Universidade Federal de Pernambuco
Centro de Informática - CIn

Graduação em Ciência da Computação

Aprendizado por Quantização Vetorial usando Distâncias Adaptativas

Telmo de Menezes e Silva Filho

Trabalho de Graduação

Recife
17 de novembro de 2009

Universidade Federal de Pernambuco
Centro de Informática - CIn

Telmo de Menezes e Silva Filho

Aprendizado por Quantização Vetorial usando Distâncias Adaptativas

Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Centro de Informática - CIn da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: *Profa. Dra. Renata Maria Cardoso Rodrigues de Souza*

Recife
17 de novembro de 2009

A meus pais, minhas irmãs e todos que me acompanharam
e me apoiaram até aqui.

Agradecimentos

Chego, então, a um grande marco na vida de um estudante: o Trabalho de Conclusão de Curso. Foram vinte anos e muitas experiências até esse momento. Antes de qualquer coisa, devo agradecer àqueles que são os maiores responsáveis por eu ter chegado a esse ponto: meus pais. Agradeço a Deus, pois jamais poderia ter tido melhores pais. Esforçados, diligentes, inteligentíssimos e competentes em tudo que fazem, sempre fizeram de tudo para dar à sua família o melhor que pudessem. Ensinaram-me a ser um homem correto e que para tudo há a hora certa. Amigos e sempre presentes, nunca me deixaram desamparado. Ajudaram-me com tarefas de escola (e como me cobravam isso!), me castigaram em momentos de erro e me parabenizaram em momentos de acerto. Tenho certeza de que não poderia ter tido melhor criação e espero ser, um dia, um pai tão bom quando eles foram para mim. Peço desculpas se algum dia eu os decepcionei e aproveito que escrevendo é mais fácil para dizer o que não costumo dizer a eles, embora sejam eles os maiores merecedores disso no mundo: Pai, Mãe, eu amo vocês dois.

Devo citar também minhas duas irmãs, Talita e Larissa, por terem (apesar das discussões, bem mais comuns em tempos passados) me dado apoio quando vim morar em Recife, longe de meus pais. Reconheço aqui que em grande parte das discussões que tivemos, elas estavam tentando me ensinar algo ou me mostrar que errei em algum momento e eu, relutante, acabava brigando. Muito do que sei devo a elas e tenho certeza de que também não poderia ter tido irmãs melhores.

Costumo dizer que a vida de uma pessoa vai passando cada vez mais rápido e ficando cada vez mais complexa. O Ensino Fundamental e os dois primeiros anos do Ensino Médio parecem durar uma eternidade. A criança (e mais tarde o adolescente) sente como se aquela rotina de escola nunca fosse acabar, a calmaria de fazer provas de quando em quando, sem preocupações. Então vem o terceiro ano do Ensino Médio e o Vestibular aparece como primeira grande atribulação na vida do estudante. O ano passa rápido: é como se houvesse passado um mês dos anos anteriores. Na faculdade as coisas pioram ainda mais e a sensação é de que toda a duração da graduação é menor do que a do pré-vestibular. O curso não é fácil: não há tempo para nada, pesquisas, projetos, provas e noites em claro e, quando se percebe, já chegou o fim.

Agradeço à Professora Renata, pelos 2 anos de atenção e orientação, período em que fui monitor da cadeira de Estatística e comecei a me aventurar com a tal da pesquisa científica.

Todas as fases citadas, bem como as pessoas que me acompanharam nelas, foram de uma importância sem fim para eu ter conseguido chegar onde cheguei. Muito obrigado a meus amigos e minha namorada, pelo apoio e por não me deixar perder a cabeça ante às dificuldades do curso.

Assim como são as pessoas, são as criaturas!
—DIDI MOCÓ SONRISÉLPIO COLESTEROL NOVALGINO MUFUMBBO (Os Trapalhões)

Resumo

O aprendizado por quantização vetorial (*AQV*) é uma família de algoritmos baseados em protótipos proposta por Kohonen que tem sido extensivamente estudada e aplicada em diversas áreas, devido a sua simplicidade e eficiência. O *AQV* básico visa a achar regiões de classe discriminativas no espaço de dados de entrada, representadas por subconjuntos de protótipos.

O objetivo desse trabalho é fazer um estudo sobre o uso de distâncias adaptativas que considerem características das regiões de classe representadas por subconjuntos de vetores de protótipos e alcancem resultados melhores em comparação ao algoritmo *AQV* original, que usa distância Euclidiana padrão. São utilizados três tipos de distâncias adaptativas: uma distância com pesos diferentes para cada variável e única para todos os protótipos, uma distância com pesos diferentes para cada variável e que é diferente para cada classe e uma distância com pesos diferentes para cada variável e que é diferente para cada protótipo.

Palavras-chave: Aprendizado por Quantização Vetorial, Aprendizado por Protótipos, Aprendizagem de Máquina, Distância Adaptativa

Abstract

Learning Vector Quantization (LVQ) is a family of algorithms based on prototypes proposed by Kohonen which has been extensively studied and applied in many areas, due to their simplicity and efficiency. The basic LVQ aims at finding discriminative class regions in the input data space, which are represented by subsets of prototypes.

The goal of this work is to make a study about the utilization of adaptive distances which take into account the features of the class regions represented by subsets of prototypes and reach better results when compared to the original LVQ, which uses standard Euclidean distance. Three different types of adaptive distances are used: one distance has different weights for every variable and is unique to all the prototypes, one has different weights for every variable and is different for each class and one has different weights for each variable and is different for each prototype.

Keywords: Learning Vector Quantization, Prototype Learning, Machine Learning, Adaptive Distance

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

O Aprendizado por Quantização Vetorial (a partir de agora referido como AQV, neste trabalho) é uma família de algoritmos baseados em protótipos proposta por Kohonen [5] que tem sido extensivamente estudada e aplicada em diversas áreas, devido a sua simplicidade e eficiência. O AQV básico visa a encontrar regiões de classe discriminativas no espaço de dados de entrada, representadas por subconjuntos de protótipos. O algoritmo de aprendizagem começa escolhendo um subconjunto de protótipos para cada classe (de forma aleatória) e, iterativamente, procura pelo protótipo que é o vizinho mais próximo de cada padrão. Esse protótipo é atualizado, de acordo com o resultado da classificação, de forma que a regra do vizinho mais próximo minimize a probabilidade de erro de classificação média esperada. Quando as iterações param, os protótipos atualizados que representam as regiões de classe devem estar próximos aos padrões de treinamento em sua classe. A classe de um novo padrão é a mesma do seu protótipo mais próximo.

A família de algoritmos AQV consiste de algumas modificações do procedimento AQV básico, com o intuito de alcançar uma melhor aproximação das regiões de decisão ótimas de Bayes, convergência mais rápida ou mais robusta ou incorporar métricas mais flexíveis ([1], [4] - [5], [7]). A distância entre os pontos de dados, determinada pela métrica escolhida, tem um papel muito importante no AQV e em algoritmos de cluster. Além disso, clustering particional [6], bem como o AQV, fornece partições naturais dos dados de entrada em classes ou regiões de classes que tentam otimizar um critério (por exemplo, a soma de inércias intra-classe no algoritmo k-means e probabilidade de erro de classificação média esperada no AQV). Entretanto, se esses métodos admitem um modelo estático de distância, isso pode torná-los ineficientes em capturar adequadamente características das classes em situações em que o conjunto de dados contém grupos de diferentes formas e tamanhos ou formas iguais, mas volumes diferentes.

Para resolver o problema, De Carvalho *et al* apresentou métodos fuzzy de clustering particional utilizando duas formas de distâncias que mudam a cada iteração para representar melhor as regiões de classes: uma distância única (nesse trabalho referida como distância simples) que é a mesma para todos os protótipos e outra distância que não só muda a cada iteração, como também é diferente para cada classe. Essas distâncias são chamadas de distâncias Euclidianas adaptativas. Nesse trabalho, características dos clusters são capturadas pela dispersão dos elementos pertencentes aos clusters, com relação aos seus protótipos. A importância dessas distâncias se deve ao fato de que ela leva em consideração a estrutura interna dos clusters e, dessa forma, o algoritmo é capaz de reconhecer classes de formas e tamanhos diferentes, além de variáveis de comportamentos diferentes.

A contribuição deste trabalho de graduação é introduzir o uso de três tipos de distâncias

adaptativas em conjunto com um algoritmo da família AQV. As distâncias utilizadas são: uma distância simples, única para todos os protótipos, uma distância por classe, que tem pesos diferentes para cada variável e para cada classe, e uma distância por protótipo que tem pesos diferentes para cada variável e para cada protótipo. Aqui, a regra de aprendizagem é baseada no Aprendizado por Quantização Vetorial Otimizado (AQV1O - uma versão otimizada do AQV1 introduzida por Kohonen [5]) e a busca do protótipo mais próximo é realizada admitindo-se distâncias euclidianas adaptativas. O objetivo é usar distâncias que considerem características das regiões de classes representadas por subconjuntos de vetores de protótipos e alcance um melhor desempenho em comparação ao AQV1 original, que usa distância euclidiana padrão.

O trabalho está organizado da seguinte maneira: o Capítulo 2 apresenta uma breve explanação sobre Aprendizagem Supervisionada, o Capítulo 3 explica de forma mais detalhada o funcionamento do AQV básico, bem como do AQV1O e explica superficialmente alguns outros algoritmos da família AQV. No Capítulo 4 são apresentadas as três distâncias adaptativas utilizadas e, no Capítulo 5, é apresentado o classificador AQV1O que as utiliza. O Capítulo 5 também descreve uma análise de desempenho, considerando dois conjuntos de dados simulados com classes superpostas. A avaliação de desempenho é baseada na acurácia de predição avaliada numa simulação Monte Carlo com 100 replicações. Esse mesmo capítulo mostra uma análise de performance considerando sete conjuntos de dados reais e a acurácia de precisão é avaliada por um método de validação cruzada com 10 partições. Por último, o Capítulo 6 traz as observações finais.

CAPÍTULO 2

Aprendizagem Supervisionada

Este trabalho baseia-se em um algoritmo da família AQV e essa família consiste de algoritmos de Aprendizagem Supervisionada baseada em protótipos. Portanto, é interessante analisar brevemente o que já está consolidado na literatura desses campos, a fim de situar e contextualizar melhor o método adiante proposto.

O objetivo da Aprendizagem Supervisionada (*AS*) é construir um modelo da distribuição das classes a partir de características de predição, fornecidas pelos exemplos extraídos de um conjunto de treinamento. O classificador resultante passar a ser usado para designar rótulos de classes para instâncias de teste cuja classe é desconhecida, embora as características de predição sejam conhecidas [8]. Alguns dos algoritmos que baseiam-se em *AS* são: redes neurais, Support Vector Machines, Classificadores Estatísticos e Classificadores baseados em protótipos. Cada um deles tem formas diferentes de atingir o objetivo da *AS*, mas não se pode dizer se há um melhor do que os outros, pois isso varia de problema em problema.

2.1 Redes Neurais

Consiste de um grande número de unidades (neurônios artificiais) que partilham conexões e são geralmente divididos em três grupos: unidades de entrada, que recebem a informação, unidades de saída, que disponibilizam o resultado do processamento da rede, e as unidades internas, mais conhecidas como unidades escondidas [9]. Durante a classificação, o sinal de entrada se propaga a partir dos neurônios de entrada e passa por toda a rede. Cada neurônio da rede tem um valor de ativação e, para passar o sinal adiante, sua função de ativação deve exceder esse valor. O cálculo da função de ativação envolve a soma das contribuições de todos os neurônios que mandaram seus sinais para a unidade em questão. A cada contribuição de cada neurônio é multiplicada por um peso, que define as conexões entre unidades [9].

O processo de treinamento de uma Rede Neural baseada em Perceptron consiste em utilizar o erro da rede para ajustar os pesos das conexões entre os neurônios, bem como seus valores de ativação.

Um problema comum ao se trabalhar com Multilayer Perceptron é a determinação da quantidade de neurônios na camada escondida. Um número pequeno de neurônios pode deixar a rede com baixa capacidade de generalização (underfitting) e o contrário pode levar ao overfitting [9].

2.2 Support Vector Machines (SVM)

SVM é a técnica de Aprendizagem Supervisionada mais nova e está entre as mais robustas e eficientes ([8], [9]). *SVM* baseia-se na idéia de criar uma margem, ou seja, maximizar a mínima distância entre o hiperplano que separa as classes (o *SVM* básico suporta apenas problemas binários) e as instâncias de dados. Isso já se mostrou capaz de reduzir o erro esperado de generalização [9] A questão é que a maior parte dos problemas reais não pode ser separados por hiperplanos em seu espaço de dados. *SVM*, então, introduz o conceito de espaço transformado de dados, cuja idéia é aumentar a dimensão dos dados até torná-los separáveis por um hiperplano. Com isso, qualquer conjunto consistente pode se tornar separável [9]. Os detalhes de como isso é feito no *SVM* são um tanto quanto complexos e fogem ao escopo deste trabalho.

