Preparativos para o Novo Acordo da Basiléia
Parte 5: Redes Neurais Artificiais

Jeffrey S. Morrison

Preparing for Basel II
Part 5: Artificial Neural Networks

This article continues the series dealing with modeling requirements for Basel II. Previous articles have focused on the problems of missing data, model-building strategies, special challenges in model validation, and time to default. This article looks at a completely different approach to modeling PD (probability of default) and LGD (loss given default) — artificial neural networks.

Este artigo dá continuidade à série que trata dos requisitos de modelagem do Novo Acordo da Basiléia. Os artigos anteriores concentraram-se nos problemas da falta de dados, das estratégias de modelagem, de desafios específicos da validação de modelos e do tempo que precede a inadimplência. Este artigo analisa uma abordagem totalmente diferente da modelagem de PI (probabilidade de inadimplência) e PCI (perda em caso de inadimplência) — as redes neurais artificiais.

When credit and repayment relationships are extraordinary complex, it becomes far more difficult to arrive at the probability of default and loss given default. Previous articles in this series offered a variety of strategies, such as transforming the predictor variables, discretizing and binning the data, constructing dummy variables, including interaction terms, and developing segmentation schemes using CART.[1]

However, it is possible that even these procedures do not go far enough. Developing interaction variables, for example, usually

Quando os relacionamentos de crédito e pagamento do principal são excepcionalmente complexos, torna-se muito mais difícil chegar à probabilidade de inadimplência e à perda em caso de inadimplência. Os artigos anteriores desta série ofereceram diversas estratégias, como transformação das variáveis preditivas, descrédito e agrupamento dos dados, construção de variáveis *dummy*, inclusão dos termos de interação e desenvolvimento de esquemas de segmentação por meio do CART (Classification and Regression Trees — Árvores de Classificação e Regressão).[1]

Mas é possível que até esses procedimentos não sejam suficientes. O desenvolvimento de variáveis de interação, por exemplo, costuma exigir conhecimento prévio para que a especificação seja útil. Técnicas como a regressão logística podem revelar-se menos precisas do que métodos concebidos para lidar com dados não-lineares

As redes neurais artificiais, freqüentemente chamadas simplesmente de redes neurais, usam uma abordagem matemática para representar o processamento da informação de maneira semelhante à empregada pelo cérebro humano.[2] Criadas originalmente para reconhecer e classificar padrões, as RNAs também podem ser usadas em aplicativos de agrupamento e previsão.

A terminologia associada ao desenho de RNAs pode ser tão diferente daquilo a que estamos habituados que até construtores profissionais de modelos se perdem na nomenclatura. Em um estudo[3] de 1997, J. Stuart McMenamin procurou superar o abismo entre os métodos econométricos tradicionais, como a análise de regressão, e os conceitos que cercam as redes neurais. Ele explicou que as redes neurais são, essencialmente, modelos não-lineares capazes de aproximar uma ampla variedade de processos geradores de dados. Além disso, no núcleo da maioria das RNAs está um procedimento recomendado para modelagem da probabilidade de inadimplência — a função logística.

Assim sendo, vamos definir alguns termos das RNAs por meio de comparações com os termos que usamos anteriormente nesta série:

Na terminologia das regressões, estimamos ponderações ou coeficientes. Na terminologia

requires prior knowledge for the specification to be useful. Techniques such as logistic regression may prove less accurate than methods designed to handle nonlinear data.

An artificial neural network (ANN), often just referred to as a neural net, uses a mathematical approach to represent the processing of information in a fashion similar to the human brain.[2] Originally designed for pattern recognition and classification, ANN also can be used for clustering and prediction applications.

The terminology associated with designing ANNs can be so different from anything we're used to that even professional model-building practitioners can get lost in the nomenclature. In a 1997 paper [3], J. Stuart McMenamin attempts to bridge the gap between traditional econometric methods, such as regression analysis, and the concepts surrounding neural networks. He explains that neural networks are essentially nonlinear models that can approximate a wide variety of data-generating processes. Furthermore, at the heart of most ANNs is a recommended procedure for modeling probability of default — the logistic function.

So let's define some ANN terms by comparing them with terms used earlier in this series:

In regression terminology, we estimate parameter weights or coefficients. In ANN terminology, these are called connection strengths.

