Algoritmos 2 2021/2

Documentação do Trabalho Prático 01 da Disciplina de Algoritmos 2

Matheus Prado Miranda

Departamento de Ciência da Computação, Universidade Federal de Minas Gerais

9 de janeiro de 2022

Palavras-Chave
kNN ; Árvore ; Classificação ; Métricas ; Teste ; Treinamento

1 Introdução

Esta documentação lida com o trabalho prático proposto sobre aplicações de algoritmos geométricos. Nele, é necessário implementar o algoritmo para classificação baseado nos k vizinhos mais próximos (kNN), utilizando a estrutura de árvores k-dimensionais para armazenar os conjuntos de pontos recebidos. Esse algoritmo tem como finalidade realizar previsões dos rótulos de novas instâncias e avaliar sua qualidade, através de conjuntos de treinamento e teste. Para atribuir esses rótulos, o algoritmo considera que as instâncias são pontos no espaço n-dimensional e verifica a proximidade entre elas de acordo com uma função de distância, que nesse trabalho é a função de distância euclidiana.

São fornecidos, para o cálculo desejado, diversas bases de dados que possuem informações sobre as classes e atributos de uma dada relação. Com isso, faz-se necessário o tratamento de dados e a implementação de algoritmos para ambos a árvore k-dimensional e o kNN, tendo em vista obter a classificação e suas seguintes estatísticas: acurácia ; precisão ; revocação.

2 Modelagem Computacional

Para a resolução, cria-se 3 classes para o tratamento da árvore k-dimensional e do kNN: "Node"; "kd_tree"; "x-NN", sendo que as duas últimas haviam sendo especificadas na proposta como obrigatórias. Todas essas possuem métodos e atributos que auxiliam no tratamento dos dados e execuções dos algoritmos.

É escolhida, como principal forma de armazenamento dos dados de suas respectivas bases, a estrutura lista da biblioteca padrão. Além dessa, faz-se necessário o uso de outra estrutura, especificada também na proposta, a fila de prioridades, que tem o objetivo de armazenar os k vizinhos mais próximos de cada ponto de teste e também as distâncias entre o respectivo ponto e os demais. Por último, dicionários são utilizados como forma de representação para os rótulos dos vizinhos encontrados, auxiliando no processo de verificação para aquele rótulo que deverá ser o escolhido para a previsão.

Por último, utiliza-se funções e módulos, para auxiliar nos métodos principais, da linguagem para a realização de funções específicas, tais como criação de matrizes de forma eficiente, seleção aleatória de amostras, cálculos com diferentes operadores e expressões e também pré-implementação de estruturas necessárias.

3 Especificações do Ambiente

- Windows, 10 Home, Sistema Operacional de 64 bits ;
- Linguagem Python ;
- Intel(R)Core(TM) I-7-8550U CPU @ 1.80GHz, 16,0 GB RAM .

4 Estruturas de Dados

Tratando-se da solução adotada, as classes, primeiramente, são uma alternativa eficiente escolhida para tratar diretamente das especificações das três “entidades” que são relevantes para o trabalho. Com elas, é possível ter um acesso às informações desejadas sem alto custo de ambos memória e tempo.

Já a lista é escolhida como forma de armazenamento dos dados da base e para auxiliar em diversaos métodos, pois ela possui operações eficientes, de bom custo benefício, e é de fácil manipulação e compreensão. Isso verifica-se, por exemplo, no fato de que várias listas são compostas de tuplas de elementos e são facilmente manipuláveis, com baixo custo, para acesso, inserção e remoção.

A fila de prioridades, por sua vez, é bastante flexível e possui pontos positivos em relação à alocação de memória, que pode ser feita de forma dinâmica. Além disso, já existe um módulo de sua pré-implementação na biblioteca padrão da linguagem.

Por último, dicionários são utilizados por, além de custos de criação, acesso e manipulação baixos, evitarem chaves com elementos repetidos e consequentemente poderem ter seus identificadores utilizados como contadores para verificar o número de vezes em que o rótulo se repete nos vizinhos mais próximos, classificando assim a respectiva instância.

