Lista de exercícios Matemática Computacional
Parte B – Prof. Dr. Reinaldo Rosa - 2020

Denis M. A. Eiras

Exercício 1 - Descrição

1-Simulação de Sinais Estocásticos com GRNG1.py com N valores de medidas.

1.1. Utilize o algoritmo e gere 10 sinais para cada família com N elementos:
N1: 64; N2:128; N3:256; N4:512; N5:1024; N6: 2048; N7:4096; N8: 8192

1.2. Escreva um algoritmo em Python que permita, tendo como entrada cada um dos sinais acima, obter sua forma normalizada entre 0 e 1, obter o respectivo Histograma e calcular os 4 momentos estatísticos respectivos.

1.3. Organize todos os dados num dataset (instancias x atributos) e tente agrupá-los com a técnica K-means para caracterizar, se houver, classes nos espaço de parâmetros composto por variância, skewness e kurtosis.

Exercício 1 – Detalhes da implementação

Para a resolução do Exercício 1, foi implementado o programa principal exercicio1.py para executar o exercício por completo, que utiliza as funções dos programas exercicio1_1.py, exercicio1_2.py e exercicio1_3.py, que contém as soluções dos itens 1.1, 1.2 e 1.3.

As funcões dos exercícios 1.1 a 1.3 foram encapsuladas em arquivos python separados e implementadas de forma a serem re-utilizadas por outros programas. Para isso, foram implementados alguns parâmetros extras, que indicam se os valores devem ser normalizados, os nomes dos arquivos gerados, arrays com as famílias, número de sinais, valores de k para o Kmeans e métodos dos cotovelo, além de parâmetros que controlam a exibição das figuras geradas na tela.

Todas as funções são explicadas nos respectivos arquivos, incluindo a descrição do objetivo, métodos utilizados e os parâmetros de entrada e saída.

Exercício 1.1

O arquivo exercicio1_1.py contém a função “gerador_de_sinais_aleatorios”, que gera uma quantidade parametrizada sinais por família, e armazenas os resultados no arquivo .csv de nome também parametrizado, contendo as colunas valor, que é gerado do gerador randômico não gaussiano, família, representando a quantidade de valores da família, e a coluna sinal, um número que identifica unicamente cada sinal.

Exercício 1.2

No arquivo exercicio1_2.py, a função “gerador_de_momentos”, utiliza um arquivo csv como entrada, no mesmo formato do arquivo de saída do exercício1_1.py. Os valores podem ser normalizados entre 0 e 1, caso o parâmetro is_normalizar_valores seja True. Em seguida, calcula o histograma, variância, assimetria e curtose, para cada conjunto de valores de cada par sinal / família e salva a figura dos histogramas. Um arquivo de saída .csv com as colunas variância, assimetria e curtose é gerado, para ser reutilizado no exercício 1.3.

Exercício 1.3

O arquivo exercicio1_3.py implementa a função “k_means_e_metodo_do_cotovelo”, que recebe o arquivo csv com os momentos estatísticos, gerados na função do exercicio1_2.py, e executa o algoritmo K-means para tentar agrupar classes nos espaços de parâmetros variância, assimetria e curtose, exibindo exibir os agrupamentos graficamente. Quatro técnicas são podem ser utilizadas para se para descobrir o melhor k:

- Técnica do método do cotovelo – “distorcao_km_inertia” - implementação própria;
- Técnica do método do cotovelo – “distorcao_yellowbrick” - pacote yellowbrick;
- Técnica da silhueta - “silhueta_yellowbrick” -
- Técnica Calinski Harabasz - “calinski_harabasz_yellowbrick” - pacote yellowbrick;

O método do cotovelo implementado executa o agrupamento K-means no conjunto de dados para um intervalo de valores para k (digamos de 1 a 10) e, em seguida, para cada valor de k, utiliza a soma dos quadrados das distâncias geradas pelo próprio k-means do scikit-learn para gerar um gráfico, onde a determinação do cotovelo é visual. Se o gráfico de linhas parecer um braço, então o "cotovelo" (o ponto de inflexão na curva) é o melhor valor de k.

O pacote yellobrick implementa, por padrão, seu próprio método do cotovelo, usando uma pontuação denominada “distortion”, calculada a partir da soma das distâncias quadradas de cada ponto ao seu centróide. Outras métricas também podem ser usadas, como a pontuação “silhouette”, que utiliza o coeficiente médio da silhueta para todas as amostras ou a pontuação “calinski_harabasz”, que calcula a taxa de dispersão entre e dentro de clusters. A determinação do melhor k é apresentada graficamente quando é possível determiná-lo pelos algoritmos.

Análise

Algumas execuções do exercicio1.py exibiram histogramas com dados bastante aleatórios, independentemente do número de valores da família, confirmando a implementação do gerador randômico não-gaussiano sem classe de universalidade via PDF, como exibem as figuras 1.a e 1.b.

