Machine Learning, Deep Learning, Cross Learning, and Hybrid Models

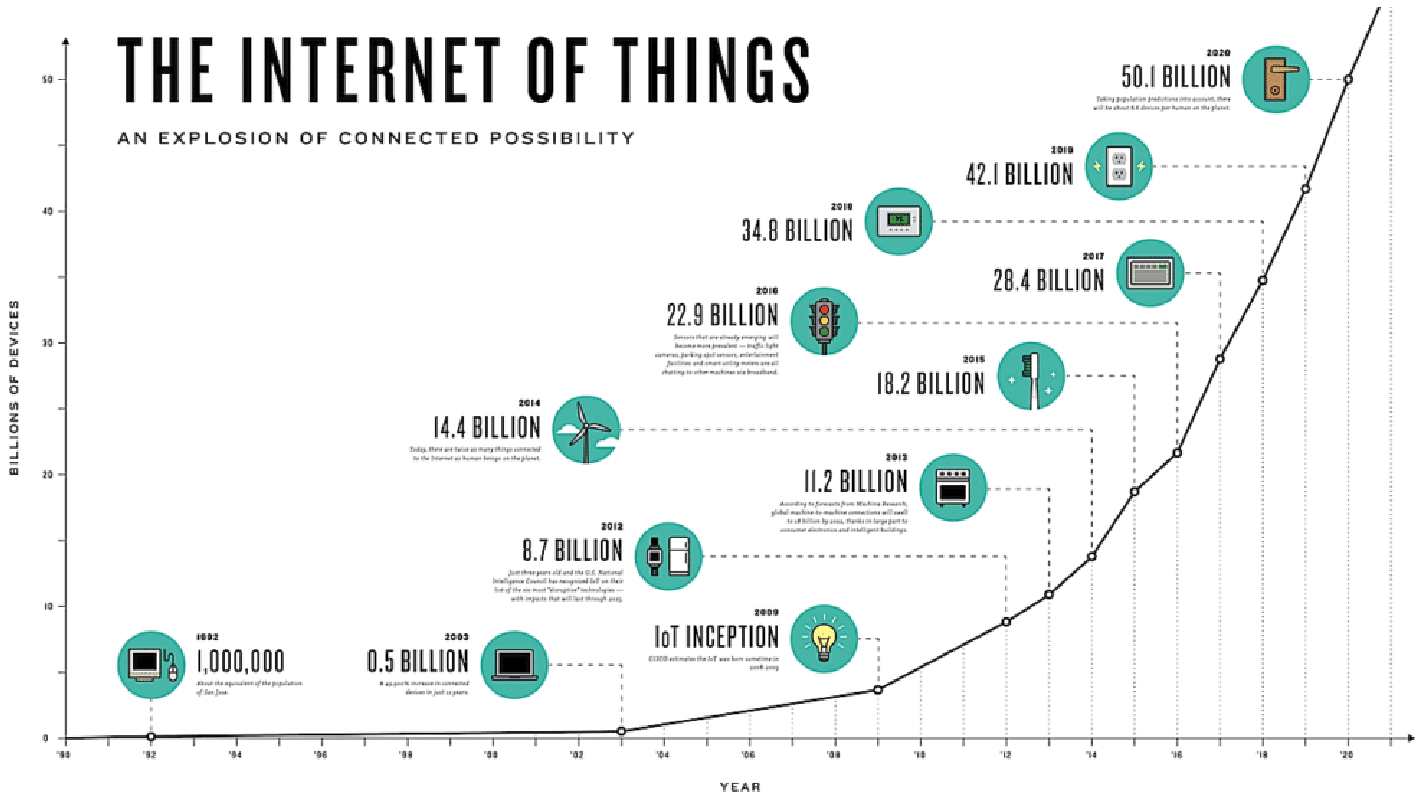
Week 3: Explanatory and ML methods

UNIVERSITY *of* NICOSIA

Machine Learning – Present & Future

The first steps in ML development were done in the early 1800. However, ML usage has been exponentially increased after the 90s due to:

- ➢ **Advances in ML algorithms**
- ➢ **Data availability**
- ➢ **Computational power and storage**



Advances in ML algorithms

1805	Legendre - *Least square errors*
1958	Rosenblatt – *Perceptron (ANN)*
1965	Ivakhnenko - *Deep Learning (DNN)*
1986	Rumelhart – *Back propagation*
1989	LeCun – *Convolutional NNs (CNNs)*
1992	Boser – *Support Vector Machines (SVMs)*
1997	Hochreiter– *Recurrent NNs (RNNs)*
1998	Brin & Page – *Google Search*
2006	Hinton- *Efficient Deep Learning*

Data availability

1991	*Word Wide Web (www)*
2000	*Broadband*
2004	*Facebook & Web 2.0*
2005	*YouTube & 1 billion web users*
2007	*iPhone & smartphone revolution*
2010	*Cloud & 20 exabytes of data*
2012	*1 billion Facebook users*
2014	*More smartphones than people*
2017	*2.5 quintillion bytes of data*

Computational power and storage

1965	*Moore's law*
1999	*Nvidia GeForce & GPUs*
2002	*Amazon web services*
2004	*Map reduce algorithm*
2005	*99% decrease in storage cost*
2006	*Hadoop*
2009	*Spark and advanced GPUs*
2017	*Google TPU (Tensor Processing Unit)*

Machine Learning

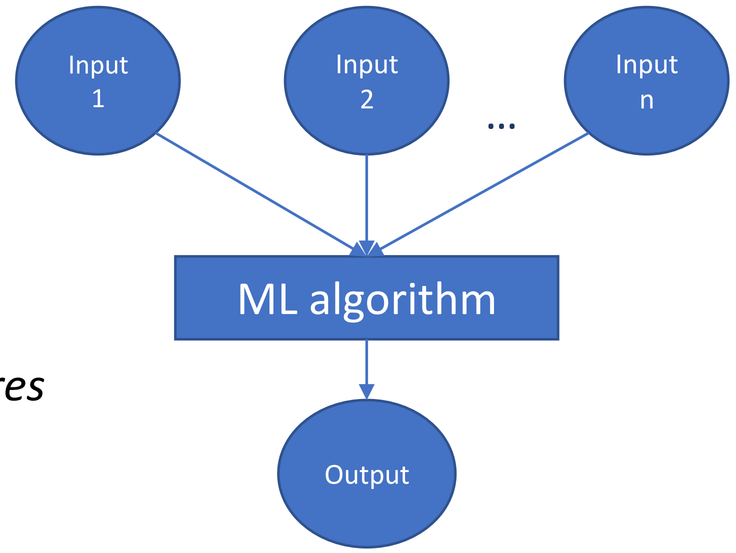
- ✓ Involves a variety of algorithms that exploit **big data** in order to solve regression*, classification, and clustering problems
- ✓ Their strength, when compared to statistical models, is that ML algorithms **make few or no assumptions** about the data generation process, thus being more **generic and flexible**
- ✓ As new and more data become available, ML algorithms are improved in terms of accuracy by learning from their previous mistakes**

* in forecasting applications, this typically includes auto-regression approaches
** in forecasting applications, this may refer to meta-learners that adjust the new forecasts based on past errors and backcasting processes considered during training

Machine Learning: *Supervised learning*

✓ They **know** the **correct answers to past problems** and try to find relationships between the variables provided as input so that the target* variable(s) is accurately approximated by the predictor* variable(s)

- Neural Networks
- Decision trees
  - ✓ Random Forest
  - ✓ Gradient boosting trees
- Support Vector Machine



