Dr. George Karraz, Ph. D.

Neural Networks in Business Applications

Outline

- Customer Loan Classification
- Bankruptcy Prediction
- Exchange rate Forecasting
- Conclusions

1.Customer Loan Approval

- **Problem Statement.** Many stores are now offering their customers the possibility of applying for a loan directly at the store, so that they can proceed with the purchase of relatively expensive items without having to put up the entire capital all at once.

- Initially the offering of consumer loans was found only in connection with expensive purchases, such as cars, but it is now commonly offered at major department stores for purchases of washing machines, televisions, and other consumer goods.

- The loan applications are filled out at the store and the consumer deals only with the store clerks for the entire process.

- The store, however, relies on a financial company (often a bank) that handles such loans, evaluates the applications, provides the funds, and handles the credit recovery process when a client fails to fulfill the repayment schedule.

- For this study, there were 1000 records of consumer loan applications that were granted by a bank, together with the indication whether each loan had been always paid on schedule or there had been any problem.

- The provided data did not make a more detailed distinction about the kind of problem encountered by those "bad" loans, which could range from a single payment that arrived late to a complete defaulting on the loan.

ANN Application to Loan Approval

- We use an ANN to classify the consumer loans and identify potentially risky loans, that is, loans that might not be repaid on schedule.

Input variables

- Each application had ***15 variables*** that included the number of members of the household with an income, the amount of the loan requested, whether or not the applicant had a phone in his/her house, etc.
- Some of these variables were numerical (*e.g.* the number of relatives), while other used a digit as a label to indicate a specific class (*e.g.* the values 0,1,2,3 of variable 8 referred to four different classes of employment).

	Input variables	Variable value
1	N° of relatives	from 1 to total components
2	N° of relatives with job	from 0 to total components
3	Telephone number	0,1
4	Real estate	0,1
5	Residence seniority	from 0 to date of loan request
6	Other loans	0,1,2
7	Payment method	0,1
8	Job type	0,1,2,3
9	Job seniority	from 0 to date of loan request
10	Net monthly earnings	integer value
11	Collateral	0,1,2
12	Loan type	0,1,2,3
13	Amount of loan	integer value
14	Amount of installment	integer value
15	Duration of loan	integer value
	Computed output variable	
1	Repayment probability	from 0 to 100
	Desired output variable	
1	Real result of grant loan	0 if repayment irregular or null 100 if repayment on schedule

Table 1: Input and output variables

- For each record, a single output variable indicated whether the loan reached was extinguished without any problem (Z=100) or with some problem (Z=0).

- **Network topology**: Multi-Layer Perceptron (MLP) with one input layer, one hidden layer and one output layer. We use a network with 10 nodes in hidden layer.
- **Training Algorithm**: For training ANN, Bayesian has been used to train the ANN rather than back-propagation algorithm.
- In its *a-posteriori* analysis, the bank classified loans with Z=0 as "bad loans". In the provided data, only about 6% of the loans were classified as "bad".

The output variable

- The single output of our network turned out to be in the range from -30 to +130, whereas the corresponding "real" output was limited to the values Z=0 or Z=100. A negative value of the output would indicate a very bad loan and thus negative values were clamped to zero; similarly, output values above 100 were assigned the value of 100.
- A 30% tolerance was used on the outputs so that loans would be classified as "good" if the ANN computed a value above 70, and "bad" is their output was less then 30.
- Loans that fell in the intermediate band [30, 70] were left as "unclassified". The rationale for the existence of the "unclassified" band is to provide an alarm requesting a more detailed examination of those loans, possibly by requesting additional information from the applicant.

Training set and test set

- Of the available 1000 cases, ***400*** cases used to train the ANN.
- Then this specific ANN was supplied with the remaining ***600*** cases of the testing set.
- This set contained ***38*** cases that had been classified as bad (Z=0), while the remaining ***562*** cases had been repaid on schedule.
- Figure 1 shows the output of ANN for all 600 cases analyzed. Each point on the horizontal axis corresponds to one of the 600 cases, with the output computed by the ANN shown by a diamond or a cross.

