Redes Neuro-Fuzzy Evolutivas Embarcadas em Sistemas Microcontrolados

Israel Mendes[1], Pyramo Costa[1], Luis Bergo[1] e Daniel Leite[2]

[1] Programa de Pós Graduação em Engenharia Elétrica
Pontifícia Universidade Católica de Minas Gerais, Brasil
[2] Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação, Brasil
professorisraelt@gmail.com, pyramo@pucminas.br, danfl7@dca.fee.unicamp.br

Abstract. Este trabalho propõe o estudo de modelos de inferência neuro-fuzzy evolutivos (DENFIS) para identificação de sistemas dinâmicos. O aspecto evolutivo da abordagem neuro-fuzzy proposta se refere a adaptação estrutural do modelo a partir de fluxos de dados. Particularmente, este trabalho enfatiza sistemas DENFIS embarcados em microcontroladores para o propósito de portabilidade. Avaliações de desempenho computacional mostram que o modelo DENFIS embarcado reproduz com boa precisão os resultados obtidos em simulações em software. Além disso, o tempo de resposta do modelo embarcado se mostrou apropriado para uso em ambiente online. Um exemplo de aplicação utilizando dados reais de frenagem e acoplamento de trens em movimento foi considerado para ilustrar o desempenho do sistema neuro-fuzzy proposto.
abstract environment.

Keywords: Redes Neuro-Fuzzy, Processos Dinâmicos, Sistemas Evolutivos, Sistemas Embarcados, Inteligência Computacional

1 Introdução

Métodos convencionais para modelagem de sistemas dinâmicos se tornam imprecisos ou inadequados frente à necessidade de controlar processos cada vez mais complexos. Reconhecimento de voz e imagens, predição de séries temporais, controle robusto e classificação semi-supervisionada são exemplos de aplicações que requerem modelos estruturalmente adaptativos de acordo com fluxos de dados em ambiente online. Sistemas conexionistas evolutivos (ECOS) são abordagens no contexto de modelagem dinâmica não-linear e não-estacionária apropriadas para lidar com fluxos de dados. Estes sistemas são inspirados no comportamento de redes neurais biológicas e na evolução de indivíduos durante seu ciclo de vida: aprendendo a partir da experiência, herança e mudança gradual. Conhecimento é gerado a partir de tarefas repetitivas e de fluxos de dados produzidos através de percepções e enviados ao cérebro [1] [2] [3] [4]. O desenvolvimento de redes neurais evolutivas é gradual, i.e., regras e neurônios não são fixos nem pré-definidos, mas gerados sempre que novos dados são suficientemente informativos e não são

comuns ao modelo/entendimento atual. A principal diferença entre ECOS e demais sistemas conexionistas é que ECOS podem ser construídos e adaptados ao longo do tempo sem a necessidade de usufruir de dados de instantes passados. Algoritmos de construção e adaptação de modelos ECOS são dirigidos a fluxos de dados online [5] [7].

Segundo [3] [5] [6], um sistema conexionista é considerado evolutivo quando: (i) utiliza um algoritmo de aprendizagem incremental, i.e., um algoritmo que procede apenas um passo sobre os dados disponibilizados e em seguida os descarta; (ii) identifica e adapta gradualmente sua estrutura e parâmetros a novas condições; (iii) reconhece padrões espaço-temporais sem esquecer padrões anteriores; (iv) pode disponibilizar continuamente um sumário das regras que regem o seu comportamento.

Ressalta-se que o propósito de ECOS é diferente do propósito de redes neurais construtivas [11]. Enquanto redes neurais construtivas tentam otimizar um critério de adequação geralmente baseado na precisão da saída estimada, ECOS enfatizam não somente precisão, mas também interpretabilidade e escalabilidade de sistemas de informação. A literatura em reconhecimento neural de padrões refere-se a esta questão como dilema da estabilidade-plasticidade [12]. ECOS procuram contrabalancear (estabelecer um compromisso entre) adaptação paramétrica e estrutural para capturar mudanças abruptas e graduais de sistemas não-estacionários. A idéia é conciliar parsimoniosamente plasticidade e estabilidade.

Em [10] ressalta-se as semelhanças e diferenças entre modelos conexionistas e a teoria da computação evolucionária. Por exemplo, abordagens genéticas podem ser empregadas para adaptar parâmetros de ECOS, mas não é esta a característica que determina a evolução online desses sistemas. A literatura de sistemas evolutivos faz uma clara distinção entre técnicas evolucionárias, i.e. técnicas inspiradas na teoria da evolução de Darwin e noções de genética, e técnicas evolutivas de aprendizagem. A primeira não requer adaptabilidade online do modelo a partir de fluxos de dados nem adaptação estrutural.

