Mineração de Dados - tp 3
KNN

Thales Filizola Costa
thalesfc@dcc.ufmg.br

23 de janeiro de 2013

Sumário

1 INTRODUÇÃO

Classificação é o problema de identificar para qual de um conjunto de categorias uma nova observação pertence, com base no treinamento utilizando dados com observações cuja a categoria é conhecida.

O *k-nearest neighbor*(KNN), um dos primeiros algoritmos de classificação criados, é de fácil entendimento. A ideia do algoritmo é classificar uma nova instância com base nas instâncias de treino. Sabendo os k vizinhos mais semelhantes somos capazes de predizer a classe da nova observação.

Este trabalho prático propõem a utilização do algoritmo de *k-nearest neighbor*(KNN) para classificação de uma base de dados reais.

A seção 4 retrata a base de dados utilizada neste trabalho, tendo como foco a análise de stemming e de retirada de stopwords e relacionando os mesmos com os efeitos na acurácia e na performance do algoritmo. Na seção 5 apresentamos uma discussão sobre a forma de escolha do parâmetro k. A seção 6 retrata todos os experimentos executados.

2 ALGORITMO DE *K-NEAREST NEIGHBOR*

2.1 Descrição

O algoritmo de *k-nearest neighbor* é um método para classificação de objetos baseado na proximidade com exemplos de treinos em relação ao espaço de atributos. O funcionamento do algoritmo é apresentado abaixo:

1. Dado um conjunto de trainamento da forma:$(x_1, y_1), (x_2, y_2) \cdots (x_n, y_n)$.
2. Assumimos que $x_i = (fi_1, fi_2 \cdots fi_d)$ um vetor d dimencional de features, onde cada posição t é a frequência do téssimo termo no ponto i.
3. y_i é a classe $\{0, 1\}$ do ponto i.
4. A tarefa: determinar y_{new} para todos os x_{new}.

Algoritmo de kNN:

5. Encontre os k pontos mais próximos de x_{new}, usando uma métrica de distância, e.g. distância euclideana.
6. Classifique y_{new} = voto majoritário dentre os k pontos mais próximos.

Durante a execução do algoritmo de kNN, não é necessário estimar nenhum parâmetro, como por exemplo é feito no classificador Rocchio(centroides) ou no Naive Bayes(prioris e probabilidades condicionais). kNN simplesmente memoriza todos os exemplos de treino e compara o documento de teste com

eles. Por essa razão, kNN é chamado de *memory-based learnig* ou *instance-based learning*. É sempre interessante ter a maior quantidade de instâncias de treino possíveis, entretanto, no kNN ter essas instâncias é um custo para o classificador. Maiores detalhes serão apresentados à seguir, quando falamos sobre a complexidade do algoritmo.

2.2 Complexidade

Assim, o custo computacional do algoritmo de kNN para classificação de um simples ponto é $O(nd)$, onde n é o número de pontos no treinamento e d é a quantidade de dimensões do espaço de atributos, no nosso caso será o tamanho do vocabulário. O custo do algoritmo é esse pois para classificar um ponto, calculamos a distância desse ponto com todos os n pontos de treinamento. O custo de calcular a distância entre dois pontos é d, logo o custo geral é $O(nd)$.

3 MÁQUINA DE TESTE

Todos as execuções e testes do programa implementado foram executadas na máquina: `claro.grad.dcc.ufmg.br`

4 BASE DE DADOS

A base de dados utilizada consiste em *reviews* de usuários para filmes. Para cada *review* são fornecidos o comentário e a classificação do comentário, que pode ser positiva (1) ou negativa (0).

4.1 Vocabulário

Nessa subseção queremos determinar se, para os próximos experimentos, vamos utilizar ou não *stemming* e se vamos retirar as *stopwords*.

A tabela 1 apresenta um sumários das diferentes configurações para tratamento de palavras. Nesses experimentos, o número de vizinhos utilizados foi dez (`k=10`) e a semelhança entre os pontos foi calculada utilizando a distância euclideana (`d=euclidean`).

De acordo com a tabela acima, podemos ver que o stemming reduziu bem a dimencionalidade dos dados (próximo de $\frac{1}{3}$) com melhoria na acurácia, por isso será adotado. Entretanto, a remoção das *stopwords* não apresentou ganhos tão grandes de redução de dimencionalidade e levou a perda de acurácia, portanto essa prática não será adotada.

