Clustering de Dados Biomédicos com Algoritmos baseados em Critérios Entrópicos

Frederico Mendes de Morais

Dissertação submetida ao Instituto Superior de Engenharia do Porto para a obtenção de grau de Mestre em Engenharia de Computação e Instrumentação Médica

Orientador no ISEP
Doutor Jorge Manuel Fernandes dos Santos
Departamento de Matemática

Porto, 11 de Novembro de 2012

Agradecimentos

Em primeiro lugar, agradeço ao Professor Jorge Santos pela oportunidade que me proporcionou de realizar este trabalho. Agradeço-lhe pela orientação, ajuda e o tempo dedicado, sem os quais não seria possível finalizar esta dissertação.

Agradeço também à Ana pela paciência e pelas palavras de conforto, que tornaram a conclusão desta fase numa tarefa menos árdua.

Por último, agradeço aos meus pais pela dedicação, motivação e o apoio que me deram desde o início da minha vida académica, não deixando perder o rumo até à meta final. Sem eles não teria a possibilidade de investir na minha formação.

Resumo

A procura de padrões nos dados de modo a formar grupos é conhecida como aglomeração de dados ou *clustering*, sendo uma das tarefas mais realizadas em mineração de dados e reconhecimento de padrões. Nesta dissertação é abordado o conceito de entropia e são usados algoritmos com critérios entrópicos para fazer *clustering* em dados biomédicos. O uso da entropia para efetuar *clustering* é relativamente recente e surge numa tentativa da utilização da capacidade que a entropia possui de extrair da distribuição dos dados informação de ordem superior, para usá-la como o critério na formação de grupos (*clusters*) ou então para complementar/melhorar algoritmos existentes, numa busca de obtenção de melhores resultados. Alguns trabalhos envolvendo o uso de algoritmos baseados em critérios entrópicos demonstraram resultados positivos na análise de dados reais. Neste trabalho, exploraram-se alguns algoritmos baseados em critérios entrópicos e a sua aplicabilidade a dados biomédicos, numa tentativa de avaliar a adequação destes algoritmos a este tipo de dados. Os resultados dos algoritmos testados são comparados com os obtidos por outros algoritmos mais "convencionais" como o k-médias, os algoritmos de *spectral clustering* e um algoritmo baseado em densidade.

Abstract

The search for patterns in data in order to find groups is known as clustering and is one of the most important tasks in pattern recognition and data mining. In this work, we explore entropy and the usefulness of some entropic algorithms to cluster biomedical data. The use of entropy in clustering is recent and comes as an attempt to take advantage from its capability of extracting higher order information from the data and use it as a criterion to form clusters or to improve/complement already existing algorithms, as a way of achieving better results. Some work with entropic algorithms have shown positive results on clustering real data. In this work, we have explored some entropic algorithms and its aplication to biomedical datasets, to evaluate if this algorithms are suitable to this kind of data. The results from these algorithms are compared with those obtained with more conventional ones, such as k-means, spectral clustering and density based clustering.

Conteúdo

Agradecimentos iv

Resumo vi

Abstract viii

Conteúdo x

Lista de Figuras xi

Lista de Tabelas xiii

1. Introdução 1
1.1 Objetivos do Trabalho 2
1.2 Organização e Conteúdo 2

2. Enquadramento e Definições 5
2.1 Reconhecimento de Padrões e Mineração de Dados 5
2.2 Aglomeração de dados - Clustering 7
2.2.1 Algoritmos de Clustering 8
2.3 Clustering Entrópico 11
2.3.1 Entropia e Teoria da Informação 11
2.3.2 Algoritmos com Critérios Entrópicos 13

3. Algoritmos de Clustering 15
3.1 Conjuntos de dados 15
3.1.1 Dados de Testes 15
3.1.2 Dados Biomédicos 16
3.2 Minimum Entropy Clustering 18
3.2.1 Modelo de Entropia Mínima 19
3.2.2 Algoritmo Iterativo 20
3.2.3 Implementação em MATLAB 21
3.3 LEGClust 22

3.3.1 Matriz de proximidade entrópica . . . 23
3.3.2 Formação de clusters com o algoritmo hierárquico . . . 24
3.3.3 Implementação em MATLAB . . . 25
3.4 minCentropy . . . 26
3.4.1 Definição da função objetivo . . . 26
3.4.2 Implementação do algoritmo . . . 27
3.5 Spectral Clustering . . . 29
3.5.1 Shi & Malik - Normalized Cut . . . 29
3.5.2 Ng & Jordan - K-Eigenvectors . . . 30
3.5.3 Perona & Freeman - Affinity Factorization . . . 31
3.6 K-Médias . . . 31
3.7 Mean Shift . . . 32
3.8 Adjusted Rand Index . . . 33

4. Clustering de Dados Biomédicos . . . 35
4.1 Testes Exploratórios . . . 35
4.1.1 Testes com dados artificiais bidimensionais . . . 35
4.1.2 Testes com dados reais . . . 44
4.2 Aplicação a dados Biomédicos . . . 46

5. Conclusão . . . 51

Bibliografia . . . 54

Lista de Figuras

2.1 Paradigmas da análise de dados. 6

3.1 Conjuntos de dados artificiais bidimensionais. 17

4.1 Resultados obtidos com Minimum Entropy Clustering nos conjuntos Kites e Citroen com dois clusters. 36
4.2 Resultados obtidos com Minimum Entropy Clustering nos conjuntos Anthills e Eye com três clusters. 37
4.3 Resultados obtidos com Minimum Entropy Clustering o conjunto Starfish com cinco clusters. 38
4.4 Resultados obtidos com os conjuntos Three e Kites usando o LEGClust. 39
4.5 Resultados obtidos com os conjuntos Epiglottis e Eye usando o LEGClust. 40
4.6 Resultados obtidos com o conjunto Citroen usando o LEGClust. . . . 41
4.7 Resultados obtidos com o conjunto Anchor usando o LEGClust. . . . 42
4.8 Resultados obtidos com o conjunto Cross usando o LEGClust. 43
4.9 Resultados obtidos com o conjunto Starfish usando o LEGClust. . . . 44
4.10 Resultados obtidos com o conjunto Anthills usando o LEGClust. . . . 44

Lista de Tabelas

3.1 Conjunto de dados reais usados para o teste dos algoritmos. 16
3.2 Conjuntos de dados biomédicos 18
3.3 Pseudo-código do algoritmo Minimum Entropy Clustering. 21
3.4 Exemplo de uma matriz de dissimilaridade entrópica 24
3.5 Exemplo de uma matriz de proximidade entrópica 24
3.6 Pseudo-código do algoritmo LEGClust 25
3.7 Pseudo-código do algoritmo minCentropy 28
3.8 Pseudo-código do algoritmo K-means 32
3.9 Tabela de contingência relacionando duas partições 33

4.1 Resultados exploratórios obtidos com os conjuntos de dados reais. . . 45
4.2 Resultados (índice ARI) da aplicação dos algoritmos baseados em critérios entrópicos e do algoritmo k-médias em dados biomédicos. . . 46
4.3 Resultados (índice ARI) da aplicação dos algoritmos de spectral clustering e do algoritmo mean shift em dados biomédicos. 47
4.4 Resultados (índice ARI) da aplicação das diferentes implementações em MATLAB e JAVA do algoritmo Minimum Entropy Clustering e do algoritmo k-médias em dados biomédicos. 48
4.5 Resultados (índice ARI) da aplicação dos diferentes algoritmos usados em dados biomédicos. 50

Capítulo 1

Introdução

A interpretação de dados e a descoberta de nova informação através dos mesmos, conduz muitas vezes ao encontro de uma solução para problemas da atualidade. A aglomeração de dados, conhecida como *clustering*, é uma das tarefas mais comuns na área de análise de dados. Incluída na área de reconhecimento de padrões e também mineração de dados, a aglomeração consiste no agrupamento dos dados, criando grupos que partilham caraterísticas idênticas. Os dados não possuem qualquer identificação, não sendo conhecidas as classes dos objetos, o que faz do *clustering* uma operação de aprendizagem não supervisionada.

Existem várias técnicas de *clustering* para encontrar grupos de objetos idênticos nos dados. As técnicas mais comuns envolvem a medida de distâncias entre os dados. O algoritmo k-médias é o exemplo mais comum do uso de distâncias para encontrar grupos (*clusters*) nos dados. Outros algoritmos fazem uso de diferentes técnicas. O modelo de *clustering* entrópico é baseado na medida da entropia que quantifica a incerteza dos dados. Para obter os *clusters*, estes algoritmos usam a entropia na sua função objetivo, também referida como critério de *clustering*, em conjunto com outras técnicas como algoritmos hierárquicos.

O método de *clustering* hierárquico é bastante usado em áreas como a biologia e a medicina. O *spectral clustering*, que combina grafos com vetores próprios é muito usado na segmentação de imagens. Outros algoritmos encontram os *clusters* com base na densidade dos dados. O algoritmo k-médias e os algoritmos hierárquicos, são muito frequentemente utilizados, devido à sua facilidade de uso, implementação simples e resultados aceitáveis.

Os dados biomédicos têm características particulares. Muitas vezes apresentam-se incompletos e com erros. Nem todas as técnicas de *clustering* são adequadas para

dados desta natureza. Estes dados podem não possuir agrupamentos naturais entre os objetos, dificultando a pesquisa por *clusters*.

1.1 Objetivos do Trabalho

O objetivo principal deste trabalho é testar em dados biomédicos alguns algoritmos de *clustering* baseados em entropia. São escolhidos algoritmos que usam o modelo de entropia para criar conjuntos de objetos nos dados. Os dados biomédicos são extraídos do domínio público e têm de ser pré-processados antes de poderem ser usados nos algoritmos. É necessário fazer uma seleção das características a analisar e corrigir possíveis erros ou campos em falta nos dados.

Posteriormente os dados são introduzidos nos algoritmos que os irão processar. Não existe um resultado esperado, sendo que os resultados obtidos, podem não ser os mais adequados. Algoritmos mais convencionais, como o k-médias, são também usados por forma a comparar os resultados obtidos com os outros algoritmos. No final os resultados de todos os algoritmos são avaliados através de um índice de performance, sendo assim possível avaliar o método que obtém melhores resultados. Os algoritmos serão avaliados tendo em conta os resultados obtidos, a quantidade de interação necessária e a capacidade computacional requerida.

1.2 Organização e Conteúdo

Esta dissertação encontra-se organizada por capítulos. Neste capítulo fez-se uma pequena introdução ao tema, descrevendo de modo generalizado o âmbito de todo o trabalho e os seus objetivos.

No capitulo 2 são apresentados os conceitos teóricos relevantes para uma melhor compreensão do trabalho realizado, nomeadamente o enquadramento na área de desenvolvimento e a definição de *clustering*, sendo mencionadas as técnicas mais proeminentes e dados alguns exemplos. É ainda definido o conceito de entropia, referindo os principais desenvolvimentos e alguns exemplos de algoritmos com critérios entrópicos.

No capitulo 3 são apresentados os conjuntos de dados biomédicos e os conjuntos de dados com os quais foram efetuados alguns testes à implementação dos algoritmos. São apresentados também todos os algoritmos usados para processar os dados biomédicos e é feita uma descrição individual de cada algoritmo.

No capitulo 4 são apresentados os resultados obtidos com os dados biomédicos e ainda os resultados obtidos nos testes aos algoritmos. É feita também a discussão dos resultados neste capitulo. Por ultimo no capitulo 5 são tiradas algumas conclusões sobre o trabalho desenvolvido.

Capítulo 2

Enquadramento e Definições

A análise de dados é uma tarefa que permite a extração de grande quantidade de informação sobre um determinado conjunto de dados. A organização dos dados por grupos permite um melhor entendimento dos mesmos. A criação de grupos com caraterísticas semelhantes é um processo denominado por aglomeração de dados, mais conhecido pelo termo *clustering*. O *clustering* é uma das principais tarefas utilizadas no reconhecimento de padrões (*pattern recognition*) e em mineração de dados (*data mining*).

2.1 Reconhecimento de Padrões e Mineração de Dados

As áreas de reconhecimento de padrões e mineração de dados partilham entre si grande parte das suas tarefas. No reconhecimento de padrões a análise dos dados é feita com o objetivo de aprendizagem através dos dados. A mineração dos dados tem o objetivo de extrair dos dados informação desejada. Apesar de serem duas áreas com nomes diferentes partilham a mesma premissa de explorar e analisar dados. Os métodos usados nestas duas áreas são muito frequentemente aplicados ao processo de descoberta de conhecimento.

O reconhecimento de padrões pode ser definido com dois objetivos: exploração e confirmação. Na exploração, o objetivo é compreender os dados analisando as suas caraterísticas ou a sua estrutura, não possuindo informação prévia. Na confirmação, o objetivo é confirmar uma hipótese prévia ou um conjunto de fatos baseados nos dados [1]. Na mineração de dados os objetivos são semelhantes: verificação ou confirmação e exploração ou descoberta.

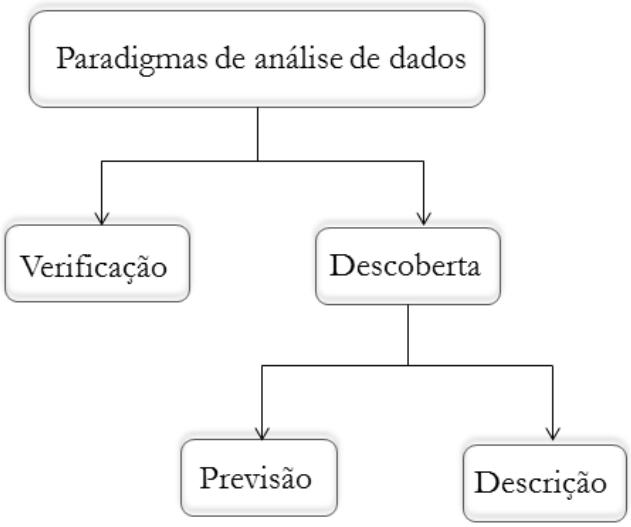


Fig. 2.1: Representação hierárquica das relações entre os diferentes paradigmas da análise de dados.

Na figura 2.1 é possível visualizar graficamente as relações entre os diferentes paradigmas mencionados. A verificação ou confirmação é um paradigma adequado para testar hipóteses e confirmar fatos conhecidos à priori. O paradigma de descoberta ou exploração tem por objetivo encontrar nova informação. A descoberta pode ainda ser dividida em: (*i*) previsão, onde são procurados padrões que permitam prever a evolução dos dados (*ii*) descrição, onde os dados são processados de modo a tornarem-se mais claros, são aplicadas técnicas de *clustering*, sumarização e visualização [2].

O reconhecimento de padrões foca-se na vertente de previsão, usando um conjunto de dados para treinar o modelo de previsão de modo a saber o comportamento dos dados desconhecidos ao modelo (conjunto de teste). As tarefas de previsão são denominadas de aprendizagem. A aprendizagem pode ser supervisionada, envolvendo dados com classes conhecidas, e não supervisionada que envolve dados sem classes. Das tarefas efetuadas, destacam-se a classificação (*classification*), aglomeração (*clustering*), sumarização (*summarization*), regressão (*regression*), modelos de dependência e modelos de deteção de mudanças e desvios. Seguidamente são explicadas de modo genérico estas diferentes tarefas:

- *Classificação*: os dados são classificados segundo os seus atributos por uma função específica. Estes são divididos em dois conjuntos: treino e teste. No conjunto de treino cada objeto dos dados é representado por um vetor com atributos e uma classe. O modelo de classificação é construído através da

análise das relações entre os atributos e a classe de cada objeto no conjunto de dados de treino. Este modelo pode depois classificar outros dados, como por exemplo os do conjunto de teste. Este processo de classificação é também referido como aprendizagem supervisionada (*supervised learning*).

- *Clustering*: os dados são agrupados através da identificação das suas caraterísticas, formando *clusters* ou grupos de objetos, que partilham características entre si. O agrupamento ou aglomeração é feito mediante parâmetros definidos pelo utilizador. Os parâmetros são baseados nos atributos dos dados.

- *Sumarização (summarization)*: consiste no resumo ou na generalização dos dados. Estes são simplificados resultando um conjunto de dados mais pequeno onde a informação mais relevante está agregada.

- *Regressão*: descreve as relações entre variáveis independentes e dependentes. Atribui um valor médio à relação entre uma variável dependente e uma independente, quando todas as outras variáveis independentes possuem valores fixos.

- *Modelos de dependência*: modelos que descrevem as relações significantes entre as variáveis contidas nos dados.

- *Modelos de deteção de mudanças e desvios*: modelos que se focam em encontrar mudanças significantes dos atributos dos dados em relação a valores estipulados.