5 Algoritmos

Nessa seção, explica-se os principais métodos e funções implementados para tratar as atividades propostas.

Primeiro, tem-se uma função para tratar os dados no arquivo e adicioná-los à listas de acordo com o necessário. Nela, outras funções da linguagem, como "split"e "startswith", foram usadas para lidar com possíveis diversas dimensões, classes e disposições dos dados em relação a caracteres especiais e espaços em branco. Logo depois, há outra função para criar os conjuntos de treinamento e teste. Essa, por sua vez, recebe uma lista, gerada na função anterior, com todas as instâncias de uma base. Com isso, ela divide esta lista aleatoriamente, com o auxílio da função "random.sample", em conjuntos na proporção de 70% e 30%.

Na segunda parte, a classe que trata dos nós é implementada. Para fins de auxílio no algoritmo de construção da árvore k-dimensional, cada nó possui atributos: itens ; filho da esquerda ; filho da direita ; mediana. Agora, já na classe "kd_tree", há o método importante de criação da árvore k-dimensional, de acordo com o modelo proposto em [1]. Esse método recebe inicialmente a lista com os pontos de treinamento e a profundidade 0 da raíz, e primeiro verifica qual o eixo atual. Logo depois, há uma condição para verificar se o nó atual é uma folha ou não. Caso ele seja uma folha, retorna-se um nó em que todos os seus atributos são None, a não ser o atributo itens que recebe o ponto atual. Caso ele não seja uma folha, calcula-se a mediana da lista e depois atribui-se valores das listas utilizando índices com a mediana. Depois, faz-se chamadas recursivas para as profundidades em sequência com esses novos valores, para os nós filhos da esquerda e da direita. Por último, atribui-se ao nó atual os valores de filho da esquerda, filho da direita, None e mediana, e então esse nó é retornado.

Na terceira parte, trata-se dos métodos mais importantes da classe "x-NN", que constrói a árvore com o conjunto de treinamento, realiza o algoritmo de k vizinhos mais próximos e calcula as métricas para a classificação.

Primeiro, tem-se o método de busca na árvore, proposto também pelo [1], mas adaptado com ideias do [4] para uma busca k-dimensional e não em apenas 2 dimensões. Ele recebe como parâmetros inicialmente o nó raíz, o ponto alvo do conjunto de treino, a profundidade, número de vizinhos e a lista de vizinhos vazia que representará o heap. Com isso, ele verifica se a árvore existe e se o nó atual é um nó intermediário ou folha. Caso seja um nó intermediário, atribui-se os nós filhos à duas novas variáveis e é feita uma chamada recursiva para o nó filho da direita. Além disso, calcula-se a distância euclidiana entre o ponto alvo e a atribuição da chamada recursiva, que é a lista de vizinhos, e também a distância para a outra seção do espaço. Com essas distâncias, é verificado se a distância euclidiana é maior do que essa seção ou se a lista de vizinhos ainda tem menos vizinhos do que o especificado, e se isso ocorre, é feita outra chamada recursiva para buscar nessa outra seção. Com o resultado dessa chamada, avalia-se com uma função auxiliar se os pontos encontrados são mais próximos do que os já existentes. Caso sejam, eles são colocados na fila de prioridades por ora. Ao fim disso, é retornado a fila de prioridades. Agora, caso seja uma folha, calcula-se a distância euclidiana para armazenar na fila de prioridades junto ao ponto, e retorna-se essa fila.

Em segundo nessa parte e por último , com os vizinhos mais próximos encontrados, é preciso classificar o conjunto de teste. A função de classificar recebe os vizinhos mais próximos e o número de vizinhos, e apenas vê qual o rótulo predominante. Depois de classificar, chega o momento de obter as estatísticas, segundo informações em [3]. Para elas, é feita uma função que recebe o conjunto de teste, as predições, as classes e os atributos. Com isso, outra função auxiliar encarrega-se de obter a matriz de confusão da classificação ao calcular os valores de tp, tn, fp e fn ("true positive", "true negative", "false positive"e "false negative"), verificando as classificações

da predição e aquelas do teste. Com esses valores, é possível calcular as métricas acurácia, precisão e revocação e retorná-las.

