FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Aprendizagem Automática

2018/2019

Assignment 2

Relatório

Turno Prático: P2

Realizado por:

45679, Diogo Silvério
47525, Pedro Xavier

Professores:

Ludwig Krippahl
Joaquim Ferreira da Silva

8 de Dezembro de 2018

1. Introdução

O objetivo deste trabalho é examinar o desempenho de três algoritmos de *clustering* diferentes através do problema de agrupamento de eventos sísmicos. Os dados do problema são todos os eventos sísmicos de magnitude maior ou igual a 6.5 na escala de Richter durante um período de 100 anos (1917-2017) obtidos pela USGS. Será feita a parametrização destes algoritmos e serão calculados os índices de validação dos clusters gerados, de modo a que se possa comparar o desempenho final de cada um relativamente aos outros e relacioná-los à informação existente referente às falhas tectónicas conhecidas.

No conjunto de dados fornecido, cada linha corresponde a um evento sísmico e cada sismo é caracterizado por diversas *features* incluindo a longitude, a latitude e um número inteiro que representa a falha mais próxima ao epicentro do sismo. Apesar dos eventos sísmicos terem mais características apenas estas 3 são relevantes no contexto do problema de *clustering*.

2. Pré processamento de dados

Como se trata de um problema de aprendizagem não supervisionada, não será feita nenhuma separação entre conjuntos de treino e teste, não sendo necessário fazer *shuffle* dos dados para garantir uma distribuição proporcional em ambos os conjuntos. Também não será feita a estandardização dos dados, pois estes serão utilizados para calcular distâncias e qualquer alteração ao valor das coordenadas dos sismos irá distorcer os verdadeiros valores das distâncias em questão. Como os algoritmos K-Means e DBSCAN utilizam a distância euclidiana, os resultados obtidos seriam também eles distorcidos.

Como será necessário calcular medidas de distância entre os diferentes eventos sísmicos e tal não é possível com coordenadas geográficas, a latitude e a longitude terão de ser convertidas em coordenadas ECEF (Earth-Centered, Earth-Fixed). Para tal, é utilizada a seguinte fórmula:

$$x = RADIUS * \cos\left(latitude * \frac{\pi}{180}\right) * \cos\left(longitude * \frac{\pi}{180}\right)$$
$$y = RADIUS * \cos(latitude * \frac{\pi}{180}) * \sin(longitude * \frac{\pi}{180})$$
$$z = RADIUS * \sin(latitude * \frac{\pi}{180})$$

3. Explicação dos Algoritmos

K-Means

O algoritmo K-Means é um método de *clustering* particional, exclusivo, completo, e baseado em protótipos. Particional porque divide os dados em grupos ao mesmo nível, exclusivo porque cada ponto pertence a apenas um cluster, completo porque todos os eventos sísmicos irão pertencer a um cluster e baseado em protótipo porque cada sismo irá pertencer ao cluster representado pelo protótipo mais próximo.

O algoritmo consiste em agrupar os dados num número k de clusters dado como parâmetro e tentar encontrar o melhor agrupamento desses k clusters. Os *clusters* são definidos a partir da proximidade de um ponto a cada centróide, o ponto médio de um *cluster* mais próximo de si que é calculado através da média de todos os pontos do cluster. O centróide é um ponto fictício que não pertence aos dados. Os protótipos gerados por este algoritmo podem ser vistos como diagramas de Voronoi.

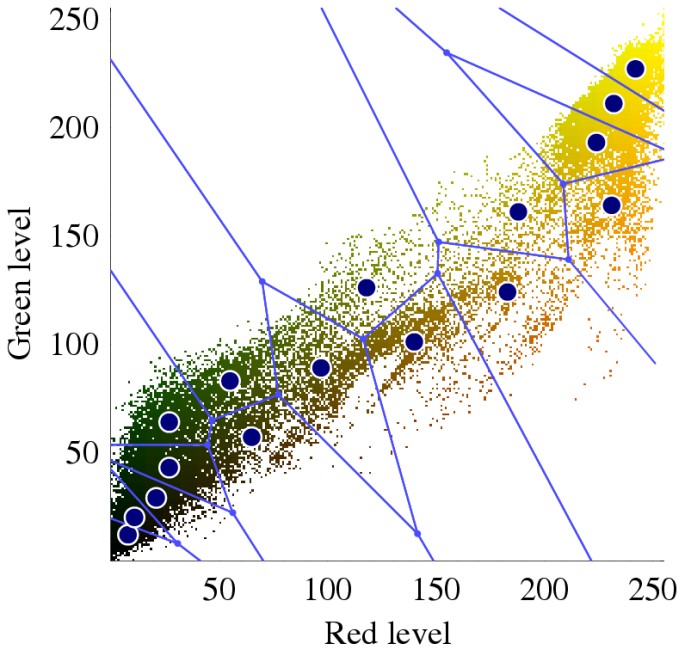


Fig. 1: Diagrama de Voronoi resultante do algoritmo K-Means[1]

Apesar de ser conhecido como um algoritmo, o K-Means é apenas um método de *clustering* e existem diversos algoritmos que o implementam. O algoritmo utilizado neste trabalho foi o algoritmo de Lloyd que começa por definir os k centróides de forma aleatória e atribui cada ponto ao centróide mais próximo. A seguir são feitos dois passos. Primeiro é calculado o ponto médio dos pontos de cada cluster através da distância euclidiana. Depois são actualizados os centróides e todos os pontos atribuindo cada um ao centróide mais próximo de

[1] Dcoetzee assumed (based on copyright claims). Own work assumed (based on copyright claims). Public Domain, https://commons.wikimedia.org/w/index.php?curid=752687

si. Estes dois passos são repetidos até que haja convergência, quando os centróides dos clusters deixarem de ser alterados.

Este algoritmo será negativamente afectado pelos pontos ruidosos, pois estes irão puxar a posição dos centróides para mais próximo de si, provocando erros. O algoritmo também está dependente de um passo inicial aleatório que irá afectar os resultados. Outra desvantagem é que o algoritmo assume que os clusters têm formas bem definidas, compactas e que estão bem separados o que não é verdade para o nosso caso.

DBSCAN

O algoritmo DBSCAN (Density-based spatial clustering of applications with noise) é como indicado no nome um algoritmo baseado na densidade de pontos. O algoritmo tem dois parâmetros: o valor do raio da vizinhança, ε, e o número mínimo de pontos, $minPts$. Este opera com base na vizinhança de cada ponto, $N_\varepsilon(p)$, e classifica um ponto como *core point* se este tiver no mínimo $minPts$ na sua vizinhança, ou seja, pontos que estejam a uma distância menor ou igual a ε, $|N_\epsilon(p)| \geq minPts$. Os pontos contidos na vizinhança de p dizem-se directamente alcançáveis por p. Os pontos q são alcançáveis a partir de p se forem directamente alcançáveis a partir de p ou se forem alcançáveis a partir de um *core point* c que é alcançável por p.

