André Georghton Cardoso Pacheco

Agregação de classificadores neurais via integral de Choquet com respeito a uma medida *fuzzy*

Vitória

2016

André Georghton Cardoso Pacheco

Agregação de classificadores neurais via integral de Choquet com respeito a uma medida *fuzzy*

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para a obtenção do grau de Mestre em Informática

Universidade Federal do Espírito Santo - UFES

PPGI - Programa de Pós-Graduação em Informática

Orientador: Prof. Dr. -Ing. Renato A. Krohling

Vitória

2016

André Georghton Cardoso Pacheco

Agregação de classificadores neurais via integral de Choquet com respeito a uma medida *fuzzy*

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para a obtenção do grau de Mestre em Informática

O trabalho apresentado foi avaliado e aprovado pela banca examinadora no dia 15 de julho de 2016 em Vitória, Espírito Santo, Brasil.

Prof. Dr. -Ing. Renato A. Krohling
Orientador

Prof. Dr. João Paulo Papa
Membro externo

Profa. Dra. Maria Claudia Silva Boeres
Membro interno

Vitória
2016

Declaração de autoria

Eu, André Georghton Cardoso Pacheco, declaro que este trabalho foi escrito e desenvolvido por mim. Toda e qualquer ajuda recebida durante todo o processo de desenvolvimento foi devidamente reconhecida. Além disso, certifico que todas as fontes de informação e literatura utilizadas são indicadas na dissertação.

André Georghton Cardoso Pacheco

Vitória

2016

Agradecimentos

Agradeço primeiramente aos meus pais que foram o suporte fundamental por toda minha caminhada. Sem eles, nada disso seria possível.

À minha namorada, Mariane, por toda paciência e compreensão desde os tempos de graduação.

Ao professor orientador Renato Krohling, fonte de estímulo e enorme conhecimento. Obrigado pela oportunidade, a mim concedida, de trabalhar ao seu lado.

Aos professores da banca, Maria Claudia Silva Boeres e João Paulo Papa, por aceitarem o convite de participar da mesma.

Aos amigos de laboratório por todas as discussões de ideias e ajuda prestada ao longo deste trabalho.

"A ciência da computação está tão relacionada aos computadores quanto a astronomia aos telescópios, a biologia aos microscópios, ou a química aos tubos de ensaio. A ciência não estuda ferramentas. Ela estuda como nós as utilizamos e o que descobrimos com elas" - Edsger Dijkstra

Resumo

Classificação de dados pode ser aplicada em diversos problemas reais, tais como: reconhecer padrões em imagens, diferenciar espécies de plantas, classificar tumores benignos e malignos, dentre outros. Muitos desses problemas possuem padrões de dados difíceis de serem identificados, o que requer, consequentemente, técnicas mais avançadas para sua resolução. Ao longo dos anos, diversos algoritmos de classificação foram desenvolvidos para abordar esses problemas, todavia, não existe um classificador capaz de ser a melhor opção em todas as situações. Baseado nisso, surge o conceito de sistema baseado em elenco, no qual, mais de uma metodologia é utilizada em conjunto para solucionar um determinado problema. Por ser uma metodologia simples e eficaz, elenco de classificadores vem sendo aplicado em diversos problemas de classificação com intuito de melhorar o desempenho e de aumentar confiabilidade do resultado final. Entretanto, para que o elenco seja capaz de promover melhorias, uma boa técnica de agregação deve ser aplicada. Neste trabalho, duas contribuições são apresentadas: primeiramente será apresentado o uso de três classificadores baseado em redes neurais artificiais, sendo uma rede neural multicamadas *feedforward* usando o algoritmo de treinamento de Levenberg-Marquardt, uma rede neural do tipo máquina de aprendizado extremo (ELM), e uma máquina de Boltzmann restrita discriminativa (DRBM), além de um classificador convencional do tipo K vizinhos mais próximos (KNN). A seguir é proposta uma metodologia de agregação de elenco de classificadores utilizando a integral de Choquet com respeito a uma medida *fuzzy* obtida através da técnica de Análise de Componentes Principais (PCA). Por fim, tal metodologia é aplicada para agregar os classificadores obtidos para *benchmarks* convencionais da literatura, para grande base de dados e os resultados são promissores.

Palavras-chaves: Classificação de dados. Elencos de classificadores. Integral de Choquet. Medida *fuzzy*. Aprendizado profundo.

Abstract

Data classification appears in many real-world problems, e.g., recognition of image patterns, differentiation among species of plants, classifying between benign and malignant tumors, among others. Many of these problems present data patterns, which are difficult to be identified, thus requiring more advanced techniques to be solved. Over the last few years, various classification algorithms have been developed to address these problems, but there is no classifier able to be the best choice in all situations. So, the concept of ensemble systems arise, which more than one methodology is used together to solve a particular problem. As a simple and effective methodology, ensemble of classifiers have been applied in several classification problems, aiming to improve performance and increase reliability of the final result. However, in order to improve the classification accuracy, an affective aggregation of classifiers must be performed. In this work, we present two contributions: first, we describe three classifiers based on neural networks, a multilayer feedforward trained by Levenberg-Marquardt algorithm; an extreme learning machine (ELM); and a discriminative restricted Boltmann machine (DRBM). Furthermore, we use conventional classifier k-nearest neighbors (KNN). Next, we propose an aggregation methodology to ensemble of classifiers using Choquet integral with respect to a fuzzy measure obtained by principal component analysis (PCA). Then, we apply this methodology to aggregate the classifiers performed to conventional benchmarks, for large database and the results are promising.

Keywords: Data classification. Ensemble of classifiers. Choquet Integral. Fuzzy measure. Deep Learning.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Em aprendizado de máquina, sistemas de aprendizagem por elencos são inspirados no comportamento humano em tomadas de decisões. É inerente ao ser humano buscar mais de uma fonte de opinião especialista ao se deparar com uma decisão importante a ser tomada, principalmente nas áreas médica, financeira e social (Štefka; Holeňa, 2010). Um exemplo clássico é a coleta de mais de uma opinião médica relacionada a uma possível intervenção cirúrgica. O indivíduo ouve cada uma das fontes especialistas, neste caso os médicos, considera a opinião de cada uma delas e toma a decisão final, ou seja, realizar ou não o processo cirúrgico, como ilustrado na figura 1. Esse é um processo bastante comum para um ser humano e vem sendo aplicado em diversos problemas que envolvem inteligência computacional (Polikar, 2006).

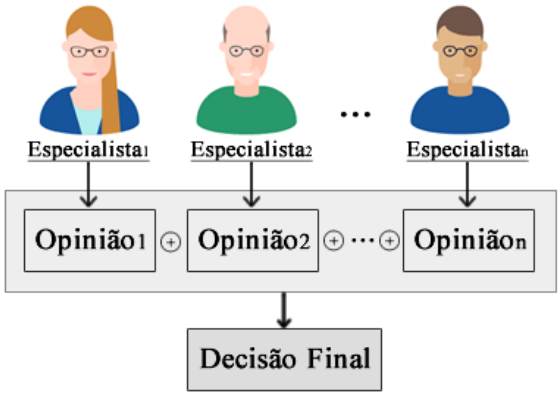


Figura 1 – Processo de tomada de decisão considerando mais de uma opinião

Informações obtidas por fontes diferentes possuem características diferentes. Partindo deste princípio, a agregação de informações é uma tarefa que consiste em produzir, a partir de um conjunto de informações relacionadas, uma única informação que é capaz de extrair as principais características das demais. Atualmente, abordagens que utilizam fusão de informação vêm sendo utilizadas amplamente em problemas como: classificação de dados (Štefka; Holeňa, 2015; Mousavi; Eftekhari, 2015; Seokho; Cho; Kang, 2015), tomada de decisão (Polikar, 2006; Tapia-Rosero et al., 2016), predição de séries temporais (Merigó; Palacios-Marqués; Ribeiro-Navarrete, 2015; Kourentzes; Petropoulos, 2015), dentre outros.

No contexto de classificação de dados, a agregação de classificadores tem como objetivo obter uma classificação mais confiável por meio de um conjunto de diferentes algoritmos de classificação atuando sobre a mesma base de dados. Ao invés de buscar o melhor classificador, busca-se um conjunto de classificadores combinados por um algoritmo de agregação. De maneira geral, um conjunto de classificadores simples pode ser melhor do

que um único classificador complexo (Chen; He; Li, 2010). Baseado nisso, ao longo do tempo diversos algoritmos de agregação foram aplicados para um conjunto de classificadores a fim de se obter melhores resultados. Xu, Krzyżak e Suen (1992) combinaram a teoria de Dempster-Shafer, formalismo Bayesiano e o princípio de votação para reconhecimento de escrita. Em Krawczyk e Woźniak (2016) a agregação é realizada por meio de um procedimento para determinar pesos de uma média ponderada. Já em Štefka e Holeňa (2015), os autores utilizam integral *fuzzy* para realizar agregação. Em Cao (2012) e He et al. (2007) a agregação é obtida por meio da integral de Choquet.

Uma das técnicas utilizadas para agregação de classificadores é a integral de Choquet (Choquet, 1954; Höhle, 1982), que é capaz de realizar a fusão de um conjunto de classificadores levando em consideração a saída individual de cada classificador com respeito a uma medida *fuzzy*. A medida *fuzzy* foi proposta por Sugeno (1974) e é uma forma natural para realizar avaliações envoltas de incerteza e subjetividade. A grande dificuldade em utilizar a integral de Choquet como operador de agregação é justamente determinar quais são os 2^n valores [1] que definem a medida *fuzzy* relacionada ao conjunto a ser agregado. Com isso, Sugeno (1974) propôs também a medida λ-*fuzzy*, uma simplificação da medida *fuzzy*, no qual n valores devem ser determinados. Desde então diversas abordagens supervisionadas foram desenvolvidas com intuito de identificar os valores da medida λ-*fuzzy* (Lee; Leekwang, 1995; Chen; Wang, 2001; Takahagi, 2007; Larbani et al., 2011). Todavia, esses métodos possuem deficiências relacionadas a convergência lenta, necessidade de informação humana e/ou alto consumo de tempo computacional (Krishnan; Kasim; Bakar, 2015).

Baseado nisso, Kojadinovic (2004) propôs uma metodologia não supervisionada, que utiliza conceito de entropia, como alternativa para estimar as medidas *fuzzy* de um conjunto de dados. Porém, tal método sofre com a "maldição da dimensionalidade" (Kojadinovic, 2004) pois o mesmo necessita de um grande conjunto de dados para uma boa estimativa das medidas. Diante das dificuldades dos métodos supervisionados e da fragilidade da abordagem de Kojadinovic, recentemente Rowley, Geschke e Lenzen (2015) propuseram um método não supervisionado, baseado em análise de componentes principais (PCA) (Jolliffe, 2002) e que não possui os requisitos da abordagem de Kojadinovic, capaz de estimar as medidas *fuzzy* de um conjunto de dados. Neste trabalho a medida *fuzzy* é estimada por meio deste último método.

[1] Dado um conjunto X com n elementos, uma medida *fuzzy* requer o cálculo de 2^n valores referentes aos subconjuntos de X. Por exemplo, caso $X = \{a, b, c\}$, deve-se calcular a medida *fuzzy* para cada subconjunto $\{a, b, c, ab, ab, bc, abc, \varnothing\}$

1.1 Objetivo

A abordagem não supervisionada para estimar medida *fuzzy*, desenvolvida por Rowley, Geschke e Lenzen (2015), foi aplicada com sucesso para problemas de tomada de decisão multicritério utilizando a integral de Choquet. Grande parte das metodologias de agregação baseadas em integrais *fuzzy* utilizam a medida simplificada λ-*fuzzy* (Štefka; Holeňa, 2015; Cao, 2012). Tendo em vista as limitações das abordagens para determinar, tanto a medida *fuzzy*, quanto a medida λ-*fuzzy*, o objetivo principal deste trabalho é desenvolver uma metodologia de agregação de classificadores utilizando a integral de Choquet com respeito a uma medida *fuzzy* estimada por meio do novo método proposto por Rowley, Geschke e Lenzen (2015).

1.2 Motivação

Atualmente, tecnologias derivadas de sistemas de aprendizagem de máquina estão sendo utilizadas em diversas atividades da sociedade moderna: buscas na web, recomendações em redes sociais ou em sistemas de *e-commerce*, aplicativos em *smarTVs* e *smartphones*, dentre diversas outras (LeCun; Bengio; Hinton, 2015). Com a crescente demanda dessas tecnologias, algoritmos de aprendizagem de máquina vêm sendo aprimorados e dentre esses algoritmos a classificação de dados se destaca.

Muitos dos problemas de classificação de dados possuem padrões difíceis de serem identificados e, por conta disso, ao longo dos anos, diversos classificadores foram desenvolvidos com objetivo de atacar esses problemas. Classificadores baseados em redes neurais são frequentemente utilizados em classificação de dados. Além disso, uma nova abordagem, conhecida como aprendizado profundo (do inglês: *deep learning*) tem se mostrado uma alternativa promissora para classificação de padrões (Bengio, 2009). Neste contexto, foi utilizado neste trabalho classificadores baseados em redes neurais, tais como: uma rede neural *feedforward* multicamada, uma máquina de aprendizado extremo e uma máquina de Boltzmann restrita discriminativa, baseada em aprendizado profundo. Todavia, não existe um "super-classificador" capaz de ser a melhor solução para todos os problemas. Cada algoritmo tem suas vantagens e desvantagens. Além disso, um classificador sozinho dificilmente é capaz de identificar todas as correlações entre os atributos de entrada e saída de uma determinada base de dados (Chen; He; Li, 2010). Devido a isso surgiu a ideia de agregação de elenco de classificadores.

O trabalho de Dasarathy e Sheela (1979) foi um dos primeiros a discutir o uso de mais de um classificador atuando em uma base de dados simultaneamente. Mesmo após décadas, com frequência, novas abordagens continuam sendo proposta (Krawczyk; Woźniak, 2016). Ao utilizar uma metodologia de agregação que faz uso da integral de Choquet, automaticamente herda-se a problemática questão: como determinar a medida

fuzzy. Como já mencionado, muitos trabalhos abordam essa questão que também é um problema em aberto. A metodologia proposta por Rowley, Geschke e Lenzen (2015) para estimar medida *fuzzy* foi aplicada para problemas de tomada de decisão. Portanto, o objetivo é estender a abordagem para problemas de classificação, que tanto quanto é do nosso conhecimento, não foi desenvolvido ainda (Krohling, 2015).

1.3 Estrutura

O restante deste trabalho está organizado na seguinte estrutura:

- No capítulo 2 são apresentados os conceitos básicos relacionados a classificação de dados. Além disso, quatro classificadores são descritos, sendo três deles baseados em redes neurais.
- No capítulo 3 são descritas as metodologias relacionadas à agregação de elencos de classificadores. Nele, são descritos todos os conceitos necessários para que a agregação seja alcançada, incluindo a proposta deste trabalho para estimar uma medida *fuzzy* para elencos de classificadores.
- No capítulo 4 são apresentados os resultados experimentais. A metodologia de agregação de elenco de classificadores proposta neste trabalho é aplicada a diversos *benchmarks* e seus resultados são comparados e analisados. Além disso, é realizado um estudo de caso com uma base de dados criada exclusivamente para este trabalho.
- No capítulo 5 o trabalho é sumarizado, são apresentadas conclusões e é apontada a direção para trabalhos futuros.

1.4 Lista de publicações

As publicações obtidas ao longo do trabalho são listadas a seguir:

1. Hrasko, Pacheco e Krohling (2015): é proposta uma metodologia híbrida entre uma rede neural artificial (RNA) e uma máquina de Boltzmann restrita (RBM) para abordar o problema de predição de séries temporais. Apesar de neste trabalho o problema de predição em si não ser abordado, os algoritmos utilizados para tal tarefa também são utilizados aqui para classificação e são descritos no capítulo 2.
2. Krohling e Pacheco (2015): é proposto o método A-TOPSIS utilizado para ranquear algoritmos baseado no desempenho dos mesmos. Nesta dissertação o A-TOPSIS é utilizado para comparar os resultados no capítulo 4 e descrito no anexo C.

3. Pacheco e Krohling (2016c): é realizado um estudo utilizando a máquina de Boltzmann restrita discriminativa (DRBM) para grandes bases de dados não extraídas de imagens. Nesta dissertação a DRBM é descrita no capítulo 2 e aplicadas para grandes bases no capítulo 4.

4. Pacheco e Krohling (2016b): é proposta uma metodologia de agregação de classificadores utilizando a integral de Choquet com respeito a medida λ-*fuzzy*. Este é artigo é um trabalho preliminar que utiliza diversos conceitos discutidos ao longo desta dissertação.

5. Pacheco e Krohling (2016a): este é o artigo final na qual é proposta toda metodologia descrita e discutida nesta dissertação. O trabalho esta submetido e ainda sobre revisão.

2 Classificação de dados

A classificação de dados está presente em diversos problemas reais, tais como: reconhecer padrões em imagens, diferenciar espécies de plantas, classificar tumores benignos e malignos, dentre outros (Aggarwal, 2014). Este problema é um dos tópicos mais ativos na área de aprendizado de máquina. Basicamente, o problema de classificação consiste em determinar o rótulo de algum objeto, baseado em um conjunto de atributos extraídos do mesmo. Para que isso ocorra é necessário um conjunto de treinamento com instâncias na qual os rótulos dos objetos são conhecidos. Devido a isso, na terminologia de aprendizado de máquina, a classificação de dados é um problema de aprendizado supervisionado (Kodratoff, 2014). Formalmente, a classificação de dados pode ser definida da seguinte forma:

Definição 2.1. (Goncalves, 2015) *Dado um conjunto de entradas* $X_i = \{x_1, ..., x_a\}$ *com* N *instâncias, sendo* $i = \{1, ..., N\}$ *e* a *o número de atributos de* X*, e um conjunto de rótulos de classificação* $R = \{r_1, ..., r_b\}$*, com* $b \geq 2$*, é montado um conjunto de dados de treinamento* $T = \{(X_1, c_1), ..., (X_i, c_i)\}$ *no qual cada entrada* X_i *é vinculada a um rótulo* $c_i \in R$*. O objetivo de um algoritmo de classificação é aprender, a partir de* T*, uma correlação entre os atributos de entrada tal que: dado uma nova entrada não rotulada* $\hat{X} = \{\hat{x_1}, ..., \hat{x_a}\}$*, o classificador seja capaz de determinar o rótulo vinculado a ela.*

Um exemplo clássico de classificação é o problema da *Iris* (Fisher, 1936). Neste problema, existe um conjunto de flores do gênero *Iris* que são divididas em 3 grupos (rótulos): *setosa*, *virginica* e *versicolor*. Sendo assim, o objetivo é determinar a qual grupo uma determinada flor pertence baseado nas medidas de sépalas e pétalas das mesmas. A figura 2 ilustra um processo de classificação. Inicialmente as sépalas e pétalas devem ser extraídas em um pré-processamento. As medidas extraídas são processadas e suas características extraídas. Por fim, é realizada a classificação das flores. Neste exemplo os valores de X serão as medidas de comprimento e largura das sépalas e pétalas e R assumirá os rótulos *setosa*, *virginica* e *versicolor*.

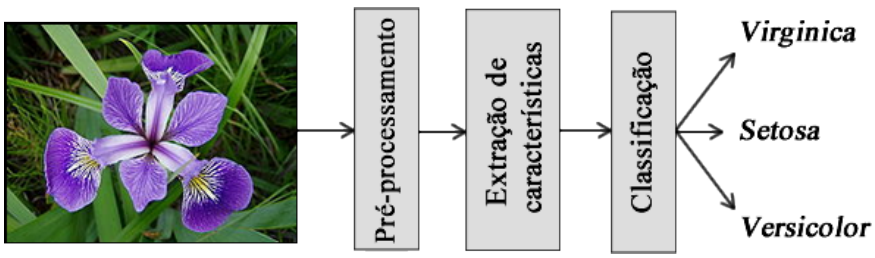


Figura 2 – Exemplo do processo de classificação de flores do gênero *Iris*

A saída de um classificador pode ser apresentada de duas formas (Aggarwal, 2014):

- *Rótulo discreto:* neste caso, um rótulo é retornado para cada amostra.
- *Valor numérico:* neste caso, é retornado um valor numérico para cada rótulo. Esses valores, normalmente entre [0,1], representam o nível de aceitação da hipótese de que a amostra em questão pertença a um determinado rótulo. Normalmente, o rótulo escolhido é aquele que possui o maior valor.

A vantagem de classificadores com valores numéricos é que se torna possível comparar o grau de aceitação de cada rótulo e ranqueá-las, se for necessário. Além disso, é comum utilizar a função *softmax* [1] em classificadores com valores numéricos, pois a mesma transforma esses valores em probabilidades (Heaton, 2015). Sem a *softmax* a saída é apenas valores numéricos. Com ela, os valores representam a probabilidade de cada rótulo para uma dada amostra. Considerando um problema com três rótulos de classificação, {A,B,C}, na figura 3 são ilustrados os dois tipos de classificadores já considerando o uso da função *softmax* no classificador de valor numérico.

Figura 3 – Ilustração de um classificador abstrato e um classificador de valor numérico

Por fim, é importante destacar a diferença entre problemas de classificação e de *clusterização* (também conhecido como agrupamento). A *clusterização* tem como objetivo determinar quantos grupos/rótulos de classes existem em uma determinada base de dados. Portanto, a classificação assume que as classes já são conhecidas previamente através de um processo de agrupamento. Algoritmos de *clusterização* utilizam similaridades entre atributos das variáveis de entrada sem conhecimento prévio de qual grupo a mesma pertence, caracterizando assim, um problema não supervisionado. Essa definição também difere dos problemas de classificação, pelo fato de aprender através de amostras com rótulos conhecidos, é considerado um problema supervisionado.

Existem diversos algoritmos para classificação de dados conhecidos. Neste trabalho, para compor o elenco de classificadores, serão utilizados três algoritmos baseados em redes neurais e um convencional, são eles: uma rede neural artificial *feedforward*, uma máquina

[1] A função *softmax* é definida como: $\sigma(z)_k = \frac{e^{z_k}}{\sum_{j=1}^{k} e^{z_j}}$

de aprendizado extremo (do inglês: *extreme learning machine* - ELM), uma máquina de Boltzmann restrita discriminativa (do inglês: *discriminative restricted Boltzmann machine* - DRBM) e, por fim, o K-vizinhos mais próximos (do inglês: *K-nearest neighbors* - KNN). Os algoritmos baseados em rede neural são descritos nas subseções a seguir e o KNN é descrito no anexo A.

