

Data Mining Techniques

Chapter 7: Artificial Neural Networks

Artificial Neural Networks	2
Neural network example	3
Inputs and output	4
Neural network process	5
What is a neural net?	6
A neuron	7
Feed-forward neural net example	8
How neural nets learn	9
Neural net heuristics	10
Choosing training sample	11
Preparing data	12
Other considerations	13

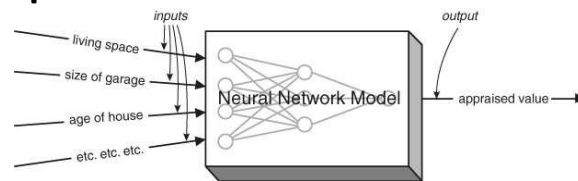
Artificial Neural Networks

- Biological neural networks: human brain enables people to generalize from experience.
- Artificial neural networks: computers generalize and learn from experience.
- *Black box* technique: little insight into how the model works or what the computer has done.
- History: ideas developed in computer science and artificial intelligence.
- Case study: real estate appraisal (p 213–8):
 - realtors combine features of a house (size, age, neighborhood, etc.) to come up with a valuation;
 - computer can do the same thing via a neural network model (see next slide).

© Iain Pardoe, 2006

2 / 13

Neural network example



- Inputs well understood, but how to combine (weight) them is unknown.
- Output well understood.
- Experience is available (lots of training data).
- Training presents known examples (data from previous sales) to the network so that it can learn how to calculate the sales price:
 - e.g., iteratively adjust weights to find best ones for predicting price (minimize overall error).

© Iain Pardoe, 2006

3 / 13

Inputs and output

- Inputs usually standardized or rescaled to lie between -1 and $+1$ (to avoid inputs with large ranges dominating the network):
 - quantitative: standardize (XLMiner: “normalize”);
 - quantitative (alternative): $2(X - m_X)/r_X$, where m_X is halfway between the minimum and maximum and r_X is the range;
 - ordered qualitative/categorical: map different categories to numbers between -1 and $+1$;
 - unordered qualitative/categorical: create $-1/+1$ inputs for pairs of categories.
- Output is usually in the range 0 to 1 (or sometimes -1 to $+1$) and needs to be mapped back to original scale for deployment.

© Iain Pardoe, 2006

4 / 13

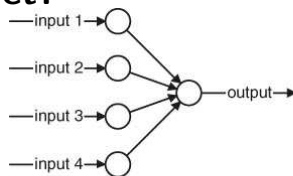
Neural network process

- Each iteration (adjusting weights) called a *generation* or *epoch*.
- Final epoch when cannot reduce error rate further:
 - fits training data best, but probably overfits.
- Earlier epoch usually fits validation data better.
- Neural networks for directed data mining (classification or prediction):
 - identify inputs/output;
 - standardize data;
 - set up network topology (see later);
 - train network with training sample;
 - use validation sample to select weights that minimize errors;
 - evaluate final model using test sample;
 - deploy model to predict unknown output for known inputs.

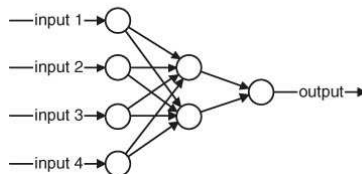
© Iain Pardoe, 2006

5 / 13

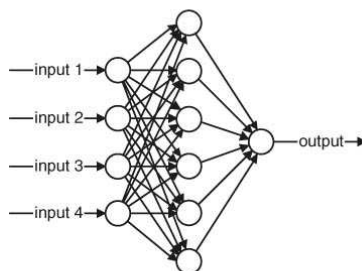
What is a neural net?



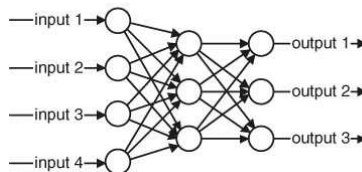
A very simple neural network that takes four inputs and produces an output. This result of training this network is equivalent to the statistical technique called logistic regression.



This network has a middle layer called the *hidden layer*, which makes the network more powerful by enabling it to recognize more patterns.



Increasing the size of the hidden layer makes the network more powerful but introduces the risk of overfitting. Usually, only one hidden layer is needed.



A neural network can produce multiple output values.

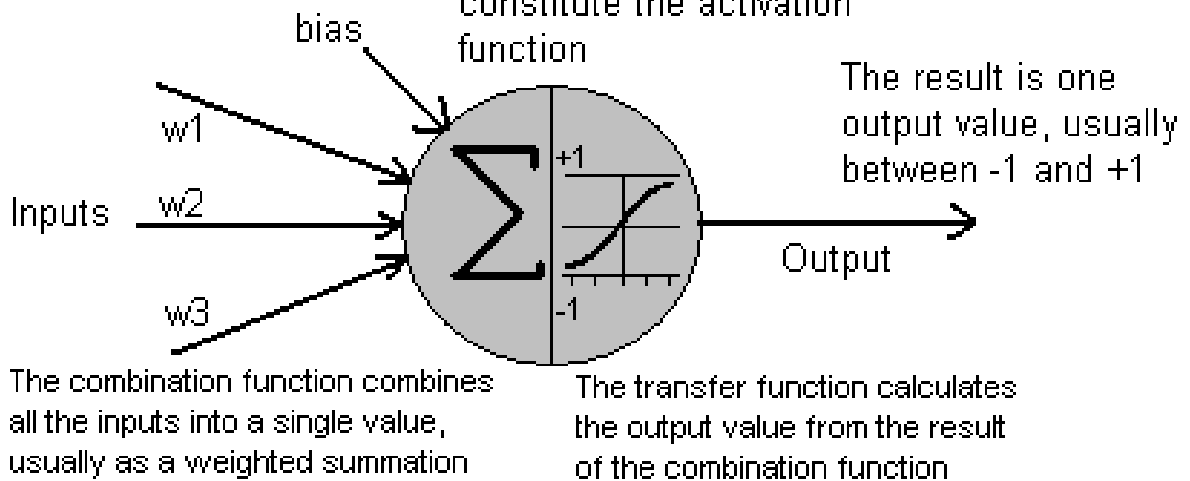
© Iain Pardoe, 2006

6 / 13

A neuron

Each input has its own weight,
plus there is an additional
weight called the bias