Neste diagrama com minPts = 4, o ponto A e os outros pontos vermelhos são *core points*, porque a vizinhança desses pontos de raio ε contém pelo menos 4 pontos (incluindo o próprio ponto). Como são todos alcançáveis uns pelos outros então formam um cluster. Os pontos B e C não são *core points*, mas são alcançáveis a partir de A através de *core points* e por isso também pertencem ao cluster. O ponto N é um *noise point* porque não é nem *core point* nem é directamente alcançável.

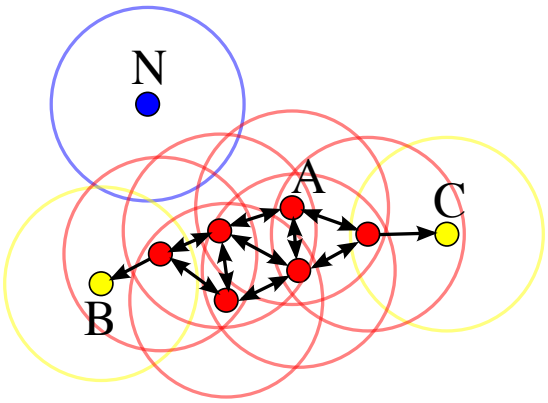


Fig. 2: Diagrama de pontos classificados pelo DBSCAN durante a sua execução[2]

O algoritmo funciona da seguinte maneira. Todos os pontos são visitados e cada ponto é classificado da seguinte forma: caso tenha menos pontos na sua vizinhança do que o mínimo definido, $|N_\epsilon(p)| < minPts$, será presumido que este ponto seja ruído. Caso contrário, é criado um cluster e todos os pontos q dentro da vizinhança deste ponto, $q \in N_\varepsilon(p)$, serão adicionados ao cluster. Por fim, se

[2] Chire - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=17045963

algum ponto q na vizinhança de p tiver igual ou maior número de pontos na sua vizinhança do que o mínimo definido, $|N_\epsilon(p)| \geq minPts$, então os dois clusters serão fundidos.

Ao contrário do *K-Means*, este algoritmo consegue lidar com ruído, pontos *noise*, e consegue também encontrar clusters de diversas formas. Porém o algoritmo é bastante sensível à variação de densidade.

Gaussian Mixture Model

O Mixture Model é um modelo probabilístico utilizado em Estatística para representar a presença de sub-populações dentro de uma população geral, sem que haja a necessidade de um conjunto de dados que identifique uma certa sub-população a que cada dado individual pertença. O Gaussian Mixture Model é uma variante deste método estatístico e pode ser utilizada como método de *clustering* em que é usada uma mistura de distribuições Gaussianas adicionando um peso a cada uma destas distribuições. Para cada ponto é calculada a probabilidade de pertencer a um cluster.

O algoritmo funciona da seguinte maneira. Assumimos que temos que um modelo probabilístico dos nossos dados, X:

$$P(X, Y) = P(X|Y)P(Y)$$

Em que Y é uma variável que pode incluir parâmetros ou variáveis ocultas que associam cada ponto a um cluster. Se assumirmos que θ é um conjunto de parâmetros e Z é o conjunto de variáveis ocultas, podemos definir a função *likelihood* como sendo:

$$L(\theta; X, Z) = p(X, Z|\theta)$$

Em que maximizar a *likelihood* de ambos os dados que conhecemos, X, e os que desconhecemos, Z, dado os parâmetros, θ, é igual a maximizar a *likelihood* dos parâmetros. Se tivermos o Z podemos calcular os pontos onde a função de *likelihood* é máxima e se tivermos os parâmetros, podemos estimar o valor de Z. Formalmente, este é o algoritmo Expectation Maximization.

Primeiro é feito o passo de *Expectation*, onde calculamos a probabilidade de um ponto vir de uma das distribuições gaussianas assumindo um conjunto de parâmetros:

$$\gamma(Z_{nk}) = \frac{\pi_k \mathcal{N}(X|\mu_k, \Sigma_k)}{\sum_{j=1}^{k} \pi_j \mathcal{N}(X|\mu_j, \Sigma_j)}$$

Depois é feito o passo de *Maximization*, em que quando sabemos que a função é máxima, recalculamos os parâmetros:

$$\pi_k = \frac{N_k}{N} \qquad \mu_k = \frac{1}{N_k}\sum_{n=1}^{N} \gamma(Z_{nk}) X_n \qquad \Sigma_k = \frac{1}{N_k}\sum_{n=1}^{N} \gamma(Z_{nk})(X_n - \mu_k)$$

Estes dois passos são repetidos até que haja convergência.

4. Índices De Validação de Clusters

Ao contrário do que acontece em aprendizagem supervisionada onde é possível medir o erro comparando o conjunto de teste com as previsões obtidas, em aprendizagem não supervisionada tal não é possível. Geralmente em problemas de *clustering* apenas podemos considerar o quão bons são os clusters gerados em relação à informação que pretendemos obter. Para fazer validação de clusters utilizamos então um índice interno de forma medir o quão bons são os nossos clusters sem recorrer a qualquer informação exterior.

Silhouette Score

O *Silhouette Score* é um índice interno que mede a distância média de um ponto i a todos os outros pontos do seu cluster, $a(i)$, e a distância média de um ponto i a todos os pontos do cluster mais próximo, $b(i)$.

$$s(i) = \frac{b(i) - a(i)}{\max\left(a(i), b(i)\right)}$$

$s(i)$ toma valores próximos de 1 quando os clusters são muito compactos e estão muito afastados uns dos outros, e valores próximos de -1 quando os clusters não têm estrutura e se sobrepõem. Ou seja, este índice permite-nos saber o quão compactos e afastados são os *clusters* computados. No contexto do problema, este índice não será muito útil devido à forma das falhas tectónicas ser linear e por serem muito próximas umas das outras.