2.1 Rede neural artificial

2.1.1 Conceitos básicos

Uma rede neural artificial é um um sistema de processamento paralelo de informações constituído pela interconexão de unidades básicas de processamento, denominadas neurônios, que tem a propensão natural para armazenar conhecimento experimental e torná-lo disponível para uso (Haykin, 2007). Todo conhecimento adquirido pela rede se dá através de um algoritmo de aprendizagem, cuja função é modificar os pesos das conexões entre os neurônios da rede, conhecidos como pesos sinápticos, de forma ordenada a fim de alcançar o mapeamento desejado. O neurônio artificial é a menor unidade de processamento de uma rede neural, que recebe sinais de entrada e produz sinais de saída. O modelo de neurônio mais utilizado é o *perceptron*, ilustrado na figura 4, que é composto por: m entradas $\{x_1, ..., x_m\}$), m pesos sinápticos $\{w_1, ..., w_m\}$, uma variável de deslocamento linear b (do inglês: *bias*), uma saída intermediária u e a saída final y.

$$y = f(\sum_{i=1}^{m} x_i w_i + b) \tag{2.1}$$

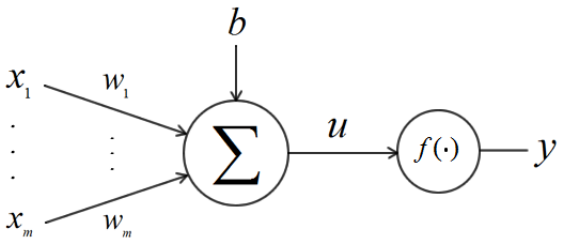


Figura 4 – O modelo do neurônio *perceptron*

A função f é conhecida como função de ativação ou função de transferência. Dentre as funções de ativação a mais utilizada é a *sigmóide* e a tangente hiperbólica, definidas pelas equações 2.2 e 2.3, respectivamente.

$$f(u) = \frac{1}{1 + e^{-u}} \tag{2.2}$$

$$f(u) = \frac{e^u - e^{-u}}{e^u + e^{-u}} \tag{2.3}$$

A arquitetura de uma rede neural está relacionada com a maneira pela qual os neurônios da mesma estão organizados. Para isso, a rede é dividida em três tipos de camadas: a de entrada, a escondida/oculta e a de saída. As camadas de entrada e saída são intuitivas e representam o número de entradas e saídas do problema em questão. Já a escondida é a camada que fará a maior parte do processo de aprendizagem da rede. Normalmente uma rede neural possui uma camada de entrada, uma camada de saída e k camadas escondidas, no qual k é definido empiricamente e varia de acordo com o problema. Com isso forma-se uma rede de múltiplas camadas como mostrado a figura 5.

Redes neurais do tipo *feedforward* são redes de múltiplas camadas nas quais a informação só propaga em um sentido. No caso, os sinais provenientes dos neurônios de uma camada só podem estimular os neurônios da camada seguinte, não existindo realimentação. Na figura 5 a rede neural de múltiplas camadas é do tipo *feedforward*, com uma camada de entrada, com m entradas, k camadas escondidas e uma camada de saída com n saídas.

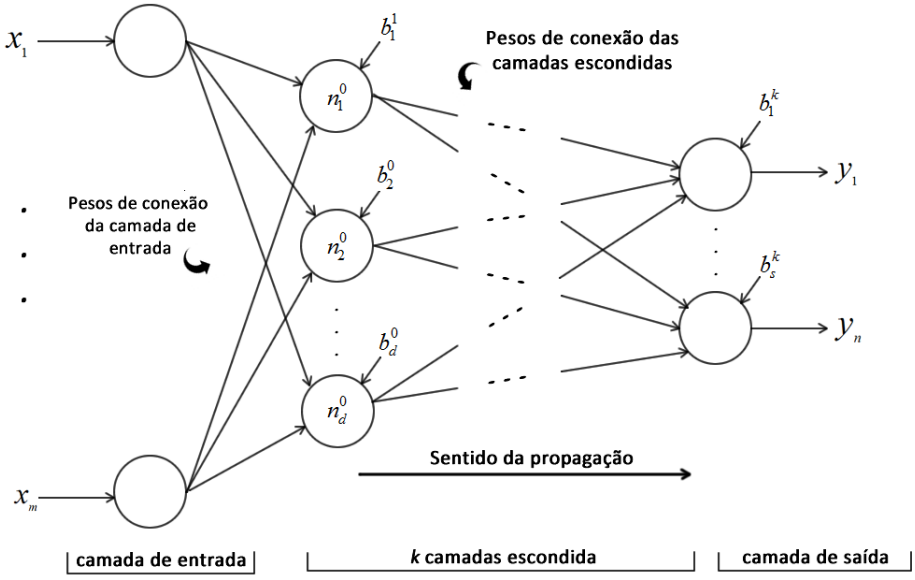


Figura 5 – Rede neural multi-camada do tipo *feedforward*

Toda rede é modelada na forma matricial. Tomando como exemplo a primeira camada da rede da figura 5, os neurônios $[n_1^0, n_2^0, \cdots, n_d^0]$ são obtidos da seguinte forma:

$$[n_1^0, n_2^0, \cdots, n_d^0] = [x_1, \cdots, x_m] \begin{bmatrix} w_{11}^0 & \cdots & w_{1d}^0 \\ \vdots & \ddots & \vdots \\ w_{m1}^0 & \cdots & w_{md}^0 \end{bmatrix} + [b_1^0, b_2^0, \cdots, b_d^0] \tag{2.4}$$

De forma análoga a informação é propagada até que as saídas $[y_1, \cdots, y_n]$ sejam obtidas.

Após especificada a arquitetura da rede, é necessário um algoritmo para determinar a melhor configuração do conjunto de pesos de conexão e *bias*. Esse processo é conhecido como treinamento da rede, no qual a rede aprende os pesos e *bias* através de exemplos que relacionam as entradas e saídas do problema a ser solucionado, caracterizando assim, uma abordagem de aprendizado supervisionado.

2.1.2 Treinamento da rede neural *feedforward*

2.1.2.1 Algoritmo *backpropagation*

Um dos algoritmos mais utilizados para treinamento de uma rede neural artificial é o *backpropagation* (Rumelhart; Hinton; Williams, 1988). A ideia do algoritmo é estimar os valores dos pesos e *bias* minimizando o erro entre a saída desejada e a saída obtida pela rede utilizando o gradiente descendente. Dessa forma, o gradiente de cada peso indicará o quanto o mesmo deve ser modificado para que a saída obtida se aproxime da saída desejada.

O erro entre a saída desejada e a saída obtida é descrito pela equação 2.5 e é conhecido como erro quadrático médio (MSE, do inglês: *mean square error*). O gradiente, nada mais é do que a derivada do erro com respeito aos pesos, como descrito pela equação 2.6.

$$E = \frac{1}{T}\sum_{a=1}^{T}(\hat{y}_t - y_t)^2 \tag{2.5}$$

$$\Delta\mathbf{W} = -\eta\frac{\partial E}{\partial \mathbf{W}} \tag{2.6}$$

onde T é igual ao número de amostras do conjunto de treinamento, $\hat{y}$ a saída desejada e y a saída obtida pela rede. Para facilitar a manipulação matemática o vetor de *bias* é incluído na matriz de pesos $\mathbf{W}$. Aproveitando a modelagem matricial descrita pela equação 2.4, a nova modelagem é descrita da seguinte maneira:

$$[n_1^0, n_2^0, \cdots, n_d^0] = [x_1, \cdots, x_m, 1]\begin{bmatrix} w_{11}^0 & \cdots & w_{1d}^0 \\ \vdots & \ddots & \vdots \\ w_{m1}^0 & \cdots & w_{md}^0 \\ b_1^0 & \cdots & b_d^0 \end{bmatrix} \tag{2.7}$$

No cálculo do gradiente da equação 2.6 η é uma constante conhecida como taxa de aprendizado, que indica o tamanho do passo do gradiente rumo a minimização e o sinal negativo indica a descida do gradiente no espaço de pesos, ou seja, a busca por uma alteração no peso que reduza o valor do erro. Sendo assim, existem três situações relacionadas ao gradiente:

- *Gradiente zero:* não é necessária nenhuma alteração no peso.
- *Gradiente negativo:* o peso deve ser incrementado para diminuir o erro.
- *Gradiente positivo:* o peso deve ser decrementado para diminuir o erro.

Baseado nisso, é gerada a seguinte regra de atualização dos pesos da rede:

$$\mathbf{W}^{t+1} = \mathbf{W}^t + \Delta\mathbf{W}^t \tag{2.8}$$

Nesta etapa o problema em questão é calcular a derivada da equação 2.6. Sendo assim, é obtida a seguinte equação:

$$\Delta w = \eta(\hat{y} - y)f(u)'x \tag{2.9}$$

onde w representa o peso que está sendo treinado, x é a entrada do neurônio que está sendo ponderado pelo peso w, $f(u)'$ é a derivada da função de ativação e u é obtido pela equação 2.1. O caso mais simples é quando o peso em questão pertence à camada de saída. Neste caso, a equação 2.9 é utilizada. Todavia, quando o peso em questão pertence a uma camada escondida, sua atualização é mais complicada pois não é possível ajustar o peso a partir da comparação $\hat{y} - y$, já que y não é conhecido. Além disso, o peso de uma camada escondida exerce influência em todas as saídas da rede, uma vez que a saída de seu neurônio alimenta todos os neurônios da camada seguinte. Para solucionar isso, o algoritmo *backpropagation* introduziu o conceito de erro local para cada neurônio. No caso de um neurônio de saída, o erro continua sendo calculado da mesma forma, ou seja, comparando com o valor desejado. Porém, para um neurônio de camada oculta, o erro é calculado a partir dos erros locais da camada seguinte. Sendo assim, primeiramente é calculado o erro da camada de saída e em sequência, esse erro é propagado para trás, como sugere o nome do algoritmo. Sendo assim, os ajustes dos neurônios são calculados da seguinte forma:

$$\Delta w_{ij} = \eta x_i \delta_j \tag{2.10}$$

onde w_{ij} é o peso que conecta o neurônio i de uma camada ao neurônio j da camada seguinte, x_i é, ao mesmo tempo, a saída do neurônio i de uma camada e a entrada do neurônio j da camada seguinte e δ_j corresponde ao erro local do neurônio j, que é calculado da seguinte forma:

$$\delta_j = \begin{cases} f(u_j)'(\hat{y_j} - y_j), & \text{se } j \text{ é neurônio de camada de saída} \\ f(u_j)' \sum_{k=1}^{K} w_{kj}\delta_k, & \text{se } j \text{ é neurônio de camada escondida} \end{cases} \tag{2.11}$$

onde u_j é o resultado da equação 2.1 para o neurônio j, $\hat{y_j}$ e y_j são, respectivamente, a saída desejada e saída obtida do neurônio de saída j, w_{kj} é o peso que liga o neurônio j ao

neurônio k da camada seguinte e K é o número de neurônios da camada posterior à camada na qual se encontra o neurônio j. A derivada da função de transferência $f(u_j)'$ depende, obviamente, da função a ser utilizada. No algoritmo 1 é apresentado o pseudocódigo do *backpropagation*. O critério de parada do algoritmo é um parâmetro de projeto da rede neural, podendo ser: um número de iterações pré-definido, um erro mínimo ou verificação de convergência do erro.

Algoritmo 1: *backpropagation*

```
inicialização:
    Preparar conjunto de dados de entrada e saída T de acordo com a definição 2.1;
    Informar número de camadas ocultas e neurônios para cada uma dela(s);
    Informar η;
repita
    para cada amostra do conjunto de treinamento T faça
        para cada peso de conexão w faça
            Calcular ΔW utilizando as equações 2.11 e 2.10;
            Atualizar W de acordo com a equação 2.8;
        fim para
    fim para
Até número de iterações pré-determinado ou erro mínimo satisfeito
retornar: pesos e bias treinados
```

2.1.2.2 *Backpropagation* utilizando o algoritmo Levenberg-Marquardt

A metodologia clássica do algoritmo *backpropagation*, apresentada até aqui, possui algumas desvantagens relacionadas a tempo de convergência e convergência para mínimos locais. Existem diferentes otimizações do algoritmo com objetivo de melhorar o seu desempenho. Uma das mais conhecidas é o Levenberg-Marquardt *backpropagation*, versão utilizada neste trabalho. Os fundamentos do algoritmo Levenberg-Marquardt (LMA) foram introduzidos por Levenberg (1944) e expandido por Marquardt (1963). A metodologia para aplicação do LMA à treinamento de redes neurais foi introduzida mais tarde por Hagan e Menhaj (1994).

O LMA é um algoritmo híbrido baseado no método de Newton (Polyak, 2007) e no gradiente descendente. Assim como no método de Newton, o LMA é baseado no cálculo de derivadas de segunda ordem. As derivadas primeiras, utilizadas no *backpropagation* clássico podem ser organizadas em forma de matriz. Essa matriz é conhecida como Jacobiana e é descrita pela equação a seguir:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial E}{\partial w_1} \\ \vdots \\ \frac{\partial E}{\partial w_n} \end{bmatrix} \tag{2.12}$$

As derivadas segundas também são organizadas em forma de matriz, sendo essa conhecida como Hessiana e descrita pela seguinte equação:

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 E}{\partial w_1^2} & \cdots & \frac{\partial^2 E}{\partial w_1 \partial w_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial w_n \partial w_1} & \cdots & \frac{\partial^2 E}{\partial w_n^2} \end{bmatrix} \tag{2.13}$$

A regra de atualização dos pesos da rede, segundo o método de Newton, é descrita por (Hagan; Menhaj, 1994):

$$\mathbf{W}^{t+1} = \mathbf{W}^t - \mathbf{H}^{-1}\mathbf{G} \tag{2.14}$$

onde $\mathbf{G}$ é o gradiente da rede neural, que pode ser calculado como $\mathbf{G} = \mathbf{J}^T\mathbf{E}$, tal que $\mathbf{J}^T$ é a transposta de $\mathbf{J}$ e $\mathbf{E}$ é o vetor de erros da rede. A matriz Hessiana é difícil de ser obtida. Por conta disso, ela é estimada da seguinte forma:

$$\mathbf{H} \approx \mathbf{J}^T\mathbf{J} \tag{2.15}$$

O LMA propõe a seguinte modificação na equação 2.14:

$$\mathbf{W}^{t+1} = \mathbf{W}^t - (\mathbf{H} + \gamma\mathbf{I})^{-1}\mathbf{G} \tag{2.16}$$

sendo $\mathbf{I}$ a matriz identidade. Por fim, a regra final de atualização dos pesos é descrita por:

$$\mathbf{W}^{t+1} = \mathbf{W}^t - (\mathbf{J}^T\mathbf{J} + \gamma\mathbf{I})^{-1}\mathbf{J}^T\mathbf{E} \tag{2.17}$$

A constante γ é conhecida como fator de amortecimento e é ela quem garante que o algoritmo seja híbrido. Diminuir o valor γ significa privilegiar o método de Newton. Por outro lado, quando o valor γ é aumentado, o gradiente descendente se torna o favorecido. O método de Newton é mais rápido e mais preciso para encontrar um erro mínimo, então a ideia é iniciar com um valor pequeno para γ, que por sua vez é decrementado caso o erro E diminua e incrementado caso aumente (Hagan; Menhaj, 1994). No algoritmo 2 é descrito o pseudocódigo do LMA *backpropagation* onde γ_i é o valor inicial de γ, γ_a e γ_d são os fatores de aumento e diminuição.

2.1.2.3 A técnica *mini-batch*

Uma técnica comumente utilizada para melhorar o desempenho, não só do *backpropagation* e do Levenberg-Marquardt*backpropagation*, mas de algoritmos de treinamento de redes neurais em geral, é conhecida como *mini-batch*. A ideia do *mini-batch* é, basicamente, dividir o conjunto de treinamento, em um número predeterminado de partes, com objetivo

Algoritmo 2: LMA *backpropagation*

inicialização:
    Preparar conjunto de dados de entrada e saída T de acordo com a definição 2.1;
    Informar número de camadas ocultas e neurônios para cada uma dela(s);
    Informar γ_i, γ_a e γ_d;
repita
    para cada amostra do conjunto de treinamento T **faça**
        para cada peso de conexão w **faça**
            Calcular a matriz **J** utilizando conceitos do *backpropagation*;
            Calcular a matriz $\mathbf{W}^{t+1}$ utilizando a equação 2.17;
        fim para
        se erro diminuir **então**
            $\gamma_i = \gamma_i \times \gamma_d$;
        se erro aumentar **então**
            $\gamma_i = \gamma_i \times \gamma_a$;
        fim se
    fim para
Até número de iterações pré-determinado ou erro mínimo satisfeito
retornar: pesos e bias treinados

de agilizar o processo de treinamento e evitar *overfiting* (Hinton, 2010). Na figura 6 é ilustrado um pequeno exemplo da técnica de *mini-batch*. Nele um conjunto de treinamento de apenas 8 amostras é dividido, de maneira aleatória, em 2 *batchs*. Obviamente o tamanho do *batch* varia de acordo com o tamanho do conjunto de treinamento e escolhas comuns estão no intervalo 10-100 (Hinton, 2010).

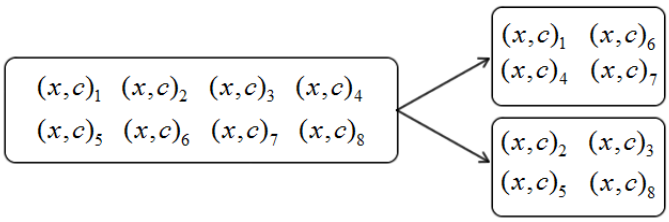


Figura 6 – Exemplo de aplicação da técnica *mini-batch*

2.2 Máquina de aprendizado extremo (ELM)

2.2.1 Conceitos básicos

A máquina de aprendizado extremo (do inglês: *extreme learning machine* - ELM) é um algoritmo de aprendizado baseado em uma rede neural artificial de apenas uma camada oculta (Huang; Zhu; Siew, 2004) (Huang; Zhu; Siew, 2006). O princípio de funcionamento da ELM é o mesmo de uma rede neural artificial, todavia a metodologia de treinamento

de uma ELM não é baseada em gradiente descendente. Com isso o algoritmo escapa das principais deficiências do *backpropagation*: convergência lenta e convergência para mínimos locais. Segundo Huang, Zhu e Siew (2006) o treinamento de uma ELM pode ser milhares de vezes mais rápido do que o treinamento via *backpropagation*.

A arquitetura da ELM é a mesma que a ilustrada na figura 5 quando $k = 1$. O vetor $\mathbf{X}$ é a entrada da rede. Os pesos de conexão da camada de entrada são alocados em uma matriz denominada $\mathbf{W}$ e já os da camada oculta em uma matriz denominada $\boldsymbol{\beta}$. Os *bias* dos neurônios da camada oculta também são alocados em $\mathbf{W}$, da mesma forma da equação 2.7, e os *bias* da camada oculta não são utilizados na ELM. A modelagem matricial do ELM é descrita a seguir:

$$\mathbf{X} = [x_1, \cdots, x_m, 1] \qquad \mathbf{W} = \begin{bmatrix} w_{11} & \cdots & w_{1d} \\ \vdots & \ddots & \vdots \\ w_{m1} & \cdots & w_{md} \\ b_1 & \cdots & b_d \end{bmatrix} \qquad \boldsymbol{\beta} = \begin{bmatrix} \beta_{11} & \cdots & \beta_{1s} \\ \vdots & \ddots & \vdots \\ \beta_{d1} & \cdots & \beta_{ds} \end{bmatrix} \qquad \mathbf{Y} = [y_1, \cdots, y_s] \tag{2.18}$$

onde m é o número de neurônios de entradas, d é o número de neurônios na camada oculta e s é o número de neurônios de saída da rede.

2.2.2 Treinamento da ELM

O treinamento da ELM é realizado de maneira analítica, diferentemente da abordagem iterativa do *backpropagation*. A matriz $\mathbf{W}$ é gerada de maneira aleatória e não é alterada até o fim do algoritmo. Portanto, o objetivo do treinamento da ELM é encontrar a matriz de pesos $\boldsymbol{\beta}$, baseado na matriz de saída $\mathbf{Y}$ e na matriz de pesos aleatórios $\mathbf{W}$, por meio da resolução de um sistema linear. Para isso, o primeiro passo é determinar a matriz $\mathbf{H}$ da seguinte maneira:

$$\mathbf{H}^i = [x_1^i, \cdots, x_m^i, 1] \begin{bmatrix} w_{11} & \cdots & w_{1d} \\ \vdots & \ddots & \vdots \\ w_{m1} & \cdots & w_{md} \\ b_1 & \cdots & b_d \end{bmatrix} \Rightarrow \mathbf{H} = \begin{bmatrix} f(H^1) \\ f(H^2) \\ \vdots \\ f(H^N) \end{bmatrix}_{N \times d} \tag{2.19}$$

onde a função $f(.)$ é a função de transferência da camada e $i = \{1, \cdots, N\}$, sendo N o número de amostras do conjunto de treinamento. Portanto, a matriz $\mathbf{H}$ armazena o resultado de todos os neurônios da camada oculta obtidos a partir de $\mathbf{X}$ e $\mathbf{W}$. Uma vez determinada a matriz $\mathbf{H}$, para se obter os pesos da matriz $\boldsymbol{\beta}$ deve ser solucionado o seguinte sistema linear:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{Y} \rightarrow \boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{Y} \tag{2.20}$$

onde $\mathbf{H}^{\dagger}$ é a inversa generalizada de Moore–Penrose (Serre, 2002) da matriz $\mathbf{H}$. O pseudo-código do treinamento de uma ELM é apresentado no Algoritmo 3.

