CS 133 - Introduction to Computational and Data Science

Instructor: Renzhi Cao

Computer Science Department

Pacific Lutheran University

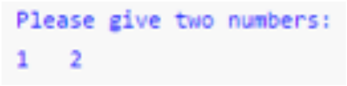
Spring 2017

Announcement

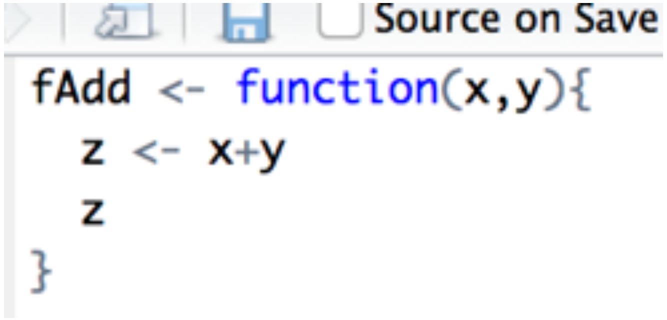
- *Read book.*
- *Final project*
- *Today we are going to learn machine learning.*

Machine learning - Neural Network

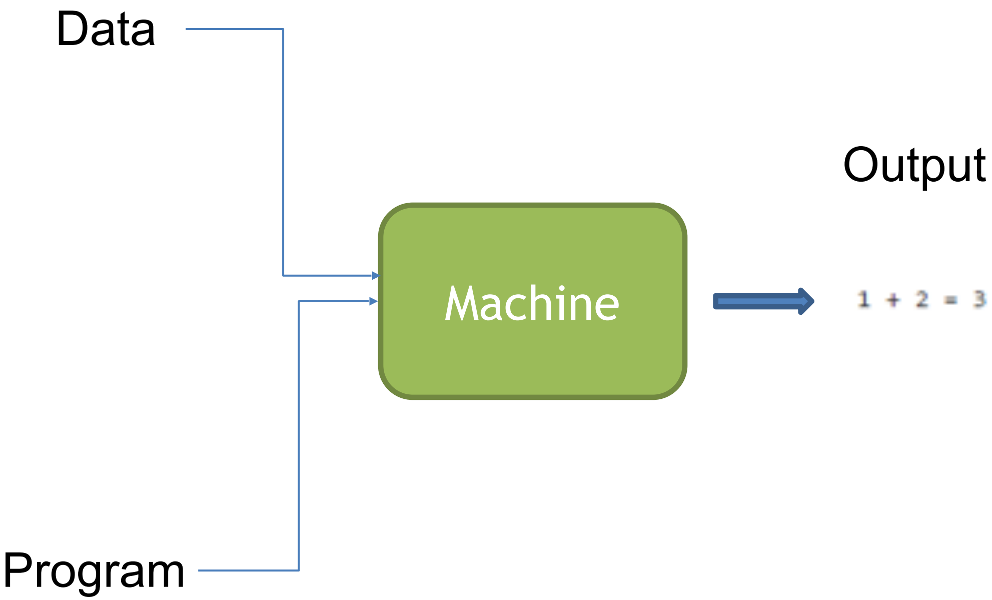
Traditional Programming



```
Please give two numbers:
1   2
```



```
fAdd <- function(x,y){
  z <- x+y
  z
}
```



