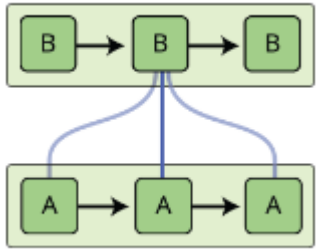
MET CS688 C1

WEB ANALYTICS AND MINING

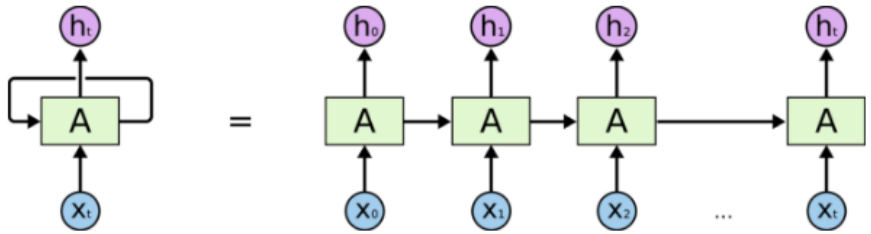
ZLATKO VASILKOSKI

RECURRENT NEURAL NETWORKS

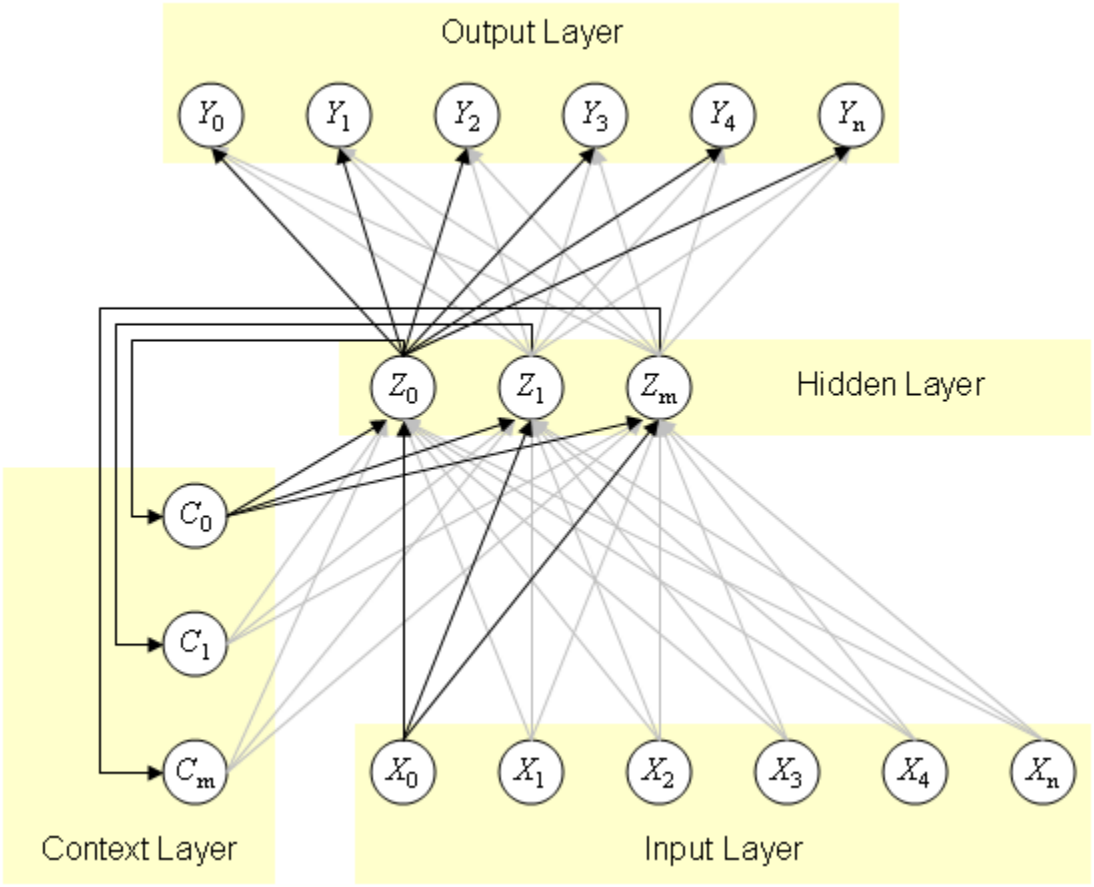


Recurrent Neural Networks

- Attention and Augmented Recurrent Neural Networks work with sequences of data like text, audio and video.
- Contain Context layer
- A special variant – “long short-term memory” LSTM networks
- Very powerful, remarkable results in many tasks including
  - translation,
  - voice recognition, and
  - image captioning.



An unrolled recurrent neural network.



