Universidade Federal do ABC

Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas

INF317B - Tópicos em Inteligência Artificial: Machine Learning

Implementação do algorítmo KNN e análise de desempenho.

Alexandre Miccheleti Lucena RA: 21201910023

Professor: André Kazuo Takahata
Professor: Ricardo Suyama

Março
2020

Conteúdo

1 Introdução

No contexto de engenharia, há uma crescente demanda na solução de problemas de reconhecimento de padrões, principalmente no intuito de automatizar processos e decisões. Nesse sentido, os avanços em técnicas de inteligência artificial, e mais especificamente no aprendizado de máquinas têm mostrado eficiência em abordar este tipo de problema.

A fim de introduzir os estudos neste tipo de abordagem, este trabalho apresenta um estudo do desempenho do algoritmo de classificação supervisionada denominado *k-Nearest Neighbors*, a partir de sua implementação mais básica e compreensão de seu funcionamento, aplicando-o à conjuntos de dados artificiais e utilizando-se de métricas de avaliação, e técnica de validação de dados.

2 Materiais e Métodos

Nesta seção é apresentado o algoritmo estudado, explicando o seu funcionamento, bem como os conjuntos de dados utilizados nos testes do mesmo. Também nesta seção, encontram-se detalhes sobre o ambiente de simulação onde foram desenvolvidos estes testes, e métricas utilizadas na validação dos resultados.

2.1 O Algoritmo *k*-NN

O algoritmo k-vizinhos mais próximos (*k-NN do inglês k-nearest neighbors*) é um algoritmo não paramétrico bastante conhecido e difundido no tratamento de problemas de reconhecimento de padrões e aprendizado de máquina. Um dos primeiros trabalhos a explorar o critério de menor distância no contexto de classificação foi (COVER; HART, 1967), trazendo detalhes de sua convergência e probabilidade de erro para o caso de 1 vizinho.

Como o nome sugere, um novo dado é classificado baseando-se na *distância* do mesmo relação à k outros dados (ou pontos) conhecidos. A localização de um dado do conjunto está associada às suas coordenadas dentro de um espaço de atributos e portanto representado por um ponto neste espaço. A

distância ou proximidade de um novo dado em relação aos k dados conhecidos no espaço de atributos (seus vizinhos) irá determinar à qual classe o mesmo pertence.

Em outras palavras, dado um conjunto de dados conhecido, o k-NN classifica um novo dado (vetor de características), comparando-o com a classificação de seus k pontos mais próximos (menor distância), atribuindo o novo dado à classe na qual a maioria destes k vizinhos pertence.

Apesar de sua simplicidade, é possível obter resultados interessantes quando aplicado a problemas de classificação, inclusive sendo capaz de atuar para conjuntos de dados não separáveis linearmente. A Figura 1 ilustra o comportamento do algoritmo para um conjunto de dados num espaço de dois atributos.

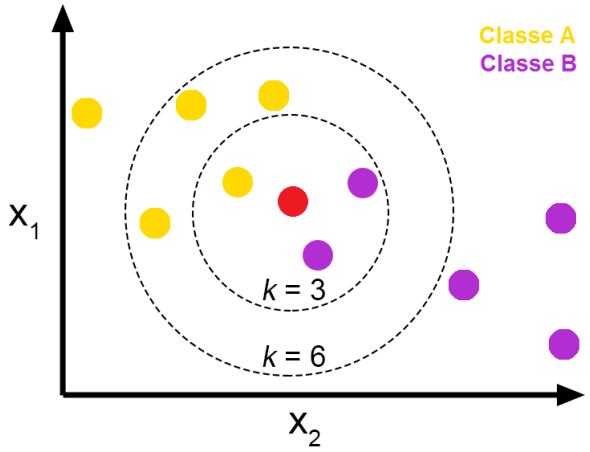

Figura 1: Representação do funcionamento do algoritmo k-NN.

O desempenho deste algoritmo está diretamente ligado à escolha do número de k-vizinhos que serão utilizados na comparação. Geralmente, a utilização de valores maiores para k, implica em uma suavização das fronteiras de decisão. Além disso, é recomendada a utilização de números ímpares para k evitando assim possíveis empates (DOUGHERTY, 2012). Uma desvantagem do algoritmo k-NN é sua susceptibilidade a ruídos locais, o que pode ser amenizado com um aumento do valor de k. Porém, isso também afeta a percepção de detalhes na geometria das fronteiras de decisão.

2.2 Conjunto de Dados

Os dados utilizados para os testes de classificação com o algoritmo k-NN, podem ser divididos em três partes: dados propostos pelo professor, dados gerados por funções, e o *dataset Olivetti Faces*.