Algumas das mais citadas abordagens no contexto de ECOS incluem: Simple Evolving Connectionist System (SECoS) [5] e Evolving Fuzzy Neural Network (EFuNN) [8]. SECoS foi proposto em [9] [10] com o propósito de identificação de fonemas. Estes sistemas são redes neurais artificiais com três camadas de neurônios, sendo que uma delas, a camada intermediária, evolui de acordo com o fluxo de dados. Em outras palavras, o número de neurônios da camada evolutiva de uma rede neural SECoS não é pré-determinado, mas encontrado de acordo com novas informações extraídas dos dados. EFuNN [8] é uma rede fuzzy-neural evolutiva de cinco camadas, sendo uma de entrada, uma com neurônios de agregação, uma com neurônios evolutivos, uma com neurônios de fusão (quantização), e uma camada de saída. EFuNN é equipada com um algoritmo incremental próprio para lidar com bases de dados grandes, possivelmente ilimitadas [3]. EFuNN foi aplicado com relativo sucesso na predição da série temporal caótica Mackey-Glass [8]. Posteriormente, em reconhecimento de fonemas, EFuNN mostrou-se superior a abordagem Learning Vector Quantization (LVQ).

Outros modelos de ECOS, tais como o Hybrid Neuro-Fuzzy Inference System (HyFIS) [3] e o Dynamic Evolving Neuro-Fuzzy Inference System (DENFIS) [6], foram propostos posteriormente. Sistemas fuzzy e neuro-computação são complementares em termos de interpretação e flexibilidade de aprendizagem, o que motiva a computação neuro-fuzzy. HyFIS e DENFIS fundamentalmente codificam bases de regras fuzzy em arquiteturas neurais dispostas em camadas, e.g., uma regra fuzzy Takagi-Sugeno evolutiva pode ser representada através de neurônios e conexões que ligam as entradas destas redes neurais a uma saída [3] [5] [6].

A inferência na abordagem DENFIS pode ser considerada econômica em termos de ocupação de memória com relação as abordagens HyFIS, SECoS e EFuNN [5] [6]. Em análise comparativa, demonstrou-se que modelos DENFIS são mais rápidos para processamento de certos tipos de dados numéricos. Velocidade de processamento é um aspecto essencial em se tratando de sistemas embarcados. A partir dos estudos em [5] e [6], várias investigações, como [14] e [15], buscaram implementar modelos evolutivos em hardware. É notável nestes estudos limitações dos modelos em função de restrições do hardware disponível.

A proposta deste trabalho é associar a tecnologia de sistemas embarcados a eficiência das atuais técnicas de modelagem evolutiva. Em particular, consideramos a abordagem de modelagem DENFIS. Conduzimos experimentos considerando um problema de frenagem parcial de trens em movimento para acoplamento suave de vagões sem a necessidade de parada. A parada de trens para acoplamento usualmente é o gargalo de operações de logística de transporte ferroviário. Considerou-se fluxos de dados reais de sensores de distância e velocidade relativa entre os trens para mostrar o comportamento do modelo DENFIS embarcado em microcontrolador. A saída do modelo DENFIS deve ditar/estimar a pressão a ser aplicada no freio do trem que se aproxima por trás para evitar colisão e descarrilamento. O modelo é desenvolvido a partir de dados nunca vistos e deve aprender novos conceitos e criar novas regras em tempo real.

O restante deste trabalho é estruturado da seguinte forma: a Seção 2 apresenta o método de clusterização evolutiva adotado para granular fluxos de dados. Métodos de clusterização incrementais são fundamentais na construção de modelos neuro-fuzzy quando a base de dados não é disponível de antemão. A Seção 3 descreve o algoritmo de adaptação dos parâmetros e estrutura do modelo DENFIS. A Seção 4 apresenta as rotinas para implementação do modelo DENFIS em microcontrolador. Na Seção 5 foram utilizados dados reais para avaliar o desempenho do sistema proposto na tarefa de frenagem parcial e acoplamento de trens. A Seção 6 conclui o trabalho e sugere propostas para trabalhos futuros.

2 Clusterização Evolutiva

O método de clusterização evolutiva ECM [3] tem um papel chave nos processos de desenvolvimento e aprendizagem de modelos evolutivos neuro-fuzzy. Este método de clusterização dispensa a disponibilidade *a priori* de toda a base de

dados. Seu princípio de funcionamento é baseado na partição iterativa do espaço de entrada em sub-conjuntos (clusters).

Em ECM, basicamente, um limiar de distância, D_{thr}, é considerado para decidir quando criar novos clusters, excluir ou expandir os clusters existentes. O limiar D_{thr} se refere ao raio máximo que um cluster pode ter - consideramos norma-2 ou distância Euclidiana, logo clusters se referem a hiperesferas de diâmetro máximo igual a $2 \times D_{thr}$ formadas no espaço de entrada q-dimensional. A distância Euclidiana entre dois vetores, por exemplo, $x = (x_1, ..., x_q)$ e $y = (y_1, ..., y_q)$, é dada por:

$$\|x - y\|_2 = \left(\sum_{i=1}^{q} |x_i - y_i|^2 \right)^{1/2} \tag{1}$$

Medidas de distância, como (1), são utilizadas para decidir se uma amostra é suficientemente próxima ao centro de um cluster. Em geral, qualquer medida de distância pode ser empregada. Os pontos centrais dos clusters são utilizados como referência (protótipos), i.e., são usados no algoritmo de aprendizado de DENFIS para cálculo do cluster mais próximo a uma determinada amostra de dados [5]. Mudanças no limiar D_{thr} durante o treinamento provocam alterações na acuidade da base de regras do DENFIS. Uma solução de compromisso deve ser encontrada para cada caso [3] [6]. Ressalta-se que os centros de clusters formados no ECM são equivalentes aos neurônios da camada evolutiva de uma rede EFuNN [5]. Uma vez que um cluster atinge seu diâmetro máximo, ele não pode ser mais alterado; eventualmente pode ser excluído.

O procedimento de clusterização ECM pode ser sumarizado no seguinte algoritmo:

Passo1: Ler amostra x^k*, onde* $k = 1, ...$ *é índice de tempo*
Passo2: Se $k = 1$*, criar cluster* C^1*, com raio* $r^1 = 0$ *e centro* $c^1 = x^k \rightarrow$*Passo 1*
Passo3: Calcular $D = \left\| x^k - c^j \right\|_2$*, e* $S^k = D + r^j$*, onde* c^j *é o centro do cluster mais próximo,* r^j *é o raio deste cluster.*
Passo4: Se $D < r^j$*, a amostra* x^k *pertence àquele cluster.* $\rightarrow$ *Passo 1*
Passo5: Se $S^k > 2 \times D_{thr}$*, então criar cluster* C^{novo} *com raio* $r^{novo} = 0$ *e centro* $c^{novo} = x^k \rightarrow$ *Passo 1*
Passo6: Se $S^k < 2 \times D_{thr}$*,então o cluster* C^j *é expandido. O seu novo raio passa a ser:* $r^{atualizado} = S^k/2 \rightarrow$ *Passo 1*

(Algoritmo do método de clusterização evolutiva ECM)

O algoritmo acima salienta a essência de abordagens dirigidas a fluxo de dados onde amostras são lidas e descartadas uma a uma. Dados históricos são dispensáveis, e a evolução dos clusters acontece gradualmente, em uma base incremental.

3 Sistema Neuro-Fuzzy Evolutivo DENFIS

Redes neurais DENFIS tomam como base a informação refletida nos dados de entrada e um grupo de m regras para estimar o valor de saída de um sistema dinâmico [3] [6]. Na fase de treinamento, redes DENFIS utilizam o método ECM para agrupar/clusterizar vetores de entrada similares. Este agrupamento dinâmico baseado no algoritmo ECM fornece características evolutivas ao DENFIS. Em fase de treinamento, os centros dos clusters ECM formam o antecedente de regras fuzzy. Além disso, pares de dados de entrada e saída são usados para identificar os parâmetros dos consequentes das regras fuzzy.

Sistemas de inferência fuzzy Takagi-Sugeno, codificados na arquitetura de redes neurais DENFIS, são descritos por um conjunto de regras do tipo:

$$\text{SE } x_1 \text{ é } R_{11} \text{ e ... e } x_q \text{ é } R_{1q} \text{ ENTÃO } y_1 \text{ é } f_1(x_1, \ldots, x_q)$$

$$\text{SE } x_1 \text{ é } R_{21} \text{ e ... e } x_q \text{ é } R_{2q} \text{ ENTÃO } y_2 \text{ é } f_2(x_1, \ldots, x_q)$$

$$\ldots$$

$$\text{SE } x_1 \text{ é } R_{m1} \text{ e ... e } x_q \text{ é } R_{mq} \text{ ENTÃO } y_m \text{ é } f_m(x_1, \ldots, x_q)$$

onde m é o número de regras; e q é o número de variáveis. $R_{ij}, i = 1, \ldots, q; j = 1, \ldots, m$ são conjuntos fuzzy. A pertinência de uma amostra $x = (x_1, ..., x_q)$ no conjunto fuzzy R_{ij} é μ_{ij}. Consideramos funções de pertinência triangulares definidas pelos parâmetros (a, b, c), logo:

$$\mu(x_i) = \begin{cases} 0 & \text{se } x_i \leq a; \\ \frac{x_i - a}{b - a} & \text{se } a \leq x_i \leq b; \\ \frac{x_i - c}{c - b} & \text{se } b \leq x_i \leq c; \\ 0 & \text{se } x_i \geq a. \end{cases}$$

A correspondência entre os parâmetros de uma função de pertinência triangular genérica μ e os parâmetros determinados pelo algoritmo ECM é: $(a, b, c) = (c^j - r^j, c^j, c^j + r^j)$. A Fig. 1 ilustra o processo de criação das funções de pertinência (antecedentes) de cada regra a partir de m clusters gerados pelo algoritmo ECM.

