Forecasting with (Artificial) Neural Networks

Prof. Galit Shmueli

Forecasting Analytics

We've already seen
methods designed **for cross-sectional** data
used for forecasting time series

Linear regression

$Y_t = \beta_0 + \beta_1 t + \beta_2 \text{ season}_1 + \beta_3 \text{ season}_2 + \ldots$

$\log(Y_t) = \beta_0 + \beta_1 t + \beta_2 \text{ season}_1 + \beta_3 \text{ season}_2 + \ldots$

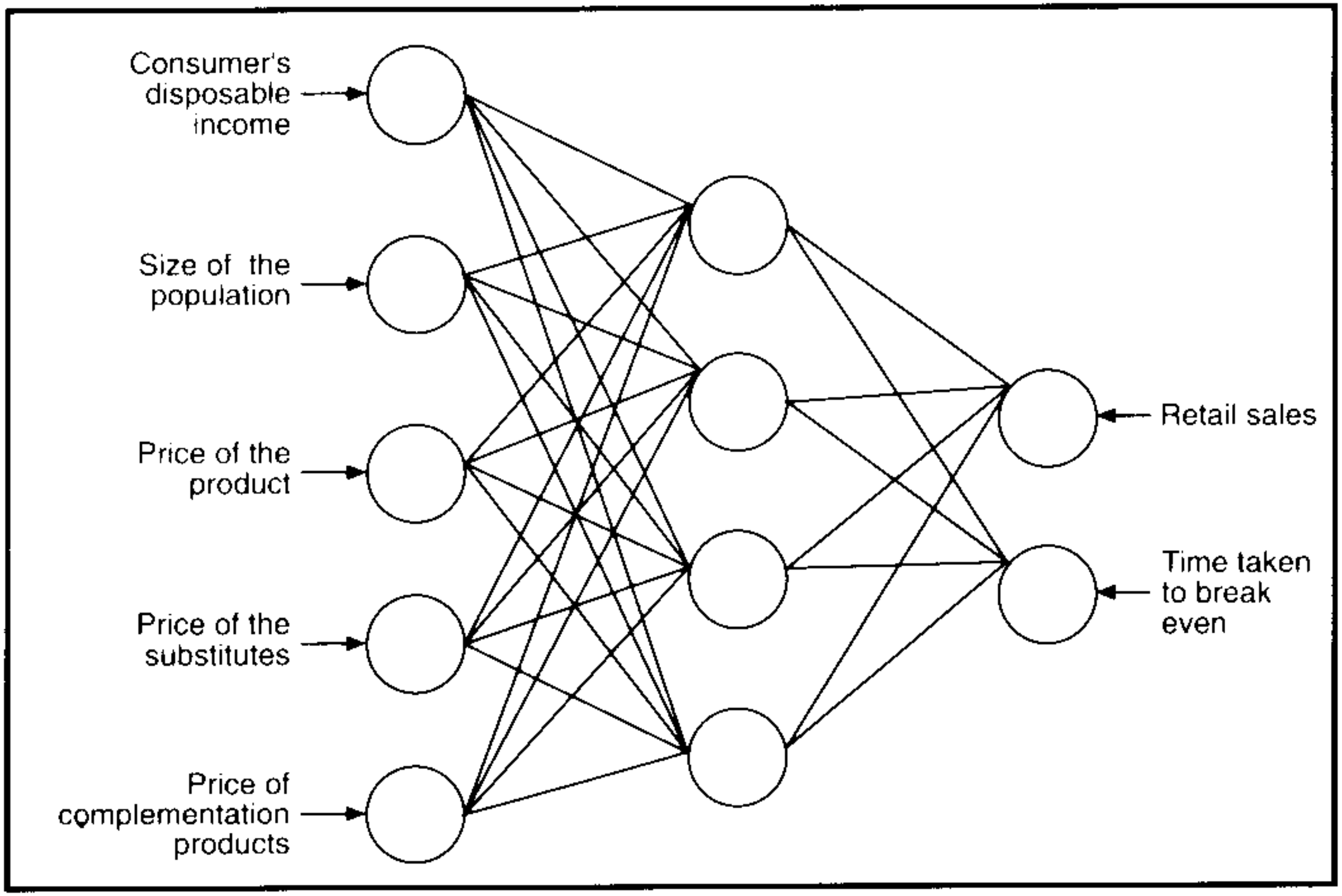
$Y_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \ldots$ (AR)

$Y_t = \beta_0 + \beta_1 X_{t-1} + \beta_2 Z_{t-2} + \ldots$

Logistic regression (binary outcome)

$\text{Logit}(Y_t) = \beta_0 + \ldots$ lags, external predictors, trend, seasonality

Among data mining algorithms for predicting cross-sectional data,



neural nets are popular for forecasting time series

NN Forecasting: Empirical Results

Tourism

Finance (trading)

Renewable energy

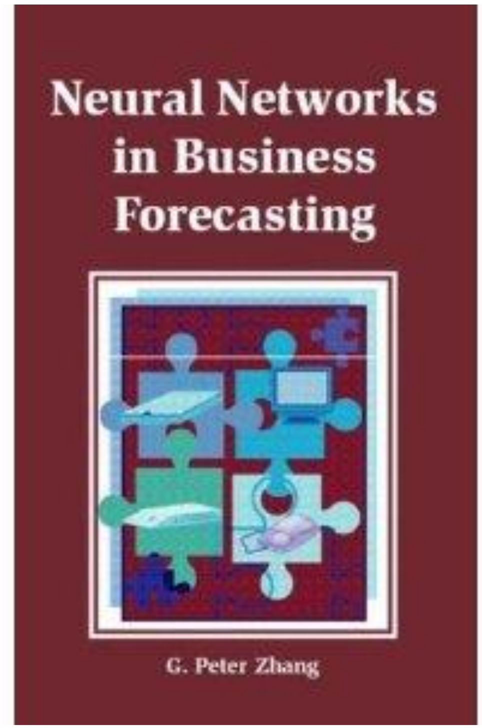
Mixed results compared to other methods

Seem to work best with high-frequency data

Very popular in 1990-2000's

"The development of ANNs is still an art rather than a science"

(p.69)



Available @ LRC

The idea: capture a complex relationship between output and inputs by creating **layers of derived variables**

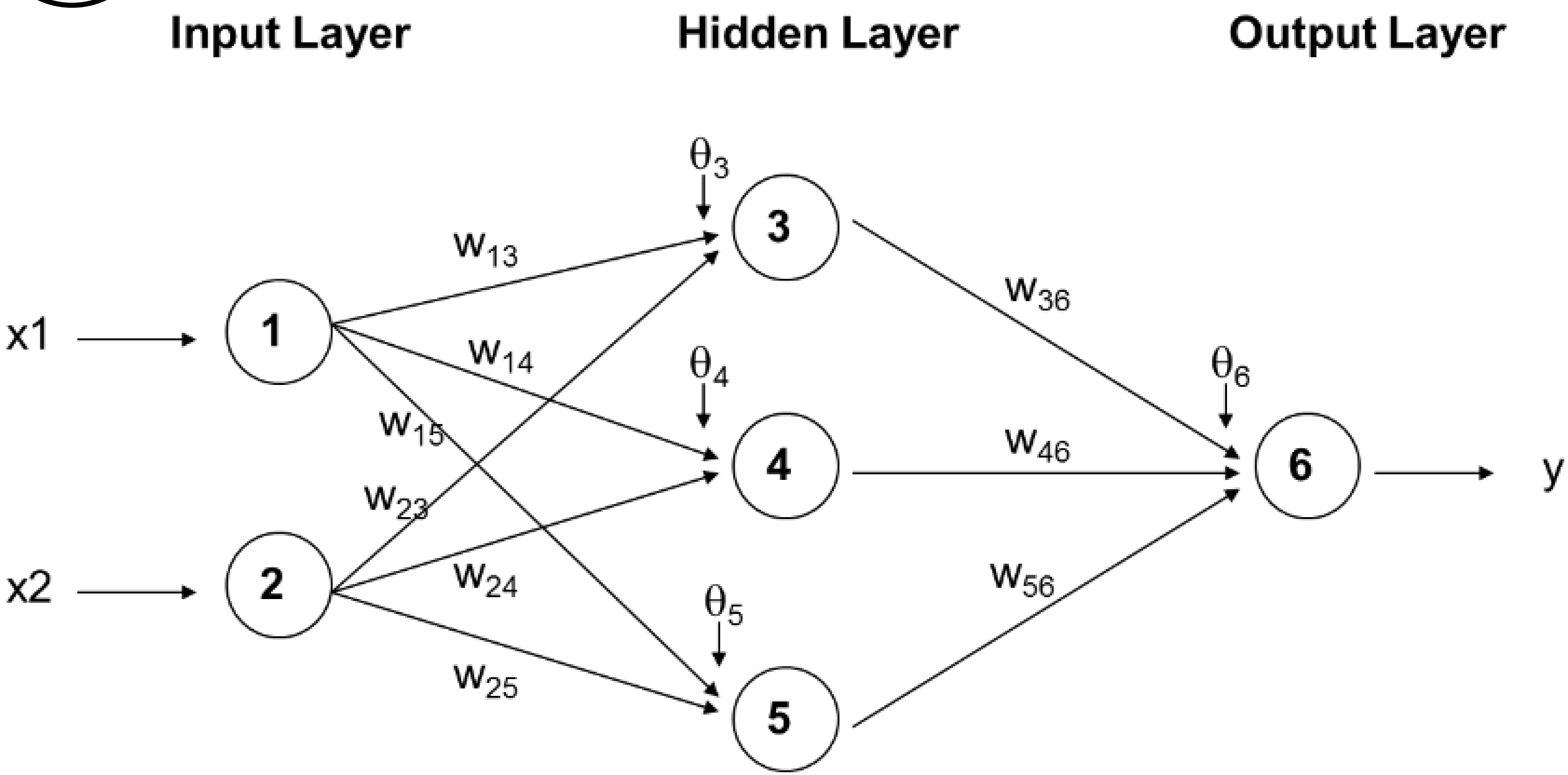
Y = output variable
X = original input variable
g(x) = derived input variable

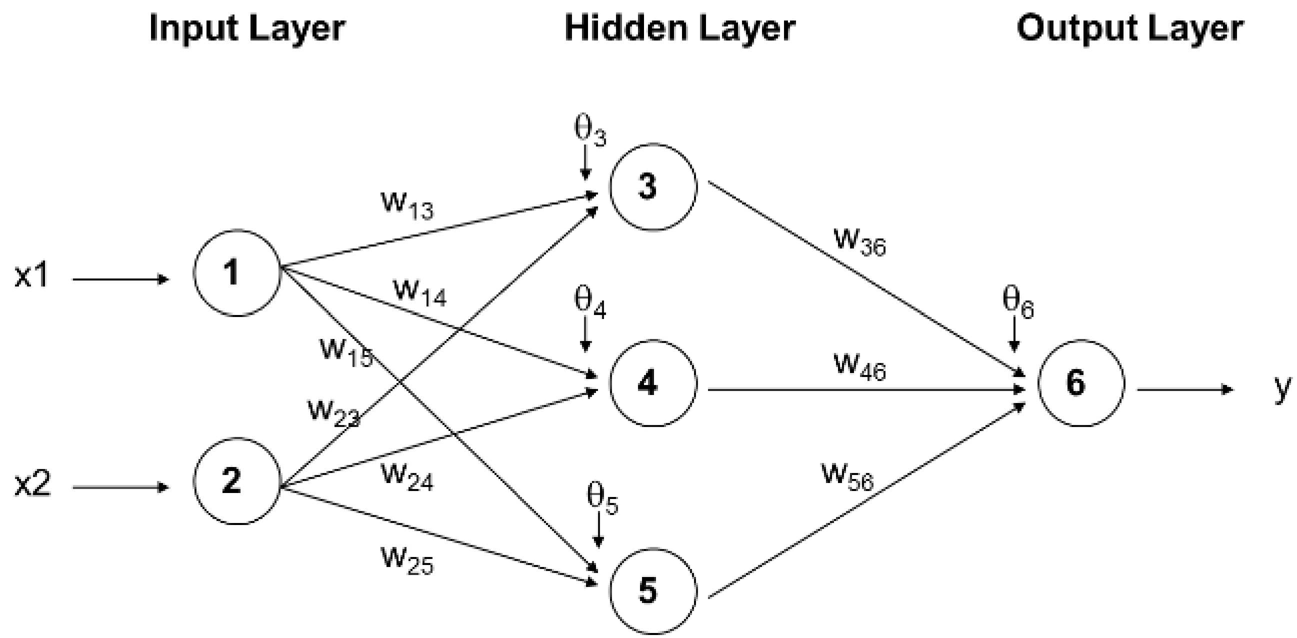
Which derived variables did we see in linear regression for forecasting?

$$y = g_1\,(g_2(\ \ldots g_k(X)\ldots)$$

Multi-layer feed-forward, fully connected
Neural Net Architecture

“node” = derived variable





This example: single hidden layer
Output from one layer is input into next layer

bias = controls contribution of node *j*

Output of node *j* = $\mathrm{g}\left(\theta_j + \sum_{j=1}^{p} w_{ij} x_i\right)$

weights

Activation function
(common: linear, exponential, s-shaped)

Example: consumer acceptance of cheese

Obs.	*Fat Score*	*Salt Score*	*Acceptance*
1	0.2	0.9	1
2	0.1	0.1	0
3	0.2	0.4	0
4	0.2	0.5	0
5	0.4	0.5	1
6	0.3	0.8	1