Estes métodos são efetuados através de algoritmos especializados, como árvores de decisão, redes neuronais, algoritmos genéticos, algoritmos de *clustering*, redes *Bayesianas*, etc.

2.2 Aglomeração de dados - Clustering

O *Clustering* é um método de aprendizagem não supervisionada que consiste na descoberta de grupos (*clusters*) de objetos com características semelhantes, sem se conhecer a classe a que pertencem. O *clustering* é uma ação exploratória com o objetivo de encontrar estruturas nos dados. A ausência do conhecimento prévio das classes dos objetos distingue *clustering* de classificação, denominada de aprendizagem supervisionada, o que torna o *clustering* numa tarefa mais complicada [3]. São

procurados padrões nos dados que distingam os objetos, aqueles que apresentam padrões semelhantes são agrupados no mesmo *cluster*. Intuitivamente, objetos do mesmo *cluster* são mais semelhantes entre si do que objetos de *clusters* diferentes. *Clustering* pode ser definido como: dado um conjunto de dados com n objetos, encontrar K grupos com base numa medida de similaridade, de modo a que a similaridade entre objetos do mesmo grupo seja elevada enquanto a similaridade entre objetos de grupos diferentes seja baixa. Os *clusters* podem variar entre si, em termos de tamanho, forma e densidade.

A presença de ruído nos dados dificulta a descoberta de *clusters*. O *cluster* ideal pode ser definido como um grupo de objetos isolado e compacto, mas na realidade um *cluster* é algo subjetivo, dependente da intenção do utilizador. Existe uma grande variedade de técnicas para efetuar *clustering*. A variedade de algoritmos que implementam essas técnicas é ainda maior.

2.2.1 Algoritmos de Clustering

A grande maioria dos algoritmos de *clustering* podem ser agrupados pela função objetivo que implementam ou através da comparação dos *clusters* que detetam [4] . A função objetivo determina o resultado do procedimento de *clustering* dos dados, pelo que, muitos algoritmos fazem a aplicação dessa função aos dados. A função objetivo é também referida como o critério de *clustering*. Em [4] vários algoritmos de *clustering* são usados e é feita uma análise comparativa entre os resultados obtidos pelos diferentes algoritmos. A semelhança entre as funções objetivo dos diferentes algoritmos foi avaliada pela igualdade entre os *clusters* encontrados nos resultados.

De um modo generalizado, os algoritmos de *clustering* podem ser divididos em dois tipos: particionais e hierárquicos. Alguns algoritmos usam estes dois métodos em conjunto com, por exemplo, medidas de densidade, grafos ou medidas de entropia.

Algoritmos Hierárquicos

Os algoritmos hierárquicos criam uma hierarquia de *clusters* nos dados. Os *clusters* podem ser criados através de dois procedimentos distintos: aglomeração ou divisão. Na aglomeração o algoritmo atribui cada objeto a um único *cluster*, juntando os *clusters* de acordo com uma medida de similaridade ou dissimilaridade entre pares de objetos. Deste modo, são gerados vários *clusters* no início, sendo progressivamente fundidos até existir um só *cluster* contendo todos os objetos. Entre os processos mais utilizados para efetuar a fusão dos *clusters* podemos referir os seguintes: (i)

ligação singular (*single-link*) (*ii*) ligação completa (*complete-link*) e (*iii*) método de Ward (*Ward's method*).

No procedimento de divisão ocorre o oposto, inicialmente existe um grande *cluster* que contém todos os objetos e é dividido em *clusters* menores, até cada objeto ser atribuído a um *cluster* individual. Os algoritmos BIRCH [5], AGNES [6], ROCK [7], CURE [8] e Chameleon [9] são alguns exemplos de *clustering* hierárquico.

Algoritmos Particionais

Os algoritmos particionais encontram todos os *clusters* em simultâneo como uma partição dos dados. O algoritmo mais simples que implementa este método é o k-médias. É um algoritmo muito popular e largamente utilizado para *clustering* devido à sua facilidade de implementação, simplicidade, eficiência e boa interatividade. É usado em muitos algoritmos como um método auxiliar para obter uma partição inicial dos dados.

O algoritmo k-médias tem sido expandido e melhorado. Algumas das suas variações incluem a utilização de caraterísticas adicionais como o tamanho mínimo dos *clusters* e a junção e divisão de *clusters*. Os algoritmos ISODATA [10] e FORGY [11] são duas variações do k-médias bastante usadas no reconhecimento de padrões. No algoritmo k-médias cada objeto dos dados é atribuído a um *cluster* único, enquanto que no algoritmo *Fuzzy C-Means* [12], uma variação do k-médias, cada objeto pode pertencer a vários *clusters*. Outro algoritmo que implementa uma variação do k-médias, denominado *Bisecting K-Means* [13], envolve o uso de *clustering* hierárquico de divisão. O algoritmo *X-Means* [14] encontra automaticamente o número de *clusters* K nos dados implementando o critério de informação Baysiano. O algoritmo *K-Medoid* [15] encontra os *clusters* usando a mediana em vez da média usada pelo k-médias. Todas estas variações têm por base o método usado pelo k-médias e introduzem novos parâmetros que necessitam de ser definidos pelo utilizador.

Algoritmos de Densidade

Os *clusters* podem ser definidos no espaço do conjunto de dados como regiões de grande densidade separadas por regiões de baixa densidade. Alguns algoritmos são baseados nesta definição, procurando regiões mais densas conetadas no espaço dos dados para encontrar os *clusters*. Os algoritmos *Jarvis-Patrick* [16], DBSCAN [17] e *Mean Shift* [18] são três exemplos de algoritmos de *clustering* baseados na densidade dos objetos no espaço. Procuram as regiões conectadas com maior densidade,

estimando a densidade pelo método da janela de Parzen.

Algoritmos baseados na Teoria de Grafos

Os dados são representados em forma de grafo, onde cada objeto é um vértice e são feitas ligações entre os vértices avaliadas pela similaridade entre cada par de objetos. Inicialmente existe um só *cluster* dado pelo próprio grafo. Os novos *clusters* são formados através de cortes sucessivos nos. Num grafo, um corte corresponde à remoção de vértices, dividindo o grafo. O corte mínimo é a remoção do menor número de vértices necessário para efetuar um corte. O resultado do corte faz com que o *cluster* inicial seja dividido em dois *clusters*.

O uso de grafos em conjunto com vetores próprios da matriz Laplaciana são a base do algoritmo *spectral clustering*. A matriz Laplaciana é baseada numa matriz de afinidade construída através de uma medida de similaridade. A medida mais usada é a distância Euclidiana entre dois vetores. O corte normalizado usa vetores próprios de uma matriz Laplaciana normalizada para fazer *clustering* usando o segundo vetor próprio mais pequeno. Os vetores próprios podem ser também utilizados em conjunto com outros algoritmos de *clustering*.

Validação de Clusters

Os algoritmos de *clustering* encontram *clusters* nos dados, mesmo que não haja grupos de objetos bem definidos. Um conjunto de dados com vários objetos espalhados aleatoriamente pelo espaço, não formando grupos bem definidos, é um conjunto que, pela sua natureza, não possui uma solução de *clustering* óbvia. Diferentes *clusters* podem ser encontrados, se forem usados diferentes algoritmos ou se, no mesmo algoritmo, forem variados os valores dos parâmetros.

A validação dos *clusters* refere-se a procedimentos que avaliam o resultado de uma forma quantitativa e objetiva. A avaliação é feita através de índices baseados em critérios internos, relativos e externos. Índices baseados em critérios internos avaliam o ajuste entre os *clusters* encontrados e o próprio conjunto de dados. Índices baseados em critérios relativos comparam diferentes resultados de *clustering* gerados por algoritmos diferentes avaliando qual o melhor resultado mediante um certo critério. Índices externos avaliam a performance fazendo corresponder os *clusters* com informação conhecida à priori, nomeadamente, as classes verdadeiras dos dados. O critério externo de validação é normalmente o mais utilizado. Os índices Jaccard [19], *Rand Index* [20] e *Adjusted Rand Index* [21] são exemplos de validação

de *clusters* usando o critério externo.

2.3 Clustering Entrópico

2.3.1 Entropia e Teoria da Informação

O conceito de entropia foi inicialmente aplicado em Termodinâmica, como uma medida de energia mensurável em transformações que envolvem temperaturas. Claude Shannon introduziu o termo entropia em comunicações e transmissão de informação, desenvolvendo a teoria da comunicação, que mais tarde deu origem a uma nova área de investigação denominada teoria da informação.

A informação não tem uma definição universal, sendo bastante subjetiva. Do ponto de vista da comunicação, a informação pode ser definida como uma mensagem constituída por símbolos, por exemplo, uma frase é constituída por letras e espaços, onde a frase é a mensagem e as letras e os espaços são os símbolos que constituem a mensagem. Associada a esta mensagem está uma determinada quantidade de incerteza. Quando os símbolos que constituem a mensagem são consecutivos, isto é, acontecem repetidamente com uma frequência específica, pode ser determinada a probabilidade de ocorrência dos vários símbolos numa certa fase do processo. O acontecimento de novos símbolos está relacionado com símbolos anteriores, assim a probabilidade de um novo acontecimento existir está dependente do acontecimento anterior.

Claude Shannon na teoria da informação, define entropia como a medida de incerteza envolvida num novo acontecimento, ou seja, a incerteza de um novo resultado. Dada uma variável aleatória X tal que $X = \{x_1, x_2, ...x_n\}$ com probabilidades $P = (p_1, p_2...p_n)$, a entropia de Shannon é dada por:

$$H(X) = -\sum_{i=1}^{n} p_i log_2 p_i \quad . \tag{2.1}$$

A entropia de X não depende dos valores de X mas do valor da probabilidade de cada X. Uma nova medida de entropia foi apresentada por Rényi:

$$H_\alpha = \frac{1}{1-\alpha} log \left(\sum_{k=1}^{n} p_k^\alpha \right) \quad . \tag{2.2}$$

Esta medida de entropia é também conhecida como a medida generalizada ou simplesmente como a entropia de Rényi de α-ordem. Outra medida de entropia

diferente foi definida por Havrda e Charvat como:

$$H_\alpha = \frac{1}{1-\alpha} log\left(\sum_{k=1}^{n} p_k^\alpha - 1\right) \quad . \tag{2.3}$$

É muito semelhante à entropia de Rényi, sendo equivalente a Shannon e Rényi na maximização da entropia. Para mais que uma variável aleatória, a entropia de Shannon tem de ser alterada. Introduz-se o conceito de entropia condicional, entropia conjunta e informação mutual. Usando a entropia de Shannon, a entropia conjunta, $H(X,Y)$, é obtida por:

$$H(X,Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i,y_j) log[p(x_i,y_j)] \quad . \tag{2.4}$$

A entropia condicional de uma variável aleatória Y está relacionada com a probabilidade de Y dado que x_i já aconteceu e pode ser calculada por:

$$H(Y|x_i) = -\sum_{j=1}^{m} p(y_i|x_j) log[p(y_i|x_j)] \quad . \tag{2.5}$$

A informação mutua pode ser descrita como uma dependência entre duas variáveis aleatórias X e Y onde uma variável contém informação sobre a outra variável. É definida por

$$I(X,Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i,y_j) log\frac{p(x_i,y_j)}{p(x_i)p(y_j)} \quad . \tag{2.6}$$

A entropia relativa, outro conceito muito importante na teoria da informação, é uma medida de dissimilaridade entre duas distribuições de probabilidade definida por

$$D(p|q) = \sum_x p(x) log\frac{p(x)}{q(x)} \quad , \tag{2.7}$$

onde p e q são distribuições de probabilidade diferentes. A entropia dos dados tem de ser estimada, isto é, as probabilidades $p(x_i)$ ou $p(y_i|x_i)$ necessitam de ser estimadas para se calcular a entropia. Para a estimativa da função densidade de probabilidade podem ser utilizados métodos não paramétricos como a janela de Parzen. Para uma variável de dimensão d a estimativa usando a janela de Parzen é definida por:

$$\hat{p}(x) = \frac{1}{nh^d}\sum_{j=1}^{n} K\left(\frac{x-x_j}{h}\right) \quad , \tag{2.8}$$

onde h é o parâmetro que define a largura da janela de Parzen. A entropia de Shannon é estimada aplicando (2.8) à equação (2.1). O método da janela de Parzen também pode ser aplicado às derivações da entropia de Shannon, como a entropia de Rényi e Havrda-Charvat.

2.3.2 Algoritmos com Critérios Entrópicos

Os algoritmos com critérios entrópicos fazem uso das entropias apresentadas anteriormente como medidas de dissimilaridade. Em muitos algoritmos a entropia é usada em conjunto com outros métodos. Em [22] é feita uma implementação do conhecido algoritmo k-médias com uma função objetivo baseada em entropia. O algoritmo MECA [23] usa a entropia em conjunto com *fuzzy clustering*. O algoritmo COOLCAT [24] usa a entropia para efetuar *clustering* em dados categóricos. Outros algoritmos procuram maximizar a entropia para efetuar *clustering* [25] e [26]. O algoritmo LEGClust [27] usa a entropia em conjunto com grafos e usa um método hierárquico para fazer *clustering*. Já o algoritmo MEC (*Minimum Entropy Clustering*) [28] usa a entropia em conjunto com uma partição inicial dos dados dada por outro método de *clustering*. Outros algoritmos usam entropia com grafos abrangentes [29] e também aplicada aos vetores existentes no sub-espaço dos dados [30]. A entropia é ainda usada como medida de proximidade ou de relação entre os *clusters* [31].

Capítulo 3

Algoritmos de Clustering

Neste capítulo são apresentados os algoritmos de *clustering* usados no âmbito deste trabalho, aplicados a conjuntos de dados biomédicos. Estes conjuntos de dados são de domínio público e foram retirados de diferentes repositórios *online*.

Os algoritmos *Minimum Entropy Clustering* e *LEGClust* foram implementados em MATLAB para o propósito dos objetivos deste trabalho. Foi usada a distribuição de 2011 versão b (R2011b 7.13.0.564). Os algoritmos *minCentropy*, *Spectral Clustering* e a função *Adjusted Rand Index* foram retirados da comunidade de utilizadores do MATLAB. Nas secções seguintes, é feita a descrição detalhada de cada algoritmo usado, do conjunto de dados de teste onde se efetuaram experiências preliminares e dos conjuntos de dados biomédicos utilizados.

3.1 Conjuntos de dados

3.1.1 Dados de Testes

Antes dos algoritmos serem aplicados a conjuntos de dados biomédicos, foi efetuada uma fase de experiências de teste usando conjuntos de dados artificiais e reais.

O conjunto de dados artificiais é constituído por dados bidimensionais. Este conjunto de dados é utilizado em [27] e [32] , onde são demonstrados vários resultados usando os mesmos. Em [28], [33] e [34] também são usados conjuntos de dados do mesmo tipo para demonstrar a eficácia dos algoritmos propostos. O conjunto de dados reais usado foi retirado do repositório *UCI Machine Learning*.

Sendo o uso deste tipo de dados bastante frequente devido à facilidade de visualização dos resultados, decidiu-se pela sua utilização neste trabalho para que

se pudesse avaliar os resultados dos vários algoritmos testados. Tendo em conta que foi feita uma implementação dos algoritmos propostos em [27] e [28], o uso de conjuntos de dados semelhantes permitiu também comparar resultados e validar a implementação.

Na tabela 3.1 são apresentadas as características dos conjuntos de dados reais usados nas experiências de teste. O conjunto de dados Breast Cancer, Ecoli, Glass, Haberman, Iris, Segmentation e Wine foram retirados de [35]. Nas figuras 3.1 estão representados os conjuntos de dados artificiais bidimensionais, usados em [32].

Tab. 3.1: Conjunto de dados reais usados para o teste dos algoritmos.

