Algoritmo K-NN

Noemi Pereira Scherer[1], Elaine Sangali[2]

[1]Departamento da Computação – Universidade Tecnológica Federal do Paraná (UTFPR)
Campo Mourão – PR – Brazil.

{noemischerer13@gmail.com, e.nani92@gmail.com

Abstract. *This meta-article describes the steps to programming an algorithm named by k-Nearest Neighbors algorithm (K-NN). The goal is use a group of sample and test of months of year to apply the Euclidian Distance and selecting the results between a k valor, witch is k nearest neighbors more closing of founded distance, storing in a confusion matrix.*

Resumo. *Este meta-artigo descreve os passos para implementar um algoritmo de classificação conhecido como k-Nearest Neighbors algorithm (K-NN). O seu objetivo é utilizar um conjunto de amostras e teste dos meses do ano para aplicar uma distância Euclidiana e selecionando o resultado através de um valor de K, que é K vizinhos mais próximos da distância encontrada, armazenando em uma matriz de confusão.*

1. Introdução.

K-NN é um método de classificação sub-ótimo, que não possui processamento na fase de treinamento, pois não é necessário estimar as distribuições de probabilidades das classes. É necessário um grande número de padrões de treinamento, pois as tarefas de estimativa e de classificação são fundidas em um único valor.

São utilizados cinco etapas, normalização das instâncias do treino e teste utilizando o Min-Max, o cálculo da distância Euclidiana de uma classe do teste para todas as de treino, retornando um valor no qual é encontrado o menor dentre eles atráves de um K vizinhos mais próximos, a classificação, ou seja, classificar a qual mês do ano o conjunto de treino pertence e verificar se está correta de acordo com o mês do conjunto teste. Por fim, é gerado uma matriz de confusão com a quantidade de acerto e erros.

2. Materiais.

Para a elabaração do código considera-se:

- Linguagem de programação Java;
- IDE NetBeans;
- Conjunto de teste do tipo data, com 1200 linhas, 24 instâncias para classe e 12 classes, uma para cada mês do ano;

- Conjunto de treino do tipo data, com 3600 linhas, 24 instâncias para cada classe e 12 classes, uma para cada mês do ano.

3. O código.

3.1. Visão geral.

Para cada conjunto de dados e teste, são armazenados em uma matriz, para facilitar o encontro dos valores, e normalizada cada uma delas utilizando o Min-Max. Para a normalização, é percorrido cada instância de cada linha da matriz, encontrar o menor e maior valor, denominado min e max respectivamente, e utilizar a fórmula de (valor atual – min)/(min-max).

É criado uma matriz de confusão, no qual a linha e colunas são valores de 0 até o 11, que representão os meses do ano (0 – Janeiro, 11 – Dezembro).

Percorre a todas as linhas da matriz de treino, e para cada instância de cada linha é calculado a distância Euclidiana em relação a todas instâncias de todas as linhas da matriz de treino, os resultados são armazenados em uma lista de distâncias associados a classe do mês que pertencem. A distância Euclidiana, é a raiz quadrada da subtração de cada valor da instância teste menos o valor da instância treino elevado ao quadrado.

Organiza-se as lista de distância em ordem crescente, verifica o valor de K passado pelo usuário para determinar o valor da lista. Por exemplo, se K=1, utiliza-se apenas o primeiro elemento da lista. Se K=50, utiliza-se as 50 preimeiras distâncias da lista, e verifica a classe do mês associada que possui maior frequência, se Janeiro é o mês que mais aparece nos 50 primeiros números, então a classificação é Janeiro.

Por fim, deve-se verificar se o algoritmo acertou, ou seja, se o mês associado a distâncias corresponde ao mês da linha da matriz de teste. Se sim, é salvo na matriz de confusão corresponde ao mês do teste e treino. Se não, também é salvo na matriz de confusão do mesmo jeito. Por fim, é calculado uma precisão total dos acertos.

3.1. Estrutura e Definições.

O código se encontra no Pacote nomeiado Knn, no qual possui duas classes Java, a principal que é o Knn.java e a auxiliar para organizar a lista de distâncias, NomeClasse.java.

- Knn.java: Possui nove métodos + o método principal main:
  - leDadosArquivo(): lê os dados do arquivo de teste e treino e armazena os valore em uma matriz de String;
  - geraMatrizConfusao(): Responsável por inicializar a matriz de confusão, com linhas e colunas correspondende as classes dos meses;
  - normalizaDados(): Recebe as matrizes de dados teste e coluna como paramêtro, aplica a normalização Min-Max e retorna uma nova matriz correspondente;