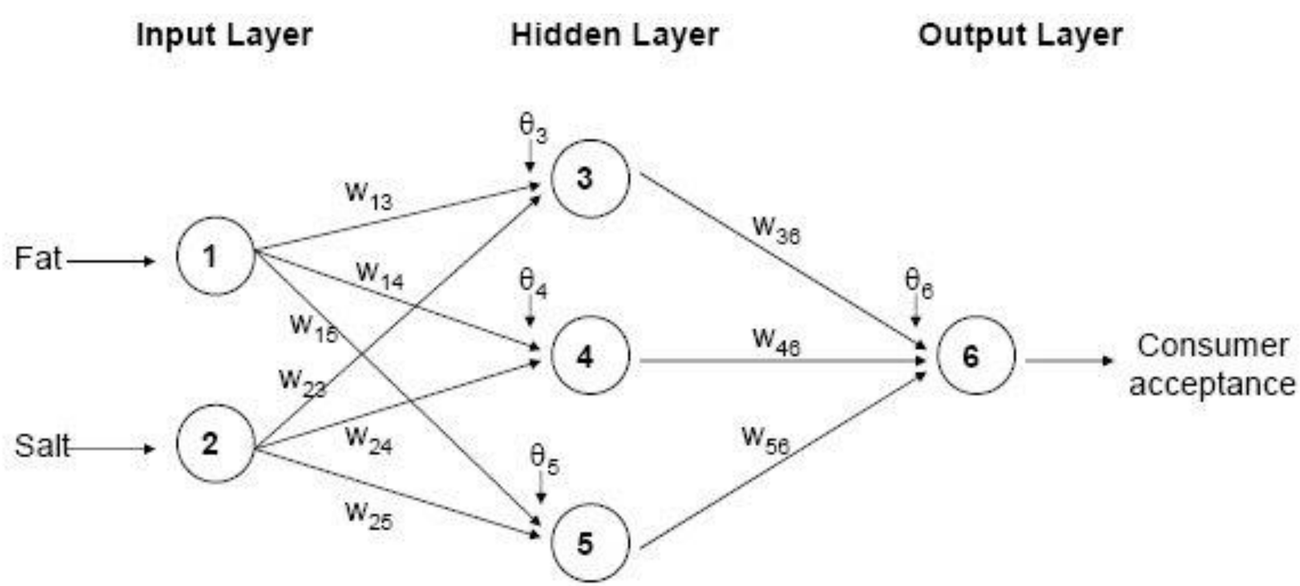
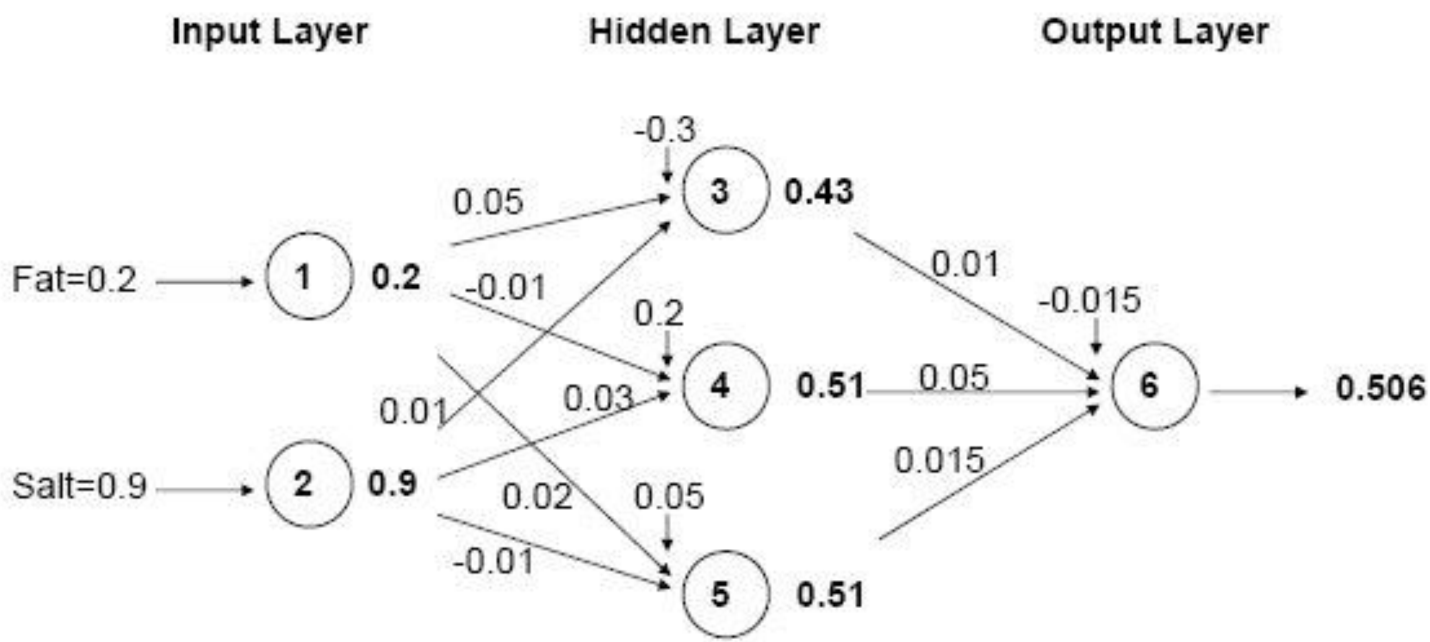


Figure 9.2: Neural network for the tiny example. Circles represent nodes, $w_{i,j}$ on arrows are weights, and θ_j are node bias values.



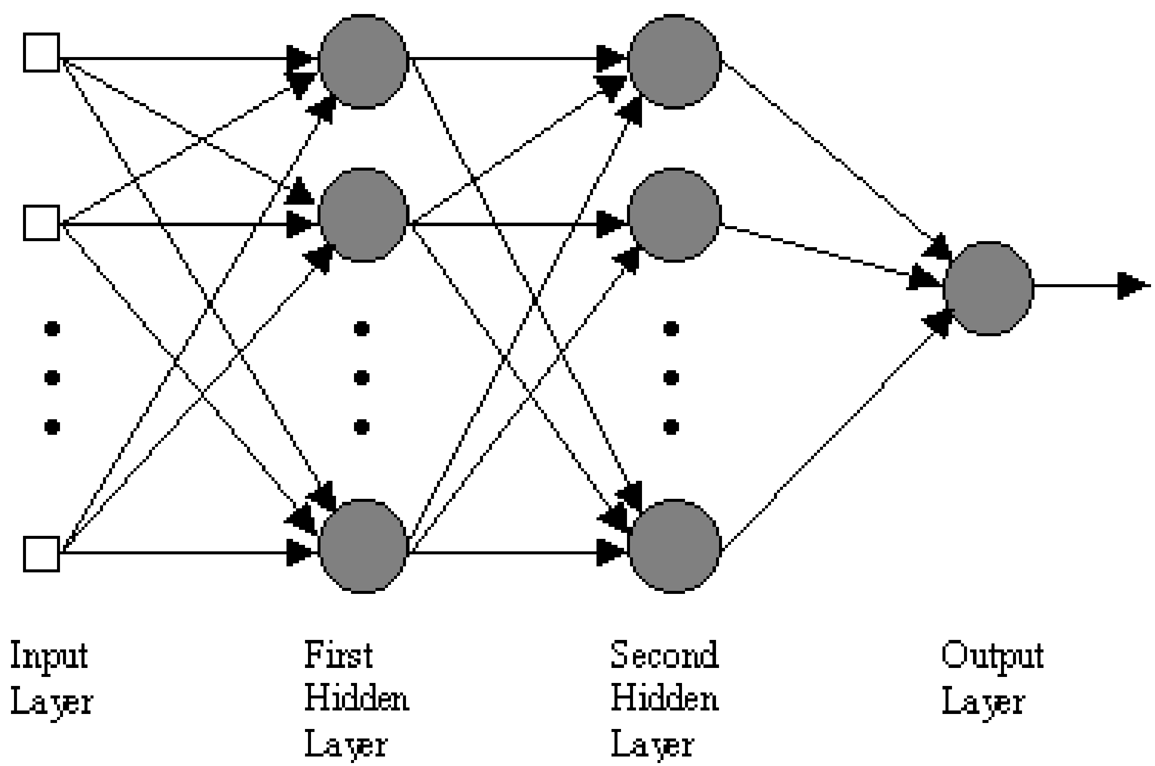
>0.5
Classify as 1

Figure 9.3: Computing node outputs (in boldface type) using the first observation in the tiny example and a logistic function.

$$output_3 = \frac{1}{1 + e^{-[-0.3+(0.05)(0.2)+(0.01)(0.9)]}} = 0.43$$

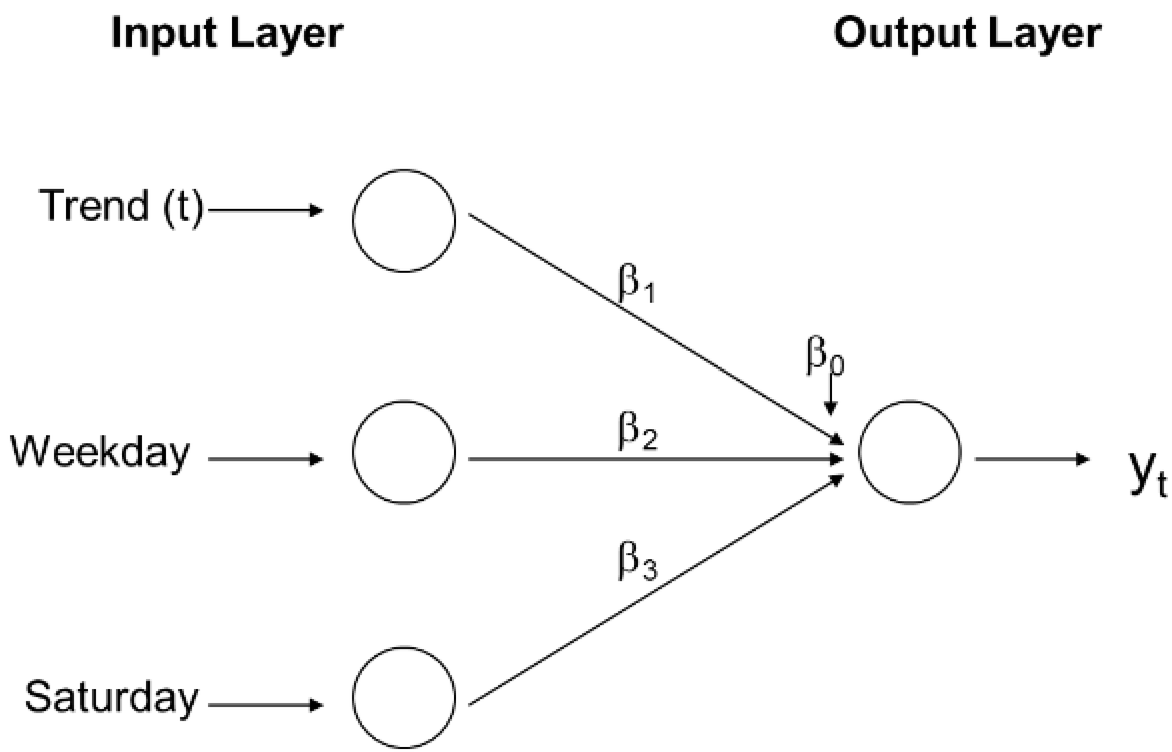
$$output_6 = \frac{1}{1 + e^{-[-0.015+(0.01)(0.43)+(0.05)(0.507)+(0.015)(0.511)]}} = 0.506$$

More layers, more complexity



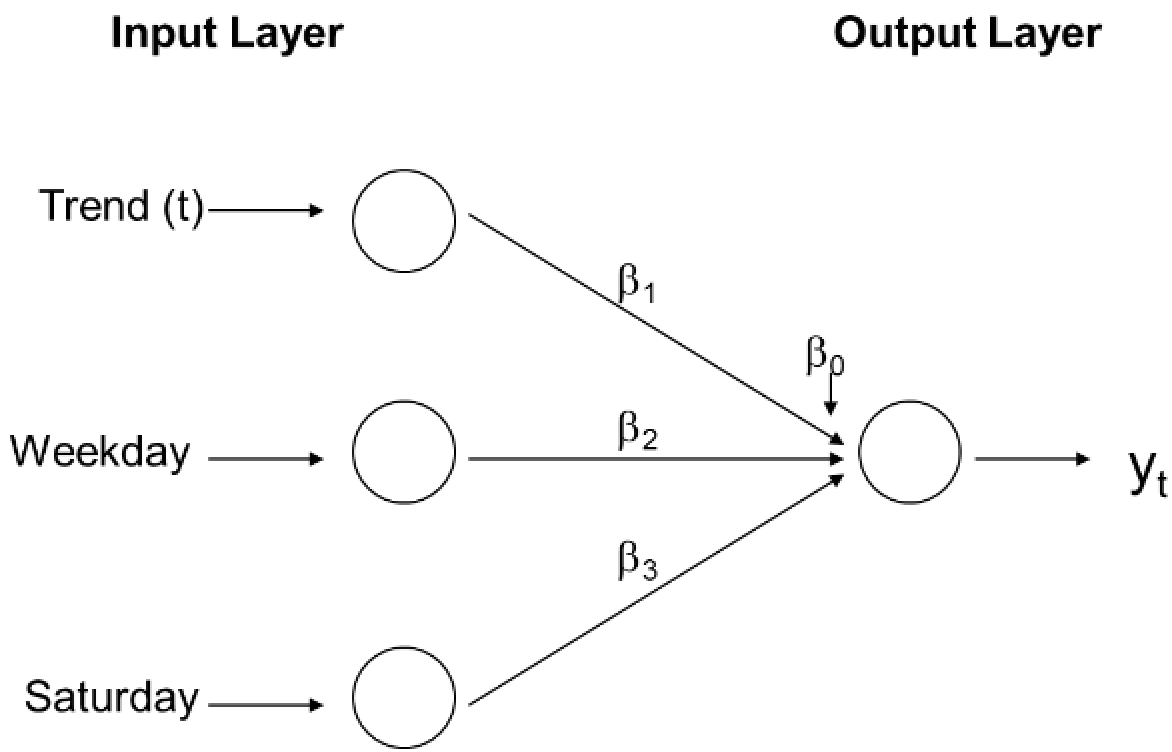
Most popular in forecasting: **single** hidden layer

Example #1: NN with no hidden layers
g= linear activation function



$$Y_t = \beta_0 + \beta_1\, t + \beta_2\, \text{Weekday} + \beta_3\, \text{Saturday}$$

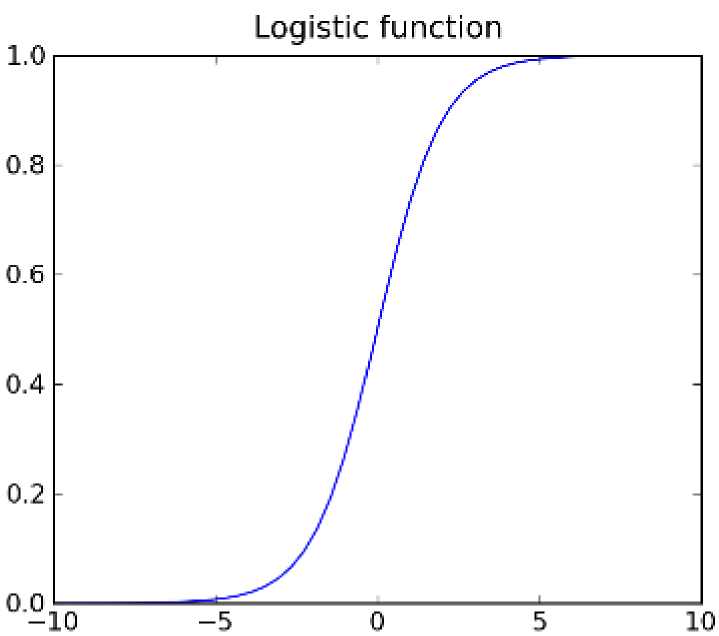
Example #2: NN with no hidden layers g= exponential activation function



