Guilherme Artém dos Santos

Máquinas de Boltzmann restrita usando otimização via enxame de partículas para classificação de dados

Vitória

2017

Guilherme Artém dos Santos

Máquinas de Boltzmann restrita usando otimização via enxame de partículas para classificação de dados

Monografia apresentada como Projeto de Graduação em Engenharia de Computação da Universidade Federal do Espírito Santo, como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação

Universidade Federal do Espírito Santo - UFES

Departamento de Informática

Orientador: Renato A. Krohling

Coorientador: André G. C. Pacheco

Vitória

2017

Agradecimentos

Agradeço à minha família, por todo carinho e apoio durante toda a vida, e à educação recebida, que me fizeram chegar até onde estou.

À minha namorada, Maria, por todo o apoio que me deu durante os últimos meses.

Agradeço ao orientador prof. Dr.-Ing. Renato Krohling e ao coorientador M.Sc. André Pacheco, pela orientação, direções, disponibilidade e paciência.

Aos professores prof. Dr. Thiago Oliveira dos Santos e prof. M.Sc. Carlos Alexandre Siqueira da Silva por aceitarem o convite para participarem da banca, mesmo sob pouca antecedência.

Aos colegas de laboratório por toda as explicações, conversas, ajuda e tempo disponibilizado desde 2015.

Ao PET Engenharia de Computação do qual eu participei, e à tutora, a prof Roberta Lima Gomes e a todos os colegas que participaram comigo, que me ajudaram durante toda a graduação, e me fizeram uma pessoa melhor.

Aos colegas, professores e equipe de apoio que estiveram presentes antes e durante a graduação, sem os quais o percurso teria sido impossível.

Descobrir consiste em olhar para o que todo mundo está vendo
e pensar uma coisa diferente.
- Roger Von Oech

Resumo

Técnicas de aprendizado de máquinas, como redes neurais, vêm sendo amplamente aprimoradas nos últimos anos, e aplicadas em diversas áreas do conhecimento para a resolução de vários tipos de problemas, como classificação de dados, clusterização e predição. Um tipo de rede neural muito utilizada é a máquina de Boltzmann restrita (RBM), uma rede com treinamento não supervisionado, que tem como uma de suas funções a extração de características. O treinamento clássico de RBMs é baseado na técnica de gradiente descendente, que possui uma propensão à queda em mínimos locais, o que pode ser um problema desse método de treinamento. Muitas meta-heurísticas foram desenvolvidas para problemas de otimização a fim de evitar mínimos locais. Uma dessas meta-heurísticas é a otimização por enxame de partículas (PSO), um algoritmo bio-inspirado que possui uma dinâmica de população a fim de tentar evitar esse problema. Neste trabalho é proposta uma máquina de Boltzmann restrita utilizando o PSO como algoritmo para treinamento, denominada de PSO-RBM. Para o modelo são realizados diversos experimentos com *benchmarks* convencionais da literatura. Por fim, o PSO-RBM é comparado com a versão original do algoritmo e resultados mostram que para os experimentos de classificação a PSO-RBM obteve resultados similares à treinada pela maneira clássica.

Palavras chave: Máquinas de Boltzmann restrita. Otimização por enxame de partículas. Classificação de dados.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Recentemente abordagens que utilizam aprendizado de máquina (do inglês: *Machine Learning*) vêm sendo amplamente desenvolvidas, principalmente utilizando redes neurais profundas (do inglês: *Deep Neural Networks*, DNNs), sendo usada para a resolução de diversos problemas, como classificação e predição de séries temporais.

Redes neurais artificiais (do inglês: *artificial neural networks*, ANNs) são modelos computacionais inspirados nas redes neurais biológicas do cérebro humano. Uma rede neural é um sistema de computação paralelamente distribuido, constituído de unidades de processamento simples que tem a propensão natural de armazenar conhecimento experimental e torná-lo disponível para o uso (HAYKIN, 2007). A rede adquire conhecimento por processos de treinamento. Em redes neurais convencionais se utiliza o treinamento dito supervisionado, no qual são utilizados padrões de treinamento com dados de entrada e saída da rede, de forma a calcular os pesos sinápticos dos neurônios. Um dos métodos mais utilizados é o *backpropagation* (RUMELHART; HINTON; WILLIAMS, 1988).

Máquinas de Boltzmann restrita (do inglês: *restricted Boltzmann machines*,RBMs) são um subtipo de ANNs que é amplamente utilizada para a construção de DNNs. RBMs foram propostas por Smolensky (1986) mas só começaram a ser utilizadas com o desenvolvimento de novos algoritmos de treinamento, principalmente o *Contrastive Divergence* (CD) (HINTON, 2002), e a partir dele o treinamento de DNNs (HINTON; OSINDERO; TEH, 2006). Diferentemente das ANNs convencionais, as RBMs possuem um treinamento não supervisionado, no qual a rede aprende a distribuição dos dados de entrada, podendo funcionar como um extrator de características dos dados (HINTON, 2014).

Um possível problema do *contrastive divergence* é o fato dele ser uma técnica baseada em gradiente descendente, que possui suscetibilidade a queda em mínimos locais. No campo da otimização, diversas meta-heurísticas foram desenvolvidas com dinâmica de população, de forma a fazer uma busca em caráter global e tentar obter uma menor propensão à mínimos locais. Uma dessas meta-heurísticas é a otimização por enxame de partículas (do inglês: *Particle Swarm Optimization*, PSO), que possui diversas atualizações para melhorar seu desempenho em relação à queda em mínimos locais, e que foi utilizada nesse trabalho como forma alternativa de treinamento ao *contrastive divergence*.

1.1 Motivação e Justificativa

Na área de otimização, muitos algoritmos desenvolvidos com base em derivadas ou gradientes têm a tendência de cair em mínimos locais. Várias meta-heurísticas foram desenvolvidas de forma a tentar obter melhores resultados, dentre elas a otimização por enxame de partículas.

O algoritmo clássico para treinamento de máquinas de Boltzmann restrita é o *contrastive divergence*, que é baseado em gradiente, e portanto herda os problemas de mínimos locais para certas bases. Nesse trabalho, pretende-se avaliar o uso do PSO como uma alternativa de treinamento de RBMs em problemas de classificação, e verificar seu há uma melhora em comparação ao treinamento clássico.

1.2 Objetivo

Este trabalho tem como objetivo o estudo do desempenho de máquinas de Boltzmann restrita com treinamento via otimização por enxame de partículas e averiguar se há uma melhora em relação à RBMs treinadas de maneira tradicional em problemas de classificação de dados.

1.3 Estrutura

O restante do trabalho está organizado na seguinte maneira;

- No capítulo 2 são apresentados os conceitos básicos para o desenvolvimento do trabalho.
- No capítulo 3 é apresentada a PSO-RBM desenvolvida nesse trabalho.
- No capítulo 4 são descritos os experimentos realizados e seus resultados, além de uma análise dos mesmos.
- No capítulo 5 trabalho é sumarizado, são apresentadas conclusões e é apontada a direção para trabalhos futuros.

2 Conceitos Básicos

Neste capítulo serão apresentados conceitos básicos necessários para o desenvolvimento do trabalho. Na seção seção 2.1 é introduzido o problema de classificação de dados. As seções 2.2 e 2.3 introduzem os conceitos básicos de redes neurais artificiais e máquinas de Boltzmann restrita, respectivamente e a seção 2.4 introduz otimização por enxame de partículas.

2.1 Classificação de Dados

O problema de classificação de dados está presente em diversos campos do conhecimento, como diagnósticos médicos, caracterização e filtragem de documentos, análise de redes sociais e análise de dados biológicos (AGGARWAL, 2014). O problema de classificação de dados pode ser descrito como:

"Dado um conjunto de dados de treinamento com seus respectivos rótulos, determinar os rótulos para dados de testes não rotulados."(AGGARWAL, 2014)

A classificação de dados pode ser formalmente definida por:

Definição 2.1. (GONÇALVES, 2015) Dado $X = \{X_1, \cdots, X_d\}$ um conjunto com d atributos para classificação e $L = \{l_1, l_2, \cdots, l_q\}$ um conjunto com q rótulos, onde $q \geq 2$. Considere um conjunto de dados de treinamento D composto de N instancias na forma $\{(x_1, c_1), (x_2, c_2), \cdots, (x_n, c_n)\}$. Neste conjunto cada x_i corresponde a um vetor $(x_i, \cdots, x_d)$ que guarda valores para os atributos de X e cada $c_i \in L$ corresponde a um único rótulo. O objetivo da tarefa de classificação é de aprender de D uma função y (i.e. classificador) que, dado uma instância não rotulada $t = (x, ?)$ é capaz de predizer corretamente seu rótulo c.

Um exemplo clássico de classificação é o problema da *Iris* (FISHER, 1936). Neste problema há um conjunto de flores do gênero *Iris* que são divididas em três grupos: *setosa, virginica* e *versicolor*. Assim, o objetivo é determinar a qual grupo cada flor pertence baseado nas medidas de suas sépalas e pétalas. Neste exemplo X são as medidas de comprimento e largura das sépalas e pétalas, e os valores de L são os rótulos *setosa, virginica* e *versicolor*. Neste trabalho, a classificação será feita por uma abordagem baseada em máquinas de Boltzmann restrita e redes neurais *feedforward*.

2.2 Redes Neurais Artificiais

Uma rede neural artificial é, em sua forma mais geral, uma abordagem para tentar modelar a maneira como o cérebro realiza uma tarefa particular ou função de interesse (HAYKIN, 2007). ANNs podem ser usadas em vários problemas, como processamento de imagem (HU et al., 2016), classificação de dados (BHARDWAJ et al., 2016), predição (KUREMOTO et al., 2014) e diversos outros. Assim como as redes biológicas, ANNs possuem como unidades básicas de processamento os neurônios. Os neurônios artificias foram inspirados no funcionamento de neurônios biológicos. Um exemplo de neurônio é o *perceptron*, mostrado na figura 1.

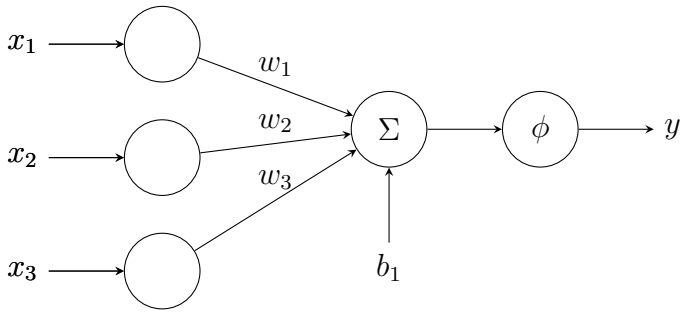


Figura 1 – Modelo de um *Perceptron*

onde (x_1, x_2 e x_3) é um vetor de entradas, (w_1, w_2 e w_3) é um vetor de pesos, b_1 é um elemento de deslocamento linear (do inglês: *bias*), y é a saída e ϕ é uma função de ativação, que possui a função de regular a saída do neurônio, de forma análoga à membrana de disparo em neurônios biológicos. O *perceptron* é descrito matematicamente pela equação 2.1:

$$y = f(\mathbf{x}, \mathbf{w}, b) = \phi(\sum_{i}^{n}(x_i w_i) + b) \tag{2.1}$$

onde $\mathbf{x}$ é o vetor de entradas, $\mathbf{w}$ é o vetor de pesos que multiplicam as entradas, n é o número de entradas, b o peso de *bias* e ϕ a função de ativação. Para um neurônio deve sempre haver o mesmo número de elementos nos vetores de entrada e de peso, enquanto o *bias* é um elemento único.

As funções de ativação ϕ mais utilizadas são a função *sigmóide* e a função tangente hiperbólica, definidas pelas equações 2.2 e 2.3, respectivamente.

$$\phi(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

$$\phi(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.3}$$

Uma das características de uma rede neural é a sua arquitetura, a maneira pela qual seus neurônios estão organizados. As redes são formadas por três tipos de camadas: a de entrada, as ocultas e a de saída. As camadas de entrada e saída representam as entradas e saídas do problema. Já as camadas ocultas são as camadas em que ocorre a maior parte do processamento e processo de aprendizagem. Além disso, é nas conexões entre os neurônios em que é codificado o conhecimento da rede. Normalmente uma rede neural possui uma camada de entrada, uma camada de saída e k camadas escondidas. Um exemplo de rede neural *feedforward* é mostrado na figura 2.

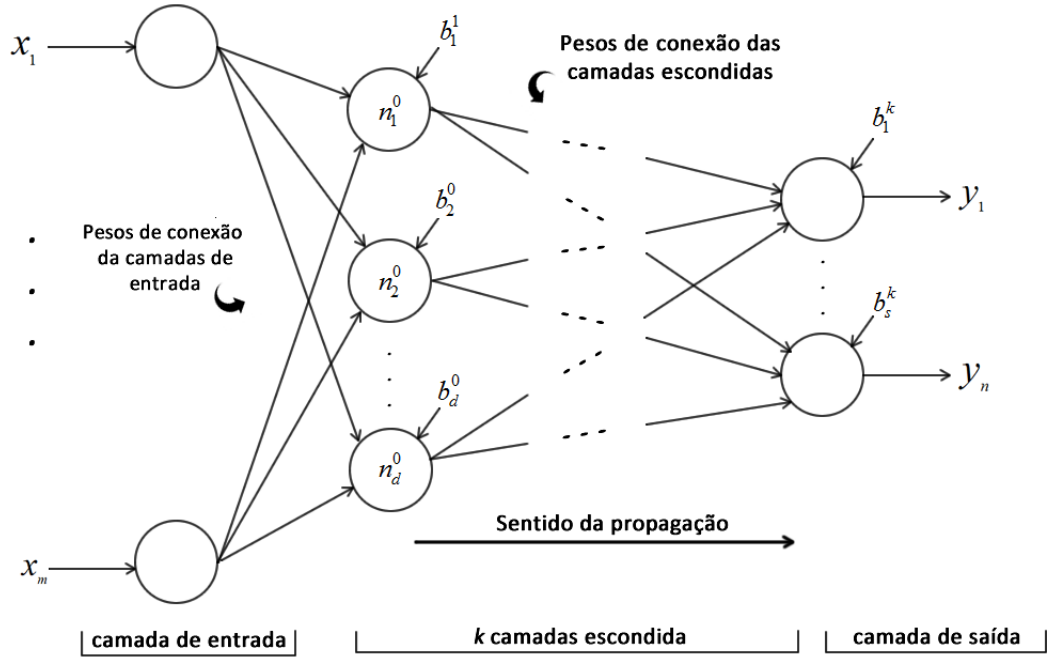


Figura 2 – Rede Neural *feedforward* com k camadas escondidas, retirada de Pacheco (2016a)

Rede *feedforward* é o tipo de rede na qual a informação só trafega em um sentido, onde os sinais dos neurônios de uma entrada só podem estimular os neurônios da camada seguinte. Um dos algoritmos para treinamento para esse tipo de rede é o Levenberg-Marquardt *backpropagation* (HAGAN; MENHAJ, 1994) (abreviado por LMBP), que utiliza-se de uma aproximação do método de Newton, de modo que a atualização dos pesos da rede se dá por:

$$\Delta w_i^t = -[\mathbf{J}'\mathbf{J} + \lambda \mathbf{I}]^{-1}\mathbf{J}'\mathbf{E} \tag{2.4}$$

onde $\mathbf{J}$ é a matriz das derivadas primeiras do erro pelos pesos w_i, chamada de Jacobiana, λ é o fator de amortecimento e $\mathbf{E}$ é a matriz de erros. O fator de amortecimento λ deve ser decrementado caso o erro diminua entre duas iterações e incrementado caso contrário. No algoritmo 1 é apresentado o pseudo-código do LMBP.

Algoritmo 1: *Levenberg-Marquardt backpropagation*

Entradas:
    Conjunto de dados de entrada $\mathbf{X}$ e de saída $\mathbf{Y}$
    Números de camadas ocultas e neurônios em cada camada oculta
Inicialização:
    Inicializar a matriz de pesos $\mathbf{W}$ aleatoriamente
    Inicializar λ
Repita:
    Para cada peso w_i **faça**:
        Calcular Δw_i^t a partir da equação 2.4
        $w_i^{t+1} \leftarrow \Delta w_i^t + w_i^t$
        Caso: erro anterior $>$ erro atual:
            Decrementar λ
        Caso contrário:
            Incrementar λ
Até número máximo de épocas pré-determinado ou erro mínimo satisfeito
retornar: Matriz de pesos $\mathbf{W}$ treinado

2.3 Máquina de Boltzmann Restrita

Máquinas de Boltzmann restrita (SMOLENSKY, 1986; HINTON, 2002) são redes estocásticas compostas por duas camadas, uma visível e uma oculta. Ela é baseada no modelo estatístico de energia e é capaz de funcionar como um extrator de características (HINTON, 2014). Uma RBM não relaciona padrões de entrada e de saída de um conjunto de treinamento, possuindo apenas um conjunto de entradas, assim seu treinamento é feito de modo não supervisionado. Além disso, RBMs podem ser utilizadas como filtros, reconstrutores de sinais ou análise de componentes principais (do inglês: *Principal Component Analysis* (PCA)) (BU et al., 2015).