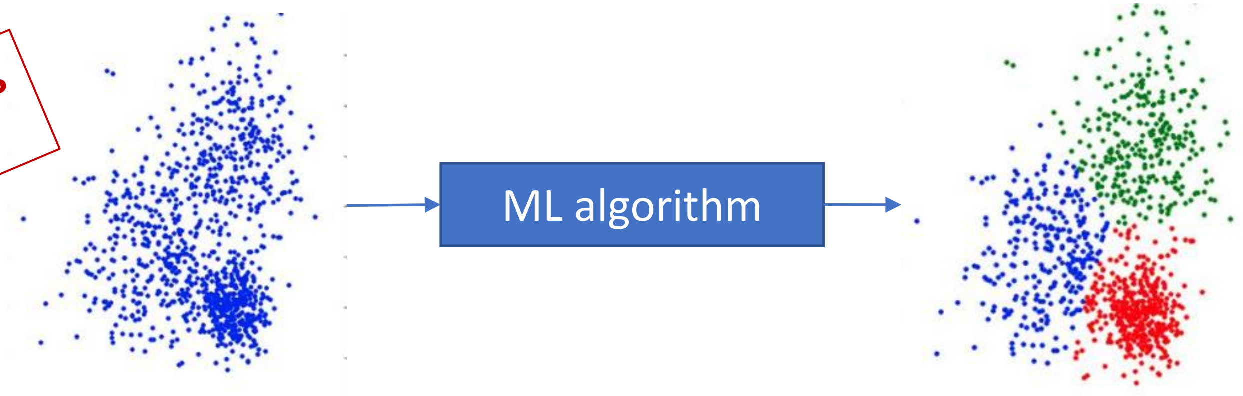
* *Input, Independent, Regressor, Explanatory, Predictor* variables or just *Features*
** *Output, Dependent, Target or Explained* variables

Machine Learning: *Unsupervised learning*

✓ They **do not know** the **correct answers to past problems** and try to explore the possible relationships between the variables provided as input so that the objects under examination are properly grouped

- K-means clustering
- Hierarchical clustering
- Gaussian mixture model
- Recommender system

Typically not used for forecasting purposes
at lest not for producing forecasts directly

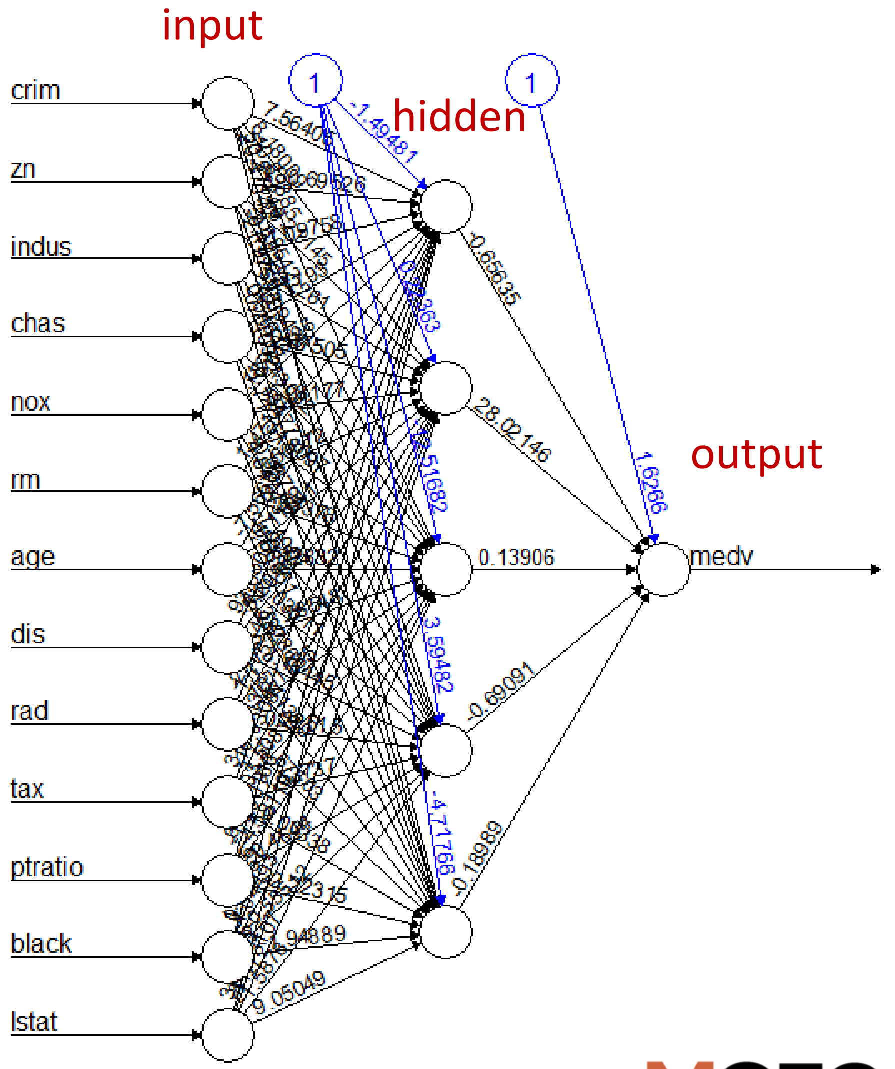


UNIC

MOFC

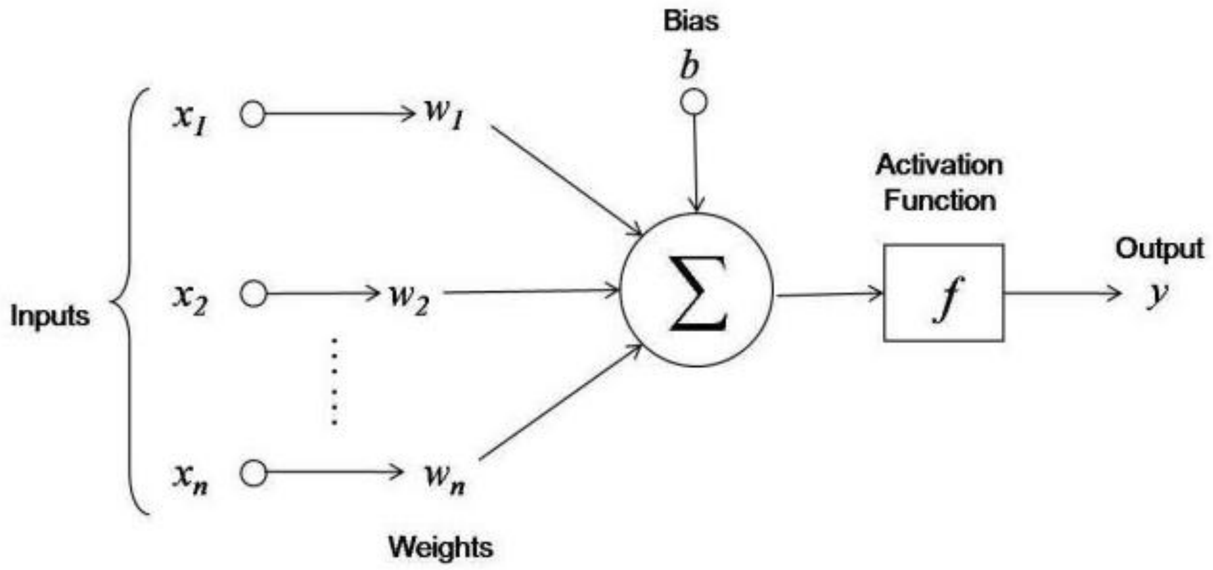
Neural Networks

- Allow for **non-linear** regression
- The model connects the regressor variables, assigned to the **input layer**, to the target variables, assigned to the **output layer**, by using one or more additional, **hidden layers** for performing the computations
- The underlying relationships are identified through a process that mimics the way the **human brain** operates, i.e., by starting from zero knowledge and learning through trial and error.