Algoritmo 3: *treinamento ELM*

inicialização:
    Preparar conjunto de dados de entrada e saída T de acordo com a definição 2.1;
    Informar número de neurônios para a camada oculta;
    Inicializar $\mathbf{W}$ de maneira aleatória;
para cada amostra de treinamento do conjunto T **faça**
    Calcular $\mathbf{H}^{i}$ por meio da equação 2.19;
fim para
Obter a matriz $\mathbf{H}$ por meio de $\mathbf{H}^{i}$;
Obter a matriz $\boldsymbol{\beta}$ por meio do sistema linear da equação 2.20;
retornar: pesos e bias treinados

Devido ao fato do treinamento da ELM ser executado de forma analítica, o mesmo é realizado de maneira mais rápida do que um método iterativo (Huang; Zhu; Siew, 2006). Todavia a abordagem possui suas fraquezas. A primeira delas é relacionada a inicialização aleatória dos pesos da matriz $\mathbf{W}$. Pode ocorrer dos valores obtidos para $\mathbf{W}$ desencadear, ao fim do processo, em uma matriz $\boldsymbol{\beta}$ que proporcione um resultado final ruim. Por conta disso, existem trabalhos que visam otimizar a escolha dos valores de $\mathbf{W}$ por meio de algoritmos evolutivos (Han; Yao; Ling, 2013). Outra fraqueza é o fato do algoritmo trabalhar com a inversa generalizada da matriz $\mathbf{H}$. Caso a rede possua muitas amostras e muitos neurônios na camada oculta, obter a inversa generalizada de $\mathbf{H}$ pode demandar bastante recurso computacional.

2.3 Máquinas de Boltzmann

Neste trabalho a máquina de Boltzmann discriminativa restrita foi utilizada como um dos classificadores do elenco. Todavia, para descrever o modelo da mesma se faz necessário, primeiramente, abordar a máquina de Boltzmann restrita, sua precursora. A máquina de Boltzmann restrita (do inglês: *restricted Boltzmann machine*, RBM) é uma rede estocástica amplamente utilizada para compor redes de crença profundas (do inglês: *deep belief networks*, DBN) (Hinton; Osindero; Teh, 2006). A RBM é capaz de extrair características de um conjunto de dados por meio de treinamento não supervisionado. Devido a isso, abordagens que utilizam RBM's, para compor uma DBN, foram desenvolvidas como primeiro estágio de um classificador baseado em redes neurais artificiais (Salama; Hassanien; Fahmy, 2010; Tamilselvan; Wang, 2013). Na figura 7 é mostrado uma arquitetura híbrida baseada em composições de RBM's alimentando uma rede neural, constituindo assim, uma abordagem semi-supervisionada.

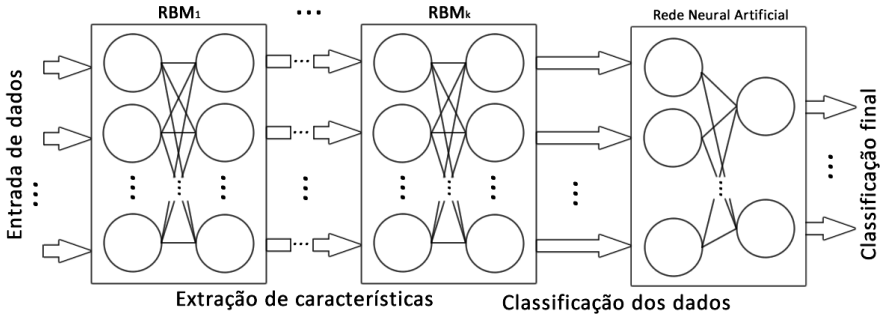


Figura 7 – Arquitetura híbrida semi-supervisionada

Baseado na arquitetura híbrida, Larochelle e Bengio (2008) e Larochelle et al. (2012) propuseram um modelo completo, baseado em RBM, que engloba tanto a abordagem supervisionada, quanto a não supervisionada, denominado máquina de Boltzmann restrita discriminativa (do inglês: *discriminative restricted Boltzmann machine*, DRBM). Além dos dados, a DRBM necessita dos rótulos de classificação de cada amostra. Com isso, o algoritmo é capaz de extrair as características do conjunto, assim como uma RBM, e classificar a amostra, sem a necessidade de acoplar um outro algoritmo supervisionado à abordagem. Até o momento a DRBM vem sendo utilizada para classificação de dados extraídos de imagens, principalmente classificação de dígitos escritos à mão (Larochelle et al., 2012; Ji et al., 2014; Papa et al., 2015). Neste trabalho, o algoritmo foi utilizado para bases que não são extraídas de imagens, nos experimentos com *benchmarks* da literatura, e para uma base extraída de imagem no estudo de caso.

2.3.1 Máquina de Boltzmann restrita (RBM)

2.3.1.1 Conceitos básicos

A máquina de Boltzmann restrita (Smolensky, 1986; Hinton, 2002) é, basicamente, uma rede estocástica constituída por duas camadas: visível e oculta. A camada de unidades visíveis representam os dados observados e está conectada à camada oculta, que por sua vez, deverá aprender a extrair características desses dados (Memisevic; Hinton, 2010). Originalmente, a RBM foi desenvolvida para dados binários, tanto na camada visível quanto na camada oculta. Essa abordagem é conhecida como Bernoulli-Bernoulli RBM (BBRBM). Devido ao fato de existir problemas onde é necessário processar outros tipos de dados, Hinton e Salakhutdinov (2006) propuseram a Gaussian-Bernoulli RBM (GBRBM), que utiliza uma distribuição normal para modelar os neurônios da camada visível. Nesta seção, serão descritos os conceitos básicos referentes a abordagem GBRBM.

Na RBM as conexões entre neurônios são bidirecionais e simétricas. Isso significa que existe tráfego de informação em ambos os sentidos da rede. Além disso, para simplificar procedimentos de inferência, neurônios de uma mesma camada não estão conectados entre

si. Sendo assim, só existe conexão entre neurônios de camadas diferentes, por isso a máquina é restrita. Na figura 8 é mostrado uma RBM com m neurônios na camada visível $(v_1, ..., v_m)$, n neurônios na camada oculta $(h_1, ..., h_n)$, sendo $(a_1, ..., a_m)$ e $(b_1, ..., b_n)$ os vetores de *bias* e por fim $\mathbf{W}$ a matriz de pesos das conexões. Daqui até o fim desta seção, o conjunto $(\mathbf{W}, \mathbf{a}, \mathbf{b})$ será denominado $\boldsymbol{\theta}$.

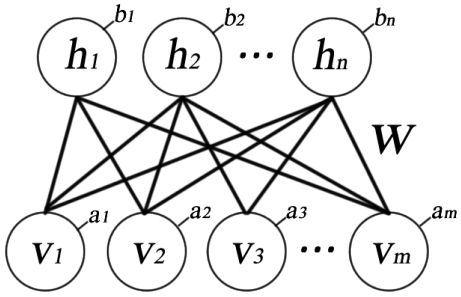


Figura 8 – Máquina de Boltzmann restrita

A RBM é um modelo baseado em energia, isso significa que a distribuição de probabilidade conjunta da configuração (**v**,**h**) é obtida pela equação 2.21, sendo a função de energia descrita pela equação 2.22.

$$p(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta}) = \frac{e^{-E(\mathbf{v},\mathbf{h};\boldsymbol{\theta})}}{\sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h};\boldsymbol{\theta})}} \tag{2.21}$$

$$E(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta}) = -\sum_{i=1}^{m} \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_{j=1}^{n} b_j h_j - \sum_{i,j=1}^{m,n} \frac{v_i}{\sigma^2} h_j w_{ij} \tag{2.22}$$

A probabilidade que a rede atribui a um vetor visível, $\mathbf{v}$, é dada pela soma de todas as probabilidades dos vetores escondidos $\mathbf{h}$, calculados por:

$$p(\mathbf{v}; \boldsymbol{\theta}) = \frac{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h};\boldsymbol{\theta})}}{\sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h};\boldsymbol{\theta})}} \tag{2.23}$$

Como a RBM é restrita, ou seja, não possui conexões de neurônios entre uma mesma camada, as distribuições de probabilidade de $\mathbf{h}$ dado $\mathbf{v}$ e de $\mathbf{v}$ dado $\mathbf{h}$ são descritas pelas equações 2.24 e 2.25, respectivamente.

$$p(\mathbf{h}|\mathbf{v}; \boldsymbol{\theta}) = \prod_j p(h_j|\mathbf{v}) \tag{2.24}$$

$$p(\mathbf{v}|\mathbf{h}; \boldsymbol{\theta}) = \prod_i p(v_i|\mathbf{h}) \tag{2.25}$$

Baseado na versão GBRBM (Hinton; Salakhutdinov, 2006), no qual a camada visível é contínua e a oculta binária, as distribuições condicionais são descritas pelas equações 2.26 e 2.27.

$$p(h_j = 1|\mathbf{v};\boldsymbol{\theta}) = \phi(b_j + \sum_{i=1}^{m} v_i w_{ij}) \tag{2.26}$$

$$p(v_i = v|\mathbf{h};\boldsymbol{\theta}) = N(v|a_i + \sum_{j=1}^{n} h_j w_{ij}, \sigma^2) \tag{2.27}$$

onde $\phi(x) = \frac{1}{1+e^{-x}}$, a função logística, e N é uma distribuição de probabilidade normal, com média v e desvio padrão σ^2, normalmente utilizado como 1.

2.3.1.2 Treinamento de uma RBM

Basicamente, o objetivo do treinamento da RBM é estimar $\boldsymbol{\theta}$ que faça com que a energia da rede diminua (Hinton, 2010). Como $p(\mathbf{v};\boldsymbol{\theta})$ é a distribuição dos dados de entrada, $\boldsymbol{\theta}$ pode ser estimado a partir da maximização de $p(\mathbf{v},\boldsymbol{\theta})$ ou, de maneira equivalente, log $p(\mathbf{v},\boldsymbol{\theta})$. Sendo assim, o gradiente descendente de log $p(\mathbf{v},\boldsymbol{\theta})$, com respeito a $\boldsymbol{\theta}$ é calculado por:

$$\frac{\partial p(\mathbf{v},\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \langle v_i h_j \rangle_d - \langle v_i h_j \rangle_m \tag{2.28}$$

onde as componentes $\langle v_i h_j \rangle_d$ e $\langle v_i h_j \rangle_m$ são usadas para denotar as expectativas computadas sob os dados e o modelo, respectivamente.

A estimativa de $\langle v_i h_j \rangle_d$ é obtida de maneira simples através das equações 2.26 e 2.27. Todavia, obter uma estimativa de $\langle v_i h_j \rangle_m$ é muito mais difícil. Isso pode ser feito por meio do amostrador de Gibbs utilizando dados aleatórios alimentando a camada visível. Todavia, esse procedimento pode consumir um longo tempo para obter um resultado adequado. Felizmente, um procedimento mais rápido, denominado *contrastive divergence* (CD), foi proposto por Hinton (2002) cuja ideia é alimentar a camada visível com dados de treinamento e executar o amostrador de Gibbs apenas uma vez, como ilustrado na figura 9. Esta etapa foi denominada por Hinton (2002) como reconstrução.

Para aplicar o algoritmo CD, o primeiro passo é igualar a camada visível $\mathbf{v_0}$ aos dados de entradas e, logo após, estimar a camada oculta $\mathbf{h_0}$ por meio da equação 2.26. Com isso, $\left\langle \mathbf{v}\mathbf{h}^\mathbf{T} \right\rangle_d = \mathbf{v_0}\mathbf{h_0^T}$. Em seguida, a partir de $\mathbf{h_0}$, deve-se estimar $\mathbf{v_1}$ por meio da equação 2.27. De maneira similar, a partir de $\mathbf{v_1}$, estima-se $\mathbf{h_1}$, novamente por meio

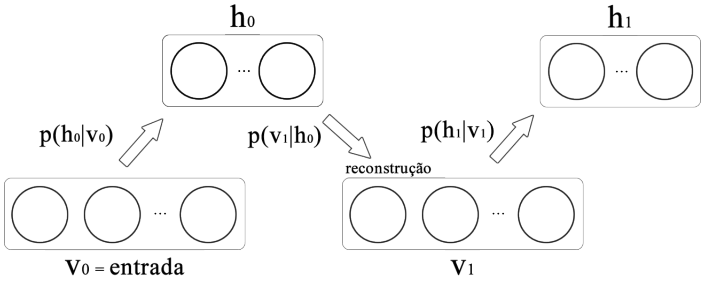


Figura 9 – Processo de reconstrução do algoritmo CD para RBM

da equação 2.26. Com isso, $\left\langle \mathbf{vh^T} \right\rangle_m = \mathbf{v_1 h_1^T}$. Por fim, o conjunto de parâmetros $\boldsymbol{\theta}$ são atualizados da seguinte forma:

$$\mathbf{W^{t+1}} = \mathbf{W^t} + \Delta\mathbf{W^t} \rightarrow \Delta\mathbf{W^t} = \hat{\eta}(\mathbf{v_0 h_0^T} - \mathbf{v_1 h_1^T}) - \hat{\rho}\mathbf{W^t} + \hat{\alpha}\Delta\mathbf{W^{t-1}} \tag{2.29}$$

$$\mathbf{a^{t+1}} = \mathbf{a^t} + \Delta\mathbf{a^t} \rightarrow \Delta\mathbf{a^t} = \hat{\eta}(\mathbf{v_0} - \mathbf{v_1}) + \hat{\alpha}\Delta\mathbf{a^{t-1}} \tag{2.30}$$

$$\mathbf{b^{t+1}} = \mathbf{b^t} + \Delta\mathbf{b^t} \rightarrow \Delta\mathbf{b^t} = \hat{\eta}(\mathbf{h_0} - \mathbf{h_1}) + \hat{\alpha}\Delta\mathbf{b^{t-1}} \tag{2.31}$$

sendo que $(\mathbf{W}, \mathbf{a}, \mathbf{b})$ são inicializados de maneira aleatória. No algoritmo 4 é descrito o pseudocódigo do algoritmo CD.

Algoritmo 4: *treinamento RBM*

inicialização:
    Preparar conjunto de dados de entrada;
    Informar número de neurônios para camada oculta $\mathbf{h}$;
    Inicializar os parâmetros $\hat{\eta}$, $\hat{\rho}$ e $\hat{\alpha}$;
    Inicializar $\boldsymbol{\theta}$ de maneira aleatória;
repita:
    Igualar a camada visível $\mathbf{v}_0$ aos dados de entrada;
    Estimar a camada oculta $\mathbf{h}_0$ por meio da equação 2.26;
    Estimar, a partir de $\mathbf{h}_0$, a camada visível $\mathbf{v}_1$ por meio da equação 2.27;
    Estimar, a partir de $\mathbf{v}_1$, a camada oculta $\mathbf{h}_1$ por meio da equação 2.26;
    Atualizar $\boldsymbol{\theta}$ por meio das equações 2.29, 2.30 e 2.31;
Até número de iterações pré-determinado ou erro mínimo satisfeito
retornar: $\boldsymbol{\theta}$ treinado

Os parâmetros $\hat{\eta}$, $\hat{\rho}$ e $\hat{\alpha}$ são conhecidos como taxa de aprendizado, fator de decaimento e *momentum*. (Hinton, 2010) sugere $\hat{\eta} = 0.01$, $\hat{\rho} = [0.01, 0.0001]$ e $\hat{\alpha} = 0.5$ para iteração menor do que 5 e $\hat{\alpha} = 0.9$ caso contrário. Trabalhos relacionados a escolhas destes parâmetros têm sido desenvolvidos (Liu; Zhang; Sun, 2014; Papa et al., 2015; Papa; Scheirer; Cox, 2016).

2.3.2 Máquina de Boltzmann restrita discriminativa (DRBM)

2.3.2.1 Conceitos básicos

A máquina de Boltzmann restrita discriminativa é, basicamente, um incremento na RBM com objetivo de torná-la um algoritmo semi-supervisionado para problemas de classificação. A ideia principal do método é incorporar na RBM os rótulos de cada classe na camada de entrada, e com isso, determinar a distribuição conjunta, tanto dos dados, quanto dos rótulos, de uma maneira discriminativa, ou seja, calcular a probabilidade de cada classe dado uma certa amostra (Papa et al., 2015).

Além das características comuns à RBM, a DRBM possui uma camada de rótulos, representada por $\mathbf{y}$, os pesos de das conexões entre $\mathbf{y}$ e $\mathbf{h}$, representado por $\mathbf{U}$ e o vetor de bias, $\mathbf{c}$. Na figura 10 é ilustrada uma DRBM no qual as ligações entre a camada de rótulos e a camada oculta estão destacadas em azul. O conjunto $\boldsymbol{\theta}$, definido da seção 2, será acrescido de $\mathbf{c}$ e $\mathbf{U}$.

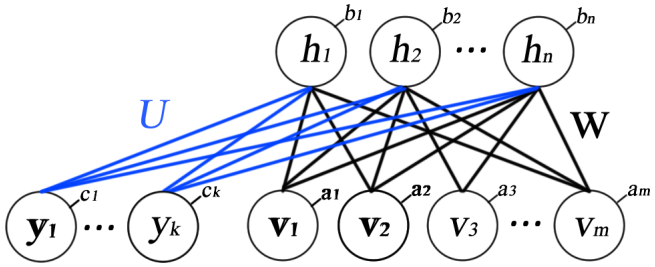


Figura 10 – A máquina de Boltzmann restrita discriminativa

O modelo matemático da DRBM é bastante similar ao da RBM. Sendo assim, serão apresentadas nesta seção as diferenças entre as abordagens tendo como base a modelagem descrita na seção anterior. Para começar, são descritas pelas equações 2.32 e 2.33 a distribuição de probabilidade conjunta da configuração $(\mathbf{y}, \mathbf{v}, \mathbf{h})$ e a função de energia, respectivamente:

$$p(\mathbf{y}, \mathbf{v}, \mathbf{h}; \boldsymbol{\theta}) = \frac{e^{-E(\mathbf{y},\mathbf{v},\mathbf{h};\boldsymbol{\theta})}}{\sum_{\mathbf{y},\mathbf{v},\mathbf{h}} e^{-E(\mathbf{y},\mathbf{v},\mathbf{h};\boldsymbol{\theta})}} \tag{2.32}$$

$$E(\mathbf{y}, \mathbf{v}, \mathbf{h}; \boldsymbol{\theta}) = -\sum_{i=1}^{m} \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_{j=1}^{n} b_j h_j - \sum_{z=1}^{k} c_z y_z - \sum_{i,j=1}^{m,n} \frac{v_i}{\sigma^2} h_j w_{ij} - \sum_{z,j=1}^{k,n} y_z h_j u_{zj} \tag{2.33}$$

Das equações 2.32 e 2.33 as distribuições de probabilidade condicionais são obtidas. A distribuição de probabilidade da camada visível dado a camada oculta é idêntica a equação 2.27 da seção anterior. Já a probabilidade da camada oculta dado a camada de rótulos

e camada visível e da probabilidade da camada de rótulos dado a camada oculta, são descritas pelas equações 2.34 e 2.35, respectivamente.

$$p(h_j = 1|\mathbf{y}, \mathbf{v}; \boldsymbol{\theta}) = \phi(b_j + \sum_{z=1}^{k} y_z u_{zj} + \sum_{i=1}^{m} v_i w_{ij}) \tag{2.34}$$

$$p(y_z = 1|\mathbf{h}; \boldsymbol{\theta}) = \frac{e^{(c_z + \sum_{j=1}^{n} h_j u_{zj})}}{\sum_{l=1}^{k} e^{(c_l + \sum_{j=1}^{n} h_j u_{lj})}} \tag{2.35}$$

2.3.2.2 Treinamento de uma DRBM

O treinamento de uma DRBM segue os mesmos princípios do treinamento da RBM e por isso será apresentado de forma sucinta. O aprendizado dos pesos de conexão $\mathbf{W}$ e os vetores de *bias* $\mathbf{a}$ e $\mathbf{b}$, são obtidos pelas equações 2.29, 2.30 e 2.31. Todavia, para o processo de amostragem da camada oculta é utilizado a equação 2.34, como mostrado na figura 11. Além disso, é necessário treinar os pesos de conexão $\mathbf{U}$ e o vetor *bias* $\mathbf{c}$. O processo é similar e é descrito pelas equações 2.36 e 2.37. Na figura 11 também é ilustrado o processo de amostragem da camada de rótulos a partir da camada oculta utilizando a equação 2.35.

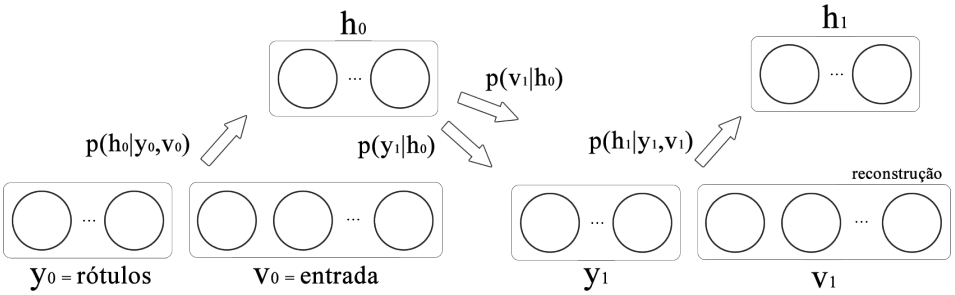


Figura 11 – Processo de reconstrução do algoritmo CD para DRBM

$$\mathbf{U^{t+1}} = \mathbf{U^t} + \Delta\mathbf{U^t} \rightarrow \Delta\mathbf{U^t} = \hat{\eta}(\mathbf{y_0 h_0^T} - \mathbf{y_1 h_1^T}) - \hat{\rho}\mathbf{U^t} + \hat{\alpha}\Delta\mathbf{U^{t-1}} \tag{2.36}$$

$$\mathbf{c^{t+1}} = \mathbf{c^t} + \Delta\mathbf{c^t} \rightarrow \Delta\mathbf{c^t} = \hat{\eta}(\mathbf{y_0} - \mathbf{y_1}) + \hat{\alpha}\Delta\mathbf{c^{t-1}} \tag{2.37}$$

Por fim, dado uma amostra $\mathbf{x}$ relacionada a um conjunto de rótulos $e = (1, 2, ..., k)$, a classificação da DRBM pode ser obtida por:

$$p(y^e|\mathbf{x}) = \frac{e^{-F(\mathbf{x}, y^e)}}{\sum_{y^* \in \{1,2,...,k\}} e^{-F(\mathbf{x}, \mathbf{y}^*)}} \tag{2.38}$$

O rótulo relacionado a amostra será aquele com maior probabilidade. A função $F(\mathbf{x}, y^e)$ é conhecida como "energia livre" e pode ser computada por:

$$F(\mathbf{x}, \mathbf{y}) = c_e + \sum_{j=1}^{n} \varphi(\sum_{i=1}^{m} x_i w_{ij} + u_{yj} + b_j) \tag{2.39}$$

onde $\varphi(z) = log(1 + exp(z))$ é conhecida como *softplus*. É importante salientar que o rótulo $\mathbf{y}$ é um vetor binário no qual apenas a posição e assume valor 1 quando $\mathbf{y}$ representa o rótulo e. No algoritmo 5 é apresentado o pseudocódigo para treinamento da DRBM.