Uma RBM possui uma camada visível com m neurônios e uma camada oculta com n neurônios, sendo que cada neurônio da camada visível está conectado a todos os neurônios da camada oculta, não existindo conexão entre neurônios da mesma camada, por esse motivo ela é restrita. Durante o treinamento da rede, os neurônios da camada oculta aprendem a modelar a distribuição de probabilidade da camada visível. Um exemplo de RBM é mostrado na figura 3. Nesse caso, $\mathbf{W}$ é a matriz $m \times n$ de pesos das conexões entre as camadas, $\mathbf{a}$ o vetor *bias* referente à camada visível, $\mathbf{b}$ o vetor *bias* referente à camada oculta, $\mathbf{v}$ o conjunto dos neurônios visíveis e $\mathbf{h}$ o conjunto de neurônios ocultos. O conjunto de parâmetros da rede $(\mathbf{W}, \mathbf{a}, \mathbf{b})$ será denominado $\boldsymbol{\theta}$.

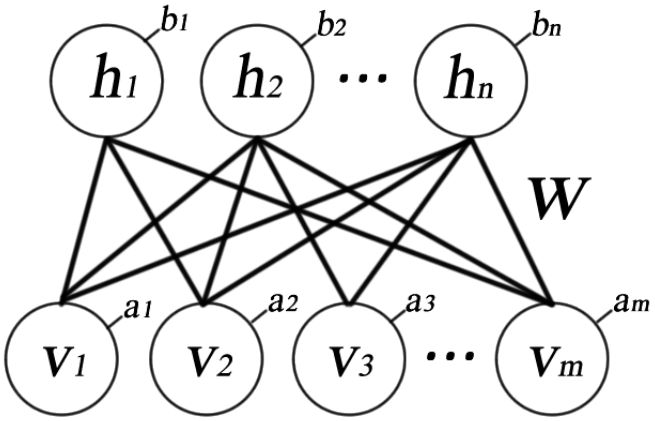


Figura 3 – Máquina de Boltzmann restrita, retirada de Pacheco (2016a)

A distribuição de probablidade conjunta da configuração de uma RBM $p(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta})$ é calculada pela equação 2.5:

$$p(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta}) = \frac{e^{-E(\mathbf{v},\mathbf{h};\boldsymbol{\theta})}}{\sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h};\boldsymbol{\theta})}} \tag{2.5}$$

Originalmente as RBMs foram desenvolvidas para dados binários, ou seja, seus valores devem ser 0 ou 1. O modelo de energia para rede binária é descrita pela equação 2.6:

$$E(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta}) = -\sum_{i=1}^{m} a_i v_i \ -\sum_{j=1}^{n} b_j h_j \ -\sum_{i=1}^{m}\sum_{j=1}^{n} v_i w_{ij} h_j \tag{2.6}$$

Deste modelo, pode-se determinar as probabilidades condicionais de um neurônio visível a partir dos neurônios ocultos e vice-versa, como é descrito pelas equações 2.7 e 2.8:

$$P(v_i = 1|\mathbf{h}; \boldsymbol{\theta}) = \ \phi(a_i + \sum_{j=1}^{n} w_{ij} h_j) \tag{2.7}$$

$$P(h_j = 1|\mathbf{v}; \boldsymbol{\theta}) = \ \phi(b_j + \sum_{i=1}^{m} w_{ij} v_i) \tag{2.8}$$

onde ϕ é a função de ativação do neurônio.

Uma modificação muito utilizada de RBMs são as *Gaussian-Bernoulli* RBMs, que possuem camada visível contínua e camada oculta binária. Nesse caso a energia da rede é descrita pela equação 2.9 e a probabilidade condicional da camada visível é descrita pela equação 2.10.

$$E(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta}) = -\sum_{i=1}^{m} \frac{(v_i - a_i)^2}{2\sigma_i^2} \ -\sum_{j=1}^{n} b_j h_j \ -\sum_{i=1}^{m}\sum_{j=1}^{n} \frac{v_i}{\sigma_i^2} w_{ij} h_j \tag{2.9}$$

$$P(v_i = 1|\mathbf{h}; \boldsymbol{\theta}) = \ N(v|a_i + \sum_{j=1}^{n} w_{ij} h_j, \sigma^2) \tag{2.10}$$

onde N é uma distribuição normal com média v e desvio padrão σ, normalmente utilizado como 1.

O treinamento de uma RBM consiste em ajustar os parâmetros de $\boldsymbol{\theta}$ de forma a minimizar a energia da rede (HINTON, 2010). Um meio de estimar $\boldsymbol{\theta}$ é a partir da maximização de $p(\mathbf{v};\boldsymbol{\theta})$, que é a distribuição de probabilidade dos dados de entrada, ou da mesma forma da maximização de $\log p(\mathbf{v};\boldsymbol{\theta})$ que poderia ser estimado por meio do gradiente descendente. Infelizmente esse método é muito difícil de ser calculado e demanda muito tempo.

O método para treinamento de RBMs denominado *contrastive divergence*, (CD) proposto por Hinton (2002) é muito mais rápido. O passo básico desse algoritmo consiste em alimentar a RBM com uma entrada de treino e reconstruir essa entrada por meio de um amostrador de Gibbs apenas uma vez como mostrado na figura 4. A partir dos resultados obtidos dessa amostragem, $\boldsymbol{\theta}$ é atualizado. Isso é repetido um número predefinido de vezes, conhecido como épocas, ou até que um erro mínimo seja alcançado. O pseudo-código CD é descrito no algoritmo 2.

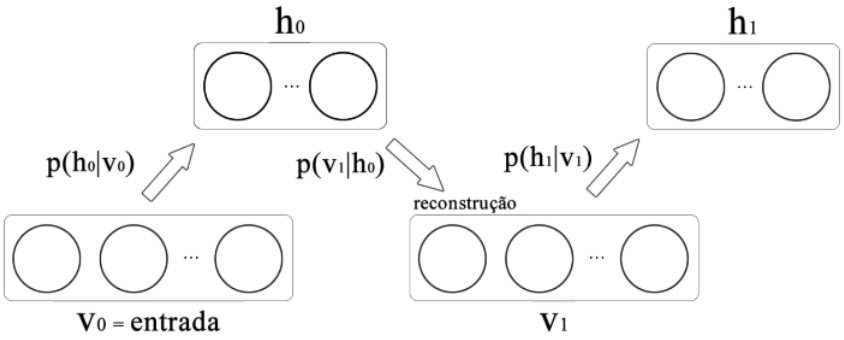


Figura 4 – Processo de reconstrução do algoritmo CD para RBM, retirada de Pacheco (2016a)

As equações para atualizar $\boldsymbol{\theta}$ são descritas por 2.11, 2.12 e 2.13:

$$\mathbf{W}^{t+1} = \mathbf{W}^t + \Delta\mathbf{W}^t \rightarrow \Delta\mathbf{W}^t = \eta(\mathbf{v}_0\mathbf{h}_0^T - \mathbf{v}_1\mathbf{h}_1^T) \ - \lambda\mathbf{W}^t + \alpha\Delta\mathbf{W}^{t-1} \tag{2.11}$$

$$\mathbf{a}^{t+1} = \mathbf{a}^t + \Delta\mathbf{a}^t \rightarrow \Delta\mathbf{a}^t = \eta(\mathbf{v}_0 - \mathbf{v}_1) + \alpha\Delta\mathbf{a}^{t-1} \tag{2.12}$$

$$\mathbf{b}^{t+1} = \mathbf{b}^t + \Delta\mathbf{b}^t \rightarrow \Delta\mathbf{b}^t = \eta(\mathbf{h}_0 - \mathbf{h}_1) + \alpha\Delta\mathbf{b}^{t-1} \tag{2.13}$$

Os parâmetros η, λ e α são conhecidos, respectivamente, como taxa de aprendizado, fator de decaimento e *momentum*. Recomenda-se o uso dos valores destes parâmetros como $\eta = 0.01$, $\lambda = [0.01, 0.0001]$ e $\alpha = 0.5$ antes da quinta iteração e $\alpha = 0.9$ após a mesma (HINTON, 2010).

Algoritmo 2: *Treinamento RBM*

inicialização:
    Preparar conjunto de dados de entrada
    Informar número de neurônios na camada oculta $\mathbf{h}$
    Inicializar de maneira aleatória $\boldsymbol{\theta}$
repita:
    Igualar a camada visível $\mathbf{v}_0$ aos dados de entrada
    Estimar $\mathbf{h}_0$ através da equação 2.8
    A partir de $\mathbf{h}_0$ estimar $\mathbf{v}_1$ utilizando a equação 2.7
    A partir de $\mathbf{v}_1$ estimar $\mathbf{h}_1$ utilizando a equação 2.8 ou equação 2.10 caso GBRBM
    Atualizar os parâmetros de $\boldsymbol{\theta}$ através das equações 2.11, 2.12 e 2.13
Até número de épocas pré-determinado ou erro mínimo satisfeito
retornar: $\boldsymbol{\theta}$ treinado

Um possível problema da RBM, é o seu treinamento usar um algoritmo baseado em gradiente descendente, o que pode implicar na queda em mínimos locais. Neste trabalho foi desenvolvido um método de treinamento alternativo, usando a otimização por enxame de partículas, que é descrita a seguir.

2.4 Otimização por enxame de partículas

A otimização por enxame de partículas é um algoritmo populacional inspirado na interação entre agentes sociais entre si e com o meio em que se encontram. Exemplos desse comportamento na natureza são cardumes de peixes, bandos de pássaros e enxames de abelhas.

O PSO foi proposto por Eberhart e Kennedy (1995) e tem como objetivo buscar a solução ótima de uma função objetivo em um espaço de busca através da troca de informações entre os indivíduos da população, determinando a trajetória que cada um deve tomar. No PSO cada partícula é uma possível solução para o problema, e pode ser entendida como um indivíduo em uma população, uma analogia são os pássaros em um bando. Esses pássaros exploram uma região, que é determinada pela função objetivo a fim de encontrar a solução ótima para o problema, como mostrado na figura 5.

As partículas são inicializadas com posições e velocidades aleatórias no espaço do problema a ser otimizado. A cada iteração, cada partícula se move nesse espaço, baseando-se na melhor posição pela qual essa partícula já passou e a melhor posição conhecida globalmente. O PSO possui duas topologias básicas que denotam como as partículas se comunicam: a topologia global e a topologia local, mostradado na figura 6.

Na topologia global, cada partícula comunica o melhor valor encontrado com todas as demais, resultando em um valor global único para todas elas. Já na topologia local, cada

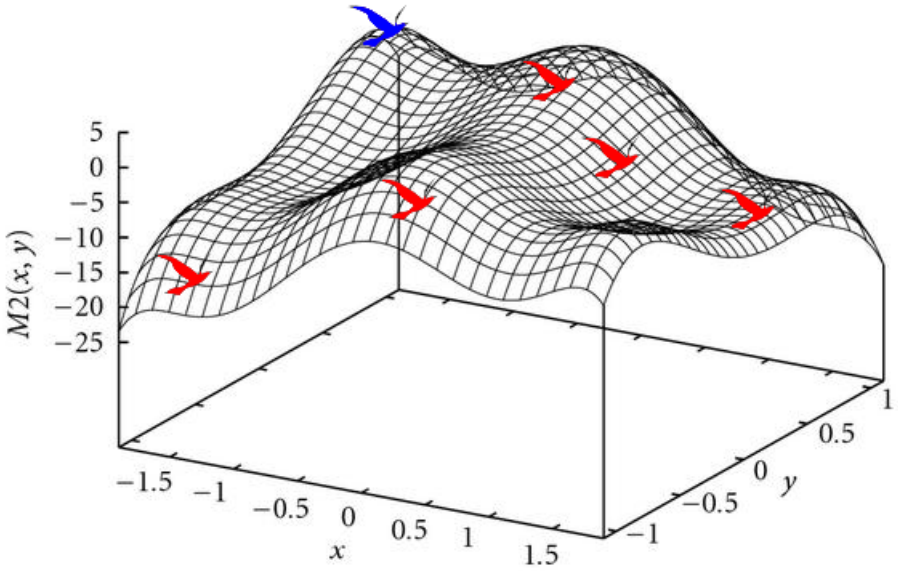


Figura 5 – Analogia de um bando de pássaros em um espaço de busca de 3D. Considerando um problema de maximização, o pássaro azul representa a partícula de solução ótima do problema. Retirado de Pacheco (2016b)

partícula só se comunica com seus vizinhos, o que resulta em cada uma ter seu próprio "ótimo da vizinhança" conhecido.

Essas topologias influenciam o tempo de convergência do algoritmo e na sua suscetibilidade à queda em ótimos locais. Na topologia global, por existir apenas um ótimo global, as partículas tendem a se aproximar desse ótimo resultando em uma convergência mais rápida, porém com o risco de cair em um ótimo local. Já na topologia local, pelo retardo da informação (caso um ótimo seja encontrado ele tem que ser passado por todas as partículas uma a uma) as partículas tendem a se deslocar mais, o que resulta em um maior tempo de convergência, com o benefício de ter mais resistência a ótimos locais.

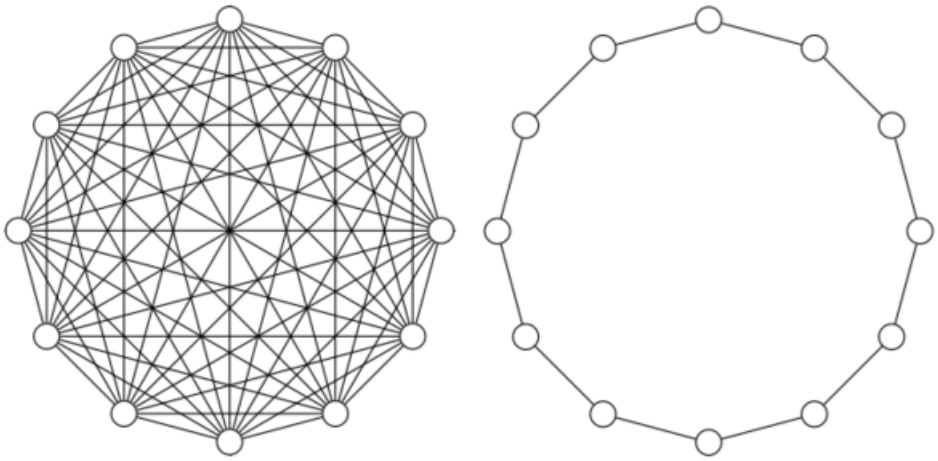
Dada uma população com N partículas, as equações para a atualização da velocidade e da posição, respectivamente, de uma partícula i em uma iteração são descritas por 2.14 e 2.15.

$$v_i(t+1) = wv_i(t) + c_1r_1(p_{best_i} - x_i) + c_2r_2(g_{best_i} - x_i) \tag{2.14}$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{2.15}$$

onde:

- $v_i(t)$: velocidade da partícula i na iteração t



(a) Topologia Global (b) Topologia Local

Figura 6 – Topologias do PSO.

- $x_i(t)$: posição da partícula i na iteração t
- w: coeficiente de inércia
- p_{best_i}: melhor posição pela qual a partícula i passou
- g_{best}: melhor posição global
- c_1 e c_2: constantes de aceleração referentes às melhores posições local e global, respectivamente
- r_1 e r_2: números aleatórios extraidos de uma distribuição de probabilidade uniforme no intervalo $[0, 1]$

As condições de paradas normalmente utilizadas no PSO são um número máximo de iterações $maxiter$ e um número de iterações em que não ocorre atualização do ótimo global $nmax$. O pseudo-código do PSO é descrito no algoritmo 3:

Algoritmo 3: *Algoritmo PSO*

inicialização:
    Inicializar as constantes c_1, c_2, r_1 e r_2 e o tamanho da população N do PSO
    Inicializar aleatoriamente a população de tamanho N
repita:
    para cada partícula i:
        Atualizar v_i de acordo com a equação 2.14
        Atualizar x_i de acordo com a equação 2.15
        Calcular a aptidão da partícula i pela função objetivo
        Atualizar p_{best_i} (melhor posição da partícula) se a aptidão calculada for melhor
        Atualizar g_{best_i} (melhor posição da população) se a aptidão calculada for melhor
até Condições de parada serem atendidas
retornar: g_{best}

3 Treinamento de máquinas de Boltzmann restrita via otimização por enxame de partículas

Técnicas de otimização baseadas em derivadas e gradientes possuem uma propensão a cair em mínimos locais. Diversas meta-heurísticas com dinâmica de população foram desenvolvidas, como a otimização por enxame de partículas e algoritmos genéticos, que fazem uma busca global das soluções diminuindo assim as chances de quedas em mínimos locais (OJHA; ABRAHAM; SNÁŠEL, 2017).

