UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

EXPLORANDO CAMINHOS DE MÍNIMA INFORMAÇÃO EM GRAFOS PARA PROBLEMAS DE CLASSIFICAÇÃO SUPERVISIONADA

ALAN KAZUO HIRAGA

ORIENTADOR: PROF. DR. ALEXANDRE LUÍS MAGALHÃES LEVADA

São Carlos - SP
Abril /2014

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

EXPLORANDO CAMINHOS DE MÍNIMA INFORMAÇÃO EM GRAFOS PARA PROBLEMAS DE CLASSIFICAÇÃO SUPERVISIONADA

ALAN KAZUO HIRAGA

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Processamento de Imagens e Sinais: Algoritmos e Arquitetura.
Orientador: Prof. Dr. Alexandre Luís Magalhães Levada

São Carlos - SP
Abril/2014

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

H668ec

Hiraga, Alan Kazuo.
Explorando caminhos de mínima informação em grafos para problemas de classificação supervisionada / Alan Kazuo Hiraga. -- São Carlos : UFSCar, 2014.
77 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2014.

1. Ciência da computação. 2. Reconhecimento de padrões. 3. Teoria dos grafos. 4. Campos aleatórios. 5. Informação de Fisher. 6. Validação cruzada. I. Título.

CDD: 004 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Explorando caminhos de mínima informação em grafos para problemas de classificação supervisionada"

Alan Kazuo Hiraga

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação

Membros da Banca:

Alexandre Luis M. Levada

Prof. Dr. Alexandre Luis Magalhães Levada
(Orientador - DC/UFSCar)

Nelson Delfino d'Ávila Mascarenhas

Prof. Dr. Nelson Delfino d'Ávila Mascarenhas
(PPG-CC/UFSCar)

Prof. Dr. Denis Henrique Pinheiro Salvadeo
(UNESP/Rio Claro)

São Carlos
Maio/2014

"Você pode encarar um erro como uma besteira a ser esquecida, ou como um resultado que aponta uma nova direção".

Steve Jobs

AGRADECIMENTOS

À Deus por cuidar da milha família e me proteger.

À minha mãe Dona Agailda e meu pai Paulo pelo amor, carinho e compreensão em todos os momentos em que estive ausente.

À minha amada e futura esposa Kerlyn pela paciência, carinho, apoio e pelas palavras de incentivo.

Aos meus Irmãos Rodrigo e Jaime, aos meus sobrinhos João e Maria por acreditarem em mim.

A minha sogra Dona Silvia por ser sempre carinhosa e me apoiar.

Aos meus amigos, em especial ao Ricardo Barbosa que foi uma pessoa indispensável para a conclusão deste trabalho; ao Diego Tumitan, Chico, Rodrigo Gallis, Victor Gomes e Lucas Porto.

Ao meu orientador, Prof. Dr. Alexandre Luís Magalhães Levada pelos ensinamentos, atenção, compreensão, paciência, motivação, incentivo e confiança em mim depositado.

Aos professores do mestrado Antônio Prado, Hermes Senger, Luís Trevelin, Marcio Fernandes, Nelson Mascarenhas e Sandra Fabbri.

Aos funcionários do Departamento de Computação da UFSCar.

À CAPES, pelo concessão de bolsa durante todo período de realização deste Mestrado.

Por fim, a todos os amigos, colegas e familiares que me apoiaram nesta caminhada.

RESUMO

A classificação é uma etapa muito importante em reconhecimento de padrões, pois ela tem o objetivo de categorizar objetos a partir de um conjunto de características inerentes a ele, atribuindo-lhe um rótulo. Esse processo de classificação pode ser supervisionado, quando existe um conjunto de amostras de treinamento rotuladas que representam satisfatoriamente as classes, semi-supervisionado, quando o conjunto de amostras é limitado ou quase inexistente, ou não-supervisionado, quando não existem amostras rotuladas. Este trabalho propõe explorar caminhos de mínima informação em grafos para problemas de classificação, por meio da criação de um método de classificação supervisionado, não paramétrico, baseado em grafos, seguindo uma abordagem contextual. Esse método propõe a construção de um grafo a partir do conjunto de amostras de treinamento, onde as amostras serão representadas pelos vértices e as arestas serão as ligações entre amostras pertencentes a uma relação de adjacência. A partir da construção do grafo o método faz o calculo da informação de Fisher Local Observada, uma medida baseada no modelo de Potts, para todos os vértices, identificando o grau de informação que cada um possui. Geralmente vértices de classes distintas quando conectados por uma aresta possuem alta informação (bordas). Feito o calculo da informação, é necessário ponderar as arestas por meio de uma função que penaliza a ligação de vértices com alta informação. Enquanto as arestas são ponderadas é possível identificar e selecionar vértices altamente informativos os quais serão escolhidos para serem vértices protótipos, ou seja, os vértices que definem a região de borda. Depois de ponderadas as arestas e definidos os protótipos, o método propõe que cada protótipo conquiste as amostras oferecendo o menor caminho até ele, de modo que quando uma amostra é conquistada ela receba o rótulo do protótipo que a conquistou, ocorrendo a classificação. Para avaliar o método serão utilizados métodos estatísticos para estimar as taxas de acertos, como K-fold, Hold-out e Leave-one-out Cross-Validation. Os resultados obtidos indicam que o método pode ser um uma alternativa viável as técnicas de classificação existentes.

Palavras-chave: Classificação de Padrões, Teoria dos Grafos, Campos Aleatórios Markovianos, Informação de Fisher e Validação Cruzada.

ABSTRACT

Classification is a very important step in pattern recognition, as it aims to categorize objects from a set of inherent features, through its labeling. This process can be supervised, when there is a sample set of labeled training classes, semi-supervised, when the number of labeled samples is limited or nearly inexistent, or unsupervised, where there are no labeled samples. This project proposes to explore minimum information paths in graphs for classification problems, through the definition of a supervised, non-parametric, graph-based classification method, by means of a contextual approach. This method proposes to construct a graph from a set of training samples, where the samples are represented by vertices and the edges are links between samples that belongs to a neighborhood system. From the graph construction, the method calculates the local observed Fisher information, a measurement based on the Potts model, for all vertices, identifying the amount of information that each sample has. Generally, different class vertices when connected by an edge, have a high information level. After that, it is necessary to weight the edges by means of a function that penalizes connecting vertices with high information. During this process, it is possible to identify and select high information vertices, which will be chosen to be prototype vertices, namely, the nodes that define the classes boundaries. After the definition, the method proposes that each prototype sample conquer the remaining samples by offering the shortest path in terms of information, so that when a sample is conquered it receives the label of the winning prototype, occurring the classification. To evaluate the proposed method, statistical methods to estimate the error rates, such as Hold-out, K-fold and Leave-One-Out Cross-Validation will be considered. The obtained results indicate that the method can be a viable alternative to the existing classification techniques.

Keywords: Pattern Classification, Graph Theory, Markov Random Field, Fisher Information and Cross-Validation.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

K-NN – K - *Nearest Neighbors*

OPF – Optimum-Path Forest

SVM - Support Vector Machines

MAP - Maximum a Posteriori Probability

PCA – Principal Component Analysis

LDA – Linear Discriminant Analysis

SUMÁRIO

Capítulo 1

INTRODUÇÃO

Neste capitulo serão apresentados o contexto e a motivação desse trabalho de pesquisa, bem como a definição do problema. Os objetivos do trabalho também serão expostos e discutidos. Ao final deste capitulo será descrita a organização da dissertação.

1.1 Contexto, Motivação e Definição do problema

Reconhecimento de padrões é uma subárea do aprendizado de máquina. Ela vem sendo empregada em diversas áreas da ciência, como na detecção de objetos, automação, robótica, diagnósticos médicos, sensoriamento remoto, mineração de dados, entre outras. Ela é composta por um conjunto de técnicas que tem como objetivo principal classificar padrões ou informações, tendo como base conhecimentos prévios ou estatísticos (DUDA et al., 2000).

Uma técnica muito importante, ou até a mais importante em reconhecimento de padrões, é a classificação de padrões, que consiste em classificar um conjunto de *"n"* amostras e determinar de maneira correta a qual das *"m"* classes cada amostra pertence. Para isso, de cada amostra são extraídos atributos. Atributos são características inerentes à amostra, por exemplo, se a amostra a ser classificada se trata de uma planta, esses atributos podem ser a altura da planta, grossura do caule, presença de frutos e tamanho do fruto. Normalmente é selecionado um subconjunto das amostras rotuladas, para que o método de classificação possa "aprender" como cada classe se comporta, e encontrar um limiar ou regra de decisão, que é capaz de classificar essas amostras em suas respectivas classes. Esta etapa é chamada aprendizagem (DUDA et al., 2000).

A aprendizagem pode ser classificada como supervisionada, quando todas as amostras do conjunto de treinamento são rotuladas, podendo estimar uma função que classifica cada amostra em sua classe, também chamado de aprendizado indutivo. Não-supervisionada, quando todas as amostras não são rotuladas, esse método procura unir conjuntos de amostras similares dependendo da quantidade de classes existentes. E por fim a semi-supervisionada, nesse caso a grande maioria das amostras não tem rótulos, entretanto existe um pequeno conjunto de amostras rotuladas (PAPA et al., 2009).

Este trabalho tem o seu foco em aprendizado supervisionado, pois esse tipo de aprendizado possibilita a criação de um classificador robusto, além de possibilitar a comparação de resultados utilizando alguns métodos de estimação de taxas de erros, como '*Leave-One-Out Cross-Validation*' e '*Holdout*' (WEBB, 2002).

Classificar padrões não é uma tarefa fácil, e isto se deve principalmente pela sobreposição entre as classes, isto é, existem classes que possuem comportamentos parecidos, desse modo pode haver erros nas regiões de sobreposição (DUDA et al., 2000).

Pensando em minimizar esses erros este trabalho propõe investigar caminhos de mínima informação em grafos para problemas de classificação supervisionada, através do desenvolvimento de um método de classificação contextual baseado em grafos. A abordagem contextual será empregada com o auxilio da teoria dos grafos e modelos Markovianos contextuais.

1.2 Objetivo

Este trabalho tem como objetivo explorar caminhos de mínima informação, por meio da criação de uma abordagem de classificação supervisionada, não paramétrica, baseada em grafos, com o propósito de investigar e delimitar as regiões de fronteiras entre as classes por meio da seleção de protótipos informativos.

1.3 Metodologia de Desenvolvimento do Trabalho

Dado um conjunto de amostras com o seus respectivos atributos e rótulos, a ideia básica do método consiste em construir uma representação topológica das mesmas, gerando assim um grafo. Esse grafo pode ser gerado a partir dos métodos de estimação de vizinhança. Assim pode-se fixar um raio e procurar na vizinhança os *"n"* nodos que estão dentro desse raio/volume ("*E-ball*"), ou fixar uma quantidade *"k"* de vizinhos e procurar os *"k"* nodos mais próximos ("*K-NN*"), de modo que essa relação de adjacência define um modelo contextual para o método.

Com a representação topológica são calculadas as informações contextuais de cada nodo. Esse cálculo advém do modelo de *Potts*, uma representação baseada em campos aleatórios Markovianos (BOYKOV et al., 1998). Com o intuito de quantificar quais as amostras mais informativas do conjunto de treinamento, foi utilizada a informação de Fisher Local Observada, uma métrica baseada em teoria da informação (LEVADA et al., 2008). Cada nodo tem um rótulo atribuído a ele, e nesta fase são escolhidos os protótipos com maiores informações. Por fim as arestas do grafo são ponderadas pela soma das informações dos vértices ligantes. Posteriormente por meio do algoritmo "*Dijkstra multi-source*" (CORMEN et al.,2009), são derivadas árvores de caminhos ótimos, menor caminho que minimiza a Informação de Fisher, com o intuito de classificar as amostras. A motivação para utilizar a informação de Fisher como peso das arestas consiste na observação de que ligações que cruzam bordas são altamente informativas, o que representa uma penalização no cálculo das distâncias entre amostras de classes distintas. Em outras palavras, caminhar extensivamente dentro de um cluster seria bem menos custoso do que atravessar uma fronteira.

1.4 Organização do Trabalho

Esta dissertação está organizada da seguinte forma:

- O Capítulo 2 apresenta o levantamento bibliográfico acerca dos conceitos de Teoria dos grafos, Problemas e algoritmos em grafos, Tipos de

representação, Classificação de padrões e Campos aleatórios Markovianos. Também são descritos trabalhos relacionados à proposta deste projeto de mestrado.

- O capítulo 3 discute o método de classificação proposto, bem como os métodos de validação a serem utilizados como medidas quantitativas de desempenho.
- O capítulo 4 apresenta os resultados e discussões.
- Por fim, o capítulo 5 apresenta as conclusões do trabalho, contribuições e trabalhos futuros.

Capítulo 2

REVISÃO BIBLIOGRÁFICA

2.1 Teoria dos Grafos

A Teoria dos grafos é um ramo da matemática que ganhou notoriedade nos séculos 19 e 20, pois permitiu descrever fenômenos de diversas áreas, por exemplo, desenhar e colorir mapas, infraestrutura de redes de comunicação, planejamento de tarefas e classificação de padrões (STEEN, 2010).

2.1.1 Grafo

Um grafo é descrito como um diagrama que pode representar situações do mundo real. Ele consiste de um conjunto de pontos, chamados de vértices, e linhas que unem os vértices, chamados de arestas, veja a Figura 2-1. A notação de um grafo é proveniente da teoria dos conjuntos (CLARK e HOLTON,1995).

A definição formal de um grafo é dada como segue. Um grafo *G* é uma tupla *G = (V, A)* onde *V* é um conjunto finito não nulo de vértices e *A* é um conjunto finito, que pode ser nulo, de arestas, onde para cada aresta são atribuídas um par de vértices ordenados ou não *(u,v)*, chamados de vértices de extremidade. Como exemplo o grafo da Figura 2.1, em que:

V = {v1,v2,v3,v4,v5}, A = {a1,a2,a3,a4,a5}

a1 ↔ (v1,v5), a2 ↔ (v5,v2), a3 ↔ (v2,v4), a4 ↔ (v4,v3), a5 ↔ (v4,v4)

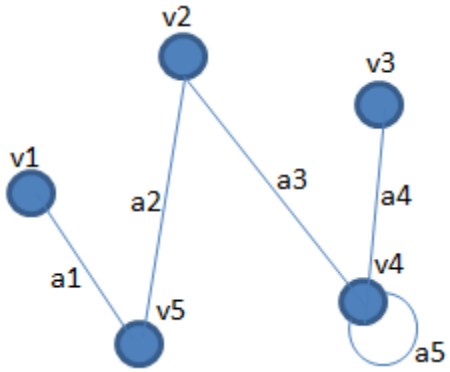


Figura 2-1 - Grafo G=(V,A) (Fonte: adaptado de CLARK e HOLTON,1995)

Nota-se que a aresta *a5* tem os vértices de extremidade idênticos, isso é chamado de laço.

2.1.2 Nomenclaturas e definições

Existem algumas nomenclaturas importantes na teoria dos grafos, aqui serão expostas as mais pertinentes ao trabalho (CLARK e HOLTON,1995).

Grau: é a quantidade de arestas incidentes ao vértice, no grafo da Figura 2-1, os vértices *v1* e *v3* tem grau igual a um, os vértices *v2* e *v5* tem grau igual a dois e o vértice *v4* tem grau igual a quatro.

Grafo ponderado: cada aresta tem um custo.

Grafo direcionado: a aresta direciona (com uma seta) os vértices de extremidade, tornando possível ir de um vértice *v1* para um *v2*, mas não o oposto, conforme a Figura 2-2, por exemplo.

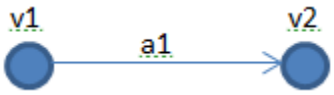


Figura 2-2 - Grafo direcionado (Fonte: CLARK e HOLTON,1995)

Grafo não-direcionado: é possível ir tanto do vértice *v1* para o *v2* quanto o oposto, conforme a Figura 2-3.

Figura 2-3 - Grafo não-direcionado (Fonte: adaptado de CLARK e HOLTON,1995)

Arestas paralelas: são duas arestas que ligam os mesmos vértices, Figura 2-4.

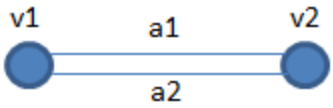


Figura 2-4 - Grafo com arestas paralelas (Fonte: CLARK e HOLTON,1995)

Laço: o vértice é ligado a ele mesmo por uma aresta.

Grafo simples: é um grafo não-direcionado, sem laços, e sem arestas paralelas.

Grafo completo: é um grafo simples, onde cada vértice está ligado aos demais vértices.

Grafo regular: todos os vértices tem o mesmo grau.

Subgrafo: considere os grafos $G = (V_G,A_G)$ e $H = (V_H,A_H)$, H é subgrafo de G se e somente se, $V_H \subseteq V_G$ e $A_H \subseteq A_G$, veja a Figura 2-5.

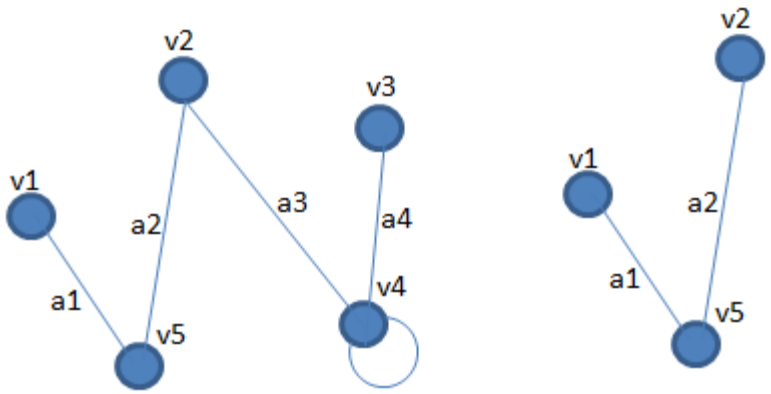


Figura 2-5 - Grafo G e Grafo H (Fonte: adaptado de CLARK e HOLTON,1995)

Caminho: consiste em uma sequência de vértices que são interligados por arestas, onde existe um vértice de origem e um vértice de destino, sem a repetição de vértices, conforme ilustra o exemplo a seguir com a definição do grafo $I = (V_I,A_I)$ onde

$V_I = \{v1,v2,v3,v4\}$ e $A_I = \{(v1,v2),(v1,v3),(v1,v4),(v2,v3),(v2,v4),(v3,v4)\}$

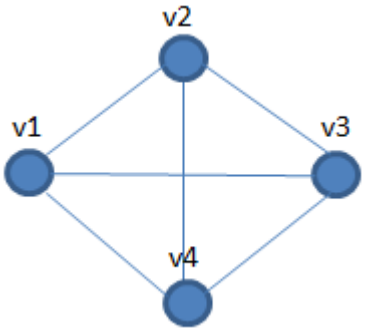


Figura 2-6 - Grafo I (Fonte: adaptado de CLARK e HOLTON,1995)

Os possíveis caminhos do vértice *v1* (origem) ao vértice *v2* (destino) são:

C_1 = {(**v1**,**v2**)}

C_2 = {(**v1**,v3),(v3,**v2**)}

C_3 = {(**v1**,v4),(v4,**v2**)}

C_4 = {(**v1**,v4),(v4,v3),(v3,**v2**)}

Grafo conexo: é um grafo que independente dos vértices existe um caminho entre eles.

Ciclo: é um caminho fechado.

Árvore: é um grafo conexo e sem a presença de ciclos.

Floresta: é um conjunto de árvores.

Árvore geradora: é um subgrafo um grafo G que é uma árvore.

2.2 Problemas e algoritmos em grafos

Nesta seção serão expostos e exemplificados alguns problemas que envolvem grafos. Também serão apresentados algoritmos que resolvem esses problemas.

2.2.1 Árvore geradora mínima

Vamos supor o seguinte problema, exposto em CLARK e HOLTON (1995): certas vilas em uma área necessitam ser unidas a um sistema de abastecimento de água localizado em uma das vilas. O sistema é constituído de dutos que ligam as

torres de água de duas vilas. Para quaisquer duas vilas, sabe-se o quanto custaria para construir um duto para interliga-las. A pergunta que surge é: qual seria a forma mais econômica de construir esse sistema? Para resolver esse problema tem-se que achar uma árvore geradora (pois todos os vértices devem estar obrigatoriamente conectados), e como o custo deve ser considerado, deve-se achar uma árvore geradora onde o custo é mínimo, ou seja, árvore geradora mínima. Existem dois principais algoritmos que encontram a árvore geradora mínima, o Kruskal e o Prim. Para explicá-los considere o grafo ponderado da Figura 2-7.

