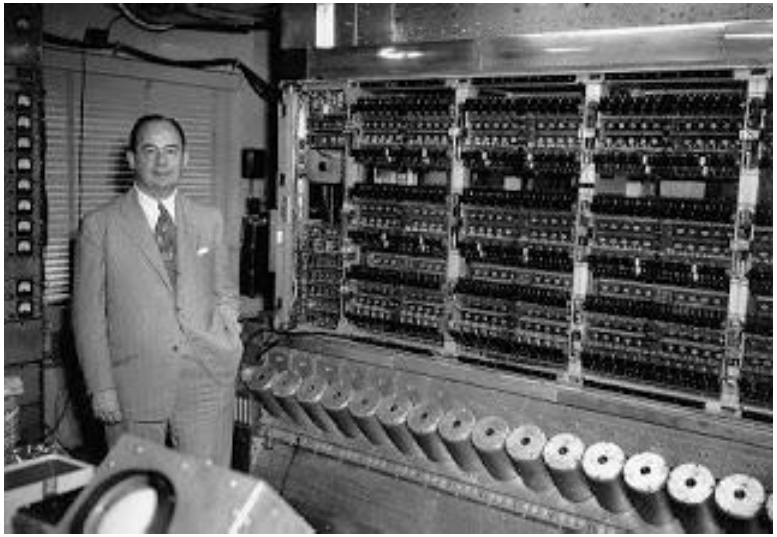
Machine Learning Introduction

Dr. Paul H. Comitz

pcomitz@live.com



Professor John Von Neumann and one of the first “Von Neumann” machines

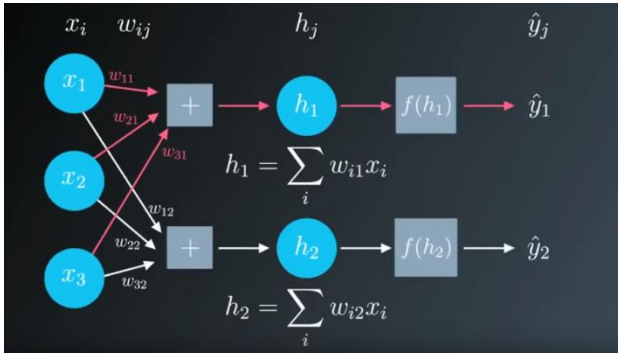
Agenda

- Machine Learning
- Regression
- Neural Networks
  - Gradient Descent
- TensorFlow
  - Linear Regression
  - Simple NN
  - Classifier

Objective

A short session introducing selected Machine learning topics. No prior knowledge is assumed.

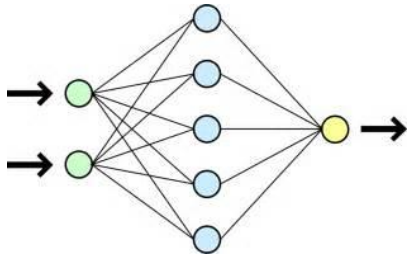
- Regression
- Neural Networks
- Classification



Slides, discussion, hands-on

Context

- Focus on Machine Learning
  - Regression, Neural Nets, Classifier
- Use TensorFlow 1.2
- Use Python 3.5 (other versions may work)
- Use Anaconda on Windows 10 (other environments will work)
- A little math and computer science experience will help
  - Not required
  - Beginner level – *for this material*



- Approximately 2ish hours with short break

What is machine learning ?

Write a program

Easy right ?

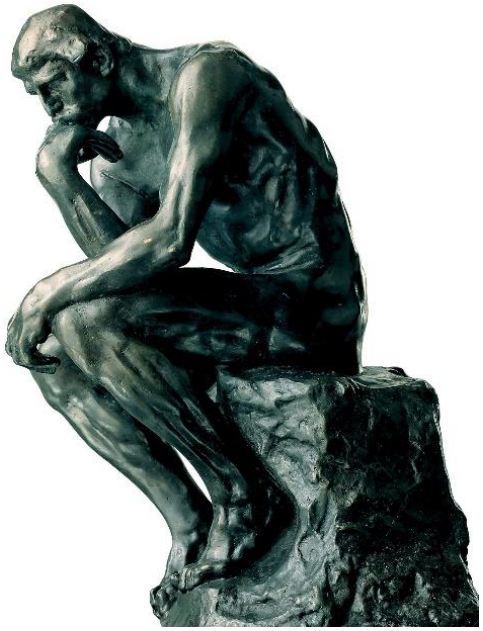
What if the objects are very different?

Not so fast ….

Google I/O 2017 Machine Learning APIs by Example
https://www.youtube.com/watch?v=ETeeSYMGZn0

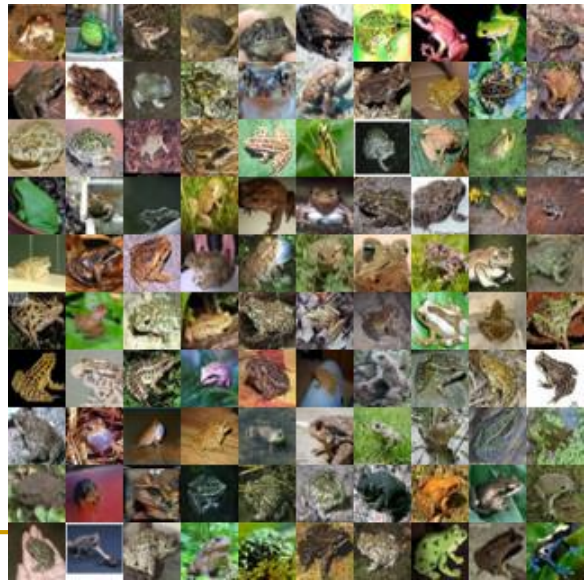
The Big Idea

- Don’t want to write programs based on rules
- Write code that will find the patterns
- Write code that will *learn* from data
  - All kinds of data
    - Video, text, images
- This is often referred to a *machine learning*



