Trabalho Prático de Inteligência Artificial

Bruno Asti Baradel	726499	Pablo Ferreira Laranjo	726557
Lucas Granja Toniello	726560	Vanderlei Jesus de Andrade	726590

Introdução

Este trabalho consistiu em implementar três algoritmos de agrupamento: `k-médias`, `single-link` e `average-link`. A partir disso, eles foram utilizados para agrupar três conjuntos de objetos. A saída de cada algoritmo foi então transformada em gráficos e, em seguida, estes gráficos foram comparados com os gráficos reais. Uma análise então pôde ser feita com base na comparação dos resultados entre os algoritmos, assim como entre cada algoritmo e o resultado real. Então, foi utilizado o Índice `Rand` Ajustado, implementado por nós, para calcular a similaridade entre *clusters*. Uma análise sobre estes resultados foi feita, buscando encontrar a melhor partição para cada algoritmo.

Algoritmos

O primeiro passo para iniciar o trabalho foi implementar os três algoritmos necessários. O algoritmo `k-médias` foi implementado em *Python*. Seu funcionamento é baseado na classificação de um determinado número pré-definido *K* de *clusters* e de iterações, e tem como função de classificação a distância entre o objeto e o centróide. A cada iteração, o centróide dos *clusters* é atualizado e os cálculos são refeitos.

Em seguida, os outros dois algoritmos, `single-link` e `average-link`, também foram implementados, mas desta vez em *C++*. O `single-link` forma a matriz de distâncias calculando todas as distâncias entre todos o objetos em busca da menor delas; os dois objetos mais próximos formam um *cluster*; e a matriz de distâncias é atualizada com base na distância entre esse *cluster* e os outros objetos. Em cada iteração será formado um *cluster* entre os dois objetos ou *clusters* com menor distâncias entre eles. Já o `average-link` funciona de forma semelhante ao `single-link`, exceto na forma em que calcula as distâncias, onde a distância entre dois *clusters* se dá pela média entre os objetos dos *clusters*.

Os três algoritmos recebem como entrada os arquivos `c2ds1-2sp.txt`, `c2ds3-2g.txt` e `monkey.txt`, cada um representando um conjunto de objetos. Como saída, eles produzem um arquivo com as partições com os números de *clusters* fornecidos separando eles por quebras de linha. Para formatar a saída no formato `.clu` , foi desenvolvido um algoritmo secundário chamado `formatador.py` que usa como base a saída dos algoritmos.

Rand

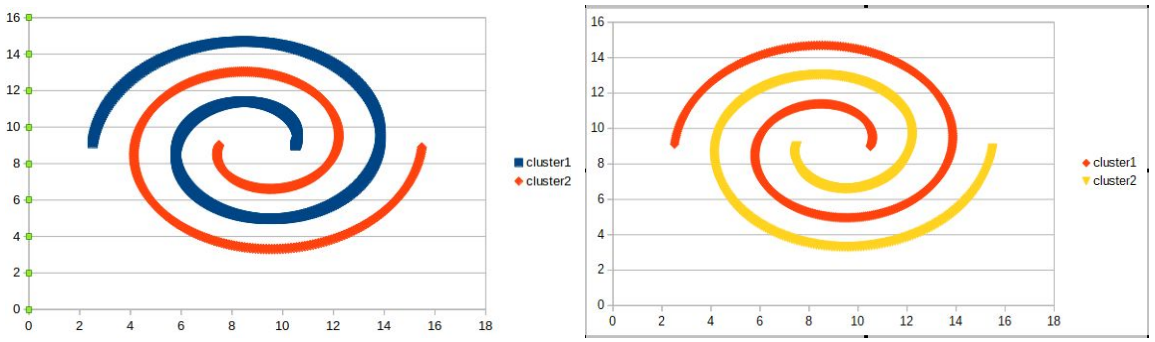
O Índice `Rand` Ajustado foi implementado por nós para este trabalho. Sua implementação foi feita em *Python*. Ele funciona gerando uma matriz de confusão a partir dos valores reais e dos valores previstos e, em seguida, calcula o índice `Rand`, que gera a similaridade entre as partições reais e previstas que foram produzidas.

Comparando Resultados

A partir da criação dos gráficos, da geração da matriz de confusão e o cálculo de similaridade do algoritmo `Rand`, é possível fazer uma comparação de cada algoritmo para identificar quais são mais semelhantes com os gráficos reais.

