Relatório de Iniciação Científica
Técnicas Espectrais para Agrupamento Múltiplo

Mateus Matias dos Santos, aluno
Eduardo Bezerra da Silva, orientador

Agosto de 2015

Resumo

Este projeto de pesquisa visa o aprofundamento em técnicas espectrais de redução de dimensionalidade, especificamente a baseada em autovetores e autovalores[1], com o objetivo de facilitar o agrupamento de conjuntos de dados (*datasets*) que possuem uma distribuição reminiscente da distribuição normal ou Gaussiana, ou tantas dimensões que cause inconsistências no resultado de algoritmos de agrupamento. Essa técnica, no contexto desse projeto, é usada para tratamento de dados astronômicos correspondentes à estrelas, possibilitando o agrupamento de sistemas estelares que possuem a mesma origem (um *open cluster*). A linguagem de programação utilizada é o Python versão 3.3.
Palavras-chave: grafo, distribuição normal, eigenvectors, clustering, estrelas, *open cluster*

[1] *eigenvectors e eigenvalues.*

Sumário

1 Introdução

A redução de dimensionalidade é de vital importância ao lidar com datasets dispostos de forma normal e com muitas dimensões, pois facilita a comparação e, consequentemente, a aplicação de algoritmos de agrupamento, já que se estes baseiam principalmente em estabelecer uma relação entre os dados. Uma das áreas em que é necessária a redução de dimensionalidade de datasets é na Astronomia. Entre os diversos problemas que requerem o auxílio da computação, destaco o *Stellar Cluster Membership Assignment*, definido como "o problema de segregar o campo e estrelas que pertencem a um mesmo aglomerado em um dado campo de catálogos que são gerados a partir de imagens de um telescópio"[1]. Os dados de entrada são as coordenadas x e y de cada estrela; o z, porém, não é conhecido, o que impossibilita o uso de algoritmos de agrupamento tradicionais. Por isso, é necessária a redução de dimensionalidade do dataset, para que se possa aplicar tais técnicas. O método de redução de dimensionalidade usado é aquele segundo Belkin e Niyogi[2], valendo-se de *eigenmaps* e técnicas espectrais para clustering, aplicado a um dataset bi-dimensional gerado aleatoriamente nos padrões dos dados destacados anteriormente.

2 Desenvolvimento

2.1 Funções

A base da implementação consiste em três funções:

(i) *create_dataset(N, cov1, cov2, mean1=[1,1], mean2=[1,1])*, onde N é o número de elementos, $cov1$ é a matriz de covariância que gera o primeiro agrupamento; $cov2$ é a matriz de covariância que gera o segundo agrupamento; $mean1$ indica a posição x e y do ponto médio do primeiro agrupamento e $mean2$ indica a posição x e y do ponto médio do segundo agrupamento. O retorno dessa função é uma matriz $R^{N\times 3}$ que contém os valores x e y de cada objeto e uma coluna destinada a informar a qual cluster o objeto pertence. O objetivo dessa função é gerar um dataset que consista em dois agrupamentos gerados através da distribuição normal multivariada e depois plotá-lo na tela. As bibliotecas utilizadas são:

  (i.i) numpy, que conta com inúmeras funcionalidades matemáticas prontas. Nesse caso, foi utilizada a função *numpy.random.multivariate_normal(mean1, cov1, N)*, que recebe como parâmetro um ponto central, uma matriz de covariância e um número de elementos e gera um dataset aleatório seguindo o padrão multivariado normal; e

  (i.ii) matplotlib, usado para plotar[2] o dataset gerado em um gráfico.