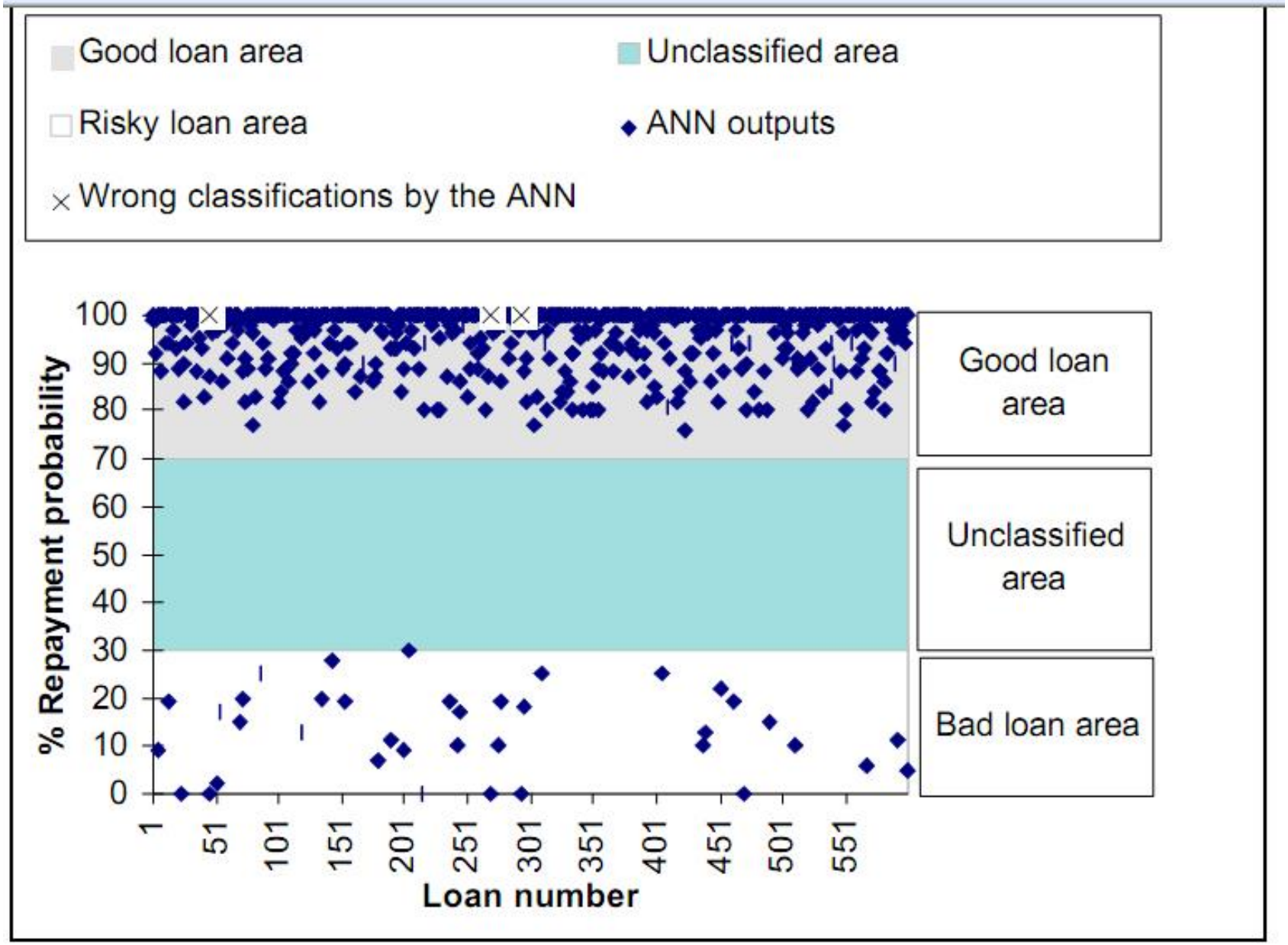


Figure 1: Loan classification by ANN

Experiment results

- Clearly the ANN separates the given cases into two non-overlapping bands: the good ones near the top and the bad ones near the bottom. No loan was left unclassified, so in this case there would have been no cases requiring additional (human) intervention.
- The ANN made exactly 3 mistakes in the classification of the test cases: those were 3 cases that the ANN classified as "good" loans, whereas in reality they turned out to be "bad". Manual, *a-posteriori* inspection of the values of their input variables did not reveal any obvious symptoms that they were "problem cases".
- What could have likely happened is that the applicant did not repay the loan as schedule due to some completely unforeseen and unpredictable circumstance. This is also supported by the fact that the bank officers themselves approved those 3 loans, thus one must presume that they did not look too risky at application time.

- The ANN, however, was more discriminating than the bank officers since the ANN would have denied 35 loan applications that scored less than 30.
- As is turns out, all those 35 loans had problems with their repayments and thus the bank would have been well advised to heed the network's classification and to deny those 35 applications.

2. Bankruptcy Prediction

- *Bankruptcy prediction* problem is important to creditors, auditors and shareholders.
- There have been a lot of work on developing neural networks to predict bankruptcy using financial ratios.
- The ANN selected in the design phase for this problem was a three-layer feed-forward ANN using back-propagation.
- The data for training the network consist of a small set of numbers for well-known *financial ratios*, and data were available on the bankruptcy outcomes corresponding to known data sets.

Application design

- There are five input nodes, corresponding to five financial ratios:
  - X1: Working capital/total assets
  - X2: Retained earnings/total assets
  - X3: Earnings before interest and taxes/total assets
  - X4: Market value of equity/total debt
  - X5: Sales/total assets

  (The same financial ratios selected by Altman, 1968)
- Two output nodes gives the final classification (one indicating a bankrupt firm or and one indicating non-bankrupt firm).
  - For instance, a bankrupt training case that has its bankrupt output node valued at 0.85 and its non-bankrupt node at 0.17.