Nome	Objetos	Caraterísticas	Classes
Breast Cancer	699	10	2
Ecoli	336	8	8
Glass	214	10	6
Haberman	306	4	2
Iris	150	5	3
Segmentation	210	20	7
Wine	178	14	3

3.1.2 Dados Biomédicos

Neste trabalho foram usados 24 conjuntos de dados biomédicos. Os conjuntos são de domínio publico e estão disponíveis para *download*. O conjunto de dados Breast Cancer W., Column 2C, Column 3C, Lung Cancer-uci e Spectf estão disponíveis em [35]; Asthma, Babies, Growth, Surgery, Weights e Xray foram retirados de [36]; Acath, Diabetes, Dmd, Dominican, Gbsg, Pbc, Prostate e Stress echo foram retirados de [37] e por último, Depress, Lung cancer-ucla e Lung function foram retirados de [38]. Todos os conjuntos de dados possuem características numéricas, pois é um dos requisitos para os algoritmos testados. Conjuntos de dados categóricos não podem ser utilizados com os algoritmos usados neste trabalho. Cada conjunto de dados apresenta-se em forma de tabela, com dimensões $N \times M$ ou $N \times N$, sendo N o número de linhas, correspondente a cada objeto desse conjunto de dados e M o número de colunas, correspondente às características de cada objeto. Certos conjuntos possuem grande número de objetos, como por exemplo, os conjuntos Acath ou Asthma. Outros conjuntos possuem maior número de características, como no caso do conjunto Lung Cancer-uci, onde o número de características é quase quatro vezes maior que o número de objetos. Na tabela 3.2 estão sumariados os atributos

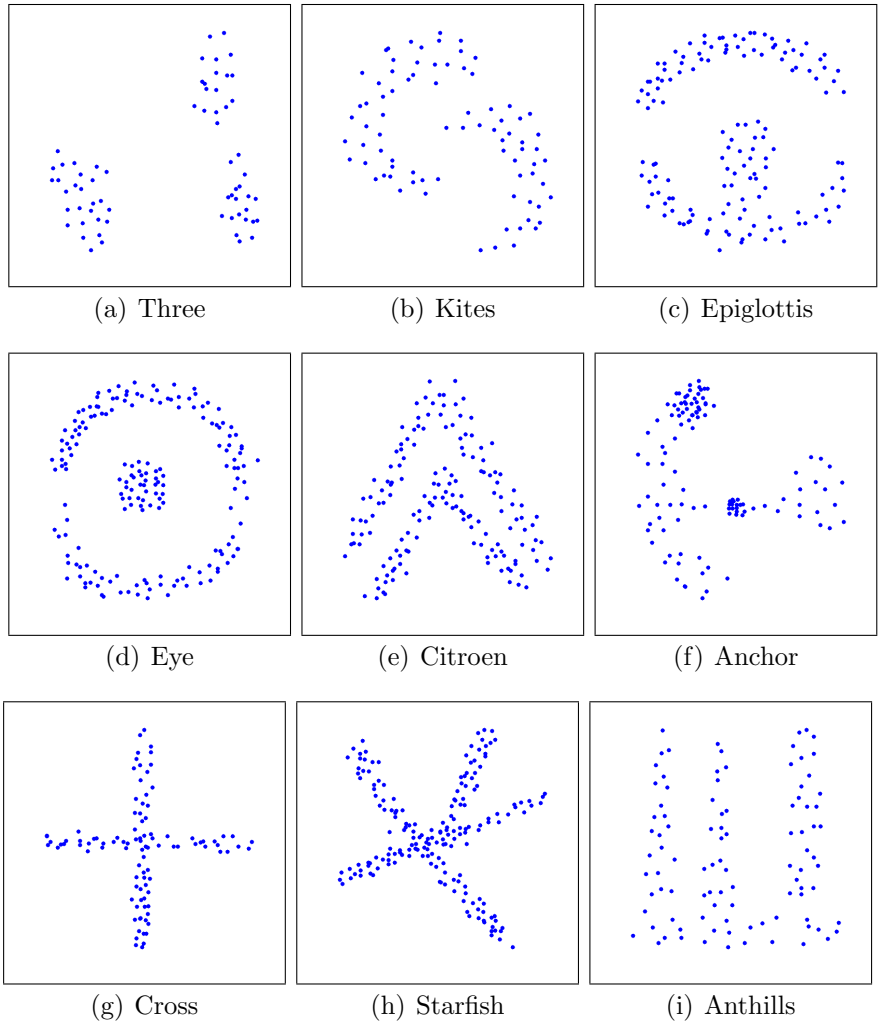


Fig. 3.1: Conjuntos de dados artificiais bidimensionais.

destes conjuntos de dados. Todos os conjuntos de dados biomédicos usados possuem um campo denominado *Classe*. Este campo permite saber a classe a que pertence cada objeto e consequentemente, permite validar o resultado de *clustering*. Muitos dos conjuntos de dados continham erros que poderiam impossibilitar o seu uso. Esses erros consistiam em campos nulos, isto é, alguns campos da tabela eram vazios, ou continham um valor não numérico. A correção dos conjuntos de dados foi feita manualmente, com a ajuda do MATLAB. Os conjuntos de dados foram importados com o MATLAB de modo a serem editados, onde foram removidos os objetos com campos inválidos.

Tab. 3.2: Conjuntos de dados biomédicos.

Nome	Objetos	Caraterísticas	Classes
Acath	2258	5	2
Asthma	2464	3	2
Babies	256	3	2
Breast cancer W.	699	10	2
Column 2C	310	6	2
Column 3C	310	6	3
Column full	620	6	4
Depress	294	10	2
Diabetes	366	13	3
Dmd	185	5	3
Dominican	318	3	2
Gbsg	686	7	2
Growth	277	6	2
Lung cancer-uci	27	102	3
Lung cancer-ucla	327	4	2
Lung function	150	36	4
Pbc	276	13	2
Prostate	483	19	2
Spectf	248	88	2
Stress echo	558	15	2
Surgery	126	6	2
Weights	550	10	4
Xray	150	10	2

3.2 Minimum Entropy Clustering

O algoritmo *Minimum Entropy Clustering* (MEC) foi desenvolvido para a análise de expressão genética. Tem por base a teoria de informação e o conceito de entropia. Os autores desenvolveram um modelo de entropia mínima, com base na entropia de Shannon e de Havrda-Charvat. O algoritmo, de uma forma muito generalizada, consiste em dois passos repetidos iterativamente (i) aplicação do modelo aos dados (ii) avaliação do resultado e consequente ação. Estes dois passos são repetidos até o algoritmo não conseguir fazer mais nenhuma alteração nos dados. Quando mais nenhuma alteração nos dados for possível significa que foram encontrados os *clusters* pretendidos. Seguidamente são explicados os modelos de entropia mínima e o algoritmo iterativo relativos ao algortitmo *Minimum Entropy Clustering*.

3.2.1 Modelo de Entropia Mínima

O modelo tem por base a entropia de Shannon e através desta os autores definem o critério de entropia mínima, do inglês, *minimum entropy clustering criterion*. Este critério consiste na aplicação da entropia condicional:

$$H(C|X) = -\int \sum_{j=1}^{m} p(c_j|x) log\, p(c_j|x) \quad . \tag{3.1}$$

Posteriormente, os autores aplicam ao critério a entropia de Havrda-Charvat. Desta forma teremos:

$$\hat{J} = \begin{cases} 1 - \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} p^{\alpha}(c_j|x_i) & \alpha > 1 \\ -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} p(c_j|x_i) ln p(c_j|x_i) & \alpha = 1 \\ \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} p^{\alpha}(c_j|x_i) - 1 & 0 < \alpha < 1 \end{cases} \tag{3.2}$$

que representa o critério de entropia mínima a ser usado no algoritmo. Para o uso deste critério é necessário estimar a probabilidade condicionada:

$$p(c_j|x_i) \quad . \tag{3.3}$$

Para esta estimativa os autores dão preferência a métodos não paramétricos, indicando que métodos paramétricos não são apropriados para a análise de genes. Duas técnicas de estimativa não paramétrica são aplicadas: janela de Parzen (*Parzen Window*) e k-vizinhos mais próximos (*k-nearest neighbors*).

Usando o método da janela de Parzen, queremos estimar o valor da função *p(x)* no ponto x. Para tal é feita uma janela *R(x)* à volta do ponto x. Deste modo pode-se estimar (3.3) dividindo o número de pontos do *cluster* maior (*cj*) dentro da janela, pelo número de todas as amostras contidas na janela *R(x)*, resultando em:

$$\frac{k(x|c_j)}{k(x)} \quad , \tag{3.4}$$

onde k é numero de pontos existente dentro de *R(x)*. Usando o método de vizinhos mais próximos, (3.3) é estimado por:

$$\frac{v(x)}{v(x|c_j)} \quad , \tag{3.5}$$

onde v é o volume da janela *R(x)*. Aplicando o método da janela de Parzen em (3.2) obtém-se:

$$\hat{J} = \begin{cases} 1 - \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}(\frac{k(x|c_j)}{k(x)})^{\alpha} & \alpha > 1 \\ -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{k(x|c_j)}{k(x)}log(\frac{k(x|c_j)}{k(x)}) & \alpha = 1 \\ \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}(\frac{k(x|c_j)}{k(x)})^{\alpha} - 1 & 0 < \alpha < 1 \end{cases} \tag{3.6}$$

Sendo 3.6 o critério de *clustering* final a ser utilizado pelo algoritmo.

3.2.2 Algoritmo Iterativo

O critério (3.6) não é aplicado diretamente aos dados. Os autores desenvolveram um algoritmo iterativo de modo a reduzir a entropia partindo de uma partição inicial, dada por outro método de *clustering*. Neste caso, o algoritmo *Minimum Entropy Clustering* procura por uma solução ideal partindo de uma partição já existente dada pelo algoritmo k-Médias.

A solução ideal é encontrada através do rearranjo da partição inicial. O rearranjo consiste na atribuição de um objeto x dos dados ao *cluster* que contém a maioria dos seus vizinhos. Suponhamos que um ponto x na partição inicial, foi atribuído ao *cluster* c_i, mas no entanto a maioria dos vizinhos de x pertencem a um *cluster* diferente de c_i, digamos c_j. Suponhamos também que n_j vizinhos de x pertencem ao *cluster* c_j e n_i vizinhos de x pertencem ao *cluster* c_i, ao ponto em que $n_i < n_j$. Como n_i é inferior a n_j, passando x para o *cluster* c_j a entropia de x será menor, o que indica que x pertence ao *cluster* c_j, pois a incerteza de x em relação ao *cluster* c_j é menor.

A entropia é calculada através da aplicação do critério (3.6) a cada ponto iterativamente. O calculo é feito para x pertencente ao *cluster* c_j e para x pertencente ao *cluster* c_i. São calculados dois valores de entropia: H e H'. Sendo que H é a entropia do sistema antes da mudança de x para um novo *cluster* e H' a entropia depois de x ser mudado de *cluster*. É calculada a diferença (h) entre as duas entropias. Quando h é inferior a 0, x é atribuído a um novo *cluster*. O calculo de h é feito por:

$$h = \sum_{y}(H'(C|y) - H(C|y)) \quad . \tag{3.7}$$

O cálculo de h e o consequente rearranjo, caso se verifique, é feito até não haver

mais rearranjos possíveis.

3.2.3 Implementação em MATLAB

O algoritmo MEC encontra-se disponível numa implementação em JAVA em [39]. Para o trabalho desta tese, foi desenvolvida uma implementação em MATLAB, baseada no artigo que descreve o algoritmo [28]. Na tabela (3.3) está descrito o pseudo-código deste algoritmo.

Tab. 3.3: Pseudo-código do algoritmo Minimum Entropy Clustering.

```
importar dados
inicializar variáveis
obter partição pelo k-médias
enquanto não houver mudanças
   para cada objeto
      encontrar vizinhos
      contar o número de pontos no cluster de x (ci)
      contar o número de pontos no cluster de x (cj)
      se cj > ci
      então encontrar h
      se h <0
         mudar x para cj
      terminar se
      terminar se
   terminar para
   contar mudanças
   se mudanças = 0
   então não houve mudanças
   terminar se
terminar enquanto
```

Um conjunto de dados inicial tem de ser fornecido ao algoritmo. O conjunto de dados apenas pode ter valores numéricos. A importação dos dados é feita através da função apropriada para o formato dos dados. Apenas foram usados dados em formato *text* e *MAT*.

Após serem importados, os dados são normalizados, isto é, a média de cada linha dos dados é colocada a 0 e o desvio padrão a 1. O algoritmo possui três parâmetros que devem ser escolhidos pelo utilizador:

- k - número de *clusters* para a partição inicial.

- σ - tamanho da janela de Parzen.
- α - valor que determina a equação a usar do critério 3.6.

O algoritmo é iterativo, percorre todos os pontos do conjunto de dados através de ciclos. Primeiramente são encontrados os vizinhos do objeto que se encontram dentro do intervalo $[x - \sigma, x + \sigma]$, que constituem os limites da janela de Parzen. É feita a contagem de quantos objetos pertencem ao *cluster* do objeto e quantos objetos pertencem a outro *cluster*. Se houver mais objetos de *cluster* diferente, procede-se ao cálculo de h como apresentado em 3.7. Se a maioria dos vizinhos pertencer ao mesmo *cluster* então o ciclo avança para o próximo objeto. O valor de h é verificado segundo a condição $h < 0$. Se a condição se verificar, x é mudado para o *cluster* dos vizinhos. O ciclo continua até não haver mais alterações que possam ser feitas. O resultado final é muito dependente da partição inicial e dos parâmetros definidos pelo utilizador. No capítulo 4 são apresentados resultados para experiências feitas com os dados de teste e com dados biomédicos usando a implementação do algoritmo MEC. São também apresentados resultados da implementação do algoritmo em JAVA.

3.3 LEGClust

O algoritmo LEGClust é baseado na entropia quadrática de Rényi e usa um método de *clustering* baseado em camadas de sub-grafos para formar os *clusters*. A entropia de Rényi, é definida por:

$$H(X) = -log(\int_c [f(x)]^2 dx) \quad , \tag{3.8}$$

sendo $f(x)$ a função de densidade de probabilidade que pode ser estimada usando método não paramétrico da janela de Parzen por:

$$f(x) = \frac{1}{Nh^m} \sum_{i=1}^{N} G(\frac{x - x_i}{h}, I) \quad . \tag{3.9}$$

Aplicando (3.9) em (3.8) obtém-se:

$$\hat{H} = -log(\frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} G(x_i - x_j; 0, 2h^2 I)) \quad , \tag{3.10}$$

onde N é o número de objetos, G um *kernel*, m a dimensão de X, h o tamanho da janela de Parzen e I a matriz identidade de ordem M. Usando o *kernel* Gaussiano:

$$G(x;0,I) = \frac{1}{(2\pi)^{\frac{m}{2}}\,|I|^{\frac{1}{2}}} exp(-\frac{1}{2}x^T I^{-1}x) \quad , \tag{3.11}$$

em (3.10), obtemos finalmente uma expressão para estimar a entropia quadrática de Rényi:

$$\hat{H} = -log\left[\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{1}{(2\pi)^{\frac{m}{2}}\,|2h^2I|^{\frac{1}{2}}} exp(-\frac{1}{2}(x_i - x_j)^T(2h^2I)^{-1}(x_i - x_j))\right] \quad . \tag{3.12}$$

O algoritmo LEGCLust divide-se em duas grandes componentes, *i*) uma matriz de proximidade entrópica e *ii*) um método hierárquico para a formação de *clusters*.

3.3.1 Matriz de proximidade entrópica

Através de (3.12) é calculada a matriz de dissimilaridade entrópica. Esta matriz $N \times N$, onde N são os objetos do conjunto de dados, consiste num mapa que relaciona os objetos entre si. A todos os objetos, um por um, é aplicada a equação (3.12). Tendo em conta um número específico de vizinhos do objeto, é calculada a entropia do objeto em relação aos seus vizinhos.

A partir da matriz de dissimilaridade entrópica é formada a matriz de proximidade entrópica, que consiste numa matriz $N \times M$, onde N são todos os objetos e M são as camadas em que o objeto se liga a um dos seus vizinhos (M é igual ao número de vizinhos). Na primeira coluna encontram-se todos os objetos. Nas colunas seguintes encontram-se os objetos com os quais os objetos da primeira coluna se ligam. As ligações são feitas entre as camadas até todos os objetos se ligarem entre si.

A título de exemplo, para uma melhor compreensão, a tabela 3.4 e 3.5 representam uma matriz de dissimilaridade e uma de proximidade, respetivamente. O exemplo contem seis objetos, onde foram escolhidos dois vizinhos, consequentemente a matriz de proximidade contém duas camadas. Na matriz de proximidade é feita a relação entre as entropias dos objetos. Objetos com entropias mais baixas relacionam-se na primeira camada da matriz de proximidade e assim sucessivamente, isto é, o par de objetos com entropia menor, são os primeiros a serem conectados.

Tab. 3.4: Exemplo de uma matriz de dissimilaridade entrópica.

X	1	2	3	4	5	6
1		3.25	5.67	-	-	-
2	2.89		4.12	-	-	-
3	6.87	5.63		-	-	-
4	5.89	-	2.34		-	-
5	-	-	-	9.21		4.95
6	-	-	-	2.73	6.45	

Tab. 3.5: Exemplo de uma matriz de proximidade entrópica.