Podemos também utilizar índices externos para validar os clusters com informação externa fazendo um paralelo entre classificação e *clustering*. É criada de forma análoga uma *confusion matrix* como aquela utilizada em aprendizagem supervisionada, mas em que consideramos todos os pares de pontos $N(N-1)/2$, tal que:

- True Positive: é um par da mesma falha do mesmo cluster
- True Negative: é um par de falhas diferentes em clusters diferentes
- False Positive: é um par de falhas diferentes em clusters iguais
- False Negative: é um par da mesma falha em clusters diferentes

	Mesma Falha	**Falhas Diferentes**
Mesmo Cluster	True Positive	False Positive
Clusters Diferentes	False Negative	True Negative

Tabela 1: Confusion matrix para aprendizagem não supervisionada

Rand Index

O *Rand Index* é análogo à *accuracy* vista em aprendizagem supervisionada. Este score calcula a fracção de pares de pontos que pertencem ao mesmo cluster e à mesma falha mais os pares que pertencem a clusters e falhas diferentes sobre todos os pares de pontos.

$$RAND = \frac{TP + TN}{N(N-1)/2}$$

Adjusted Rand Index

O *Adjusted Rand Index* é semelhante ao *Rand Index* mas o *Rand Index* não tem em consideração a possibilidade de fazer *clustering* de um par de pontos que corresponde aos grupos com os quais os comparamos. O *Adjusted Rand Index* corrige este problema subtraindo o valor esperado caso o *clustering* feito não esteja correlacionado com o grupo com o qual está a ser comparado. Um valor igual a 0 indica que não há correlação.

$$ARI = \frac{RI - Expected\ RI}{\max(RI) - Expected\ RI}$$

Precision

A precisão representa, para cada cluster, a fracção do número de pares de pontos bem classificados, que pertençam à mesma falha, sobre todos os pontos desse cluster.

$$precision = \frac{TP}{TP + FP}$$

Recall

O Recall é a fracção de pares de uma falha que foram agrupados no mesmo cluster sobre todos os pares dessa falha.

$$recall = \frac{TP}{TP + FN}$$

Esta não será uma boa métrica pois para valores baixos de k no *K-Means* e c no *Gaussian Mixture Model*, quando houverem muitos poucos clusters, a maioria dos pontos irá pertencer aos mesmos clusters e por isso teremos valores muito altos de Recall. O mesmo acontece para o DBSCAN, mas quando o $\mathrm{epsilon}$ aumenta, pois teremos menos clusters, mas que são maiores e por isso a *Recall* irá aumentar. Este índice não nos permite extrair informação relevante sobre o problema.

F1Measure

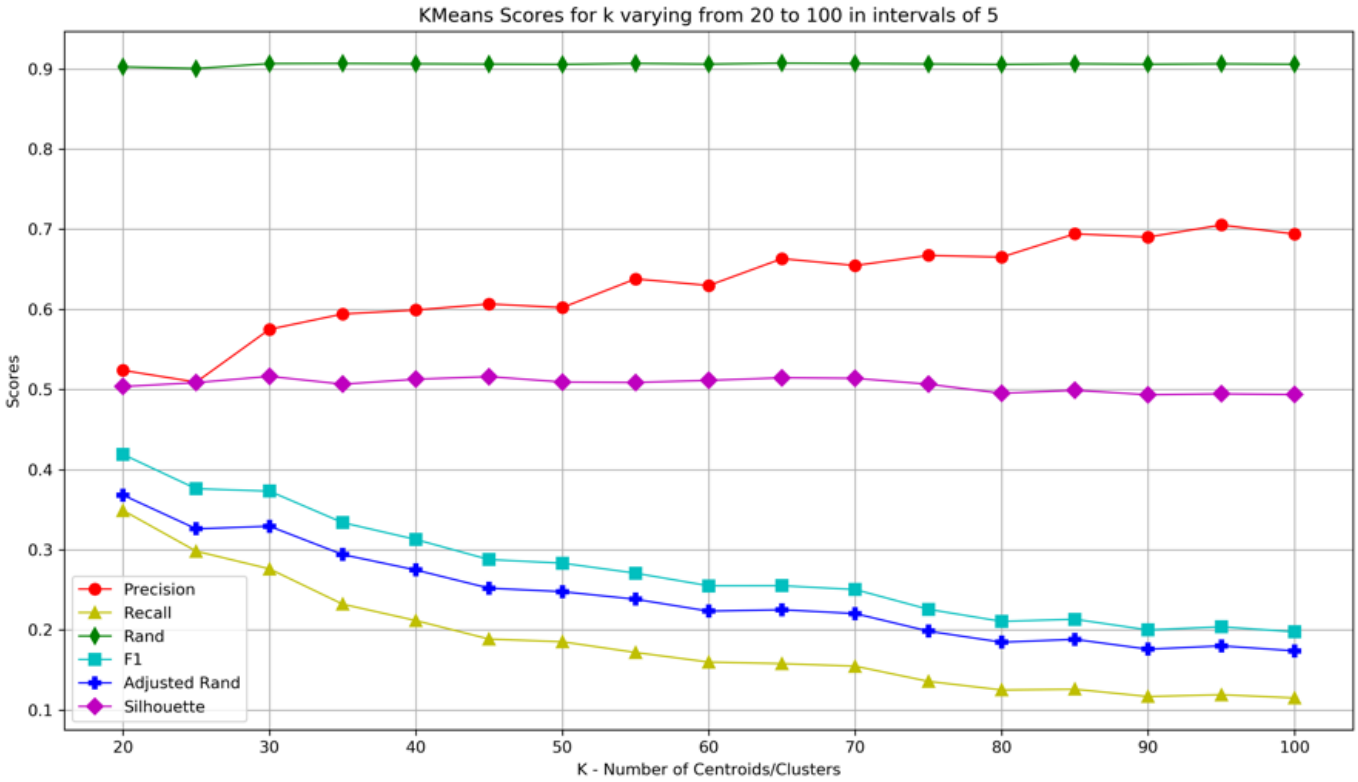
O *F1 Measure* relaciona a precisão e o *Recall* através de uma média harmónica o que permite encontrar um equilíbrio entre ambos os scores.

$$F1 = 2\frac{precision * recall}{precision + recall}$$

5. Análise e Discussão de Resultados

K-Means

Como referido anteriormente, o K-Means atribui cada ponto ao cluster cujo centróide é mais próximo de si e esse centróide é calculado com base na média dos pontos de cada protótipo. Ora como estamos a utilizar todos os pontos do conjunto de dados, aqueles sem falhas associadas, *noise*, irão desviar os clusters resultantes do algoritmo devido ao facto de contribuírem também para o cálculo do ponto médio de cada protótipo. Como se pode ver nas figuras seguintes, houve uma melhoria em todos os índices externos depois de retirados os pontos *noise*.