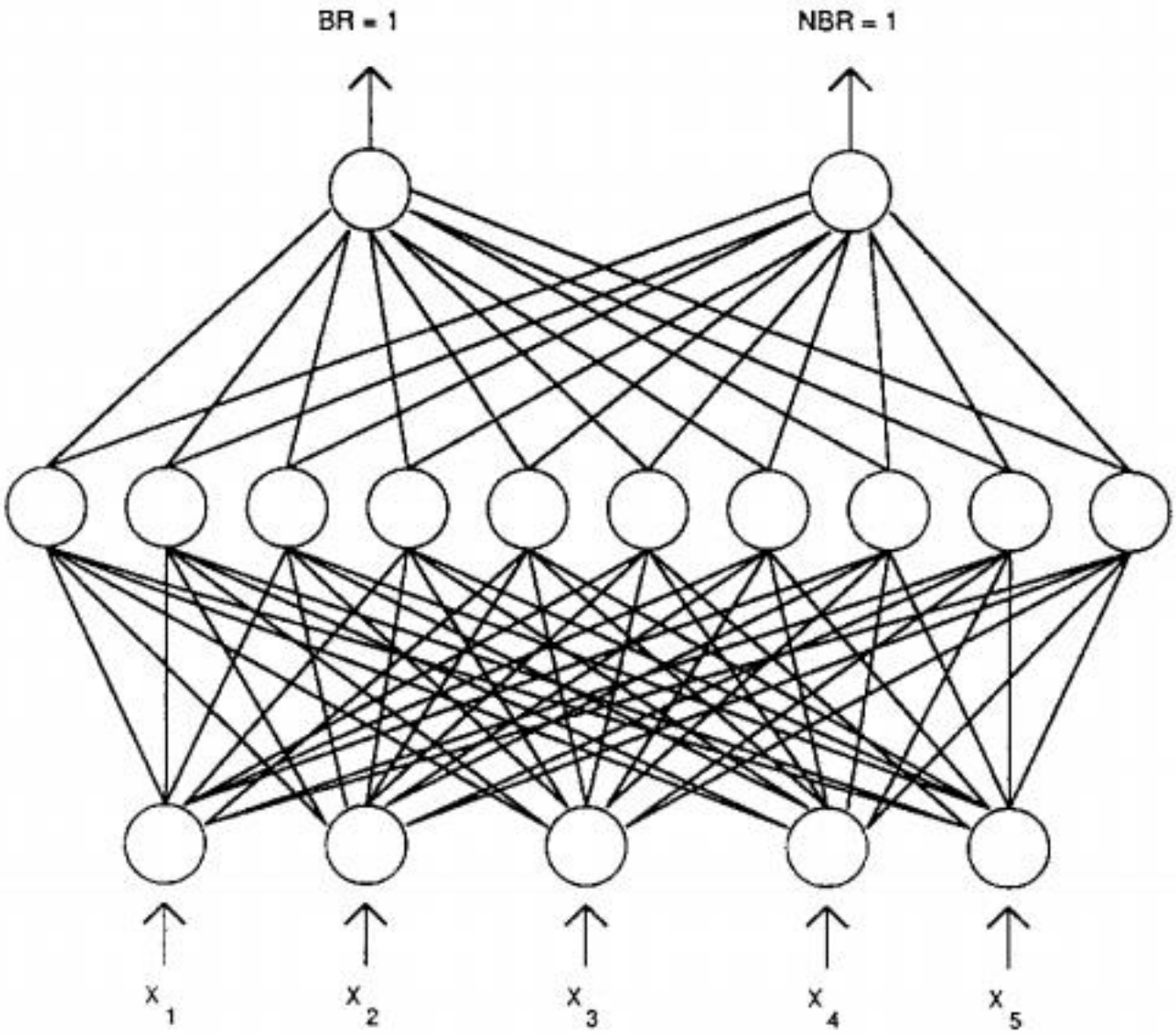
Atman's five financial ratios

- A company's *total assets* consists of *current assets* and *long term assets*.
- The *working capital* is *current assets* minus the *current liabilities*.
- The *current liabilities* include short term loans, accounts payable, taxes due, etc.
- The firm's *liabilities* consists of *current liabilities* and *long term debts*.
- The *retained earnings* means the accumulation of the firm's earnings since the firm's inception.
- The firm's *total assets* is based on a) the *total liabilities* and b) *shareholder's equity*.
- The *shareholder's equity* consists of the *capital* raised in share offerings and the *retained earnings*.

- The data source consists of financial ratios for 129 firms, 65 of which went bankruptcy and 64 non-bankrupt during the period between 1975 and 1982.
- *Financial ratios* were calculated for each of the *five* aspects shown above, each of which became the input for one of the five input nodes.
- For each set of data, the actual result, whether or not bankruptcy occurred, can be compared to the neural network's output to measure the performance of the network and monitor the training.

ANN Architecture

- This work uses *BRAINMAKER*, a neural network software package (California Scientific Software) which implement back-propagation training algorithm, to construct and test the ANN.
- The architecture of the ANN is shown in the following figure with:
  - 5 input nodes
  - 10 hidden nodes
  - 2 output nodes


BR = 1
NBR = 1
X 1
X 2
X 3
X 4
X 5

Training

- The data set, consisting of 129 firms, was partitioned into a training set and a test set. The training set of 74 firms consisted of 38 that went bankrupt and 36 that did not. The needed financial ratios were computed and stored in the input file to the neural network and in a file for a conventional *discriminant analysis* program for comparison of the two techniques.
- The neural network has three important parameters to be set: learning threshold, learning rate, and momentum.
  - The learning threshold allows the developer to vary the acceptable overall error for the training case.
  - The learning rate and momentum allow the developer to control the step sizes the network uses to adjust the weights as the errors between computed and actual outputs are fed back.