Como o *contrastive divergence* é um algoritmo baseado em gradientes, neste trabalho foi desenvolvido um treinamento para RBM utilizando meta-heurísticas com o objetivo de investigar o seu comportamento. Foi escolhido o PSO com topologia local para o treinamento, pois é um algoritmo que apresenta bons resultados (MENDEL; KROHLING; CAMPOS, 2011). O tipo de RBM utilizada foi uma Gaussian-Bernoulli, pois como sua entrada é contínua, esta pode ser usada para uma gama maior de problemas.

Para o uso do PSO, é necessário definir a função objetivo a ser otimizada. No caso de uma RBM essa função deve ser escolhida de forma que a RBM treinada possua características semelhantes a uma RBM treinada pelo *contrastive divergence*. Como durante o algoritmo CD é utilizado um amostrador de Gibbs para executar uma reconstrução da rede, e a RBM tem como característica o aprendizado da distribuição de probabilidade das entradas, a função objetivo escolhida foi a raiz quadrada do erro quadrático médio (RMSE) da reconstrução.

Para a adquirir esse erro, as entradas passam por um passo do amostrador de Gibbs como mostrado na figura 4. O cálculo desse erro é descrito pela equação 3.1.

$$f(x) = \sqrt{\frac{\sum_{i=1}^{n} (v_{i_0} - v_{i_1})^2}{n}} \tag{3.1}$$

onde $f(x)$ é a função objetivo, n é o número de entradas de treinamento da rede, v_{i_0} a i-ésima entrada de treinamento, v_{i_1} é a reconstrução pelo amostrador de Gibbs da entrada v_{i_0}.

Para o treinamento da RBM os *bias* e os pesos das conexões das redes são passados para o PSO para serem otimizados. No processo de otimização, o PSO envia os parâmetros para a RBM para calcular a sua função objetivo. Quando finalizado, o processo de otimização devolve à rede os pesos e os *bias* otimizados. Na figura 7 é ilustrado o procedimento

de treino.

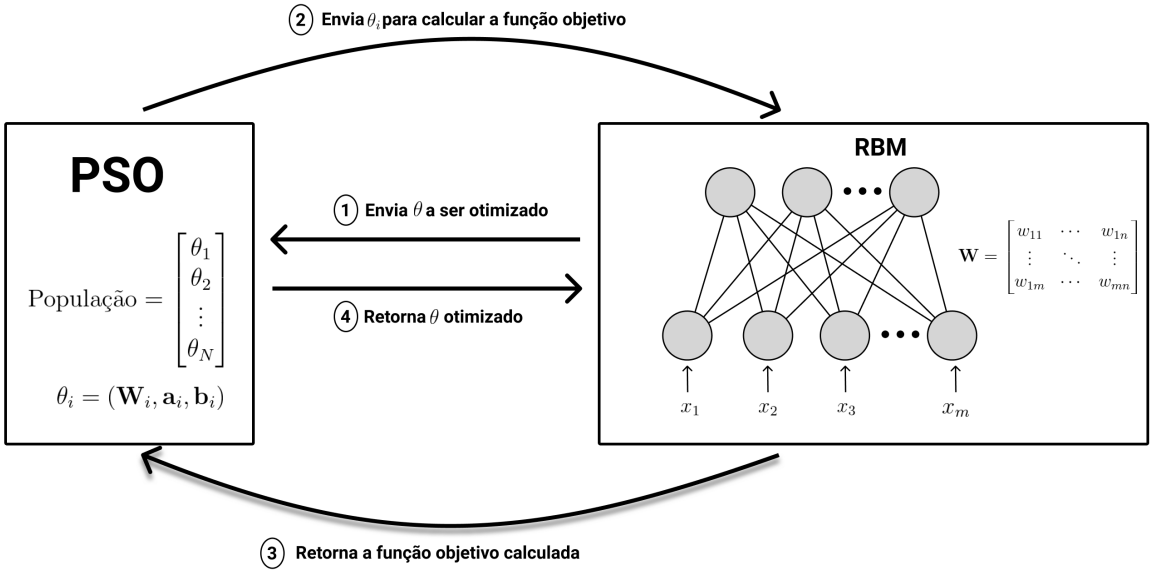


Figura 7 – Treinamento da RBM utilizando PSO. Na etapa 1 os parâmetros são enviados ao PSO para serem otimizados. Na etapa 2 o PSO envia os parâmetros para a RBM para calcular o erro de reconstrução, que é devolvido na etapa 3. Na etapa 4 o PSO retorna os parâmetris otimizados para a RBM

3.1 RBM como um estágio de pré-processamento para classificação de dados

Para realizar a classificação de dados, nesse trabalho a RBM foi utilizada como um estágio de pré-processamento dos dados. Neste procedimento, primeiro a RBM é treinada como na seção anterior. Em seguida, é calculada a camada escondida para todos os dados de treinamento. O resultado dessa amostragem é usado como entrada para o treinamento da rede *feedforward*, que fará a classificação dos dados. Na etapa de teste, mais uma vez os dados são amostrados pela RBM, e então, passam pela rede *feedforward* para realizar a classificação, como mostrado na figura 8.

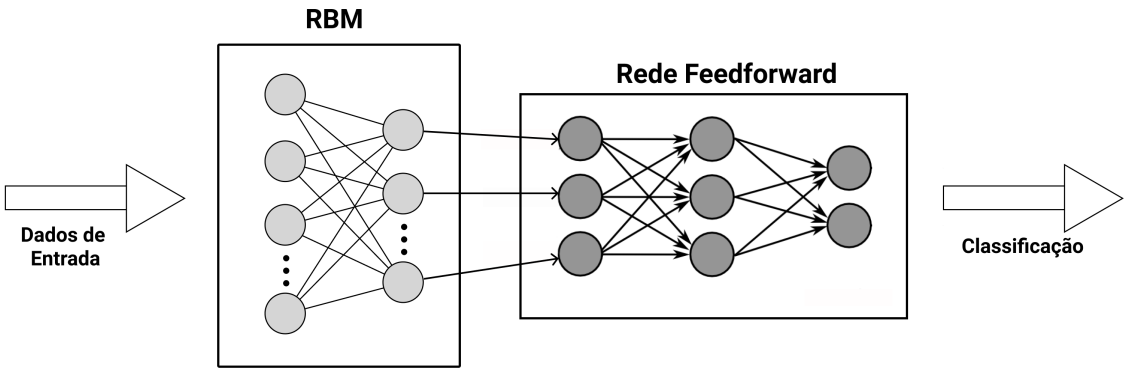


Figura 8 – Classificação de dados com RBM e rede *feedforward*

4 Resultados Experimentais

Neste capítulo serão apresentados os resultados experimentais obtidos neste trabalho. Na seção 4.1 são descritas as bases utilizadas, na seção 4.2 é realizado um estudo sobre a convergência no processo de treinamento da RBM com o *contrastive divergence* e PSO, e na seção 4.3 são discutidos os experimentos para classificação de dados e seus resultados. Todos os esperimentos realizados nesse trabalho foram executados em um computador com processador *Intel(R) Core(TM) i7-490* CPU @ 3.60 GHz com 16 Gb de memória *RAM* com Sistema Operacional *Windows 10 Pro 64bits* e utilizando o ambiente de desenvolvimento MATLAB 2015.

4.1 Bases de Dados

Nesse experimento foram utilizadas oito bases de dados retiradas do *UCI Machine Learning Repository* (LICHMAN, 2013), cujas características de número de amostras, atributos e rótulos são apresentadas na tabela 1.

- A base *Iris* é uma das bases mais conhecidas na literatura. Ela representa medições em plantas do tipo Iris que são distribuidas em três classes com 50 elementos cada, onde cada classe representa um tipo dessa planta.
- A base *Wine* representa a análise química de 13 componentes em três tipos de vinhos, com as classes bem distribuidas.
- A base *seeds* representa medições em sementes de 3 variedades de trigo, com 70 amostras cada.
- A base *sonar* representa padrões obtidos ao enviar ondas de sonar em diversos ângulos e frequências a objetos cilíndricos (representando minas) e a rochas. Os valores representam a energia de cada padrão de frequência.
- A base *column* representa atributos da coluna vertebral de pacientes, usados para determinar se há algum problema com o mesmo.
- A base *Indian Liver Patient Dataset (ILPD)* representa uma base para determinar se um paciente possui alguma doença no fígado. Nessa base, 71% dos dados pertencem a uma classe, e 29% à outra.
- A base *Credit Australia* representa uma base para aprovação de crédito. Essa base possui uma mescla de atributos contínuos, nominais com poucos valores e nominais com muitos valores.

- A base *Cancer* indica características em amostras de câncer de mama.

Base de Dados	Número de amostras	Número de atributos	Número de rótulos
Cancer	699	9	2
Column	310	6	2
Credit Australia	690	14	2
Iris	150	4	3
ILPD	583	10	2
Seeds	210	7	3
Sonar	208	60	2
Wine	178	13	3

Tabela 1 – Atributos das bases de dados utilizadas

4.2 Estudo de Convergência

Foi realizado um estudo da convergência das RBMs, com o treinamento via PSO, a fim de verificar o comportamento do RMSE da reconstrução calculado com o passar das iterações do treinamento. Nesse estudo as redes foram executadas para um número de neurônios na camada escondida variando entre 10 e 50, e um número máximo de iterações do algoritmo de 50, 100, 200, 300, 400 e 500. Para cada configuração, o teste foi repetido 30 vezes a fim de se realizar cálculos estatísticos. Este procedimento foi repetido para todas as bases de dados da seção 4.1. O treinamento é realizado com 70% dos dados, e os restantes são usados para testes. Para a separação dos dados, eles são primeiro embaralhados, e então são separadas as duas partições, que são utilizadas em todos os experimentos. Isso é feito para todas as bases.

O PSO foi utilizado com uma população de 30 partículas. O mesmo se comporta bem com populações entre 20 e 100 indivíduos, não havendo grande diferença de desempenho entre elas (BRATTON; KENNEDY, 2007). As constantes de aceleração c_1 e c_2 foram utilizadas com valor 2.05 cada e o coeficiente de inércia w com valor 0.729, que são os valores padrão utilizados no PSO (MENDEL; KROHLING; CAMPOS, 2011). Nesse experimento apenas o número de iterações foi utilizado como critério de parada.

Desse experimento, os melhores resultados de RMSE, para cada base, são apresentados na tabela 2. No anexo A são apresentados os resultados completos para esse estudo. Na figura 9 são apresentados os gráficos de convergência do treinamento com o PSO, para cada configuração de neurônios escondidos. Nessa figura, os gráficos da esquerda mostram todas as iterações, porém limitando o valor do RMSE em 2, enquanto os gráficos da direita mostram apenas as 50 primeiras iterações, mas com o valor completo do RMSE, para uma melhor análise do comportamento inicial.

Base	**Neurônios na camada escondida**	**Número de iterações**	**RMSE de reconstrução**
Iris	10	500	0.424 ± 0.0384
Seeds	10	500	0.506 ± 0.0389
Column	10	500	0.395 ± 0.0447
Wine	10	500	0.613 ± 0.0477
ILPD	10	500	0.513 ± 0.0338
Cancer	10	500	0.502 ± 0.0297
Credit Australia	10	500	0.594 ± 0.0339
Sonar	10	500	0.445 ± 0.0640

Tabela 2 – Resultados de RMSE com o PSO

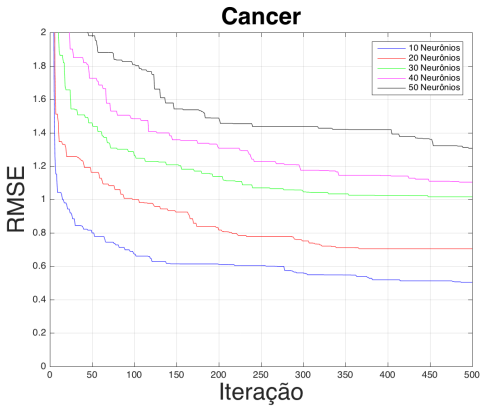
A fim de comparação, o mesmo experimento foi realizado com uma RBM tradicional. Na tabela 3 é apresentado uma comparação dos melhores valores de RMSE obtidos pelos dois métodos. Na tabela 4 é apresentada uma comparação do tempo de treinamento entre os resultados, e na figura 10 é mostrado um comparativo da convergência do RMSE com redes treinadas via PSO e via CD.

Base	**PSO-RBM RMSE de reconstrução**	**CD-RBM RMSE de reconstrução**
Iris	0.424 ± 0.0384	$\mathbf{0.307 \pm 0.0140}$
Seeds	0.506 ± 0.0389	$\mathbf{0.286 \pm 0.0135}$
Column	0.395 ± 0.0447	$\mathbf{0.221 \pm 0.0092}$
Wine	0.613 ± 0.0477	$\mathbf{0.246 \pm 0.0054}$
ILPD	0.513 ± 0.0338	$\mathbf{0.238 \pm 0.0050}$
Cancer	0.502 ± 0.0297	$\mathbf{0.312 \pm 0.0056}$
Credit Australia	0.594 ± 0.0339	$\mathbf{0.330 \pm 0.0040}$
Sonar	0.445 ± 0.0640	$\mathbf{0.235 \pm 0.0037}$

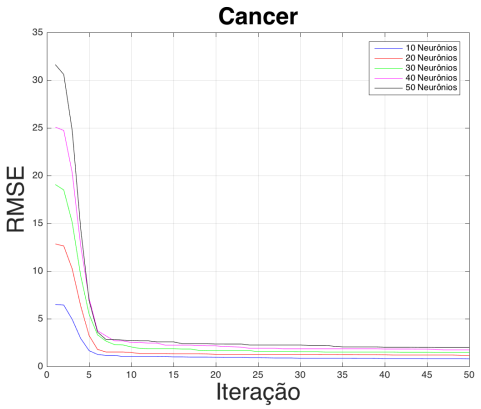
Tabela 3 – Comparação de RMSE entre RBMs treinadas via PSO e via CD

Base	**PSO-RBM tempo em seg**	**CD-RBM tempo em seg**
Iris	5.99	**0.09**
Seeds	13.95	**0.11**
Column	13.54	**0.03**
Wine	21.64	**0.08**
ILPD	26.36	**0.13**
Cancer	23.62	**0.28**
Credit Australia	26.99	**0.23**
Sonar	78.18	**0.25**

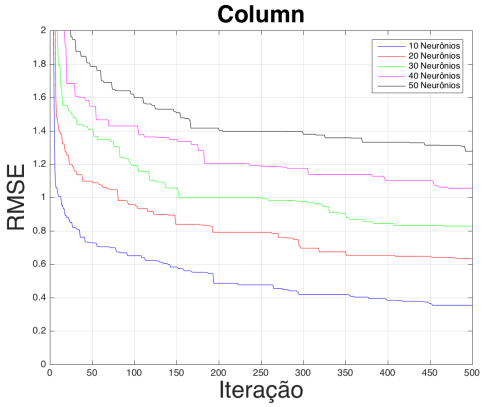
Tabela 4 – Tempos de treinamente para redes selecionadas



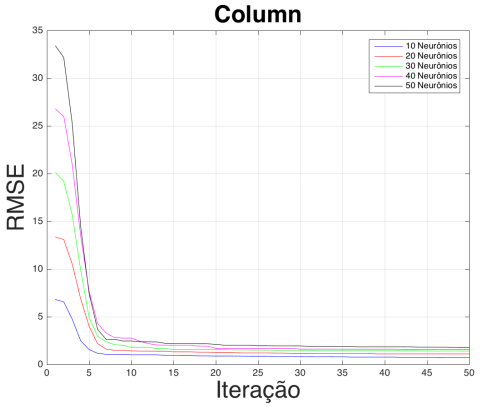
(a) Base *Cancer* 500 iter.



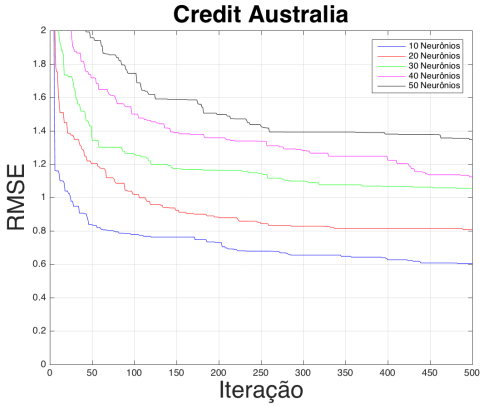
(b) Base *Cancer* 50 iter.



