 Resultados

Nota-se que a porcentagem total dos dados de treino variam, e são utilizadas porções de tamanho 25%, 50%, e 100% para analisar o impacto na acurácia do classificador em dados com maior volume de treino. São utilizados dados matriciais de teste e treino, com normalizações min_max e zscore, e distâncias euclideana e de manhattan. Utilizando os valores de $k = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]$.

Conforme descrita a configuração de teste proposta teremos o total de resultados igual a 40 por cada porcentagem de treino (25%, 50%, e 100%). Por exemplo, utilizando 25% de daos de treino, o k irá percorrer 10 vezes (do elemento 1 até o elemento 19), para cada k será utilizado a distância euclidiana e de manhattan e também min_max e zscore , logo o número de resultados é

$$resultados = 10 * 2 * 2 = 40 * 3 = 120$$

7.1 Formato dos resultados

Cada um desses 120 elementos de resultado possuem o seguinte formato

```
-----------------------------------------------
-----------------------------------------------
| K = 9 | Percent: 100 % |
| Normalization: z_score | Distance: manhattan |

| Hit rate: 92.60000000000001 % |
| Error rate: 7.40 %            |

        confusion_matrix:

        [94, 0, 0, 0, 0, 3, 1, 0, 0, 2]
        [0, 94, 3, 0, 0, 1, 0, 0, 1, 1]
        [0, 0, 89, 6, 0, 1, 1, 0, 3, 0]
        [1, 0, 3, 92, 1, 0, 2, 0, 0, 1]
        [0, 0, 0, 1, 96, 1, 1, 0, 1, 0]
        [0, 2, 0, 3, 0, 94, 1, 0, 0, 0]
        [0, 2, 0, 1, 0, 1, 96, 0, 0, 0]
        [0, 0, 0, 1, 1, 0, 0, 97, 0, 1]
        [4, 4, 0, 1, 1, 3, 2, 0, 80, 5]
        [0, 0, 0, 0, 2, 1, 1, 2, 0, 94]
-----------------------------------------------
-----------------------------------------------
```

8 Conclusão

De acordo com a acurácia dos testes foi feita a seguinte comparação:

- Utilizando 25% dos dados de treino, dentre os 40 resultados o melhor resultado de acurácia foi 90%.
- Utilizando 50% dos dados de treino, dentre os 40 resultados o melhor resultado de acurácia foi 91%.
- Utilizando 100% dos dados de treino, dentre os 40 resultados o melhor resultado de acurácia foi 92.60000000000001 %.