Stopwords	Stem	Tamanho do vocabulário	Acurácia	Tempo de execução∗
Não	Não	31892	0.6325%	88.33 s
Não	Não	31393	0.51%	85.89 s
Não	**Sim**	21562	0.6675%	63.81 s
Sim	**Sim**	21127	0.5175%	62.59 s

Tabela 1: Sumário das diferentes configurações para tratamento de strings. ∗ Média de 5 execuções

4.2 Distribuição das classes

A distribuição das classes determina a proporção de exemplos da partição que são das respectivas classes. Conforme podemos ver pela figura 1, a partição de treino é desbalanceada em relação ao teste, visto que as proporções entre as classes são completamente diferentes ($\frac{1}{3}$ no treino e 1 no teste).

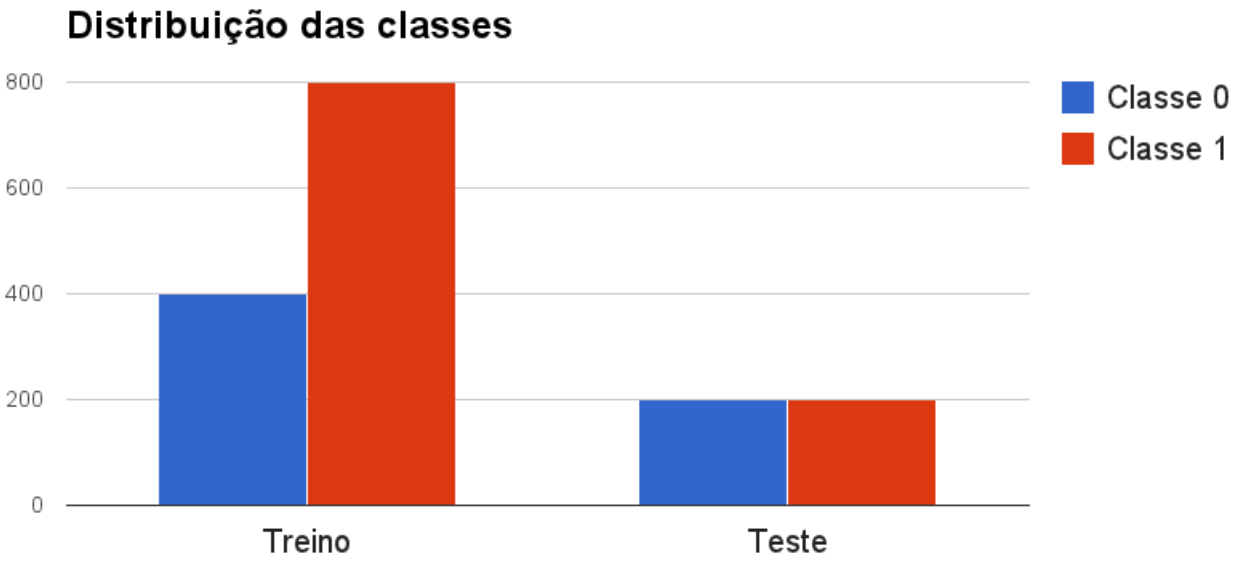

Figura 1: Distribuição das classes na base de dados.

Tal distribuição atrapalha o algoritmo de classificação, pois com o dado treino existe uma probabilidade maior de classificar os pontos como sendo da classe 1.

5 SELECIONANDO O VALOR DE K

O objetivo deste trabalho prático é simular um ambiente real de utilização do algoritmo de KNN. Como consequente não podemos utilizar a acurácia na partição de teste para escolhermos o melhor valor de k.

A solução é empregarmos a técnica de *leave-one-out* nos dados de treino utilizando diferentes valores de k, com isso somos capazes de estimar qual o valor de k que melhor se comporta para a base de dados.

5.1 Leave-one-out

A técnica de *leave-one-out* é apresentada abaixo, tal técnica foi escolhida devido à sua resistência à variância e sua facilidade de computar.

1. Para cada exemplo $e_i = (f, c)$ no treino:
   (a) Treino o classificador de KNN usando todos os pontos, menos e_i
   (b) Teste a acurácia do classificador para e_i
2. Repita o processo para todos os i pontos
3. Acurácia = média da acurácia de todos os pontos.

5.2 Resultados

A configuração dos parâmetros utilizada nesses testes foi: uso de stemming sem a retirarada das stopwords.

Usando a técnica de *leave-one-out*, testamos diversos valores de k usando distância Euclideana e de Jaccard.

As tabelas 2 e 3 apresentam respectivamente os resultados obtidos para as distâncias Euclideana e Jaccard. Dos diversos valores de k testados reportamos: no campo de treino (leave-one-out), a acurácia e o tempo de execução; no campo de teste, somente a acurácia. A figura 2 apresenta um resumo das acurácias na partição de treino.