Algoritmo 5: *treinamento DRBM*

inicialização:
    Preparar conjunto de dados de entrada e saída T de acordo com a definição 2.1;
    Informar número de neurônios para camada oculta $\mathbf{h}$;
    Inicializar os parâmetros $\hat{\eta}$, $\hat{\rho}$ e $\hat{\alpha}$;
    Inicializar $\boldsymbol{\theta}$ de maneira aleatória;
repita:
    Igualar a camada visível $\mathbf{v}_0$ aos dados de entrada;
    Igualar a camada de rótulos $\mathbf{y}_0$ aos rótulos relacionados às entradas;
    Estimar a camada oculta $\mathbf{h}_0$ por meio da equação 2.34;
    Estimar, a partir de $\mathbf{h}_0$, a camada visível $\mathbf{v}_1$ por meio da equação 2.27;
    Estimar, a partir de $\mathbf{h}_0$, a camada de rótulos $\mathbf{y}_1$ por meio da equação 2.35;
    Estimar, a partir de $\mathbf{v}_1$ e $\mathbf{y}_1$, a camada oculta $\mathbf{h}_1$ por meio da equação 2.34;
    Atualizar $\boldsymbol{\theta}$ por meio das equações 2.30, 2.31, 2.36 e 2.37;
Até número de iterações pré-determinado ou erro mínimo satisfeito
Obter as probabilidades dos rótulos de classificação por meio da equação 2.38;
retornar: probabilidades dos rótulos de classificação

3 Agregação de classificadores

Neste capítulo será discutido a agregação de elenco de classificadores. Primeiramente serão abordados a análise de componentes principais, a integral de Choquet e a medida *fuzzy*, conceitos básicos utilizados neste trabalho para que a agregação seja alcançada. Feito isso, são descritos o elenco de classificadores e a agregação propriamente dita.

3.1 Conceitos básicos

3.1.1 Análise de Componentes Principais (PCA)

A análise de componentes principais (do inglês: *principal component analysis*, PCA) é um método de análise de dados multivariados que pode ser usado para interpretar bases de dados multivariadas a fim de reduzir sua dimensionalidade (Jolliffe, 2002). Análise de dados multivariados é utilizada em diversas áreas de pesquisa como alternativa para tratar grandes quantidades de dados. Matematicamente, PCA é definido como uma transformação linear ortogonal que transforma um conjunto de variáveis correlatas em um outro conjunto menor de variáveis não correlacionadas denominadas componentes principais.

O primeiro passo para se aplicar PCA é o cálculo de uma matriz de variância-covariância. Dado um conjunto de dados, $X = (X_1, X_2, ..., X_n)$, com $X_i = (x_i^1, x_i^2, ..., x_i^d)$, a matriz de variância-covariância de $\mathbf{X}$, denominada $\mathbf{R}$, é obtida da seguinte forma:

$$\mathbf{R} = \begin{array}{c} \\ X_1 \\ X_2 \\ \vdots \\ X_n \end{array} \begin{array}{c} \begin{array}{cccc} X_1 & X_2 & ... & X_n \end{array} \\ \begin{bmatrix} var(X_1) & cov(X_1, X_2) & \dots & cov(X_1, X_n) \\ cov(X_2, X_1) & var(X_2) & \dots & cov(X_2, X_n) \\ \vdots & \vdots & \ddots & \vdots \\ cov(X_n, X_1) & cov(X_n, X_2) & \dots & var(X_n) \end{bmatrix} \end{array} \tag{3.1}$$

onde $var(\mathbf{Y}) = E[(\mathbf{Y} - E(\mathbf{Y}))^2]$ e $cov(\mathbf{Y}, \mathbf{Z}) = E[(\mathbf{Y} - E(\mathbf{Y}))(\mathbf{Z} - E(\mathbf{Z}))]$, sendo E o valor esperado. Em sequência, calcula-se os autovalores $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, ..., \lambda_d)$ e os autovetores $\mathbf{a} = (a_1, a_2, ..., a_d)$ relacionados a $\mathbf{R}$, da seguinte forma:

$$\det(\mathbf{R} - \boldsymbol{\lambda}\mathbf{I}) = 0 \tag{3.2}$$

onde $\mathbf{I}$ é a matriz identidade e os valores de $\boldsymbol{\lambda}$ que solucionam a equação 3.2 serão os autovalores associados a $\mathbf{R}$. Para obter os autovetores, soluciona-se o sistema linear obtido

através da seguinte equação:

$$(\lambda \mathbf{I} - \mathbf{R})\mathbf{a} = 0 \tag{3.3}$$

onde o vetor $\mathbf{a}$ que solucione a equação 3.3 para um determinado autovalor λ é o seu autovetor associado. Por fim, as componentes principais são calculadas por meio da combinação linear entre os autovetores e os dados de entrada $\mathbf{X}$ da seguinte forma:

$$CP_i = \mathbf{a}X_i \tag{3.4}$$

no algoritmo 6 é descrito o pseudocódigo para computar PCA.

Algoritmo 6: *Análise de componentes principais*

1 **inicialização:** Preparar conjunto de dados de entrada e saída X;
2 Calcular a matriz de variância-covariância $\mathbf{R}$ relacionada a $\mathbf{X}$ de acordo com a equação 3.1;
3 Calcular os autovalores $\boldsymbol{\lambda}$ associados a $\mathbf{R}$ de acordo com a equação 3.2;
4 Calcular os autovetores $\mathbf{a}$ associados a $\mathbf{R}$ de acordo com a equação 3.3;
5 **para** cada elemento $\mathbf{X}_i$ **faça**
6 Calcular a componente principal relacionada de acordo com a equação 3.4;
7 **fim para**
8 **retornar:** componentes principais

As componentes principais apresentam as seguintes propriedades (Jolliffe, 2002):

- A variância da *i-ésima* componente principal CP_i, é igual ao valor do autovalor λ_i relacionado a ela.
- A componente principal com maior autovalor representará a componente com maior variância.
- O total de variância das variáveis originais é igual ao somatório dos autovalores, que por sua vez, é igual ao total de variância das componentes principais.
- As componentes principais não têm correlação entre si.

Considerando as propriedades acima, calcula-se a contribuição C_i (variância individual) de cada componente de acordo:

$$C_i = \frac{\lambda_i}{\sum_{i=1}^{d} \lambda_i} \times 100 \tag{3.5}$$

Para realizar a redução de dimensionalidade, escolhe-se as componentes principais com as maiores contribuições. Todavia, neste trabalho a técnica de PCA não é utilizada com este propósito.

3.1.2 Medida *fuzzy*

A noção de medida generaliza os conceitos habituais de comprimento, de área, de volume etc (Barros; Bassanezi, 2006). O conceito de medida *fuzzy* foi proposto por Sugeno (1974), e é uma forma natural para realizar avaliações envoltas de incerteza e, principalmente, subjetividade, como por exemplo, avaliar preço de uma obra de arte, determinar valor de remuneração de um trabalho, avaliar valores de indenizações judiciais, avaliar preço de uma jóia, dentre outros. Sugeno percebeu que apesar da subjetividade e incerteza, envolvida na avaliação, existem características lógicas inseridas nas mesmas, por exemplo, se uma jóia A é maior e de melhor qualidade que uma jóia B, conclui-se que A deve possuir uma medida de avaliação maior do que B. Assim, é de se esperar que B seja um subconjunto de A, então $\mu(A) > \mu(B)$ sendo μ uma medida subjetiva ou não (Barros; Bassanezi, 2006).

Para apresentar os conceitos de medida *fuzzy*, primeiramente se faz necessário definir o que é espaço mensurável e medida. O espaço mensurável é descrito pelo par $(\Omega, \mathbb{A})$ que consiste de um universo Ω e de uma σ-algebra de conjunto mensuráveis de Ω (Pedrycz; Gomide, 1998).

Definição 3.1. (Pedrycz; Gomide, 1998) $\mathbb{A}$ é uma σ-algebra, com $\mathbb{A} \neq \varnothing$, se são satisfeitas:

1. $A \in \mathbb{A} \Rightarrow \bar{A} \in \mathbb{A}$, onde $\bar{A}$ é o complementar de A.
2. $A_k \in \mathbb{A}, k \in \mathbb{N} \Rightarrow \bigcup_k A_k \in \mathbb{A}$.

Definição 3.2. (Pedrycz; Gomide, 1998) Uma medida μ é uma função definida em um espaço mensurável $(\Omega, \mathbb{A})$, que não assume valores negativos e satisfaz os seguintes postulados:

1. $\mu(\varnothing) = 0$.
2. μ é σ-aditiva, ou seja, para uma família de pares de conjuntos disjuntos, $A_i \in \Omega$, $A_i \cap A_j = \varnothing (i = 1, 2, ..., i \neq j)$, a medida da união dos conjuntos A_i é dada por:

$$\mu(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} \mu(A_i) \tag{3.6}$$

3. Se A_i, (tal que $i = 1, 2, ...$) é uma sequência crescente de conjuntos mensuráveis então:

$$\lim_{i \to \infty} \mu(A_i) = \mu(\lim_{i \to \infty} A_i) \tag{3.7}$$

Uma das medidas σ-aditiva mais conhecidas, e utilizada comumente, é a medida de probabilidade, onde $P(\bigcup_{i=1}^{\infty}) = \sum_{i=1}^{\infty} P(A_i)$.

Embora as medidas σ-aditivas sejam comuns e úteis para um grande número de aplicações, a propriedade de σ-aditividade é muito forte e acaba excluindo, da teoria de medidas, conceitos que nos parecem passíveis de mensuração. Imagine que deseja-se "medir" a produtividade de um grupo de trabalhadores de uma determinada indústria. Seja $\mu(A)$ e $\mu(B)$ as medidas de produtividade de dois subconjuntos destes trabalhadores. Nesse caso, não é razoável que μ seja necessariamente aditiva, isto é, $\mu(A \cup B) = \mu(A) + \mu(B)$, ainda que $A \cap B = \varnothing$. Pode ocorrer $\mu(A \cup B) < \mu(A) + \mu(B)$ se houver incompatibilidade entre as operações entre os trabalhadores dos grupos A e B (Barros; Bassanezi, 2006). Assim como no exemplo anterior, em muitas áreas existem fenômenos que devem ser "medidos"mas não é razoável esperar aditividade na medida sugerida. Em resumo, se funções aditivas são admissíveis para medir incertezas, então deve haver um espaço para funções não aditivas também (Pedrycz; Gomide, 1998).

Com principal objetivo de flexibilizar a σ-aditividade Sugeno (1974) propôs a medida *fuzzy* que generaliza a medida σ-aditiva através da substituição da propriedade de σ-aditividade pela propriedade de monotonicidade (Pedrycz; Gomide, 1998). Os valores da medida *fuzzy* não representam apenas a importância de cada objeto, mas também a importância de todas as possíveis combinações entre eles (Krishnan; Kasim; Bakar, 2015).

Definição 3.3. (Sugeno, 1974) Uma função $\mu : \mathbb{A} \to [0, 1]$ é uma medida fuzzy se ela satisfaz:

1. $\mu(\varnothing) = 0$ e $\mu(\Omega) = 1$ (condição de fronteira).
2. $\mu(A) \leq \mu(B)$ se $A \subseteq B, \forall A, B \in \mathbb{A}$ (condição de monotonicidade).

A definição de medida *fuzzy* nem sempre é a mesma na literatura. Por exemplo, é comum exigir apenas que tal medida seja monótona e positiva. No entanto, entre as diversas definições, o que há em comum é a exigência da condição de monotonicidade e que a medida do conjunto vazio seja zero (Barros; Bassanezi, 2006). A tripla $(\Omega, \mathbb{A}, \mu)$ é conhecida como espaço de medida *fuzzy* e é importante observar que a medida *fuzzy* atua sobre espaços de conjuntos clássicos e não deve ser confundida com os próprios conjuntos *fuzzy*. A condição de fronteira interpreta que o conjunto vazio não tem importância, portanto $\mu(\varnothing) = 0$. Por outro lado, o conjunto completo, aquele com todos os elementos do conjunto, tem importância máxima, sendo assim $\mu(\Omega) = 1$ (Krishnan; Kasim; Bakar, 2015). Devido a condição de monotonocidade a medida *fuzzy* pode expressar três tipos de interação. Supondo A e B dois subconjuntos, tal que $\mu(A)$ e $\mu(B)$ são as medidas *fuzzy* de A e B, respectivamente, e a combinação dos conjuntos A e B denotada por $A \bigcup B$, então a interação entre esses conjuntos é descrita a seguir (Bonetti et al., 2012):

- Se $\mu(A \bigcup B) = \mu(A) + \mu(B)$, então A e B compartilham um efeito aditivo, em outras palavras, são independentes entre si.

- Se $\mu(A \bigcup B) < \mu(A) + \mu(B)$, então A e B compartilham um efeito sub-aditivo, ou seja, são redundantes entre si.
- Se $\mu(A \bigcup B) > \mu(A) + \mu(B)$, então A e B compartilham um efeito super-aditivo, expressam um efeito sinérgico.

3.1.3 Integral de Choquet

A integral de Choquet foi introduzida, inicialmente no contexto da teoria da capacidade por Choquet (1954), foi utilizada como integral com respeito a uma medida *fuzzy* por Höhle (1982) e redescoberta mais tarde por Murofushi e Sugeno (1989) e Murofushi e Sugeno (1991). De maneira geral, a integral de Choquet é um operador de agregação com respeito a qualquer medida *fuzzy* (Grabisch, 1995). A grande utilização dessa técnica se deve, principalmente, ao fato de seu modelo considerar não só a importância de cada atributo individual a ser agregado, mas também a interação entre eles (Cao, 2012). A definição da integral de Choquet é descrita da seguinte forma:

Definição 3.4. (Murofushi; Sugeno, 1989) Seja μ uma medida *fuzzy* em um conjunto finito X, então a integral de Choquet da função $f : X \to [0, \infty]$, com respeito a medida *fuzzy* μ é definida como:

$$\int_C f d\mu = \sum_{i=1}^{n} (f(x_i) - f(x_{i-1}))\mu(\hat{A}_i) \tag{3.8}$$

onde $\hat{A}_i \subset X$ para $i = 1, 2, ..., n$, $f(x_1) \leq f(x_2) \leq f(x_3) \leq ... \leq f(x_n)$ e $f(x_0) = 0$.

Expandindo a integral da definição 3.4, dado que $X = \{x_1, x_2, ..., x_n\}$ sejam n objetos com avaliações (do inglês: *ratings*) $f(x_1), f(x_2), ..., f(x_n)$ tal que $f(x_1) \leq f(x_2) \leq ... \leq f(x_n)$ e as medidas *fuzzy* desses objetos sejam $\mu(\hat{A}_1), \mu(\hat{A}_2), ..., \mu(\hat{A}_n)$, é obtido:

- Primeiro termo: $f(x_1).\mu(\hat{A}_1)$, onde $\mu(\hat{A}_1) = \mu(\{x_1, x_2, ..., x_n\})$
- Segundo termo: $(f(x_2) - f(x_1)).\mu(\hat{A}_2)$, onde $\mu(\hat{A}_2) = \mu(\{x_2, x_3, ..., x_n\})$
- Terceiro termo: $(f(x_3) - f(x_2)).\mu(\hat{A}_3)$, onde $\mu(\hat{A}_3) = \mu(\{x_3, x_4, ..., x_n\})$
- *n-ésimo* termo: $(f(x_n) - f(x_{n-1})).\mu(\hat{A}_n)$, onde $\mu(\hat{A}_n) = \mu(x_n)$

Por simplificação, considerando apenas três termos, o conceito da integral de Choquet é ilustrado na figura 12, na qual cada termo expandido anteriormente é a área de um retângulo. Esse conceito é válido para os n termos da integral.

Como já mencionado anteriormente, a integral de Choquet pode ser implementada com qualquer medida *fuzzy* e um dos grandes entraves do método é justamente determinar

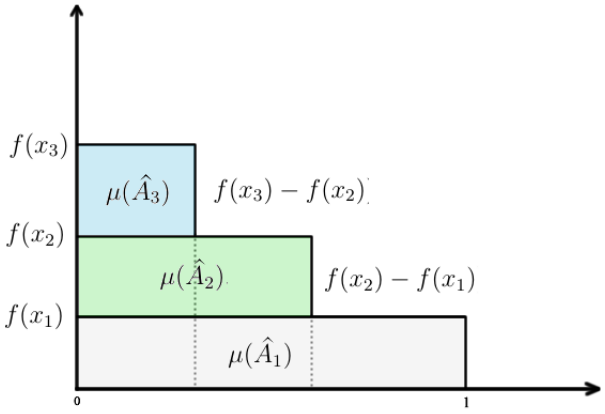


Figura 12 – Ilustração da integral de Choquet

a medida *fuzzy* dos objetos do conjunto a ser agregado. Assim como em uma agregação via média ponderada é necessário determinar o peso de cada objeto, para se obter a agregação via integral de Choquet é necessário determinar as medidas *fuzzy*. No anexo B o cálculo da integral de Choquet é exemplificado.

3.2 Elenco de classificadores

A ideia da utilização de sistemas baseados em elencos vem sendo investigada há décadas. Os trabalhos de Dasarathy e Sheela (1979) e Hansen e Salamon (1990) foram precursores em introduzir a ideia de elencos e utilizar redes neurais para tal tarefa, respectivamente. Desde então diversos trabalhos utilizando elencos de classificadores foram desenvolvidos com os mais diversos nomes, tais como: composição de classificadores, mistura de especialistas, combinação de classificadores, comitê de classificadores, dentre outros (Polikar, 2006). Um elenco de classificadores simples pode ser melhor do que um único classificador complexo, pois um classificador sozinho quase sempre não é capaz de identificar todas as correlações entre as características de entrada e de saída de uma base de dados (Chen; He; Li, 2010). Como diversos algoritmos funcionam realizando buscas que podem levar a um ótimo local, e consequentemente prejudicar o resultado final, computacionalmente e estatisticamente a agregação de um conjunto de classificadores reduz o risco de escolher uma hipótese ruim. Todavia, nada impede de que um único classificador seja melhor do que todo o elenco em determinada base de dados (Polikar, 2006).

De maneira geral, existem duas abordagens utilizadas para combinar classificadores de um elenco (Polikar, 2006):

- *Seleção de classificadores:* cada classificador do elenco é treinado para se tornar um especialista em determinada região de características da base de dados.

- *Agregação de classificadores:* todos os classificadores do elenco são treinados em toda região de características da base de dados.

Métodos baseados em combinação de classificadores se tornaram ferramentas padrões para melhorar a qualidade da classificação de dados (Štefka; Holeňa, 2015). Em vez de utilizar apenas um classificador, um elenco de classificadores é criado e cada avaliação individual é considerada para formar a classificação final. Neste trabalho a combinação dos classificadores do elenco é realizada via metodologia de agregação como ilustrado na figura 13.

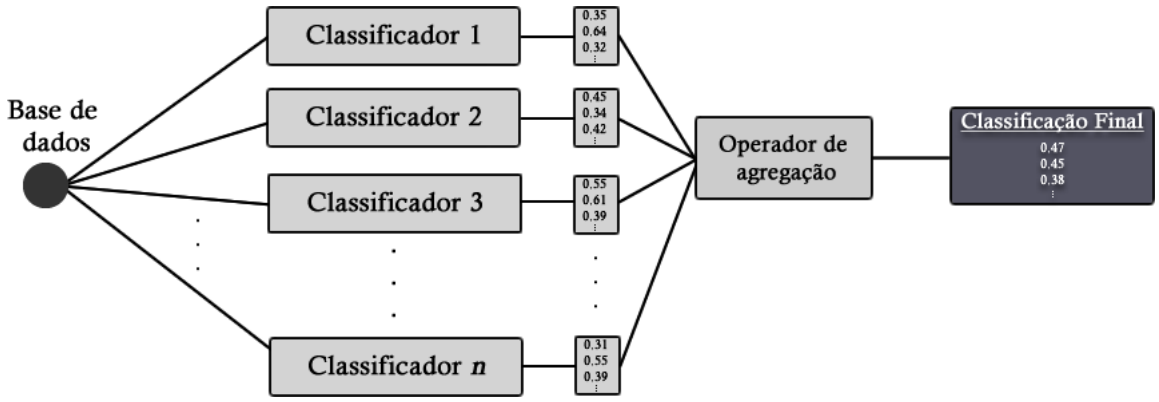


Figura 13 – Elenco de classificadores combinados via agregação

3.2.1 Metodologia de agregação

Como ilustrado na figura 13 para metodologia de agregação ser aplicada se faz necessário, obviamente, um operador de agregação, que neste caso será a integral de Choquet com respeito à medida *fuzzy*. Para isso, o primeiro passo é organizar a saída de cada classificador do elenco na seguinte matriz de decisão:

$$\mathbf{M}_k = \begin{bmatrix} c_{11}^k & c_{12}^k & \cdots & c_{1j}^k \\ c_{21}^k & c_{22}^k & \cdots & c_{2j}^k \\ \vdots & \vdots & \ddots & \vdots \\ c_{i1}^k & c_{i2}^k & \cdots & c_{ij}^k \end{bmatrix} \tag{3.9}$$

onde cada coluna representa um classificador do elenco ($j = 1, 2, ..., J$), sendo J o número de classificadores, cada linha representa um rótulo da base de dados ($i = 1, 2, ..., I$), sendo I o número de rótulos, k é uma amostra da base de dados e c_{ij}^k é a hipótese de classificação, c^k é conhecida como função suporte referente à amostra k da base de dados. Cada amostra

da base de dados possuirá sua própria matriz de decisão $\mathbf{M}_k$. Essa matriz considera cada classificação individual do elenco e será nela que o operador de agregação atuará.

Uma vez montada a matriz de decisão $\mathbf{M}_k$, o próximo passo é realizar a agregação das linhas da mesma, ou seja, agregação das hipóteses de classificação para cada rótulo, a fim de se obter o resultado final em um vetor coluna denominado $\mathbf{C}_f$, como ilustrado na figura 14, no qual a integral $\int_C$ é a integral de Choquet da definição 3.4. O rótulo da classificação final do elenco será aquele que possui o maior valor em $\mathbf{C}_f$, ou seja, calcula-se o *arg max*($\mathbf{C}_f$). Nesta etapa se faz necessário estimar as medidas *fuzzy* entre os classificadores do elenco.