Testing

- The ANN was tested in two ways: by using the test data set and by comparison with discriminant analysis. The test set consisted of 27 bankrupt and 28 non-bankrupt firms.
- The ANN was able to correctly predict 81.5% of the bankrupt cases and 82.1% of the nonbankrupt cases.
  - Overall, the ANN did much better predicting 22 out of the 27 actual cases (the discriminant analysis predicted only 16 cases correctly).
  - An analysis of the errors showed that 5 of the bankrupt firms classified as non-bankrupt were also misclassified by the discriminant analysis method. A similar situation occurred for the non-bankrupt cases.
- The result of the testing showed that ANN implementation is at least as good as the conventional approach. An accuracy of about 80% is usually acceptable for ANN applications. At this level, a system is useful because it automatically identifies problem situations for further analysis by a human expert.

3. Exchange Rate Forecasting

- *Exchange rate forecasting* is an important and challenging task due to the unpredictability of the foreign exchange rate.
- Neural networks have general *nonlinear function mapping capability*.
- Neural networks are nonparametric *data-driven* models which impose little prior assumptions on the underlying processes from which data are generated.
- So neural networks are suitable for forecasting exchange rates.
- But building a *neural network forecaster* is a non-trivial task.
- Although there exists many types of neural networks, *multi-layer feed-forward networks* are still the most commonly-used.

Neural network for time series forecasting

- Neural networks are able to capture the underlying pattern or autocorrelation structure within a time series even when the underlying law governing the system is unknown or too complex to describe.
- The most popular neural network used for forecasting is the feed-forward multilayer network (MLP), given in the Figure 3.1.
- For a *univariate* time series forecasting problem, the *inputs* of the network are the past, lagged observations of the data series and the *outputs* are the future values.
- The network represented in Figure 3.1 is a mapping function of the form:

$$y_{t+1} = f(y_t, y_{t-1}, \ldots, y_{t-p})$$

where y_t is the observations at time t and p is the dimension of the input vector or the number of past observations related to the future value.