X	Camada 1	Camada 2
1	2	3
2	1	3
3	2	1
4	3	1
5	6	4
6	4	5

3.3.2 Formação de clusters com o algoritmo hierárquico

A formação de *clusters* é feita a partir da matriz de proximidade entrópica. Cada camada da matriz representa um sub-grafo. A aglomeração de todos os sub-grafos constitui *clusters* até todos os objetos pertencerem ao mesmo *cluster*. Cada sub-grafo é constituído ligando cada objeto ao respetivo objeto de cada camada da matriz de proximidade, criando assim *clusters* em cada sub-grafo. Para ligar os sub-grafos, os autores desenvolveram um método hierárquico aglomerativo. Com a primeira camada são criados os *clusters* primários, sendo seguidamente conetados os objetos dos *clusters* da segunda camada e pela mesma forma os objetos das camadas seguintes.

O algoritmo é hierárquico pois inicia a ligação dos objetos da camada mais baixa até à camada mais alta, sendo esta a última camada. Em cada camada as ligações entre os objetos são orientadas, isto é, suponhamos que o objeto 1 se liga ao objeto 2, então faz-se uma ligação orientada de $1 \rightarrow 2$. Se o objeto 2 por sua vez se liga ao objeto 1, então a ligação terá de ser orientada em dois sentidos, $1 \leftrightarrow 2$. Estas orientações permitem contabilizar as ligações entre os *clusters* de cada sub-grafo. Os *clusters* são aglomerados conforme o número de ligações entre os objetos que os constituiem. Por exemplo, os objetos do *cluster A* possuem 3 ligações com os objetos do *cluster B* e 5 ligações com os objetos do *cluster C*. Como A e C possuem

mais ligações entre si, o algoritmo funde os dois *clusters* em um só. Este processo é feito para todas as camadas, juntando assim todos os *clusters* entre as camadas de modo hierárquico.

3.3.3 Implementação em MATLAB

A implementação deste algoritmo foi baseada na descrição feita em [27]. O pseudo-código pode ser encontrado na tabela 3.6. Tal como descrito em [27], inicialmente

Tab. 3.6: Pseudo-código do algoritmo LEGClust.

```
importar dados
inicializar variáveis
para cada objeto
   encontrar os vizinhos mais próximos
   calcular os vetores
   calcular a matriz de dissimilaridade
terminar para
construir matriz de proximidade
construir a primeira camada
para cada camada
   para cada objeto
      juntar clusters com maior número de ligações
   terminar para
terminar para
mostrar solução
```

são definidos três parâmetros necessários para o funcionamento do algoritmo:

- M - número de vizinhos do objeto.
- h - tamanho da janela de Parzen.
- k - número de ligações mínimas para juntar os *clusters*.

O valor destes parâmetros é definido pelo utilizador e influencia o resultado final. Os vizinhos mais próximos do objeto são encontrados através de uma função especifica do MATLAB que retorna os M vizinhos mais próximos. É feito um mapa de vetores entre os vizinhos e o objeto e são calculados os vetores que ligam o objeto a cada vizinho. Com estes vetores, aplicados à equação (3.12), é construída a matriz de dissimilaridade. Com esta matriz é construída a matriz de proximidade, que por sua vez, é usada na criação dos *clusters* elementares, que se encontram na primeira

camada. O algoritmo hierárquico usa os *clusters* elementares como base para ligar os objetos das camadas seguintes e juntar os *clusters* que tenham k ou mais ligações entre si. É necessário definir um ponto de corte, isto é, o número desejado de *clusters* a encontrar. O algoritmo inicia o processo de junção dos *clusters* onde se formam vários até existir um só *cluster*. Este ponto de corte é definido também pelo utilizador e o algoritmo retorna a solução com o número de *clusters* desejado, isto é, definido pelo ponto de corte. Caso não haja um valor de *clusters* igual ao ponto de corte, o algoritmo seleciona o número mais próximo e retorna essa solução. No capítulo seguinte são apresentados os resultados obtidos com as experiências feitas aos dados de teste e dados biomédicos usando esta implementação.

3.4 minCentropy

Baseado numa aproximação do ponto de vista da teoria da informação, o algoritmo minCentropy usa um método de cálculo de entropia muito semelhante aos métodos descritos em [33]. O algoritmo encontra-se disponível em [40] e já implementado em MATLAB. Os autores desenvolvem um novo critério baseado na entropia de Havrda-Charvat em conjunto com a estimativa de densidade por janela de Parzen.

3.4.1 Definição da função objetivo

Os autores procuram minimizar a entropia condicional através de:

$$C = argmin H(X|C) \quad . \tag{3.13}$$

Sendo a entropia de Havrda-Charvat:

$$H_\alpha = (2^{1-\alpha})^{-1} \left[\sum_{k=1}^{K} p_k^\alpha - 1 \right] \quad , \tag{3.14}$$

atribuindo a α o valor 2, é possível definir a entropia quadrática condicional de Havrda-Charvat como:

$$H_2(X|C) = \sum_{k=1}^{K} p(c_k) H_2(X|C = c_k) \quad . \tag{3.15}$$

Aplicando (3.15) a (3.13), os autores definem a função objetivo como:

$$C = argmin \left\{ \sum_{k=1}^{K} p(c_k) H_2(X|C = c_k) \right\} \quad . \tag{3.16}$$

Para a estimativa da densidade de probabilidade, os autores usam o método da janela de Parzen. É usado o *kernel* gaussiano:

$$G(x - a, \sigma^2) = \frac{1}{(2\pi\sigma)^{\frac{d}{2}}} exp\left(\frac{-\|x - a\|^2}{2\sigma^2} \right) \quad , \tag{3.17}$$

onde σ é a largura da janela e a é o centro da janela. A estimativa da densidade de X é dada por:

$$p(x) = \frac{1}{N} \sum_{i=1}^{N} G(x - x_i, \sigma^2) \quad . \tag{3.18}$$

Aplicando (3.18) em (3.15), obtém-se:

$$H_2(X) = 1 - \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} G(x_i - x_j, 2\sigma^2) \quad . \tag{3.19}$$

Combinando (3.19) com a função objetivo definida em (3.16), obtém-se a função final para o calculo da entropia:

$$CE(C) = \sum_{k=1}^{K} \frac{\sum_{x_i, x_j \in c_k} exp\left(\frac{-\|x_i - x_j\|^2}{4\sigma^2} \right)}{n_k} \quad , \tag{3.20}$$

onde K é o número de *clusters*, n_k é o número de objetos no *cluster* e σ é o tamanho da janela.

3.4.2 Implementação do algoritmo

Este é um algoritmo iterativo que percorre todos os objetos dos dados. Em cada iteração, um objeto é considerado para ser movido para um novo *cluster*, caso o objeto seja semelhante aos objetos incluídos no *cluster*. O algoritmo faz as iterações sobre uma partição prévia dos dados. Esta partição pode ser fornecida pelo utilizador ou calculada usando o método de *k-médias*. Na tabela 3.7 está descrito o pseudo-código do algoritmo *minCentropy*.

Um conjunto de dados é importado, bem como uma partição desse mesmo conjunto (caso a partição não seja fornecida, o algoritmo através do *k-médias* cria uma

Tab. 3.7: Pseudo-código do algoritmo minCentropy.

```
importar dados
inicializar variáveis
obter partição inicial com k-médias
calcular a matriz com a similaridade entre os pares de
objetos
calcular a matriz de similaridade objeto-a-cluster
flag = verdadeiro
enquanto flag = verdadeiro fazer
   flag = falso
   para cada objeto fazer
      calcular entropia
      se entropia > 0 então
         mudar objeto de cluster
         flag = verdadeiro
      terminar se
   terminar para
terminar enquanto
mostrar solução
```

partição). Algumas variáveis são inicializadas e são construídas duas matrizes especiais para efetuar o cálculo da entropia:

- Matriz de similaridade entre pares de objetos (*Pairwaise Similarity Matrix*) - consiste numa matriz $N \times N$ onde são guardados os valores da semelhança entre pares de objetos.
- Matriz de similaridade objeto-a-cluster (*Point-to-Cluster Similarity Matrix*) - consiste num vetor $N \times K$ onde são guardados os valores da semelhança entre um objeto e os objetos contidos num *cluster*.

Muito semelhante ao algoritmo descrito em [28] , o algoritmo *minCentropy* percorre através de ciclos todos os objetos dos dados, verificando a entropia do objeto num *cluster* e a entropia mudando o objeto para um *cluster* diferente. A condição para a mudança implica que a entropia seja maior que 0. Determinados parâmetros têm de ser definidos pelo utilizador:

- k - número de partições inicial.
- σ - tamanho da janela de Parzen.
- *nrun* - número de vezes que o algoritmo deverá ser corrido.

- C - este parâmetro é opcional, consiste na partição inicial, caso não seja fornecido, o algoritmo elabora uma partição.

3.5 Spectral Clustering

Os algoritmos de *spectral clustering* descritos em [34] , [41] e [42] , demonstraram obter bons resultados, razão pela qual também foram escolhidos para serem usados neste trabalho. Três implementações diferentes de *spectral clustering* são usadas. Estes três algoritmos implementados em MATLAB podem ser encontrados em [43].

3.5.1 Shi & Malik - Normalized Cut

O agrupamento dos dados é feito com base na teoria de grafos. O conjunto de dados é interpretado como um aglomerado de pontos todos conectados, onde cada ponto é visto como um vértice (*edge*) da rede formada pelas ligações. Os autores propõem um critério para fazer a partição dos dados baseado em grafos, denominado *normalized cut*. Usando este critério em conjunto com vetores próprios da matriz Laplaciana o algoritmo consegue efetuar partições nos dados. Enumeram-se abaixo os passos que o algoritmo percorre até encontrar os *clusters*.

1. Criar um grafo G a partir de um conjunto de dados.
2. Atribuir uma etiqueta (*label*) aos vértices do gráfico (*weighted graph*).
3. Sumarizar grafo em duas matrizes W e D.
4. Encontrar os menores valores próprios através de $(D - W)x = \lambda Dx$.
5. Usando o segundo menor valor próprio na matriz Laplaciana, encontrar o ponto de corte onde *Ncut* é mínimo e fazer partição.
6. Verificar se a partição atual deve ser repartida, através da estabilidade do corte.

W é uma matriz simétrica de dimensões $N \times N$, onde são guardadas as distâncias entre os vértices. D é uma matriz diagonal de dimensões $N \times N$, preenchida por 0 exceto na sua linha diagonal, onde são guardados os valores de d, somatório das ligações do vértice a todos os outros vértices do grafo. *Ncut* é definido pela equação 3.21. Esta equação representa o critério *normalized cut*, onde $cut(A, B)$ representa

um corte fazendo duas partições A e B, e $assoc(A, V)$ representa o total de ligações dos vértices em A com todos os vértices do grafo. Por sua vez, $assoc(B, V)$ representa as ligações de B com todos os vértices do grafo. Ao utilizador é apenas pedido o número de *clusters* que deseja encontrar.

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)} \quad . \tag{3.21}$$

3.5.2 Ng & Jordan - K-Eigenvectors

O algoritmo proposto em [34] também recorre ao uso de vetores próprios da matriz Laplaciana. Introduz o uso de partições através do algoritmo k-médias. Abaixo são enumerados os passos do algoritmo para um conjunto de dados $S = \{s_1, ...s_n\}\, s_1, ...s_n$.

1. Criar matriz de afinidade definida por $A_{ij} = exp(-\left\|s_i - s_j\right\|^2 / 2\sigma^2)$.
2. Criar D, uma matriz diagonal onde o elemento (i, i) é a soma da linha i na matriz A.
3. Criar matriz Laplaciana L definida por $D^{1/2}AD^{-1/2}$.
4. Encontrar os k maiores vetores próprios em L.
5. Criar vetor X colocando os k maiores vetores próprios numa coluna.
6. Criar matriz Y atraves de X, aplicando $Y_{ij} = X_{ij}/(\sum_j X_{ij}^2)^{\frac{1}{2}}$ a todas as linhas de X.
7. Usando Y, criar *clusters* com k-médias.
8. Atribuir o ponto original s_i ao *cluster* j se e só se a linha i da matriz Y pertencer ao $clusterj$.

A matriz de afinidade (*affinity matrix*) pode ser vista como um grafo, onde as linhas e as colunas estão todas ligadas entre si, representando as ligações entre os vértices de um grafo, e o valor dos campos da matriz representa os próprios vértices. O parâmetro σ é encontrado de forma automática pelo algoritmo. O utilizador apenas introduz o número de *clusters* a encontrar.

3.5.3 Perona & Freeman - Affinity Factorization

Em [42] os autores exploram o trabalho de *Shi* e *Malik* para desenvolver um algoritmo simples que faça uso do conceito de criação de fatores a partir da matriz de afinidade. Seguidamente são enumerados os passos efetuados pelo algoritmo.

1. Criar matriz de afinidade A_{ij} entre os pares de elementos.
2. Calcular p, associado ao maior valor próprio de A.
3. Definir o primeiro plano F como o conjunto de objetos i cujo p_i correspondente não seja igual a 0.

A_{ij} é uma medida de similaridade entre dois objetos do conjunto de dados. Quando os dois objetos são muito semelhantes, A_{ij} tem um valor alto. Se a semelhança entre os dois objetos é reduzida, consequentemente A_{ij} terá um valor baixo, muito perto de zero. p é uma função aplicada a todos os objetos, um por um. A função é definida por:

$$p = argmin \sum_{i,j=1}^{N} (A_{i,j} - \hat{p}_i \hat{p}_j)^2 \quad . \tag{3.22}$$

Os *clusters* são formados calculando p de cada objeto em A_{ij}. Objetos com p iguais pertencem ao mesmo *cluster*. Com este algoritmo o utilizador não define nenhum parâmetro, o algoritmo encontra os *clusters* automaticamente. No capitulo 4 é feita uma apreciação desta característica analisando os resultados obtidos.

3.6 K-Médias

O algoritmo k-médias faz uma partição dos dados em quantos *clusters* o utilizador desejar. O algoritmo é iterativo, o utilizador define k *clusters* que pretende encontrar e o algoritmo cria k centróides no conjunto de dados. Através de várias iterações, os objetos são atribuídos aos *cluster* dependendo da sua distância a cada centróide. É usada a distância Euclidiana para relacionar os objetos com os centróides. A cada iteração, também os centróides são recalculados. As iterações são feitas até que a soma de todas as distâncias Euclidianas dos objetos aos seus centróides;

$$J = \sum_{j=1}^{k} \sum_{n \in S_j} \|x_n - c_j\|^2 \quad , \tag{3.23}$$

seja mínima. Na tabela 3.8 é mostrado o pseudo-codigo do algoritmo k-médias. Foi usada a implementação do algoritmo de k-médias fornecida com o programa MATLAB. Em [44] pode ser encontrada a função que implementa este algoritmo.

Tab. 3.8: Pseudo-código do algoritmo K-means.

importar dados
definir n número de clusters
implementar n centroides
enquanto o critério de finalização não for atingido **fazer**
  para cada objeto **fazer**
    calcular a distância de cada objeto aos centroides
  terminar para
  calcular matriz com a distância entre pares de objetos
  re-calcular os centroids
  para cada objeto na matriz **fazer**
    mudar objeto para o cluster que contem o centroide mais próximo
  terminar para
terminar enquanto

3.7 Mean Shift

O algoritmo *Mean Shift* é bastante usado no reconhecimento de padrões, mais especificamente na análise e processamento de imagem. É um procedimento para localizar os máximos da função de densidade dos objetos do conjunto de dados. O procedimento é iterativo e usa um *kernel*, $K(x_i - x)$, aplicado a cada objeto, para determinar a densidade do conjunto de objetos à sua volta. O *kernel* é Gaussiano definido por:

$$K(x_i - x) = e^{c\|x_i - x\|^2} \quad . \tag{3.24}$$

A média da densidade dos objetos é definida por:

$$m(x) = \frac{\sum_{i=1}^{n} x_i g(\left\| \frac{x - x_i}{h} \right\|^2)}{\sum_{i=1}^{n} g(\left\| \frac{x - x_i}{h} \right\|^2)} - x \quad . \tag{3.25}$$

A mudança do *kernel* de um objeto para o próximo é indicada por um vetor orientado na direção da densidade máxima. O procedimento pode ser enumerado

com (*i*) aplicar (3.25) aos dados até encontrar o ponto máximo da densidade (*ii*) encontrar a área de atração desse ponto, encontrando todos os pontos à volta cuja densidade converge para o centro (*iii*) agrupar todos os objetos dentro da área de atração no mesmo *cluster*. O algoritmo de *Mean Shift* usado pode ser encontrado em [18].

