FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Aprendizagem Automática

17/18

Assignment 2

Report

Turno P1

Prof. Ludwig Krippahl

Ana Simão 42978

Pedro Vieira 42610

Índice

1. Introdução

O intuito deste trabalho é comparar o desempenho de diversos algoritmos de agrupamento, de forma a consolidar a implementação e o funcionamento próprio de cada uma deles. Para tal é fornecido um conjunto de dados reais, referente a eventos sísmicos que ocorreram nos últimos 100 anos, cuja magnitude é igual ou superior a 6,5. Estes episódios sísmicos serão então agrupados de forma a que estejam relacionados entre si bem como relacionados à informação existente referentes às falhas tectónicas conhecidas atualmente e que são, na maioria das vezes, a causa destes sismos. Para tal serão aplicados três algoritmos distintos: *K-Means*, *DBSCAN* e *Gaussian Mixture Models*.

Os dados fornecidos estão organizados de forma a que cada linha corresponde a um evento sísmico, e cada sismo possui vinte e três valores distintos, separados por vírgulas, onde o último valor foi inserido especificamente para o contexto deste projeto, e indica a falha (cada uma representada por um número inteiro) mais próxima (distância) do sismo em causa, os restantes valores foram analisados e selecionados (detalhes na secção 3).

2. Algoritmos de *Clustering*

Clustering é uma técnica que tem como objectivo encontrar agrupamentos nos dados de forma a que no mesmo conjunto estejam presentes exemplos semelhantes e que exemplos distintos estejam presentes em grupos distintos. Estes conjuntos ou grupos são chamados de *clusters*.

Este procedimento é muito utilizado em aprendizagem não supervisionada e é útil para auxiliar na compreensão dos dados e das suas relações e assim tirar conclusões sobre conjuntos de dados muitos extensos ou complexos; outra aplicação pode ser a de reduzir (sintetizar) os dados criando clusters protótipos com propriedades abstratas que substituem um conjunto de dados. Isto pode melhorar a performance da análise ou classificação de exemplos.

O termo *clustering* é utilizado de duas formas distintas, pode representar o método utilizado para executar o agrupamento de dados e pode também referir-se ao conjunto de *clusters* criado no agrupamentos de dados. Dependendo do método utilizado, vão ser criados diferentes tipos de *clustering* (assumindo que o termo se refere ao conjunto de *clusters*). Existe o *clustering* particionado, que divide os dados em vários grupos no mesmo nível e existe também o *clustering* hierarquizado em que os grupos estão *nested* em grupos maiores, em árvore. O conjunto de *clusters* pode também ser classificado como exclusivo (cada exemplo pertence apenas a um *cluster*); como sobreposto (os exemplos podem pertencer a mais do que um *cluster)*; ou como *fuzzy* (cada exemplo pertence a todos os *clusters* existentes com um determinado peso que pode tomar valores entre 0 e 1 (inclusivé), se considerarmos que a soma dos pesos de um exemplo tem de ser

igual a 1, então o *fuzzy clustering* torna-se probabilístico. Outra forma de classificação distingue entre *clustering* completo (todos os exemplos estão atribuídos a um *cluster*) e parcial (existem exemplos que não estão atribuídos a nenhum cluster pois podem ser considerados ruído ou irrelevantes). Por fim podemos ainda considerar *clusters* bem separados (estes dependem apenas dos dados e todos os pontos de um determinado *cluster* são mais semelhantes entre si do que com quaisquer outros pontos fora desse *cluster*); *clusters* baseados em protótipos (os exemplos são agrupados conforme o protótipo mais próximo); *clusters* baseados na continuidade (o *cluster* é formado não pela distância aos outros pontos mas sim pela continuidade dos dados) e *clusters* baseados na densidade (os grupos são formados em zona de alta densidade de pontos).

Como a eficiência do *clustering* está intrinsecamente relacionada ao tipo de dados que estão a ser analisados, bem como do tipo de conclusões que se pretende retirar desta análise, não existe nenhuma "fórmula" pré definida e é necessário para cada caso escolher de entre os inúmeros algoritmos disponíveis atualmente. De seguida apresentamos os algoritmos considerados para este projeto.

2.1. K-Means

Segundo as classificações de *clustering* apresentadas na secção anterior, podemos afirmar que o algoritmo *K-Means* é um método exclusivo, particional, completo e baseado em protótipos. O seu funcionamento é bastante simples e consiste em dividir os dados, o melhor possível, por um determinado número k de *clusters*. Esses *clusters* são definidos através da proximidade dos pontos a um dado protótipo (os chamados centróides) que é encontrada através do cálculo da média dos pontos do grupo, gerando um diagrama de *Voronoi* (Fig. 1). É importante referir que o parâmetro k, ou seja, o número de clusters e de centróides criados pelo algoritmo é pré-definido e o *K-Means* não tenta encontrar nenhum valor para esse parâmetro.

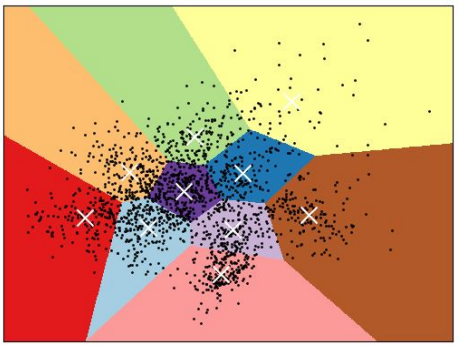


Fig. 1: Exemplo de diagrama de Voronoi resultante da aplicação do K-Means

Quando o algoritmo começa a sua execução, é necessário definir qual o conjunto inicial dos k centróides que vão ser utilizados. Para tal, pode ser utilizado o método *forgy,* que escolhe k pontos aleatório do conjunto de dados como protótipos e de seguida atribui cada ponto ao centróide mais próximo. Um outro método de inicializar o algoritmo é através da partição aleatória, em que após a seleção dos protótipos, os pontos do conjunto de dados são atribuídos a um *cluster* aleatório. Feito isto, os centróides são redefinidos de forma a que se tornem nos pontos médios dos pontos do seu *cluster* e é feita uma nova passagem por todos os exemplos de forma a atribuí-los ao grupo do centróide mais próximo. A partir daí, estes dois passos de recalcular os pontos médios e de atribuir os pontos ao *cluster* mais próximo vão sendo repetidos até ser atingido um estado de convergência em que o centro dos *clusters* não muda com uma nova iteração.

Assim sendo, o correto *clustering* por parte deste algoritmo está fortemente dependente da escolha correto do valor de k, ou seja, de sabermos ou não o número de *clusters* que realmente estão presentes nos dados. No entanto, apesar de esta ser uma condição necessária, não é uma condição suficiente, no sentido em que, mesmo que a escolha do k seja acertada, se os dados não assumirem uma forma compatível com algoritmo, ou seja, se não forem globulares ou se os *clusters* tiverem dispersões muito diferenciadas, os resultados não vão ser os ideais.

É também possível deduzir através destas características, que o resultado final da aplicação do *K-Means* está inteiramente dependente da *seed* aleatória que utilizou para dar início ao seu processo de *clustering*. Por esta razão, deve-se realizar múltiplos testes e fazer uma média dos resultados para que a resposta final convirja para o resultado verdadeiro.

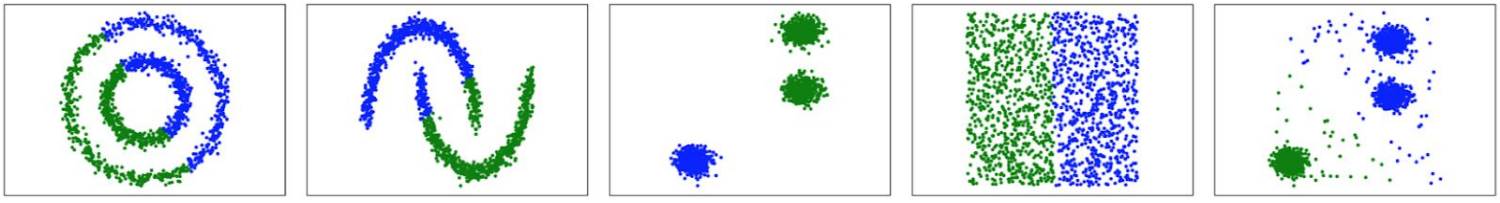


Fig. 2: Exemplos da aplicação do K-Means com k=2

2.2. *DBSCAN*

O algoritmo *DBSCAN* (*Density-based spatial clustering of applications with noise)*, é baseado na densidade dos pontos. Para tal, o algoritmo agrupa, no mesmo *cluster*, pontos que estejam na vizinhança ou que sejam alcançáveis a partir dessa vizinhança (Fig. 3). Isto significa que é necessário definir dois parâmetros: o número mínimo de pontos numa região para que esta seja considerado uma região densa e assim formar um *cluster* (isto permite deixar de fora pontos que estão em regiões de baixa densidade - $MinPts$); e o parâmetro ε que define o raio máximo

da vizinhança de um ponto. Conceptualmente, este último parâmetro representa uma circunferência, descrita com centro em cada um dos pontos, todos os outros pontos que se encontrem contidos dentro desse limite serão considerados vizinhos do ponto central. Para além disso, quando um exemplo da amostra tem de facto $MinPts$ vizinhos a uma distância inferior a ε, esse ponto é designado de *core point* e todos os seus vizinhos e os vizinhos destes vão ser considerados do mesmo *cluster*.

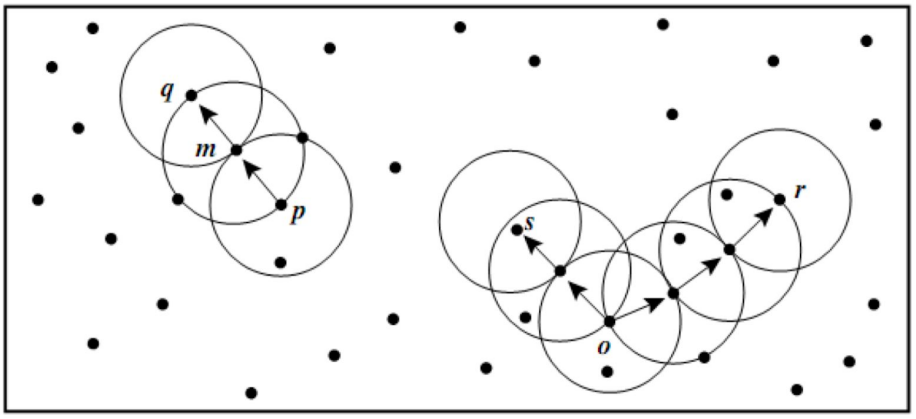


Fig. 3: Visualização da conectividade dos pontos no DBSCAN

Tendo em conta a definição do algoritmo é possível verificar que este apresenta características de parcialidade, na medida em que existe a possibilidade de alguns pontos não terem pelo menos $MinPts$ vizinhos num raio de ε, fazendo com que, tendo em conta a especificação do algoritmo, sejam considerados como ruído e não sejam colocados em nenhum dos *clusters*. O método de escolha e otimização dos parâmetros anteriormente referidos será explanada na próxima secção no entanto, tendo em conta o comportamento do algoritmo, é possível concluir que se se o valor escolhido de ε for muito pequeno, uma grande parte dos dados não será agrupada e, por outro lado, se o valor que foi escolhido for muito alto, os *clusters* tenderão a ficar fundidos e a maioria dos exemplos será atribuído ao mesmo *cluster*.

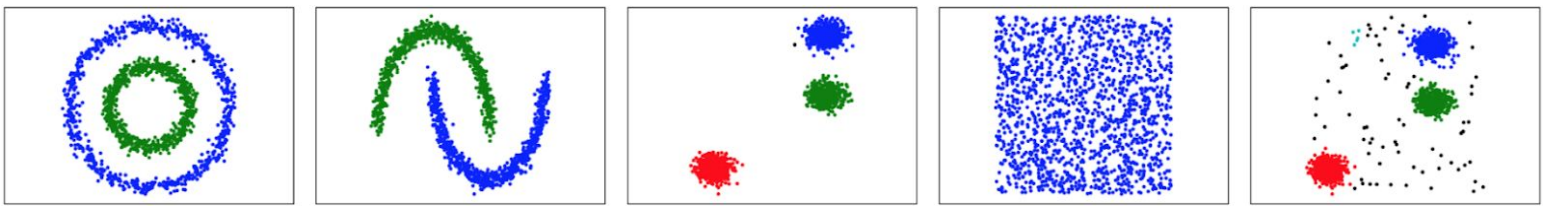


Fig. 4: Exemplos da aplicação do DBSCAN

2.2.1. Implementação

Como foi referido anteriormente, o algoritmo necessita de dois parâmetros para proceder à sua inicialização, $MinPts$ e ε.