O consequente da j-ésima regra DENFIS é uma função linear afim:

$$f_j(x_1, ..., x_q) = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_q x_q \tag{2}$$

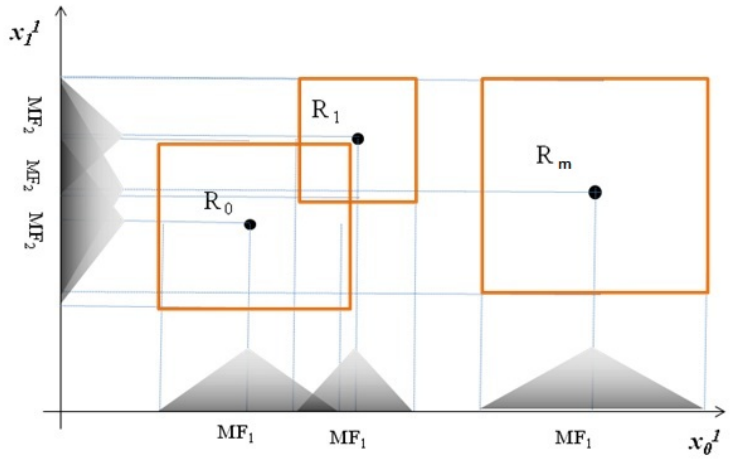

Fig. 1. Formação de funções de pertinência fuzzy a partir de clusters ECM

Sistemas de inferência fuzzy Takagi-Sugeno com funções polinomiais de ordem zero:

$$f_j(x_1, ..., x_q) = C_j \text{ onde } j = 1, ..., m$$

podem ser considerados para o propósito de classificação [14]. Similarmente, funções f_j de ordens superiores podem prover uma aproximação local mais precisa da função de um processo ao custo de um maior tempo de processamento e memória.

A estimação dos consequentes das regras DENFIS é realizada a partir de p pares entrada-saída disponibilizados pelo fluxo de dados:

$$\left([x_{i1}, x_{i2}, \ldots, x_{iq}], y_i\right), i = 1, 2, \ldots, p$$

Sugerimos o método dos mínimos quadrados (LSE) [16] para cálculo do vetor de parâmetros $B = [b_0, ..., b_q]^T$ em (2). Basicamente, B pode ser calculado diretamente a partir de:

$$B = (A^T A)^{-1} A^T y$$

onde

$$A = \begin{pmatrix} 1 & x_{11} & x_{12} & x_{1q} \\ 1 & x_{21} & x_{22} & x_{2q} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & x_{p1} & x_{p2} & x_{pq} \end{pmatrix}$$

e

$$y = [y_1, y_2, \cdots, y_p]^T$$

Em suma, os parâmetros de f_j, $j = 1, ..., m$, são calculados através de algumas amostras de entrada e saída acumuladas. As amostras são descartadas logo após o procedimento.

Um vetor de entrada x pode ativar múltiplas regras DENFIS devido a sobreposição dos conjuntos fuzzy antecedentes. Desta forma, a saída da rede neural y é dada pela média ponderada das regras ativas, i.e., regras R_j cujo grau de ativação:

$$\mu_j = T(\mu_{1j}, ..., \mu_{ij}, ..., \mu_{qj}) > 0$$

onde T é uma T-norma, e.g., o mínimo. Assumindo Γ regras ativas, $1 \leq \Gamma \leq m$, tem-se

$$y = \frac{\sum_{j=1}^{\Gamma} \mu_j f_j(x_0, ..., x_q)}{\sum_{j=1}^{\Gamma} \mu_j} \tag{3}$$

4 Sistemas Evolutivos Embarcados

Reproduzir softwares em versões embarcadas com mínima perda de eficiência se justifica pela redução do consumo de energia, redução de custo e portabilidade [17].

Estudos como [17] e [18] avaliaram formas de flexibilizar métodos inteligentes visando produzir versões mais rápidas e mais baratas em termos de memória. Em [13] foi proposta uma rede neuro-fuzzy adaptativa embarcada em FPGA (matriz de contatos reprogramável). Entretanto, o estudo limita-se ao escopo experimental. O desenvolvimento de microcontroladores com arquiteturas aprimoradas [21] [22] abriu novas possibilidades a sistemas inteligentes embarcados. Nesta seção apresentamos o procedimento DENFIS embarcado.

Um esquema embarcado para implementação de sistemas evolutivos DENFIS é mostrado na Fig. 2. Na figura, os periféricos dependem de requisitos de projeto. O núcleo (contendo o procedimento de modelagem DENFIS) é comum, e pode ser migrado para diferentes projetos. Define-se tempo de resposta como o tempo decorrido desde o recebimento de um sinal por um periférico de entrada até a produção de outro sinal por um periférico de saída devido ao primeiro sinal. Em geral, dados de entrada provenientes de sensores são recebidos via porta serial ou conversor analógico-digital. Dados de saída podem ser exportados via porta serial ou usados por um mecanismo PWM para modulação por largura de pulso.