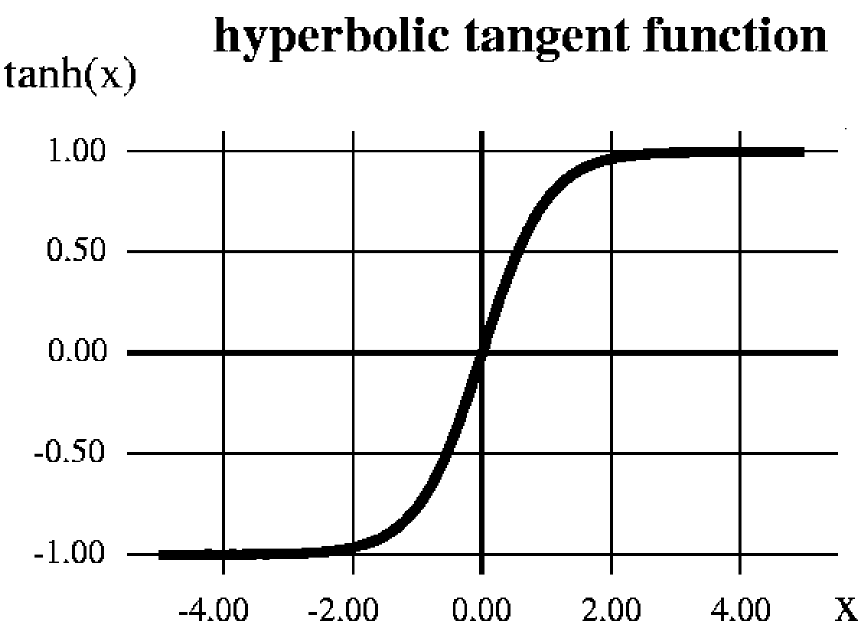
$$\log(Y_t) = \beta_0 + \beta_1 t + \beta_2 \text{ Weekday} + \beta_3 \text{ Saturday}$$

S-shaped (*sigmoidal*) activation functions

"Squash" large and small values
Maintain near-linearity in midrange

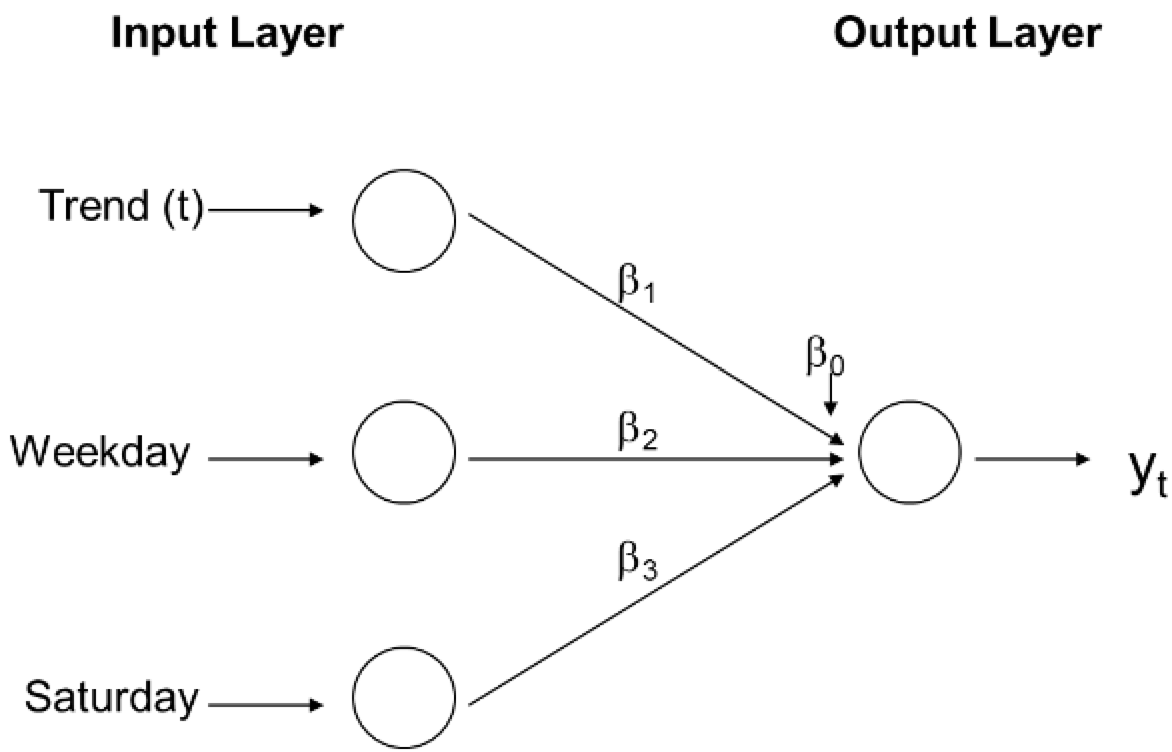


$$g(s) = \frac{1}{1 + e^{-s}}$$



$$g(s) = -1 + \frac{2}{1 + e^{-s}}$$

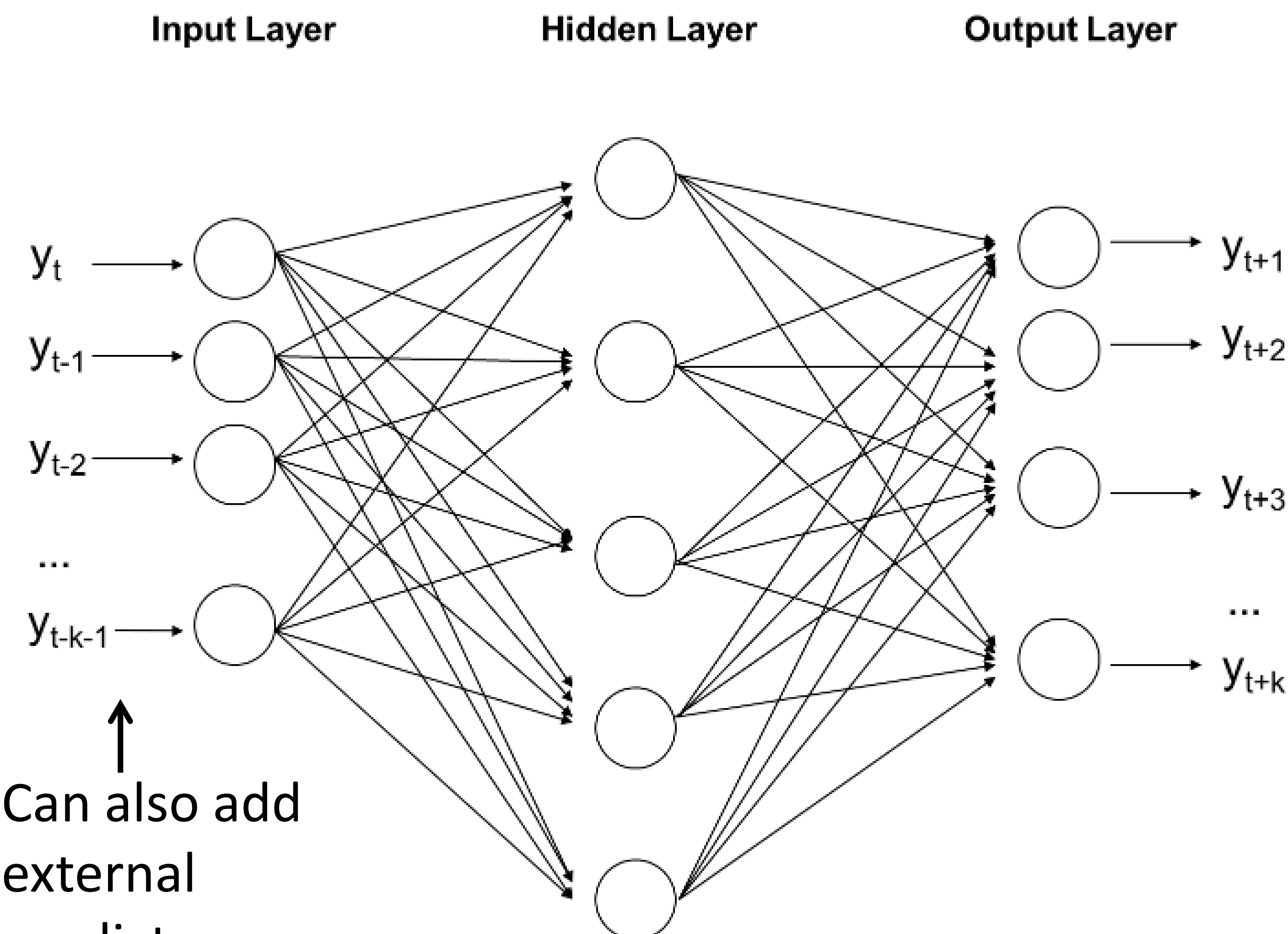
Example #3: NN with no hidden layers
g= logit activation function



$$\text{logit}(Y_t = 1) = \beta_0 + \beta_1\, t + \beta_2\, \text{Weekday} + \beta_3\, \text{Saturday}$$

NN architecture for roll-forward **forecasting**

(single hidden layer example)



Can also add external predictors

Behind the scenes:

Training the network (weight estimation)

Iterative error minimization: node-specific errors used for updating weights

$$w_{ij}^{new} = w_{ij}^{old} + lr \times (err_j) \times output_i$$

Learning Rate in [0,1]

Backpropagation: compute errors from last layer to first

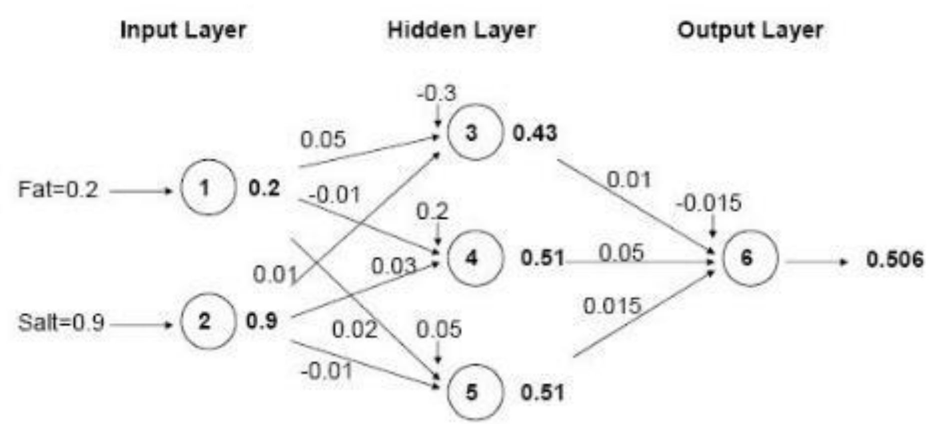
Output node

$$err_j = out_j(1\text{-}out_j)(y_j\text{-}out_j)$$

Hidden node

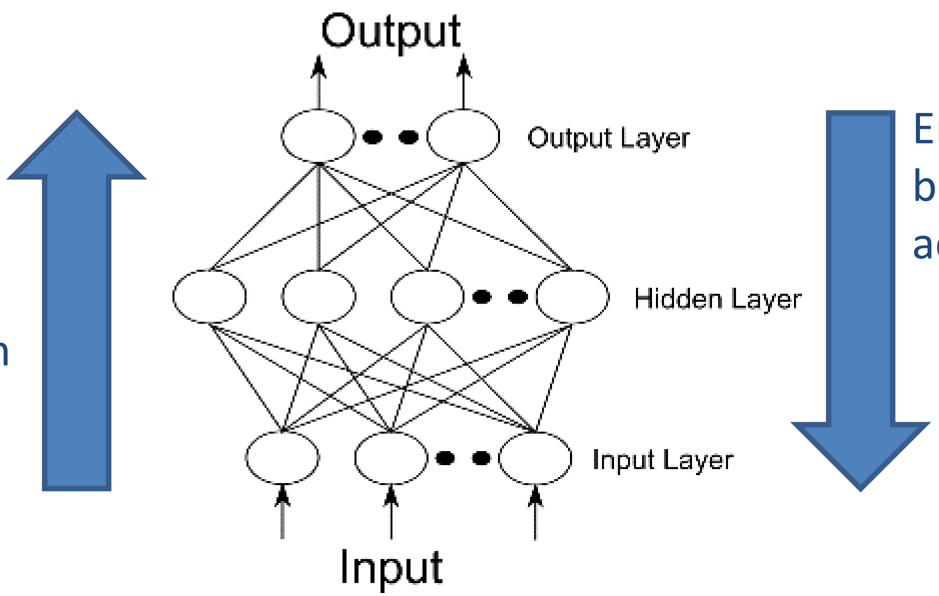
$$err_j = out_j(1\text{-}out_j)\sum_k err_k\, w_{jk}^{old}$$

Initialization: random values in [-0.05, +0.05] = random prediction



Backpropagation is the most popular error minimization algorithm in NN software

Activity from input flows forward



Error signal flows backward to adjust weights

Its greatest strength is in non-linear solutions to ill-defined problems

"**momentum**" keeps weights put

But, it can get stuck in local minima and can be slow

$$w_{ij}^{new} = w_{ij}^{old} + (1-\mathbf{m})\, lr \times (err_j) \times output_i + \mathbf{m}(w_{ij}^{old} - w_{ij}^{older})$$

Backpropagation: two options

Case updating (XLMiner)

- Weights updated after each record is run through the network
- Completion of all records through the network is one **epoch** (=sweep or iteration)
- After one epoch is completed, return to first record and repeat the process

Batch Updating

- All records in the training set are fed to the network before updating takes place
- In this case, the error used for updating is the sum of all errors from all records

When to stop

- When weights change very little from one iteration to the next
- When the misclassification rate reaches a required threshold
- When a limit on runs is reached

Danger: Overfitting

With sufficient iterations, neural nets can easily overfit the training data

To avoid overfitting:

- Track error in validation data
- Limit iterations
- Limit complexity of network
- Cross-validation

REQUIRED USER INPUT

#1: Choose predictors
#2: Pre-process data
#3: Specify network architecture
#4: Specify algorithm parameters

For binary forecasts
#5: Determine cutoff value

Step #1: Choice of predictors

NN highly dependent on quality of predictors

#2: Pre-processing

Numerical forecast:

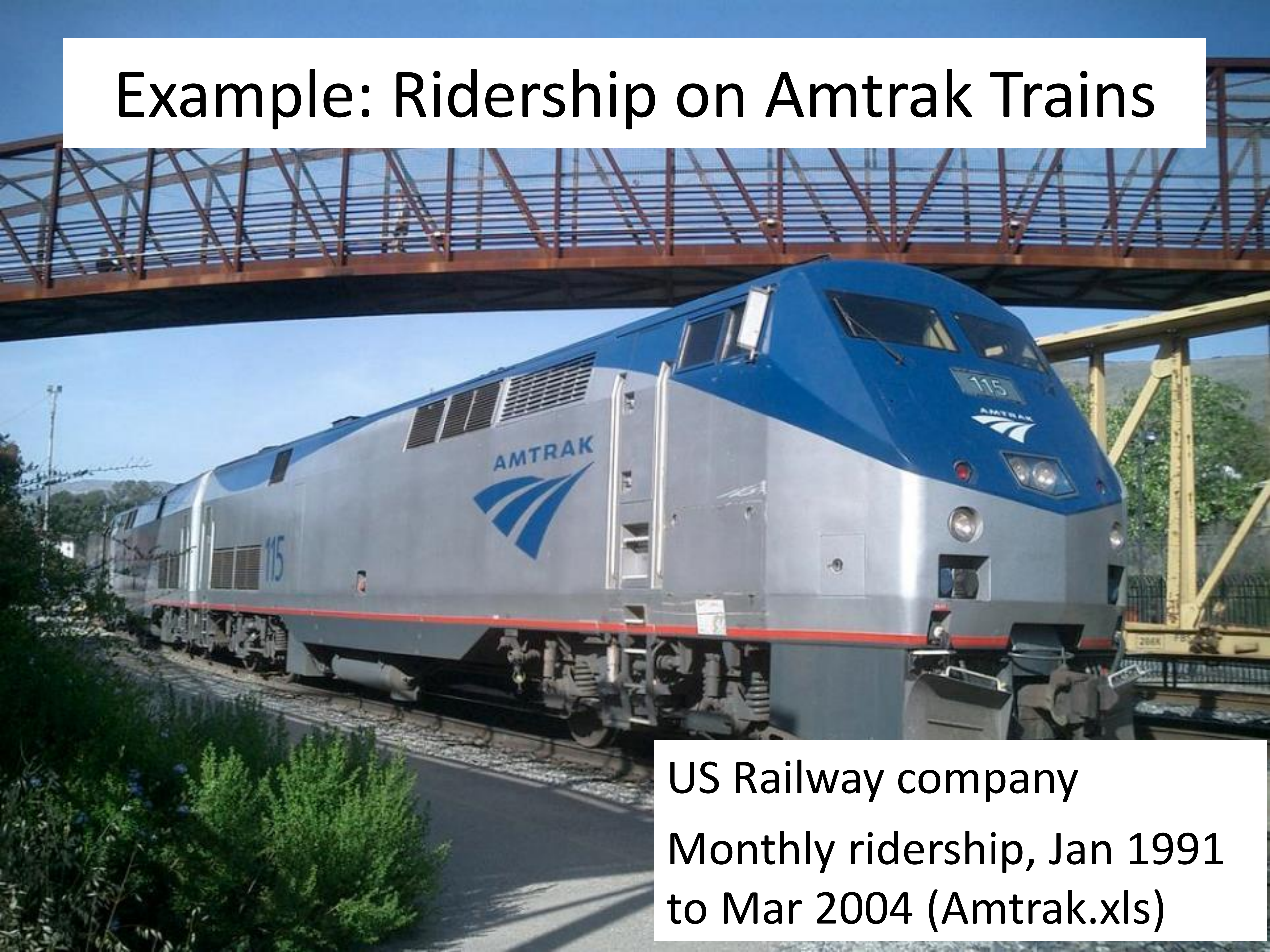
- **Transform** (e.g., log) skewed variables
- **De-seasonalize, de-trend**
- **Scale** to [0,1] (logistic) or [-1,1] (tanh)

controversial

Binary forecast:

- Create one **dummy** variable

Example: Ridership on Amtrak Trains



US Railway company

Monthly ridership, Jan 1991 to Mar 2004 (Amtrak.xls)

Amtrak: predictor choice & pre-processing

Need to account for seasonality

Several options:

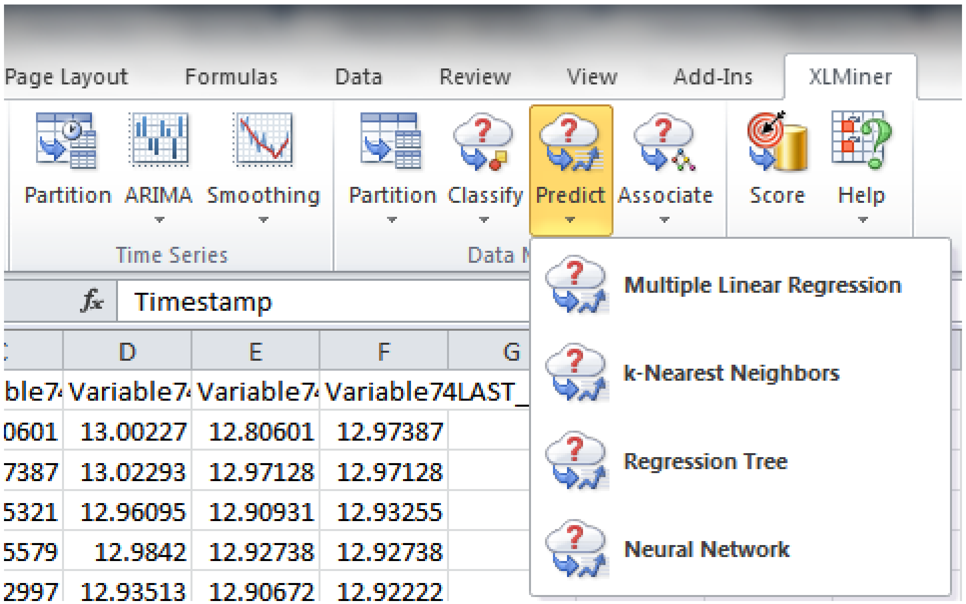
- Create 12 lags
- Create 11 dummies
- De-seasonalize the series first

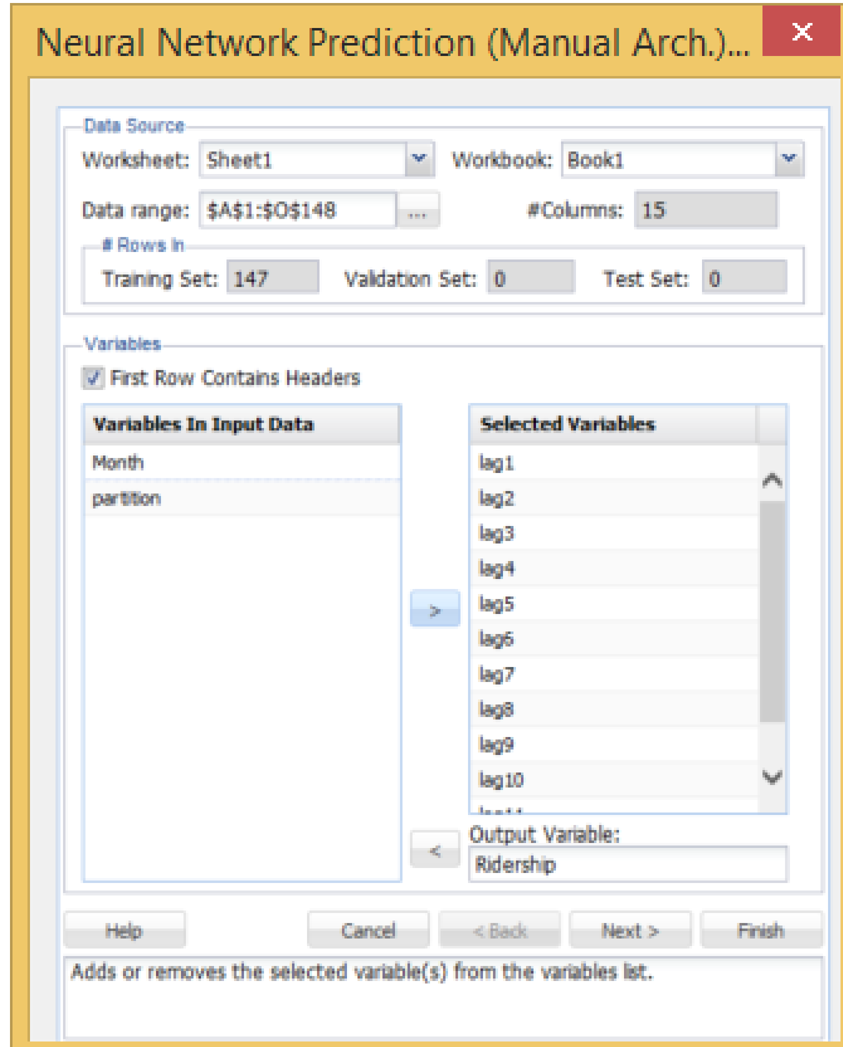
Create 12 lags

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	**Month**	**Ridership**	**lag1**	**lag2**	**lag3**	**lag4**	**lag5**	**lag6**	**lag7**	**lag8**	**lag9**	**lag10**	**lag11**	**lag12**
2	Jan-92	1615	1814	1676	1725	1596	2013	1940	1862	1975	1812	1973	1621	1709
3	Feb-92	1557	1615	1814	1676	1725	1596	2013	1940	1862	1975	1812	1973	1621
4	Mar-92	1891	1557	1615	1814	1676	1725	1596	2013	1940	1862	1975	1812	1973
5	Apr-92	1956	1891	1557	1615	1814	1676	1725	1596	2013	1940	1862	1975	1812
6	May-92	1885	1956	1891	1557	1615	1814	1676	1725	1596	2013	1940	1862	1975
7	Jun-92	1623	1885	1956	1891	1557	1615	1814	1676	1725	1596	2013	1940	1862
8	Jul-92	1903	1623	1885	1956	1891	1557	1615	1814	1676	1725	1596	2013	1940
9	Aug-92	1997	1903	1623	1885	1956	1891	1557	1615	1814	1676	1725	1596	2013
10	Sep-92	1704	1997	1903	1623	1885	1956	1891	1557	1615	1814	1676	1725	1596
11	Oct-92	1810	1704	1997	1903	1623	1885	1956	1891	1557	1615	1814	1676	1725

Data Partition: Validation period = last 12 months

NN in XLMiner





Also in Classification menu!

Choose “Manual Network” to set the architecture

#3: Specify network architecture

Number of hidden layers

Most popular: single hidden layer

Number of nodes in hidden layer(s)

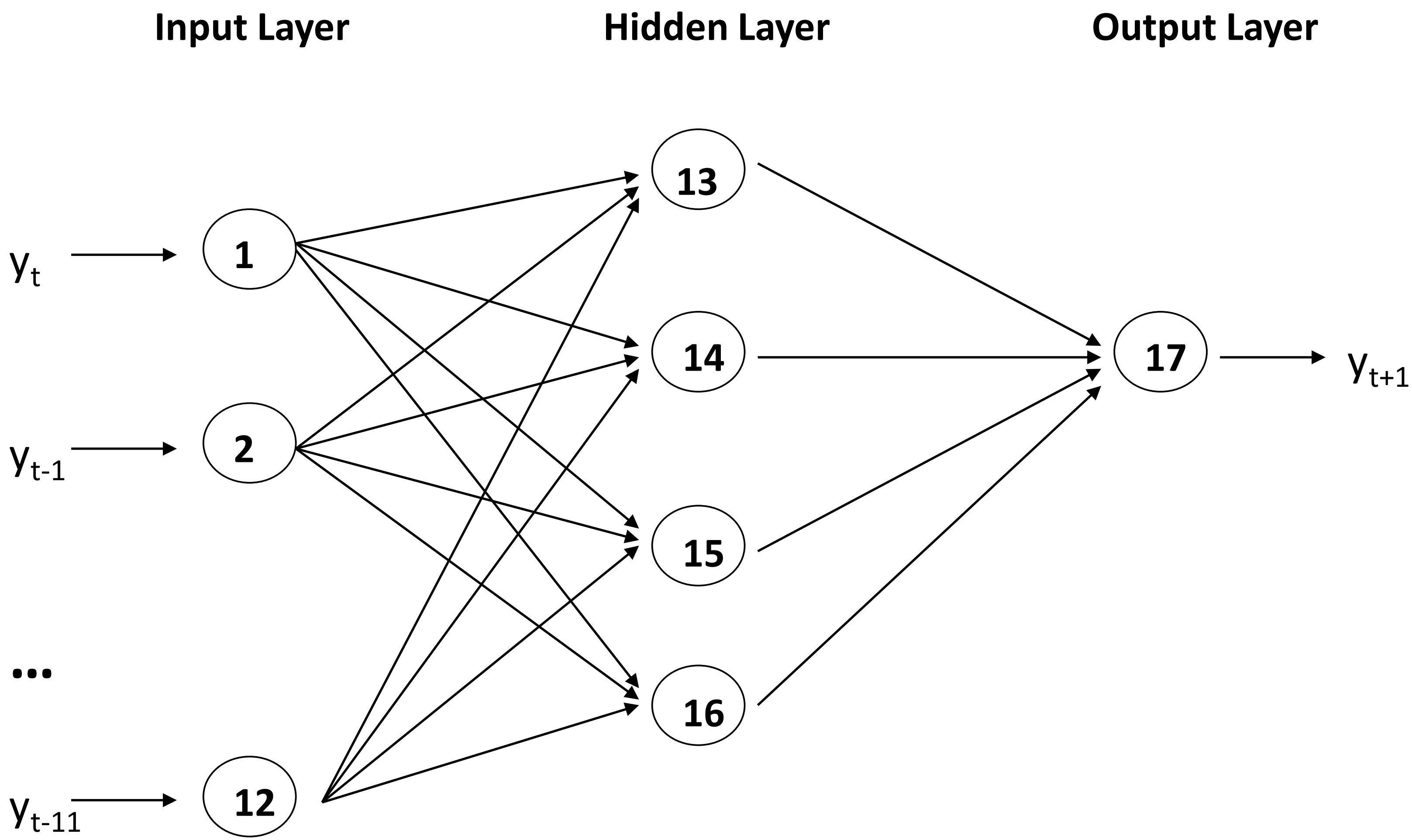
More nodes capture complexity, but increase chances of overfitting

Number of output nodes

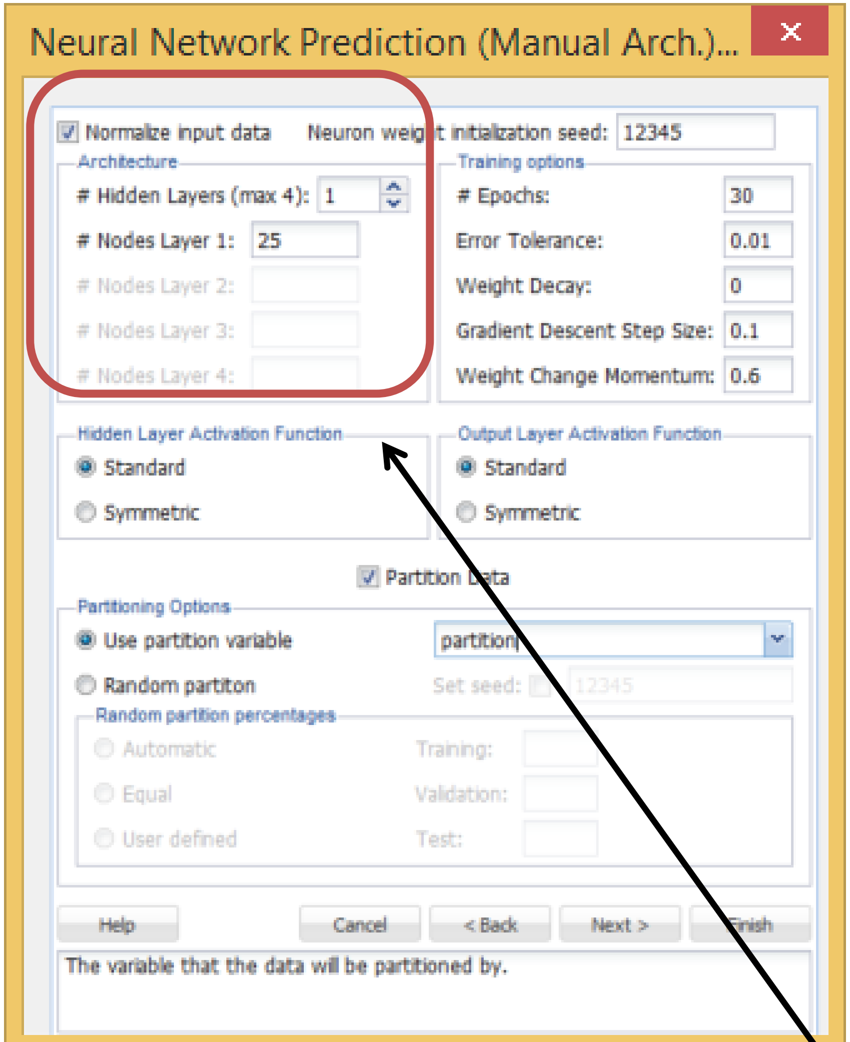
For classification: one node per class (in binary case can also use only one)