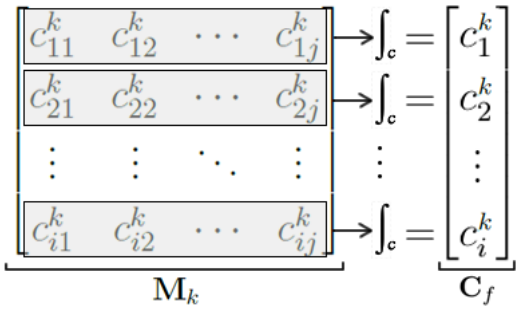


Figura 14 – Agregando a matriz de decisão dos classificadores

3.2.2 Metodologia para cálculo da estimativa da medida *fuzzy*

Estimar os valores da medida *fuzzy* de um conjunto na prática é um tópico bastante revisitado. Partindo deste princípio, para atacar este problema, Rowley, Geschke e Lenzen (2015) propuseram uma metodologia não supervisionada que visa escapar da "maldição da dimensionalidade" encontrada por Kojadinovic (2004). Nesta abordagem o conceito de importância de um determinado objeto é substituído por um conceito alternativo de quão única uma informação é mensurada por um objeto do conjunto. Rowley, Geschke e Lenzen (2015) aplicaram essa nova metodologia para problemas de tomada de decisão, na qual o conceito de informação única é extraído de uma matriz de tomada de decisão montada a partir de avaliações baseadas nas alternativas e critérios do problema. Neste trabalho, essa mesma ideia será utilizada para extrair a medida *fuzzy* do elenco de classificadores a partir da matriz de decisão $\mathbf{M}_k$ que, como mostrado na seção anterior, é montada a partir das hipóteses de classificação e não de avaliações de um processo de tomada de decisão, como no método original (Krohling, 2015).

3.2.2.1 Número equivalente de objetos não interativos

A partir de uma matriz de variância-covariância $\mathbf{R}$, definida pela equação 3.1, é possível obter um número equivalente de objetos não interativos, denotado por J^*, de

um determinado conjunto (Rowley; Geschke; Lenzen, 2015). Tomando como exemplo um conjunto com cinco objetos, se todos eles forem independentes entre si, é obtida uma matriz $\mathbf{R}_1$, que se normalizada, $\mathbf{R}_1 = I_{5\times5}$, onde $\mathbf{I}$ é a matriz identidade. Neste caso, tem-se $J^* = 5$. Assumindo agora que os objetos 5 e 4 são quase completamente correlatos tem-se a matriz $\mathbf{R}_2$ dada por:

$$\mathbf{R}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0.99 \\ 0 & 0 & 0 & 0.99 & 1 \end{bmatrix} \tag{3.10}$$

neste caso, é esperado $J^* \approx 4$, pois os objetos 4 e 5 podem ser considerados como o mesmo. Em um caso extremo, em que todos os objetos são quase completamente correlacionados entre si, tem-se a matriz $\mathbf{R}_3$:

$$\mathbf{R}_3 = \begin{bmatrix} 1 & 0.99 & 0.99 & 0.99 & 0.99 \\ 0.99 & 1 & 0.99 & 0.99 & 0.99 \\ 0.99 & 0.99 & 1 & 0.99 & 0.99 \\ 0.99 & 0.99 & 0.99 & 1 & 0.99 \\ 0.99 & 0.99 & 0.99 & 0.99 & 1 \end{bmatrix} \tag{3.11}$$

nesta situação, espera-se $J^* \approx 1$, ou seja, apenas um objeto pode descrever os demais.

Utilizando o conceito de critérios não equivalentes e considerando as demais definições deste capítulo, uma medida *fuzzy* $\mu(A_i)$, para qualquer subconjunto $A_i \in \Omega$, pode ser definido como (Rowley; Geschke; Lenzen, 2015):

$$\mu(A_i) = \frac{J^*(A_i)}{J^*(\Omega)} \tag{3.12}$$

Uma maneira para calcular J^* utilizando PCA é descrito na próxima seção.

3.2.2.2 Determinação do número de objetos não iterativos utilizando PCA

Os princípios da análise de componentes principais, definidos na seção 3.1.1, podem ser utilizados para calcular o número de objetos não interativos equivalente, seja para todo conjunto Ω ou um subconjunto A_i. A metodologia para determinar J^* aplica os dois primeiros passos do PCA utilizando a matriz de decisão $\mathbf{M}_k$ da equação 3.9. O procedimento completo é descrito na forma de passos a seguir (Rowley; Geschke; Lenzen, 2015):

Passo 1: a partir de $\mathbf{M}_k$ determinar uma matriz $\mathbf{P}_i$ para cada subconjunto $A_i \in \Omega$. A matriz $\mathbf{P_i}$ será formada a partir das colunas de $\mathbf{M}_k$, por exemplo, se um subconjunto A_1 possui os classificadores c_1 e c_2 do elenco, utiliza-se as colunas de $\mathbf{M}_k$ referentes a eles.

Passo 2: de acordo com a equação 3.1, determinar a matriz de variância-covariância $\mathbf{R}_i$ para cada uma das matrizes $\mathbf{P}_i$.

Passo 3: calcular os autovalores $\boldsymbol{\lambda^i} = (\lambda_1^i, \lambda_2^i, ..., \lambda_d^i)$ relacionados a cada uma das matrizes $\mathbf{R}_i$.

Passo 4: calcular o valor de $J^*(A_i)$ de acordo com a seguinte equação:

$$J^*(Ai) = \sum_{z=1}^{d} \{\lambda_z^i | \lambda_z^i < 1\} + |\{\lambda_z^i | \lambda_z^i \geq 1\}| \tag{3.13}$$

onde o operador $|\ |$ denota o número de elementos do conjunto finito. No anexo B é descrito um exemplo numérico ilustrativo desta metodologia.

A tabela 1 descreve os valores dos autovalores, variância e variância acumulada quando PCA é aplicado para as matrizes $\mathbf{R}_1$, $\mathbf{R}_2$ e $\mathbf{R}_3$ da seção 3.2.2.1. Baseado nisso é possível calcular os valores de J^* de acordo com a equação 3.13. Em todos os casos os valores de J^* vão ao encontro dos valores esperados para cada matriz (Rowley; Geschke; Lenzen, 2015): 5 para $\mathbf{R}_1$, o esperado é 5; 4.01 para $\mathbf{R}_2$, o esperado é 4; e 1.04 para $\mathbf{R}_3$, o esperado é 1.

Matriz $\mathbf{R}_1$				Matriz $\mathbf{R}_2$				Matriz $\mathbf{R}_3$			
CP	$\boldsymbol{\lambda}$	V	V_a	CP	$\boldsymbol{\lambda}$	V	V_a	CP	$\boldsymbol{\lambda}$	V	V_a
1	1	20%	20%	1	1.99	39,8%	39,8%	1	4,96	99.2%	99.2%
2	1	20%	40%	2	1	20%	59,8%	2	0.01	0.2%	99.4%
3	1	20%	60%	3	1	20%	79,8%	3	0.01	0.2%	99.6%
4	1	20%	80%	4	1	20%	99,8%	4	0.01	0.2%	99.8%
5	1	20%	100%	5	0.01	0.02%	100%	5	0.01	0.2%	100%

Tabela 1 – PCA para as matrizes da seção 3.2.2.1. CP são as componentes principais, V a contribuição de variância de cada componente e V_a a variância acumulada

Por fim, estimada a medida *fuzzy* a integral de Choquet é utilizada como operador de agregação do elenco de classificadores. A metodologia completa de agregação é descrita no algoritmo 7.

Algoritmo 7: *metodologia de agregação*

inicialização:
    Definir os classificadores do elenco;
    Obter o conjunto de saída de cada classificador do elenco;
para cada amostra do conjunto de dados **faça**
    Montar a matriz $\mathbf{M}_k$ de acordo com a equação 3.9;
    Determinar as matrizes $\mathbf{P}_i$ relacionadas aos subconjuntos A_i;
    Determinar as matrizes de variância-covariância $\mathbf{R}_i$ relacionadas a P_i;
    Determinar os autovalores $\boldsymbol{\lambda}^i$ relacionados a $\mathbf{R}_i$;
    Calcular o valor de $J^*(A_i)$ de acordo com a equação 3.13;
    Estimar a medida *fuzzy* de acordo com a equação 3.12;
    Obter $\mathbf{C}_f$ agregando as linhas de $\mathbf{M}_k$ utilizando a integral de Choquet;
    Obter o rótulo de classificação através de $arg\ max(\mathbf{C}_f)$;
fim para
retornar: conjunto de rótulos da classificação agregado

4 Resultados experimentais

Neste capítulo são apresentados os resultados experimentais para a agregação de um elenco de classificadores utilizando os classificadores neurais, descritos no capítulo 2, além do classificador convencional KNN, utilizado para comparação. Primeiramente, o elenco é aplicado a diversos *benchmarks* do repositório UCI (Lichman, 2013) em experimentos divididos em duas partes: a primeira parte utilizando pequenas bases de dados e a segunda parte utilizando grandes bases de dados, tanto em número de amostras, quanto em número de atributos de entradas. Em seguida, é realizado um estudo de caso em uma nova base de dados desenvolvida para este trabalho. Além disso, os resultados do algoritmo de agregação proposto neste trabalho são comparados com duas metodologias clássicas de agregação, são elas (Krawczyk; Woźniak, 2016):

- *Média das hipóteses:* cada hipótese de classificação da matriz $\mathbf{M}_k$ é agregada por meio de uma média aritmética simples gerando o vetor $\mathbf{C_f}$, na qual a classificação final é obtida fazendo $arg\ max(\mathbf{C_f})$.
- *Voto majoritário:* cada classificador atua em uma dada amostra resultando em um determinado rótulo. O rótulo mais encontrado entre os classificadores, ou seja, o mais "votado", será o resultado da classificação final.

Para comparar todos os algoritmos utilizados foi aplicado o teste não paramétrico de Friedman seguido do teste *pos hoc* de Wilcoxon com intuito de identificar possíveis diferenças estatísticas entre os métodos. O teste de Friedman irá verificar a hipótese nula de que os resultados possuam a mesma mediana. Para isso é escolhido um $p_{valor} = 0.05$, no qual, se o teste retornar um valor menor do que este há evidência estátística que ao menos um dos algoritmos é significativamente diferente (Derrac et al., 2011). Caso o teste de Friedman indique a diferença estatística é necessário um teste *pos hoc*, neste caso o teste de Wilcoxon, que é realizado com os algoritmos par a par. Este teste irá verificar a hipótese nula de que os algoritmos sejam semelhantes. Para isso novamente é escolhido $p_{valor} = 0.05$, que de forma análoga ao teste de Friedman, se o teste retornar um valor menor do que o escolhido existe evidência estatística que o par analisado é significativamente diferente (Derrac et al., 2011). Além disso, para complementar a comparação é utilizado o algoritmo A-TOPSIS (Krohling; Pacheco, 2015), que retorna um ranking dos melhores algoritmos baseado no desempenho dos mesmos. Para mais informações do A-TOPSIS consulte o anexo C.

Por fim, todos os procedimentos foram realizados em uma máquina com processador *intel(R) core(TM) i7-4790* CPU @ 3.60 GHz, com 16 Gb de memória *RAM* e utilizando o

ambiente de desenvolvimento MATLAB 2015. Neste ambiente, a programação é realizada através de funções e *scripts*, codificados em linguagem de programação própria. Esta ferramenta, apesar de simples, é bastante poderosa, devido à facilidade de codificação e agilidade no processo de prototipação dos algoritmos, permitindo que eles sejam testados de maneira rápida antes de serem codificados no ambiente real onde serão utilizados. Além disso, foram utilizadas a *toolbox* de rede neural, nativa do MATLAB, e a *toolbox DeeBnet* (Keyvanrad; Homayounpour, 2014), que implementa algoritmos relacionados à máquina de Boltzmann. Todos os algoritmos implementados estão disponíveis no endereço virtual http://goo.gl/wEU4c0 como código aberto para fins acadêmicos.

4.1 Resultados para pequenas bases de dados

Nesta seção serão apresentados os resultados experimentais para pequenas bases de dados. Todas as bases utilizadas aqui são descritas na tabela 2 e, como pode ser observado, cada uma delas possui configurações diferentes em relação a número de amostras, atributos e classes. As características de cada base também possuem suas peculiaridades, como por exemplo: a base *iris*, que possui dados bem comportados e distribuídos; a base *spam*, que possui 57 atributos de entradas; e por fim a base *statlog* que é mal distribuída, 80% dos dados estão no rótulo de classe 1, dentre os 7 possíveis, e o desafio é conseguir acurácia acima de 99% de acerto.

Base de Dados	**# de amostras**	**# de atributos**	**# de rótulos**
Cancer	699	9	2
Credit Australia	690	14	2
Diabetic	1151	19	2
Iris	150	4	3
Spam	4601	57	2
Statlog	14500	9	7
Wine	178	13	3

Tabela 2 – Bases de dados utilizadas na primeira parte do experimento

A configuração dos parâmetros dos classificadores foram obtidas de maneira empírica e são descritas a seguir:

- Rede neural *feedforward* (NN): a rede utilizada possui duas camadas ocultas com 25 neurônios cada, utiliza *mini-bacth* de 10 partições para bases de dados com mais de 1000 amostras, o algoritmo de treinamento executa no máximo 1000 iterações, e os valores de γ são: $\gamma_i = 0,001$, $\gamma_a = 0.1$ e $\gamma_d = 10$ (Hagan; Menhaj, 1994).
- Máquina de aprendizado extremo (ELM): a ELM utilizada possui 100 neurônios na camada oculta para base de dados *spam* e 50 neurônios para as demais bases.

- Máquina de Boltzmann restrita discriminativa (DRBM): a DRBM utilizada possui 100 neurônios na camada oculta, utiliza *mini-bacth* de 10 partições para bases de dados com mais de 1000 amostras, o algoritmo executa no máximo 300 iterações e os parâmetros $\hat{\eta}, \hat{\rho}$ e $\hat{\alpha}$ são os mesmos propostos por Hinton (2010) descritos na seção 2.3.1.

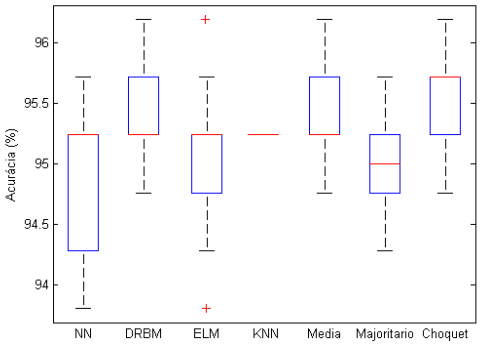
- K vizinhos mais próximos (KNN): como descrito no anexo A, o valor de k é igual a $\sqrt{|T|}$, sendo $|T|$ o número de amostras do conjunto de treinamento (Jirina; Jirina, 2011).

Para executar os experimentos, cada base de dados da tabela 2 foi embaralhada e dividida em 70% para treinamento e 30% para testes. Cada algoritmo, com exceção do KNN que não é estocástico, foi executado 30 vezes para cálculos estatísticos do resultado. Neste ponto é importante observar que o embaralhamento deve ser o mesmo para execução do elenco de classificadores. Caso contrário, não há garantia que as amostras a serem agregadas sejam as mesmas. A métrica de desempenho escolhida é a acurácia de classificação, ou seja, a porcentagem de acerto dos rótulos. O desempenho de cada algoritmo, em termos de média e desvio padrão, é descrito na tabela 3 e os *boxplots* dos desempenhos são exibidos na figura 15. Nesta parte do experimento não serão apresentados os tempos computacionais de execução dos algoritmos pois os mesmos são, em sua maior parte, muito pequenos.

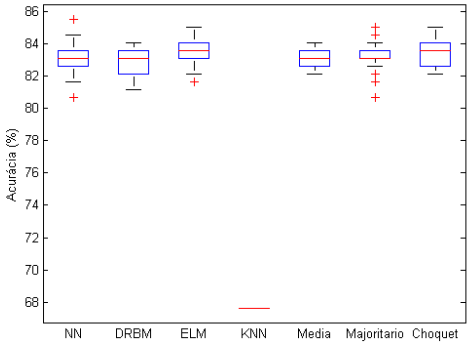
Acurácia em %							
Bases	**Classificadores**				**Agregadores**		
	NN	**DRBM**	**ELM**	**KNN**	**Média**	**Major.**	**Choquet**
Cancer	$94,87 \pm 0,62$	$95,39 \pm 0,28^*$	$95,07 \pm 0,50$	$95,23 \pm 0$	$94,93 \pm 0,42$	$95,38 \pm 0,35$	$\mathbf{95,57 \pm 0,35}$
CredAus	$83,05 \pm 1,07$	$82,88 \pm 0,80$	$83,38 \pm 0,81^*$	$67,63 \pm 0$	$83,14 \pm 0,65$	$83,15 \pm 0,98$	$\mathbf{83,53 \pm 0,85}$
Diabetic	$71,02 \pm 2,31$	$60,48 \pm 1,69$	$\mathbf{72,37 \pm 1,09}^*$	$61,74 \pm 0$	$69,84 \pm 3,03$	$70,05 \pm 0,97$	$71,42 \pm 0,97$
Iris	$95.62 \pm 1,36^*$	$89,41 \pm 0,95$	$94,81 \pm 1,96$	$95,55 \pm 0$	$95,11 \pm 1,47$	$95,62 \pm 1,42$	$\mathbf{95,70 \pm 1,29}$
Spam	$93,61 \pm 0,80^*$	$90,80 \pm 0,56$	$89,43 \pm 0,57$	$72,75 \pm 0$	$93,40 \pm 0,61$	$92,02 \pm 0,37$	$\mathbf{93,78 \pm 0,65}$
Statlog	$\mathbf{99,59 \pm 0,06}^*$	$92,08 \pm 0,04$	$98,30 \pm 0,08$	$97,73 \pm 0$	$98,72 \pm 0,01$	$99,32 \pm 0,01$	$98,73 \pm 0,01$
Wine	$95,91 \pm 2,16^*$	$95,15 \pm 1,28$	$91,13 \pm 2,80$	$67,92 \pm 0$	$96,16 \pm 1,44$	$95,22 \pm 1,76$	$\mathbf{97,18 \pm 1,34}$

Tabela 3 – Desempenho dos classificadores para cada uma das bases de dados da primeira parte do experimento. O asterisco indica o melhor classificador do elenco e o negrito o melhor desempenho global, considerando a acurácia média

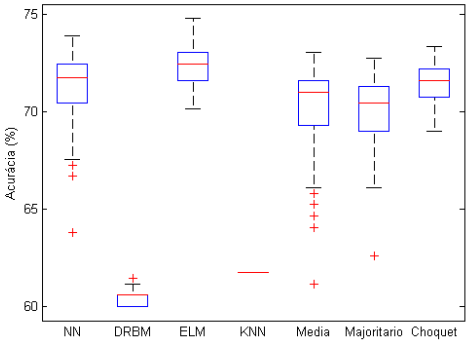
Analisando resultados da tabela 3 e os *boxplots* da figura 15 é possível notar que a agregação via integral de Choquet obteve o melhor resultado para a maioria das bases de dados utilizadas. Apesar da melhora na acurácia de classificação não ser extremamente significativa, a principal vantagem da abordagem é assegurar uma acurácia final quase sempre superior ao melhor classificador do elenco. Todavia, nos casos em que dois dos quatro classificadores obtiveram desempenhos bem abaixo do melhor, como nas bases *diabetic* e *statlog*, todas as metodologias de agregação foram afetadas fazendo com que a



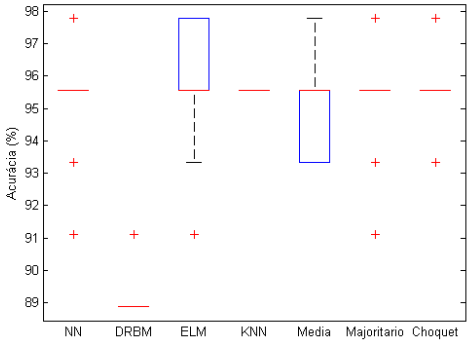
(a) *Cancer*



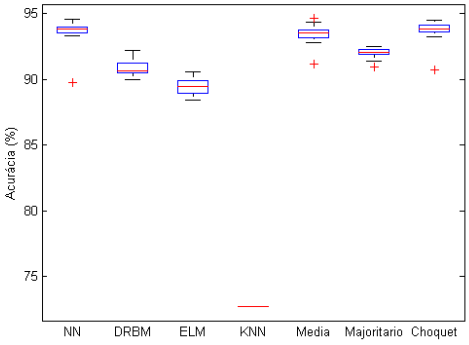
(b) *Credit Australia*



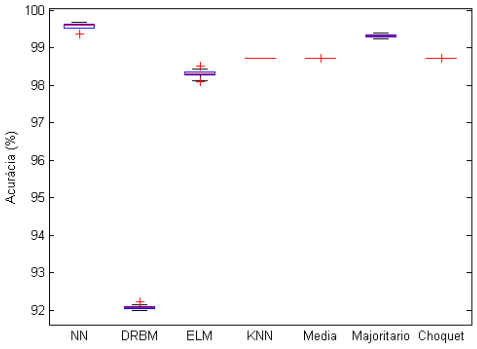
(c) *Diabetic*



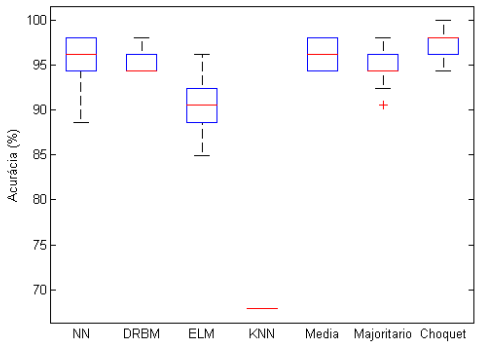
(d) *Iris*



(e) *Spam*



(f) *Statlog*



(g) *Wine*

Figura 15 – *Boxplots* do desempenho dos classificadores para cada uma das bases de dados da primeira parte do experimento

acurácia dos classificadores individuais fossem as melhores, neste caso NN e ELM para as respectivas bases. Por outro lado, em outras bases de dados como *credAus*, *spam* e *wine*, mesmo com o KNN tendo um desempenho bem ruim, ainda assim a agregação via Choquet foi capaz de melhorar o resultado da classificação final. As agregações por meio da média das hipóteses e do voto majoritário também obtiveram bons resultados quando comparados com os classificadores do elenco e, levando em consideração a simplicidade destes dois métodos, os resultados mostram que elas também são efetivas. Por fim, considerando apenas o elenco, é possível observar que nenhum dos classificadores individuais obteve o melhor desempenho em todas as bases de dados; a NN foi melhor em quatro bases, ELM em duas, DRBM em uma e o KNN em nenhuma. Com isso os resultados mostram que a metodologia de agregação do elenco garante uma classificação mais confiável, ao passo que não existe o classificador perfeito para todos os problemas.