Elman's RNN

COGNITIVE SCIENCE **14,** 179–211 (1990)

Finding Structure in Time

JEFFREY L. ELMAN
University of California, San Diego

- Introduced by Elman in 1990 with intention to analyze language.

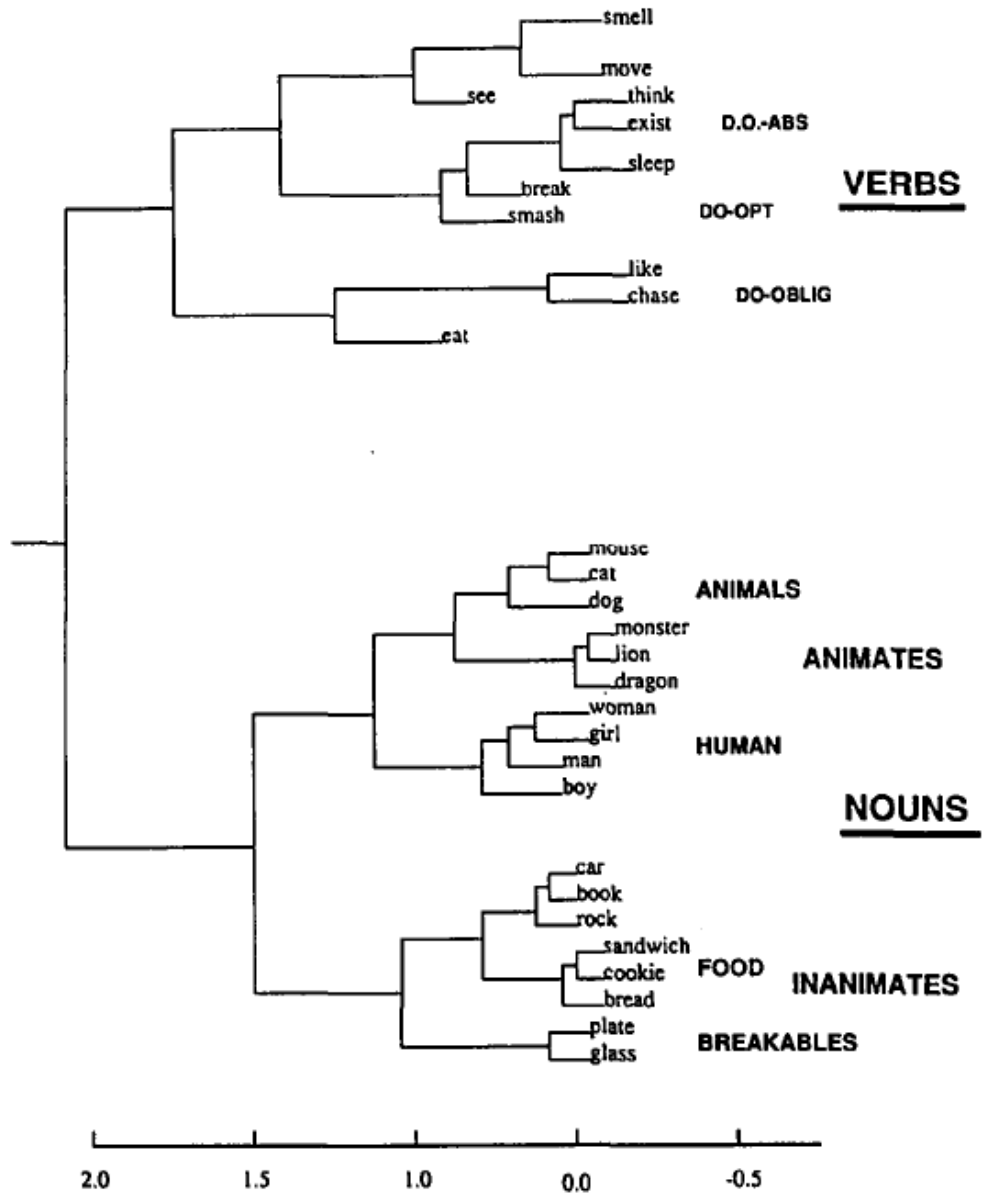


Figure 7. Hierarchical cluster diagram of hidden unit activation vectors in simple sentence prediction task. Labels indicate the inputs which produced the hidden unit vectors; inputs were presented in context, and the hidden unit vectors averaged across multiple contexts.

Recurrent Neural Nets in R – RSNNS package

- Enables a use of wide variety of network types and possible applications.
- A convenient feature in R is the inclusion of datasets along with software packages.
- This is important for NNS standardized tests as well as for examples of the usage of the package.
- Use the list *snnsData()* for all of the available datasets in RSNNS.

