UNIVERSIDADE ESTADUAL DO CEARÁ

CENTRO DE CIÊNCIAS E TECNOLOGIA

MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

FRANCISCO WAGNER COSTA AQUINO

ESTRATÉGIAS PARA PREVISÃO DE SÉRIES TEMPORAIS COM CLUSTERIZAÇÃO DE DADOS, ANÁLISE INDEPENDENTE E SELEÇÃO AUTOMÁTICA DE PREVISORES

FORTALEZA, CEARÁ
2011

FRANCISCO WAGNER COSTA AQUINO

ESTRATÉGIAS PARA PREVISÃO DE SÉRIES TEMPORAIS COM CLUSTERIZAÇÃO DE DADOS, ANÁLISE INDEPENDENTE E SELEÇÃO AUTOMÁTICA DE PREVISORES

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Estadual do Ceará, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientação: Prof. Dr. Gustavo Augusto Lima de Campos; Prof. Dr. Jerffeson Teixeira de Souza.

Área de Concentração: Inteligência Computacional.

FORTALEZA, CEARÁ
2011

FRANCISCO WAGNER COSTA AQUINO

ESTRATÉGIAS PARA PREVISÃO DE SÉRIES TEMPORAIS COM CLUSTERIZAÇÃO DE DADOS, ANÁLISE INDEPENDENTE E SELEÇÃO AUTOMÁTICA DE PREVISORES

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Estadual do Ceará, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Aprovada em __/__/ 2011

BANCA EXAMINADORA

_____

Prof. Dr. Gustavo Augusto Lima de Campos (Orientador)
Universidade Estadual do Ceará - UECE

_____

Prof. Dr. Jerffeson Teixeira de Souza (Orientador)
Universidade Estadual do Ceará - UECE

_____

Prof. Dr. Thelmo Pontes de Araújo
Universidade Estadual do Ceará - UECE

_____

Prof. Dr. Tibérius de Oliveira e Bonates
Universidade Federal Rural do Semi-Árido - UFERSA

AGRADECIMENTOS

Agradeço aos meus orientadores, o professor Dr. Gustavo Augusto Lima de Campos e o professor Dr. Jerffeson Teixeira de Souza, pelas orientações, por acreditaram no meu trabalho e por todo apoio e incentivo nas horas mais necessárias.

Aos membros da banca, o professor Dr. Thelmo Pontes de Araújo e o professor Dr. Tibérius de Oliveira e Bonates que tão gentilmente se colocaram a disposição para avaliar esse trabalho.

Agradeço ao Mestrado Acadêmico em Ciências da Computação da Universidade Estadual do Ceará (UECE) pelo apoio ao desenvolvimento dessa pesquisa. Aos meus amigos que me ajudaram e batalharam junto comigo durante toda esta jornada. Em especial ao Abner pelas consultorias em redes neurais, ao André pelas dicas sobre avaliação, ao Dayvison por ser um grande companheiro nesta conquista e a todos os amigos que cederam uma parte de seus processadores para concluir as baterias de testes.

Agradeço aos meus pais, Vicente e Lucivanda, e minha irmã, Wládia, que sempre me apoiaram em todos os momentos da minha vida.

Agradeço aos meus familiares, em especial à minha noiva, Grasiely, por estarem sempre ao meu lado e compreenderem a minha ausência nos momentos difíceis da minha caminhada.

Agradeço a Deus pelas oportunidades que me foram dadas e por ter colocado todas estas pessoas na minha vida.

“Não é porque certas coisas são difíceis que nós não ousamos.
É justamente porque não ousamos que tais coisas são difíceis!”
Sêneca

RESUMO

Este trabalho busca apresentar um método para previsão de séries temporais que se utiliza da estratégia de dividir para conquistar na busca da minimização do erro na previsão. O algoritmo proposto realiza a seleção de exemplos através da clusterização dos dados via rede de Kohonen, com estratégias para aumentar a densidade dos dados. Para cada *cluster*, é gerada automaticamente, através de algoritmos genéticos, uma rede previsora MLP (considerando atributos de entrada, janela de tempo, topologia da rede e calibragem dos parâmetros) otimizada para aquela classe. Também foram definidos dois comitês de máquinas que aliam as informações de semelhança entre os padrões de entrada advindas da clusterização com a combinação de conhecimentos dos especialistas obtida através da essência do comitê de máquinas. Todas as estratégias apresentadas constituem o portfólio de estratégias de previsão do sistema que utiliza uma seleção automática de previsores, dotada de uma grade tridimensional dos desempenhos dos mesmos, para definir qual a melhor estratégia para realizar a previsão de cada padrão de entrada apresentado. A avaliação do algoritmo foi realizada em séries temporais econômicas onde foi possível notar que a seleção automática dos previsores, mesmo utilizando estratégias de previsão medianas se avaliadas em dados gerais, conseguiu uma baixa taxa de erro com pequena variação entre os resultados.

Palavras-chave: Previsão, Séries Temporais, Clusterização, *Backpropagation*, Comitê de Máquinas.

ABSTRACT

This paper presents a method for forecasting time series that uses the strategy of divide and conquer in pursuit of minimizing the error in forecasting. The proposed algorithm performs the sample selection through the clustering of data using Kohonen network, with strategies to increase data density. For each cluster, is generated automatically using genetic algorithms, a MLP network (considering the input attributes, time window, the network topology and calibration of parameters) optimized for that class. Two committees machines were also set that combine the information of similarity between the input patterns arising from clustering with the combination of expert knowledge gained through the essence of the committee machines. All strategies presented are strategies for the portfolio of a system that uses an automatic selection of predictors, having a three-dimensional grid of the performance thereof, to define the best strategy to accomplish the prediction of each input pattern presented. The algorithm evaluation was conducted in economic time series where it was possible to notice that the automatic selection of predictors, using regular forecast strategies if data is evaluated in general, achieved a low error rate with little variation between the results.

Keywords: *Prediction, Times Series, Clustering, Backpropagation, Committee Machines.*

LISTA DE ABREVIATURAS E SIGLAS

ARMA: *Auto-Regressive Moving Average* (Modelos auto regressivos de médias móveis)

ARFIMA: *Autoregressive fractionally integrated moving average*.

ARIMA: *Auto-Regressive Integrated Moving Average* (Modelos auto regressivos de médias móveis integradas)

BCM: *Bayesian Committee Machine* (Comitê de Máquinas Baiesiano)

BMU: *Best Matching Unit* (Melhor neurônio)

CMEF: *Committee Machine with Empirical Formulas* (Comitê de Máquinas com fórmulas Empíricas)

GA: *Genetic Algorithm* (Algoritmo Genético)

GARCH: *Generalized Autoregressive Conditional Heteroskedasticity* (Heterocedasticidade condicional auto-regressiva generalizada)

ICMS: Imposto Sobre Circulação de Mercadorias

IFR: Índice de Força Relativa

MAE: *Mean Absolute Error*

MAPE: *Mean Absolute Percentage Error*

MLP: *Multilayer Perceptron* (*Perceptron* de múltiplas camadas)

RBF: *Radial Basis Function*

SOM: *Self-Organising Maps* (Mapas auto organizáveis)

RMSE: *Root Mean Square Error* (Erro quadrático médio)

RNA: Redes Neurais Artificiais

LISTA DE FIGURAS

LISTA DE EQUAÇÕES

SUMÁRIO

1. INTRODUÇÃO

A ideia de conseguir acertar (prever) eventos é de interesse e, em alguns casos, necessidade das pessoas. Para isto, muitas pesquisas foram realizadas com sucesso nesta área, em diversos domínios de aplicação, entre eles: predição de séries temporais financeiras (Dablemont, 2003), previsão de vasão de rios (Atiya, 1999), modelagem de séries temporais biomédicas (Coyle, 2005) e predição de tráfego de rede (Doulamis, 2003).

Série temporal é uma coleção de observações realizadas em ordem cronológica. As formas de se trabalhar na previsão de séries temporais se tornaram mais complexas com o tempo, sempre na busca de reduzir a distância entre o valor previsto e o valor desejado na previsão. Para Kalyvas (2001), o melhor a se fazer é tentar reduzir tanta incerteza envolvida. Outro problema é saber quais dados (atributos) podem ajudar na previsão de uma série temporal. A calibragem de um sistema previsor, na sua forma mais simples, consiste em várias simulações com ajustes de parâmetros e atributos utilizados em um grande experimento com tentativas e erros.

As principais contribuições deste trabalho são a utilização de dados clusterizados com a rede de Kohonen para tratar cada neurônio da grade como uma classe de dados distinta e realizar o projeto de uma rede previsora otimizada especificamente para os dados daquela classe. Na etapa do projeto da rede previsora, a responsabilidade do usuário é apenas informar as séries temporais disponíveis. Todo o processo é automatizado, da topologia da rede até a seleção dos atributos (que são séries temporais), empregando-se algoritmo genético e os subconjuntos de exemplos associados aos seus respectivos neurônios na grade (seleção de exemplos). Esta é uma forma de dividir o problema em partes menores, mais específicas, facilitando o trabalho dos previsores reduzindo a necessidade de generalização dos mesmos, por agora trabalharem apenas com um conjunto restrito (e semelhante) de dados.

Mais contribuições deste trabalho são a utilização do erro médio de cada previsor, não apenas como uma medida de avaliação do sistema, como também uma medida de segurança na tomada de decisão de qual previsor é responsável por cada padrão de entrada, realizando assim, a seleção automática dos previsores, fazendo com que a união de vários métodos de previsão apenas razoáveis resultem em uma estratégia que se destaca pelo seu bom desempenho. Também foram definidos dois comitês de máquinas que aliam as informações de semelhança entre os padrões de entrada advindas da clusterização com a combinação de conhecimentos dos especialistas obtida através do comitê de máquinas.

McNelis (2005) diz que, para aplicar métodos de previsão e diagnósticos, é necessário ter um conhecimento particular para pré-processamento dos dados visando uma convergência ótima. A proposta deste trabalho busca reduzir o trabalho do usuário do sistema de previsão no que diz respeito à configuração dos parâmetros, pois todas são automatizadas. Além da facilidade de utilização, esta proposta se mostra eficaz também ao conseguir reduzir significativamente o erro médio das previsões em todas as medidas de avaliação realizadas, inclusive melhorando a qualidade dos resultados, como pode ser observado pelo baixo desvio padrão dos resultados.

Para avaliar o desempenho da proposta, foram realizados testes utilizando séries temporais econômicas na tentativa de prever o preço de fechamento de ativos no dia seguinte (D+1). Foram utilizadas nove séries temporais de três ativos distintos num período compreendido entre os anos de 2000 e 2008, onde a medida de avaliação utilizada primariamente é o RMSE, porém também foram aferidos o MAPE e o MAE dos resultados obtidos, assim como o desvio padrão de todas as medidas nas trinta execuções realizadas.

1.1. Motivação

Os seres humanos tentam se antecipar a determinados eventos há séculos, desde o tempo onde a maior preocupação era se ia chover muito ou não naquela estação até hoje quando se tenta descobrir qual será o preço de fechamento de um determinado ativo na bolsa de valores. Para realizar as previsões, eles se baseiam nos eventos passados para tentar prever como será no futuro, ao "anotar" o comportamento passado, associado a alguma medida de tempo, estão criando séries temporais. Dependendo das características da mesma, a tarefa de previsão se torna difícil ao ponto de não ser possível acertar com exatidão. Começa, então, a busca pelo erro mínimo na previsão.

O projeto de uma RNA, quando bem estruturado, com todos seus parâmetros de treinamento otimizados e uma arquitetura adequada, resulta em uma melhora na sua capacidade de generalização, na tolerância a ruídos e em um aprendizado mais exato e rápido, consequentemente garantindo uma boa previsão. Este processo de calibração de redes neurais é uma tarefa que exige uma série de testes, sendo necessário realizar ajustes nos parâmetros conforme seja a análise dos resultados obtidos, tarefas que requerem certo conhecimento do usuário, tanto nas heurísticas de previsão como no domínio de aplicação do problema. Além da calibração da rede previsora, os dados utilizados no treinamento da mesma também afetam a qualidade das previsões.

Este trabalho apresenta uma abordagem para realizar previsão de séries temporais que busca minimizar a diferença entre as previsões obtidas pela abordagem e o resultado esperado para os mesmos. Também é uma motivação para este trabalho retirar do usuário a responsabilidade por toda a calibração do previsor requerendo que o mesmo apenas forneça as séries temporais disponíveis para o domínio de aplicação. As tarefas de decidir a topologia das redes utilizadas, escolher quais atributos realmente serão utilizados como entrada de cada rede, selecionar quais exemplos serão fornecidos como conjunto de entrada no treinamento e até mesmo avaliar o desempenho de cada previsor ficam sob a responsabilidade do sistema.

1.2. Objetivos

1.2.1. Objetivo geral

De uma forma geral, este trabalho tem como objetivo apresentar uma abordagem para realizar a previsão de séries temporais dividindo o problema em partes menores, utilizando estratégias de previsão com altas taxas de erros, se avaliadas em todo o conjunto de teste, para montar um sistema com baixa taxa de erro e pequena variação dos resultados. Tudo de forma automática, sem sobrecarregar o usuário.

1.2.2. Objetivos específicos

Especificamente, esta abordagem se propõe a:

1. Melhorar a taxa de erro através da clusterização dos dados por rede de Kohonen e projeto automático de redes MLP específicas para cada classe de dados;
2. Apresentar dois comitês de máquinas que utilizam os conceitos de semelhança dos dados obtidos pela clusterização e as redes especialistas criadas pelo projeto automático;
3. Introduzir o conceito da análise independente de previsores e propor uma estratégia que monta uma grade tridimensional para permitir que várias estratégias medianas formem um sistema com alto desempenho;
4. Comparar o desempenho da proposta com uma abordagem bastante utilizada na literatura que é a rede MLP com *backpropagation*;
5. Aplicar a heurística proposta no problema real da previsão de séries temporais econômicas;

6. Reduzir a sobrecarga do usuário tendo a única tarefa de apresentar para o sistema as séries temporais disponíveis.

1.3. Estrutura da dissertação

O presente trabalho encontra-se organizado da seguinte forma: o segundo capítulo apresenta alguns trabalhos relacionados à área de previsão de séries temporais, mostrando algumas abordagens utilizadas para esta tarefa, assim como ferramentas que auxiliam na minimização do erro, como: algoritmos genéticos e projeto automático, clusterização dos dados e comitê de máquinas. No terceiro capítulo serão abordadas todas as tecnologias utilizadas ao longo do trabalho, como as redes neurais, tanto de Kohonen como as MLP, algoritmos genéticos, seleção de atributos, projeto automático de redes neurais e comitê de máquinas. O quarto capítulo apresenta a abordagem proposta, esta foi dividida em módulos que podem ser visualizados entre as seções do capítulo. No quinto capítulo são apresentados os testes realizados com a implementação da abordagem descrita no capítulo anterior, algumas variações são feitas e também é realizada a avaliação dos resultados obtidos, em seguida é apresentada a conclusão do trabalho. Por fim, as referências bibliográficas utilizadas para a confecção deste trabalho são apresentadas.

2. ABORDAGENS PARA PREVISÃO DE SÉRIES TEMPORAIS

A proposta deste trabalho envolve previsão de séries temporais, clusterização de dados, seleção de atributos, projeto automático de redes MLP e comitê de máquinas. Devido à amplitude deste trabalho, não foram encontrados na literatura trabalhos relacionados que utilizem todas as tecnologias aqui empregadas. Desta forma os trabalhos relacionados apresentados neste capítulo, trabalham com, pelo menos, uma das tecnologias aplicadas atreladas a outros aspectos não mencionados.

O problema da previsão de séries temporais tem sido estudado há bastante tempo. Em 1970, George Box e Jenkins (Box, 1970) aplicaram modelos auto-regressivos de médias móveis (ARMA ou ARIMA) na tentativa de encontrar semelhanças em trechos passados da série temporal, de forma a ajudá-los na tarefa da previsão de dados futuros. Em 1986, foi criado um modelo de processos de heterocedasticidade condicional auto regressiva generalizada (GARCH) para modelar a volatilidade de séries temporais e estimar a variância condicional (Bollerslev 1986).

Abelém (1994) comparou o desempenho de redes MLP com o modelo Box-Jenkins aplicados na previsão do preço do ouro no mercado internacional. A conclusão foi que as RNAs se saíram superiores para aquelas séries temporais. Comrie (1997) mostrou que as redes MLP apresentam resultados um pouco superiores quando comparadas a métodos de regressão tradicionais, na previsão do nível de ozônio em diferentes cidades. Cho (2003) mostrou que as redes MLP se saíram melhores que o modelo ARIMA na previsão de turistas em Hong Kong, principalmente em casos onde não existe um padrão definido, como no caso da oscilação do número de turistas oriundos do Japão devido à crise asiática.

De Castro (2001) obteve melhores resultados com redes RBF (*Radial basis Function*) em comparação a métodos estatísticos convencionais em diversos domínios de aplicação de séries temporais: desde números de pacotes numa rede local, passando pela taxa de câmbio do dólar americano em relação ao franco suíço até a intensidade de manchas solares. Figueredo (2008) comparou o desempenho de redes RBF e o modelo Box-Jenkins aplicados aos dados de vendas de uma linha de produtos. Como resultado: ambas tiveram um bom desempenho, porém as redes RBF se mostraram melhores por apresentar menor RMSE.

Gomes (2005) utilizou redes neurais recorrentes Elman e Jordan para previsão de séries temporais econômicas, como a taxa de inflação dos Estados Unidos. Em comparação com os modelos ARIMA e ARFIMA, as redes neurais tiveram resultados superiores. Bastos (2010) comparou o desempenho de redes MLP e redes de Elman na previsão da arrecadação de

ICMS da Secretaria da Fazenda do Ceará. Como resultado: ambas apresentaram bons resultados com vantagens para as redes MLP.

Segundo Menezes (2006), o fato de muitas séries temporais serem ruidosas, não-lineares e, em alguns casos, caóticas, faz com que os modelos de RNA, geralmente, apresentem desempenho melhores que técnicas lineares tradicionais. Ainda em Menezes (2006), chega-se à conclusão que as técnicas lineares convencionais têm cedido cada vez mais espaço para técnicas não-lineares que conseguem capturar, com mais eficácia, a dinâmica dos sistemas complexos.

Além de redes neurais, Chagas (2008) utilizou redes bayesianas na previsão de crescimento de fluxos de caixa, em comparação com métodos econométricos tradicionais (Testes de regressão múltipla, por exemplo). Os resultados ficaram próximos.

Após várias pesquisas comparando diversos modelos de redes neurais com modelos matemáticos e estatísticos, começaram a surgir trabalhos buscando otimizar o desempenho das próprias séries, onde os desempenhos das mesmas são comparados com elas mesmas sem a abordagem para otimização proposta.

Para Vellido (1999), a utilização em conjunto de algoritmos genéticos e RNAs vem crescendo, já que os sistemas resultantes identificam e mapeiam melhor as relações não-lineares entre variáveis, trabalham com dados incompletos e ruidosos e não requisitam qualquer suposição *a priori* sobre os dados. Kim (2000) utilizou algoritmos genéticos para definir os pesos entre as conexões da rede neural. Moura (2006) utilizou um GA na sua dissertação para buscar uma configuração inicial e realizou diversos testes modificando (manualmente) a topologia da rede e os atributos na camada de entrada na previsão de tendência de ações do mercado brasileiro.

Iyoda (2000) utilizou algoritmo A* para definir a quantidade de neurônios na camada intermediária. Boozarjomehry (2001) utilizou algoritmos genéticos para automatizar o *design* de redes neurais e obteve bons resultados no processo de neutralização de pH. Sikora (2008) utilizou algoritmos genéticos na tentativa de automatizar a quantidade de neurônios na camada intermediária, conseguindo bons resultados nas topologias encontradas para previsão de preços do mercado elétrico. Kalyvas (2001) utilizou GAs para definir a quantidade de neurônios da camada de entrada e para as camadas intermediárias (que podiam ser até duas). A quantidade de neurônios na camada de entrada definida pelo GA era o tamanho da janela de tempo do único atributo utilizado no conjunto de entrada.

De Castro (2001) sugere a utilização de algoritmos genéticos para obtenção dos modelos que melhor representem os dados analisados. Amaral (2010) utilizou algoritmos

genéticos para determinação da melhor configuração das redes MLP e Elman (nós da camada de entrada e intermediária e taxa de aprendizagem das redes), cuja *fitness* era o RMSE, conseguindo uma melhora significativa apenas para as redes MLP.

Apesar de Antunes (2001) falar que a classificação não tem obtido muita atenção nas séries temporais, alguns trabalhos na área são encontrados. Chappelier (1996) classifica assinaturas levando em consideração a sequência de toques da caneta para "desenhá-la". Mantegna (1999) e Keogh (2002) utilizaram o algoritmo *Hierarchical Clustering* que busca agrupar os dados similares para formar *clusters* até que não existam dados sem classificação.

Lendasse (1998) clusteriza os dados utilizando redes de Kohonen de uma dimensão na previsão das séries temporais da competição de Santa Fé. Fu (2001) utilizou SOM para encontrar padrões dentro de séries temporais econômicas. Lin (2005), utilizou mapas auto-organizáveis e redes RBF para previsão de séries temporais, na sua implementação os neurônios da grade de Kohonen são utilizados para determinar os centros das bases radiais. Siqueira (2006) clusterizou os dados utilizando redes de Kohonen, depois fez uma análise da quantidade de dados que existia em cada *cluster*. As classes que não atingiam a quantidade mínima de dados estipulado por ele eram excluídas do modelo.

Os comitês de máquinas têm apresentado bons resultados na previsão de séries temporais. Os comitês são mais populares com o uso de redes neurais, mas existem outras implementações. Chen (2006) usou comitês de fórmulas empíricas (CMEF – *Committee Machine with Empirical Formulas*) para previsão, seus resultados foram melhores que cada fórmula trabalhando em separado. Tresp (2000) utilizou uma BCM (*Bayesian Committee Machine)* onde usou previsores baseados em processos de regressão gaussianos.

Adeodato (2009) conseguiu bons resultados com a implementação mais simples dos comitês que é a média aritmética das saídas dos especialistas, em seu trabalho, ele utilizou a saída de 30 redes MLP e calculou a média aritmética destas saídas para definir a saída geral do sistema.