(ii) *en_heat(n_samples, ep, t, D)*, onde *n_samples* é o número de elementos presentes no *dataset*, *ep* (epsilon) e t são paramêtros oriundos da escolha pelos métodos *e-neighborhoods* e *heat kernel*[2], e D é a matriz dos dados aos quais será aplicada a redução dimensional. O objetivo dessa função é implementar as técnicas de redução de dimensionalidade descritas por Belkin e Niyogi[2] e plotar o resultado. As bibliotecas utilizadas são:

[2]Desenhar

(ii.i) numpy, pela função numpy.linalg.eig(R), que retorna os eigenvalues e eigenvectors de uma matriz R;

(ii.ii) matplotlib, usado para plotar o dataset reduzido em um gráfico; e

(ii.iii) scikit-learn, abreviado sklearn, que possui funções relacionadas ao aprendizado de máquina. Nessa função, foi chamada a função *sklearn.preprocessing.normalize* com o propósito de normalizar os dados do dataset de entrada.

(iii) *pca_plot(X, n_samples)*, onde X é a matriz a qual será aplicada o PCA e *n_samples* é o número de elementos presentes no *dataset*. É uma implementação simples que reduz a dimensionalidade do dataset para 1 e plota o resultado. As bibliotecas utilizadas são:

(iii.i) numpy;

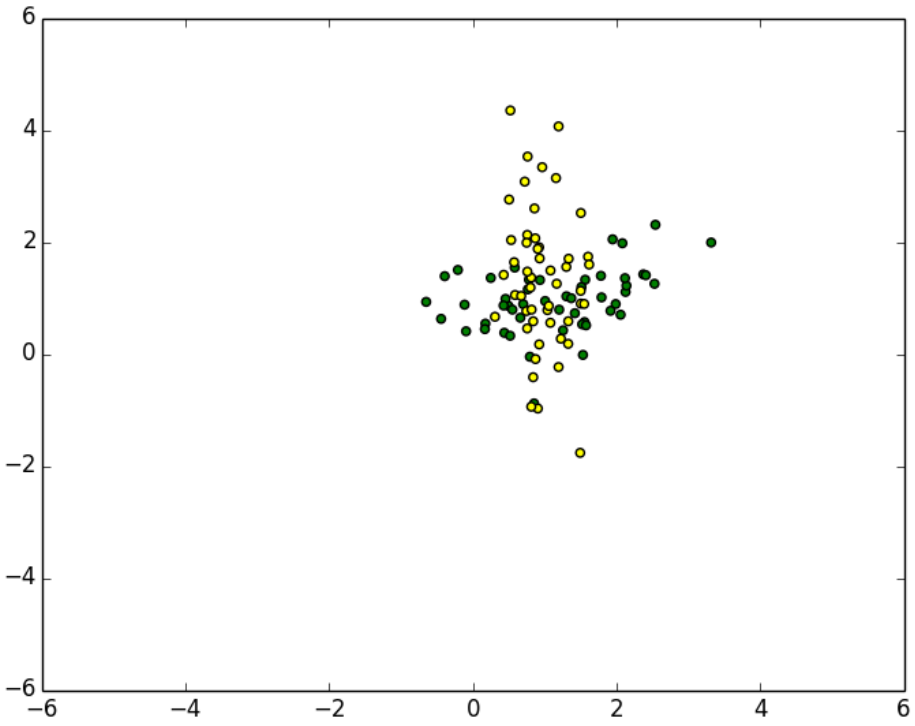
(iii.ii) matplotlib; e

(iii.iii) sklearn, pela função *sklearn.decomposition.PCA*, uma implementação do algoritmo PCA.

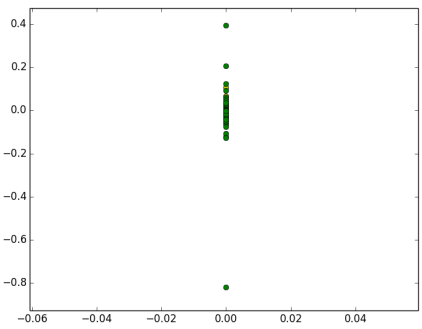
2.2 Comparações com PCA

Nessa seção serão mostrados resultados comparativos de cinco datasets gerados pela função *create_dataset*, nos quais foram aplicadas as funções *en_heat* e *pca_plot*.