In regression, the parameter estimate for a particular variable is called its slope coe-

fficient. In neural network terms, we call it the tilt parameter.

In regression, we have a term for constants — the y-intercept. In neural network terminology, these are called bias parameters.

The objective for both processes is to find a set of parameter weights (or connection strengths) that will make the errors as small as possible. In regression, this process tends to be rather direct. In neural networks, however, the solution can be much iterative and sometimes quite a bit slower.

Back-propagation. *Although a variety of parameter estimation techniques abound in the industry, back-propagation may be most popular. Basically, this is a repetitive process of using the estimated parameters to compute the forecasting errors where an adjustment is made so the next pass of errors is smaller. Back-propagation is an example of supervised learning because it requires knowledge of the "correct answers" as it learns the data and finds acceptable parameter estimates.*

The Neural Network Structure

Figure 1 shows the most basic type of structure you will find in neural network software packages. This is called a feed-forward architecture. In this type of network, information is passed from one layer to the next in a one-directional fashion. No information is channeled laterally or backwards in the network. At the bottom of Figure 1, we see the process begins with the predictive variables

das RNAs, referimo-nos a isso como forças de ligação.

Em uma regressão, a estimativa parametral de uma variável específica qualquer é chamada de coeficiente de inclinação. Na linguagem das redes neurais, ela é chamada de *tilt*.

Em uma regressão, há um termo para as constantes — o intercepto y. Nas redes neurais, esses parâmetros são chamados de viés.

O objetivo dos dois processos é identificar um conjunto de ponderações parametrais (ou forças de ligação) que leve aos menores erros possíveis. Esse processo, em uma regressão, tende a ser bastante direto. Nas redes neurais, contudo, a solução pode ser iterada e, em alguns casos, bem mais lenta.

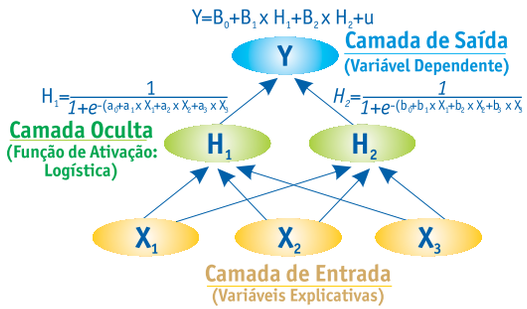
Retropropagação. Embora diversas técnicas de estimativa parametral abundem no mercado, a retropropagação parece ser a mais popular delas. Basicamente, trata-se de um processo repetitivo de uso dos parâmetros estimados para calcular os erros de previsão, fazendo ajustes para que os erros da passagem seguinte sejam menores. A retropropagação é um exemplo de aprendizado supervisionado porque exige conhecimento das "respostas certas" à medida que toma conhecimento dos dados e identifica estimativas parametrais aceitáveis.

A Estrutura da Rede Neural

A Figura 1 mostra o tipo mais simples de estrutura encontrado em pacotes de *software* de redes neurais. É a chamada arquitetura de alimentação adiante (*feed-forward architecture*). Em redes desse tipo, a informação é passada unidirecionalmente de uma camada para a seguinte.

Figura 1

Estrutura de uma Rede Neural Comum



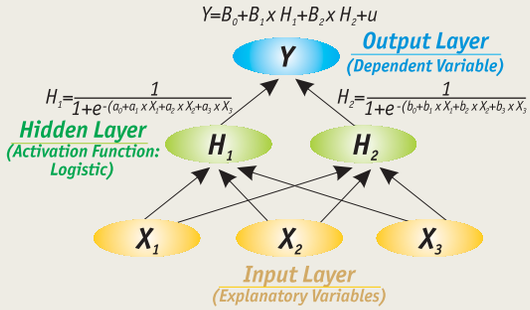
Não há canalização lateral ou retrógrada de informação na rede. Na parte inferior da Figura 1, vemos que o processo se inicia com as variáveis preditivas da camada de entrada (*input-layer*). Essas variáveis seriam dados como a relação empréstimo/valor, tipo de empréstimo, *score* de crédito etc.