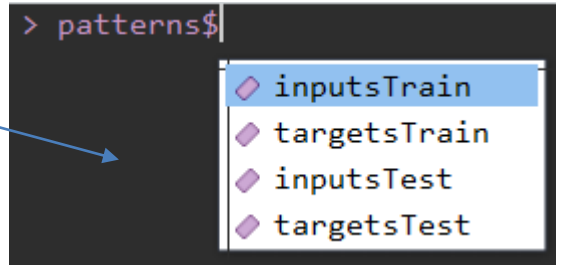
- These functions can be used to pick the right columns according to their names.
  - *inputColumns()* Extracts all columns from a matrix whose column names begin with "in"
  - *outputColumns()* Extracts all columns from a matrix whose column names begin with "out"
- Functions *splitForTrainingAndTest()* can be used to split the data in a training and a test set.

RSNNS – Access & Splitting the Data

- Read the raw data into R from a CSV file ("EIS_Data.csv“ on Blackboard)
- Access and split the data into Train and Test data.

```
# Example: RSNNS
# Separate Data into train and test
inputs <- temp.CSV$Actual.Consumption..kWh. # used for Training
targets <- temp.CSV$Expected.Consumption..kWh. # (Regression data) used as Targets
# Split the input data into train and test specified by the argument ratio=0.15
patterns <- splitForTrainingAndTest(inputs, targets, ratio = 0.15)
```

	Dat Table	
Inputs		Outouts
1		1
2	Train	2
-	Data	-
-		-
850		850
851		851
-	Test	-
-	Data	-
1000		1000

