Simulação de Sistemas de Estoque e Metamodelagem através de Redes Neurais Artificiais

Adelmo Luis Cechin
UNISINOS,
Programa Interdisciplinar de Pós-Graduação em Computação Aplicada,
Av.Unisinos, 950, São Leopoldo, Brasil
acechin@unisinos.br

Resumo: Este artigo apresenta um novo algoritmo para adquirir um projeto experimental ótimo, quando Redes Neurais Artificiais são treinadas para aprender metamodelos. Métodos para a computação de um projeto experimental apropriado ou para a determinação de uma região adequada para medir os dados são importantes se os custos de aquisição dos dados são mais importantes que a computação do metamodelo. Esta computação tipicamente impõe um certo custo, tais como computação ou tempo de processamento humano (custo de mensuração). Como este é o caso para a maior parte das aplicações, um bom projeto experimental possui um impacto profundo sobre o metamodelo obtido. Resultados são mostrados para um estudo de caso envolvendo um sistema de armazenamento e uma comparação do algoritmo proposto com os dados aleatórios padronizados é apresentada. Além das simulações, a metodologia é também útil para aplicações usando um projeto multi-estágio e aquisição de dados.

Palavras-chaves: Projeto Experimental, Redes Neurais Artificias, Sistemas de Armazenamento.

1 Introdução

A possibilidade de simular um sistema complexo demais para ser calculado diretamente através de equações simples apresenta uma série de vantagens: economia para o projetista do sistema, tanto em termos de tempo quanto em termos de recursos, avaliação dos principais fatores que afetam o desempenho do sistema e previsão de situações inesperadas. A simulação permite também o ajuste do assim chamado *gap semântico* (diferença entre o sistema real e o modelo de simulação) provendo, mesmo a operadores experientes, indicações sobre quais partes do sistema são mais importantes tendo influência global sobre os resultados e quais possuem uma influência mais localizada. Há também situações de risco de vida onde outras soluções não são aplicáveis. Contudo, os dois maiores problemas do uso de simulações são (a) o tempo necessário para executar uma simulação e (b) a

interpretação dos resultados da mesma em termos dos fatores e configuração do sistema.

Na prática, a redução do tempo de simulação é obtida através do aumento do *gap semântico*, simplificando o modelo, reduzindo o número de variáveis consideradas ou diminuindo o tempo de simulação. Como desvantagem, estas simplificações prejudicam a exatidão dos resultados obtidos. Além disso, muitas vezes não é possível uma simplificação do modelo.

Uma vez que um modelo representa um sistema real, deseja-se ainda que o mesmo possa ser utilizado como sub-modelo de um modelo maior. Por exemplo, o modelo de um sistema de estoque pode ser usado como parte de um sistema de logística, um modelo de uma máquina pode ser usado como parte de um modelo de uma indústria, etc. Se a simulação destes sub-modelos não puder ser realizada em tempo hábil, a simulação do sistema que o inclui ficará limitada. Assim, faz-se necessário o desenvolvimento de um *metamodelo*. Um metamodelo representa o processo de simulação de um modelo. O objetivo do metamodelo é aproximar-se, tanto quanto possível, da resposta da simulação do modelo. A grande vantagem do uso do metamodelo reside no tempo reduzido de processamento e, portanto, permite a sua utilização em modelos mais complexos e simulações incluindo vários componentes.

Redes Neurais Artificiais, por serem não-lineares, livres e adaptáveis [1] podem ser usadas como metamodelos. Do ponto de vista teórico, uma RNA é um aproximador universal multidimensional [2]. Isto possibilita estimativas de valores da saída do metamodelo para diversas configurações de parâmetros, bem como o estudo do comportamento global do sistema simulado em função de seus parâmetros. Por exemplo, RNAs têm sido usadas como metamodelos em sistemas de armazenamento [3], sistemas de emergência em hospitais [4], sistemas de predição de depósitos de carvão [5], na análise econômica de projetos de risco [6], simulação de sistemas fabris estruturados em *job-shop* [7], na computação de intervalos da confiança em simulações discretas [8][9][10], etc..