2.3 Classificadores Estatísticos

Técnicas multivariadas e que podem ser divididas em duas abordagens: paramétrica (quando é necessário fazer suposições sobre a distribuição dos dados) e não-paramétrica.

O Naive Bayes é uma técnica paramétrica que se baseia na probabilidade a posteriori para classificar um objeto.

Os métodos Kernel e k-Vizinhos são técnicas não-paramétricas que se buscam estimar uma função de densidade local para classificar os objetos e não fazem suposições sobre a distribuição dos dados.

2.3.1 Naive Bayes

Método baseado no conceito de Máxima Probabilidade a Posteriori [8]. A idéia é de que, dado um problema com um conjunto de classes (cujas probabilidades a priori são conhecidas), pode-se designar como classe de uma instância x a classe que tiver maior probabilidade a posteriori para a presença de x. Essa probabilidade a posteriori pode ser calculada com o Teorema de Bayes, mas isso pode ser difícil, dadas as probabilidades condicionais dos atributos. No entanto, para simplificar, o Naive Bayes desconsidera essas dependências e tem se mostrado eficiente, apesar dessa simplificação [8].

2.3.2 Função Kernel

O Método Kernel busca estimar funções de densidades não-paramétricas para classificar as observações (Souza, 1999). A função Kernel estima a densidade $f(x)$ (caso unimodal) através de uma amostra aleatória $x_1, x_2, \ldots, x_n$, por:

$$f(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{2.1}$$

onde h é a amplitude do intervalo (x-h,x+h) centrado em x.

$K(x, x_i, h)$ é a função Kernel e vários critérios para essa função existem. Assim, a equação anterior pode ser reescrita como: $f(x) = \frac{1}{nh} \sum_{i=1}^{n} K(x, x_i, h)$ e h passa a ser chamado de parâmetro

de suavização. Para usar o Método Kernel, é necessário especificar a função Kernel e o parâmetro de suavização, que pode ser escolhido pela otimização do erro médio quadrático.

Um dos critérios que podem ser escolhidos para a função Kernel é a janela de Parzen (Michie et al., 1994):

$$K(x,x_i,h) = \frac{1}{\sqrt{2h}} \exp\left\{ -\frac{1}{2}\left(\frac{x-x_i}{h}\right)^2 \right\} \tag{2.2}$$

Nesse caso, um h muito grande faz com que $K(x,x_i,h)$ mude muito lentamente com relação a x, suavizando a estimativa de $f(x)$. Por outro lado, um h pequeno faz com que seja produzida uma estimativa muito irregular de $f(x)$, pois $f(x)$ será uma superposição de distribuições normais com variâncias pequenas centradas na amostra.

2.3.3 k-Vizinhos mais Próximos (k-NN)

Um dos mais antigos algoritmos de classificação não-paramétricos [8]. Sua implementação é bastante simples: para classificar um exemplo, calcula-se sua distância para todas as instâncias de treinamento, para achar os k mais próximos e a classe do exemplo é definida como a classe mais freqüente entre os k vizinhos mais próximos.

Essa forma de classificação pode trazer problemas. Por exemplo, supondo na situação demonstrada na Figura 2.1, que o parâmetro k tem valor 3 e que a classe verdadeira do exemplo representado pelo círculo verde seja a classe triângulo. Com esse valor de k o algoritmo iria identificar como vizinhos mais próximos 2 triângulos e 1 quadrado, ou seja, a classe da maioria é triângulo e a classificação do exemplo seria correta. Mas, se o valor de k fosse 5, seriam identificados mais dois quadrados e a classificação do exemplo seria incorreta.

Percebe-se que é preciso achar um valor de k que evite erros como esse. Valores altos de k geralmente reduzem os efeitos do ruído nos dados, mas tornam as fronteiras entre as classes menos distintas. Além disso, não se escolhe valores pares para k, pois isso poderia levar a empates na escolha da classe.

Uma adaptação que é feita ao $k-NN$ básico para evitar o problema relatado acima é a ponderação das distâncias, com isso, elementos mais distantes influenciam menos na escolha da classe do exemplo. O peso para cada elemento pode assumir um valor $1/d$ em que d é o valor da distância do elemento ao padrão de exemplo. A idéia de vizinho mais próximo está por trás dos algoritmos AQV, pois, no seu processo de treinamento, procura-se pelo protótipo mais próximo ao exemplo.

2.4 Classificadores baseados em protótipos

Outro conceito fundamental para o AQV é a Aprendizagem baseada em protótipos, em que um subconjunto de protótipos é escolhido no espaço de dados de entrada para representar as regiões de classes. O processo de treinamento de algoritmos baseados em protótipos consiste em tentar posicionar os protótipos de forma ótima, ou seja, que aproxime a melhor representação das classes. Para classificação, identifica-se o protótipo mais relacionado ao dado de

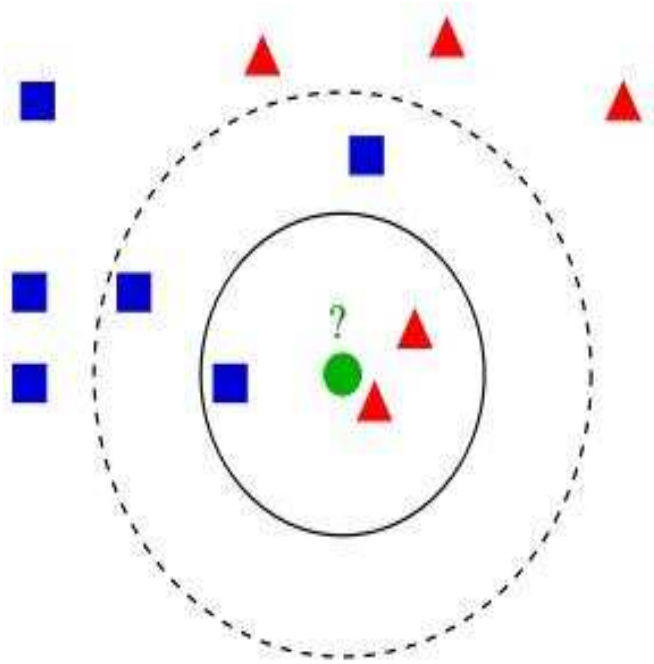

Figura 2.1 Exemplo de k-NN, circunferência menor representa $k = 3$ e circunferência maior representa $k = 5$

exemplo e assume-se que a classe do dado é a mesma do protótipo. Pode-se citar nesse campo o classificador baseado em k-Médias.

2.4.1 Classificador baseado em k-Médias

O algoritmo k-Médias (em inglês k-Means) original é um algoritmo de clustering não-supervisionado, que define partições e centros de partições em um conjunto de dados sem rótulos de classes. O k é um parâmetro que representa o número de centros de partições que o algoritmo deve encontrar e o algoritmo move os centros com o intuito de minimizar a variação interna das partições. Assim, dado um conjunto inicial de centros (tipicamente escolhidos aleatoriamente entre as observações do conjunto de treinamento), o algoritmo k-Means alterna os seguintes passos até sua convergência:

1. para cada centro, identificar o subconjunto de pontos que são mais próximos a ele do que a qualquer outro centro;

2. computa-se a média de cada atributo para os pontos em cada partição e esse vetor de médias torna-se o novo centro daquela partição.

Uma alteração interessante do k-Médias é a adaptação do algoritmo para torná-lo um classificador supervisionado [10]. Com isso, os passos do novo algoritmo são três:

1. aplicar o k-Médias particional para os dados de cada classe separada, usando R protótipos por classe;

2. designar um rótulo de classe para cada um dos protótipos;

3. classificar um novo padrão x para a classe do protótipo mais próximo.

As figuras 2.2 e 2.3 mostram um exemplo com três classes e dois atributos. São usados $R = 5$ protótipos por classe. Pode-se perceber na figura 2.2 que uma certa quantidade dos protótipos aproximam-se das fronteiras de decisão, o que ocorre devido à forma como o classificador baseado em k-médias é construído: como para cada classe utiliza-se um processo k-médias separado, uma classe não tem como afetar o posicionamento dos protótipos da outra. Esse posicionamento dos protótipos pode levar a erros de classificação para elementos próximos às fronteiras de decisão. Na figura 2.3, um processo *AQV* foi realizado com os protótipos resultantes de um classificador baseado em k-Médias anterior. Nota-se que os protótipos foram afastados das fronteiras de decisão. A curva pontilhada é a fronteira de decisão de Bayes.

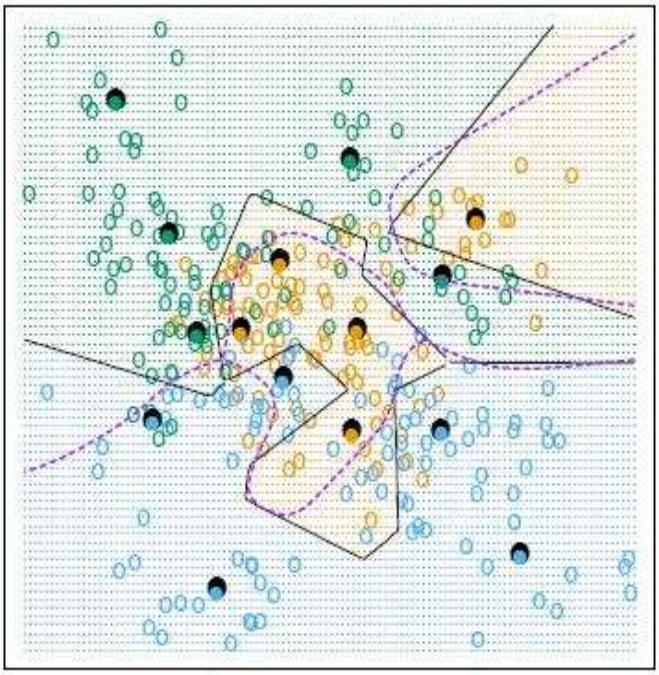

Figura 2.2 k-Médias com 5 protótipos por classe

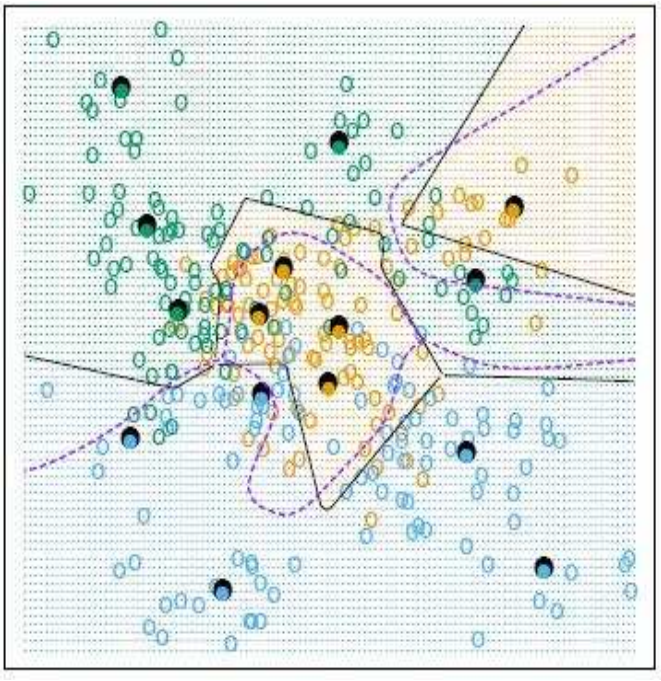

Figura 2.3 AQV com 5 protótipos por classe

CAPÍTULO 3

Aprendizado por Quantização Vetorial

Como já foi citado nesse trabalho, o nome Aprendizado por Quantização Vetorial se refere a toda uma família de algoritmos, como AQV1, AQV2, AQV3 e AQV1O. O AQV foi proposto por Kohonen, o mesmo criador dos Mapas Auto-organizáveis (em inglês, Self-organizing maps ou SOM). AQV e SOM, são de certa forma relacionados, pois são baseados em protótipos e seus processos de convergência consistem em fazer os protótipos se distribuirem pelo espaço de dados, a fim de representarem melhor as regiões de classes [5]. No entanto, o AQV é um algoritmo de aprendizado supervisionado e o SOM é não-supervisionado. Além disso, o algoritmo SOM define vizinhanças ao redor do neurônio "vencedor", de forma que os neurônios da vizinhança também são afetados em certo grau. Esse conceito não é utilizado no AQV básico.

3.1 Decisão Ótima

O problema da decisão ótima ou reconhecimento estatístico de padrões consiste tradicionalmente em aproximar a função que indica a probabilidade de uma amostra x pertencer a determinada classe C, com a mínima taxa de erros de classificação. Ao achar essa função, $f(x)$, admite-se que x pertence à classe que tem o valor máximo de $f(x)$.

O AQV, no entanto, é baseado numa filosofia completamente diferente. Supondo que e xistem $S_1,\ldots,S_K$ classes. Seja $\Omega = \{(x_i,y_i)\}(i = 1,\ldots,n)$ um conjunto de treinamento. Cada padrão i é descrito por um vetor de p atributos quantitativos $x_i = (x_i^1,\ldots,x_i^p)$ e um atributo discreto de classe Y que toma valores no conjunto discreto $1,\ldots,K$. Seja $\{(w_m,y_m)\}(m = 1,\ldots,M)$ um conjunto de protótipos, cada um representado por um vetor de p atributos quantitativos $w_m = (w_m^1,\ldots,w_m^p)$ e um atributo discreto de classe Y que toma valores no conjunto discreto $1,\ldots,K$.