The combination function and
transfer function together
constitute the activation
function



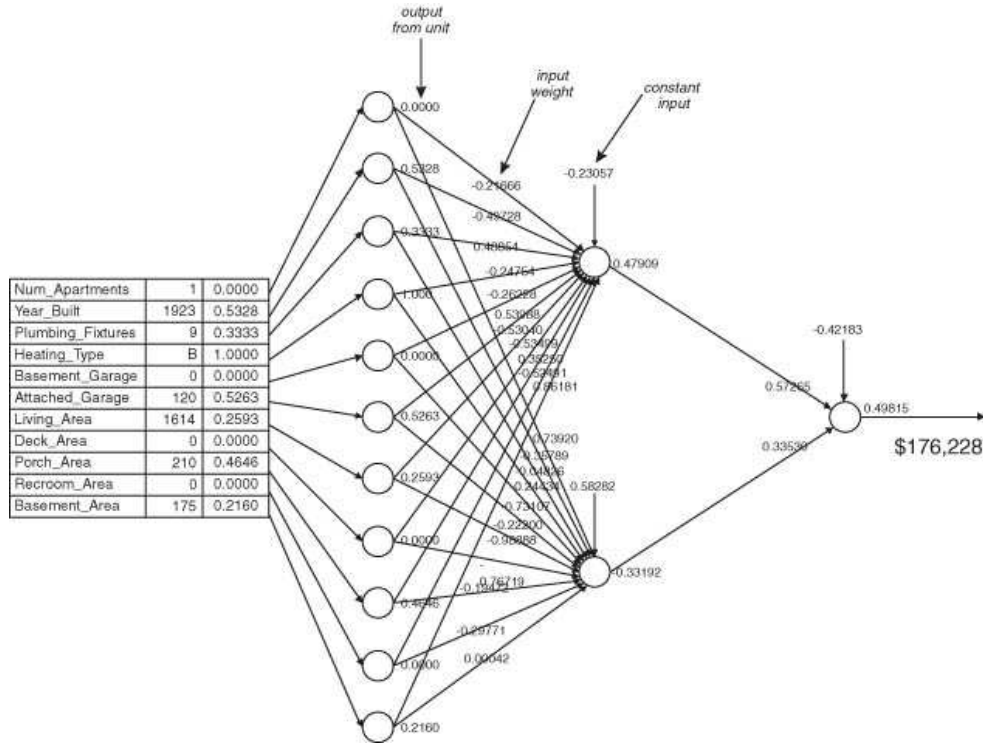
XLMiner has two types of transfer function:

- standard (logistic);
- symmetric (hyperbolic tangent).

© Iain Pardoe, 2006

7 / 13

Feed-forward neural net example



© Iain Pardoe, 2006

8 / 13

How neural nets learn

- Learn using *back propagation*:
 - feed classification/prediction errors back through network and adjust weights to reduce errors.
- Alternative methods, e.g., *conjugate gradient*:
 - see p. 230 (*jolly green giant*).
- After a number of iterations/generations/epochs, errors and weights no longer change and algorithm stops.
- Tuning parameters:
 - *learning rate* (step size for gradient descent);
 - weight change *momentum*.

© Iain Pardoe, 2006

9 / 13

Neural net heuristics

- # hidden layers and units/nodes per layer:
 - trial and error;
 - too low—poor fit;
 - too high—can recognize more patterns but may overfit;
- Size of training set:
 - aim for at least 30 per weight, e.g., for 15 inputs, a hidden layer with 10 nodes, and 1 output, aim for $30[10(15+1)+(10+1)] = 5130$ observations.
- learning rate (0.1–0.9):
 - start high, then decrease step size for gradient descent.
- weight change momentum (0–2):
 - allows network to find solution quicker.

© Iain Pardoe, 2006

10 / 13

Choosing training sample

- Coverage of values for all features (inputs and output).
- Number of features (variable selection), e.g., use background knowledge, intuition, or methods like decision trees first.
- Size of training set (see previous slide).
- Proportions of output categories (perhaps consider oversampling rare cases).

© Iain Pardoe, 2006

11 / 13

Preparing data

- Features with quantitative continuous values:
 - transform (standardize or map to $(-1, +1)$);
 - *bin* using quartiles or quintiles if highly skewed (or use log transform).
- Features with ordered discrete (integer) values:
 - map to $(-1, +1)$;
 - thermometer codes (see p. 238).
- Features with qualitative/categorical values:
 - use dummy indicator variables (1s and 0s);
 - alternative: effects coding (1s, 0s, and -1 s);
 - with L levels/categories only need $L - 1$ variables.
- Other types of features, e.g., dates and addresses, need special handling.

© Iain Pardoe, 2006

12 / 13

Other considerations

- Interpreting results: convert output to quantitative prediction for prediction network or probability for classification network.
- Neural networks for time series.
- Neural networks are hard to understand (black box), but *sensitivity analysis* can offer limited insight into input effects on a test set.
- Self-organizing maps (aka Kohonen networks) use undirected neural networks for clustering.
- Examples:
 - p. 252–4 in book;
 - BMW airbags—neural networks used to determine which airbags to deploy in a crash based on characteristics of the crash.