Guo (2004) usou comitê na previsão de preços para reduzir a dificuldade da tarefa. Em sua pesquisa, ele combinou a saída das redes através de conexões que eram determinadas pela probabilidade de cada uma das redes capturarem o mapeamento da entrada. Os resultados foram superiores aos obtidos com redes MLP e RBF comuns e, também, um comitê de máquinas cuja saída era a média aritmética.

Villanueva (2006) compara alguns comitês de máquinas:

- *Ensemble-Averaging*: trata-se do cálculo da média aritmética das saídas, já explicado anteriormente;

- *Ensemble-bagging: Bagging* vem de *Bootstrap aggregating*. trata-se de uma técnica de geração de conjuntos distintos de treinamento a partir de um único conjunto, via re-amostragem, com reposição (Villanueva, 2006);
- Mistura de especialistas: dividiu o espaço de entrada em regiões e treinou uma rede neural com os dados de cada região dividida, as saídas de todas as redes treinadas são enviadas para outra rede (Rede *Gating*) que calcula a saída final do sistema.

Nos testes de Villanueva, os melhores resultados foram obtidos com a mistura de especialistas, seguido dos métodos *ensemble* e, por fim, redes MLP simples.

3. FUNDAMENTAÇÃO TEÓRICA

3.1. Redes Neurais

As redes neurais artificiais (RNA) foram inspiradas na estrutura e comportamento dos neurônios biológicos existentes no cérebro humano. Cada neurônio está conectado com alguns outros, formando uma rede e realizando um processamento contínuo e paralelo. Em 1943, com o trabalho de McCulloch e Pitts (McCulloch, 1943) foi proposto o primeiro modelo de neurônio artificial. A Figura 1 ilustra o esquema de neurônio artificial de McCullock e Pitts.,

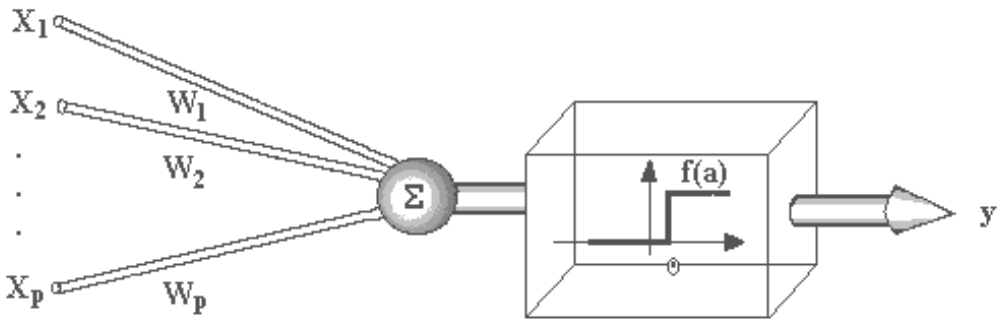


Figura 1 – Esquema de neurônio artificial de McCullock e Pitts (ICMC, 2011)

A Equação (1) descreve como a saída de cada neurônio é calculada:

$$y = f\left(\sum_{i=1}^{p} x_i w_i\right) \tag{1}$$

onde:

- X_i representa as p entradas;
- W_i representa os p pesos associados às entradas. Basicamente as entradas são multiplicadas por seus respectivos pesos, podendo gerar sinais positivos (excitatórios) ou negativos (inibitórios);
- Σ representa o combinador linear, que executa o somatório dos sinais calculados pelos produtos das entradas por seus pesos;
- $f(a)$ representa a função de ativação da saída do combinador linear, neste caso, a função degrau;
- y representa a saída calculada.

Outras publicações iniciais importantes foram: (a) um modelo básico de auto-organização da rede por Hebb (1949) e (b) o modelo *Perceptron* de aprendizado supervisionado por Rosemblatt (1958). Desde então, a ideia vem amadurecendo, funções de

ativação foram propostas e, segundo Hung (1996), como os neurônios podem ser conectados de formas diferentes, várias arquiteturas de redes neurais foram criadas. As principais funções de ativação são:

- **Função Sigmóide:** Esta função, formalizada pela Equação (2), com comportamento gráfico ilustrado na Figura 2, pode assumir valores v entre 0 e 1, onde $v \in \mathbb{R}$

$$f(a) = sgm(a) = \frac{1}{1+e^{-a}} \tag{2}$$

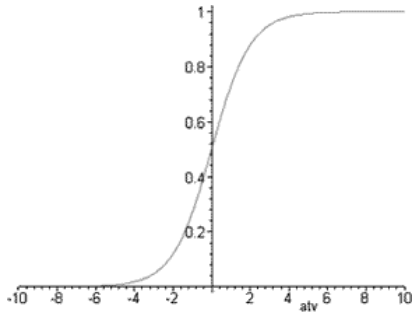


Figura 2 – Gráfico da função sigmoide.

- **Função Tangente Hiperbólica:** Esta função, formalizada pela Equação (3), com comportamento gráfico ilustrado na Figura 3, pode assumir valores v entre -1 e 1, onde $v \in \mathbb{R}$.

$$f(a) = \tanh(a) = \frac{1 - e^{-a}}{1 + e^{-a}} \tag{3}$$

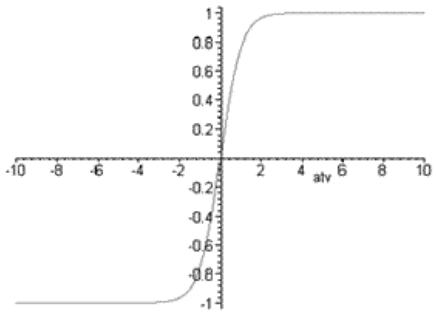


Figura 3 – Gráfico da função tangente hiperbólica**.**

Segundo Guo (2004), a principal vantagem de uma rede neural é que ela é capaz de inferir (mapear) relacionamentos "escondidos" entre os dados. Para que isto ocorra, a rede deve passar por uma fase de treinamento. Haykin (2001) define aprendizado de RNA como um processo pelo qual os parâmetros de uma rede neural são adaptados através de um processo contínuo de estimulação pelo ambiente no qual a rede está inserida. O tipo de

aprendizado é determinado pela forma em que a mudança dos parâmetros ocorre. Os métodos de aprendizado de RNAs se dividem em duas categorias:

- **Aprendizado Supervisionado**: Este tipo de treinamento, segundo Moura (2006), utiliza em sua estrutura uma espécie de instrutor que confere o quanto a rede está próxima de uma solução aceitável. No treinamento supervisionado, um conjunto de dados é apresentado como entrada para a rede e, associado a este conjunto de entrada, é apresentado um conjunto de saída. O algoritmo de aprendizado busca ajustar os pesos das conexões entre os neurônios de tal forma que, para o conjunto de entrada informado, a rede seja capaz de calcular uma saída o mais próximo possível da saída informada.
- **Aprendizado Não Supervisionado**: Conhecido também como aprendizado auto-supervisionado por não possuir instrutores. Este tipo de treinamento utiliza apenas as informações das entradas e realiza o aprendizado através de agrupamentos e conceitos de vizinhança. Muito utilizado para encontrar padrões entre os dados.

O domínio de aplicação das redes neurais é vasto, mas tende a cair nas seguintes categorias gerais:

- Aproximação de funções ou análise de regressão, incluindo a previsão de séries temporais e modelagem;
- Classificação, reconhecimento de sequência, detecção de novidade e tomada de decisão sequencial;
- Processamento de dados, incluindo filtragem, agrupamento.

3.1.1. MLP

A rede *multilayer perceptron* (MLP) representa uma generalização do *perceptron* proposto por Rosemblatt (1958). Sua arquitetura consiste em várias camadas compostas por nós (neurônios) onde cada neurônio de uma camada possui ligação com todos os neurônios da camada seguinte. A rede MLP possui uma camada de entrada, que atua como os sensores da rede captando os estímulos do ambiente; pode ter uma ou mais camadas intermediárias, que é onde a maior parte do processamento é realizada; e uma camada de saída onde o resultado final é concluído e apresentado. Não existem conexões entre a saída de um neurônio e algum outro neurônio localizado em uma camada anterior ao primeiro, ou seja, não existem ciclos,

fato que caracteriza uma rede *feedforward.* A Figura 4 exibe uma rede MLP com todos os pesos entre as conexões.

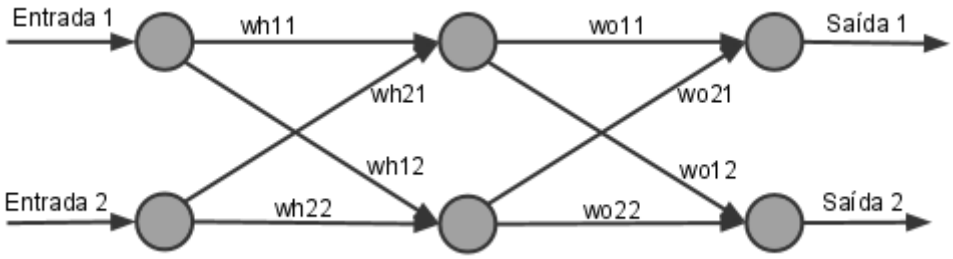


Figura 4 - Rede MLP com pesos entre conexões. Adaptado de Crochat (2009).

Na figura, W_{ij}^{h} representa o peso correspondente da ligação entre o neurônio da camada de entrada i e o neurônio j da camada oculta (*hidden*); W_{ij}^{o} representa o peso correspondente da ligação entre o neurônio da camada oculta i e o neurônio j da camada de saída (*output*). Um dos algoritmos mais utilizados para treinar a rede MLP é o *Backpropagation,* proposto por Rumelhart (1986). Este algoritmo busca retropropagar o erro entre a saída desejada e a saída obtida visando minimizar o erro total da rede, no decorrer do treinamento. O algoritmo *backpropagation*, adaptado de Crochat (2009), pode ser resumido nos seguintes passos:

- A saída de cada um dos M neurônios da camada oculta $H_{1\ldots M}$ é calculada conforme a Equação (4) a seguir:

$$H_m = sgm\left(\sum_{i=1}^{N_i} I_i W_{im}^{h}\right) \tag{4}$$

  onde $smg(.)$ é a função sigmoide definida em (2), I_i representam os estímulos vindos da camada de entrada (*input*) e N_i é a quantidade de neurônios na camada de entrada;

- A saída de cada um dos N neurônios da camada de saída $O_{1\ldots N}$ é calculada empregando-se (5) a seguir:

$$O_n = sgm\left(\sum_{i=1}^{N_h} H_i W_{ni}^{o}\right) \tag{5}$$

  onde N_h é a quantidade de neurônios na camada intermediária;

- Um passo na retropropagação para um padrão de entradas e saídas específicas começa com o cálculo do erro nas n saídas dos neurônios (ε_n^os). Este erro é a diferença entre a saída calculada O_n e a saída desejada D_n multiplicada pela

derivada da sigmóide da saída do neurônio $\varphi(O_n)$, conforme ilustrada na Equação (6).

$$\varepsilon_n^o = (D_n - O_n)\varphi(O_n) \quad (6)$$

onde a derivada da sigmoide $\varphi(x)$ é calcula a partir da Equação (7);

$$\varphi(x) = \frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right) = \frac{e^{-x}}{(1+e^{-x})^2} = sgm(x)(1 - sgm(x)) \quad (7)$$

- O erro é calculado pelo somatório do produto entre os erros dos neurônios da camada seguinte e seus pesos respectivos, multiplicado pela derivada da sigmoide da saída do neurônio, conforme Equação (8) que ilustra a retropropagação para a camada intermediária;

$$\varepsilon_m^h = \sum_{i=1}^{N_o} (O_i W_{mi}^o)\, \varphi(H_m) \quad (8)$$

- Após a definição do erro, deve-se calcular o gradiente, ou delta, de cada conexão dos neurônios da camada intermediária realizando o produto entre a saída do neurônio e o erro do neurônio da camada seguinte, utilizando a Equação (9).

$$\Delta W_{ij}^o = H_i \varepsilon_j^o \quad (9)$$

- Para calcular o gradiente de cada conexão dos neurônios da camada de entrada com os neurônios da camada intermediária, utiliza-se a Equação (10).

$$\Delta W_{ij}^h = I_i \varepsilon_j^h \quad (10)$$

- Após a definição do gradiente, o último passo é a atualização dos pesos das conexões para concluir o processo de atualização dos pesos e correção dos erros. Cada peso é calculado agregando-se ao valor dele ao produto do gradiente calculado para o mesmo e a taxa de aprendizado ϕ definida (*a priori*) para o treinamento, conforme Equação (11).

$$\mathrm{W}_{ij} = \mathrm{W}_{ij} + (\Delta \mathrm{W}_{ij} \phi) \quad (11)$$

No algoritmo acima foi citada a taxa de aprendizagem do treinamento. Além deste existem outros parâmetros a serem configurados para a rede. Cada um tem suas implicações:

- **Taxa de aprendizagem:** Taxas de aprendizagem altas implicam em maiores mudanças nos pesos (conforme percebido na Equação (11)), o que em geral aumenta a intensidade de atualização dos pesos, causando uma maior oscilação do modelo. Deve-se buscar um meio termo para evitar oscilações e garantir um treinamento rápido.
- **Quantidade de épocas:** Responsável por indicar a quantidade de vezes que o conjunto de entrada deve ser apresentado à rede durante o treinamento. Uma maior quantidade de épocas implica em um maior aprendizado por parte da rede para aquele conjunto de dados, porém reduz a capacidade de generalização da rede. Segundo Magri (2011), deve-se encontrar um ponto ótimo de parada com erro mínimo e capacidade de generalização máxima. Como é muito difícil de existir tal ponto, normalmente procura-se encontrar um meio termo.
- **Quantidade de camadas intermediárias:** A arquitetura MLP não impõe restrições em relação à quantidade de camadas ocultas na topologia de rede, porém deve-se tomar cuidado, pois é nestas camadas que os mapeamentos não lineares entre entradas e saídas são processados.
- **Quantidade de neurônios:** A arquitetura MLP também não impõe restrições em relação à quantidade de neurônios em cada camada.
    - A camada de entrada deve refletir os atributos (variáveis) que são fornecidas ao sistema.
    - As camadas intermediárias são mais complicadas de se definir um valor por não terem relação direta nem com o conjunto de entrada, nem com o conjunto de saída. Em geral, redes com um pequeno número de neurônios possuem uma melhor capacidade de generalização, porém um número pequeno de neurônios pode ser insuficiente para modelar o problema (Zhang, 1998).
    - A camada de saída é a mais fácil de definir a quantidade de neurônios, pois está diretamente relacionada com o problema em análise, com o que se deseja prever.

Ainda em relação ao aprendizado, deve-se tomar cuidado para não realizar um superaprendizado onde a rede não fica “viciada” nos dados de entrada e reduz a capacidade de generalização ao serem apresentados dados fora dos padrões treinados, caracterizando

overfitting. Segundo Kohavi (1995), é possível identificar um *overfitting* quando os dados apresentados estão bem modelados, mas as previsões estão com altas taxas de erro. Isso quer dizer que o sistema está se viciando no conjunto de treinamento e está perdendo a capacidade de generalizar para dados externos a este conjunto.

3.1.2. Redes de Kohonen

Segundo Kohonen (1997), um mapa auto organizável de Kohonen, ou SOM (*Self-Organising Map*), é um algoritmo neural não supervisionado usado para extrair regularidades estatísticas presentes em conjuntos de dados de entrada. As redes de Kohonen permitem representar dados multidimensionais em espaços de dimensões bem menores. Segundo Liao (2005), é uma classe de redes neurais de poucas dimensões (geralmente duas) treinadas por um processo iterativo e não supervisionado ou procedimento auto-organizável.

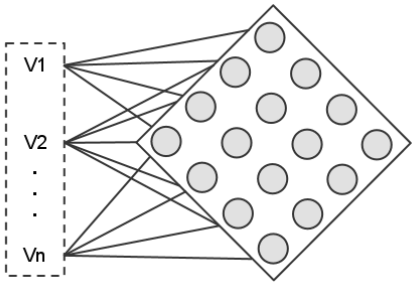


Figura 5 – Rede de Kohonen.

A Figura 5 ilustra uma rede de Kohonen de duas dimensões e o conjunto de entrada $V_{1\ldots n}$, onde cada neurônio da grade está totalmente conectado às n unidades de entrada V, através de pesos sinápticos agrupados no vetor w. Esta rede tem uma estrutura de propagação *feedforward* com apenas uma camada computacional de neurônios, normalmente dispostos em uma configuração linear, retangular ou hexagonal (Bastos, 2007).

O algoritmo de treinamento da rede de Kohonen é não supervisionado e utiliza uma estratégia com três passos envolvidos na formação do mapa auto-organizável: competição, cooperação e adaptação. Para realizar o treinamento, devem-se seguir os seguintes passos:

- Inicializar randomicamente todos os pesos sinápticos, para que nenhuma organização prévia seja imposta à rede;
- Para cada entrada, deve-se definir quem é o BMU (*Best Matching Unit*), chamado a partir de agora como “Melhor neurônio”, que é o neurônio com menor distância euclidiana da entrada. Esta é a parte competitiva do

algoritmo, onde o neurônio vencedor será ativado. A distância euclidiana é calculada conforme Equação (12), onde V é o vetor de entrada, W é o vetor de pesos do neurônio e n é o tamanho do vetor;

$$\mathrm{d}_E(V,W) = \sqrt{\sum_{i=1}^{n}(V_i - W_i)^2} \tag{12}$$

- Sabendo qual o melhor neurônio, calcula-se o tamanho do raio de abrangência que reduz o seu tamanho a cada iteração conforme Equação (13). Considere t o contador da iteração para definir qual será o tamanho do raio naquela iteração e T a quantidade total de iterações; σ_0 representa o raio inicial, ou seja, na primeira iteração, obtido através de (14); e λ é uma constante de tempo que é calculada através da função exponencial definida na em (15);

$$\sigma(\mathrm{t}) = \sigma_0 e^{(-\frac{t}{\lambda})} \tag{13}$$

$$\sigma_0 = \frac{\max(numLinhas, numColunas)}{2} \tag{14}$$

$$\lambda = \frac{T}{\log \sigma_0} \tag{15}$$

- De posse do raio, resta apenas atualizar os pesos dos neurônios (vizinhos) que estão dentro do perímetro do raio calculado conforme (13). Representando a parte cooperativa e adaptativa do algoritmo. Os novos valores para o peso são calculados conforme (16);

$$\mathrm{W(t+1) = W(t) + \Theta(t)L(t)(V(t) - W(t))} \tag{16}$$

- A influência da distância entre o neurônio em questão e o melhor neurônio é calculada através da função gaussiana definida em (17). Considerando $dist$ como a distância euclidiana entre o neurônio e o melhor neurônio;

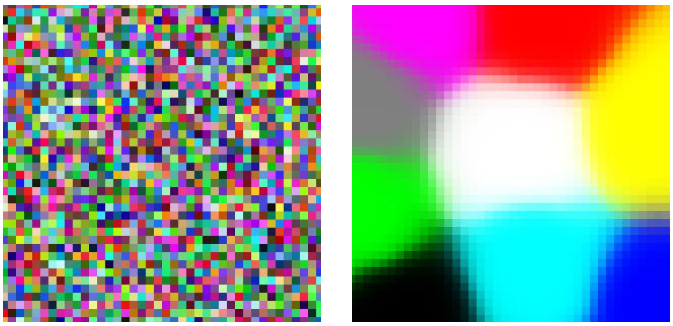
$$\Theta(\mathrm{t}) = e^{(-\frac{dist^2}{2*\sigma^2(t)})} \tag{17}$$

- A taxa de aprendizado, que também é reduzida ao longo das iterações é calculada empregando-se (18). Considerando L_0 a taxa de aprendizado definida (*a piori*) para o treinamento.

$$L(t) = L_o e^{(-\frac{t}{\lambda})} \tag{18}$$

Segundo Magri (2011), o fato de $\sigma(t)$ ser reduzido com o passar do tempo caracteriza uma vizinhança mais restrita e, portanto, mais especializada. Já o motivo de $L(t)$ também decrescer com o tempo é para evitar que dados novos, apresentados após um longo período de treinamento, venham a comprometer seriamente o conhecimento que já está sedimentado.

Uma forma de exemplificar o funcionamento da rede de Kohonen é mapear as cores definidas num vetor tridimensional (RGB: *Red, Green, Blue*) para duas dimensões apenas. A Figura 6(a) exibe a situação dos pesos sinápticos antes do treinamento e a Figura 6(b) mostra o resultado final do processamento. É possível notar que os pesos foram atualizados de maneira tal que os *cluster*s ficaram facilmente visíveis.



(a) **(b)**

Figura 6 – Pesos sinápticos antes do treinamento **(a).** Pesos sinápticos após o treinamento **(b)**

As redes de Kohonen utilizam o conceito de vizinhança, tanto para atualizar os pesos durante o treinamento, como para definir o padrão que mais se assemelha ao dado de entrada apresentado. Segundo Barreto (2001), elas possuem várias características interessantes. Dentre elas se destacam sua poderosa habilidade de classificação e suas propriedades de abstração/generalização resultantes do aprendizado. Após o treinamento, a grade se torna seletivamente sintonizada a vários padrões de entrada (Kohonen, 1997).

3.1.3. Casamento de Densidade com Redes de Kohonen

Uma característica importante das redes de Kohonen é o casamento de densidade. Esta característica implica que a distribuição dos dados de entrada não é uniforme, existem variações de tal forma que as regiões dos espaços de entrada de onde os vetores de treinamento são retirados com maior frequência são mapeadas para domínios maiores no espaço de saída. Em contrapartida, as regiões de entrada de onde os vetores de treinamento

são retirados com menor frequência são mapeadas para domínios menores no espaço de saída (Rodrigues, 2010).