Assume-se que, para cada classe $k(k = 1,\ldots,K)$, é designado um subconjunto de protótipos, de forma que o número total de protótipos é M. Então, para cada padrão de treinamento x, encontra-se o protótipo w que tem a menor distância Euclidiana de x. Os protótipos podem ser escolhidos de tal forma que aqueles que pertencem a classes diferentes não se misturem, embora as distribuições das classes se justaponham. Com isso, apenas os protótipos mais próximos das regiões de bordas das classes são, de fato, importantes para a decisão ótima. Então uma boa aproximação de $f(x)$ não é necessária em todos os protótipos. É mais importante, portanto, posicionar os protótipos no espaço de dados de forma que a regra do vizinho mais próximo utilizada para a classificação minimize a probabilidade de erro esperada.

3.2 O AQV1

Supondo que um certo número de protótipos é designado para cada classe k, pela regra do 1-vizinho mais próximo, determina-se que o padrão i extraído de Ω pertence à mesma classe à qual pertence o protótipo m mais próximo a i. Assim,

$$c = \arg\min_m \left\{ d_E(x_i, w_m) = \sqrt{\sum_{j=1}^{p} (x_i^j - w_m^j)^2} \right\} \tag{3.1}$$

define o índice do w_m mais próximo a x_i de acordo com a distância Euclidiana d_E. Logo, c, o índice do "vencedor", depende do padrão x_i e de todos os protótipos w_m.

Seja $x_i(t)$ uma instância de dados e $w_m(t)$ os valores seqüenciais de w_m nos instantes $t = 0, 1, 2, \ldots$. Tais valores podem ser achados, de forma a minimizar a taxa de erros de classificação, segundo o processo de aprendizagem a seguir.

1. **Inicialização**

   1.1 Seleciona-se aleatoriamente M padrões distintos do conjunto de treinamento Ω, produzindo o subconjunto inicial de protótipos $\{w_1(t), \ldots, w_M(t)\}$ no passo $t = 0$.

   1.2 Inicie a taxa de aprendizagem $\alpha(t)$ com um valor preferencialmente pequeno, como 0.1.

2. **Passo de atualização dos protótipos**: Escolha um padrão de treinamento x_i de Ω aleatoriamente com reposição.

   2.1 Defina o protótipo vencedor w_c tal que

   $c = \arg\min_{m=1,\ldots,M} d_E(x_i, w_m)$

   2.2 Se classe(x_i) = classe(w_c) faça $w_c(t+1) = w_c(t) + \alpha(t)[x_i - w_c(t)]$.

   2.3 Se classe(x_i) $\neq$ classe(w_c) faça $w_c(t+1) = w_c(t) - \alpha(t)[x_i - w_c(t)]$.

3. **Atualização da taxa de aprendizagem**: A taxa de aprendizagem deve decrescer até zero com cada iteração. Repetir do passo 2 até convergir.

A taxa de aprendizagem $\alpha(t)$ decresce, normalmente, monotonicamente com o tempo. Recomenda-se que α comece pequena, por exemplo 0.1. A fórmula exata para a atualização de $\alpha(t)$ não é crucial[5], podendo até decrescer linearmente até zero, desde que o número de passos de treinamento seja suficiente.

3.3 O AQV1 Otimizado

Nesta variação do AQV1 básico (nesse trabalho chamada de $AQV1O$), uma taxa de aprendizagem individual $\alpha_m(t)$ é designada para cada protótipo w_m. Assim, o seguinte processo de aprendizagem é obtido:

1. **Inicialização**

   1.1 Seleciona-se aleatoriamente M padrões distintos do conjunto de treinamento Ω, produzindo o subconjunto inicial de protótipos $\{w_1(t),\ldots,w_M(t)\}$ no passo $t=0$.

   1.2 De $m=1$ até M inicie $\alpha_m(t)$.

2. **Passo de atualização dos protótipos**: Escolha um padrão de treinamento x_i de Ω aleatoriamente com reposição.

   2.1 Defina o protótipo vencedor w_c tal que

   $c = \arg\min_{m=1,\ldots,M} d_E(x_i, w_m)$

   2.2 Se classe(x_i) = classe(w_c) faça $w_c(t+1) = w_c(t) + \alpha_c(t)[x_i - w_c(t)]$.

   2.3 Se classe(x_i) $\neq$ classe(w_c) faça $w_c(t+1) = w_c(t) - \alpha_c(t)[x_i - w_c(t)]$.

   2.4 Atualize

   $$\alpha_c(t+1) = \frac{\alpha_c(t)}{1 + s(t)\alpha_c(t)}$$

   onde $s(t) = +1$ se x_i e w_c pertencem à mesma classe, e s(t)=-1 se x_i e w_c pertencem a classes diferentes

3. **Critério de parada**: Se todos os $\alpha_m(t) = 0 (m = 1,\ldots,M)$ então o algoritmo convergiu. Caso contrário, repetir o passo 2.

É preciso ter uma precaução quanto a essa forma de atualização das taxas de aprendizagem, pois $\alpha_c(t)$ pode também aumentar. Deve-se evitar que $\alpha_c(t)$ torne-se maior do que 1, o que pode ser imposto no próprio algoritmo. Para valores iniciais dos α_m pode-se escolher algo entre 0.3 e 0.5. O $AQV1O$ foi o algoritmo escolhido como base para o método com distâncias adaptativas apresentado nesse trabalho de graduação.

3.4 Outras variações

Existem ainda várias outras modificações do AQV básico. Em geral, essas modificações não alteram a forma como o algoritmo toma as decisões na classificação. São alterações na etapa de aprendizagem. Existem versões que atualizam mais de um protótipo a cada iteração, como o AQV2 e o AQV3, e versões batch, como o AQV1-Batch.

3.4.1 O AQV2

Nessa variação do AQV básico, dois protótipos w_a e w_b que são os vizinhos mais próximos, segundo a distância Euclidiana, de um padrão x_i, retirado aleatoriamente do conjunto de treinamento Ω, são atualizados simultaneamente. Um dos protótipos deve pertencer à classe correta e outro a uma classe errada, respectivamente. Assim, x_i deve cair numa zona de valores chamada de "janela"definida entre w_a e w_b. Sejam d_a e d_b as distâncias Euclidianas entre x_i e w_a e w_b, respectivamente. Então, x_i cairá numa "janela"de largura W se

$$\min\left(\frac{d_a}{d_b}, \frac{d_a}{d_b}\right) > s, \; onde \; s = \frac{1-W}{1+W} \tag{3.2}$$

Recomenda-se uma largura de "janela"W de 0.2 a 0.3.

3.4.2 O AQV3

O AQV2 não leva em conta o que pode acontecer com a localização dos protótipos a longo prazo. Assim, parece necessário introduzir correções que garantam que os protótipos continuem aproximando as distribuições das classes. Daí vem o AQV3:

$$\begin{aligned} w_a(t+1) &= w_a(t) + \alpha(t)[x_i(t) - w_a(t)] \\ w_b(t+1) &= w_b(t) - \alpha(t)[x_i(t) - w_b(t)] \end{aligned} \tag{3.3}$$

onde w_a e w_b são os protótipos mais próximos a x_i, x_i e w_a pertencem à mesma classe e x_i e w_b pertencem a classes diferentes. Além disso, x_i deve estar dentro da "janela"de largura W descrita na equação 3.2. A diferença do $AQV3$ para o $AQV2$ é que o $AQV3$ permite que os dois protótipos escolhidos sejam da mesma classe. Quando x_i, w_a e w_b pertencem à mesma classe, faz-se:

$$w_k(t+1) = w_k(t) + \varepsilon\alpha(t)[x_i(t) - w_k(t)], \; para \; k \in a, b \tag{3.4}$$

Numa série de experimentos, foram achados valores aplicáveis de ε entre 0.1 e 0.5, relacionados a $W = 0.2$ ou 0.3. O valor ótimo de ε demonstra ter uma relação direta com o tamanho W da "janela", menor para janelas mais estreitas. Com o algoritmo descrito, o posicionamento ótimo dos protótipos parece não mudar com a continuação do processo de aprendizagem.

3.4.3 O AQV1-Batch

O AQV1 básico pode ser expresso numa forma batch como segue:

1. Para se iniciar os protótipos, pode-se utilizar um processo não-supervisionado, como o *SOM*, precedente, em que a classificação de $x(t)$ ainda não é levada em consideração.

2. Os $x(t)$ são apresentados de novo, desta vez listando-os, junto com seus atributos de classe sob os protótipos vencedores correspondentes.

3. Determina-se as classes dos protótipos de acordo com a classe mais freqüente nas suas listas de padrões.

4. Para cada protótipo m, tome para seu novo valor do vetor de código a entidade

$$w_m^* = \frac{\sum_{t'} s(t')x(t')}{\sum_{t'} s(t')} \tag{3.5}$$

   onde os $x(t')$ representam os valores dos padrões listados sob o protótipo m e os $s(t')$ indicam se os $x(t')$ pertencem à mesma classe do protótipo m ou não.

5. Repita a partir de 2 algumas vezes.

Para manter a estabilidade, pode ser necessário checar o valor de $\sum_{t'} s(t')$, pois, se for negativo, não se deve atualizar o protótipo. Um fator importante é que, ao contrário do AQV1 básico, o AQV1-Batch permite a mudança dinâmica da classe dos protótipos.

CAPÍTULO 4

Distâncias Adaptativas

A métrica utilizada para a escolha do protótipo "vencedor"nos algoritmos citados acima era a distância Euclidiana. Essa distância tende a capturar áreas circulares ou volumes esféricos. Isso pode tornar os métodos ineficientes em capturar adequadamente as características das classes, em situações em que o conjunto de dados contém partições de diferentes formas e tamanhos ou formas parecidas, mas volumes diferentes. Para resolver esse problema, seria necessário tornar a métrica utilizada, no caso a distância Euclidiana, mais robusta contra esses fatores.

Com esse intuito, foram apresentados por De Carvalho *et al* métodos de clustering (não-supervisionados) utilizando duas formas de distâncias que mudam a cada iteração para representar melhor as regiões de classes: uma distância única (nesse trabalho referida como distância simples) que é a mesma para todos os protótipos e outra distância que não só muda a cada iteração, como também é diferente para cada classe. Essas distâncias são chamadas de distâncias Euclidianas adaptativas.

Características das classes são capturadas pela dispersão dos elementos com relação aos protótipos dos clusters aos quais pertencem. Tais distâncias levam em consideração a estrutura interna dos clusters, sendo, dessa forma, capazes de reconhecer classes de formas e tamanhos diferentes. Além disso, são também mais robustas contra variações de escalas entre as variáveis.

Neste capítulo são apresentadas as três distâncias adaptativas utilizadas nesse trabalho: a Distância Simples (muda a cada iteração, mas é única, ou seja, é a mesma para todos os protótipos), a Distância por Classe (muda a cada iteração e é diferente para cada classe) e a Distância por Protótipo (muda a cada iteração e é diferente para cada protótipo).

4.1 Distância Euclidiana Adaptativa Simples

Ao processar o conjunto de padrões de treinamento, uma função de critério baseada na regra do vizinho mais próximo pode ser definida como:

$$J = \sum_{i=1}^{n} d_{EAS}(x_i(t), w_c(t))\delta_{ci} \tag{4.1}$$

onde d_{EAS} é uma distância adaptativa (nesse caso, a Distância Euclidiana Adaptativa Simples, referida daqui em diante como *DEAS*) entre um padrão $x_i(t)$ e um protótipo $w_c(t)$ na iteração t tal que c é o índice do w_m $(m = 1,\ldots,M)$ mais próximo ao x_i e $\delta_{ci} = 1$ se $x_i(t)$ é da mesma classe que $w_c(t)$, senão $\delta_{ci} = 0$.

O critério acima representa a soma das distâncias entre padrões e os respectivos protótipos vencedores que são da mesma classe. O cálculo da distância *DEAS* é como segue:

$$d_{EAS}(x_i(t), w_c(t)) = \sqrt{\sum_{j=1}^{p} \lambda^j (x_i^j(t) - w_c^j(t))^2} \tag{4.2}$$

A adaptatividade de *DEAS* é expressa pelo vetor de pesos $\lambda = (\lambda^1, \ldots, \lambda^p)$ que representa a informação da dispersão das regiões de classe.