2.2.1 Dados propostos pelo professor

Os primeiros conjuntos de dados utilizados referem-se a 6 arquivos no formato *.txt*, disponibilizados pelo professor, contendo valores numéricos associados a pontos em um espaço de atributos bidimensional acompanhados da identificação de qual classe pertencem. As distribuições dos pontos podem ser visualizadas na representação da Figura 2.

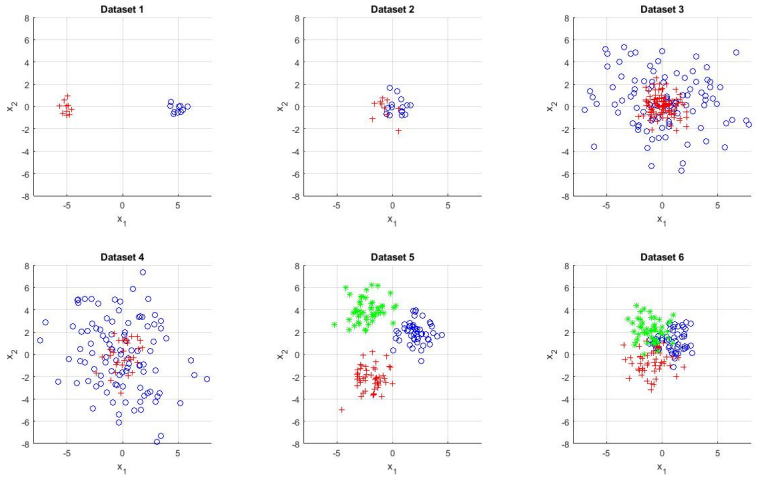

Figura 2: Ilustração dos pontos contidos nos *datasets* de 1 a 6.

2.2.2 Dados gerados por funções

A segunda parte dos conjuntos de dados são *datasets* artificiais gerados por um grupo de 6 funções criadas para o MATLAB por Jeroen Kools (KOOLS, J., 2020), que podem ser ajustadas conforme parâmetros de inicialização. Os códigos das funções originais podem ser encontrados nas referências. Para o uso neste trabalho, foram ajustados os parâmetros de

forma arbitrária, bem como feitas adaptações para atender a versão do MATLAB e numeração do identificador das classes (*labels*). A versão ajustada das funções e que foram utilizadas neste trabalho encontram-se no link para repositório disponível no Apêndice deste trabalho.

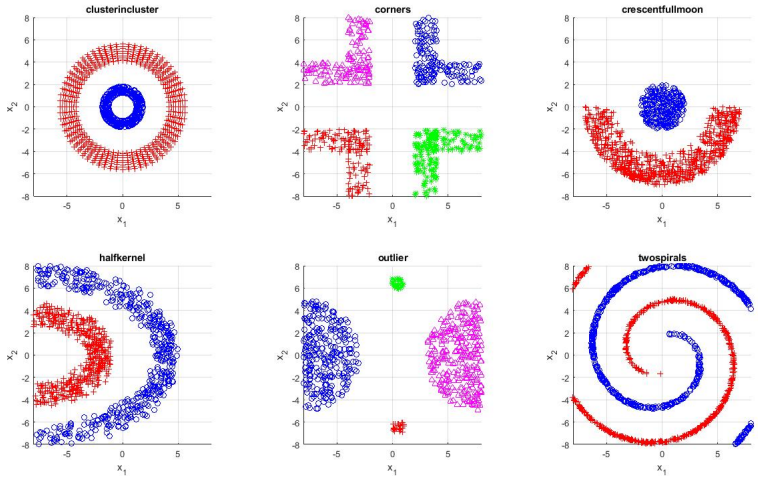

Figura 3: Ilustração dos pontos gerados através funções propostas.

2.2.3 *Dataset Olivetti Faces*

Neste trabalho, também foram feitos testes com um conjunto de dados composto for fotos de rostos, conjunto popularmente conhecido como *Olivetti Faces* (SAMARIA; HARTER, 1994). Este *dataset* é composto por 400 imagens de 64×64 pixels (10 imagens para cada um dos 40 sujeitos). A versão utilizada neste trabalho é uma versão disponível em (ROWEIS, S., 2020) adaptada para utilização no matlab e já possui um pre-processamento que rearranja as imagens em vetores de 1×4096. Algumas amostras deste *dataset* são apresentadas na Figura 4.

Figura 4: Exemplos de amostras das imagens contidas no *dataset Olivetti Faces*.