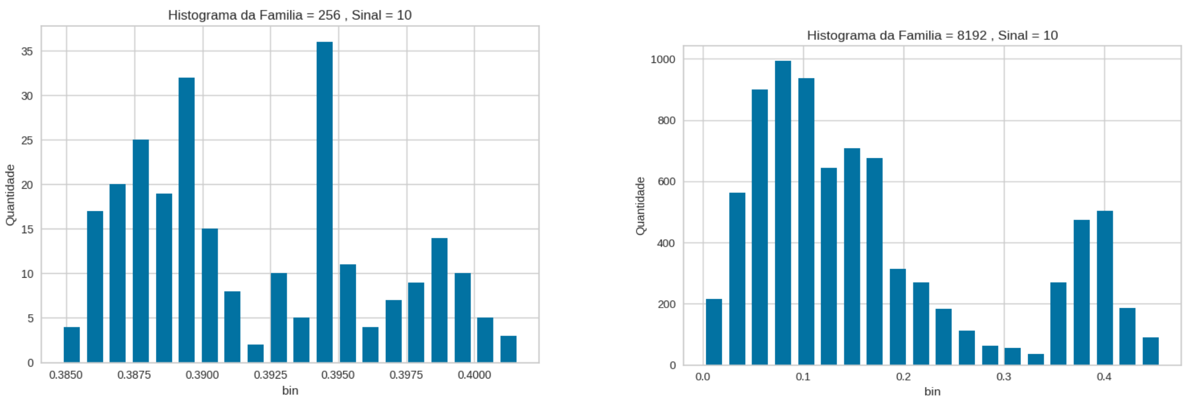

Figura 1. a) Um histograma da família 256; b) Um histograma da família 8192

Para verificar se há classes nos espaço de parâmetros composto por variância, assimetria e curtose, o algoritmo K-means foi executado com k entre 2 e 7.

A primeira estratégia para identificar o melhor k, foi analisar o gráficos gerados para cada k, que exibem as áreas dos momentos estatisticos na diagonal principal e a distribuição dos pontos para cada par de momento estatístico.

Nas figuras 2.a e 2.b, observando a assimetria, no gráfico central, e a curtose, no gráfico inferior direito, percebe-se que há uma menor intersecção das áreas entre os agrupamentos nos gráficos de k=2 e k=3, indicando que esses parâmetros k podem ser os melhores valores a serem considerados como agrupadores desses momentos estatísticos. Isto se confirma ao observar os agrupamentos gerados nos gráficos que compreendem assimetria e curtose. O mesmo não pode ser dito da variância, onde a intersecção dos agrupamentos é grande.

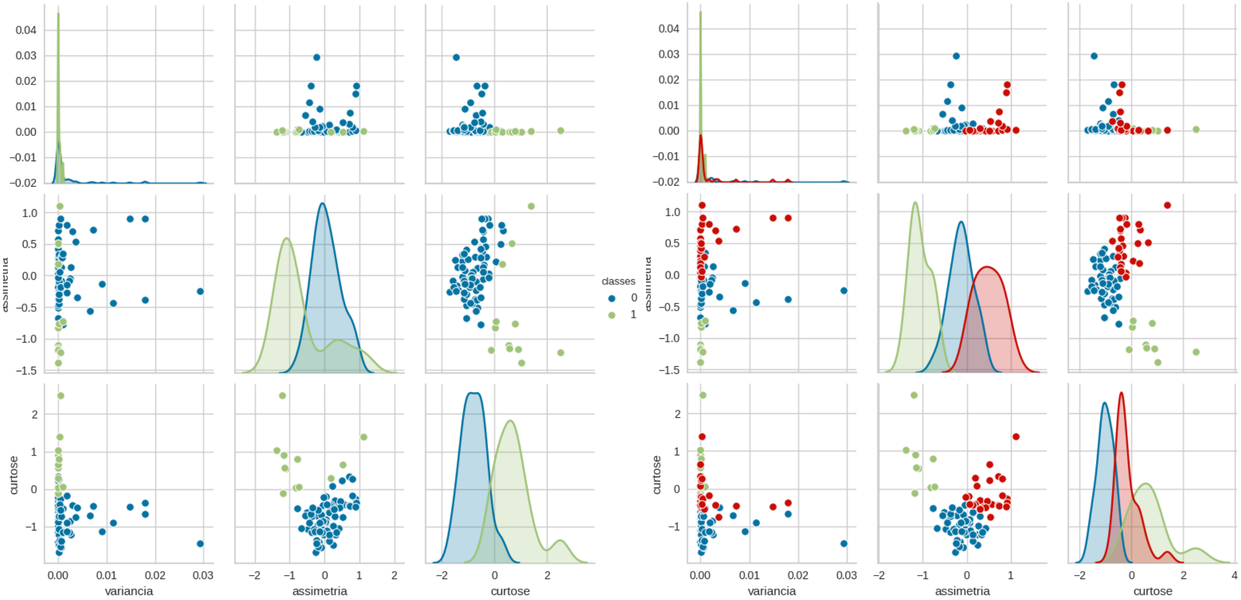

Figura 2. a) K-means para k=2; b) K-means para k=3.