Abaixo, encontram-se as comparações entre a partição real e a partição mais próxima encontrada por algum dos algoritmos:

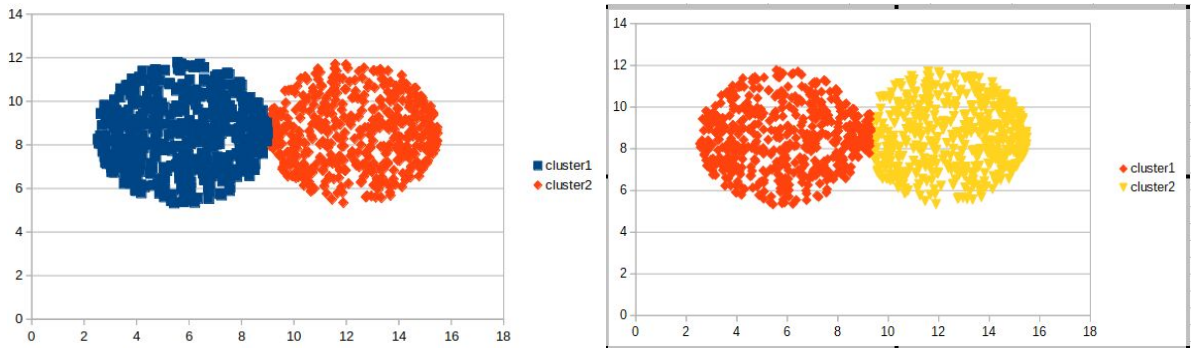
Para o gráfico em espiral, o melhor algoritmo foi o `Single-Link`.



`Real` `Single-Link`

Através do cálculo do índice `Rand`, executado a partir do algoritmo do `single-link`, foi obtido índice `Rand` de 1,0 de similaridade. Sendo assim é possível observar que o gráfico que foi gerado pelo algoritmo `single-link` é mais semelhante à partição real, quando comparado com os outros dois algoritmos: `k-médias`, apresentando índice `Rand` de aproximadamente 0,034, pelo fato de possuir um alto rendimento ao separar conjuntos de dados englobados em clusters; e `average-link`, com índice `Rand` de aproximadamente 0,4728. O fato do `single-link` ser mais similar a essa partição se dá pois os *clusters* são encadeados, ou seja, qualquer ponto em comum em um *cluster* está mais próximo ao mesmo *cluster* do que qualquer outro ponto que não pertence a esse *cluster*. Como temos uma partição com dois *clusters* no gráfico `Single-Link`, é possível observar que um ponto que pertence ao `cluster1` não está próximo de um ponto do `cluster2`.

Para o gráfico com dois círculos, o algoritmo que obteve melhor desempenho foi o `K-médias`.



`Real` `K-médias`

Neste caso o algoritmo obteve um índice `Rand` de aproximadamente 0,8872, o que indica alta similaridade. Isso se deve ao fato de que o k-médias possui alto rendimento ao

separar conjuntos de dados englobados em *clusters*, como no caso acima, onde não há nenhuma separação entre os dados. Os outros algoritmos obtém resultados muito ruins neste exemplo, com o `single-link` em particular apresentando índice rand de 0.0003. Devido a aglomeração dos dados, o `single-link` não consegue distinguir quais dados pertencem a cada *cluster*, pois faz seus pareamentos essencialmente de maneira aleatória.

Por fim, para o gráfico do macaco, o algoritmo mais efetivo foi, novamente, o `single-link`..

`Real` `Single-Link`

Neste caso, o algoritmo obteve um índice `Rand` de aproximadamente 0,8344. Como foi discutido no gráfico de espiral, pelo fato do algoritmo conseguir agrupar conjuntos de dados mais próximos do mesmo *cluster*, é possível observar no gráfico `single-link` que os *clusters* 2 ao 8 estão mais próximos de si mesmo; apenas o `cluster1` não consegue distinguir quais dados pertencem a cada *cluster*. Os outros algoritmos tiveram índice `Rand` semelhantes entre si: o `k-médias` obteve 0,5945, e o `average-link` 0,4967.

Conclusão

A partir dos gráficos obtidos pelos algoritmos, é importante observar que cada algoritmo tem sua similaridade de acordo com o agrupamento dos *cluster*. Para os casos aqui apresentados, o algoritmo `single-link` apresentou um comportamento mais próximo ao esperado em duas ocasiões. Por outro lado, o algoritmo `average-link` não conseguiu apresentar um resultado melhor que os outros em nenhuma partição. Entretanto, podemos ver no gráfico de círculos que o `single-link`, até então supostamente o melhor algoritmo para a tarefa, obteve o pior índice Rand de todos: 0,0003. Com isso, fica claro que os três algoritmos são úteis para situações mais específicas: o `single-link` é útil para partições com clusters separados; já o `k-médias`, para partições com clusters aglomerados; por fim, o `average-link`, apesar de não se mostrar o melhor algoritmo para nenhuma solução, obteve resultados medianos sem muitos valores fora da média. A disposição dos objetos influenciam consideravelmente no resultado esperado dos algoritmos, tornando-os complementares e de igual importância para a classificação do dados.