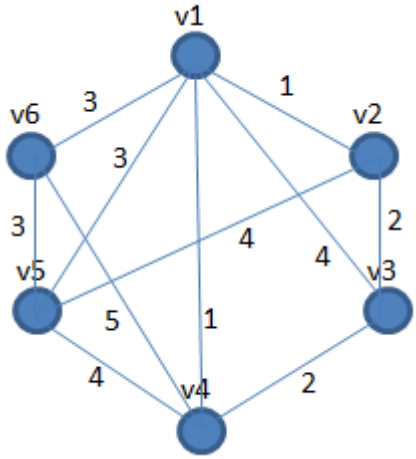


Figura 2-7 - Grafo ponderado (Fonte: CLARK e HOLTON,1995)

2.2.1.1 Kruskal

Definido por Joseph Kruskal em 1956 (KRUSKAL, 1956) é um algoritmo que encontra a árvore geradora mínima de um grafo conexo, ou uma floresta geradora mínima de um grafo não conexo. É um algoritmo do tipo guloso, pois procura resolver o problema de árvore geradora mínima escolhendo sempre a aresta de menor custo a cada passo (escolha ótima local).

O algoritmo começa recebendo por parâmetro um grafo $G = (V, A)$, e tem como saída um grafo $T = (V, A_T)$. No início o conjunto de vértices de T recebe os vértices de G, desse modo T passa a ser uma floresta, depois disso todas as arestas de G são adicionadas a uma fila de prioridade. Esta fila F é ordenada pelos pesos das arestas. Enquanto a quantidade de elementos do conjunto A_T for menor que a quantidade de elementos do conjunto V menos um, o algoritmo retira o primeiro elemento (E) da fila, verifica se o par de vértices, que forma a aresta, pertence à mesma árvore, caso não pertençam, o conjunto de arestas A_T recebe o elemento (E) (CORMEN,2009).

Algoritmo 2.1 – Kruskal (Fonte: adaptado de CORMEN,2009)

Entrada: Grafo G =(V, A);

Saída: Árvore geradora mínima T = (V, A_T);

Estruturas auxiliares: Fila de prioridade F.

1. T.V = G.V // O conjunto dos Vértices da árvore T recebe os Vértices do grafo G, T agora é uma floresta e o conjunto das arestas A_T é inicialmente vazio
2. F = G.A // A Fila recebe todas as arestas
3. F.Ordenar() // A Fila ordena as arestas por ordem crescente
4. **Enquanto** |T.A_T| < |T.V| -1, **Faça** // Enquanto o número de arestas for menor que o número de vértices faça
5. Remova de F o primeiro elemento E // Aresta de menor peso
6. **Se** Busca-Arvore(E.u) ≠ Busca-Arvore(E.v), **Então** // Se os vértices u e v não pertencerem a mesma árvore então
7. T.A = T.A U E // Une as arestas do conjunto T.A com o {E}
8. **Retorne** T

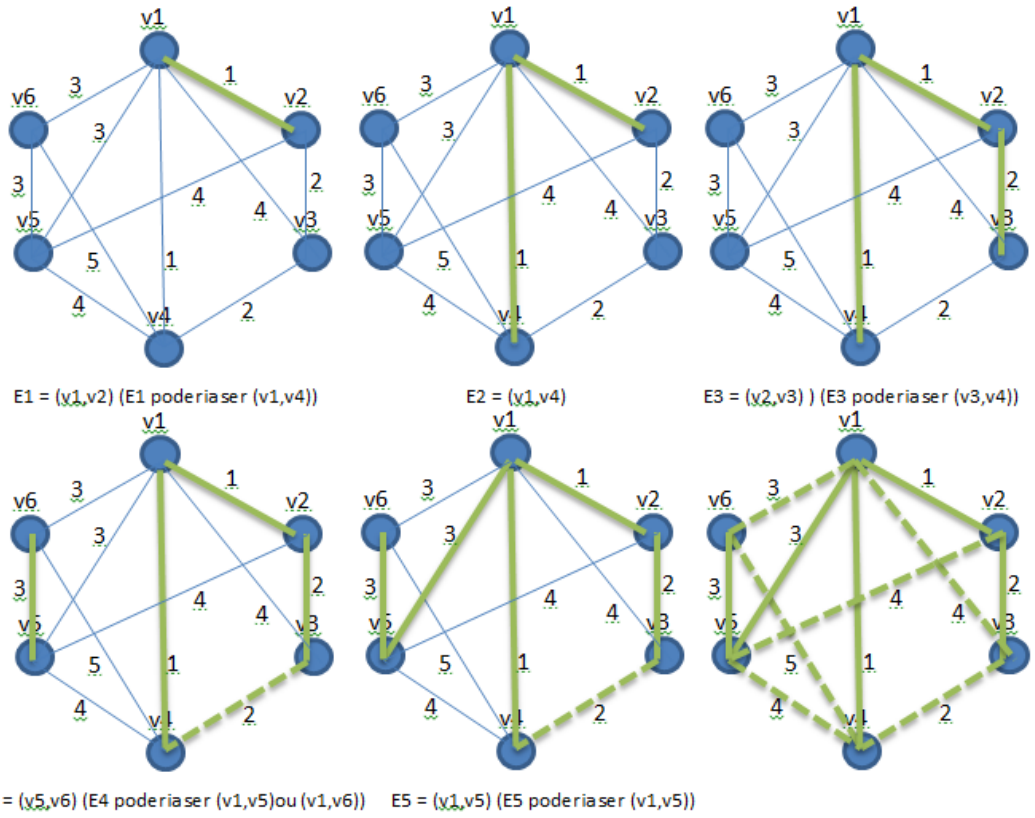


Figura 2-8 - Construção da Árvore geradora mínima utilizando o algoritmo Kruskal (Fonte: CLARK e HOLTON,1995)

2.2.1.2 Prim

Proposto por Robert C. Prim em 1957 (PRIM,1957), assim como o Kruskal é um algoritmo guloso que encontra a árvore geradora mínima. Funciona, no entanto, somente para grafos conexos.

O algoritmo começa recebendo por parâmetro um grafo *G = (V, A)*, e tem como saída um grafo $T = (V, A_T)$. No inicio o conjunto de vértices de *T* é formado por apenas um elemento *u*, que pertence ao conjunto de vértices de *G*. A partir desse elemento *u*, são adicionados a uma fila *F* todas as arestas que estão ligadas a *u*, e a fila é ordenada pelos pesos das arestas. Enquanto a quantidade de vértices de *T* for menor que a quantidade de vértices de *G*, o algoritmo retira a menor aresta da fila, verifica se esta aresta não forma laço, caso não forme laços, une aos conjuntos do grafo *T* tanto o vértice {*v}* quanto a aresta {E}. Note que, quando {v} se une ao conjunto de vértices do grafo *T*, as arestas de *v* também entram na Fila de prioridade que novamente é ordenada (CORMEN,2009).

Diferente do Kruskal, no algoritmo Prim o grafo *T*, em qualquer iteração, sempre é uma árvore, no entanto o custo final do grafo *T* é o mesmo.

Algoritmo 2.2 – Prim (Fonte: adaptado de CORMEN,2009)

Entrada: Grafo G =(V, A);
Saída: Árvore geradora mínima $T = (V, A_T)$;
Estruturas auxiliares: Fila de prioridade F.

```
1. T.V = u ∈ G.V // O conjunto de vértices da Árvore T recebe o vértice aleatório u que pertence ao grafo G
2. F = u.A // A Fila recebe todas as arestas do vértice u
3. F.Ordenar() // A Fila ordena as arestas por ordem crescente
4. Enquanto |T.V| < |G.V| , Faça
5.     Remova de F o primeiro elemento E // Aresta de menor peso
6.     Se Não-Ciclo(E), Então // Se o elemento E a ser inserido não formar ciclo então
7.         T.A_T = T.A_T U E // Une as arestas do conjunto T.A com {E}
8.         T.V = T.V U E.v// Une os vértices do conjunto T.V com { v}
9.         Inserir-Fila(E.v)// Insere todas as arestas do novo vértice na fila
10.        F.Ordenar();
11. Retorne T
```

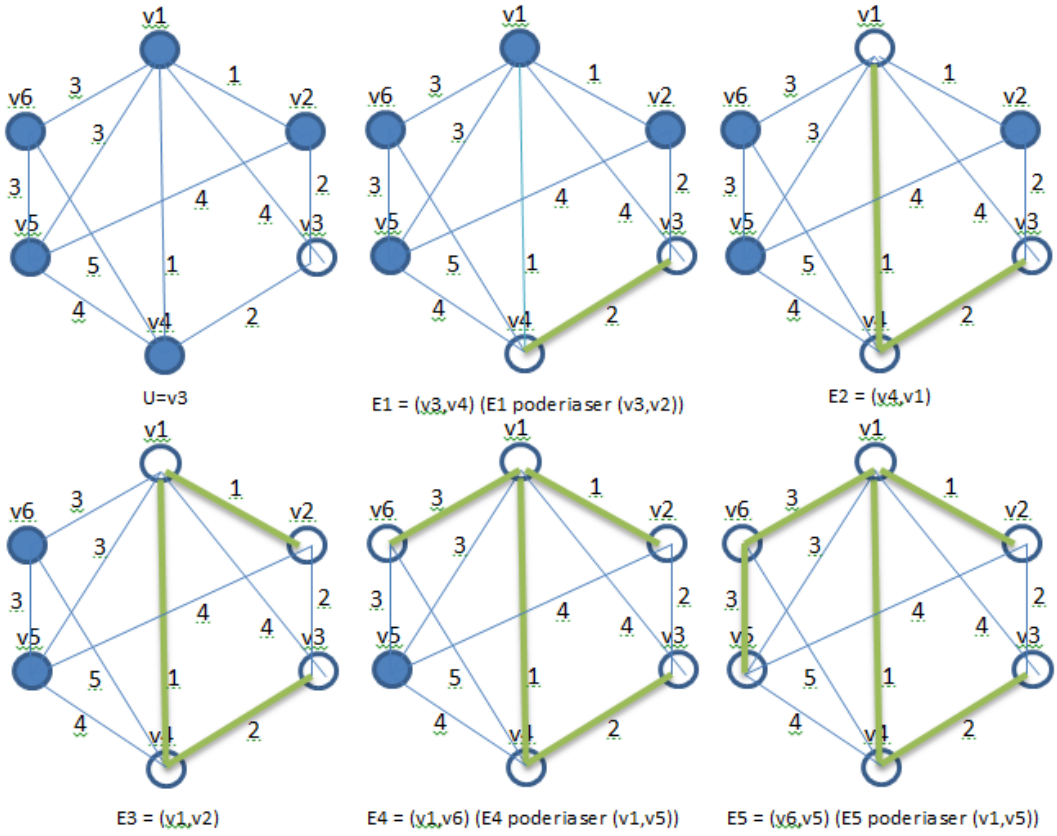


Figura 2-9 - Construção da Árvore geradora mínima utilizando o algoritmo Prim (Fonte: CLARK e HOLTON,1995)

2.2.2 Caminhos mínimos

Para ilustrar esse conceito, considere o seguinte cenário: suponha que em uma rede de computadores existe um computador mestre que envia mensagens para todos os outros computadores interligados a rede (*broadcast*). No entanto nem sempre é possível interligar todos os computadores da rede diretamente ao mestre. Desse modo alguns computadores recebem as mensagens do mestre por meio de outros computadores, e quando isso ocorre existe um atraso, que depende da quantidade de computadores que repassaram a mensagem. A pergunta que surge é: qual seria o melhor caminho entre o mestre e todos os outros computadores, para que esse atraso seja o menor possível? (CLARK e HOLTON,1995).

Este problema é conhecido como problema do caminho mínimo, e apesar de também gerar uma árvore como solução, como o problema de árvore geradora mínima, é diferente, pois busca encontrar o caminho mínimo de um vértice s para todos os demais vértices de G. Para a solução deste problema existem diversos algoritmos, no entanto o mais usado e conhecido é o algoritmo de *Dijkstra*.

O algoritmo de *Dijkstra* começa recebendo como parâmetro um grafo *G = (V,A)* e um vértice *s*, que será o vértice raiz, e tem como saída uma árvore $T=(V,A_T)$ de caminhos mínimos. No início dois vetores são inicializados, um de distâncias *D*, que são iniciados com os maiores valores possíveis, e outro de caminhos *L*, que é iniciado com valores nulos, denominados vetor de ligação *L*. É atribuído zero ao vetor de distância na posição *s*, e *s* ao vetor de caminhos na mesma posição, pois *s* é o vértice raiz. Enquanto o vetor de distância não for nulo, o algoritmo retira o vértice *u* de menor distância do vetor de distância, une o vértice *u* ao conjunto de vértices de *T*, e recalcula todas as distâncias dos vértices ligados a *u* (CORMEN,2009). Para retornar uma árvore, este algoritmo usa o vetor de caminhos para formar o conjunto de arestas do grafo *T*.

Algoritmo 2.3 – Dijkstra (Fonte: adaptado de CORMEN,2009)

Entrada: Grafo G =(V, A), vértice s (origem);
Saída: Árvore de caminhos mínimos T = (V, A_T);
Estruturas auxiliares: vetor de distâncias D, vetor de ligação L.

```
1. Para todo v ∈ G.V faça // inicia os vetores D e L
2.      D[v] = ∞
3.      L[v] = nulo
4. D[s] = 0 //D na posição s recebe 0
5. L[s] = s //L na posição s recebe s
6. Enquanto |D| ≠ Ø , Faça
7.      u = retiraMenorDistancia(D) // retira o vértice de menor distância
8.      T.V = T.V U {u} //Une ao conjunto de vértices do grafo T o vértice u
9.      Para todo v adjacente a u faça // recalcula as distâncias dos vértices adjacentes ao vértice u
10.             Se D[v]> D[u] + pesoDaAresta(u,v) então
11.                     D[v] = D[u] + pesoDaAresta(u,v)
12.                     L[v]=u
13. Para todo v ∈ G.V faça
14.     T.A_T = T.A_T U (v,L[v])
15. Retorne T
```

2.3 Tipos de representação de grafos

Para representar um grafo pode-se fazer um diagrama formado por círculos que simbolizam os vértices e retas para identificar as arestas, como na Figura 2-10. Apesar de ser um tipo de representação de fácil visualização, não é adequada computacionalmente, por ser de difícil implementação (STEEN, 2010). No entanto existem outras formas de se representar um grafo.

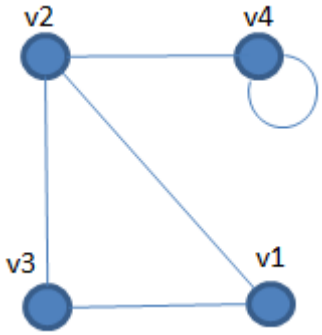


Figura 2-10 - Representação de um Grafo por meio de um diagrama (Fonte: adaptado de STEEN, 2010)

2.3.1 Matriz de adjacência

Considere um grafo $G = (V, A)$. A matriz de adjacência de G é uma matriz nxn, que se denota por M, onde $M = (m_{ij})$ para $1 \leq i,j \leq n$ e $n = |V|$, $m_{ij} = 1$ quando $\{i,j\} \in A$ e $m_{ij} = 0$ caso contrário, em grafos não ponderados (STEEN, 2010).

Assim o grafo da Figura 2-10 é representado da seguinte forma:

	v1	v2	v3	v4
v1	0	1	1	0
v2	1	0	1	1
v3	1	1	0	0
v4	0	0	0	1

Figura 2-11 – Matriz de adjacência (Fonte: adaptado de STEEN, 2010)

Para um grafo ponderado em vez do valor um coloca-se o valor do peso da aresta.

2.3.2 Matriz de incidência

Considere um grafo *G = (V, A)*. A matriz de incidência de *G* é uma matriz *nxm*, que denota-se por *M*, onde $M = (m_{ij})$, para $1 \leq i \leq n$, $1 \leq j \leq m$, $n = |V|$ e $m = |A|$, $m_{ij} = 1$ quando o vértice *j* é incidente à aresta *i*, e $m_{ij} = 0$ caso contrário, em grafos não ponderados (STEEN, 2010).

Assim o grafo da Figura 2-10 é representado da seguinte forma:

	{v1,v2}	{v1,v3}	{v2,v3}	{v2,v4}	{v4,v4}
v1	1	1	0	0	0
v2	1	0	1	1	0
v3	0	1	1	0	0
v4	0	0	0	1	2

Figura 2-12 - Matriz de incidência (Fonte: adaptado de STEEN, 2010)

O vértice v4 é um laço, assim ele é incidente duas vezes.

2.3.3 Lista de adjacência

Considere um grafo *G = (V, A)*, a lista de adjacência de *G* são listas associadas a cada um dos elementos do conjunto de vértices do grafo *G*, cada lista guarda os vértices adjacentes ao elemento que ela esta associada (CLARK e HOLTON,1995).

Assim o grafo da Figura 2-10 é representado da seguinte forma:

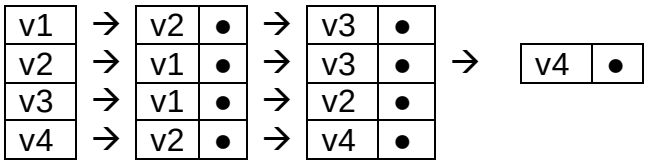


Figura 2-13 - Lista de adjacência (Fonte: adaptado de CLARK e HOLTON,1995)

2.4 Classificação de padrões

Classificação de padrões é uma etapa em reconhecimento de padrões, que tem por objetivo classificar objetos, da forma mais confiável possível, em sua respectiva classe. Um objeto é formado por um vetor de atributos, e por meio desses atributos é que se pode identificar a qual classe o mesmo pertence (DUDA et al.,

2000). Existem dois tipos distintos de classificação, a supervisionada e a não-supervisionada.

2.4.1 Classificação supervisionada

Esse tipo de classificação é caracterizado pelo conhecimento prévio dos rótulos do conjunto de treinamento, assim é possível fazer ajustes no classificador em função dos erros ou acertos obtidos. No entanto, nem sempre é possível garantir que todo o conjunto de treinamento seja classificado de forma correta (PAPA et al., 2008). Ela pode ser do tipo paramétrica, quando assume um modelo paramétrico conhecido para os dados, ou não paramétrica quando isso não ocorre.

2.4.1.1 K-vizinhos mais próximos ou K-NN

É uma técnica não paramétrica que se baseia em classificar um objeto a partir dos seus vizinhos mais próximos. A quantidade de vizinhos não é fixa, no entanto em muitos casos deve ser maior que um. O conjunto de treinamento é usado somente na fase de execução da classificação, não existe uma fase anterior de treinamento, por isso ela é considerada uma técnica de aprendizado preguiçoso (CUNNINGHAM e DELANY, 2007). No exemplo a seguir, Figura 2-14, considere duas classes, quadrado e triângulo, e a quantidade de vizinhos igual a cinco, foi escolhido uma quantidade ímpar para não haver empate. O círculo será o objeto a ser classificado.

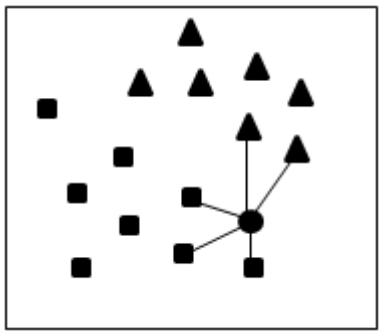


Figura 2-14 – K-NN (Fonte: adaptado de CUNNINGHAM e DELANY, 2007)

O objeto está conectado a três quadrados e a dois triângulos, sendo assim ele será classificado como quadrado. A probabilidade de o círculo ser da classe quadrado é dado pela seguinte expressão.

$$p(\omega_j|x) = \frac{k_j}{N} \qquad (1)$$

onde kj é o número de ocorrências de elementos da classe *ωj*, e N é a quantidade de vizinhos (WEBB, 2002).