Para calcular a densidade em problemas de alta dimensionalidade utilizando redes de Kohonen é utilizado o conceito de mistura de modelos, que representa a distribuição por meio de uma combinação linear de algumas funções de *kernel* (Bishop, 1995). A densidade é calculada por (19).

$$p(x) = \sum_{j=1}^{m} P(j)p(x|j) \tag{19}$$

onde m representa a quantidade de neurônios da SOM; $P(j)$ é a probabilidade *a priori* do dado pertencer à célula do diagrama de Voronoi correspondente ao neurônio j, calculada em (20); e $p(x|j)$ representa a densidade condicional do vetor desejado x para aquele *kernel*.

$$P(j) = \frac{N_j}{N} \tag{20}$$

onde N_j representa a quantidade de dados mapeados para a célula j do diagrama de Voronoi; e N representa a quantidade total dos dados apresentados. Sendo que a somatória de todas as probabilidades deve ser igual a 1; e $0 \leq P(j) \leq 1$. Já as probabilidades p(x|j) devem ser normalizadas para satisfazer a condição em (21). Para funções de *kernel*, uma escolha comum é a gaussiana.

$$\int p(x|j)dx = 1 \tag{21}$$

3.2. Algoritmos Genéticos

Os algoritmos genéticos são algoritmos de busca baseados nos mecanismos de seleção natural e genética que combinam a sobrevivência entre os melhores com uma forma estruturada de troca de informação genética entre dois indivíduos para formar uma estrutura heurística de busca (Linden, 2006). Para a definição deste algoritmo as teorias da seleção natural de Darwin foram observadas, assim como a genética natural de Mendel.

As soluções possíveis para o problema de otimização são representadas em uma estrutura simbólica denominada cromossomo. A função de objetivo do problema é representada por uma função *fitness* que informa o quão boas (aptas) são as soluções para o problema em questão. Inicialmente, um conjunto de cromossomos aleatórios é gerado (formando a população inicial). Estes elementos são selecionados em pares que passam por

operações de cruzamento, onde as características dos cromossomos são combinadas, visando formar novos indivíduos mais aptos (com melhor função de avaliação). Os cromossomos também passam por operações de mutação, onde as características dos cromossomos são alteradas aleatoriamente. A cada ciclo, os indivíduos mais aptos formam uma nova geração e assim por diante até que algum outro critério de parada seja alcançado. A Figura 7 possui um exemplo de algoritmo genético.

Algoritmo 1: Algoritmo Genético

```
função algoritmo_genetico (populacao , calc_fitness ) retornar um_individuo
        entradas: populacao //conjunto de indivíduos
                  calc_fitness //função que calcula a aptidão do indivíduo
1       repetir
2           nova_populacao ← conjunto vazio;
3           para i de 1 até tamanho(populacao) faça
4               x ← seleção_randômica(populacao,calc_fitness);
5               y ← seleção_randômica(populacao,calc_fitness);
6               filho ← reproduz(x ,y );
7               se (pequena probabilidade aleatória) então filho ← mutação(filho );
8               adicionar filho a nova_populacao;
9           populacao ← nova_populacao;
10      até algum indivíduo estar adaptado o suficiente ou ter decorrido tempo suficiente
11      retornar o melhor indivíduo em populacao de acordo com calc_fitness
```

Figura 7 – Algoritmo Genético. Adaptado de Russel (1995) por Sikora (2011).

Os cromossomos, na grande maioria das vezes são representados como um vetor de bits, onde cada bit (ou grupo de bit) é responsável por um parâmetro da solução, assim como o cromossomo biológico consiste de genes que são blocos de sequências de DNA e controlam uma característica hereditária específica do indivíduo. As principais operações realizadas no algoritmo genético estão enumeradas na Figura 8:

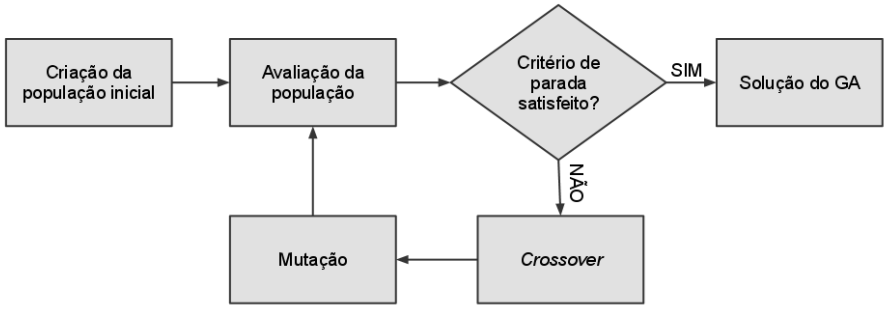


Figura 8 – Fluxograma com as principais operações do GA.

- **Geração da população inicial:** a geração da população inicial é importante, pois é responsável por fornecer o ponto de partida do algoritmo. A maneira

mais comum de geração é através da geração aleatória que busca criar uma população inicial diversificada;

- **Avaliação da população:** etapa onde cada indivíduo da população é avaliado para definir uma medida de aptidão para cada um deles. A função de avaliação leva em consideração a função objetivo que se deseja otimizar;
- **Cruzamento:** também conhecido como *crossover*, nesta etapa é realizado o cruzamento de informações entre dois indivíduos selecionados através do sorteio (geralmente usa a roleta viciada onde os indivíduos mais aptos, ou seja, com melhor avaliação possuem maior probabilidade de serem escolhidos), a Figura 9(a) ilustra um dos métodos para realizar o cruzamento de cromossomos. Uma das vantagens do GA vem deste ponto, pois as chances de dois indivíduos bons serem escolhidos são altas e neste ponto suas características são mescladas na busca de criar um indivíduo ainda melhor;
- **Mutação:** nesta etapa, quando um indivíduo é selecionado para sofrer mutação, o algoritmo realiza alguma alteração aleatória no gene deste cromossomo, a Figura 9(b) ilustra um dos métodos para realizar a mutação de cromossomos. Esta medida busca aumentar a diversificação da população, tentando levar o cromossomo para algum outro ponto no espaço de soluções;
- **Critério de parada:** para evitar que o algoritmo fique processando eternamente, é necessário definir critérios de parada. Os mais comuns são: tempo, quantidade de gerações, valor na função objetiva, muito tempo sem melhorias.

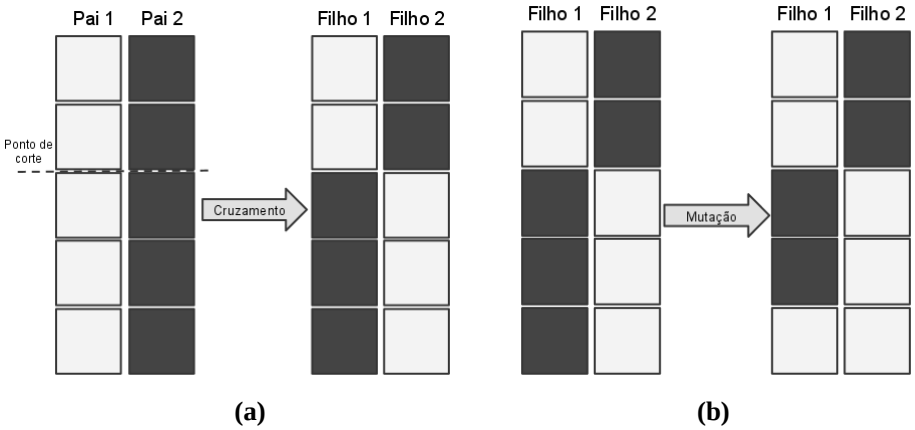


Figura 9 – Operação de crossover com um ponto de corte **(a)**. Operação de mutação ocorrida no Filho 1 **(b)**.

Os algoritmos genéticos possuem uma larga aplicação em muitas áreas especificas, por exemplo: programação genética, gerenciamento de redes, computação evolutiva,

composição musical, otimização evolutiva multicritério, problemas de otimização complexos, ciências biológicas, etc.

3.3. Seleção de Atributos

Para realizar o treinamento de uma rede neural MLP, além de definir o que a rede deve aprender (classificação, previsão, etc.), deve-se definir quais dados serão utilizados como entrada para a mesma. Segundo Heaton (2005), cada neurônio de entrada deve representar uma variável independente que tem alguma relação com a saída da rede neural. Por exemplo, em séries temporais, cada instância representa um ponto no tempo diferente e os atributos trazem informações associadas àquele momento. Na previsão de séries temporais, cada atributo é uma série temporal relacionada à(s) série(s) que se deseja realizar a previsão.

A seleção de atributos consiste em selecionar um subconjunto do conjunto de atributos disponíveis que maximize o desempenho da rede e minimize o ruído adicionado ao modelo. A seleção de atributos (variáveis) utilizadas no processo de treinamento influi bastante na qualidade do aprendizado da rede. Dependendo do domínio de aplicação, existem diversas variáveis disponíveis para análise, entretanto muitas destas variáveis são irrelevantes ou redundantes. No caso das redes neurais, a introdução de mais atributos na camada de entrada não necessariamente estará introduzindo novas informações, mas poderá estar introduzindo novos ruídos.

Quanto às características dos atributos, estes podem ser:

- **Atributos de relevância fraca:** são atributos que, em alguns momentos melhoram o desempenho da ferramenta de aprendizado. Removê-los implica em nenhuma ou baixa redução no desempenho;
- **Atributos de relevância forte:** são atributos que sempre melhoram o desempenho da ferramenta. Não utilizá-los implica em perdas significativas de desempenho;
- **Atributos irrelevantes:** são atributos que não contribuem para o desempenho da ferramenta, pelo contrário, apenas adicionam ruído ao sistema;
- **Atributos redundantes:** são atributos que, apesar de serem relevantes, as informações obtidas a partir deles já são conhecidas através de outros atributos.

Agrupando os trabalhos de Souza (2004) e Guyon (2003), existem quatro tipos de algoritmos para seleção de atributos. Os **Filtros** (ou *filters*) escolhem os atributos

independentes do algoritmo de aprendizagem. Os ***Wrappers*** utilizam a ferramenta de aprendizagem em forma de caixa preta como parte da medida de avaliação da solução (Kohavi, 1996), conforme ilustrado na Figura 10. Os filtros gastam poucos recursos computacionais em relação aos algoritmos *wrappers*, porém têm baixo desempenho. Os algoritmos *wrappers* têm desempenho muito superior aos filtros, porém requerem mais recursos computacionais devido ao fato de realizar treinamentos (caixa preta) para avaliar cada possível solução. Os algoritmos **Híbridos** procuram encontrar um meio termo entre os dois tipos de algoritmos já citados buscando combinar a velocidade de processamento dos *filters* e os bons resultados dos *wrappers*. Nos algoritmos **Embutidos**, o processo de seleção de atributos está inserido dentro do algoritmo de treinamento da ferramenta de aprendizagem.

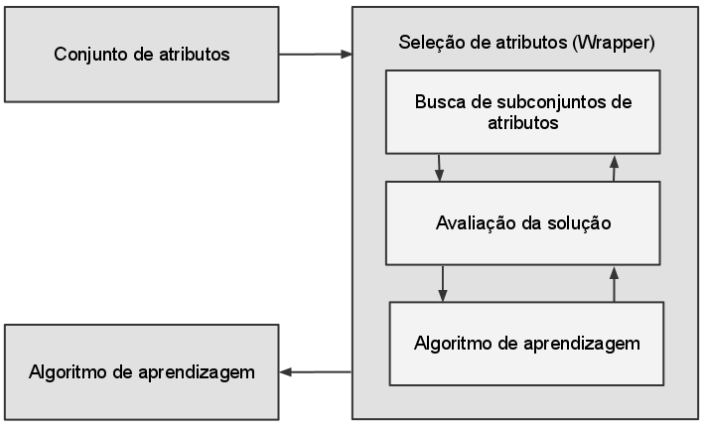


Figura 10 - Fluxograma para seleção de atributos dos algoritmos tipo Wrapper . Adaptadas de Kohavi (1996).

Os principais benefícios esperados de uma boa seleção de atributos são: a quantidade de dados de treinamento será menor, o tempo de execução do treinamento será menor, a taxa de erro da ferramenta de treinamento será menor, os modelos serão mais simples e precisos, e os custos com armazenamento serão reduzidos (Souza, 2004).

3.4. Projeto automático de Redes Neurais

Redes neurais com o mesmo conjunto de entrada, mas com estruturas distintas possuem significativa diferença no seu desempenho. Vários estudos foram realizados na tentativa de definir uma configuração para a rede neural de forma a obter melhores resultados. Por exemplo, Lipmann (1987), Lapedes (1988), Cybenko (1988) e Zhang (1994) concluíram que redes MLP duas camadas ocultas são suficientes para a resolução da grande maioria dos problemas; Zhang (1998) conclui que uma camada oculta é suficiente para a maioria dos

problemas de previsão, porém, em determinados problemas duas camadas ocultas levam a melhores resultados.

Zhang (2003) realizou um levantamento de alguns trabalhos na área e concluiu que a maioria dos estudos encontrados (29 artigos) se baseou apenas em uma camada intermediária, sendo que somente sete artigos consideraram suas redes com duas camadas intermediárias. Existem algumas heurísticas para determinar a quantidade de neurônios nas camadas ocultas relacionadas ao número de entrada, porém não é um padrão que possa ser seguido para todos os tipos de problema (Zhang, 1998). A grande maioria dos autores determina um número de entradas de maneira empírica ou por tentativa (Moura, 2006).

A tarefa de definir a melhor configuração para uma rede neural tem se mostrado um trabalho exaustivo através de sequências de tentativas e erros. O esforço se torna ainda mais oneroso quando se procura calibrar outras características da rede além da quantidade de camadas intermediárias, como a quantidade de neurônios nas camadas intermediárias, a taxa de aprendizado ou a quantidade máxima de épocas utilizadas na etapa de treinamento. Surgiu então a necessidade de automatizar este trabalho de tentativa e erro, ou seja, a criação de algoritmos de projeto automático para redes neurais.

Existem vários algoritmos para buscar, de forma automática, uma topologia de rede neural ótima para um determinado problema, alguns deles foram listados por Iyoda (2000):

- **Métodos construtivos:** procuram encontrar a arquitetura ótima para o problema partindo de uma arquitetura mínima. Como exemplo tem-se o algoritmo *cascade-correlation* (CASCOR) proposto por Fahlman (1990);
- **Métodos de poda:** iniciam com uma arquitetura mais complexa e buscam melhorar seu desempenho com a poda de unidades ou conexões;
- **Métodos inspirados em técnicas estatísticas:** são algoritmos baseados em modelos de regressão por busca de projeção. Como exemplo tem-se o Aprendizado por Busca de projeção, proposto por Von Zuben (1996);
- **Métodos otimizados por meta-heurísticas:** modelam o problema para serem executados em meta-heurísticas na busca de uma arquitetura ótima. Iyoda (2000) utiliza o algoritmo A*, onde os nós da árvore de busca contém informações como as soluções possíveis para o problema e os arcos contém informações sobre custos.

As redes evolutivas, que utilizam meta-heurísticas, consistem em uma forma para buscar a configuração que melhor se adeque ao problema em questão, sem a necessidade de

um especialista no assunto para fazer análises detalhadas nos parâmetros de calibração da rede.

3.5. Comitê de Máquinas

Os comitês de máquinas são um tipo de arquitetura que combina a saída de vários especialistas como, por exemplo, várias redes neurais em uma saída apenas. Segundo Haykin (2001), estas máquinas empregam a estratégia "Dividir para Conquistar", onde uma tarefa computacional complexa é resolvida dividindo-a em um número de tarefas computacionais simples e então combinando as soluções destas tarefas. A arquitetura mais conhecida, ilustrada na Figura 11, consiste de um grupo de especialistas, cada um contendo uma função de regressão geral como redes neurais, árvores de decisão ou algum outro tipo de algoritmo (Sharkey, 1996). Cada especialista executa a mesma tarefa e seus resultados são agrupados numa estrutura que irá combinar os resultados e gerar a saída final do sistema.

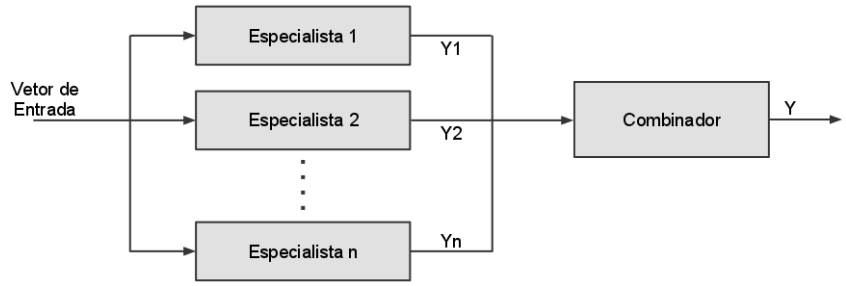


Figura 11 - Arquitetura padrão do comitê de máquina.

A forma mais simples de combinar o resultado é o cálculo da média aritmética dos resultados de todos os n especialistas (22) e costuma apresentar bons resultados (Naftaly, 1997; Adeodato, 2009).

$$Y = \frac{\sum_{i=1}^{n} Y_i}{n} \tag{22}$$

Segundo Chen (2006), como a saída final é resultado da combinação da saída de vários especialistas, o erro também será combinado. Mais especificamente, considerando que o comitê busca aproximar uma saída D, onde a saída de cada componente especialista do comitê possui um erro associado e_i, ou seja:

$$Y_i = D + e_i \tag{23}$$

então o erro quadrático na saída do i-ésimo especialista será:

$$E_i = (Y_i - D)^2, \tag{24}$$

e o erro quadrático médio considerando os n especialistas:

$$E_{médio} = \frac{1}{n}\sum_{i=1}^{n} E_i \tag{25}$$

Bishop (1995) aponta alguns aspectos desfavoráveis apresentados pelas RNAs que motivaram o estudo do comitê de máquinas (Villanueva, 2006), como: a convergência para um mínimo local e o risco de sobre-ajuste e sub-ajuste. Segundo Bishop (1995), Haykin (1999) e Villanueva (2006), os ganhos de desempenhos com o comitê, comparado com uma única máquina de aprendizado são: melhor capacidade de generalização, diminuição da variância do modelo e maior tolerância a ruídos nos dados.

As máquinas de comitê se dividem em duas categorias: estruturas estáticas (*ensemble*), onde as saídas dos previsores são combinadas através de uma estrutura estática independentes da entrada, por exemplo: média de *ensemble* e reforço; e as estruturas dinâmicas, onde as saídas dos previsores são combinadas de formas distintas conforme os dados de entrada diferem entre si, por exemplo: mistura hierárquica e mistura de especialistas na qual as saídas das redes especialistas são combinadas não linearmente através de módulo específico.

4. ABORDAGEM

A abordagem proposta neste trabalho busca minimizar o erro na previsão de séries temporais. O algoritmo proposto para resolver o problema de previsão combina adequadamente os algoritmos genéticos e as redes neurais MLP e de Kohonen em comitês de redes. Mais especificamente, o problema de previsão foi decomposto em cinco subproblemas: seleção de grupos de exemplos empregando redes de Kohonen, seleção de atributos para treinamento de redes MLPs, projeto e parametrização automática de redes MLP empregando algoritmos genéticos, aproximação da função previsão empregando um comitê das redes resultantes e seleção automática de previsores através da avaliação independente dos previsores da grade.

Este capítulo foi dividido em mais seis seções. A Seção 4.1 apresenta um esboço do algoritmo proposto. A Seção 4.2 fala sobre como foi realizada a seleção de exemplos utilizando redes de Kohonen. A Seção 4.3 explica todo o processo do projeto automático das redes previsoras para cada neurônio da grade, esta seção também possui subseções explicando o motivo de algumas decisões tomadas durante a formalização da abordagem. A Seção 4.4 apresenta os comitês de máquinas criados para aproveitar todo o conhecimento adquirido com a clusterização dos dados e a criação de redes específicas para cada padrão de entrada. A Seção 4.5 propõe a análise independente dos previsores da grade e a Seção 4.6 utiliza este raciocínio para apresentar a seleção automática dos previsores da classe.

4.1. Algoritmo para a Previsão com Séries Temporais

O problema de previsão considera que são conhecidas informações (dados) a respeito do presente e do passado de um processo e que se deseja conhecer informações a respeito do futuro do processo. Estas informações são séries temporais dos valores dos diversos atributos envolvidos no processo de previsão. Primeiramente, na etapa de aprendizagem dos dados, o algoritmo proposto emprega uma rede de Kohonen para agrupar (clusterizar) os dados a respeito do passado em subconjuntos de exemplos que representem adequadamente a densidade das informações do passado. Posteriormente, o algoritmo utiliza os subconjuntos associados aos neurônios na grade da rede para o treinamento de redes MLPs associadas. Durante a etapa de treinamento, o algoritmo utiliza algoritmos genéticos para a seleção de atributos, projeto e parametrização automática das redes MLPs que são avaliadas juntamente com todas as estratégias de previsores disponíveis para o sistema. Finalmente, na etapa de

operação, o algoritmo de seleção automática de previsores se encarrega de definir qual estratégia é responsável por definir a saída para cada padrão de entrada apresentado.

Pode-se dizer, que o algoritmo proposto para esta abordagem se divide em duas etapas, uma de pré-processamento e aprendizagem (algoritmo *offline*, ilustrado na Figura 12) que requer uma fatia maior de tempo e recursos computacionais, onde é realizada a clusterização dos dados de treinamento, configuração das redes MLP associadas aos neurônios da grade de Kohonen e também é realizada a avaliação dos previsores disponíveis para o sistema; e outra fase (*online*, Figura 13) onde a previsão, de fato, é realizada após a seleção automática do melhor previsor para aquele padrão de dados informado como entrada. Os dados disponíveis foram divididos em três partes: 50% para o conjunto de treinamento, 25% para o conjunto de validação e 25% para o conjunto de teste. Devido ao fato do algoritmo de seleção automática de previsores utilizar como insumo os resultados do pré-processamento (realizado pelo primeiro algoritmo), o tempo de resposta deste algoritmo é mínimo.

Algoritmo 1: Pré-Processamento

Data: $dadosTreinamento, dadosValidacao, estrategias$
Result: $kohonen, dadosPrevisores$

```
1  kohonen ⇐ treinarRedeKohonen(dadosTreinamento);
2  foreach kohonen.getNeuronios() do
3  |  dados ⇐ buscarDados(neuronio, dadosTreinamento, kohonen);
4  |  dadosPrevisor ⇐ projetarPrevisor(dados);
5  |  dadosPrevisores.add(dadosPrevisor, neuronio);
6  end
7  foreach dadosValidacao do
8  |  neuronio ⇐ kohonen.getBMU(dado);
9  |  foreach estrategias do
10 |  |  previsao ⇐ estrategia.getPrevisao(dadoValidacao.getEntrada();
11 |  |  erro ⇐ calcularErro(previsao, dadoValidacao.getSaida());
12 |  |  dadosPrevisores.addErro(erro, neuronio, estrategia);
13 |  end
14 end
15 salvarRede(kohonen);
16 salvarPrevisores(dadosPrevisores);
```

Figura 12 - Algoritmo de Pré-processamento.