Neural Networks - Perceptron

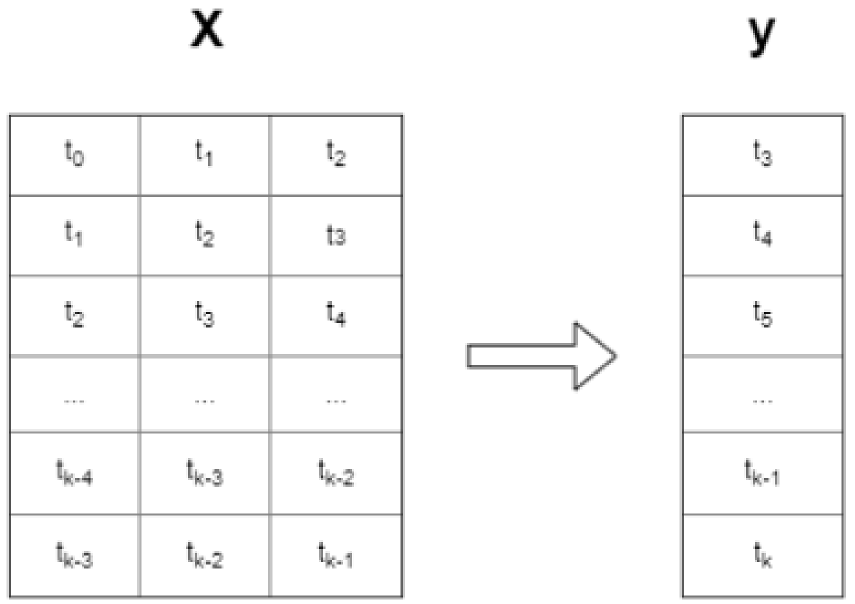
- Each layer consists of one or more nodes, and each **node** is a perceptron, being similar to a multiple linear regression model: This is why we typically call NNs **Multilayer Perceptrons** (MLPs)
- The perceptron feeds the signal produced by a multiple linear regression into an **activation function** that may be nonlinear, such as a *sigmoid* or a *tanh* one. Accordingly, relevant **scaling** transformations should be considered
- Different activation functions can be used for each layer. Typically, the output layer uses a linear (identity) function, to be able to **account for trend**, while the rest a sigmoid or *relu* one
- When NNs employ identity functions across all layers, then they approximate linear regression and ARIMA models



Neural Networks - Training

- The dataset is divided into individual **observations**, i.e., matches of regressor and target variables
- Each observation is sequentially imported to the network and, based on its results, the weights are updated. If the initial weights were a good guess, then the update is minor, while if the weights resulted in bad accuracy, the opposite happens. However, this also depends on the *learning rate*.
- In fact, more than one observation may be imported to the network simultaneously. These are called **batches**: Smaller batches allow for *more detailed learning* but are easier to lead to *overfitting*
- Since the weights are updated from the output layer to the input layer, i.e., from back to front, this process of learning i called “**backpropagation**”
- In practice, there are numerous ways of applying the backpropagation algorithm, considering variations of it so that the model
  - ✓ is more robust to over-fitting
  - ✓ converges faster

Neural Networks for time series forecasting

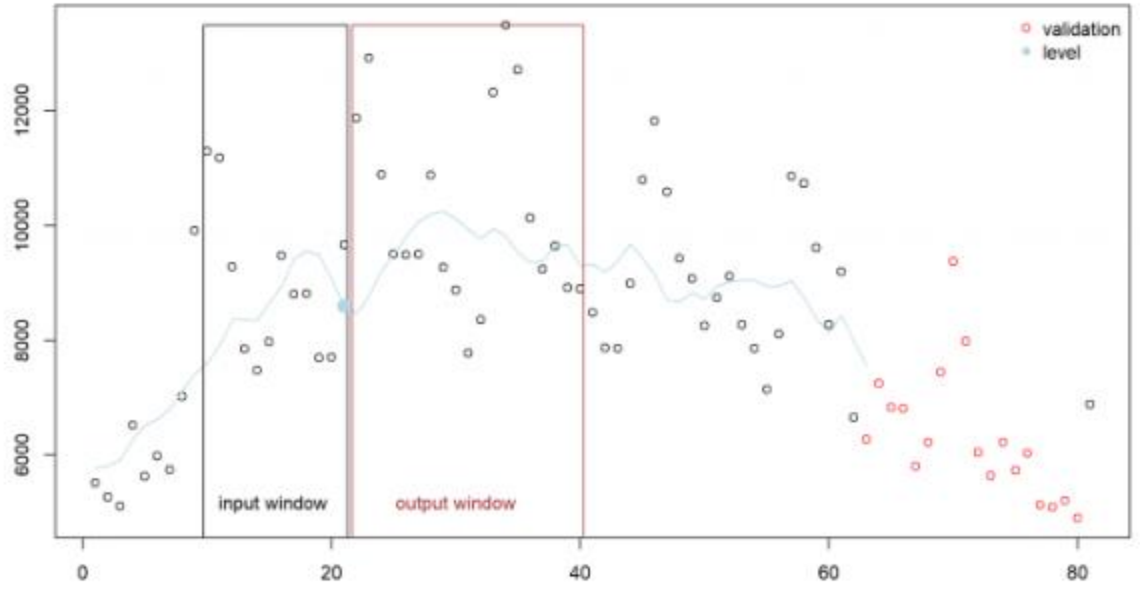


x

t_0	t_1	t_2
t_1	t_2	t_3
t_2	t_3	t_4
...	...	...
t_{k-4}	t_{k-3}	t_{k-2}
t_{k-3}	t_{k-2}	t_{k-1}

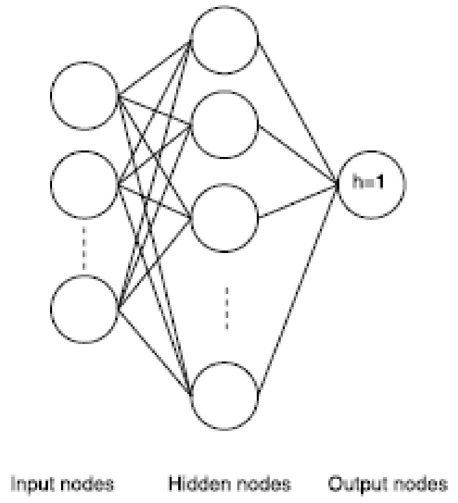
y

t_3
t_4
t_5
...
t_{k-1}
t_k

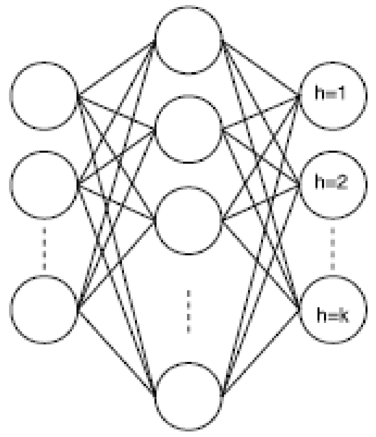


- ✓ For time series forecasting, the regression model is converted into an auto-regression model where each observation Yt is predicting using past observations Yt-1, Yt-2,…. Yt-k
- ✓ However, using additional features as external variables is always possible
- ✓ Determining the **look-back window**, *k*, is one of the hyper-parameters that should be specified before training

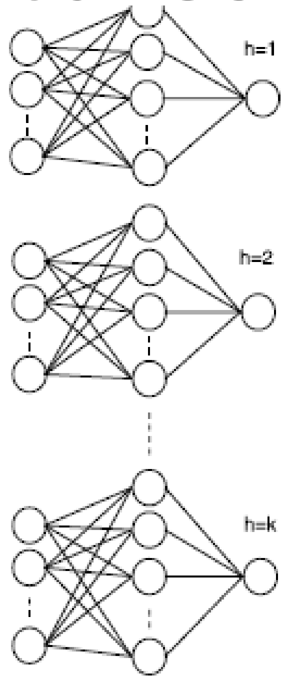
Neural Networks - Architectures



Iterative/Reccurrent



Direct



Multi-NN

- ✓ Produce an one-step-ahead forecast. Then, use this forecast, along with the actual past observations, to compute the following one-step-ahead forecast, and so on, till *h* forecasts are produced.
- ✓ Directly produce *h* forecasts. Each output node represents a forecasting horizon.
- ✓ Each NN is specialized in predicting at a different forecasting horizon.