2.3 Ambiente de Simulação

O computador utilizado possui sistema operacional *Windows 10* de 64 bits, 16 GB de memória RAM e processador Intel Core i7-2600 CPU @ 3.4GHz. Todas as simulações foram implementadas dentro do ambiente do *software* MATLAB versão 2018a. Foram elaborados os códigos das funções que implementam o classificador k-NN, bem como um algoritmo de ordenação próprio que posteriormente terá seu desempenho comparado com a função de ordenação nativa do MATLAB.

2.4 Validação do Algoritmo

2.4.1 Método *Holdout*

A validação dos resultados foi feita através do método *holdout*. Este método consiste na divisão do conjunto de dados em duas partes: teste e treinamento. O conjunto de treinamento, é utilizado na etapa de aprendizado e ajuste de parâmetros do classificador, de forma treiná-lo para dados futuros. O conjunto de testes, simula a apresentação de novos dados, onde

a capacidade de generalização do algoritmo é testada e seu desempenho avaliado. O conjunto de dados pode sofrer uma subamostragem aleatória de seus elementos, de maneira a diversificar os resultados de diferentes conjuntos treino-teste. Dessa forma o desempenho é avaliado com base na média de diversas repetições do experimento.

A performance deste método de validação pode variar conforme o tamanho escolhido para a divisão de conjuntos treino-teste. Se o conjunto de treinamento for muito pequeno, é observada uma grande variância nos resultados do modelo. Por outro lado, se este conjunto torna-se muito grande, há pouca confiança na acurácia obtida (DOUGHERTY, 2012). Geralmente a partição reservada para o treinamento é maior que a reservada para a etapa de teste, onde o conjunto de treinamento consiste de 70% à 80% da quantidade total de amostras disponíveis. Neste trabalho optou-se por utilizar cerca 70% dos dados para treinamento e 30% para testes, para todos os casos. Na ocasião onde a divisão exata não foi possível, foi realizado um arredondamento para o próximo número inteiro.

2.4.2 Métricas de Avaliação

A principal forma de avaliar o desempenho do algoritmo utilizado foi a partir da construção de uma matriz de confusão. Conhecidas as classes existentes no conjunto de treinamento, essa matriz é construída de maneira a relacionar as entrada e saídas do algoritmo, contabilizando em cada posição da matriz a relação e entre a classe real de um dados, e a classe predita pelo algoritmo.

A matriz de confusão permite o cálculo de outras métricas, como acurácia, revocação, precisão e outras (DOUGHERTY, 2012). Algumas delas foram utilizadas também para avaliar o algoritmo e serão explicadas a seguir:

- **Acurácia:** Dado um conjunto de treino avaliado pelo classificador, a acurácia pode ser compreendida como a proporção de quantas amostras foram classificadas de maneira correta em relação à todas as amostras avaliadas; isto é, quantas das amostras avaliadas foram identificadas como pertencentes à classe na qual de fato pertencem. A Acurácia pode ser expressa por 1:

$$Acurácia = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

- **Precisão:** A precisão para uma determinada classe pode ser interpretada como a quantidade de acertos dentre todas as amostras que receberam aquela classificação. A Precisão pode ser calculada por 2:

$$Precisão = \frac{TP}{TP + FP} \tag{2}$$

- **Revocação:** A revocação para uma determinada classe é interpretada como a proporção entre a quantidade de acertos em relação a quantidade de amostras apresentadas ao classificador que realmente pertencem a esta classe. A Revocação pode ser obtida por 3:

$$Revocação = \frac{TP}{TP + FN} \tag{3}$$

3 Resultados e Discussão

O algoritmo foi implementado no ambiente MATLAB de maneira a ser possível selecionar dois tipos de algoritmo de ordenação: o *nativo* do software MATLAB e um algortimo de ordenação implementado manualmente (*Insertion Sort*). Apesar de ambas as opções, devido a ineficiência do algoritmo implementado manualmente e a grande quantidade de amostras nos dados, a simulação utilizando o *Insertion Sort* atingiu durações proibitivas para a conclusão deste relatório.

Por este motivo, os resultados contendo a utilização do *Insertion Sort* não são apresentadas. Todos os códigos utilizados nas simulações, bem como os conjuntos de dados estão disponíveis em um repositório do GitHub [1].

3.1 Dados propostos

Na primeira parte das simulações, foram realizados testes utilizando-se os conjuntos de dados propostos de 1 à 6, apresentados na seção 2. Num

[1] https://github.com/alemlucena/Projeto_KNN

primeiro momento, avaliou-se o tempo de execução do algoritmo para cada *dataset* e diferentes valores de k. Os resultados são apresentados na Tabela 1.