Do método Multiplicador de Lagrange, o vetor de pesos $\lambda = (\lambda^1, \ldots, \lambda^p)$, que minimiza o critério J na Equação 4.1 com $\lambda^j > 0$ e $\prod_{j=1}^{p} \lambda^j = 1$, é atualizado de acordo com a seguinte expressão:

$$\lambda^j = \frac{\left[\prod_{h=1}^{p} \left(\sum_{m=1}^{M} \sum_{x_i \in w_m} (x_i^h - w_m^h)^2\right)\right]^{\frac{1}{p}}}{\sum_{m=1}^{M} \sum_{x_i \in w_m} (x_i^j - w_m^j)^2} \tag{4.3}$$

Onde $h = 1, \ldots, p$ é o conjunto dos índices das variáveis, $m = 1, \ldots, M$ é o conjunto dos índices dos protótipos e os x_i representam os padrões designados a cada protótipo w_m. Assim, para cada variável h, é feito um somatório

$$\Delta^h = \sum_{m=1}^{M} \sum_{x_i \in w_m} (x_i^h - w_m^h)^2 \tag{4.4}$$

da diferença quadrática entre os valores de cada protótipo e os padrões afetados por eles. Faz-se, então, um produtório de todos os Δ^h e depois calcula-se a raiz de ordem p desse produtório (p corresponde ao número de variáveis). Por fim, para se obter o peso λ^j de cada variável j, divide-se o produtório calculado anteriormente pelo Δ^j correspondente à variável. Com isso a Equação 4.3 pode ser reescrita da seguinte forma:

$$\lambda^j = \frac{\left[\prod_{h=1}^{p} \left(\Delta^h\right)\right]^{\frac{1}{p}}}{\Delta^j} \tag{4.5}$$

Seguindo as restrições citadas acima, o produtório de todos os pesos λ^j calculados deve ser igual a 1 e, para toda variável j, $\lambda^j > 0$.

Esse tipo de distância deve capturar as diferenças nas distribuições das variáveis melhor do que a distância Euclidiana padrão, assim, se essas disparidades não ocorrerem (por exemplo com dados normalizados), espera-se um desempenho similar ao da distância Euclidiana padrão. Além disso, como é a mesma para todos os protótipos, não acrescenta robustez contra alta disparidade de formas e volumes de classes e nem das regiões internas das classes representadas pelos protótipos.

4.2 Distância Euclidiana Adaptativa por Classe

Uma abordagem para descrever melhor as formas e volumes variados das classes é manter um vetor de pesos λ_k diferente para cada classe k.

Ao processar o conjunto de padrões de treinamento, uma função de critério baseada na regra do vizinho mais próximo pode ser definida como:

$$J = \sum_{k=1}^{K} \sum_{x_i \in C_k} d_{EAC}(x_i(t), w_c(t)) \delta_{ci} \tag{4.6}$$

onde d_{EAC} é uma distância adaptativa (nesse caso, a Distância Euclidiana Adaptativa por Classe, referida daqui em diante como *DEAC*) entre um padrão $x_i(t)$ e um protótipo $w_c(t)$ na iteração t tal que c é o índice do w_m $(m = 1, \ldots, M)$ mais próximo ao x_i, C_k é o conjunto de protótipos pertencentes à classe k e $\delta_{ci} = 1$ se $x_i(t)$ é da mesma classe que $w_c(t)$, senão $\delta_{ci} = 0$.

O critério acima representa a soma das distâncias entre padrões x_i classificados corretamente e os respectivos protótipos vencedores w_c que são da mesma classe k. O cálculo da distância *DEAC* é como segue:

$$d_{EAC}(x_i(t), w_c(t)) = \sqrt{\sum_{j=1}^{p} \lambda_k^{\,j} (x_i^j(t) - w_c^j(t))^2} \tag{4.7}$$

A adaptatividade de *DEAC* é expressa pelo vetor de pesos $\lambda_k = (\lambda_k^1, \ldots, \lambda_k^p)$ que representa a informação da dispersão das regiões de cada classe.

Do método Multiplicador de Lagrange, o vetor de pesos $\lambda_k = (\lambda_k^1, \ldots, \lambda_k^p)$, que minimiza o critério J na Equação 4.6 com $\lambda_k^{\,j} > 0$ e $\prod_{j=1}^{p} \lambda_k^{\,j} = 1$, é atualizado de acordo com a seguinte expressão:

$$\lambda_k^{\,j} = \frac{\left[\prod_{h=1}^{p} \left(\sum_{x_i \in C_k} (x_i^h - w_c^h)^2\right)\right]^{\frac{1}{p}}}{\sum_{x_i \in C_k} (x_i^j - w_c^j)^2} \tag{4.8}$$

Onde $h = 1, \ldots, H$ é o conjunto dos índices das variáveis, $k = 1, \ldots, K$ é o conjunto dos índices das classes, C_k é o conjunto dos protótipos da classe k e x_i representa um padrão que foi designado corretamente à classe k, ou seja a classe de x_i é igual à do seu protótipo mais próximo, de acordo com a distância *DEAC*, w_c. Assim, para cada classe k, é feito um somatório para cada variável h

$$\Delta_k^h = \sum_{x_i \in C_k} (x_i^h - w_c^h)^2 \tag{4.9}$$

da diferença quadrática entre os valores de cada protótipo da classe k e os padrões afetados corretamente por eles. Faz-se, então, um produtório de todos os Δ_k^h e depois calcula-se a raiz de ordem p desse produtório (p corresponde ao número de variáveis). Por fim, para se obter o peso $\lambda_k^{\,j}$ de cada variável j, para cada classe k, divide-se o produtório calculado anteriormente pelo Δ_k^j correspondente à variável. Com isso a Equação 4.8 pode ser reescrita da seguinte forma:

$$\lambda_k^{\,j} = \frac{\left[\prod_{h=1}^{p} \left(\Delta_k^h\right)\right]^{\frac{1}{p}}}{\Delta_k^j} \tag{4.10}$$

Seguindo as restrições citadas acima, o produtório de todos os pesos $\lambda_k^{\,j}$ calculados para a classe k deve ser igual a 1 e, para toda variável j, $\lambda_k^{\,j} > 0$.

A *DEAC* captura as diferenças entre formas e volumes das classes, assim, se as classes não apresentarem diferença sensível nessas características *DEAC* deve ter desempenho próximo do da distância Euclidiana padrão. Apesar do ganho que a *DEAC* pode trazer, ela não considera as sub-regiões que podem existir nas classes e que são representadas pelos seus protótipos.

4.3 Distância Euclidiana Adaptativa por Protótipo

Um novo tipo de distância adaptativa é introduzido agora, que tem como diferencial o fato de que cada protótipo w_m tem o seu vetor de pesos λ_m.

Ao processar o conjunto de padrões de treinamento, uma função de critério baseada na regra do vizinho mais próximo pode ser definida como:

$$J = \sum_{m=1}^{M} \sum_{x_i \in w_m} d_{EAP}(x_i(t), w_m(t))\delta_{ci} \tag{4.11}$$

onde d_{EAP} é uma distância adaptativa (nesse caso, a Distância Euclidiana Adaptativa por Protótipo, referida daqui em diante como *DEAP*) entre um padrão $x_i(t)$ e um protótipo $w_m(t)$ na iteração t tal que $x_i(t)$ foi afetado pelo protótipo $w_m(t)$ e $\delta_{mi} = 1$ se $x_i(t)$ é da mesma classe que $w_m(t)$, senão $\delta_{mi} = 0$.

O critério acima representa a soma das distâncias entre os M protótipos e os padrões afetados corretamente por eles. O cálculo da distância *DEAP* é como segue:

$$d_{EAP}(x_i(t), w_m(t)) = \sqrt{\sum_{j=1}^{p} \lambda_m^j (x_i^j(t) - w_m^j(t))^2} \tag{4.12}$$

A adaptatividade de *DEAP* é expressa pelo vetor de pesos $\lambda_m = (\lambda_m^1, \ldots, \lambda_m^p)$ que representa a informação da dispersão da sub-regiõe de classe representada pelo protótipo w_m.

Do método Multiplicador de Lagrange, o vetor de pesos $\lambda_m = (\lambda_m^1, \ldots, \lambda_m^p)$, que minimiza o critério J na Equação 4.11 com $\lambda_m^j > 0$ e $\prod_{j=1}^{p} \lambda_m^j = 1$, é atualizado de acordo com a seguinte expressão:

$$\lambda_m^j = \frac{\left[\prod_{h=1}^{p} \left(\sum_{x_i \in w_m} (x_i^h - w_m^h)^2\right)\right]^{\frac{1}{p}}}{\sum_{x_i \in w_m} (x_i^j - w_m^j)^2} \tag{4.13}$$

Onde $h = 1, \ldots, p$ é o conjunto dos índices das variáveis e os x_i representam os padrões designados a cada protótipo w_m. Assim, para cada protótipo w_m, é feito um somatório para cada variável h

$$\Delta_m^h = \sum_{x_i \in w_m} (x_i^h - w_m^h)^2 \tag{4.14}$$

da diferença quadrática entre os valores do protótipo w_m e os padrões x_i afetados corretamente por ele. Faz-se, então, um produtório de todos os Δ_m^h e depois calcula-se a raiz de ordem p desse produtório (p corresponde ao número de variáveis). Por fim, para se obter o peso λ_m^j de

cada variável j, para cada protótipo m, divide-se o produtório calculado anteriormente pelo Δ_m^j correspondente à variável. Com isso a Equação 4.13 pode ser reescrita da seguinte forma:

$$\lambda_m^j = \frac{\left[\prod_{h=1}^{p} \left(\Delta_m^h\right)\right]^{\frac{1}{p}}}{\Delta_m^j} \tag{4.15}$$

Seguindo as restrições citadas acima, o produtório de todos os pesos λ_m^j calculados para o protótipo m deve ser igual a 1 e, para toda variável j, $\lambda_m^j > 0$.

Essa distância deve capturar a dispersão dos regiões representadas por cada um dos protótipos e, com isso, expressar as sub-regiões internas de cada classe. A probabilidade de existirem sub-regiões internas às classes é maior para bases de dados maiores e com outliers. Portanto, para bases de dados pequenas *DEAP* deve ter desempenho próximo ao da distância Euclidiana padrão.

CAPÍTULO 5

O AQV1O com Distâncias Euclidianas Adaptativas

Aplicando os conceitos anteriormente, tem-se um AQV1O que utiliza distâncias adaptativas como métrica para determinação do protótipo mais próximo (chamado daqui em diante de $AQV1O-DEA$). Novos passos devem ser adicionados ao algoritmo original: um passo de inicialização dos pesos e outro de cálculo dos mesmos após cada atualização de protótipo.

5.1 O algoritmo

O $AQV1O-DEA$ é um algoritmo online e isso trouxe um problema, pois as distâncias adaptativas só haviam sido usados antes com algoritmos batch (em particular, algoritmos de clustering). Isso significa que o passo de atualização da distância acontecia após cada vez que todo o conjunto de treinamento era apresentado, pois em algoritmos batch só depois de apresentar todos os elementos de treinamento é que os protótipos são atualizados. A utilização dessas distâncias de forma online acrescenta a necessidade de que a distância seja atualizada a cada apresentação de um elemento de treinamento e atualização de protótipo, pois, se isso não for feito a cada iteração, a distância se tornará incoerente com o posicionamento dos protótipos.

A seguir, a solução para esse problema e o algoritmo $AQV1O-DEA$ proposto são explicados para cada uma das três distâncias adaptativas utilizadas.

5.1.1 AQV1O-DEA com Distância Euclidiana Adaptativa Simples

5.1.1.1 O cálculo dos somatórios usados para calcular os pesos

Para a distância *DEAS* de forma online, o somatório Δ^j das diferenças quadráticas entre os padrões e seus protótipos vencedores para cada variável (definido na Equação 4.4 e utilizado na Equação 4.5) é atualizado a cada vez que um protótipo sofre atualização por ter sido o vencedor de um padrão $x(t)$. Após isso, o vetor de pesos λ^j é recalculado.

Mas essa atualização online dos Δ^j levanta uma nova questão: se o somatório Δ^j é modificado a cada atualização dos protótipos, o fator com que cada protótipo contribuiu previamente para o somatório torna-se desatualizado a cada nova modificação dos valores dos protótipos. A fim de resolver esse problema, a cada atualização de protótipo w_c provocada pela afetação de um padrão x_i, o somatório $\Delta^j(t)$ se torna um somatório ponderado entre $\Delta^j(t-1)$ e a diferença entre x_i^j e w_c^j. Para expressar a atualização constante dos protótipos, a taxa de aprendizagem $\alpha_c(t)$ correspondente a w_c no instante t foi escolhida para compor os pesos do somatório. Assim:

$$\Delta^j(t) = ((1-\alpha_c(t))\Delta^j(t-1)) + (\alpha_c(t)(x_i^j - w_c^j)^2) \tag{5.1}$$

Com isso, o Δ^j deve ser capaz de expressar melhor a atualização constante dos protótipos, produzindo λ^j mais coerentes. Experimentos mostram que, para qualquer valor de $\alpha_c(t)$, os valores dos pesos λ^j mantêm as restrições citadas em 4.1.

5.1.1.2 Treinamento

Os passos do treinamento do algoritmo $AQV1O-DEA$ com distância *DEAS* são os seguintes:

1. **Inicialização Batch**

   1.1 Escolha M padrões distintos do conjunto de dados de treinamento Ω aleatoriamente, produzindo o subconjunto inicial de protótipos $\{w_1(t),\ldots,w_M(t)\}$ no passo t=0.