Em seguida, vemos que a informação é passada para a camada oculta (*hidden layer*), onde ficam os principais mecanismos da rede neural — as funções de ativação. É aqui que residem as funções logísticas ou outros mecanismos não-lineares. No caso de uma função logística, o valor é limitado entre 0 e 1. sob algumas condições, um dos nós assumirá valor próximo a 1, simulando uma condição de disparo ou ativação num neurônio humano. Sob condições diferentes, a condição de disparo ou ativação não se verifica e o nó assume um valor mais próximo de 0.

Quanto maior for o número de nós na camada oculta, mais complexa será a rede. Costuma caber ao usuário decidir quantos nós serão incluídos na camada oculta — ou seja, a complexidade de que a rede precisará para produzir a melhor previsão possível. Normalmente, os melhores resultados se verificam entre dois e cinco nós. Se houver nós demais nessa camada,

Figure 1

Common Neural Network Structure



at the input layer. These variables would be data such as LTV, loan type, debt to income, credit score, and so forth.

Next, we see this information is passed to the hidden layer containing the primary workhouses of the neural network–the activation functions. This is where the logistic functions or other nonlinear mechanisms reside. For a logistic function, the value is bounded between 0 and 1. Under some conditions, a node will take on a value close to 1, implying a firing or activation condition in a human neuron. Under other conditions, the firing or activation condition is not implied, as the node takes on a value closer to 0.

The more nodes you specify in the hidden layer, the more complex the network. It's typically up to the user to decide how many nodes to include in the hidden layer–that is, how much complexity the network will need to produce the best forecast possible. Usually, two to five nodes in the hidden layer will provide the best results. If too many nodes are included in this layer, the network can "overfit" the data, making it look predictive on training data but result in poor perfor-

mance when new information is presented.

Finally, we see the process ends at the output layer where the information from the previous layer is put together often in a linear fashion — not unlike what we might see in a regression equation.

In addition to the terminology already presented, other terms may be encountered when dealing with neural networks. The term epoch represents a network's pass through the data. Many neural networks are designed to make numerous passes through the data to find the most appropriate parameter values. This is called training the network. Each time the data is presented, the information may be recorded or randomized. The learning rate of a neural net reflects the step size when the weights are adjusted, as might be the case in back propagation. Often it is up to the user to provide input into these parameters — parameters that can dramatically affect the accuracy of the network.

Neural nets process the information in a fashion similar to the human brain.

As redes neurais processam a informação de forma análoga ao cérebro humano.

Essential Key Features

Neural network software comes in a variety of flavors, each with its own unique user interface to make things simple while providing the flexibility needed to build good

a rede pode "exagerar" no encaixe dos dados, fazendo com que pareça preditiva com os dados de treinamento, mas resultando em baixo desempenho quando forem apresentadas novas informações.

Finalmente, vemos que o processo se conclui na camada de saída, onde as informações da camada anterior são reunidas, freqüentemente de forma linear — parecida com a que poderíamos ver numa equação de regressão.

Além da terminologia apresentada acima, podem ser encontrados outros termos ao lidar com redes neurais. O termo época representa uma passagem da rede pelos dados. Muitas redes neurais são concebidas para fazer diversas passagens até que se identifiquem os valores parametrais mais adequados. O fato é conhecido como treinamento da rede. A cada apresentação dos dados, as informações podem ser registradas ou randomizadas. A taxa de aprendizado de rede neural reflete o tamanho do passo de ajuste das ponderações, como poderia acontecer no caso da retropropagação. Muitas vezes cabe ao usuário dar entrada a esses parâmetros — que podem afetar dramaticamente a precisão da rede.

Características Essenciais

Há diversos *softwares* de redes neurais, cada

um com sua própria interface para simplificar as coisas ao mesmo tempo que proporciona a flexibilidade necessária para construir bons modelos. Independentemente do *software* escolhido, é necessário estudar, com cuidado, quatro aspectos do pacote.

Interface de dados. O *software* escolhido deve ter uma excelente interface que permita visualizar os dados, realizar análises preliminares e deixar todos na forma apropriada para a construção de modelos.

