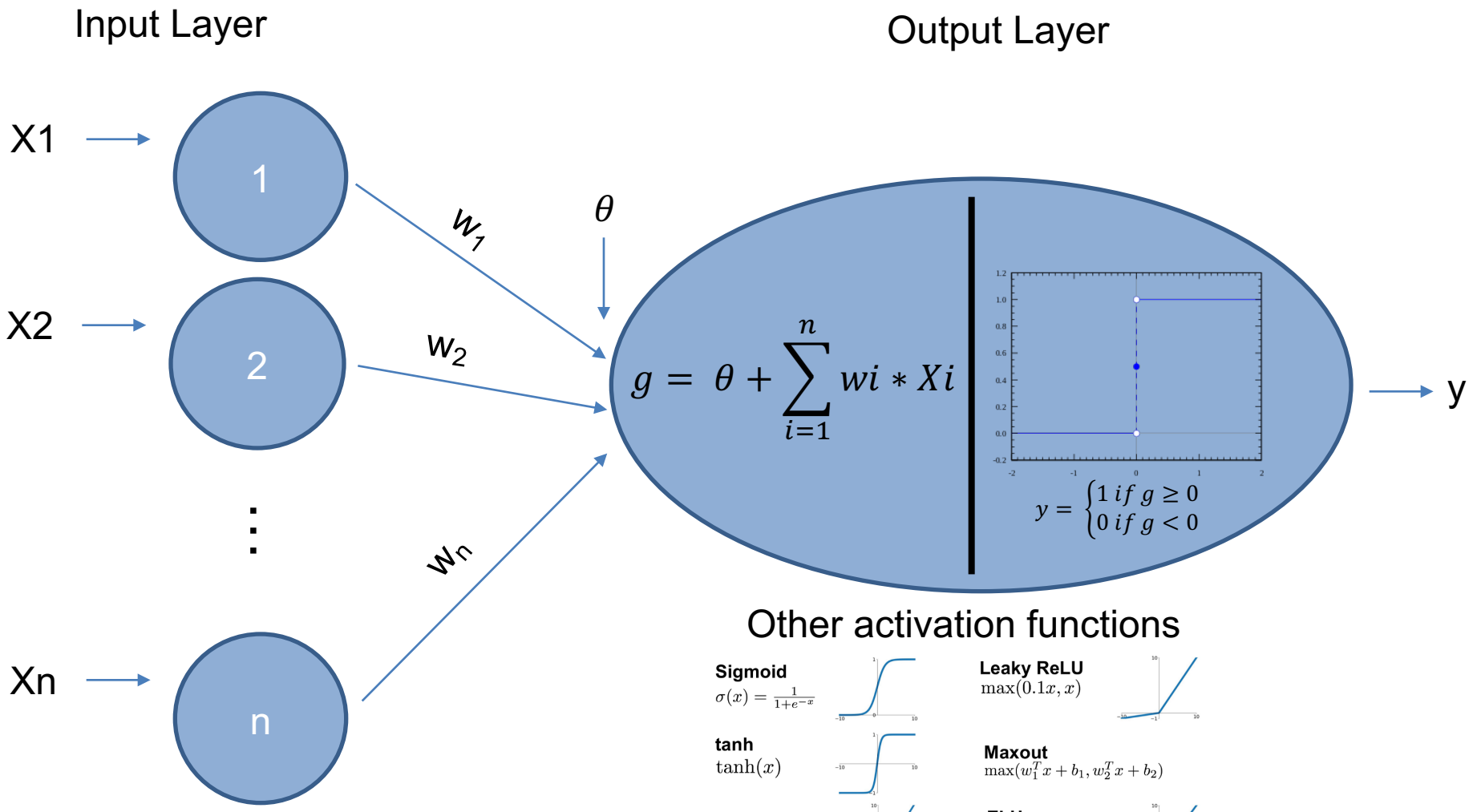
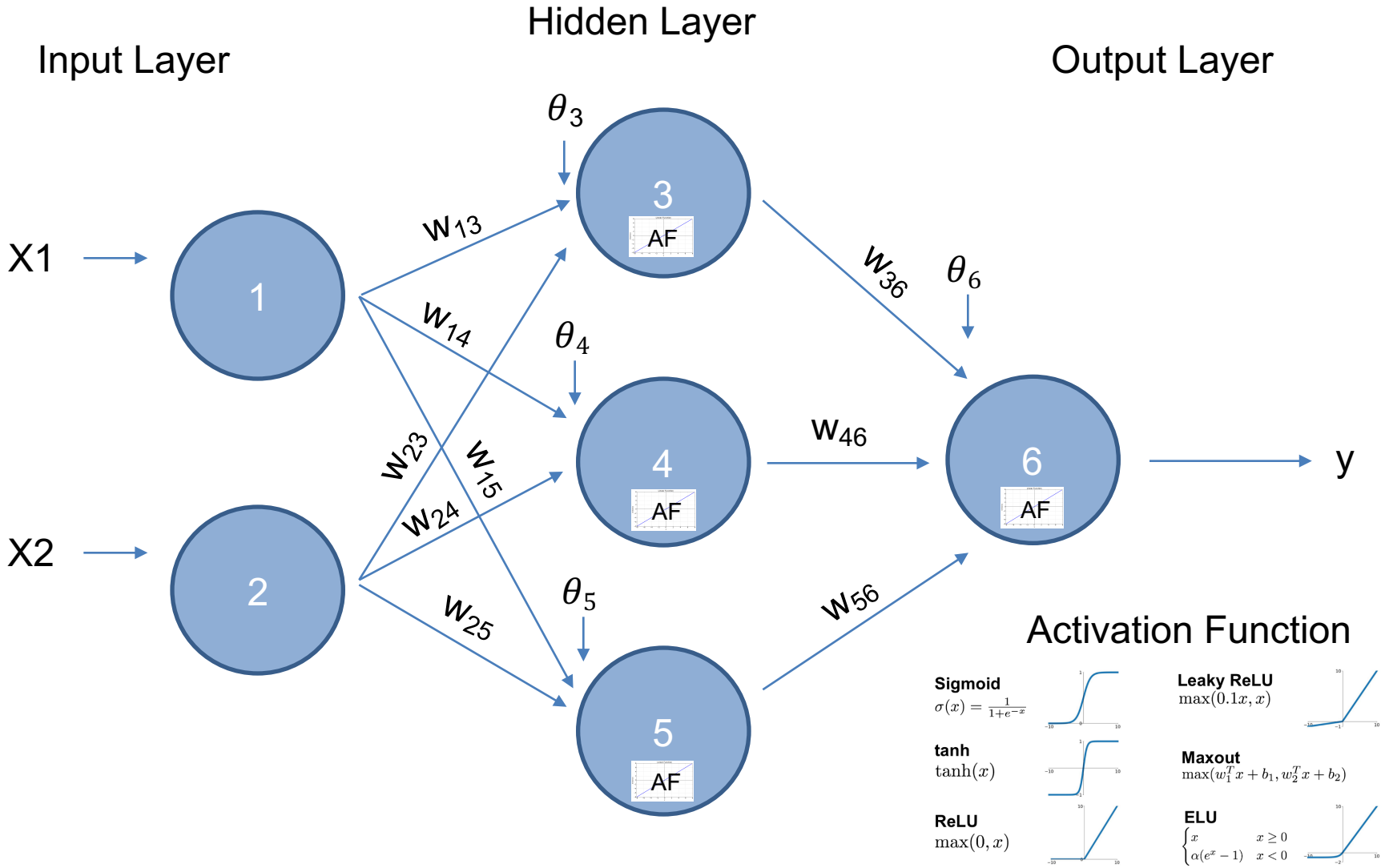
Unit 13