For numerical prediction: use one

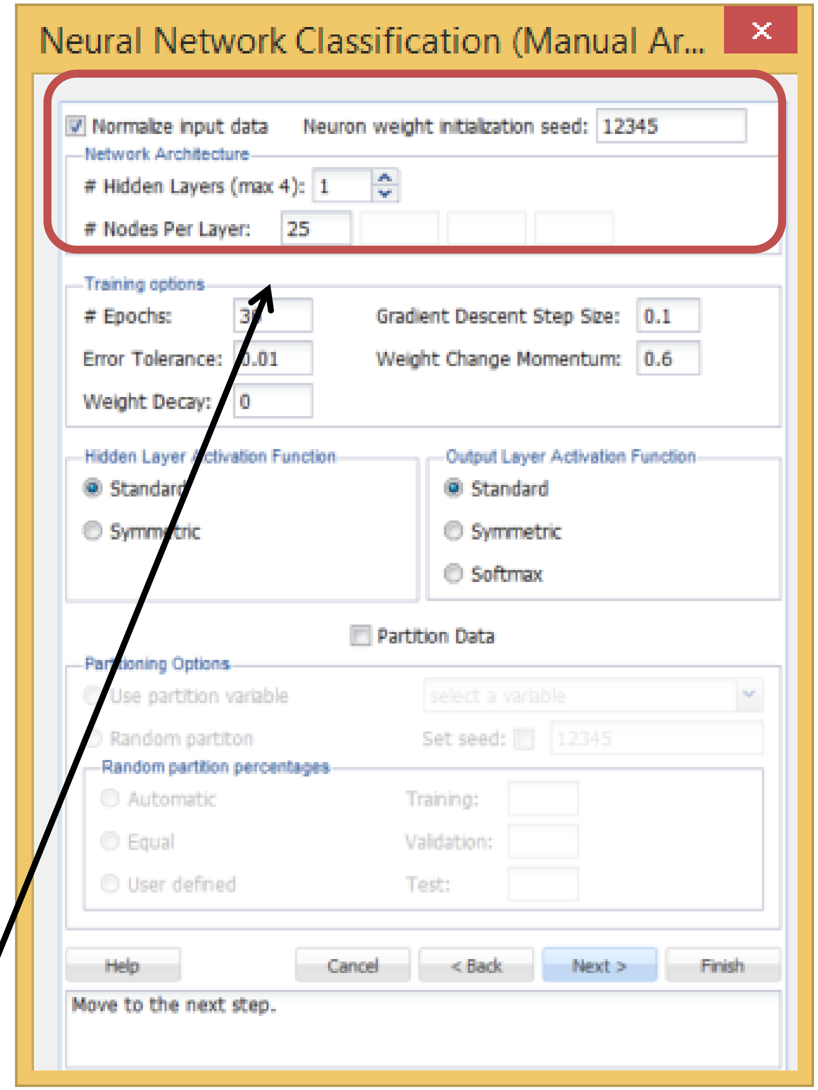
One-step-ahead forecasts
12-lag network; one hidden layer (4 nodes)



Numerical forecast



Binary forecast



Choice of network architecture

#4: Specify algorithm parameters

“Learning Rate” (lr)

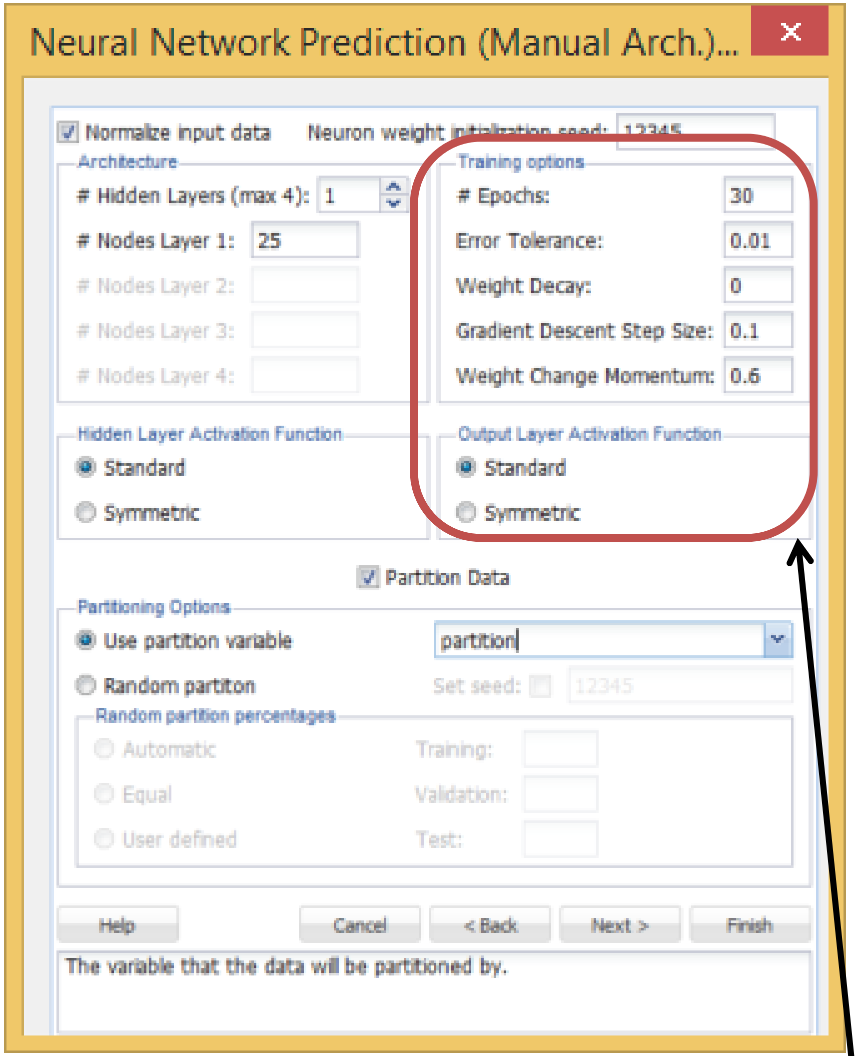
Low values “down-weight” the new information from errors at each iteration; This slows learning, but reduces tendency to overfit to local structure

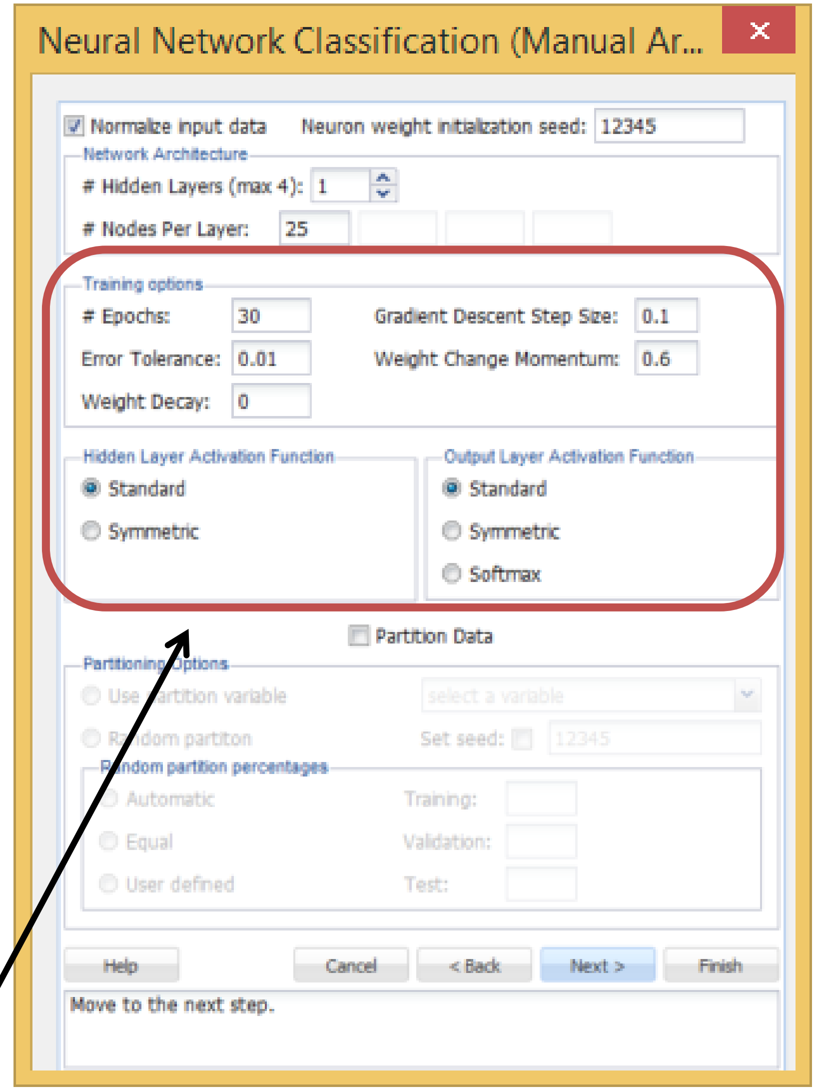
Momentum (“weight change momentum”)

High values keep parameters changing in same direction as previous iteration; helps avoid overfitting to local structure, but also slows learning

Numerical forecast

Binary forecast





Choice of algorithm parameters

#5: Determine cutoff value (for binary forecasts)

Cutoff on probability to obtain binary classification

Tendency of probabilities to cluster around 0.5

Use validation period to choose cutoff

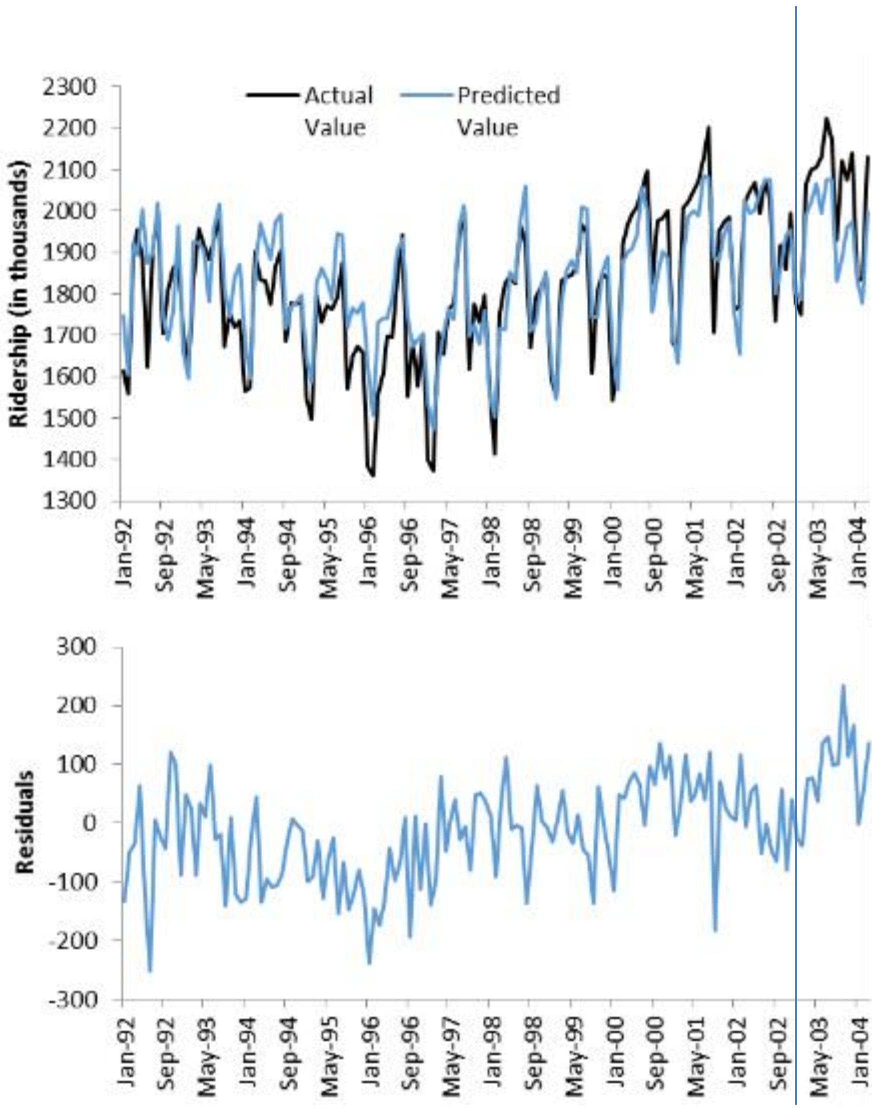
Output for Amtrak Data

Training Data Scoring - Summary Report

Total sum of squared errors	RMS Error	Average Error
480587.1499	59.66494	-22.519

Validation Data Scoring - Summary Report

Total sum of squared errors	RMS Error	Average Error
92607.91465	87.84831	108.9948