O teste de Friedman foi aplicado para este experimento e obteve $p = 0.006469$ indicando que ao menos um algoritmo é significativamente diferente. Sendo assim, na sequência foi aplicado o teste de Wilcoxon e o resultado do mesmo é descrito na tabela 4. Analisando especificamente a integral de Choquet, os valores obtidos pelo teste indicam diferença estatística em relação ao KNN, a DRBM e a média. Nos demais pares contendo a integral, o teste não foi capaz de identificar diferenças entre os algoritmos. Como os resultados da tabela 3 não apontam grandes ganhos em termos de acurácia, o resultado do teste é coerente.

Comparação	p
DRBM - Choquet	0.015625
KNN - Choquet	0.031250
AVG - Choquet	0.031250
FNN - DRBM	0.046875
AVG - DRBM	0.031250
KNN - NN	0.046875
MV - KNN	0.046875

Tabela 4 – Resultados par a par do teste de Wilcoxon com valores de p abaixo de 0.05 para as bases de dados da primeira parte do experimento

Para complementar o teste estatístico, foi utilizado o algoritmo A-TOPSIS para gerar um ranking dos algoritmos. Para utilizar o A-TOPSIS é necessário ponderar a média e o desvio padrão do desempenho dos algoritmos. A tabela 5 descreve a variação do ranking dando peso igual a ambos até desconsiderar o desvio padrão. A variação do ranking mostra que ao longo da alteração dos pesos a agregação via integral de Choquet se mantém como melhor alternativa. Quando a média do desempenho dos algoritmos passa a ser privilegiada, a NN passa a subir no ranking seguida pelo ELM. Mesmo o KNN sendo beneficiado pelo fato de não possuir desvio padrão, o algoritmo só não é o último quando a ponderação é igual, o que já era esperado considerando o desempenho do mesmo.

Variação peso [media, desvio]	**Ranking**
[0.5, 0.5]	Choquet $\prec$ Maj. $\prec$ Media $\prec$ NN $\prec$ ELM $\prec$ KNN $\prec$ DRBM
[0.6, 0.4]	Choquet $\prec$ Maj. $\prec$ Media $\prec$ NN $\prec$ ELM $\prec$ DRBM $\prec$ KNN
[0.7, 0.3]	Choquet $\prec$ Maj. $\prec$ Media $\prec$ NN $\prec$ ELM $\prec$ DRBM $\prec$ KNN
[0.8, 0.2]	Choquet $\prec$ Maj. $\prec$ NN $\prec$ Media $\prec$ ELM $\prec$ DRBM $\prec$ KNN
[0.9, 0.1]	Choquet $\prec$ NN $\prec$ Maj. $\prec$ Media $\prec$ ELM $\prec$ DRBM $\prec$ KNN
[1.0, 0.0]	Choquet $\prec$ NN $\prec$ Maj. $\prec$ Media $\prec$ ELM $\prec$ DRBM $\prec$ KNN

Tabela 5 – Ranking dos classificadores obtido pelo A-TOPSIS para primeira parte do experimento variando o peso da média e desvio padrão do desempenho

Uma boa configuração de pesos é 75% para média dos desempenhos e 25% para o desvio padrão. Dessa forma o ranking leva em consideração o desvio e não privilegia o KNN. Na figura 16 é mostrado gráfico de barras do ranking para esta configuração.

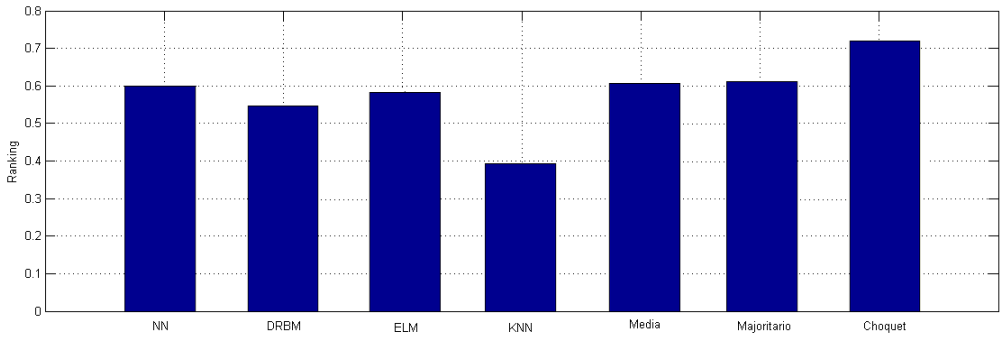


Figura 16 – Ranking dos classificadores obtido pelo A-TOPSIS para bases de dados da primeira parte do experimento

4.2 Resultados para grandes bases de dados

Nesta seção serão apresentados os resultados experimentais para grandes bases de dados seguindo os mesmos moldes da seção anterior. As bases utilizadas aqui são descritas na tabela 6 e podem ser divididas em dois grupos: aquelas com muitas amostras, *covtype*, *higgs* e *susy*, sendo as duas últimas extratos de um milhão das bases originais, e aquelas com número elevado de atributos de entradas, *DNA* e *isolet*.

A configuração dos parâmetros dos classificadores foram obtidas de maneira empírica e são descritas a seguir:

- Rede neural *feedforward* (NN): a rede utilizada possui duas camadas ocultas com 50 neurônios cada, utiliza *mini-bacth* de 100 partições, o algoritmo de treinamento executa no máximo 1000 iterações, e os valores de γ são: $\gamma_i = 0,001$, $\gamma_a = 0.1$ e $\gamma_d = 10$ (Hagan; Menhaj, 1994).

Base de Dados	# de amostras	# de atributos	# de classes
DNA	3186	180	3
Covtype	581012	54	7
Higgs	1000000	28	2
Isolet	7797	617	26
Susy	1000000	18	2

Tabela 6 – Bases de dados utilizadas na segunda parte do experimento

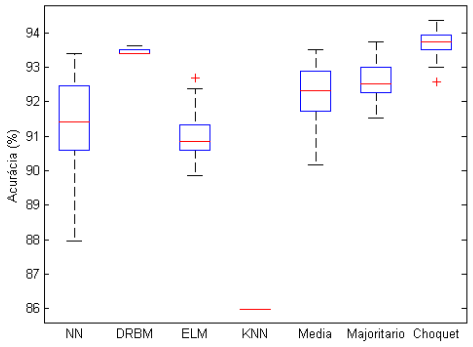
- Máquina de aprendizado extremo (ELM): a ELM utilizada possui 700 neurônios na camada oculta para base *isolet* e 250 para as demais.
- Máquina de Boltzmann restrita discriminativa (DRBM): a DRBM utilizada possui 700 neurônios na camada oculta para base isolet e 300 para as demais, utiliza também *mini-bacth* de 100 partições, executa no máximo 300 iterações e os parâmetros $\hat{\eta}, \hat{\rho}$ e $\hat{\alpha}$ são os mesmos propostos por Hinton (2010) descritos na seção 2.3.1.
- K vizinhos mais próximos (KNN): como descrito no anexo A, o valor de k é igual a $\sqrt{|T|}$, sendo $|T|$ o número de amostras do conjunto de treinamento (Jirina; Jirina, 2011).

Assim como na seção anterior, com exceção do KNN, cada classificador foi executado 30 vezes para cada *benchmark*. Além disso, as bases de dados da tabela 6 foram embaralhadas e divididas em 70% para treinamento e 30% para testes e a métrica de desempenho escolhida continua sendo a acurácia de classificação. O desempenho de cada algoritmo, em termos de média e desvio padrão é descrito na tabela 7 e os *boxplots* dos desempenhos são exibidos na figura 17.

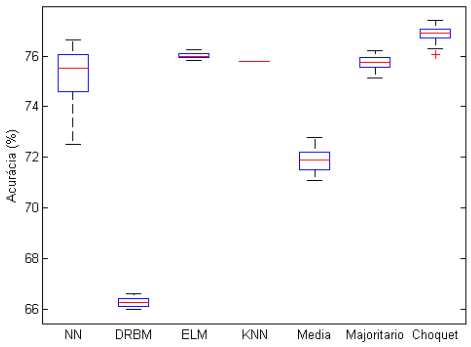
Acurácia em %							
Bases	**Classificadores**				**Agregadores**		
	NN	**DRBM**	**ELM**	**KNN**	**Média**	**Major.**	**Choquet**
DNA	$91,36 \pm 1,34$	$93,45 \pm 0,07^*$	$90,59 \pm 0,75$	$85,98 \pm 0$	$92.18 \pm 0,92$	$92.64 \pm 0,57$	$\mathbf{93.69 \pm 0.38}$
Covtype	$75,22 \pm 1,09$	$66,25 \pm 0,17$	$76,01 \pm 0,11^*$	$75,81 \pm 0$	$71,84 \pm 0,42$	$75,75 \pm 0,24$	$\mathbf{76,85 \pm 0,29}$
Higgs	$63,21 \pm 1,19$	$63,40 \pm 0,30$	$63,99 \pm 0,09^*$	$59,84 \pm 0$	$64,01 \pm 0,96$	$63,75 \pm 0,64$	$\mathbf{64,70 \pm 0,72}$
Isolet	$89,41 \pm 1,7$	$93,74 \pm 0,16^*$	$86,81 \pm 0,60$	$88,24 \pm 0$	$93,73 \pm 0,26$	$\mathbf{93.93 \pm 0,31}$	$93.75 \pm 0,16$
Susy	$78,14 \pm 0,65$	$76,39 \pm 0,32$	$\mathbf{79,39 \pm 0,29}$	$70,88 \pm 0$	$78,38 \pm 0,59$	$78,14 \pm 0,54$	$78,58 \pm 0,65$

Tabela 7 – Desempenho dos classificadores para cada uma das bases de dados da segunda parte do experimento. O asterisco indica o melhor classificador do elenco e o negrito o melhor desempenho global, considerando a acurácia média

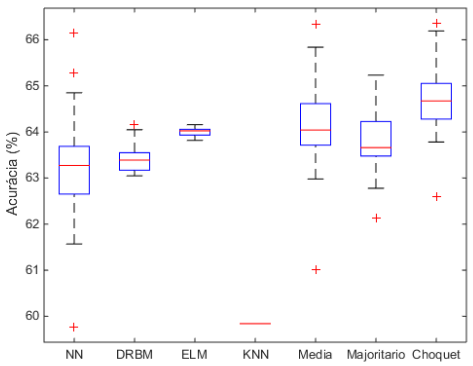
Observando a tabela 7 e os *boxplots* da figura 17 verifica-se que, assim como para pequenas bases, o desempenho da agregação por meio da integral de Choquet se sobressaiu comparado às demais metodologias de agregação e aos classificadores do elenco. Nas bases



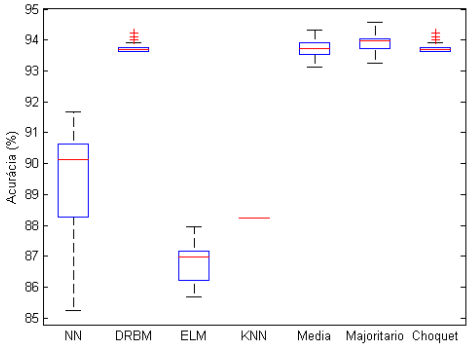
(a) *DNA*



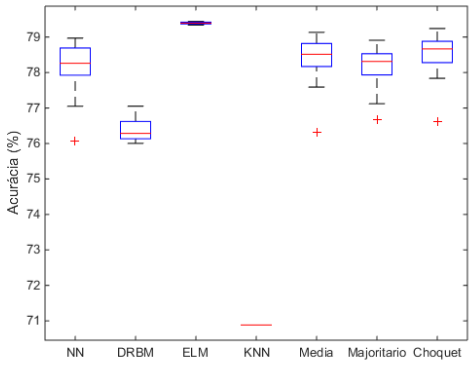
(b) *Covtype*



(c) *Higgs*



(d) *Isolet*



(e) *Susy*

Figura 17 – *Boxplots* do desempenho dos classificadores para cada uma das bases de dados da segunda parte do experimento

de dados *DNA*, *covtype* e *higgs* a metodologia de agregação via integral de Choquet obteve o melhor desempenho. Já nas bases *isolet* e *susy* os melhores desempenhos foram obtidos pela metodologia de voto majoritário e pela ELM, respectivamente, seguidas pela integral de Choquet. Considerando apenas o elenco, novamente nenhum dos classificadores individuais é capaz de ser o melhor em todas as bases. Neste experimento o ELM foi melhor nas bases *covtype*, *higgs* e *susy* e a DRBM nas bases *DNA* e *isolet*. Embora não tenha tido o melhor desempenho em nenhum *benchmark* deste experimento a NN esteve sempre próximo do melhor, tendo um desempenho constante nas cinco bases, diferentemente da DRBM que apresentou um desempenho muito ruim para *covtype*. Neste ponto, vale ressaltar que a DRBM se mostrou promissora principalmente em bases de dados com número elevado de atributos de entradas, o que vai ao encontro do trabalho de Bu et al. (2015), no qual os autores utilizaram a RBM como metodologia para reduzir dimensionalidade de grandes bases de dados. Dentre as metodologias de agregação a média ficou quase sempre abaixo das demais e o voto majoritário foi melhor do que a integral de Choquet apenas uma vez, justamente na base *isolet*. Sendo assim, novamente é possível afirmar que a agregação tornou o resultado final mais confiável. Por fim, tendo em vista o tamanho das bases, vale ressaltar que nesta etapa do experimento uma melhora pequena na acurácia final é capaz de afetar um grande número de amostras, principalmente nas bases *covtype*, *higgs* e *susy*.

Na tabela 8 são descritos os tempos computacionais de execução de cada um dos algoritmos. Dentre os classificadores a rapidez da ELM se destaca perante aos demais. O algoritmo cumpre a afirmação de Huang, Zhu e Siew (2006) de ser capaz de obter resultados milhares de vezes mais rápidos do que uma NN. A DRBM e a NN fornecem bons resultados mas possui alto custo computacional principalmente quando o número de atributos da base aumentam siginificativamente, como o caso da *isolet*. Já o KNN sofre com o número de amostras, quanto mais elas crescem mais tempo o algoritmo consome. Isso se confirma nas bases *susy* e *higgs*, onde o KNN é extremamente lento. No contexto dos agregadores, é possível observar que a integral de Choquet é o mais lento entre eles. Esse resultado é esperado uma vez que o cálculo da medida *fuzzy* demanda mais esforço computacional do que o cálculo da média e do voto majoritário. Como para a agregação o que interessa é a saída dos classificadores, quando o número de entradas cresce, caso das bases *DNA* e *isolet*, os mesmos não influenciam no tempo final. Por outro lado, quando o número de amostras cresce, casos das bases *covtype*, *higgs* e *susy*, o tempo computacional aumenta, obviamente.

De maneira análoga a primeira parte dos experimentos, o teste de Friedman foi aplicado e obteve $p = 0.031111$ indicando que ao menos um algoritmo é significativamente diferente. Novamente o teste de Wilcoxon é executado e o resultado do mesmo, descrito na tabela 9, indica diferença estatística apenas paras os pares KNN - Choquet e NN - Choquet. Novamente o resultado é coerente com os valores da tabela 7.

Bases	Tempos em seg.						
	Classificadores				Agregadores		
	NN	**DRBM**	**ELM**	**KNN**	**Média**	**Major.**	**Choquet**
DNA	$255,33 \pm 12,32$	$184,85 \pm 56,39$	$0,15 \pm 0,08$	$1,52 \pm 0$	$0,07 \pm 0,0$	$0,1 \pm 0,0$	$0,84 \pm 0,01$
Covtype	$1426,5 \pm 157,2$	$127,91 \pm 19,25$	$24,51 \pm 0,60$	$2294,6 \pm 0$	$3,7 \pm 0,17$	$2,4 \pm 0,12$	$64,39 \pm 1,4$
Higgs	$2332,3 \pm 86,54$	$322,32 \pm 115,2$	$41,91 \pm 1,60$	$6748,3 \pm 0$	$5.7 \pm 0,19$	$10,1 \pm 0,25$	$99,4 \pm 1,8$
Isolet	$3440,4 \pm 462,9$	$2839,9 \pm 234,2$	$0,40 \pm 0,02$	$41,31 \pm 0$	$0.43 \pm 0,0$	$0,29 \pm 0,0$	$4,15 \pm 0,1$
Susy	$1712,9 \pm 49,75$	$608,15 \pm 96,68$	$42,97 \pm 1,14$	$5984,6 \pm 0$	$5,9 \pm 0,23$	$9,8 \pm 0,41$	$89,3 \pm 1,91$

Tabela 8 – Tempo computacional de execução de cada algoritmo cada uma das bases de dados da segunda parte do experimento

Comparação	p
FNN - Choquet	0.042500
KNN - Choquet	0.042500

Tabela 9 – Resultados par a par do teste de Wilcoxon com valores de p abaixo de 0.05 para as bases de dados da segunda parte do experimento

O teste estatístico foi mais uma vez complementado com o algoritmo A-TOPSIS. A tabela 10 descreve a variação do ranking de acordo com o os pesos da média e desvio padrão do desempenho. De acordo com os resultados obtidos pelo A-TOPSIS a integral de Choquet em primeiro e o voto majoritário em segundo se mantiveram como melhores opções independente da variação do peso. Na medida que o desvio padrão perde importância, o KNN desce no ranking e a ELM, juntamente com a média das hipóteses sobe.

Variação peso [media, desvio]	**Ranking**
[0.5, 0.5]	Choquet $\prec$ Maj. $\prec$ KNN $\prec$ DRBM $\prec$ ELM $\prec$ Media $\prec$ NN
[0.6, 0.4]	Choquet $\prec$ Maj. $\prec$ ELM $\prec$ DRBM $\prec$ KNN $\prec$ Media $\prec$ NN
[0.7, 0.3]	Choquet $\prec$ Maj. $\prec$ Media $\prec$ ELM $\prec$ DRBM $\prec$ KNN $\prec$ NN
[0.8, 0.2]	Choquet $\prec$ Maj. $\prec$ Media $\prec$ ELM $\prec$ NN $\prec$ DRBM $\prec$ KNN
[0.9, 0.1]	Choquet $\prec$ Maj. $\prec$ Media $\prec$ NN $\prec$ ELM $\prec$ DRBM $\prec$ KNN
[1.0, 0.0]	Choquet $\prec$ Maj. $\prec$ Media $\prec$ NN $\prec$ ELM $\prec$ DRBM $\prec$ KNN

Tabela 10 – Ranking dos classificadores obtido pelo A-TOPSIS para segunda parte do experimento variando o peso da média e desvio padrão do desempenho

Novamente, uma boa configuração de pesos é 75% para média dos desempenhos e 25% para o desvio padrão. Dessa forma o ranking leva em consideração o desvio e não privilegia o KNN. Na figura 18 é mostrado o gráfico de barras do ranking para esta configuração.

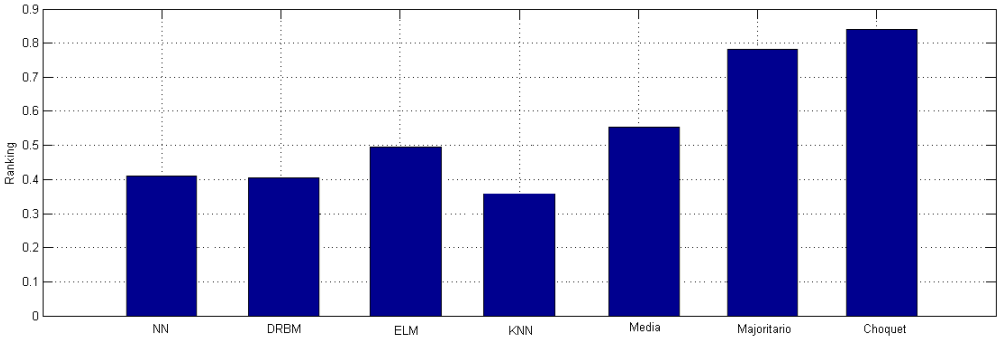


Figura 18 – Ranking dos classificadores obtido pelo A-TOPSIS para segunda parte do experimento

4.3 Estudo de caso

Para realizar o estudo de caso foi criada uma base de dados com imagens de vogais utilizando diversas fontes de escritas diferentes. Cada vogal possui 276 amostras totalizando 1380 amostras na base completa. Na figura 19 é mostrado diversas amostras do conjunto de dados na qual cada vogal é representada por uma imagem 30 x 30 pixels. Portanto, a base vogais 900 atributos de entrada e 5 rótulos de classificação.

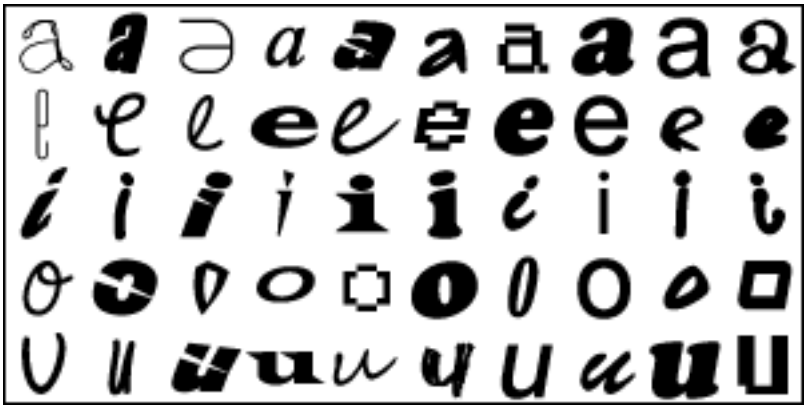


Figura 19 – Exemplos de amostras da base de dados vogais

Para classificar a nova base de dados foi montado o elenco de classificadores com as seguintes configurações de parâmetros, também obtidas de maneira empírica:

- Rede neural *feedforward* (NN): a rede utilizada possui duas camadas ocultas com 30 neurônios cada, utiliza *mini-bacth* de 10 partições, o algoritmo de treinamento executa no máximo 1000 iterações, e os valores de γ são: $\gamma_i = 0,001$, $\gamma_a = 0.1$ e $\gamma_d = 10$ (Hagan; Menhaj, 1994).
- Máquina de aprendizado extremo (ELM): a ELM utilizada possui 250 neurônios na camada oculta.

