Trabalho Prático 1

Henrique Furst Scheid - 2017014898

Janeiro 2022

Implementação

O algoritmo

O algoritmo implementado desempenha a função de classificar pontos de acordo com sua proximidade espacial a pontos anteriormente avaliados. O mecanismo para tal avaliação consiste em um registro *a priori* de um conjunto de dados pertencente ao mesmo 'espaço' dos dados a serem avaliados. Desse modo, durante a classificação resgatam-se do registro um número arbitrário (oferecido pelo usuário do algoritmo) os dados espacialmente mais próximos do dado a se classificar. A classificação resultante corresponde à classificação presente na maioria de tais dados resgatados. Trata-se, pois, de uma implementação do algoritmo **k-nearest-neighbours**.

Denominam-se os dados oferecidos *a priori* de **conjunto de aprendizado** e os dados a serem classificados pelo algoritmo de **conjunto de teste**. O algoritmo está implementado em *python*.

Estruturas de dados

kdtree

O aspecto chave do algoritmo está no armazenamento e resgate de dados espaciais, com número arbitrário de dimensões. Foi utilizada uma árvore k-dimensional (*k-d tree*) com vértices estruturais e valores registrados exclusivamente em folhas. Dessa forma, o resgate de dados próximos a um ponto oferecido consistem em uma pesquisa (de complexidade logarítmica) a uma folha de valor mais próximo que atua como 'centro', seguido de 'retrocessos' a vértices em seu ramo para a adição de suas folhas vizinhas mais próximas (espacialmente, os pontos dentro dentro de um raio partindo de tal centro).

A implementação da *kdtree* consistiu na criação de duas classes principais: $KDtree$ e $BaseNode$. Esta se ramifica em duas subclasses $Node$ e $Leaf$, que representam os elementos da árvore.

KDtree possui métodos para:

- Construção de uma kdtree
- Pesquisa de pontos (retorna o ponto salvo mais próximo do ponto pesquisado)
- Resgate de todas as folhas que ramificam de um nó (retorna um conjunto de pontos)
- Exibição da kdtree, utilizado para testes

BaseNode possui métodos para:

- Construção de um vértice com valor interno e referência para um vértice pai.
- Resgate do vértice pai

Node possui métodos para:

- Construção de um vértice BaseNode com valor comparador e referência ao vértice paterno, adicionado de:
  - dois BaseNodes filhos (esquerdo para valores menores que o comparador, direito para valores maiores que o comparador)
  - dimensão do espaço à qual o comparador pertence
- Configuração e resgate do vértice filho esquerdo
- Configuração e resgate do vértice filho direito

Leaf possui métodos para:

- Construção de um vértice BaseNode com valor de classificação e referência ao vértice paterno, adicionado das coordenadas espaciais do ponto referente

k-nearest-neighbours

Além de armazenar os pontos do conjunto de aprendizado em uma *kdtree*, a execução do algoritmo exige computação sucessiva de pontos vizinhos do ponto de teste P. Tais vizinhos são armazenados em uma **fila de prioridade** (implementada no módulo *heapq* da biblioteca padrão de *python*). A fila de prioridade permite a ordenação de novos vizinhos mediante menor distância com relação a P a cada inserção, de forma que pontos mais distantes ocupam as folhas. Tal fila é sempre "podadada" para manter apenas os K vizinhos mais próximos de P.

Algoritmos

Construção de kdtree

A construção de uma árvore k-dimensional é feita através da construção recursiva de nós. Uma lista inicial com todos os pontos é separada "maiores" e "menores" através da comparação com o valor mediano de uma das dimensões. Um novo nó é criado cujos filhos esquerdos são o ramo "menores" e os direitos o ramo "maiores". Tal processo é feito recursivamente para novas dimensões até que haja no máximo 2 pontos, instante no qual folhas são criadas para armazená-los e o algoritmo retorna.

```
constroiArvore(lista_pontos, dimensão, nó_pai):
    Se 'lista_pontos' está vazio:
        retorne
    Se 'lista_pontos' possui 2 pontos:
        crie um novo nó filho de 'nó_pai'
        o valor do novo nó é a mediana dos pontos na dimensão atual
        crie duas folhas para armazenar os pontos:
            direita para o ponto maior que a mediana na dimensão atual
            esquerda para o outro
        insira as folhas como filhas do nó
        retorne o novo nó

    Se 'lista_pontos' possui 1 ponto:
        retorne uma folha com o ponto

    Senão:
        crie um novo nó filho de 'nó_pai'
        o valor do novo nó é a mediana dos pontos da dimensão atual
        divida 'lista_pontos' em pontos maiores e menores que a mediana
            na dimensão atual

        filho direito do novo nó = constoiArvore(pontos maiores,
                                                 próxima dimensão,
                                                 nó atual)
        filho esquerdo do novo nó = constroiArvore(pontos menores,
                                                 próxima dimensão,
                                                 nó atual)
        retorne o novo nó
```

Consulta de ponto mais próximo

Para resgatar dados a partir de um ponto oferecido, basta comparar recursivamente o valor do input com o valor do comparador no nó em sua dimensão correspondente, a começar da raiz da árvore. Se o valor do input for maior, pesquise recursivamente no filho da direita. Caso contrário, pesquise recursivamente no filho da esquerda. Chegando-se em uma folha, retorne-a: este é o valor desejado.

Resgatar as folhas a partir de um nó

Dado um nó da kdtree, crie uma lista vazia. Adicione a essa lista as folhas dos nós do seu filho à esquerda, depois as folhas dos seus filhos da direita. Para encontrar tais filhos, execute este mesmo algoritmo, recursivamente.

```
folhas(nó):
    Se 'nó' é vazio:
        retorne lista vazia
    Se 'nó' é uma folha:
        retorne uma lista contendo 'nó'
    Senão:
        crie uma lista vazia
        adicione à lista o valor de folhas(filho da esquerda de 'nó')
        adicione à lisa o valor de folhas(filho da direita de 'nó')
        retorne a lista
```

K-nearest neighbours

1. Dado um ponto P do conjunto de teste para classificação e um número K de vizinhos como parâmetro, o algoritmo pesquisa em sua kdtree (habitada pelo conjunto de aprendizado) o ponto X mais próximo de P.
2. Em seguida, o algoritmo retrocede para o pai X' de tal ponto. Se as folhas que ramificam de X totalizam um número de vizinhos menor que K, retroceda para o pai de X', X'' e pegue as folhas que retrocedem de X''. Repita o processo até encontrar K vizinhos.
3. Se o valor comparador do próximo pai a se retroceder estiver a uma distância maior do que a distância máxima entre os vizinhos atuais e o ponto P, interrompa laço. Senão, retroceda novamente, remova os vizinhos mais distantes para permanecer com K vizinhos sendo considerados.

Análise experimental

Foram escolhidos os conjuntos de dados