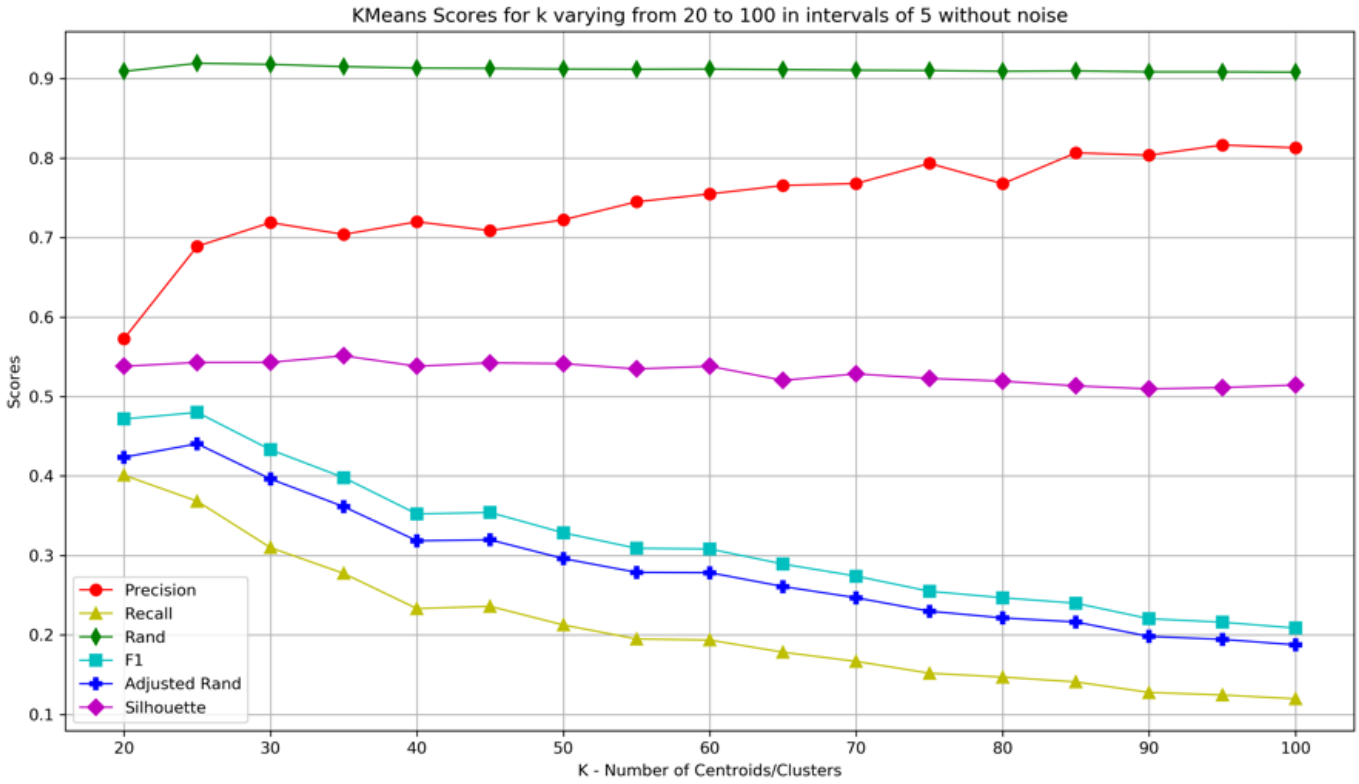


Fig. 3: Gráfico dos scores dos índices de validação em função valor de k.
O primeiro com pontos ruidosos e o segundo sem

Precision (k=95)	Recall (k=20)	Rand (k=25)	F1 (k=20)	Adjusted Rand Index (k=20)	Silhouette Score (k=55)
0.811	0.424	0.916	0.501	0.457	0.541

Tabela 2: Melhor valor de cada índice e o k correspondente, quando executado o K-Means sem noise

Podemos observar que os valores dos índices apresentam o mesmo comportamento para ambos os casos, mas que quando são removidos os pontos sem falhas associadas, os *scores* aumentam significativamente.

Os valores abaixo de 20 foram descartados pois não fazem sentido neste conjunto de dados o que pode ser verificado com o aumento elevado dos valores dos índices de validação. Valores acima de 100 também podem ser desprezados pois a partir deste valor os índices estabilizam, não apresentando grandes melhorias.

Analisando os 6 índices diferentes é possível observar o seguinte:

- *Silhouette* – o valor mantém-se sempre à volta dos 0.5 independentemente do valor de k. Isto acontece porque este score assume que os clusters são compactos e que estarão longe uns dos outros. Isto não acontece porque os eventos sísmicos estão maioritariamente espalhados ao longo das linhas das falhas tectónicas da terra.

- *Precision* – Quanto maior for o número de clusters menor será o número de pares de pontos que pertencem ao mesmo cluster, mas que pertence a falhas diferentes o que explica o aumento do score com o aumento do número de clusters.
- *Recall* – este score diminui conformo o aumento do número de clusters pois o número de pontos da mesma falha agrupados em clusters diferentes começa também a aumentar.
- *Rand* – mantém-se quase constante com um valor de 0.9. Isto acontece porque o denominador deste score é constante e o número de pares de pontos de falhas diferentes em clusters diferentes é bastante elevado.
- F1 – o score diminui conforme o número de clusters cresce.
- *Adjusted Rand Index* – tal como o *Recall* e o *F1*, este score diminui conforme o aumento de k. É possível também perceber que existe bastante aleatoriedade devido à diferença entre o *Rand* e este índice.

Após esta análise, pode-se concluir que o melhor valor de k será 20. Este valor é escolhido com base no valor do Adjusted Rand Index que apesar de não ter o melhor valor é a métrica mais fidedigna pois dá maior peso aos eventos sísmicos bem agrupados de acordo com a falha a que estão associados.

DBSCAN

Como referido anteriormente, o DBSCAN constrói clusters com base na densidade de pontos e ao contrário do *K-Means*, não é completo, consegue construir clusters de variadas formas e lidar com ruído. Os pontos são visitados e cada ponto é classificado da seguinte forma: caso tenha menos pontos na sua vizinhança do que o mínimo definido, $|N_\epsilon(p)| < minPts$, será presumido que este ponto seja um *noise point*. Caso contrário, é criado um cluster e todos os pontos q dentro da vizinhança deste ponto, $q \in N_\varepsilon(p)$, serão adicionados ao cluster. Por fim, se algum ponto q na vizinhança de p tiver igual ou maior número de pontos na sua vizinhança do que o mínimo definido, $|N_\epsilon(p)| \geq minPts$, então os dois clusters serão fundidos.