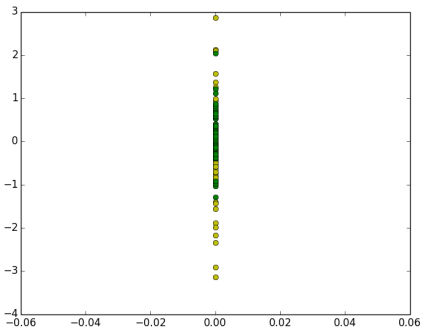
2.2.1 Dataset 1



N=100, cov1=[[0.1, 0], [0, 2]], cov2=[[0.9, 0.2], [0.2, 0.3]]

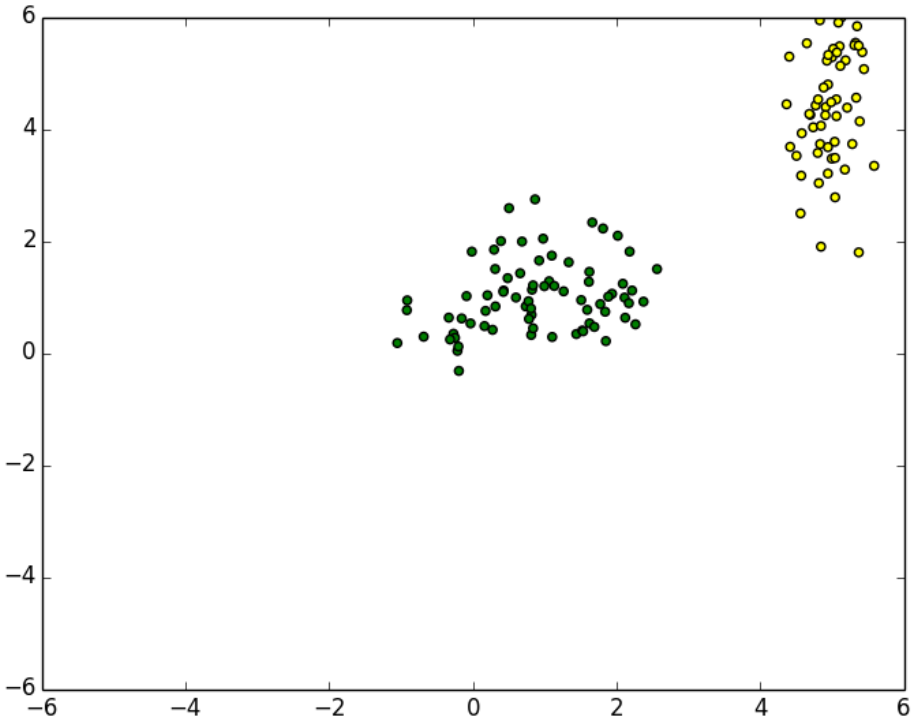


(a) Método Espectral

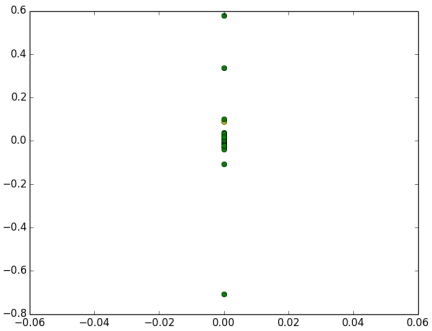


(b) PCA

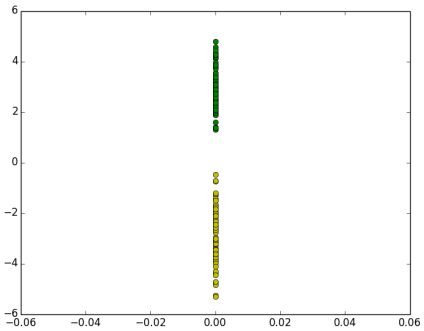
2.2.2 Dataset 2



N=150, cov1=[[0.1, 1], [0, 2]], cov2=[[0.9, 0.2], [0.2, 0.3]], mean1=[5, 5]