Como o método calcula todas as distâncias da amostra a ser classificada para todas as amostras do conjunto de treinamento, antes de verificar quais amostras estão mais próximas, o tempo de execução desse método é altamente dependente do tamanho do conjunto de treinamento (CUNNINGHAM e DELANY, 2007).

2.4.1.2 Árvore de decisão

A classificação de um objeto ocorre por meio de uma série de testes (*if-then*) que são feitos nos ramos da árvore. Como as folhas são rotuladas com as classes, a classificação ocorre quando o objeto atinge uma folha (RUSSEL e NORVIG, 2003). É uma técnica não paramétrica, que permite solucionar problemas de classificação não linear ou com dados complexos, muito utilizada em mineração de dados, por aceitar que o conjunto de dados seja composto por diversos tipos de variáveis. No entanto, ao contrário do *K-NN*, essa técnica exige a fase de treinamento. Segundo LEVADA et al. (2010) “uma possível desvantagem desse método é que o tempo de treinamento pode ser excessivamente longo no caso de um grande conjunto de dados”.

2.4.1.3 Discriminante quadrático e linear

Essa técnica paramétrica classifica o objeto atribuindo ele à classe mais provável, ou seja, a classe que maximiza a seguinte expressão, pela regra de Bayes.

$$P(C_i|x) = \frac{(P(C_i) * P(x|C_i))}{P(x)} \qquad (2)$$

Desconsiderando $P(x)$, que é uma constante para todas as classes, tem-se:

$$P(C_i|x) = P(C_i) * P(x|C_i) \quad (3)$$

A probabilidade a *priori* de $P(C_i)$ é dado por:

$$P(C_i) = \frac{n_i}{\sum_j n_j} \quad (4)$$

onde n_i é a quantidade de elementos da classe C_i e $\sum_j n_j$ é a somatória de todos os elementos. Suponha-se que $P(x|\ C_i)$ é dado pela função de densidade de probabilidade de uma distribuição Gaussiana, descrito na seguinte expressão:

$$P(x|\ C_i) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} \exp\{-\frac{1}{2}(\vec{x} - \vec{\mu}_i)^T \Sigma_i^{-1}(\vec{x} - \vec{\mu}_i)\} \quad (5)$$

onde $\vec{\mu}_i$é o vetor de média da classe i e Σ_i é a matriz de covariância da classe i.

Agora tem-se:

$$P(C_i|x) = P(C_i) * \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} \exp\{-\frac{1}{2}(\vec{x} - \vec{\mu}_i)^T \Sigma_i^{-1}(\vec{x} - \vec{\mu}_i)\} \quad (6)$$

Aplicando log na expressão, tem-se a função discriminante quadrática.

$$P(C_i|x) = \log(P(C_i)) - \frac{1}{2}\log(|\Sigma_i|) - \frac{1}{2}(\vec{x} - \vec{\mu}_i)^T \Sigma_i^{-1}(\vec{x} - \vec{\mu}_i) \quad (7)$$

Caso todas as matrizes de Covariâncias forem idênticas $\Sigma_i = \Sigma$, a função discriminante torna-se linear, seguindo a equação:

$$P(C_i|x) = \log(P(C_i)) - \frac{1}{2}(\vec{x} - \vec{\mu}_i)^T \Sigma^{-1}(\vec{x} - \vec{\mu}_i) \quad (8)$$

O classificador que maximiza $P(C_i|x)$ é conhecido como classificador MAP (‘*maximum a posteriori probability*’) (THEODORIDIS e KOUTROUMBAS, 2009). Quando ele usa a função discriminante quadrática ele recebe o nome de Classificador Discriminante Quadrático ou Bayesiano Quadrático, e por sua vez

quando faz uso da função discriminante linear recebe o nome de Classificador Discriminante Linear ou Bayesiano Linear.

2.4.1.4 Support Vector Machines (SVM)

É uma técnica não paramétrica, que tenta encontrar hiperplanos ótimos de separação entre classes, por meio de vetores de suporte. Veja um exemplo que considera duas classes, como ilustra a Figura 2-15. Na fase de treinamento são encontrados dois vetores de suporte, um para cada classe, o hiperplano ótimo será a reta que mais se distancia de ambos os vetores de suporte (DUDA et al., 2000).

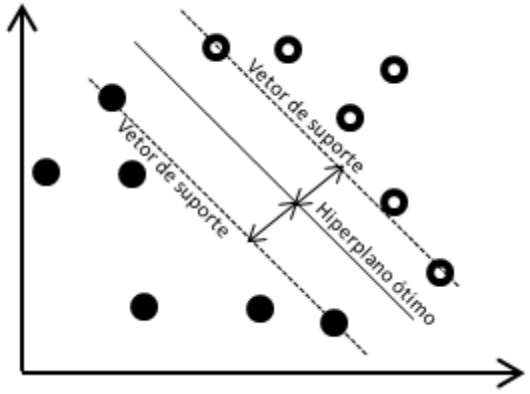


Figura 2-15 – SVM (Fonte: DUDA et al., 2000)

No entanto, a grande maioria dos problemas de classificação não é separável linearmente. Para isso, o SVM faz uso de um artificio, que consiste em mapear os dados em um espaço de maior dimensão. Contudo é necessário fazer uso de funções de *Kernel*, os tipos mais comuns são polinomiais, gaussianas e sigmoides (MELGANI e BRUZZONE, 2004). A princípio, SVM era utilizado para problemas de classificação binária. No entanto, pode ser empregado em problema de múltiplas classes, decompondo o mesmo em vários problemas de classificação binária (DUAN e KEERTHI, 2005). A grande vantagem é o tempo de classificação, já o tempo de treinamento é um fator limitante, pois é altamente influenciado pela quantidade de elementos do conjunto de treinamento, e a dimensionalidade dos dados (THEODORIDIS e KOUTROUMBAS, 2009).

2.4.1.5 Redes neurais

Inspirado no estudo do cérebro humano, mais especificamente no neurônio, é uma técnica que faz uso de um modelo matemático para simular o funcionamento de

uma rede de neurônios. Uma das redes mais conhecidas é a *Perceptron*, composta por uma camada de entrada, uma camada (simples) ou mais (multicamadas) ocultas e uma camada de saída. As camadas da rede são formadas por neurônios, como na Figura 2-16. O aprendizado de uma rede neural ocorre na fase de treinamento, onde os pesos dos neurônios são ajustados utilizando um algoritmo de aprendizado, como o de retropropagação. Essa técnica possui como grande limitante, redes *Perceptrons* separam classes somente com superfícies de decisão linear (*Perceptron* Simples). No entanto, é possível combinar várias redes para gerar superfícies mais complexas. Contudo, o desempenho é altamente prejudicado (HAYKIN, 2001).

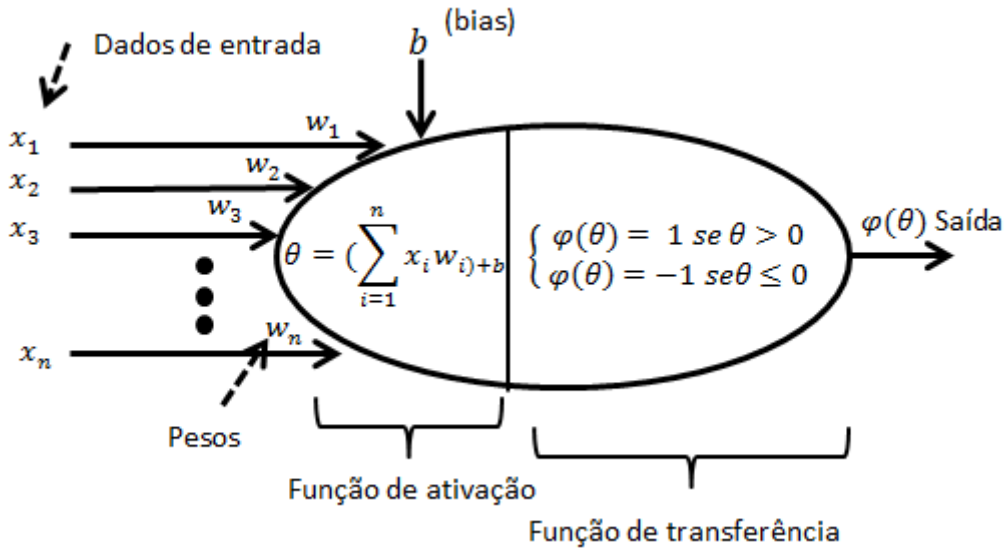


Figura 2-16 - Representação matemática do neurônio (Fonte: adaptado de HAYKIN, 2001)

2.4.2 Classificação não supervisionada

Ao contrário da classificação supervisionada, nesse tipo de classificação o conjunto de treinamento não é rotulado, pois rotular corretamente um conjunto é muitas vezes um processo que desprende muito tempo, esforço e dinheiro. No entanto, pode-se fazer uso de um conjunto de treinamento excessivamente grande, e assim tentar encontrar padrões por meio de agrupamentos consistentes (*clusters*) (DUDA et al., 2000). O principal processo de classificação não supervisionada é a clusterização, que faz a aglutinação de dados similares.

Os algoritmos de clusterização podem ser divididos em categorias, como Hierárquicos, Sequenciais, Baseados na otimização de funções de custo e Lógica nebulosa (*Fuzzy*). O algoritmo mais conhecido é a *K-Means*, que consiste em fixar *K* quantidade de classes de modo aleatório (centroides), depois associar cada objeto ao centroide mais próximo, após isso o centroide é atribuído ao objeto de centro gravitacional. O algoritmo tem fim quando não há mudança nos centroides (FRAHLING e SOHLER, 2006).

2.5 Campos aleatórios Markovianos

Um campo aleatório Markoviano é uma coleção de variáveis aleatórias, tal que a probabilidade de um elemento assumir um determinado valor, depende somente do comportamento de seus vizinhos, e não de todas as variáveis do campo (LEVADA et al., 2009).

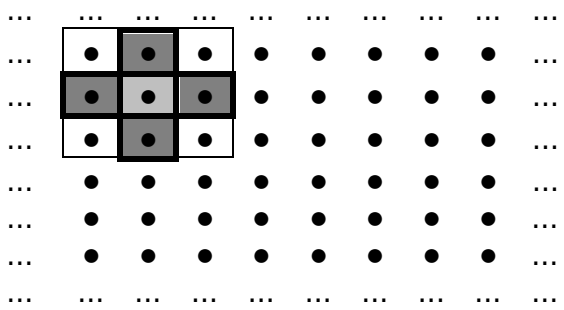


Figura 2-17 - Ilustração de um Campo aleatório Markoviano em um reticulado bidimensional (Fonte: LEVADA et al., 2009)

No caso da Figura 2-17 os elementos sempre tem a mesma quantidade de vizinhos. Já em um campo aleatório Markoviano definido em um grafo com topologia irregular (caso estudado nesta proposta), o número de vizinhos não é fixo ao longo de todo o campo, que dificulta a estimação dos parâmetros dos modelos (LEVADA et al., 2009).

Para a ilustração acima existem diversos tipos de sistemas de vizinhanças.

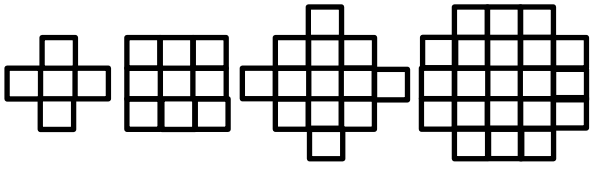


Figura 2-18 - Diferentes sistemas de vizinhanças (Fonte: LEVADA et al., 2009)

Para calcular a probabilidade de um campo aleatório Markoviano com dados discretos, o modelo mais utilizado é o modelo de *Potts* (BOYKOV et al., 1998).

2.5.1 Modelo de Potts

O modelo de *Potts*, um modelo Markoviano para dados discretos, é um modelo que estuda os efeitos coletivos como consequência de interações locais. A função densidade condicional local do modelo de *Potts* e dada pela seguinte expressão (LEVADA et al., 2009):

$$p(x_i = m|\beta, \eta_i) = \frac{e^{\beta U_i(m)}}{\sum_{l=1}^{c} e^{\beta U_i(m)}} \qquad (9)$$

onde β é uma parâmetro que controla a dependência dos elementos do campo (também conhecido como inverso da temperatura), η_i é o sistema de vizinhança local e $U_i(m)$ é o número de ocorrências do rótulo m no sistema de vizinhança local.

Sendo assim, a probabilidade de x_i ser da classe m depende dos seus vizinhos e do grau de dependência que os mesmos exercem sobre x_i, que é dado pelo parâmetro β. Quanto maior o β maior será a dependência.

Uma medida interessante pra extração de informação a partir desses dados discretos são medidas baseadas na teoria da informação, que nesse trabalho será introduzida utilizando a Informação de Fisher (PRINCIPE e XU, 1999). Um dos objetivos desse trabalho consiste em investigar a utilização da Informação de Fisher Local Observada, como ferramenta para quantificar o nível de informação presente nas amostras pertencentes ao conjunto de treinamento (grafo induzido pelas amostras), por meio de um procedimento análogo a detecção de bordas em grafos (PAVAN e PELILLO, 2003). A motivação para essa ideia advém do fato de que em problemas de classificação as bordas das regiões estão diretamente relacionadas às superfícies de decisão (DUDA et al., 2000). Um exemplo concreto seriam as máquinas de vetores de suporte, que buscam identificar superfícies de separação lineares que otimizam certos critérios definidos em termos dos vetores de suporte, que são os protótipos de classes diferentes mais próximos entre si (e portanto na região de fronteira).

2.6 Trabalhos relacionados

A classificação de padrões utilizando grafos vem sendo utilizada para classificação não-supervisionada e semi-supervisionada. Geralmente é empregada com o propósito de agrupar dados similares por meio do método de clusterização (SCHAEFFER,2007), utilizando a técnica de *K-means*. Essa abordagem é muito utilizada nas áreas de mineração de dados (LONG et al., 2006), pesquisas WEB (MATSUO et al., 2006), estudos dos genes (JIANG et al., 2004), entre outras. Uma técnica que está sendo explorada é a utilização de métodos de *Kernel* para agrupar dados, com o objetivo de resolver problemas relacionados à linearidade (CULP e MICHAILIDIS, 2008) (KULIS et al., 2009) (VISHWANATHAN et al., 2010) em classificação semi-supervisionada. Ainda em classificação semi-supervisionada, alguns trabalhos propõem utilização de grafos para propagação de rótulos entre seus vizinhos não rotulados para aprendizado transdutivo (SZUMMER e JAAKKOLA, 2001)(ZHOU e SCHOLKOPF, 2004).

No entanto esse tipo de abordagem é pouco utilizado na classificação supervisionada. Apesar disso, neste tópico serão apresentados dois trabalhos que estão fortemente relacionados com este trabalho.

2.6.1 Classificação baseada em K-associação

É um método de classificação supervisionado, não paramétrico baseado em grafos. Esse método gera um grafo a partir do método K-NN modificado, chamado de K-associado, que liga todos os vértices aos seus K vértices mais próximos e de mesmo rótulo, mantendo as duplas ligações (se o vértice A se liga com o vértice B e o oposto também ocorre, à medida que o grafo é criado, as duas ligações são mantidas) (BERTINI,2011). Uma vez gerado o grafo é calculada a medida de pureza de cada componente conexa, ou seja, o grau de entrelaçamento entre as diferentes classes, quanto menor o grau de pureza, maior será o entrelaçamento. Este cálculo é feito pela expressão:

$$\phi_\alpha = \frac{|G_\alpha|}{2K} \qquad (10)$$

onde ϕ_α é o grau de pureza, $|G_\alpha|$ é a média do grau da componente α e K é o tamanho da vizinhança considerada para construir o grafo.

Levando em consideração o grau de pureza, o método propõe a geração de um grafo K-associado ótimo, que tem por objetivo aumentar o grau de pureza de cada componente, utilizando o *K* mais adequado a cada uma delas. Sendo assim cada componente tem seu respectivo K_α.

A classificação ocorre da seguinte forma, dado o conjunto de treinamento, é gerado um grafo K-associado ótimo, e para cada componente é calculado o grau de pureza. Para a classificação de um elemento, o mesmo é inserido no grafo e ligado aos *K* (maior *K* do grafo) vizinhos mais próximos de cada componente. Depois disso a classificação ocorre pelas seguintes equações:

$$P(v_y \in C_\alpha | \Lambda_{vy}) = \frac{P(\Lambda_{vy} | v_y \in C_\alpha)\, P(v_y \in C_\alpha)}{P(\Lambda_{vy})} \qquad (11)$$

onde $P(v_y \in C_\alpha | \Lambda_{vz})$ é a probabilidade a *posteriori* de acordo com a regra de Bayes, e a probabilidade condicional é dada por:

$$P(\Lambda_{vy} | v_y \in C_\alpha) = \frac{|\{\Lambda_{vy,K_\alpha} \cap C_\alpha\}|}{K_\alpha} \qquad (12)$$

onde Λ_{vy,K_α} representa o conjunto dos K_α vizinhos mais próximos do elemento vy. A constante normalizadora do denominador da Expressão 11 é dada por:

$$P(\Lambda_{vz}) = \sum_{N_{vz},\beta \neq 0} P(\Lambda_{vy} | v_y \in C_\beta)\, P(C_\beta) \qquad (13)$$

Por fim a probabilidade a *priori* é definida como:

$$P(v_y \in C_\alpha) = \frac{\phi_\alpha}{\sum_{N_{vy},\beta \neq 0} \phi_\beta} \qquad (14)$$

onde ϕ_α é o grau de pureza do componente α e $\sum_{N_{vz}\beta \neq 0} \phi_\beta$ é a somatória do grau de pureza de todos os componentes ligantes ao elemento vz. Para decisão final computa-se a soma das probabilidades a *posteriori* dos componentes das classes, dada por:

$$P(v_y|\omega_i) = \sum_{class(C_\alpha)=\omega_i} P(v_y \in C_\alpha | \Lambda_{vy}) \quad (15)$$

onde $P(v_z|\omega_i)$ é a probabilidade do elemento v_y ser da classe ω_i e C_α componente α. Sendo assim a regra de decisão final é:

$$\varphi(v_y) = \arg\ max\{P(v_y|\omega_1), \dots, P(v_y|\omega_M)\} \quad (16)$$

onde $\varphi(v_z)$ é a classe atribuída ao elemento (BERTINI et al., 2011).

Considere o seguinte exemplo dado por Bertini et al. (2011), três classes com cinco componentes, (Figura 2-19). Calcula-se a probabilidade a *priori* $P(v_y \in C_\alpha)$ para todas as componentes.

$$P(v_y \in C_1) = \frac{\phi_1}{\phi_1 + \phi_2 + \phi_4 + \phi_5} = \frac{0.97}{0{,}97 + 0{,}81 + 0{,}5 + 0{,}79} \approx 0{,}32$$

$$P(v_y \in C_2) = \frac{\phi_2}{\phi_1 + \phi_2 + \phi_4 + \phi_5} = \frac{0{,}81}{0{,}97 + 0{,}81 + 0{,}5 + 0{,}79} \approx 0{,}26$$

$$P(v_y \in C_4) = \frac{\phi_4}{\phi_1 + \phi_2 + \phi_4 + \phi_5} = \frac{0.5}{0{,}97 + 0{,}81 + 0{,}5 + 0{,}79} \approx 0{,}17$$

$$P(v_y \in C_5) = \frac{\phi_5}{\phi_1 + \phi_2 + \phi_4 + \phi_5} = \frac{0.79}{0{,}97 + 0{,}81 + 0{,}5 + 0{,}79} \approx 0{,}25$$

Calculando a probabilidade a *posteriori* $P(\Lambda_{vy}|v_y \in C_\alpha)$ para cada componente se tem:

$$P(\Lambda_{vy}|v_y \in C_1) = \frac{1}{4}$$

$$P(\Lambda_{vy}|v_y \in C_2) = \frac{1}{3}$$

$$P(\Lambda_{vy}|v_y \in C_4) = \frac{1}{2}$$

$$P(\Lambda_{vy}|v_y \in C_5) = \frac{1}{3}$$

Desse modo se tem:

$$P(\Lambda_{vz}) = \frac{1}{4} \times 0{,}32 + \frac{1}{3} \times 0{,}26 + \frac{1}{2} \times 0{,}17 + \frac{1}{3} \times 0{,}25 = 0{,}3343$$

Assim pela regra de Bayes tem-se:

$$P(v_y \in C_1|\Lambda_{vy}) = \frac{\frac{1}{4} \times 0{,}32}{0{,}3343} \approx 0{,}24$$

$$P(v_y \in C_2|\Lambda_{vy}) = \frac{\frac{1}{3} \times 0{,}26}{0.3343} \approx 0{,}26$$

$$P(v_y \in C_4|\Lambda_{vy}) = \frac{\frac{1}{2} \times 0{,}17}{0{,}3343} \approx 0{,}25$$

$$P(v_y \in C_5|\Lambda_{vy}) = \frac{\frac{1}{3} \times 0{,}25}{0{,}3343} \approx 0{,}25$$

Sabendo que a componente C_4 e C_5 são da mesma classe, tem-se:

$$P(v_y|\omega_1) = P(v_y \in C_1|\Lambda_{vy}) = 0{,}24$$
$$P(v_y|\omega_2) = P(v_y \in C_2|\Lambda_{vy}) = 0{,}26$$
$$P(v_y|\omega_3) = P(v_y \in C_4|\Lambda_{vy}) + P(v_y \in C_5|\Lambda_{vy}) = 0{,}5$$

Pela expressão $\varphi(v_y)$ o elemento recebe o rótulo da classe ω_3.