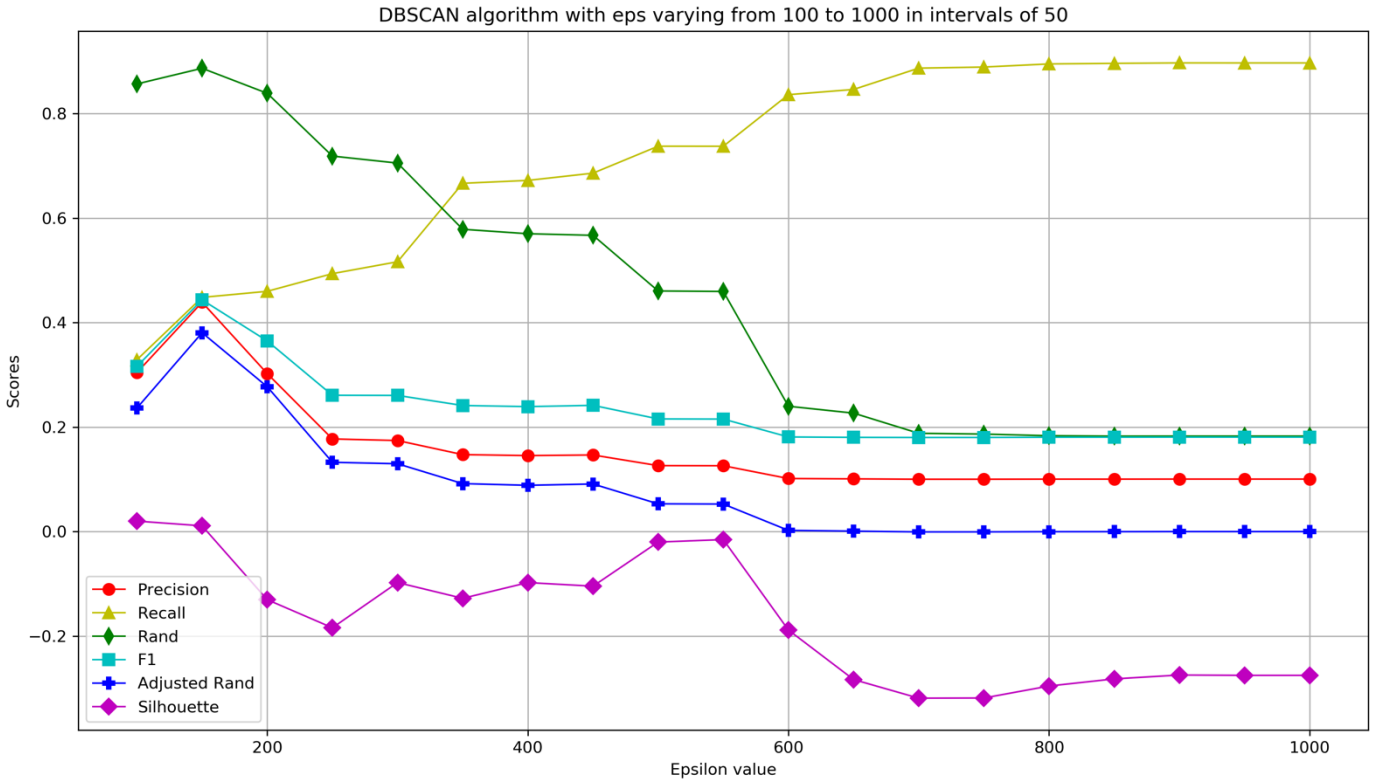


Fig. 4: Gráfico dos scores dos índices de validação em função do valor de epsilon

Precision (eps=150)	Recall (eps=900)	Rand (eps=150)	F1 (eps=150)	Adjusted Rand Index (eps=150)	Silhouette Score (eps=100)
0.439	0.898	0.887	0.444	0.381	0.020

Tabela 3: Melhor valor de cada índice e o epsilon correspondente

O valor de *epsilon* representa o tamanho da vizinhança de cada ponto logo quando este aumenta, o número de clusters diminui, mas o seu tamanho aumenta pois haverá um maior número de pontos dentro das vizinhanças uns dos outros.

O *epsilon* varia entre 50km e 1000km pois valores menores que 50 seriam muito pequenos e não agrupariam os eventos sísmicos, classificando a maioria ou todos os pontos como ruído, e valores maiores que 1000 formariam *clusters* muito grandes que não teriam qualquer valor no contexto do problema e dos dados.

Analisando os 6 índices diferentes é possível observar o seguinte:

- *Silhouette* – mantém-se negativo consoante o aumento do *epsilon*. Isto pode ser explicado pelo facto deste algoritmo conseguir construir clusters com forma linear que é pouco compacta e vai contra a expectativa deste índice.
- *Precision* – diminui com o aumento do *epsilon* pois este algoritmo é parcial. Os pontos que não pertencem a quaisquer clusters por estarem muito afastados são marcados com -1 independentemente da falha a que pertençam, mas durante o

cálculo dos índices, estes serão considerados um *cluster*. Ou seja, haverá um cluster com todos os pontos mais afastados de todos os outros. Por esse motivo o número de *false positives* irá ter um grande aumento. Este problema irá afectar também o *Recall*, *Rand* e *F1*.

- *Recall* – com o aumento do epsilon, este índice aumenta, pois, o tamanho dos clusters aumenta abrangindo maior número de pontos o que irá diminuir o número de *false negatives*.
- *Rand* – diminui porque o número de *true negatives* irá diminuir bastante devido à diminuição do número de *clusters.*
- F1 – diminui, tendo sempre um valor baixo como consequência dos resultados de *precision* e *recall*.
- *Adjusted Rand Index* – este diminui tal como a maioria com o aumento do epsilon. Apesar disso será uma melhor métrica por ter em conta o problema do *cluster* com todos os pontos ruído.

Após esta análise, pode-se concluir que o melhor valor de epsilon será 150. Este valor é escolhido com base no valor do *Adjusted Rand Index* que apesar de não ter o melhor valor é a métrica mais fidedigna pois tem em conta o problema do *cluster* com os pontos ruídos que invalida os outros índices externos.

Método de escolha do parâmetro ε

O método de escolha do *epsilon* no *paper* original referente ao algoritmo[3] envolve calcular a distância de cada ponto ao seu 4º vizinho mais próximo. Depois todas as distâncias devem ser ordenadas em sentido decrescente e deverá ser feito um gráfico com esta informação denominado o *sorted k-dist graph* pelos autores. Este gráfico dá-nos uma ideia da densidade dos pontos no conjunto de dados e um possível intervalo de valores para o *epsilon*. Traçado o gráfico de distâncias, será então feita uma procura pelo "cotovelo" do mesmo. O ponto que dará o valor do *epsilon* ideal marcará o início do "cotovelo" e todos os pontos à esquerda deste serão considerados ruído. Caso seja possível calcular o ruído dos dados este deverá ser calculado, que foi o caso obtendo um valor de 18% de ruído. A percentagem de ruído foi obtida através do seguinte cálculo:

$$noise\ percentage = \frac{pontos\ sem\ falha\ atribuída}{total\ de\ pontos} * 100$$

Este valor pode ser aceite por nós ou então podemos procurar outro valor à direita deste caso não aparente ser o melhor valor de *epsilon*.

[3] A density-based algorithm for discovering clusters in large spatial databases with noise (1996). Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu.