- Máquina de Boltzmann restrita discriminativa (DRBM): a DRBM utilizada possui 750 neurônios na camada oculta, utiliza *mini-bacth* de 10 partições, executa no máximo 300 iterações e os parâmetros $\hat{\eta}, \hat{\rho}$ e $\hat{\alpha}$ são os mesmo propostos por Hinton (2010) descritos na seção 2.3.1.

- K vizinhos mais próximos (KNN): como descrito no anexo A, o valor de k é igual a $\sqrt{|T|}$, sendo $|T|$ o número de amostras do conjunto de treinamento (Jirina; Jirina, 2011).

Assim como nos experimentos anteriores, com exceção do KNN, cada classificador foi executado 30 vezes para base vogal embaralhada e dividida em 70% para treinamento e 30% para testes e a métrica de desempenho escolhida continua sendo a acurácia de classificação. O desempenho de cada algoritmo, em termos de média e desvio padrão é descrito na tabela 11 e o *boxplot* dos desempenhos é exibido na figura 20.

Acurácia em %							
Base	**Classificadores**				**Agregadores**		
	NN	**DRBM**	**ELM**	**KNN**	**Média**	**Major.**	**Choquet**
Vogais	$86,90 \pm 2,19$	$90,46 \pm 0,18^{*}$	$80,04 \pm 1,72$	$82,85 \pm 0$	$88,56 \pm 0,74$	$89,09 \pm 0,85$	$\mathbf{90.75 \pm 0.88}$

Tabela 11 – Desempenho dos classificadores para base vogais. O asterisco indica o melhor classificador do elenco e o negrito o melhor desempenho global, considerando a acurácia média

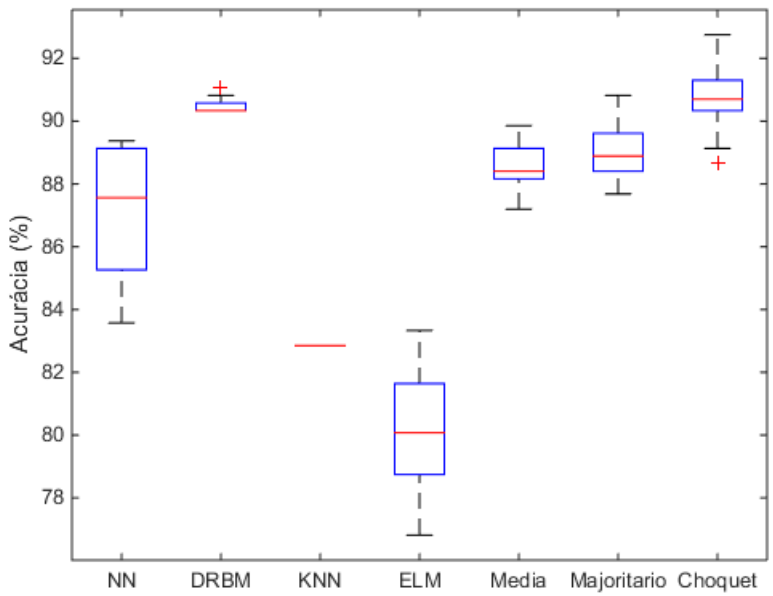


Figura 20 – *Boxplot* do desempenho dos classificadores para base vogais

Os resultados descritos na tabela 11 apontam a agregação via integral de Choquet com melhor desempenho entre os agregadores e a DRBM dentre os classificadores. Mesmo com a ELM e o KNN obtendo resultados muito inferiores quando comparado a DRBM, a integral de Choquet foi capaz de identificar o melhor desempenho. Novamente a DRBM demonstra o seu bom desempenho para bases de dados com número de atributos de entradas elevados, neste caso 900 entradas. No *boxplot* da figura 20 é confirmado o bom desempenho da agregação por Choquet em relação a média e o voto majoritário. Comparando o desempenho da DRBM e Choquet nesse mesmo *boxplot* é possível observar que a agregação por Choquet possui uma mediana ligeiramente acima mas os dados são mais dispersos do que a DRBM.

Com objetivo de investigar um pouco mais a fundo o comportamento dos classificadores e da integral de Choquet como operador de agregação, nas tabelas 12, 13, 14, 15 e 16 são descritas as matrizes de confusão para cada um dos métodos. Analisando os resultados das tabelas, de maneira geral, os principais erros de classificação ocorrem entre as vogais *a*, *o* e *u*. O KNN chama a atenção por ser o único algoritmo a classificar a vogal *i* 100% corretamente. Por outro lado, o desempenho na vogal *a* é muito ruim, comparado aos demais. A maior dificuldade da NN é identificar as vogais *a* e *o*, fazendo com que o algoritmo perca desempenho. Já a ELM possui um ótimo resultado para a vogal *i*, mas o desempenho cai muito ao classificar as demais. A DRBM e a integral de Choquet possuem matrizes de confusão mais constantes em todos os rótulos. A agregação por Choquet é capaz de melhorar o desempenhos da DRBM nas vogais *i* e *u*, piora nas vogais *a* e *o* e praticamente empatam na vogal *e*. Esse resultado vai ao encontro do desempenho global de ambos os algoritmos que é bem próximo.

Valores em %					
	a	e	i	o	u
a	**81.3889**	1.6270	2.6190	8.1349	6.2302
e	1.9907	**88.7500**	2.6389	6.3889	0.2315
i	2.2093	0.5039	**96.3178**	0.7752	0.1938
o	8.2479	3.0769	0.5983	**81.7521**	6.3248
u	7.6950	1.0638	0.4610	4.7163	**86.0638**

Tabela 12 – Matriz de confusão da NN para base vogais

Valores em %					
	a	e	i	o	u
a	**89.6032**	1.1508	3.6111	4.2857	1.3492
e	2.2222	**90.7407**	4.2130	2.8241	0
i	2.2093	1.2016	**95.5814**	0.7364	0.2713
o	3.4615	1.3675	2.5641	**90.0855**	2.5214
u	6.9504	0.4255	2.0922	3.8652	**86.6667**

Tabela 13 – Matriz de confusão para DRBM na base vogais

Valores em %					
	a	e	i	o	u
a	**71.9048**	3.0556	8.2143	9.2063	7.6190
e	4.0278	**76.2963**	7.5000	10.7870	1.3889
i	0.5426	0.6977	**98.4496**	0.1550	0.1550
o	7.6068	3.5470	7.0085	**78.9316**	2.9060
u	8.9716	1.6312	7.5887	7.5177	**74.2908**

Tabela 14 – Matriz de confusão para ELM na base vogais

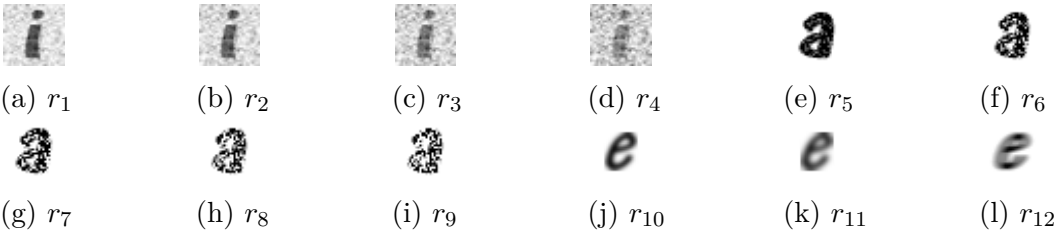
Valores em %					
	a	e	i	o	u
a	**59.5238**	3.5714	7.1429	16.6667	13.0952
e	0	**84.7222**	8.3333	5.5556	1.3889
i	0	0	**100.0000**	0	0
o	1.2821	5.1282	3.8462	**85.8974**	3.8462
u	6.3830	1.0638	3.1915	5.3191	**84.0426**

Tabela 15 – Matriz de confusão para KNN na base vogais

Valores em %					
	a	e	i	o	u
a	**87.4603**	1.1508	3.2143	5.4762	2.6984
e	1.6204	**90.6944**	3.9352	3.7037	0.0463
i	2.2093	0.9302	**96.6667**	0.1163	0.0775
o	3.1624	1.8803	1.8803	**89.5726**	3.5043
u	5.7447	0.7801	1.2411	3.1206	**89.1135**

Tabela 16 – Matriz de confusão para integral de Choquet na base vogais

Para finalizar este estudo de caso, foram introduzidas novas amostras com ruídos e modificações nas vogais. O objetivo desta modificação é verificar a robustez dos classificadores e da agregação por Choquet. Na figura 21 são ilustrados os diferentes ruídos inseridos nas amostras. Na tabela 17 é descrito o desempenho dos classificadores e o resultado da agregação dos mesmos via integral de Choquet.



(a) r_1 (b) r_2 (c) r_3 (d) r_4 (e) r_5 (f) r_6

(g) r_7 (h) r_8 (i) r_9 (j) r_{10} (k) r_{11} (l) r_{12}

Figura 21 – Ruídos inseridos nas vogais

Os resultados descritos na tabela 17 mostram que a integral de Choquet foi capaz de classificar todas as amostras corretamente garantindo certa robustez à ruído. Do elenco de classificadores a NN e a DRBM classificaram quase todas as amostras corretamente,

Classificadores	**Classificação das amostras com ruídos**											
	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}	r_{11}	r_{12}
NN	**i**	**i**	**i**	**i**	**a**	**a**	**a**	**a**	o	**e**	**e**	**e**
DRBM	**i**	**i**	a	a	**a**	**a**	**a**	**a**	**a**	**e**	**e**	**e**
ELM	**i**	**i**	e	e	**a**	**a**	o	o	o	o	o	o
KNN	**i**	**i**	**i**	**i**	**a**	**a**	**a**	o	o	**e**	**e**	o
Choquet	**i**	**i**	**i**	**i**	**a**	**a**	**a**	**a**	**a**	**e**	**e**	**e**

Tabela 17 – Classificação das amostras com os ruídos ilustrados na figura 21. Em negrito os rótulos corretos

errando apenas um e dois rótulos, respectivamente. Em seguida vem o KNN com três erros e por último a ELM que foi o pior classificador do teste.

Finalizando o estudo de caso, foram realizadas modificações nas vogais, como mostrado na figura 22. Os resultados para a classificação dessas amostras são descritos na tabela 18.

(a) m_1 (b) m_2 (c) m_3 (d) m_4 (e) m_5 (f) m_6

(g) m_7 (h) m_8 (i) m_9

Figura 22 – Modificações inseridas nas vogais

Classificadores	**Classificação das amostras**								
	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9
NN	e	e	**u**	a	u	**i**	**e**	a	i
DRBM	**o**	**o**	**u**	a	u	**i**	**e**	a	i
ELM	e	e	**u**	o	o	**i**	**e**	o	i
KNN	**o**	**o**	**u**	**e**	o	**i**	o	o	i
Choquet	**o**	**o**	**u**	**e**	u	**i**	**e**	o	i

Tabela 18 – Classificação das amostras com as modificações ilustradas na figura 22. Em negrito os rótulos corretos

Neste último teste a integral de Choquet foi capaz de classificar corretamente 6 das 9 vogais propostas. Nos 3 erros cometidos pela agregação, as amostras m_5, m_8 e m_9, todos os classificadores erram suas classificações individuais. Vale ressaltar, que na amostra m_4 apenas o KNN acertou o rótulo correto e a integral de Choquet foi capaz de identificar esse acerto. Neste teste a NN não demonstrou a mesma eficácia do teste anterior e a DRBM e o KNN obtiveram os melhores desempenhos do elenco.

5 Conclusão

Neste trabalho foram apresentados fundamentos básicos relacionados a classificação de dados bem como quatro algoritmos de classificação, sendo três baseados em redes neurais, uma rede neural *feedforward* (NN), uma máquina de aprendizado extremo (ELM) e uma máquina de Boltzmann restrita discriminativa (DRBM), e um classificador convencional, o K-vizinhos mais próximos (KNN). Dentre os quatro classificadores apresentados dois deles representam algoritmos clássicos da área, NN e KNN, e os outros dois, ELM e principalmente a DRBM, são metodologias recentes na quais diversos trabalhos ainda propõem melhorias para o uso dos mesmos. No que tange a agregação de elenco de classificadores, uma abordagem baseada na integral de Choquet como operador de agregação foi proposta. Para utilizar a integral de Choquet é necessário o uso de uma medida *fuzzy*. Determinar tal medida é um tópico problemático e revisitado na literatura. Sendo assim, a principal contribuição deste trabalho foi o desenvolvimento de uma metodologia para estimar uma medida *fuzzy* para o elenco de classificadores utilizando a análise de componentes principais.

O algoritmo de agregação de classificadores proposto neste trabalho foi aplicado à diferentes *benchmarks* da literatura e seus resultados foram discutidos de maneira mais detalhada ao longo experimentos apresentados. Além disso, foi realizado um estudo de caso para uma base de dados de vogais criada especificamente para este trabalho. De maneira geral a metodologia apresentou bons resultados, tanto para pequenas bases de dados, quanto para grandes. Quando comparada com outros dois algoritmos de agregação comumente utilizados na literatura, a média das hipóteses e voto majoritário dos classificadores do elenco, a integral de Choquet se destaca. Na maioria dos experimentos, o método conseguiu melhorar o resultado final da classificação. Partindo do princípio que nenhum classificador do elenco foi capaz de ser o melhor em todas as bases de dados, a principal contribuição da agregação foi o fato da mesma acompanhar o classificador de melhor desempenho do elenco, gerando uma resposta final mais confiável. Os classificadores e as metodologias de agregação foram comparadas utilizando o A-TOPSIS e o algoritmo confirmou o bom desempenho da integral de Choquet. Todavia, os resultados mostraram que o método possui limitações. Em certos experimentos, quando pelo menos metade do elenco obtem um resultado bem abaixo do melhor, a agregação não foi capaz de acompanhar o classificador do elenco de melhor desempenho. Além disso, os tempos computacionais de execução apresentados nos experimentos para grandes bases de dados mostram que a integral de Choquet foi o método de agregação mais lento dentre os utilizados e isso pode acarretar um problema na medida que as bases de dados ultrapassam a barreira de milhões de amostras.

A limitação exposta nos resultados indica que a metodologia de agregação baseada na integral de Choquet ainda tem potencial para ser melhorada. Metodologias para estimar medida *fuzzy* continuam em foco. Como trabalhos futuros pretende-se utilizar técnicas de aprendizado profundo para auxiliar no cálculo de PCA e na estimativa da medida *fuzzy* e, consequentemente, melhorar o cálculo da integral de Choquet. Além disso, outro objetivo futuro é incorporar procedimentos de *dropout* de maus classificadores do elenco, em tempo de execução do algoritmo, com intuito de evitar que os mesmos contribuam negativamente no resultado final. No contexto de eficiência, é interessante otimizar a implementação bem como utilizar GPU para paralelizar o processamento do cálculo da medida *fuzzy*. Por fim, deseja-se estender o método proposto neste trabalho para agregar um elenco de preditores de séries temporais.

Referências

Aggarwal, C. C. *Data classification: algorithms and applications*. Minneapolis, USA: CRC Press, 2014.

Barros, L. C. de; Bassanezi, R. C. *Tópicos de lógica fuzzy e biomatemática*. São Paulo, Brasil: Grupo de Biomatemática, Instituto de Matemática, Estatística e Computação Científica (IMECC), 2006.

Bengio, Y. Learning deep architectures for AI. *Foundations and trends in Machine Learning*, v. 2, n. 1, p. 1–127, 2009.

Bonetti, A. et al. Modelling group processes and effort estimation in project management using the Choquet integral: An MCDM approach. *Expert Systems with Applications*, v. 39, n. 18, p. 13366–13375, 2012.

Bu, Y. et al. Restricted Boltzmann machine: a non-linear substitute for PCA in spectral processing. *Astronomy & Astrophysics*, v. 576, p. A96, 2015.

Cao, Y. Aggregating multiple classification results using Choquet integral for financial distress early warning. *Expert Systems With Applications*, v. 39, n. 2, p. 1830–1836, 2012.

Chen, J.; He, Q.; Li, Y. Different types of classifiers combination based on Choquet integral. *IEEE International Conference on Systems Man and Cybernetics*, p. 4056–4061, 2010.

Chen, T.-Y.; Wang, J.-C. Identification of λ-fuzzy measures using sampling design and genetic algorithms. *Fuzzy Sets and Systems*, v. 123, n. 3, p. 321–341, 2001.

Choquet, G. Theory of capacities. *Annales de l'institut Fourier*, v. 5, p. 131–295, 1954.

Dasarathy, B. V.; Sheela, B. V. A composite classifier system design: concepts and methodology. *Proceedings of the IEEE*, v. 67, n. 5, p. 708–713, 1979.

Derrac, J. et al. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, v. 1, n. 1, p. 3–18, 2011.

Fisher, R. A. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, v. 7, n. 2, p. 179–188, 1936.

Fukunaga, K.; Narendra, P. M. A branch and bound algorithm for computing k-nearest neighbors. *IEEE Transactions on Computers*, v. 100, n. 7, p. 750–753, 1975.

Goncalves, E. C. *Novel Classifier Chain Methods for Multi-label Classification Based on Genetic Algorithms*. Tese (Doutorado) — Universidade Federal Fluminense, Brasil, 2015.

Grabisch, M. Fuzzy integral in multicriteria decision making. *Fuzzy sets and Systems*, v. 69, n. 3, p. 279–298, 1995.

Hagan, M. T.; Menhaj, M. B. Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks*, v. 5, n. 6, p. 989–993, 1994.

Han, F.; Yao, H.-F.; Ling, Q.-H. An improved evolutionary extreme learning machine based on particle swarm optimization. *Neurocomputing*, v. 116, p. 87–93, 2013.

Hansen, L. K.; Salamon, P. Neural network ensembles. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, n. 10, p. 993–1001, 1990.

Haykin, S. *Neural Networks: A Comprehensive Foundation (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2007.

He, Q. et al. Choquet fuzzy integral aggregation based on g-lambda fuzzy measure. *International Conference on Wavelet Analysis and Pattern Recognition*, v. 1, p. 98–102, 2007.

Heaton, J. *Artificial intelligence for humans, volume 3: deep learning and neural networks*. Chesterfield, England: Heaton Research Inc, 2015.

Hinton, G. A practical guide to training restricted Boltzmann machines. *Momentum*, v. 9, n. 1, p. 926, 2010.

Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural computation*, v. 14, n. 8, p. 1771–1800, 2002.

Hinton, G. E.; Osindero, S.; Teh, Y.-W. A fast learning algorithm for deep belief nets. *Neural computation*, v. 18, n. 7, p. 1527–1554, 2006.

Hinton, G. E.; Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *Science*, v. 313, n. 5786, p. 504–507, 2006.

Höhle, U. Integration with respect to fuzzy measures. *Proc. IFAC Symposium on Theory and Applications of Digital Control*, p. 35–37, 1982.

Hrasko, R.; Pacheco, A. G. C.; Krohling, R. A. Time series prediction using restricted Boltzmann machines and backpropagation. *Procedia Computer Science*, v. 55, p. 990–999, 2015.

Huang, G.-B.; Zhu, Q.-Y.; Siew, C.-K. Extreme learning machine: a new learning scheme of feedforward neural networks. *IEEE International Joint Conference on Neural Networks*, v. 2, p. 985–990, 2004.

Huang, G.-B.; Zhu, Q.-Y.; Siew, C.-K. Extreme learning machine: theory and applications. *Neurocomputing*, v. 70, n. 1, p. 489–501, 2006.

Hwang, C.; Yoon, K. Multiple attributes decision making methods and applications. *Berlin: Springer-Verlag*, 1981.

Ji, N. et al. Discriminative restricted Boltzmann machine for invariant pattern recognition with linear transformations. *Pattern Recognition Letters*, v. 45, p. 172–180, 2014.

Jirina, M.; Jirina, M. J. *Classifiers Based on Inverted Distances*. Prague, Czech Republic: INTECH Open Access Publisher, 2011.

Jolliffe, I. *Principal component analysis (2nd Edition)*. New York, USA: Springer, 2002.

Keyvanrad, M. A.; Homayounpour, M. M. A brief survey on deep belief networks and introducing a new object oriented MATLAB toolbox (deebnet v2. 1). *arXiv preprint arXiv:1408.3264*, 2014.

Kodratoff, Y. *Introduction to machine learning*. Palo Alto, USA: Morgan Kaufmann, 2014.

Kojadinovic, I. Estimation of the weights of interacting criteria from the set of profiles by means of information-theoretic functionals. *European Journal of Operational Research*, v. 155, n. 3, p. 741–751, 2004.

Kourentzes, N.; Petropoulos, F. Forecasting with multivariate temporal aggregation: The case of promotional modelling. *International Journal of Production Economics, in press*, 2015.

Krawczyk, B.; Woźniak, M. Untrained weighted classifier combination with embedded ensemble pruning. *Neurocomputing*, v. 196, p. 14–22, 2016.

Krishnan, A. R.; Kasim, M. M.; Bakar, E. M. N. E. A. A short survey on the usage of Choquet integral and its associated fuzzy measure in multiple attribute analysis. *Procedia Computer Science*, v. 59, p. 427–434, 2015.

Krohling, R. A. Comunicado privado. PPGI/UFES. 2015.

Krohling, R. A.; Pacheco, A. G. A-TOPSIS–an approach based on TOPSIS for ranking evolutionary algorithms. *Procedia Computer Science*, v. 55, p. 308–317, 2015.

Larbani, M. et al. A novel method for fuzzy measure identification. *International Journal of Fuzzy Systems*, v. 13, n. 1, p. 24–34, 2011.

Larochelle, H.; Bengio, Y. Classification using discriminative restricted Boltzmann machines. *Proceedings of the 25th international conference on Machine learning*, p. 536–543, 2008.

Larochelle, H. et al. Learning algorithms for the classification restricted Boltzmann machine. *The Journal of Machine Learning Research*, v. 13, n. 1, p. 643–669, 2012.

LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature*, v. 521, n. 7553, p. 436–444, 2015.

Lee, K.-M.; Leekwang, H. Identification of λ-fuzzy measure by genetic algorithms. *Fuzzy Sets and Systems*, v. 75, n. 3, p. 301–309, 1995.

Levenberg, K. A method for the solution of certain problems in least squares. *Quarterly of Applied Mathematics*, v. 11, n. 2, p. 164–168, 1944.

Lichman, M. *UCI Machine Learning Repository*. 2013. Disponível em: <http://archive.ics.uci.edu/ml>.