Neural Networks and Time Series

The Perceptron

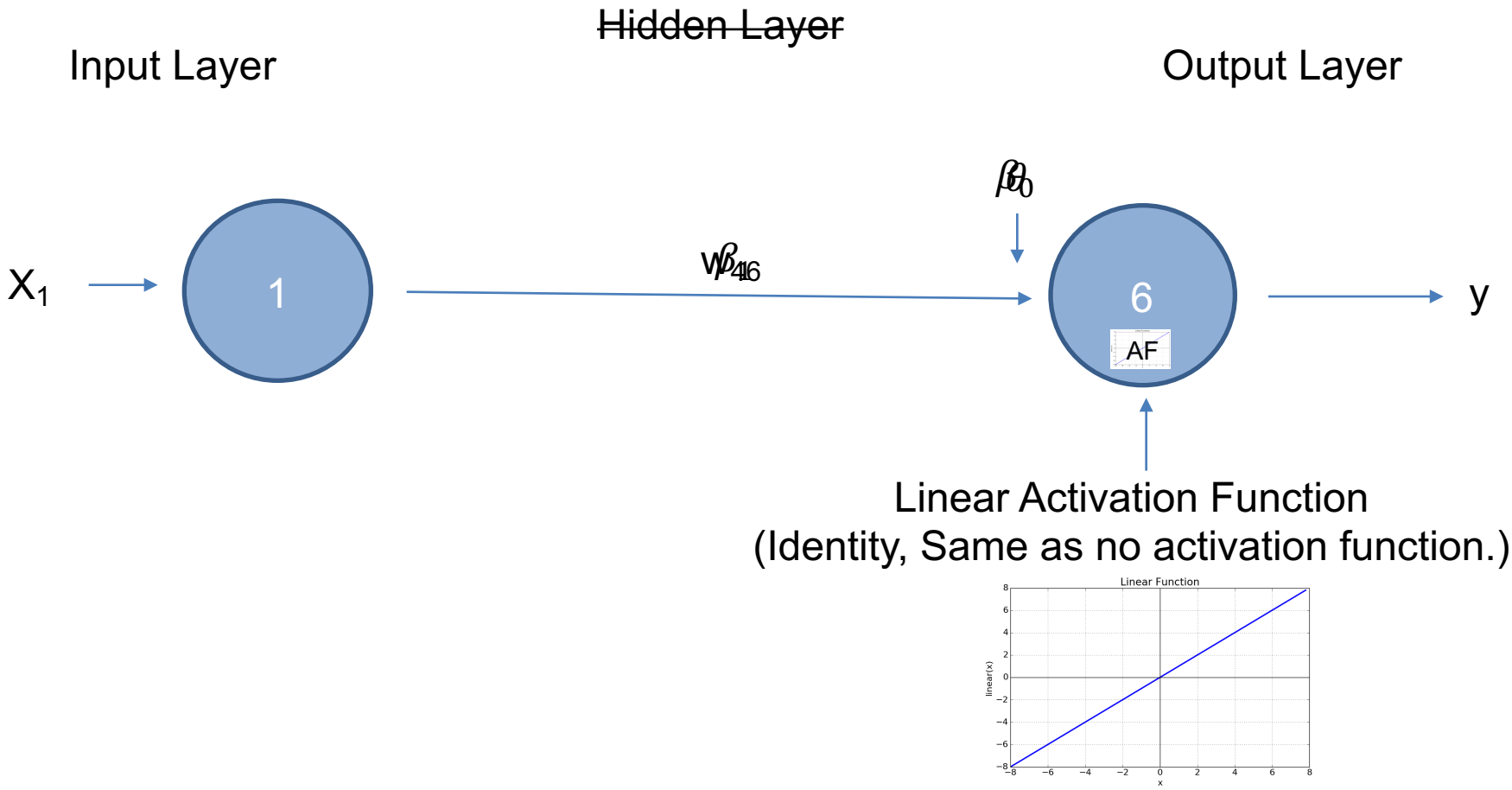


Multilayered Perceptron (MLP) (The Neural Network Model)