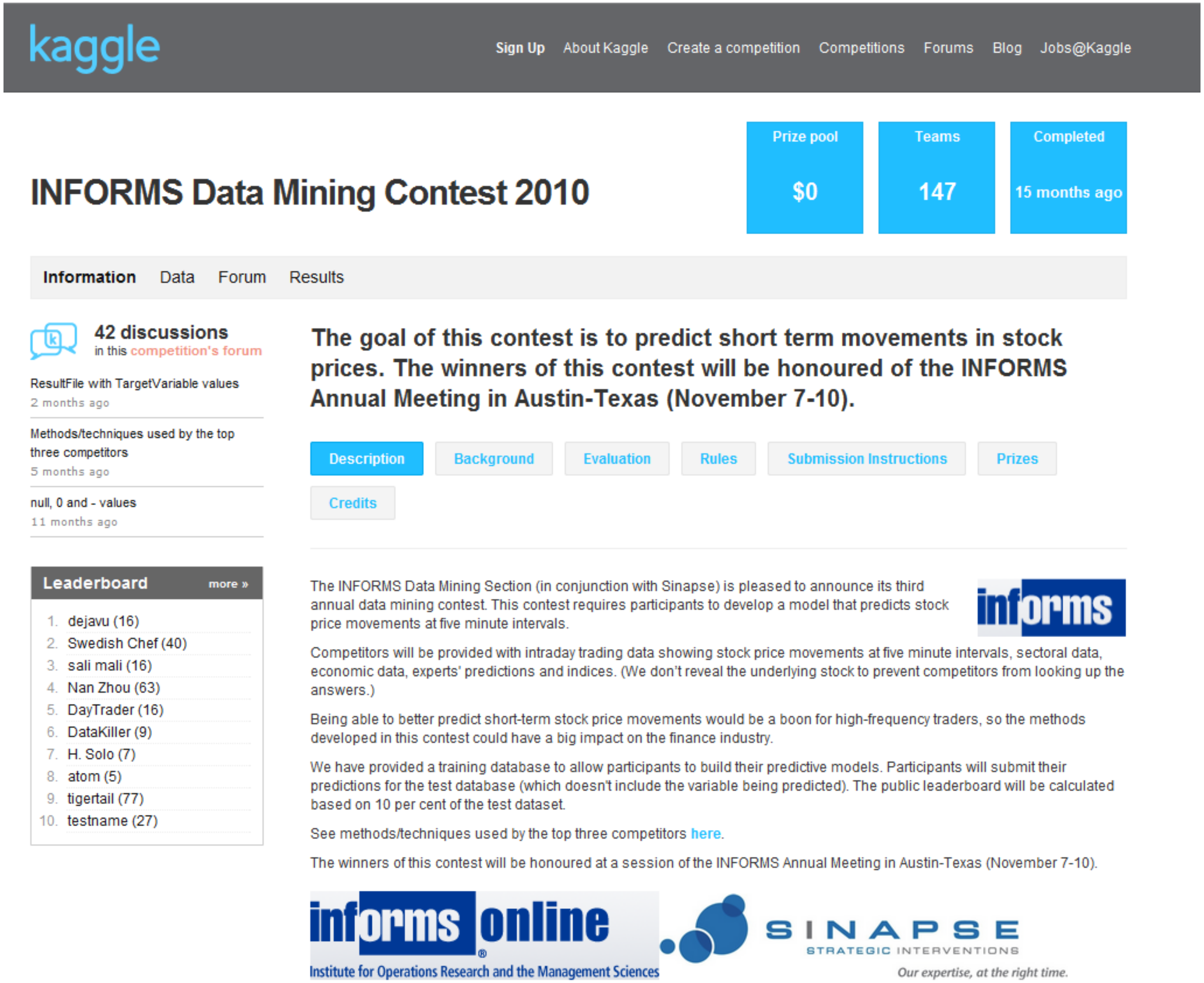
Neural Net Output: Weights

Inter-Layer Connections Weights

	Input Layer												
Hidden Layer 1	lag1	lag2	lag3	lag4	lag5	lag6	lag7	lag8	lag9	lag10	lag11	lag12	Bias
Neuron 1	0.094363	0.089113	-0.28193	0.186411	0.112626	-0.01388	0.084388	0.174799	-0.138412084	0.110711	-0.10088	0.050513	0.113553
Neuron 2	-0.13106	-0.11478	0.03908	-0.16469	0.18817	0.094872	-0.17799	0.011372	0.107325093	-0.09091	0.026824	-0.08468	-0.23351
Neuron 3	-0.05192	0.100877	0.187221	0.107114	-0.15845	0.010803	0.212407	0.100309	-0.039006765	0.028133	0.001924	-0.1263	-0.15984
Neuron 4	-0.17678	0.098353	-0.12386	-0.13332	0.152321	-0.00064	-0.10709	-0.19652	0.015738799	-0.04464	-0.19617	0.134691	-0.06318
Neuron 5	0.136766	0.09493	0.154444	-0.03275	-0.18182	-0.12637	0.129922	0.05468	0.184708802	-0.11113	0.111348	-0.02541	-0.21254
Neuron 6	-0.16624	-0.15226	0.140214	-0.05597	-0.14339	0.23508	-0.01131	0.023592	-0.082750668	0.128931	-0.06134	-0.1881	0.082961
Neuron 7	-0.05989	0.039449	0.130059	0.150713	0.143888	0.012385	0.054855	-0.19699	-0.070875592	-0.1214	0.147518	-0.0777	-0.18328
Neuron 8	0.207265	0.133408	0.009683	0.054705	0.064394	-0.20669	0.056141	0.049528	-0.098872477	-0.03393	0.170564	0.275894	-0.04292
Neuron 9	-0.2116	0.205475	-0.10094	-0.07188	0.208448	0.183452	0.113602	-0.20405	-0.127668703	0.194025	-0.10929	-0.08565	-0.02061
Neuron 10	-0.10531	-0.13455	-0.01836	-0.0786	-0.18627	-0.13996	-0.19312	0.086334	-0.013205001	-0.1613	-0.05508	0.159183	0.068046
Neuron 11	0.143901	-0.12167	0.137977	-0.19178	0.117482	0.110498	-0.05979	-0.11244	0.082764924	0.007036	0.100683	-0.17005	-0.04769
Neuron 12	0.0696	-0.16996	-0.20068	0.067829	-0.00152	-0.08275	0.145232	0.055731	0.178512226	-0.04142	0.171361	0.198928	0.001946
Neuron 13	-0.16783	-0.03541	0.156842	0.067154	0.104142	-0.09283	-0.0083	-0.15163	-0.148568254	0.013667	-0.14446	0.107524	0.127401
Neuron 14	-0.19345	-0.11864	-0.03495	-0.18377	-0.12617	0.005803	-0.2228	0.027523	-0.133374249	0.042096	0.033819	-0.29488	-0.23379
Neuron 15	-0.27131	0.071415	0.059467	0.141694	-0.10162	0.213105	-0.01848	0.166539	0.167778211	0.166755	-0.17243	-0.16167	-0.08627
Neuron 16	0.014701	-0.03312	-0.11008	0.092236	0.122182	0.266399	-0.07818	-0.04183	-0.085435642	-0.07044	0.15316	0.063612	-0.21804
Neuron 17	-0.00822	-0.07925	-0.18449	-0.0626	-0.10022	-0.33199	0.028086	0.076626	0.034647321	0.078682	0.066142	0.347728	0.010336
Neuron 18	0.236285	-0.02065	-0.00649	0.111023	-0.0282	-0.09795	-0.10312	-0.16528	0.038761721	-0.03257	0.107128	-0.08084	0.067993
Neuron 19	-0.04639	0.046747	-0.03122	0.083796	-0.106	0.127822	0.064177	-0.11659	-0.000774462	0.074779	-0.09748	-0.24961	-0.0971
Neuron 20	0.149388	0.091825	0.049603	0.137897	0.147355	0.02578	-0.06084	-0.082	0.078952566	-0.05911	-0.12131	0.065262	0.191851
Neuron 21	0.184703	0.046131	-0.06781	-0.10849	0.001512	-0.06847	-0.00177	-0.12078	-0.18104508	0.099871	0.130226	0.244464	-0.0557
Neuron 22	-0.10019	0.10691	-0.00531	0.084597	-0.04283	-0.02033	-0.07095	-0.01539	0.052016621	-0.09284	-0.07568	0.085528	-0.03866
Neuron 23	0.00625	-0.17297	-0.09192	-0.00371	-0.18202	-0.11592	-0.03354	-0.13492	0.167417488	-0.20521	0.073613	0.201603	0.030787
Neuron 24	0.103663	0.182899	-0.14429	-0.18449	-0.14227	0.069527	0.14129	-0.12456	0.160120911	-0.03184	-0.00566	-0.14792	-0.12782
Neuron 25	0.102095	-0.11403	0.070987	-0.00605	0.06183	-0.21457	0.295604	-0.00591	-0.241719907	-0.01579	-0.04408	0.114197	0.206415

	Hidden Layer 1													
Output Layer	Neuron 1	Neuron 2	Neuron 3	Neuron 4	Neuron 5	Neuron 6	Neuron 7	Neuron 8	Neuron 9	Neuron 10	Neuron 11	Neuron 12	Neuron 13	N
Response	0.981875	-0.71218	0.074669	-0.36605	-0.37494	-0.61346	-0.44477	0.926251	-0.187995762	-0.23692	-0.13581	0.186516	-0.30472	

CASE 9.3 IN TEXTBOOK:
FORECASTING STOCK PRICE MOVEMENTS



kaggle

Sign Up About Kaggle Create a competition Competitions Forums Blog Jobs@Kaggle

INFORMS Data Mining Contest 2010

Prize pool	Teams	Completed
$0	147	15 months ago

Information Data Forum Results

42 discussions
in this competition's forum

ResultFile with TargetVariable values
2 months ago

Methods/techniques used by the top three competitors
5 months ago

null, 0 and - values
11 months ago

Leaderboard more »

1. dejavu (16)
2. Swedish Chef (40)
3. sali mali (16)
4. Nan Zhou (63)
5. DayTrader (16)
6. DataKiller (9)
7. H. Solo (7)
8. atom (5)
9. tigertail (77)
10. testname (27)

The goal of this contest is to predict short term movements in stock prices. The winners of this contest will be honoured of the INFORMS Annual Meeting in Austin-Texas (November 7-10).

Description Background Evaluation Rules Submission Instructions Prizes Credits

The INFORMS Data Mining Section (in conjunction with Sinapse) is pleased to announce its third annual data mining contest. This contest requires participants to develop a model that predicts stock price movements at five minute intervals.

informs

Competitors will be provided with intraday trading data showing stock price movements at five minute intervals, sectoral data, economic data, experts' predictions and indices. (We don't reveal the underlying stock to prevent competitors from looking up the answers.)

Being able to better predict short-term stock price movements would be a boon for high-frequency traders, so the methods developed in this contest could have a big impact on the finance industry.

We have provided a training database to allow participants to build their predictive models. Participants will submit their predictions for the test database (which doesn't include the variable being predicted). The public leaderboard will be calculated based on 10 per cent of the test dataset.

See methods/techniques used by the top three competitors here.

The winners of this contest will be honoured at a session of the INFORMS Annual Meeting in Austin-Texas (November 7-10).

informs online
Institute for Operations Research and the Management Sciences

SINAPSE
STRATEGIC INTERVENTIONS
Our expertise, at the right time.

Is it possible?

http://www.kaggle.com/c/informs2010/forums/t/38/is-it-possible-to-predict-stock-price-movements-at-five-minute-intervals

Data: http://kaggle.com/c/informs2010/data

kaggle

ProfG About Kaggle Create a competition Competitions Forums Blog Jobs@Kaggle

Thank you for accepting the rules. You can now download data files.

INFORMS Data Mining Contest 2010

Prize pool	Teams	Completed
$0	147	15 months ago

Information **Data** Submissions Forum Results

42 discussions
in this competition's forum

ResultFile with TargetVariable values
2 months ago

Methods/techniques used by the top three competitors
5 months ago

null, 0 and - values
11 months ago

Leaderboard more »

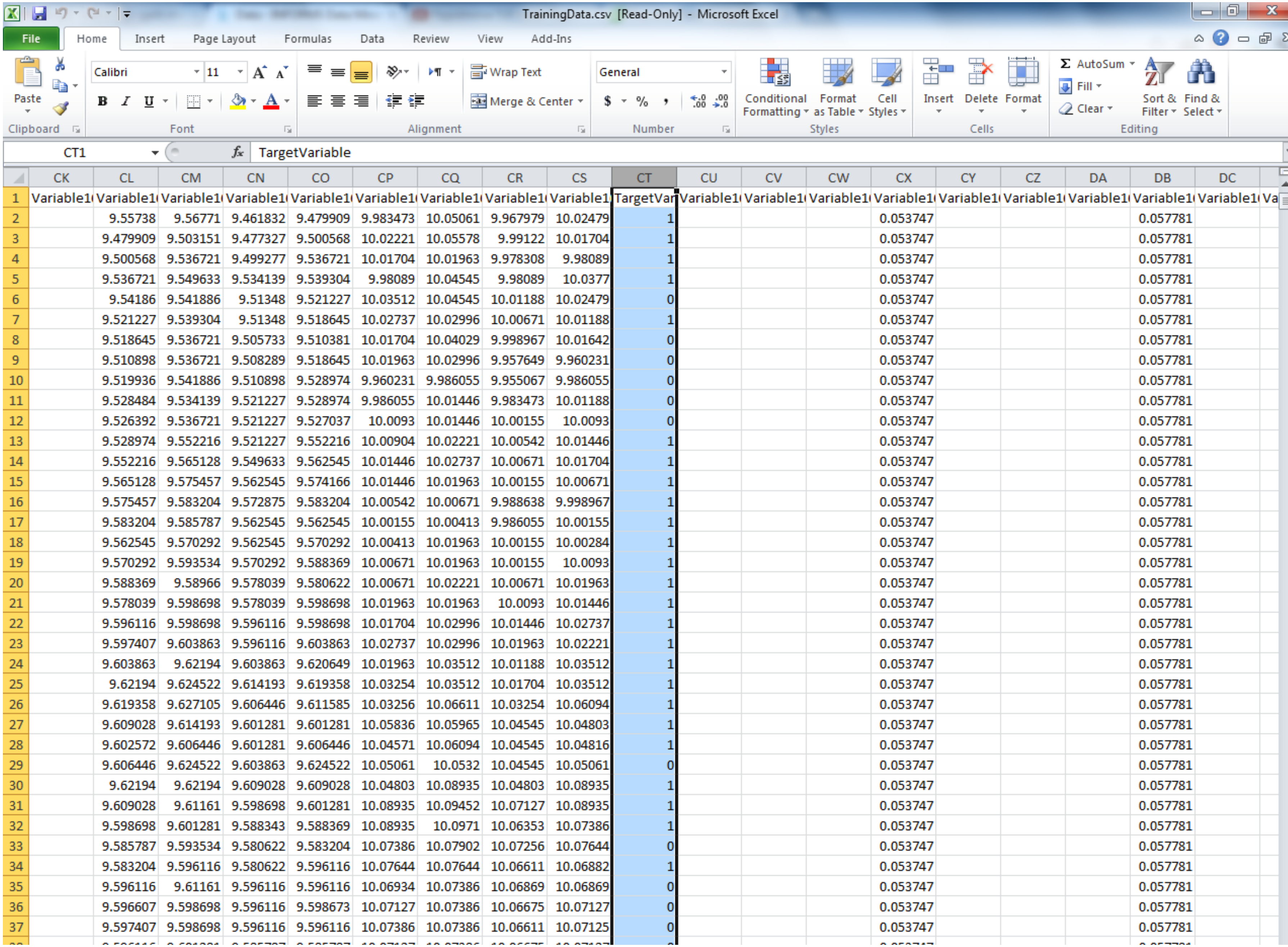
1. dejavu (16)
2. Swedish Chef (40)
3. sali mali (16)

INFORMS Data Mining Contest 2010 Data Files

File Name	Available Formats
TestData	.zip (4.49 mb)
TrainingData	.zip (9.55 mb)

The datasets are time series, which include 609 explanatory variables. The explanatory variables include stock prices, sectoral data, economic data, experts' predictions and indexes. These variables are named Variable... in the datasets. The first column in each file is the timestamp. The binary variable to be predicted is called TargetVariable.

The training dataset contains 5922 observations. The test dataset contains 2539 observations and follows chronologically from the training dataset. All observations are taken at 5 minutes interval.