Um estudo de RNAs tipicamente inclui uma fase de aquisição de dados, uma fase de treinamento da RNA e uma fase de testes do desempenho da RNA. Esta última usa uma base de dados diferente da base de treinamento. Outras metodologias de teste de RNAs incluem validação cruzada e *leave-one-out*. Estas metodologias são usadas quando a fase de aquisição de dados não está sob o controle do especialista ou quando a aquisição dos dados é realizada em apenas uma fase independente (tipicamente anterior) do cálculo da RNA. Neste caso, como não é possível realizar novas medidas, há um esforço na obtenção da melhor RNA possível para os dados disponíveis. O foco do estudo passa a ser a RNA e não mais os dados, como se o custo de obtenção de novos dados fosse infinito. Certamente, a aquisição de um dado tem um custo ou humano ou computacional e não há disponibilidade irrestrita dos mesmos para certa aplicação. Contudo, os dados adquiridos por simulação, apesar de caros computacionalmente, podem ser obtidos e controlados *on-line*. Se o metamodelo gerado apresenta um erro insatisfatório, novos dados podem ser gerados. Esta diferença altera completamente a etapa de aquisição de dados e teste das RNAs, tornando o processo iterativo entre estas duas etapas. A estratégia usada para controlar este processo, a geração de dados, o cálculo da resposta da simulação, influencia o desempenho e depende do metamodelo utilizado. Assim, neste artigo propomos uma estratégia de geração de um projeto experimental e a partir do mesmo,

a obtenção do metamodelo neural. Regiões do espaço de entrada do metamodelo, onde este apresenta um erro inaceitável, são investigadas com um detalhe maior.

Este artigo está dividido nas seguintes seções: na seção 2 apresentamos simplificadamente o metamodelo neural, na seção 3 apresentamos a aplicação: um sistema de controle de estoque, na seção 4 apresentamos a metodologia na forma de um algoritmo e na seção 5 apresentamos resultados comparativos entre o algoritmo proposto e a geração uniforme dos dados dentro do intervalo válido para os parâmetros da simulação. Finalmente, na seção 6 apresentamos as conclusões.

2 Redes Neurais Artificiais

RNAs são usadas como metamodelos não-lineares multidimensionais. A RNA discutida neste artigo pertence à classe Multilayer Perceptron (MLP) [1]. As funções base são funções sigmoidais (logística ou tangente hiperbólica), dadas por $a_j = sig(s_j) = \frac{1}{1+e^{-s_j}}$. Nesta equação, s_j representa a soma ponderada das entradas do neurônio j e a_j representa a saída do neurônio j. Nesta equação, $s_j = \sum_{i=1}^{n} w_{ji} a_i + b_j$, onde w_{ji} representa o peso sináptico do neurônio i para o neurônio j, a_i representa a saída do neurônio i, b_j o bias do neurônio j e n é o número de neurônios antecessores conectados ao neurônio j. A ativação dos neurônios $a^{(k)}$ em uma camada k pode ser representada na forma matricial e calculada através da seguinte equação: $a^{(k)} = sig\left(W^{(k,k-1)} a^{(k-1)} + b^{(k)}\right)$, onde $W^{(k,k-1)}$ é a matriz dos pesos da camada *k-1* para camada *k*, $a^{(k)}$ é a ativação dos neurônios na camada k e $b^{(k)}$ é o bias da camada *k*. Assim, para uma RNA com 3 camadas temos: uma camada de entrada ($a^{(1)}$), uma camada intermediária ($a^{(2)}$) e uma camada de saída ($a^{(3)}$). A RNA é calculada primeiro avaliando a equação anterior para $k=2$ e então para $k=3$. Por definição, o valor de $a^{(1)}=x$, onde x corresponde aos parâmetros de entrada do metamodelo e $y=a^{(3)}$, onde y são as variáveis de saída do metamodelo. Por exemplo, a figura a seguir mostra uma RNA 4-16-3.