Ainda analisando as tabelas 2 e 3, usando os valores de acurácia gerado pelo treino, $k = 30$ é o melhor valor, entretanto, analisando os valores gerados pela acurácia no teste, vemos que o valor de $k = 7$ obtem a melhor acurácia. A diferença no melhor valor de k para o traino e o teste é explicado pelo fato da distribuição de classes ser bem diferente.

	TREINO		**TESTE**
Valor de k	**Acurácia(%)**	**Tempo de execução (s)**	**Acurácia (%)**
k=1	0.6258	281.48	0.615
k=3	0.6433	282.09	0.6075
k=5	0.6775	280.63	0.6425
k=7	0.68	280.36	0.65
k=9	0.6958	285.94	0.63
k=15	0.68	284.73	0.6275
k=30	**0.6925**	**285.96**	**0.5925**
k=40	0.69	282.77	0.5725
k=50	0.695	285.1	0.5725
k=100	0.6825	300.13	0.5225

Tabela 2: *Leave-one-out* na partição de treino usando distância Euclideana.

	TREINO		**TESTE**
Valor de k	**Acurácia(%)**	**Tempo de execução (s)**	**Acurácia (%)**
k=1	0.653	759.91	0.605
k=3	0.683	753.78	0.6525
k=5	0.7075	751.28	0.65
k=7	0.7108	770.02	0.6875
k=9	0.7108	759.36	0.6775
k=15	0.7325	760.58	0.6625
k=20	0.7408	762.81	0.665
k=30	**0.7433**	**757.56**	**0.64**
k=40	0.7441	763.13	0.6375
k=50	0.7325	754.4	0.6225
k=100	0.7016	767.52	0.57

Tabela 3: *Leave-one-out* na partição de treino usando distância Jaccard.

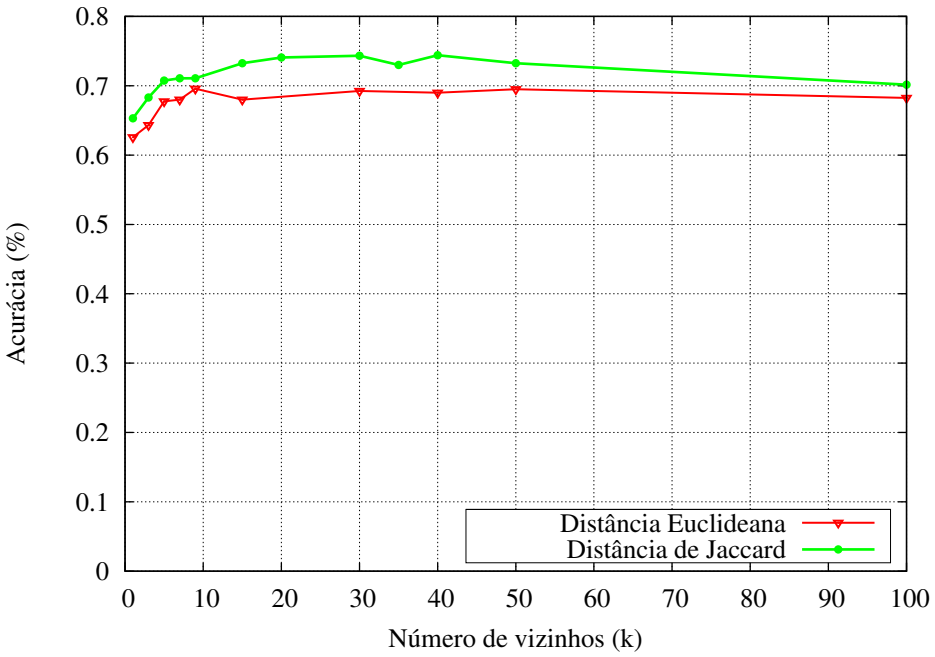

Figura 2: Gráfico com resumo da técnica leave one out.

6 EXPERIMENTOS

6.1 Métricas de Distâncias

Nessa seção vamos testar a qualidade de diversas métricas de distância no calculo do knn. A configuração dos parâmetros utilizada nesses testes foi: uso de stemming, sem a retirarada das stopwords e `k=7`.

Podemos ver pela figura 3 que os melhores resultados foram obtidos pelas distâncias Euclideanas e de Jaccard.