   1.2 Inicie os pesos das variáveis, fazendo de $j=1$ até p, $\lambda^j = 1$.

   1.3 De $i=1$ até n liste cada x_i de Ω sob seu protótipo vencedor w_c.

   1.4 De $m=1$ até M atualize os valores de cada w_m de acordo com a Equação 3.5.

   1.5 De $j=1$ até p calcule o valor de cada Δ^j e utilize-os para atualizar os λ^j (Equação 4.5)

   1.6 De $m=1$ até M faça $\alpha_m(t) = 0.3$.

2. **Passo de atualização dos protótipos**: escolha um padrão x_i de Ω aleatoriamente com reposição

   2.1 Defina o protótipo vencedor w_c tal que

   $c = \arg\min_{m=1,\ldots,M} d_{EAS}(x_i, w_m)$
   onde *dEAS* é a distância *DEAS* calculada segundo a Equação 4.2

   2.2 Se classe(x_i) = classe(w_c) faça

   2.2.1 $w_c(t+1) = w_c(t) + \alpha_c(t)[x_i - w_c(t)]$.

   2.2.2 De $j=1$ até p atualize os valores dos somatórios Δ^j usando a Equação 5.1.

   2.2.3 De $j=1$ até p atualize os valores dos pesos λ^j usando a Equação 4.5.

   2.3 Se classe(x_i) $\neq$ classe(w_c) faça $w_c(t+1) = w_c(t) - \alpha_c(t)[x_i - w_c(t)]$

   2.4 Atualize

   $$\alpha_c(t+1) = \frac{\alpha_c(t)}{1 + s(t)\alpha_c(t)}$$

   onde $s(t) = +1$ se x_i e w_c pertencem à mesma classe, e s(t)=-1 se x_i e w_c pertencem a classes diferentes

3. **Critérios de parada**: Se para todo $\alpha_m(t) \leq 0.00005$ $(m = 1,\ldots,M)$ ou se, por 10 ou mais iterações, nenhum dos w_m sofreu alteração em seus valores ou se o somatório das distâncias calculadas entre os w_c e os x_i corretamente classificados for menor ou igual a 10^{-5} (indicando um posicionamento dos protótipos que aproxima o posicionamento ótimo) ou ainda se o número de iterações for igual a 100 vezes o número de protótipos, pare o treinamento, senão vá para 2.

5.1.2 AQV1O-DEA com Distância Euclidiana Adaptativa por Classe

5.1.2.1 O cálculo dos somatórios usados para calcular os pesos

Para a distância *DEAC* de forma online, o somatório Δ_k^j das diferenças quadráticas entre os valores de cada protótipo da classe k e os padrões afetados corretamente por eles (definido na Equação 4.9 e utilizado na Equação 4.10) é atualizado a cada vez que um protótipo da classe k sofre atualização por ter sido o vencedor de um padrão $x(t)$. Após isso, o vetor de pesos λ_k^j para a classe k é recalculado.

Mas essa atualização online dos Δ_k^j levanta uma nova questão: se o somatório Δ_k^j é modificado a cada atualização dos protótipos da classe k, o fator com que cada protótipo da classe k contribuiu previamente para o somatório torna-se desatualizado a cada nova modificação dos valores dos protótipos. A fim de resolver esse problema, a cada atualização de protótipo w_c (que pertence à classe k) provocada pela afetação de um padrão x_i, o somatório $\Delta_k^j(t)$ se torna um somatório ponderado entre $\Delta_k^j(t-1)$ e a diferença entre x_i^j e w_c^j. Para expressar a atualização constante dos protótipos, a taxa de aprendizagem $\alpha_c(t)$ correspondente a w_c no instante t foi escolhida para compor os pesos do somatório. Assim:

$$\Delta_k^j(t) = ((1-\alpha_c(t))\Delta_k^j(t-1)) + (\alpha_c(t)(x_i^j - w_c^j)^2) \tag{5.2}$$

Com isso, o Δ_k^j deve ser capaz de expressar melhor a atualização constante dos protótipos, produzindo λ_k^j mais coerentes. Experimentos mostram que, para qualquer valor de $\alpha_c(t)$, os valores dos pesos λ_k^j mantêm as restrições citadas em 4.2.

5.1.2.2 Treinamento

Os passos do treinamento do algoritmo $AQV1O-DEA$ com distância *DEAC* são os seguintes:

1. **Inicialização Batch**

   1.1 Escolha M padrões distintos do conjunto de dados de treinamento Ω aleatoriamente, produzindo o subconjunto inicial de protótipos $\{w_1(t),\ldots,w_M(t)\}$ no passo t=0.

   1.2 Inicie os pesos das variáveis para cada classe, fazendo de $k = 1$ até K (K é o número de classes) de $j = 1$ até p, $\lambda_k^j = 1$.

   1.3 De $i = 1$ até n liste cada x_i de Ω sob seu protótipo vencedor w_c.

   1.4 De $m = 1$ até M atualize os valores de cada w_m de acordo com a Equação 3.5.

1.5 De $k = 1$ até K de $j = 1$ até p calcule o valor de cada Δ_k^j e utilize-os para atualizar os λ_k^j (Equação 4.10)

1.6 De $m = 1$ até M faça $\alpha_m(t) = 0.3$.

2. **Passo de atualização dos protótipos**: escolha um padrão x_i de Ω aleatoriamente com reposição

2.1 Defina o protótipo vencedor w_c tal que

$c = \arg\min_{m=1,\ldots,M} d_{EAC}(x_i, w_m)$
onde *dEAC* é a distância *DEAC* calculada segundo a Equação 4.7

2.2 Se classe(x_i) = classe(w_c) faça

2.2.1 $w_c(t+1) = w_c(t) + \alpha_c(t)[x_i - w_c(t)]$.

2.2.2 De $j = 1$ até p atualize os valores dos somatórios Δ_k^j (onde k é o índice da classe do protótipo w_c) usando a Equação 5.2.

2.2.3 De $j = 1$ até p atualize os valores dos pesos λ_k^j (onde k é o índice da classe do protótipo w_c) usando a Equação 4.10.

2.3 Se classe(x_i) $\neq$ classe(w_c) faça $w_c(t+1) = w_c(t) - \alpha_c(t)[x_i - w_c(t)]$

2.4 Atualize

$$\alpha_c(t+1) = \frac{\alpha_c(t)}{1 + s(t)\alpha_c(t)}$$

onde $s(t) = +1$ se x_i e w_c pertencem à mesma classe, e s(t)=-1 se x_i e w_c pertencem a classes diferentes

3. **Critérios de parada**: Se para todo $\alpha_m(t) \leq 0.00005$ $(m = 1, \ldots, M)$ ou se, por 10 ou mais iterações, nenhum dos w_m sofreu alteração em seus valores ou se o somatório das distâncias calculadas entre os w_c e os x_i corretamente classificados for menor ou igual a 10^{-5} (indicando um posicionamento dos protótipos que aproxima o posicionamento ótimo) ou ainda se o número de iterações for igual a 100 vezes o número de protótipos, pare o treinamento, senão vá para 2.

5.1.3 AQV1O-DEA com Distância Euclidiana Adaptativa por Protótipo

5.1.3.1 O cálculo dos somatórios usados para calcular os pesos

Para a distância *DEAP* de forma online, o somatório Δ_m^j das diferenças quadráticas entre os valores do protótipo w_m e os padrões x_i afetados corretamente por ele (definido na Equação 4.14 e utilizado na Equação 4.15) é atualizado a cada vez que um protótipo w_m sofre atualização por ter sido o vencedor de um padrão $x(t)$. Após isso, o vetor de pesos λ_m^j para o protótipo w_m é recalculado.

Mas essa atualização online dos Δ_m^j levanta uma nova questão: se o somatório Δ_m^j é modificado a cada atualização do protótipo w_m, o fator com que w_m contribuiu previamente para o seu

somatório torna-se desatualizado a cada nova modificação. A fim de resolver esse problema, a cada atualização de protótipo w_c provocada pela afetação de um padrão x_i, o somatório $\Delta_c^j(t)$ se torna um somatório ponderado entre $\Delta_c^j(t-1)$ e a diferença entre x_i^j e w_c^j. Para expressar a atualização constante dos protótipos, a taxa de aprendizagem $\alpha_c(t)$ correspondente a w_c no instante t foi escolhida para compor os pesos do somatório. Assim:

$$\Delta_c^j(t) = ((1-\alpha_c(t))\Delta_c^j(t-1)) + (\alpha_c(t)(x_i^j - w_c^j)^2) \tag{5.3}$$

Com isso, o Δ_c^j deve ser capaz de expressar melhor a atualização constante dos protótipos, produzindo λ_c^j mais coerentes. Experimentos mostram que, para qualquer valor de $\alpha_c(t)$, os valores dos pesos λ_c^j mantêm as restrições citadas em 4.3.

5.1.3.2 Treinamento

Os passos do treinamento do algoritmo $AQV1O-DEA$ com distância $DEAP$ são os seguintes:

1. **Inicialização Batch**

   1.1 Escolha M padrões distintos do conjunto de dados de treinamento Ω aleatoriamente, produzindo o subconjunto inicial de protótipos $\{w_1(t), \ldots, w_M(t)\}$ no passo t=0.

   1.2 Inicie os pesos das variáveis para cada protótipo, fazendo de $m = 1$ até M de $j = 1$ até p, $\lambda_m^j = 1$.

   1.3 De $i = 1$ até n liste cada x_i de Ω sob seu protótipo vencedor w_c.

   1.4 De $m = 1$ até M atualize os valores de cada w_m de acordo com a Equação 3.5.

   1.5 De $m = 1$ até M de $j = 1$ até p calcule o valor de cada Δ_m^j e utilize-os para atualizar os λ_m^j (Equação 4.15)

   1.6 De $m = 1$ até M faça $\alpha_m(t) = 0.3$.

2. **Passo de atualização dos protótipos**: escolha um padrão x_i de Ω aleatoriamente com reposição

   2.1 Defina o protótipo vencedor w_c tal que

   $c = \arg\min_{m=1,\ldots,M} d_{EAP}(x_i, w_m)$
   onde $dEAP$ é a distância $DEAP$ calculada segundo a Equação 4.12

   2.2 Se classe(x_i) = classe(w_c) faça

   2.2.1 $w_c(t+1) = w_c(t) + \alpha_c(t)[x_i - w_c(t)]$.

   2.2.2 De $j = 1$ até p atualize os valores dos somatórios Δ_c^j usando a Equação 5.3.

   2.2.3 De $j = 1$ até p atualize os valores dos pesos λ_c^j usando a Equação 4.15.

   2.3 Se classe(x_i) $\neq$ classe(w_c) faça $w_c(t+1) = w_c(t) - \alpha_c(t)[x_i - w_c(t)]$

2.4 Atualize

$$\alpha_c(t+1) = \frac{\alpha_c(t)}{1+s(t)\alpha_c(t)}$$

onde $s(t) = +1$ se x_i e w_c pertencem à mesma classe, e s(t)=-1 se x_i e w_c pertencem a classes diferentes

3. **Critérios de parada**: Se para todo $\alpha_m(t) \leq 0.00005$ $(m = 1,\ldots,M)$ ou se, por 10 ou mais iterações, nenhum dos w_m sofreu alteração em seus valores ou se o somatório das distâncias calculadas entre os w_c e os x_i corretamente classificados for menor ou igual a 10^{-5} (indicando um posicionamento dos protótipos que aproxima o posicionamento ótimo) ou ainda se o número de iterações for igual a 100 vezes o número de protótipos, pare o treinamento, senão vá para 2.

5.2 O problema da inicialização dos protótipos

Um problema ao desenvolver um método de construção de conjunto de protótipos incremental é como gerar um conjunto de protótipos inicial. Para resolver esse problema, uma simulação Monte Carlo com um número fixo de inicializações do conjunto inicial de protótipos é utilizada. Para cada inicialização, um conjunto de protótipos atualizados é obtido de acordo com a convergência do algoritmo e a média do conjunto de protótipos atualizados entre essas inicializações é obtida. O objetivo é obter os melhores valores para o conjunto de protótipos atualizados e melhorar a performance do classificador.

5.3 Avaliação de desempenho com conjuntos de dados reais

Para uma primeira análise da performance do algoritmo $AQV1O - DEA$, ele foi testado utilizando as três distâncias Euclidianas adaptativas citadas ($DEAS$, $DEAC$ e $DEAP$) e comparado com o algoritmo $AQV1O$ com distância Euclidiana padrão (chamada de DE nas tabelas de resultados deste Capítulo) utilizando os dados de sete conjuntos de dados extraídos do repositório UCI machine learning[1]. O método foi testado duas vezes para cada conjunto de dados, uma vez com os dados originais e outra com os dados normalizados no intervalo de 0 a 1. Foi utilizada simulação Monte Carlo de 5 replicações, em que em cada replicação foi feita uma validação cruzada de 10 partições. Assim, foram obtidos cinqüenta valores de taxa de erro de classificação e os valores apresentados são a média entre eles. Além disso, para cada partição da validação cruzada, foi feita uma estimação dos protótipos, como citado em 5.2, em que foram usadas dez inicializações. A taxa de aprendizagem inicial de cada protótipo foi de 0.3. Os números de protótipos para cada experimento foram escolhidos após testes. Não foram usados métodos de estimativa do número de protótipos.