	CK	CL	CM	CN	CO	CP	CQ	CR	CS	CT	CU	CV	CW	CX	CY	CZ	DA	DB	DC	
1	Variable1	Variable1	Variable1	Variable1	Variable1	Variable1	Variable1	Variable1	Variable1	TargetVar	Variable1	Variable1	Variable1	Variable1	Variable1	Variable1	Variable1	Variable1	Variable1	Va
2		9.55738	9.56771	9.461832	9.479909	9.983473	10.05061	9.967979	10.02479	1				0.053747				0.057781		
3		9.479909	9.503151	9.477327	9.500568	10.02221	10.05578	9.99122	10.01704	1				0.053747				0.057781		
4		9.500568	9.536721	9.499277	9.536721	10.01704	10.01963	9.978308	9.98089	1				0.053747				0.057781		
5		9.536721	9.549633	9.534139	9.539304	9.98089	10.04545	9.98089	10.0377	1				0.053747				0.057781		
6		9.54186	9.541886	9.51348	9.521227	10.03512	10.04545	10.01188	10.02479	0				0.053747				0.057781		
7		9.521227	9.539304	9.51348	9.518645	10.02737	10.02996	10.00671	10.01188	1				0.053747				0.057781		
8		9.518645	9.536721	9.505733	9.510381	10.01704	10.04029	9.998967	10.01642	0				0.053747				0.057781		
9		9.510898	9.536721	9.508289	9.518645	10.01963	10.02996	9.957649	9.960231	0				0.053747				0.057781		
10		9.519936	9.541886	9.510898	9.528974	9.960231	9.986055	9.955067	9.986055	0				0.053747				0.057781		
11		9.528484	9.534139	9.521227	9.528974	9.986055	10.01446	9.983473	10.01188	0				0.053747				0.057781		
12		9.526392	9.536721	9.521227	9.527037	10.0093	10.01446	10.00155	10.0093	0				0.053747				0.057781		
13		9.528974	9.552216	9.521227	9.552216	10.00904	10.02221	10.00542	10.01446	1				0.053747				0.057781		
14		9.552216	9.565128	9.549633	9.562545	10.01446	10.02737	10.00671	10.01704	1				0.053747				0.057781		
15		9.565128	9.575457	9.562545	9.574166	10.01446	10.01963	10.00155	10.00671	1				0.053747				0.057781		
16		9.575457	9.583204	9.572875	9.583204	10.00542	10.00671	9.988638	9.998967	1				0.053747				0.057781		
17		9.583204	9.585787	9.562545	9.562545	10.00155	10.00413	9.986055	10.00155	1				0.053747				0.057781		
18		9.562545	9.570292	9.562545	9.570292	10.00413	10.01963	10.00155	10.00284	1				0.053747				0.057781		
19		9.570292	9.593534	9.570292	9.588369	10.00671	10.01963	10.00155	10.0093	1				0.053747				0.057781		
20		9.588369	9.58966	9.578039	9.580622	10.00671	10.02221	10.00671	10.01963	1				0.053747				0.057781		
21		9.578039	9.598698	9.578039	9.598698	10.01963	10.01963	10.0093	10.01446	1				0.053747				0.057781		
22		9.596116	9.598698	9.596116	9.598698	10.01704	10.02996	10.01446	10.02737	1				0.053747				0.057781		
23		9.597407	9.603863	9.596116	9.603863	10.02737	10.02996	10.01963	10.02221	1				0.053747				0.057781		
24		9.603863	9.62194	9.603863	9.620649	10.01963	10.03512	10.01188	10.03512	1				0.053747				0.057781		
25		9.62194	9.624522	9.614193	9.619358	10.03254	10.03512	10.01704	10.03512	1				0.053747				0.057781		
26		9.619358	9.627105	9.606446	9.611585	10.03256	10.06611	10.03254	10.06094	1				0.053747				0.057781		
27		9.609028	9.614193	9.601281	9.601281	10.05836	10.05965	10.04545	10.04803	1				0.053747				0.057781		
28		9.602572	9.606446	9.601281	9.606446	10.04571	10.06094	10.04545	10.04816	1				0.053747				0.057781		
29		9.606446	9.624522	9.603863	9.624522	10.05061	10.0532	10.04545	10.05061	0				0.053747				0.057781		
30		9.62194	9.62194	9.609028	9.609028	10.04803	10.08935	10.04803	10.08935	1				0.053747				0.057781		
31		9.609028	9.61161	9.598698	9.601281	10.08935	10.09452	10.07127	10.08935	1				0.053747				0.057781		
32		9.598698	9.601281	9.588343	9.588369	10.08935	10.0971	10.06353	10.07386	1				0.053747				0.057781		
33		9.585787	9.593534	9.580622	9.583204	10.07386	10.07902	10.07256	10.07644	0				0.053747				0.057781		
34		9.583204	9.596116	9.580622	9.596116	10.07644	10.07644	10.06611	10.06882	1				0.053747				0.057781		
35		9.596116	9.61161	9.596116	9.596116	10.06934	10.07386	10.06869	10.06869	0				0.053747				0.057781		
36		9.596607	9.598698	9.596116	9.598673	10.07127	10.07386	10.06675	10.07127	0				0.053747				0.057781		
37		9.597407	9.598698	9.596116	9.596116	10.07386	10.07386	10.06611	10.07125	0				0.053747				0.057781		

Does stock price forecasting work? Experience of a stock trader

http://youtu.be/UmGIGEJMmN8?t=13m15s

Competition winners' experience

Tried lots of lagging and differencing of different predictors

Variable selection (logistic regression with variable selection)

Variable 74 (open, close, high, low)

- Order 12 differencing of var74
- Take lag 13 of target

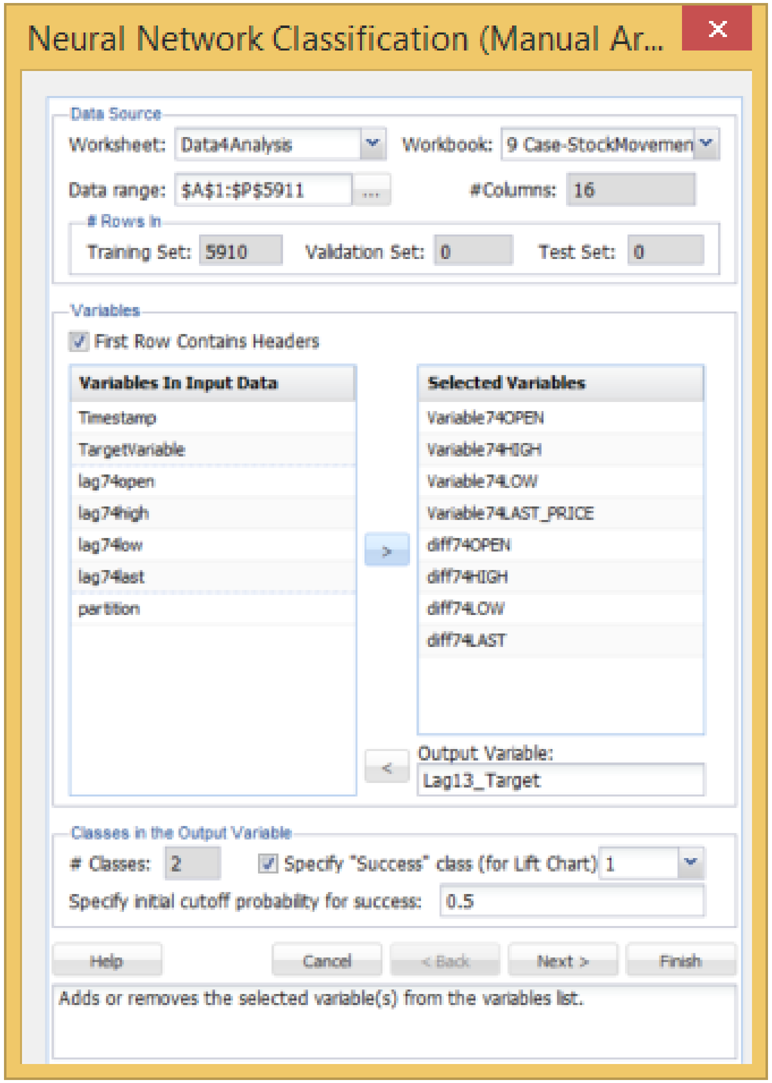
Try forecasting with a neural net

Pre-processing

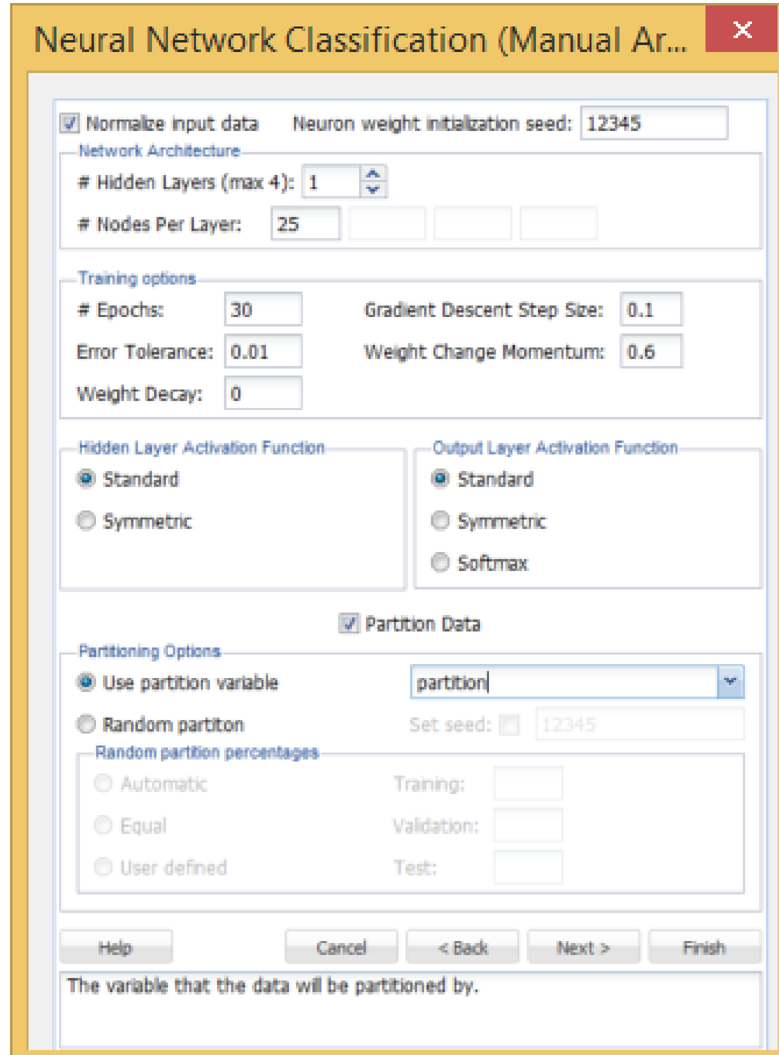
Timestam	TargetVar	Variable74OPEN	Variable74HIGH	Variable74LOW	Variable74LAST_PRICE	lag74oper	lag74high	lag74low	lag74last	diffOPEN	diffHIGH	diffLOW	diffLAST	Lag13_Targ
40182.4	1	12.80601178	13.00227249	12.80601178	12.97386634									
40182.4	1	12.97386634	13.02293152	12.97128396	12.97128396									
40182.4	1	12.95320731	12.96095445	12.90930689	12.93254829									
40182.41	1	12.95578969	12.98419585	12.92738353	12.92738353									
40182.41	0	12.92996591	12.93513067	12.90672451	12.92221878									
40182.41	1	12.92480116	12.92996591	12.89381262	12.9196364									
40182.42	0	12.90414213	12.94029542	12.86540647	12.86540647									
40182.42	0	12.87573598	12.88864787	12.82150604	12.82150604									
40182.42	0	12.81117653	12.81634129	12.75952897	12.79051751									
40182.43	0	12.79051751	12.79051751	12.74919946	12.74919946									
40182.43	0	12.74919946	12.80084702	12.74919946	12.80084702									
40182.43	1	12.80084702	12.88090073	12.80084702	12.88090073	12.80601	13.00227	12.80601	12.97387	-0.00516	-0.12137	-0.00516	-0.09297	
40182.44	1	12.89381262	12.92221878	12.89381262	12.91447165	12.97387	13.02293	12.97128	12.97128	-0.08005	-0.10071	-0.07747	-0.05681	1
40182.44	1	12.91447165	12.94029542	12.91447165	12.94029542	12.95321	12.96095	12.90931	12.93255	-0.03874	-0.02066	0.005165	0.007747	1
40182.44	1	12.94029542	12.95578969	12.91447165	12.91705402	12.95579	12.9842	12.92738	12.92738	-0.01549	-0.02841	-0.01291	-0.01033	1
40182.45	1	12.92221878	12.92221878	12.90155976	12.91447165	12.92997	12.93513	12.90672	12.92222	-0.00775	-0.01291	-0.00516	-0.00775	1
40182.45	1	12.91188927	12.97903109	12.91188927	12.97903109	12.9248	12.92997	12.89381	12.91964	-0.01291	0.049065	0.018077	0.059395	0
40182.45	1	12.97644871	12.99710774	12.94804256	12.94804256	12.90414	12.9403	12.86541	12.86541	0.072307	0.056812	0.082636	0.082636	1
40182.46	1	12.95837207	12.9661192	12.95062494	12.96353682	12.87574	12.88865	12.82151	12.82151	0.082636	0.077471	0.129119	0.142031	0
40182.46	1	12.96095445	12.96095445	12.92221878	12.92221878	12.81118	12.81634	12.75953	12.79052	0.149778	0.144613	0.16269	0.131701	0

XLMiner: Classification > Neural Network > Manual

Inputs , Output



Architecture



Training Data Scoring - Summary Report

Cutoff probability value for success (UPDATABLE)	0.5

Confusion Matrix		
	Predicted Class	
Actual Class	1	0
1	1758	281
0	114	1218

Error Report			
Class	# Cases	# Errors	% Error
1	2039	281	13.78127
0	1332	114	8.558559
Overall	3371	395	11.71759

Performance	
Success Class	1
Precision	0.939103
Recall (Sensitivity)	0.862187
Specificity	0.914414
F1-Score	0.899003