O algoritmo de pré-processamento é responsável por fazer a clusterização dos dados de entrada através do treinamento de uma rede de Kohonen utilizando os dados de treinamento como conjunto de entrada (linha 1). Após a rede estar treinada, é necessário definir as redes

previsoras (MLP) para cada *cluster* (neurônio da grade de Kohonen). Para cada neurônio são coletados todos os dados associados a ele (dados que, quando apresentados a rede, ativam o neurônio em questão), o método $buscarDados()$ utiliza o SOM treinado para encontrar todos os dados dentro do conjunto $dadosTreinamento$ em que o neurônio em questão é o "melhor neurônio" (linha 2).

Estando de posse do subconjunto de dados que tem o neurônio em questão como BMU, o próximo passo é configurar uma rede previsora (MLP) otimizada para a classe de dados representada pelo neurônio em questão, realizando a seleção de atributos e o projeto automático do previsor utilizando apenas os dados da classe em questão (linha 3). O objeto $dadosPrevisor$ obtido pelo método $projetarPrevisor()$ é uma estrutura que, além da topologia da rede que será utilizada para aquele *cluster*, possui outras informações como: os atributos que serão utilizados como entrada para a previsão, a janela de tempo de cada atributo e a taxa de erro médio calculada para aquela rede após testes com um conjunto de validação. Após todas as redes MLPs previsoras serem treinadas com os seus respectivos subconjunto de dados (classe associada à rede), resta apenas avaliar o desempenho de todas as estratégias de previsão que podem ser utilizadas pelo sistema utilizando os dados do conjunto de validação (linhas 7-14). Estes valores também são armazenados no objeto $dadosPrevisor$ onde, para cada neurônio, cada estratégia do leque de opções possui um erro associado àquela classe de dados. A tarefa de pré-processamento é finalizada com a persistência dos dados gerados em arquivo para que estas informações sejam utilizadas pelo algoritmo de seleção automática de previsor (linhas 15, 16).

Algoritmo 2: Seleção automática de previsor

Data: $kohonen, dadosPrevisores, dadosTeste$
Result: $saida$

```
1 foreach dadosTeste do
2 |   neuronio ⇐ kohonen.getBMU(dadoTeste);
3 |   dadosPrevisor ⇐ dadosPrevisores.obterPrevisor(neuronio);
4 |   estrategia ⇐ dadosPrevisor.getMelhorEstrategia(neuronio);
5 |   saida ⇐ estrategia.getPrevisao(dadoTeste.getEntrada());
6 end
```

Figura 13 - Algoritmo de seleção automática de previsor.

O segundo algoritmo é responsável por selecionar automaticamente a melhor estratégia para realizar a previsão de cada um dos dados do conjunto de teste ($dadosTeste$), até então

desconhecidos para o sistema. Conforme já mencionado, ele necessita do mapa de Kohonen, das redes previsoras associadas a cada neurônio da grade que foram geradas e persistidas pelo algoritmo de pré-processamento e das avaliações de cada uma das estratégias calculadas *a priori*. Para cada instância no conjunto de teste, o algoritmo verifica a classe da instância, ou seja, conforme o neurônio vencedor da rede de Kohonen (BMU) que for ativado com a apresentação da instância (linha 2). Em seguida, o algoritmo seleciona e obtém as informações a respeito da rede previsora previamente treinada e dos desempenhos médios das estratégias associadas ao BMU (linha 3). Baseado nas informações dos previsores, o sistema seleciona o previsor com a menor taxa de erro médio associada àquele *cluster* (linha 4). Esta estratégia selecionada é a responsável por definir a saída para aquele dado apresentado.

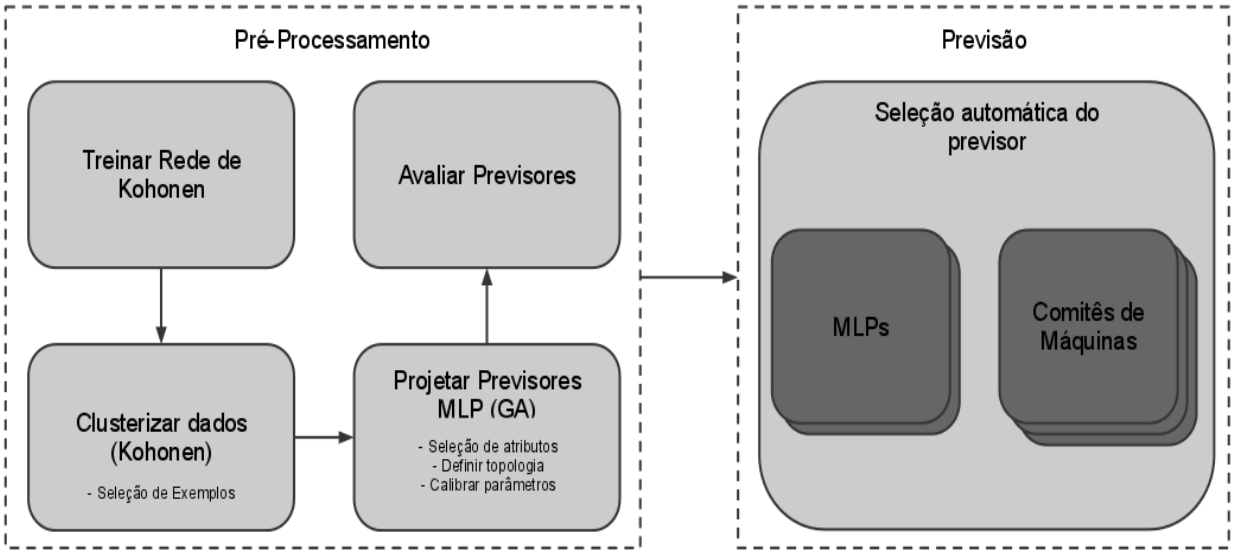


Figura 14 – Fluxograma macro da abordagem proposta.

A Figura 14 exibe um fluxograma macro da abordagem proposta, destacando as duas etapas do algoritmo (pré-processamento e seleção automática do previsor), assim como os principais módulos de cada um. Alguns destes módulos foram detalhados nas próximas seções.

4.2. Seleção de exemplos via Rede de Kohonen

Um ponto a se pensar na criação de uma estrutura de previsão de séries temporais é a qualidade dos exemplos utilizados no treinamento. Os problemas já surgem na hora de quantificar a qualidade de um determinado exemplo: este pode ser bom na aprendizagem de um padrão de entrada e péssimo para outro. Uma possível solução seria realizar o treinamento

apenas com os exemplos bons para o dado de entrada em questão. O desafio é como saber quais os melhores exemplos para uma entrada até então nunca apresentada ao sistema.

Esta abordagem busca resolver esse problema clusterizando o conjunto de exemplos de treinamento, consequentemente, gerando um classificador empregando uma rede de Kohonen. Conforme mencionado, esta etapa visa gerar subconjuntos de exemplos, onde cada subconjunto identifica uma classe de dados detectada pela rede de Kohonen, para o treinamento das redes MLPs previsoras. Mais especificamente, após a rede de Kohonen ser treinada, obtém-se o neurônio vencedor para cada exemplo do conjunto de exemplos e, em seguida, os subconjuntos de exemplos associados a estes neurônios.

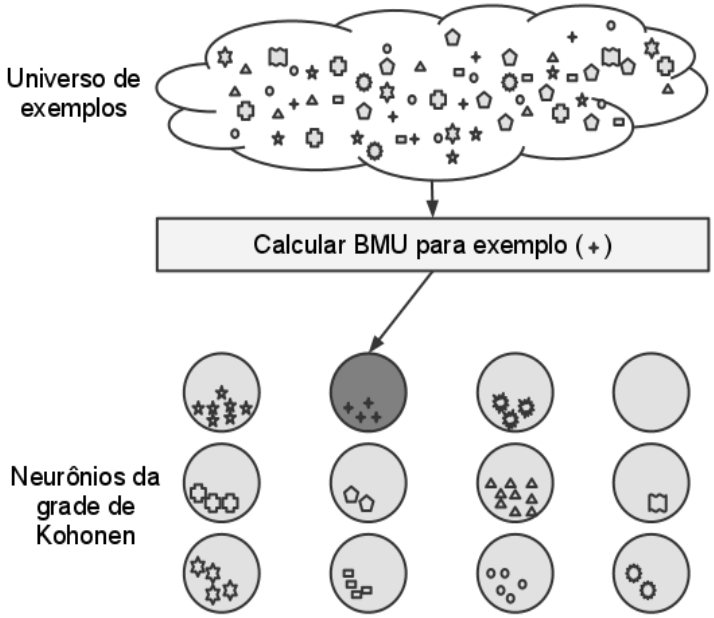


Figura 15 – Seleção de exemplos utilizando SOM.

A Figura 15 exibe como o agrupamento e, consequentemente, como a seleção de exemplos é realizada utilizando a rede de Kohonen. Considerando que a rede foi treinada, cada exemplo do conjunto de exemplos foi apresentado à rede e os neurônios vencedores foram identificados. O dado de entrada foi associado ao neurônio e inserido em um dos subconjuntos associado a uma das onze classes detectadas no conjunto de exemplos. Ao final de todo o processo, cada exemplo estará associado a alguma classe de dados.

4.2.1. Neurônios com poucos exemplos associados

Em séries temporais, principalmente as caóticas, dificilmente os dados são uniformes, ou seja, enquanto alguns subconjuntos possuem vários exemplos associados, existem outros com poucos exemplos, o que pode ocasionar previsões piores nas redes criadas para trabalhar

com os dados destes *cluster*s de menor densidade. Neste trabalho são apresentadas duas estratégias para se trabalhar com estes neurônios de baixa densidade.

A estratégia para aumentar a densidade de exemplos de subconjuntos de baixa densidade, consistiu na inserção de exemplos pertencentes aos subconjuntos associados aos neurônios vizinhos dos neurônios aos quais os subconjuntos de baixa densidade estão associados.

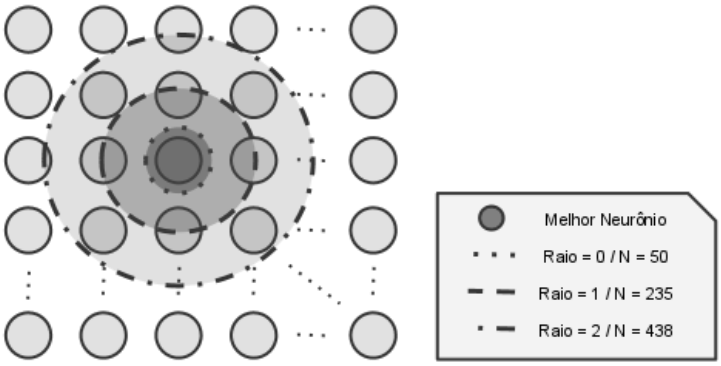


Figura 16 – Relaxamento na restrição de proximidade.

A Figura 16 ilustra este processo de relaxamento na restrição de proximidade (semelhança). A estratégia é aumentar o raio gradativamente até atingir uma quantidade mínima de exemplos selecionados. O número de exemplos mínimo ideal a serem coletados não pode ser definido por uma constante, este deve ser analisado caso a caso. Deve-se buscar um meio termo para não ter problemas por utilizar poucos exemplos no conjunto de treinamento e, ao mesmo tempo, evitar apresentar dados fora do padrão que a rede está sendo treinada, que é o que ocorre quando o raio de aceitação está muito grande. Desta forma, é possível aumentar a quantidade de exemplos sempre buscando selecionar os dados mais semelhantes dentro do universo do conjunto de treinamento.

Uma segunda estratégia para tratar o problema dos neurônios com poucos exemplos associados é a utilização do casamento de densidade com redes de Kohonen (Seção 3.1.3), onde o cálculo da probabilidade *a priori* do dado pertencer àquele neurônio atua como uma espécie de glosador, reduzindo a responsabilidade dos neurônios com poucos dados associados (de baixa densidade). Ao contrário de Siqueira (2006) que simplesmente excluía do modelo as classes menos densas, esta estratégia utiliza todas as classes, mas de forma ponderada, levando em consideração a densidade das mesmas. Essa estratégia é utilizada neste trabalho apenas no comitê de máquinas por diagrama de Voronoi (Seção 4.4.2).

4.3. Projeto das redes previsoras

A rede de Kohonen permite que, para cada entrada, sejam selecionados apenas os exemplos mais similares a ela, pode-se então dizer que cada neurônio representa um *cluster*, uma classe de dados. Estes exemplos são utilizados para o treinamento de uma rede específica para realizar a previsão daquela entrada. Cada classe de dados é independente das demais e pode ser manipulada de formas distintas. Estando de posse dos exemplos de cada classe, é possível buscar a melhor configuração para uma rede de previsão (MLP, *feedforward* com *backpropagation*) específica para trabalhar com as entradas mapeadas para aquele *cluster*. A Figura 17 exibe como o processo é realizado: para cada neurônio, a seleção de atributos e o projeto automático de uma rede MLP são realizados, inclusive janela de tempo, onde apenas os dados daquela classe serão utilizados.

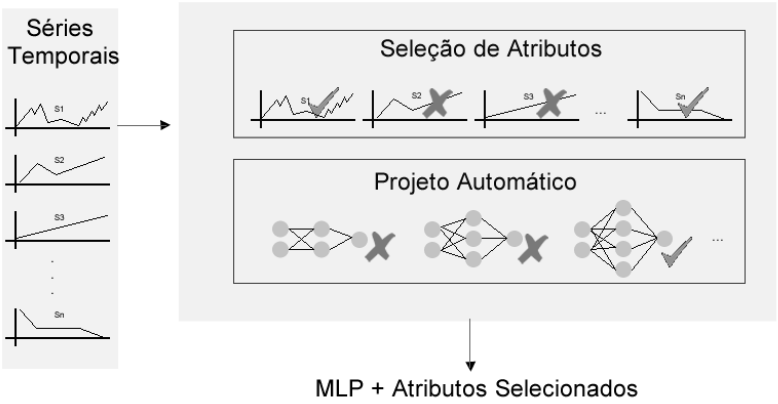


Figura 17 – Processo de treinamento da rede de previsão.

A seleção de atributos e o projeto automático de cada rede neural são realizados simultaneamente por um algoritmo genético onde cada cromossomo define quais atributos (quais séries temporais) são utilizados como entrada e qual a estrutura da rede MLP. Junto com os atributos que podem ser utilizados, o sistema também permite que sejam determinadas janelas de tempo independentes para cada um dos atributos, ou seja, pode-se utilizar como entrada apenas a última observação do atributo Atb_1 (D+0), as três últimas observações do atributo Atb_2 (D+0, D-1 e D-2) e até mesmo nenhuma observação do atributo Atb_3. Na prática, cada unidade da janela de tempo é considerada como um atributo, ou seja, o Atb_2 que possui uma janela de tamanho 3 (três) é dividido em três atributos: Atb_2 em D+0, Atb_2 em D-1 e Atb_2 em D-2.

O tamanho máximo da janela de tempo é parametrizável, sendo que, quanto maior o valor do parâmetro, maior será o tamanho do cromossomo. Deve-se tomar cuidado para não

ficar com um cromossomo muito grande, pois isto aumenta o espaço de soluções, o que acaba dificultando a tarefa do GA na busca de uma solução de boa qualidade.

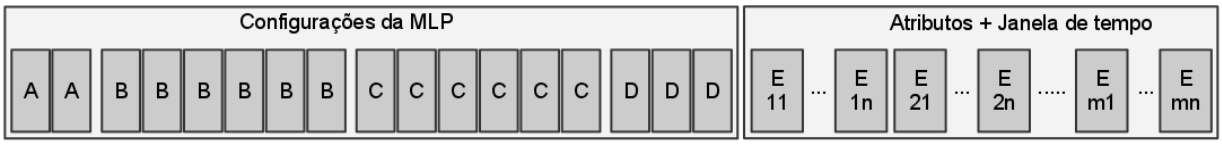


Figura 18 – Estrutura do cromossomo para o projeto da rede previsora.

A Figura 18 ilustra a estrutura do cromossomo para o projeto da rede previsora. Nela pode-se perceber que o cromossomo se divide em dois blocos principais que são responsáveis respectivamente pelas configurações da rede MLP e os atributos (com janela de tempo) que são enviados como entrada para a rede MLP definida no bloco anterior. O cromossomo é um vetor de valores binários, cujo tamanho é variável e cada posição do mesmo (considerando a Figura 18) representa:

- **A:** as duas posições iniciais do cromossomo definem a quantidade de camadas ocultas que a rede terá. Este valor é calculado pela soma dos valores nas posições "A", ou seja: $A_1 + A_2$. Desta forma, a quantidade de camadas ocultas irá variar entre 0 e 2 camadas;
- **B:** as seis posições seguintes definem a quantidade de neurônios nas camadas intermediárias. Este valor é calculado ao considerar estas seis posições como uma sequência de números na base binária e o valor final será acrescido em 1. Por exemplo: a sequência "100101" significa que a rede terá 37+1=38 neurônios nas camadas intermediárias, ou seja: $(B_1B_2B_3B_4B_5B_6)_2 + 1_{10}$. Desta forma, a quantidade de neurônios nas camadas intermediárias da MLP irá variar entre 1 e 64 neurônios;
- **C:** as seis posições seguintes definem a taxa de aprendizado da rede. Assim como a série anterior, esta sequência também é considerada como base binária, a diferença é que, ao invés de adicionar 1, o resultado será dividido por 100. Usando novamente a sequência "100101" como exemplo, o resultado seria 37/100=0,37. O cálculo se dá pela equação: $(B_1B_2B_3B_4B_5B_6)_2/100$. Desta forma, a taxa de aprendizado da rede irá variar entre 0 e 0,63;
- **D:** as três posições seguintes, últimas no que diz respeito às configurações da rede definem a quantidade de épocas pelo qual o treinamento irá passar. O

cálculo se dá pela função exponencial: $10^{(D_1+D_2+D_3)}$. Desta forma, a quantidade de épocas no treinamento pode ser 1, 10, 100 ou 1000.

- **E:** esta sequência é responsável por definir os atributos que serão utilizados como entrada para a rede. O tamanho desta sequência é dinâmico e calculado por $M * N$, onde M representa a quantidade de atributos disponíveis e N representa o tamanho máximo da janela de tempo para cada atributo. Ou seja, para cada atributo, existirá uma sequência de N posições que definirá o tamanho da janela de tempo para aquele atributo em específico. O cálculo se dá pela equação: $E_{i1} + E_{i2} + \cdots + E_{in}$**.** Desta forma, o tamanho da janela de tempo de cada atributo é distinto e irá variar entre 0 e N. Vale ressaltar que atributos com tamanho de janela igual a zero significa que aquele atributo não será utilizado como entrada para aquela rede.

Após definir a estrutura de previsão de cada classe, tem-se um mecanismo de previsão, onde é possível mapear, através da rede de Kohonen, qual a rede MLP (treinada apenas com os exemplos daquela classe) e atributos de uma determinada entrada que devem ser utilizados para minimizar o erro na previsão. A Figura 19 ilustra como cada neurônio da rede de Kohonen está associado a uma rede MLP e a um subconjunto de atributos para realizar a previsão de entradas pertencentes àquela classe.

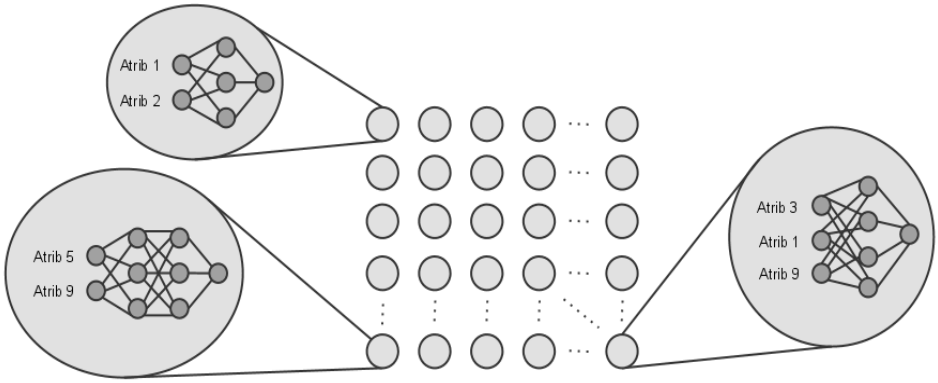


Figura 19 – MLPs distintas associadas aos neurônios na grade.

4.3.1. Projeto automático e seleção de atributos

Para compor a estrutura da rede previsora é necessário definir a topologia de rede (quantidade de camadas, quantidade de neurônios em cada camada, taxa de aprendizado e quantidade de épocas) e os atributos que serão utilizados como entrada para a mesma (considerando o tamanho da janela de tempo de cada atributo como um atributo, ou seja, uma entrada para a rede). Alguns questionamentos ocorrem para realizar esta tarefa, tais como:

- Que método para realizar a seleção de atributos será utilizado?
- Qual metodologia será empregada para otimizar a calibragem da rede neural?

Ambos são questionamentos antigos e que já obtiveram bastante atenção dos pesquisadores. Na literatura, existem diversos métodos específicos para realizar a seleção de atributos, assim como também há um leque de algoritmos para configurar redes neurais automaticamente. Algumas delas são mencionadas nas seções 3.3 e 3.4, respectivamente. Cada algoritmo resolve o seu problema de forma isolada, porém, ao se pensar em realizar as duas tarefas em um mesmo sistema, surge outro questionamento:

- O que deve ser realizado primeiro, a seleção de atributos ou o projeto automático?

Para realizar a seleção de atributos usando algoritmos do tipo *wrapper* (que utiliza o previsor para medir a sua eficiência) é necessário que a rede previsora já esteja definida, ou seja, que o projeto automático já tenha sido realizado anteriormente. Por outro lado, avaliar topologias de redes no projeto automático necessita que os atributos da camada de entrada estejam definidos para medir o desempenho da mesma.

Alguns testes foram realizados, iniciando pela seleção de atributos (neste caso, a topologia da rede foi definida de forma manual). Terminada a seleção, realizou-se o projeto automático empregando somente os atributos recém selecionados. O resultado observado foi que a topologia considerada ótima, na grande maioria das execuções, foi exatamente a topologia definida manualmente pois a seleção de atributos já tinha definido o melhor subconjunto de atributos para aquela topologia, logo, dificilmente seria encontrada outra topologia que melhorasse o desempenho do sistema. Os mesmos testes foram realizados iniciando pelo projeto automático e os resultados foram similares.