Tempo de execução (ms)					
k / Dataset nº	1	3	5	7	9
1	0.12	0.30	0.13	0.13	0.12
2	0.13	0.34	0.14	0.14	0.15
3	1.63	2.09	1.80	1.79	1.79
4	1.42	2.19	1.79	1.13	0.94
5	1.93	2.51	1.99	1.16	1.07
6	1.31	1.60	1.26	1.20	1.18

Tabela 1: Tempo de execução médio de 10 realizações, para cada dataset e diferentes valores de k.

Concomitantemente, foram calculados os valores de Acurácia, Precisão e Revocação, também para cada um dos datasets, a partir da média de 10 realizações para cada valor k. Estes resultados são apresentados nas Tabelas 2, 3 e 4, respectivamente.

Acurácia					
k / Dataset nº	1	3	5	7	9
1	100.00%	100.00%	100.00%	100.00%	100.00%
2	67.14%	80.00%	72.86%	74.29%	58.57%
3	76.33%	77.67%	78.67%	80.67%	79.00%
4	76.67%	73.33%	75.38%	72.31%	75.13%
5	98.00%	99.11%	99.11%	99.11%	99.78%
6	72.89%	75.56%	78.89%	83.33%	77.56%

Tabela 2: Acurácia média de 10 realizações, para cada dataset e diferentes valores de k.

Precisão					
k / Dataset nº	1	3	5	7	9
1	100.00%	100.00%	100.00%	100.00%	100.00%
2	70.02%	72.50%	78.17%	76.83%	69.93%
3	74.65%	84.36%	88.55%	94.37%	97.54%
4	82.92%	82.46%	87.78%	81.34%	82.75%
5	97.17%	98.14%	97.31%	97.76%	99.52%
6	66.24%	63.75%	72.44%	77.79%	66.28%

Tabela 3: Precisão média de 10 realizações, para cada dataset e diferentes valores de .

Revocação					
k / Dataset nº	1	3	5	7	9
1	100.00%	100.00%	100.00%	100.00%	100.00%
2	78.33%	96.33%	82.67%	90.00%	72.00%
3	77.07%	68.53%	68.31%	65.75%	61.47%
4	87.63%	81.11%	80.95%	82.60%	85.62%
5	96.24%	99.23%	100.00%	100.00%	100.00%
6	67.57%	80.64%	87.54%	86.80%	79.91%

Tabela 4: Revocação média de 10 realizações, para cada dataset e diferentes valores de k.

Foi possível também estimar as fronteiras de decisão referente ao classificador k-NN. A Figura 5 apresenta as fronteiras estimadas para diferentes valores de k no Dataset 3.

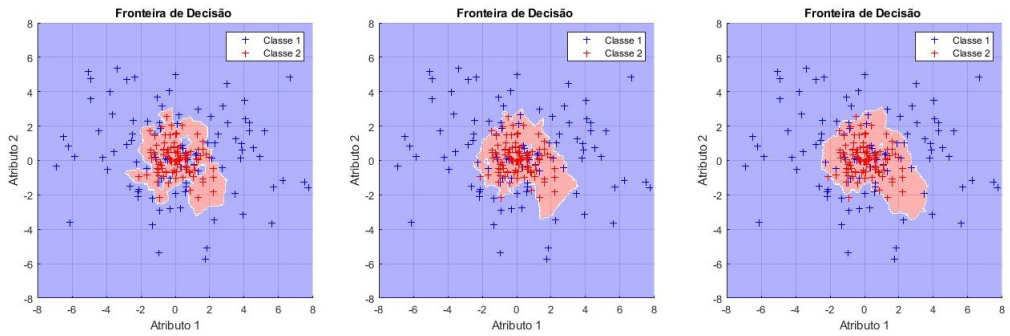

Figura 5: Comparação entre fronteira de decisão obtida no Dataset 3 para diferentes valores de k.

Da Figura 5 é possível observar a suavização da fronteira de decisão a partir do aumento de do valor k, que passa a ter maior influência quando o conjunto de dados apresenta a sobreposição.

3.2 Dados de funções

Na segunda parte das simulações, os testes foram feitos a partir dos conjuntos de dados gerados pelas funções MATLAB. O tempo de execução também foi mensurado, numa média de 10 realizações, e é apresentado na Tabela 5.