A Familiar Neural Network: SLR

$$y = \beta_0 + \beta_1 x_1 + \varepsilon$$



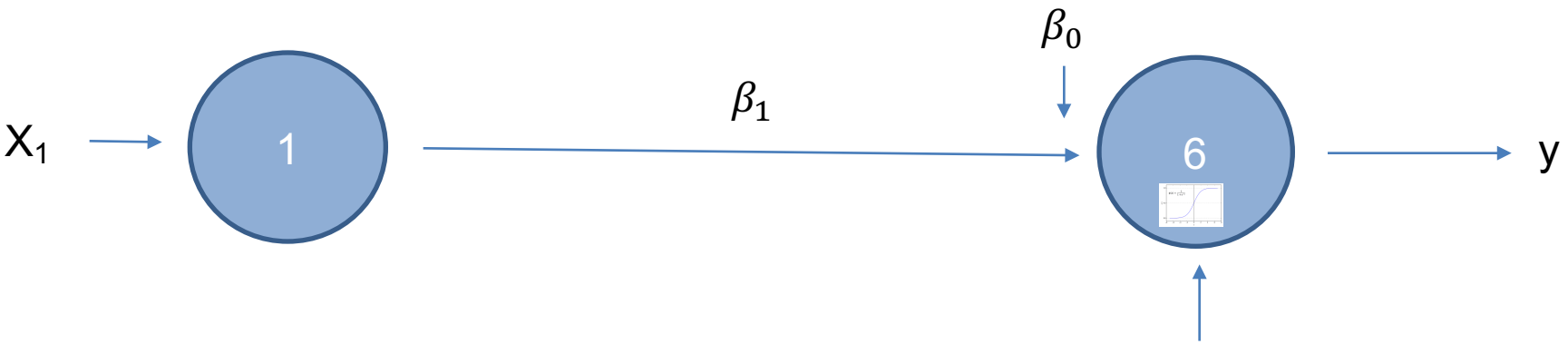
Another Familiar Neural Network

~~Hidden Layer~~

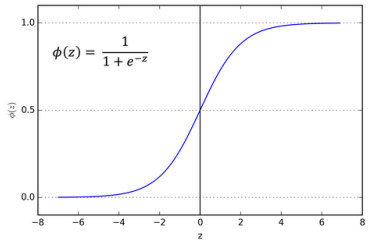
Input Layer

Output Layer

$$logit(y) = \beta_0 + \beta_1 x_1 + \varepsilon$$

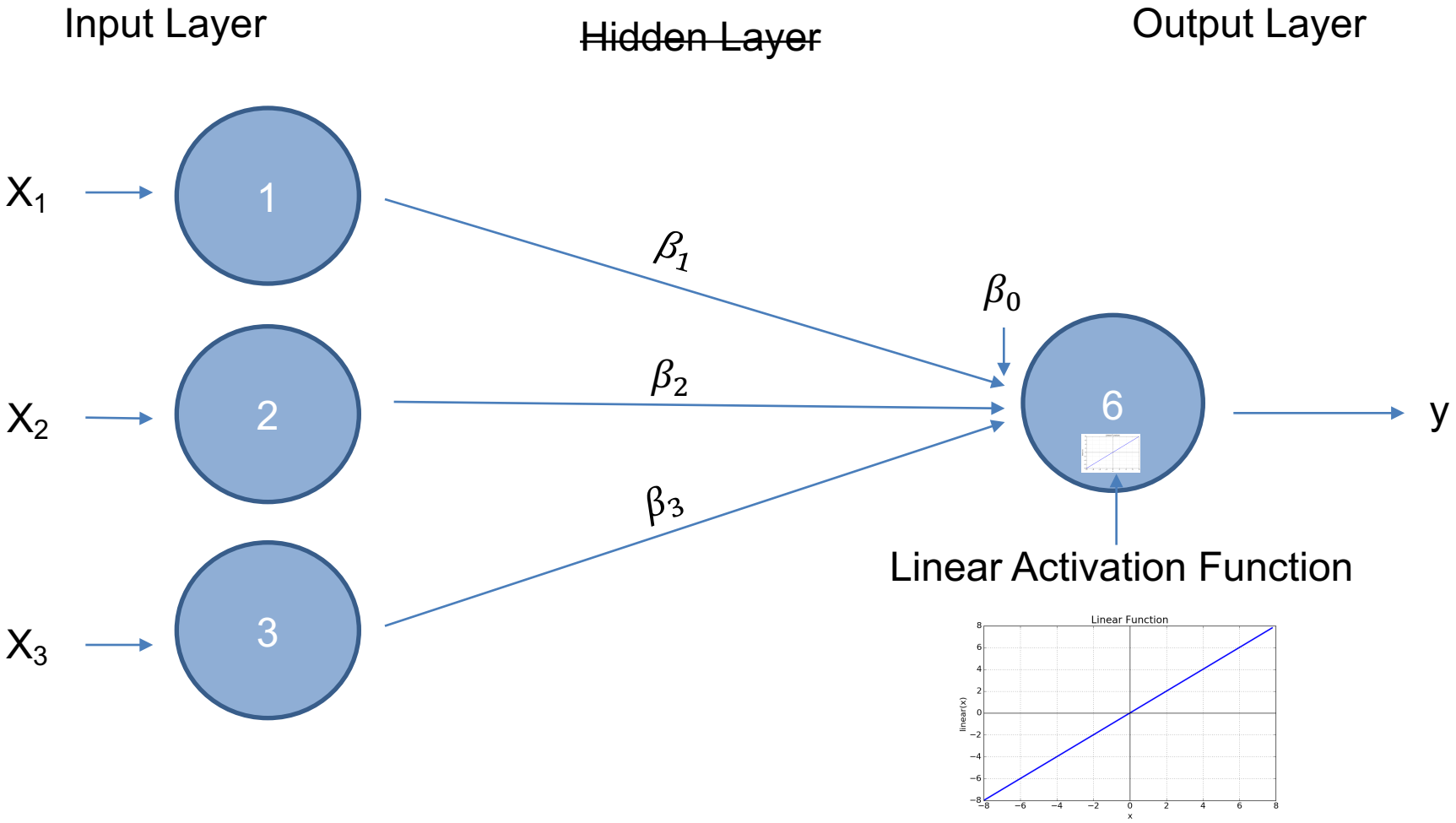


Logistic Activation Function



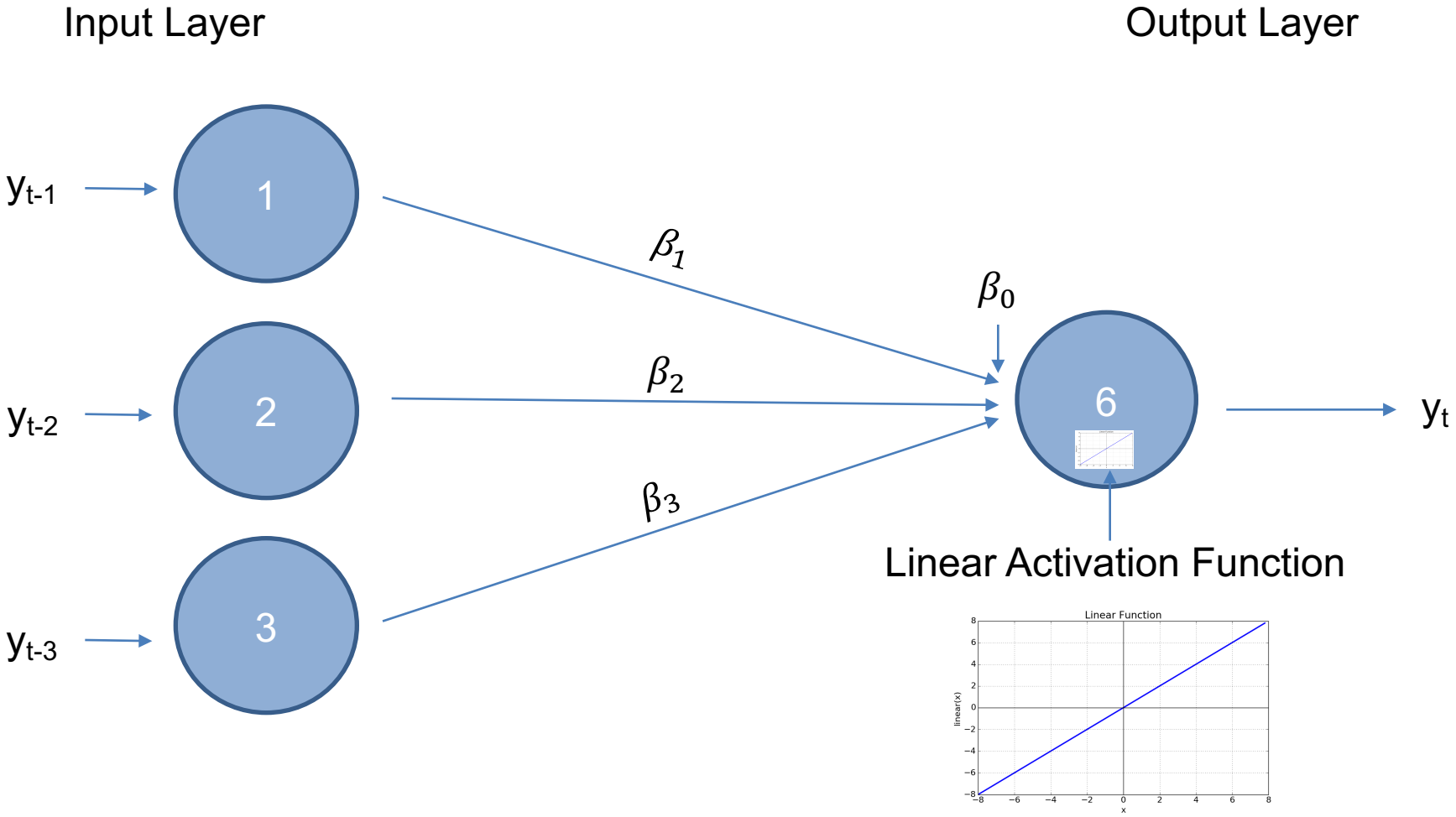
MLR as a Neural Network: MLR

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \varepsilon$$



Time Series as a Neural Network: AR(3)

$$y_t = \beta_0 + \beta_1 y_{t-1} + \beta_2 y_{t-2} + \beta_3 y_{t-3} + \varepsilon$$



What if?

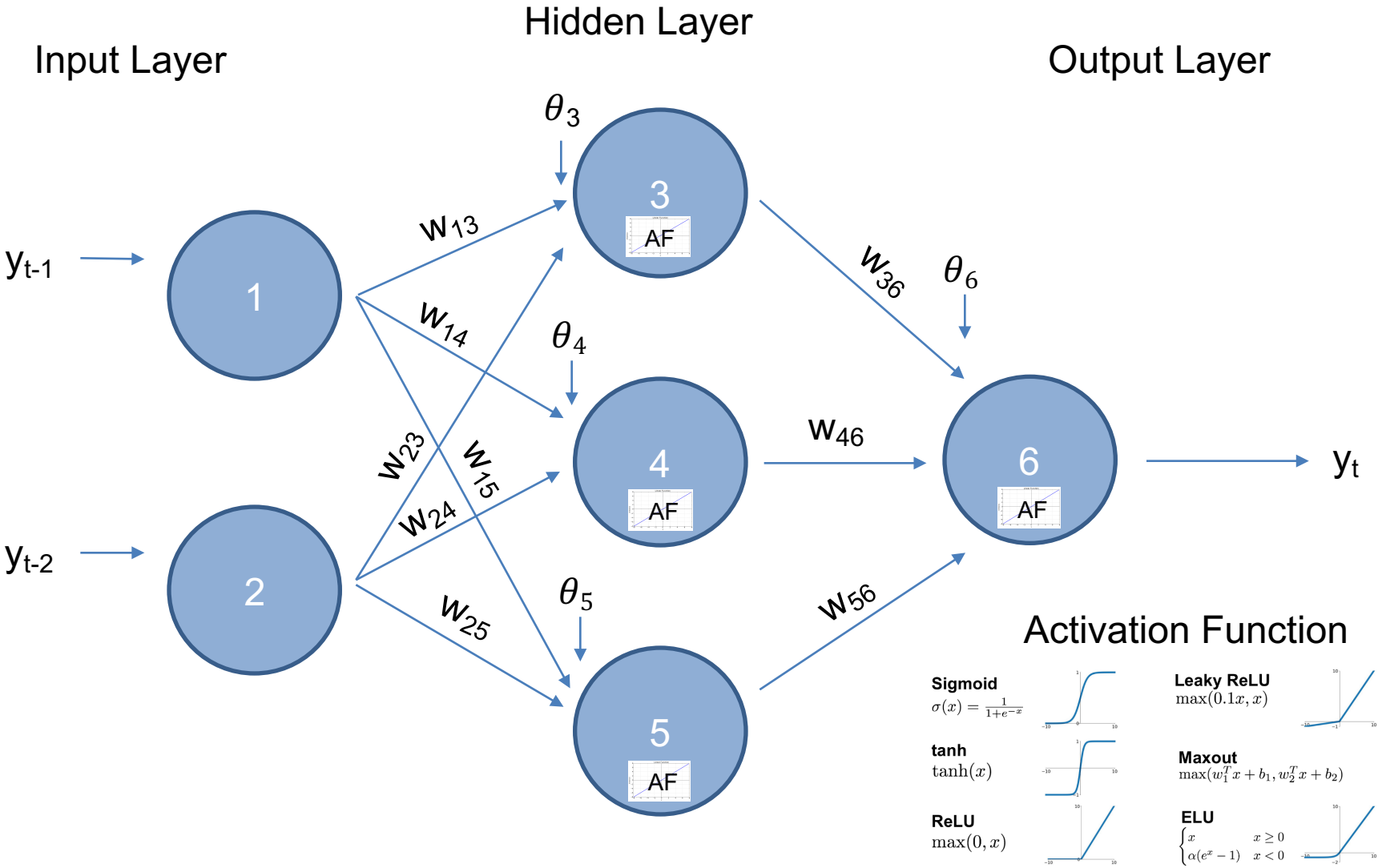
What if the relationship between y_t and the past is non-linear?