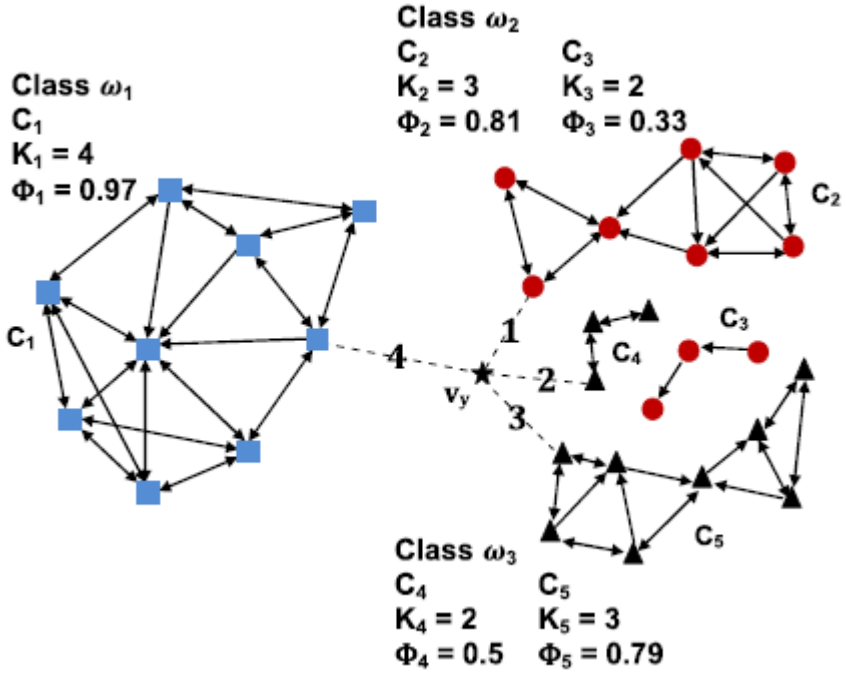


Figura 2-19 - Exemplo de classificação (Fonte: BERTINI,2011)

2.6.2 Classificação utilizando Floresta de caminhos ótimos

Existem dois tipos de classificação utilizando Floresta de caminhos ótimos (OPF – *Optimum-Path Forest*), a não supervisionada e a supervisionada. Dentro da classificação supervisionada também se tem uma subdivisão, a com grafo K-NN e a com grafo completo. No entanto, para este estudo, será apresentado o método mais abrangente, o OPF com supervisão e grafo completo (método não paramétrico) (PAPA et al.,2009).

O algoritmo OPF é dividido em duas etapas: uma de treinamento e outra de classificação.

Na etapa de treinamento é gerado um grafo do conjunto de treinamento, cada elemento do conjunto é um vetor de atributos, que é representado por um vértice no grafo, onde o mesmo é ligado a todos os outros vértices (grafo completo) por meio de arestas ponderadas. A ponderação das arestas ocorre por meio de uma função de distância. Após gerar o grafo, o método deve escolher os protótipos de cada classe (no mínimo um por classe), esses protótipos são as amostras da região de fronteira entre as classes. Para fazer essa escolha é executado o algoritmo de Árvore geradora mínima. A seguir, percorre-se o grafo encontrando as amostras conectadas de classes com rótulos diferentes, as quais serão os protótipos. Cada protótipo deve conquistar as amostras restantes do grafo, o que é feito por meio do Algoritmo Dijkstra generalizado, ou seja, em vez do algoritmo original que tem apenas um ponto raiz, esse algoritmo possui vários pontos raízes, dependendo da quantidade de protótipos (EKLUND et al., 1996).

Outra diferença significativa é a função de custo a ser minimizada. Ao invés da função de custo “*FSum*”, que faz a somatória de todos os custos até atingir a raiz, adotada pelo algoritmo de Dijkstra, essa metodologia adota a função “*FMax*”, onde o custo é determinado pelo maior custo entre amostra e raiz, com o intuito de minimizar alguns problemas encontrados, como o problema da cadeia (chain problem). Ao executar o Dijkstra generalizada o grafo gerado é, na verdade, uma floresta de caminhos de menor custo, e cada protótipo é uma raiz, de custo zero, e os vértices recebem o custo da função “*FMax*”. Quando um vértice protótipo conquista um vértice normal, o vértice normal recebe o rótulo do vértice protótipo. A Figura 2-20 ilustra o procedimento.

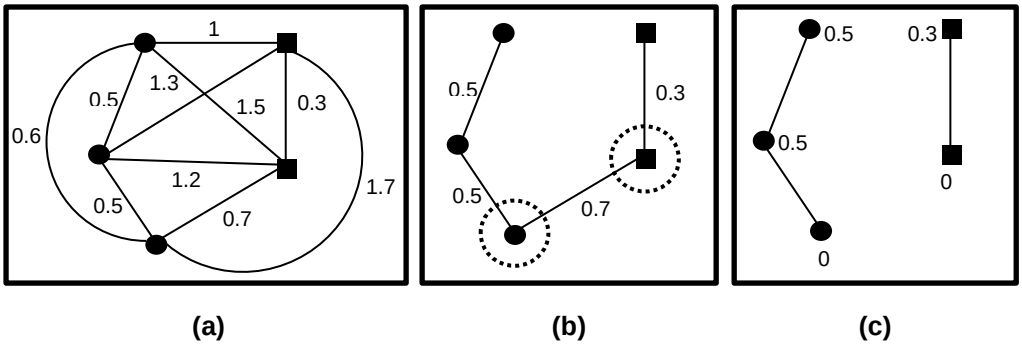


Figura 2-20 - Sequência da fase de treinamento. (a)Grafo completo (b)Escolha dos Protótipos (c)Floresta de caminhos mínimos (Fonte: adaptado de PAPA et al.,2009)

Na etapa de classificação uma amostra (vértice) é inserida na Floresta de caminhos mínimos, a mesma é ligada a todos os vértices, e o vértice que oferecer o menor custo, em relação a função de custo "*FMax*", é a que conquista a amostra rotulando-a. Após isso, a amostra é retirada e outra amostra pode ser classificada (PAPA et al.,2009). Figura 2-21.

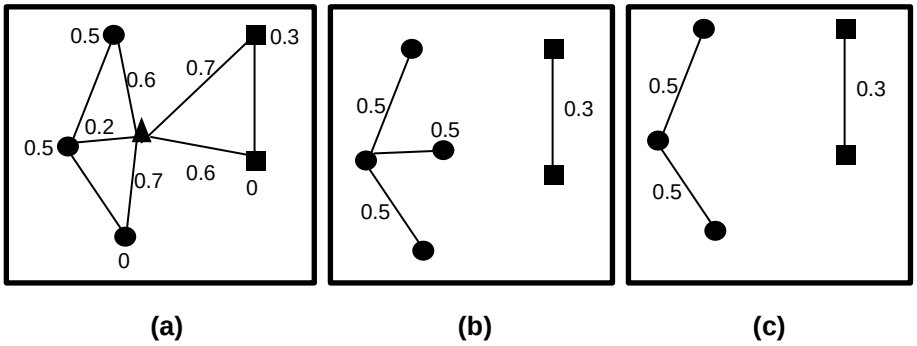


Figura 2-21 - Sequência da fase de classificação. (a)Inserção da amostra (b)Classificação (c)Retirada da amostra mínimos (Fonte: adaptado de PAPA et al.,2009)

2.7 Considerações finais

Neste capítulo foi apresentada a revisão bibliográfica deste projeto de mestrado, onde foram abordados conceitos de grafos, seu problemas e algoritmos, bem como os tipos de representação; classificação de padrões (supervisionado e não supervisionado), e as principais metodologias de classificação, campos aleatórios Markovianos e o modelo de Potts. Além disso, foram apresentados

alguns trabalhos relacionados a este projeto. O objetivo da revisão bibliográfica apresentada, juntamente com os principais trabalhos relacionados, é servir de base para um melhor entendimento da proposta de trabalho apresentada no capítulo seguinte.

Capítulo 3

PROPOSTA DO TRABALHO

Neste capítulo serão apresentados o método proposto, bem como os métodos de validação a serem utilizados como medidas quantitativas de desempenho.

O objetivo deste trabalho é propor um método para explorar caminhos de mínima informação em grafos para problemas de classificação supervisionada, seguindo uma abordagem contextual. O método proposto seleciona protótipos com o propósito de definir as amostras que delimitam as regiões de fronteira entre classes vizinhas (vértices de alta informação). Essa seleção é feita por meio da construção de um grafo conexo e utilizando um modelo contextual baseado em Campos Aleatórios Markovianos (modelo de *Potts*), para calcular a Informação de Fisher Observada Local de cada amostra do conjunto de treinamento. Após selecionar os protótipos, o método propõe a competição entre eles para rotular as amostras a serem classificadas. O protótipo que oferecer o caminho com menor informação, ou seja, o caminho de menor custo é o que conquista a amostra e a rotula. Caminhos de mínima informação serão aqueles que cruzarem o menor número de fronteiras, o que de forma intuitiva representa um comportamento desejado em problemas de classificação de padrões.

3.1 Método proposto

Na metodologia proposta, o sistema tem como entrada vetores de atributos, que posteriormente serão os vértices de um grafo induzido. Utilizando os vetores de atributos o grafo é construído calculando-se as distâncias entre os vetores, e definindo a relação de vizinhança (adjacência) entre eles. Na fase de treinamento todos os vértices são rotulados, e para cada vértice é calculada sua Informação de Fisher Observada Local, como tentativa de mensurar o nível de informação presente

em cada amostra do conjunto de treinamento. Em seguida, as arestas do grafo são ponderadas, e os vértices com maiores informações são selecionados como protótipos ou sementes (serão raízes das árvores de caminhos mínimos). Na etapa de classificação, uma amostra é inserida no grafo, calcula-se a distância dela para as demais amostras e com isso é definida a relação de adjacência. A seguir, é feito o cálculo da Informação de Fisher da amostra, e suas arestas são ponderadas. Por meio do algoritmo *Dijkstra* generalizado ("*Multi-source Dijkstra*") a amostra é classificada, recebendo o rótulo do protótipo mais próximo. A Figura 3-1 mostra o diagrama de blocos do método proposto.

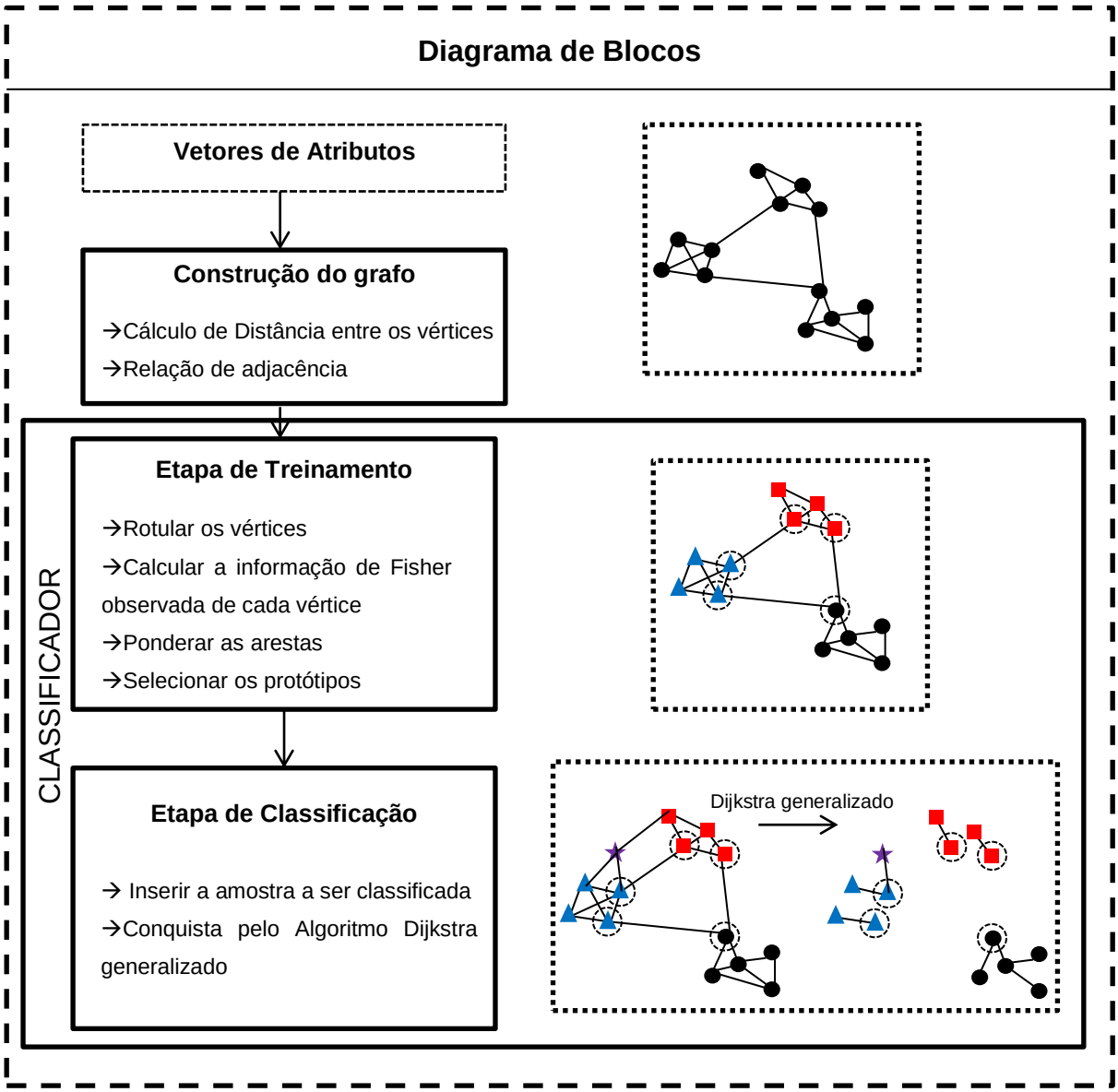


Figura 3-1 - Diagrama de Blocos do Método

3.1.1 Construção do Grafo

Para a construção do grafo é preciso primeiramente calcular as distâncias entre os vetores de atributos, para isso existem várias distâncias que serão consideradas, são elas a City-Block, Euclidiana, Mahalanobis e a medida de similaridade do cosseno (NGUYEN e BAI, 2010).

Dados dois vetores de atributos $\vec{U} = (u_1, u_2, \dots, u_n)$ e $\vec{V} = (v_1, v, \dots, v_n)$, para n>0, o cálculo das distâncias é dado pelas expressões da Tabela 3-1.

Tabela 3-1 - Expressões de distâncias

Tipo de Distância	Expressão	Expressões auxiliares
City-Block	$D_{cb}(\vec{U},\vec{V}) = \sum_{i=1}^{n} \lvert u_i - v_i \rvert$	
Euclidiana	$D_e(\vec{U},\vec{V}) = \sqrt{\sum_{i=1}^{n} (u_i - v_i)^2}$	
Mahalanobis	$D_m(\vec{U},\vec{V}) = \sqrt{(\vec{U}-\vec{V})^T \Sigma^{-1} (\vec{U}-\vec{V})}$	$\Sigma = \frac{1}{n}\sum_{i=1}^{n} (\vec{x_i} - \vec{\mu})(\vec{x_i} - \vec{\mu})^T$
Medida de Similaridade do Cosseno	$D_C(\vec{U},\vec{V}) = \frac{\vec{U}.\vec{V}}{\lVert\vec{U}\rVert \lVert\vec{V}\rVert}$	$\lVert\vec{X}\rVert = \sqrt{\sum_{i=1}^{n} {x_i}^2}$

Antes de se fazer os cálculos das distâncias, deve-se representar o grafo a ser construído por meio de uma matriz de adjacência *N x N*, onde *N* é o número de vértices (vetores de atributos). Como serão calculadas todas as distâncias entre todos os vértices, ao final dos cálculos de distâncias tem-se um grafo completo. No entanto, isso não é interessante, pois como o foco desse trabalho é calcular a informação de cada vértice, se o grafo for completo a informação de todos os vértices será a mesma (PRINCIPE e XU, 1999), o que não revela detalhes sobre o conjunto de dados. Para resolver esse problema foram adotados dois métodos de

geração de grafos, o "*K-NN*" que liga os *K* vértices mais próximos, e o "*E-ball*" que fixa um raio *R* e procura os vértices que estão dentro dos limites desse volume.

Para executar os métodos de geração de grafos tem-se que ter a certeza de que o grafo é conexo, pois isso será importante na etapa de treinamento, mais precisamente na definição dos protótipos. Caso o grafo seja desconexo, amostras de classes diferentes nem sempre se conectam, quando isso ocorre as bordas são limitadas às amostras de mesma classe. Desse modo, os protótipos escolhidos não serão amostras pertencentes a regiões de fronteira entre classes distintas, conforme ilustra a Figura 3-2. Para isso o algoritmo Prim é utilizado para gerar a árvore geradora mínima, que será o grafo base, onde os métodos de geração de grafos irão iniciar o seu processo. A Figura 3-3 mostra o fluxograma dessa etapa.