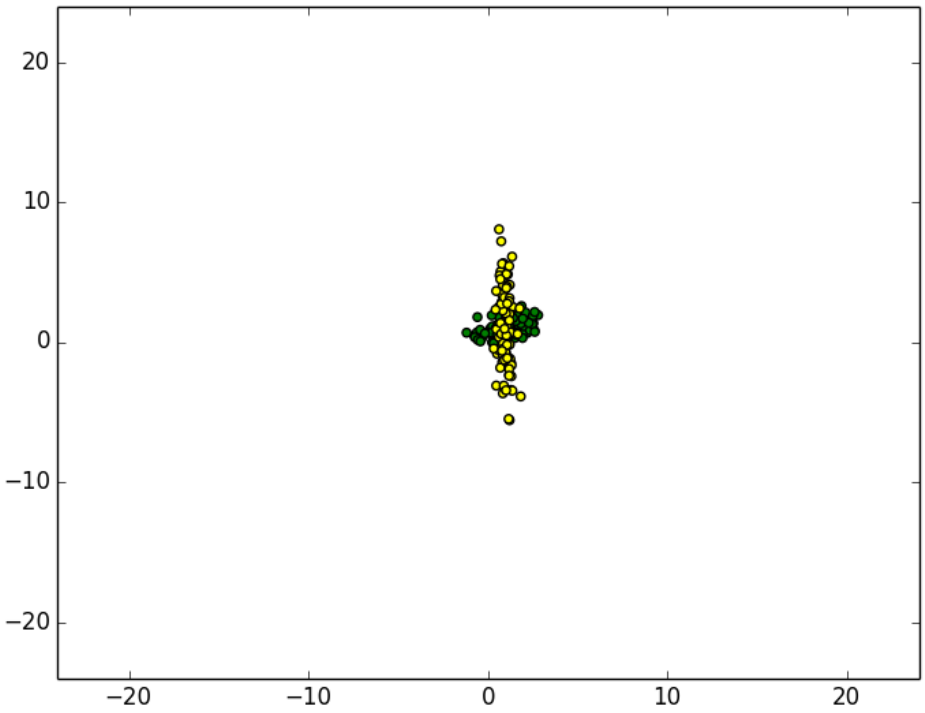


(a) Método Espectral

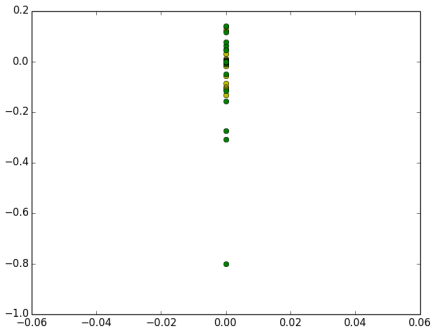


(b) PCA

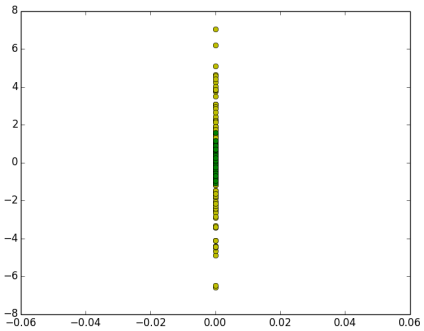
2.2.3 Dataset 3



N=200, cov1=[[0.1, 0], [0, 8]], cov2=[[0.9, 0.2], [0.2, 0.3]]