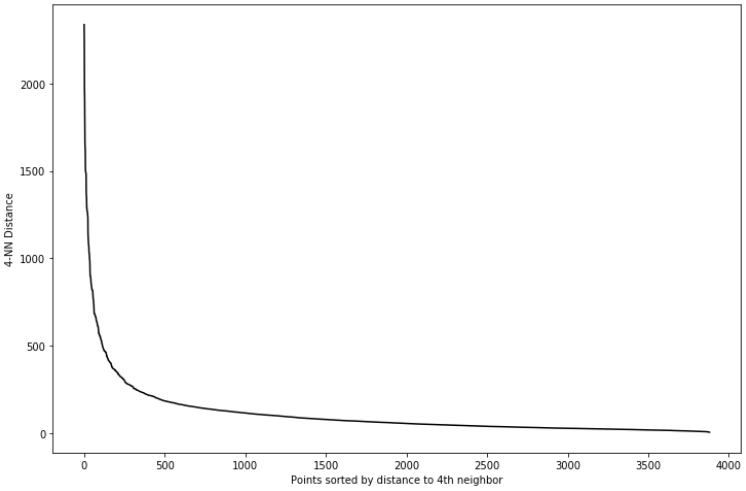
Começando pelo parâmetro $MinPts$, os autores do algoritmo *DBSCAN* referem que, após múltiplas experiências, concluíram que para *data sets* de 2 *features* o parâmetro não surtia grandes alterações nos resultados quando este tinha um valor superior a 4. No entanto, apesar dos pontos no *data set* do problema atual não apresentarem características bidimensionais, em termos práticos e para o contexto deste problema, verificamos que não existe uma grande alteração quando se passa de 2 para 3 *features* pois estas 3 *features* são representativas e calculadas através das duas base (latitude e longitude). Por esta razão, seguimos a pesquisa realizada e utilizamos, para todas as iterações da aplicação do algoritmo *DBSCAN*, o mesmo valor para o parâmetro $MinPts$, 4. Caso o problema requeresse dados de dimensões elevadas, por exemplo superiores a 10 *features*, seria proveitoso aplicar métodos do estudo e variação do $MinPts$ de forma a verificar o(s) melhor(es) valor(es) para os dados a serem clusterizados.

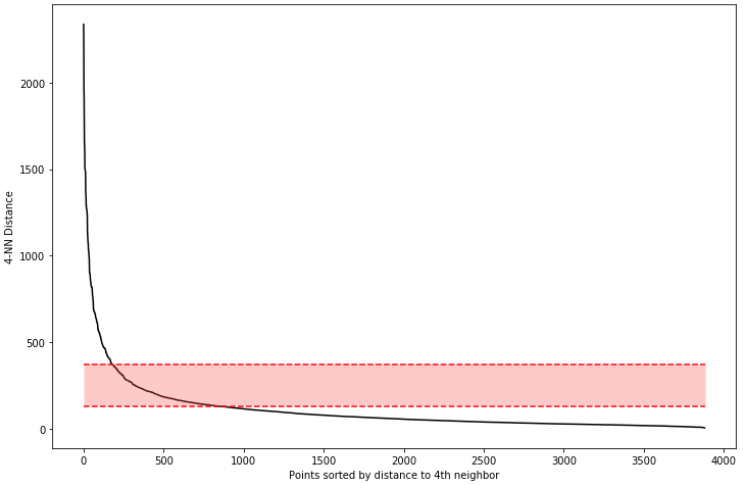
Relativamente ao parâmetro ε, os autores referem duas formas para obter valores adequados ao problema, no entanto, é necessário realizar um método de *preprocessing* de forma poder tirar conclusões. Este processo envolve a construção de um *k-distance graph*, nomeadamente um *4-dist graph* pois é o valor do $MinPts$ que assumimos inicialmente, que consiste em, para cada ponto, obter a distância ao seu 4º vizinho mais próximo, utilizando um classificador *K-Nearest Neighbors* com o parâmetro $k = 4$, e guardar todas estas distâncias numa lista para que depois se realize o *plot* do gráfico com estes valores (garantido que, quando apresentado no gráfico, as distâncias seguem um critério de ordenação descendente).

Após ter sido realizado este passo, os métodos sugeridos pelos autores para a obtenção do ε são os seguintes:

1. Tendo o gráfico *4-dist* construído, o *data scientist* deverá, manualmente, analisá-lo e encontrar o valor do y do *elbow* gerado através das distâncias obtidas. Conceptualmente, este "cotovelo" representa a separação entre zonas consideradas de grande densidade (em que a distância aos seus vizinhos é mínima) e zonas escassas (em que, no pior caso, o 4º vizinho é um dos pontos que se encontra localizado numa das zonas de alta densidade). Assim, o parâmetro ε tomará o mesmo valor do y escolhido. No entanto, como é possível verificar (Fig. 5) e ao contrário do que é apresentado pelos autores, a escolha do *elbow* para o *data set* atual não é intuitiva pois não existe uma clara separação entre os dois tipos de zonas: apenas se consegue concluir que, de zona para zona, as densidades tendem a diminuir gradualmente. Por este motivo, não conseguimos, com um grau elevado de confiança, escolher um dos pontos ao longo do *elbow* para definir o parâmetro ε, mas sim definir um intervalo fechado de valores nos quais consideramos que o cotovelo é

incidente ($y \in [130, 370]$, Fig. 6), e utilizá-lo para testar, posteriormente, a *performance* do algoritmo.





Figs. 5 e 6: 4-dist graph gerado para o data set e intervalo do elbow escolhido tendo em conta os k-dist graph, respetivamente

Paralelamente, tendo em conta a especificação do método acima descrito, é possível constatar que estamos perante um índice interno, pois apenas é utilizado as informações dos próprios pontos do *data set*, sem o auxílio de dados exteriores, o que nem sempre poderá gerar resultados relevantes pois este método apenas tem em conta a estrutura interna dos dados.

2. Alternativamente, os autores referem um outro método que já não envolve a necessidade de escolher o *elbow* manualmente e que pode ser adaptado para qualquer tipo de *data set* de forma mais automatizada. Para isto, apenas é necessário determinar de antemão, através da análise das informações externas dos dados em questão, qual a percentagem de pontos que serão considerados como ruído. Tendo calculado este valor, o método de seleção passa por ignorar os primeiros x pontos e obter o valor da 4º distância do ponto $x+1$. Para o contexto deste trabalho, uma possível escolha da percentagem de barulho do *data set* poderia incidir sobre os pontos sísmicos que não pertencem a nenhuma falha (ou seja, *fault = -1*) o que resultaria na remoção dos primeiros 701 pontos do *k-dist graph*. Desta forma, o ε obtido teria o valor do y (ou seja, a distância ao quarto vizinho) do ponto $701+1$, ou seja, $\varepsilon = y \approx 148$ (Fig. 7).

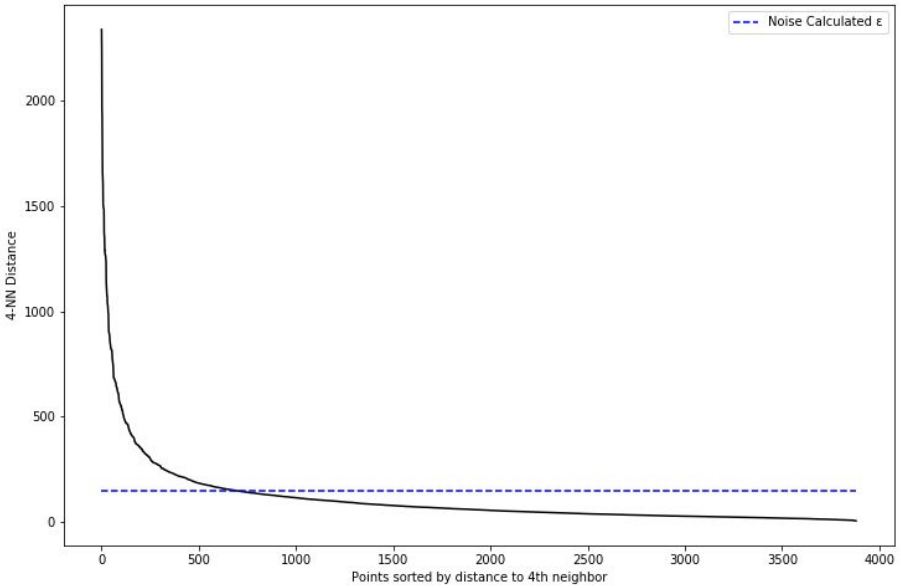
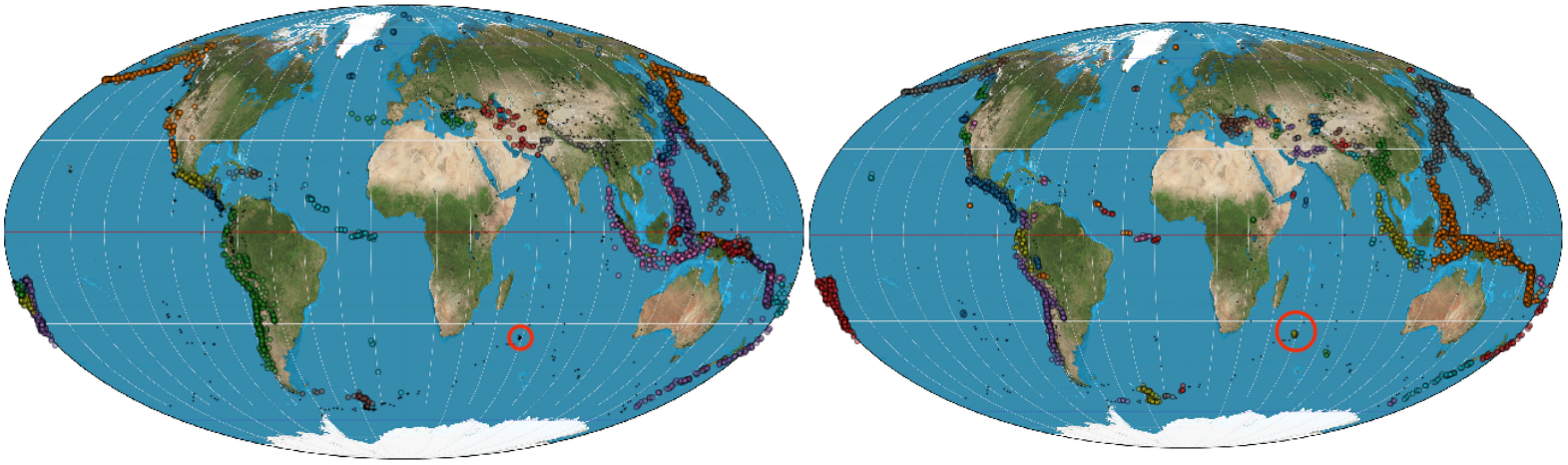


Fig. 7: ε escolhido tendo em conta a percentagem de noise (y = 148)

Inversamente ao que foi referido no ponto anterior, este método já representa uma implementação de índice externo pois a percentagem de *noise* é obtida e escolhida consoante as *labels* externas. No entanto, tendo em conta a remoção dos pontos com *faults* a -1, uma das desvantagens desta abordagem incide no facto do *DBSCAN* considerar uma definição diferente para pontos ruidosos. Especificamente, se considerarmos o mapa original do mundo usando as falhas originais para criar agrupamentos é possível verificar que, para qualquer um dos valores de ε do intervalo obtido anteriormente, pontos que foram considerados ruidosos serão colocados em *clusters* pelo algoritmo (Figs. 8 e 9) fazendo com que o resultado obtido através deste processo possa não ser a escolha mais correta.



Figs. 8 e 9: Exemplo de pontos que logicamente poderão ser considerados como ruidoso (fault = -1) e os mesmos pontos, aos olhos do DBSCAN, não são considerados como ruído, respetivamente

2.3. Gaussian Mixture Models

Estatisticamente falando, um *mixture model* é um modelo probabilístico usado para representar a presença de subgrupos dentro de uma grupo geral, sem que para isso seja necessário possuir um conjunto de dados que identifique explicitamente quais são esses subgrupos. Em problemas de aprendizagem não supervisionada este é um método utilizado para efectuar *clustering* probabilístico.

Este algoritmo de *clustering* é probabilístico, completo e os seus *clusters* são modelados com distribuições gaussianas, logo não representam apenas a média dos pontos como também a sua covariância. Desta forma podemos adaptar o modelo maximizando a *likelihood* dos dados observados (utilizando o algoritmo EM). Este processo vai ser semelhante ao do algoritmo *K-Means* mas os dados vão ser atribuídos a cada *cluster* com uma determinada probabilidade. Uma das vantagens deste método consiste no facto de, através do processo de *clustering,* obtemos um *generative model* do conjunto de dados que permite não só analisar os dados existentes como criar novos exemplos que estão de acordo com as características do conjunto de dados. Uma outra utilização útil passa por comparar dois conjuntos de dados de modo a aferir as suas diferenças.

Analisando em maior detalhe a implementação deste método, considere-se, por exemplo, que existe um modelo probabilístico para a distribuição de um conjunto de dados **X**, com algumas distribuições aleatórias **Y**, a partir das quais os pontos de dados são desenhados:

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

Como referido anteriormente este modelo tem muitas vantagens mas para tal é necessário encontrar **Y**. Se considerarmos que **Y** pode ser decomposto no

conjunto de parâmetros θ e no conjunto de variáveis ocultas Z, a função de *likelihood* para os parâmetros θ dada a informação completa sobre X e Z seria:

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

Tendo em conta que Z não é conhecido, é possível recorrer ao método de Expectativa-Maximização (EM) para resolver esse problema iterativamente.