[1] http://www.ics.uci.edu/mlearn/MLRepository.html

5.3.1 Wisconsin Breast Cancer

5.3.1.1 Dados da base

Endereço: http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original)
Número de instâncias: 699
Atributos:

1. Sample code number: id number
2. Clump Thickness: 1 - 10
3. Uniformity of Cell Size: 1 - 10
4. Uniformity of Cell Shape: 1 - 10
5. Marginal Adhesion: 1 - 10
6. Single Epithelial Cell Size: 1 - 10
7. Bare Nuclei: 1 - 10
8. Bland Chromatin: 1 - 10
9. Normal Nucleoli: 1 - 10
10. Mitoses: 1 - 10
11. Classe: (2 para benigno e 4 para maligno)

Distribuição das classes: 458 instâncias da classe benigno e 241 instâncias da classe maligno
Missing values e remoção de atributos: A base tem 16 instâncias com missing values que foram retiradas paras os experimentos, ou seja, foram utilizadas 683 instâncias. O atributo 1, que é apenas um número de identificação de cada instância não foi utilizado.

5.3.1.2 Resultados

Foram utilizados 44 protótipos para classe benigno e 24 protótipos para classe maligno.

Tabela 5.1 Dados Originais (Wisconsin Breast Cancer)

	DE	DEAS	DEAC	DEAP
Erro Médio	4.915779160%	3.805378478%	4.306682699%	4.129734956%
Desvio Padrão	3.128501574%	1.861413747%	2.467887058%	2.836033672%

Tabela 5.2 Dados Normalizados (Wisconsin Breast Cancer)

	DE	DEAS	DEAC	DEAP
Erro Médio	5.003161940%	3.982650685%	3.809653777%	13.081020728%
Desvio Padrão	2.945300019%	2.352905757%	2.323495400%	7.564690191%

5.3.1.3 Avaliação

É um problema fácil, com taxas de erro baixas. Não há diferença significativa entre os desempenhos do método com distâncias adaptativas e do método com distância padrão, exceto para a distância *DEAP* que teve o desempenho bastante prejudicado com a normalização.

5.3.2 Iris

5.3.2.1 Dados da base

Endereço: http://archive.ics.uci.edu/ml/datasets/Iris
Número de instâncias: 150
Atributos:

1. Comprimento da sépala em cm
2. Largura da sépala em cm
3. Comprimento da pétala em cm
4. Largura da pétala em cm
5. Classe: (0 para Iris Setosa, 1 para Iris Versicolour e 2 para Iris Virginica)

Distribuição das classes: 50 instâncias para cada classe
Missing values e remoção de atributos: A base não tem missing values e todos os atributos foram utilizados.

5.3.2.2 Resultados

Foram utilizados 5 protótipos para cada classe.

Tabela 5.3 Dados Originais (Iris)

	DE	DEAS	DEAC	DEAP
Erro Médio	5.733333333%	4.133333333%	4.000000000%	3.466666667%
Desvio Padrão	6.172133998%	4.835619914%	4.856209061%	4.900830602%

Tabela 5.4 Dados Normalizados (Iris)

	DE	DEAS	DEAC	DEAP
Erro Médio	5.866666667%	4.133333333%	3.066666667%	4.000000000%
Desvio Padrão	6.119001297%	5.853851185%	4.895275153%	4.856209061%

5.3.2.3 Avaliação

É um problema fácil, com taxas de erro baixas. Não há diferença significativa entre os desempenhos do método com distâncias adaptativas e do método com distância padrão e nem entre a utilização dos dados originais e dos dados normalizados.

5.3.3 Pima Indians Diabetes

5.3.3.1 Dados da base

Endereço: http://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes
Número de instâncias: 768
Atributos:

1. Número de gravidezes
2. Concentração de glucose no plasma
3. Pressão diastólica (mm Hg)
4. Largura da dobra de pele do tríceps (mm)
5. Insulina (mu U/ml)
6. Índice de massa corporal (kg/m^2)
7. Função de pedigree para diabete
8. Idade em anos
9. Classe: (0 para teste negativo para diabetes e 1 para teste positivo para diabetes)

Distribuição das classes: 500 instâncias da classe teste negativo para diabetes e 268 instâncias da classe teste positivo para diabetes
Missing values e remoção de atributos: A base não tem missing values e todos os atributos foram utilizados.

5.3.3.2 Resultados

Foram utilizados 50 protótipos para classe 0 e 25 protótipos para classe 1.

Tabela 5.5 Dados Originais (Pima Indians Diabetes)

	DE	DEAS	DEAC	DEAP
Erro Médio	35.265208476%	31.042036910%	30.984278879%	30.311688312%
Desvio Padrão	5.592204645%	5.349565333%	4.882172148%	5.031463534%

Tabela 5.6 Dados Normalizados (Pima Indians Diabetes)

	DE	DEAS	DEAC	DEAP
Erro Médio	33.280246070%	30.055707450%	30.050239234%	30.289815448%
Desvio Padrão	4.973693834%	5.118895591%	4.597292975%	4.784526032%

5.3.3.3 Avaliação

Testes de hipóteses foram feitos para comprovar o melhor desempenho do algoritmo $AQV1O-DEA$ em relação ao algoritmo $AQV1O$. O erro médio apresentado nas tabelas é resultado de uma média entre cinqüenta taxas de erro, ou seja $n_1 = n_2 = 50$ e, com isso, o valor da tabela t de Student aproxima o valor da tabela Z. Foi utilizado um nível de significância de 5% (valor crítico -1.64). A hipótese nula H_0 é de que $x_1 - x_2 \geqq 0$ em que x_2 equivale ao erro médio do $AQV1O$ utilizando os dados originais nos 3 primeiros testes e ao erro médio do $AQV1O$ utilizando os dados normalizados nos 3 últimos testes e x_1 corresponde ao erro médio do algoritmo com a distância Euclidiana adpatativa correspondente ao teste. E a hipótese alternativa H_a é de que $x_1 - x_2 < 0$.

1. **Resultados para os dados originais:**

Teste 1 (*DEAS*): $t \approxeq -3.8587, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância *DEAS* tem desempenho melhor do que o $AQV1O$.

Teste 2 (*DEAC*): $t \approxeq -4.0777, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância *DEAC* tem desempenho melhor do que o $AQV1O$.

Teste 3 (*DEAP*): $t \approxeq -4.6562, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância *DEAP* tem desempenho melhor do que o $AQV1O$.

2. **Resultados para os dados normalizados:**

Teste 1 (*DEAS*): $t \approxeq -3.1946, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância *DEAS* tem desempenho melhor do que o $AQV1O$.

Teste 2 (*DEAC*): $t \approxeq -3.3722, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância *DEAC* tem desempenho melhor do que o $AQV1O$.

Teste 3 (*DEAP*): $t \approxeq -3.0639, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância *DEAP* tem desempenho melhor do que o $AQV1O$.

De acordo com os resultados dos testes para o conjunto *PimaIndiansDiabetes*, o $AQV1O-DEA$ teve desempenho melhor do que o $AQV1O$ para as três distâncias apresentadas, sendo que as distâncias *DEAC* e *DEAP* tiveram melhores desempenhos.

5.3.4 Glass Identification

5.3.4.1 Dados da base

Endereço: http://archive.ics.uci.edu/ml/datasets/Glass+Identification
Número de instâncias: 214
Atributos:

1. Número de identificação da instância: 1 to 214
2. RI: índice de refração
3. Na: Sódio (unidade de medida: percentagem no peso do óxido correspondente, a mesma dos atributos 4-10)
4. Mg: Magnésio
5. Al: Alumínio
6. Si: Silício
7. K: Potássio
8. Ca: Cálcio
9. Ba: Bário
10. Fe: Ferro
11. Classe: (São 7 classes, que representam 7 tipos de vidros diferentes)

Distribuição das classes: 70 instâncias da classe 1, 76 instâncias da classe 2, 17 instâncias da classe 3, 0 instâncias da classe 4, 13 instâncias da classe 5, 9 instâncias da classe 6 e 29 instâncias da classe 7
Missing values e remoção de atributos: A base não tem missing values, no entanto a classe 4 não tem instâncias e, portanto foi desconsiderada. O atributo 1, que é apenas um número de identificação de cada instância não foi utilizado.

5.3.4.2 Resultados

Foram utilizados 15 protótipos para classe 1, 15 protótipos para classe 2, 6 protótipos para classe 3, 3 protótipos para classe 5, 3 protótipos para classe 6 e 15 protótipos para classe 7.

Tabela 5.7 Dados Originais (Glass Identification)

	DE	DEAS	DEAC	DEAP
Erro Médio	43.342750486%	36.904918753%	37.348309178%	36.549865111%
Desvio Padrão	10.943766111%	9.815947985%	10.542030465%	10.034249921%

Tabela 5.8 Dados Normalizados (Glass Identification)

	DE	DEAS	DEAC	DEAP
Erro Médio	44.831313131%	41.776359872%	40.736991028%	40.566183575%
Desvio Padrão	9.656235993%	9.947452048%	10.482755899%	11.291172592%

5.3.4.3 Avaliação

Testes de hipóteses foram feitos para comprovar o melhor desempenho do algoritmo $AQV1O-DEA$ em relação ao algoritmo $AQV1O$. O erro médio apresentado nas tabelas é resultado de uma média entre cinqüenta taxas de erro, ou seja $n_1 = n_2 = 50$ e, com isso, o valor da tabela t de Student aproxima o valor da tabela Z. Foi utilizado um nível de significância de 5% (valor crítico -1.64). A hipótese nula H_0 é de que $x_1 - x_2 \geqq 0$ em que x_2 equivale ao erro médio do $AQV1O$ utilizando os dados originais nos 3 primeiros testes e ao erro médio do $AQV1O$ utilizando os dados normalizados nos 3 últimos testes e x_1 corresponde ao erro médio do algoritmo com a distância Euclidiana adpatativa correspondente ao teste. E a hipótese alternativa H_a é de que $x_1 - x_2 < 0$.

1. **Resultados para os dados originais:**

Teste 1 (*DEAS*): $t \approxeq -3.0965, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância *DEAS* tem desempenho melhor do que o $AQV1O$.

Teste 2 (*DEAC*): $t \approxeq -2.7895, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância *DEAC* tem desempenho melhor do que o $AQV1O$.

Teste 3 (*DEAP*): $t \approxeq -3.2351, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância *DEAP* tem desempenho melhor do que o $AQV1O$.

2. **Resultados para os dados normalizados:**

Teste 1 (*DEAS*): $t \approxeq -1.5582$, $t < -1.64$. Assim, existe evidência de que o $AQV1O-DEA$ com distância *DEAS* tem desempenho equivalente ao do $AQV1O$.

Teste 2 (*DEAC*): $t \approxeq -2.0313$, $t < -1.64$. Assim, existe evidência de que o $AQV1O-DEA$ com distância *DEAC* tem desempenho melhor do que o $AQV1O$.

Teste 3 (*DEAP*): $t \approxeq -2.0299$, $t < -1.64$. Assim, existe evidência de que o $AQV1O-DEA$ com distância *DEAP* tem desempenho melhor do que o $AQV1O$.

De acordo com os resultados dos testes para a base *Glass Identification*, o $AQV1O-DEA$ teve desempenho em geral melhor do que o $AQV1O$, sendo equivalente apenas quando utilizou-se a distância *DEAS* com os dados normalizados, sendo que a distância *DEAP* teve o melhor desempenho.

5.3.5 Heart Disease

5.3.5.1 Dados da base

Endereço: http://archive.ics.uci.edu/ml/datasets/Heart+Disease
Número de instâncias: 303
Atributos:

1. Idade
2. Sexo
3. Cp
4. Trestbps
5. Chol
6. Fbs
7. Restecg
8. Thalach
9. Exang
10. Oldpeak
11. Slope
12. Ca
13. Thal
14. Classe: (0 para saudável e 1 para doente)

Distribuição das classes: 150 instâncias da classe 0 e 120 instâncias da classe 1
Missing values e remoção de atributos: A base tem 33 instâncias com missing values que foram retiradas paras os experimentos, ou seja, foram utilizadas 270 instâncias. A base tem, na verdade, 75 atributos, mas apenas 14 deles são utilizados. Além disso, ela tem cinco classes: classe 0 para indivíduos saudáveis e 1,2,3 e 4 indicam níveis de doença cardíaca. Mas, nestes experimentos, o problema foi reduzido a 2 classes: saudável e doente.

5.3.5.2 Resultados

Foram utilizados 18 protótipos para classe 0 e 9 protótipos para classe 1.