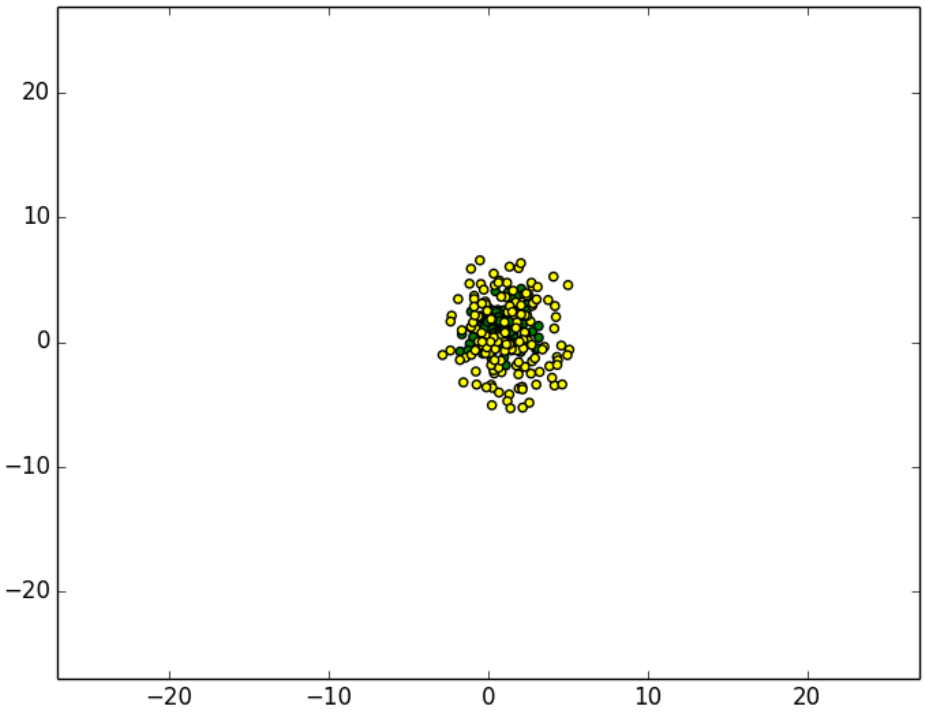


(a) Método Espectral

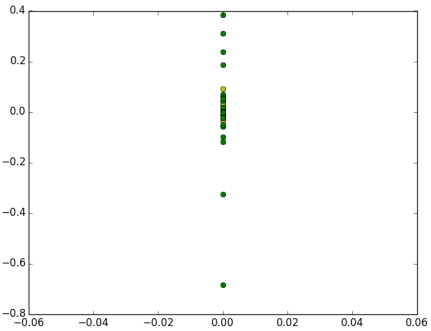


(b) PCA

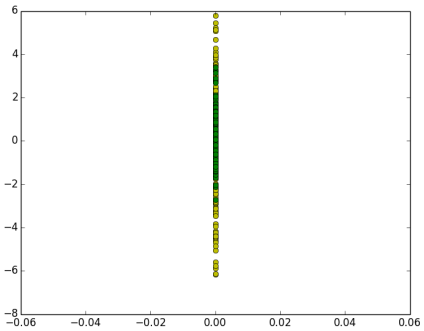
2.2.4 Dataset 4



N=300, cov1=[[3, 0], [0, 9]], cov2=[[1, 0.2], [0.2, 2.1]]