Tempo de execução (ms)					
Dataset nº \ k	1	3	5	7	9
clusterincluster	26.4	28.6	27.9	27.7	27.7
corners	27.4	28.8	27.1	26.4	26.7
crescentfullmoon	26.8	28.4	26.0	26.1	26.0
halfkernel	26.9	29.1	26.4	27.1	27.2
outlier	10.4	11.3	10.4	10.5	10.1
twospirals	102.2	105.5	103.3	102.9	105.0

Tabela 5: Tempo de execução médio de 10 realizações, para cada função e diferentes valores de k.

Para cada uma das funções, e diferentes valores de k, também foram realizados as medidas de Acurácia, Precisão e Revocação. Entretanto, todas

as realizações (sem exceção) obtiveram 100% para todas as métricas, o que pode ser justificado pela distribuição dos dados no espaço de atributos e suas fronteiras bem definidas. Por este motivo, os resultado não são apresentados em tabelas.

3.3 Olivetti Faces

Na última etapa das simulações, foram realizados os testes com o conjunto, novamente obtendo a média de 10 realizações para diferentes valores de k. Os resultados são apresentados na Tabela 6.

Olivetti Faces - Métricas					
k	1	3	5	7	9
Tempo (ms)	209.4	202.2	197.7	199.4	197.1
Acurácia	89.92%	81.67%	73.92%	70.33%	63.67%
Precisão	77.33%	-	57.33%	-	50.00%
Revocação	64.69%	64.00%	85.00%	50.00%	39.83%

Tabela 6: Tempo de execução médio de 10 realizações, para cada dataset e diferentes valores de k.

A partir da Tabela 6, podemos observar a influência do parâmetro k na acurácia obtida, de maneira que o melhor desempenho foi para $k = 1$. Isso pode ser justificado do ponto de vista da possível sobreposição entre os vetores de atributo do conjunto de dados, onde a utilização de mais vizinhos pode *confundir* o classificador. Afim de ilustrar este fenômeno, é apresentada a matriz de confusão na Figura 6 para $k = 1$, $k = 5$ e $k = 9$.

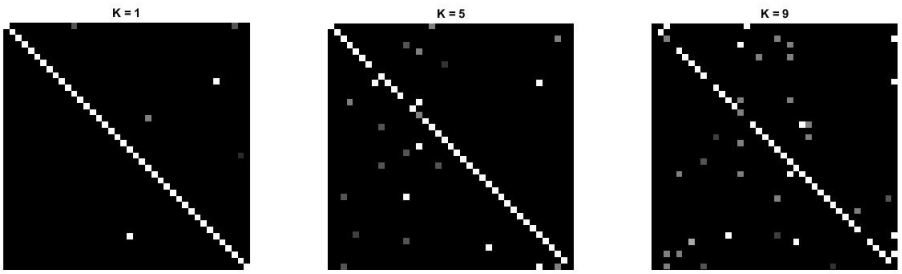

Figura 6: Comparação da matriz de confusão para diferentes valores de k - k=1, k=5, k=9 da esquerda para a direita, respectivamente.

4 Conclusão

A partir dos testes realizados, foi possível verificar o comportamento do algoritmo k-NN e avaliar seu desempenho através de métricas clássicas para problemas de classificação. É importe notar, que cada uma das diferentes métricas contribui para avaliar uma característica diferente do algoritmo. Pôde ser observado de maneira clara a influência da variação do parâmetro de número de vizinhos k, e que a escolha do mesmo está diretamente ligada ao tipo de conjunto de dados que é utilizado. Apesar de uma implementação simples, o desempenho do k-NN está fortemente associado à eficiência do algoritmo de ordenação utilizado na determinação das menores distâncias dos vizinhos, influenciando no seu tempo de execução. Assim, algoritmo provou sua capacidade de atuar em problemas de classificação não separáveis linearmente.

Referências

COVER, T.; HART, P. Nearest neighbor pattern classification. *IEEE transactions on information theory*, IEEE, v. 13, n. 1, p. 21–27, 1967.

DOUGHERTY, G. *Pattern recognition and classification: an introduction*. [S.l.]: Springer Science & Business Media, 2012.

KOOLS, J. *6 functions for generating artificial datasets*. [S.l.]: MATLAB Central File Exchange, 2020. ⟨https://www.mathworks.com/matlabcentral/fileexchange/41459-6-functions-for-generating-artificial-datasets⟩. Acesso: 08-03-2020.

ROWEIS, S. *Olivetti Faces*. 2020. ⟨https://cs.nyu.edu/~roweis/data.html⟩. Acesso: 08-03-2020.

SAMARIA, F. S.; HARTER, A. C. Parameterisation of a stochastic model for human face identification. In: IEEE. *Proceedings of 1994 IEEE workshop on applications of computer vision*. [S.l.], 1994. p. 138–142.