UNIC

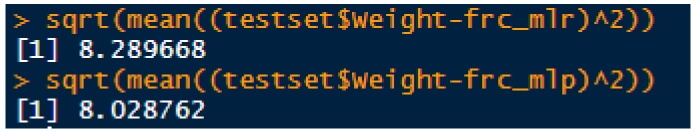
MOFC

Neural Networks in R (1/2)

```
dataset <- read.csv("PATH/Heights-Weights.csv", stringsAsFactors = F)

#Multiple linear regression
trainset <- head(dataset, 25) # data used for training
testset <- tail(dataset, 5) # data used for testing
mlr <- lm(Weight ~ Age + Height + ShoeSize, data=trainset)
frc_mlr <- as.numeric(predict(mlr, testset))

#Simple NN - neuralnet
trainset_s <- trainset #data used for training - scaled
trainset_s$Age <- (trainset_s$Age-min(trainset_s$Age))/(max(trainset_s$Age)-min(trainset_s$Age))
trainset_s$Height <- (trainset_s$Height-min(trainset_s$Height))/(max(trainset_s$Height)-min(trainset_s$Height))
trainset_s$ShoeSize <- (trainset_s$ShoeSize-min(trainset_s$ShoeSize))/(max(trainset_s$ShoeSize)-min(trainset_s$ShoeSize))
trainset_s$Weight <- (trainset_s$Weight-min(trainset_s$Weight))/(max(trainset_s$Weight)-min(trainset_s$Weight))

testset_s <- testset # data used for testing
testset_s$Age <- (testset_s$Age-min(trainset$Age))/(max(trainset$Age)-min(trainset$Age))
testset_s$Height <- (testset_s$Height-min(trainset$Height))/(max(trainset$Height)-min(trainset$Height))
testset_s$ShoeSize <- (testset_s$ShoeSize-min(trainset$ShoeSize))/(max(trainset$ShoeSize)-min(trainset$ShoeSize))

library(neuralnet)
set.seed(512)
mlp1 <- neuralnet(Weight ~ Age + Height + ShoeSize, data=trainset_s,
            hidden = 1, act.fct = "logistic", linear.output = FALSE)
frc_mlp_s <- as.numeric(predict(mlp1, testset_s))
frc_mlp <- frc_mlp_s*(max(trainset$Weight)-min(trainset$Weight))+min(trainset$Weight)
data.frame(frc_mlp,testset$Weight)

#RMSE of linear and non-linear models
sqrt(mean((testset$Weight-frc_mlr)^2))
sqrt(mean((testset$Weight-frc_mlp)^2))
```

It is important to scale each variables into 0-1 to facilitate training

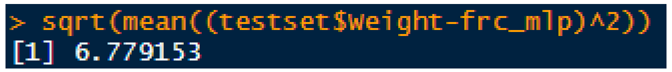
One may also consider ensembles of multiple NNs, even of the same hyper-parameters, to mitigate parameter uncertainty due to weight initialization



```
> sqrt(mean((testset$Weight-frc_mlr)^2))
[1] 8.289668
> sqrt(mean((testset$Weight-frc_mlp)^2))
[1] 8.028762
```

Neural Networks in R (2/2)

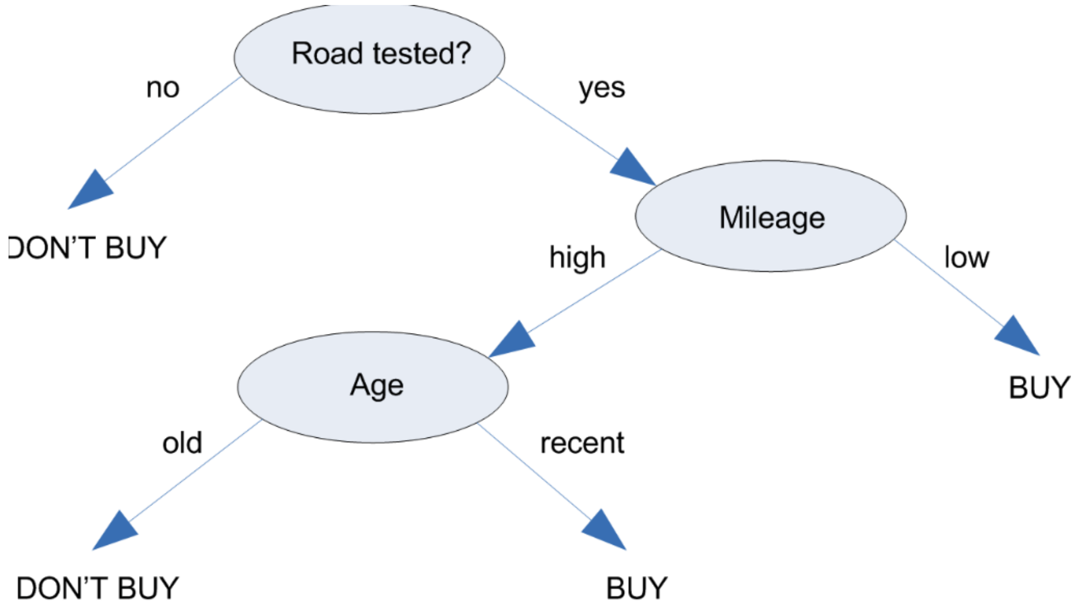
```
library(RSNNS)
set.seed(512)
mlp2 <- mlp(x=trainset_s[,c("Age", "Height", "ShoeSize")],
        y=trainset_s$Weight, size = c(5), maxit = 100,
        learnFunc = "SCG",
        hiddenActFunc = "Act_Logistic", linOut = FALSE)
frc_mlp_s <- as.numeric(predict(mlp2, testset_s[,c("Age", "Height", "ShoeSize")]))
frc_mlp <- frc_mlp_s*(max(trainset$Weight)-min(trainset$Weight))+min(trainset$Weight)
data.frame(frc_mlp,testset$Weight)
sqrt(mean((testset$Weight-frc_mlp)^2))
```



```
> sqrt(mean((testset$Weight-frc_mlp)^2))
[1] 6.779153
```