Assumindo que o conjunto de dados X foi desenhado a partir de um conjunto de distribuições gaussianas, é possível criar uma mistura de distribuições gaussianas, adicionando diferentes distribuições gaussianas com pesos diferentes, de modo a sua soma seja igual a 1. Em mais do que uma dimensão, as distribuições gaussianas:

$$\mathcal{N}(x|\mu, \Sigma) = \frac{1}{(2\pi|\Sigma|)^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

Sendo que Σ representa a matriz de covariância. O mixture model de k gaussianas, onde o π_k representa o peso de cada distribuição Gaussiana, é:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$$

Para cada ponto de dados x, vamos criar uma variável z que assume o valor de 1 em uma de suas dimensões e 0 em todas as outras, onde as dimensões correspondem às distribuições gaussianas. Isso significa que z especifica a partir de qual distribuição gaussiana o ponto x vai ser desenhado. Também podemos definir a distribuição conjunta de x e z como: **p(x, z) = p(z) p(x | z)**.

A probabilidade marginal de qualquer z_k ser igual a um, marginalizada para todos os valores possíveis de x, é igual ao peso da gaussiana correspondente.

Uma vez que $z_k = 1$ para um determinado ponto significa que esse ponto foi extraído da distribuição k, então a probabilidade condicional dos pontos dado que $\mathbf{z_k = 1}$ é a gaussiana respectiva:

$$p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)$$

Assim é então possível aplicar o método EM. Usando a regra de *Bayes*, a probabilidade de $z_k = 1$ dado o ponto x é conhecida, bem como os dados sobre

distribuição de probabilidades de x, podemos escrever a expressão da probabilidade posterior de que a gaussiana k seja responsável pelo ponto x:

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\mu_j, \Sigma_j)}$$

É possível encontrar a solução de *likelihood* dos parâmetros π, μ e Σ se os valores do γ (z_{nk}) forem conhecidos, que por sua vez dependem tanto do X quanto dos parâmetros. Mais uma vez, a solução é usar o método Expectation-Maximization. Partindo de valores aleatórios para π, μ e Σ, o valor de o γ (z_{nk}) e dos parâmetros é recalculado iterativamente até a convergência.

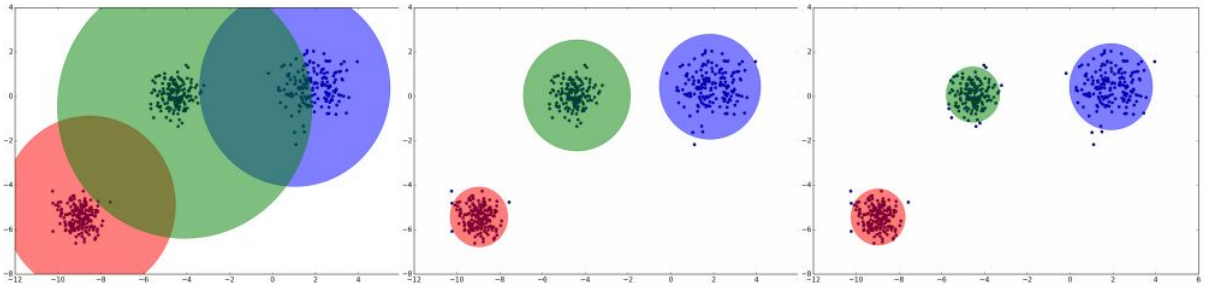


Fig. 10: Exemplo de três iterações do do método EM para o cálculo de um modelo de mistura de 3 distribuições gaussianas

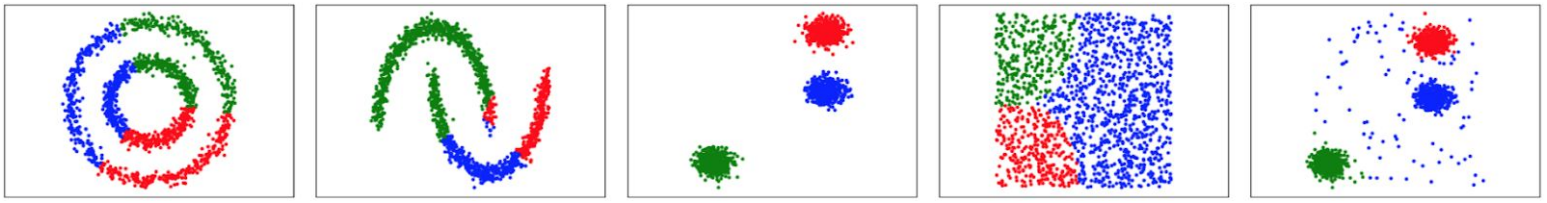


Fig. 11: Exemplos de aplicações do GMM, com c=3

3. Clustering Validation

Tendo em conta que estamos a abordar aprendizagem não supervisionada não podemos simplesmente validar a *performance* dos algoritmos calculando o erro entre as previsões efetuadas e as classes dos dados.

Assim sendo, é necessário (tal como na escolha do algoritmo a utilizar) fazer uso de uma medida de avaliação que esteja coerente com os dados específicos de cada problema. Uma forma de avaliação passa por analisar a semelhança entre dados do mesmo *cluster* e de *clusters* diferentes.

Para tal podemos fazer uso de índices internos e índices externos.

3.1. *Internal Indexes*

A validação interna de *clusters*, como o nome sugere, utiliza apenas informação interna de *clustering* para avaliar a correta estruturação dos *clusters*

Uma das suas utilizações é a estimativa do número de clusters a ser criados.

3.1.1. *Silhouette Score*

O *Silhouette Score* é um índice interno que é utilizado para medir a coesão e separação de *clusters*, verificando, para cada ponto, a sua semelhança para com os restantes pontos do *cluster* que lhe foi atribuído comparativamente aos outros *clusters*. Desta forma, é possível deduzir que esta validação tenderá a devolver melhores resultados quando aplicado a *clusters* globulares e bem definidos. No entanto, este índice não tem em conta características como o formato do *cluster* nem a dispersão intra-*cluster* dos pontos.

Seja *a(i)* a distância média entre o ponto *i* e todos os outros pontos do mesmo *cluster*, e seja *b(i)* a distância média entre o ponto *i* e os pontos do *cluster* mais próximo (a menor distância de *i*), podemos calcular o *silhouette score* para o ponto *i* através da seguinte fração:

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

De forma a obter o valor do *silhouette score* para um *cluster*, é necessário calcular a média dos *silhouette scores* de todos os pontos do grupo. Este *score* pode tomar valores entre -1 e 1, sendo que quanto mais próximo de 1, mais distantes estão os *clusters* (*a* e *b*, no exemplo), logo existe uma maior diferenciação entre os pontos de cada grupo.

3.2. *External Indexes*

Inversamente ao que foi referido na secção anterior, os *external indexes* representam uma classe de validação de *clusters* que utiliza informações externas, conhecidas a priori, para formalizar uma análise personalizada consoante o problema que se pretende resolver. Estes dados exteriores poderão representar *labels* de classes, por exemplo. Para este trabalho, é possível obter esta informação através do campo "*fault*" do *data set* que nos indica a falha geológica a que um evento sísmico pertence.

De forma a calcular os diversos índices externos, é necessário calcular a *confusion matrix* do *clustering* para obter os *true/false positives* e *true/false*

negatives. No entanto, ao contrário do que se pretendia em problemas de classificação, a utilização da *confusion matrix* clássica é inadequada para problemas de clusterização. De forma a solucionar este problema, foi necessário validar cada combinação de pares de pontos através de uma *contingency table*, pois não podemos assumir que as *labels* geradas por um algoritmo são iguais às *labels* externas, ao invés de todos os pontos individualmente, tendo em conta as falhas a que estes pertencem (informação externa) e os clusters em que estes ficaram alojados (informação interna).

	Clusters Iguais	*Clusters* Diferentes
Mesma Falha	*True Positive*	*False Negative*
Falha Diferente	*False Positive*	*True Negative*

Tabela 1: Contingency Table

3.2.1. *Rand Index*

O *rand index* representa a fração de todos os pares de pontos em que quer as *labels* dos *clusters* obtidos quer as *labels* conhecidas a priori, para esses pontos, estão em concordância. Ou seja, existe concordância caso a *cluster label* para o ponto a seja igual à do ponto b e ambos são da mesma *fault*. Para além disso, caso ambos os pontos tenham sido colocados em *clusters* diferentes e, olhando para o *data set*, estes têm também são provenientes de falhas diferentes, também existe concordância.

Este *index* assume valores entre 0 e 1, sendo que 0 indica que os dois *clusters* não partilham nenhum par de pontos e 1 indicando que estes são exatamente os mesmos.

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

3.2.2. *Precision*

A precisão representa, para cada *cluster*, a fração do número de pares de pontos bem classificados para aquele cluster, ou seja, que pertencem à mesma falha, sobre todos os pontos que foram agrupados pelo algoritmo no mesmo *cluster*.

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

3.2.3. *Recall*

Alternativamente à precisão, o *recall* calcula a fração de todos os pares de eventos sísmicos que tiveram origem na mesma falha geológica e que foram agrupados no mesmo *cluster* computado através do algoritmo de *clustering* escolhido.

$$recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

3.2.4. *F1 Measure*

O *F1 Measure* pode ser interpretado como uma média ponderada da precisão e do recall, ou seja esta métrica tem em consideração tanto os falsos positivos e como os falsos negativos e pode tomar valores entre 0 e 1, sendo que quanto maior o resultado melhor a performance medida. De notar que a contribuição relativa da precisão e do recall para o F1 Measure é semelhante.

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

3.2.5. *Adjusted Rand Index*

O *Adjusted Rand Index* surge como forma de colmatar uma lacuna do *Rand Index* que consiste em não explicar a possibilidade de *clustering* de pares de exemplos que coincidirem com os grupos aos quais são comparados. Ou seja o *ARI* mede não só a correta separação de elementos de classes diferentes como também a relação entre elementos da mesma classe. Podemos assim dizer que este *index* dá uma maior peso à relação entre elementos do que à relação entre um exemplo e a sua classe, fazendo com que o *ARI* avalie a capacidade do algoritmo de separar elementos que pertencem a classes distintas. Este índice varia de -1 a 1, onde o valor 0 indica que não existe correlação.

$$ARI = (RI - Expected\_RI) / (max(RI) - Expected\_RI)$$

4. *Detalhes de implementação*

Como referido no capítulo introdutório, o conjunto de dados fornecido representa eventos sísmicos, sendo que cada um deles possui vinte e dois *features* mais uma coluna de informação sobre as falhas. Tendo em conta o contexto do problema, muitas dessas *features* são irrelevantes para o processo de *clustering* e por isso mesmo apenas selecionámos as *features* da latitude e longitude que indicam em que local no planeta ocorreu o sismo (mantendo também a informação sobre as falhas).

Apesar de, como referido, a latitude e a longitude providenciarem uma localização exata do evento no globo, não é possível calcular distâncias pontos com essas medidas, por esse motivo foi necessário converter essa localização para coordenadas esféricas (x, y, z) utilizando as seguintes fórmulas (de notar que *RADIUS* representa o raio da Terra que foi assumido como sendo 6371 km):

$$x = RADIUS * cos(latitude * \pi/180) * cos(longitude * \pi/180)$$
$$y = RADIUS * cos(latitude * \pi/180) * sin(longitude * \pi/180)$$
$$z = RADIUS * sin(latitude * \pi/180)$$

Ao contrário do primeiro projeto, neste problema foi tomada a decisão de não proceder à *standardization* dos dados tendo em conta que estes se referem a localizações geográficas (latitude e longitude) ou posteriormente a distâncias (x, y, z) e portanto os dados e as suas relações ficariam desvirtuadas. Para além disso, como as funções de distância, nomeadamente a *Euclidean distance* que é utilizada em algoritmos de *clustering* como o *K-Means* e o *DBSCAN*, são extremamente sensíveis a diferenças de escala entre os valores, consideramos que é deveras importante manter as posições originais de forma a permitir que a função de distância apresente uma clara separação entre placas geológicas.

Tendo em conta os algoritmos utilizados para a aprendizagem e o próprio contexto do problema, também foi considerado que não faria sentido proceder a um *shuffle* nos dados, pois estes não seriam separados em diferentes conjuntos e porque a sua ordem também não interfere no comportamento dos métodos aplicados nem é necessário realizar processos de obtenção de resultados usando métodos *unbiased*.

5. Avaliação

Para este problema, o *data set* disponibilizado contém informações externas (as falhas a que cada evento sísmico pertence) que nos ajudam a encontrar uma *ground-truth* do que seria expectável obter no melhor caso, independentemente do

algoritmo de *clustering* utilizado (Fig. 12). Para além disso, também é possível verificar que os eventos sísmicos utilizados pertencem a 27 falhas distintas. Nesta secção será feita uma análise do vários resultados obtidos para cada um dos algoritmos e serão feitas algumas conclusões provenientes dessa análise.