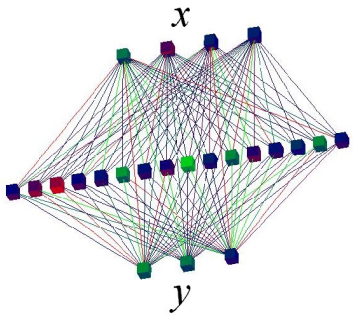

Fig. 1. RNA com 4 neurônios na camada de entrada, 16 na intermediária e 3 neurônios na camada de saída.

Algumas RNAs não usam uma função sigmoidal na saída e outras possuem pesos ligando as entradas diretamente às saídas (chamados *short-cut connections*). RNAs utilizando *short-cut connections* são denotadas por colchetes (4-[16]-3) e RNAs não utilizando a função sigmoidal na saída são denotadas pelo símbolo + (4-16+3).

3 Aplicação: Sistemas de Controle de Estoque

Sistemas de estoque [11] são importantes para regular o fluxo de produtos ao longo de uma cadeia produtiva. Isto ocorre devido a incertezas no fornecimento destes produtos e à economia de escala. Esta mostra que os menores gastos na cadeia são obtidos quando é possível colocar à disposição exatamente um item à disposição no momento e lugar de seu consumo. Devido às condições de produção e consumo tal ideal não pode ser atingido. Um sistema de estoque permite uma aproximação deste ponto ótimo. Contudo, cada cadeia produtiva possui suas próprias características e, portanto, projetar um estoque de forma ideal adequado a ela depende de vários fatores (custo de manutenção do estoque, custo de perda de produtos perecíveis, custo de transporte para o estoque, etc.).

Tipicamente, sistemas de estoque possuem dois parâmetros fundamentais que controlam a entrada e saída de produtos: o número mínimo de produtos no estoque (s) e o tamanho do estoque (S) ou número máximo de produtos no estoque. O número mínimo determina um limite, que quando atingido, dispara um pedido de compras do produto. A quantidade típica em um pedido de compras é S - *número de produtos ainda no estoque*. A partir deste pedido, decorre um tempo até que o pedido seja atendido, durante o qual o estoque deveria continuar suprindo o consumo. Contudo, pode ocorrer que durante este tempo o consumo seja maior que a capacidade do estoque. Neste caso, ocorre um custo adicional ao sistema pela perda de clientes e não atendimento da demanda. Assim, custos incidem sobre cada etapa do processo, sendo estes divididos em: (a) custo do pedido, igual a um custo fixo do pedido mais o preço do produto comprado (32,0R$ + *quantidade* × 3,0R$), (b) custo de manutenção do estoque, proporcional ao número de itens no estoque e ao tempo que os itens ficam armazenados no mesmo (1,0R$ × *quantidade* × *período de armazenamento*) e (c)

custo de perda de clientes por falta de produtos fornecidos (5,0R$ × *quantidade não suprida* × *período em falta do produto*). Todos estes custos são somados e resultam em um custo total.

Para permitir uma visualização das relações existentes entre estas variáveis, a figura a seguir apresenta o custo total do processo, em função do tamanho do estoque *S* e da diferença *d*=*S*-*s* (*s* = limite para efetuar novo pedido de compra). A diferença *d* pode ser interpretada como o número máximo de produtos consumidos a partir do qual um novo pedido de compra deve ser disparado, ou seja, se o estoque está normalmente cheio (*d* pequeno) ou não (*d* grande). Valores baixos de *d* indicam estoques cheios e muitos pedidos feitos em tempos curtos. A partir desta figura, nota-se que em geral, para estoques grandes (*S* grande), outros fatores não influenciam muito no custo total. Isto explica a tendência geral dos administradores em insistir nos grandes estoques como uma boa estratégia, caso não haja um bom controle do consumo dos produtos ou na falta de estudos que permitam obter boas previsões. Contudo, o custo mínimo é obtido para estoques pequenos (sistemas que se aproximam do ideal *just-in-time*). Porém, estoques pequenos exigem um bom controle da demanda, podendo ter seu custo rapidamente incrementado para valores pequenos de *d*. Para estoques pequenos, manter o estoque cheio (*d* pequeno) não é uma boa estratégia, apresentando custos elevados.