Classificação e/ou previsão. O *software* precisa oferecer as rotinas adequadas para a construção de modelos de PI e PCI. Quando se está construindo um modelo de probabilidade de inadimplência para atender às exigências do Novo Acordo, pode haver interesse por uma ferramenta preditiva e não meramente classificadora. É preciso verificar se o *software* é capaz de extrair probabilidades estimadas. Para modelagem de PCI, o melhor pode ser uma rede neural de regressão geral (RNRR). As RNRRs freqüentemente são de rápida estimativa, mas às vezes são um pouco lentas para fazer previsões baseadas em novos dados. Mas um bom pacote de rede neural deve ser suficientemente flexível para incluir todas essas características.

Análise de sensibilidade. Esta característica indica as variáveis preditivas mais importantes

Back-propagation is the most popular parameter estimation technique.

A retropropagação é a técnica de estimativa parametral mais popular.

models. Regardless of which software you choose, look carefully at how the package does four things.

Data interface. *The software you select should have an excellent interface to allow you to look at your data, perform preliminary analysis on it, and get it in the right form for model building.*

Classification and/or prediction. *The software needs to have the proper routines available for building PD and LGD models. If you are building a probability of default model for Basel requirements, then you may be interested in a prediction tool rather than a pure classification tool. Be sure to determine whether the software offers the capability of extracting probability estimates. For LGD modeling, you might want a general regression neural network (GNRR). GNRR often is quick to estimate but sometimes a little slow to make predictions on new data. However, a good neural net package should be flexible enough to include all these capabilities.*

Sensitivity analysis. *This feature indicates the model's most important predictor variables. Sensitivity analysis can be done in a variety of ways[4], but because nonlinearity issues surround a neural net model, take care*

to evaluate each variable. In other words, the impact of each variable may depend not only on its value, but also on the interaction of other variables. So perform this analysis by using graphical plots of the data across the range of input values for each predictor. Sometimes the software automatically performs this type of analysis by removing the variable from the model and examining the impact on the output value. It would be wise to have looked into the sensitivity issue before deployment, as regulators expect you to know what variables play the most important role in your model.

Implementation code. *Implementing a PD model derived from a logistic regression is relatively simple; the most difficult part is performing a basic exponential function. However, implementing a neural network model can be a complex and difficult task. A good neural net software package should have a deployment engine that automatically generates the code necessary to score new data in languages like C (a programming language, similar to Visual Basic–VB — but much quicker). Deployment software should be able to handle data cleansing transformations and to perform the appropriate me-*

The primary engine in the network resides in the hidden layer.

Os principais mecanismos da rede neural estão na camada oculta.

do modelo. A análise de sensibilidade pode ser feita de diversas maneiras[4] mas, por causa dos aspectos de não-linearidade que acompanham os modelos de rede neural, é necessário ter cautela, avaliando cada uma das variáveis. Em outras palavras, o impacto de cada variável pode depender não só do seu valor, mas também da interação entre as demais variáveis. Assim, a análise deve usar plotagens gráficas dos dados com a gama de valores de entrada para cada variável preditiva. Às vezes o *software* realiza automaticamente esse tipo de análise, removendo a variável do modelo e examinando o impacto sobre o valor de saída. Pode ser recomendável examinar a questão da sensibilidade antes da aplicação, uma vez que os reguladores esperam que saibamos quais variáveis desempenham as funções mais importantes em nossos modelos.

Código de implementação. Implementar um modelo de PI derivado de um regressão logística é algo relativamente simples; seu aspecto mais difícil é realizar uma função exponencial básica. Mas implementar um modelo de rede neural pode ser uma tarefa complexa e difícil. Um bom pacote de *software* de rede neural deve incluir um engenho de aplicação que gere automaticamente o código necessário para fazer o *score* de novos dados em linguagens como C (uma linguagem de pro-

gramação parecida com Visual Basic-VB — mas muito mais rápida). O *software* de aplicação deve ser capaz de lidar com transformações de limpeza de dados e usar os métodos adequados para operar com os dados ausentes. O código deve, ainda, ser escrito com suficiente eficiência para fazer, em tempo oportuno, o *score* do total da carteira.