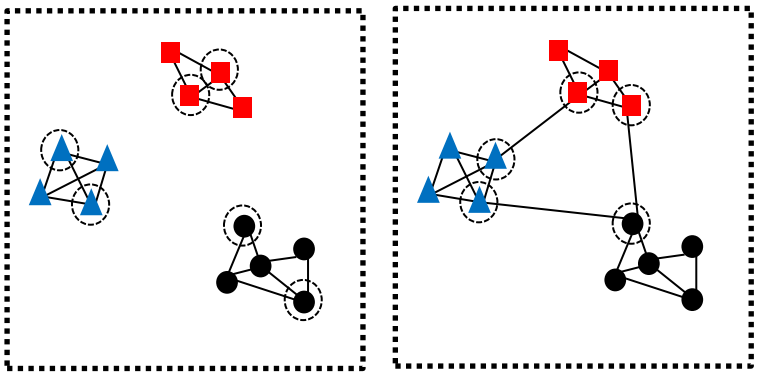


Figura 3-2 - Escolha dos protótipos (a) sem o algoritmo de Prim (b) com o algoritmo de Prim

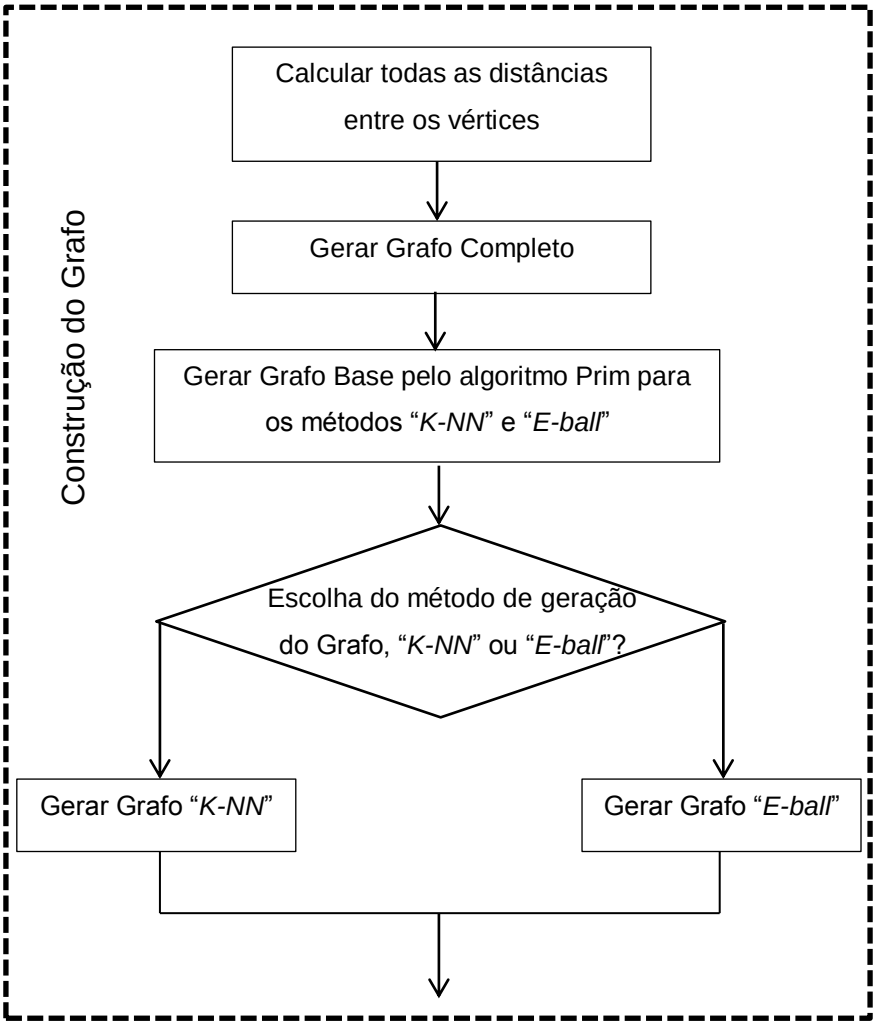


Figura 3-3 - Fluxograma da Construção do Grafo

3.1.2 Etapa de Treinamento

Nesta etapa são rotulados os vértices, e para cada vértice calculada a sua informação de Fisher Local Observada (LEVADA et al., 2009) (derivação no Apêndice 1), dado pela expressão:

$$\varphi_i = \frac{\sum_{l=1}^{q}\sum_{k=1}^{q}[(U_i(m)-U_i(l))(U_i(m)-U_i(k))e^{\beta\left(U_i(l)+U_i(k)\right)}]}{\sum_{l=1}^{q}\sum_{k=1}^{q}e^{\beta\left(U_i(l)+U_i(k)\right)}} \quad (17)$$

onde φ_i é a informação do vértice i, q é a quantidade de classes, U_i é um vetor de q posições, onde cada posição representa a quantidade de vértices ligantes de

determinada classe ao vértice i, m é a posição no vetor U_i da classe do vértice i , e o valor de β critico é dado pela expressão:

$$\beta = \ln(1+\sqrt{q}) \qquad (18)$$

Como a quantidade de vértices ligantes nem sempre é a mesma (Figura 3-4) deve-se atribuir pesos a cada ligação, esse peso é dados pela expressão:

$$p_i = \frac{k}{l_i} \qquad (19)$$

onde p_i é o peso do vértice i, k é a relação de adjacência (K-NN) e l_i é o total de ligantes do vértice i (grau do vértice i).

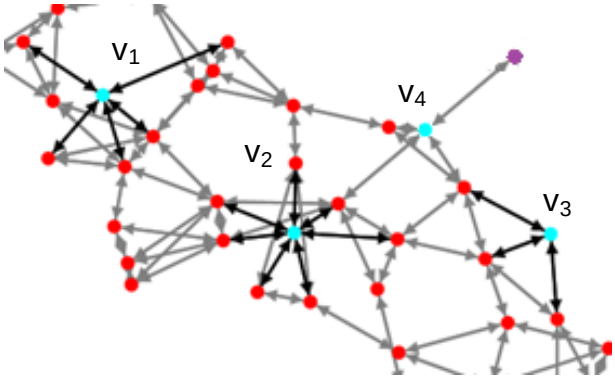


Figura 3-4 - Grafo k=3

Sendo assim a expressão 17 agora é dada por:

$$\varphi_i = \frac{\sum_{l=1}^{q}\sum_{k=1}^{q}[(U_i(m).p_i - U_i(l).p_i)(U_i(m).p_i - U_i(k).p_i)e^{\beta(U_i(l).p_i+U_i(k).p_i)}]}{\sum_{l=1}^{q}\sum_{k=1}^{q}e^{\beta(U_i(l).p_i+U_i(k).p_i)}} \qquad (20)$$

Para exemplificar considere os vértices v_1, v_2, v_3 e v_4, calculando a informação de Fisher pela Equação 17 para cada vértice têm-se:

$\varphi_1 = 0{,}00002902$

$\varphi_2 = 0{,}00000013$

$\varphi_3 = 0{,}00412746$

$\varphi_4 = 0{,}01535970$

Utilizando a Equação 20 temos:

$\varphi_1 = \varphi_2 = \varphi_3 = 0{,}0041$

$\varphi_4 = 0{,}04189$

Em outras palavras, isso significa que neste trabalho foi utilizada uma versão modificada do modelo de Potts, dada por:

$$p(x_i = m|\beta, \eta_i) = \frac{e^{\beta U_i(m) p_i}}{\sum_{l=1}^{c} e^{\beta U_i(m) p_i}} \qquad (21)$$

Note que utilizando a Equação 17 os vértices v_3 e v_4 são os mais informativos. No entanto v_3 possui ligações com vértices de mesma classe, sendo que para o método proposto ele deve possuir a mesma informação que v_1 e v_2. Por outro lado, v_4 possui ligação com um vértice de outra classe, algo que é altamente informativo. Utilizando a Equação 20 isso não ocorre, o vértice mais informativo é somente o v_4.

Nesse ponto, cada vértice tem um valor de informação associado. Esse valor será utilizado para ponderar as arestas, atribuindo ao peso w_{ij} o valor da soma das informações dos vértices v_i e v_j. Desse modo uma aresta que conecta dois vértices de alta informação (tipicamente arestas que cruzam a fronteira de decisão) terá maior custo que uma aresta que conecta dois vértices de menor informação. Assim torna-se possível distinguir os limiares de decisão na fase de classificação. No entanto, tem-se que selecionar vértices protótipos, para que o algoritmo Dijkstra generalizado (CORMEN,2009) possa subdividir o grafo em árvores de caminhos ótimos, a partir dos protótipos. Vértices protótipos são aqueles que possuem alta informação quando comparado aos demais de mesma classe, esse protótipos são selecionados a fim de caracterizar a forma da classe, uma vez que vértices de mais alta informação representam as fronteiras de decisão.

A Figura 3-5 mostra o fluxograma dessa etapa.

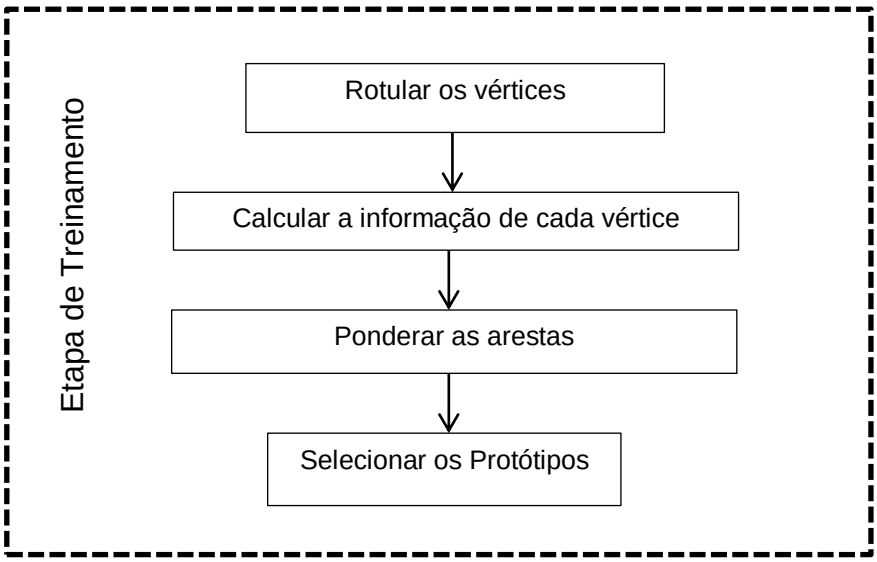


Figura 3-5 – Fluxograma da Etapa de Treinamento

3.1.3 Etapa de Classificação

Para a classificação, uma amostra é inserida no grafo, sendo feito o calculo da distância dela para as demais amostras de treinamento. Em seguida é definida a relação de adjacência ("*K-NN*" ou "*E-ball*"), calcula-se a Informação de Fisher Local da amostra e as arestas são ponderadas. Para classificá-la deve-se encontrar o menor caminho entre a amostra e um dos protótipos escolhidos na etapa de treinamento, o protótipo que oferecer o caminho de menor custo irá rotular a amostra com a classe a que ele pertence.

O algoritmo que promove essa disputa entre os diversos protótipos é o *Dijkstra* generalizado. Ele tem a execução similar ao *Dijkstra* "tradicional", no entanto em vez de ter um vértice raiz, ele possui diversos vértices raízes. O algoritmo começa recebendo como parâmetro um grafo $G = (V,A)$ e um conjunto de vértices S, que serão os vértices protótipos, e tem como saída uma floresta $F = (V,A_F)$ de caminhos mínimos. No início dois vetores são inicializados, um de distâncias, que é iniciado com infinito, e outro de caminhos, que é iniciado com valores nulos. São atribuídos zeros ao vetor de distância nas posições dos elementos pertencentes ao conjunto de vértices *S*, e o próprio elemento ao vetor de caminhos nas mesmas posições, pois os elementos do conjunto *S* são os vértices raízes. Enquanto o vetor de distância não for vazio, o algoritmo retira o vértice *u* de menor distância do vetor de distância, une o vértice *u* ao conjunto de vértices de *F*, e recalcula todas as distâncias dos vértices ligados a *u* (CORMEN,2009). Para retornar uma Floresta, este algoritmo usa o vetor de caminhos para formar o conjunto de arestas do grafo *F*. Veja o Algoritmo 3.1. A Figura 3-6 mostra o fluxograma dessa etapa.

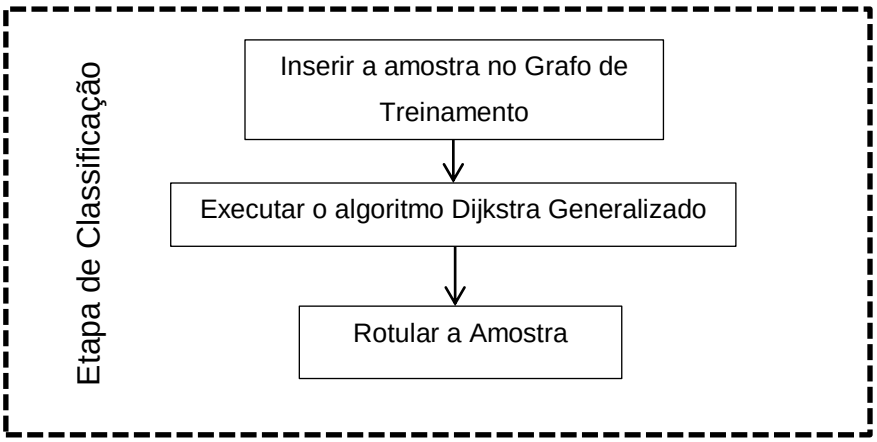


Figura 3-6 - Fluxograma da Etapa de Classificação

Algoritmo 3.1 - Dijkstra Generalizado (Fonte: adaptado de CORMEN,2009)

Entrada: Grafo G =(V, A), vértices S (protótipos);

Saída: Floresta de caminhos mínimos F = (V, A_F);

Estruturas auxiliares: vetor de distâncias D, vetor de ligação L.

```
1. Para todo v ∈ G.V faça // inicia os vetores D e L
2.      D[v] = ∞
3.      L[v] = nulo
4. Para todo s ∈ S faça
5.      D[s] = 0 //D na posição s recebe 0
6.      L[s] = s //L na posição s recebe s
7. Enquanto |D| ≠ Ø , Faça
8.      u = retiraMenorDistancia(D) // retira o vértice de menor distância
9.      F.V = F.V U {u} //Une ao conjunto de vértices do grafo T o vértice u
10.     Para todo v adjacente a u faça // recalcula as distâncias dos vértices adjacentes ao vértice u
11.             Se D[v]> D[u] + pesoDaAresta(u,v) então
12.                     D[v] = D[u] + pesoDaAresta(u,v)
13.                     L[v]=u
14. Para todo v ∈ G.V faça
15.     F.A_F = F.A_F U (v,L[v])
16. Retorne F
```

Existem métodos estatísticos para avaliar e comparar as diversas metodologias de classificação, que tem por objetivo mensurar o grau de generalização do classificador, ou seja, estimar a acurácia. Neste trabalho serão utilizados o método de validação *Resubstitution*, nas etapas de Geração do Grafo e Treinamento com o propósito de estimar quais protótipos geram as maiores taxas de acertos, otimizando essas etapas com base na escolha e definição de parâmetros. Além disso, os métodos de validação cruzada *K-fold*, *Holdout* e *Leave-one-out* serão utilizados na etapa de Classificação para estimar o grau de generalização do classificador.

3.2.1 Resubstitution Validation

Nesse método o classificador é treinado e testado com todo o conjunto de dados. É a validação mais simples de todas, no entanto como os dados de treinamento são idênticos aos dados de teste. Nesse cenário, as taxas de acertos podem não ser reais quando testados com outros dados (problema de *overfitting*) (KOHAVI, 1995), ou seja, é um processo que gera um taxa otimista de acertos.

3.2.2 K-fold Cross-Validation

A validação cruzada (*Cross-Validation*) é um método estatístico que divide o conjunto de dados em dois segmentos, um usado para o treinamento e o outro para teste. Essa divisão ocorre basicamente da seguinte forma: todo o segmento de dados é dividido em *K* partes, *K-1* partes são usadas na fase de treinamento e uma parte é utilizada na fase de teste, contudo todas as partes devem ser testadas, sendo assim ocorre *K* iterações (RODRÍGUEZ et al., 2010) , veja a Figura 3-7.

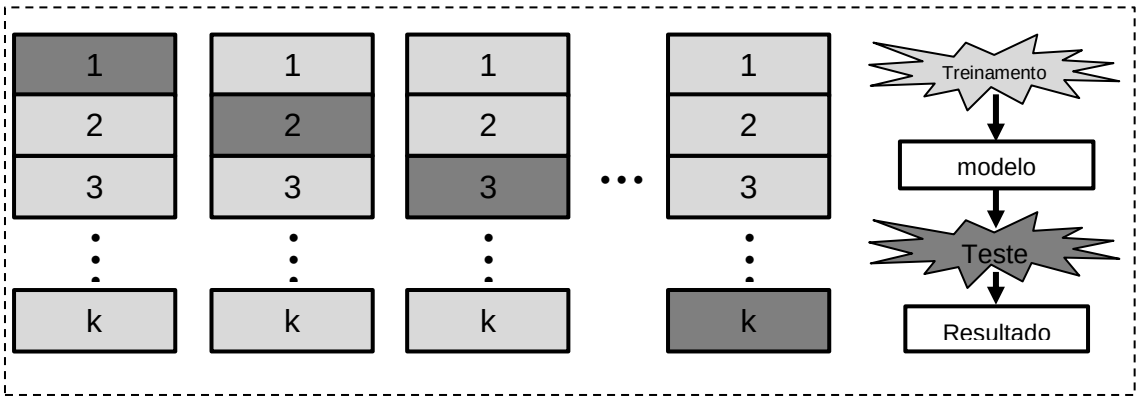


Figura 3-7 - K-fold Cross-Validation (Fonte: adaptado de VIAENE et al., 2005)

3.2.3 Holdout Validation

O *Holdout* divide o conjunto de dados em duas partes (*2-fold Cross-Validation*), uma de treinamento e outra de teste, não necessariamente idênticas, fazendo isso existe uma redução do problema de *overfitting*. Porém a grande desvantagem desse procedimento é a divisão dos conjuntos de treinamento e teste. Caso o conjunto de teste for muito fácil ou muito difícil pode-se ter uma distorção na

taxa de acertos/erro. Para resolver esse problema pode-se executar o mesmo processo varias vezes e depois tirar a média. No entanto, esse processo deve ocorrer de maneira sistemática, uma vez que todas as amostras devem estar presentes na fase de treinamento e teste na mesma proporção (KIM, 2009). Quando executado de forma correta gera uma taxa realista de acertos.

3.2.4 Leave-One-Out Cross-Validation

Esse método retira uma amostra para o teste e o restante das amostras fará parte do conjunto de treinamento, isso é repetido para as *N* amostras do conjunto (*N-fold Cross-Validation*). É útil quando a quantidade de amostras é limitada ou escassa, no entanto é o método mais utilizado para avaliar um classificador, pois gerar uma taxa pessimista de acertos (CAWLEY, 2006).

3.2.5 Coeficiente Kappa

Para validar os resultados foi utilizado o coeficiente Kappa, que é uma medida estatística que determina a concordância entre avaliadores distintos. Em classificação supervisionada esse coeficiente determina o grau de concordância a *posteriori*, por meio da matriz de confusão, onde as colunas representam os valores verdadeiros, e as linhas representam os valores obtidos (esperado e obtido). O coeficiente Kappa pode variar de 1 até abaixo de 0, onde 1 representa a total concordância, ou seja, todas as amostras foram classificadas de modo correto, e igual ou abaixo de 0 representa nenhuma concordância. A expressão para o cálculo do coeficiente Kappa a partir da matriz de confusão X é dada por:

$$\widehat{K} = \frac{N\sum_{i=1}^{C} c_{ii} - \sum_{i=1}^{C} x_{i+}x_{+i}}{N^2 - \sum_{i=1}^{C} x_{i+}x_{+i}} \qquad (22)$$

onde x_{i+} é a soma dos elementos da linha i, x_{+i} é a soma dos elementos da coluna i, C é o número de classes e N é o número total de observações (CONGALTON, 1991).

Para exemplificar o cálculo do coeficiente Kappa considere a Matriz confusão:

4	1	0
0	3	2
0	0	5

Sendo assim N=15 e C=3, calculando Kappa:

$$\sum_{i=1}^{C} c_{ii} = (c_{00} + c_{11} + c_{22}) = (4 + 3 + 5) = 12$$

$$\sum_{i=1}^{C} x_{i+}x_{+i} = (x_{0+}x_{+0} + x_{1+}x_{+1} + x_{2+}x_{+2}) = (5.4 + 5.4 + 5.7) = 75$$

$$\widehat{K} = \frac{N\sum_{i=1}^{C} c_{ii} - \sum_{i=1}^{C} x_{i+}x_{+i}}{N^2 - \sum_{i=1}^{C} x_{i+}x_{+i}} = \frac{15(12) - 75}{15^2 - 75} = \frac{105}{150} = \mathbf{0,70}$$

3.3 Considerações Finais

Neste Capítulo foi apresentada a metodologia que se pretende adotar para explorar caminhos de mínima informação em grafos para problemas de classificação. O objetivo dessa metodologia é estudar como a informação contextual presente em um grafo pode ajudar na classificação supervisionada não paramétrica de objetos, por meio da investigação da topologia do grafo, induzido pelas amostras de treinamento durante a seleção de protótipos, para a construção de uma floresta de caminhos mínimos. Também foram apresentados métodos estatísticos para avaliar essa metodologia, com o proposito de identificar a viabilidade da mesma na classificação supervisionada.

Capítulo 4

RESULTADOS E DISCUSSÕES

4.1 Desenvolvimento da Ferramenta

Foi implementada um solução computacional baseada na metodologia proposta. Para isso foi utilizado o ambiente *Visual Studio 2010* de programação e a linguagem C# e todos os métodos foram implementados utilizando somente as bibliotecas padrão do C#.