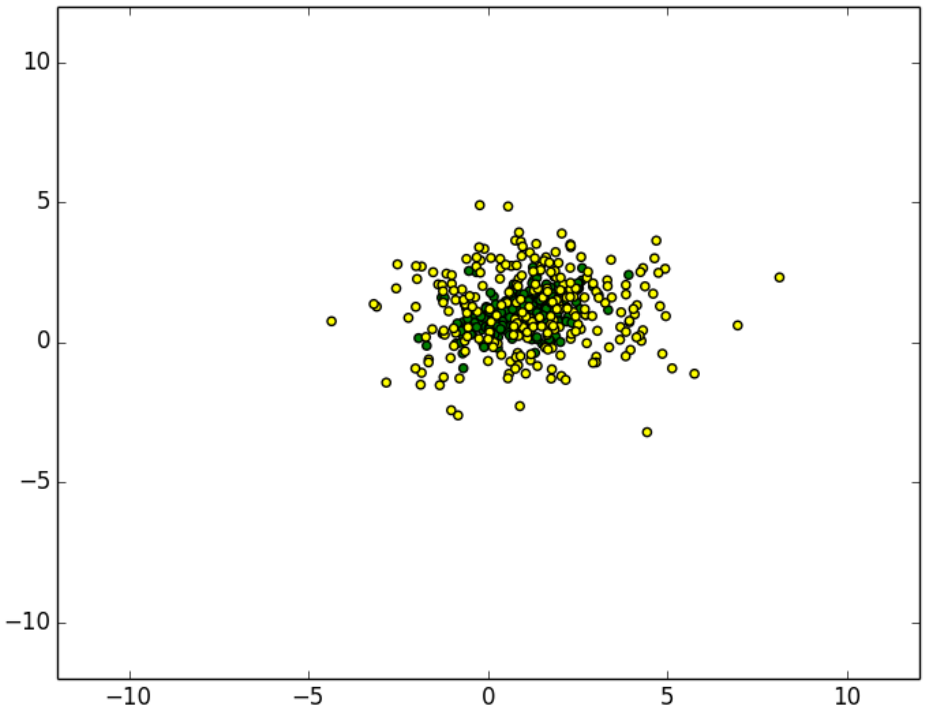


(a) Método Espectral

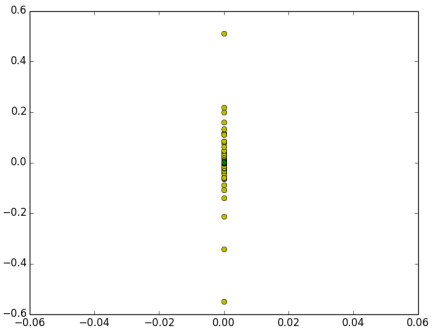


(b) PCA

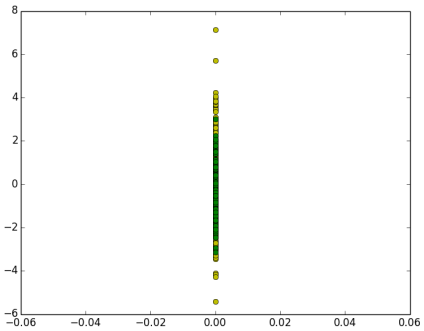
2.2.5 Dataset 5



N=500, cov1=[[4, 0], [0, 2]], cov2=[[0.9, 0.2], [0.2, 0.3]]



(a) Método Espectral



(b) PCA

3 Conclusão

Em retrospecto, a experiência de trabalhar com Python foi positiva. Em termos de desenvolvimento, é uma linguagem bem diferente de C, a única que o autor possuía conhecimento, porque conta com uma sintaxe mais simples e bibliotecas específicas voltadas para a área científica. Houve alguns momentos que o progresso foi afetado por problemas específicos ao lidar com a biblioteca numpy, por esta possuir um tipo específico de *array* (o *ndarray*). Tais obstáculos foram ultrapassados com uma leitura mais profunda sobre o funcionamento da biblioteca. Outros problemas relacionados ao funcionamento da linguagem Python foram resolvidos com pesquisas sobre o tema. Desse projeto, o autor pôde alcançar um aprendizado básico em álgebra linear, matemática aplicada e Python, que definitivamente serão úteis no futuro. O autor agradece ao CNPq e ao CEFET/RJ pelo apoio no desenvolvimento desta pesquisa.

Referências

[1] Bezerra, E., De Lima, L., Krone-Martins, A. (2014) *Spectral Dimensionality Reduction Applied to Stellar Cluster Membership Assignment*, Many Faces of Distances, Campinas, Brasil.

[2] Belkin, M., Niyogi, P. (2001) *Laplacian Eigenmaps and spectral techniques for embedding and clustering*, NIPS 14, pp. 585–591, MIT Press.