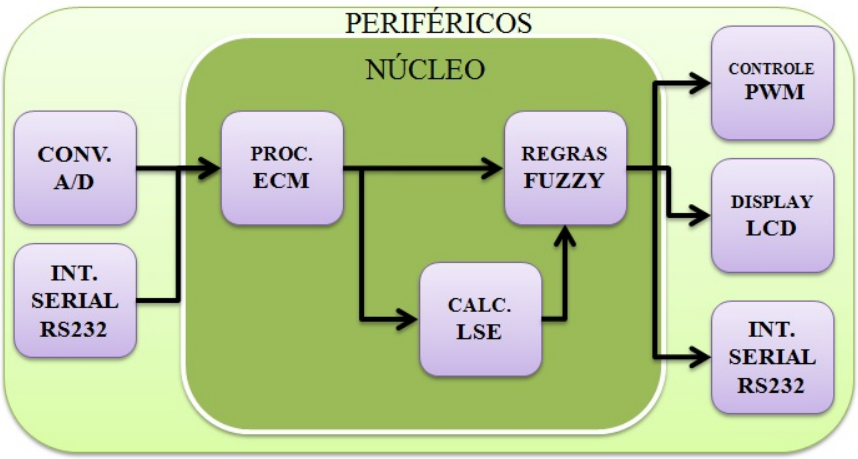

Fig. 2. Blocos funcionais de um sistema evolutivo embarcado

4.1 Clusterização Evolutiva Embarcada

Nesta seção apresentamos o procedimento ECM embarcado. Propomos a formatação a seguir para armazenamento dos clusters na memória *RAM* do microcontrolador:

Estrutura cluster:
Variável Ponto flutuante: Coordenada centro $x0$
...
Variável Ponto flutuante: Coordenada centro xd
Variável Ponto flutuante: Raio
Variável Bit: Status
Variável Inteiro: Número de amostras associadas ao cluster

(DENFIS embarcado: Formação de clusters para o método ECM)

Nesta estrutura são considerados 4 Bytes para variáveis do tipo ponto flutuante, 1 Byte para inteiros, e $0,125$ Bytes para 1 bit [20]. Considerando d como a dimensão do espaço de entrada, nesta estrutura, o espaço em *RAM* para um cluster pode ser determinado por:

$$EpRAM = 4d + 4 + 1 + 0,125$$

$$EpRAM = 4d + 5,125 \tag{4}$$

Mapeamos toda a memória *RAM* disponível por clusters. Esta estratégia possibilita avaliar o número máximo de clusters em fase de desenvolvimento. Ressaltamos que alguns endereços são reservados para configuração do dispositivo e devem ser preservados. A Tabela I apresenta o número máximo de clusters alocáveis nos microcontroladores avaliados.

Tabela I - Número máximo de clusters por dispositivo

Modelo	*RAM* (Bytes)	Clusters 3 Dim	Clusters 4 Dim	Clusters 5 Dim.
PIC18F4550	2000	110	90	76
PIC18F4680	3300	176	144	121
PIC24F128GA010	8000	303	252	217
DSPIC30F4013	2000	96	80	69
PIC32MX360F512L	32000	1305	1110	978

Apresentamos a seguir uma versão simplificada do algoritmo ECM, compatível com a estrutura apresentada nesta seção.

FUNCAO PROCECM
INICIO
Alocar clusters na memória, Raio = 0, Status = 0
FAZER //Loop eterno
Ler nova amostra nos sensores
Calcular distância entre amostra e centro dos clusters existentes
SE (nenhum cluster ativo)
Criar cluster (buscar e ativar o primeiro cluster com status = 0)
Localizar cluster ativo mais próximo n
SE (distancia $> D_{thr}$*)*
Criar cluster (buscar e ativar primeiro cluster com status = 0)
SE (distância $< D_{thr}$*) E (distância* $\leq$ *Raio)*
Incrementar número de amostras associadas ao cluster n
SE NÃO
Expandir cluster n
Verificar critério de exclusão de clusters (poda)
FIM - FAZER
FIM

(DENFIS embarcado: Algoritmo do método ECM)

A estrutura proposta nesta seção foi validada com o microcontrolador PIC24F128GA010 [22]. Experimentos apresentaram tempo de resposta de aproximadamente 1.48 microssegundos para o algoritmo de criação de um cluster. A expansão de um cluster existente leva 77.36 microssegundos para ser processada. Outras operações são processadas em 45.33 microssegundos em média.