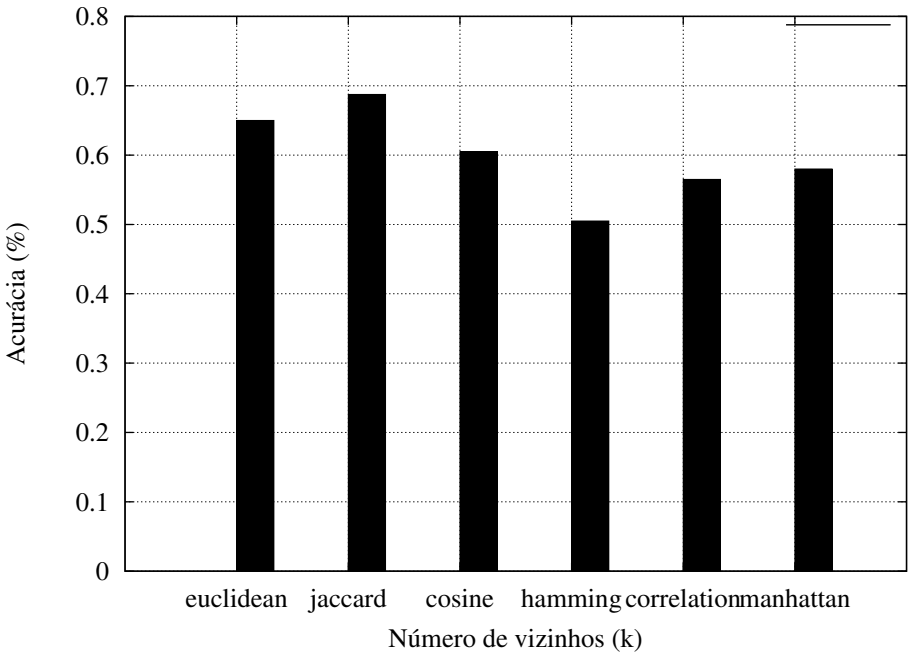

Figura 3: Gráfico com resultado final para diversas distâncias.

6.2 Variaçaõ no Tamanho de Entrada

Nessa seção vamos testar a estabilidade do algoritmo de knn com a variação no tamanho de entrada. A configuração dos parâmetros utilizada nesses testes foi: uso de stemming, sem a retirarada das stopwords, `k=7` e distância de jaccard.

A metodologia adota será utilizar i% do treino para classificar toda a base de teste. Sendo que o valor de i vai de 10% até 100%.

Podemos ver pelas figuras 4 e 5 que tanto o tempo de execução como a acurácia dos classificadores aumenta linearmente com a quantidade de dados no treino. Essas observações são importantes porque: (i) comprovam que complexidade do algoritmo é linear com o número de treinamento; (ii) mostram que a acurácia do algoritmo é proporcional ao número de treinamento. Logo se um pesquisador quiser melhorar a acurácia do classificador, uma maneira é investir em mais exemplos; e (iii) demonstra que existe um trade-off entre aumentar o número de treino para melhorar a acurácia, mas sofrer com o aumento do tempo de execução do algoritmo.

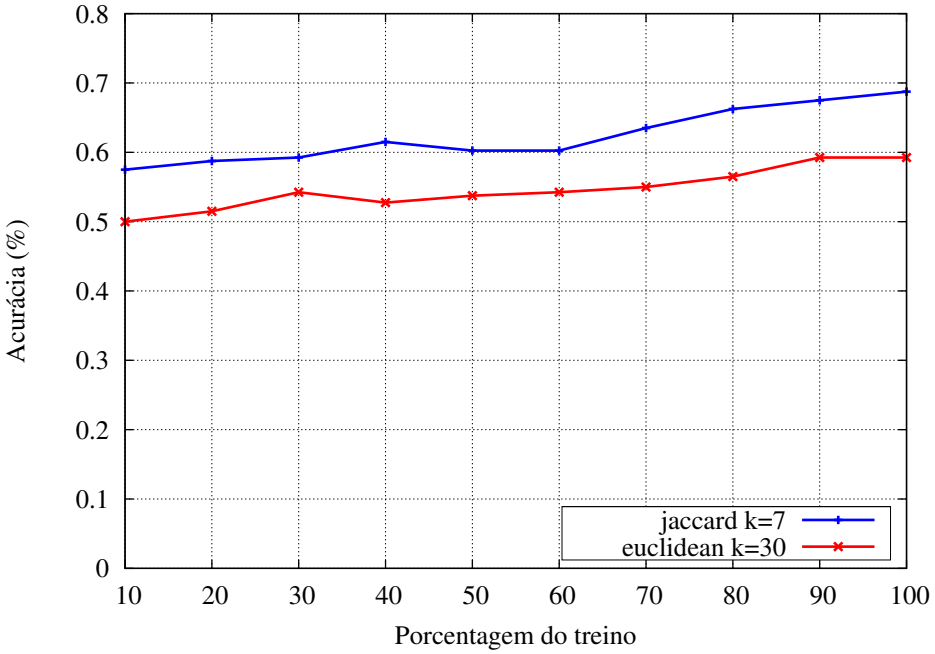

Figura 4: Acurácia dos classificadores para diferentes porcentagens de treino.