Example:

$$y_t = e^{\beta_0} + e^{\beta_1 y_{t-1}} + e^{\beta_2 y_{t-2}}$$

It can be shown that a neural net with one hidden layer can approximate any continuous function.

Multilayered Perceptron (MLP) (The Neural Network Model)



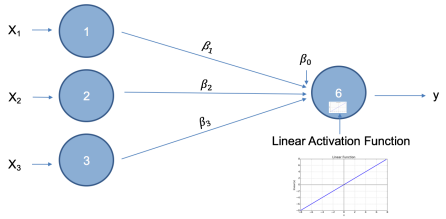
DataScience@SMU

Concept Check

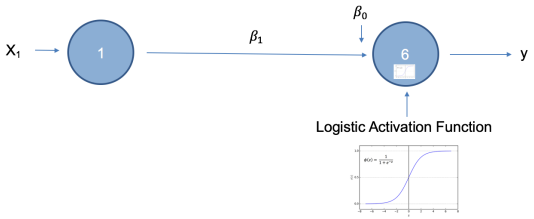
Match the model with the appropriate neural network:

1. Simple Linear Regression Model
2. AR(3)
3. Multiple Linear Regression Model
4. Non-linear model
5. Log linear model.

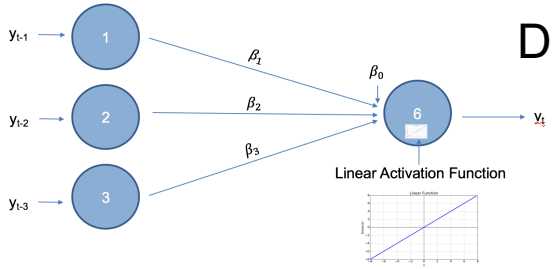
A



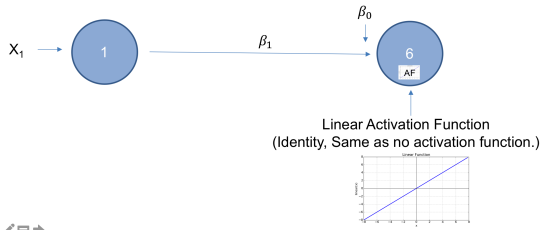
C



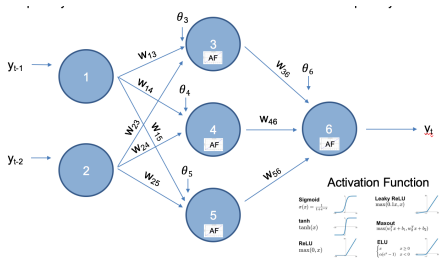
B



D



E

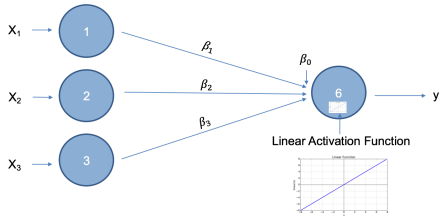


Concept Check

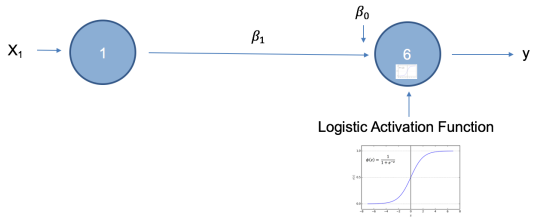
Match the model with the appropriate neural network:

1. Simple Linear Regression Model D
2. AR(3)B
3. Multiple Linear Regression Model A
4. Non-linear model E
5. Log linear model. C

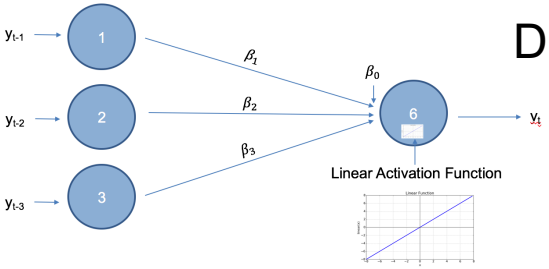
A



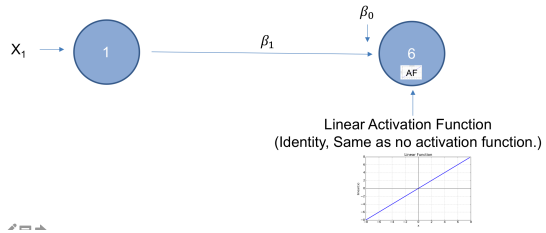
C



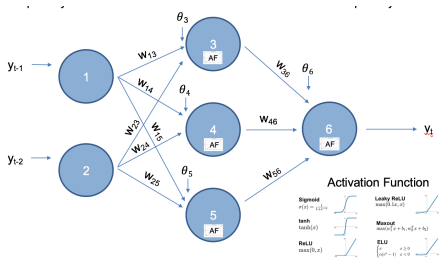
B



D



E



Concept Check

2. True / False: The input nodes have activation functions.

3. TRUE / FALSE: Adding 1 or more “hidden layers” allows for estimation of non-linear relationships between the inputs and the outputs.

4. TRUE / FALSE A single Hidden Layer can approximate any continuous function (model).

5. TRUE / FALSE: Gradient decent is the only option for estimating the weights and biases.

Concept Check

2. True / False: The input nodes have activation functions.

3. TRUE / FALSE: Adding 1 or more “hidden layers” allows for estimation of non-linear relationships between the inputs and the outputs.

4. TRUE / FALSE A single Hidden Layer can approximate any continuous function (model).

5. TRUE / FALSE: Gradient decent is the only option for estimating the weights and biases.

DataScience@SMU

MLPs and R

mlp {nnfor}

Multilayer Perceptron for time series forecasting

Description

This function fits MLP neural networks for time series forecasting.

Usage

```
mlp(y, m = frequency(y), hd = NULL, reps = 20, comb = c("median",
  "mean", "mode"), lags = NULL, keep = NULL, difforder = NULL,
  outplot = c(FALSE, TRUE), sel.lag = c(TRUE, FALSE),
  allow.det.season = c(TRUE, FALSE), det.type = c("auto", "bin", "trg"),
  xreg = NULL, xreg.lags = NULL, xreg.keep = NULL,
  hd.auto.type = c("set", "valid", "cv", "elm"), hd.max = NULL,
  model = NULL, retrain = c(FALSE, TRUE), ...)
```