Tabela 5.9 Dados Originais (Heart Disease)

	DE	DEAS	DEAC	DEAP
Erro Médio	40.592592593%	35.629629630%	34.444444444%	33.703703704%
Desvio Padrão	8.724866120%	8.722619801%	9.820482113%	9.705786417%

Tabela 5.10 Dados Normalizados (Heart Disease)

	DE	DEAS	DEAC	DEAP
Erro Médio	29.555555555%	25.777777778%	24.666666667%	22.962962963%
Desvio Padrão	8.741855240%	9.070949838%	9.687309086%	7.812075693%

5.3.5.3 Avaliação

Testes de hipóteses foram feitos para comprovar o melhor desempenho do algoritmo $AQV1O-DEA$ em relação ao algoritmo $AQV1O$. O erro médio apresentado nas tabelas é resultado de uma média entre cinqüenta taxas de erro, ou seja $n_1 = n_2 = 50$ e, com isso, o valor da tabela t de Student aproxima o valor da tabela Z. Foi utilizado um nível de significância de 5% (valor crítico -1.64). A hipótese nula H_0 é de que $x_1 - x_2 \geqq 0$ em que x_2 equivale ao erro médio do $AQV1O$ utilizando os dados originais nos 3 primeiros testes e ao erro médio do $AQV1O$ utilizando os dados normalizados nos 3 últimos testes e x_1 corresponde ao erro médio do algoritmo com a distância Euclidiana adpatativa correspondente ao teste. E a hipótese alternativa H_a é de que $x_1 - x_2 < 0$.

1. **Resultados para os dados originais:**

Teste 1 (*DEAS*): $t \approxeq -2.8445, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância *DEAS* tem desempenho melhor do que o $AQV1O$.

Teste 2 (*DEAC*): $t \approxeq -3.3094, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância *DEAC* tem desempenho melhor do que o $AQV1O$.

Teste 3 (*DEAP*): $t \approxeq -3.7324, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância *DEAP* tem desempenho melhor do que o $AQV1O$.

2. **Resultados para os dados normalizados:**

Teste 1 (*DEAS*): $t \approxeq -2.1205, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância *DEAS* tem desempenho melhor do que o $AQV1O$.

Teste 2 (*DEAC*): $t \approxeq -2.6493, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância *DEAC* tem desempenho melhor do que o $AQV1O$.

Teste 3 (*DEAP*): $t \approxeq -3.9762, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância *DEAP* tem desempenho melhor do que o $AQV1O$.

De acordo com os resultados dos testes para o conjunto *HeartDisease*, o $AQV1O-DEA$ teve desempenho melhor do que o $AQV1O$. A distância *DEAP* teve um melhor desempenho.

5.3.6 Image Segmentation

5.3.6.1 Dados da base

Endereço: http://archive.ics.uci.edu/ml/datasets/Image+Segmentation
Número de instâncias: 2310
Atributos:

1. Region-centroid-col
2. Region-centroid-row
3. Region-pixel-count
4. Short-line-density-5
5. Short-line-density-2
6. Vedge-mean
7. Vegde-sd
8. Hedge-mean
9. Hedge-sd
10. Intensity-mean
11. Rawred-mean
12. Rawblue-mean
13. Rawgreen-mean
14. Exred-mean
15. Exblue-mean
16. Exgreen-mean
17. Value-mean
18. Saturation-mean
19. Hue-mean
20. Classe: (0 para brickface, 1 para sky, 2 para grass, 3 para path, 4 para window, 5 para cement e 6 para foliage)

Distribuição das classes: 330 instâncias para cada classe
Missing values e remoção de atributos: A base não tem missing values e todos os atributos foram utilizados.

5.3.6.2 Resultados

Foram utilizados 30 protótipos para cada classe.

Tabela 5.11 Dados Originais (Image Segmentation)

	DE	DEAS	DEAC	DEAP
Erro Médio	15.497835498%	13.281385281%	12.658008658%	12.692640693%
Desvio Padrão	2.552846931%	2.038828890%	2.476127459%	2.391074709%

Tabela 5.12 Dados Normalizados (Image Segmentation)

	DE	DEAS	DEAC	DEAP
Erro Médio	18.137564990%	13.859369860%	13.198644658%	14.357548944%
Desvio Padrão	3.563625876%	3.216727556%	3.239607548%	2.654089057%

5.3.6.3 Avaliação

Testes de hipóteses foram feitos para comprovar o melhor desempenho do algoritmo $AQV1O-DEA$ em relação ao algoritmo $AQV1O$. O erro médio apresentado nas tabelas é resultado de uma média entre cinqüenta taxas de erro, ou seja $n_1 = n_2 = 50$ e, com isso, o valor da tabela t de Student aproxima o valor da tabela Z. Foi utilizado um nível de significância de 5% (valor crítico -1.64). A hipótese nula H_0 é de que $x_1 - x_2 \geqq 0$ em que x_2 equivale ao erro médio do $AQV1O$ utilizando os dados originais nos 3 primeiros testes e ao erro médio do $AQV1O$ utilizando os dados normalizados nos 3 últimos testes e x_1 corresponde ao erro médio do algoritmo com a distância Euclidiana adpatativa correspondente ao teste. E a hipótese alternativa H_a é de que $x_1 - x_2 < 0$.

1. **Resultados para os dados originais:**

Teste 1 (*DEAS*): $t \approxeq -4.7971, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância *DEAS* tem desempenho melhor do que o $AQV1O$.

Teste 2 (*DEAC*): $t \approxeq -5.6463, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância *DEAC* tem desempenho melhor do que o $AQV1O$.

Teste 3 (*DEAP*): $t \approxeq -5.6710, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância *DEAP* tem desempenho melhor do que o $AQV1O$.

2. **Resultados para os dados normalizados:**

Teste 1 (*DEAS*): $t \approxeq -6.3014, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância *DEAS* tem desempenho melhor do que o $AQV1O$.

Teste 2 (*DEAC*): $t \approxeq -7.2514, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância *DEAC* tem desempenho melhor do que o $AQV1O$.

Teste 3 (*DEAP*): $t \approxeq -6.0154, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O - DEA$ com distância *DEAP* tem desempenho melhor do que o $AQV1O$.

De acordo com os resultados dos testes para o conjunto *ImageSegmentation*, o $AQV1O - DEA$ teve desempenho bem melhor do que o $AQV1O$. A distância *DEAC* apresentou o melhor desempenho.

5.3.7 Wine

5.3.7.1 Dados da base

Endereço: http://archive.ics.uci.edu/ml/datasets/Wine
Número de instâncias: 178
Atributos:

1. Álcool
2. Ácido málico
3. Cinzas
4. Alcalinidade das cinzas
5. Magnésio
6. Fenóis totais
7. Flavonóides
8. Fenóis não-flavonóides
9. Proantocianinas
10. Intensidade da cor
11. Matiz
12. OD280/OD315 dos vinhos diluídos
13. Prolina
14. Classe: (1 a 3)

Distribuição das classes: 59 instâncias para classe 1, 71 instâncias para classe 2 e 48 instâncias para classe 3
Missing values e remoção de atributos: A base não tem missing values e todos os atributos foram utilizados.

Tabela 5.13 Dados Originais (Wine)

	DE	DEAS	DEAC	DEAP
Erro Médio	35.238476092%	30.161807705%	29.137168903%	29.261222910%
Desvio Padrão	10.236972463%	10.247026343%	10.425752055%	9.103687077%

Tabela 5.14 Dados Normalizados (Wine)

	DE	DEAS	DEAC	DEAP
Erro Médio	12.035345717%	5.102812177%	4.512297902%	4.403336773%
Desvio Padrão	6.259272209%	5.544260197%	5.051764733%	5.000920825%

5.3.7.2 Resultados

Foram utilizados 5 protótipos para cada classe.

5.3.7.3 Avaliação

Testes de hipóteses foram feitos para comprovar o melhor desempenho do algoritmo $AQV1O-DEA$ em relação ao algoritmo $AQV1O$. O erro médio apresentado nas tabelas é resultado de uma média entre cinqüenta taxas de erro, ou seja $n_1 = n_2 = 50$ e, com isso, o valor da tabela t de Student aproxima o valor da tabela Z. Foi utilizado um nível de significância de 5% (valor crítico -1.64). A hipótese nula H_0 é de que $x_1 - x_2 \geqq 0$ em que x_2 equivale ao erro médio do $AQV1O$ utilizando os dados originais nos 3 primeiros testes e ao erro médio do $AQV1O$ utilizando os dados normalizados nos 3 últimos testes e x_1 corresponde ao erro médio do algoritmo com a distância Euclidiana adpatativa correspondente ao teste. E a hipótese alternativa H_a é de que $x_1 - x_2 < 0$.

1. **Resultados para os dados originais:**

Teste 1 ($DEAS$): $t \approxeq -2.4784, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância $DEAS$ tem desempenho melhor do que o $AQV1O$.

Teste 2 ($DEAC$): $t \approxeq -2.9527, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância $DEAC$ tem desempenho melhor do que o $AQV1O$.

Teste 3 ($DEAP$): $t \approxeq -3.0852, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância $DEAP$ tem desempenho melhor do que o $AQV1O$.

2. **Resultados para os dados normalizados:**

Teste 1 ($DEAS$): $t \approxeq -5.8625, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância $DEAS$ tem desempenho melhor do que o $AQV1O$.

Teste 2 ($DEAC$): $t \approxeq -6.6135, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância $DEAC$ tem desempenho melhor do que o $AQV1O$.

Teste 3 ($DEAP$): $t \approxeq -6.7359, t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância $DEAP$ tem desempenho melhor do que o $AQV1O$.

De acordo com os resultados dos testes para o conjunto *Wine*, o $AQV1O-DEA$ teve desempenho melhor do que o $AQV1O$ para as três distâncias apresentadas. Chama atenção a diferença entre os resultados com os dados originais e com os dados normalizados. A distância *DEAP* obteve o melhor desempenho.

5.3.8 Avaliação geral para os conjuntos de dados reais

Como foi mostrado nas seções anteriores pelos resultados dos experimentos e testes de hipóteses, o algoritmo $AQV1O-DEA$ apresenta desempenho melhor do que o algoritmo $AQV1O$ que utiliza distância Euclidiana padrão, como era esperado. Além disso, não houve muita diferença nos resultados com dados originais e com dados normalizados, exceto para a base *Wine*, que apresentou uma disparidade muito grande desses resultados. O desempenho geral pode ser observado nas tabelas abaixo.

Tabela 5.15 Resultados dos algoritmos para os dados originais dos conjuntos de dados reais

Conjunto	DE	DEAS	DEAC	DEAP
Cancer	4.915779160%	3.805378478%	4.306682699%	4.129734956%
Iris	5.733333333%	4.133333333%	4.000000000%	3.466666667%
Pima	35.265208476%	31.042036910%	30.984278879%	30.311688312%
Glass	43.342750486%	36.904918753%	37.348309178%	36.549865111%
Heart	40.592592593%	35.629629630%	34.444444444%	33.703703704%
Segmentation	15.497835498%	13.281385281%	12.658008658%	12.692640693%
Wine	35.238476092%	30.161807705%	29.137168903%	29.261222910%

Tabela 5.16 Resultados dos algoritmos para os dados normalizados dos conjuntos de dados reais

Conjunto	DE	DEAS	DEAC	DEAP
Cancer	5.003161940%	3.982650685%	3.809653777%	13.081020728%
Iris	5.866666667%	4.133333333%	3.066666667%	4.000000000%
Pima	33.280246070%	30.055707450%	30.050239234%	30.289815448%
Glass	44.831313131%	41.776359872%	40.736991028%	40.566183575%
Heart	29.555555555%	25.777777778%	24.666666667%	22.962962963%
Segmentation	18.137564990%	13.859369860%	13.198644658%	14.357548944%
Wine	12.035345717%	5.102812177%	4.512297902%	4.403336773%

5.4 Avaliação de performance com conjuntos de dados sintéticos

Experimentos com dois conjuntos artificiais de dados quantitativos em $\Re^2$ mostrando alguma superposição de classes e uma análise de performance correspondente do $AQV1O-DEA$ são consideradas nesta seção. A idéia é conseguir uma avaliação do algoritmo $AQV1O-DEA$ em

comparação com o algoritmo $AQV1O$ que usa distância Euclidiana padrão. A avaliação é feita baseada na acurácia de predição medida através da taxa de erro de classificação.

Nesses experimentos, 100 replicações do conjunto de dados com propriedades estatísticas idênticas são obtidas e, para cada uma, conjuntos de treinamento (75% do conjunto de dados original) e de teste (25% do conjunto de dados original) são aleatoriamente gerados. A taxa de erro de classificação estimada corresponde à média das taxas de erro encontradas nas 100 réplicas do conjunto de teste.

Cada classe nos conjuntos de dados quantitativos foi constituída de acordo com duas distribuições normais independentes.

5.4.1 Configuração 1

Os dados neste conjunto distribuem-se segundo os seguintes parâmetros:

a) Classe 1: $\mu_1 = 45$, $\mu_2 = 30$, $\sigma_1^2 = 100$, $\sigma_2^2 = 16$, $n = 200$;

b) Classe 2: $\mu_1 = 70$, $\mu_2 = 38$, $\sigma_1^2 = 81$, $\sigma_2^2 = 36$, $n = 200$;

c) Classe 3: $\mu_1 = 45$, $\mu_2 = 42$, $\sigma_1^2 = 100$, $\sigma_2^2 = 16$, $n = 200$;

d) Classe 4: $\mu_1 = 42$, $\mu_2 = 20$, $\sigma_1^2 = 81$, $\sigma_2^2 = 36$, $n = 200$;

A Figura 5.1 mostra o conjunto de dados.