Desafios de Modelagem

Construir modelos de regressão é tanto uma arte quanto uma ciência. Projetar uma rede neural eficaz apresenta desafios semelhantes. Ao construir um modelo de PI ou PCI por meio de uma técnica de regressão, é preciso decidir, no mínimo, que variáveis incluir e eliminar e optar entre transformar as variáveis ou dividi-las em variáveis preditivas mais definidas. Ao construir um modelo de PI ou PCI através de uma rede neural, é necessário decidir quais as variáveis preditivas de entrada serão utilizadas, quantos nós deverá haver na camada oculta, qual o número de épocas a serem rodadas, qual a taxa de aprendizado especificada e, em alguns casos, definir a rotina de treinamento mais adequada.

Às vezes é possível não haver a convergência do algoritmo de estimativa de um modelo de regressão logística por causa de algum problema. Mas a falta de convergência é óbvia porque nor-

O software escolhido deve ter uma excelente interface.

The software you select should have an excellent interface.

thods to account for missing data. The code also should be written in a manner efficient enough to score the entire portfolio in a timely manner.

Modeling Challenges

Building regression models is both art and science. Designing an effective neural network poses similar challenges. In building a PD or LGD model using a regression technique, you must decide at the very least which variables to include and which to eliminate, and whether to transform the variables or break them up into more discrete predictors. In building a PD or LGD model using a neural net, you still have to decide which input predictors to use, how many nodes should be present in the hidden layer, the number of epochs to run, which learning rate to specify, and sometimes which training routine is appropriate.

It occasionally is possible that the estimation algorithm in a logistic regression model will not "converge" because of one problem or another. However, lack of convergence is obvious because warning or error messages are usually produced. In neural networks, it is possible that the network will converge on a solution that

is more local than global — meaning that the process does not find the overall best solution. This is part of the trial-and-error methodology associated with neural nets. Some schools of thought say that rather than a single model, you should build a variety of models that vary the number of nodes in the hidden layer, learning rates, and other parameters needed by the neural net software.

Summary

Neural networks can offer the analyst a means to build flexible nonlinear models for portfolios with credit behavior that are potentially too complex for traditional techniques such as regression analysis.[5] One advantage is that the network can automatically determine what these nonlinearities may look like — a big plus if the relationship within the data are not well understood beforehand. These networks are generally designed around a three-layer architecture — the input layer, the hidden layer, and the output layer. The primary engine in these networks resides in the hidden layer through activation functions. Although neural networks are by design able to capture nonlinearities in the data, their use is not without human judgment. Since there are many different neural net software packages commercially available — each with its own variety of estimation algorithms, optimization routines, input parameters, and graphical interfaces —, software-specific experience is needed to build an effective predictive model.

malmente são produzidas mensagens de alerta ou erro. Nas redes neurais, é possível que a rede venha a convergir para uma solução mais local do que global — ou seja, pode ser que o processo não identifique a melhor solução genérica. Isso faz parte da metodologia de tentativa e erro associada às redes neurais. Algumas escolas de pensamento afirmam que, em vez de um só modelo, devemos construir diversos modelos com diferente números de nós na camada oculta, taxas de aprendizado e outros parâmetros exigidos pelo *software* de rede neural.

Sumário

As redes neurais podem oferecer ao analista um meio de construir modelos não-lineares flexíveis para carteiras cujo comportamento de crédito possa ser complexo demais para técnicas tradicionais, como a análise de regressão.[5] Uma vantagem é o fato de que a rede é capaz de determinar automaticamente a aparência que as não-linearidades podem ter — o que é importante se a relação entre os dados não for bem compreendida de antemão. Essas redes costumam ser concebidas em torno de uma arquitetura em três camadas — de entrada, oculta e de saída. O principal engenho dessas redes reside na camada oculta, operando através de funções de ativação. Embora as redes neurais sejam, por natureza, capazes de capturar características não-lineares dos dados, elas não operam livres de julgamento humano. Como há muitos pacotes de *software* de rede neural disponíveis no mercado — cada um com sua própria seleção de algoritmos de estimativa, rotinas de otimização, parâmetros de entrada e interfaces gráficas

—, é preciso ter experiência no uso do *software* escolhido para construir um modelo preditivo eficaz.