Arguments

`y`	Input time series. Can be ts or msts object.
`m`	Frequency of the time series. By default it is picked up from y.
`hd`	Number of hidden nodes. This can be a vector, where each number represents the number of hidden nodes of a different hidden layer.
`reps`	Number of networks to train, the result is the ensemble forecast.
`comb`	Combination operator for forecasts when reps > 1. Can be "median" "mode" (based on KDE estimation) and "mean".
`lags`	Lags of y to use as inputs. If none provided then 1:frequency(y) is used. Use 0 for no univariate lags.
`keep`	Logical vector to force lags to stay in the model if sel.lag == TRUE. If NULL then it keep = rep(FALSE,length(lags)).
`difforder`	Vector including the differencing lags. For example c(1,12) will apply first and seasonal (12) differences. For no differencing use 0. For automatic selection use NULL.
`sel.lag`	Automatically select lags. Can be TRUE or FALSE.
`allow.det.season`	Permit modelling seasonality with deterministic dummies.
`det.type`	Type of deterministic seasonality dummies to use. This can be "bin" for binary or "trg" for a sine-cosine pair. With "auto" if ony a single seasonality is used and periodicity is up to 12 then "bin" is used, otherwise "trg".
`xreg`	Exogenous regressors. Each column is a different regressor and the sample size must be at least as long as the target in-sample set, but can be longer.
`xreg.lags`	This is a list containing the lags for each exogenous variable. Each list is a numeric vector containing lags. If xreg has 3 columns then the xreg.lags list must contain three elements. If NULL then it is automatically specified.
`xreg.keep`	List of logical vectors to force lags of xreg to stay in the model if sel.lag == TRUE. If NULL then all exogenous lags can be removed. The syntax for multiple xreg is the same as for xreg.lags.
`hd.auto.type`	Used only if hd==NULL. "set" fixes hd=5. "valid" uses a 20% validation set (randomly) sampled to find the best number of hidden nodes. "cv" uses 5-fold cross-validation. "elm" uses ELM to estimate the number of hidden nodes (experimental).
`hd.max`	When hd.auto.type is set to either "valid" or "cv" then this argument can be used to set the maximum number of hidden nodes to evaluate, otherwise the maximum is set automatically.
`model`	A previously trained mlp object. If this is provided then the same model is fitted to y, without re-estimating any model parameters.
`retrain`	If a previous model is provided, retrain the network or not.

Example: SWA Delays

Time-Series Objects

Description

The function `ts` is used to create time-series objects.

`as.ts` and `is.ts` coerce an object to a time-series and test whether an object is a time series.

Usage

```
ts(data = NA, start = 1, end = numeric(), frequency = 1,
   deltat = 1, ts.eps = getOption("ts.eps"), class = , names = )
as.ts(x, ...)
is.ts(x)
```

Arguments

`data`	a vector or matrix of the observed time-series values. A data frame will be coerced to a numeric matrix via `data.matrix`. (See also 'Details'.)
`start`	the time of the first observation. Either a single number or a vector of two integers, which specify a natural time unit and a (1-based) number of samples into the time unit. See the examples for the use of the second form.
`end`	the time of the last observation, specified in the same way as `start`.
`frequency`	the number of observations per unit of time.

SWATrain = ts(SWA$arr_delay[1:141],start=c(2004,1),frequency = 12)

Example: SWA Delays

Usage

```
ts(data = NA, start = 1, end = numeric(), frequency = 1,
   deltat = 1, ts.eps = getOption("ts.eps"), class = , names = )
```

SWATrain = ts(SWA$arr_delay[1:141],start=c(2004,1),frequency = 12)

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2004	13120	22475	20515	21990	35527	55436	33230	27633	8968	32832	32451	22603
2005	19312	13047	15767	10640	14835	13343	41759	33651	20378	14612	14779	17714
2006	10348	19869	35146	19741	19245	24010	24266	23433	21932	34415	21877	37388
2007	21002	29386	45611	33272	50451	68692	57515	43492	23120	24249	23971	45271
2008	29376	49046	53617	38033	38636	52185	32304	28792	10722	16144	19611	55393
2009	17425	10681	29266	35972	27638	40251	24670	27757	15066	34744	8586	33124
2010	22406	26278	29812	19568	35272	37969	37685	23385	22920	32746	26566	41375
2011	32556	41939	27700	42095	59774	30285	25742	25312	23182	20094	14175	21073
2012	13844	14743	33830	24490	23615	36834	36789	28226	16621	16182	16534	30062
2013	19379	16559	23709	32022	46882	45227	43761	38579	29059	26282	28879	60143
2014	59555	30659	38806	36031	55136	68654	60061	39171	24802	34870	31845	53497
2015	33976	32441	33603	44298	78256	62907	56052	44223	19853			

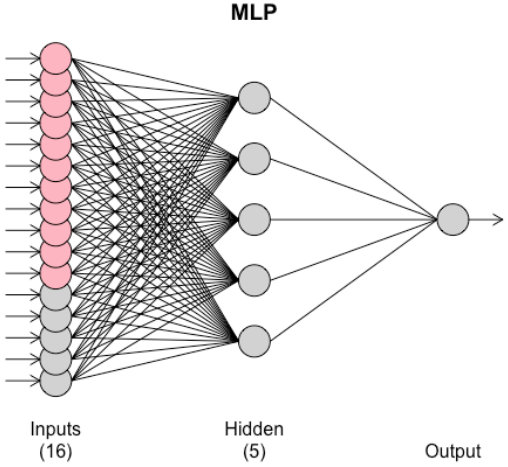
DataScience@SMU

Example: SWA Delays

```
#Divide dataset into training and test set.  The last 36 months in the test set.
SWATrain = ts(SWA$arr_delay[1:141],start= c(2004,1),frequency = 12)
SWATest = ts(SWA$arr_delay[142:177],start = c(2015,10),frequency = 12)
set.seed(2)
fit.mlp = mlp(SWATrain,reps = 50,comb = "mean")
fit.mlp
```

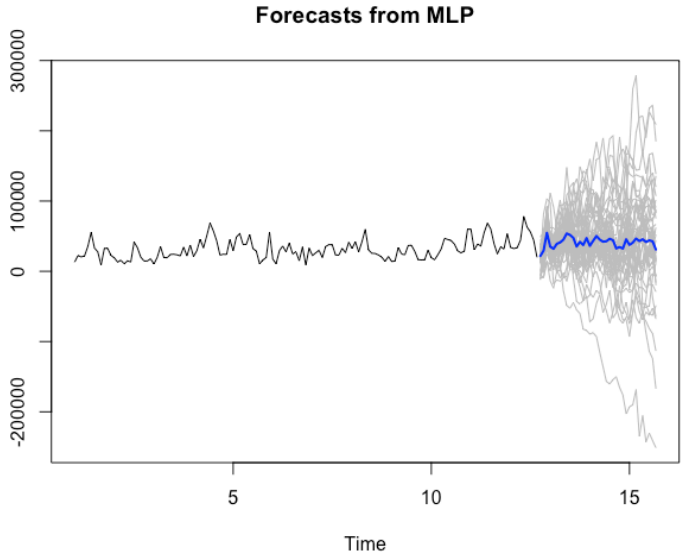
```
MLP fit with 5 hidden nodes and 50 repetitions.
Series modelled in differences: D1.
Univariate lags: (1,2,3,4,5)
Deterministic seasonal dummies included.
Forecast combined using the mean operator.
MSE: 5043757.8409.
```

```
# Visualize the Neural Network
plot(fit.mlp)
```



Note: pink nodes are for seasonal dummies and grey nodes are for lagged inputs

```
fore.mlp = forecast(fit.mlp, h = 36)
plot(fore.mlp)
```




```
ASE = mean((SWATest - fore.mlp$mean)^2)
ASE
```

```
[1] 317604251.9
```

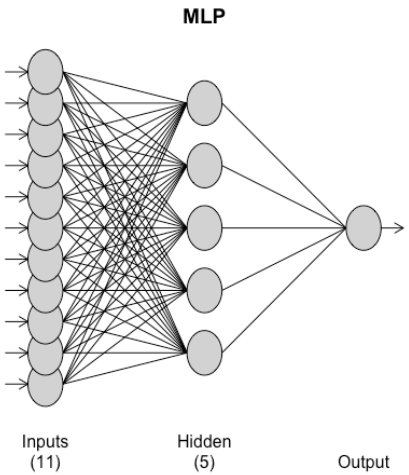
Example: SWA Delays

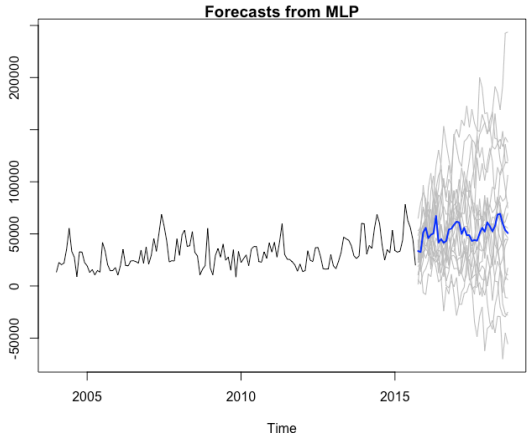
```
fit.mlp = mlp(SWATrain, lags = c(1,2,3,4,5,6,7,8,9,10,11,12),
allow.det.season = FALSE)
set.seed(2)
fit.mlp
```

```
MLP fit with 5 hidden nodes and 20 repetitions.
Series modelled in differences: D1.
Univariate lags: (1,2,3,4,5,6,7,8,9,11,12)
Forecast combined using the median operator.
MSE: 5598324.2681.
```

```
plot(fit.mlp)
fore.mlp = forecast(fit.mlp, h = 36)
plot(fore.mlp)
ASE = mean((SWATest - fore.mlp$mean)^2)
ASE
```

```
[1] 456929879.5
```