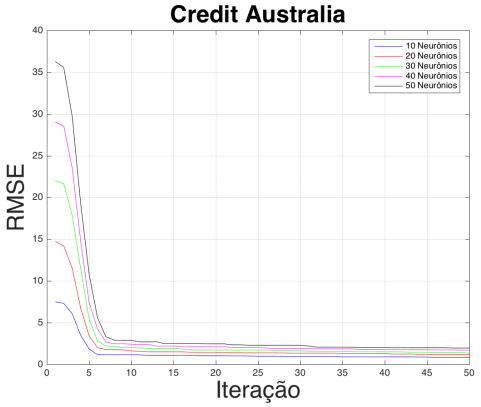
(c) Base *Column* 500 iter.



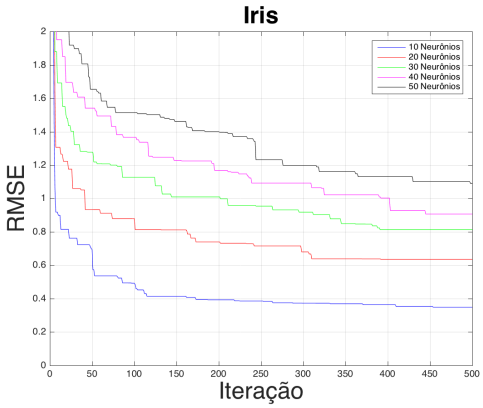
(d) Base *Column* 50 iter.



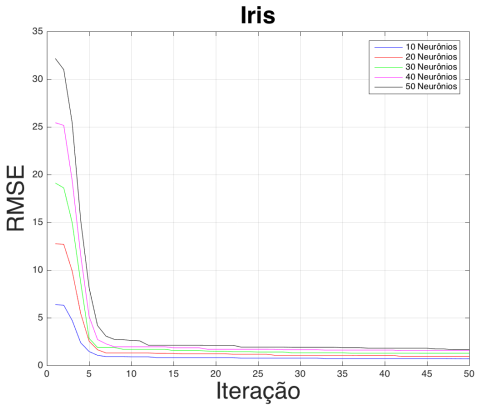
(e) Base *Credit Australia* 500 iter.



(f) Base *Credit Australia* 50 iter.

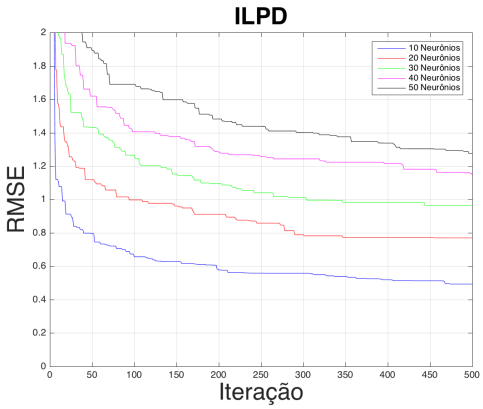


(g) Base *Iris* 500 iter.

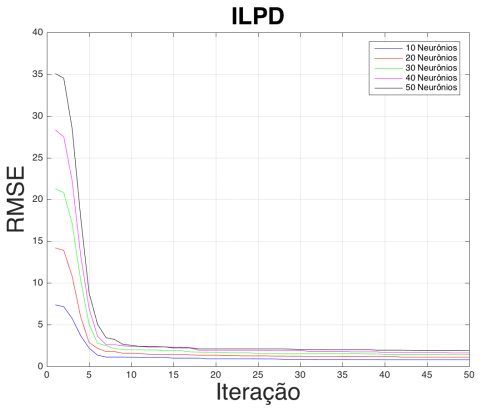


(h) Base *Iris* 50 iter.

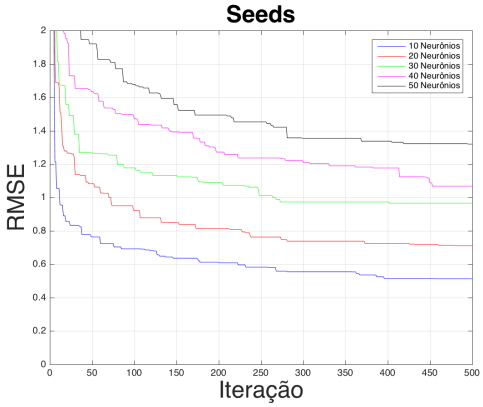
Figura 9 – Estudo da Convergência com PSO



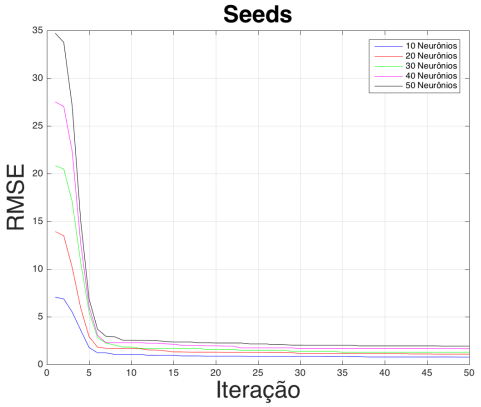
(i) Base *ILPD* 500 iter.



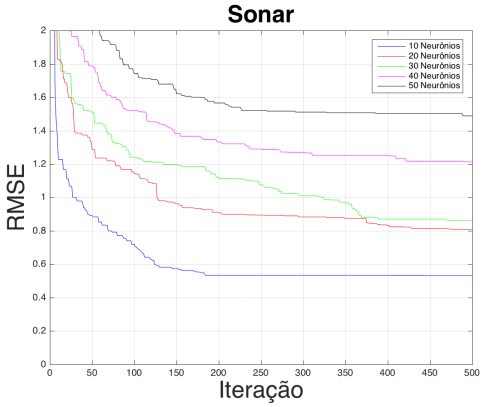
(j) Base *ILPD* 50 iter.



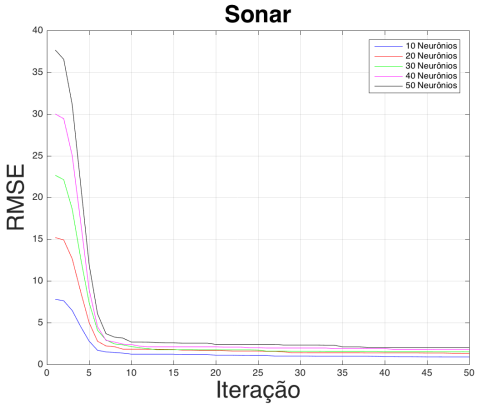
(k) Base *Seeds* 500 iter.



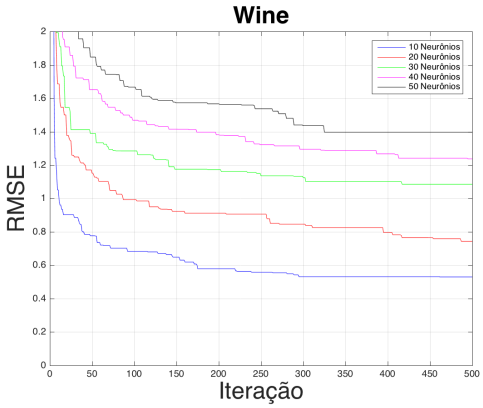
(l) Base *Seeds* 50 iter.



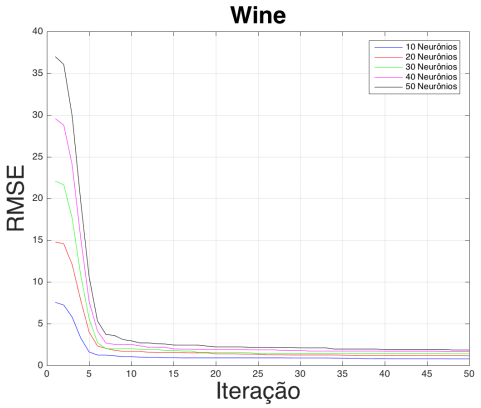
(m) Base *Sonar* 500 iter.



(n) Base *Sonar* 50 iter.

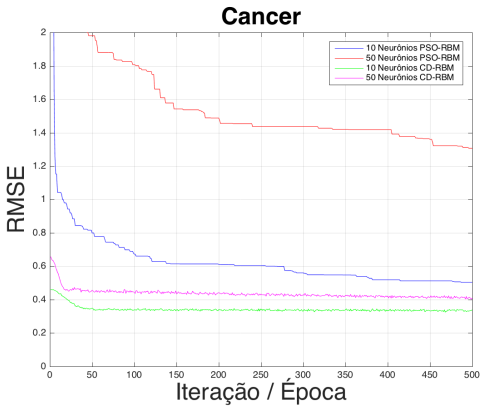


(o) Base *Wine* 500 iter.

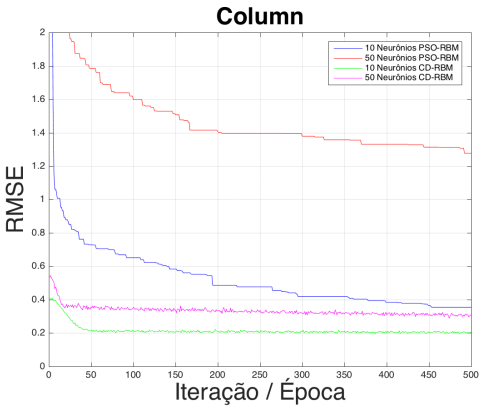


(p) Base *Wine* 50 iter.

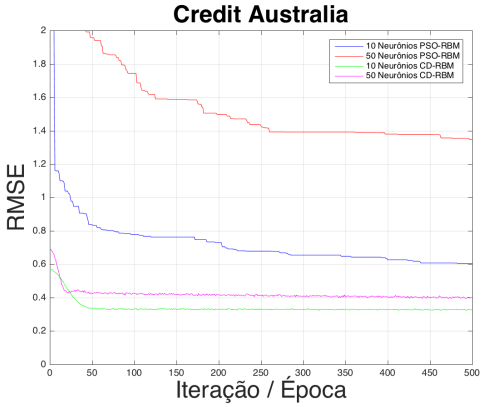
Figura 9 – Estudo da Convergência com PSO (cont.)



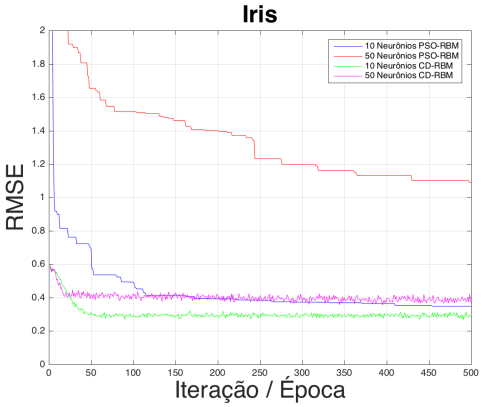
(a) Base *Cancer*



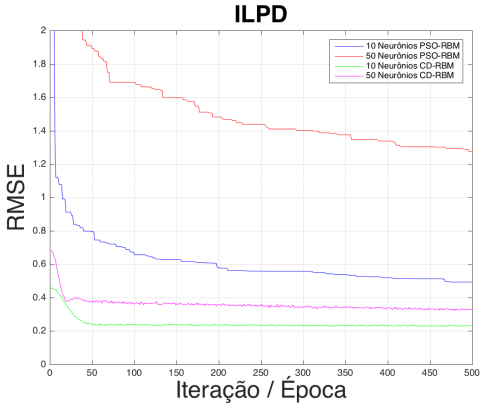
(b) Base *Column*



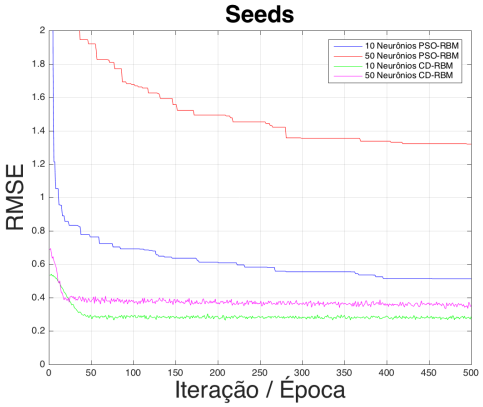
(c) Base *Credit Australia*



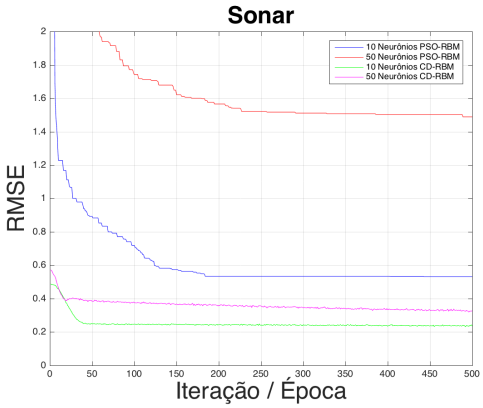
(d) Base *Iris*



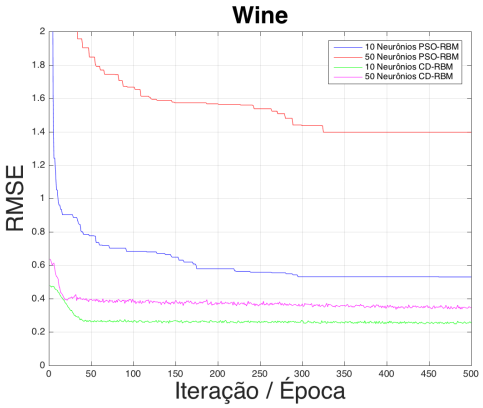
(e) Base *ILPD*



(f) Base *Seeds*



(g) Base *Sonar*



(h) Base *Wine*

Figura 10 – Comparativo de convergência entre RBMs treinadas com PSO e com RBM

Neste estudo, pode se facilmente observar que, para todas as bases, a rede treinada com o *contrastive divergence* apresenta menor valor de RMSE e tempo de treinamento do que as redes treinadas com o PSO. Outro aspecto que pode ser notado é que o RMSE aumenta com o aumento do número de neurônios na camada escondida. Esse comportamento é ainda mais acentuado nas redes treinadas pelo PSO que apresenta uma grande variação no seu RMSE. Um fator também importante é o tempo de treinamento que é significativamente maior quando o PSO é utilizado. Isto se dá pelo problema se tornar cada vez mais complexo, com mais variáveis a serem otimizadas.

Na figura 9, pode-se notar que o PSO inicia com valores muito altos de RMSE, mas com uma descida rápida na curva, porém, após cerca de 10 iterações apresenta uma convergência mais lenta que tende a estabilizar após aproximadamente 250 iterações. Pode se também observar de forma clara a constatação anterior, de que redes com mais neurônios na camada escondida possuem um maior RMSE.

Analisando a figura 10, pode-se observar que as redes treinadas via CD tendem a convergir mais rápido, estabilizando entre 50 e 100 épocas, e com um menor RMSE que as redes treinadas via PSO.

4.3 Classificação de dados

O principal experimento realizado nesse trabalho foi o uso da RBM como um estágio de pré-processamento, para a da classificação de dados com uma rede neural, como descrito na seção 3.1.