3.8 Adjusted Rand Index

O índice *Adjusted Rand Index* (ARI) foi usado neste trabalho como forma de validar os resultados dos algoritmos de *clustering*. A sua implementação em MATLAB está disponível em [45]. O ARI é um índice que mede a correspondência entre duas partições do mesmo conjunto de dados. A correspondência é obtida com base numa tabela denominada tabela de contingência. De uma forma simplificada, o ARI avalia a correspondência entre o resultado de *clustering* e as classes reais do conjunto de dados (as classes são conhecidas à priori). É feita a verificação dos objetos pertencentes aos *clusters* em comparação com as suas classes verdadeiras. Suponhamos a existência de um resultado de *clustering* V e as classes reais U do conjunto de dados. Suponhamos agora a existência de dois sub-conjuntos, pertencentes a cada partição: r pertencente a U e c pertencente a V. A tabela 3.9 representa a tabela de contingência que relaciona U com V e r com c, respetivamente.

Tab. 3.9: Tabela de contingência relacionando duas partições, U e V.

		Cluster				
		V_1	V_2	...	V_c	Soma
Classe	U_1	n_{11}	n_{12}	...	n_{1c}	$n_{1.}$
	U_1	n_{12}	n_{22}	...	n_{2c}	$n_{2.}$
	...	...	...	...	...	...
	U_r	n_{r1}	n_{r2}	...	n_{rc}	$n_{r.}$
	Soma	$n_{.1}$	$n_{.2}$	...	$n_{.c}$	$n_{..}$ = n

O valor do índice ARI é calculado por:

$$ARI = \frac{\binom{n}{2}\sum_{r=1}^{R}\sum_{c=1}^{C}\binom{n_{rc}}{2} - \left[\sum_{r=1}^{R}\binom{n_{r.}}{2}\sum_{c=1}^{C}\binom{n_{.c}}{2}\right]}{\frac{1}{2}\binom{n}{2}\left[\sum_{r=1}^{R}\binom{n_{r.}}{2} + \sum_{c=1}^{C}\binom{n_{.c}}{2}\right] - \left[\sum_{r=1}^{R}\binom{n_{r.}}{2}\sum_{c=1}^{C}\binom{n_{.c}}{2}\right]} \quad . \tag{3.26}$$

O índice assume valores entre 0 e 1, sendo que valores perto de 0 são menos

satisfatórios que valores mais perto de 1.

Capítulo 4

Clustering de Dados Biomédicos

Neste capítulo são documentados os resultados das experiências realizadas com os algoritmos de *clustering*. Primeiramente foram efetuadas experiências com conjuntos de dados artificiais bidimensionais e dados reais, com o objetivo de testar a implementação dos algoritmos. Posteriormente avançou-se para experiências dos algoritmos usando conjuntos de dados biomédicos.

4.1 Testes Exploratórios

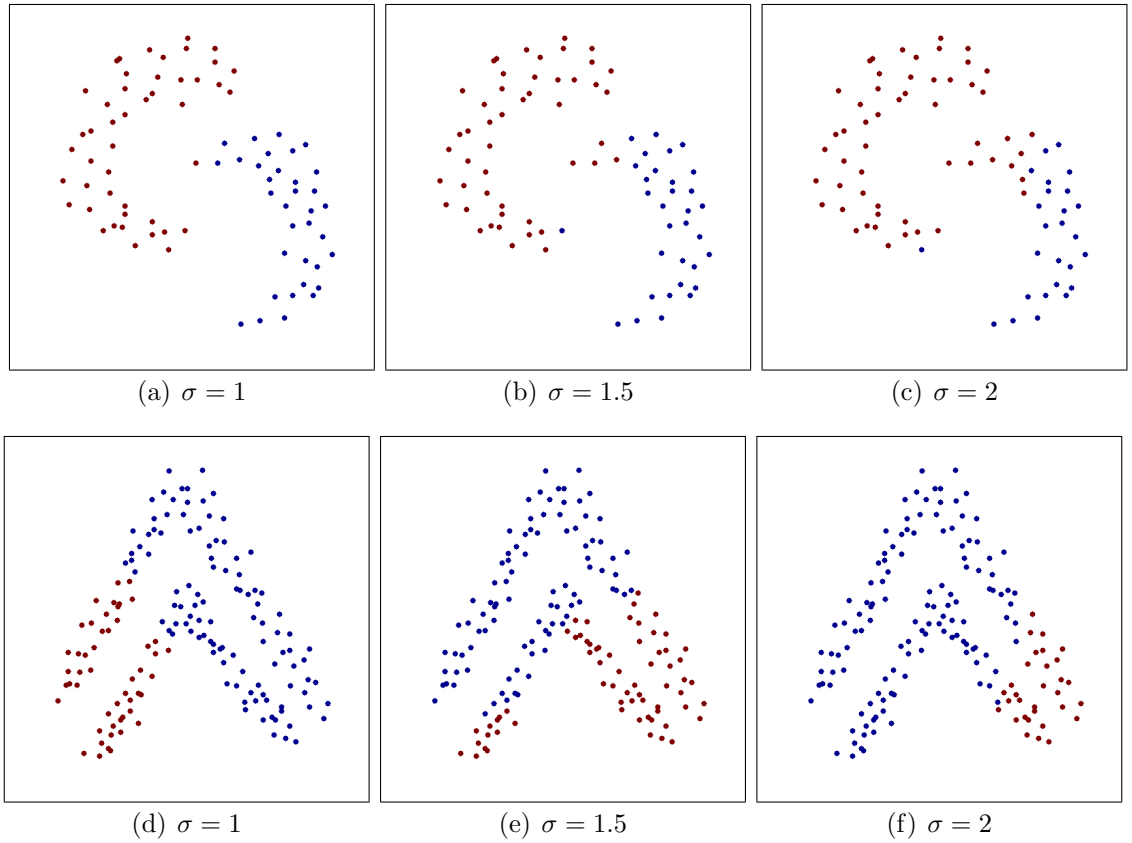
Os algoritmos *Minimum Entropy Clustering* e LEGClust foram implementados em MATLAB e de modo a validar a sua implementação, foram efetuados testes com conjuntos de dados bidimensionais e ainda alguns conjuntos de dados reais. Foram efetuados alguns testes envolvendo outros algoritmos de modo a comparar resultados. Vários dos algoritmos usados necessitam da definição de determinados parâmetros pelo utilizador. Durante os testes foram efetuadas algumas experiências com variações de valores dos parâmetros, de modo a encontrar os valores que produzissem os melhores resultados.

4.1.1 Testes com dados artificiais bidimensionais

Minimum Entropy Clustering

O algoritmo *Minimum Entropy Clustering* (MEC) possui três parâmetros que devem ser definidos pelo utilizador, o número de *clusters*, o tamanho da janela de Parzen e o critério de entropia a usar. O número de *clusters* foi escolhido mediante a inspeção visual à priori dos conjuntos de dados bidimensionais.

Na figura 4.1 estão representados os resultados para os conjuntos Kites e Citroen. Ambos possuem dois aglomerados de pontos distintos, por isso a solução ideal seria dois *clusters*. Foram usados diferentes valores para definir o tamanho σ da janela de Parzen. Na figura 4.1 (a), (b) e (c) estão representados os resultados obtidos com o conjunto Kites de dois *clusters*. Aumentando o valor de σ acima de 1.5, a formação dos *clusters* não representa a estrutura do conjunto. Com o conjunto Citroen, os *clusters* formados não são os ideais. Aumentando o valor de σ apenas faz com que um dos *clusters* englobe mais pontos. Valores de σ acima de 1.5 não produzem resultados aceitáveis.

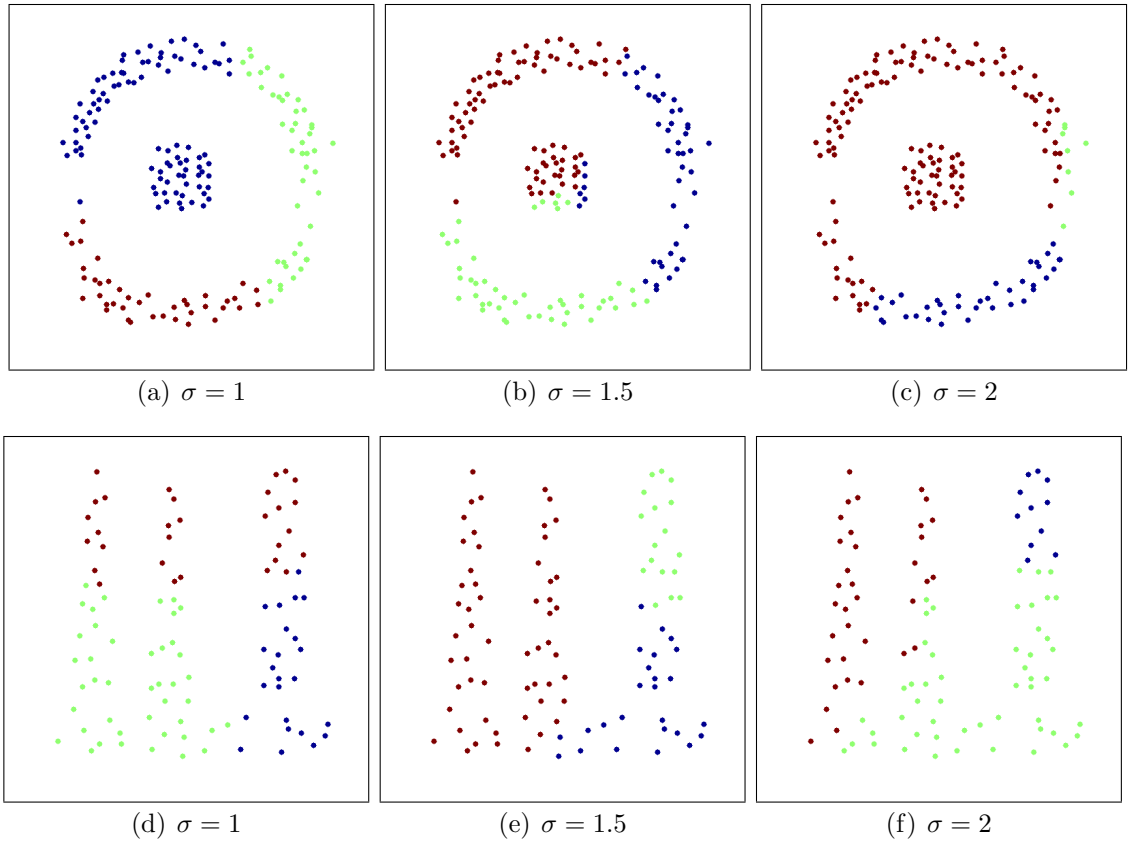


(a) $\sigma = 1$ (b) $\sigma = 1.5$ (c) $\sigma = 2$

(d) $\sigma = 1$ (e) $\sigma = 1.5$ (f) $\sigma = 2$

Fig. 4.1: Resultados obtidos com Minimum Entropy Clustering nos conjuntos Kites e Citroen com dois clusters.

Na figura 4.2 estão representados os conjuntos Anthills e Eye onde foram encontrados três *clusters*. Aumentando o parâmetro σ faz com que os *clusters* englobem mais pontos, como demonstrado em (c) com o conjunto Eye. Neste, os três *clusters* ideais seriam compostos por um grupo de pontos circular ao centro e dois grupos em cima e em baixo com forma côncava. Os *clusters* encontrados pelo algoritmo em Eye não refletem corretamente os grupos naturais contidos nos dados, pelo que



(a) $\sigma = 1$ (b) $\sigma = 1.5$ (c) $\sigma = 2$

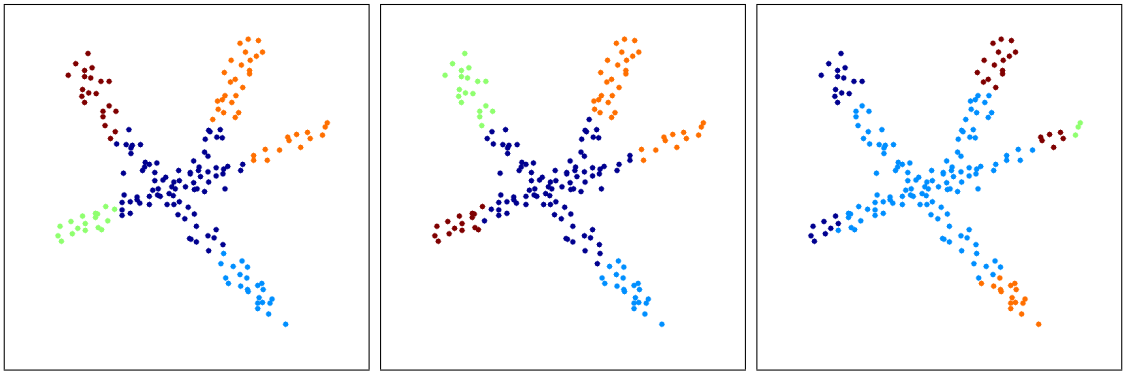
(d) $\sigma = 1$ (e) $\sigma = 1.5$ (f) $\sigma = 2$

Fig. 4.2: Resultados obtidos com Minimum Entropy Clustering nos conjuntos Anthills e Eye com três clusters.

não representam a estrutura contida nos dados.

No conjunto Anthills os *clusters* encontrados também não refletem a estrutura dos dados. Dado que, idealmente, seriam de esperar três *clusters* em que cada um representa uma das estruturas com forma triangular. O aumento de σ acima de 1.5 volta a obter resultados pouco favoráveis.

Na figura 4.3 estão representados os resultados obtidos com o conjunto Starfish. Foram procurados cinco *clusters* de modo a corresponder a uma das possíveis soluções. Aumentando σ não influenciou positivamente os resultados e não foi possível ultrapassar o valor de 1.4. O conjunto de dados Starfish possui muitos pontos, pelo que o aumento de σ permite que mais vizinhos entre os pontos sejam encontrados, o que implica mais tempo de processamento no cálculo da entropia, uma vez que esta é calculada através dos vizinhos do ponto em questão. Os *clusters* encontrados não são correspondentes à estrutura dos dados, idealmente cinco *clusters* onde cada um devia corresponder a cada fila de pontos formada a partir do centro, resultando na forma de uma estrela.



(a) $\sigma = 1$ (b) $\sigma = 1.2$ (c) $\sigma = 1.4$

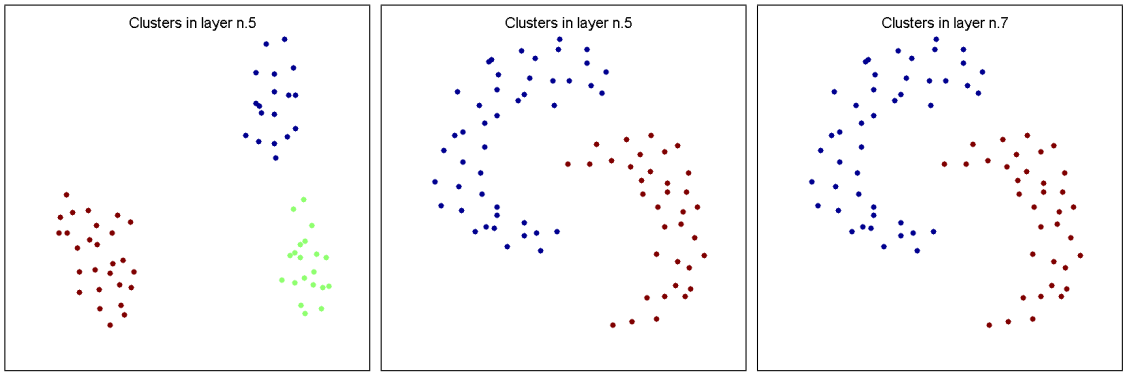
Fig. 4.3: Resultados obtidos com Minimum Entropy Clustering o conjunto Starfish com cinco clusters.

Analisando os resultados de todos os conjuntos de dados, é possível afirmar que valores de σ acima de 1.5 não obtêm resultados favoráveis. Valores abaixo de 1 não apresentam qualquer alteração à partição inicial feita pelo k-médias. Tendo em conta que σ determina o tamanho da janela de Parzen, com valores muito pequenos a janela não consegue encontrar vizinhos suficientes. Valores no intervalo de 1 a 1.5 apresentam resultados mais coerentes. Para as experiências seguintes será usado o valor 1 para σ.

