Aprendizado Não Supervisionado

Prof. Thomas da Silva Paula

Trabalho I - Implementação K-means

Descrição

Como vimos em aula, agrupamento de dados é uma das técnicas amplamente utilizadas em aprendizado não supervisionado. Técnicas de agrupamento de dados visam dividir os dados em grupos (*clusters*) que são significativos, úteis ou ambos. Grupos são considerados significativos quando tentam capturar a estrutura natural dos dados e são considerados úteis quando são ponto inicial para outras tarefas (e.g. resumo de dados).

O K-means é um dos algoritmos mais influentes na área de mineração de dados, principalmente por ser simples, intuitivo e ter bons resultados. Este trabalho busca exercitar justamente nosso entendimento sobre esse algoritmo clássico, onde vamos implementar uma versão simples do K-means do zero. Por simples, deve-se entender que nenhuma das otimizações do K-means precisa ser implementada (e.g. estruturas de dados mais eficientes ou cálculos otimizados).

Ponto importante: o algoritmo K-means pode ser implementando utilizando bibliotecas Python como Numpy e Pandas, porém você não deve utilizar a função pronta do K-means do Scikit Learn! O objetivo do trabalho é justamente fazer a sua implementação, mostrando o seu entendimento do algoritmo.

Leia com atenção todos os itens e pergunte ao professor caso tenha qualquer dúvida.

Requisitos

1. Criar método com nome *do_kmeans_clustering*, que receba um parâmetro k, um parâmetro X e um parâmetro *random_state*, representando número de clusters, dados e inteiro a ser usado como seed (para função random), respectivamente.
   (a) O método deve possuir dois retornos, sendo que o primeiro são os centróides e segundo são os grupos finais.
   (b) Assuma que X terá shape $(num\_instances, num\_features)$, onde $n\_features = 2$.
   (c) Assuma que os centróides finais tenham shape $(k, num\_features)$.
   (d) Assuma que os clusters (labels) sejam uma lista de tamanho *num_instances*.
2. A inicialização dos centróides dos clusters deve ser aleatória e baseada no *random_state*.
3. O algoritmo deve executar até que os clusters não mudem. Portanto, não temos um número máximo de iterações.
4. Devem ser implementados ao menos um método de plot para verificar os dados iniciais e um método de plot para verificar os grupos finais em relação aos dados.
   (a) No plot dos grupos, deve-se mostrar também os centróides de cada grupo de uma forma destacada (cor diferente ou outro símbolo, por exemplo).
5. Para o treinamento, deve-se utilizar os valores $k = 3$ e $random\_state = 30$.
6. A base de dados usada é sintética e para tal deve ser utilizado o método *make_blobs* do *sample_generator* da biblioteca *Scikit Learn*. Neste caso, deve-se utilizar o $random\_state = 100$.

Perguntas

Após realizar sua implementação base, faça testes de acordo com as perguntas abaixo e escreva suas percepções.

1. Realize experimentos com diferentes valores para o *random_state* na inicialização de centróides. O que você pode observar?
2. Realize experimentos com diferentes valores de k (número de clusters). O que você pode observar? O que acontece se você variar o valor de k e o *random_state* na inicialização de centróides, ao mesmo tempo?
3. O que você aprendeu ao realizar essa implementação?

Entrega Final

- Jupyter notebook com os experimentos realizados (.ipynb e um .html ou .pdf).
- Relatório em formato .pdf com suas respostas para as perguntas e suas conclusões em geral, seja em relação à implementação ou aos experimentos e desafios que você enfrentou. Caso prefira, os resultados e conclusões podem ser colocados diretamente no Jupyter notebook, desde que fique bem explicado e claro.