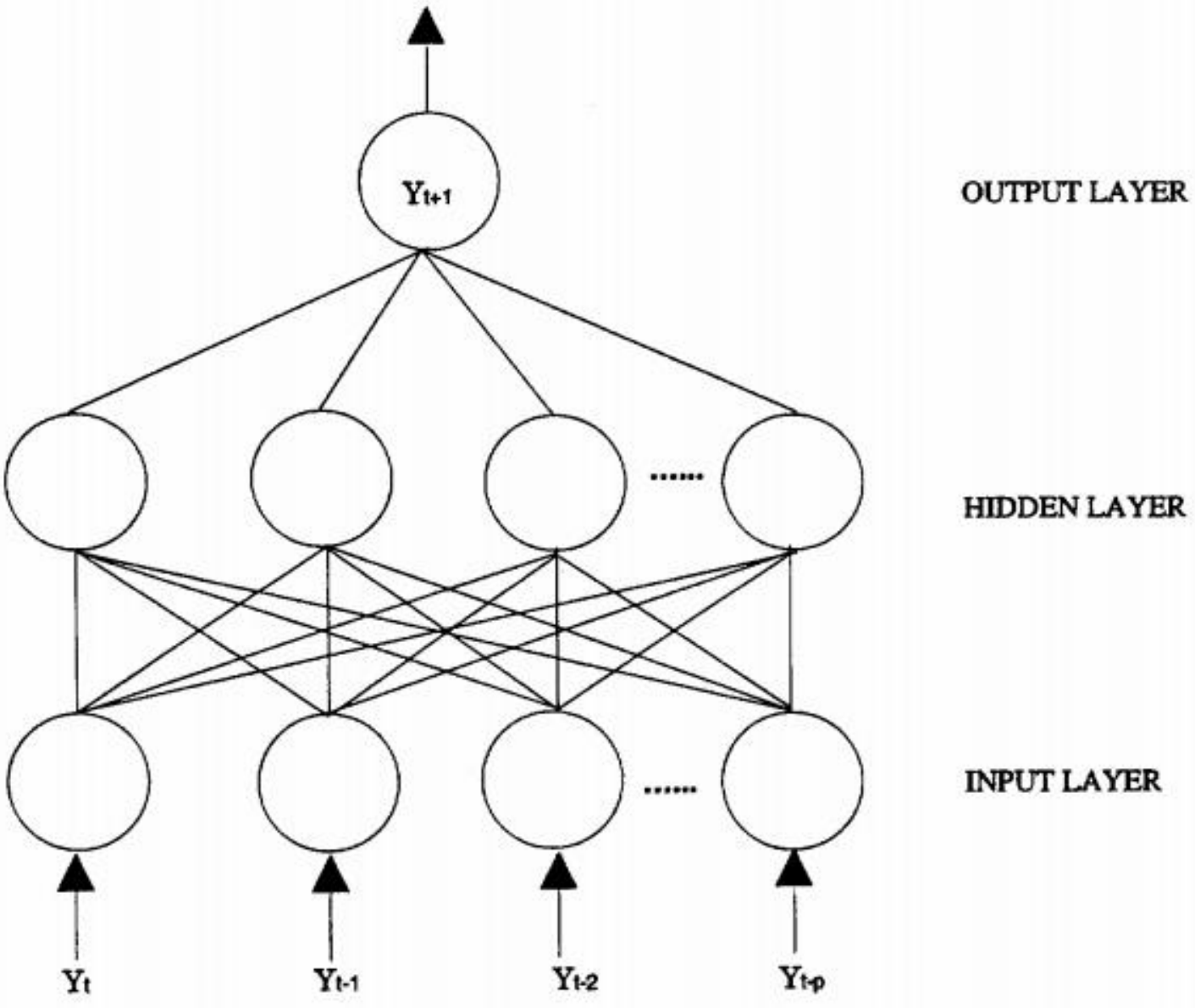


Figure 3.1 A feedforward neural network for time series forecasting

Data representation in training a neural network

- Suppose we have N time-lagged observations $y_1, y_2, \ldots, y_N$ in the training set and we need the one-step-ahead forecast, then using a network with p input nodes and one output node, we have N- p training patterns.
- The first training pattern is composed of $y_1, y_2, \ldots, y_p$ as the inputs and y_{p+1} as the target output.
- The second training pattern contain of $y_2, y_3, \ldots, y_{p+1}$ as the inputs and y_{p+2} as the target output.
- The final training pattern is $y_{N-p}, y_{N-p+1}, \ldots, y_{N-1}$ as the inputs and y_N as the target output.
- See the Figure 3.2 for an illustration