De um modo geral, os resultados obtidos pelo algoritmo *Minimum Entropy Clustering* com os dados artificiais bidimensionais, não possuem o *clustering* ideal que representa a estrutura dos dados. Nestes conjuntos de dados artificiais com formas bem definidas, é relativamente fácil encontrar grupos através da inspeção visual e prever o resultado que seria ideal. Com o conjunto de dados Kites, o resultado obtido aproxima-se muito do ideal. Conjuntos de dados com grupos bem definidos e espaçados como no caso do conjunto Kites, o algoritmo *Minimum Entropy Clustering* não apresenta dificuldade em encontrar os *clusters* pretendidos. No entanto, com grupos de pontos interligados ou muito próximos, o algoritmo implementado apresenta dificuldades em encontrar os *clusters* ideais. Em [28] os resultados apresentados com um conjunto de dados artificial bidimensional semelhante, representam melhor a estrutura dos dados, pelo que admitimos que possa haver diferenças na implementação dos algoritmos ou que tenham sido usados nesse trabalho outros parâmetros não referidos.

LEGClust

O algoritmo LEGClust possui três parâmetros cujo valor deve ser definido pelo utilizador. O número de vizinhos M, o número de ligações entre *clusters* K e o tamanho da janela de Parzen h. O valor da janela de Parzen foi definido por: $h = 2\sigma\left(\frac{4}{(m+2)N}\right)^{\frac{1}{m+4}}$ como sugerido em [27], para todas as experiências com o algoritmo LEGClust. Foi variado apenas o número de vizinhos e o número de ligações entre *clusters*.



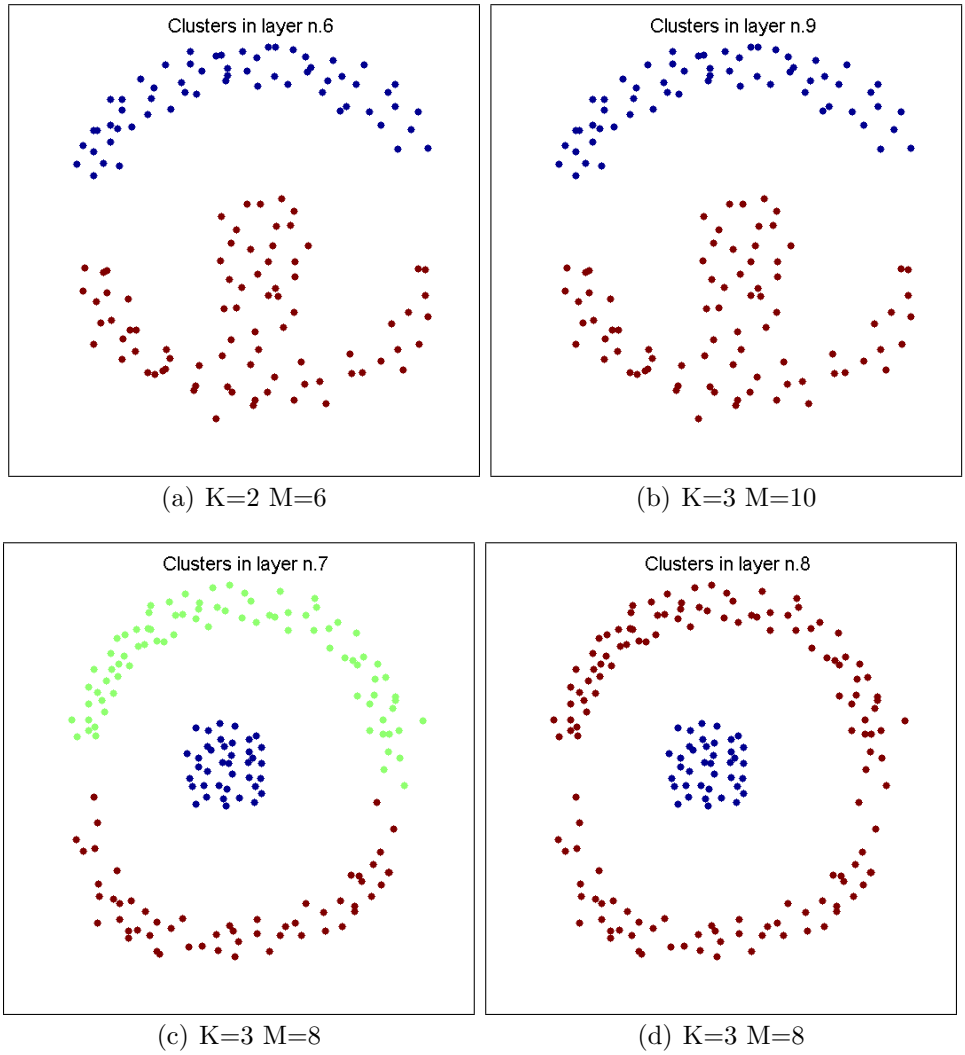
(a) K=2 M=5 (b) K=2 M=5 (c) K=3 M=10

Fig. 4.4: Resultados obtidos com os conjuntos Three e Kites usando o LEGClust.

Na figura 4.4 estão representados os resultados obtidos na experiência de teste com o algoritmo LEGClust usando os conjuntos Three e Kites. Estes dois conjuntos possuem aglomerados de pontos bem definidos. Através da inspeção visual do conjunto Three, intuitivamente são encontrados três *clusters* diferentes, representados na figura 4.4 (a) por cores diferentes. No conjunto Kites é possível identificar dois *clusters* diferentes representados na figura 4.4 (b) e (c). Usando o valor 2 e 3 para K no conjunto Kites foi possível encontrar dois *clusters*. Aumentando o valor de K tornou-se necessário aumentar também o valor de M. O aumento de K implica que só *clusters* com K ou acima de K ligações são agrupados, sendo que são necessárias mais camadas para ocorrer a junção de *clusters* que possuam acima de duas ligações. Dado que M é o número de vizinhos e consequentemente o número de camadas entrópicas, foi necessário aumentar M de modo a acontecer a junção de mais *clusters*.

Em ambos os conjuntos de dados os resultados traduzem a estrutura dos mesmos. Os resultados obtidos em (b) e (c) são iguais, independentemente da troca de parâmetros. Em (b) com $K = 2$ a solução proposta com dois *clusters* é encontrada na camada cinco enquanto que em (b) com $K = 3$ a solução é encontrada na décima

camada. Para ambos os valores a solução encontrada é adequada.



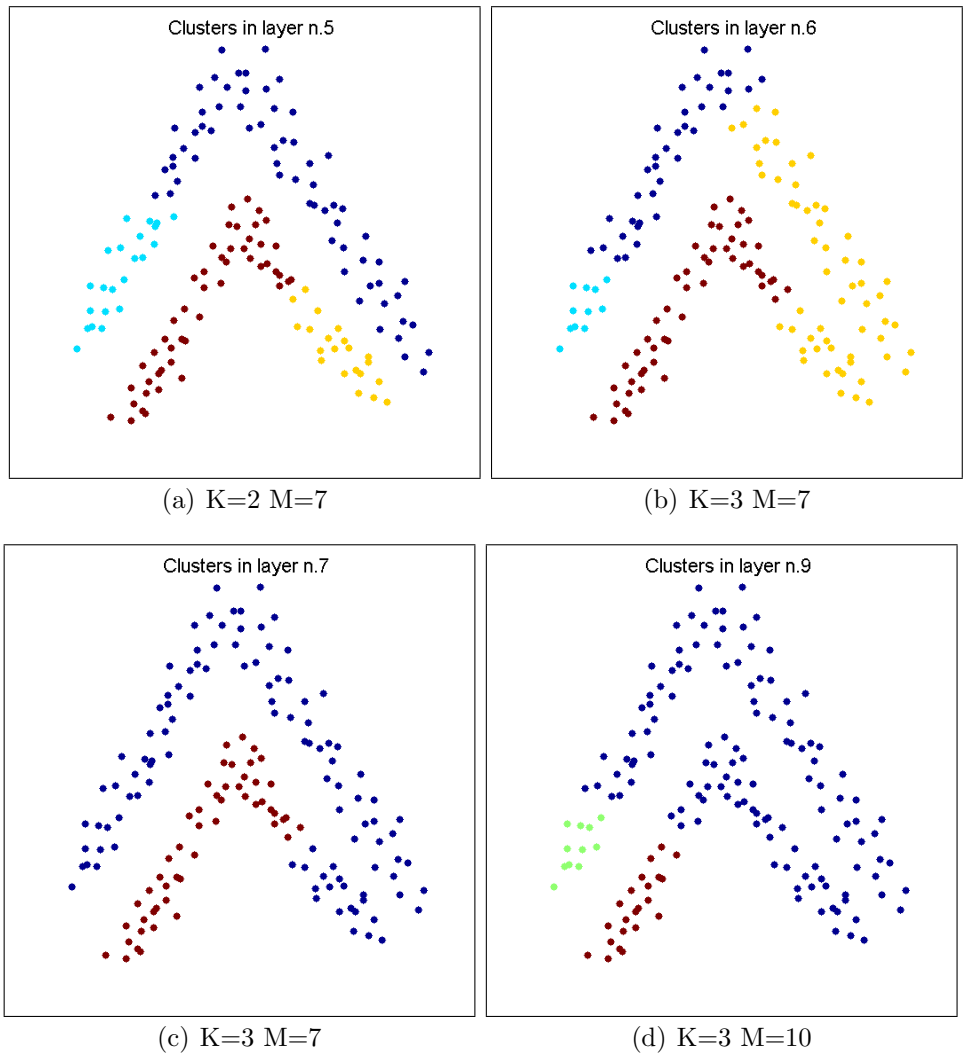
(a) K=2 M=6 (b) K=3 M=10

(c) K=3 M=8 (d) K=3 M=8

Fig. 4.5: Resultados obtidos com os conjuntos Epiglottis e Eye usando o LEGClust.

A figura 4.5 representa os resultados obtidos com os conjuntos Epiglottis e Eye. Os valores usados de K mantiveram-se 2 e 3. No conjunto Epiglottis é possível distinguir dois *clusters* enquanto que no conjunto Eye três *clusters* distintos podem ser identificados. Na figura 4.5 (a) e (b) estão representados os *clusters* formados com K igual a 2 e 3 respetivamente. Em (c) esta representada uma solução do conjunto Eye com três *clusters* e em (d) uma solução com dois *clusters* usando os mesmos parâmetros. Em ambos os conjuntos, os resultados obtidos traduzem a estrutura dos dados. Em (c) foram encontrados com sucesso três *clusters* na camada 7 e em (d) dois *clusters* como seriam escolhidos por inspeção visual, sendo um *cluster* em forma de circulo ao centro e um *cluster* a volta em forma de anel.

No conjunto Epiglottis a mudança de parâmetros obteve os mesmos resultados.



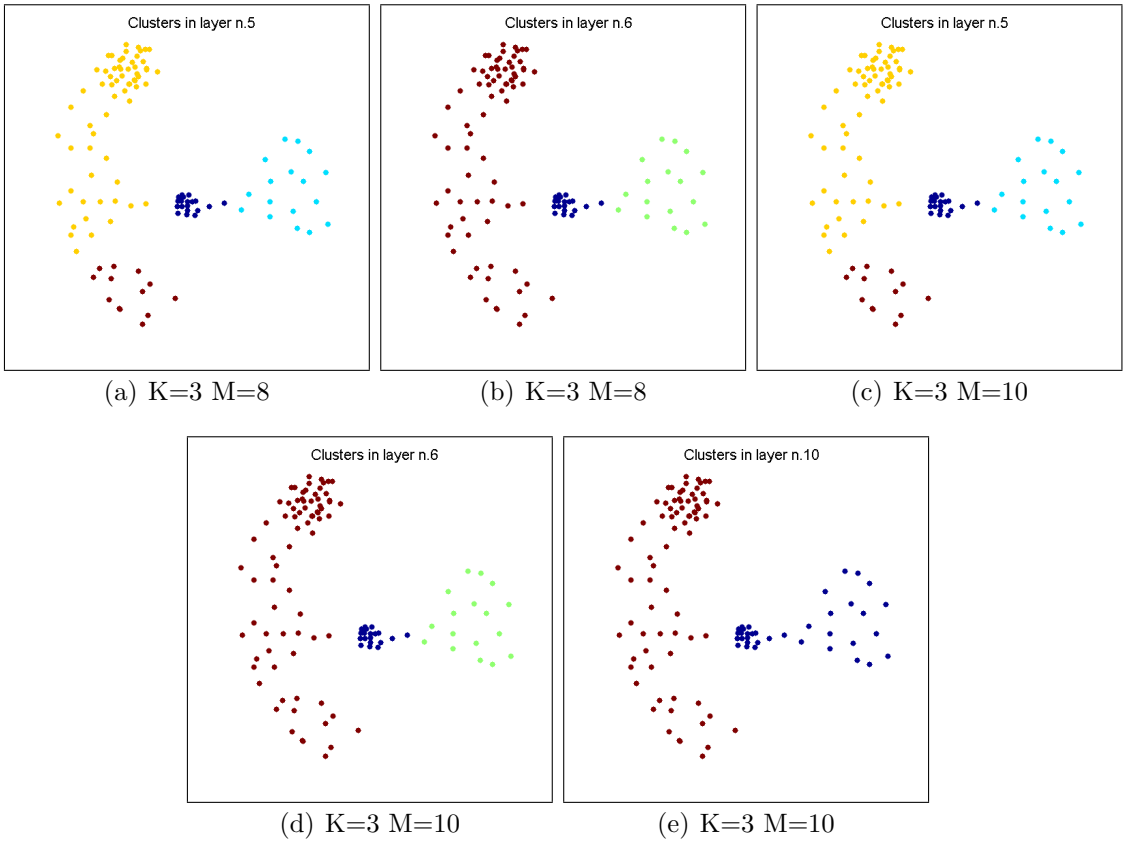
(a) K=2 M=7 (b) K=3 M=7

(c) K=3 M=7 (d) K=3 M=10

Fig. 4.6: Resultados obtidos com o conjunto Citroen usando o LEGClust.

A figura 4.6 representa os resultados obtidos com o conjunto Citroen. Neste conjunto é possível identificar dois aglomerados de pontos distintos, dispostos paralelamente. Na figura 4.5 (a) e (b) encontram-se as soluções com quatro *clusters* usando $K = 2$ e 3 respetivamente. O número de vizinhos manteve-se igual. Em (c) está representada a solução com dois *clusters* usando 7 vizinhos e em (d) uma solução com três *clusters* usando 10 vizinhos. Dos quatro resultados apresentados, nenhum representa a estrutura dos dados. O conjunto em (c) está mais próximo dos *clusters* ideais. Comparando (a) com (b) é possível notar a diferença no *cluster* amarelo. Com $K = 3$ é englobado no *cluster* amarelo um grupo de pontos que não corresponde à disposição dos dados. Consequentemente em (c), que corresponde à

camada seguinte à camada apresentada em (b), são formados dois *clusters*, onde o *cluster* castanho deveria englobar o pequeno grupo de pontos azul, fazendo deste modo os dois *clusters* ideais esperados.



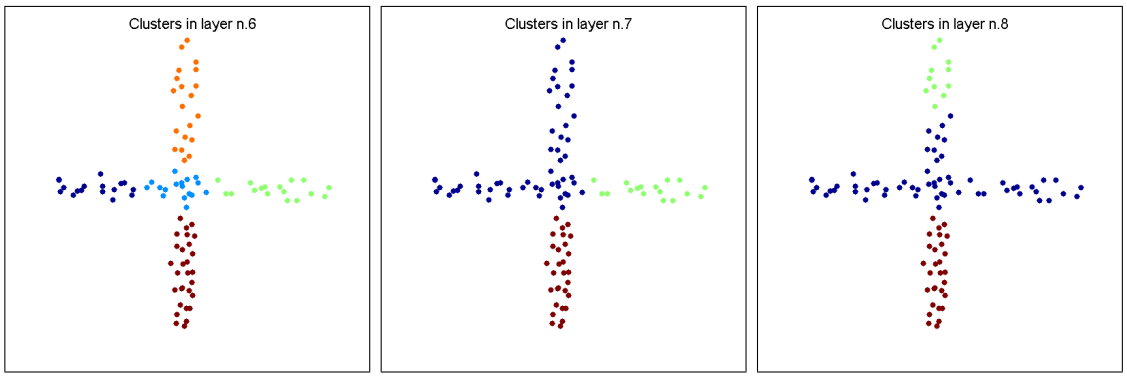
(a) K=3 M=8 (b) K=3 M=8 (c) K=3 M=10

(d) K=3 M=10 (e) K=3 M=10

Fig. 4.7: Resultados obtidos com o conjunto Anchor usando o LEGClust.