Liu, K.; Zhang, L. M.; Sun, Y. W. Deep Boltzmann machines aided design based on genetic algorithms. *Applied Mechanics and Materials*, v. 568, p. 848–851, 2014.

Marquardt, D. W. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, v. 11, n. 2, p. 431–441, 1963.

Memisevic, R.; Hinton, G. E. Learning to represent spatial transformations with factored higher-order Boltzmann machines. *Neural Computation*, v. 22, n. 6, p. 1473–1492, 2010.

Merigó, J. M.; Palacios-Marqués, D.; Ribeiro-Navarrete, B. Aggregation systems for sales forecasting. *Journal of Business Research*, v. 68, n. 11, p. 2299–2304, 2015.

Mousavi, R.; Eftekhari, M. A new ensemble learning methodology based on hybridization of classifier ensemble selection approaches. *Applied Soft Computing*, v. 37, p. 652–666, 2015.

Murofushi, T.; Sugeno, M. An interpretation of fuzzy measures and the Choquet integral as an integral with respect to a fuzzy measure. *Fuzzy sets and Systems*, v. 29, n. 2, p. 201–227, 1989.

Murofushi, T.; Sugeno, M. A theory of fuzzy measures: representations, the Choquet integral, and null sets. *Journal of mathematical Analysis and Applications*, v. 159, n. 2, p. 532–549, 1991.

Pacheco, A. G. C.; Krohling, R. A. Aggregation of neural classifiers using choquet integral with respect to a fuzzy measure. *Applied Soft Computing*, 2016. Submetido.

Pacheco, A. G. C.; Krohling, R. A. Agregação de elenco de classificadores utilizando integral de choquet com respeito a medida λ-fuzzy. In: XLVIII Simpósio Brasileiro de Pesquisa Operacional. Vitória, ES, 2016. No prelo.

Pacheco, A. G. C.; Krohling, R. A. Classificação de grandes bases de dados utilizando máquina de Boltzmann restrita discriminativa. In: XLVIII Simpósio Brasileiro de Pesquisa Operacional. Vitória, ES, 2016. No prelo.

Papa, J. P. et al. Model selection for discriminative restricted Boltzmann machines through meta-heuristic techniques. *Journal of Computational Science*, v. 9, p. 14–18, 2015.

Papa, J. P.; Scheirer, W.; Cox, D. D. Fine-tuning deep belief networks using harmony search. *Applied Soft Computing*, v. 46, p. 875–885, 2016.

Pedrycz, W.; Gomide, F. *An introduction to fuzzy sets: analysis and design*. Cambridge, USA: Mit Press, 1998.

Polikar, R. Ensemble based systems in decision making. *IEEE Circuits and systems magazine*, v. 6, n. 3, p. 21–45, 2006.

Polyak, B. T. Newton's method and its use in optimization. *European Journal of Operational Research*, v. 181, n. 3, p. 1086–1096, 2007.

Rowley, H. V.; Geschke, A.; Lenzen, M. A practical approach for estimating weights of interacting criteria from profile sets. *Fuzzy Sets and Systems*, v. 272, p. 70–88, 2015.

Rumelhart, D. E.; Hinton, G. E.; Williams, R. J. Learning representations by back-propagating errors. *Cognitive modeling*, v. 5, n. 3, p. 1, 1988.

Salama, M. A.; Hassanien, A. E.; Fahmy, A. A. Deep belief network for clustering and classification of a continuous data. *IEEE International Symposium on Signal Processing and Information Technology*, p. 473–477, 2010.

Seokho, K.; Cho, S.; Kang, P. Multi-class classification via heterogeneous ensemble of one-class classifiers. *Engineering Applications of Artificial Intelligence*, v. 43, p. 35–43, 2015.

Serre, D. *Matrices: theory and applications*. New York: Springer, 2002.

Smolensky, P. Information processing in dynamical systems: Foundations of harmony theory. *DTIC Document*, 1986.

Štefka, D.; Holeňa, M. Dynamic classifier aggregation using fuzzy integral with interaction-sensitive fuzzy measure. *10th International Conference on Intelligent Systems Design and Applications*, p. 225–230, 2010.

Štefka, D.; Holeňa, M. Dynamic classifier aggregation using interaction-sensitive fuzzy measures. *Fuzzy Sets and Systems*, v. 270, p. 25–52, 2015.

Sugeno, M. *Theory of fuzzy integrals and its applications*. Tese (Doutorado) — Tokyo Institute of Technology, Japão, 1974.

Takahagi, E. A fuzzy measure identification method by diamond pairwise comparisons: AHP scales and Grabish's graphical interpretation. *Knowledge-Based Intelligent Information and Engineering Systems*, p. 316–324, 2007.

Tamilselvan, P.; Wang, P. Failure diagnosis using deep belief learning based health state classification. *Reliability Engineering & System Safety*, v. 115, p. 124–135, 2013.

Tapia-Rosero, A. et al. Fusion of preferences from different perspectives in a decision-making context. *Information Fusion*, v. 29, p. 120–131, 2016.

Xu, L.; Krzyżak, A.; Suen, C. Y. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man and Cybernetics*, v. 22, n. 3, p. 418–435, 1992.

ANEXO A – K vizinhos mais próximos

O algoritmo K vizinhos mais próximos (do inglês: K *neareast neighbours*, KNN) foi proposto por Fukunaga e Narendra (1975) e até hoje é bastante utilizado para problemas de classificação. A ideia principal do algoritmo é determinar o rótulo de classificação de uma amostra baseado nas amostras vizinhas advindas de um conjunto de treinamento. O algoritmo é bastante simples e de fácil compreensão. Na figura 23 é ilustrado um exemplo de aplicação do algoritmo para um problema de classificação com dois rótulos de classe e com $k = 7$. No exemplo, são aferidas as distâncias de uma nova amostra, representada por uma estrela, às demais amostras de treinamento, representadas pelas bolinhas azuis e amarelas. Das sete amostras de treinamento mais próximas da nova amostra, 4 são do rótulo A e 3 do rótulo B. Portanto, como existem mais vizinhos do rótulo A, a nova amostra receberá o mesmo rótulo deles, ou seja, A.

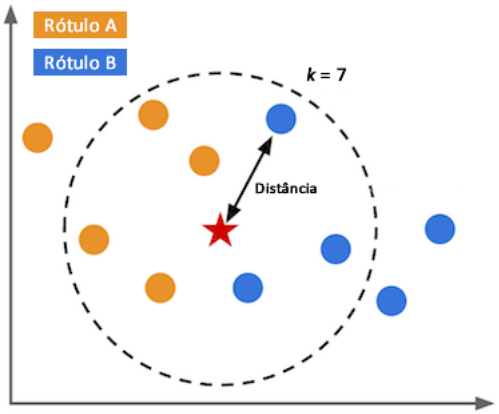


Figura 23 – Exemplo de classificação do KNN com dois rótulos de classe e $k = 7$

Dois pontos chaves que devem ser determinados para aplicação do KNN são: a métrica de distância e o valor de k. Para métrica de distância a mais utilizada é a distância Euclidiana, descrita por:

$$D = \sqrt{(p_1 - q_1)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{A.1}$$

onde $P = (p_1, \cdots, p_n)$ e $Q = (q_1, \cdots, q_n)$ são dois pontos n-dimensionais. O valor k é determinado de maneira empírica, não existe um valor único para a constante, a mesma varia de acordo com a base de dados. Neste trabalho, o valor de k é igual a $\sqrt{|T|}$, sendo $|T|$ o número de amostras do conjunto de treinamento (Jirina; Jirina, 2011).

A grande vantagem do KNN é sua abordagem simples de ser compreendida e implementada. Todavia, calcular distância é tarefa custosa e caso o problema possua grande número de amostras o algoritmo pode consumir muito tempo computacional. Além disso, o método é sensível à escolha do k. O pseudocódigo do KNN é apresentado no Algoritmo 8.

Algoritmo 8: *K vizinhos mais próximos*

inicialização:
    Preparar conjunto de dados de entrada e saída T de acordo com a definição 2.1;
    Informar o valor de k;
para cada nova amostra **faça**
    Calcular distância para todas as amostras de T de acordo com a equação A.1
    Determinar o conjunto das $k's$ distâncias mais próximas
    O rótulo com mais representantes no conjunto dos $k's$ vizinhos será o escolhido
fim para
retornar: conjunto de rótulos de classificação

ANEXO B – Exemplo de cálculo da medida *fuzzy* e integral de Choquet

Neste anexo será apresentado um exemplo de estimativa da medida *fuzzy*, utilizando a metodologia proposta neste trabalho, e cálculo da integral de Choquet baseado na medida estimada. Considerando $\mathbf{M}$ uma matriz de decisão para uma amostra qualquer obtida a partir de um elenco com 3 classificadores, c_1, c_2 e c_3 aplicados a um problema de classificação com 3 rótulos de classificação tem-se:

$$\mathbf{M} = \begin{bmatrix} 0.2152 & 0.1223 & 0.2896 \\ 0.2116 & 0.2041 & 0.3291 \\ 0.5732 & 0.6736 & 0.3813 \end{bmatrix} \tag{B.1}$$

Para este caso, $\Omega = \{c_1, c_2, c_3\}$ e os subconjuntos $A = \{\{\varnothing\}, \{c_1\}, \{c_2\}, \{c_3\}, \{c_1, c_2\}, \{c_1, c_3\}, \{c_2, c_3\}, \{c_1, c_2, c_3\}\}$. Como o elenco possui 3 classificadores, A possui 2^3 subconjuntos, denominados A_i, com $i = 0, \cdots, 7$.

De agora em diante será exemplificado, a partir da matriz $\mathbf{M}$, o procedimento para estimar as medidas *fuzzy*, descrita na seção 3.2.1. Para isso, cada subconjunto A_i irá dispor de uma matriz $\mathbf{P}_i$ advinda de $\mathbf{M}$. Exibir todas as matrizes é algo dispendioso, portanto $\mathbf{P}_i$ será exemplificado para os subconjuntos $A_7 = \{c_1, c_2, c_3\}$, $A_4 = \{c_1, c_2\}$, $A_1 = \{c_1\}$ e $A_0 = \{\varnothing\}$, sendo o restante calculado de maneira análoga. Obviamente, para o conjunto vazio a matriz $\mathbf{P}_0 = [\varnothing]$ e para o conjunto $\{c_1, c_2, c_3\}$, $\mathbf{P}_7 = \mathbf{M}$. Para os demais subconjuntos a matriz relacionada será construída a partir dos valores de cada classificador participantes deste subconjunto, da seguinte forma:

$$\mathbf{P_4} = \begin{bmatrix} 0.2152 & 0.1223 \\ 0.2116 & 0.2041 \\ 0.5732 & 0.6736 \end{bmatrix} \qquad \mathbf{P_1} = \begin{bmatrix} 0.2152 \\ 0.2116 \\ 0.5732 \end{bmatrix} \tag{B.2}$$

A partir das matrizes $\mathbf{P}_i$ são obtidas as matrizes de variância-covariância $\mathbf{R}_i$:

$$\mathbf{R}_7 = \begin{bmatrix} 0.0432 & 0.0611 & 0.0086 \\ 0.0611 & 0.0885 & 0.0130 \\ 0.0086 & 0.0130 & 0.0021 \end{bmatrix} \qquad \mathbf{R}_4 = \begin{bmatrix} 0.0432 & 0.0611 \\ 0.0611 & 0.0885 \end{bmatrix} \qquad \mathbf{R}_1 = \begin{bmatrix} 0.0432 \end{bmatrix} \tag{B.3}$$

E os autovalores associados as matrizes $\mathbf{R}_i$ serão $\lambda_7 = [0, 0.0009, 0.1329]$, $\lambda_4 = [0.0006, 0.1310]$ e $\lambda_1 = [0.0432]$. Com isso, é possível calcular os valores de $J^*(A_i)$ e consequentemente a

medida *fuzzy* da seguinte forma:

$$\begin{aligned}
\mu(A_0) &= \frac{J^*(A_0)}{J^*(\Omega = A_7)} = \frac{0}{0.1338} = 0 \\
\mu(A_7) &= \frac{J^*(A_7)}{J^*(\Omega = A_7)} = \frac{0.1338}{0.1338} = 1 \\
\mu(A_4) &= \frac{J^*(A_4)}{J^*(\Omega = A_7)} = \frac{0.1317}{0.1338} = 0.9843 \\
\mu(A_1) &= \frac{J^*(A_1)}{J^*(\Omega = A_7)} = \frac{0.0432}{0.1338} = 0.3328
\end{aligned} \tag{B.4}$$

As demais estimativas são realizadas de maneira análoga. Ao final, o algoritmo fornecerá as seguintes medidas:

$$\begin{array}{rrrr}
\mu(A_0) = 0 & \mu(A_1) = 0.3328 & \mu(A_2) = 0.6616 & \mu(A_3) = 0.0158 \\
\mu(A_4) = 0.9843 & \mu(A_5) = 0.3384 & \mu(A_6) = 0.6774 & \mu(A_7) = 1
\end{array}$$

Dada as medidas, as linhas da matriz $\mathbf{M}$ são agregadas por meio da integral de Choquet. A fórmula geral da agregação, de acordo com a seção 3.1.3 será:

$$\int f d\mu = f(x_1)\mu(\{x_1, x_2, x_3\}) + (f(x_2) - f(x_1))\mu(\{x_2, x_3\}) + (f(x_3) - f(x_2))\mu(\{x_3\}) \tag{B.5}$$

Todavia as entradas $f(x)$, que serão os valores da matriz $\mathbf{M}$, precisam ser ordenados e isso exige atenção pois altera a ordem dos subconjuntos de A. O cálculo das integrais com os valores já ordenados são descritos a seguir:

$$\begin{aligned}
r_1 &= 0.1223 \times \mu\{c_2, c_1, c_3\} + (0.2152 - 0.1223) \times \mu\{c_1, c_3\} + (0.2896 - 0.2152) \times \mu\{c_3\} \\
r_1 &= 0.1223 \times 1 + 0.0929 \times 0.3384 + 0.0744 \times 0.015 = 0.1549
\end{aligned}$$

$$\begin{aligned}
r_2 &= 0.2041 \times \mu\{c_2, c_1, c_3\} + (0.2116 - 0.2041) \times \mu\{c_1, c_3\} + (0.3291 - 0.2116) \times \mu\{c_3\} \\
r_2 &= 0.2041 \times 1 + 0.0075 \times 0.3384 + 0.1175 \times 0.015 = 0.2084
\end{aligned}$$

$$\begin{aligned}
r_3 &= 0.3813 \times \mu\{c_3, c_1, c_2\} + (0.5732 - 0.3813) \times \mu\{c_1, c_2\} + (0.6736 - 0.5732) \times \mu\{c_2\} \\
r_3 &= 0.3813 \times 1 + 0.1919 \times 0.9843 + 0.1004 \times 0.6616 = 0.6366
\end{aligned} \tag{B.6}$$

Para finalizar, os rótulos r_1, r_2 e r_3 formam o vetor coluna $\mathbf{C_f}$:

$$\mathbf{C_f} = \begin{bmatrix} 0.1549 \\ 0.2084 \\ \mathbf{0.6366} \end{bmatrix} \tag{B.7}$$

onde o rótulo de classificação final será r_3.

ANEXO C – O algoritmo A-TOPSIS

Normalmente, ao utilizar algoritmos estocásticos os mesmos são executados diversas vezes para diferentes *benchmarks*. Uma grande dificuldade é comparar esses algoritmos e definir qual é o melhor. É comum o uso de teste estatístico para comparar esses algoritmos, porém, além de ser uma tarefa dispendiosa, o teste não indica qual é o melhor algoritmo utilizado e o número de testes cresce significativamente caso se utilize muitos algoritmos. Sendo assim, o A-TOPSIS (Krohling; Pacheco, 2015) é um método alternativo, baseado no TOPSIS (do inglês: *Technique for Order Preference by Similarity to Ideal Solution* (Hwang; Yoon, 1981), para solucionar o problema de ranqueamento e comparação de algoritmos.

O A-TOPSIS é aplicado nos valores de média e desvio padrão dos algoritmos para os *benchmarks*. Seja $A = \{a_1, \cdots, a_k\}$ um grupo de K algoritmos estocásticos que pretende-se comparar, de acordo com o desempenho dos mesmos em um grupo de N *benchmarks* $B = \{b_1, \cdots, bn\}$, sendo $n = 1, \cdots, N$ e $k = 1, \cdots, K$. Como os algoritmos são executados diversas vezes para cada *benchmark*, são calculadas as matrizes de média e desvio padrão da seguinte forma:

$$\mathbf{M}_\mu = \begin{array}{c} \\ a_1 \\ \vdots \\ a_k \end{array} \begin{array}{c} \begin{array}{ccc} b_1 & \dots & b_n \end{array} \\ \begin{bmatrix} \mu_{11} & \dots & \mu_{1n} \\ \vdots & \ddots & \vdots \\ \mu_{k1} & \dots & \mu_{kn} \end{bmatrix} \end{array} \qquad \mathbf{M}_\sigma = \begin{array}{c} \\ a_1 \\ \vdots \\ a_k \end{array} \begin{array}{c} \begin{array}{ccc} b_1 & \dots & b_n \end{array} \\ \begin{bmatrix} \sigma_{11} & \dots & \sigma_{1n} \\ \vdots & \ddots & \vdots \\ \sigma_{k1} & \dots & \sigma_{kn} \end{bmatrix} \end{array} \tag{C.1}$$

sendo μ_{kn} e σ_{kn} a média e o desvio padrão do algoritmo a_k para o *benchmark* b_n, respectivamente. A partir das matrizes de C.1 o A-TOPSIS é capaz de realizar o ranking dos melhores algoritmos em relação ao grupo de *benchmarks* utilizados como ilustra a figura 24.

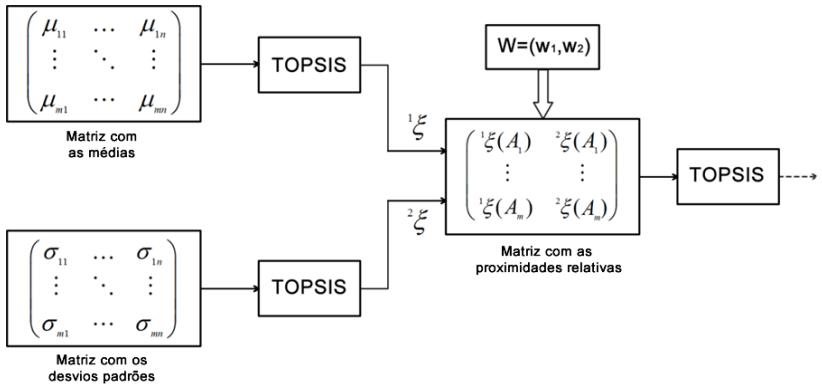


Figura 24 – Ilustração do funcionamento do A-TOPSIS

O funcionamento do algoritmo é descrito na forma de passos a seguir:

Passo 1: normalizar as matrizes $\mathbf{M}_\mu$ e $\mathbf{M}_\sigma$.

Passo 2: o A-TOPSIS herda do TOPSIS os conceitos de critérios de benefício - quanto maior melhor - e custo - quanto menor melhor. Como o desempenho dos *benchmarks* é medido em termos de acurácia, este é considerado um critério de benefício. Com isso, devem ser identificadas a solução ideal positiva A^+ (benefício) e a solução ideal negativa A^- (custo), para as duas matrizes $\mathbf{M}_\mu$ e $\mathbf{M}_\sigma$, da seguinte forma:

$$\begin{aligned} A^+ &= (p_1^+, p_2^+, \cdot, p_k^+) \;\; \because \;\; p_j^+ = (max_i \;\; p_{ij}, j \in J_1; \;\; min_i \;\; p_{ij}, j \in J_2;) \\ A^- &= (p_1^-, p_2^-, \cdot, p_k^-) \;\; \because \;\; p_j^- = (min_i \;\; p_{ij}, j \in J_1; \;\; max_i \;\; p_{ij}, j \in J_2;) \end{aligned} \tag{C.2}$$

onde J_1 e J_2 representam os critérios de benefício e custo, respectivamente. Além disso, como esse passo é aplicado para as duas matrizes $\mathbf{M}_\mu$ e $\mathbf{M}_\sigma$, p_{ij} é uma generalização para os valores das μ e σ de cada uma delas.

Passo 3: calcular a distância Euclidiana de A^+ e A^- para cada algoritmo A_k, das duas matrizes, da seguinte forma:

$$\begin{aligned} d_i^+ &= \sqrt{\sum_{j=1}^{n}(p_j^+ - p_{ij})^2}, \;\; \text{com} \;\; i = 1, \cdots, k \\ d_i^- &= \sqrt{\sum_{j=1}^{n}(p_j^- - p_{ij})^2}, \;\; \text{com} \;\; i = 1, \cdots, k \end{aligned} \tag{C.3}$$

Passo 4: calcular a proximidade relativa de cada algoritmo $\xi(A_k)$, para cada uma das matrizes, com respeito as distâncias calculadas no passo anterior:

$$\xi(A_k) = \frac{d_i^-}{d_i^+ + d_i^-}, \;\; \text{com} \;\; i = 1, \cdots, k \tag{C.4}$$

Para a matriz $\mathbf{M}_\mu$ deve ser gerado a $\xi^1(A_k)$ e para a matriz $\mathbf{M}_\sigma$ a $\xi^2(A_k)$

Passo 5: após o cálculo das proximidades relativas para ambas as matrizes, deve ser gerada a matriz $\mathbf{G}$:

$$\mathbf{G} = \begin{bmatrix} \xi^1(A_1) & \xi^2(A_1) \\ \vdots & \vdots \\ \xi^1(A_k) & \xi^2(A_k) \end{bmatrix} \tag{C.5}$$

Neste caso, é aplicado um peso (w_1, w_2) para cada ξ, no qual w_1 pondera a contribuição da média e w_2 pondera a contribuição do desvio padrão. Obviamente a soma dos pesos deve ser igual a 1. Sendo assim, a matriz ponderada é obtida da seguinte forma:

$$\mathbf{G} = \begin{bmatrix} w_1\xi^1(A_1) & w_2\xi^2(A_1) \\ \vdots & \vdots \\ w_1\xi^1(A_k) & w_2\xi^2(A_k) \end{bmatrix} \tag{C.6}$$

Passo 6: nesta etapa os passos 2, 3 e 4 são aplicados para a matriz $\mathbf{G}$, gerando um ξ_G, no qual, os melhores algoritmos são aqueles com maior valor. A partir de ξ_G é gerado o ranking dos melhores algoritmos.