# Algoritmo k-NN

**Bruno Mendes de Souza[1], Mairieli Santos Wessel[1]**

[1]Departamento de Ciência da Computação – Universidade Tecnológica Federal do Paraná (UTFPR)
BR 369 – km 0,5 – Caixa Postal 271 – Campo Mourão – PR – Brazil

{brunosouza,mairieliw}@alunos.utfpr.edu.br

***Resumo.*** *O algoritmo k-NN (k-Nearest Neighbor) classifica uma instância de acordo com as classes dos k (k ≥ 1) vizinhos mais próximos, pertencentes a uma base de treinamentos dada. Este artigo descreve a implementação do algoritmo k-NN, utilizando a distância Euclidiana e Z-score, a fim de avaliar seu desempenho para diferentes valores de k. A avaliação do impacto de usar mais ou menos instâncias no conjunto de treinamento também será avaliada.*

## 1. Introdução

O algoritmo de classificação baseado no vizinho mais próximo NN (Nearest Neighbor) é amplamente utilizado para reconhecer padrões. A base de seu funcionamento está em encontrar o vizinho mais próximo de uma determinada instância. O algoritmo k-NN (k-Nearest Neighbor) pertence a um grupo de técnicas onde são encontrados os k vizinhos mais próximos, ao invés de apenas um vizinho mais próximo.

O k-NN (k-Nearest Neighbor) classifica uma instância de acordo com as classes dos k (k ≥ 1) vizinhos mais próximos, pertencentes a uma base de treinamentos dada. O algoritmo calcula a distância da instância para cada um dos elementos do conjunto de treinamento, e então os ordena do mais próximo ao mais distante. Dos elementos ordenados selecionam-se apenas os k primeiros, que servem de parâmetro para a regra de classificação.

A regra de classificação e a função que calcula a distância entre duas instâncias são dois pontos importantes no algoritmo k-NN. A regra de classificação diz como o algoritmo vai tratar a importância de cada um dos k elementos mais próximos. À função de distância cabe a tarefa identificar quais são os k-NN.

## 2. Materiais e Métodos

O algoritmo k-NN foi implementado utilizando a distância Euclidiana e para a normalização o Z-Score. A normalização dos dados com Z-score tem como base a média e o desvio padrão da característica. Para o utilizar o Min-Max é necessário saber os possíveis valores máximos e os valores mínimos de cada característica, já no Z-Score não é necessário. Escolhemos utilizar o Z-Score porque não sabemos qual valor máximo e mínimo que cada característica pode ter.

O algoritmo implementado inicia normalizando o conjunto de treino e teste com Z-Score. Logo após é selecionado o conjunto de instâncias para treino. Para cada uma das instâncias do conjunto de teste, calcula-se a distancia euclidiana no conjunto de treino. São escolhidos os k vizinhos com menor valor de distância para cada uma das

instâncias de teste. Define-se a classe de cada uma das instâncias pelo maior número de classes dos k vizinhos e o resultado é incrementado na matriz de confusão.

Utilizando a matriz de confusão calcula a taxa de acerto do algoritmo implementado: taxa = soma(diagonal_principal) / soma(total_matriz).

## 3. Resultados

Após a implementação do algoritmo, testamos o seu desempenho com k = 1, 3, 5, 7, 9 e 11 e utilizando 25%, 50% e 100% das instâncias para treino, os resultados serão apresentados abaixo.

**Tabela 1. Taxas de acerto do Algoritmo k-NN**

| **k = 1** | | **k = 3** | | **k = 5** | |
|---|---|---|---|---|---|
| 25% | 0.6483 | 25% | 0.6458 | 25% | 0.6550 |
| 50% | 0.6525 | 50% | 0.6842 | 50% | 0.6942 |
| 100% | 0.6675 | 100% | 0.7025 | 100% | 0.7192 |
| **k = 7** | | **k = 9** | | **k = 11** | |
| 25% | 0.6433 | 25% | 0.6867 | 25% | 0.6583 |
| 50% | 0.7025 | 50% | 0.6992 | 50% | 0.6925 |
| 100% | 0.7192 | 100% | 0.7250 | 100% | 0.7225 |

Segue a matriz de confusão para o melhor caso encontrado, com k=9 utilizando 100% das instâncias de treino.

**Tabela 1. Matriz de confusão para k=9 e 100% das instâncias de treino**

| | Jan | Fev | Mai | Abr | Jun | Jul | Ago | Set | Out | Nov | Dez |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Jan | **63** | 13 | 6 | 3 | 3 | 1 | 0 | 2 | 3 | 2 | 1 |
| Fev | 29 | **52** | 7 | 0 | 2 | 0 | 2 | 1 | 2 | 5 | 0 |
| Mai | 4 | 14 | **130** | 10 | 2 | 2 | 5 | 10 | 10 | 11 | 1 |
| Abr | 4 | 0 | 5 | **76** | 0 | 0 | 1 | 3 | 9 | 2 | 0 |
| Jun | 17 | 3 | 1 | 0 | **68** | 1 | 0 | 1 | 1 | 8 | 0 |
| Jul | 3 | 2 | 2 | 0 | 31 | **53** | 0 | 1 | 4 | 3 | 1 |
| Ago | 0 | 2 | 7 | 2 | 0 | 2 | **72** | 0 | 1 | 0 | 14 |
| Set | 2 | 1 | 8 | 0 | 2 | 0 | 0 | **75** | 7 | 5 | 0 |
| Out | 0 | 3 | 8 | 0 | 1 | 6 | 0 | 11 | **71** | 0 | 0 |
| Nov | 4 | 6 | 12 | 1 | 5 | 2 | 0 | 6 | 11 | **53** | 0 |
| Dez | 1 | 2 | 7 | 0 | 4 | 0 | 12 | 5 | 0 | 4 | **65** |

## 4. Conclusões

Desenvolvendo o algoritmo e analisando os resultados com k = 1, 3, 5, 7, 9 e 11 e utilizando 25%, 50% e 100% das instâncias para treino, a melhor taxa de acerto obtida é 0.7250, utilizando k = 9 e 100% das instâncias de treino.

Analisando os resultados contidos na Tabela 1, a qual contém as taxas de acerto do algoritmo k-NN, observamos que para todos os valores de k testados, utilizar um menor número de instâncias no conjunto de treinamento diminui as taxas de acerto.

## 5. Referências

Deepak Sinwar, Rahul Kaushik (2014) “Study of Euclidean and Manhattan Distance Metrics using Simple K-Means Clustering”.