Uma vez treinada e aplicada a rede neural, resta o problema de explicar seu desenho tanto aos reguladores quanto à linha de negócios. É preciso estar preparado para perguntas como "por que uma conta recebe um *score* elevado enquanto outras que parecem semelhantes recebem *scores* muito menores?". Embora uma boa análise de sensibilidade possa ajudar a responder essas perguntas, é preciso enfrentar a formidável tarefa de explicar um modelo, possivelmente complexo, que será inevitavelmente visto por algumas pessoas como uma "caixa preta". Dado que os modeladores que usam abordagens de regressão estão dotados de estratégias para aproximar, em certa medida, relações não-lineares, pode surgir o argumento de que o uso de redes neurais para estimativas de PI e PCI só vale a pena quando seu valor preditivo é substancialmente maior.[6] Assim sendo, a sugestão é de que as redes neurais sejam usadas no processo de modelagem como parte de um processo de campeão-desafiante em que o vencedor, sendo uma rede neural, somente seja aplicado se proporcionar um aumento substancial do poder de previsão.

Notas

1. MORRISON, Jeffrey S., "*Preparing for Modeling Requirements in Basel II, Part 2: Modeling Strategies*". The RMA Journal, maio de 2004.

2. As RNAs foram inicialmente utilizadas no setor de defesa para reconhecer e classificar submarinos inimigos. O conceito original por detrás

Once the neural network has been trained and deployed, there remains the problem of explaining its design to both the regulators and the line of business. You must be prepared for such questions as why our account receives a high score while others that appear very similar receive a much lower score. Although a good sensitivity analysis can help answer these questions, you still face the formidable task of explaining a possibly complex model that will inevitably be seen by some as a "black box". Given that modelers using regression approaches are equipped with strategies to approximate nonlinear relationships to some extent, it may be argued that the use of neural networks for PD and LGD estimates is worthwhile only when their incremental predictive value is substantial.[6] Therefore, I recommend using neural networks in the modeling process as part of a champion-challenger strategy where the winner, if it is found to be a neural network, is only deployed given a substantial increase in predictive power.

Notes

1. *MORRISON, Jeffrey S., "Preparing for Modeling Requirements in Basel II, Part 2: Modeling Strategies". The RMA Journal, May 2004.*

2. *ANN was used early on in the defense industry to correctly recognize and classify enemy submarines. The initial concepts behind the approach began to surface in the 1940s, but the exponential growth in computer technology has allowed neural ne-*

tworks to be applied in almost every industry imaginable. In the credit world, perhaps one of the most prominent successes includes fraud prediction using transactional data. Because of the highly technical nature inherent in any discussion of the subject, we will provide only a high level overview of the topic and focus our attention on its value in modeling and implementing requirements for Basel II.

3. MCMENAMIN, J. Stuart, "Why not Phi? A Primer for Neural Networks for Forecasting", Regional Economic Research, Inc., April 1997.

4. YAO, J.T., "Sensitivity Analysis for Data Mining", Department of Computer Science, University of Regina, Canada. Research publication.

5. ATYIA, Amir F., "Bankruptcy Prediction for Credit Risk Using Neural Networks: A Survey and New Results", IEEE Transaction of Neural Networks Vol. 12, No. 4, July 2001.

6. FEDENCZUK, Leon L., "To Neural or Not to Neural? SUGI27 — Data Mining Techniques", SAS User's Group publication.

da abordagem começou a emergir na década de 1940, mas o avanço exponencial da informática permitiu que as redes neurais fossem aplicadas a praticamente qualquer setor. No mundo do crédito, um de seus maiores sucessos talvez seja a previsão de fraudes por meio de dados transacionais. Por causa da natureza altamente técnica de qualquer discussão desse assunto, forneceremos apenas um panorama geral e concentraremos nossa atenção em seu valor para a modelagem e implementação em atendimento às exigências do Novo Acordo da Basiléia.

3. MCMENAMIN, J. Stuart, "*Why not Phi? A Primer for Neural Networks for Forecasting*", Regional Economic Research, Inc., abril de 1997.

4. YAO, J.T., "*Sensitivity Analysis for Data Mining*", Department of Computer Science, University of Regina, Canada. Research publication.

5. ATYIA, Amir F., "*Bankruptcy Prediction for Credit Risk Using Neural Networks: A Survey and New Results*", IEEE Transaction of Neural Networks Vol. 12, No. 4, julho de 2001.

6. FEDENCZUK, Leon L., "*To Neural or Not to Neural?* SUGI27 — Data Mining Techniques", SAS User's Group publication.