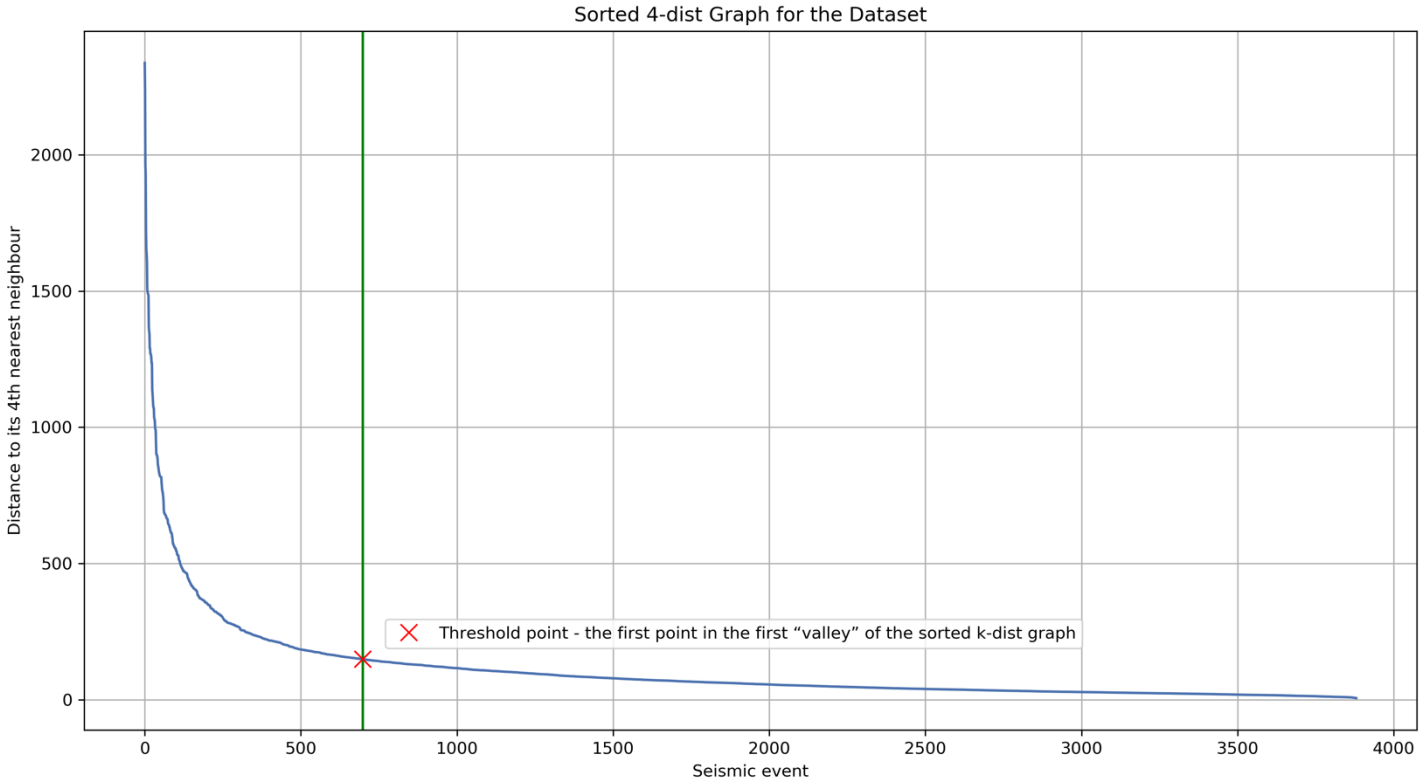


Fig. 5: Gráfico das distâncias de um evento sísmico ao seu 4º vizinho ordenadas por ordem decrescente

A linha verde no gráfico representa o ponto *threshold* que nos dará o valor ideal do *epsilon*. Neste caso o valor calculado do *epsilon* foi igual a 148. É possível ver que a maioria dos pontos se encontra a uma distância muito pequena uns dos outros.

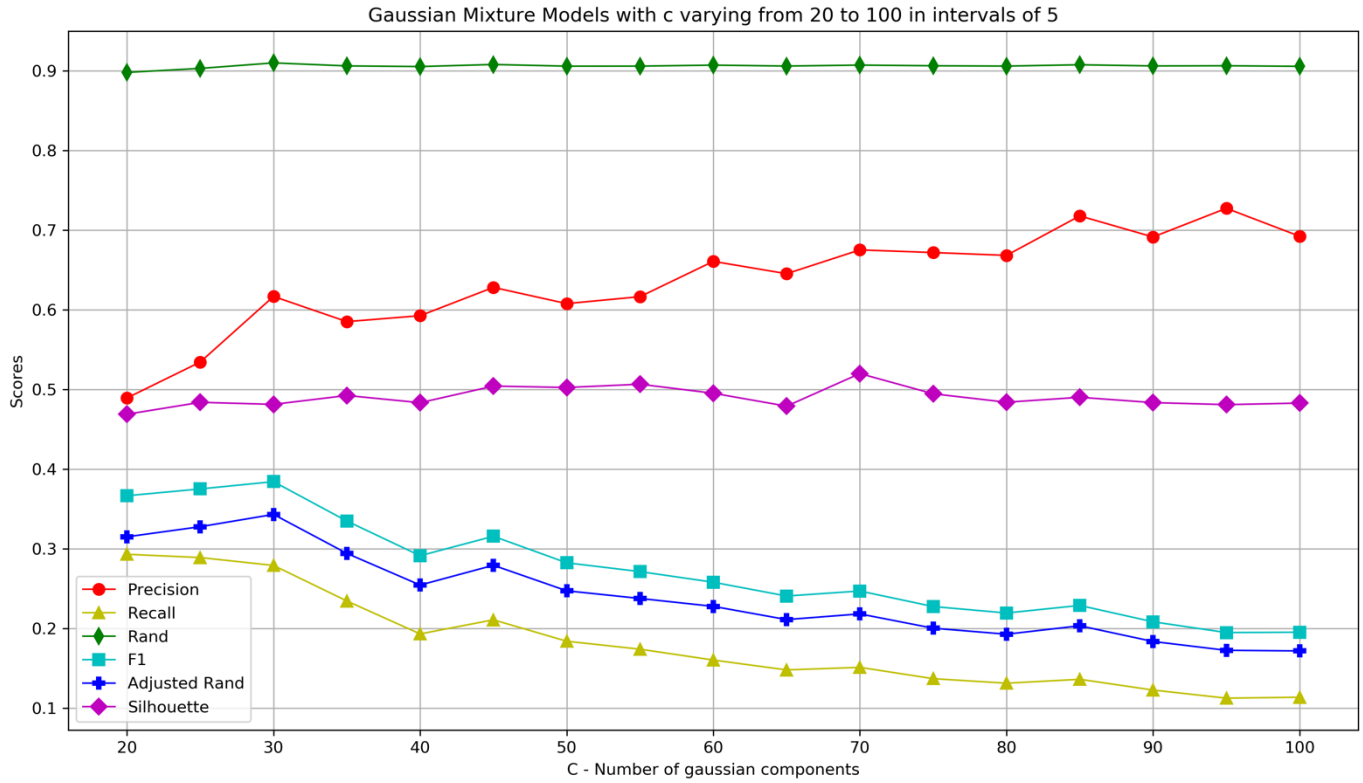
Comparando o valor deste *epsilon* com aquele que escolhemos, $eps = 150$, com base na nossa análise dos índices de validação é possível dizer que este valor é aceitável tendo uma diferença mínima daquele que escolhemos.