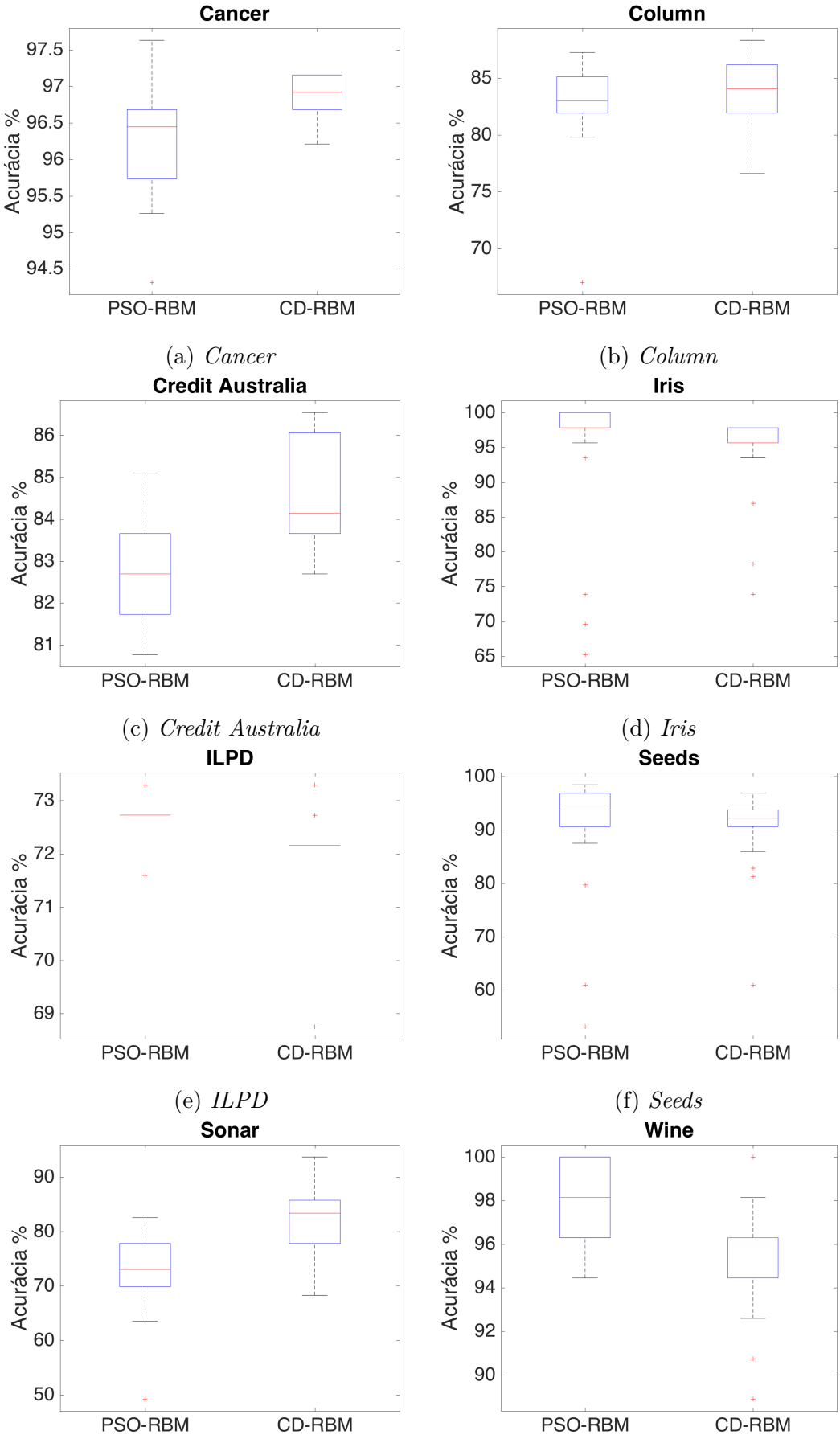
Nesse experimento foi realizado um estudo de sensibilidade, variando o número de neurônios na camada escondida da RBM entre 10 e 50, e da camada intermediária da *feedforward* também entre 10 e 50. No experimento, foram utilizadas as bases descritas na seção 4.1. Os dados foram também separados em dois conjuntos com 70% dos dados para treinamento e os 30% restantes para testes.

Para o PSO, foram utilizados os mesmos parâmetros da seção 4.2, e como critério de parada foram utilizados um número máximo de 300 iterações e 10 iterações sem mudança de *gbest*. Para a rede *feedforward*, foi utilizada a *toolbox* do MATLAB de redes neurais, com treinamento via LMBP, μ com valor inicial de 0.1 e número máximo de épocas 1000.

O mesmo experimento foi também executado com RBMs utilizando o treinamento via *contrastive divergence*. Os melhores resultados para cada rede são apresentados na tabela 5 e ilustrados na figura 11 em forma de boxplots. Os resultados completos para o teste encontram-se no anexo B.

Base	**Taxa de acerto PSO-RBM**	**Taxa de acerto CD-RBM**
Iris	$\mathbf{94.64 \pm 10.63}$	94.57 ± 5.47
Seeds	91.15 ± 10.09	$\mathbf{91.20 \pm 6.80}$
Column	83.01 ± 3.51	$\mathbf{83.62 \pm 3.09}$
Wine	$\mathbf{98.09 \pm 1.57}$	95.37 ± 2.27
ILPD	$\mathbf{72.75 \pm 0.28}$	72.10 ± 0.67
Cancer	96.30 ± 0.77	$\mathbf{96.87 \pm 0.32}$
Credit Australia	82.77 ± 1.32	$\mathbf{84.52 \pm 1.27}$
Sonar	73.17 ± 6.76	$\mathbf{82.06 \pm 5.76}$

Tabela 5 – Melhores resultados de classificação com treinamento pelo PSO



(a) *Cancer* (b) *Column*

(c) *Credit Australia* (d) *Iris*

(e) *ILPD* (f) *Seeds*

(g) *Sonar* (h) *Wine*

Figura 11 – Boxplots dos testes realizados

4.4 Análise dos Resultados

No experimento de classificação de dados, a rede treinada com o PSO possui melhor resultado para as bases *Iris*, *Wine* e *ILPD* enquanto a rede treinada com o CD foi melhor para as outras bases. Exceto para a base *Cancer*, a diferença de desempenho é muito pequena, não passando de 2.5%. Na base *Cancer*, porém, a diferença de acurácia é de aproximadamente 9%. Quanto ao desvio padrão, a RBM treinada com PSO obteve melhores resultados para as bases *Wine* e *ILPD*, apenas, o que indica uma maior robustez na rede treinada pelo CD.

Na figura 11 pode-se observar que há uma maior dispersão dos dados ao redor da mediana nos boxplots referentes às redes treinadas com o PSO nas bases *Cancer*, *Seeds* e *Wine*, enquanto que, para as bases *Column* e *Credit Australia*, isso ocorre para treinamento com o CD.

Algo que pode ser notado foi que, embora os melhores valores de RMSE foram todos para redes com 10 neurônios na camada escondida, na maioria dos casos essas redes não foram as que obtiveram melhor acurácia.

Como pode ser também observado na tabela 4, uma desvantagem do treinamento pelo PSO é o seu elevado tempo de convergência. Esse tempo de convergência é comum para meta-heurísticas, pelo fato de sua busca global e dinâmica de população.

Para comparar os algoritmos, foi utilizado o teste não paramétrico de Friedman com o 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 $pvalor = 0.05$, no qual, se o teste retornar um valor menor do que este há evidência estatística que um dos algoritmos é significativamente diferente (DERRAC et al., 2011). Para esse experimento o teste de Friedman obteve $p = 0.4795$, o que mostra que não há uma diferença estatística entre os métodos.

Foi realizado um estudo com o algoritmo A-TOPSIS (KROHLING; PACHECO, 2015) para fazer o ranqueamento dos algoritmos, baseado na acurácia. Nesse estudo, o peso da média foi variado de 0.5 a 1, e do desvio padrão de 0.5 a 0. A tabela 6 mostra os rankings obtidos. O algoritmo A-TOPSIS é descrito no anexo C.

Analisando a tabela 6, observa-se que a RBM treinada pelo *contrastive divergence* foi ranqueada em primeiro para todas as combinações de peso para média e desvio padrão, o que indica que o treinamento pelo CD foi melhor nesse experimento.

Um comportamento estranho foi observado para a base *ILPD*, onde as taxas de acertos para a ambos os treinamentos da RBM se mantiveram, para todas as configurações de neurônios próximas a 72%. Além disso, constata-se também, na figura 11, que para essa base, há pouca dispersão nos boxplots de ambos os treinamentos. Como discutido na

		Ranking	
Média	**Desvio Padrão**	**CD-RBM**	**PSO-RBM**
0.5	**0.5**	1	2
0.6	**0.4**	1	2
0.7	**0.3**	1	2
0.8	**0.2**	1	2
0.9	**0.1**	1	2
1.0	**0.0**	1	2

Tabela 6 – Ranqueamento dos Algoritmos pelo A-TOPSIS

seção 4.1, a base *ILPD* é desbalanceada com 71% das entradas pertencendo a uma classe. Investigando assim esse comportamento, foi possível notar que para todas as configurações de redes, o classificador indicava que as entradas pertenciam à classe da maioria em grande parte dos casos, originando assim esse resultado.

5 Conclusão

Máquinas de Boltzmann restrita vem sendo muito empregadas em diversos trabalhos recentemente. Porém seu treinamento é derivado da técnica de gradiente descendente, que em problemas de otimização possui propensão à queda em mínimos locais. Nesse trabalho foi desenvolvido um treinamento para máquinas de Boltzmann restrita, utilizando como base a otimização via enxame de partículas como uma alternativa para o treinamento tradicional.

Foi realizado com a RBM com treinamento via PSO um estudo para analisar seu comportamento durante o treinamento, verificando a convergência do erro de reconstrução, e um experimento de classificação. Nos experimentos de classificação, a RBM treinada via PSO obteve resultados similares à treinada via CD, sem diferenças estatísticas segundo o teste de Friedman, porém, com um tempo de treinamento muito superior. Além disso, nos experimentos de convergência, a RBM treinada pelo PSO apresenta um maior erro de reconstrução.

Para as bases de dados e configurações de rede utilizadas nesse trabalho, o treinamento através do *contrastive divergence* se mostrou superior em todos os quesitos estudados. Possíveis trabalhos futuros são: o estudo com outras configurações ou modelos mais avançados de PSO, por exemplo o PSO com o uso de pulos caóticos, o estudo com treinamento utilizando computação paralela, a fim de obter tempos de treinamento mais viáveis, e o uso de outras funções objetivo para o treinamento.

Referências

AGGARWAL, C. C. *Data Classification: Algorithms and Applications*. 1st. ed. [S.l.]: Chapman & Hall/CRC, 2014.

BHARDWAJ, A. et al. A genetically optimized neural network model for multi-class classification. *Expert Systems with Applications*, v. 60, p. 211–221, 2016.

BRATTON, D.; KENNEDY, J. Defining a standard for particle swarm optimization. In: IEEE. *Swarm Intelligence Symposium, 2007. SIS 2007*. [S.l.], 2007. p. 120–127.

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

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*, Elsevier, v. 1, n. 1, p. 3–18, 2011.

EBERHART, R.; KENNEDY, J. Particle swarm optimization. *Proceedings of 1995 IEEE International Conference on Neural Networks*, p. 1942–1948, 1995.

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

GONÇALVES, E. C. *Novel classifier chain methods for multi-label classification based on genetic algorithms*. Tese (Doutorado) — Universidade Federal Fluminense, Niterói, 9 2015.

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.

HAYKIN, S. *Neural Networks: A comprehensive foundation, 3. ed.* New Jersey: Prentice Hall, 2007.

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

HINTON, G. Where do features come from? *Cognitive science*, v. 38, n. 6, p. 1078–1101, 2014.

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.

HU, Y. et al. Dense crowd counting from still images with convolutional neural networks. *Journal of Visual Communication and Image Representation*, v. 38, p. 530–539, 2016.

HWANG, C.-L.; YOON, K. *Multiple attributes decision making methods and applications*. Berlin: Springer-Verlag, 1981.

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

KUREMOTO, T. et al. Time series forecasting using a deep belief network with restricted Boltzmann machines. *Neurocomputing*, v. 137, p. 47–56, 2014.

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

MENDEL, E.; KROHLING, R. A.; CAMPOS, M. Swarm algorithms with chaotic jumps applied to noisy optimization problems. *Information Sciences*, Elsevier, v. 181, n. 20, p. 4494–4514, 2011.

OJHA, V. K.; ABRAHAM, A.; SNÁŠEL, V. Metaheuristic design of feedforward neural networks: A review of two decades of research. *Engineering Applications of Artificial Intelligence*, Elsevier, v. 60, p. 97–116, 2017.

PACHECO, A. G. *Agregação de classificadores neurais via integral de Choquet com respeito a uma medida fuzzy*. Dissertação de mestrado — Universidade Federal do Espírito Santo, Vitória, 2016.

PACHECO, A. G. C. *Otimização por enxame de partículas – PSO*. 2016. Disponível em: <http://pachecoandre.com.br/artigos/otimizacao-por-enxame-de-particulas-pso/>.

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

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

ANEXO A – Resultados do Estudo de Convergência

Nesse apêndice são apresentados os resultados completos do estudo realizado na seção seção 4.2. Esse estudo tem por finalidade verificar o comportamento do RMSE da rede com o decorrer do treinamento, para algumas configurações de neurônios na camada escondida. Nas próximas seções são apresentados os resultados desse estudo e os tempos de treinamento, para as bases discutidas na seção 4.1.

A.1 Base *Iris*

Base	**Neurônios na camada escondida da RBM**	**Número de iterações / épocas**	**RMSE CD-RBM**	**RMSE PSO-RBM**
Iris	10	50	0.317 ± 0.0142	0.703 ± 0.0569
Iris	10	100	0.316 ± 0.0135	0.615 ± 0.0458
Iris	10	200	0.307 ± 0.0146	0.519 ± 0.0517
Iris	10	300	0.313 ± 0.0109	0.468 ± 0.0495
Iris	10	400	$\mathbf{0.307 \pm 0.0140}$	0.438 ± 0.0414
Iris	10	500	0.312 ± 0.0166	$\mathbf{0.424 \pm 0.0384}$
Iris	20	50	0.336 ± 0.0166	1.033 ± 0.0815
Iris	20	100	0.344 ± 0.0214	0.942 ± 0.0841
Iris	20	200	0.339 ± 0.0237	0.807 ± 0.0421
Iris	20	300	0.337 ± 0.0154	0.749 ± 0.0506
Iris	20	400	0.334 ± 0.0210	0.692 ± 0.0552
Iris	20	500	0.329 ± 0.0170	0.688 ± 0.0497
Iris	30	50	0.367 ± 0.0241	1.342 ± 0.0899
Iris	30	100	0.372 ± 0.0196	1.192 ± 0.0999
Iris	30	200	0.365 ± 0.0157	1.086 ± 0.0751
Iris	30	300	0.354 ± 0.0225	0.993 ± 0.0789
Iris	30	400	0.348 ± 0.0208	0.942 ± 0.0826
Iris	30	500	0.352 ± 0.0208	0.868 ± 0.0620
Iris	40	50	0.398 ± 0.0207	1.588 ± 0.0980
Iris	40	100	0.392 ± 0.0235	1.450 ± 0.1071
Iris	40	200	0.383 ± 0.0232	1.301 ± 0.0850

Iris	40	300	0.378 ± 0.0182	1.233 ± 0.0769
Iris	40	400	0.376 ± 0.0199	1.145 ± 0.0995
Iris	40	500	0.363 ± 0.0228	1.071 ± 0.0851
Iris	50	50	0.424 ± 0.0243	1.847 ± 0.1473
Iris	50	100	0.412 ± 0.0241	1.674 ± 0.1241
Iris	50	200	0.401 ± 0.0157	1.493 ± 0.1157
Iris	50	300	0.398 ± 0.0254	1.433 ± 0.1048
Iris	50	400	0.395 ± 0.0175	1.315 ± 0.1064
Iris	50	500	0.387 ± 0.0212	1.255 ± 0.0799

Tabela 7 – Resultados do estudo de convergência para a base *Iris*

Base	**Neurônios na camada escondida da RBM**	**Número de iterações / épocas**	**Tempo CD-RBM (s)**	**Tempo PSO-RBM (s)**
Iris	10	50	0.01	1.02
Iris	10	100	0.02	2.02
Iris	10	200	0.04	4.01
Iris	10	300	0.07	5.99
Iris	10	400	0.09	7.98
Iris	10	500	0.11	9.97
Iris	20	50	0.02	1.66
Iris	20	100	0.03	3.29
Iris	20	200	0.06	6.56
Iris	20	300	0.09	9.83
Iris	20	400	0.12	13.07
Iris	20	500	0.15	16.38
Iris	30	50	0.02	2.24
Iris	30	100	0.03	4.46
Iris	30	200	0.07	8.90
Iris	30	300	0.10	13.29
Iris	30	400	0.13	17.74
Iris	30	500	0.16	22.14
Iris	40	50	0.02	2.80
Iris	40	100	0.04	5.55
Iris	40	200	0.07	11.07
Iris	40	300	0.11	16.55
Iris	40	400	0.14	22.07

Iris	40	500	0.18	27.54
Iris	50	50	0.02	3.35
Iris	50	100	0.04	6.64
Iris	50	200	0.08	13.22
Iris	50	300	0.12	19.81
Iris	50	400	0.16	26.40
Iris	50	500	0.20	33.00

Tabela 8 – Tempos do estudo de convergência para a base *Iris*

A.2 Base *Seeds*

Base	**Neurônios na camada escondida da RBM**	**Número de iterações / épocas**	**RMSE CD-RBM**	**RMSE PSO-RBM**
Seeds	10	50	0.301 ± 0.0098	0.759 ± 0.0449
Seeds	10	100	0.294 ± 0.0101	0.666 ± 0.0417
Seeds	10	200	0.293 ± 0.0109	0.595 ± 0.0469
Seeds	10	300	0.292 ± 0.0131	0.538 ± 0.0383
Seeds	10	400	$\mathbf{0.286 \pm 0.0135}$	0.515 ± 0.0373
Seeds	10	500	0.291 ± 0.0108	$\mathbf{0.506 \pm 0.0389}$
Seeds	20	50	0.327 ± 0.0107	1.112 ± 0.0672
Seeds	20	100	0.329 ± 0.0113	0.981 ± 0.0408
Seeds	20	200	0.321 ± 0.0158	0.881 ± 0.0537
Seeds	20	300	0.321 ± 0.0107	0.837 ± 0.0462
Seeds	20	400	0.316 ± 0.0140	0.785 ± 0.0571
Seeds	20	500	0.309 ± 0.0123	0.760 ± 0.0446
Seeds	30	50	0.355 ± 0.0165	1.373 ± 0.0549
Seeds	30	100	0.353 ± 0.0151	1.254 ± 0.0731
Seeds	30	200	0.343 ± 0.0131	1.133 ± 0.0670
Seeds	30	300	0.344 ± 0.0175	1.065 ± 0.0540
Seeds	30	400	0.341 ± 0.0142	1.027 ± 0.0503
Seeds	30	500	0.329 ± 0.0140	0.985 ± 0.0538
Seeds	40	50	0.385 ± 0.0171	1.648 ± 0.0838
Seeds	40	100	0.382 ± 0.0150	1.483 ± 0.0620
Seeds	40	200	0.379 ± 0.0141	1.306 ± 0.0688
Seeds	40	300	0.369 ± 0.0121	1.277 ± 0.0729
Seeds	40	400	0.359 ± 0.0128	1.220 ± 0.0628