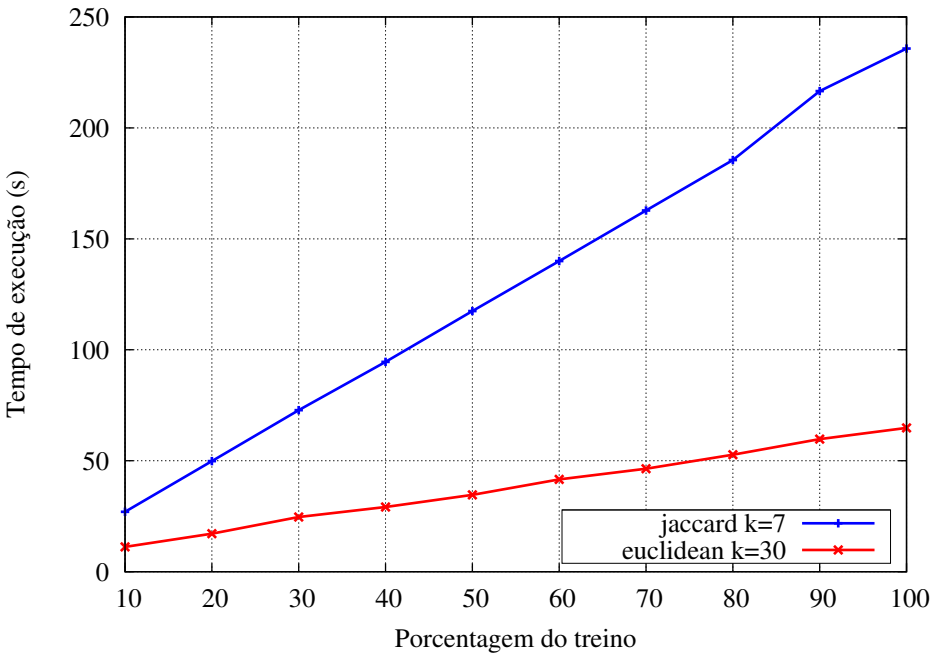

Figura 5: Tempo de execução dos classificadores para diferentes porcentagens de treino.

7 CONCLUSÃO

A realização deste trabalho prático permitiu algumas conclusões sobre o algoritmo de knn.

1. A distribuição das classses no treino é muito diferente da distribuição das mesmas no teste. Essa diferença nas distribuições tem muita probabilidade de acontecer na implementação de sistemas reais, ao longo desse relatório foi demostrado que esse efeito pode afetar a acurácia do classificador e deve ser levado em consideração em implementações.

2. O efeito do stemming na base de dados obteve uma surpreendente melhoria na acurácia, o que sugere que a substituição de sinônimos possa ser explorada como trabalhos futuros para uma melhora ainda mais significativa dos resultados.

3. A técnica de *leave-one-out* foi executada usando as distâncias Euclideana e de Jaccard. Tal técnica mostrou estabilidade ao obter melhores valores de k bem similares nas duas métricas. Entretando, devido à distribuição das classes ser bem diferente no treino/teste, os valores de acurácia obtidos (através do leave-one-out) não foram compatíveis com os valores obtidos no teste.

4. Os experimentos feitos com a técnica de leave-one-out mostram que o valor de k não afeta o tempo de execução do algoritmo. Além disso, os resultados obtidos com a variação no tamanho de entrada mostram que o algoritmo tem custo computacional linearmente proporcional ao tamanho do treino. Esses dois resultados comprovam a complexidade do algoritmo derivada na seção 2.2.

Referências

[1] Michael Hahsler. A Comparison of Commonly Used Interest Measures for Association Rules. `http://michael.hahsler.net/research/association_rules/measures.html`, 2011. [Online; accessed 13-October-2012].

[2] Mohammed Zaki and Wagner Meira Jr. *Fundamentals of Data Mining Algorithms*. Cambridge University Press, 2010.