A abordagem proposta neste trabalho realiza a seleção de atributos e o projeto automático da rede ao mesmo tempo. Um algoritmo genético foi utilizado para esta tarefa onde na estrutura do seu cromossomo ilustrado na Figura 18 mapeia tanto a camada de entrada (os atributos com janela de tempo) quanto alguns parâmetros da rede e a topologia das camadas intermediárias (projeto automático da rede). A função de *fitness* desse algoritmo considera o erro médio da rede codificada no cromossomo.

4.3.2. Clusterização antecedendo projeto de previsores

Outro ponto da abordagem que reflete diretamente nos resultados do sistema é a ordem em que são realizados o projeto das redes previsoras e a clusterização para seleção de exemplos. Para cada uma das situações foi elaborada uma estratégia, ou seja:

1. **Projeto antes da clusterização:** o projeto automático das MLPs é executado (conforme solução proposta na seção 4.3.1) levando-se em consideração o conjunto de treinamento total e, em seguida, é realizada a clusterização dos dados em subconjuntos de exemplos, que serão utilizados no treinamento das redes MLPs previsoras. A Figura 20 ilustra esta situação, ou seja, em que todas as redes MLPs possuem a mesma topologia e os mesmos atributos de entrada, resultantes da etapa de projeto automático realizada previamente.
2. **Projeto após a clusterização:** primeiro os exemplos do conjunto de treinamento são agrupados por similaridade (conforme ilustrado na Figura 15) em subconjuntos de treinamento e, em seguida, estes subconjuntos são utilizados nas seleções dos atributos e nos projetos automáticos das redes MLPs previsoras. A Figura 19 ilustra esta situação, ou seja, cada rede previsora possui atributos de entrada e topologia específica, resultantes do treinamento realizado com o seu respectivo subconjunto de exemplos.

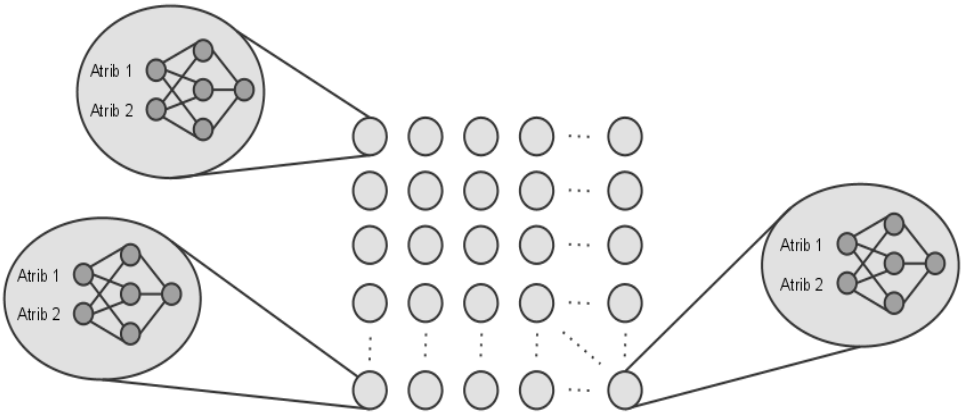


Figura 20 - MLPs com topologias idênticas associadas aos neurônios na grade.

Desta forma, a estratégia 1 projeta uma rede otimizada para trabalhar com todos os dados apresentados, ou seja, uma solução global. Ao final da estratégia, tem-se várias redes MLP (utilizando os mesmos atributos como entrada e a mesma topologia da rede) que se diferenciam em termos dos pesos obtidos ao final dos processos de treinamentos envolvendo os diversos subconjuntos de exemplos obtidos com a clusterização dos exemplos disponíveis.

Por outro lado, a estratégia 2 gera várias redes distintas, cada uma com topologia e atributos específicos obtidos a partir do treinamento com os mesmos subconjuntos de exemplos providenciados na estratégia 1, garantindo assim um conjunto de redes com soluções especificas e locais.

Observou-se que é melhor realizar o projeto a partir da clusterização dos dados:

- **A clusterização é independente:** a clusterização ocorre nas duas estratégias, entretanto o desempenho dos previsores é altamente influenciado pelos exemplos utilizados no seu treinamento;
- **O projeto dos previsores abrange espaço de soluções bem maior:** a solução do projeto dos previsores busca encontrar os pesos sinápticos e a topologia ótimos associados a cada subconjunto de exemplos;
- **Os previsores ficam mais especializados:** cada uma das redes previsoras não sofre influência de dados que não pertencem à classe que ela representa, sendo responsável por gerar uma solução que leva em consideração o aprendizado daquela classe.

O tempo de processamento necessário para executar a segunda estratégia é bem maior que o tempo de execução da primeira, pois o projeto do previsor é realizado apenas uma vez na primeira estratégia e N vezes, na segunda, onde N representa a quantidade de neurônios existentes na grade de Kohonen. Não se pode afirmar que o tempo da segunda é N vezes maior que o da primeira, pois enquanto o projeto da primeira é realizado com todos os exemplos disponíveis, os projetos da segunda são realizados com subconjuntos dos exemplos disponíveis, o que reduz o tempo de processamento.

4.3.3. Utilização de cache no algoritmo genético

Na Seção 4.3 foi definido que as redes previsoras são projetadas automaticamente por algoritmos genéticos, um GA específico para cada rede. O desempenho desta proposta depende, fortemente, da eficiência do algoritmo genético que depende do cálculo de uma função *fitness* que leva em consideração o treinamento *backpropagation* de uma rede MLP. Assim, o tempo para avaliar cada cromossomo em uma população pode variar entre segundos e horas, dependendo da estrutura da rede neural, da quantidade de atributos utilizados na camada de entrada e da quantidade de exemplos em um subconjunto de treinamento.

Uma das características dos algoritmos genéticos é o elitismo, ou seja, manter os indivíduos mais aptos ao longo das gerações. Em algumas implementações também não há restrições em relação à repetição de indivíduos em mais de uma geração. Nestes casos, deve-se evitar a repetição do cálculo da *fitness* desses indivíduos. Neste trabalho, de maneira a reduzir o tempo de processamento total do algoritmo, aplicou-se *cache* ao GA, mantendo-se uma estrutura que associa os cromossomos aos seus valores de *fitness*. Assim, antes de avaliar, o algoritmo checa se o indivíduo pertence à estrutura, evitando-se assim o recálculo de sua *fitness*. A adaptação do algoritmo genético pode ser visualizada na Figura 21.

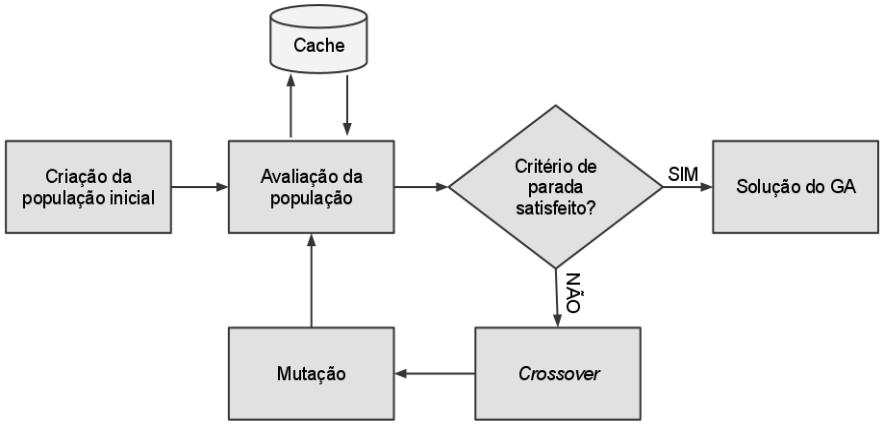


Figura 21 - Adaptação do GA com inclusão de cache.

O cálculo da *fitness* para o GA com cache (f_{CACHE}) do cromossomo é realizado através da equação definida em (26):

$$f_{CACHE} = \frac{\sum_{i=1}^{N} f_i}{N} \tag{26}$$

dado que $N \leq M$, onde f_i é o valor da função *fitness* comum, calculada pela i-ésima vez; N é a quantidade de cálculos de *fitness* já realizados para aquele cromossomo e M representa um parâmetro que indica a quantidade máxima que o mesmo cromossomo deve ser calculado, ou seja, o valor máximo de N.

Como a função *fitness* não é determinística, o parâmetro M é uma proteção contra resultados fora da curva normal. Se o M for alto, a probabilidade de ter um f confiável serão maiores, porém este parâmetro influi bastante no tempo de execução do GA, quanto maior o M, mais vezes um mesmo cromossomo será calculado e maior será o tempo gasto.

Os ganhos com esta abordagem em termos do tempo de execução (em milissegundos) por geração em um algoritmo genético utilizado para seleção de atributos podem ser visualizados na Figura 22.

Pode-se notar que, nas gerações iniciais, os tempos de execução por geração são similares, porém, à medida que a lista de cromossomos calculados (já em cache) vai aumentando, o tempo de execução do GA com cache cai drasticamente, enquanto que o GA comum mantem-se relativamente estável durante todas as gerações.

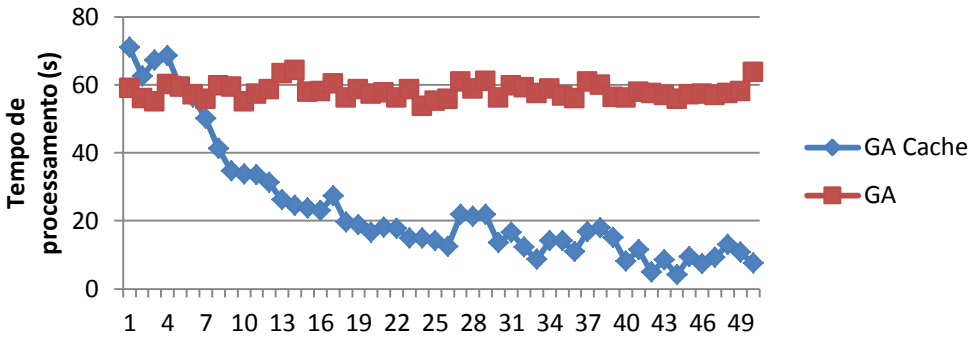


Figura 22 - Tempo de processamento por geração dos GAs.

4.4. Comitê de Máquinas

O algoritmo proposto nesta dissertação também emprega a ideia de comitês de máquinas, ambos de estrutura dinâmicas com mistura de especialistas, onde todo o conceito de classe de dados e proximidade (semelhança) é aproveitado. Duas formulações de comitês foram propostas e testadas. Cada formulação está associada a um dos dois modos empregados na tentativa de aumentar a densidade de exemplos nos subconjuntos obtidos com o processo de clusterização do conjunto de exemplos. A primeira formulação, conforme destacado na Subseção 4.4.1, utiliza a informação respeito da distância entre o padrão de entrada apresentado e os pesos dos neurônios na grade da rede de Kohonen. A segunda, em destaque associada ao modo descrito na Subseção 4.4.2 utiliza informações a respeito dos diagramas de Voronoi dos neurônios na grade, da probabilidade do padrão de entrada pertencer às classes de dados associadas a estes neurônios e das densidades de dados nestas classes.

4.4.1. Primeira formulação: influência da distância

A Figura 23 ilustra o funcionamento deste comitê. Um dado de entrada é submetido às as redes previsoras e a um módulo que calcula a influência da distância entre cada neurônio e o neurônio que representa a classe do dado de entrada apresentado. Este valor é utilizado

como um peso nas conexões entre a saída da rede e o módulo que faz a combinação das saídas das redes previsoras.

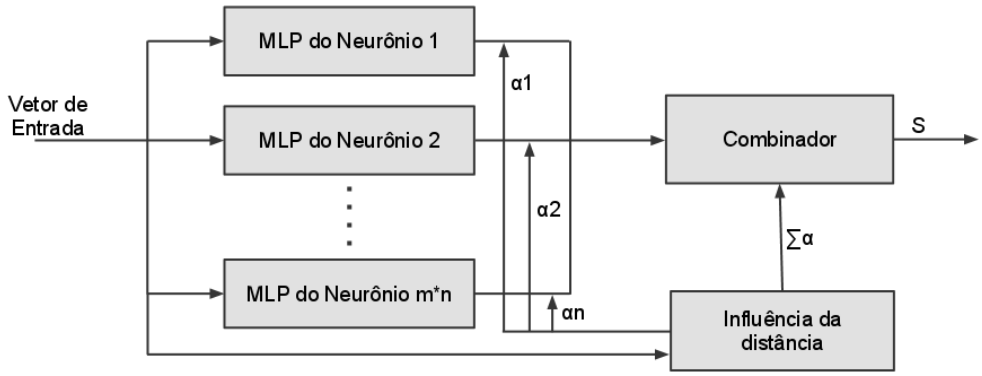


Figura 23 - Comitê de máquinas com influência da distância.

O valor previsto é calculado através de uma combinação de todas as saídas das redes previsoras associadas aos neurônios da grade de Kohonen. O valor da saída de uma rede previsora é ponderado ($\propto$) pelo valor da distância de seu neurônio em questão ao neurônio BMU (neurônio que foi ativado pela entrada), conforme definido em (27):

$$S = \frac{\sum_{i=1}^{N} \alpha_i S_i}{\sum_{i=1}^{N} \alpha_i} \tag{27}$$

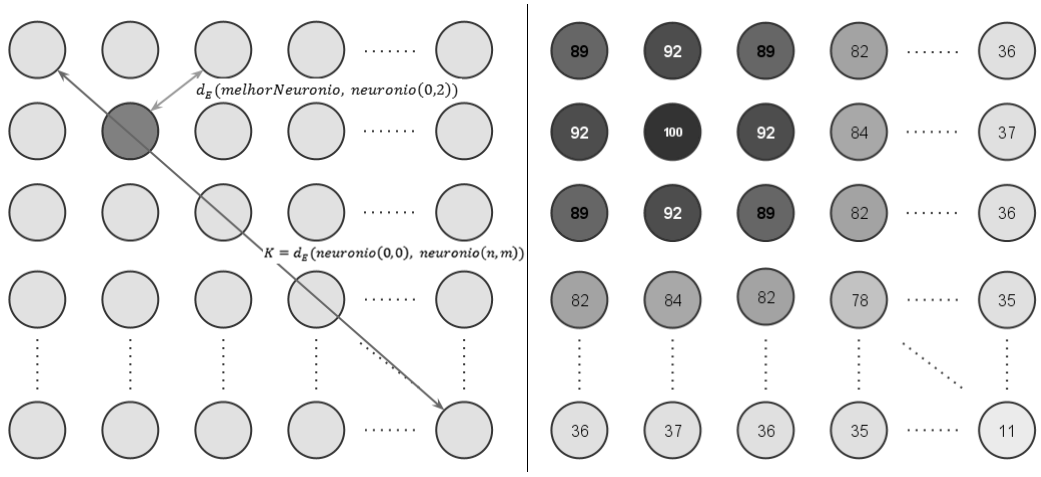
onde S_i representa a saída do neurônio i e α_i é calculado conforme indicado em (28):

$$\alpha_i = \frac{d_E(\text{melhorNeuronio},\ \text{neuronio}_i)}{K} \tag{28}$$

onde K representa a distância euclidiana (d_E) entre os pontos de uma das diagonais da rede de Kohonen, conforme ilustrado na Figura 24(a), e calculado conforme (29):

$$K = d_E\big(neuronio(0,0), neuronio(n,m)\big) \tag{29}$$

onde n representa a quantidade de linhas da grade de Kohonen e m representa a quantidade de colunas da mesma, e K permite obter valores $\propto_i$ normalizados. A Figura 24(b) ilustra a participação percentual (x100) de cada neurônio (α_i) no cálculo da saída final do sistema, para o exemplo ilustrado na Figura 31(a), ou seja, quando o neurônio $(1,1)$ foi ativado (BMU).



(a) **(b)**

Figura 24 - Cálculo da distância para o melhor neurônio e cálculo do K **(a).** Participações percentuais das saídas de cada neurônio da grade na composição da saída resultante. **(b)**

Na Figura 24(b) é possível notar que os neurônios mais próximos do neurônio escolhido, ou seja, os mais similares ao padrão da entrada possuem uma participação maior na saída resultante do comitê. Desta forma, a tarefa de previsão foi dividida entre vários previsores com o diferencial que as saídas das redes mais aptas a realizar a previsão terão um peso maior no resultado final. A junção destes valores aumenta a acurácia das previsões.

4.4.2. Segunda formulação : diagrama de Voronoi

Outra estratégia para comitês de máquinas que foi pensada para realizar a previsão utilizando o conceito de proximidade é a utilização de diagramas de Voronoi. Os diagramas de Voronoi têm a capacidade de criar seções (células) onde todos os elementos contidos nela têm em comum o centro mais próximo. O funcionamento deste comitê utiliza os conceitos do casamento de densidade com redes Kohonen apresentados na Subseção 3.1.3.

O cálculo da saída deste comitê leva em consideração tanto a distância que cada neurônio (classe de dados) está da entrada em questão quanto à densidade de exemplos associado a cada neurônio na grade, conforme (30):

$$Y = \frac{\sum_{j=1}^{N} P(j)P(x'|j)y_j}{\sum_{j=1}^{N} P(j)P(x'|j)} \tag{30}$$

onde $P(j)$, definida neste trabalho conforme (29), determina a probabilidade *a priori* de x', o padrão de entrada, pertencer à célula do diagrama de Voronoi do neurônio j:

$$P(j) = \frac{N_j}{N} \tag{31}$$

onde N_j representa a quantidade de exemplos associada ao neurônio j, e N representa a quantidade total de entradas utilizadas no treinamento. $P(x'|j)$ é a função densidade de

probabilidade condicional da entrada x' no neurônio j e foi formalizada através da função gaussiana em (32):

$$P(x'|j) = e^{-\frac{(x\prime - w_j)}{2*\sigma_j^2}} \quad (32)$$

onde σ_j, formalizado em (33), representa o campo de ativação do neurônio, que é calculado através da distância euclidiana entre o peso do neurônio j e as entradas mapeadas para aquele neurônio, ou seja, os padrões pertencentes à classe representada pelo neurônio j:

$$\sigma_j = \frac{\sum_{i=1}^{N_j} d_E(W_j, x'_i)}{N_j} \quad (33)$$

Com esta abordagem, o ponto médio dos dados de cada classe será o centro de uma célula do diagrama de Voronoi. A saída de cada um dos previsores é ponderada pela probabilidade daquela rede representar o padrão de entrada apresentado, ou seja, classes muito distantes terão pesos menores (similar ao comitê na primeira formulação), e pela densidade de cada classe, ou seja, as classes com poucos dados associados terão peso menor, trazendo mais confiança às redes treinadas com mais dados pertencentes àquela classe.

4.5. Análise independente dos previsores da grade

Cada rede previsora pode ser avaliada com exemplos de teste da mesma classe cujos exemplos foram utilizados no treinamento. O algoritmo proposto considera o RMSE (34) de cada rede como mais uma informação empregada na tomada de decisão sobre as diversas possibilidades que a previsão pode ocorrer:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(O_i - D_i)^2}{n}} \quad (34)$$

onde O_i representa a saída calculada, D_i é a saída desejada e n a quantidade de exemplos.

O RMSE das redes indica como a previsão deve ser realizada para qualquer valor de entrada. Por exemplo, caso alguma entrada esteja associada a uma rede com baixo RMSE, pode ser mais seguro utilizar somente esta rede específica da classe para realizar a previsão. Entretanto, caso a entrada esteja associada a uma rede com alto RMSE, o algoritmo pode simplesmente se abster de realizar uma previsão para aquela entrada.

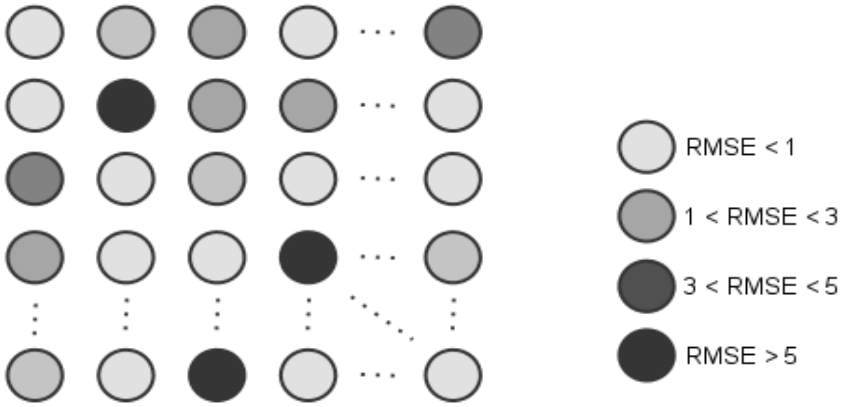


Figura 25 - Erros nas redes associadas aos neurônios da grade.

A Figura 25 exibe um exemplo de uma rede de Kohonen e os respectivos erros das redes previsoras associadas aos neurônios na grade. Cores mais claras representam classes cujas redes possuem baixa taxa de erro. Se o limite de erro aceitável for 5 (cinco), por exemplo, entradas pertencentes às classes da cor mais escura deverão ser tratadas de maneira adequada, já que o RMSE nestes casos estão indicando pouca segurança. Entretanto, é possível melhorar a rede de previsão das classes com taxas de erros altas ou, até mesmo, buscar alguma outra estratégia que melhore as previsões dos dados daquela classe sem interferir ou impactar nas redes de classes com baixa taxa de erro.

A análise independente dos previsores da grade pode continuar atualizando as taxas de erro médio dos *clusters* mesmo após a fase de treinamento. Esta característica permite um aprendizado *online*, onde o sistema continua avaliando seu desempenho podendo mudar seu comportamento caso o cenário tenha mudado.

4.6. Seleção automática dos previsores da classe

Se o sistema tiver a sua disposição vários previsores (não importando se estes forem de estratégias distintas) é possível definir qual o melhor previsor para aquela classe através da análise independente dos previsores da grade. Para que isto seja possível, é necessário ter um módulo responsável por realizar a seleção automática do melhor previsor para cada classe de dados. Este previsor deve manter uma estrutura para armazenar o desempenho (erro médio) de cada previsor disponível nas previsões dos dados daquela classe de dados. A estrutura seria semelhante à apresentada na Figura 25, diferindo apenas no acréscimo de uma nova dimensão para armazenar as mesmas informações para as diversas estratégias utilizadas, conforme ilustrada na Figura 26.