4.2 Aprendizado no DENFIS Embarcado

Propomos a alocação do maior número possível de clusters em memória *RAM*. Entretanto, durante o treinamento, algum espaço deve estar temporariamente disponível para o método LSE (3). Após o treinamento, este espaço em memória

é reutilizado. A estratégia *"dividir e conquistar"* é proposta para reduzir a ocupação da memória *RAM* durante o treinamento. Considerando d como a dimensão do espaço de entrada, e n o número de clusters formados a partir do *dataset* de treinamento, as matrizes do LSE são preenchidas com grupos de d clusters a cada iteração do cálculo, até que todos os n clusters sejam processados. Esta varredura de todos os n clusters em sub-grupos de matrizes quadradas, torna as matrizes A, A^T e P do método LSE quadradas de dimensão $d \times d$, economizando espaço em memória *RAM*. A matriz de coeficientes B é de ordem $d \times n$, pois esta depende do número de clusters. O algoritmo proposto a seguir ilustra a estratégia descrita para viabilizar o método LSE:

PROCEDIMENTO CALCULA LSE
FAZER contador = 0 até contador = n
Busca os d clusters ativos a partir de contador
Se achados < d clusters
Busca os d clusters ativos regredindo de n
Roda LSE
Armazena conseqentes
Soma d ao contador
FIM(FAZER)
FIM

O espaço total em memória *RAM* necessário para processar o algoritmo de treinamento do DENFIS é calculado por:

$$EpRAM = 5d^2 + d + d \times n$$

Entretanto, todos os componentes de cada matriz são variáveis do tipo ponto flutuante, então o cálculo passa a ser:

$$EpRAM = 20d^2 + 4d \times (1 + n) \quad (5)$$

A Tabela II apresenta a ocupação da memória *RAM* em um microcontrolador
PIC24F128GA010 [22], recebendo a estrutura do ECM e a estrutura de treinamento com algoritmo LSE.

Tabela II - Ocupação da memória *RAM* para processamento do LSE e ECM

Ocupação em *Bytes*			
	50 clusters	75 clusters	100 clsuters
3 dimensões	2442	3342	4242
4 dimensões	2790	3790	4790
5 dimensões	3178	4278	5378

Os experimentos apresentados nesta seção enfatizam que dados de dimensões elevadas limitam a eficiência do sistema embarcado. Métodos para redução em tempo real da densidade dos dados estão nas propostas de trabalhos futuros. Após o treinamento, a memória temporária é liberada para outras etapas do algoritmo DENFIS. O tempo total para treinamento no caso do acoplamento de trens em movimento é apresentado na seção seguinte.

5 Acoplamento de Trens em Movimento

Foram utilizados os dados de [19], onde é proposto o controle dinâmico e evolutivo do acoplamento de dois trens em movimento. O acoplamento de locomotivas auxiliares visa aumento de torque para a subida de rampas. O modelo baseado em EGNN *(Evolving Granular Neural Network)* controla a pressão do cilindro de freio da locomotiva de torque auxiliar, até que a mesma se acople de forma suave à composição principal, e lhe forneça torque extra. Em suma, em [19] busca-se reproduzir as ações do controlador humano especialista no acoplamento. Ações deste operador foram amostradas em um banco de dados contendo dados de velocidade, distância entre trens, e ponto de aceleração do motor. Propomos a implementação do modelo DENFIS embarcado e simulado software, para a mesma tarefa. Resultados serão comparados com a abordagem em EGNN, e com o operador humano. Dentre os microcontroladores avaliados, capacidade de armazenamento de clusters, e precisão na leitura de sensores levaram à escolha do microcontrolador PIC24FJ128GA010, trabalhando em hardware com clock a cristal de 120Mhz. Este dispositivo possui 10 *bits* de resolução no periférico que realiza a leitura dos sensores, caso se opte pela leitura de amostras diretamente sobre eles [22]. Para avaliar o desempenho embarcado, foi desenvolvido um sistema de interface entre o microcontrolador e o banco de dados utilizado em [19], este sistema é apresentado na Fig. 4.

O algoritmo de treinamento DENFIS foi executado embarcado. Utilizou-se 312 pares entrada-saída, e o tempo total para o treinamento a partir destes dados foi de 2104,79 milissegundos. A Tabela III apresenta a ocupação dos recursos do microcontrolador programado para o estudo de caso considerado. O baixo grau de ocupação dos recursos do dispositivo enfatiza a eficiência no processo de produção das rotinas embarcadas. Por outro lado, na Tabela IV, um ganho de velocidade pode ser observado no sistema embarcado, por efeito do processamento dedicado.

Tabela III - Ocupação dos recursos da memória do microcontrolador.

Recurso	Ocupação
Memória de programa (FLASH)	26 %
Memória *RAM*	38 %

Tabela IV - Tempo de resposta para inferência DENFIS

Implementação	Tempo de resposta
Embarcada (clock 120MHz)	45.33 microssegundos
Software (dual core clock 2.3GHz)	1,41 milissegundos

Na Fig. 3 é apresentada a curva de frenagem obtida nos modelos DENFIS simulado em software e embarcado, em relação ao comportamento do operador apresentado em [19]. Pode-se observar na figura uma tendência do sistema embarcado a acompanhar ações bruscas de frenagem, apresentado baixo *overshoot* e *undershoot*. No escopo do problema, a frenagem brusca é necessáaria em função da inércia da locomotiva.