O DBSCAN poderá ser utilizado para estudar as zonas do planeta onde ocorre maior número de eventos sísmicos. Depois de obtida esta informação poderão ser estudadas estas zonas para se tentar perceber o que têm em comum.

Gaussian Mixture Model

Como já foi referido, o Gaussian Mixture Model é um modelo probabilístico utilizado em Estatística para representar a presença de sub-populações dentro de uma população geral sem que haja a necessidade de um conjunto de dados que identifique uma certa sub-população a que cada dado individual pertença. Para cada ponto é calculada a probabilidade de pertencer a um cluster. O algoritmo suporta clusters de diferentes tamanhos e formas como o DBSCAN, mas irá ser afectado pelo ruído no conjunto de dados quando calcula a média e a covariância resultando numa distorção da forma dos

clusters formados. Como se pode ver nas figuras seguintes, houve uma melhoria em todos os índices externos depois de retirados os pontos ruidosos.



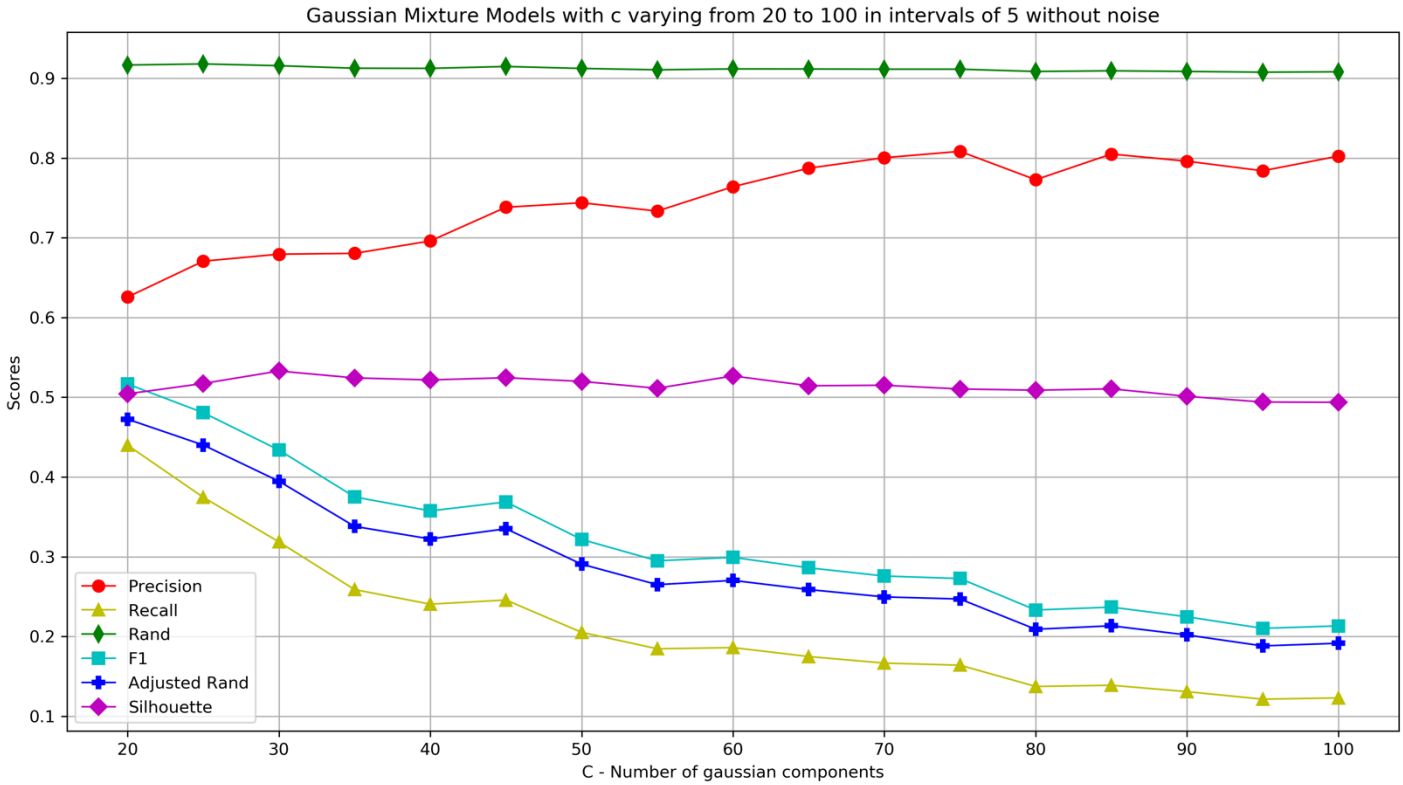


Fig. 6: Gráfico dos scores dos índices de validação em função do valor de componentes. O primeiro com pontos ruidosos e o segundo sem

Precision (c=75)	Recall (c=20)	Rand (c=25)	F1 (c=20)	Adjusted Rand Index (c=20)	Silhouette Score (c=30)
0.808	0.440	0.918	0.517	0.473	0.533

Tabela 4: Melhor valor de cada índice e o c correspondente, quando executado o Gaussian Mixture model sem noise

Podemos observar que os valores dos índices apresentam o mesmo comportamento para ambos os casos, mas que quando são removidos os pontos sem falhas associadas, estes aumentam significativamente.

Os valores abaixo de 20 foram descartados pois não fazem sentido neste conjunto de dados o que pode ser verificado com o aumento elevado dos valores dos índices de validação. Valores acima de 100 podem também ser desprezados pois a partir de deste valor os índices estabilizam, não apresentando grandes melhorias.

Analisando os 6 índices diferentes é possível observar o seguinte:

- *Silhouette* – tal como acontece no *K-Means* o valor do índice mantém-se sempre à volta dos 0.5 independentemente do valor de c. Isto acontece porque este score assume que os clusters são compactos e que estarão longe uns dos outros. Isto não acontece porque os eventos sísmicos estão maioritariamente espalhados ao longo das linhas das falhas tectónicas da terra.
- *Precision* – Quanto maior for o número de clusters menor será o número de pares de pontos que pertencem ao mesmo cluster, mas que pertence a falhas diferentes o que explica o aumento do score com o aumento do número de clusters.
- *Recall* – este score diminui conformo o aumento do número de clusters pois o número de pontos da mesma falha agrupados em clusters diferentes começa também a aumentar.
- *Rand* – mantém-se quase constante com um valor de 0.9. Isto acontece porque o denominador deste score é constante e o número de pares de pontos de falhas diferentes em clusters diferentes é bastante elevado.
- F1 – o score diminui conforme o número de clusters cresce seguindo a evolução do *Recall* e *Precision*.
- *Adjusted Rand Index* –decresce conforme o número de componentes aumenta.