Validation Data Scoring - Summary Report

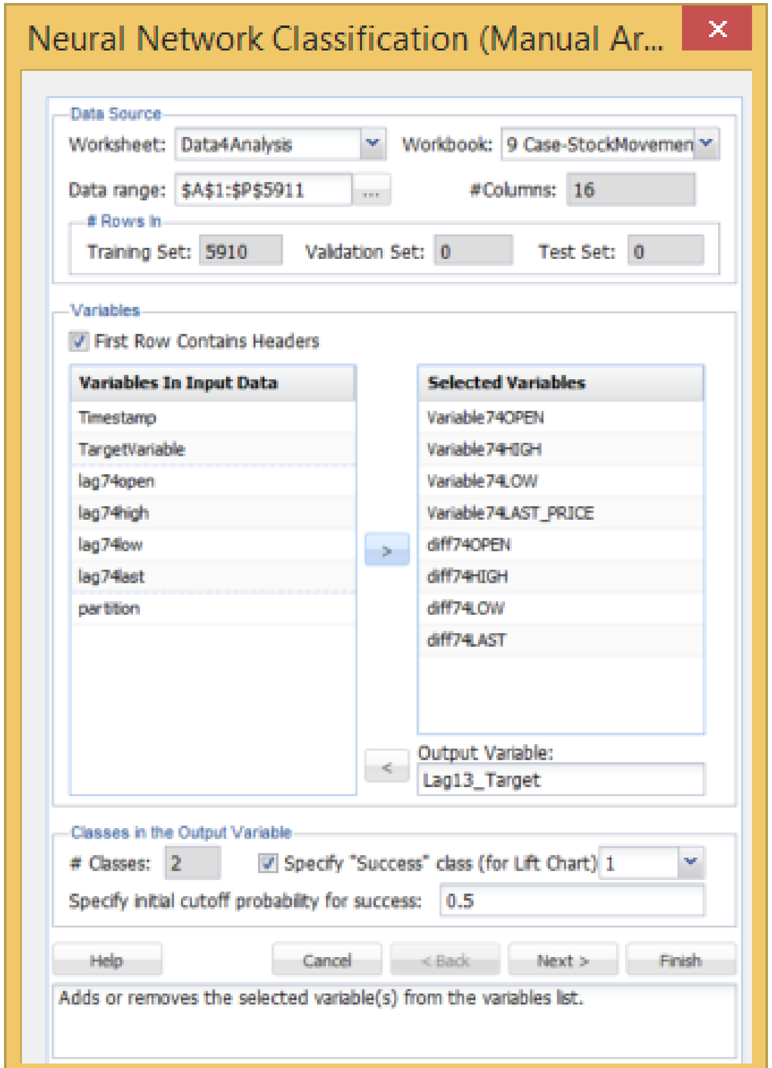
Cutoff probability value for success (UPDATABLE)	0.5

Confusion Matrix		
	Predicted Class	
Actual Class	1	0
1	994	584
0	2	959

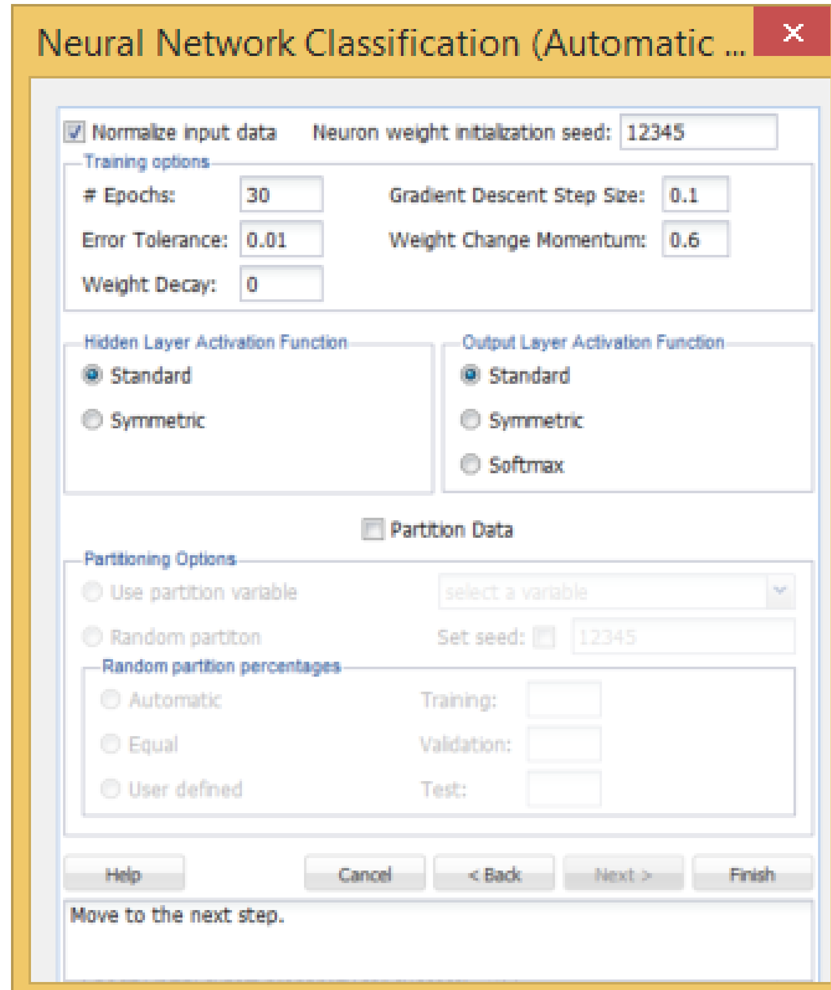
Error Report			
Class	# Cases	# Errors	% Error
1	1578	584	37.00887
0	961	2	0.208117
Overall	2539	586	23.07995

Performance	
Success Class	1
Precision	0.997992
Recall (Sensitivity)	0.629911
Specificity	0.997919
F1-Score	0.772339

XLMiner: Classification > Neural Network > Automatic Network



Tries different #hidden layers, (max #nodes around 5,6)



Any better architecture?

Error Report

				Error Report. Training Partition.				Error Report. Validation Partition.			
Net ID	# Layers	# Neurons (Layer 1	# Neurons (Layer 2	T: # Errors	T: % Errors	T: % Sensitivity	T: % Specificity	V: # Errors	V: % Errors	V: % Sensitivity	V: % Specificity
Net 1	1	1		441	13.08	84.6	90.47	861	33.91	45.5	99.9
Net 2	1	2		388	11.51	87.2	90.47	425	16.74	73.95	98.54
Net 3	1	3		385	11.42	87.2	90.69	521	20.52	67.24	99.58
Net 4	1	4		397	11.78	85.78	91.97	722	28.44	54.31	99.9
Net 5	1	5		387	11.48	86.95	90.92	559	22.02	64.77	99.69
Net 6	1	6		447	13.26	83.23	92.12	820	32.3	48.1	99.9
Net 7	2	1	1	436	12.93	83.91	91.89	821	32.34	48.04	99.9
Net 8	2	1	2	436	12.93	83.96	91.82	822	32.37	47.97	99.9
Net 9	2	1	3	440	13.05	83.72	91.89	831	32.73	47.4	99.9
Net 10	2	1	4	439	13.02	83.82	91.82	835	32.89	47.15	99.9
Net 11	2	1	5	448	13.29	83.37	91.82	873	34.38	44.74	99.9
Net 12	2	2	1	405	12.01	85.29	92.12	1093	43.05	30.74	100
Net 13	2	2	2	399	11.84	85.38	92.42	1116	43.95	29.28	100
Net 14	2	2	3	405	12.01	85.09	92.42	966	38.05	38.85	99.9
Net 15	2	2	4	401	11.9	85.29	92.42	792	31.19	49.87	99.9
Net 16	2	2	5	403	11.95	85.19	92.42	853	33.6	46.01	99.9
Net 17	2	3	1	425	12.61	84.01	92.57	779	30.68	50.7	99.9
Net 18	2	3	2	407	12.07	84.8	92.72	709	27.92	55.13	99.9
Net 19	2	3	3	402	11.93	85.04	92.72	707	27.85	55.26	99.9
Net 20	2	3	4	397	11.78	85.58	92.27	510	20.09	68	99.48
Net 21	2	3	5	388	11.51	86.37	91.74	700	27.57	55.7	99.9
Net 22	2	4	1	426	12.64	84.21	92.19	823	32.41	47.91	99.9
Net 23	2	4	2	404	11.98	84.89	92.79	686	27.02	56.59	99.9

Can we do better with **logistic regression**?

Regression Model

Input Variables	Coefficient	Std. Error	Chi2-Statistic	P-Value	Odds
Intercept	-6.57467	1.703238	14.90038521	0.000113	0.001395266
Variable74OPEN	11.93931	9.672523	1.523629029	0.217071	153171.6954
Variable74HIGH	60.68003	10.35533	34.33707045	4.63E-09	2.25424E+26
Variable74LOW	-0.52959	10.54919	0.002520236	0.959961	0.588846266
Variable74LAST_PRICE	-71.5196	10.48729	46.50748533	9.13E-12	8.69864E-32
diff74OPEN	-13.1919	6.664462	3.918202318	0.047766	1.86559E-06
diff74HIGH	-48.724	7.428234	43.02432779	5.41E-11	6.90955E-22
diff74LOW	-14.8319	7.246665	4.189073956	0.040685	3.61894E-07
diff74LAST					

Popular among competitors, especially due to variable selection capabilities

Training Data Scoring - Summary Report

Cutoff probability value for success (UPDATABLE)

Confusion Matrix		
	Predicted Class	
Actual Class	1	0
1	1918	121
0	194	1138

Error Report			
Class	# Cases	# Errors	% Error
1	2039	121	5.934281511
0	1332	194	14.56456456
Overall	3371	315	9.344408187

Performance	
Success Class	1
Precision	0.908144
Recall (Sensitivity)	0.940657
Specificity	0.854354
F1-Score	0.924115

Validation Data Scoring - Summary Report

Cutoff probability value for success (UPDATABLE)

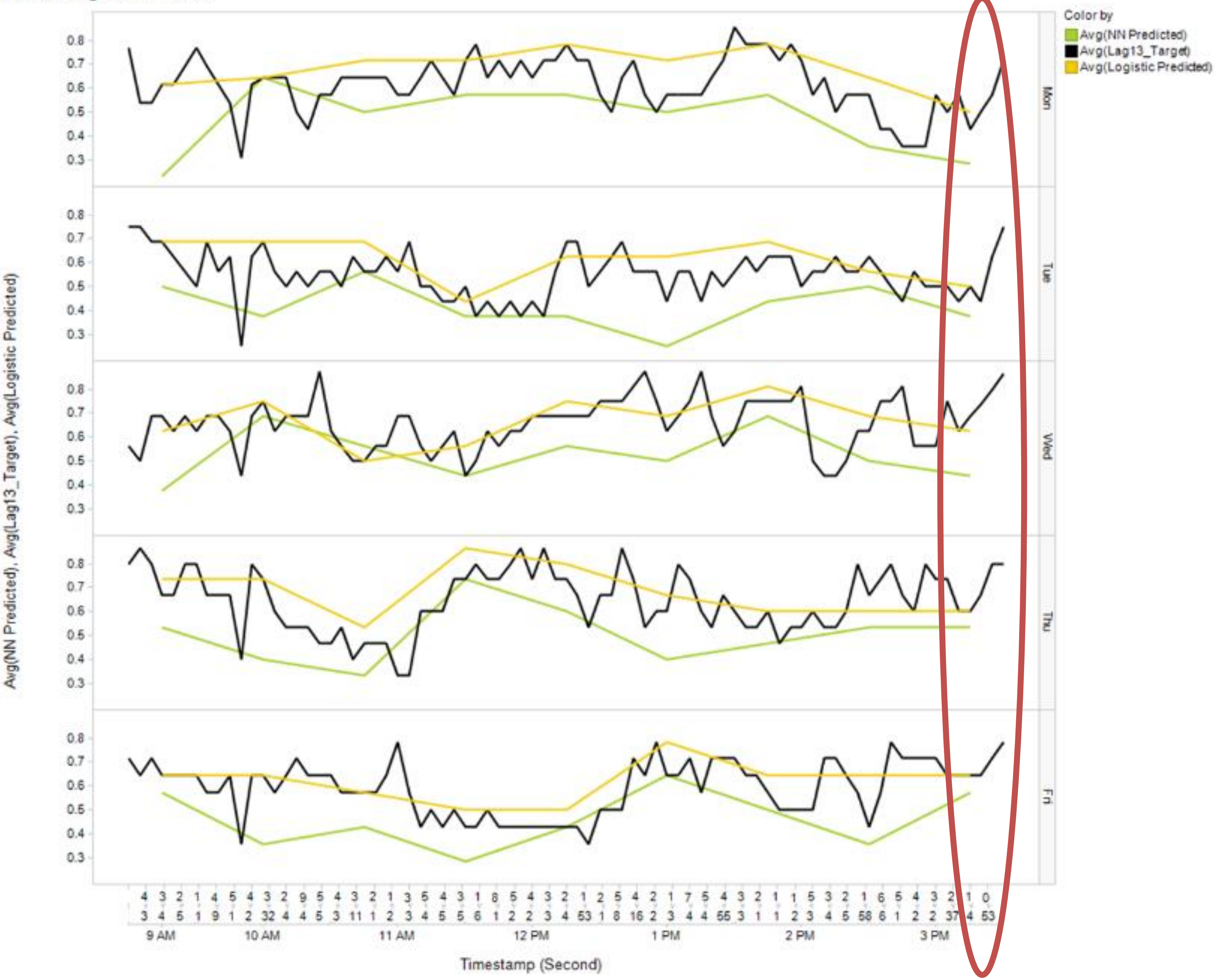
Confusion Matrix		
	Predicted Class	
Actual Class	1	0
1	1501	77
0	185	776

Error Report			
Class	# Cases	# Errors	% Error
1	1578	77	4.879594423
0	961	185	19.25078044
Overall	2539	262	10.31902324

Performance	
Success Class	1
Precision	0.890273
Recall (Sensitivity)	0.951204
Specificity	0.807492
F1-Score	0.91973

Further improvements: look at the forecasts and actuals!

Forecasting Performance



Here's what we did

Output:
$\mathit{Target}_{t\text{-}13}$

Inputs (8 predictors):
$\mathit{Var74OPEN}_t$
$\mathit{Var74OPEN}_t - \mathit{Var74OPEN}_{t\text{-}12}$

How can this forecasting method be implemented in practice?

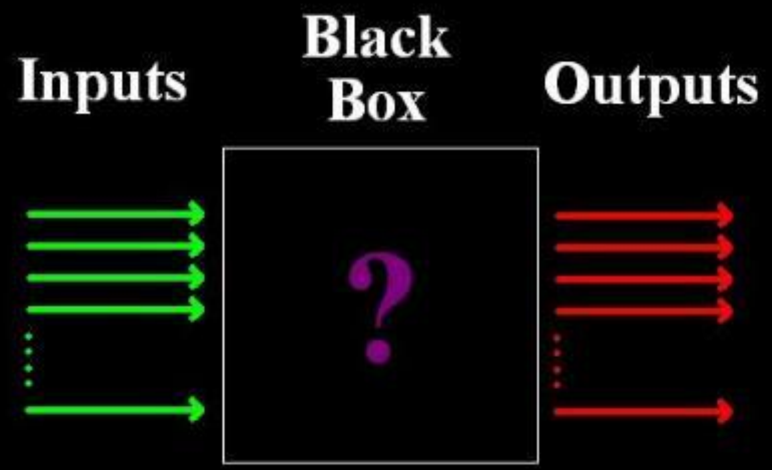
For what purpose?

Forecasting with NN: Advantages & weaknesses

Data-driven
Automated
Numerical and binary forecasts

Pre-processing: controversial
Requires many training periods

Blackbox
Over-fitting
Can become computationally intensive
No variable-selection mechanism



NN Forecasting in practice

Tourism

Finance (trading)

Renewable energy

Mixed results compared to other methods
Seem to work best with high-frequency data