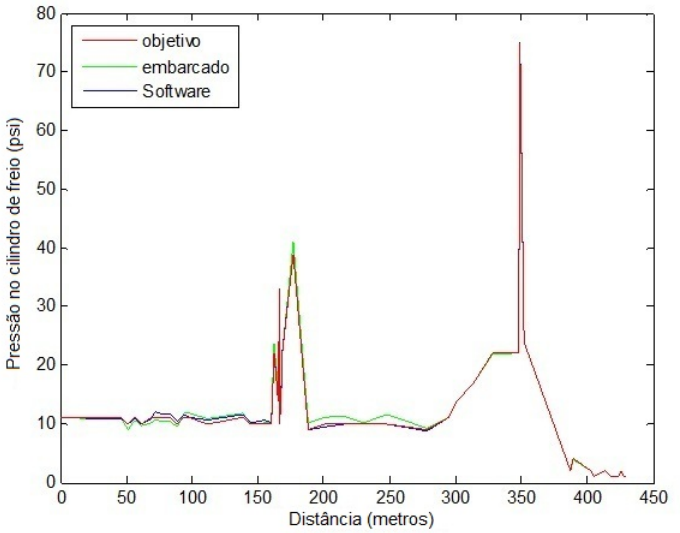

Fig. 3. Curva de pressão imposta ao cilindro de freio em relação à distancia entre os trens.

Destacamos na figura, a versatilidade de DENFIS. Abordagens embarcadas e simulações em software, mesmo com recursos de processamento diferentes, são igualmente precisas. Medimos a precisão através do erro médio quadrático (MSE): Tomando y_d como a saída desejada, e y_m como a saída obtida no modelo, o erro para n amostras é calculado por:

$$MSE = (\frac{1}{n}\Sigma_{i=1}^{n}(y_d - y_m)^2\ ) \tag{6}$$

O modelo DENFIS avaliado apresentou erro médio quadrático de 0,0564 para a versão embarcada, e 0,0399 para a versão simulada em software. Obtivemos

maior precisão do que a abordagem EGNN, na qual [19] obteve erro quadrático de 2,1. Este erro pode ser considerado pequeno em relação à características físicas do problema, como inércia da grande massa da locomotiva, imprecisão nos sensores e desgastes no sistemas de frenagem.

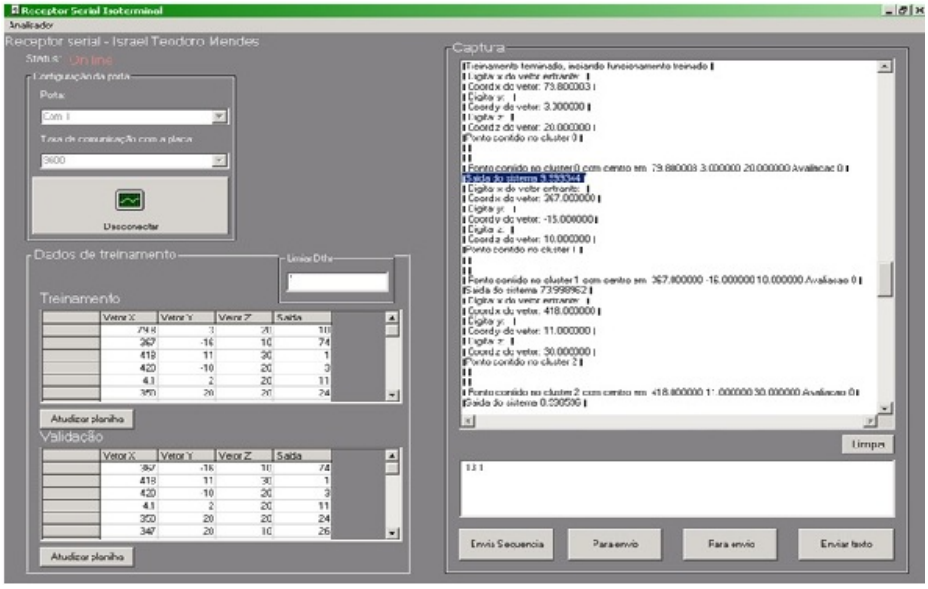

Fig. 4. DENFIS embarcado: interface de comunicação

6 Conclusão

Apresentamos neste estudo uma investigação de sistemas evolutivos embarcados, buscando modelar processos dinâmicos. Propusemos e avaliamos uma abordagem neuro-fuzzy com aprendizado incremental que possibilita modelagem de sistemas dinâmicos não lineares e não estacionários em ambientes com recursos limitados. Demonstrou-se com o estudo do acoplamento dinâmico que a abordagem neuro-fuzzy embarcada reproduz os resultados obtidos em software com boa precisão. A comparação com simulações em plataforma computacional enfatizou a velocidade do processamento embarcado dedicado. Microcontroladores são mais rápidos, uma vez que interagem diretamente com o hardware de controle da planta.