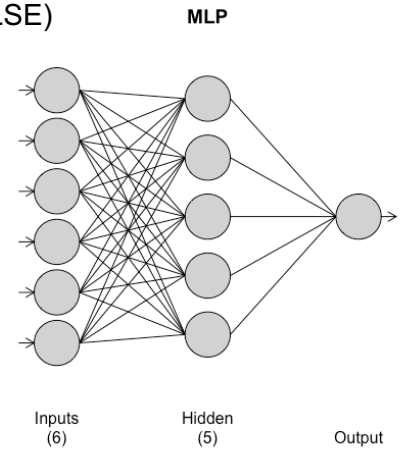


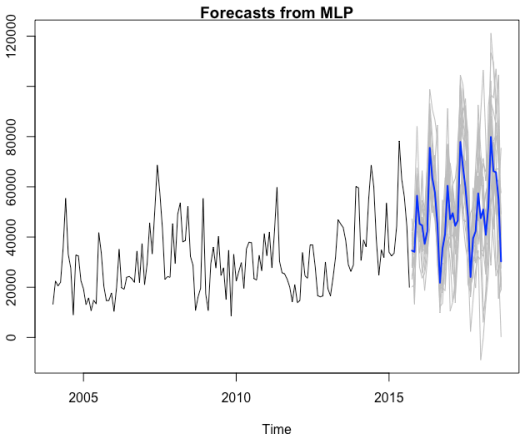

```
fit.mlp = mlp(SWATrain, difforder = c(12), allow.det.season = FALSE)
set.seed(2)
fit.mlp
```

```
MLP fit with 5 hidden nodes and 20 repetitions.
Series modelled in differences: D12.
Univariate lags: (1,2,3,6,9,12)
Forecast combined using the median operator.
MSE: 32068872.9302.
```

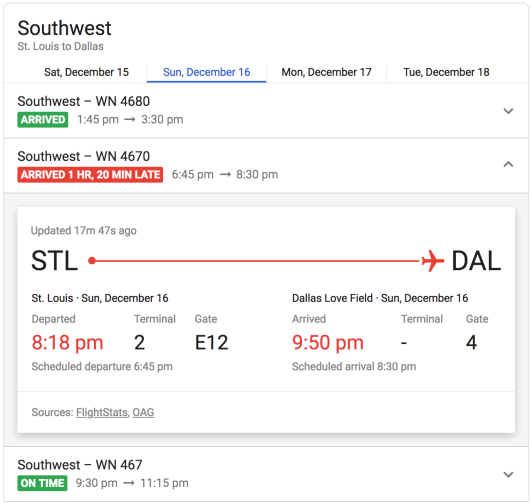
```
plot(fit.mlp)
fore.mlp = forecast(fit.mlp, h = 36)
plot(fore.mlp)
ASE = mean((SWATest - fore.mlp$mean)^2)
ASE
```

```
[1] 221327081.7
```





Example: SWA Delays



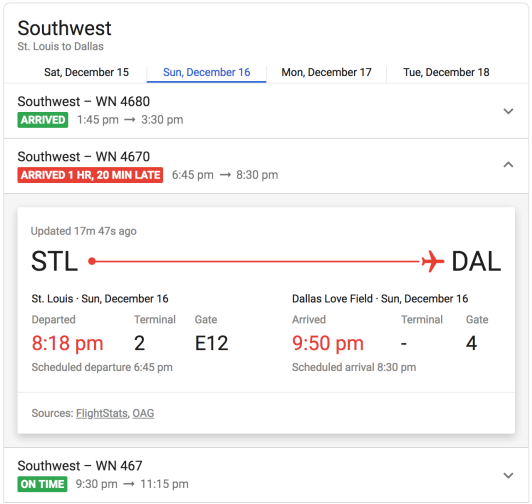
Model: AR(12):

$$(1-0.44B- 0.02B^2 +0.12B^3 -0.08B^4 +0.00B^5 -0.02B^6 -0.06B^7 +0.09B^8 -0.06B^9 -0.07B^{10} -0.02B^{11} -0.37B^{12})(X_t – 34934) = a_t$$

```
f = fore.arma.wge(SWA$arr_delay,phi = c(0.44, 0.02, -0.12, 0.08, 0.00, 0.02, 0.06, -0.09, 0.06, 0.07, 0.02, 0.37), n.ahead = 30, lastn = TRUE,limits = FALSE)
ASE = mean((SWA$arr_delay[148:177]- f$f)^2)
ASE
```

ASE: 309880104

Example: SWA Delays



$$\phi_{13}(1 - B^{12})X_t = a_t$$

```
#Factor Table suggested s = 12
SWA.12 = artrans.wge(SWA$arr_delay[1:141],c(rep(0,11),1))
p = aic5.wge(SWA.12,p = 0:18, q = 0:2)
#AIC picks p = 13
AR13 = est.arma.wge(SWA.12,p = 13)
f = fore.aruma.wge(SWA$arr_delay,phi = AR13$phi, s = 12, n.ahead = 36, lastn = TRUE)
ASE = mean((SWA$arr_delay[142:177] - f$f)^2)
ASE
```

ASE: 215312956

SWA Delay Comarison

AR(12)

ASE: 309880104

$$\phi_{13}(1 - B^{12})X_t = a_t$$

ASE: 215312956

$Best\ NN$

ASE: 2213 27081.7

The seasonal ARIMA model performed best in terms of ASE in this analysis although all are very close and leaders could easily change if the horizon was changed, another sample was taken or if a different number of repeats is used in the NN.

DataScience@SMU

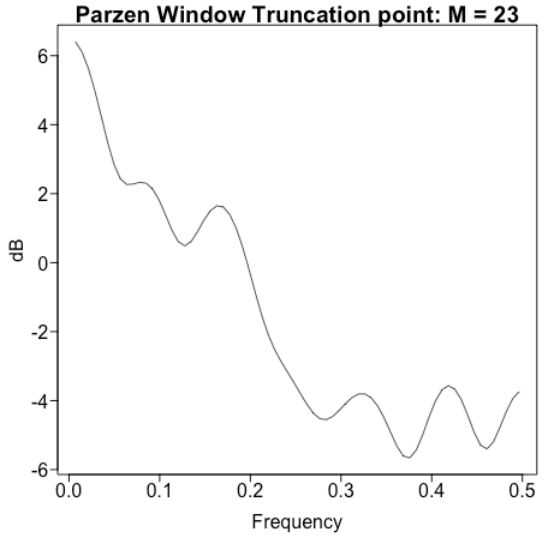
Concept Check

Inspect the spectral density estimate of the Southwest airlines arrival delay data from the last example below. There appears to be some evidence of a peak at 0, around .08 and around .16. This would correspond to wandering behavior, annual period and bi-annual behavior. Which r call below fits 100 neural networks that includes a (1-B), ($1-B^6$) and ($1-B^{12}$) seasonal factor and also allows for AR fitting of the residuals after the differencing?