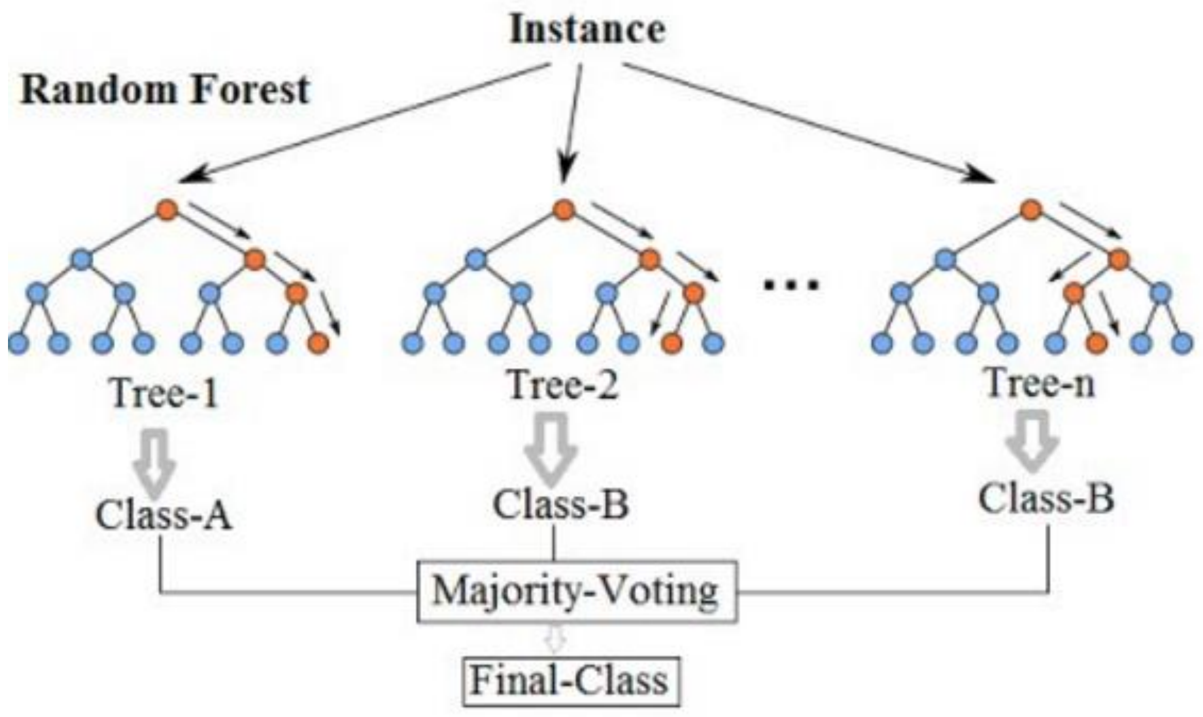
Decision trees

- Can be used for performing a tree-like recursive partitioning of the input space, thus dividing it into regions, called the *terminal leaves*
- Then, based on the inputs provided, tests are applied to decision nodes in order to define which leave should be used for forecasting
- The partition rules are defined so that the overall error is reduced at each step by the maximum extent (greedy approach)



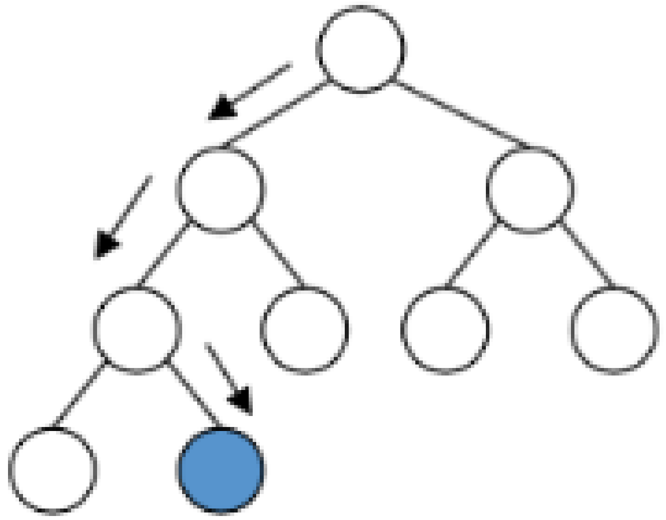
Random forests

- RF expand the concept of RT by combining the results of **multiple RTs**
- Each RT depends on the values of a random vector
  - ✓ sampled **independently** and with the **same distribution**
  - ✓ RF is more **robust to outliers** and **over-fitting**, even for limited samples of data



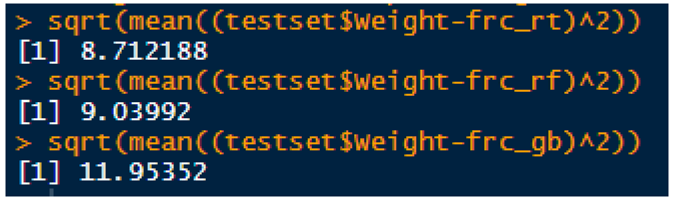
Gradient boosting trees

- This method is similar to RF, but instead of generating multiple independent trees, it builds **one tree at a time**, each new tree **correcting** the errors made by the previously trained one
- Thus, although GBT is more specialized than RF in forecasting the target variable, is more sensitive to over-fitting.
- Cross-validation procedures are typically considered to mitigate over-fitting.



Decision trees in R

```
#Regression tree
library(rpart)
rt <- rpart(Weight ~ Age + Height + ShoeSize,
            method="anova", data=trainset)
frc_rt <- as.numeric(predict(rt, testset))

#Random forest
library(randomForest)
rf <- randomForest(Weight ~ Age + Height + ShoeSize,
                   data = trainset, ntree = 1000)
frc_rf <- as.numeric(predict(rf, testset))

#Gradient boosting
library(gbm)
gb =gbm(Weight ~ Age + Height + ShoeSize, data = rbind(trainset,trainset),
        distribution = "gaussian", n.trees = 50,
        shrinkage = 0.01, interaction.depth = 4)
frc_gb <- as.numeric(predict(gb, testset, n.trees = 50))

sqrt(mean((testset$Weight-frc_rt)^2))
sqrt(mean((testset$Weight-frc_rf)^2))
sqrt(mean((testset$Weight-frc_gb)^2))
```

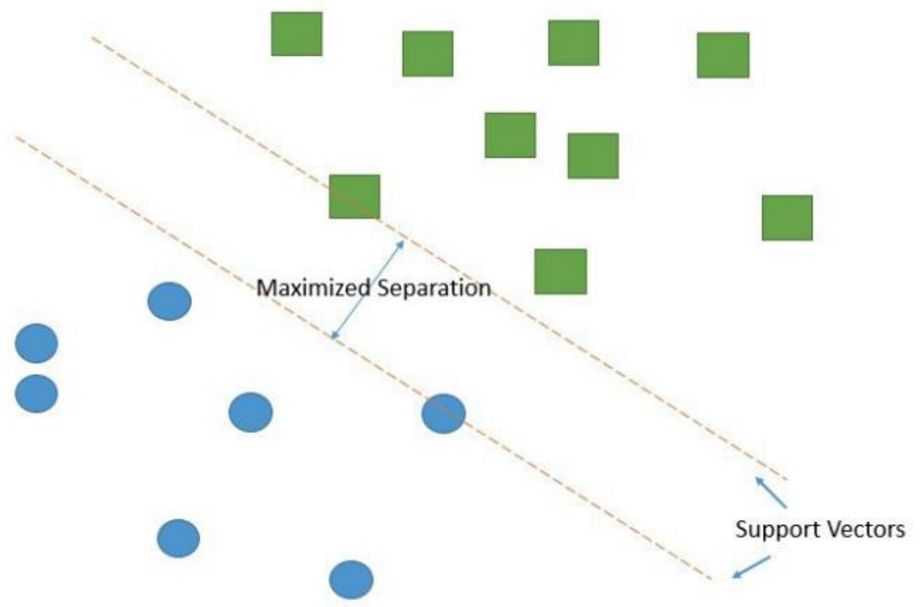


```
> sqrt(mean((testset$Weight-frc_rt)^2))
[1] 8.712188
> sqrt(mean((testset$Weight-frc_rf)^2))
[1] 9.03992
> sqrt(mean((testset$Weight-frc_gb)^2))
[1] 11.95352
```

Support Vector Regression

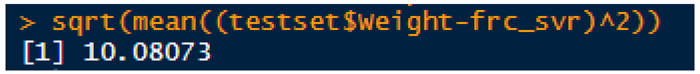
- SVR generates forecasts by identifying the hyper-plane that maximizes the margin between two classes and minimizes the total error under tolerance

- Thus, an SVR model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible.