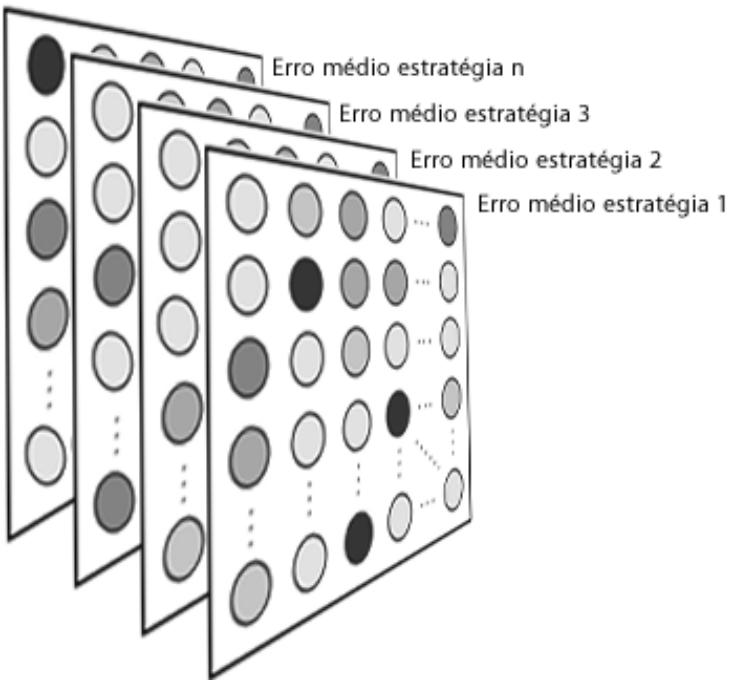


Figura 26 - Erros dos previsores associados aos neurônios da grade por estratégia de previsão.

Desta forma, é criada uma grade tridimensional do desempenho médio de cada previsor em cada um dos neurônios da grade. A Figura 27 ilustra o processo de seleção automática da melhor estratégia.

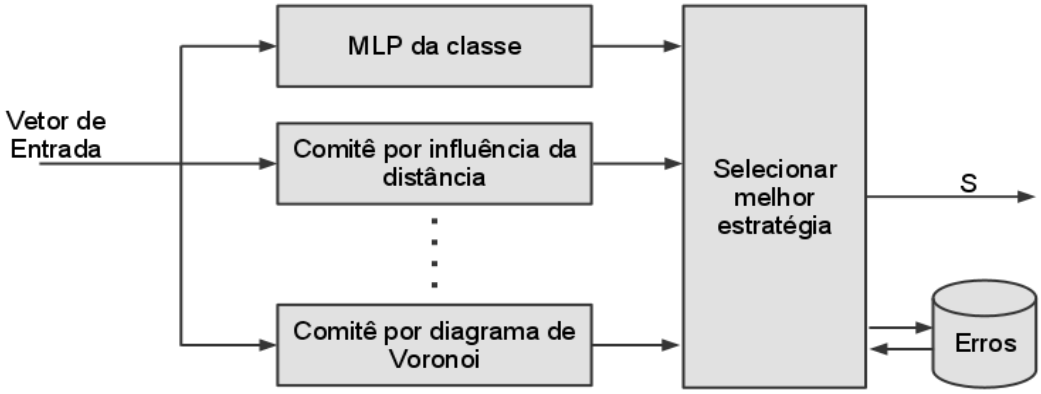


Figura 27 - Seleção automática da melhor estratégia.

Para cada padrão de entrada a ser previsto, os valores de entrada são submetidos a todas as estratégias disponíveis e cada uma irá realizar o cálculo da saída independentemente dos demais. O modulo responsável seleciona a estratégia que possui o menor erro associado às previsões dos dados daquela classe de dados. A saída desta estratégia será a saída resultante da previsão.

Após cada previsão, os desempenhos das estratégias são armazenados na estrutura do módulo de seleção para possibilitar que os mesmos continuem sendo avaliados durante todas

as previsões garantindo assim que a melhor estratégia de cada classe sempre seja selecionada para determinar as saídas daquela classe.

Apesar do algoritmo apresentado na Figura 12 avaliar qual a melhor estratégia antes e utilizar apenas aquela para determinar a saída do sistema, é aconselhável que todas as estratégias sejam avaliadas para, assim como na análise independentes de previsores, manter uma caraterística *online* na execução do algoritmo.

5. AVALIAÇÃO DA ABORDAGEM PROPOSTA

Esta seção busca avaliar a abordagem proposta no Capítulo 4. Esta avaliação foi realizada considerando os diversos aspectos envolvidos na descrição do algoritmo de previsão, proposto na Seção 4.1, como, por exemplo, a seleção de atributos e de exemplos, bem como o projeto e parametrização automática das redes MLPs, a ordem em que estes estas atividades devem ocorrer e a integração destas até a geração do algoritmo de previsão. Assim, além da seção que descreve os dados que foram utilizados na avaliação do algoritmo, as outras seções foram ordenadas conforme a ordem adotada para descrever o sistema previsor no Capítulo 4, buscando identificar as melhorias de desempenho do sistema previsor à medida que os módulos ilustrados na Figura 14 são inseridos no sistema até a composição do algoritmo como um todo. Esta sequência tem o objetivo de provar a modularidade da abordagem, onde cada módulo adicionado ao sistema ajuda, de alguma forma, no bom desempenho obtido no resultado final.

5.1. Materiais e Métodos

Para avaliar o algoritmo foi desenvolvida uma ferramenta de previsão em linguagem Java. Foram utilizadas séries temporais econômicas. Para validar a eficácia, foram executados 30 (trinta) testes independentes utilizando 9 (nove) séries temporais da empresa Vale (VALE5) no período compreendido entre os anos 2000 e 2008. O comportamento do preço de fechamento da VALE5 no período avaliado pode ser visualizado na Figura 28.

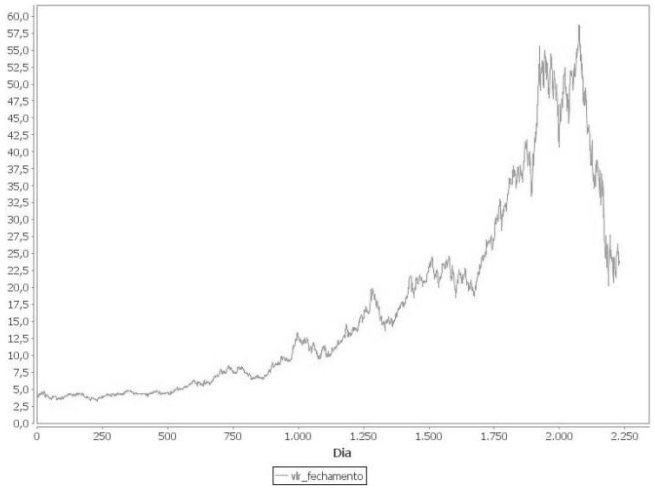


Figura 28 - Evolução do preço de fechamento da VALE5.

Também é possível flexibilizar o tamanho da janela de cada atributo entre 1 (D+0) e 3 (D+0, D-1, D-2) dias, onde cada dia é considerado um atributo distinto, totalizando 27 (3*9=27) atributos a disposição do sistema. As séries temporais utilizadas foram:

1) **Preço de abertura:** representa o preço inicial do ativo no início das negociações de um determinado dia;
2) **Preço de fechamento:** representa o preço final do ativo no final das negociações;
3) **Máxima do dia:** representa o maior preço atingido pelo ativo;
4) **Mínima do dia:** representa o menor valor observado para o ativo;
5) **Volume de negociações:** representa o volume de negócios realizados com o ativo;
6) **Variação percentual em relação ao dia anterior:** representa a evolução ou involução percentual em relação ao dia anterior;
7) **Indicador de alta ou baixa:** representa, de forma binária, se houve uma alta ou baixa em relação ao dia anterior;
8) **Índice de Força Relativa** (**IFR de 14 dias):** possibilita observar a força de uma tendência, se o mercado está comprado ou vendido;
9) **Diferença entre médias de 9 e 40 dias:** estratégia utilizada nas análises técnicas para definir se o ativo está em tendência de alta ou baixa no curto prazo. Quando a média menor está acima da média maior representa uma tendência de alta, ou seja, se a diferença entre as médias for positiva, o ativo está em tendência de alta. A quantidade de dias utilizados nas médias móveis indica se a observação é para curto, médio ou longo prazo.

Além das séries da Vale, também foram realizados testes utilizando outras séries temporais, como os preços da petrolífera Petrobrás (PETR4) e da siderúrgica Usiminas (USIM5). As mesmas nove séries temporais de cada empresa e o mesmo período foram considerados nos testes. Todos os preços e volumes foram coletados no site da BOVESPA. As demais informações: variação, indicador de alta ou baixa, IFR e diferença de média móvel foram calculadas após a coleta dos dados, considerando-se, respectivamente as seguintes equações:

$$V = \frac{Preço_D - Preço_{D-1}}{Preço_{D-1}} \tag{35}$$

$$Ind_{a\_b} = \begin{cases} 1, se\ Preço_D > Preço_{D-1} \\ 0, se\ Preço_D < Preço_{D-1} \end{cases} \tag{36}$$

$$\mathrm{IFR}_{14} = \frac{100}{(1 + \left(\frac{P}{N}\right))} \tag{37}$$

$$\mathrm{Diff}_{9\_40} = \frac{\sum_{i=0}^{8} Preço_{D-i}}{9} - \frac{\sum_{j=0}^{39} Preço_{D-j}}{40} \tag{38}$$

onde $Preço_{D-1}$ representa o preço de fechamento do ativo no dia $D-1$, P representa a média de todas as variações positivas ocorridas no período observado e N representa a média das variações negativas.

Os conjuntos de treinamento, validação e teste foram formados através de sorteios sem reposição dentro do universo de dados e os mesmos conjuntos sorteados foram utilizados em todos os testes. A principal medida de avaliação considerada foi o *Root Mean Square Error* (RMSE) do conjunto de testes, como medidas alternativas, também foram calculadas o MAPE (*Mean Absolute Percentage Error*) e o MAE (*Mean Absolute Error*) das previsões.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{O_i - D_i}{D_i}\right| \tag{39}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|O_i - D_i| \tag{40}$$

onde O_i representa o valor obtido na previsão i, D_i representa o valor desejado pela previsão i; e n representa a quantidade de dados previstos.

Todas as medidas representam uma diferença entre o valor previsto e o valor desejável, ou seja, quanto menor a taxa de erro, mais preciso é o modelo de previsão. O objetivo da previsão é descobrir o dia seguinte (D+1) do preço de fechamento do ativo. Conforme indicado na introdução deste capítulo, os cenários de testes foram alocados à sequência de seções visando avaliar e discutir cada um dos módulos integrados ao sistema. Para cada cenário, foram realizadas 30 execuções onde as três medidas de avaliação supracitadas foram calculadas assim como o desvio padrão de cada uma delas. Como parâmetro de desempenho, a ser utilizado como controle, foi empregado um cenário de testes simples, porém amplamente utilizado na literatura:

A. Controle: denominação empregada por ser base para comparação de todas as outras estratégias, consiste em uma arquitetura simples que utiliza apenas uma MLP, refinada com seleção de atributos e projeto automático considerando todo o conjunto de treinamento. Apesar de ser o teste mais simples, reflete a grande

maioria das utilizações de redes neurais para previsão de séries temporais na literatura. Os resultados alcançados neste cenário podem ser visualizados na Tabela 1.

Tabela 1 - Resultados dos testes com o cenário A.

Cenário	RMSE	σ_{RMSE}	MAPE	σ_{MAPE}	MAE	σ_{MAE}
A	1,886	0,284	0,069	0,004	0,921	0,065

As próximas seções propõem mais 9 (nove) novos cenários visando comparar os resultados obtidos pela integração parcial dos módulos até o algoritmo de previsão, com o cenário de controle (A) e os cenários anteriores avaliados na sequência, até ser possível gerar uma conclusão com a análise dos resultados. A seguir, as denominações dos oito cenários:

- **(b)** projeto e seleção de atributos antes da clusterização;
- **(c)** clusterização antes do projeto e seleção de atributos;
- **(d)** comitê por influência da distância;
- **(e)** comitê por diagrama de Voronoi;
- **(f)** comitê de média (*Ensemble-Averaging*);
- **(g)** projeto e seleção de atributos antes da clusterização com erro baixo;
- **(h)** clusterização antes do projeto e seleção de atributos com erro baixo;
- **(i)** clusterização antes do projeto e seleção de atributos com erro baixo e comitê por influência da distância;
- **(j)** seleção automática de previsores.

5.2. Clusterização antes do projeto de previsores

Conforme já discutido na Seção 4.3.2, a ordem em que a clusterização é realizada afeta no desempenho dos previsores. Para avaliar qual a melhor forma, os seguintes cenários foram levantados:

B. **Projeto + Atributos antes da clusterização:** utilizando a mesma estrutura da rede e atributos definidos no cenário A, várias redes foram treinadas após a seleção de exemplos pelo processo pela clusterização dos exemplos no conjunto de treinamento realizado pela rede de Kohonen (Seção 4.2). Desta forma, a cada neurônio da grade foi associada uma rede de mesma configuração mas com os pesos "otimizados" para a classe de dados do neurônio (conforme ilustrado na Figura 20). A saída resultante é definida pela rede MLP associada à classe do dado de entrada, calculada através da Equação (5).

C. **Clusterização antes do Projeto + Atributos:** aplica a clusterização (Seção 4.2) e, em seguida, realiza o projeto automático e a seleção de atributos para cada *cluster* utilizando apenas os exemplos daquela classe (Seção 4.3). A diferença deste para o teste B é que cada classe de dados tem uma rede com configuração distinta e otimizada para um determinado subconjunto de exemplos (conforme arquitetura ilustrada na Figura 19). Inclusive os atributos de entrada (e sua janela de tempo) podem variar entre as classes (neurônios da grade). A saída resultante é definida pela rede MLP associada à classe do dado de entrada, calculada através da Equação (5).

Os resultados obtidos com os cenários B e C podem ser observados na Tabela 2 e no gráfico ilustrado na Figura 29.

Tabela 2 - Resultados dos testes com os cenários B e C.

Cenário	RMSE	σ_{RMSE}	MAPE	σ_{MAPE}	MAE	σ_{MAE}
A	1,886	0,284	0,069	0,004	0,921	0,065
B	3,160	0,501	0,074	0,009	1,106	0,074
C	1,462	0,471	0,045	0,006	0,719	0,078

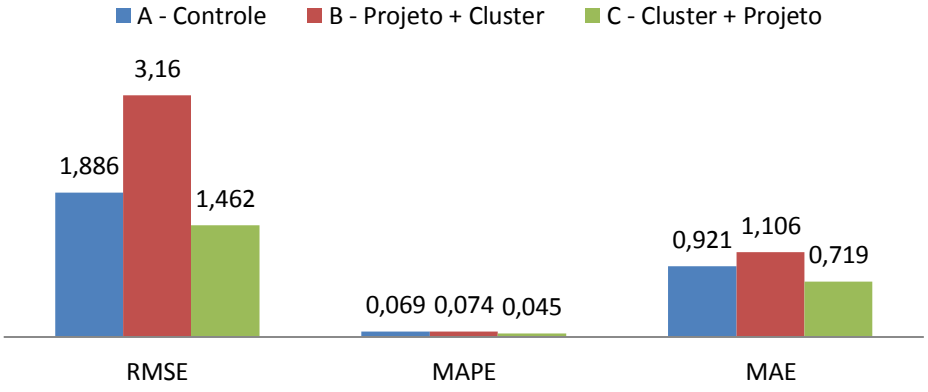


Figura 29 - Gráfico dos resultados dos testes com os cenários B e C.

Os resultados comprovam o que foi discutido na Subseção 4.3.2 mostrando que o cenário C possui uma taxa de erro médio menor que o cenário B em todas as medidas calculadas. Desta forma, pode-se concluir que realizar a seleção de atributos e o projeto automático dos previsores de cada classe de dados detectadas pela rede de Kohonen é melhor que utilizar uma estrutura de rede previsora única. O cenário B possui desempenho pior que o cenário C devido ao fato de o primeiro utilizar, em todos os *clusters*, redes de mesma topologia, mesma parametrização e mesmos atributos como entrada, sendo que esta rede foi projetada para trabalhar com uma variedade maior de padrões de dados de entrada. Já no cenário C, cada rede foi projetada para trabalhar apenas com o padrão de entrada da classe à

qual a mesma está associada, tornando-as especialistas para aquele padrão e dividindo a responsabilidade entre as redes de cada *cluster*.

Em relação ao teste de controle (cenário A), a clusterização de dados apresentou melhores resultados apenas quando aplicada antes do projeto automático dos previsores de cada *cluster*. Apesar do cenário C apresentar resultados melhores que A, seu desvio padrão ficou alto em relação ao teste de controle. Após uma análise mais detalhada, verificou-se nos casos de erros acima da média que as redes MLP associadas a alguns *clusters* estavam com taxa de erro alta, fato que elevou o erro médio do sistema como um todo.

5.3. Comitês de Máquinas

Dois comitês que utilizam o conceito de proximidade entre os padrões de entrada foram apresentados neste trabalho (Seções 4.4.1 e 4.4.2). Os seguintes cenários foram levantados para a avaliação destes comitês:

D. **Comitê por influência da distância:** aplica a clusterização (Seção 4.2), depois faz o projeto automático e seleção de atributos para cada *cluster* utilizando apenas os exemplos daquela classe (Seção 4.3) e gera um comitê de máquinas por influência de distância (Subseção 4.4.1) utilizando os dados da clusterização e todos os previsores associados aos neurônios da rede de Kohonen. A saída resultante é calculada pelo comitê através da Equação (27).

E. **Comitê por diagrama de Voronoi:** aplica a clusterização (Seção 4.2), depois faz o projeto automático e seleção de atributos para cada *cluster* utilizando apenas os exemplos daquela classe (Seção 4.3) e gera um comitê de máquinas por diagrama de Voronoi (Subseção 4.4.2) utilizando os dados da clusterização e os previsores de cada *cluster*. A saída resultante é calculada pelo comitê através da Equação (30).

F. **Comitê de média (*Ensemble-Averaging*):** este comitê, amplamente aplicado na literatura, foi utilizado como parâmetro de comparação dos desempenhos dos comitês propostos neste trabalho. Aplica a clusterização (Seção 4.2), depois faz o projeto automático e seleção de atributos para cada *cluster* utilizando apenas os exemplos daquela classe (Seção 4.3) e gera um comitê de máquinas onde a saída resultante é a média aritmética da saída de todas as redes previsoras dos *clusters*, conforme Equação (22).

Os resultados obtidos com os cenários D, E e F podem ser observados na Tabela 3 e no gráfico ilustrado na Figura 30.

Tabela 3 - Resultados dos testes com os cenários D, E e F.

Cenário	RMSE	σ_{RMSE}	MAPE	σ_{MAPE}	MAE	σ_{MAE}
D	1,764	0,727	0,049	0,018	0,803	0,284
E	1,516	0,735	0,045	0,012	0,799	0,309
F	1,917	0,927	0,054	0,021	0,831	0,299

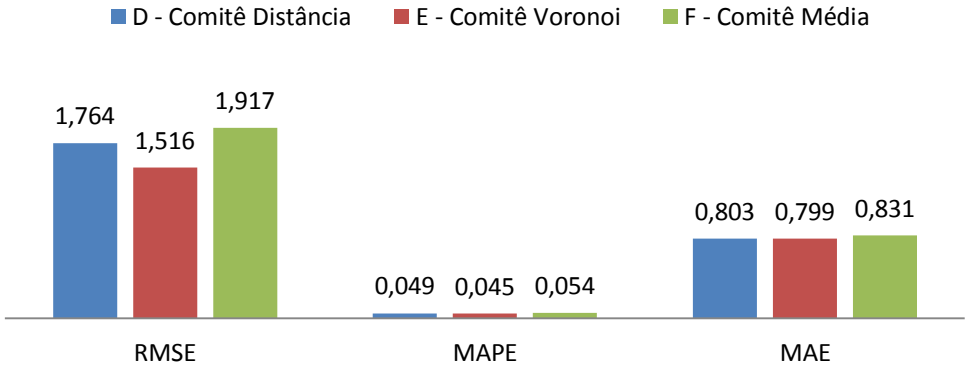


Figura 30 - Gráfico dos resultados dos testes com os cenários D, E e F.

Os resultados mostram que os comitês por influência da distância e por diagrama de Voronoi obtiveram desempenhos próximos e ambos tiveram acurácias melhores que o comitê padrão por média aritmética. A vantagem dos comitês propostos para o definido no cenário F é que ambos ponderam as saídas das redes levando em consideração a proximidade dos padrões apresentados ou a densidade dos dados, enquanto que o comitê por média aritmética trata todas as redes igualmente. A diferença entre os comitês por influência da distância e por diagrama de Voronoi é que o segundo, além de considerar a semelhança entre os dados, utiliza a probabilidade *a priori* (Equação (31)) para reduzir o peso das redes que possuíam poucos dados associados ao seu *cluster*.

Todos os comitês conseguiram resultados melhores que o teste de controle, exceto o RMSE do comitê por média aritmética (Cenário F), fato que indica que o cenário F possui erros mais dispares que os demais comitês, pois o RMSE é a medida mais sensível a este tipo de dados. Já nos desvios padrões, os resultados novamente foram melhores para o teste de controle. O motivo do desvio padrão elevado é o mesmo do explicado na seção anterior, pois os comitês utilizam as mesmas redes.

5.4. Utilização do erro na tomada de decisão

O erro médio dos previsores pode ser utilizado como uma medida para auxiliar o processo de tomada de decisão do sistema dando uma maior segurança sobre a precisão de cada previsor (Seção 4.5). O gráfico na Figura 31 exibe o erro médio das redes MLP associadas a cada um dos neurônios da grade de Kohonen aferidos no conjunto de treinamento e no conjunto de testes de uma das execuções dos testes. Neste gráfico é possível notar que, em geral, os erros nos conjuntos de validação e testes seguem a mesma tendência, ou seja, utilizar a avaliação dos previsores no conjunto de validação para ferramenta de auxílio na avaliação do conjunto de testes é viável e traz bons resultados, conforme pode ser observado ao longo desta seção.