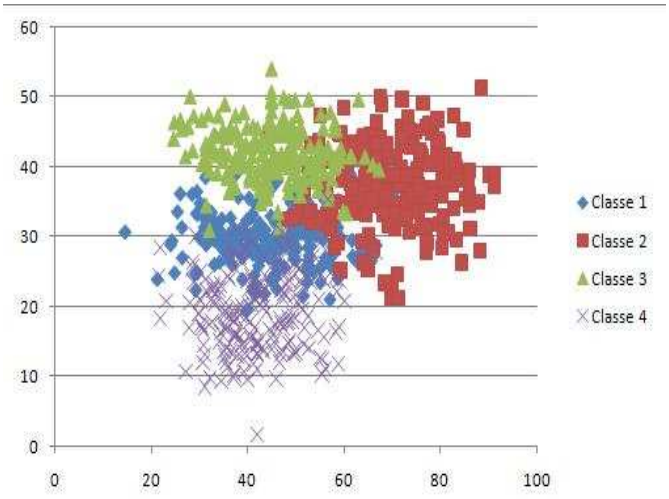

Figura 5.1 Dados gerados com a Configuração 1

Como mencionado inicialmente, 100 replicações desse conjunto de dados foram feitas na estrutura de uma simulação Monte Carlo, para estimar a taxa de erro de lassificação. Dentro dessa estrutura, 10 replicações para escolher conjuntos de protótipos iniciais foram também feitas na estrutura de outra simulação Monte Carlo (estimação dos protótipos citada em 5.2). Foram utilizados 20 protótipos para cada classe. Nesse conjunto os dados foram distribuídos

de forma que as classes tenham tamanhos, formas e volumes parecidos, mas distribuem-se de forma desigual nos dois eixos, ou seja, as variáveis tem comportamentos diferentes. Assim, espera-se que a distância *DEAS* apresente o melhor desempenho. Os valores da média e do desvio-padrão da taxa de erro de classificação para os algoritmos $AQV1O-DEA$ e $AQV1O$ foram os seguintes:

Tabela 5.17 Resultados (Configuração de dados sintéticos 1)

	DE	DEAS	DEAC	DEAP
Erro Médio	25.975%	23.715%	23.955%	23.95%
Desvio Padrão	3.884945692%	3.103643183%	3.568305344%	3.441062204%

Testes de hipóteses foram feitos para comprovar o melhor desempenho do algoritmo $AQV1O-DEA$ em relação ao algoritmo $AQV1O$. O erro médio apresentado nas tabelas é resultado de uma média entre cem taxas de erro, ou seja $n_1 = n_2 = 100$ e, com isso, o valor da tabela t de Student aproxima o valor da tabela Z. Foi utilizado um nível de significância de 5% (valor crítico -1.64). A hipótese nula H_0 é de que $x_1 - x_2 \geqq 0$ em que x_2 equivale ao erro médio do $AQV1O$ utilizando os dados originais nos 3 primeiros testes e ao erro médio do $AQV1O$ utilizando os dados normalizados nos 3 últimos testes e x_1 corresponde ao erro médio do algoritmo com a distância Euclidiana adpatativa correspondente ao teste. E a hipótese alternativa H_a é de que $x_1 - x_2 < 0$.

Teste 1 (*DEAS*): $t \approxeq -4.5450$, $t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância *DEAS* tem desempenho melhor do que o $AQV1O$.

Teste 2 (*DEAC*): $t \approxeq -3.8293$, $t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância *DEAC* tem desempenho melhor do que o $AQV1O$.

Teste 3 (*DEAP*): $t \approxeq -3.9019$, $t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O-DEA$ com distância *DEAP* tem desempenho melhor do que o $AQV1O$.

De acordo com os resultados dos testes para o conjunto de dados gerado com base na Configuração 1, o $AQV1O-DEA$ teve desempenho melhor do que o $AQV1O$ para as três distâncias apresentadas, tendo a distância *DEAS* apresentado melhor desempenho, como era esperado.

5.4.2 Configuração 2

Os dados neste conjunto distribuem-se segundo os seguintes parâmetros:

a) Classe 1: $\mu_1 = 45$, $\mu_2 = 22$, $\sigma_1^2 = 289$, $\sigma_2^2 = 144$, $n = 200$;

b) Classe 2: $\mu_1 = 70$, $\mu_2 = 28$, $\sigma_1^2 = 16$, $\sigma_2^2 = 361$, $n = 150$;

c) Classe 3: $\mu_1 = 50$, $\mu_2 = 42$, $\sigma_1^2 = 36$, $\sigma_2^2 = 36$, $n = 100$;

d) Classe 4: $\mu_1 = 47$, $\mu_2 = 10$, $\sigma_1^2 = 144$, $\sigma_2^2 = 9$, $n = 250$;

A Figura 5.2 mostra o conjunto de dados.

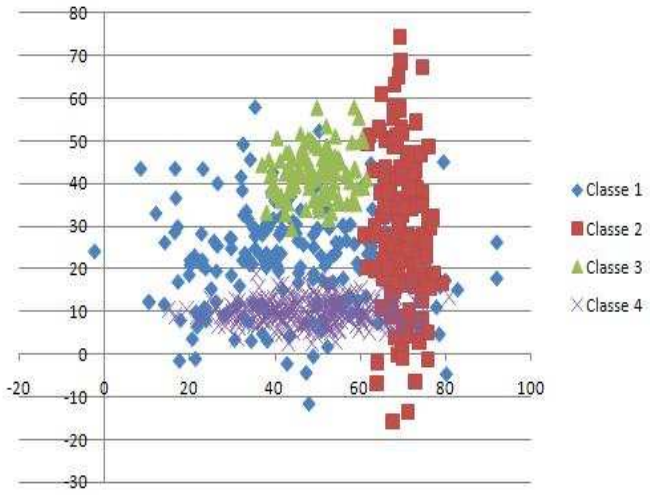

Figura 5.2 Dados gerados com a Configuração 2

Como mencionado inicialmente, 100 replicações desse conjunto de dados foram feitas na estrutura de uma simulação Monte Carlo, para estimar a taxa de erro de lassificação. Dentro dessa estrutura, 10 replicações para escolher conjuntos de protótipos iniciais foram também feitas na estrutura de outra simulação Monte Carlo (estimação dos protótipos citada em 5.2). Foram utilizados 20 protótipos para a classe 1, 15 protótipos para a classe 2, 10 protótipos para a classe 3 e 25 protótipos para a classe 4. Nesse conjunto os dados foram distribuídos de forma que as classes tenham tamanhos, formas e volumes diferentes. Assim, espera-se que a distância *DEAC* apresente o melhor desempenho. Os valores da média e do desvio-padrão da taxa de erro de classificação para os algoritmos $AQV1O - DEA$ e $AQV1O$ foram os seguintes:

Tabela 5.18 Resultados (Configuração de dados sintéticos 2)

	DE	DEAS	DEAC	DEAP
Erro Médio	29.642045454%	26.727272727%	26.420454545%	26.721590909%
Desvio Padrão	3.652947113%	3.765360061%	3.573021406%	4.391812650%

Testes de hipóteses foram feitos para comprovar o melhor desempenho do algoritmo $AQV1O - DEA$ em relação ao algoritmo $AQV1O$. O erro médio apresentado nas tabelas é resultado de uma média entre cem taxas de erro, ou seja $n_1 = n_2 = 100$ e, com isso, o valor da tabela t de Student aproxima o valor da tabela Z. Foi utilizado um nível de significância de 5% (valor crítico -1.64). A hipótese nula H_0 é de que $x_1 - x_2 \geqq 0$ em que x_2 equivale ao erro médio do $AQV1O$ utilizando os dados originais nos 3 primeiros testes e ao erro médio do $AQV1O$ utilizando os dados normalizados nos 3 últimos testes e x_1 corresponde ao erro médio do algoritmo com a distância Euclidiana adpatativa correspondente ao teste. E a hipótese alternativa H_a é de que $x_1 - x_2 < 0$.

Teste 1 (*DEAS*): $t \approxeq -5.5560$, $t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O - DEA$ com distância *DEAS* tem desempenho melhor do que o $AQV1O$.

Teste 2 (*DEAC*): $t \approxeq -6.3047$, $t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O - DEA$ com distância *DEAC* tem desempenho melhor do que o $AQV1O$.

Teste 3 (*DEAP*): $t \approxeq -5.1124$, $t < -1.64$. Assim, existe uma forte evidência de que o $AQV1O - DEA$ com distância *DEAP* tem desempenho melhor do que o $AQV1O$.

De acordo com os resultados dos testes para o conjunto de dados gerado com base na Configuração 2, o $AQV1O - DEA$ teve desempenho melhor do que o $AQV1O$ para as três distâncias apresentadas, tendo a distância *DEAC* apresentado melhor desempenho, como era esperado.

5.4.3 Avaliação geral para os conjuntos de dados sintéticos

Como foi mostrado nos resultados dos experimentos e testes de hipóteses, o algoritmo $AQV1O - DEA$ apresenta desempenho melhor do que o algoritmo $AQV1O$ que utiliza distância Euclidiana padrão. Isso ocorreu porque os dados foram gerados de forma a simular classes com características diferentes, como forma, volume e tamanho. Assim, fica demonstrado a maior capacidade do algoritmo aqui proposto de considerar disparidades entre as regiões de classes no espaço de dados.

A distância *DEAS* mostrou-se melhor em cenários que têm classes de formas e tamanhos semelhantes, mas as variáveis se comportam de formas diferentes e a distância *DEAC* mostrou-se melhor para situações em que as classes tem formas e tamanhos diferentes.

CAPÍTULO 6

Considerações finais

Esse trabalho apresentou o uso de três tipos distâncias adaptativas com um algoritmo da família *AQV*, mais especificamente o $AQV1O$. O objetivo de usar essas distâncias era alcançar métricas que considerassem as características das regiões de classes representadas por protótipos e, com isso, obter melhores resultados comparando-se com o $AQV1O$ original, que utiliza distância Euclidiana padrão.

Foram apresentados todos os passos do novo método e, para comprovar sua capacidade de gerar resultados melhores do que o algoritmo original, o método foi testado e avaliado utilizando-se sete conjuntos de dados reais, além de dois conjuntos de dados sintéticos. Os experimentos e testes de hipóteses comprovaram o melhor desempenho do $AQV1O-DEA$ em quase todas as situações testadas. A distância *DEAS* mostrou-se melhor em cenários que têm classes de formas e tamanhos semelhantes, mas as variáveis se comportam de formas diferentes e a distância *DEAC* mostrou-se melhor para situações em que as classes tem formas e tamanhos diferentes. Não foram criados cenários que fariam a distância *DEAP* ter o melhor desempenho. Como essa distância representa a dispersão dos dados em relação a cada protótipo, ela deve ter o melhor resultado para bases com presença de outliers e ruído.

Como atividades futuras, planeja-se testar o algoritmo proposto com outras métricas, bem como generalizá-lo para outros tipos de dados, em especial os Dados Simbólicos. Além disso, é preciso criar cenários com outliers e ruído que comprovem a maior robustez da distância *DEAP* em relação às outras distâncias nessas situações.

APÊNDICE A

Assinaturas

Renata Maria Cardoso Rodrigues de Souza
Orientadora

Telmo de Menezes e Silva Filho
Aluno

Referências Bibliográficas

[1] Biehl, M., Ghosh, A. and Hammer, B.: Learning vector quantization: The dynamics of winner-takes-all alforithms. Neurocomputing, 69, 660-670, (2006) 1

[2] Diday, E. et Govaert, G. 1977. Classification Automatique avec Distances Adaptatives. R.A.I.R.O. Informatique Computer Science 11 (4), 329-349.

[3] De Carvalho, F.A.T., Tenório, C.P. and Cavalcanti Junior, N.L.: Partitional Fuzzy Clustering Methods Based on Adaptive Quadratic Distances. Fuzzy Sets and Systems, 157, 21, 2833-2857, (2006)

[4] Duda, R.O. Hart P.E. and Stock, D.G: Pattern Classification. John Wiley, (2000) 1

[5] Kohonen, T.: Self-Organizing Maps, Third Ed., Springer, Berlim (2001) 1, 3, 3.2

[6] Jain, A.K. and Dubes, R.C.: ALgorithms for Clustering Data. Prentice Hall, (1998) 1

[7] Sánches, J.S. and Marqués, A.I.: An LVQ-based adaptive algorithm for learning from very small codebooks. Neurocomputing, 69, 922-927, (2006) 1

[8] Aly, M.: Survey on Multiclass Classification Methods, (2005) 2, 2.2, 2.3.1, 2.3.3

[9] Kotsiantis, S.B.: Supervised Machine Learning: A Review of Classification Techniques, (2007) 2.1, 2.2

[10] Hastie, T., Tibshirani, R. and Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference and Prediction, Springer (2001) 2.4.1