6 Experimentos e Resultados

Nessa seção, serão relatados os resultados para os experimentos realizados com 10 bases de dados. Essas bases possuem diferentes tipos de dados, não nominais, com diversas dimensões, mas no entanto apenas 2 classes. Para a verificação do método, escolhe-se bases que possuem um número de dados variando entre baixas e altas quantidades. A partir disso, define-se um número fixo 2 de vizinhos e é feito, como método para obter um resultado confiável das métricas, a média aritmética dos valores obtidos em 5 execuções.

A apresentação dos resultados encontrados estará logo abaixo na seguinte tabela.

Base de Dados	Número de Dados	Classes	Acurácia	Precisão	Revocação	Número de Vizinhos	Tempo de Execução
banana	5300	-1.0 ; 1.0	0,873	0,890	0,880	2	28,692 s
haberman	306	positive ; negative	0,649	0,592	0,327	2	4,480 s
twonorm	7400	0 ; 1	0,949	0,946	0,950	2	137,961 s
phoneme	5404	0 ; 1	0,898	0,919	0,939	2	35,630 s
bupa	345	1 ; 2	0,575	0,625	0,422	2	3,580 s
monk-2	432	0 ; 1	0,905	0,978	0,809	2	4,794 s
appendicitis	106	0 ; 1	0,802	0,891	0,857	2	2,172 s
heart	270	1 ; 2	0,596	0,653	0,601	2	4,893 s
australian	690	0 ; 1	0,521	0,618	0,425	2	4,547 s
ring	7400	0 ; 1	0,741	0,988	0,475	2	128,276 s

Figura 1: Tabela dos experimentos realizados

Com isso, é percebido que o método implementado com o modelo proposto possui um resultado satisfatório para as instâncias. Mesmo com um número muito baixo de vizinhos próximos estabelecido, as predições são, majoritariamente, corretas. Aquelas que não obtiveram valores ótimos nas métricas são explicadas pelo fato de possuírem poucas instâncias de dados, um resultado esperado devido aos conceitos do algoritmo, que favorece bases com riqueza de dados.

Conclusão

Este trabalho lida com a implementação de um método de classificação baseado nos k-vizinhos mais próximos utilizando a estrutura árvore k-dimensional. A abordagem utilizada é a utilização de classes, módulos e estruturas da linguagem Python, além dos principais algoritmos adaptados para a atividade. Com isso, é possível realizar uma interação direta entre as classes, que por sua vez, possibilita o uso dos métodos específicos para a criação de estruturas e funções que tratam da abordagem especificada para a implementação.

Por meio da resolução, pratica-se conceitos do tópico de algoritmos geométricos aplicados praticamente para a solução de problemas comuns e importantes na área de computação. Verifica-se, então, a importância de relacionar algoritmos e estruturas com a finalidade de aprimorar os resultados em ambas as questões de tempo e resultado.

Durante a implementação, houve importantes desafios a serem superados como por exemplo: leitura e armazenamento dos dados ; implementação de novas funções para tratamento desses dados ; compreensão e bom uso das estruturas da linguagem ; compreensão e implementação da árvore k-dimensional ; compreensão e implementação dos métodos de classificação ; implementação de métodos para obtenção das estatísticas da classificação.

Referências

[1] Computational Geometry: Algorithms and Aplications (2000), De Berge, Mark ; Van Kreveld, Marc ; Overmars, Mark ; Schwarzkopf, Otfried. Segunda Edição, Springer Verlag. url = http://www.cs.uu.nl/geobook/

[2] Slides da Disciplina de Algoritmos 2 (2022), Renato Vimieiro. Departamento de Ciência da Computação, Universidade Federal de Minas Gerais. Disponibilizado pela plataforma Microsoft Teams

[3] Confusion Matrix For Your Multi-Class Machine Learning Model (2020), Mohajon, Joydwip. Towards Data Science. url = https://towardsdatascience.com/confusion-matrix-for-your-multi-class-machine-learning-model-ff9aa3bf7826

[4] KD-Tree Nearest Neighbor Data Structure (2020). Stable Sort. US. url - https://www.youtube.com/watch?v=Glp7THUpGow