Data

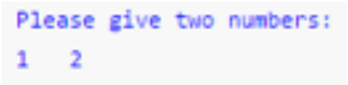
Program

Machine

Output

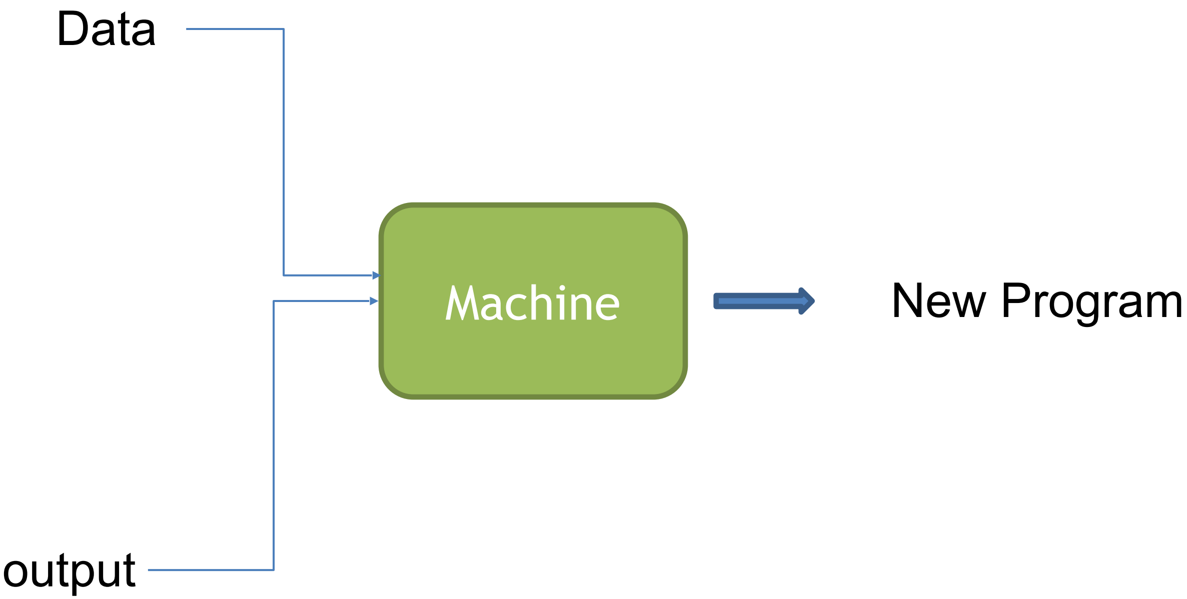
```
1 + 2 = 3
```

What is Machine learning?



```
Please give two numbers:
1   2
```

3



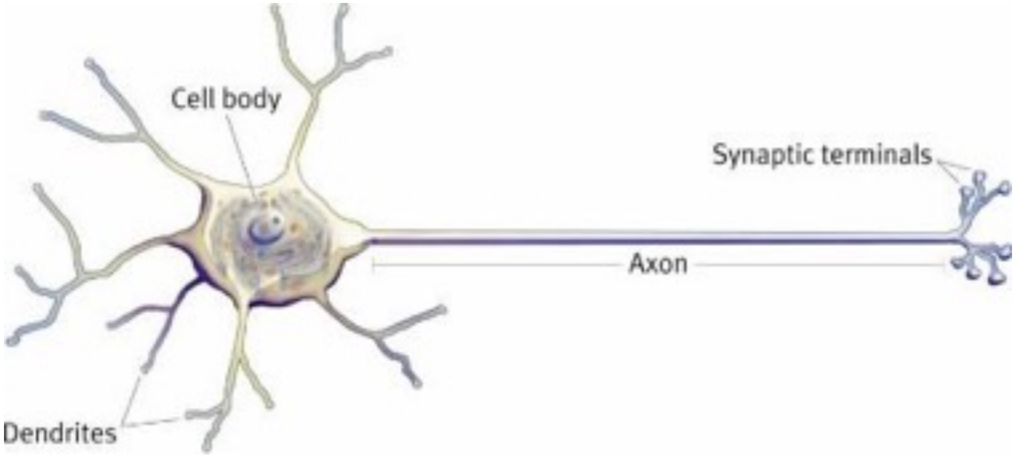
Data

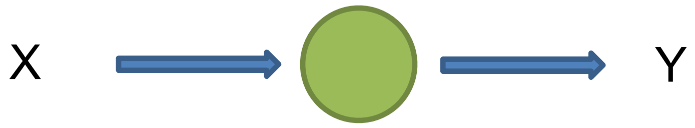
output

Machine

New Program

Neural Network





$$\underset{\text{output}}{Y} = \underset{\text{Weight}}{w} * \underset{\text{Input}}{x}$$