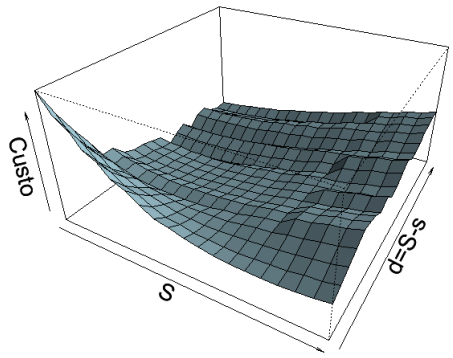

Fig. 2. Custo total do sistema de estoque para diferentes configurações de gerenciamento do estoque e simulação de 120 meses de funcionamento do estoque. *S*= tamanho do estoque, *d* = diferença entre tamanho do estoque e limite mínimo para disparo de um novo pedido.

O custo relativamente uniforme para estoques grandes e um incremento rápido para estoques pequenos mostra um comportamento diferenciado dos custos, dependendo da combinação dos parâmetros *S* e *s* e indicam o uso de metamodelos que possuam tais características em sua função base (funções usadas pelo metamodelo como base para compor a função do metamodelo; no caso das RNAs, funções sigmoidais). Estas características não são particulares do sistema de estoque, mas aparecem tipicamente em tais aplicações. RNAs, pelo uso de funções sigmoidais, apresentam tais

propriedades e, portanto, são adequadas como metamodelos para aplicações como *job-shops*, sistemas de filas, etc..

4 Metodologia de Aquisição de Metamodelos

A seguir será apresentado o algoritmo para definição do projeto experimental usando metamodelos neurais.

Para calcular um metamodelo, dados são inicialmente necessários. Portando, inicialmente (fase **Inicialização** no algoritmo abaixo) define-se um projeto experimental a ser aplicado. Por exemplo, gera-se um projeto fatorial completo (todas combinações dos valores das variáveis), definindo as condições sob as quais a simulação será realizada. Um conjunto de teste é gerado e erros são calculados. A geração do conjunto de teste segue uma distribuição de probabilidade visando gerar pontos em toda a região de interesse.

Em uma etapa seguinte (fase **Iteração** no algoritmo abaixo), dados no conjunto de teste em regiões onde a RNA não obteve um bom desempenho são incluídos no conjunto de treinamento e retirados do conjunto de teste. Para manter o tamanho do conjunto de teste, novos dados de teste são gerados e incluídos neste conjunto. O critério para transferência de dados entre treinamento e teste é a média do erro. Caso o erro de um dado seja maior que a média, ele é transferido para o treinamento.

```
Inicialização:
 (R, T) ← definição inicial dos dados de treinamento e de teste
 treinamento da RNA com os dados em R e teste com T
 cálculo do erro da RNA aos dados de teste T
Iteração:
 for certo número de iterações or
 while o erro da RNA < erro limite
       m ← média do erro dos dados em T
       for todos os dados d em T
              if (erro do dado d) > m
                     transfere o dado d de R para T
                     d' ← gera um novo dado
                     r' ← resposta do sistema para d'
                            (por simulação)
                     inclui (d',r') em T
       treinamento da RNA com os dados em R e  teste com T
       cálculo do erro da RNA aos dados de teste T
 return RNA, R, T, erro
```

5 Resultados e Discussão

Como ilustração do método acima, a figura abaixo apresenta os resultados quando aplicada à função seno entre 0 e 2π com ruído uniforme de 10%, utilizando uma RNA com estrutura 1-[2]+1. Observa-se que há um limite no desempenho da RNA de

aproximadamente 3%, sendo este rapidamente atingido após terem sido acrescentados 15 dados ao conjunto de treinamento. Acréscimos posteriores no número de dados de treinamento não trazem ganhos de exatidão.