A figura 4.7 representa os resultados obtidos com o conjunto Anchor. Este conjunto possui aglomerados de pontos ligados entre si. Foi usado um valor de 3 para K e foi variado o valor dos vizinhos M. Na figura 4.7 (a) e (b) são usados 8 vizinhos, formando quatro *clusters* em (a) e três *clusters* em (b). Em (c), (d) e (e) são usados 10 vizinhos, sendo que em (e) é proposta uma solução com apenas dois *clusters*. No caso deste conjunto os resultados obtidos refletem a estrutura de dados até mesmo quando são encontrados quatro *clusters*. Para K igual a 3 com M igual a 8, o resultado de dois *clusters* não foi possível ser obtido. Dado que em (e) dois *clusters* são obtidos na camada 10, sendo que em (b) M é igual a 8, apenas são processadas 8 camadas, não sendo camadas suficientes para processar os *clusters* que se formariam em camadas mais elevadas, como por exemplo, a camada 10 em (e). Aumentando o valor de M de 8 para 10, permitiu que fossem encontrados dois

clusters, fazendo assim uma nova solução possível.



(a) K=3 M=8 (b) K=3 M=8 (c) K=3 M=10

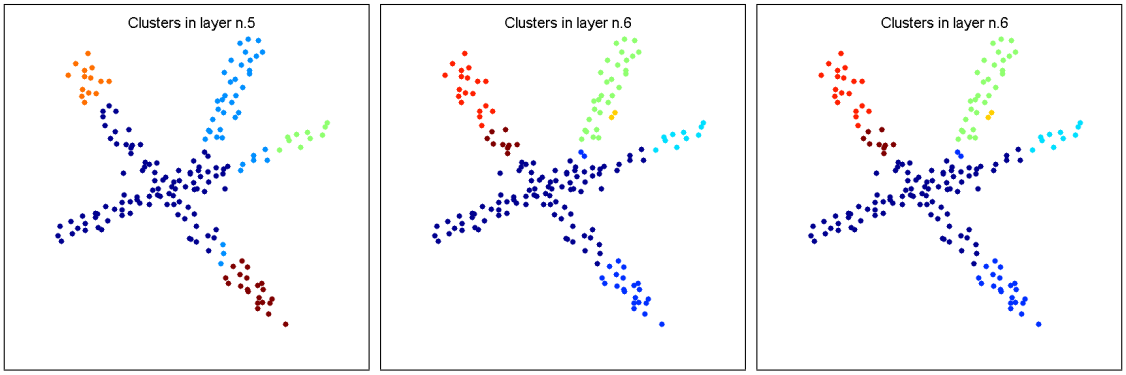
Fig. 4.8: Resultados obtidos com o conjunto Cross usando o LEGClust.

Os resultados obtidos para o conjunto Cross estão representados na figura 4.8. Em (a) são propostos cinco *clusters*, em (b) são propostos três *clusters* usando 8 vizinhos e em (c) são propostos também três *clusters* aumentando o número de vizinhos para 10. Este aumento influenciou os *clusters* encontrados em (c) que são distintos dos *clusters* em (b). Idealmente a estrutura dos dados no conjunto Cross pode ser representada por quatro grupos de pontos.

O algoritmo LEGCLust implementado não encontrou nenhuma solução com apenas quatro *clusters*. Foram encontradas soluções com cinco e três *clusters*. A solução com cinco *clusters* em (a) representa a estrutura dos dados e pode-se considerar um resultado aceitável. O resultado em (b) com três *clusters* pode também ser considerado aceitável. Em (c) M é aumentado, alterando a forma como os três *clusters* são encontrados, apesar de não refletir corretamente a estrutura dos dados, o resultado aparenta ser válido.

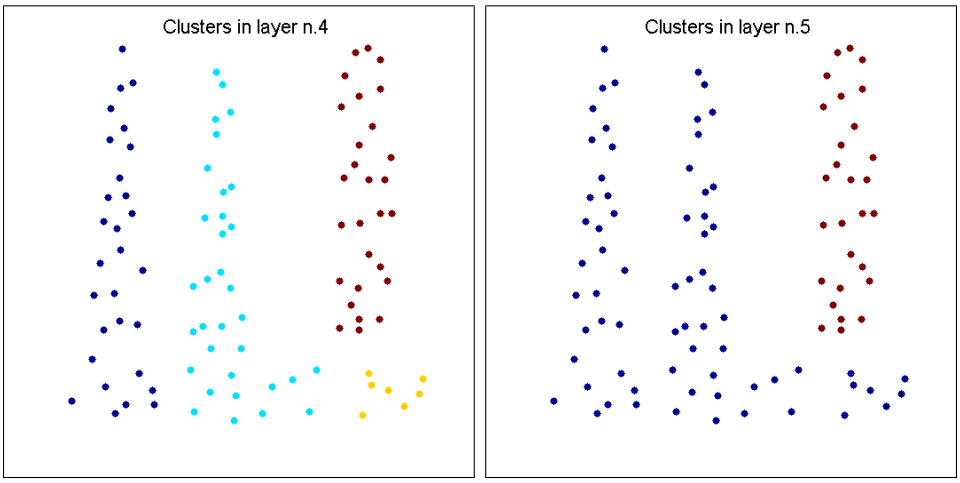
A figura 4.9 mostra os resultados obtidos com o conjunto Starfish. Em (a) foram encontrados cinco *clusters* usando $K = 2$ e $n = 8$ vizinhos, enquanto que em (b) usando os mesmos vizinhos mas variando K para 3 foram encontrados sete *clusters*. Em (c) foi aumentado o número de vizinhos para 10, mantendo o valor de K, o resultado manteve-se o mesmo. Os resultados obtidos com este conjunto de dados são pouco favoráveis.

Na figura 4.10 estão representados os *clusters* encontrados no conjunto Anthills usando $K = 2$ e $n = 6$ vizinhos. São representados quatro *clusters* em (a) e dois *clusters* em (b). Nesta figura estão demonstrados os melhores resultados obtidos com K igual a 2 para este conjunto de dados. Os *clusters* ideais a serem encon-



(a) K=2 M=8 (b) K=3 M=8 (c) K=3 M=10

Fig. 4.9: Resultados obtidos com o conjunto Starfish usando o LEGClust.



(a) K=2 M=6 (b) K=2 M=6

Fig. 4.10: Resultados obtidos com o conjunto Anthills usando o LEGClust.

trados seriam três, no entanto o algoritmo apenas encontrou soluções com quatro e dois *clusters*. A solução representada em (a) contém quatro *clusters*, que através duma inspeção visual, a solução aparenta ser válida dado que os *clusters* estão bem diferenciados. De um modo geral, a implementação do algoritmo LEGClust demonstrou melhores resultados com o conjunto de dados artificial do que o algoritmo *Minimum Entropy Clustering*. A diferença é notória nos conjuntos de dados Kites, Eye e Anthills.

4.1.2 Testes com dados reais

Os testes com dados reais foram efetuados apenas com os algoritmos baseados em critérios entrópicos. Os resultados são avaliados pelo índice ARI e comparados com

os resultados obtidos com o algoritmo k-médias.

Tab. 4.1: Resultados exploratórios obtidos com os conjuntos de dados reais.

	LEGClust K=2 M=10	LEGClust K=3 M=12	Minimum Entropy Clustering MATLAB	minCentropy	k-médias
Breast Cancer	-0.0103	-0.0151	0.8392	0.8345	0.8285
Ecoli	0.0379	0.0379	0.5095	0.3424	0.4696
Glass	0.1141	0.0833	0.2427	0.1956	0.2515
Haberman	-0.0042	0.0049	-0.0018	-0.0008	-0.0001
Iris	0.4531	0.4531	0.5556	0.6007	0.6201
Segmentation	0.1076	0.0178	0.1444	0.1683	0.23377
Wine	0.1975	0.0059	0.8975	0.9326	0.8975

Na tabela 4.1 são demonstrados os resultados dos algoritmos com critérios entrópicos obtidos com conjuntos de dados reais, avaliados segundo o índice ARI. Valores mais próximos de 1 são considerados bons. O índice ARI toma valores entre 0 e 1, no entanto inesperadamente vários resultados apresentam-se com valor negativo, são casos onde o resultado de *clustering* não corresponde minimamente às classes conhecidas à priori. Estão realçados os melhores valores obtidos para cada conjunto de dados. Para o algoritmo *LEGClust* foram usados os parâmetros de $K = 2$ com $M = 10$ e $K = 3$ com $M = 12$. Em [27] os autores sugerem usar 10% do número total de objetos contido nos dados como valor para M, no entanto, através dos testes feitos ao algoritmo com os dados bidimensionais, verificou-se que valores muito acima dos escolhidos, produziam resultados iguais. Pelo que foram escolhidos os valores que produziam resultados aceitáveis e com os quais o algoritmo demorava menos tempo a processar.

Para o algoritmo *Minimum Entropy Clustering* foram usados os parâmetros de $\sigma = 1$ e $\alpha = 1$. Com o algoritmo *minCentropy* foram usados os parâmetros de $\sigma = 1$ e $\alpha = 1.5$. O algoritmo k-médias obteve os melhores resultados. O algoritmo *LEGClust* obteve alguns resultados positivos com $K = 2$. Os conjuntos Iris e Whine de um modo geral obtiveram bons resultados, em contraste com o conjunto Haberman que na maioria dos algoritmos teve resultados negativos. O que demonstra que nalguns conjuntos de dados é mais fácil obter bons resultados de *clustering*. Os algoritmos *Minimum Entropy Clustering*, *minCentropy* e o k-médias, obtiveram resultados relativamente bons com os dados reais. A implementação do algoritmo *LEGClust* obteve resultados inesperados e menos bons. Em [27] alguns dos dados reais são também usados e são obtidos resultados bastante melhores.

4.2 Aplicação a dados Biomédicos

As experiências com dados biomédicos foram efetuadas após os testes exploratórios com o conjunto de dados artificiais e o conjunto de dados reais. Os valores definidos para os parâmetros durante as experiências de teste foram utilizados também para as experiências com o conjunto de dados biomédicos.

Tab. 4.2: Resultados (índice ARI) da aplicação dos algoritmos baseados em critérios entrópicos e do algoritmo k-médias em dados biomédicos.

	LEGClust K=2 M=10	LEGClust K=3 M=12	Minimum Entropy Clustering MATLAB	minCentropy	k-médias
Acath	0.0000	-0.0639	0.0202	0.1147	0.0081
Asthma	0.0159	0.0159	0.0211	0.0417	0.0201
Babies	-0.0045	-0.0051	-0.0039	0.0182	-0.0039
Breast cancer W.	-0.0201	0.0052	0.6471	0.4499	0.6909
Column 2C	-0.0621	-0.0066	0.0811	0.1625	0.1056
Column 3C	-0.0249	-0.0087	0.2506	0.2169	0.2754
Column full	0.0119	0.0017	0.1156	0.1406	0.1258
Depress	0.0000	0.0016	-0.0065	0.0406	-0.0037
Diabetes	0.0000	-0.0004	0.0634	0.0538	0.0647
Dmd	0.0361	0.0853	0.1378	0.1273	0.1991
Dominican	-0.0845	-0.0084	-0.0085	-0.0075	-0.0085
Gbsg	-0.0087	0.0046	0.0447	0.0431	0.0433
Growth	0.0698	0.0618	0.0622	0.0074	0.0659
Lung cancer-uci	0.2185	0.2185	0.2235	0.0259	0.2235
Lung cancer-ucla	-0.0041	-0.0032	0.0072	0.0072	0.0071
Lung function	0.0000	-0.0104	0.0262	0.0212	0.0261
Pbc	0.0045	-0.0046	0.2725	0.2528	0.2725
Prostate	0.0000	0.0029	-0.0003	0.0058	-0.0003
Spectf	-0.0242	-0.0125	-0.1031	-0.0869	-0.1031
Stress echo	-0.0099	-0.0035	-0.0221	-0.0055	-0.0221
Surgery	0.0009	0.0009	0.0248	0.0185	0.0248
Weights	0.0015	0.0063	0.0008	0.0036	0.0011
Xray	0.0000	-0.0291	-0.0078	0.3405	-0.0066

Na tabela 4.2 estão demonstrados os resultados obtidos com os algoritmos baseados em critérios entrópicos. Os resultados obtidos com o algoritmo k-médias estão também incluídos para comparação. OS melhores valores obtidos com cada conjunto são realçados. Tal como nos testes com o conjunto de dados reais, os algoritmos *Minimum Entropy Clustering*, *minCentropy* e o k-médias obtiveram maior número de

valores acima de 0.1. Grande quantidade dos resultados são valores negativos. Apenas dois valores obtidos com o algoritmo *LEGClust* são acima de 0.1. Os algoritmos *Minimum Entropy Clustering* e *minCentropy* fazem uso do k-médias para obter uma partição inicial, este processo pode explicar os valores aproximados que estes três algoritmos obtiveram.

Tab. 4.3: Resultados (índice ARI) da aplicação dos algoritmos de spectral clustering e do algoritmo mean shift em dados biomédicos.

	Shi Malik	Ng Weiss	Perona Freeman	Mean Shift	k-médias
Acath	0.0062	-0.0164	-0.0164	0.0000	0.0081
Asthma	-0.0037	0.0416	0.0416	0.0211	0.0211
Babies	0.0004	-0.0385	0.0000	0.0000	-0.0039
Breast cancer W.	0.0043	-0.0014	0.0026	0.1716	0.6918
Column 2C	0.0004	0.0344	-0.0007	0.0013	0.1056
Column 3C	0.0009	0.1491	0.0003	0.0084	0.2753
Column full	0.0069	0.0578	0.0003	0.0472	0.1257
Depress	0.0003	-0.0015	0.0611	0.0355	-0.0036
Diabetes	0.0000	0.0043	0.0006	-0.0001	0.0647
Dmd	0.0096	0.0085	-0.0016	0.0189	0.1991
Dominican	0.0168	-0.0015	0.0031	-0.0022	-0.0084
Gbsg	-0.0003	0.0259	-0.0004	0.0002	0.0433
Growth	0.0004	0.0025	0.0000	0.0275	0.0659
Lung cancer uci	0.2961	0.0744	0.0744	-0.0042	0.2235
Lung cancer ucla	0.0038	0.0072	0.0072	0.1431	0.0071
Lung function	0.0053	0.0949	0.0000	-0.0077	0.0262
Pbc	0.0042	0.0253	0.0142	0.0255	0.2725
Prostate	0.0143	0.0007	0.0031	-0.0034	-0.0003
Spectf	0.0228	0.0026	-0.0025	-0.0805	-0.1031
Stress echo	0.0011	-0.0021	-0.0006	-0.0188	-0.0221
Surgery	-0.0003	0.0037	-0.0038	-0.0111	0.0247
Weights	0.0008	-0.0004	0.0003	-0.0051	0.0011
Xray	-0.0098	0.3204	0.3204	-0.0294	-0.0066

A tabela 4.3 mostra os resultados obtidos com os algoritmos de *spectral clustering* e o algoritmo de Mean Shift. Os algoritmos de *spectral clustering* bem como o algoritmo de *Mean Shift* obtiveram resultados muito pouco favoráveis. Mais uma vez a grande maioria dos valores apresentam-se negativos e muito próximos de zero.

Na tabela 4.4 é feita a comparação entre os resultados obtidos com a implementação em MATLAB do algoritmo *Minimum Entropy Clustering* em relação à sua implementação em JAVA. Os resultados do algoritmo k-médias são também incluídos

Tab. 4.4: Resultados (índice ARI) da aplicação das diferentes implementações em MATLAB e JAVA do algoritmo Minimum Entropy Clustering e do algoritmo k-médias em dados biomédicos.

	Minimum Entropy Clustering MATLAB	Minimum Entropy Clustering JAVA	k-médias
Acath	0.0202	0.0067	0.0081
Asthma	0.0211	0.0012	0.0201
Babies	-0.0039	-0.0037	-0.0039
Breast cancer W.	0.6471	0.6616	0.6909
Column 2C	0.0811	0.1873	0.1056
Column 3C	0.2516	0.1581	0.2754
Column full	0.1155	0.0537	0.1257
Depress	-0.0065	-0.0081	-0.0036
Diabetes	0.0633	-0.0087	0.0647
Dmd	0.1377	0.0934	0.1991
Dominican	-0.0084	-0.0282	-0.0084
Gbsg	0.0447	0.0228	0.0433
Growth	0.0622	0.0865	0.0659
Lung cancer uci	0.2234	0.1217	0.2235
Lung cancer ucla	0.0071	0.1031	0.0072
Lung function	0.0261	0.0351	0.0262
Pbc	0.2725	-0.0078	0.2725
Prostate	-0.0003	0.5271	-0.00033
Spectf	-0.1031	-0.1083	-0.1031
Stress echo	-0.0221	0.0039	-0.0221
Surgery	0.0247	-0.0061	0.0248
Weights	0.0008	0.0131	0.0011
Xray	-0.0078	-0.0294	-0.0066

para comparação. Os resultados obtidos na implementação em JAVA são idênticos aos resultados obtidos com a implementação em MATLAB. Em comparação com as duas implementações, o k-médias obtém melhores resultados. Por último, na tabela 4.5 são mostrados todos os resultados obtidos com os diferentes algoritmos de modo a facilitar a comparação entre os mesmos.