Seeds	40	500	0.353 ± 0.0133	1.183 ± 0.0642
Seeds	50	50	0.414 ± 0.0150	1.853 ± 0.1104
Seeds	50	100	0.405 ± 0.0154	1.681 ± 0.0884
Seeds	50	200	0.393 ± 0.0150	1.511 ± 0.0749
Seeds	50	300	0.384 ± 0.0149	1.449 ± 0.0741
Seeds	50	400	0.377 ± 0.0132	1.379 ± 0.0529
Seeds	50	500	0.369 ± 0.0155	1.322 ± 0.0735

Tabela 9 – Resultados do estudo de convergência para a base *Seeds*

Base	**Neurônios na camada escondida da RBM**	**Número de iterações / épocas**	**Tempo CD-RBM (s)**	**Tempo PSO-RBM (s)**
Seeds	10	50	0.01	1.43
Seeds	10	100	0.03	2.81
Seeds	10	200	0.06	5.60
Seeds	10	300	0.08	8.38
Seeds	10	400	0.11	11.16
Seeds	10	500	0.13	13.95
Seeds	20	50	0.02	2.38
Seeds	20	100	0.04	4.72
Seeds	20	200	0.07	9.41
Seeds	20	300	0.11	14.11
Seeds	20	400	0.14	18.77
Seeds	20	500	0.18	23.48
Seeds	30	50	0.02	3.23
Seeds	30	100	0.04	6.42
Seeds	30	200	0.08	12.79
Seeds	30	300	0.12	19.16
Seeds	30	400	0.16	25.53
Seeds	30	500	0.20	31.89
Seeds	40	50	0.03	5.01
Seeds	40	100	0.06	9.92
Seeds	40	200	0.11	19.78
Seeds	40	300	0.16	29.62
Seeds	40	400	0.22	39.39
Seeds	40	500	0.27	49.41
Seeds	50	50	0.03	5.24

Seeds	50	100	0.06	10.38
Seeds	50	200	0.12	20.67
Seeds	50	300	0.17	30.97
Seeds	50	400	0.22	41.27
Seeds	50	500	0.27	51.57

Tabela 10 – Tempos do estudo de convergência para a base *Seeds*

A.3 Base *Column*

Base	**Neurônios na camada escondida da RBM**	**Número de iterações / épocas**	**RMSE CD-RBM**	**RMSE PSO-RBM**
Column	10	50	0.227 ± 0.0105	0.724 ± 0.0478
Column	10	100	0.222 ± 0.0112	0.629 ± 0.0421
Column	10	200	$\mathbf{0.221 \pm 0.0092}$	0.512 ± 0.0363
Column	10	300	0.224 ± 0.0096	0.456 ± 0.0526
Column	10	400	0.222 ± 0.0108	0.424 ± 0.0381
Column	10	500	0.223 ± 0.0121	$\mathbf{0.395 \pm 0.0447}$
Column	20	50	0.268 ± 0.0115	1.107 ± 0.0693
Column	20	100	0.264 ± 0.0119	0.963 ± 0.0520
Column	20	200	0.261 ± 0.0124	0.852 ± 0.0459
Column	20	300	0.260 ± 0.0092	0.781 ± 0.0625
Column	20	400	0.257 ± 0.0112	0.731 ± 0.0466
Column	20	500	0.255 ± 0.0094	0.687 ± 0.0509
Column	30	50	0.311 ± 0.0142	1.394 ± 0.0517
Column	30	100	0.296 ± 0.0130	1.223 ± 0.0679
Column	30	200	0.295 ± 0.0114	1.105 ± 0.0645
Column	30	300	0.288 ± 0.0106	1.042 ± 0.0631
Column	30	400	0.280 ± 0.0119	0.969 ± 0.0577
Column	30	500	0.281 ± 0.0088	0.918 ± 0.0505
Column	40	50	0.341 ± 0.0176	1.608 ± 0.0679
Column	40	100	0.334 ± 0.0154	1.476 ± 0.0589
Column	40	200	0.320 ± 0.0154	1.316 ± 0.0608
Column	40	300	0.319 ± 0.0107	1.226 ± 0.0481
Column	40	400	0.304 ± 0.0120	1.169 ± 0.0630
Column	40	500	0.299 ± 0.0086	1.136 ± 0.0708
Column	50	50	0.369 ± 0.0163	1.824 ± 0.0806

Column	50	100	0.367 ± 0.0158	1.678 ± 0.0673
Column	50	200	0.351 ± 0.0138	1.504 ± 0.0981
Column	50	300	0.341 ± 0.0141	1.428 ± 0.0712
Column	50	400	0.327 ± 0.0118	1.349 ± 0.0598
Column	50	500	0.320 ± 0.0106	1.291 ± 0.0586

Tabela 11 – Resultados do estudo de convergência para a base *Column*

Base	**Neurônios na camada escondida da RBM**	**Número de iterações / épocas**	**Tempo CD-RBM (s)**	**Tempo PSO-RBM (s)**
Column	10	50	0.02	1.38
Column	10	100	0.03	2.73
Column	10	200	0.03	5.44
Column	10	300	0.04	8.15
Column	10	400	0.04	10.84
Column	10	500	0.04	13.54
Column	20	50	0.02	2.27
Column	20	100	0.04	4.50
Column	20	200	0.07	8.95
Column	20	300	0.08	13.40
Column	20	400	0.10	17.85
Column	20	500	0.12	22.30
Column	30	50	0.03	3.07
Column	30	100	0.05	6.14
Column	30	200	0.10	12.15
Column	30	300	0.15	18.22
Column	30	400	0.19	24.15
Column	30	500	0.20	30.16
Column	40	50	0.03	4.26
Column	40	100	0.06	8.41
Column	40	200	0.12	16.81
Column	40	300	0.17	25.12
Column	40	400	0.23	33.47
Column	40	500	0.29	41.81
Column	50	50	0.03	5.34
Column	50	100	0.07	10.57
Column	50	200	0.13	21.07

Column	50	300	0.20	31.48
Column	50	400	0.28	42.55
Column	50	500	0.34	53.05

Tabela 12 – Tempos do estudo de convergência para a base *Column*

A.4 Base *Wine*

Base	**Neurônios na camada escondida da RBM**	**Número de iterações / épocas**	**RMSE CD-RBM**	**RMSE PSO-RBM**
Wine	10	50	0.252 ± 0.0057	0.812 ± 0.0451
Wine	10	100	0.251 ± 0.0087	0.723 ± 0.0316
Wine	10	200	0.252 ± 0.0074	0.667 ± 0.0447
Wine	10	300	$\mathbf{0.246 \pm 0.0054}$	0.655 ± 0.0412
Wine	10	400	0.247 ± 0.0071	0.616 ± 0.0388
Wine	10	500	0.246 ± 0.0081	$\mathbf{0.613 \pm 0.0477}$
Wine	20	50	0.295 ± 0.0110	1.166 ± 0.0601
Wine	20	100	0.286 ± 0.0077	1.075 ± 0.0390
Wine	20	200	0.285 ± 0.0063	0.961 ± 0.0517
Wine	20	300	0.281 ± 0.0065	0.932 ± 0.0556
Wine	20	400	0.277 ± 0.0092	0.911 ± 0.0449
Wine	20	500	0.272 ± 0.0072	0.879 ± 0.0398
Wine	30	50	0.327 ± 0.0137	1.453 ± 0.0604
Wine	30	100	0.323 ± 0.0142	1.331 ± 0.0586
Wine	30	200	0.312 ± 0.0111	1.191 ± 0.0517
Wine	30	300	0.309 ± 0.0106	1.139 ± 0.0586
Wine	30	400	0.304 ± 0.0089	1.138 ± 0.0534
Wine	30	500	0.298 ± 0.0065	1.076 ± 0.0459
Wine	40	50	0.358 ± 0.0119	1.695 ± 0.0820
Wine	40	100	0.350 ± 0.0098	1.548 ± 0.0800
Wine	40	200	0.340 ± 0.0118	1.424 ± 0.0644
Wine	40	300	0.335 ± 0.0110	1.338 ± 0.0654
Wine	40	400	0.326 ± 0.0100	1.301 ± 0.0605
Wine	40	500	0.321 ± 0.0083	1.261 ± 0.0564
Wine	50	50	0.382 ± 0.0141	1.912 ± 0.0762
Wine	50	100	0.378 ± 0.0111	1.746 ± 0.0719
Wine	50	200	0.369 ± 0.0119	1.571 ± 0.0566

Wine	50	300	0.354 ± 0.0118	1.525 ± 0.0670
Wine	50	400	0.348 ± 0.0108	1.478 ± 0.0684
Wine	50	500	0.341 ± 0.0088	1.445 ± 0.0646

Tabela 13 – Resultados do estudo de convergência para a base *Wine*

Base	**Neurônios na camada escondida da RBM**	**Número de iterações / épocas**	**Tempo CD-RBM (s)**	**Tempo PSO-RBM (s)**
Wine	10	50	0.01	2.22
Wine	10	100	0.03	4.39
Wine	10	200	0.05	8.69
Wine	10	300	0.08	13.00
Wine	10	400	0.11	17.32
Wine	10	500	0.13	21.64
Wine	20	50	0.02	3.73
Wine	20	100	0.03	7.34
Wine	20	200	0.07	14.74
Wine	20	300	0.10	21.82
Wine	20	400	0.14	29.06
Wine	20	500	0.17	36.46
Wine	30	50	0.02	6.04
Wine	30	100	0.05	11.89
Wine	30	200	0.09	23.63
Wine	30	300	0.14	35.47
Wine	30	400	0.18	46.57
Wine	30	500	0.23	58.17
Wine	40	50	0.03	7.52
Wine	40	100	0.05	14.73
Wine	40	200	0.10	29.40
Wine	40	300	0.15	44.10
Wine	40	400	0.21	59.68
Wine	40	500	0.25	74.54
Wine	50	50	0.03	8.26
Wine	50	100	0.06	16.09
Wine	50	200	0.11	32.05
Wine	50	300	0.16	48.06
Wine	50	400	0.21	63.90

Wine	50	500	0.26	79.93

Tabela 14 – Tempos do estudo de convergência para a base *Wine*

A.5 Base *ILPD*

Base	**Neurônios na camada escondida da RBM**	**Número de iterações / épocas**	**RMSE CD-RBM**	**RMSE PSO-RBM**
ILPD	10	50	0.244 ± 0.0077	0.773 ± 0.0359
ILPD	10	100	0.240 ± 0.0070	0.680 ± 0.0331
ILPD	10	200	0.241 ± 0.0066	0.609 ± 0.0376
ILPD	10	300	0.239 ± 0.0067	0.555 ± 0.0321
ILPD	10	400	0.239 ± 0.0063	0.515 ± 0.0395
ILPD	10	500	$\mathbf{0.238 \pm 0.0050}$	$\mathbf{0.513 \pm 0.0338}$
ILPD	20	50	0.285 ± 0.0075	1.137 ± 0.0362
ILPD	20	100	0.281 ± 0.0097	1.023 ± 0.0402
ILPD	20	200	0.275 ± 0.0073	0.904 ± 0.0456
ILPD	20	300	0.271 ± 0.0066	0.847 ± 0.0391
ILPD	20	400	0.267 ± 0.0063	0.815 ± 0.0564
ILPD	20	500	0.268 ± 0.0071	0.764 ± 0.0505
ILPD	30	50	0.321 ± 0.0079	1.409 ± 0.0484
ILPD	30	100	0.316 ± 0.0084	1.280 ± 0.0480
ILPD	30	200	0.307 ± 0.0084	1.148 ± 0.0481
ILPD	30	300	0.300 ± 0.0083	1.067 ± 0.0572
ILPD	30	400	0.295 ± 0.0074	1.042 ± 0.0540
ILPD	30	500	0.291 ± 0.0064	0.995 ± 0.0603
ILPD	40	50	0.350 ± 0.0116	1.638 ± 0.0583
ILPD	40	100	0.349 ± 0.0078	1.495 ± 0.0464
ILPD	40	200	0.337 ± 0.0095	1.361 ± 0.0411
ILPD	40	300	0.328 ± 0.0086	1.262 ± 0.0504
ILPD	40	400	0.321 ± 0.0084	1.217 ± 0.0534
ILPD	40	500	0.313 ± 0.0085	1.177 ± 0.0585
ILPD	50	50	0.375 ± 0.0107	1.866 ± 0.0499
ILPD	50	100	0.372 ± 0.0115	1.677 ± 0.0528
ILPD	50	200	0.359 ± 0.0094	1.528 ± 0.0583
ILPD	50	300	0.349 ± 0.0096	1.426 ± 0.0524
ILPD	50	400	0.338 ± 0.0088	1.388 ± 0.0582

ILPD	50	500	0.332 ± 0.0070	1.337 ± 0.0491

Tabela 15 – Resultados do estudo de convergência para a base *ILPD*

Base	**Neurônios na camada escondida da RBM**	**Número de iterações / épocas**	**Tempo CD-RBM (s)**	**Tempo PSO-RBM (s)**
ILPD	10	50	0.03	2.66
ILPD	10	100	0.05	5.27
ILPD	10	200	0.08	10.45
ILPD	10	300	0.10	15.62
ILPD	10	400	0.13	21.17
ILPD	10	500	0.13	26.36
ILPD	20	50	0.03	4.17
ILPD	20	100	0.07	8.24
ILPD	20	200	0.14	16.41
ILPD	20	300	0.20	24.58
ILPD	20	400	0.25	32.66
ILPD	20	500	0.30	41.16
ILPD	30	50	0.04	5.46
ILPD	30	100	0.08	10.74
ILPD	30	200	0.16	21.42
ILPD	30	300	0.23	32.06
ILPD	30	400	0.32	42.50
ILPD	30	500	0.39	53.08
ILPD	40	50	0.05	6.67
ILPD	40	100	0.09	13.22
ILPD	40	200	0.18	26.30
ILPD	40	300	0.27	39.35
ILPD	40	400	0.36	52.39
ILPD	40	500	0.45	65.55
ILPD	50	50	0.05	7.80
ILPD	50	100	0.10	15.51
ILPD	50	200	0.20	30.92
ILPD	50	300	0.30	46.27
ILPD	50	400	0.40	61.86
ILPD	50	500	0.50	77.22

Tabela 16 – Tempos do estudo de convergência para a base *ILPD*

A.6 Base *Cancer*

Base	**Neurônios na camada escondida da RBM**	**Número de iterações / épocas**	**RMSE CD-RBM**	**RMSE PSO-RBM**
Cancer	10	50	$\mathbf{0.312 \pm 0.0056}$	0.791 ± 0.0360
Cancer	10	100	0.316 ± 0.0068	0.688 ± 0.0245
Cancer	10	200	0.314 ± 0.0068	0.603 ± 0.0235
Cancer	10	300	0.313 ± 0.0050	0.563 ± 0.0269
Cancer	10	400	0.315 ± 0.0063	0.535 ± 0.0368
Cancer	10	500	0.312 ± 0.0064	$\mathbf{0.502 \pm 0.0297}$
Cancer	20	50	0.350 ± 0.0076	1.159 ± 0.0488
Cancer	20	100	0.344 ± 0.0077	1.023 ± 0.0397
Cancer	20	200	0.343 ± 0.0068	0.899 ± 0.0411
Cancer	20	300	0.336 ± 0.0076	0.830 ± 0.0409
Cancer	20	400	0.335 ± 0.0082	0.775 ± 0.0412
Cancer	20	500	0.337 ± 0.0077	0.758 ± 0.0401
Cancer	30	50	0.378 ± 0.0097	1.422 ± 0.0487
Cancer	30	100	0.372 ± 0.0081	1.268 ± 0.0380
Cancer	30	200	0.366 ± 0.0074	1.141 ± 0.0394
Cancer	30	300	0.364 ± 0.0095	1.046 ± 0.0450
Cancer	30	400	0.360 ± 0.0073	1.007 ± 0.0411
Cancer	30	500	0.357 ± 0.0087	0.963 ± 0.0604
Cancer	40	50	0.399 ± 0.0099	1.663 ± 0.0543
Cancer	40	100	0.398 ± 0.0111	1.511 ± 0.0591
Cancer	40	200	0.391 ± 0.0095	1.336 ± 0.0442
Cancer	40	300	0.383 ± 0.0096	1.241 ± 0.0400
Cancer	40	400	0.378 ± 0.0075	1.168 ± 0.0376
Cancer	40	500	0.374 ± 0.0088	1.146 ± 0.0524
Cancer	50	50	0.425 ± 0.0090	1.894 ± 0.0690
Cancer	50	100	0.420 ± 0.0081	1.704 ± 0.0714
Cancer	50	200	0.409 ± 0.0075	1.538 ± 0.0511
Cancer	50	300	0.402 ± 0.0073	1.419 ± 0.0420
Cancer	50	400	0.392 ± 0.0087	1.346 ± 0.0353
Cancer	50	500	0.391 ± 0.0095	1.318 ± 0.0512

