Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

DCC207 – Algoritmos 2
Prof. Renato Vimieiro

Trabalho Prático 01 – Aplicações de algoritmos geométricos

Objetivos

O objetivo central deste trabalho prático é abordar aspectos práticos de implementação dos algoritmos geométricos vistos nas aulas teóricas. Em particular, abordaremos aspectos práticos das árvores k-dimensionais para solução de problemas em outra área da computação. Dessa forma, veremos, além dos detalhes de implementação de tal estrutura, sua aplicação em outros contextos que não exatamente em geometria computacional.

Tarefas

Os alunos deverão implementar o algoritmo para classificação baseado nos k vizinhos mais próximos (chamado de kNN). O problema de classificação é um problema de Aprendizado de Máquina Supervisionado em que uma coleção de instâncias rotuladas, chamado de conjunto de treinamento, é usada para ajustar um modelo, o qual é usado para predizer os rótulos de novas instâncias. A qualidade do modelo é avaliada através de um outro conjunto de instância rotuladas, chamado de conjunto de teste. Nesse caso, o modelo é usado para predizer o rótulo das instâncias do conjunto de teste e uma série de métricas são computadas para avaliar a qualidade do aprendizado.

O kNN, como o próprio nome indica, atribui um rótulo a uma nova instância com base nos seus k vizinhos mais próximos no conjunto de treinamento. Dessa forma, o algoritmo considera que as instâncias do processo de aprendizado são pontos no espaço n-dimensional e que a proximidade entre instâncias é dada por uma função de distância (por exemplo, a distância euclidiana).

O tempo de processamento do kNN é, então, dominado pelo custo de se localizar os k pontos mais próximos da instância que se deseja classificar. Como vimos nas aulas teóricas, as árvores kd podem ser usadas para buscar pontos com um custo baixo. Sendo assim, a principal tarefa do trabalho prático é implementar o algoritmo de classificação kNN baseado na árvore kd.

Abaixo estão as principais atividades:

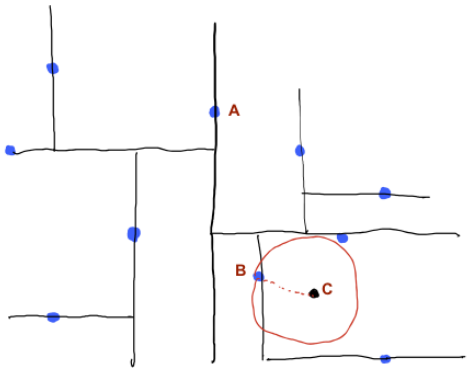
1. Implementar a classe árvore kd que receba um conjunto de pontos no espaço k-dimensional e construa a estrutura adequadamente. Os pontos devem ser

armazenados obrigatoriamente nas folhas. Não serão aceitas implementações que armazenem pontos em nós intermediários. O rótulo das instâncias (pontos) não deve ser considerados na construção da árvore.

2. Implementar a classe x-NN que recebe um conjunto de pontos treinamento e outro de teste, constrói a árvore kd com o conjunto de treinamento, e, depois, para cada ponto A no conjunto de teste, recupera os x pontos mais próximos de A e determina a classe de A pelo voto majoritário (em caso de empate escolhe arbitrariamente).
3. A classe deve fornecer as estatísticas da classificação (acurácia/precisão/revocação – links com as definições nas referências) e as classes dos pontos no conjunto de teste.

Alguns detalhes da implementação

- Os x vizinhos (pontos mais próximos) de cada ponto de teste devem ser armazenados em uma fila de prioridades limitada. Essa fila de prioridades deve ser mantida com a distância entre o ponto em questão e os demais. Sempre que um novo ponto for adicionado à fila, verifica-se se o limite de pontos foi ultrapassado, e o de menor prioridade é removido (note que eventualmente essa operação resulta em uma fila idêntica à anterior).
- Após o preenchimento da fila com os x primeiros candidatos a vizinhos mais próximos, alguns ramos da árvore podem ser descartados para economia de tempo. Para exemplificar, considere a situação da figura abaixo. Os pontos azuis são as instâncias do conjunto de treinamento, o ponto a classificar é o ponto em preto, as linhas sólidas são as divisões da árvore kd, e será utilizado somente 1 vizinho mais próximo para classificar.