Para avaliar a metodologia foram utilizadas diversas bases de dados, que possuem quantidade de atributos e amostras diferentes. Sendo assim, foi desenvolvido um método genérico onde é possível coletar dados de qualquer base que contenha um separador entre os atributos. Para coletar os dados o usuário informa o “Separador entre atributos” e seleciona o arquivo da base. Após selecionar o arquivo deve-se informar quais atributos devem ser considerados (em algumas bases existem atributos que se referem à identificação da amostra) e qual atributo se refere a classe, já que se trata de um classificador supervisionado. A Figura 4-1 mostra a interface desse método.

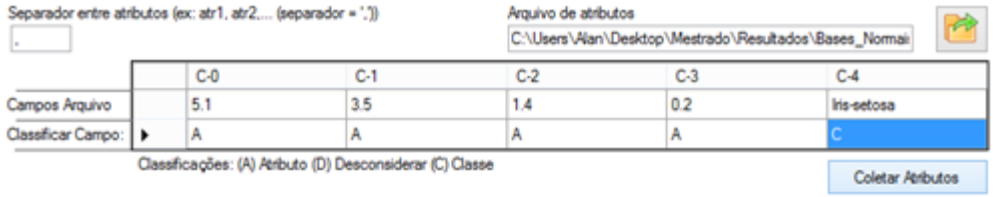


Figura 4-1 - Entrada dos dados

Foram implementadas duas estratégias diferentes. A primeira compara o método proposto com os métodos K-NN e o Classificador Bayesiano sob hipótese Gaussiana, utilizando a validação cruzada 10-Fold, e a segunda, foi implementada para analisar a seleção de protótipos e a topologia das bases de dados.

Na primeira estratégia, todos os métodos foram treinados e testados com os mesmos conjuntos de dados. A ferramenta desenvolvida gera randomicamente dez

arquivos de treinamento e dez arquivos de teste. Cada arquivo de teste contém 10% do conjunto de dados e cada arquivo de treinamento contém 90% do conjunto de dados. Primeiro são gerados os arquivos de teste e em seguida, com o completar, são gerados os arquivos de treinamento. A Figura 4-2 mostra parte dos arquivos de teste gerados.

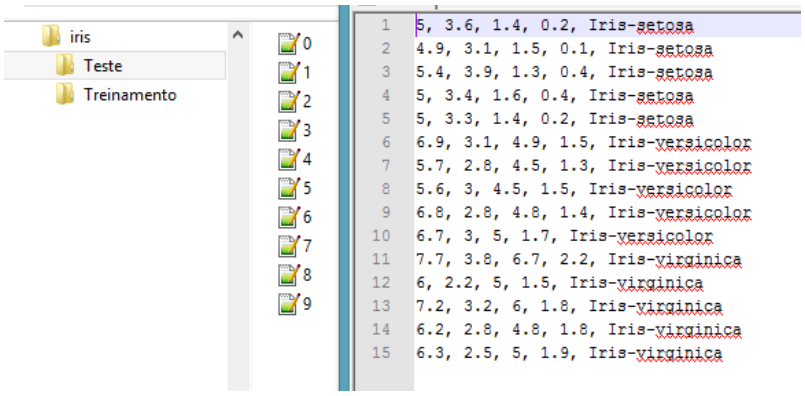
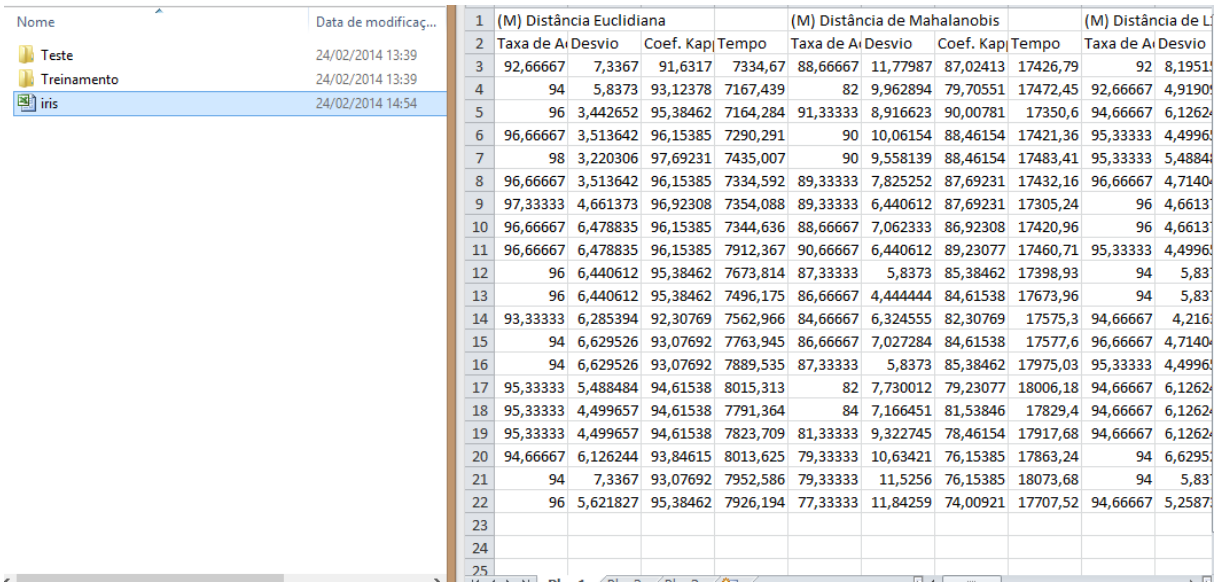


Figura 4-2 – Parte dos arquivos de teste gerados

Como a construção do grafo é um fator que tem influência na taxa de acerto no método proposto e também K-NN, foi utilizado o parâmetro K variando de 1 a 20. Além disso, são calculadas a taxa de acerto, desvio padrão, coeficiente Kappa e o tempo gasto para classificação das amostras. Todos os resultados dos testes são salvos em formato Excel na própria pasta onde se encontram os arquivos de teste e treinamento (Figura 4-3).



	(M) Distância Euclidiana				(M) Distância de Mahalanobis				(M) Distância de L
	Taxa de A	Desvio	Coef. Kap	Tempo	Taxa de A	Desvio	Coef. Kap	Tempo	Taxa de A	Desvio
	92,66667	7,3367	91,6317	7334,67	88,66667	11,77987	87,02413	17426,79	92	8,1951
	94	5,8373	93,12378	7167,439	82	9,962894	79,70551	17472,45	92,66667	4,9190
	96	3,442652	95,38462	7164,284	91,33333	8,916623	90,00781	17350,6	94,66667	6,1262
	96,66667	3,513642	96,15385	7290,291	90	10,06154	88,46154	17421,36	95,33333	4,4996
	98	3,220306	97,69231	7435,007	90	9,558139	88,46154	17483,41	95,33333	5,4884
	96,66667	3,513642	96,15385	7334,592	89,33333	7,825252	87,69231	17432,16	96,66667	4,7140
	97,33333	4,661373	96,92308	7354,088	89,33333	6,440612	87,69231	17305,24	96	4,6613
	96,66667	6,478835	96,15385	7344,636	88,66667	7,062333	86,92308	17420,96	96	4,6613
	96,66667	6,478835	96,15385	7912,367	90,66667	6,440612	89,23077	17460,71	95,33333	4,4996
	96	6,440612	95,38462	7673,814	87,33333	5,8373	85,38462	17398,93	94	5,83
	96	6,440612	95,38462	7496,175	86,66667	4,444444	84,61538	17673,96	94	5,83
	93,33333	6,285394	92,30769	7562,966	84,66667	6,324555	82,30769	17575,3	94,66667	4,216
	94	6,629526	93,07692	7763,945	86,66667	7,027284	84,61538	17577,6	96,66667	4,7140
	94	6,629526	93,07692	7889,535	87,33333	5,8373	85,38462	17975,03	95,33333	4,4996
	95,33333	5,488484	94,61538	8015,313	82	7,730012	79,23077	18006,18	94,66667	6,1262
	95,33333	4,499657	94,61538	7791,364	84	7,166451	81,53846	17829,4	94,66667	6,1262
	95,33333	4,499657	94,61538	7823,709	81,33333	9,322745	78,46154	17917,68	94,66667	6,1262
	94,66667	6,126244	93,84615	8013,625	79,33333	10,63421	76,15385	17863,24	94	6,6295
	94	7,3367	93,07692	7952,586	79,33333	11,5256	76,15385	18073,68	94	5,83
	96	5,621827	95,38462	7926,194	77,33333	11,84259	74,00921	17707,52	94,66667	5,2587

Figura 4-3 – Parte dos resultados dos testes

Na segunda estratégia a ferramenta possibilita que o usuário possa treinar a base de dados escolhendo a distância, tipo de grafo (K-NN ou E-Bal) e o parâmetro que define a relação de adjacência (K ou Raio). Para auxiliar na escolha do parâmetro foi implementado um método de “Gerar Grafo Ótimo”, baseado na escolha do tipo de grafo, o método varia a parâmetro e calcula o erro por meio da validação por resubstituição. O parâmetro que obtiver o menor erro é o selecionado. Para mostrar a variação do parâmetro pelo erro, a ferramenta gera um histograma onde x é a variação do parâmetro e y é a taxa de erro (Figura 4-4). Após a fase de Treinamento o usuário pode testar utilizando dois métodos de validação cruzada o Holdout (50% treina e 50% teste) e o Leave-One-Out.

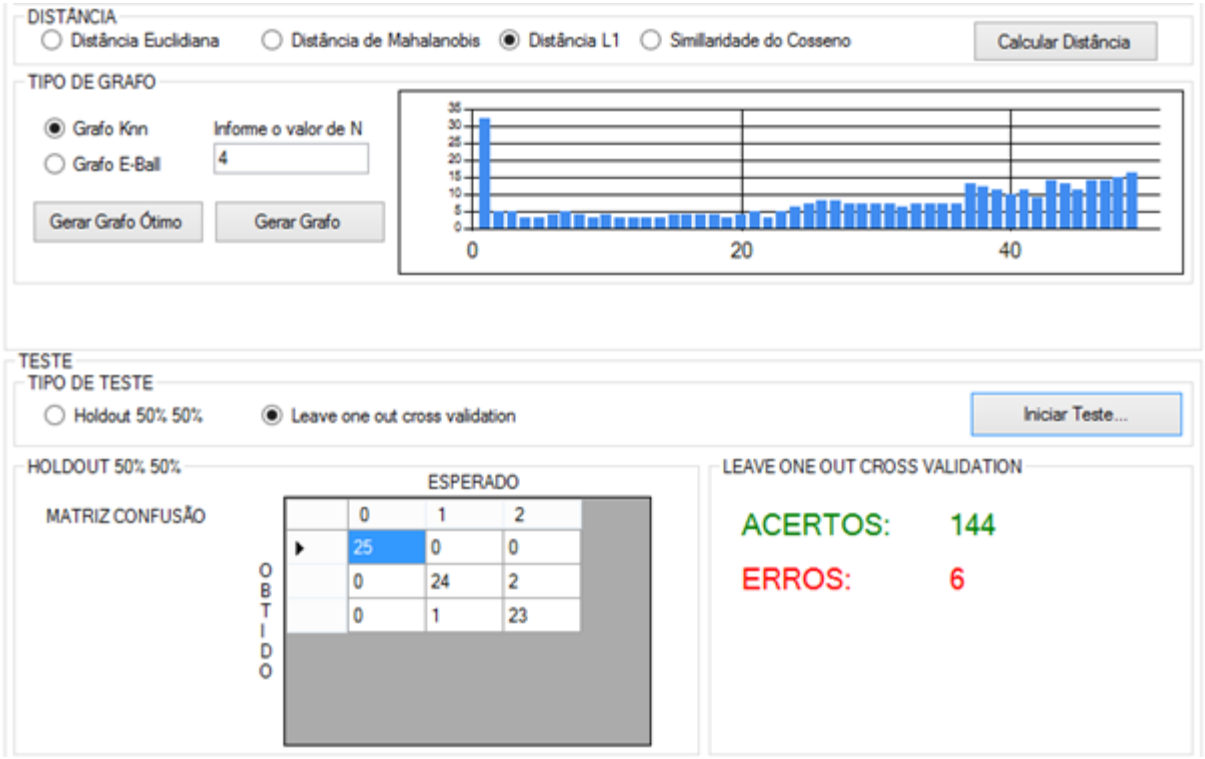


Figura 4-4 – Interface de testes do Protótipo

A ferramenta possibilita a visualização das matrizes de atributos, distância, informação de Fisher, grafo, grafo ponderado com a informação de Fisher e a matriz Dijkstra (Figura 4-5).

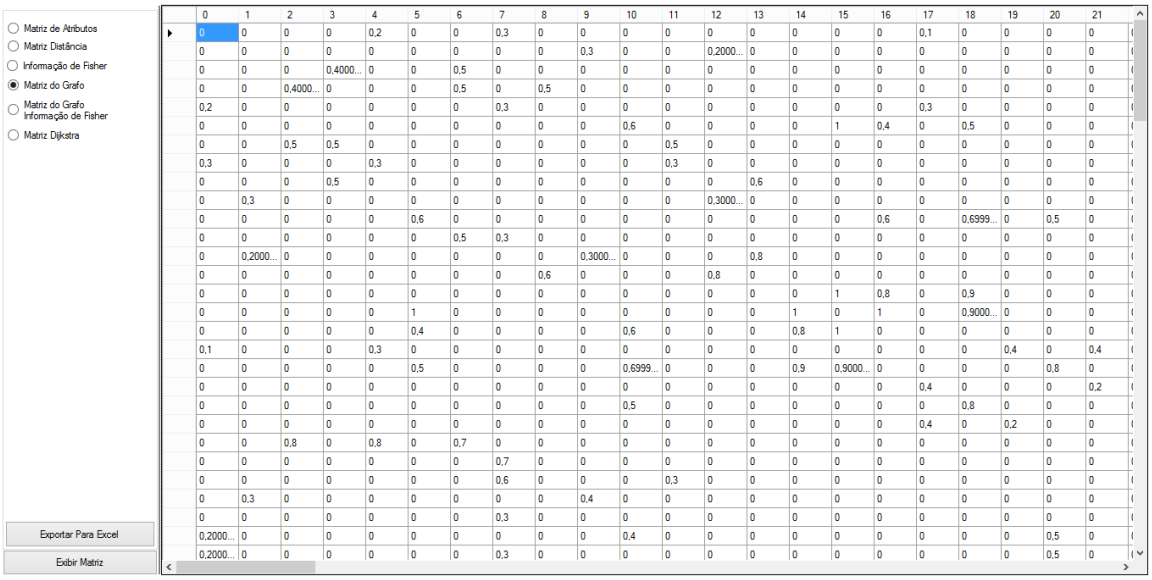


Figura 4-5 - Visualização de Matrizes

Além da visualização das matrizes é possível visualizar o grafo, esse recurso é limitado quando comparado ao plug-in disponível para o Excel *NodeXL* Figura 4-6.

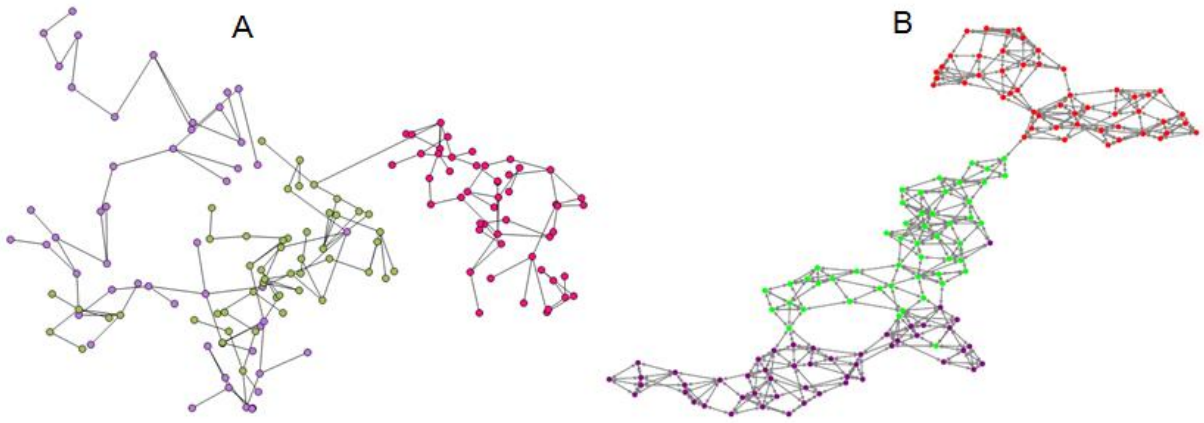
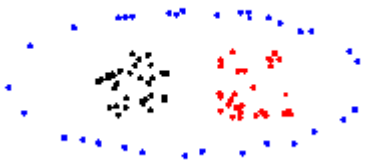
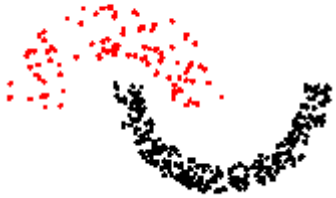
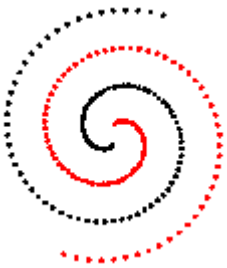


Figura 4-6 – Grafos (A)Ferramenta desenvolvida, (B) NodeXL

4.2 Base de Dados

Para avaliar e comparar a eficácia do método proposto foram utilizadas 21 bases de dados com características distintas, separadas em dois grupos: bases Artificiais (Tabela 4-1) e Reais (Tabela 4-2).

Tabela 4-1 - Base de Dados Artificiais

Base	Amostras	Atributos	Classes	Imagem
Boat	100	3	3	
Half-ring	400	3	2	
Noisy-lines	200	3	2	
Petals	100	3	4	
Saturn	200	3	2	
Spirals	200	3	2	

Fonte: http://pages.bangor.ac.uk/~mas00a/activities/artificial_data.htm

Tabela 4-2 Base de Dados Reais

Base	Amostras	Atributos	Classes	Descrição
Balance-scale	625	5	3	L (46,08%), B (7,84%) e R (46,08%)
Brainwave	182	13	2	1 (71,43%) e 2 (28,57%)
Breast cancer	569	30	2	Benign (62,74%) e Malignant (37,26%)
Credit-approval	690	16	2	+ (44,5%) e – (55,5%)
Dermatology	366	34	6	Psoriasis (30,60%), Seboreic dermatites (16,66%), Lichen planus (19,67%), Pityriasis rósea (13,39%), Cronic dermatitis (14,20%) e Pityriasis rubra pilaris (5,48%)
Diabetes	768	8	2	0 (65,10%) e 1 (34,90%)
Ecoli	336	8	8	cp (42,56%), im (22,91%), pp (15,48%), imU (10,42%), om (5,96%), omL (1,49%), imL (0,59%) e imS (0,59%)
Fertility	100	10	2	N (88%) e O(12%)
Glass	214	10	6	1 (32,71%), 2 (35,52%), 3 (7,95%), 5 (6,07%), 6 (4,20%) e 7 (13,55%)
Hayes-roth	132	6	3	1 (38,64%), 2 (38,64%) e 3 (22,72%)
Iris	150	5	3	Iris-setosa (33,33%), Iris-versicolor (33,33%) e Iris-virginica (33,33%)
Libras	360	91	15	1 (6,66%), 2 (6,66%),3 (6,66%),4 (6,66%), 5 (6,66%), 6 (6,66%), 7 (6,66%), 8 (6,66%), 9 (6,66%), 10 (6,66%), 11 (6,66%), 12 (6,66%),13 (6,66%),14 (6,66%) e 15 (6,66%)
Seeds	210	8	3	1 (33,33%), 2 (33,33%) e 3(33,33%)
Survival	306	4	2	1 (73,53%) e 2 (26,47%)
Wine	178	14	3	1 (33,14%), 2 (39,89%) e 3 (26,97%)

Fonte: http://archive.ics.uci.edu/ml/datasets

4.3 Comparações entre as metodologias

Para gerar os resultados abaixo, o parâmetro K da construção grafo, variou de 1 a 20, tanto na metodologia proposta quanto na metodologia K-NN, sendo que

para efeito de comparação foi utilizado a maior taxa de acerto de cada conjunto de dados. A Tabela 4-3 apresenta os resultados obtidos, onde D1, D2, D3 e D4 são respectivamente as distâncias Euclidiana, Mahalanobis, City-Block e a Similaridade do Cosseno. A comparação foi feita entre a metodologia proposta (M1), K-NN (M2) e o classificador Bayesiano sob hipótese Gaussiana (M3). Estão destacados em negrito os melhores resultados.