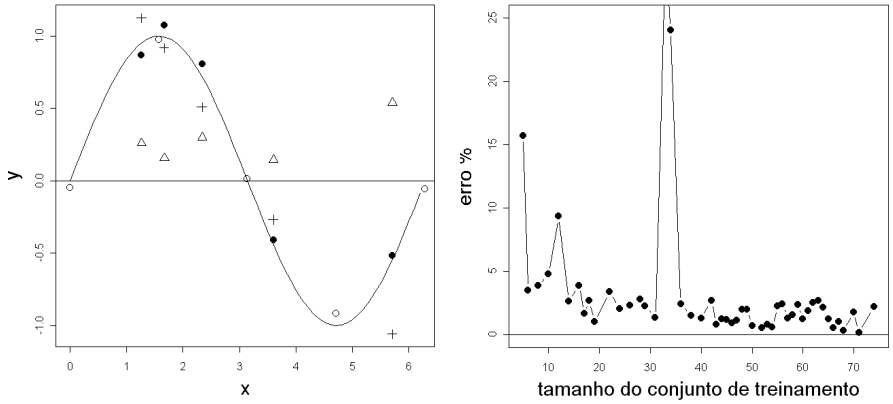

Fig. 3. Aplicação do método à função seno. Gráfico apresentando a função seno, onde círculos vazios são os dados de treinamento, círculos cheios são dados de teste, símbolo + é a resposta da RNA e triângulos representam o erro da RNA (esquerda). O erro RMS é calculado e relacionado com o "range" do seno, sendo expresso em percentual, em função do tamanho do conjunto de treinamento (direita). Experimentos realizados com $|T|$=5, ou seja, tamanho do conjunto de teste é fixo em 5.

Para a aplicação do sistema de estoque, a figura a seguir mostra o tamanho do conjunto de treinamento e a distribuição final dos pontos calculada pelo algoritmo. Para esta aplicação, uma RNA com estrutura 2-[10]+1 foi utilizada. Valores (S,d) são utilizados como entrada e o custo total do estoque são calculados para 120 meses de funcionamento do estoque. O erro é apresentado no gráfico esquerdo e a distribuição dos pontos no gráfico direito. Observa-se um decaimento do erro da base de teste com um aumento do número de dados no conjunto de treinamento. Observa-se também o comportamento típico de estabilização do erro quando o número de dados é grande o suficiente. Neste ponto, observa-se que a distribuição final dos pontos dentro do espaço de entrada dos parâmetros da simulação termina praticamente uniforme.

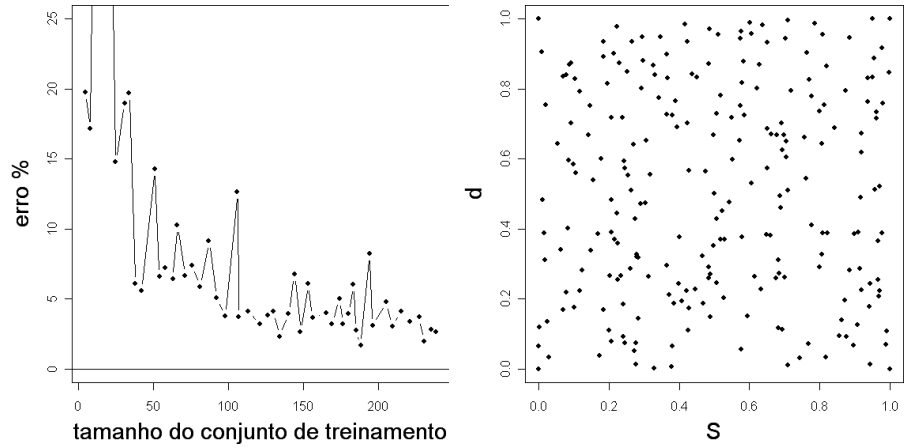

Fig. 4. Aplicação do método ao sistema de estoque. O erro da RNA ao conjunto de teste permanece dentro da faixa de 0 a 5% (esquerdo). A distribuição final dos pontos de treinamento é basicamente uniforme (direito). Experimentos realizados com $|T|=20$.