Rcode

```
library(e1071)
svr <- svm(trainset[,c("Age", "Height", "ShoeSize")], trainset$Weight,
        method="eps-regression")
frc_svr <- as.numeric(predict(svr, testset[,c("Age", "Height", "ShoeSize")]))
sqrt(mean((testset$Weight-frc_svr)^2))
```



```
> sqrt(mean((testset$Weight-frc_svr)^2))
[1] 10.08073
```

UNIC

MOFC

Hybrid ML models (1/2)

Smyl’s **M4 winning method**:
Holt-Winters (exponential smoothing) and NN models are merged into one

$$l_t = \alpha(y_t/s_t) + (1-\alpha)l_{t-1}$$
$$s_{t+m} = \gamma(y_t/l_t) + (1-\gamma)s_t$$

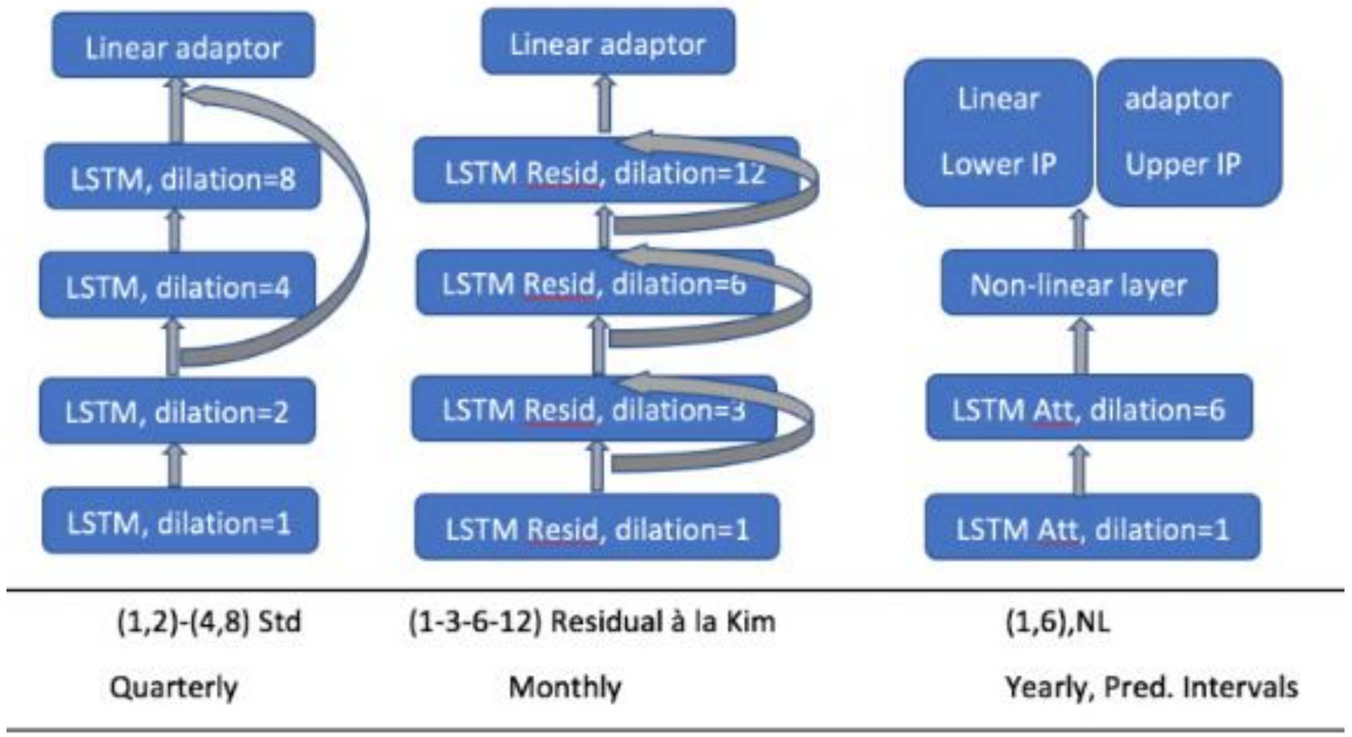
$$\widehat{y}_{t+1..t+h} = RNN(X_t) * l_t * s_{t+1..t+h}$$

Exponential Smoothing
Used for capturing level and seasonality

NN
Used for capturing trend

- Note that ***X*** refers to the preprocessed, normalized data, i.e., the **deseasonalized** time series, as provided by dividing the original series with the Holt-Winters forecasts
- Smoothing parameters are estimated **for each series individually**, while the **NN is global** and trained on all series (**cross-learning**)

Hybrid ML models (2/2)



- A special model is used per data frequency
- The forecasts are produced by considering both the particular characteristics of each series (**local** model) and the particularities of the complete dataset (**global** model)
- Hybrid approaches are effective in a sense that, if the data used for training do not display the **same seasonality and trend patterns**, using global information alone is inappropriate.

Cross learning

Montero-Manso's **M4 second-best performing method**:
ML is used for weighting the forecasts produced by various forecasting methods using time series features

- A collection of time series is used for training a meta-learner model for assigning weights to various possible forecasting methods with the goal of minimizing the average forecasting loss obtained from a weighted forecast combination
- The inputs to the meta-model are **features** that are extracted from each series, such as trend, seasonality, randomness, and autocorrelation.
- The new series are being forecasted using a weighted forecast combination, where the weights are obtained from the previously trained meta-model

Cross learning

Performance of ML methods on the M4 Competition's yearly data, as measured using the sMAPE accuracy measure

The following ML methods were considered as indicative of the different eras:

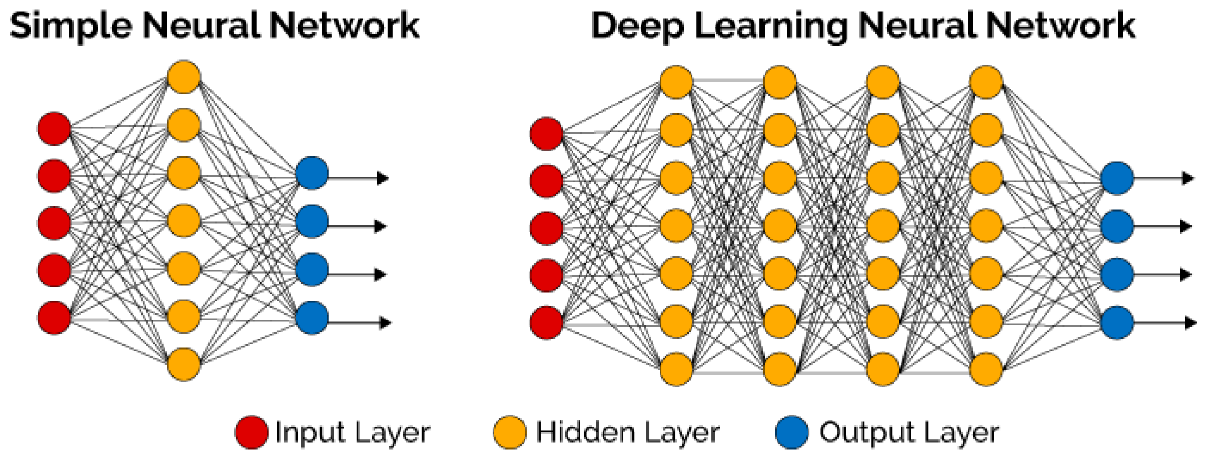
- Simple MLP trained in a "series-by-series" fashion (MLP-SBS)
- Simple MLP trained in a cross-learning fashion (MLP-CL)
- Slawek Smyl's winning submission in the M4 (ES-RNN)
- CNN based sequence-to-sequence network (CNN)
- N-BEATS neural architecture (N-BEATS)

Method	sMAPE (%)	Difference from Theta (%)
Theta	14.593	-
MLP - SBS	21.764	+ 49.1 %
MLP - CL	13.645	- 6.5 %
ES - RNN	13.176	- 9.7 %
CNN	13.086	- 10.3 %
N - BEATS	12.913	- 11.5 %

Deep Learning

- ✓ NNs that exploit multiple layers for training, so that more information is extracted, are typically called **Deep Learning** (DL) NNs.
- ✓ However, using 2, 10 or 500 layers does **not** automatically converts a NN into a DL model
- ✓ Each layer or part of the network's architecture **should structurally differ** from the other layers, in a sense that each layer should be extracting a different kind of information from the dataset.