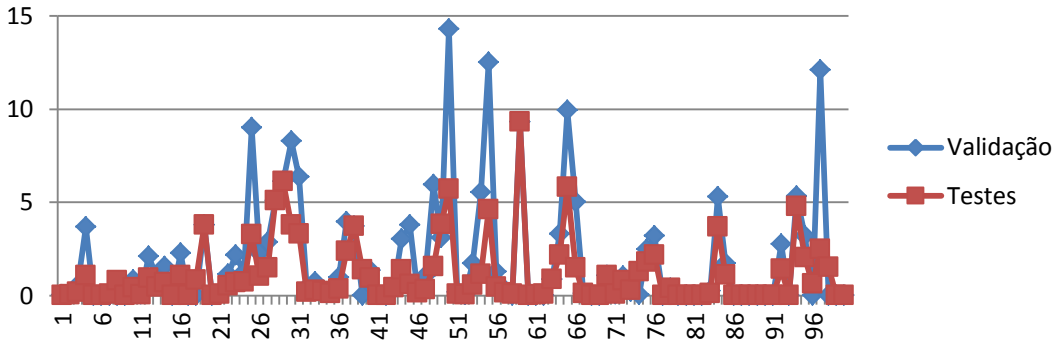


Figura 31 – Erro médio do conjunto de treinamento e do conjunto de testes calculado das redes associadas aos neurônios da grade de Kohonen.

Para a avaliação do uso do erro como ferramenta de auxílio na tomada de decisão, os cenários B e C foram reutilizados como parâmetro de comparação para os novos cenários levantados:

G. **Projeto + Atributos antes da clusterização com erro baixo:** utiliza a mesma estrutura definida no cenário B, com a diferença de o sistema se abster de realizar previsões para as classes de dados cujo erro associado a elas estão altos (Seção 4.5). Nesse caso utilizamos como ponto de corte *clusters* com taxa de erro acima de 5;

H. **Clusterização antes do Projeto + Atributos com erro baixo:** utiliza a mesma estrutura definida no cenário C, com a diferença que, assim como no cenário G, o sistema se abstém de realizar previsões para classes com erro alto.

Os resultados obtidos podem ser observados na Tabela 4 e nos gráficos da Figura 32. O gráfico da Figura 32(a) representa os testes realizados utilizando os cenários B e G, já o gráfico da Figura 32(b) representa os testes realizados utilizando os cenários C e H. A coluna ao final da tabela (*) representa a quantidade de abstenções médias observadas em cada

cenário, deve-se levar em consideração que o universo total do conjunto de testes é de 559 registros.

Tabela 4 - Resultados dos testes com os cenários B e G.

Cenário	RMSE	σ_{RMSE}	MAPE	σ_{MAPE}	MAE	σ_{MAE}	*
B	3,160	0,501	0,074	0,009	1,106	0,074	0
G	2,263	0,415	0,072	0,009	0,952	0,083	35
C	1,462	0,471	0,045	0,006	0,719	0,078	0
H	1,037	0,279	0,042	0,007	0,613	0,071	42

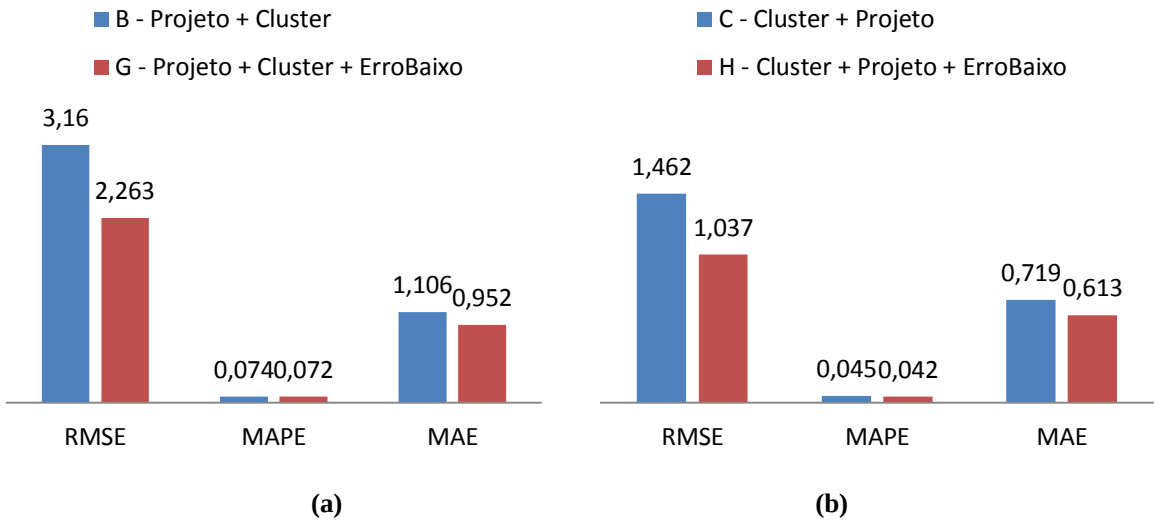


Figura 32 - Gráfico dos resultados dos testes com os cenários B e G **(a)**. Gráfico dos resultados dos testes com os cenários C e H **(b)**.

Os resultados demonstraram uma melhoria considerável obtida com a utilização da análise independente de previsores da grade. As medidas de erro avaliadas nos cenários B e C ficaram bem menores nos cenários G e H respectivamente. Apesar do cenário G apresentar melhorias, não chegou ao ponto de deixar a estratégia competitiva. Por outro lado, o cenário H além de reduzir ainda mais o erro, conseguiu também reduzir a variação dos resultados, conforme observado pelo desvio padrão do RMSE. O ponto negativo das estratégias levantadas pelos cenários G e H foram as abstenções realizadas, apesar de que em média, ficaram abaixo de 10% do conjunto total de dados apresentados. Assim, o problema dos previsores com taxas de erros altas foi amenizado por esta estratégia, onde o sistema se abstém de realizar previsões para os padrões de entrada mapeados aos neurônios com redes previsoras problemáticas.

5.4.1. Aumento gradativo do raio de aceitação

Para validar a utilização do aumento gradativo do raio de aceitação de Kohonen (definido na Subseção 4.2.1) no momento da seleção de exemplos foi realizado um teste utilizando cenários similares aos cenários C e H já descritos, a única diferença é que não foi utilizado o recurso do aumento gradativo do raio de aceitação (utilizaram apenas os dados realmente mapeados para aquele *cluster*), estes cenários foram denominados respectivamente de C' e H'. Os resultados obtidos com os cenários C' e H', em comparação com os resultados nos cenários C e H, podem ser observados na Tabela 5 e no gráfico da Figura 33. Devido à grande diferença entre os valores, o gráfico teve que ficar em escala logarítmica de base 10.

Tabela 5 - Resultados dos testes com os cenários C, C', H e H'.

Cenário	RMSE	MAPE	MAE	*
C	1,462	0,045	0,719	0
C'	247,646	0,713	8,807	0
H	1,037	0,042	0,613	42
H'	9,61	0,082	1,042	267

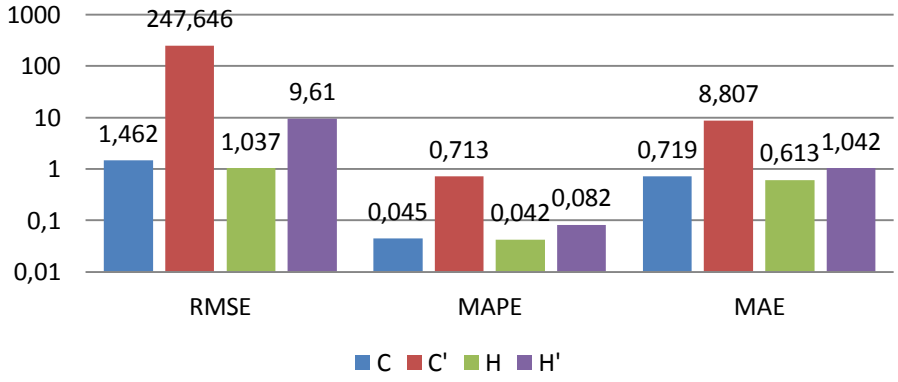


Figura 33 - Gráfico dos resultados dos testes com os cenários C, C', H e H'.

Os resultados mostram uma grande diferença entre os resultados dos cenários C e C', o motivo desta diferença é que, no segundo, a grande maioria dos previsores dos *cluster*s foi treinada com poucos dados. Esta diferença já não foi tão grande entre os cenários H e H', pois apesar de a grande maioria dos *clusters* terem sidos treinados com poucos dados, a análise independente dos previsores (Seção 4.5) mostrou ser um bom parâmetro de auxílio na tomada de decisão do algoritmo quanto a efetuar uma previsão ou se abster da mesma por falta de segurança na solução. O ponto fraco do cenário H' nesta simulação foi a alta quantidade de abstenções (muitos previsores com alta taxa de erro associada, muitos dados de entrada sem previsão) que ficou próxima de 50% dos padrões de entrada apresentados.

5.5. Seleção automática do previsor da classe

Utilizando todas as estratégias apresentadas neste trabalho e a análise independente de previsores, a utilização da seleção automática do previsor da classe é avaliada nesta seção. Para esta análise, os seguintes cenários foram levantados:

I. **Clusterização antes do Projeto + Atributos com erro baixo e comitê por influência da distância:** estratégia semelhante à do cenário H, com a diferença que, ao invés de se abster de previsões para classes com erro alto, foi aplicado o comitê de máquinas por influência da distância (Seção 4.4.1) nestas classes "problemáticas". A saída resultante é definida pela rede MLP associada à classe do dado de entrada quando está possuir um erro médio associado baixo, caso contrário, a saída é definida pelo comitê de máquinas.

J. **Seleção automática de previsores:** estratégia onde, para cada classe de dados, todos os previsores são avaliados, conforme explicado na Seção 4.5 e estendido na Seção 4.6, e o previsor selecionado para trabalhar com aquele padrão de entrada é o previsor com menor erro associado na grade tridimensional de erros associados aos neurônios, similar à apresentada na Figura 26. Para este teste os seguintes previsores estiveram envolvidos na avaliação: rede de controle (Cenário A), rede MLP projetada para a classe (Cenário C), comitê por influência da distância (Cenário D), comitê por diagrama de Voronoi (Cenário E), comitê por média aritmética (Cenário F).

Os resultados obtidos com os cenários I e J podem ser observados na Tabela 6 e no gráfico da Figura 34, o cenário H também foi utilizado como parâmetro para comparação.

Tabela 6 - Resultados dos testes com os cenários H, I e J.

Cenário	RMSE	σ_{RMSE}	MAPE	σ_{MAPE}	MAE	σ_{MAE}	*
H	1,037	0,279	0,042	0,007	0,613	0,071	42
I	1,547	0,675	0,052	0,012	0,733	0,138	0
J	0,919	0,176	0,033	0,002	0,556	0,055	0

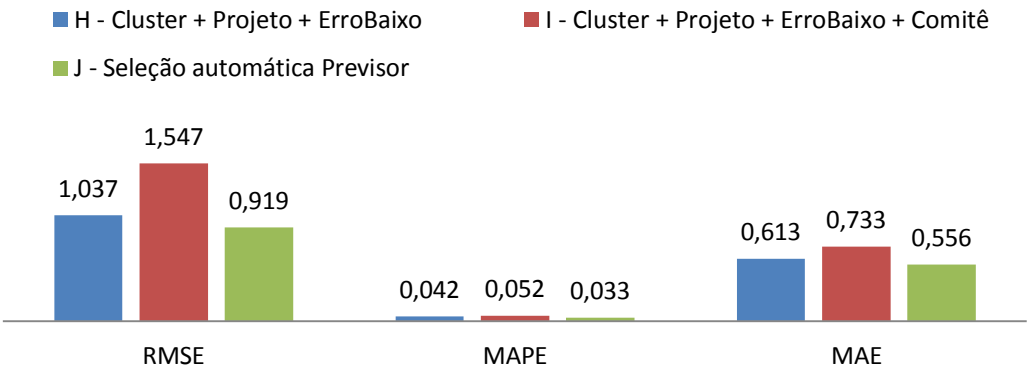


Figura 34 - Gráfico dos resultados dos testes com os cenários H, I e J.

Considerando o cenário I, é possível perceber pelos resultados que a utilização de uma estratégia apenas nas classes onde o erro está alto ficou acima dos resultados do cenário H (que tem o problema das abstenções de registros em classes com erros altos), perde-se um pouco em desempenho, eliminando-se as abstenções. Já a utilização da seleção automática dos previsores conseguiu resultados melhores que os obtidos no cenário H sem o problema das abstenções, pois todos os dados do conjunto de teste neste caso foram previstos. A tendência para o cenário J é que o desempenho do algoritmo melhore à medida que novas estratégias sejam adicionadas ao seu leque de previsores disponíveis.

O cenário J demonstrou, até esta etapa de testes, ser a melhor estratégia elaborada, pois conseguiu reduzir o erro entre 40 e 50% considerando todas as medidas de erro aferidas em relação ao cenário de controle. Inclusive todos os desvios padrões ficaram abaixo do cenário de controle, mostrando-se confiável e evidenciando que o problema de *cluster*s com redes MLP associadas com alta taxa de erro foi resolvido por esta estratégia.

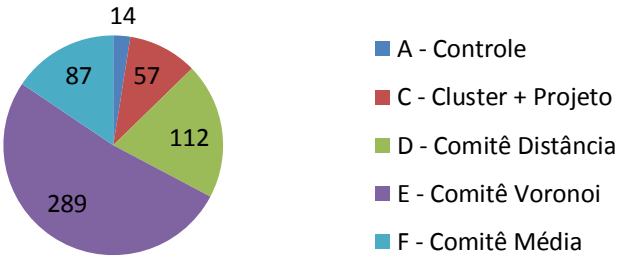


Figura 35 - Utilização dos previsores no conjunto de testes.

A Figura 35 apresenta a quantidade de vezes que cada previsor foi utilizado durante a execução de uma das baterias dos testes para o cenário J. Este cenário utilizou a clusterização dos dados e a analise independente dos previsores (por *cluster* e estratégia) para conseguir

unir várias estratégias de tal forma que o desempenho ficou melhor que cada uma das estratégias atuando isoladamente.

Analisando este gráfico é possível notar que, apesar do cenário C, no geral, ter taxa de erro médio menor que os cenários compostos por comitês (Cenários D, E e F), foi utilizado menos vezes que qualquer um dos comitês. O motivo dessa divergência é o fato de os comitês terem bom desempenho na maioria dos *cluster*s, mas apresentaram taxas de erros altas em alguns outros, fato que acabou elevando a taxa de erro global dos cenários.

A rede treinada no cenário A também foi adicionada ao portfólio de estratégias disponíveis por ser uma rede treinada com todos os dados do conjunto de treinamento, ou seja, possui uma capacidade de generalização maior. Os resultados mostraram que esta rede "genérica" foi utilizada poucas vezes, apenas em casos extremos onde nenhuma das outras estratégias apresentadas conseguiu um bom resultado.

5.6. Análise dos cenários

Após a apresentação de todas as estratégias propostas neste trabalho, um comparativo entre elas foi realizado, os erros médios descritos na Tabela 7 podem ser melhor comparados no gráfico ilustrado na Figura 36.

Tabela 7 - Resultados dos testes com todos os cenários na VALE5.

Cenário	RMSE	σ_{RMSE}	MAPE	σ_{MAPE}	MAE	σ_{MAE}	*
A	1,886	0,284	0,069	0,004	0,921	0,065	0
B	3,160	0,501	0,074	0,009	1,106	0,074	0
C	1,462	0,471	0,045	0,006	0,719	0,078	0
D	1,764	0,727	0,049	0,018	0,803	0,284	0
E	1,516	0,735	0,045	0,012	0,799	0,309	0
F	1,917	0,927	0,054	0,021	0,831	0,299	0
G	2,263	0,415	0,072	0,009	0,952	0,083	35
H	1,037	0,279	0,042	0,007	0,613	0,071	42
I	1,547	0,675	0,052	0,012	0,733	0,138	0
J	0,919	0,176	0,033	0,002	0,556	0,055	0

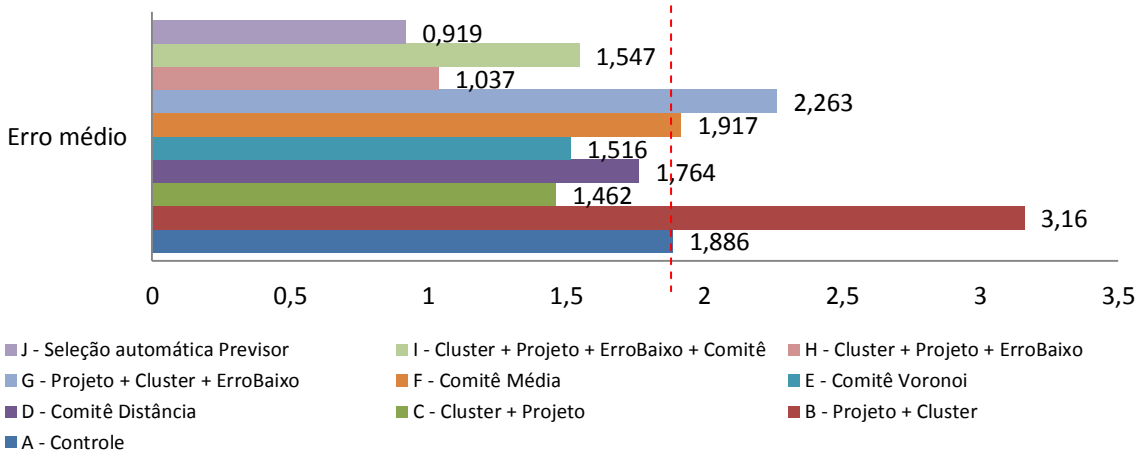


Figura 36 - Erro médio dos cenários com a VALE5.

O projeto automático das redes MLP teve desempenho melhor quando realizado após a clusterização, pois os piores resultados foram os cenários B e G, que realizavam o projeto automático antes da clusterização.

Os dois comitês de máquinas apresentados neste trabalho apresentaram bons resultados em relação ao cenário de controle, mas foram mais úteis como estratégias secundárias tanto no cenário I, como no cenário J.

A análise independente de previsores se mostrou um ótimo ponto de apoio como avaliação de segurança nos resultados, como pode ser observado: sua utilização sempre reduziu o erro em relação à mesma implementação que não a utilizava. Além disso, o fato de que os três melhores cenários encontrados utilizavam, de alguma forma, a análise independente de previsores.

Com base em todos os testes, a estratégia com menor erro médio foi aquela descrita no cenário J, seguida pelo cenário H, porém as abstenções para alguns dados, dependendo do caso/domínio de aplicação, podem ser inaceitáveis. Um fato interessante é que o cenário J utilizou a estrutura de cinco cenários com resultados relativamente imprecisos (A, C, D, E e F), realizou a análise independente dos previsores em todos os neurônios da grade, para direcionar qual estratégia deve efetuar a previsão de cada padrão de entrada para assim, conseguir um resultado bem melhor que o teste de controle (e todos os outros cenários), com o menor desvio padrão encontrado.

5.7. Outras séries temporais

Nesta seção, são apresentados os resultados dos testes aplicados na previsão de outras duas séries temporais econômicas: preço de fechamento da empresa Petrobrás (PETR4) e

preço de fechamento da empresa Usiminas (USIM5). Estes testes também consideram como atributos de entrada as nove séries temporais enumeradas no início deste capítulo. O período também ficou entre janeiro de 2000 e dezembro de 2008, a única diferença projetada foi a quantidade de baterias realizadas que, ao invés de trinta, foi de apenas uma para cada cenário.

5.7.1. Petrobrás (PETR4)

O comportamento do preço de fechamento da empresa Petrobrás (PETR4) durante o período pode ser visualizado na Figura 37. Os resultados avaliados nos testes de previsão foram detalhados no gráfico da Figura 38.

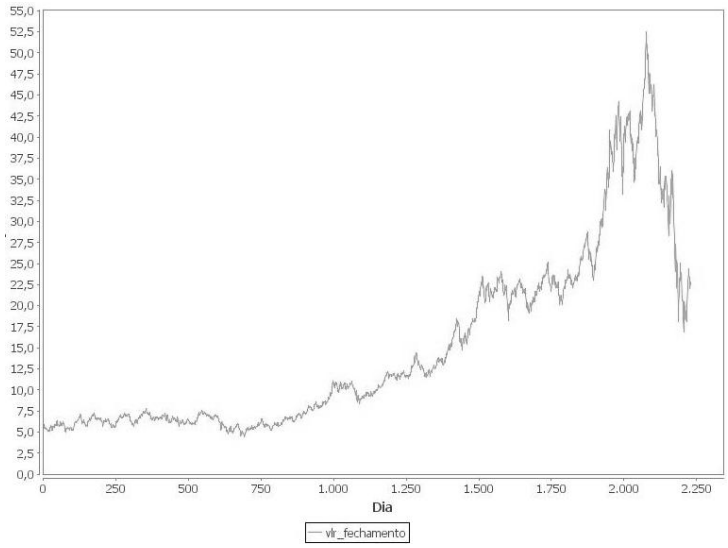


Figura 37 - Evolução do preço de fechamento da PETR4.

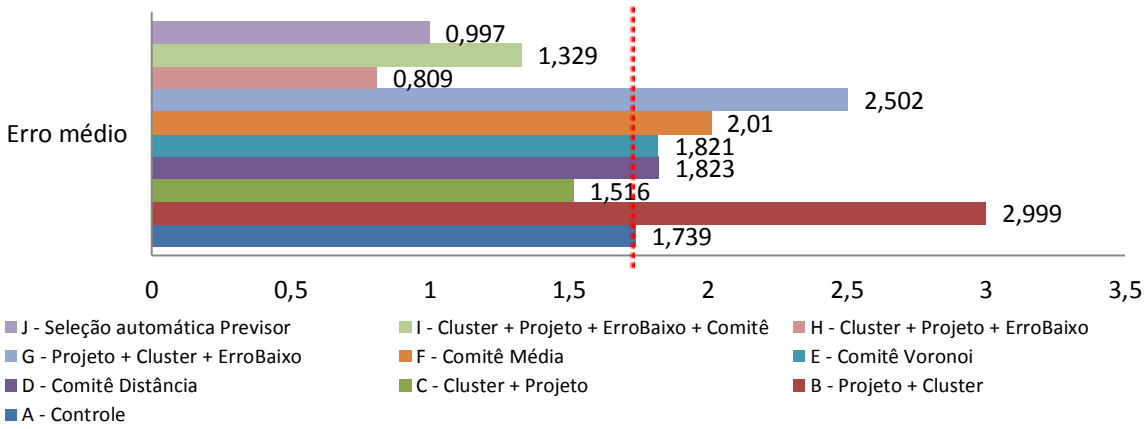


Figura 38 – Erro médio nos cenários para a PETR4.