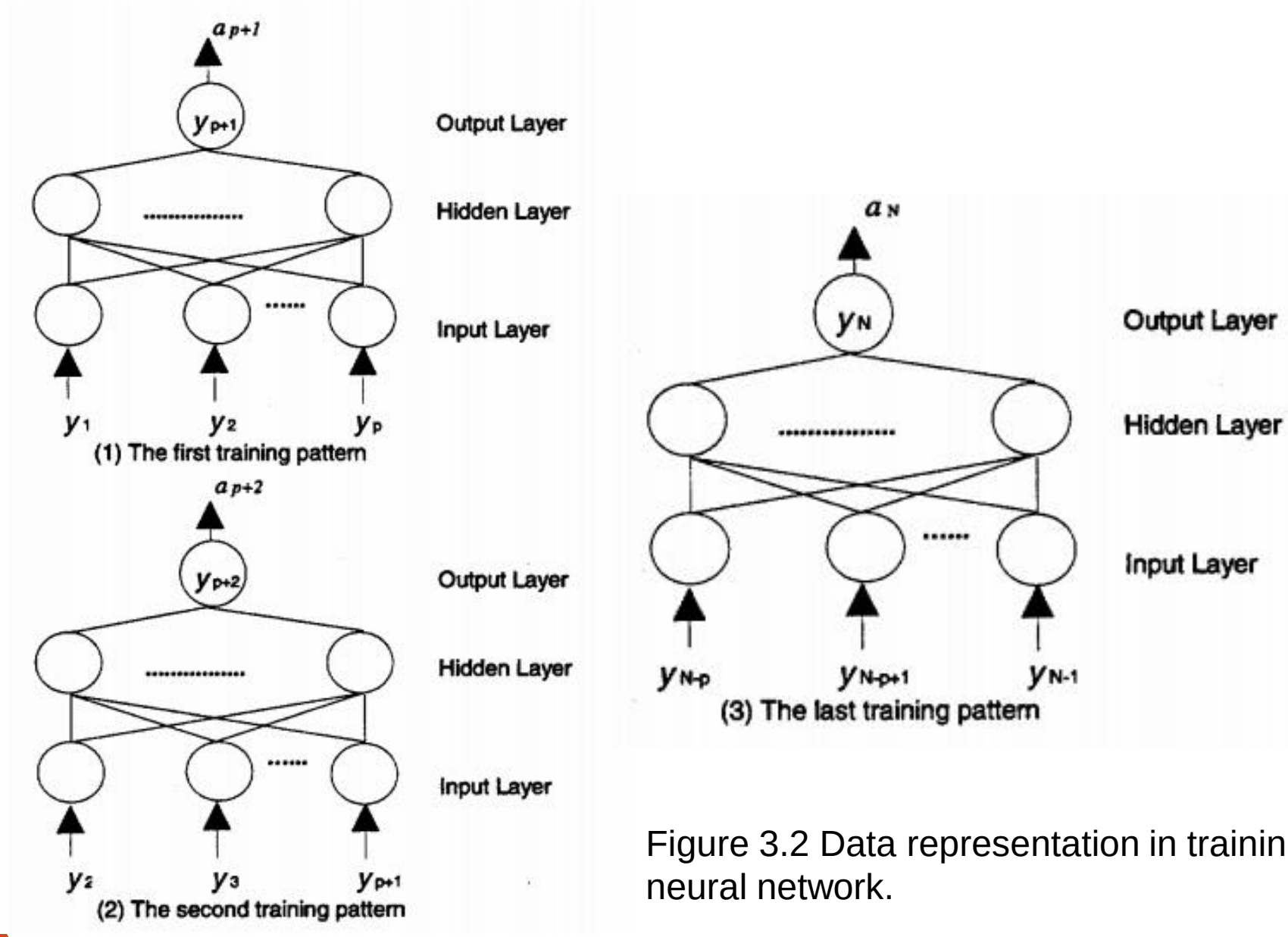


Figure 3.2 Data representation in training a neural network.

- The neural network training objective is to find the weights in order that some overall *predicted error measure* such as the sum of the squared errors (SSE) is minimized.
- For this network structure, SSE can be written as:

$$SSE = \sum_{i=p+1}^{N} (y_i - a_i)^2$$

where a_i is the output from the network.

Experiments

- Three-layer feedforward neural networks are used.
- *Logistic* (*sigmod*) activation functions are used in the hidden layer and the linear activation function is used in the output layer.
- Number of input nodes: varies from 1 to 10
- Number of hidden nodes: can be 4,8, 12, 16 and 20.
- No data normalization is required. Raw data are used.
- The large sample is consisted of 887 observations from 1976 to 1992 and the small one includes 261 data points from 1998 to 1992.
- The test sample for both cases is the 1993 data which has 52 observations.
- Three *time horizons* of one month, six months and twelve months. Forecasts are generated for the first 4 weeks, 26 weeks and 52 weeks of the test sample corresponding to the three forecast horizons.
- Focus on *one-step-ahead forecasting* (it uses the current observations to predict the next one).

Neural network training

- NN training is a nonlinear optimization.
- GRG2 (*Generalized Reduced Gradient*) software is used to train neural network rather than back-propagation algorithm.
- GRG2 is a *general-purpose nonlinear optimizer* which is widely available.
- GRG2-based system has been shown to be effective for highly nonlinear problem like NN training.
- Using GRG2, no need to specify learning rate and momentum term.
- Train each network 50 times by using 50 sets of different randomly generated initial weights. The best solution among 50 runs is used as the optimal neural network training solution.

4. Conclusions

- Neural networks have matured to the point of offering real practical benefits in many of their applications.
- Applying neural networks in business is a science and also an art.
- The design process of NN for business applications still involves much try-and error.
- Although there exists many types of neural networks, *three-layer feed-forward networks* are still the most commonly-used.
- Beside *back-propagation* algorithms, there are several other effective methods for training neural networks.