- ✓ Usual classes of DL model:
  - Convolutional NNs (CNN)
  - Recurrent NNs (RNN) – Like NNs



Deep learning with Gluon TS (1/2)

Gluon Time Series (**GluonTS**) is a tool for probabilistic time series modeling, **focusing on deep learning-based models**

GluonTS provides utilities for loading and iterating over time series datasets, **state of the art models** ready to be trained, and building blocks to define your own models and quickly experiment with different solutions.

- Train and evaluate any of the built-in models on your own data, and quickly come up with a solution for your time series tasks
- Use the provided abstractions and building blocks to create custom time series models, and rapidly benchmark them against baseline algorithms

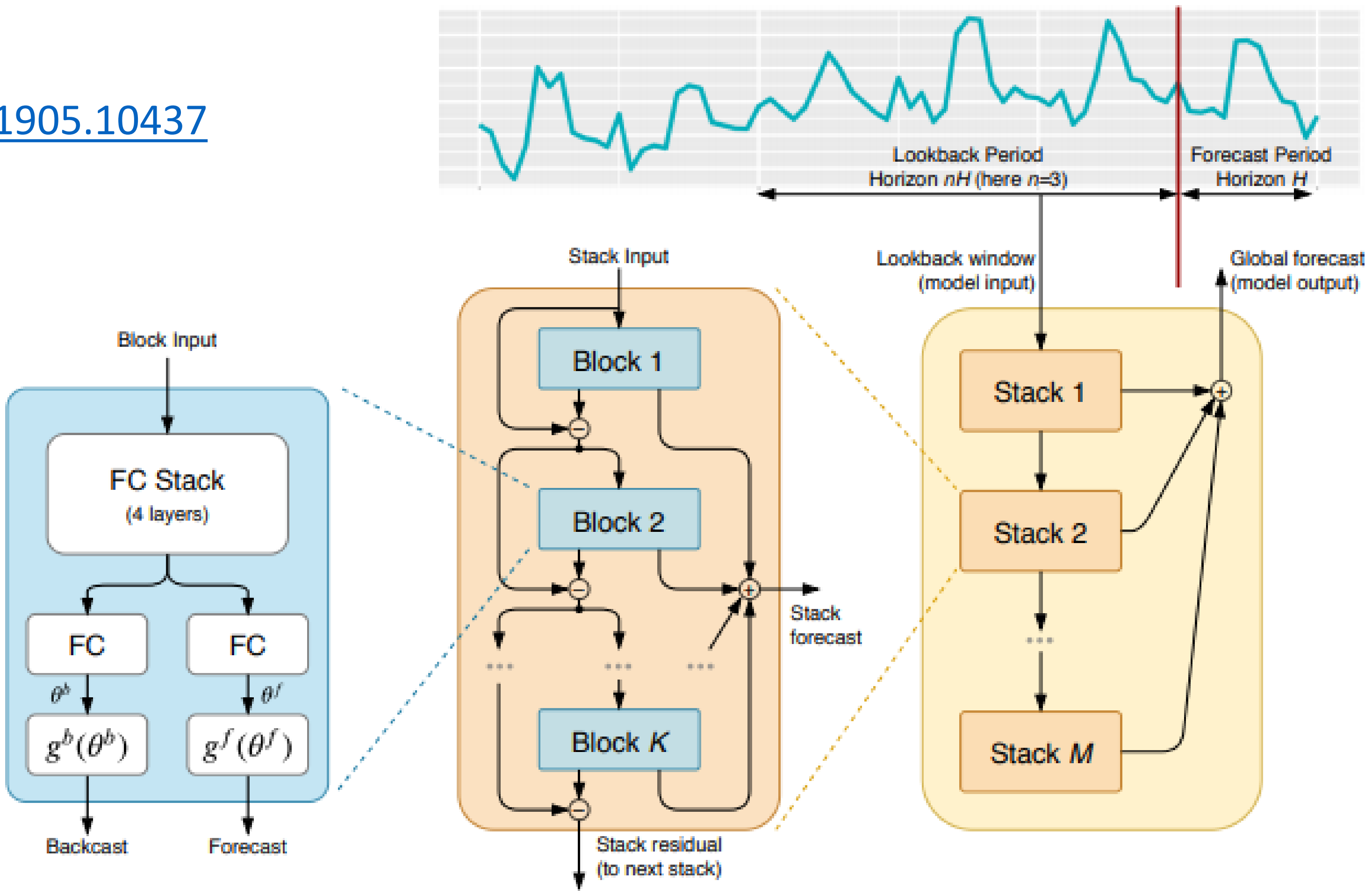
Available at:
https://gluon-ts.mxnet.io/

Deep learning with Gluon TS (2/2)