A partir dos resultados obtidos, observamos que a abordagem evolutiva DENFIS pode ser embutida em microcontrolador sem perda de desempenho e com ganhos em termos de velocidade de processamento. Em trabalhos futuros abordaremos problemas de controle e previsão de séries temporais. A avaliação do comportamento de algoritmos evolutivos em problemas com fluxo de dados de alta densidade é também uma questão que necessita de investigações futuras.

References

1. ANGELOV, P.; FILEV, D.; KASABOV, N. (Eds.) *Evolving Intelligent Systems: Methodology and Applications*. Wiley-IEEE Press Series on Computational Intelligence, 2010.
2. KASABOV, N. ECOS Evolving Connectionist Systems and the ECO learning paradigm. *Proc. INCOMP*, v. 8, p. 1232–1235, 1998.
3. KASABOV, N.; *Evolving Connectionist Systems*. Londres: Springer-Verlag & Hall/CRC, 2007.
4. LEITE, D.; COSTA, P.; GOMIDE, F. Evolving Granular Classification Neural Networks *Proc. of International Joint Conference on Neural Networks*, v. 8, p. 1–6, 2009.
5. WATTS, M. A decade of Kasabov Evolving Connectionist Systems: A review *IEEE transactions on System, Man and Cybernetics - PartC*, v. 39, p. 253–269, 2009.
6. KASABOV, N. Dynamic Evolving Neural-Fuzzy Inference Systems and Its Application for Time-Series prediction. *IEEE Transactions on Fuzzy Systems*, v. 8, p. 1736–1743, 2002.
7. KASABOV, N. Evolving Connectionist Systems: A theory and a case study on adaptative speech recognition. *IEEE Transactions on Fuzzy Systems*, v. 8, p. 3002–3007, 1999.
8. KASABOV, N. Evolving Fuzzy Neural Networks for Supervised Online Knowledge-based Learning. *IEEE transactions on System, Man and Cybernetics - PartC*, v. 8, p. 912–918, 2001.
9. KASABOV, N.; WOODFORD, B. Rule Insertion and Rule Extraction from Evolving Fuzzy neural Networks: Algorithms and Applications for Building Adptative, Intelligent Experts Systems. *IEEE International Fuzzy Systems Conference Proceedings*, v. 8, p. 1406–1411, 1999.
10. KASABOV, N. Simple Evolving Connectionist Systems and Experiments on Isolated Phoneme Recognition. *IEEE International Fuzzy Systems Conference Proceedings*, v. 8, 8p. 2000.
11. FRANCO, L; ELIZONDO, D. A.; JEREZ, J. M. (Eds.) *Constructive Neural Networks*. Springer-Verlag Berlin Heidelberg (Studies in Computational Intelligence), 2009.
12. CARPENTER, G. A.; GROSSBERG, S. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, v. 37, p. 54–115, 1987.
13. DEL CAMPO, I. J.; ECHANOBE, I. J.; BOSQUE,G.; TARELA, M. Efficient Hardware Software Implementation of an adaptative Neuro-Fuzzy Inference System. *IEEE International Fuzzy Systems Conference Procedings*, v. 8, p. 761–777, 2008.
14. DE BARROS, J. C.;DEXTER, A. L. An Evolving Fuzzy Model for Embedded Aplications. *International Symposium on Evolving Fuzzy Systems*, v. 8, p. 49–53, 2000.
15. RAVI, V.; SRINAVAS, E. R.; KASABOV, N. On line-Evolving Clustering. *International Conference on Computational Intelligence and Multimedia Applications*, v. 8, p. 347–351, 2007.
16. JANG, S. S. R; SUN, C. T.; MIZUTANI,E. *N*euro Fuzzy and Soft Computing, Prentice Hall. USA : Springer-Verlag & Hall/CRC, 2007.
17. COSTA, A.; GLORIA, A.; FARABOSCHI,P.; PAGNI, A.;RIZZOTO,G; Hardware Solutions for Fuzzy Control *Proceding of IEEE*, v. 8, p. 422–434, 1995.

18. JEREMY, B.; WILAMOWSKI, B. M.; FARABOSCHI, P.; PAGNI, A.; RIZZOTO,G. Hardware Solutions for Fuzzy Control *Proceding of IEEE*, v. 8, p. 422–434, 1995.
19. BERGO, L. Uma proposta de controle para acoplamento automático de tração ferroviária auxiliar a veículos ferroviários em movimento. *Dissertação de mestrado*, Programa de Pós Grad. Em Eng. Elétrica, Pontifícia Universidade Católica de Minas Gerais, 2011.
20. CCS Inc; *C Compiler Reference Menual. USA : CCS Inc. & Hall/CRC, 2009.*
21. MICROCHIP Inc; *DSPIC33Family Datasheet Hight Performance 16 bit Digital Signal controllers. USA : Microchip Inc. & Hall/CRC, 2007.*
22. MICROCHIP Inc; *PIC24FJ128GA010 Family Data Sheet 64/80/100-Pin General Purpose, 16-Bit Flash microcontrollers. USA : Microchip Inc. & Hall/CRC, 2009.*