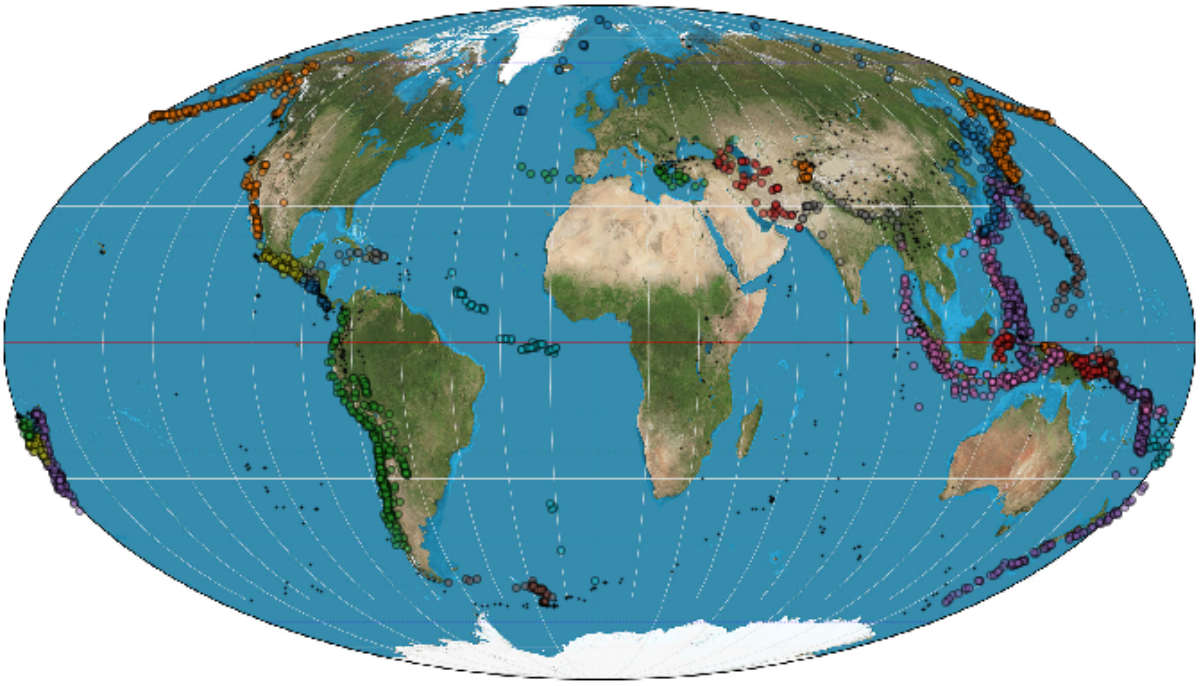


Fig. 12: Mapa do clustering real do data set tendo em conta as falhas verdadeiras

O primeiro passo no processo de análise baseia-se na obtenção de informações de como os eventos sísmicos estão agrupados na vida real, sem qualquer processamento extra, e tentar tirar mais alguns conhecimentos sobre o problema. Para isso, calculámos a incidência sísmica em cada uma das falhas distintas (para ter uma ideia do equilíbrio/desequilíbrio dos dados do sistema) e o *silhouette score* do *data set*.

Com as ocorrências calculadas, verificou-se que não existe uma uniformidade dos dados ao longo das falhas, na medida em que as que têm mais eventos sísmicos contêm entre 550 a 650 pontos e, por outro lado, as que contêm menos ocorrências albergam entre 30 a 50 pontos. Esta situação poderá estar associada à presença de atividades vulcânicas em certas falhas o que faz com que a sua ocorrência sísmica seja tendenciosa perante outras falhas. Para além disso, verificou-se que os sismos que não se encontram perto de nenhuma falha têm dominância no *data set*, perfazendo um total de 701 pontos com a *fault* a -1.

Relativamente ao *silhouette score* original, verificou-se que este ronda os -0.1097. Tendo este dado estatístico, é possível tirar algumas conclusões logo à partida: atendendo à definição deste índice, um valor que se encontra perto do 0 e que é negativo carrega o significado de que os "*clusters*" do mundo real não estão bem definidos e existe alguma sobreposição de eventos sísmicos, provenientes de

uma dada falha, na localização de uma outra. Isto deve-se ao facto de existir falhas que estão diretamente ligadas a outras, sem qualquer separação, o que poderá dificultar a escolha do *cluster* para um ponto que esteja no meio desta ligação.

Intuitivamente, o *data scientist* poderia ser levado a assumir que a melhor *clusterização* (ou seja, a que disponibilizasse um formato de *clustering* mais similar à *clusterização* real) seria aquela em que, para um dado algoritmo e respetivo parâmetro, tivesse o *score* do *silhouette* mais próximo do valor do índice original. Para verificar isso, testou-se todos os algoritmos referidos anteriormente e, para cada um deles, encontrou-se o parâmetro que minimiza a diferença entre o seu *silhouette score* e o do *plain data set* (Tabela 2**;** Figs. 13, 14 e 15).

	Plain Data Set	*K-Means* ($k = 3$)	*DBSCAN* ($\varepsilon = 262$)	*GMM* ($C = 136$)
Silhouette Score	**-0.1097**	0.4277	**-0.1099**	0.3410

Tabela 2: Valores do silhouette score obtido em cada algoritmo que mais se aproximam do silhouette original

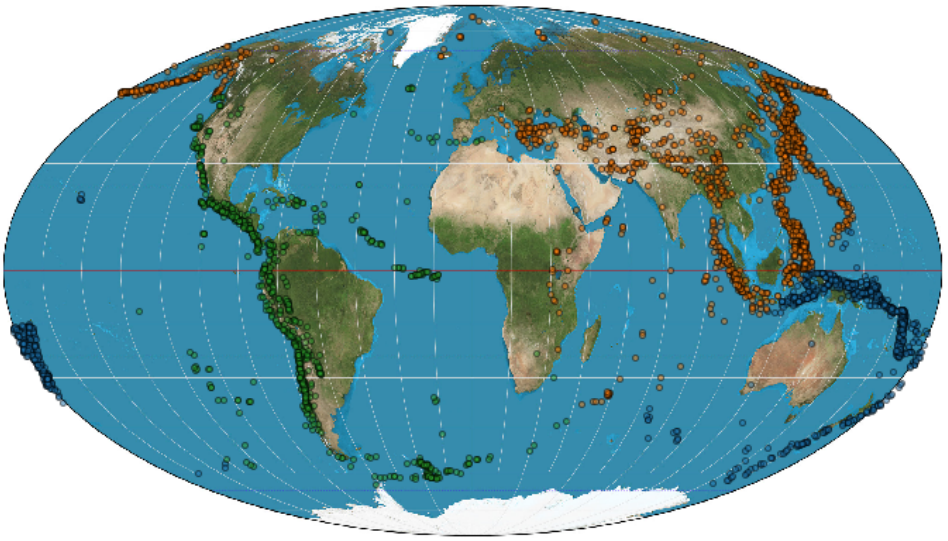


Fig. 13: Mapa do clustering gerado através do K-Means com k=3

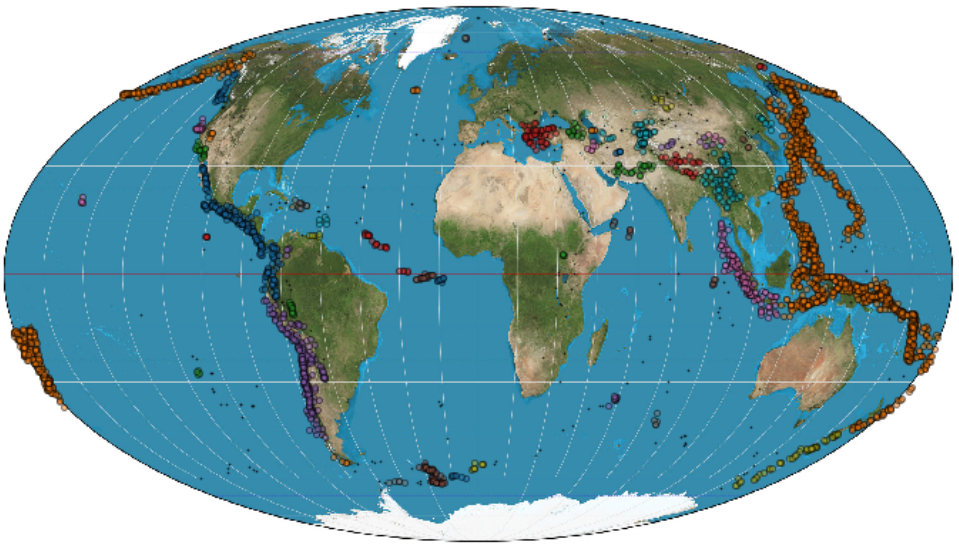


Fig. 14: Mapa do clustering gerado através do DBSCAN com e=262

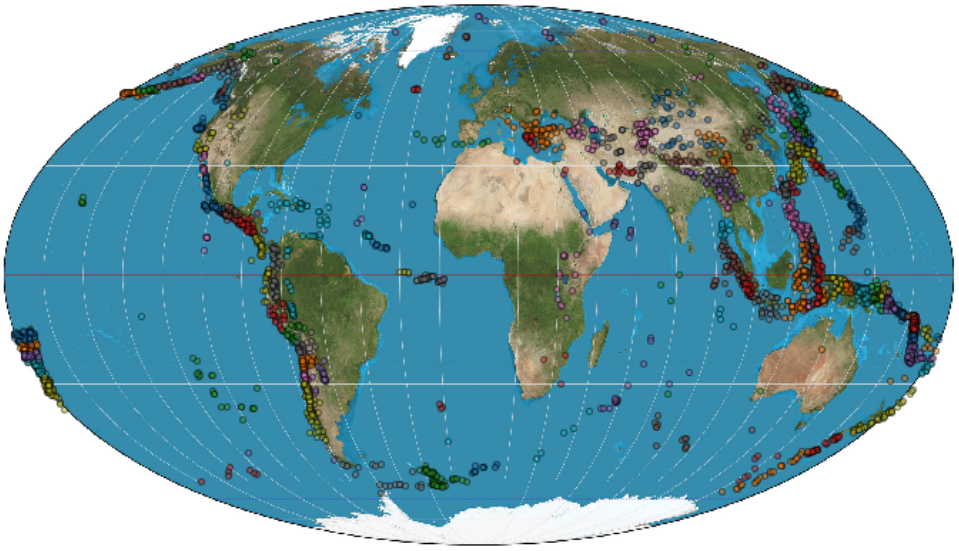


Fig. 15: Mapa do clustering gerado através do GMM com c=136

Focando apenas no *DBSCAN*, pois é aquele que apresenta o *silhouette* mais semelhante ao *score* original, verificamos que a hipótese testada não é satisfeita pois, como é possível verificar pelos *clusterings* gerados por este algoritmo, apresenta grupos completamente distintos comparativamente ao agrupamento real. Um dos sintomas que nos permite concluir isto deve-se ao facto de uma grande quantidade de pontos que deveriam pertencer a falhas distintas ficaram agrupados no mesmo *cluster* (fenómeno mais visível do lado direito da Fig. 14)**.** Desta forma, verifica-se que tentar aproximar ao máximo o *silhouette score* ao valor do índice original para obter os mesmos agrupamentos não se apresenta como a melhor alternativa. Assim sendo, considera-se que o problema deverá ser resolvido tendo como objetivo maximizar uma outra hipótese.

É importante referir que à medida que os resultados relacionados com os algoritmos em questão forem apresentados, será também efectuada uma escolha do valor ótimo para o parâmetro a variar em cada um desses algoritmos. Para tal é necessário analisar os valores apresentados pelos vários índices utilizados no projecto e assim escolher qual das métricas pretendemos maximizar. De notar também que essa opção está também interligada com o contexto do problema, em especial, com o objetivo com que estamos a efetuar o *clustering*. Assim, de forma a ter um indicador comum e que podemos facilmente validar, foi considerado que o objetivo do *clustering* é obter um agrupamento o mais semelhante possível ao mapa original (que distribui os sismos pelas falhas). Na secção da conclusão serão considerados outros casos de estudo mais específicos e, possivelmente, mais interessantes do que este.

5.1. K-Means

Devido à sua natureza do algoritmo K-Means, ou seja, o facto de efectuar um clustering completo e de utilizar protótipos, escolhendo a nova posição a cada iteração através da minimização das distâncias aos pontos mais próximos, é facilmente perceptível que os pontos que não têm qualquer falha atribuída, no *data set* original, irão influenciar negativamente a posição dos protótipos mais próximos, obrigando-os a deslocar-se para mais perto de si. Ao retirar todos os pontos que tenham a *fault* a *-1* é possível verificar (Figs. 16 e 17) que todos os índices apresentam melhorias de *performance*, mostrando que a remoção de pontos que poderão ser considerados como *noise* à partida, consoante o que se pretende obter do problema, tende a ser um processo crítico para um bom funcionamento do algoritmo. Uma forma de atenuar este efeito, sem a exclusão de pontos, passaria por utilizar a versão *K-Medoids* que poderia impedir a atração de pontos mais distantes no mesmo *cluster*. No entanto, para efeitos de realização deste trabalho prático, esta alternativa não foi tomada em consideração durante o desenvolvimento.