Tabela 17 – Resultados do estudo de convergência para a base *Cancer*

Base	**Neurônios na camada escondida da RBM**	**Número de iterações / épocas**	**Tempo CD-RBM (s)**	**Tempo PSO-RBM (s)**
Cancer	10	50	0.03	2.47
Cancer	10	100	0.06	4.79
Cancer	10	200	0.11	9.49
Cancer	10	300	0.17	14.25
Cancer	10	400	0.22	18.86
Cancer	10	500	0.28	23.62
Cancer	20	50	0.04	3.97
Cancer	20	100	0.07	7.80
Cancer	20	200	0.14	15.51
Cancer	20	300	0.21	23.15
Cancer	20	400	0.28	30.86
Cancer	20	500	0.35	38.46
Cancer	30	50	0.04	5.03
Cancer	30	100	0.08	9.95
Cancer	30	200	0.17	19.76
Cancer	30	300	0.25	29.71
Cancer	30	400	0.33	39.44
Cancer	30	500	0.41	49.24
Cancer	40	50	0.05	6.21
Cancer	40	100	0.10	12.25
Cancer	40	200	0.20	24.52
Cancer	40	300	0.29	36.41
Cancer	40	400	0.40	48.56
Cancer	40	500	0.49	60.44
Cancer	50	50	0.06	7.77
Cancer	50	100	0.11	15.38
Cancer	50	200	0.23	32.03
Cancer	50	300	0.34	47.40
Cancer	50	400	0.45	63.05
Cancer	50	500	0.57	78.91

Tabela 18 – Tempos do estudo de convergência para a base *Cancer*

A.7 Base *Credit Australia*

Base	**Neurônios na camada escondida da RBM**	**Número de iterações / épocas**	**RMSE CD-RBM**	**RMSE PSO-RBM**
Credit Australia	10	50	0.333 ± 0.0047	0.821 ± 0.0348
Credit Australia	10	100	0.332 ± 0.0052	0.737 ± 0.0262
Credit Australia	10	200	0.332 ± 0.0047	0.656 ± 0.0360
Credit Australia	10	300	0.332 ± 0.0053	0.623 ± 0.0360
Credit Australia	10	400	$\mathbf{0.330 \pm 0.0040}$	$\mathbf{0.605 \pm 0.0367}$
Credit Australia	10	500	0.330 ± 0.0049	0.594 ± 0.0339
Credit Australia	20	50	0.364 ± 0.0068	1.166 ± 0.0446
Credit Australia	20	100	0.359 ± 0.0061	1.041 ± 0.0357
Credit Australia	20	200	0.358 ± 0.0060	0.947 ± 0.0406
Credit Australia	20	300	0.353 ± 0.0042	0.899 ± 0.0366
Credit Australia	20	400	0.353 ± 0.0057	0.865 ± 0.0339
Credit Australia	20	500	0.351 ± 0.0051	0.858 ± 0.0418
Credit Australia	30	50	0.388 ± 0.0071	1.459 ± 0.0552
Credit Australia	30	100	0.382 ± 0.0073	1.289 ± 0.0353
Credit Australia	30	200	0.380 ± 0.0065	1.169 ± 0.0405
Credit Australia	30	300	0.376 ± 0.0069	1.106 ± 0.0461
Credit Australia	30	400	0.372 ± 0.0065	1.073 ± 0.0456
Credit Australia	30	500	0.371 ± 0.0045	1.032 ± 0.0503
Credit Australia	40	50	0.412 ± 0.0071	1.707 ± 0.0721
Credit Australia	40	100	0.410 ± 0.0070	1.513 ± 0.0462
Credit Australia	40	200	0.402 ± 0.0061	1.394 ± 0.0454
Credit Australia	40	300	0.398 ± 0.0065	1.301 ± 0.0389
Credit Australia	40	400	0.391 ± 0.0050	1.252 ± 0.0500
Credit Australia	40	500	0.388 ± 0.0060	1.213 ± 0.0438
Credit Australia	50	50	0.434 ± 0.0080	1.961 ± 0.0664
Credit Australia	50	100	0.429 ± 0.0094	1.710 ± 0.0497
Credit Australia	50	200	0.421 ± 0.0076	1.545 ± 0.0511
Credit Australia	50	300	0.414 ± 0.0056	1.463 ± 0.0472
Credit Australia	50	400	0.406 ± 0.0057	1.401 ± 0.0589
Credit Australia	50	500	0.402 ± 0.0072	1.378 ± 0.0479

Tabela 19 – Resultados do estudo de convergência para a base *Credit Australia*

Base	**Neurônios na camada escondida da RBM**	**Número de iterações / épocas**	**Tempo CD-RBM (s)**	**Tempo PSO-RBM (s)**
Credit Australia	10	50	0.03	3.72
Credit Australia	10	100	0.06	6.90
Credit Australia	10	200	0.12	12.59
Credit Australia	10	300	0.17	18.58
Credit Australia	10	400	0.23	26.99
Credit Australia	10	500	0.29	32.87
Credit Australia	20	50	0.04	5.15
Credit Australia	20	100	0.07	10.12
Credit Australia	20	200	0.15	20.12
Credit Australia	20	300	0.22	30.06
Credit Australia	20	400	0.29	37.59
Credit Australia	20	500	0.36	48.28
Credit Australia	30	50	0.04	6.86
Credit Australia	30	100	0.09	13.49
Credit Australia	30	200	0.18	26.81
Credit Australia	30	300	0.26	40.55
Credit Australia	30	400	0.35	52.57
Credit Australia	30	500	0.44	62.75
Credit Australia	40	50	0.05	9.07
Credit Australia	40	100	0.10	17.90
Credit Australia	40	200	0.21	36.41
Credit Australia	40	300	0.31	52.98
Credit Australia	40	400	0.42	70.50
Credit Australia	40	500	0.52	88.37
Credit Australia	50	50	0.06	11.18
Credit Australia	50	100	0.12	21.58
Credit Australia	50	200	0.23	40.07
Credit Australia	50	300	0.35	59.65
Credit Australia	50	400	0.46	77.93
Credit Australia	50	500	0.58	105.02

Tabela 20 – Tempos do estudo de convergência para a base *Credit Australia*

A.8 Base *Sonar*

Base	Neurônios na camada escondida da RBM	Número de iterações / épocas	RMSE CD-RBM	RMSE PSO-RBM
Sonar	10	50	0.247 ± 0.0032	0.887 ± 0.0520
Sonar	10	100	0.244 ± 0.0036	0.720 ± 0.0725
Sonar	10	200	0.242 ± 0.0037	0.579 ± 0.0517
Sonar	10	300	0.239 ± 0.0024	0.486 ± 0.0661
Sonar	10	400	0.237 ± 0.0035	0.472 ± 0.0664
Sonar	10	500	0.235 ± 0.0037	0.445 ± 0.0640
Sonar	20	50	0.287 ± 0.0049	1.252 ± 0.0664
Sonar	20	100	0.282 ± 0.0038	1.054 ± 0.0686
Sonar	20	200	0.276 ± 0.0043	0.870 ± 0.0680
Sonar	20	300	0.271 ± 0.0034	0.793 ± 0.0728
Sonar	20	400	0.266 ± 0.0039	0.727 ± 0.0786
Sonar	20	500	0.262 ± 0.0049	0.727 ± 0.0681
Sonar	30	50	0.321 ± 0.0052	1.529 ± 0.0592
Sonar	30	100	0.316 ± 0.0049	1.326 ± 0.0646
Sonar	30	200	0.307 ± 0.0038	1.124 ± 0.0671
Sonar	30	300	0.299 ± 0.0036	1.062 ± 0.0760
Sonar	30	400	0.293 ± 0.0045	1.012 ± 0.0777
Sonar	30	500	0.285 ± 0.0047	1.001 ± 0.0984
Sonar	40	50	0.354 ± 0.0061	1.795 ± 0.0699
Sonar	40	100	0.345 ± 0.0038	1.546 ± 0.0579
Sonar	40	200	0.332 ± 0.0055	1.370 ± 0.0871
Sonar	40	300	0.321 ± 0.0046	1.287 ± 0.0673
Sonar	40	400	0.314 ± 0.0041	1.232 ± 0.0826
Sonar	40	500	0.306 ± 0.0050	1.210 ± 0.0775
Sonar	50	50	0.383 ± 0.0052	2.042 ± 0.0840
Sonar	50	100	0.371 ± 0.0045	1.763 ± 0.0531
Sonar	50	200	0.356 ± 0.0055	1.550 ± 0.0546
Sonar	50	300	0.343 ± 0.0041	1.485 ± 0.0560
Sonar	50	400	0.333 ± 0.0048	1.432 ± 0.0686
Sonar	50	500	0.323 ± 0.0046	1.405 ± 0.0697

Tabela 21 – Resultados do estudo de convergência para a base *Sonar*

Base	**Neurônios na camada escondida da RBM**	**Número de iterações / épocas**	**Tempo CD-RBM (s)**	**Tempo PSO-RBM (s)**
Sonar	10	50	0.03	9.43
Sonar	10	100	0.05	16.93
Sonar	10	200	0.10	35.44
Sonar	10	300	0.15	48.78
Sonar	10	400	0.20	62.53
Sonar	10	500	0.25	78.18
Sonar	20	50	0.03	14.26
Sonar	20	100	0.06	28.26
Sonar	20	200	0.12	56.50
Sonar	20	300	0.18	84.74
Sonar	20	400	0.24	112.93
Sonar	20	500	0.30	141.22
Sonar	30	50	0.03	20.70
Sonar	30	100	0.06	40.86
Sonar	30	200	0.13	81.33
Sonar	30	300	0.19	122.18
Sonar	30	400	0.25	162.65
Sonar	30	500	0.31	205.23
Sonar	40	50	0.04	27.78
Sonar	40	100	0.07	55.07
Sonar	40	200	0.15	108.44
Sonar	40	300	0.21	184.65
Sonar	40	400	0.30	248.49
Sonar	40	500	0.36	310.11
Sonar	50	50	0.04	33.76
Sonar	50	100	0.08	67.13
Sonar	50	200	0.15	134.33
Sonar	50	300	0.23	201.20
Sonar	50	400	0.30	267.78
Sonar	50	500	0.37	334.43

Tabela 22 – Tempos do estudo de convergência para a base *Sonar*

ANEXO B – Resultados de Classificação

Nesse apêndice são apresentados os resultados completos do estudo realizado na seção seção 4.3. Esse estudo tem por finalidade verificar a acurácia do uso de RBMs treinadas via PSO como uma estágio de pré-processamento para classificação de dados. Nas próximas seções são apresentados os resultados desse estudo para as bases discutidas na seção 4.1.

B.1 Base *Iris*

Base	**Neurônios na camada escondida da RBM**	**Neurônios na camada escondida da FF**	**Taxa de acerto CD-RBM**	**Taxa de acerto PSO-RBM**
Iris	10	10	81.81 ± 12.80	83.84 ± 16.15
Iris	10	20	90.58 ± 9.79	85.58 ± 15.75
Iris	10	30	91.59 ± 9.43	88.62 ± 13.49
Iris	10	40	91.59 ± 8.65	88.26 ± 14.84
Iris	10	50	86.16 ± 13.54	89.20 ± 14.59
Iris	20	10	88.55 ± 11.56	86.23 ± 15.62
Iris	20	20	90.36 ± 9.72	90.51 ± 15.48
Iris	20	30	90.07 ± 11.34	88.91 ± 15.26
Iris	20	40	93.26 ± 7.00	85.94 ± 14.26
Iris	20	50	87.75 ± 11.59	87.10 ± 15.21
Iris	30	10	87.25 ± 12.23	89.57 ± 14.60
Iris	30	20	89.64 ± 12.24	88.26 ± 14.70
Iris	30	30	$\mathbf{94.57 \pm 5.47}$	$\mathbf{94.64 \pm 10.63}$
Iris	30	40	87.32 ± 12.59	90.51 ± 12.25
Iris	30	50	89.06 ± 12.28	88.33 ± 16.17
Iris	40	10	91.01 ± 10.14	83.12 ± 16.54
Iris	40	20	91.16 ± 9.94	89.64 ± 15.34
Iris	40	30	92.68 ± 7.16	91.67 ± 12.48
Iris	40	40	92.03 ± 11.45	87.39 ± 16.27
Iris	40	50	82.46 ± 19.16	93.12 ± 10.89
Iris	50	10	91.01 ± 8.88	87.54 ± 14.49
Iris	50	20	92.68 ± 8.48	95.36 ± 7.15
Iris	50	30	91.52 ± 8.42	90.51 ± 13.09
Iris	50	40	90.29 ± 12.89	91.23 ± 12.33

Iris	50	50	89.42 ± 10.99	88.12 ± 14.61

Tabela 23 – Resultados do experimento de classificação para a base *Iris*

B.2 Base *Seeds*

Base	**Neurônios na camada escondida da RBM**	**Neurônios na camada escondida da FF**	**Taxa de acerto CD-RBM**	**Taxa de acerto PSO-RBM**
Seeds	10	10	83.44 ± 13.39	80.78 ± 15.52
Seeds	10	20	86.41 ± 11.23	85.83 ± 13.37
Seeds	10	30	84.48 ± 12.93	89.74 ± 10.60
Seeds	10	40	85.99 ± 13.70	89.90 ± 9.18
Seeds	10	50	85.10 ± 11.96	82.97 ± 16.27
Seeds	20	10	83.33 ± 13.67	84.48 ± 15.41
Seeds	20	20	87.60 ± 11.15	89.64 ± 9.95
Seeds	20	30	86.93 ± 11.97	88.70 ± 11.34
Seeds	20	40	86.51 ± 11.91	89.95 ± 9.55
Seeds	20	50	85.83 ± 12.82	87.24 ± 12.95
Seeds	30	10	84.53 ± 14.90	84.27 ± 16.24
Seeds	30	20	90.57 ± 7.18	85.52 ± 15.50
Seeds	30	30	88.49 ± 10.30	91.04 ± 7.57
Seeds	30	40	87.08 ± 12.28	88.96 ± 11.63
Seeds	30	50	88.07 ± 11.81	86.51 ± 11.90
Seeds	40	10	86.72 ± 12.26	85.83 ± 15.90
Seeds	40	20	87.66 ± 10.94	88.65 ± 12.92
Seeds	40	30	88.91 ± 8.95	89.69 ± 11.58
Seeds	40	40	$\mathbf{91.20 \pm 6.80}$	87.19 ± 14.83
Seeds	40	50	88.02 ± 11.67	85.68 ± 15.76
Seeds	50	10	89.48 ± 10.13	82.29 ± 17.31
Seeds	50	20	90.52 ± 8.63	87.86 ± 12.08
Seeds	50	30	88.07 ± 11.79	90.57 ± 9.92
Seeds	50	40	88.70 ± 10.91	$\mathbf{91.15 \pm 10.09}$
Seeds	50	50	90.68 ± 10.00	86.67 ± 13.86