Suponha que a fila de prioridades já esteja preenchida com o ponto B. Se estivermos analisando a árvore cuja raiz seja a linha divisória central (ponto A), os pontos da sub-árvore da esquerda não precisam ser inspecionados, pois, certamente, não são mais próximos do ponto C que o ponto B. Isso pode ser constatado pelo fato de que o raio do círculo que passa sobre o ponto B é menor que a distância dele até a linha divisória (a diferença absoluta da coordenada x do ponto A e B é maior que o raio do círculo). Caso essa distância

fosse menor ou igual ao raio, isso implicaria na possibilidade de existir outros pontos com distância menor com respeito a C localizados na sub-árvore da esquerda. Assim, ela também deveria ser avaliada. Um raciocínio similar pode ser generalizado para o caso de mais de um vizinho.

- Nesse trabalho, usaremos a distância euclidiana entre os pontos para determinar a vizinhança.
- As implementações podem ser em C/C++ ou em Python (preferencialmente). O uso de bibliotecas que não a STL, NumPy, ou ainda aquelas que não são padrão da linguagem deve ser discutido com o professor.

O que entregar

Deverá ser entregue um relatório descrevendo a implementação, onde serão detalhadas as escolhas feitas para implementar os algoritmos; isto é, as estruturas de dados usadas, a forma de implementar e os testes realizados. Também deverá ser feita uma explicação do método com as palavras do aluno. O relatório deve conter ainda uma análise dos experimentos conduzidos para avaliar o desempenho/acurácia do método implementado. Para isso, devem ser usadas as bases de teste disponíveis no site da ferramenta Keel (ver referências). Devem ser usadas, no mínimo, 10 bases de dados. A divisão em treino e teste deve ser feita pelo aluno de forma aleatória na proporção 70-30 respectivamente.

Deverão ser observados os critérios de boas práticas de programação, incluindo documentação, identificadores de variáveis etc.

A implementação deverá ser disponibilizada em um repositório no GitHub com o código completo. O repositório deverá ser mantido privado e só tornado público após a data de entrega. Note que repositórios públicos possibilitam a outros plagiarem o trabalho mesmo sem o conhecimento do autor.

Política de Plágio

Os alunos podem, e devem, discutir soluções sempre que necessário. Dito isso, há uma diferença bem grande entre implementação de soluções similares e cópia integral de ideias. Trabalhos copiados na íntegra ou em partes de outros alunos e/ou da internet serão prontamente anulados. Caso haja dois trabalhos copiados por alunos diferentes, ambos serão anulados.

Datas

Entrega Teams: 03/01/2022 às 23h59

Política de atraso

Haverá tolerância de 1 dia na entrega dos trabalhos. Submissões feitas depois do intervalo de tolerância serão penalizadas.
- Atraso de 1 dia: 15%
- Atraso de 2 dias: 30%
- Atraso de +3 dias: não aceito

Serão considerados atrasos de 1 dia aqueles feitos após as 0h30 do dia de entrega. A partir daí serão contados o número de dias passados da data de entrega.

Referências

- Cap. 5; Computational Geometry Algorithms and Applications. Berg et al.
- Cap. 21; Algorithm Design and Applications. Goodrich e Tamassia.
- https://dataminingbook.info (seção 18.3)
- https://scikit-learn.org/stable/modules/neighbors.html
- https://pt.wikipedia.org/wiki/Precisão_e_revocação
- https://sci2s.ugr.es/keel/category.php?cat=clas