Implementação de k-NN maximizado por Algoritmos Genéticos

Vitor Bueno de Camargo[1]

[1] Departamento de Computação – Universidade Tecnológica Federal do Paraná (UTFPR)
Caixa Postal 87301 – Campo Mourão – PR – Brazil

vitorcamargo@alunos.utfpr.edu.br

Abstract. *K-Nearest Neighbor (k-NN) is one of the most popular algorithms for pattern recognition. Many researchers have found that the k-NN algorithm performs very well in its experiments on different datasets. To overcome the algorithm limitations, an improved version of k-NN is proposed during this work. Genetic Algorithm (GA) is combined with k-NN to improve its classification performance, instead of considering all training samples and taking the k-neighbors, AG is employed by choosing the best combinations of features to choose k-neighbors and then calculate the distance to classify the test samples.*

Resumo. *O k-Nearest Neighbor (k-NN) é um dos algoritmos mais populares para reconhecimento de padrões. Muitos pesquisadores descobriram que o algoritmo k-NN realiza um desempenho muito bom em seus experimentos em diferentes conjuntos de dados. Para superar as limitações do algoritmo, uma versão melhorada do k-NN é proposta durante este trabalho. Algoritmo Genético (AG) é combinado com k-NN para melhorar seu desempenho de classificação, em vez de considerar todas as amostras de treinamento e tomar os k-vizinhos, o AG é empregado escolhendo as melhores combinações de features para escolher os k-vizinhos e depois calcular a distância para classificar as amostras de teste.*

1. Introdução

A busca por vizinho mais próximo é uma das técnicas mais populares de aprendizado e classificação introduzidas por Fix e Hodges, que se provou ser um algoritmo de reconhecimento simples e poderoso. Cover e Hart mostraram que a regra de decisão funciona bem, considerando que não há conhecimento explícito dos dados disponíveis. Contudo, apesar da boa decisão dada pelo algoritmo é possível melhorar seus resultados aplicando algoritmos que gerem as melhores características afim de aprimorar ainda mais o resultado final, um bom exemplo desse algoritmo é o genético, que seleciona (aleatoriamente ou eletivamente), recombina e realiza mutações em características para melhorar o desempenho.

2. Algoritmo k-NN

O algoritmo de classificação k-NN prevê a categoria da amostra de teste de acordo com as amostras de treinamento K, que são os vizinhos mais próximos da amostra de teste, e a julga para aquela categoria que possui a maior probabilidade de categoria. Neste algoritmo, a primeira ação realizada é selecionar um subespaço de característica ideal, que possui um poder discriminatório semelhante aos dados originais, mas o número de

recursos é bastante reduzido. Assim para cada instância que será testada pelo algoritmo é necessário buscar a distância dentre essa instância com o restante das instâncias de treinamento e selecionar os k elementos com a menor distância com o elemento testado. A partir deste ponto, o elemento testado é classificado pela moda dos seus k-vizinhos.

3. Algoritmo Genético

Um algoritmo genético (ou AG) é uma técnica de pesquisa usada na computação para encontrar soluções verdadeiras ou aproximadas para problemas de otimização e busca. Eles são categorizados como heurísticas de pesquisa global e são uma classe particular de algoritmos evolutivos que usam técnicas inspiradas na biologia evolutiva, como herança, mutação, seleção e crossover (também chamado de recombinação). A evolução geralmente começa a partir de uma população de indivíduos gerados aleatoriamente e acontece em gerações. Em cada geração, a aptidão (fitness) de cada indivíduo na população é avaliada, vários indivíduos são selecionados a partir da população atual (com base em sua aptidão) e modificados para formar uma nova população e a nova população é usada na próxima iteração do algoritmo.

O algoritmo termina quando um número máximo de gerações foi produzido, ou um nível de aptidão satisfatório foi alcançado para a população, embora não haja nenhuma regra de convergência ou garantia de alcançar o valor satisfatório!

O GA tem aplicações em campos tão diversos quanto design de VLSI, processamento de imagem, redes neurais, aprendizado de máquina, programação de job shop, etc.

4. Trabalho Proposto

O principal objetivo deste trabalho é utilizar Algoritmos Genéticos para encontrar quais são as características que otimizam a taxa de acerto para o problema de reconhecimento de dígitos (10 dígitos 0 - 9). A partir de um conjunto de treinamento/teste, ao final deverá ser encontrado qual a taxa de acertos usando o classificador k-NN com todas as 132 características disponíveis nos conjunto de treinamento e teste, e logo após, implementar um código para ser usado em AG's buscando maximizar a taxa de acerto.

4.1. Ideia de Resolução Exposta

Para atingir um resultado satisfatório, foi realizado uma lógica com recursos disponíveis em AG, como mutação, recombinação genética e seleção elitista e aleatória.