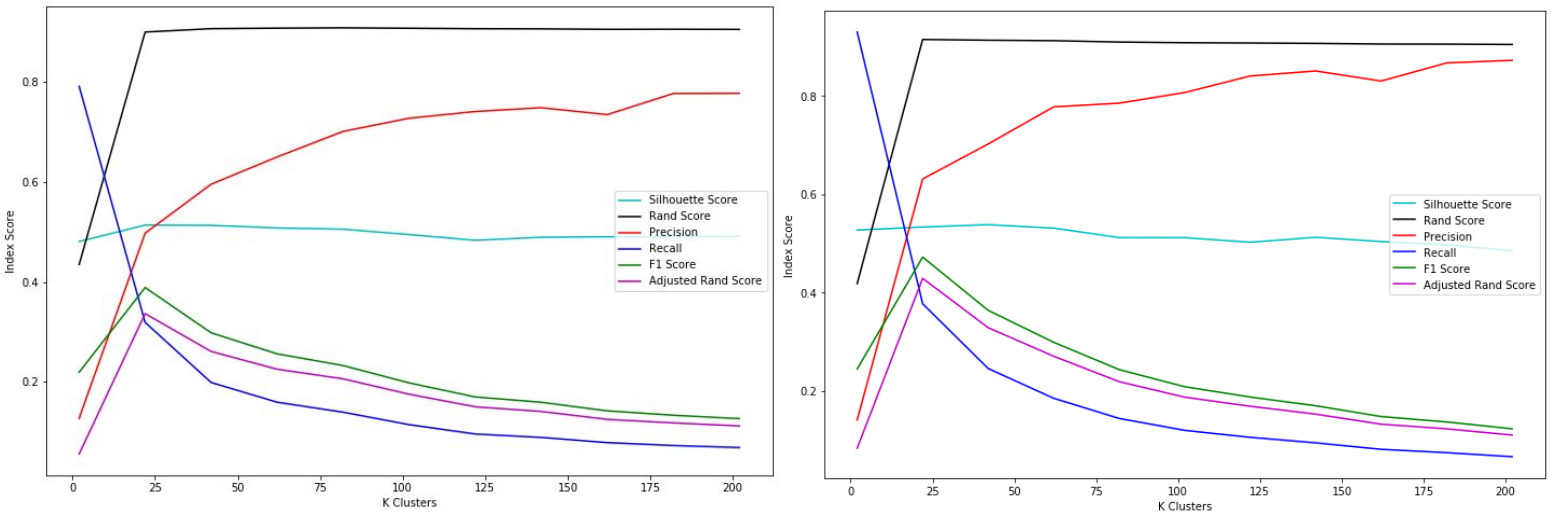


Fig. 16 e 17: Variação dos valores dos índices, com o K-Means, quando se mantém os pontos com falhas originais a -1 e quando se retiram estes, respetivamente

Observando os gráficos anteriores é possível constatar que o comportamento do algoritmo e respetivamente dos índices utilizados não se altera, ou seja, os mesmo scores continuam a aumentar ou diminuir da mesma forma em ambos os casos, apenas o seu valor absoluto sofre um aumento (pelos motivos supra-referidos). Partindo desta observação é também concluir que, aproximadamente, a partir do valor de $k = 100$, os resultados dos índices tendem a estabilizar e portanto deixam de ser relevantes para conclusões mais aprofundadas. Por um motivo semelhante, também foram descartados os valores muito baixos de k (nomeadamente $k < 20$) pois tendo em conta o contexto e *data set* do problema, bem como do comportamento do algoritmo *K-Means*, os resultados para um k muito baixo vão ser inadequados e podemos constatar isto mesmo verificando o aumento constante dos valores de todos os *scores* ao longo do aumento inicial do número de *clusters*.

Assim, apresentamos a Fig. 18 com a gama de valores já adaptada de forma a que os resultados possam ser analisados com maior detalhe.

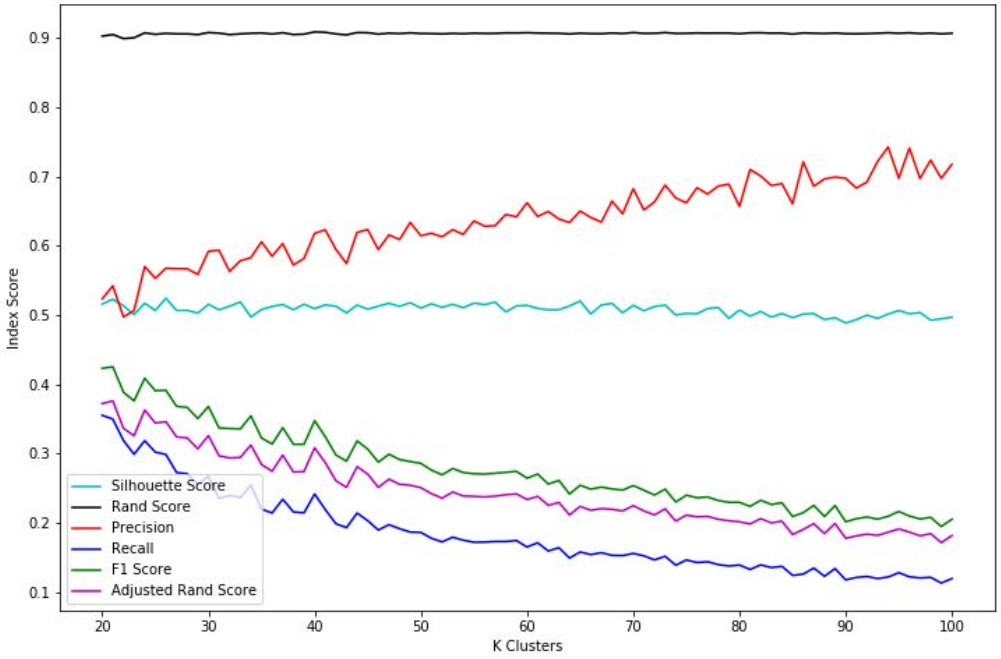


Fig. 18: Variação dos valores dos índices, com o K-Means, variando o k de 20 a 100

Silhouette Score ($k = 26$)	*Rand Index* ($k = 40$)	*Precision* ($k = 94$)	*Recall* ($k = 20$)	*F1* ($k = 21$)	*Adjusted Rand Index* ($k = 21$)
0.5244	0.9088	0.7426	0.3555	0.4254	0.3762

Tabela 3: Os melhores valores para cada índice, com o K-Means, variando o k de 20 a 100

Se voltarmos a pensar no funcionamento próprio do *K-Means*, este tende a ser uma escolha viável quando estamos perante um *data set* em que os dados estão têm uma baixa variância e uma dispersão semelhante na amostra de forma a que o algoritmo crie *clusters* globulares. Se estivermos perante este "caso perfeito" de aplicação do *K-Means*, o índice que à partida teria melhor rendimento seria o *Silhouette Score*, pois este também tende a ter melhores resultados quando analisa clusters globulares e coesos (devido ao seu modo de funcionamento que foi explicado na secção anterior). Tendo em conta que no contexto do problema em análise, os dados não estão distribuídos desta forma, o *Silhouette Score* não será uma métrica adequada para medir a eficiência dos algoritmos em teste neste trabalho. Podemos também verificar nos gráficos apresentados que o valor dado pelo *Silhouette Score* mantém-se mais ou menos constante, isto ocorre porque se o número de *clusters* for pequeno então os *clusters* formados não vão ter uma diferença de distância entre os seus pontos e o cluster mais próximo muito significativa (o que resulta num *score* mais baixo), visto que os *clusters* vão tender a ser muito grandes, devido à forma como os dados estão distribuídos, e isto ainda se agrava mais se considerarmos os pontos -1, que vão "puxar" os *clusters* para

perto de si; da mesma forma se for utilizado um valor de k elevado, vão ser gerados mais *clusters* o que vai fazer com que estes sejam mais coesos e portanto os pontos dentro do mesmo *cluster* estarão mais próximos, no entanto, quando isto acontece, a distância para o *cluster* mais próximo também diminui, o que resulta num *score* semelhante para ambas as situações.

Tendo em conta que existem 27 falhas distintas no *data set*, e portanto o ideal seria que o algoritmo dividisse os dados em 27 *clusters*, podemos ser tentados a escolher um $k = 27$ ou um valor próximo deste número, no entanto, tal como referido o conjunto de dados do problema não tem um forma, ou seja, não está distribuído de forma regular e por isso, mesmo que o k seja definido como 27 (o suposto número correto de *clusters*), o algoritmo irá atribuir os pontos a *clusters* errados ou vai juntar pontos que não deveriam pertencer ao mesmo conjunto. O resultado desta execução pode ser observado na Fig. 19.

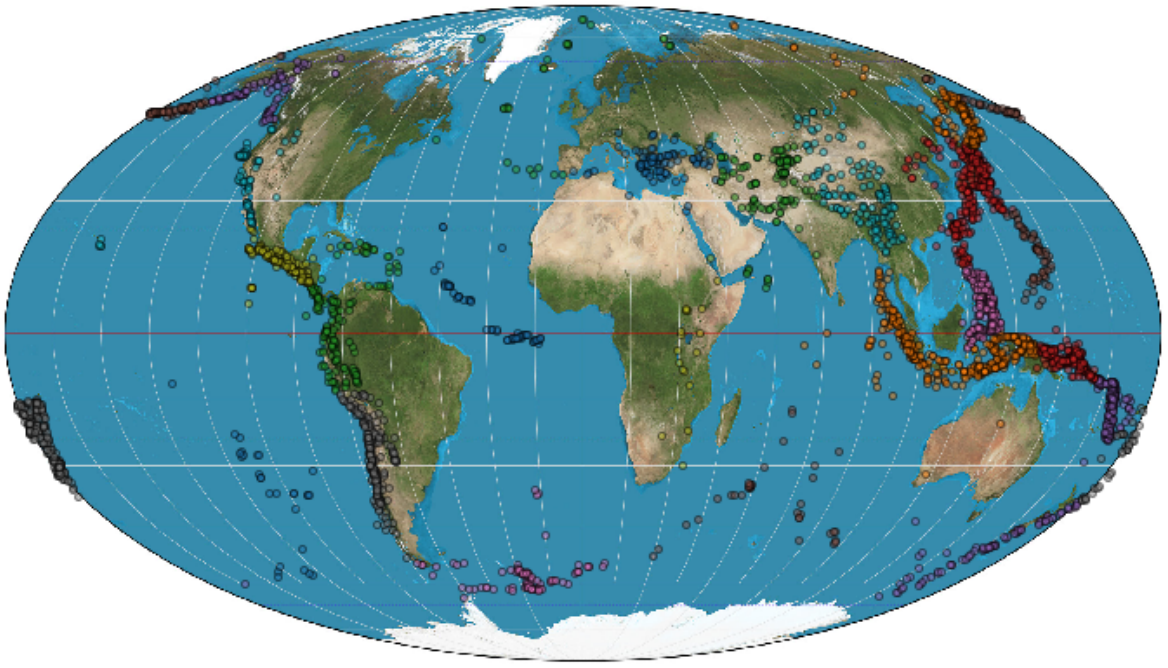


Fig. 19: Mapa do resultado da clusterização do K-Means com k=26

Olhando agora para os resultados obtidos com os valores dos índices externos, podemos verificar que a precisão tem um valor bastante elevado para um valor de k também ele elevado. Isto faz sentido visto que quanto maior o valor de k, menor o número de pontos por cluster formado e, por isso, maior o número de exemplos comparados que pertencem de facto à mesma falha dentro desse *cluster*. Embora os *clusters* formados (representados na Fig. 20) não sejam os pretendidos, ou seja, não correspondam à informação sobre as falhas, existem situações em que pode ser útil utilizar um valor de k que maximiza a precisão (falaremos um pouco mais sobre isto na Conclusão).

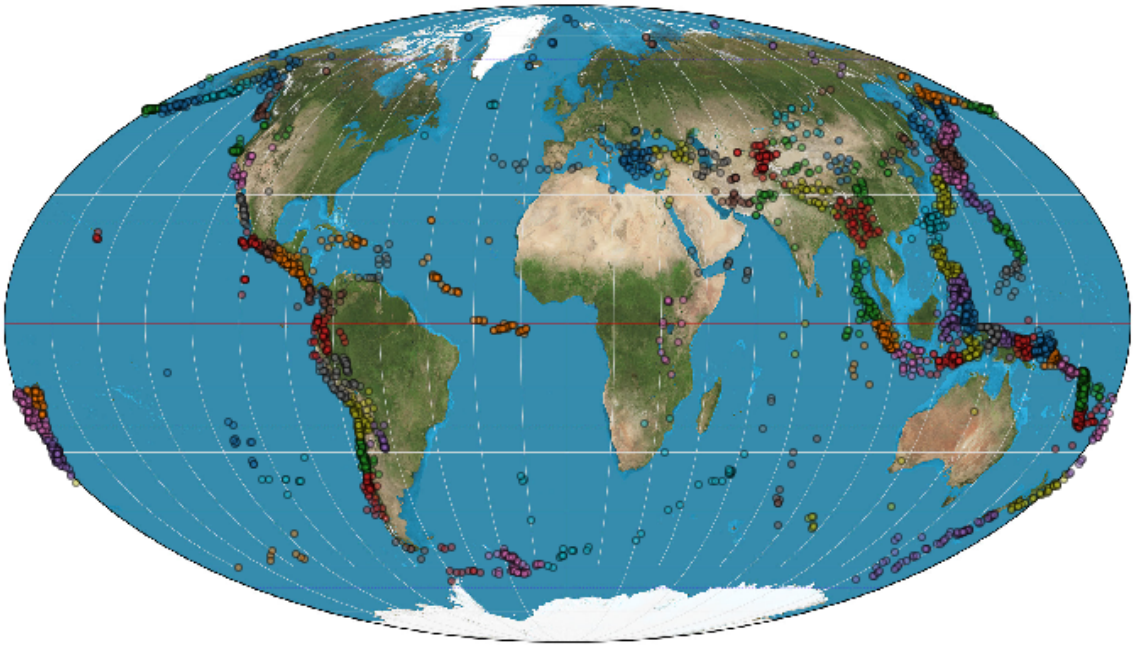


Fig. 20: Mapa do resultado da clusterização do K-Means com k=94

Por outro lado, se considerarmos o índice externo *Rand* verificamos que este tem um valor extremamente elevado quando o k assume um valor considerável, no entanto, não julgamos que este seja um indicador ideal para a escolha do parâmetro k. Esta conclusão provém de uma análise um pouco mais detalhada dos resultados, nomeadamente o número de *True Positives* e o número de *True Negatives*. Como forma de completar a nossa observação, foi também utilizado o índice *Adjusted Rand Index*, bem como o *clustering* original, ou seja, aquele dado pela informação acerca das falhas:

Índice	*Rand Index* ($k = 40$)	*Adjusted Rand Index* ($k = 21$)	*Clustering Original*
TP	182992	264917	756911
TN	6659152	6548609	6772229

Tabela 4: Comparação dos TPs e TNs para o Rand, Adjusted Rand e os valores originais

Como podemos constatar, o melhor resultado do *Rand Index* não se traduz numa atribuição correta dos pontos pelos *clusters*, sendo que o valor é grandemente influenciado pelo número de *TN*, que tem tendência a ser muito elevado quando o número de *clusters* é também considerável, ou seja, vão ocorrer muito mais situações em que os exemplos são de falhas distintas e estão agrupados em *clusters* diferentes. No entanto, isso não se traduz num *clustering* correto. Pelos motivos anteriormente referidos, consideramos o *Adjusted Rand*

Index uma métrica mais fidedigna, pois apesar de ter um valor absoluto inferior, dá uma maior peso à relação entre elementos do que à relação entre um exemplo e a sua classe, resultando em um clustering mais correto.