Em relação ao conjunto de dados biomédico, nota-se uma tendência nos resultados de vários dados em que mais de metade dos valores obtidos são abaixo de 0.1, onde parte destes valores são muito próximos de zero e outra parte são negativos. De um modo geral, com a aplicação dos algoritmos à maioria dos conjuntos de dados são obtidos valores negativos inesperados. A implementação dos algoritmos em

MATLAB pode estar na origem de valores inesperados. Esta foi feita a partir da descrição dos algoritmos pelos autores, pelo que a forma de processamento não é igual, sendo apenas baseada na descrição. A hipótese da implementação não estar o mais correta não deve ser ignorada. No entanto, mediante os resultados apresentados, o algoritmo k-médias obteve melhores resultados com dados biomédicos, seguido dos algoritmos *Minimum Entropy Clustering* e *minCentropy*.

Relembrando os testes feitos com dados artificiais bidimensionais, os resultados apontavam para uma superioridade do algoritmo LEGClust. Com o conjunto de dados real usado também nos testes e o conjunto de dados biomédicos, os algoritmos *Minimum Entropy Clustering* e *minCentropy* demonstram obter melhores resultados que a implementação do algoritmo LEGClust. De todos os algoritmos usados, o algoritmo k-médias em conjunto com os algoritmos *Minimum Entropy Clustering* e *minCentropy* apresentam melhores resultados. O algoritmo LEGCLust apresenta melhores resultados que os algoritmos de *spectral clustering*. Em conjuntos de dados muito grandes, o LEGClust e o *Minimum Entropy Clustering* demonstraram necessitar de maior tempo computacional, enquanto que o k-médias, os de *spectral clustering* e o *minCentropy* necessitam de menos tempo de processamento.

Tab. 4.5: Resultados (índice ARI) da aplicação dos diferentes algoritmos usados em dados biomédicos.

	LEGClust K=2 M=10	LEGClust K=3 M=12	MEC MATLAB	MEC JAVA	minCentropy	k-médias	Shi Malik	Ng Weiss	Perona Freeman	Mean Shift
Acath	0.0000	-0.0639	0.0202	0.0067	0.1147	0.0081	0.0062	-0.0164	-0.0164	0.0000
Asthma	0.0159	0.0159	0.0211	0.0012	0.0417	0.0201	-0.0037	0.0416	0.0416	0.0211
Babies	-0.0045	-0.0051	-0.0039	-0.0037	0.0182	-0.0039	0.0004	-0.0385	0.0000	0.0000
Breast cancer W.	-0.0201	0.0052	0.6471	0.6616	0.4499	0.6909	0.0043	-0.0014	0.0026	0.1716
Column 2C	-0.0621	-0.0066	0.0811	0.1873	0.1625	0.1056	0.0004	0.0344	-0.0007	0.0013
Column 3C	-0.0249	-0.0087	0.2506	0.1581	0.2169	0.2754	0.0009	0.1491	0.0003	0.0084
Column full	0.0119	0.0017	0.1156	0.0537	0.1406	0.1258	0.0069	0.0578	0.0003	0.0472
Depress	0.0000	0.0016	-0.0065	-0.0081	0.0406	-0.0037	0.0003	-0.0015	0.0611	0.0355
Diabetes	0.0000	-0.0004	0.0634	-0.0087	0.0538	0.0647	0.0000	0.0043	0.0006	-0.0001
Dmd	0.0361	0.0853	0.1378	0.0934	0.1273	0.1991	0.0096	0.0085	-0.0016	0.0189
Dominican	-0.0845	-0.0084	-0.0085	-0.0282	-0.0075	-0.0085	0.0168	-0.0015	0.0031	-0.0022
Gbsg	-0.0087	0.0046	0.0447	0.0228	0.0431	0.0433	-0.0003	0.0259	-0.0004	0.0002
Growth	0.0698	0.0618	0.0622	0.0865	0.0074	0.0659	0.0004	0.0025	0.0000	0.0275
Lung cancer-uci	0.2185	0.2185	0.2235	0.1217	0.0259	0.2235	0.2961	0.2961	0.0744	-0.0042
Lung cancer-ucla	-0.0041	-0.0032	0.0072	0.1031	0.0072	0.0071	0.0038	0.0072	0.0072	0.1431
Lung function	0.0000	-0.0104	0.0262	0.0351	0.0212	0.0261	0.0053	0.0949	0.0000	-0.0077
Pbc	0.0045	-0.0046	0.2725	-0.0078	0.2528	0.2725	0.0253	0.0253	0.0142	0.0255
Prostate	0.0000	0.0029	-0.0003	0.5271	0.0058	-0.0003	0.0143	0.0007	0.0031	-0.0034
Spectf	-0.0242	-0.0125	-0.1031	-0.1083	-0.0869	-0.1031	0.0228	0.0026	-0.0025	-0.0805
Stress echo	-0.0099	-0.0035	-0.0221	0.0039	-0.0055	-0.0221	0.0011	-0.0021	-0.0006	-0.0188
Surgery	0.0009	0.0009	0.0248	-0.0061	0.0185	0.0248	-0.0003	0.0037	-0.0038	-0.0111
Weights	0.0015	0.0063	0.0008	0.0131	0.0036	0.0011	0.0008	-0.0004	0.0003	-0.0051
Xray	0.0000	-0.0291	-0.0078	-0.0294	0.3405	-0.0066	-0.0098	0.3204	0.3204	-0.0294

Capítulo 5

Conclusão

A tarefa de *clustering* demonstra-se muito relevante em áreas como o reconhecimento de padrões e mineração de dados. É feita com diferentes objetivos, como a segmentação de imagens ou a análise de genes. Existem várias técnicas de *clustering* e ainda maior número de algoritmos que implementam estas técnicas. Das diferentes técnicas existentes, como o método hierárquico, *clustering* por densidade, teoria de grafos ou o uso de vetores próprios para encontrar *clusters*, destaca-se a entropia.

O objetivo deste trabalho foi testar alguns algoritmos que usam a entropia, para fazer *clustering* de dados biomédicos. Foram testados, em 24 conjuntos de dados, três algoritmos baseados em critérios entrópicos e comparados com ouros algoritmos baseados em diferentes critérios. Antes das experiências com dados biomédicos, foram feitas algumas experiências com um conjunto de dados de teste para avaliar a implementação de certos algoritmos. Os algoritmos LEGClust e *Minimum Entropy Clustering* foram implementados em MATLAB e sujeitos a testes, de modo a validar a sua implementação.

Analisando os resultados obtidos com o conjunto de dados artificiais, é possível afirmar que o algoritmo LEGClust usado, consegue identificar *clusters* com elevada precisão em dados com formas não muito complicadas. O algoritmo *Minimum Entropy Clustering* consegue também encontrar *clusters* em dados com formas bem definidas, onde os grupos naturais de pontos se encontram bem aglomerados e espaçados. Em relação às experiências de teste com dados reais, é possível concluir que os algoritmos *Minimum Entropy Clustering*, *minCentropy* e o k-médias obtêm bons resultados. O algoritmo LEGClust obteve também resultados favoráveis nos conjuntos de dados reais quando o parâmetro K possuía o valor de 2. Ainda sobre os dados reais, o conjunto Iris e o conjunto Segmentation, obtiveram bons resultados

com todos os algoritmos, em contraste com o conjunto Haberman em que nenhum dos resultados obtidos foi favorável, obtendo até alguns valores negativos.

Analisando os resultados obtidos com o conjunto de dados biomédicos, é possível afirmar que os algoritmos *Minimum Entropy Clustering*, *minCentropy* e o k-médias obtiveram os melhores resultados. Os restantes algoritmos obtiveram resultados muito pouco favoráveis, sendo a maioria deles muito próxima de zero.

Em relação aos algoritmos, os de critérios entrópicos mostraram-se muito paramétricos, isto é, necessitam da definição de vários parâmetros para o seu funcionamento. Estes parâmetros alteram o resultado, pelo que a escolha certa dos parâmetros adequados é necessária. Algoritmos como o k-médias e os algoritmos de *spectral clustering*, apenas possuem um ou dois parâmetros, aumentando a sua facilidade de interação. Os algoritmos com critérios entrópicos, devido à sua complexidade, mostraram também necessitar de maior tempo de processamento quando comparados com o algoritmo k-médias e os algoritmos de *spectral clustering*.

Em relação aos conjuntos de dados e aos valores obtidos, os dados bidimensionais mostraram-se muito úteis na avaliação dos algoritmos, pois através da inspeção visual dos mesmos foi relativamente fácil obter uma ideia do resultado final esperado e puderam ser facilmente representados graficamente. Com os dados reais e os dados biomédicos, é necessário o uso do índice de performance para avaliar o resultado. Com estes últimos é também necessário saber informação sobre as classes dos dados à priori para os poder avaliar, requisito que não é necessário com dados bidimensionais.

Os valores obtidos com os dados reais e biomédicos (especialmente com os dados biomédicos) são inesperados. Grande quantidade de valores apresenta-se muito perto de zero e alguns até negativos. Este acontecimento pode ser justificado pelos próprios dados ou até pelos algoritmos usados. Podem ser conjuntos onde não acontecem agrupamentos naturais entre os dados, como exemplo, no conjunto de dados reais usado nas experiências de teste, o conjunto Iris obteve bons resultados com todos os algoritmos, já o conjunto Haberman apenas obteve resultados negativos e muito perto de zero. A implementação dos algoritmos influencia também os resultados obtidos, pelo que a hipótese de erros de implementação pode estar na fonte de valores inesperados.

Em suma, foram testados dados biomédicos com diferentes algoritmos. Foram usados algoritmos com critérios entrópicos, algoritmos de *spectral clustering*, um algoritmo de densidade e o algoritmo k-médias. O algoritmo k-médias apresentou os melhores resultados. Dos algoritmos entrópicos usados, nomeadamente, o LEGClust, o *Minimum Entropy Clustering* e o *minCentropy*, apenas os dois últi-

mos apresentaram bons resultados com dados biomédicos. Os algoritmos de *spectral clustering* e o algoritmo de densidade *Mean Shift* apresentaram alguns resultados bons. A eleição de um dos algoritmos usados como o melhor para analisar dados biomédicos é bastante subjetivo, no entanto, com base nos resultados obtidos, o uso de algoritmos baseados em critérios entrópicos em dados biomédicos não é favorável. O algoritmo k-médias obteve um grande número de melhores resultados e tendo em conta certos pormenores como, a facilidade de implementação, o pouco tempo de processamento necessário e a inserção de apenas um parâmetro, fazem deste algoritmo a escolha mais viável para analisar dados biomédicos, em comparação com os restantes métodos testados.

Bibliografia

[1] J. Liu, J. Sun, and S. Wang. Pattern recognition: An overview. *International Journal of Computer Science and Network Security*, 6(6), June 2006.

[2] U.Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17(3), 1996.

[3] A. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 2010.

[4] A. Jain, M. Law, and J. Buhmann. Landscape of clustering algorithms. *Proc. IAPR International Conference on Pattern Recognition*, 2004.

[5] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1(2), 1997.

[6] L. Kaufman and P. Rousseeuw. Finding groups in data : An introduction to cluster analysis. *Wiley Series in Probability and Statistics*, 2005.

[7] S. Guha, R. Rastogi, and K. Shim. Rock: A robust clustering algorithm for categorical attributes. *Proceedings of the 15th International Conference on Data Engineering*, 2000.

[8] S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases. *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, 1998.

[9] G. Karypis, H. Eui-Hong, and V. Kumar. Chameleon: A hierarchical clustering algorithm using dynamic modeling. *IEEE Computer*, 1999.

[10] G. Ball and D. Hall. Isodata: a novel method of data anlysis and pattern classification. *Technical Report for Stanford Research Institute*, 1965.

[11] E. Forgy. Cluster analysis of multivariate data: efficiency vs interpretability of classifications. *Biometrics*, 1965.

[12] J. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of cybernetics*, 1973.

[13] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. *KDD workshop on text mining*, 2000.

[14] D. Pelleg and A. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. *Proceedings of the seventeenth international conference on machine learning*, 2000.

[15] L. Kaufman and P. Rousseeuw. Finding groups in data : An introduction to cluster analysis. *Wiley series in Probability and Statistics*, 2005.

[16] I. Frank and R. Todeschini. Data analysis handbook. *Elsevier Science*, 1994.

[17] F. Cao, M. Ester, W. Qian, and A. Zhou. Density-based clustering over an evolving data stream with noise. *Proceedings of Siam conference in data mining*, 2006.

[18] B. Finkston. Matlab central - mean shift clustering. `http://www.mathworks.com/matlabcentral/fileexchange/10161-mean-shift-clustering`.

[19] R. Real and J. Vargas. The probabilistic basis of jaccard's index of similarity. *Systematic Biology*, 1996.

[20] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 1985.

[21] J. Santos and M. Embrechts. On the use of the adjusted rand index as a metric for evaluating supervised classification. *Proceedings of the 19th International Conference on Artificial Neural Networks: Part II*, 2009.

[22] L. Jing, M. Ng, and J. Huang. An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data. *IEEE Transactions on Knowledge and Data Engeneering*, 19(8), August 2007.

[23] N. Karayiannis. Meca: Manimum entropy clustering algorithm. *IEEE World Congress on Computational Intelligence*, 1, June 1994.

[24] D. Barbará, J. Couto, and Y. Li. Coolcat: An entropy-based algorithm for categorical clustering. *Proceedings of the eleventh international conference on Information and knowledge management*, 1, November 2002.

[25] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. Modha. A generalized maximum entropy approach to bregman coclustering and matrix approximation. *The Journal of Machine Learning Research*, 8, August 2004.

[26] S. Ren and Y. Wang. A proof of the convergence theorem of maximum-entropy clustering algorithm. *Science China Information Sciences*, 53(6), June 2010.

[27] J. Santos, J. Sá, and L. Alexandre. Legclust - a clustering algorithm based on layered entropic subgraphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(1), January 2008.

[28] H. Li, K. Zhang, and T. Jiang. Minimum entropy clustering and applications to gene expression analysis. *Proc. IEEE Computational Systems Bioinformatics Conf.*, 31, 2004.

[29] A. Hero, B. Ma, O. Michel, and J. Gorman. Applications of entropic spanning graphs. *IEEE Signal Processing Magazine*, September 2002.

[30] C. Cheng, A. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, 1999.

[31] R. Jenssen, K. Hild II, D. Erdogmus, J. Príncipe, , and T. Eltoft. Clustering using rényis entropy. *In Proceedings of International Joint Conference on Neural Networks*, 2003.

[32] J. Santos. *Data Classification with Neural Networks and Entropic Criteria*. PhD thesis, Engineering of the University of Porto, Portugal, January 2007.

[33] N. Vinh and J. Epps. mincentropy: A novel information theoretic approach for the generation of alternative clusterings. *IEEE International Conference on Data Mining*, 2010.

[34] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, 2001.

[35] Uci machine learning repository. `http://archive.ics.uci.edu/ml/datasets.html`.

[36] J. Peat and B. Barton. Medical statistics: A guide to data analysis and critical appraisal. `http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0727918125%2CdescCd-DOWNLOAD.html`.

[37] F. Harrell. Department of biostatistics vanderbilt university. `http://biostat.mc.vanderbilt.edu/wiki/Main/DataSets`.

[38] A. Afifi. Biostatistics 406 datasets. `http://www.biostat.ucla.edu/course/406/datasets.htm`.

[39] H. Li. Minimum conditional entropy clustering. `http://alumni.cs.ucr.edu/~hli/mec/index.html`.

[40] X. Nguyen. Matlab central. `http://www.mathworks.com/matlabcentral/fileexchange/32994-the-mincentropy-algorithm-for-alternative-clustering`.

[41] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), August 2000.

[42] P. Perona and W. Freeman. A factorization approach to grouping. *Lecture Notes in Computer Science*, 1998.

[43] A. Ali. Matlab central - spectral clustering algorithms. `http://www.mathworks.com/matlabcentral/fileexchange/26354-spectral-clustering-algorithms`.

[44] Mathworks. `http://www.mathworks.com/help/stats/kmeans.html;jsessionid=c8e4de7f5158010d6bf369323b0d`.

[45] K. Wang. Matlab central - (simple) tool for estimating the number of clusters. `http://www.mathworks.com/matlabcentral/fileexchange/13916-simple-tool-for-estimating-the-number-of-clusters`.