Os resultados dos testes com as séries da PETR4 tiveram algumas diferenças em relação aos resultados obtidos com a VALE5 (ilustrados na Figura 36). A questão do projeto

automático das redes MLP antes da clusterização se manteve como a pior estratégia, Novamente os cenários B e G ficaram para trás, onde o G tem uma taxa de erro menor devido à análise independente de previsores.

Os desempenhos dos dois comitês de máquinas apresentados neste trabalho foram levemente inferiores ao teste de controle nesta execução. Em virtude da pequena diferença e levando em consideração os desvios padrões obtidos com estas estratégias para as séries da VALE, pode-se considerar que os desempenhos destes estão dentro do intervalo previsto.

A análise independente de previsores se manteve como um ótimo ponto de apoio como avaliação de segurança nos resultados, pois os três melhores cenários encontrados novamente utilizavam a análise independente de previsores.

Com base em todos os testes, a estratégia com menor erro médio foi aquela descrita no cenário H, seguida pelo cenário J. As duas melhores estratégias observadas nos testes da VALE se mantiveram na frente, porém trocaram de posição. Novamente, se levado em consideração o desvio padrão adicionado ao fato de três das estratégias (os comitês) que compõem seu leque de opções terem ficado com erro levemente maiores, é uma diferença aceitável.

5.7.2. Usiminas (USIM5)

O comportamento do preço de fechamento da empresa Usiminas (USIM5) pode ser visualizado na Figura 39. Os resultados foram detalhados no gráfico da Figura 40.

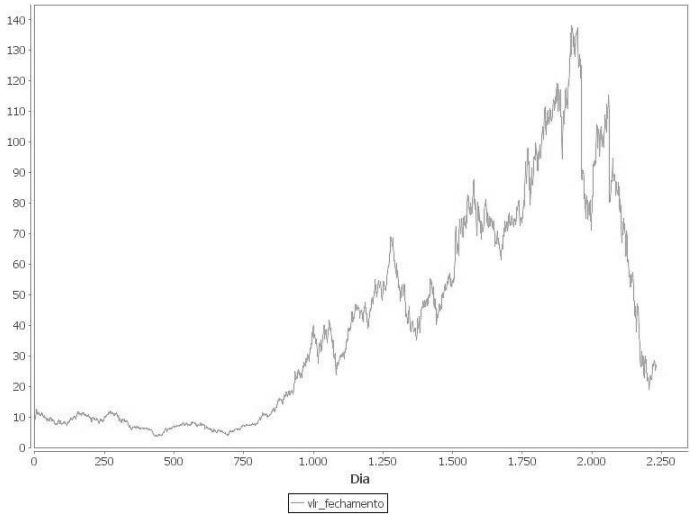


Figura 39 - Evolução do preço de fechamento da USIM5.

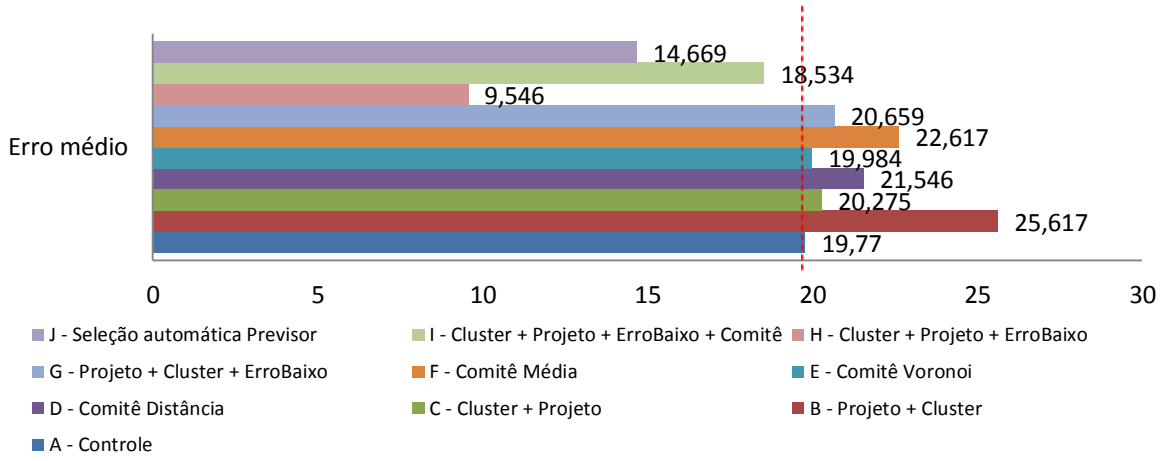


Figura 40 – Erro médio nos cenários para a USIM5.

As séries da USIM5, devido à sua grande variação no preço da ação ao longo do tempo, apresentaram taxas de erros superiores em todos os cenários (inclusive o de controle – A). Os resultados ficaram proporcionalmente similares aos resultados obtidos com a PETR4, todos os comitês e o cenário C tiverem desempenhos levemente inferiores ao cenário A (Controle), porém quando combinados no cenário J, o erro baixou consideravelmente ficando bem melhor que o cenário de controle. Os cenários G e H confirmaram a boa performance das abstenções (inclusive o cenário H obteve o melhor resultado) porém a quantidade de abstenções foi muito alta: 95 e 127 respectivamente.

Nesta simulação, o cenário H se saiu melhor que o cenário J devido ao fato de todos os previsores utilizados terem taxas de erro altas, ou seja, para o J, mesmo o melhor previsor conhecido obteve taxa de erro alta. Já o cenário H, ao encontrar um previsor com taxa de erro alta, simplesmente se absteve de realizar aquela previsão, prova disso foi a alta quantidade de abstenções (127).

5.8. Pontos negativos encontrados na abordagem

Os ganhos com a clusterização de dados, análise independente dos previsores e seleção automática dos previsores foram comprovados ao longo de todos os testes apresentados, porém alguns pontos negativos da abordagem foram levantados durante a execução dos testes.

O algoritmo de pré-processamento procura clusterizar todos os dados apresentados durante o treinamento e, para cada *cluster* é projetado uma rede MLP previsora com atributos de entrada, topologia da rede e calibragem dos principais atributos otimizados para aquela classe de dados. Toda esta automatização beneficia o usuário no sentido de não ser responsável por tomar todas estas decisões. Porém, este beneficio é refletido no tempo de

processamento do algoritmo de pré-processamento, que aumenta conforme seja desejado fazer uma busca num conjunto maior de soluções, aumentando a quantidade de gerações ou cromossomos no algoritmo genético utilizado pelo mesmo.

Conforme já mencionado na seção 4.3, aumentando a quantidade de atributos ou o tamanho da janela de tempo dos atributos, o tamanho do cromossomo do GA irá aumentar. Tal aumento amplia o espaço de soluções a ser explorado na busca por uma solução de boa qualidade. Neste ponto, deve-se tomar uma decisão: aumentar a quantidade de gerações e/ou indivíduos por geração do GA ou realizar a busca da solução de boa qualidade dentro de um subconjunto proporcionalmente menor.

Outro ponto negativo encontrado é a sensibilidade do algoritmo a previsores "problemáticos" de *cluster*s em alguns cenários. O cenário mais sensível a este problema foi o cenário C onde, se um previsor da grade foi mal configurado e está apresentando baixa acurácia, todos os dados mapeados para este *cluster* serão previstos com uma taxa de erro provavelmente alta, o que eleva o erro global médio do cenário. Os comitês de máquinas também são afetados por utilizarem todos os previsores no momento de calcular a saída, porém estes são impactados em menor escala, pois, na média, este erro acaba sendo minimizado pelas demais redes previsoras com bom desempenho.

A Figura 31 ilustra as taxas de erros das redes associadas aos neurônios da grade de Kohonen. Nesta figura também é possível verificar este problema dos *clusters* problemáticos, onde as redes MLP associadas àqueles *clusters* estão com altas taxas de erro. Através da análise independente dos previsores da grade e da perfeita divisão dos dados via clusterização foi possível apontar os *clusters* problemáticos, descartá-los e retreiná-los até que os mesmos apresentassem resultados melhores.

Os cenários G e H geralmente têm resultados melhores que os cenários B e C, respectivamente, justamente por atacarem este ponto e se absterem de realizar previsões para os padrões de entrada pertencentes aos *clusters* problemáticos ou apenas *clusters* com taxa de erro superior ao limite definido para o mesmo. O cenário J também não é afetado por este problema por escolher o previsor de menor erro para cada padrão de entrada apresentado. Caso o previsor daquele *cluster* seja problemático, o sistema busca alguma outra estratégia que contorne melhor o problema daquela classe de dado.

CONCLUSÃO

Este trabalho apresentou uma abordagem para previsão de séries temporais que utilizou clusterização dos dados para dividir o problema em problemas menores que foram trabalhados de forma independente, atrelado à seleção automática de previsores para otimizar a previsão de cada classe de dados (de cada subparte do problema), reduzindo assim a taxa de erro global do sistema. De uma forma geral, a clusterização dos dados mostrou-se válida, pois dividiu os dados em várias classes, permitindo que cada uma fosse trabalhada em separado, sem influenciar nas demais. Realizar o projeto automático e a seleção de atributos apenas com dados específicos da classe de dados gerou resultados melhores do que realizar estes passos antes da clusterização. A utilização do erro médio de cada previsor como informação auxiliar na tomada de decisão, reduziu consideravelmente o erro e mostrou ser bastante confiável, resultando nas menores taxas de erros aferidos assim como os menores desvios padrões calculados.

Na proposta mais simples da utilização da análise dos previsores (Cenário H), o erro caiu drasticamente, porém uma pequena quantidade (menos de 10%) de dados ficou sem previsão (abstenções). Na abordagem da utilização da análise de previsores mais complexa (Cenário J), em que os desempenhos de todas as estratégias são avaliados para cada *cluster*, o erro caiu em proporções similares à primeira estratégia com a vantagem de não ter havido dados sem previsões (sem abstenções).

Os comitês de máquinas, em média, geraram bons resultados na maioria dos *clusters*, porém o erro global ficou alto devido ao fato de não conseguir manter um padrão de desempenho para todos os *clusters*. Apesar de não se mostrarem uma boa estratégia como previsores de todos os dados, ofereceram grandes melhorias trabalhando como estratégias secundárias sendo adicionados ao portfólio de previsores disponíveis para a seleção automática de previsores.

Esta proposta demonstrou ser uma boa alternativa para aumentar a confiança das previsões reduzindo a diferença entre o valor previsto e o valor real (desejado), podendo ser utilizada em sistemas de apoio a tomada de decisão que precisam se antecipar para definir estratégias de investimento, definindo pontos de entrada e saída do mercado de ações, por exemplo. Este sistema seria de simples configuração pois esta estratégia não requer muitas configurações por parte do usuário.

Apesar de a validação ter sido realizada em séries temporais econômicas, não existem restrições para que esta abordagem seja utilizada em outros domínios de aplicação. Basta

reunir as séries temporais consideradas relevantes para o problema e definir para qual atributo que se deseja realizar as previsões, o que é mais um ponto positivo para abordagem: não é necessário ser um especialista na área de domínio. O projeto automático dos previsores se encarregará de desconsiderar os atributos irrelevantes, definir a topologia das redes previsoras, assim como calibrá-las. Vale ressaltar que os resultados e conclusões apresentados neste trabalho foram embasados em séries temporais econômicas. Apesar de não haver restrições para outros tipos de séries temporais, não é possível garantir que o comportamento da proposta seja o mesmo.

Um dos pontos fracos da abordagem foi o tempo necessário para o pré-processamento definido no algoritmo de pré-processamento (Figura 12), pois as dimensões da grade de Kohonen afetam consideravelmente o tempo de treinamento. Após esta fase inicial, as previsões realizadas pelo algoritmo são realizadas em questão de milésimos de segundos.

As configurações da rede de Kohonen, os atributos considerados para a clusterização e o limite mínimo de dados para treinamento das redes MLP de cada neurônio foram definidos empiricamente. Após vários testes e observações foi concluído que a quantidade de neurônios e épocas influenciam na qualidade do sistema. Para otimizar estas configurações pode-se buscar alguma heurística capaz de calibrar estes dados.

Como trabalhos futuros, pode-se testar esta abordagem aplicada a outros tipos de séries temporais, além das séries econômicas. Outro trabalho futuro seria procurar alguma forma de reduzir o erro das redes com altas taxas de erros que ficaram associadas a alguns *cluster*s ou reduzir o impacto das mesmas nos comitês de máquinas.

REFERÊNCIAS BIBLIOGRÁFICAS

Abelém, A. J. G. Redes Neurais Artificiais na Previsão de Séries Temporais. Dissertação de Mestrado, Departamento de Engenharia Elétrica, PUC-RJ, Rio de Janeiro, 1994.

Adeodato, P. J. L., Arnaud, A. L., et al., *The Role of Temporal Feature Extraction and Bagging of MLP Neural Networks for Solving the WCCI 2008 Ford Classification Challenge*., 2009.

Antunes, C. M., Oliveira, A. L., *Temporal Data Mining: An overview, Workshop on Temporal Data Mining*, 7th ACM SIGKDD *International Conference on Knowledge Discovery and Data Mining* (KDD'01), 2001

Barreto, G. A., Araújo, A. F. R., Aplicação de rede auto-organizável de Kohonen na Predição de Séries Temporais Caóticas, Departamento de Engenharia Elétrica (USP), 2001

Bastos, E. N. F., Uma rede auto-organizável construtiva para aprendizado perpétuo de padrões espaço-temporais, Dissertação de Mestrado, Universidade do Rio Grande do Sul, 2007.

Bastos, F. A. A., Previsão de receitas tributarias mediante redes neurais artificiais, Dissertação de mestrado profissionalizante, Universidade Estadual do Ceará – UECE, 2010.

Bishop, C. M., *Neural Networks for Pattern Recognition.* Clarendon Press, Oxford, 1995.

Box, George and Jenkins, Gwilym, *Time series analysis: Forecasting and control*, San Francisco: Holden-Day, 1970.

Bollerslev, T., *Generalized Autoregressive Conditional Heteroskedasticity*, Journal of Econometrics, 1986.

Boozarjomehry, R. B., Svrcek, W. Y., *Automatic design of neural network structures*, Computer and Chemical Engineering, 2001.

Chagas, R. P., Aplicação de Redes Bayesianas na previsão de crescimento de fluxos de caixa. Dissertação de mestrado profissionalizante, Escola de economia de São Paulo, 2008.

Chappelier, J. C., Grumbanch, A., *A Kohonen Map for Temporal Sequences*, In Proceedings of NEURAP'95, 1996

Chen, C. H., Lin, Z. S., *A committee machine with empirical formulas for permeability prediction*, Computer & Geosciences 32, 2006.

Crochat, P., Franklin D., *Back-Propagation Neural Network Tutorial*, http://ieee.uow.edu.au/~daniel/software/libneural/BPN_tutorial/BPN_English/BPN_English/BPN_English.html, acessado em 15/06/2008

Cybenko, G., *Continuous Valued Neural Networks with Two Hidden Layers are Sufficient*. Technical Report, Tuft University. 1988.

De castro, M.C.F. Predição Não-Linear de Séries Temporais Usando Redes Neurais RBF por Decomposição em Componentes Principais. Tese de Doutorado, Faculdade de Eng. Elétrica e Computação, UNICAMP, 2001.

Fahlman, S. E., Lebiere, C., *The Cascade-Correlation Learning Architecture*, Technical Report, School of Computer Science, Carnegie Mellon University, 1990.

Figueredo, C. J. Previsão de Séries Temporais Utilizando a Metodologia Box & Jenkins e Redes Neurais para Inicialização de Planejamento e Controle de Produção. Dissertação de Mestrado, Universidade Federal do Paraná,Curitiba, 2008.

Fu, T. C., Chung F. L., Ng, V., Luk, R., *Pattern Discovery from Stock Time Series Using Self-Organizing Maps*, Notes KDD2001 Workshop Temporal Data Mining, 2001.

Gomes, D. T. Redes Neurais Recorrentes Para Previsão de Séries Temporais de Memórias Curta e Longa. Dissertação de Mestrado, UNICAMP, 2005.

Guo, J., *Improving market clearing price prediction by using a committee machine of neural networks*, IEEE Transactions on Power Systems - IEEE TRANS POWER SYST , vol. 19, no. 4, pp. 1867-1876, 2004

Guyon, I., Elisseeff, A., *An introduction to variable and feature selection*, Journal of Machine Learning Research, vol. 3, 2003.

Haykin, S., Redes Neurais - Princípio e prática, 2ª edição, Editora Bookmam, 2001

Heaton, Jeff, *Introduction to Neural Networks with JAVA*. Heaton Research Inc., 2005

Hebb, D., *The Organization Behavior: A Neuropsycological Theory*, John Wiley & Sons, New York, 1949.

Hung, S. Y., *Integrating arbritage princing theory and artificial neural networks to support portfólio management*. Elsevier Science B.V., 1996

ICMC – USP, http://www.icmc.usp.br/~andre/research/neural, acessado em 20/06/2011.

Iyoda, E. M., Inteligência Computacional no Projeto Automático de Redes Neurais Híbridas e Redes Neurofuzzy Heterogêneas, Dissertação de Mestrado, Universidade Estadual de Campinas, 2000.

Kalyvas, E., *Using neural networks and genetic algorithms to predict stock market returns*, 2001.

Keogh, E., Kasetty, S., *On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration*. 8th ACM SIGKDD, *Conference on Knowledge Discovery and Data Mining*. Edmonton, Alberta, Canada, 2002.

Kim, K. J., Han I., *Genetic algortims approach to feature discretization in artificial networks for the prediction of stock price index*, Expert Systems with Applications 19, Pergamom, 2000.

Kohavi R., Sommerfield. D., *Feature Subset Selection Using the Wrapper Method: Overfitting and Dynamic Search Space Topology*, First International Conference on Knowledge and Data Mining (KDD-95), 1995

Kohavi, R., John, G. H., *Wrappers for features subsert selection*, 1996.

Kohonen T., *Self-organized maps*, 2a.edição, Springer-Verlag, 1997.

Lapedes, A., farber, R., *How neural nets work*. Neural information processing systems. American institute of physics. 1988.

Lendasse, A., Verleysen, M., Bodt, E., *Forecasting Time-Series by Kohonen Classification*. European Symposium on Artificial Neural Networks (ESANN'1998)

Liao,Warren T., *Clustering of time series data - A survey*, Pattern Recognition 38, Elsevier, 2005.

Lin G. F., Chen L. H., *Time Series forecasting by combining the radial basis function network and the self-organizing map*, Hydrological Processes 19, 2005.

Linden, R., Algoritmos Genéticos: Uma importante ferramenta da Inteligência Computacional, Ed. Brasport, 2006.

Lipmann, R. P., *An introduction to computing with neural nets*, IEEE ASSP Magazine. 1987.

Magri, D., Material de aula sobre Redes Neurais Artificiais, Universidade do Estado de Santa Catarina, 2011

Mantegna, R. N., *Hierarchical Structure in Financial Markets*. European Physical Journal, vol. B11, 1999.

McCulloch, W., Pitts, W., *A logical calculus of the ideas imminent in nervous activity*, Bulletin of Mathematical Biophysics 5, 1943

McNelis, P. D., *Neural Networks in Finance*, Elsevier Academic Press, 2005

Menezes, J. M. P., Redes neurais dinâmicas para predição e modelagem não-linear de series temporais, Departamento de Engenharia de Teleinformática, Universidade Federal do Ceará (UFC), 2006.

Moura, F. A., O uso de redes neurais artificiais na previsão de tendências no mercado de ações, 2006.

Naftaly, U., Intrator N., Horn, D., *Optimal ensemble averaging of neural networks*, Network: Computation in Neural Systems 8, 1997.

Rodrigues, A. C. N., Intervalo de predição em RBF: Análise e proposta de extensão. Dissertação de mestrado, Programa de pós-graduação em Ciência da Computação - UFSC, 2010.

Rosemblatt, F., *The perceptron: A probabilistic model for information storage and organization in the brain,* Psycological Review 65, 1958.

Rumelhart, D. E., Hinton, G. E., Williams, R. J., *Learning internal representation by back-propagating errors*¸ PDP Research Group (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, MIT Press, MA, 1986.

Russel, S. J., Norvig, P., *Artificial Intelligence: A Modern Approach*, Prentice Hall, 1995.

Sharkey, A. J. C., *On combining artificial neural nets*, Connection Science 8, 1996.

Sikora, G., Projeto automático de redes neurais artificiais para o problema de previsão em series temporais, 2008.

Sikora G., Uma abordagem híbrida bio-inspirada aplicada à melhora na qualidade do reconhecimento de padrões, Dissertação de mestrado, UniversidadeEstadual do Ceará, 2011.

Siqueira, E. S., Almeira, I. L., Cechin, A. L., Utilização de Mapas Auto-organizáveis no pré-tratamento de Dados de Microarranjos. CRICTE 2006

Souza, J. T., *Feature Selection with a General Hybrid Algorithm*, Tese de doutorado, Ottawa-Carleton Institute of Computer Science, 2004.

Tresp, V., *A Bayesian Committee Machine*, Neural Computation, 2000

Vellido, A., Lisboa, P. J. G., Vaughan, J., *Neural networks in bussiness: a survey of applications* (1992-1998). Expert Systems with Applications, 1999.

Villanueva, W. J. P., Comitê de Máquinas em predição de series temporais, Dissertação de mestrado, Universidade Estadual de Campinas, 2006.

Von Zuben, F. J., Modelos Paramétricos e Não-Paramétricos de Redes Neurais Artificiais e Aplicações, Universidade Estadual de Campinas, 1996.

Zhang, G., *Time series analysis and prediction by neural networks*. Optimization Methods and Software. 1994.

Zhang, G., Patuwo, B. E., Hu, M. Y., *Forecasting with artificial neural networks: The state of art*. Internationall Journal of Forecasting. 1998

Zhang, G. P., *Neural Networks in Bussiness Forecasting*. Information Science Publishing, 2003