Tabela 4-3 - Comparação entre as metodologias.

Base de dados	Método	Distância	$\bar{x}$(s)	Kappa	K(Grafo)
Boat	M1	D1	**100,00(0,00)**	**100,00**	**1**
		D2	**100,00(0,00)**	**100,00**	**1**
		D3	**100,00(0,00)**	**100,00**	**1**
		D4	72,22(9,44)	64,29	3
	M2	D1	**100,00(0,00)**	**100,00**	**1**
		D2	**100,00(0,00)**	**100,00**	**1**
		D3	**100,00(0,00)**	**100,00**	**1**
		D4	75,56(10,21)	68,57	2
	M3		97,78(4,68)	97,14	
Half-ring	M1	D1	**75,25(5,33)**	**73,95**	**19**
		D2	74,50(6,21)	73,16	20
		D3	74,50(5,37)	73,16	20
		D4	72,50(5,40)	71,05	20
	M2	D1	73,50(6,15)	72,11	3
		D2	74,00(7,56)	72,63	3
		D3	73,25(6,24)	71,84	3
		D4	69,50(7,80)	67,89	20
	M3		70,25(6,29)	68,68	
Noisy_lines	M1	D1	99,50(1,58)	99,44	1
		D2	**100,00(0,00)**	**100,00**	**1**
		D3	**100,00(0,00)**	**100,00**	**1**
		D4	95,50(4,38)	95	16
	M2	D1	99,50(1,58)	99,44	1
		D2	**100,00(0,00)**	**100,00**	**1**
		D3	**100,00(0,00)**	**100,00**	**1**
		D4	94,50(3,69)	93,89	7
	M3		**100,00(0,00)**	**100,00**	
Petals	M1	D1	**100,00(0,00)**	**100**	**8**
		D2	**100,00(0,00)**	**100**	**7**
		D3	**100,00(0,00)**	**100**	**8**
		D4	**100,00(0,00)**	**100**	**3**
	M2	D1	**100,00(0,00)**	**100**	**6**
		D2	**100,00(0,00)**	**100**	**6**

		D3	**100,00(0,00)**	**100**	**6**
		D4	**100,00(0,00)**	**100**	**3**
	M3		97,50(5,27)	96,67	
Saturn	M1	D1	92,00(3,50)	91,12	1
		D2	92,00(3,50)	91,12	1
		D3	92,00(3,50)	91,12	1
		D4	66,00(6,15)	62,22	16
	M2	D1	93,00(2,58)	92,22	1
		D2	93,00(2,58)	92,22	1
		D3	**93,50(3,37)**	**92,78**	**1**
		D4	66,50(7,84)	62,78	14
	M3		64,50(5,99)	60,56	
Spirals	M1	D1	**100,00(0,00)**	**100**	**2**
		D2	**100,00(0,00)**	**100**	**1**
		D3	**100,00(0,00)**	**100**	**1**
		D4	63,00(11,60)	58,89	11
	M2	D1	**100,00(0,00)**	**100**	**2**
		D2	**100,00(0,00)**	**100**	**1**
		D3	**100,00(0,00)**	**100**	**1**
		D4	63,50(11,07)	59,44	17
	M3		60,00(9,13)	55,56	
Balance-scale	M1	D1	90,50(4,31)	90,17	6
		D2	89,33(4,79)	88,97	14
		D3	91,00(3,35)	90,69	7
		D4	93,67(3,31)	93,45	1
	M2	D1	90,33(4,29)	90	8
		D2	90,50(4,23)	90,17	20
		D3	90,33(4,07)	90	9
		D4	**94,33(3,06)**	**94,14**	**1**
	M3		91,33(4,29)	91,03	
Brainwave	M1	D1	**71,11(14,05)**	**67,5**	**7**
		D2	**71,11(14,05)**	**67,5**	**8**
		D3	**71,11(14,05)**	**67,5**	**7**
		D4	**71,11(14,05)**	**67,5**	**8**
	M2	D1	**71,11(14,05)**	**67,5**	**18**
		D2	**71,11(14,05)**	**67,5**	**18**
		D3	**71,11(14,05)**	**67,5**	**18**
		D4	**71,11(14,05)**	**67,5**	**18**
	M3		67,78(9,00)	63,75	
Diabetes	M1	D1	75,39(5,85)	74,73	18
		D2	75,13(5,60)	74,46	10
		D3	**76,45(5,93)**	**75,81**	**10**
		D4	69,87(6,69)	69,05	6
	M2	D1	74,61(7,44)	73,92	17

		D2	75,39(6,02)	74,73	20
		D3	**76,45(5,02)**	**75,81**	**19**
		D4	70,92(6,43)	70,14	7
	M3		73,82(6,72)	73,11	
Ecoli	M1	D1	50,00(10,00)	46,55	1
		D2	86,13(6,09)	85,21	4
		D3	54,84(5,89)	51,72	1
		D4	88,06(5,28)	87,25	5
	M2	D1	52,58(5,70)	49,31	11
		D2	88,06(6,28)	87,24	9
		D3	56,13(6,49)	53,1	11
		D4	**88,71(3,80)**	**87,93**	**5**
	M3		45,16(0,00)	41,38	
Fertility	M1	D1	87,78(13,30)	84,29	5
		D2	**88,89(11,71)**	**85,71**	**3**
		D3	87,78(13,30)	84,29	5
		D4	87,78(13,30)	84,29	3
	M2	D1	87,78(13,30)	84,29	4
		D2	87,78(12,23)	84,29	3
		D3	87,78(13,30)	84,29	4
		D4	87,78(13,30)	84,29	2
	M3		84,44(21,08)	80	
Glass	M1	D1	72,22(9,44)	68,8	1
		D2	70,56(8,30)	66,99	12
		D3	72,78(11,84)	69,43	5
		D4	73,33(11,65)	70,15	4
	M2	D1	73,33(9,37)	70	1
		D2	69,44(12,35)	65,63	5
		D3	73,89(10,49)	70,63	1
		D4	**74,44(14,15)**	**71,25**	**3**
	M3		57,78(7,50)	52,5	
Hayes-Roth	M1	D1	43,08(14,14)	32,73	18
		D2	**66,15(12,67)**	**60**	**1**
		D3	43,08(14,14)	32,84	17
		D4	56,15(13,59)	48,18	1
	M2	D1	41,54(15,04)	30,91	3
		D2	**66,15(12,67)**	**60**	**1**
		D3	46,15(12,56)	36,36	6
		D4	56,15(13,59)	48,18	1
	M3		60,77(21,28)	53,64	
Iris	M1	D1	**98,00(3,22)**	**97,69**	**5**
		D2	91,33(8,92)	90,01	3
		D3	96,67(4,71)	96,15	13
		D4	**98,00(3,22)**	**97,69**	**5**

	M2	D1	**98,00(3,22)**	**97,69**	**17**
		D2	91,33(8,92)	90	3
		D3	96,67(6,48)	96,15	15
		D4	**98,00(3,22)**	**97,69**	**5**
	M3		96,67(3,51)	96,15	
Seeds	M1	D1	90,95(8,82)	90	13
		D2	**95,71(4,74)**	**95,26**	**5**
		D3	92,86(7,53)	92,11	17
		D4	91,90(5,96)	91,05	17
	M2	D1	90,95(9,38)	90	10
		D2	**95,71(6,90)**	**95,26**	**10**
		D3	91,43(7,38)	90,53	17
		D4	92,38(6,02)	91,58	14
	M3		94,29(5,41)	93,68	
Survival	M1	D1	77,00(6,56)	75,36	15
		D2	76,67(6,29)	75	15
		D3	77,33(6,81)	75,71	7
		D4	**78,00(6,13)**	**76,43**	**12**
	M2	D1	75,67(7,21)	73,93	12
		D2	75,67(6,30)	73,93	11
		D3	77,00(6,93)	75,36	19
		D4	77,00(8,23)	75,36	19
	M3		76,00(6,44)	74,29	
Breast Cancer	M1	D1	93,93(3,49)	93,7	5
		D3	93,93(2,69)	93,7	5
		D4	93,04(3,62)	92,78	5
	M2	D1	93,57(3,69)	93,33	8
		D3	**94,29(3,45)**	**94,07**	**8**
		D4	92,68(3,31)	92,41	9
Credit Approval	M1	D1	71,91(7,02)	71,06	7
		D3	75,15(4,52)	74,39	3
		D4	71,91(5,78)	71,06	12
	M2	D1	73,68(5,52)	72,88	11
		D3	**78,97(5,93)**	**78,33**	**11**
		D4	73,38(5,57)	72,58	19
Dermatology	M1	D1	88,00(4,00)	87,3	1
		D3	92,86(5,26)	92,43	7
		D4	**95,71(3,37)**	**95,45**	**1**
	M2	D1	90,29(4,30)	89,7	1
		D3	94,86(3,51)	94,55	8
		D4	**95,71(3,37)**	**95,45**	**1**
Libras	M1	D1	78,33(7,90)	76,86	3
		D3	76,67(9,94)	75,13	3

		D4	**80,33(6,37)**	**79,02**	**1**
	M2	D1	79,00(6,86)	77,5	3
		D3	80,00(7,54)	78,57	3
		D4	**80,33(6,37)**	**78,93**	**1**
Wine	M1	D1	75,00(10,62)	71,43	1
		D3	82,50(9,68)	80	1
		D4	**84,38(6,75)**	**82,14**	**1**
	M2	D1	75,00(10,62)	71,43	1
		D3	82,50(9,68)	80	1
		D4	**84,38(6,75)**	**82,14**	**1**

Para as bases de dados *Breast Cancer*, *Credit Approval*, *Dermatology*, *Libras* e *Wine* os métodos M1, M2 com a distância de Mahalanobis e o classificador Bayesiano sob hipótese Gaussiana não foram testados, pois a quantidade de atributos dessas bases é um fator limitante a ferramenta desenvolvida.

As bases *Brainwave*, *Diabetes*, *Iris* e *Seeds* a metodologia proposta obtém o mesmo resultado que a K-NN, no entanto com uma quantidade de vizinhos menor. A Tabela 4-4 mostra a melhor taxa de acerto de cada base de dados utilizando o mesmo parâmetro *K* para ambos os métodos (distância Euclidiana), a Figura 4-7 mostra o gráfico gerado utilizando os dados dessa tabela.

Tabela 4-4 - Comparação do método proposto e K-NN para o mesmo K

Bases de dados	**Método Proposto**	**K-NN**	
	Taxa de Acerto		K
Boat	**100**	**100**	1
Half-Ring	73	**73,5**	3
Noisy-Lines	**99,5**	**99,5**	1
Petals	98,75	**100**	6
Saturn	92	**93**	1
Spirals	**100**	**100**	2
Balance-scale	**90,5**	88,66	6
Brainwave	**71,11**	66,11	7
Diabetes	**74,73**	74,61	17
Ecoli	**50**	**50**	1
Fertility	86,66	**87,78**	4
Glass	**84,44**	73,33	1
Hayes-Roth	36,15	**41,54**	3
Iris	**98**	96,66	5
Seeds	90,47	**90,95**	10
Survival	**76**	75,67	12
Breast_cancer	**93,92**	92,67	5
Credit_approval	71,02	**73,68**	11

Dermatology	88	**90,29**	1
Libras	78,33	**79**	3
Wine	**75**	**75**	1

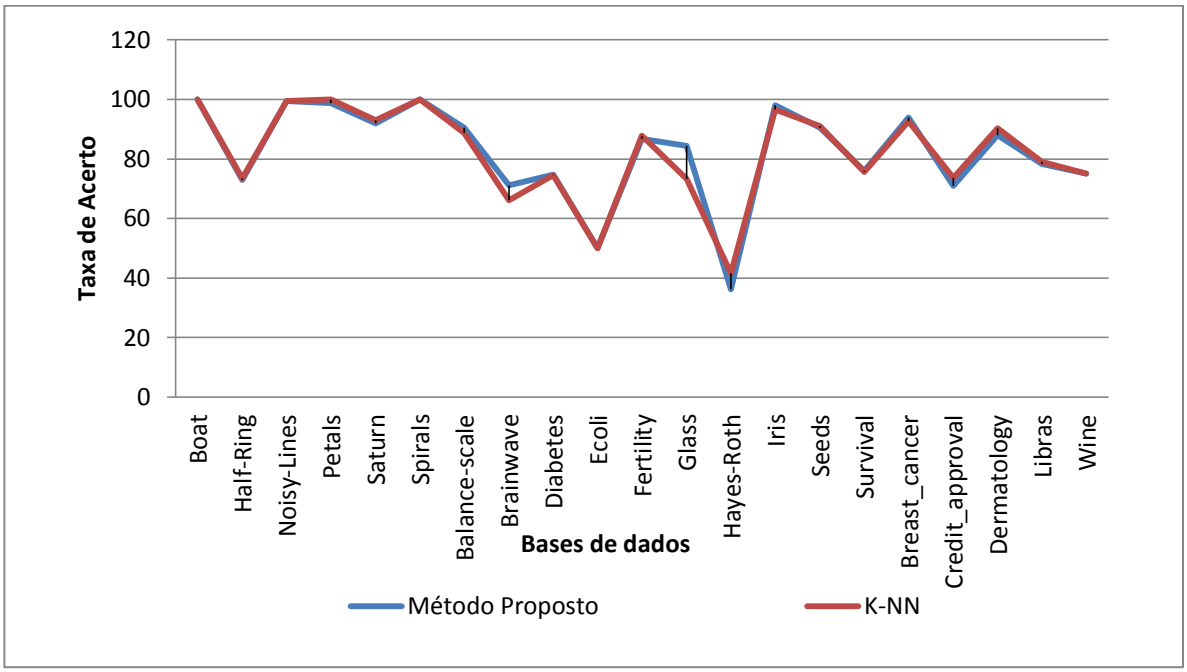


Figura 4-7 - Gráfico comparativo utilizando o mesmo parâmetro K

Em alguns casos o método proposto possui uma taxa maior de acerto, apesar disso ambos possuem comportamentos similares.

Todos os testes tiveram os seu tempo cronometrado, com o intuito de comparar o seu desempenho. A Figura 4-8 abaixo mostra o desempenho de cada classificador em função dos segundos. Em média o classificador proposto foi 37% mais lento que o método K-NN.

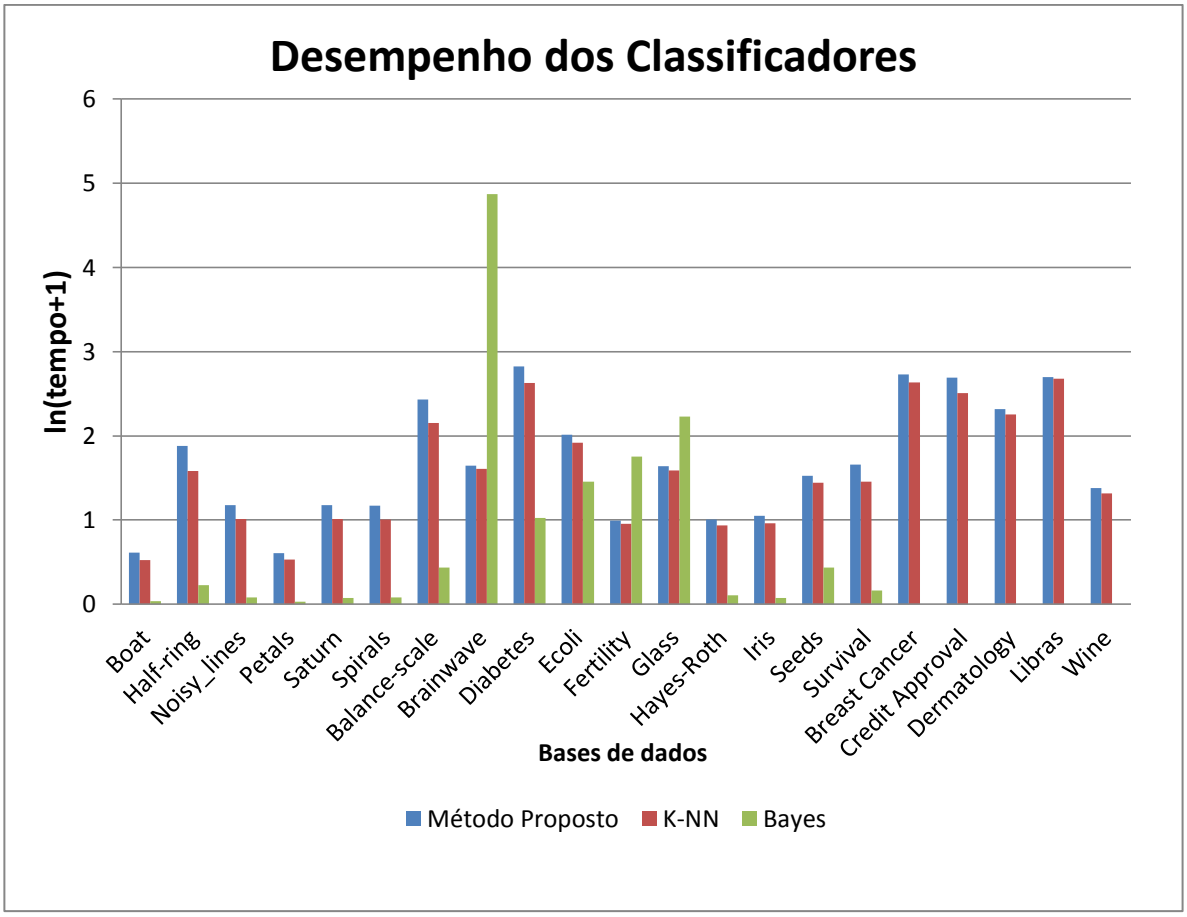


Figura 4-8 - Gráfico de desempenho

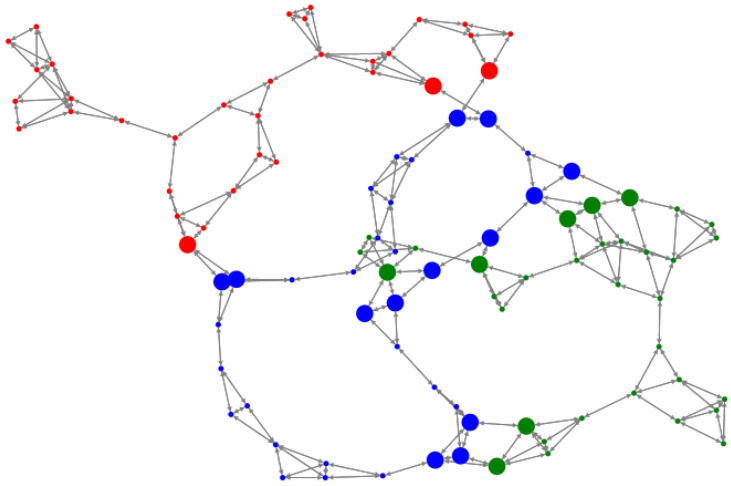
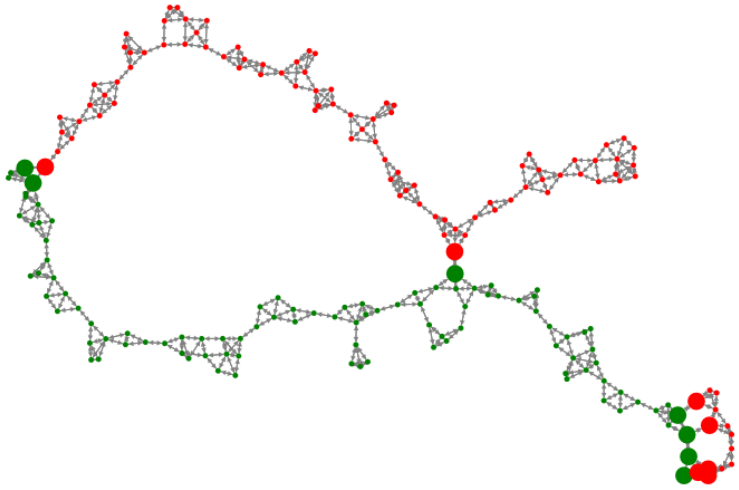
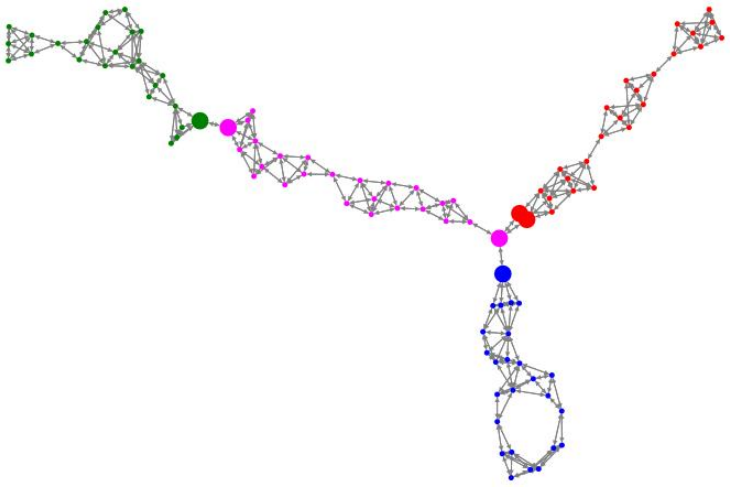
No Apêndice 2 encontra-se um link para todos os resultados gerados pela ferramenta, bem como as bases de dados.