Devido à análise efetuada, o índice maximizado para a escolha do parâmetro k foi o *Adjusted Rand Index* que atribui o valor $k = 21$ e que resulta no *clustering* apresentado na Fig. 21.

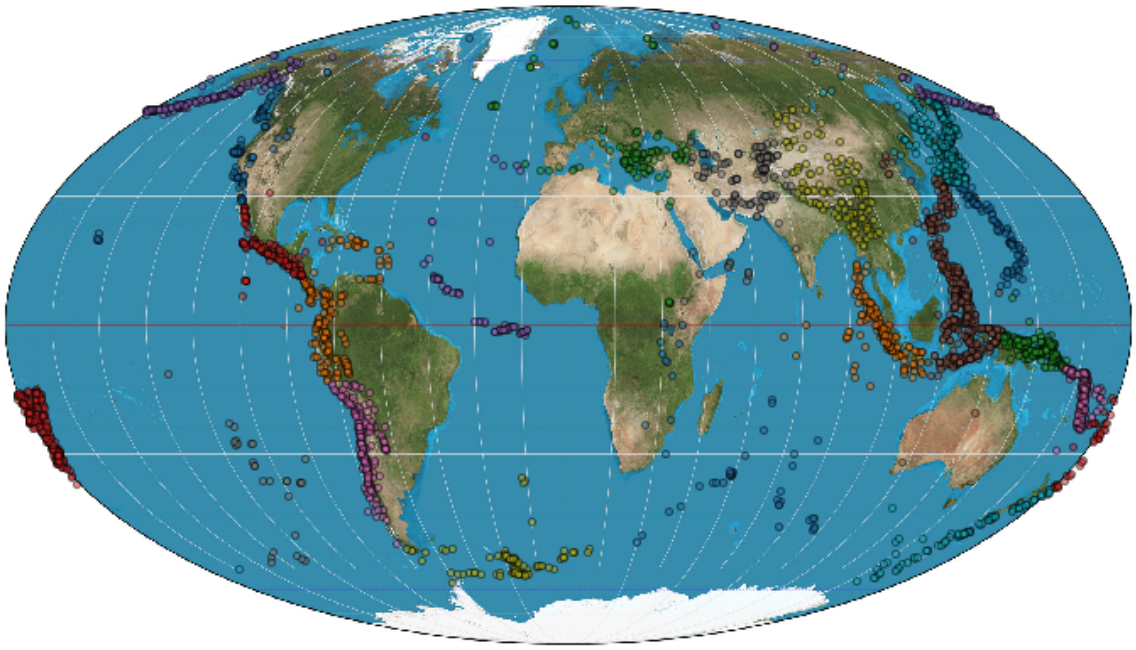


Fig. 21: Mapa do resultado da clusterização do K-Means com k=21

5.2. DBSCAN

Conforme as características teóricas anteriormente apresentadas para o algoritmo *DBSCAN*, é possível fazer algumas deduções do comportamento esperado, mesmo antes de efetuar testes práticos. O *DBSCAN* é parcial e baseado na densidade dos pontos, logo é resistente ao ruído nos dados e ao contrário do *K-Means*, não fica restringido à criação de *clusters* globulares, podendo assim lidar com clusters de várias formas e tamanhos. Tendo em conta a distribuição do *data set,* conceptualmente, o *DBSCAN* consegue encontrar *clusters* que o *K-Means* não tem capacidade de reconhecer. No entanto, devido à sensibilidade do *DBSCAN* à variação da densidade dos exemplos, o algoritmo vai ainda assim encontrar dificuldade se os pontos que deveriam ser agrupados em *clusters* distintos não tiverem uma forma definida e independente, ou seja, se dois clusters foram contínuos no espaço, então, muito provavelmente, vão ser considerados como o mesmo grupo.

Apesar de, no capítulo 2.2, ter sido referido o processo de cálculo dos supostos melhores valores para o parâmetro ε, tendo em conta a pesquisa

realizada no *paper* dos autores do *DBSCAN*, acreditamos que à semelhança do que foi realizado com o algoritmo de *clustering K-Means* é necessário também testar múltiplos valores de forma a correlacionar e verificar se os valores obtidos anteriormente realmente se apresentam como ideais para o problema em mãos.

Desta forma, após calculado o *k-dist graph* é possível encontrar o valor mínimo e o valor máximo da distância ao 4º vizinho de cada ponto, sendo aproximadamente 10 e 2300 respetivamente (não faria sentido valores menores a 10 pois tudo seria considerado ruído e superiores a 2300 pois tudo seria agrupado num só *cluster*). Tendo estes valores já é possível estabelecer um intervalo inicial para a variação do ε, resultando no seguinte gráfico de *performance* tendo em conta cada um dos índices supra-referidos.

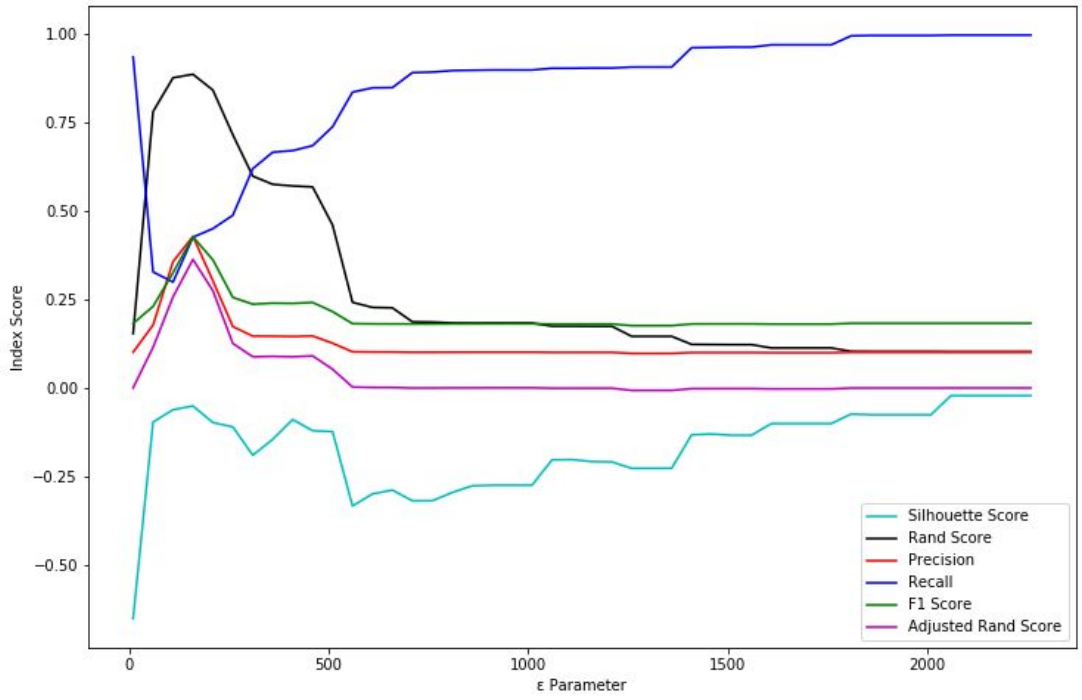
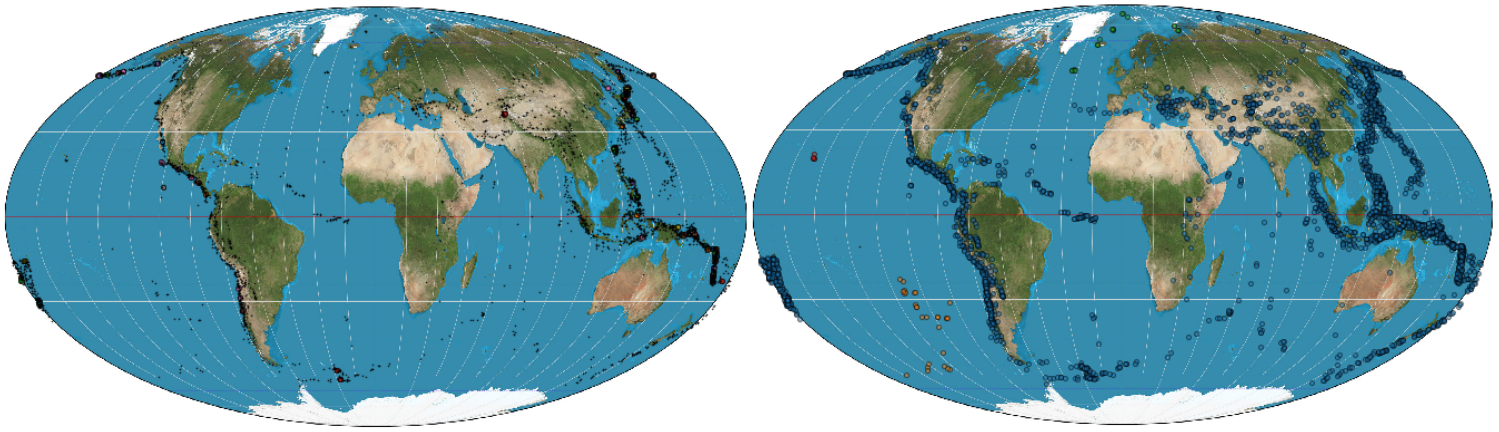


Fig. 22: Variação dos valores dos índices, com o DBSCAN, variando o ε de 10 a 2300

Logo à partida é possível tirar algumas conclusões relativamente aos valores dos índices em relação ao resultado obtido pelo algoritmo, nomeadamente o *recall*. Se considerarmos o valor do *recall* para os valores mínimos e máximos do gráfico para o parâmetro ε, verificamos que o *score* do índice é extremamente elevado e semelhante em ambos os lados do espectro. Conceptualmente, o resultado do algoritmo para um ε mínimo e máximo é altamente díspar: no primeiro caso, irá criar *clusters* apenas para as zonas de alta densidade e em que os pontos estão extremamente próximos uns dos outros; no segundo caso, o resultado esperado será uma pequena quantidade de *clusters*, possivelmente apenas um, que irá agrupar todos os pontos pois o raio de vizinhança é tão grande que é

extremamente mais fácil encontrar uma maior quantidade de pontos para um mesmo cluster. No entanto, em termos práticos, os resultados são semelhantes em qualquer um dos casos devido ao facto da grande maioria dos pontos pertencer ao mesmo cluster, pois apesar do algoritmo separar logicamente pontos clusterizados e pontos *noise*, este também atribui *labels* aos pontos ruidosos, nomeadamente o valor *-1,* o que, efetivamente, permite a representação de um *cluster*.

Assim, quando o ε é reduzido o número de *true positives* tende ser máximo e à medida que este parâmetro aumenta os *false negatives* tendem a ser 0, resultando num *score* quase perfeito para o *recall* apesar dos *clusters* gerados serem errados para o contexto do problema (como é possível verificar nas Figs. 23 e 24).



Figs. 23 e 24: Mapas do resultado da clusterização do DBSCAN com ε=20 e ε=1500, respetivamente

De forma semelhante ao que ocorreu aquando da análise do algoritmo *K-Means*, observando o gráfico anterior é possível constatar que, aproximadamente, a partir do valor de $\varepsilon = 700$, os resultados dos índices tendem a estabilizar, deixando de ser relevantes para conclusões mais aprofundadas, para além de que os *clusters* formados pelo algoritmo deixam de ser corretos para o contexto do problema. Por um motivo semelhante, também foram descartados os valores muito baixos de ε (nomeadamente $\varepsilon < 50$) pois tendo em conta o contexto e *data set* do problema, bem como do comportamento do algoritmo, o *clustering* para um ε muito baixo vai ser extremamente desajustado e podemos constatar isto mesmo verificando o aumento constante dos valores de todos os *scores* ao longo do aumento inicial do parâmetro.

Desta forma, tendo em conta o que foi descrito, apresentamos na Fig. 25 os resultados para o sub-intervalo escolhido.