Inicialmente são setadas as seguintes informações: NUMBER_STOPOVER = 100; POPULATION_RATE = 0.01 e; MUTATION_RATE = 0.01. Essas informações dizem respeito, respectivamente ao critério de parada (número de gerações que serão executadas em sequência, sendo possível aumentar esse limite em tempo de execução), a taxa usada para calcular o número da população, neste caso, é calculado 1% da quantidade de instâncias no conjunto de treinamento e por fim, a taxa de mutação, ou seja, a porcentagem (1%) de características que serão alteradas a cada nova geração.

Como o conjunto possui 1000 elementos, a cada nova geração é formada uma população de 10 instâncias, na primeira vez que o algoritmo é rodado é gerado uma população totalmente aleatória.

Para este trabalho, uma instância da população possui um conjunto de características que serão ou não avaliadas para calcular a distância euclidiana no algoritmo k-NN.

Na mudança de geração a nova população é calculada pelos seguintes passos:

(i) seleção dos 2 melhores resultados na população anterior e aplicação da recombinação genética em um ponto de características aleatório;

(ii) seleção do melhor de todos da população anterior, assim até esse ponto temos 3 elementos, 2 recombinados por instâncias com bons resultados e 1 elemento considerado muito bom sem realizar nenhuma alteração no mesmo;

(iii) para as instâncias faltantes é gerado aleatoriamente um total de 7 novos elementos (totalizando o total de 10 de população);

(iv) e por fim é realizado 10 mutações completamente aleatórias em qualquer um dos elementos já gerados pelos passos anteriores, a mutação é realizada apenas em uma característica, invertendo seu estado.

Assim, consequentemente, é gerado sempre uma população com um valor muito bom (que faz com que na busca o melhor resultado encontrado não diminua muito), valores baseados em um histórico (o que não necessariamente vai gerar valores bons sempre, mas existe uma grande chance de acontecer) e novos valores (o que garante a diversidade na população e resultado).

5. Resultados Obtidos

Como no algoritmo k-NN é possível passar um valor k para utilização no algoritmo, foi realizado 2 execuções do programa, sendo um com k = 1 e outro com k = 2.

```
########## BEST IN HISTORY:
[1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1,
## Correct Percentage: 94.1%



@@ Generation 164:
[1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1,
@@ Time: 34.75sec
@@ Correct Percentage: 94.0%


[1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1,
@@ Time: 35.65sec
@@ Correct Percentage: 93.8%


[1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1,
```

Figura 1. Resultado obtido com k = 1

Sem a utilização do algoritmo genético, o próprio k-NN já retornava um resultado de em média 89.9%, para variados k's. Já com a utilização de Algoritmos Genéticos, mas com geração de populações totalmente genéticas, o resultado variou entre 87-91%.

Contudo, para k = 1, como visto na Figura 1 foi obtido um resultado de 94.1% com o passar de 164 gerações, contudo como visto nos resultados preliminares da geração seguinte, o maior resultado não chega tão perto do maior valor encontrado em toda a história. Isso acontece porque este resultado foi encontrado muitas gerações passadas e o valor caiu em alguns décimos com o passar das gerações e não conseguiu aumentar novamente.

```
########## BEST IN HISTORY:
[1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0,
 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0,
## Correct Percentage: 94.8%



@@ Generation 78:
[1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0,
 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0,
@@ Time: 45.92sec
@@ Correct Percentage: 94.0%


[1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0,
 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0,
@@ Time: 42.36sec
@@ Correct Percentage: 93.89999999999999%


[1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0,
 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0,
@@ Time: 46.24sec
@@ Correct Percentage: 94.8%


[1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0,
 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
@@ Time: 25.41sec
@@ Correct Percentage: 90.9%
```

Figura 2. Resultado obtido com k = 4

O mesmo não aconteceu com k = 4 que, como visto na Figura 2, que com o passar das gerações continuou aumentando seu valor. No fim, obteve uma porcentagem de 94.8% em 78 gerações.

6. Conclusão

Apesar de não utilizar formas extremamente complexas no algoritmo genético, foi possível, ainda sim, alcançar um ótimo resultado final, embora tenha demorado mais gerações do que o esperado para chegar lá. O esperado para o futuro é que se ainda iterando novas gerações, ele encontre um valor ainda maior que o anterior, embora não haja certeza que o conjunto de características seja genérico o suficiente para outros testes não encontrado no conjunto dado inicialmente.

Referências

Genetic algorithms (gas). `http://www.cs.cmu.edu/~02317/slides/lec_8.pdf`. Acessado: 2019-06-10.

(2019). *Algorithms for System Identification*, pages 154–155.

Zhang, S., Li, X., Zong, M., Zhu, X., and Cheng, D. (2017). Learning k for knn classification. *ACM Transactions on Intelligent Systems and Technology*, 8:1–19.