Tabela 24 – Resultados do experimento de classificação para a base *Seeds*

B.3 Base *Column*

Base	Neurônios na camada escondida da RBM	Neurônios na camada escondida da FF	Taxa de acerto CD-RBM	Taxa de acerto PSO-RBM
Column	10	10	81.99 ± 4.33	77.09 ± 7.72
Column	10	20	81.60 ± 3.71	79.79 ± 7.53
Column	10	30	81.95 ± 4.00	79.93 ± 6.53
Column	10	40	81.63 ± 3.85	75.64 ± 11.48
Column	10	50	81.52 ± 4.37	77.48 ± 8.31
Column	20	10	82.80 ± 3.76	77.84 ± 7.20
Column	20	20	83.62 ± 3.16	80.50 ± 7.04
Column	20	30	82.59 ± 3.43	79.22 ± 7.96
Column	20	40	81.31 ± 11.13	78.79 ± 8.37
Column	20	50	82.34 ± 4.26	78.33 ± 8.72
Column	30	10	81.77 ± 3.59	80.89 ± 6.46
Column	30	20	82.38 ± 3.41	80.25 ± 6.49
Column	30	30	$\mathbf{83.62 \pm 3.09}$	81.21 ± 6.85
Column	30	40	82.30 ± 3.65	80.35 ± 7.04
Column	30	50	82.41 ± 6.02	79.61 ± 7.84
Column	40	10	82.16 ± 3.40	79.04 ± 6.64
Column	40	20	82.77 ± 3.40	$\mathbf{83.01 \pm 3.51}$
Column	40	30	82.98 ± 2.53	82.02 ± 5.55
Column	40	40	82.94 ± 3.47	79.65 ± 9.61
Column	40	50	82.80 ± 4.08	79.11 ± 7.27
Column	50	10	81.77 ± 3.25	78.90 ± 7.23
Column	50	20	83.23 ± 2.37	80.32 ± 7.08
Column	50	30	82.27 ± 3.97	81.63 ± 6.42
Column	50	40	82.41 ± 3.84	80.25 ± 7.30
Column	50	50	82.38 ± 3.76	78.09 ± 11.21

Tabela 25 – Resultados do experimento de classificação para a base *Column*

B.4 Base *Wine*

Base	**Neurônios na camada escondida da RBM**	**Neurônios na camada escondida da FF**	**Taxa de acerto CD-RBM**	**Taxa de acerto PSO-RBM**
Wine	10	10	84.38 ± 14.91	87.84 ± 13.06
Wine	10	20	90.80 ± 8.64	92.84 ± 7.73
Wine	10	30	88.46 ± 8.84	91.60 ± 11.16
Wine	10	40	90.25 ± 9.37	89.01 ± 12.71
Wine	10	50	87.35 ± 11.51	89.14 ± 11.46
Wine	20	10	85.93 ± 14.53	90.31 ± 13.30
Wine	20	20	93.40 ± 6.94	93.21 ± 11.15
Wine	20	30	94.51 ± 3.59	96.73 ± 3.46
Wine	20	40	95.37 ± 2.16	91.79 ± 12.76
Wine	20	50	89.57 ± 12.46	94.69 ± 8.75
Wine	30	10	91.23 ± 11.56	93.64 ± 10.44
Wine	30	20	91.79 ± 9.77	96.11 ± 6.92
Wine	30	30	$\mathbf{95.37 \pm 2.27}$	94.38 ± 10.34
Wine	30	40	94.14 ± 6.54	96.30 ± 7.18
Wine	30	50	95.31 ± 1.99	94.44 ± 9.28
Wine	40	10	91.91 ± 9.62	95.62 ± 6.68
Wine	40	20	93.52 ± 8.29	93.70 ± 8.87
Wine	40	30	93.70 ± 5.49	$\mathbf{98.09 \pm 1.57}$
Wine	40	40	94.44 ± 6.13	95.25 ± 8.54
Wine	40	50	95.74 ± 4.19	94.81 ± 8.60
Wine	50	10	92.96 ± 9.35	93.33 ± 11.31
Wine	50	20	90.49 ± 12.17	95.37 ± 9.28
Wine	50	30	92.53 ± 10.00	96.60 ± 5.86
Wine	50	40	94.26 ± 5.44	95.56 ± 6.41
Wine	50	50	94.51 ± 6.18	97.41 ± 2.21

Tabela 26 – Resultados do experimento de classificação para a base *Wine*

B.5 Base *ILPD*

Base	**Neurônios na camada escondida da RBM**	**Neurônios na camada escondida da FF**	**Taxa de acerto CD-RBM**	**Taxa de acerto PSO-RBM**
ILPD	10	10	72.16 ± 0.00	72.75 ± 0.35
ILPD	10	20	72.12 ± 0.26	72.63 ± 0.42
ILPD	10	30	71.59 ± 1.36	$\mathbf{72.75 \pm 0.28}$
ILPD	10	40	72.01 ± 0.58	71.61 ± 5.70
ILPD	10	50	71.69 ± 1.19	72.25 ± 1.09
ILPD	20	10	71.78 ± 1.27	72.59 ± 0.63
ILPD	20	20	71.14 ± 2.14	72.46 ± 0.79
ILPD	20	30	70.78 ± 4.14	72.18 ± 1.23
ILPD	20	40	70.23 ± 8.09	72.61 ± 0.28
ILPD	20	50	71.76 ± 1.13	72.48 ± 0.55
ILPD	30	10	$\mathbf{72.10 \pm 0.67}$	72.71 ± 0.38
ILPD	30	20	71.72 ± 1.10	72.71 ± 0.18
ILPD	30	30	70.44 ± 2.15	72.46 ± 0.57
ILPD	30	40	71.33 ± 1.53	72.50 ± 0.61
ILPD	30	50	71.16 ± 1.52	72.27 ± 0.86
ILPD	40	10	71.91 ± 0.88	72.48 ± 0.74
ILPD	40	20	71.50 ± 1.71	72.50 ± 0.81
ILPD	40	30	69.72 ± 8.07	72.37 ± 0.97
ILPD	40	40	71.21 ± 1.62	72.39 ± 0.89
ILPD	40	50	70.83 ± 1.61	72.39 ± 0.97
ILPD	50	10	71.84 ± 1.56	72.42 ± 0.94
ILPD	50	20	71.61 ± 1.17	71.93 ± 1.61
ILPD	50	30	70.74 ± 1.98	72.46 ± 0.76
ILPD	50	40	70.53 ± 1.82	71.89 ± 1.13
ILPD	50	50	71.23 ± 1.84	71.12 ± 4.03

Tabela 27 – Resultados do experimento de classificação para a base *ILPD*

B.6 Base *Cancer*

Base	Neurônios na camada escondida da RBM	Neurônios na camada escondida da FF	Taxa de acerto CD-RBM	Taxa de acerto PSO-RBM
Cancer	10	10	$\mathbf{96.87 \pm 0.32}$	95.97 ± 0.97
Cancer	10	20	96.70 ± 0.23	96.22 ± 1.04
Cancer	10	30	96.60 ± 0.90	94.50 ± 11.25
Cancer	10	40	92.32 ± 16.73	96.11 ± 0.88
Cancer	10	50	93.95 ± 12.19	$\mathbf{96.30 \pm 0.77}$
Cancer	20	10	96.54 ± 0.38	95.61 ± 2.66
Cancer	20	20	96.64 ± 0.31	96.15 ± 0.89
Cancer	20	30	96.56 ± 0.41	95.62 ± 1.26
Cancer	20	40	96.71 ± 0.41	95.06 ± 2.89
Cancer	20	50	94.20 ± 12.04	95.10 ± 4.64
Cancer	30	10	95.72 ± 5.03	95.73 ± 0.89
Cancer	30	20	96.78 ± 0.34	95.55 ± 1.22
Cancer	30	30	96.62 ± 0.32	95.18 ± 4.88
Cancer	30	40	96.26 ± 1.58	95.64 ± 1.24
Cancer	30	50	94.41 ± 12.02	95.77 ± 0.97
Cancer	40	10	96.68 ± 0.37	95.61 ± 1.29
Cancer	40	20	96.56 ± 0.39	95.67 ± 0.97
Cancer	40	30	93.81 ± 12.32	96.16 ± 0.82
Cancer	40	40	96.48 ± 0.39	94.93 ± 2.58
Cancer	40	50	96.57 ± 0.39	95.64 ± 1.38
Cancer	50	10	96.57 ± 0.34	95.06 ± 1.85
Cancer	50	20	96.68 ± 0.33	95.97 ± 0.73
Cancer	50	30	94.15 ± 11.98	95.29 ± 1.66
Cancer	50	40	96.56 ± 0.33	95.85 ± 0.91
Cancer	50	50	94.80 ± 6.62	94.79 ± 5.70

Tabela 28 – Resultados do experimento de classificação para a base *Cancer*

B.7 Base *Credit Australia*

Base	Neurônios na camada escondida da RBM	Neurônios na camada escondida da FF	Taxa de acerto CD-RBM	Taxa de acerto PSO-RBM
Credit Australia	10	10	83.73 ± 1.32	82.84 ± 1.34
Credit Australia	10	20	84.31 ± 1.47	82.71 ± 1.15
Credit Australia	10	30	83.14 ± 6.36	81.83 ± 2.94
Credit Australia	10	40	82.39 ± 7.33	81.62 ± 3.67
Credit Australia	10	50	80.14 ± 9.51	80.74 ± 6.00
Credit Australia	20	10	$\mathbf{84.52 \pm 1.27}$	82.61 ± 1.04
Credit Australia	20	20	84.29 ± 1.17	82.47 ± 0.91
Credit Australia	20	30	83.22 ± 5.27	81.94 ± 4.64
Credit Australia	20	40	83.06 ± 5.22	82.61 ± 0.99
Credit Australia	20	50	81.88 ± 7.99	82.66 ± 0.72
Credit Australia	30	10	84.07 ± 1.31	$\mathbf{82.77 \pm 1.32}$
Credit Australia	30	20	84.09 ± 1.02	82.63 ± 0.82
Credit Australia	30	30	83.41 ± 5.32	81.71 ± 4.65
Credit Australia	30	40	83.83 ± 1.23	82.02 ± 3.20
Credit Australia	30	50	84.50 ± 1.13	80.13 ± 8.48
Credit Australia	40	10	83.88 ± 1.24	82.48 ± 0.83
Credit Australia	40	20	83.91 ± 1.31	81.94 ± 2.18

Credit Australia	40	30	83.86 ± 1.12	82.47 ± 1.55
Credit Australia	40	40	82.56 ± 7.46	81.35 ± 4.95
Credit Australia	40	50	82.66 ± 5.26	82.15 ± 1.59
Credit Australia	50	10	84.09 ± 1.36	82.42 ± 1.25
Credit Australia	50	20	83.83 ± 1.34	82.52 ± 0.90
Credit Australia	50	30	83.96 ± 1.39	82.47 ± 0.93
Credit Australia	50	40	82.71 ± 5.10	82.32 ± 0.72
Credit Australia	50	50	82.76 ± 5.08	82.26 ± 2.06

Tabela 29 – Resultados do experimento de classificação para a base *Credit Australia*

B.8 Base *Sonar*

Base	**Neurônios na camada escondida da RBM**	**Neurônios na camada escondida da FF**	**Taxa de acerto CD-RBM**	**Taxa de acerto PSO-RBM**
Sonar	10	10	72.49 ± 5.75	65.13 ± 7.70
Sonar	10	20	74.34 ± 6.79	62.91 ± 8.95
Sonar	10	30	73.60 ± 7.65	62.54 ± 8.91
Sonar	10	40	74.07 ± 7.79	62.80 ± 9.54
Sonar	10	50	71.38 ± 9.66	66.24 ± 8.48
Sonar	20	10	78.15 ± 4.98	70.05 ± 3.81
Sonar	20	20	77.14 ± 8.76	69.15 ± 8.67
Sonar	20	30	75.40 ± 9.20	69.89 ± 9.08
Sonar	20	40	79.10 ± 6.66	71.59 ± 8.58
Sonar	20	50	76.14 ± 6.96	71.69 ± 9.33
Sonar	30	10	76.88 ± 6.15	$\mathbf{73.17 \pm 6.76}$
Sonar	30	20	79.05 ± 6.17	71.59 ± 7.45
Sonar	30	30	79.95 ± 7.99	71.27 ± 8.86
Sonar	30	40	77.30 ± 7.63	70.21 ± 8.89
Sonar	30	50	78.68 ± 9.43	71.75 ± 6.03
Sonar	40	10	80.11 ± 4.43	72.43 ± 6.50
Sonar	40	20	80.00 ± 4.54	72.91 ± 6.33
Sonar	40	30	77.88 ± 9.64	71.96 ± 6.73
Sonar	40	40	77.41 ± 8.09	71.59 ± 5.54
Sonar	40	50	80.21 ± 6.95	72.22 ± 5.77
Sonar	50	10	81.75 ± 5.45	72.49 ± 4.97
Sonar	50	20	80.05 ± 7.79	72.01 ± 5.77
Sonar	50	30	$\mathbf{82.06 \pm 5.76}$	70.11 ± 7.40
Sonar	50	40	77.51 ± 9.74	69.37 ± 6.68
Sonar	50	50	78.31 ± 9.30	71.38 ± 9.07

Tabela 30 – Resultados do experimento de classificação da base *Sonar*

ANEXO C – A-TOPSIS

Normalmente, ao utilizar algoritmos estocásticos, os mesmos são executados várias vezes para diferentes *benchmarks*. Uma grande dificuldade é comparar esses algoritmos e definir qual é o melhor. É comum o uso de testes estatísticos para comparar esses algoritmos, porém tais testes não indicam qual é o melhor algoritmo utilizado e como tais testes são feitos por comparações par a par, seu número cresce exponencialmente 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 os quais pretende-se comparar de acordo com o desempenho dos mesmos em um grupo de N *benchmarks* $B = b_1, \cdots, b_N$. Como os algoritmos são executados diversas vezes para cada *benchmark*, são calculadas as matrizes de média e desvio padrão da seguinte forma:

$$\mathbf{M}_\mu = \begin{array}{c} \\ a_1 \\ \vdots \\ a_K \end{array} \begin{array}{c} \begin{array}{ccc} b_1 & \cdots & b_N \end{array} \\ \begin{bmatrix} \mu_{11} & \cdots & \mu_{1N} \\ \vdots & \ddots & \vdots \\ \mu_{K1} & \cdots & \mu_{KN} \end{bmatrix} \end{array} \qquad \mathbf{M}_\sigma = \begin{array}{c} \\ a_1 \\ \vdots \\ a_K \end{array} \begin{array}{c} \begin{array}{ccc} b_1 & \cdots & b_N \end{array} \\ \begin{bmatrix} \sigma_{11} & \cdots & \sigma_{1N} \\ \vdots & \ddots & \vdots \\ \sigma_{K1} & \cdots & \sigma_{KN} \end{bmatrix} \end{array} \tag{C.1}$$

onde $\mu_k n$ e σ_{kn} são a média e o desvio padrão do algoritmo a_k para o *benchmark* b_n, respectivamente. A partir das matrizes na equação C.1, o A-TOPSIS é capaz de ranquear os algoritmos em relação ao grupo de *benchmarks* utilizados como é ilustrado na figura 12.

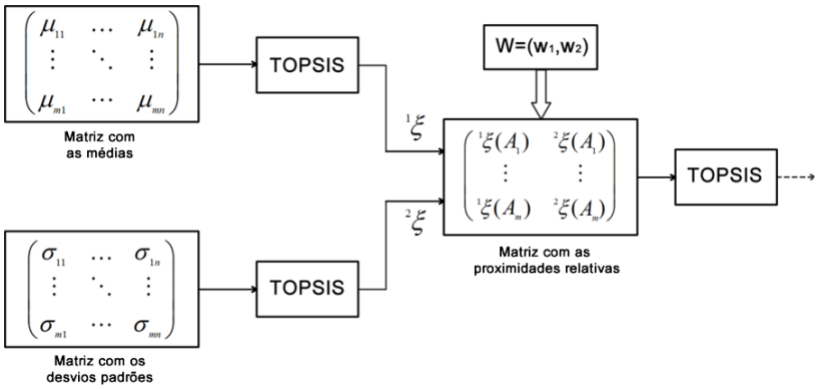


Figura 12 – Ilustração do funcionamento do A-TOPSIS. Retirada de Krohling e Pacheco (2015)

A seguir é descrito o funcionamento do algoritmo:

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

Passo 2: assim como no TOPSIS tradicional, é necessário identificar os critérios de benefício (quanto maior melhor) e custo (quanto menor melhor). Como o desempenho dos *benchmarks* é medido em termos de acurácia, então para a matriz $\mathbf{M}_\mu$ todos os critérios são benefícios, enquanto que, para a matriz $\mathbf{M}_\sigma$ todos os critérios são de custo, já que um menor desvio padrão indica um algoritmo mais estável. Com isso, devem ser identificadas a solução ideal positiva A^+ e a solução ideal negativa A^- para as matrizes $\mathbf{M}_\mu$ e $\mathbf{M}_\sigma$ da seguinte forma:

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

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

Passo 3: calcular a distância euclidiana de A^+ e A^- para cada algoritmo A_k nas duas matrizes com a seguinte equação:

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

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 calculo das proximidades relativas de 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 são aplicados os pesos com soma unitária w_1 e w_2 para cada ξ, onde w_1 se refere à contribuição da média e w_2 do desvio padrão. Assim a matriz é ponderada 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 de 3 a 4 são aplicados para a matriz $\mathbf{G}$, gerando um ξ^G, no qual os melhores algoritmos são os com maior valor. A partir de ξ^G é gerado o ranking dos melhores algoritmos.