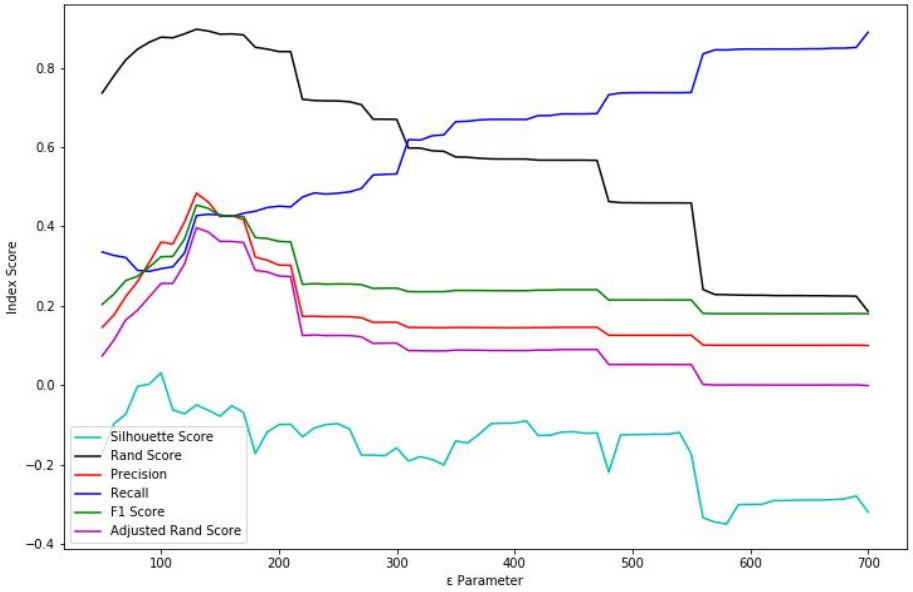


Fig. 25: Variação dos valores dos índices, com o DBSCAN, variando o ε de 50 a 700

Silhouette Score (ε = 100)	*Rand Index* (ε = 130)	*Precision* (ε = 130)	*Recall* (ε = 700)	*F1* (ε = 130)	*Adjusted Rand Index* (ε = 130)
0.0319	0.8966	0.4839	0.8890	0.4538	0.3969

Tabela 5: Os melhores valores para cada índice, com o DBSCAN, variando o ε de 50 a 700

Como já foi referido várias vezes ao longo deste relatório, o *Silhouette Score* tem resultados tão melhores quanto mais coesos, globulares e isolados estiverem os *clusters* em questão. Por isso mesmo, o *DBSCAN* é o algoritmo que apresenta um pior valor neste índice, chegando inclusivé a ser negativo na quase totalidade da execução. Isto ocorre pois os *clusters* por ele formados, em especial os pontos considerados ruído, vão apresentar uma forma mais achatada. Por este motivo o *Silhouette Score* não é um bom indicador para este algoritmo.

O comportamento anómalo referido no início desta secção referente ao índice *recall* não afeta apenas este, apresentando consequências mais abrangentes: pares de pontos que sejam marcados como *noise* pelo algoritmo e que se encontrem altamente distanciados geograficamente uns dos outros (em falhas diferentes e que estas não sejam vizinhas) serão considerados como *false positives*, algo que naturalmente nunca aconteceria (Fig. 26).

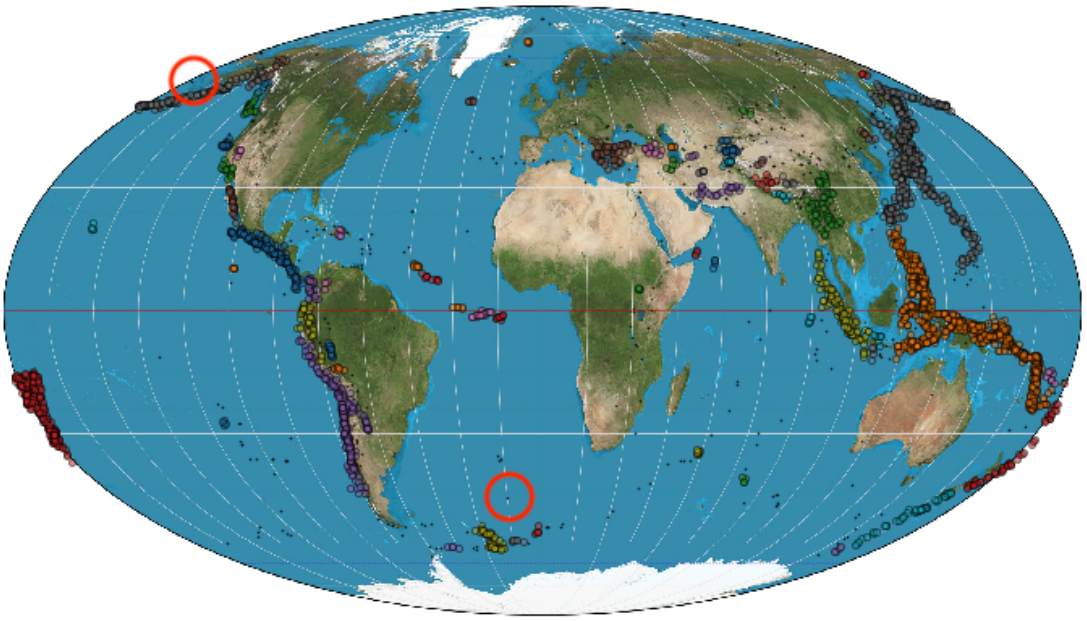


Fig. 26: Exemplos de pontos que são agora considerados como noise e que, apesar da sua separação física, serão considerados como false positives

Assim sendo, tendo em conta que o número de *TP, FN e FP* obtidos pode apresentar uma imagem incorreta da realidade devido ao facto do *DBSCAN* ser um algoritmo de *clusterização* parcial, é possível concluir que qualquer índice que utilize estes valores como base da sua veracidade (nomeadamente o *Rand*, a *Precision*, o *Recall* e, como consequência destes últimos, o *F1 Measure*) poderá ter um valor tendencioso, levando a uma análise incorreta. No entanto, atendendo à definição do *Adjusted Rand*, este índice tenderá a devolver resultados mais próximos da realidade pois irá penalizar os pontos que ficaram no mesmo *cluster* ao acaso, ou seja, dará menos importação à pseudo-aglomeração criada quando a *label* calculada é -1. Assim sendo, consideramos que o *Adjusted Rand Index* seja o melhor índice para o problema descrito na introdução, que atribui um valor de $\varepsilon = 130$ e que resulta no *clustering* da Fig. 27. Desta forma podemos concluir que este valor encontra-se dentro das expetativas inicialmente propostas no capítulo da implementação do *DBSCAN*, na medida em que $\varepsilon = 130$ pertence ao intervalo proposto para o *elbow* e o seu valor do *Adjusted Rand Index* é semelhante ao do algoritmo quando o ε é igual a 148 (valor obtido pelo 2º método), nomeadamente $\simeq 0.37$, significando que as sugestões dos autores do *DBSCAN* para descobrir e otimizar o parâmetro ε apresentam-se como válidas para o contexto do problema.

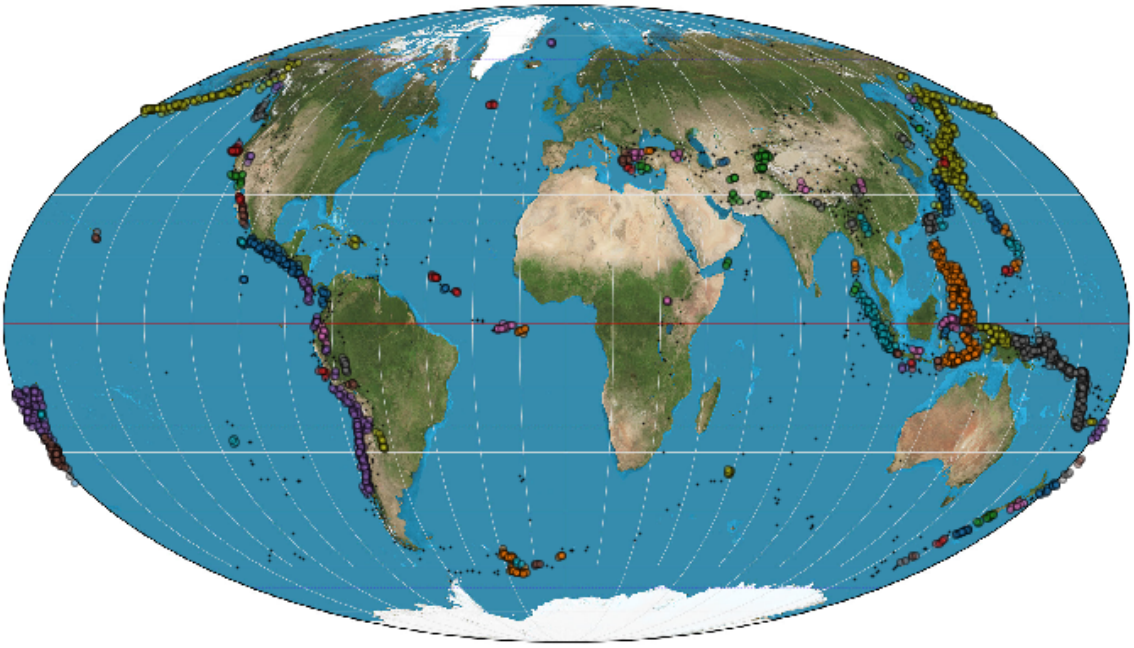
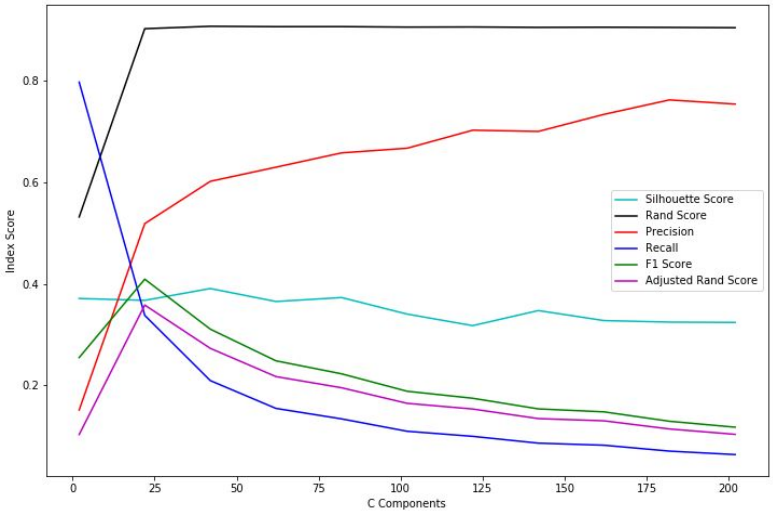
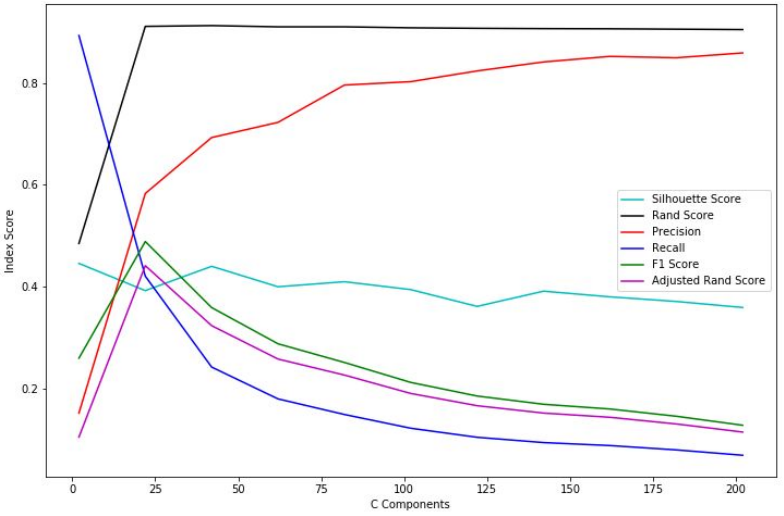


Fig. 27: Mapa do resultado da clusterização do DBSCAN com $\varepsilon = 130$

5.3. Gaussian Mixture Models

Conforme as características teóricas anteriormente apresentadas para o algoritmo *GMM*, é possível fazer algumas deduções do comportamento esperado, mesmo antes de efetuar testes práticos. Nomeadamente, o *GMM* permite acomodar clusters que têm diferentes tamanhos e estruturas de correlação dentro deles. Graças a esta característica, o algoritmo é bastante flexível, ao contrário de, por exemplo, o algoritmo *K-Means* que é um caso de aplicação específico do *GMM* em que a covariância de um *cluster* se aproxima de 0, resultando em *clusters* com uma forma esférica.





Figs. 28 e 29: Variação dos valores dos índices, com o GMM, quando se mantém os pontos com falhas originais a -1 e quando se retiram estes, respetivamente

Como é possível observar nos gráficos acima, existe uma ligeira melhoria geral nos resultados dos índices quando são retirados do *data set* os pontos que não estão atribuídos a nenhuma falha (cujo valor é -1). Isto ocorre por um motivo semelhante ao que se observava com o algoritmo *K-Means*: a *Euclidean Distance* é extremamente sensível a *outliers* no *data set* e, como o *sklearn.mixture.GMM* utiliza a *Malahanobis Distance* (uma variação da *euclidean*), os parâmetros obtidos pelo *GMM* poderão ser tendenciosos, o que influenciará o formato das gaussianas obtidas e, como consequência disso, a atribuição dos pontos aos *clusters* ficará deteriorada [6].