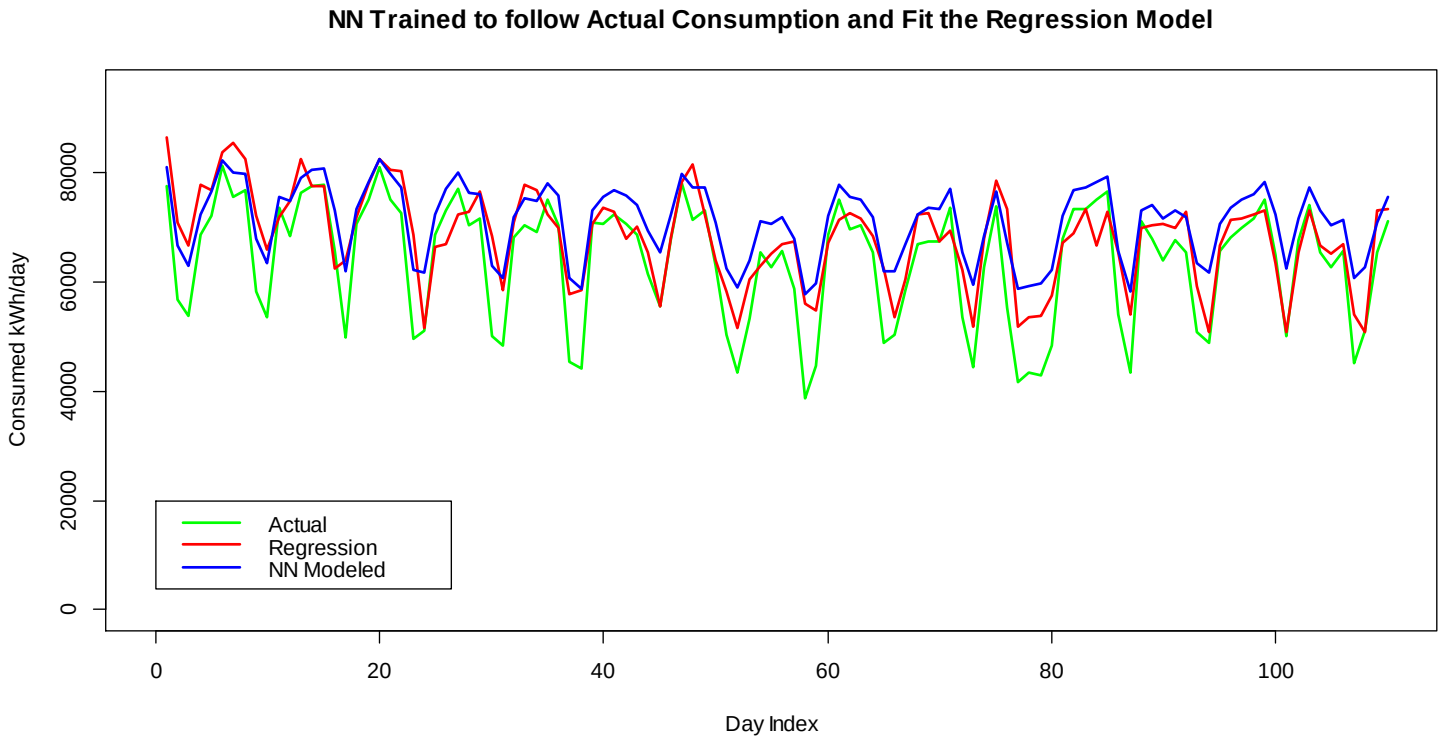


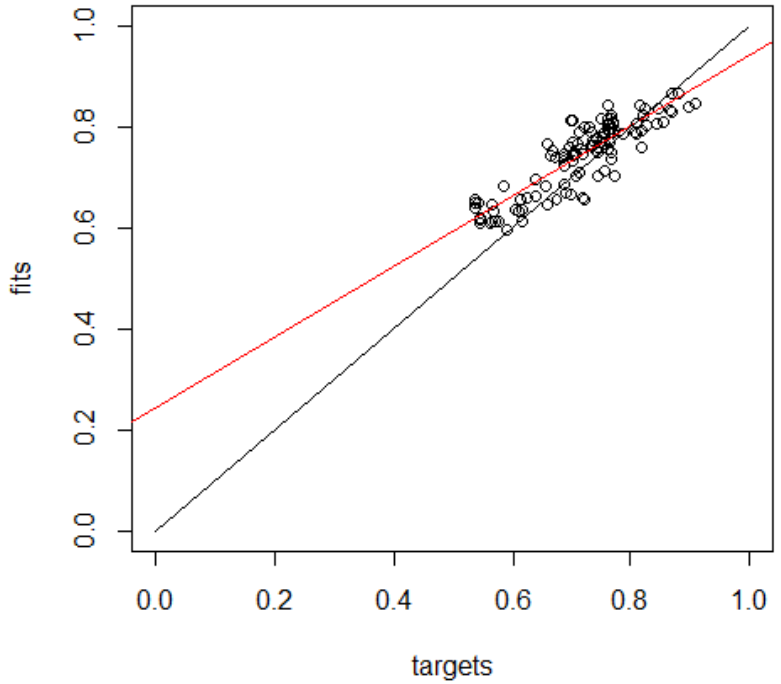
- Take input to be Actual Consumption (plant’s consumption)
- Take targets to be the Expected.Consumption..kWh (from the regression model)
- Train the Nu-Net on the Actual Consumption as an input to
- forecast reply with

Actual Data – first 100 days

The NN will be trained on the Actual Consumption (green) - the input in NN
The NN will be trained to target the Regression model data (red)

NN Trained to Match the Regression Model





After NN being trained

The NN input is Actual Consumption (green)

The NN output (blue) is very close to Xcel's Regression model data (red)

The NN match is pretty good considering the temperature, seasonal, and weekend variability of the data for which the NN is not aware of, but it learned to take them into account.

The Code

```
# Plot the data
ymax <- inputs.scale; xmax <- 100
plot(c(0,xmax),c(0,ymax),type = "n",xlab = "Day Index", ylab = "Consumed kWh/day",main = "Regression Model Fit to the Actual")
lines(inputs[1:xmax],col="green",lwd=2.5) # Plot Data 1
lines(targets[1:xmax],col="red",lwd=2.5) # Plot Data 2
legend(0.0,2e4, # places a legend at the appropriate place
       c("Actual","Regression"), # puts text in the legend
       lty=c(1,1), # gives the legend appropriate symbols (lines)
       lwd=c(2.5,2.5),col=c("green","red")) # gives the legend lines the correct color and width

# Train NN to folow Actual
# The use of an Elman network (Elman 1990) for time series regression.
model <- elman(patterns$inputsTrain, patterns$targetsTrain,
               size = c(8, 8), learnFuncParams = c(0.1), maxit = 500,
               inputsTest = patterns$inputsTest, targetsTest = patterns$targetsTest,
               linOut = FALSE)

NN.fitted.Train <- model$fitted.values*inputs.scale
NN.fitted.Test <- model$fittedTestValues*targets.scale

# Plot Train Values
ymax <- 1; xmax <- 100 # length(NN.fitted.Train)
plot(c(0,xmax),c(0,ymax),type = "n",xlab = "Day Index", ylab = "Consumed kWh/day",main = "NN Trained to follow Actual Consumption and Fit the Regression Model")
lines(patterns$inputsTrain[1:xmax],col="green",lwd=2.5) # Plot Data 1 - Train
lines(patterns$targetsTrain[1:xmax],col="red",lwd=2.5) # Plot Data 2 - Targets
lines(model$fitted.values[1:xmax], col = "blue",lwd=2.5)  # Plot Data 3 predicted
legend(0.0,0.3, # places a legend at the appropriate place
       c("Actual","Regression","NN Modeled"), # puts text in the legend
       lty=c(1,1), # gives the legend appropriate symbols (lines)
       lwd=c(2.5,2.5),col=c("green","red","blue")) # gives the legend lines the correct color and width
```