A. fit.mlp = mlp(SWATrain, difforder = 0, lags = c(1,6,12), allow.det.season = FALSE, reps = 100)

B. fit.mlp = mlp(SWATrain, difforder = c(1,6,12), allow.det.season = FALSE, reps = 100)

C. fit.mlp = mlp(SWATrain, difforder = c(1,6,12), lags = 0, allow.det.season = FALSE, reps = 100)



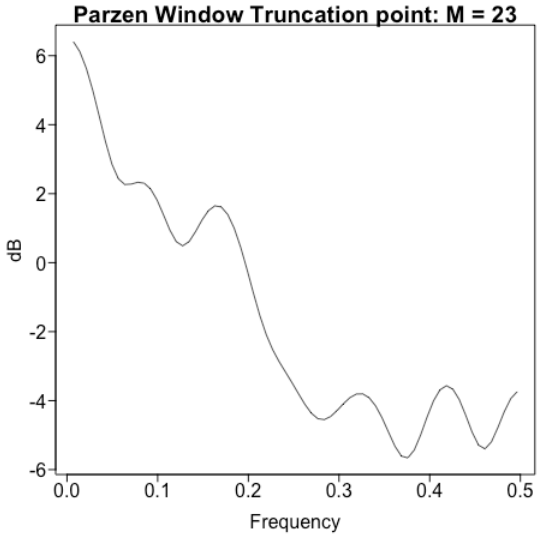
Concept Check

Inspect the spectral density estimate of the Southwest airlines arrival delay data from the last example below. There appears to be some evidence of a peak at 0, around .08 and around .16. This would correspond to wandering behavior, annual period and bi-annual behavior. Which r call below fits 100 neural networks that includes a (1-B), $(1-B^6)$ and $(1-B^{12})$ seasonal factor and also allows for AR fitting of the residuals after the differencing?

A. fit.mlp = mlp(SWATrain, difforder = 0, lags = c(1,6,12), allow.det.season = FALSE, reps = 100)

B. fit.mlp = mlp(SWATrain, difforder = c(1,6,12), allow.det.season = FALSE, reps = 100)

C. fit.mlp = mlp(SWATrain, difforder = c(1,6,12), lags = 0, allow.det.season = FALSE, reps = 100)



Concept Check

Given that the R code below will fit 100 neural networks that include a (1-B), ($1-B^6$) and ($1-B^{12}$) seasonal factor and also allows for AR fitting of the residuals after the differencing, use this code to obtain forecasts of the number of delays in the last 36 months (from October, 2015, to September, 2018.)

fit.mlp = mlp(SWATrain, difforder = c(1,6,12), allow.det.season = FALSE, reps = 100)

1. Does this model have a lower ASE than then the lowest ASE of the neural nets we fit in the example? (200083263.2) YES NO

2. Does this model have a lower ASE than then the lowest ASE of the AR(12) model we fit in a previous unit? (93354237) YES NO

3. Fit the model and find the ASE and display it in a comment below. Everyone's ASE should be different. Why?

Concept Check

Given that the R code below will fit 100 neural networks that include a (1-B), ($1-B^6$) and ($1-B^{12}$) seasonal factor and also allows for AR fitting of the residuals after the differencing, use this code to obtain forecasts of the number of delays in the last 36 months (from October, 2015, to September, 2018.)

fit.mlp = mlp(SWATrain, difforder = c(1,6,12), allow.det.season = FALSE, reps = 100)

1. Does this model have a lower ASE than then the lowest ASE of the neural nets we fit in the example? (200083263.2) YES NO

2. Does this model have a lower ASE than then the lowest ASE of the AR(12) model we fit in a previous unit? (93354237) YES NO

3. Fit the model and find the ASE and display it in a comment below. Everyone's ASE should be different. Why? Answer will vary.

DataScience@SMU

Airline Competition Among the Greats!

Now we will compare the actual models they published on the basis of the ASE calculated from forecasts of the last 36 years.

$$(1-.74B)(1+.38B^{12})(1-B^{12})(X_t-\mu)=a_t$$

AIC

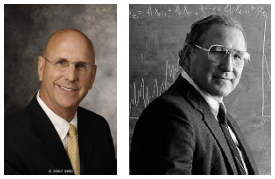
```
> Parzen$value
[1] -6.465559
```

```
> Parzen = fore.aruma.wge(airlog, d = 0, s = 12, phi = c(.74,0,0,0,0,0,0,0,0,0,0,.38,-.2812),n.ahead = 36,lastn = TRUE, limits = FALSE)
> PARZEN_ASE = mean((airlog[(144-36+1):144] - Parzen$f)^2)
> PARZEN_ASE
[1] 0.01252636
```

$$(1-B)(1-B^{12})(X_t-\mu)=(1-.4B)\,(1-.6B^{12})a_t$$

```
> Box$value
[1] -6.499275
```

```
> Box = fore.aruma.wge(airlog,d = 1, s = 12, theta = c(.4,0,0,0,0,0,0,0,0,0,0,.6,-.24),n.ahead = 36,lastn = TRUE, limits = FALSE)
> BOX_ASE = mean((airlog[(144-36+1):144] - Box$f)^2)
> BOX_ASE
[1] 0.006903242
```



$$(1+.36B+.05B^2+.14B^3+.11B^4-.04B^5-.09B^6+.02B^7-.02B^8$$
$$-.17B^9-.03B^{10}+.10B^{11}+.38B^{12})(1-B)(1-B^{12})(X_t-\mu)=a_t$$

```
> Woodward = fore.aruma.wge(airlog,d = 1, s = 12, phi = c(-.36,-.05,-.14,-.11,.04,.09,-.02, .02,.17,.03,-.10,-.38),n.ahead = 36,lastn = TRUE, limits = FALSE)
> WOODWARD_ASE = mean((airlog[(144-36+1):144] - Woodward$f)^2)
> WOODWARD_ASE
[1] 0.004185726
```

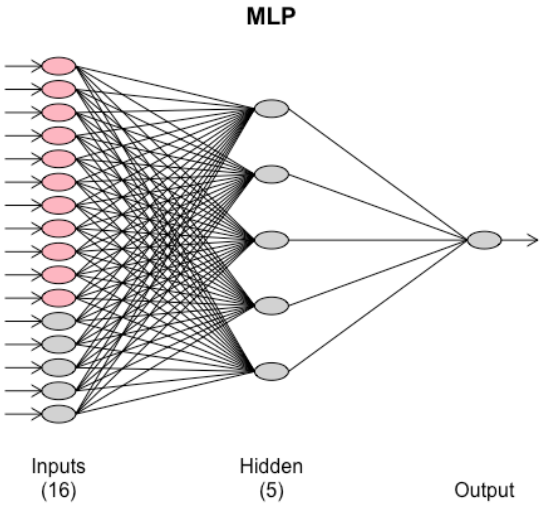
```
> Woodward$value
[1] -6.423649
```

Example: Airline Data

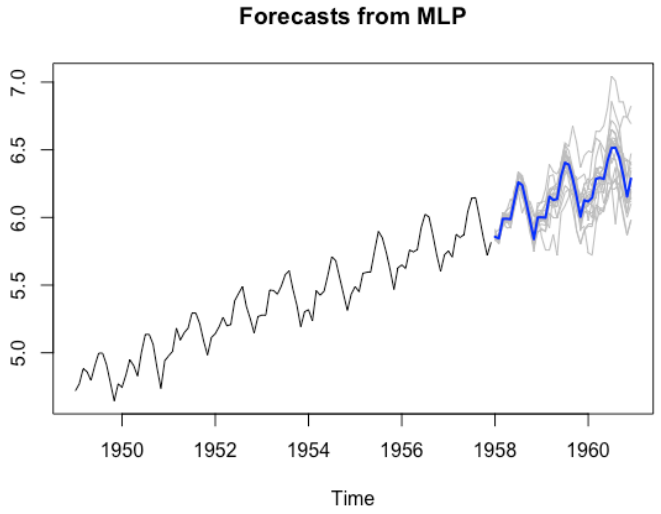
```
# First 108 months in the Training Set.
set.seed(2)
lairTrain = ts(airlog[1:108], frequency = 12, start = c(1949, 1))
 # Last 36 months in the Test set.
lairTest = ts(airlog[109:144], frequency = 12, start = c(1958, 1))
fit.mlp = mlp(lairTrain)
fit.mlp
```

```
MLP fit with 5 hidden nodes and 20 repetitions.
Series modelled in differences: D1.
Univariate lags: (1,3,4,7,8)
Deterministic seasonal dummies included.
Forecast combined using the median operator.
MSE: 0.0001.
```

```
plot(fit.mlp)
```




```
fore.mlp = forecast(fit.mlp, h = 36)
plot(fore.mlp)
```




```
ASE = mean((lairTest - fore.mlp$mean)^2)
ASE
```

```
[1] 0.01826498096
```

```
> WOODWARD_ASE
[1] 0.004185726
```

DataScience@SMU

Concept Check

We know that mlp() uses 5 hidden nodes by default, but can we do better at forecasting sales with more or less? ***hd.auto.type*** is an option in the mlp() function that can select the number of hidden nodes for you. Read the help for mlp() (?mlp) and see what the options are are for this option. We would like to forecast the last 36 observations of the log airline passengers like we did before but this time automatically select the number of hidden nodes with 5 fold cross validation. Use set.seed(2) at the beginning of the code so that we get the same answers.