Input : 2
Output: 8

$$0 = w * x - Y$$

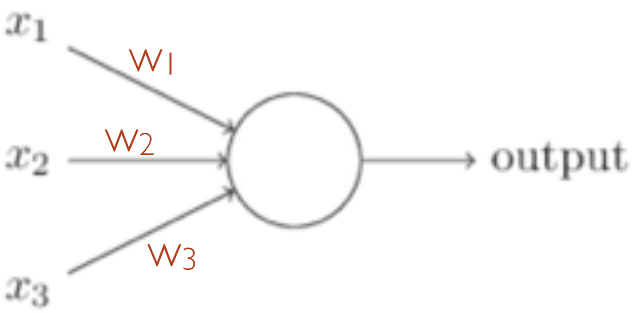
$$8 = w * 2$$

$$w = 8 \div 2$$

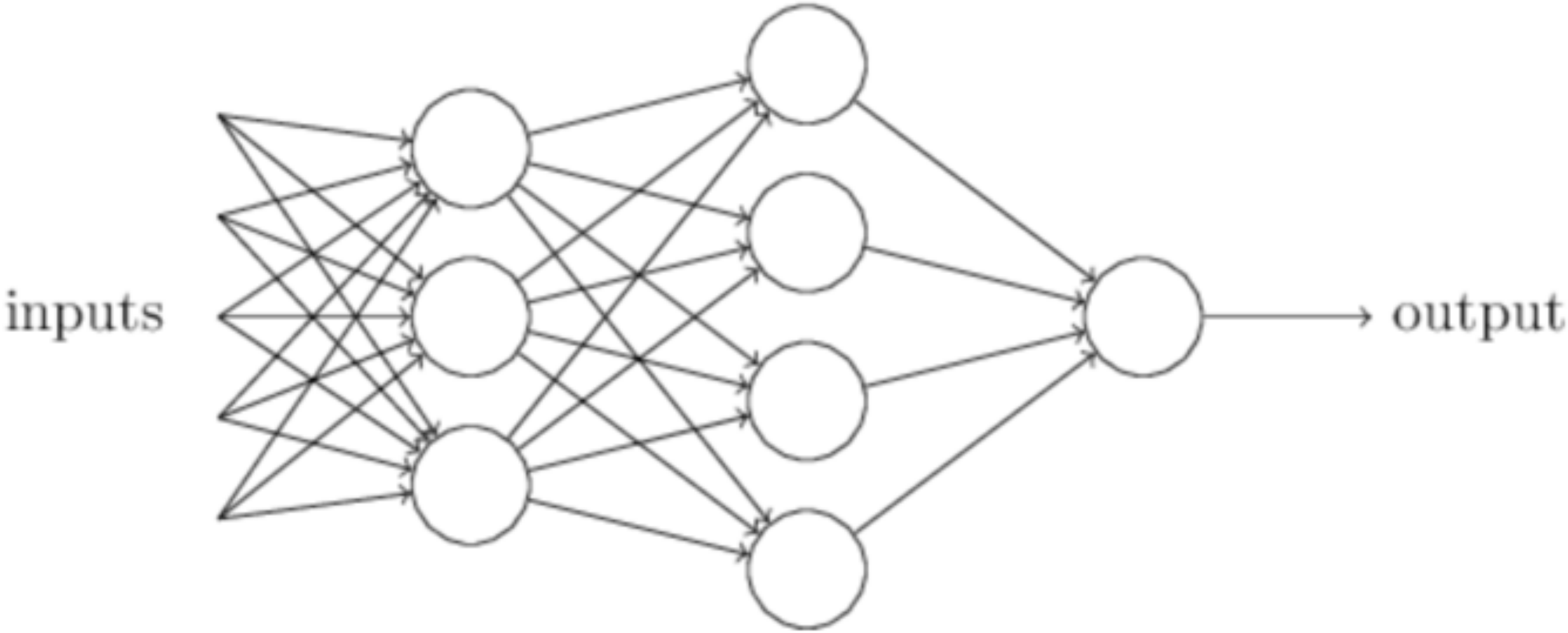
$$Error = |w * x - Y|$$

Feature units

decision units



Learned weight


inputs
output

Data preparation

ISLR's built in College Data Set which has several features of a college and a categorical column indicating whether or not the School is Public or Private.

```
#install.packages('ISLR')
library(ISLR)

print(head(College,2))
```

source: http://www.kdnuggets.com/2016/08/begineers-guide-neural-networks-r.html

Data processing

It is important to normalize data before training a neural network on it!

We use build-in scale() function to do that.

```
# Create Vector of Column Max and Min Values. apply(data, 1 for row, 2 for column, fun)
maxs <- apply(College[,2:18], 2, max)
mins <- apply(College[,2:18], 2, min)

# Use scale() and convert the resulting matrix to a data frame
scaled.data <- as.data.frame(scale(College[,2:18],center = mins, scale = maxs - mins))

# Check out results
print(head(scaled.data,2))
```

Train and Test Split

Training and testing dataset.

```
# Convert Private column from Yes/No to 1/0
Private = as.numeric(College$Private)-1
data = cbind(Private,scaled.data)

library(caTools)
set.seed(101)

# Create Split (any column is fine)
split = sample.split(data$Private, SplitRatio = 0.70)

# Split based off of split Boolean Vector
train = subset(data, split == TRUE)
test = subset(data, split == FALSE)
```

Neural Network Function

Before we actually call the neuralnetwork() function we need to create a formula to insert into the machine learning model

```
feats <- names(scaled.data)

# Concatenate strings
f <- paste(feats,collapse=' + ')
f <- paste('Private ~',f)

# Convert to formula
f <- as.formula(f)

f
```

Neural Network training

```
#install.packages('neuralnet')
library(neuralnet)
nn <- neuralnet(f,train,hidden=c(10,10,10),linear.output=FALSE)
```

```
# save your model and load it back for future usage
saveRDS(nn,"./nnModel.rds")

…

nn <- readRDS(“./nnModel.rds")
```

Predictions and Evaluations

We use the compute() function with the test data (jsut the features) to create predicted values.

```
# Compute Predictions off Test Set
predicted.nn.values <- compute(nn,test[2:18])

# Check out net.result
print(head(predicted.nn.values$net.result))
```

Predictions and Evaluations

Notice we still have results between 0 and 1 that are more like probabilities of belonging to each class.

```
predicted.nn.values$net.result <- sapply(predicted.nn.values$net.result,round,digits=0)
```

Now let's create a simple confusion matrix:

```
table(test$Private,predicted.nn.values$net.result)
```

Visualizing the Neural Net

We can visualize the Neural Network by using the plot(nn) command.

Work on your final project

- 15 mins presentation about your project
- I may give you testing data to evaluate performance of your NN model.
- Final report