Como comparação do algoritmo proposto com um algoritmo de geração aleatória dos dados, a figura a seguir apresenta o erro da RNA para ambos os algoritmos sob o mesmo número de dados na base de treinamento. O gráfico da esquerda mostra que ambos os métodos produzem metamodelos com erros similares quando o número de dados é suficientemente grande. Contudo, quando o número de dados é limitado (menos de 40 pontos para o metamodelo do estoque), o algoritmo proposto gera dados mais representativos, resultando em um erro menor (linha contínua).

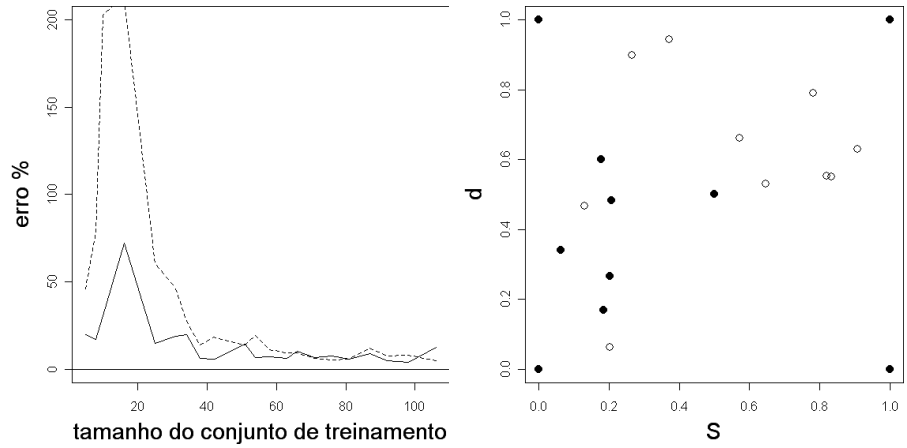

Fig. 5. Comparação entre o algoritmo proposto (linha contínua) e geração aleatória de pontos (linha tracejada) (esquerda). Dados gerados por ambos os algoritmos quando o número de dados gerados é 10 ($|R|=10$). Pontos cheios representam os dados gerados pelo algoritmo proposto e pontos vazios, os dados gerados de forma aleatória (direita).

Observa-se no gráfico dos pontos gerados (gráfico direito) que a distribuição destes para cada um dos dois algoritmos é diferente. O algoritmo proposto gera dados (círculos cheios) próximos ao quadrante (S=0,d=0) enquanto o gerador aleatório gera dados (círculos vazios) próximo ao quadrante (S=1,d=1). Certamente, o algoritmo aleatório poderia ter escolhido (por chance) o mesmo quadrante que o algoritmo proposto, contudo, este não possui a informação do erro da RNA para guiá-lo.

O erro quadrático da RNA é apresentado na Fig. 6 para três instantes de tempo do algoritmo, para $|R|$=5 (inicialmente), para $|R|$=10 e para $|R|$=25. Observa-se que, inicialmente, a região onde os erros são mais significativos é a região próxima ao quadrante (S=0,d=0) e que o erro esta está praticamente concentrado nesta região (veja Fig. 5 (direita)). À medida que o algoritmo progride, o erro global cai (veja Fig. 5, gráfico esquerdo) e a distribuição do erro torna-se mais uniforme (veja Fig. 6).

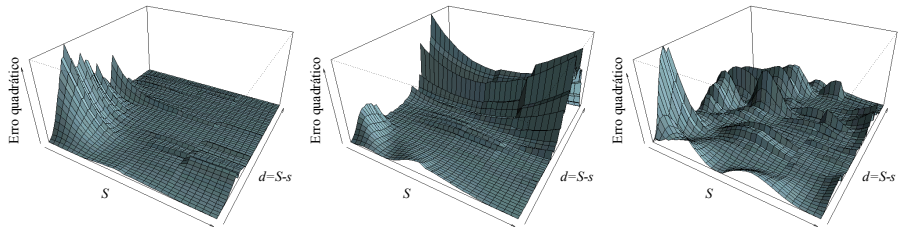

Fig. 6. Erro quadrático para a RNA para $|R|$=5 (esquerda), $|R|$=10 (centro) e $|R|$=25, em função das variáveis S e d=S-s.