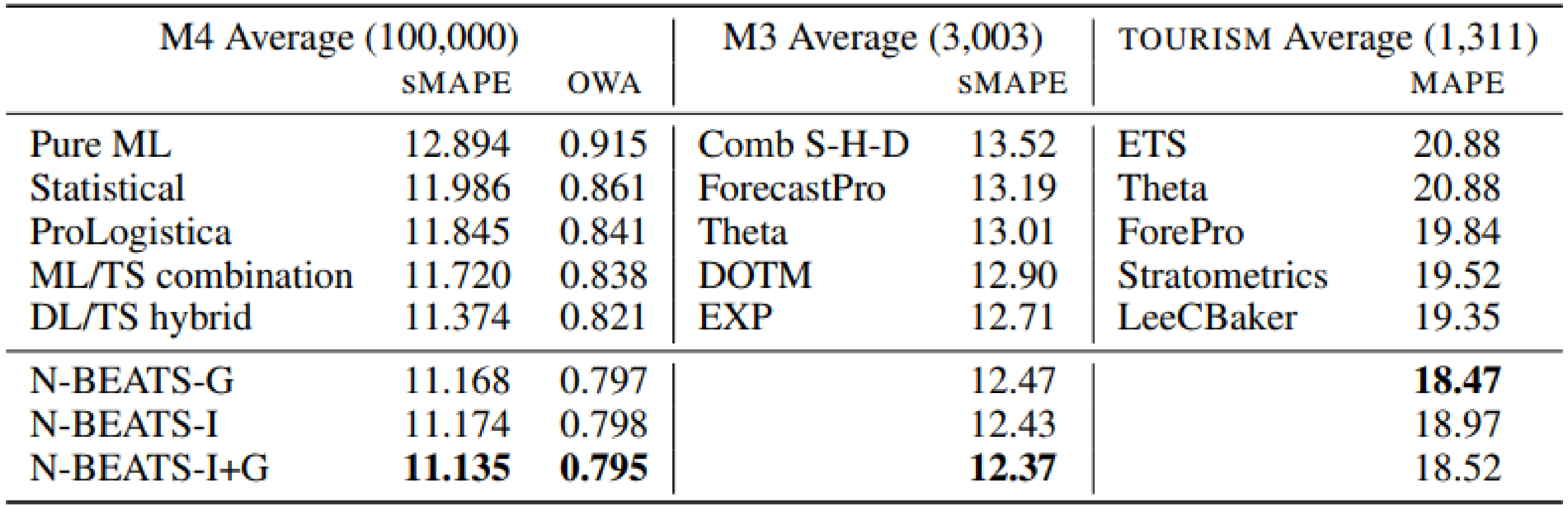
	The accuracy (sMAPE and MASE) of the M3 monthly data (1045 series), short, medium and long horizons and overall average , Tables 8 and 9 in PLOS ONE paper										
			Table 8 of PLOS ONE paper				Table 9 of PLOS ONE paper				
			sMAPE				MASE				
	Methods	**Total Number of Models**	**Short**	**Medium**	**Long**	**Average**	**Short**	**Medium**	**Long**	**Average**	**Relative Computational Complexity: Naïve 2=1**
Stat: Plos One	Naive2	1	10.78	12.46	15.08	12.77	0.76	1.05	1.35	1.05	1
	Theta	1	8.96	***10.53***	***13.19***	***10.89***	0.64	***0.89***	***1.17***	0.90	1.1
	ARIMA	1	***8.93***	11.08	13.84	11.28	**0.61**	0.89	1.17	***0.89***	49.7
	ETS	1	9.07	10.98	13.74	11.26	0.64	0.92	1.21	0.92	38.3
ML: Plos One	MLP	1	9.53	12.34	15.00	12.29	0.66	0.98	1.24	0.96	165.9
	BNN	1	9.39	12.08	14.80	12.09	0.64	0.94	1.20	0.93	95.9
DL: Gluon TS	DeepAR	50	9.23	10.91	13.60	11.25	0.65	0.90	1.23	0.93	313,000
	SimpleFeedForward	50	10.09	10.75	13.69	11.51	0.70	0.88	1.16	0.91	47,300
	Transformer	50	9.21	11.09	13.80	11.36	0.67	0.94	1.25	0.96	47,500
	WaveNet	50	10.41	10.66	13.45	11.51	0.80	0.92	1.19	0.97	306,000
	Ensemble: DeepAR+SimpleFeedForward+Transformer+WaveNet	200	**8.83**	**9.90**	**12.50**	**10.41**	***0.62***	**0.83**	**1.12**	**0.86**	714,000
Improve-ments	***Improvement of Gluon TS ensemble over Naïve 2***		***1.95***	***2.56***	***2.58***	***2.36***	***0.14***	***0.22***	***0.23***	***0.19***	
	% Improvement of Gluon TS ensemble over Naïve 2		***18.0%***	***20.5%***	***17.1%***	***18.5%***	***18.4%***	***21.0%***	***17.0%***	***18.1%***	
	Improvement of Gluon TS ensemble over Theta		**0.13**	**0.63**	**0.69**	**0.48**	**0.02**	**0.06**	**0.05**	**0.04**	
	% Improvement of Gluon TS ensemble over Theta		***1.4%***	***6.0%***	***5.2%***	***4.4%***	***3.1%***	***6.7%***	***4.3%***	***4.4%***	

Deep learning with N-Beats (1/2)

Available at:
https://arxiv.org/abs/1905.10437



Deep learning with N-Beats (2/2)



M4 Average (100,000)			M3 Average (3,003)		TOURISM Average (1,311)	
	sMAPE	OWA		sMAPE		MAPE
Pure ML	12.894	0.915	Comb S-H-D	13.52	ETS	20.88
Statistical	11.986	0.861	ForecastPro	13.19	Theta	20.88
ProLogistica	11.845	0.841	Theta	13.01	ForePro	19.84
ML/TS combination	11.720	0.838	DOTM	12.90	Stratometrics	19.52
DL/TS hybrid	11.374	0.821	EXP	12.71	LeeCBaker	19.35
N-BEATS-G	11.168	0.797		12.47		**18.47**
N-BEATS-I	11.174	0.798		12.43		18.97
N-BEATS-I+G	**11.135**	**0.795**		**12.37**		18.52

UNIC

MOFC

Training dataset size

- ✓ Empirical evidence suggest that more training samples are beneficial, given that the model has the learning capacity to make use of them.
- ✓ In an attempt to increase the number of training samples, new, artificial samples can be created by averaging pairs of the original ones.
- ✓ Empirical study:
  - ➢ Dataset consisted of the 23,000 yearly series from the M4 competition.
  - ➢ A CNN – based model was used to evaluate data augmentation.
  - ➢ Pairs of original samples were chosen and then averaged, producing new artificial training samples.

% of artificial series	Training set augmentation	sMAPE (%)
0 %	-	13.086
50 %	x 2	13.063
90 %	x 10	13.034
95 %	x 20	13.009

UNIC

MOFC

External variables and features (1/2)

The information that can be used as additional an input in ML models usually come in the following forms:

- External variables, holding information about environmental conditions that may affect the series, such as the weather, date and time of the observation, the existence of special events or promotions.
- Descriptive statistics measures holding information about the series itself, such as the central tendency, spread, shape and autocorrelation.
- Automatically generated features, usually produced by other ML algorithms

✓ Extracting such information from the series is a very challenging task from ML models.

✓ In the case of external variables, their future values are often known by the practitioners and can be used.

✓ Using additional features is a standard practice in certain areas such as demand forecasting where promotion greatly affect the time series.

✓ In the case of batch business forecasting tasks, similar to those of M3 and M4 Competitions:

- Statistical measures are mostly used for method weighting and selection tasks.
- External variables usually are not available or cannot be used effectively.

External variables and features (2/2)

Empirical study on the monthly series of the M4 competition:

- Dataset consisted of the 48,000 monthly series from the M4 competition
- A simple MLP was used for forecasting with and without additional inputs.
- The additional inputs were: strength of trend, spectral entropy, first order autocorrelation and optimal Box-Cox parameter.

Method	OWA	Difference from Theta (%)
Theta	0.907	-
Simple MLP	0.899	- 0.88 %
MLP with features	0.887	- 2.21 %

NN architectures similar to auto-encoders can be studied as a means of generating more features, without any underlying assumptions

References

- G. Zhang, P.E. Patuwo, M.Y. Hu (1998). Forecasting with artificial neural networks: The state of the art, International Journal of Forecasting, vol 14(1), pp. 35-62
- S. Makridakis, E. Spiliotis, V. Assimakopoulos (2018). Statistical and Machine Learning forecasting methods: Concerns and ways forward. PLOS ONE 13(3)
- M. Adaya, F. Collopy (1998). How Effective are Neural Networks at Forecasting and Prediction? A Review and Evaluation. Journal of Forecasting, Vol 17(5), pp. 481-495
- S. Makridakis, M. Hibon (2000). The M3-Competition: results, conclusions and implications. International Journal of Forecasting, vol 16(4), pp. 451-476
- S.F. Crone, M. Hibon, K. Nikolopoulos (2011). Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction, International Journal of Forecasting, vol 27(3), pp. 635-660
- S. Makridakis, E. Spiliotis, V. Assimakopoulos (2018). The M4 Competition: Results, findings, conclusion and way forward, International Journal of Forecasting, vol 34(4),pp. 802-808
- D. Salinas, V. Flunkert, J. Gasthaus, T. Januschowski (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks, International Journal of Forecasting, vol 36(3), pp. 1181-1191
- B. Oreshkin, D. Carpov, N. Chapados, Y. Bengio (2019). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting, arXiv preprint, arXiv:1905.10437
- P. Montero-Manso, G. Athanasopoulos, R. J. Hyndman, T. S. Talagala (2020). FFORMA: Feature-based forecast model averaging, International Journal of Forecasting, vol 36(1), pp. 86-92
- S. Smyl (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting, International Journal of Forecasting, vol 36(1), pp. 75-85

UNIC

MOFC

UNIVERSITY *of* NICOSIA