1. Did it take longer to run than without the hd.auto.type call? YES NO

2. How many hidden nodes were selected by hd.auto.type?

3. What is the ASE for this model?

Concept Check

We know that mlp() uses 5 hidden nodes by default, but can we do better at forecasting sales with more or less? ***hd.auto.type*** is an option in the mlp() function that can select the number of hidden nodes for you. Read the help for mlp() (?mlp) and see what the options are are for this option. We would like to forecast the last 36 observations of the log airline passengers like we did before but this time automatically select the number of hidden nodes with 5 fold cross validation. Use set.seed(2) at the beginning of the code so that we get the same answers.

1. Did it take longer to run than without the hd.auto.type call? YES NO

2. How many hidden nodes were selected by hd.auto.type? 2

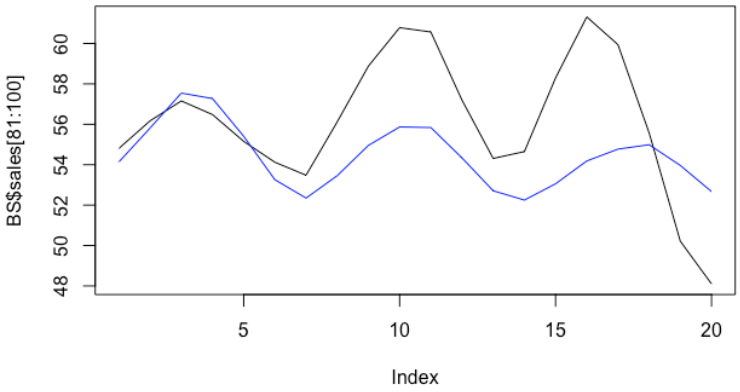
3. What is the ASE for this model? .017

DataScience@SMU

Example:
Multivariate NNs with the Sales Data

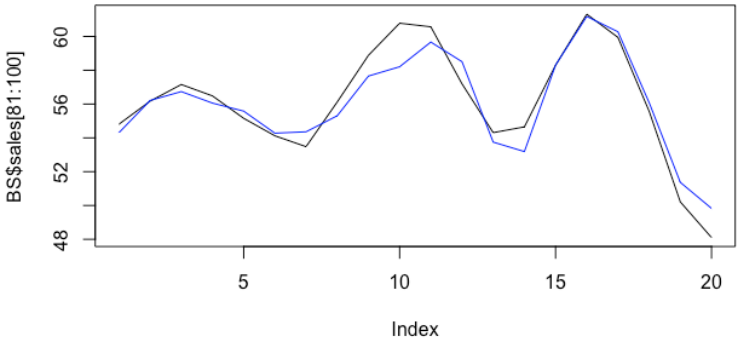
```
#BS is the Business data
# Only Time as a regressor
tBS80 = ts(BS$sales[1:80])
set.seed(2)
fit3 = mlp(tBS80)
f = forecast(fit3, h = 20)
plot(BS$sales[81:100],type = "l")
lines(seq(1,20),f$mean, col = "blue")
ASE = mean((BS$sales[81:100]-f$mean)^2)
ASE
```

```
[1] 11.46737496
```



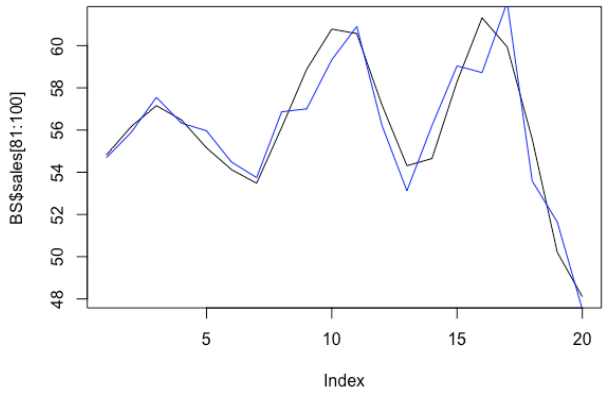

```
#With additional Regressors
set.seed(2)
tBS80 = ts(BS$sales[1:80])
tBSx = data.frame(ad_tv = ts(BS$ad_tv), ad_online = ts(BS$ad_online, frequency = 7),discount = ts(BS$discount))
fit3 = mlp(tBS80,xreg = tBSx)
f = forecast(fit3, h = 20, xreg = tBSx)
plot(BS$sales[81:100],type = "l")
lines(seq(1,20),f$mean, col = "blue")
ASE = mean((BS$sales[81:100]-f$mean)^2)
ASE
```

```
[1] 1.004041793
```



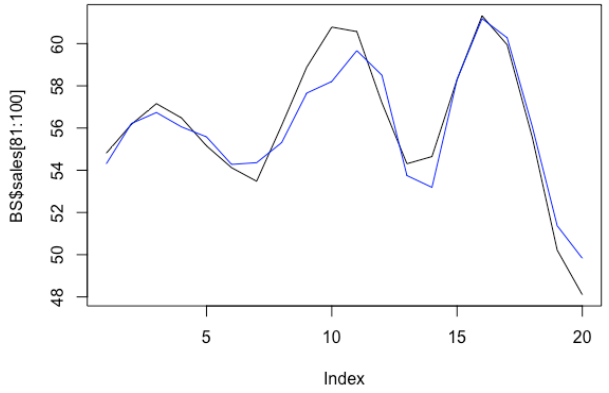
Example:
Multivariate NNs with the Sales Data

```
#ARIMA Model with Regressors
ad_tv1 = c(NA,BS$ad_tv[1:(length(BS$ad_tv)-1)])
ad_online1 = c(NA,BS$ad_online[1:(length(BS$ad_online)-1)])
BS$ad_tv1= ad_tv1
BS$ad_online1 = ad_online1
ksfit=lm(sales~ad_tv1+ad_online1+discount, data = BS)
aic.wge(ksfit$residuals,p=0:8,q=0:0)  # AIC picks p=7
fit=arima(BS$sales,order=c(7,0,0),xreg=cbind(BS$ad_tv1, BS$ad_online1,
BS$discount))
preds = forecast(fit,h = 20, xreg =
cbind(BS$ad_tv1[81:100],BS$ad_online1[81:100],BS$discount[81:100]))
plot(BS$sales[81:100],type = "l")
lines(seq(1,20),preds$mean, col = "blue")
ASE = mean((BS$sales[81:100]-preds$mean)^2)
ASE
```




```
[1] 1.5140944
```

```
#NN With additional Regressors
set.seed(2)
tBS80 = ts(BS$sales[1:80])
tBSx = data.frame(ad_tv = ts(BS$ad_tv), ad_online = ts(BS$ad_online,
frequency = 7),discount = ts(BS$discount))
fit3 = mlp(tBS80,xreg = tBSx)
f = forecast(fit3, h = 20, xreg = tBSx)
plot(BS$sales[81:100],type = "l")
lines(seq(1,20),f$mean, col = "blue")
ASE = mean((BS$sales[81:100]-f$mean)^2)
ASE
```




```
[1] 1.004041793
```

DataScience@SMU

Concept Check:

We would like to now forecast the sales data with the 3 regressors (ad_tv, ad_online and discount) but this time automatically select the number of hidden nodes with 5 fold cross validation. Remember to use set.seed(2) so that our answers match!

1. How many nodes were selected by hd.auto.type?

2. What was the corresponding ASE? (Round to 3 decimal places)

Concept Check:

We would like to now forecast the sales data with the 3 regressors (ad_tv, ad_online and discount) but this time automatically select the number of hidden nodes with 5 fold cross validation. Remember to use set.seed(2) so that our answers match!

1. How many nodes were selected by hd.auto.type? 1

2. What was the corresponding ASE? (Round to 3 decimal places)

.736

DataScience@SMU

Interview with Mikio Braun

DataScience@SMU

For Live Session

- Schumway Data
- Sunspot / Melanoma Data
- SWA DELAY DATA
- Recurrent NN

DataScience@SMU

Closing Remarks

DataScience@SMU