Após esta análise, pode-se concluir que o melhor valor de c será 20. Este valor é escolhido com base no valor do Adjusted Rand Index que apesar de não ter o melhor valor é a métrica mais fidedigna pois dá maior peso aos eventos sísmicos bem agrupados de acordo com a falha a que estão associados.

6. Conclusão

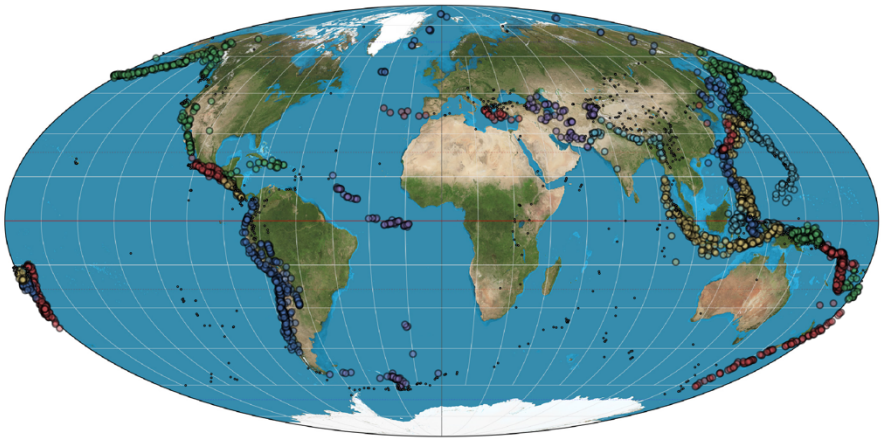


Fig. 7: Mapa dos eventos sísmicos associados às suas falhas correspondentes

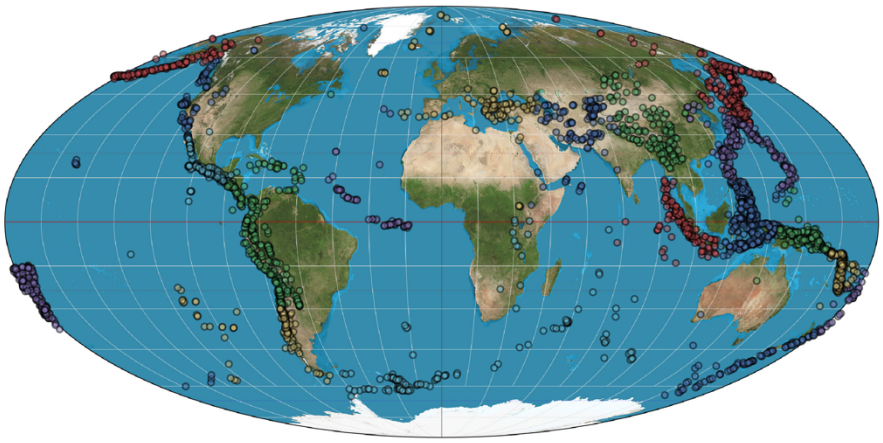


Fig. 8: Mapa dos eventos sísmicos agrupados segundo o K-Means com k = 20

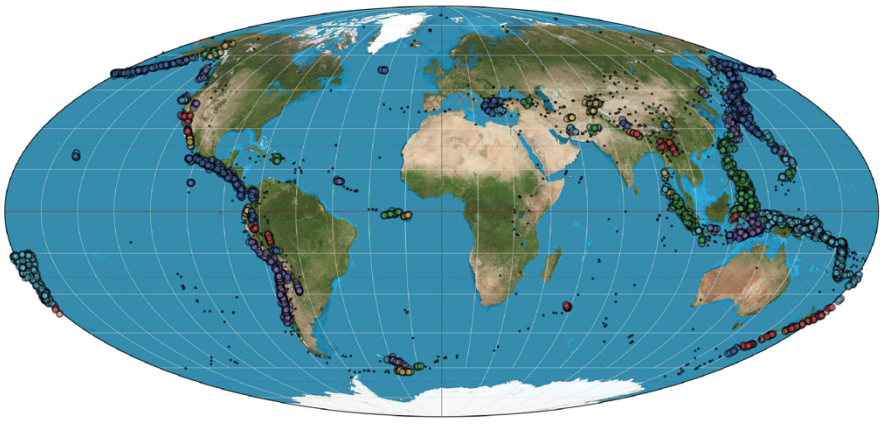


Fig. 9: Mapa dos eventos sísmicos agrupados segundo o DBSCAN com eps = 150

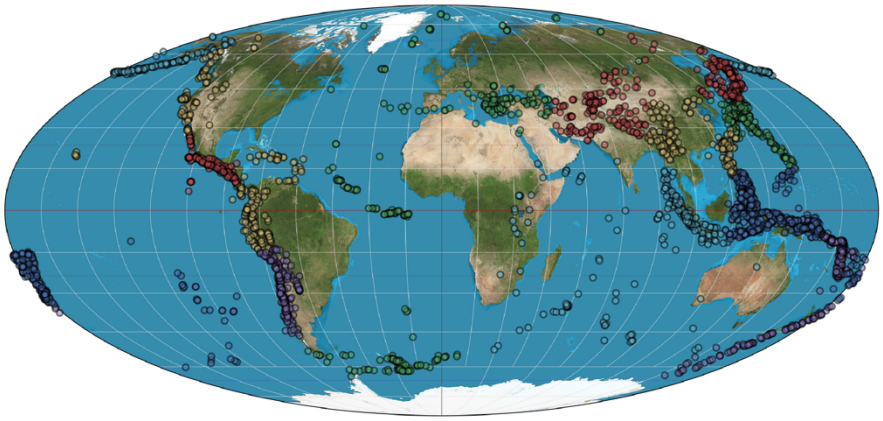


Fig. 10: Mapa dos eventos sísmicos agrupados segundo o GMM com c = 20

Neste tipo de problemas não existe uma resposta única correcta e óptima pois tudo depende do tipo de informação que pretendemos extrair dos *clusters* formados pelos algoritmos, das suas características e do conjunto de dados em questão. Neste caso não é possível determinar qual dos três algoritmos é o melhor pois cada um tem as suas vantagens e desvantagens. O K-Means permite obter um número de *clusters* idêntico ao agrupamento original dos eventos nas suas falhas correspondentes mas de uma forma mais globular, enquanto o DBSCAN permite obter clusters com formas idênticas às das falhas ou seja mais realistas mas devido ao facto de funcionar à base de densidade acaba por agrupar muitos sismos erradamente que pertencem a falhas diferentes que se encontram próximas umas das outras, enquanto o GMM permite obter clusters de formas diferentes, podendo ser considerado uma variante mais geral do *K-Means*.