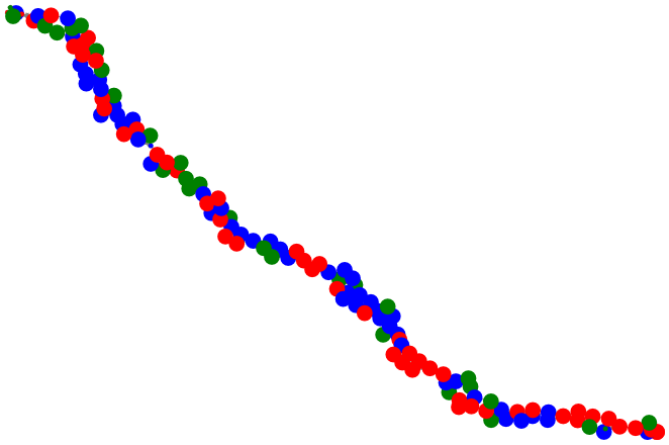
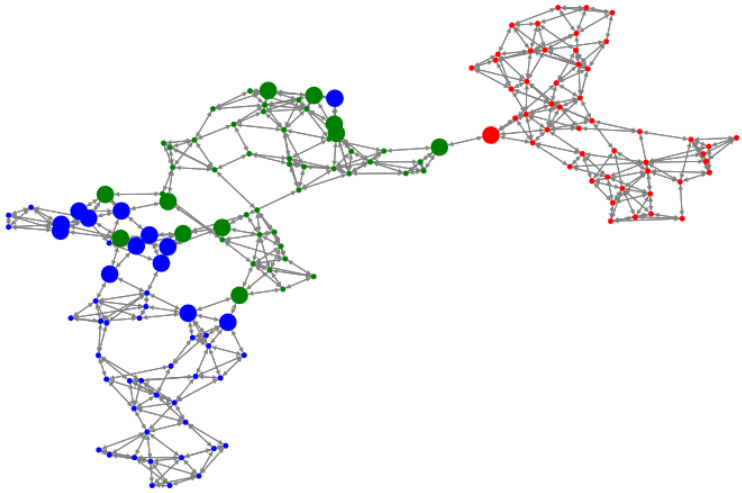
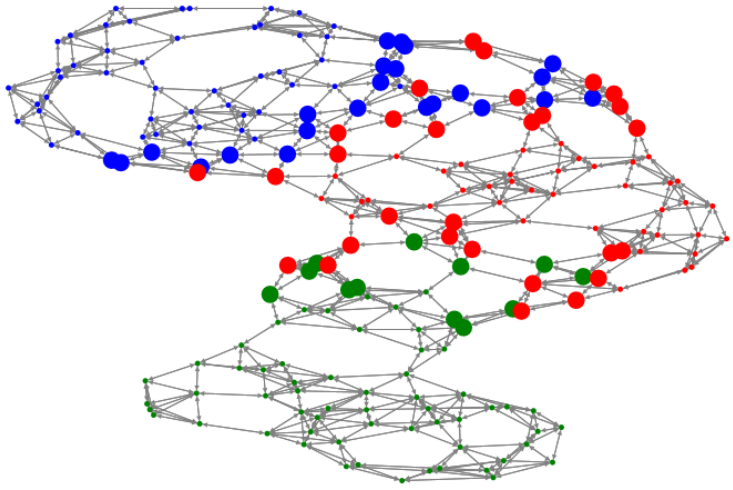
4.4 Analise da seleção dos protótipos

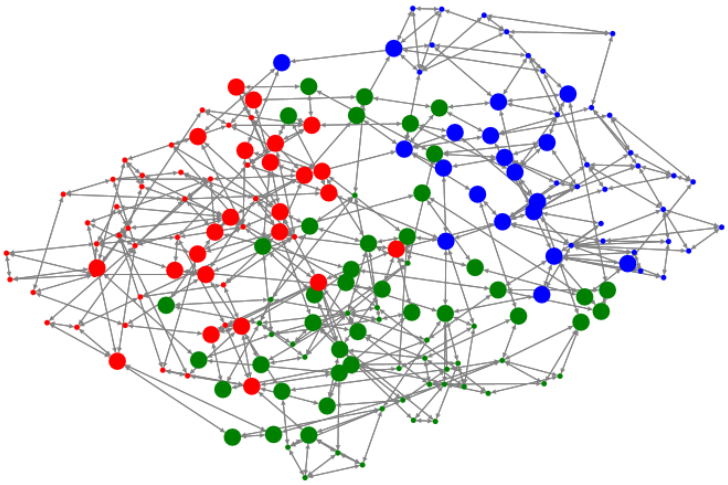
Para melhor ilustrar o método de classificação proposto foram realizadas análises geométricas a partir de alguns grafos gerados durante o processo de treinamento. A

Tabela 4-5 mostra a quantidade de protótipos e o grafo, com destaque para os protótipos.

Tabela 4-5- Construção do grafo e seleção dos Protótipos

Base de dados	Quantidade de Protótipos	Grafo
Boat	23	
Noisy_lines	14	
Petals	6	

Hayes-Roth	129	
Iris	25	
Seeds	64	

Wine	82	

A base de dados Hayes-Roth não possui uma fronteira de decisão bem definida, de modo que a metodologia proposta tende a eleger todos os vértices como protótipos. Sendo assim é justificável que ela tenha o mesmo desempenho que o método K-NN, uma vez que no caso limite de se ter o número de protótipos igual ao número de amostras, o método proposto converge exatamente para o KNN, mais precisamente ao 1-NN (note que nesse caso toda amostra será conquistada pelo vizinho mais próximo). No entanto, quando temos fronteiras de decisão bem definidas, o método é capaz de eleger poucos os protótipos, o que faz com que seu comportamento seja diferente do comportamento do método KNN.

Capítulo 5

CONCLUSÕES

Este trabalho teve como objetivo propor um classificador supervisionado, não paramétrico, baseado em grafos, com o propósito de investigar e delimitar as regiões de fronteiras entre classes por meio da seleção de protótipos informativos, utilizando o cálculo da informação de Fisher.

O método proposto se divide em três etapas: construção do grafo (Etapa 1), treinamento (Etapa 2) e teste (Etapa 3). Em uma base de dados o método primeiramente extrai os atributos e constrói o grafo a partir de uma relação de adjacência (K-NN ou E-ball). A relação de adjacência é um fator que tem influência direta no classificador, uma vez que a informação de cada amostra ou vértice depende inteiramente dos seus vizinhos.

A etapa 2 tem por objetivo delimitar a região de fronteira entre classes distintas, isso é feito utilizando o cálculo da informação de Fisher observada de cada vértice, de modo que os vértices que possuírem as maiores informações, ou seja, amostras que se encontram na margem das classes são selecionadas protótipos. Um vértice protótipo se caracteriza por estar conectado por uma ou mais arestas com um vértice de outra classe, que por sua vez também é um protótipo.

Após calculadas as informações dos vértices, cada aresta é ponderada com a soma dos vértices ligantes, o que garante ao método que vértices de classes distintas quando conectados por uma aresta terão um valor (distância) muito maior que quando conectados a vértices de mesma classe. Sendo assim o grafo possui uma característica muito importante para a etapa de classificação, arestas que conectam classes distintas possui alto valor, sendo assim cruzar bordas é altamente custoso, ou seja, se caso um protótipo de uma classe tente conquistar uma amostra pertencente a outra classe, necessariamente o mesmo deve cruzar ao menos uma borda, o que torna a conquista inviável. Por fim, a Etapa 3 classifica a amostra inserindo-a no grafo e fazendo a disputa entre os protótipos, o protótipo que oferecer o menor custo a rotula.

Na construção do grafo o método K-NN se mostrou mais simples de se trabalhar, pois não existe tanta variação do parâmetro (número inteiro) para as diferentes distâncias analisadas, para o método E-Ball o parâmetro raio (número real) possui grande variação o que dificulta os testes e investigações.

Nas bases de dados *Half-ring*, *Fertility* e *Survival* os resultados foram promissores. Mesmo que pontual foi possível observar uma melhora na taxa de acerto, o que nos fornece algumas expectativas de bom desempenho na aplicação do método em conjuntos de dados reais não explorados neste trabalho.

Apesar dos resultados terem sido similares ao K-NN, o método fornece uma representação visual do problema de classificação por meio de grafos, além de dar destaque às regiões de fronteira pela seleção dos protótipos, quantificando o nível de informação presente em cada amostra. Isso possibilita uma comparação objetiva entre os elementos do conjunto. Outro aspecto importante é a análise do grafo quanto a topologia das classes. Por exemplo, no caso da base *Seeds* a classe em verde não possui ligações com a classe azul, portanto são necessários somente dois limiares de separação para classificar essas amostras; o que não ocorre com a base de dados *Wine* que todas as classes possuem ligações, sendo necessários três limiares.

Em resumo as principais contribuições do método proposto foram:

- Classificação supervisionada baseada em grafos (é uma área de pesquisa em expansão e ainda pouco explorada).
- Método contextual baseado em campos aleatórios Markovianos.
- Utilização da informação de Fisher local no modelo de *Potts* como medida de similaridade intrínseca ao grafo (embora diferentes medidas de distâncias sejam utilizadas para induzir o grafo, o custo dos caminhos entre amostras do conjunto depende exclusivamente de características intrínsecas a ele).

Em trabalhos futuros o método pode ser utilizado para filtrar grandes bases de dados, através de um processo de seleção de amostras, onde as amostras de mais alta informação definiriam as bordas das classes (fronteiras) e as amostra de baixa informação definiriam a região central (comportamento esperado da classe).

A implementação de algoritmos para determinação automática dos parâmetros do método (distância, tipo do grafo e relação de adjacência) e a extensão do método para a classificação semi-supervisionada, utilizando uma etapa de pré-classificação das amostras também podem ser estudados, bem como a investigação da sensibilidade do método com a inclusão de uma etapa de extração de atributos (PCA ou LDA).

REFERÊNCIAS

BERTINI J. R. Jr.; ZHAO L.; MOTTA R.; LOPES A. A.. A nonparametric classification method based on K-associated graphs. Information Sciences: an International Journal, v.181 n.24, p.5435-5456, December, 2011.

BOYKOV Y.; VEKSLER O.; ZABIH R. Markov random fields with efficient approximations. In IEEE Computer Vision and Pattern Recognition Conference, p. 648-655, 1998.

CAWLEY, G. C. Leave-one-out cross-validation based model selection criteria for weighted LS-SVMS. Proc. Int. Joint Conf. Neural Netw., pp.1661 -1668 2006.

CLARK, J.; HOLTON D. A. A first Look at Graph Theory. Allied Publishers LTD.1995.

CONGALTON, R. G. A review of assessing the accuracy of classification of remotely sensed data. Remote Sensing of Enviroment, v. 37, n.1, p. 35-46, 1991.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R.; Stein, C.; et al. Introduction to Algorithms, 3rd Edition, MIT Press, 2009.

CULP M.; MICHAILIDIS G. Graph-Based Semi-Supervised Learning. IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 30, no. 1, pp. 174-179, 2008.

CUNNINGHAM P.; DELANY S. K-nearest neighbour classifiers. Technical report, UCD School of Computer Science and Informatics, 2007.

DUAN K.; KEERTHI S. S.. Which Is the Best Multiclass SVM Method? An Empirical Study. Proceedings of the Sixth International Workshop on Multiple Classifier Systems, p. 278-285, 2005.

DUDA, R.O.; HART P.E.; STORK D.G. Pattern Classification. Wiley-Interscience, 2ª edition, 2000.

EKLUND, P.W.; KIRKBY S.D.; POLLITT S.E. A Dynamic Multisource Dijkstra 's Algorithm for Vehicle Routing. Australian and New Zealand Conference on Intelligent Information Systems (ANZIIS '96), IEEE press, p.329-333, 1996.

FRAHLING, G.; SOHLER, C. A fast k-means implementation using coresets. Proceedings of the twenty-second annual symposium on Computational geometry (SoCG), p. 135-143, 2006.

HAYKIN, S. Redes neurais: princípios e prática. trad. Paulo Martins Engel. - 2.ed. - Porto Alegre: Bookman, 2001.

JIANG D.; TANG C.; ZHANG A. Cluster analysis for gene expression data: a survey. IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 11, pp. 1370-1386, 2004.

KIM, J. Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. Computational Statistics & Data Analysis, v.53 n.11, p.3735-3745, September, 2009.

KOHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (pp. 1137–1143). San Francisco, CA: Morgan Kaufmann,1995.

KRUSKAL J.B. On the shortest spanning subtree of a graph and the traveling salesman problem. Proc. Amer. Math. Soc.,1956.

KULIS B.; BASU S.; DHILLON I.; MOONEY R. Semi-supervised Graph Clustering: A Kernel Approach. Machine Learning, 2009.

LEVADA, A. L. M.; MASCARENHAS, N. D. A.; TANNUS, A. A Novel MAP-MRF Approach for Multispectral Image Contextual Classification using Combination of Suboptimal Iterative Algorithms. Pattern Recognition Letters, v. 31, p. 1795-1808, 2010.

LEVADA, A. L. M.; MASCARENHAS, N. D. A.; TANNUS, A. A Novel Pseudo-Likelihood Equation for Potts MRF Model Parameter Estimation in Image Analysis. In: 15th IEEE International Conference on Image Processing (ICIP), 2008, San Diego. Proceedings of the 15th IEEE International Conference on Image Processing (ICIP). Bryan, TX: Conference Management Services, Inc., 2008. p. 1828-1831.

LEVADA, A. L. M.; MASCARENHAS, N. D. A.; TANNUS, A. Statistical Inference on Markov Random Fields: Parameter Estimation, Asymptotic Evaluation and Contextual Classification of NMR Multispectral Images. In: Peng-Yeng Yin. (Eds.). Pattern Recognition, 1 ed., INTECH Open, 2009, p. 223-248.

LONG B.; WU X.; ZHANG Z.; YU P. S. Unsupervised learning on k-partite graphs. Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, Philadelphia, PA, USA, p. 317-326, 2006.

MATSUO Y.; SAKAKI T.; UCHIYAMA K.; ISHIZUKA M. Graph-based word clustering using a web search engine. Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, Sydney, Australia, p. 542-550, 2006.

MELGANI, F.; BRUZZONE, L. Classification of Hyperspectral Remote Sensing Images with Support Vector Machines. IEEE Transactions on Geoscience and Remote Sensing, vol. 42, No. 8, p. 1778-1790, 2004.

NGUYEN, H. V.; BAI, L. Cosine similarity metric learning for face verification. Proceedings of the 10th Asian conference on Computer vision Queenstown, New Zealand, p.709-720, 2010.

PAPA J. P.; FALCÃO A. X.; SUZUKI C. T. N. Supervised pattern classification based on optimum-path forest, International Journal of Imaging Systems and Technology, v.19 n.2, p.120-131, June 2009.

PAVAN M.; PELILLO M. A new graph-theoretic approach to clustering and segmentation. In CVPR , Vol. I, pp. 145-152, 2003.

PRIM R.C. Shortest connection networks and some generalizations. Bell System Tech. J., 1957.

PRINCIPE J. C.; XU D. Introduction to information theoretic learning. Proc. Int. Joint Conf. Neural Networks ('IJCNN'99), pp.1783 -1787, 1999.

RODRÍGUEZ J. D.; PÉREZ A.; LOZANO J. A. Sensitivity Analysis of k-Fold Cross Validation in Prediction Error Estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 3, pp. 569-575, 2010.

RUSSEL S.; NORVIG P. Artificial Intelligence, a modern Approach. 2ª edition, Prentice Hall, 2003.

SCHAEFFER S E. Graph clustering. Computer Science Review 1 (1), 27-64, 2007.

STEEN, M. S. Graph Theory and Complex Networks – An Introduction. 1ª edition, Maarten van Steen, 2010.

SZUMMER M.; JAAKKOLA T. Partially labeled classification with markov random walks. Advances in Neural Information Processing Systems, v.4.MIT, p. 945-952, 2001.

THEODORIDIS, S.; KOUTROUMBAS, K. Pattern Recognition. 5. Ed. Academic Press, 2009.

VIAENE S.; DEDENE G.; DERRIG R. A. Auto claim fraud detection using Bayesian learning neural networks, Expert Systems with Applications: An International Journal, v.29 n.3, p.653-666, October, 2005.

VISHWANATHAN S.; SCHRAUDOLPH N.; KONDOR R.; BORGWARDT K. Graph kernels. Journal of Machine Learning Research, 11, p.1201-1242, 2010.

WEBB, A. Statistical Pattern Recognition. 2ª edition London: Arnold, 2002.

ZHOU D.; SCHOLKOPF B. Learning from labeled and unlabeled data using random walks. Proceeding of the 26th DAGM Symposium on Pattern Recognition, pp 237-244, 2004.

APÊNDICE 1 : DERIVAÇÃO DA INFORMAÇÃO DE FISHER OBSERVADA LOCAL NO MODELO DE POTTS

Considere um campo aleatório Markoviano definido num grafo G=(V,E) com V={v_1, v_2,..., v_n} e um sistema de vizinhança η_i. A informação de Fisher observada local (do tipo I) para a observação x_i com relação ao parâmetro β (inverso da temperatura) é definida em termos da função densidade condicional local como:

$$\varphi(x_i) = \left[\frac{\partial}{\partial\beta}\log p(\, x_i|\eta_i, \beta)\right]^2 \quad (1)$$

onde x_i corresponde ao valor do rótulo no vértice v_1.

A seguir é mostrada a derivação de $\beta(x_i)$ em um modelo de Potts isotrópico de interação por pares. Inserindo a função densidade condicional local do modelo de Potts na equação (1) e após alguma álgebra, tem-se:

$$\varphi(x_i) = \left[\frac{\partial}{\partial\beta}\log p(\, x_i|\eta_i, \beta\,)\right]^2 = \left[U_i(x_i) - \left(\frac{\sum_{l=1}^{q} U_i(l)e^{\beta U_i(l)}}{\sum_{l=1}^{q} e^{\beta U_i(l)}}\right)\right]^2 \quad (2)$$

$$= \left[\frac{\sum_{l=1}^{q}\sum_{k=1}^{q}(U_i(x_i) - U_i(l))(U_i(x_i) - U_i(k))e^{\beta(U_i(l)+U_i(k))}}{\sum_{l=1}^{q}\sum_{k=1}^{q} e^{\beta(U_i(l)+U_i(k))}}\right]$$

APÊNDICE 2 : FERRAMENTA DESENVOLVIDA E RESULTADOS DOS TESTES

Para testar a ferramenta desenvolvida e analisar os resultados neste trabalho apresentado, o link: https://dl.dropboxusercontent.com/u/101141686/Baixar.zip disponibiliza para download todo o material utilizado, que inclui a ferramenta desenvolvida, as bases de dados e os resultados.