Utilizando o método do cotovelo implementado (fig. 3.a), método do cotovelo do pacote yellowbrick (fig. 3.b), Calinski Harabasz, (fig. 4.a) e método da Silhueta (fig. 4.b), verifica-se que o melhor k é 3. O método da silhueta têm uma melhor média e uma quantidade menor de valores abaixo de 0, para k=3.

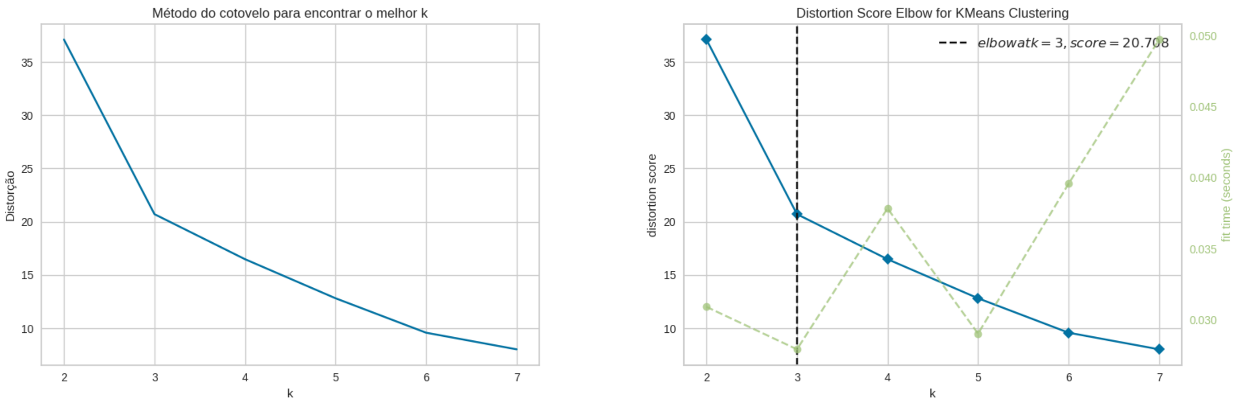

Figura 3. a) Método do cotovelo implementado; b) Método do cotovelo do pacote yellowbrick

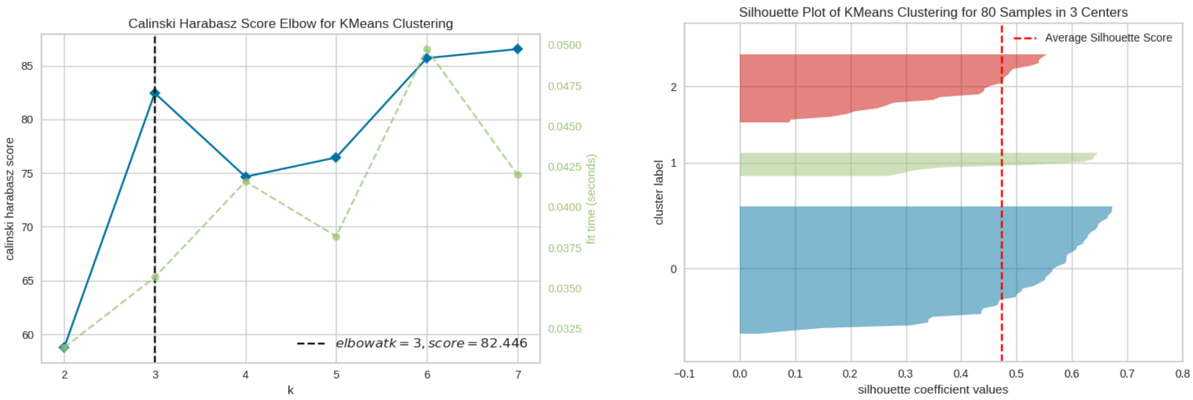

Figura 4. a) método Calinski Harabasz e b) método da Silhueta do pacote yellowbrick

Em uma possível futura implementação, uma das formas a se determinar o melhor k poderia ser através da utilização as áreas que compreendem os momentos estatísticos, exibidos na diagonal principal, através do seguinte algoritmo:

```
menor_area = Infinito
melhor_k = Nenhum

Para cada k:

    area_total_intersecao[k] = 0

    Para momento em [‘variância’, ‘assimetria’, ‘curtose’]:

        area_intersecao_momento = <area de interseccao do momento entre as classes>
        area_total_intersecao[k] = area_total_intersecao[k] + area_itersecao_momento

    se area_total_intersecao[k] < menor_area:
        menor_area = area_total_intersecao[k]
        melhor_k = k

retorna melhor_k
```

Exercício 1 – Conclusão

As análises realizadas indicam que a assimetria e a curtose são os melhores momentos estatísticos para serem utilizados como classes agrupadoras entre variância, assimetria e curtose, considerando uma série de sinais estocásticos aleatórias.

Existem diversos métodos para se interpretar os melhores agrupamentos do método K-means, mas nem sempre um método apresenta uma solução que serve para todos os casos, como é o caso do método do cotovelo, que pode conter os mesmos ângulos de inflexão para cada k. O método da silhueta pode ser uma boa opção para uma análise mais detalhada da quantidade de pontos fora da curva