6 Conclusão

Apresentamos um algoritmo para geração de dados em regiões de erro elevado, para obtenção de metamodelos neurais. Quando o número de dados disponíveis é elevado e representativo, certamente a obtenção de um bom modelo é uma tarefa simples. Contudo, quando, cada medida a ser realizada possui um custo computacional ou financeiro, a obtenção de um bom modelo utilizando um número mínimo de dados torna-se uma tarefa importante.

O método proposto apresentou melhores resultados que a simples escolha aleatória das medidas a serem realizadas e permitiu, com menos rodadas, reduzir o erro do metamodelo neural. Certamente, cada rodada exige um novo treinamento da RNA. Contudo, a aquisição dos dados é o fator crítico e, portanto, a redução do número de simulações ou experimentos é essencial.

Quando dados são medidos, tipicamente o técnico ou cientista possui seus critérios para escolher esta ou aquela variável, para escolher a faixa de valores medidos e para escolher a granularidade das medidas. A grande maioria das aplicações de RNAs a problemas práticos é aplicada sobre a base de dados medida. Contudo, observa-se que os critérios utilizados nas medições não levaram em conta o metamodelo. Idealmente, dever-se-iam realizar outros experimentos baseados na experiência ganha após uma primeira fase. A simulação permite, apesar de também estar associada a um custo

(mais computacional e menos humano), a realização de uma segunda, terceira, etc., fases.

Por fim, este trabalho representa um estagio inicial de um estudo mais profundo sobre as propriedades da metodologia proposta. O uso de RNAs como metamodelos é relativamente recente. A geração dos dados adequados dentro deste contexto é mais recente ainda. Há uma enormidade de possibilidades em termos de estratégias de geração do projeto experimental que podem ser mais adequadas que a versão apresentada aqui. Em uma situação ideal, o erro da RNA deveria condicionar uma distribuição de probabilidade a ser usada na geração. Além da estratégia em si, o efeito do número de neurônios na camada escondida, o tamanho do conjunto de teste e comparações com outros metamodelos, como os metamodelos polinomiais devem ser explorados.

Referências

1. Haykin, S., *Neural Networks, A Comprehensive Foundation*, Prentice Hall (1989).
2. Hornik, K., Stinchcombe, M., White, H., Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366 (1989).
3. Kilmer, R., Smith, A., Using artificial neural networks to approximate a discrete event stochastic simulation model (1993).
4. Robert, L. C., An emergency department simulation and a neural network metamodel. Journal of the *Society for Health Systems*, Special Issue on Simulation (1995).
5. Hurrion, R.D., Using a neural network to enhance the decision making quality of a visual interactive simulation model. *Journal of Operations Research Society*, 4(43):333–341 (1992).
6. Badiru, A., Sieger, D., Neural network as a simulation metamodel in economic analysis of risky projects. *Technical report, Department of Industrial Engineering*, University of Oklahoma (1993).
7. Cechin, A.L., Stertz, K., Comparando a Performance de Redes Neurais Artificiais como Metamodelos de Simulação. Em *Anais do XXIII Encontro Nacional de Engenharia de Produção*, Ouro Preto, v. 1, p. 1-9 (2003).
8. Shuman, L.J., Kilmer, R.A., Smith, A.E., Computing confidence intervals for stochastic simulation using neural network metamodelos. submitted to Computers and Industrial Engineering, *Special Issue on Computational Intelligence* (1998).
9. Pierreval, H., Huntsinger, R., An investigation on neural network capabilities as simulation metamodels. In *Proceedings of the 1992 Summer Computer Simulation Conference*, pages 413–417 (1992).
10. Fishwick, P., Neural network models in simulation: a comparison with traditional modeling approaches. In *Proceedings of the 1989 Winter Simulation Conference*, pages 702–710 (1989).
11. Law, A. M., Kelton, W. D., *Simulation Modeling and Analysis*. McGraw-Hill, Inc., New York, Lisbon, Singapore, Caracas (1991).