À semelhança do que ocorreu nos outros dois algoritmos, também relativamente ao *GMM* foi tomada a decisão de analisar parte do gráfico com maior detalhe deixando algumas zonas da escala, nomeadamente, a partir do valor de $c = 100$, os resultados dos índices tendem a estabilizar e portanto deixam de ser relevantes para conclusões mais aprofundadas. Por um motivo semelhante, também foram descartados os valores muito baixos de c (nomeadamente $c < 20$) pois tendo em conta o contexto e *data set* do problema, bem como do comportamento do algoritmo, os resultados para um c muito baixo seriam inadequados.

Assim, apresentamos a Fig. 30 com a gama de valores já adaptada de forma a que os resultados possam ser analisados com maior detalhe.

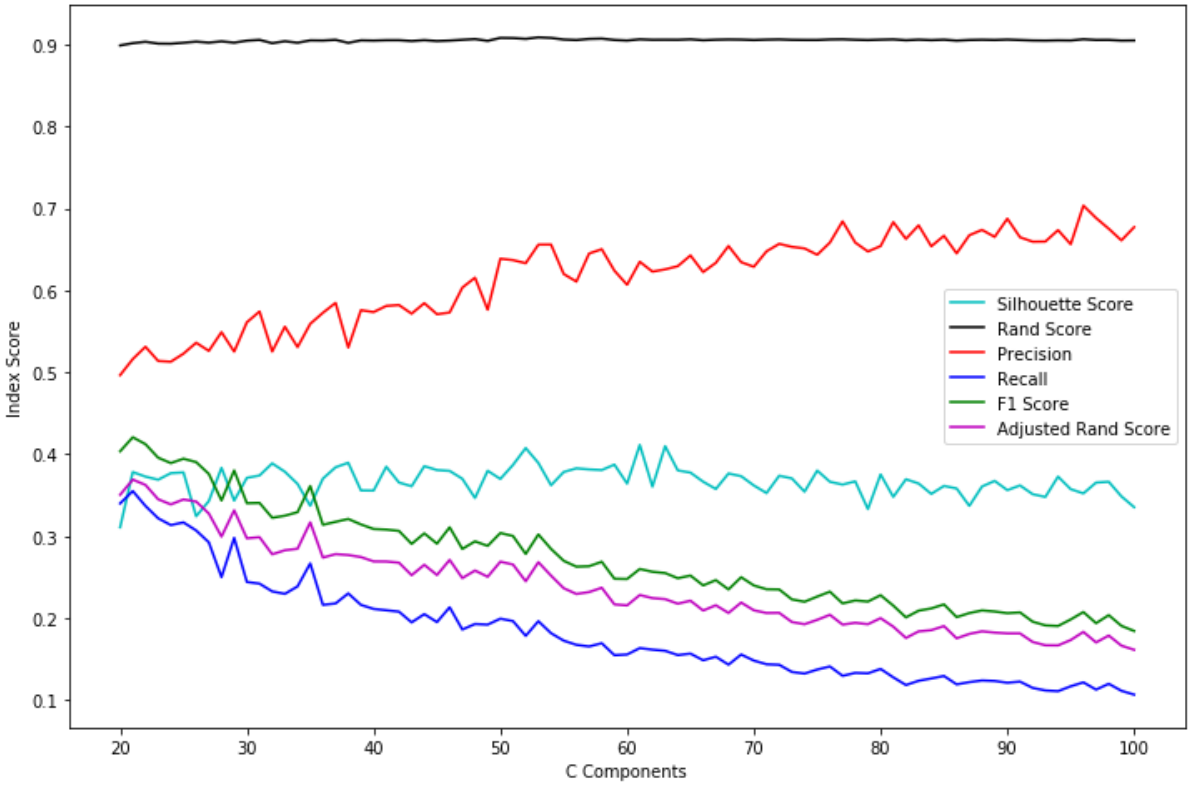


Fig. 30: Variação dos valores dos índices, com o GMM, variando o c de 20 a 100

Silhouette Score ($c = 61$)	*Rand Index* ($c = 53$)	*Precision* ($c = 96$)	*Recall* ($c = 21$)	*F1* ($c = 21$)	*Adjusted Rand Index* ($c = 21$)
0.4115	0.9089	0.7036	0.3552	0.4208	0.3691

Tabela 6: Os melhores valores para cada índice, com o GMM, variando o c de 20 a 100

À semelhança do algoritmo *K-Means*, o índice interno *Silhouette Score* mantém um valor mais ou menos constante e não será a métrica indicada para este algoritmo (pelos mesmos motivos já anteriormente explanados).

Olhando agora para os resultados obtidos com os valores dos índices externos, podemos verificar que a precisão tem um valor bastante elevado para um valor de c também ele elevado. Este facto está de acordo com o comportamento do algoritmo visto que o c representa o número de componentes que, por sua vez, tem um impacto direto no número de *clusters* gerados. Logo quanto maior o valor de c, menor o número de pontos por *cluster* formado e, por isso, maior o número de exemplos comparados que pertencem de facto à mesma falha dentro desse *cluster*. Embora quando utilizamos um grande número de *clusters* estes tenderão a não ser os pretendidos para o objectivo específico de agrupar os pontos de forma semelhante à informação das falhas.

Por outro lado, se considerarmos o índice externo *Rand* verificamos que este tem um valor extremamente elevado quando o c assume um valor considerável, no

entanto, não julgamos que este seja um indicador ideal para a escolha do parâmetro c, visto que este índice é influenciado pelo número de *True Negatives* existentes, que vão ser em grande número quando há muitos *clusters*.

Devido à análise efetuada, o índice maximizado para a escolha do parâmetro c foi o *Adjusted Rand Index* que atribui o valor $c = 21$ e que resulta no *clustering* apresentado na Fig. 31.

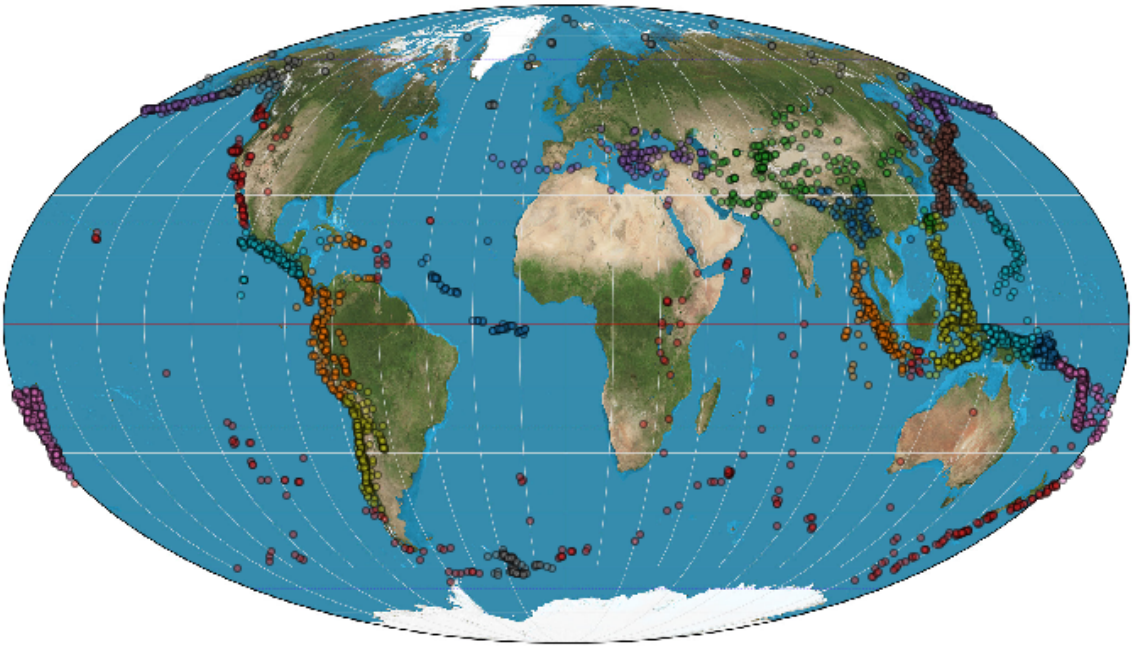


Fig. 31: Mapa do resultado da clusterização do GMM com c=21

O *GMM* é um algoritmo que depende de diversas variáveis e, apesar de neste relatório apenas ter sido explorado o parâmetro c (número de componentes), seria interessante para o contexto do problema testar e analisar de que forma os outros parâmetros influenciam a execução do algoritmo, em especial o tipo de covariância estabelecida pelo *GMM*. Este teste não foi executado por uma simples questão de falta de tempo para uma análise correta e aprofundada do assunto em questão.

6. Conclusão

Neste tipo de problemas de aprendizagem não supervisionada, pelo menos se considerarmos um contexto real com dados reais, é muito difícil ou mesmo impossível de chegar a uma solução única e geral que possa ser considerada a melhor. Como tem sido referido ao longo do documento, quando aplicamos este tipo de algoritmos e avaliamos o seu desempenho, devemos ter sempre em mente

qual o objectivo com que estamos a utilizar estes métodos, assim como o tipo de dados que estão a ser utilizados e o seu contexto na situação.

Posto isto, numa situação real, todos os resultados anteriormente explanados deveriam ser debatidos com especialistas na área, neste caso da Geologia e da Sismologia, de forma a retirar as melhores conclusões. Como não temos essa oportunidade vamos apenas descrever algumas situações que seriam interessantes no contexto deste problema e qual seria a melhor solução para cada uma delas.

Na secção anterior, aquando da escolha do índice mais apropriado para cada algoritmo e do valor do parâmetro resultante da maximização desse índice, foi considerado que objetivo principal dos algoritmos seria obter um resultado o mais semelhante possível àquele conseguido através da informação externa das falhas que nos é fornecido. Se apenas considerarmos esse cenário, o algoritmo que devolve um resultado, tanto mais realista observando os mapas resultantes como aquele que apresenta um maior valor de *Adjusted Rand Index*, é o *DBSCAN*, utilizando a escolha de parâmetro descrita na secção anterior. No entanto, acreditamos que este não é o objetivo mais interessante a considerar no contexto deste problema, tendo em conta que esse agrupamento dos sismos pelas falhas já foi efetuado pelos especialistas na área (por esse mesmo motivo temos acesso a essa informação para validar os nossos resultados).

Durante a avaliação algoritmo *K-Means* foi avaliado, foi várias vezes referido que devido ao aspeto do *data set*, este não seria uma boa escolha, de forma geral, para agrupar os exemplos, não obstante, se o objetivo não for necessariamente agrupar todos os pontos e sim encontrar pontos que funcionem como representantes de outros pontos (de forma a que a amostra de dados não seja tão numerosa/volumosa), poderíamos considerar o *K-Means* como uma boa alternativa, pois facilitaria o processo de escolha desses representantes, visto que este algoritmo é baseado em protótipos. Para tal poderia ser utilizado a métrica da precisão, para garantir que os *clusters* formados não seriam constituídos por pontos mal classificados. Ao maximizar este índice, o valor de k tenderia a ser mais elevado (este facto foi explicado na secção anterior), e por esse motivo é necessário ter em atenção um limite máximo que k pode tomar pois, se isso não for tomado em consideração, o seu valor tenderá a ser o muito próximo do número de pontos no data set o que resultará numa situação semelhante ao *overfitting*.

Considerando agora que pretendemos agrupar os dados de forma a impedir que falhas contínuas sejam agrupadas como uma única, o algoritmo que devolveria os melhores resultados seria o *GMM* pela sua flexibilidade. Por outro lado, se for pretendido que múltiplas falhas contínuas sejam agrupadas numa só, por exemplo para juntar sismos que apesar de pertencerem a falhas distintas, afetam o mesmo continente, o algoritmo mais adequado seria o *DBSCAN*. Para ambos os casos o índice externo escolhido para encontrar o valor dos parâmetro é o *Adjusted Rand Index*, pelos motivos já anteriormente explicados.

Uma outra situação seria ter como objetivo localizar as zonas do globo com maior incidência sísmica, independemente da falha que esteja associada a esses eventos. Para tal, o algoritmo mais indicado seria o *DBSCAN* pois é um algoritmo baseado na densidade dos pontos (que representa o número de eventos) e tendo em conta que apenas as zonas mais densas teriam interesse, o ε escolhido teria um valor pequeno.

Mais uma vez, é importante referir que embora este projecto pretenda simular um pouco mais do que seria a experiência de um problema de aprendizagem não supervisionada real, o conhecimento do contexto do problema e dos seus dados é simulado ao serem apresentadas informações acerca das falhas, que permitem validar de forma muito mais simples os resultados obtidos. Ainda assim, no contexto realista, existe uma grande probabilidade de não existir informação do problema em causa, ou seja, pode acontecer não existirem *labels* reais com as quais possamos comparar os resultados dos algoritmos. Numa situação deste género não poderíamos basear a nossa avaliação nos resultados de índices externos, e os resultados obtidos teriam a sua origem em índices internos.

7. Bibliografia

[1]http://scikit-learn.org/stable/modules/clustering.html
[2]http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.121.9220
[3]http://library.iugaza.edu.ps/thesis/106515.pdf
[4]https://pdfs.semanticscholar.org/52d4/8b393f3f838f2370c50af03703eee0bbd669.pdf
[5]https://mitpress.mit.edu/books/machine-learning-0
[6]https://arxiv.org/pdf/1104.4512.pdf