- calculaKnn(): Método mais importante do código, pois a ideia lógica do Knn começa nele. A sua função é percorrer todas as instâncias de cada linha do conjunto de teste, e calcular a distância Euclidiana para todas instâncias e linhas do conjunto treino (método knnParaLinhaTeste()). Assim que é recebido a classificação, ele encontra a precisão de todos os resultados chamando o metodo dadosFinais(), e imprime a nova matriz de confusão.
- knnParaLinhaTeste(): Recebe todas instâncias de cada linha do conjunto teste e aplica a distância Euclidiana para todas instâncias de todas as linhas do conjunto de treino, armazena os valores encontrados em uma lista e associa a classe do mês utilizando a classe NomeClasse.java. Organiza a lista em ordem crescente com a biblioteca do Java "Collections.sort". Recebe o valor do K, encontra a distância de acordo com K, verifica a qual classe pertence o valor encontrado e retorna a classificação final.
- calculaDistanciaEuclidiana(): Método complemento do knnParaLinhasTeste(). Ele aplica a distâcia Euclidiana para dois valores das instâncias.
- dadosFinais(): Responsável por atualizar a matriz de confusão e encontra a porcentagem de acertos (precisão). A cada acerto ou erro, ele incrementa 1 na classe correspondente da matriz de confusão.
- nomeiaClasse(): Método auxiliar que troca o mês por um valor equivalente. Por exemplo, se é Janeiro, retorna 0, Fevereiro, retorna 1.
- imprimeMatrizConfusão(): método que imprime a matriz de confusão na tela.

- NomeClasse.java: Dois atributos, o nome da classe a distância. Sua função é associar a distância a classe que pertence.

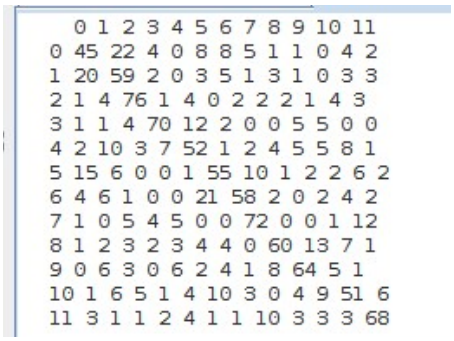
4. Análise e Resultados.

As porcentagens de acerto estão da casa dos 60%.

- Para K=1: Aproximadamente 60.83% (Figura 1).
- Para K=3: 62.083% (Figura 2).
- Para K=20: 64%.
- Para K=50: 63.58%.

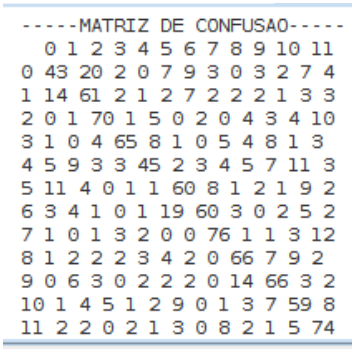
São separados aleatoriamente o conjunto de treinamento, obtivemos:

- 25% dos dados separados: Para K=1, apenas 18,58% de acertos. E Para K=20, 19.58% de acertos.
- 50% dos dados separados: Para K=1, acertou 33.91%. Para K=20, obtivemos 35.5% de acertos.



```
  0 1 2 3 4 5 6 7 8 9 10 11
0 45 22 4 0 8 8 5 1 1 0 4 2
1 20 59 2 0 3 5 1 3 1 0 3 3
2 1 4 76 1 4 0 2 2 2 1 4 3
3 1 1 4 70 12 2 0 0 5 5 0 0
4 2 10 3 7 52 1 2 4 5 5 8 1
5 15 6 0 0 1 55 10 1 2 2 6 2
6 4 6 1 0 0 21 58 2 0 2 4 2
7 1 0 5 4 5 0 0 72 0 0 1 12
8 1 2 3 2 3 4 4 0 60 13 7 1
9 0 6 3 0 6 2 4 1 8 64 5 1
10 1 6 5 1 4 10 3 0 4 9 51 6
11 3 1 1 2 4 1 1 10 3 3 3 68
```

Figura 1: Matriz de confusão: Para K=1



```
-----MATRIZ DE CONFUSAO-----
  0 1 2 3 4 5 6 7 8 9 10 11
0 43 20 2 0 7 9 3 0 3 2 7 4
1 14 61 2 1 2 7 2 2 2 1 3 3
2 0 1 70 1 5 0 2 0 4 3 4 10
3 1 0 4 65 8 1 0 5 4 8 1 3
4 5 9 3 3 45 2 3 4 5 7 11 3
5 11 4 0 1 1 60 8 1 2 1 9 2
6 3 4 1 0 1 19 60 3 0 2 5 2
7 1 0 1 3 2 0 0 76 1 1 3 12
8 1 2 2 2 3 4 2 0 66 7 9 2
9 0 6 3 0 2 2 2 0 14 66 3 2
10 1 4 5 1 2 9 0 1 3 7 59 8
11 2 2 0 2 1 3 0 8 2 1 5 74
```

Figura 2: Matriz de confusão: Para K=3

5.Conclusão

O Knn é um bom método, porém não é o melhor método de classificação, pois as taxas de acertos não são tão eficientes quanto deveria ser. Para obter o melhor resultado o K não deve ser muito pequene, mas também não muito grande, o ideal é entre a casa dos 10 aos 25. Quanto maior o número de linhas na classe treino, mais chance ele possui de acertar a classificação.

7. Referências

Campos, T. (2001) “Regra dos K vizinhos mais próximos”, http://www.vision.ime.usp.br/~teo/publications/dissertacao/node21.html, Agosto.