Machine Learning

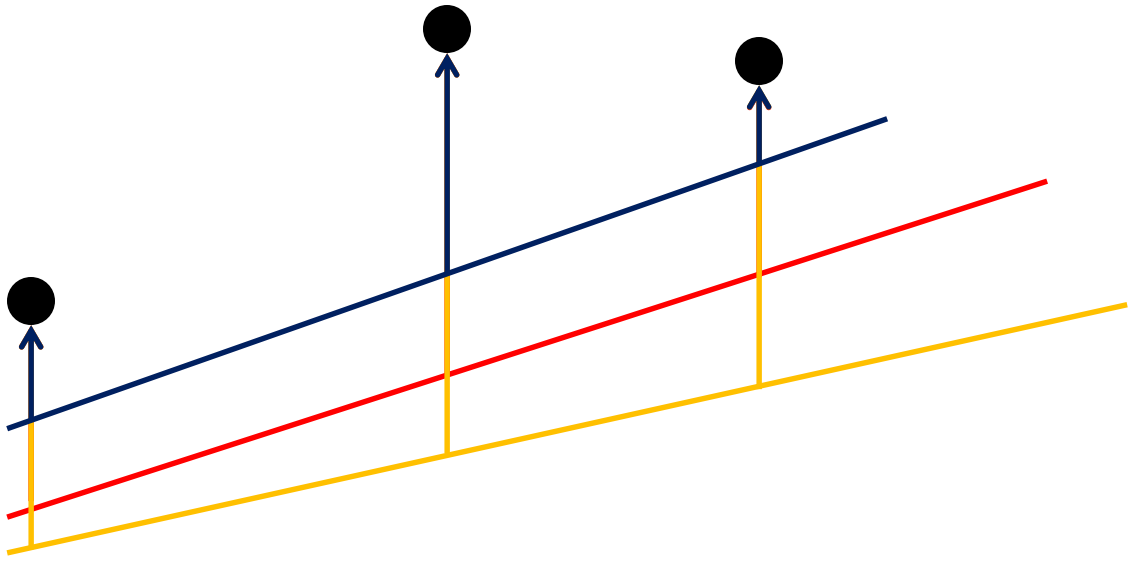
- Field of Artificial Intelligence
- Computers learn environment from data rather that rules in computer programs
- With Supervised Learning
  - Train with known data
- Classifier in browser



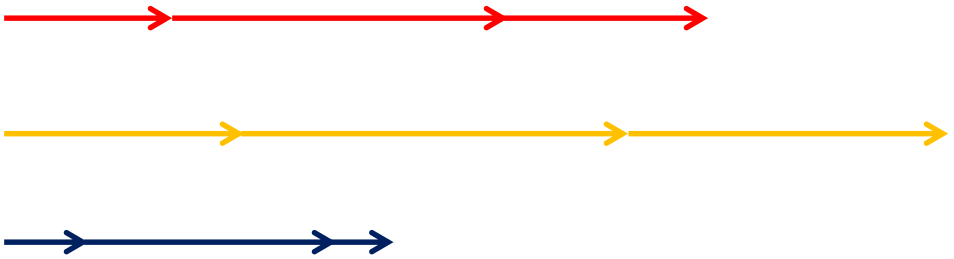
http://cs231n.stanford.edu/

Learning from Data

What is the best line that fits these points?



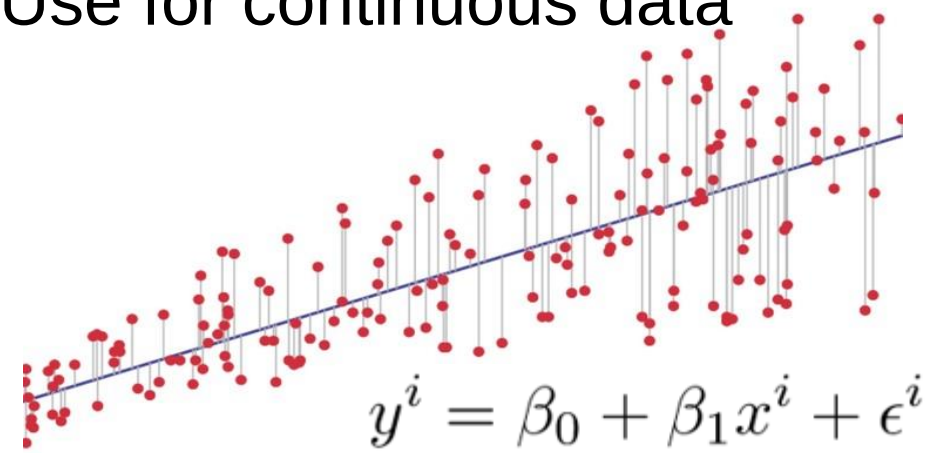
Error = sum of individual errors



Find the solution that fits the data

Linear Regression

- Move the line in the direction of decreasing error until error is minimized
  - Square the error to avoid negative distances
    - Least squares
  - Use for continuous data



$$y^i = \beta_0 + \beta_1 x^i + \epsilon^i$$

$$y^i = \beta_0 + \beta_1 x_1^i + \beta_2 x_2^i + \ldots + \beta_k x_k^i + \epsilon^i$$

x^i = independent variable, ith observation
β = regression coefficient
ε = error term

The best fit is the choice of coefficients that results in the smallest error

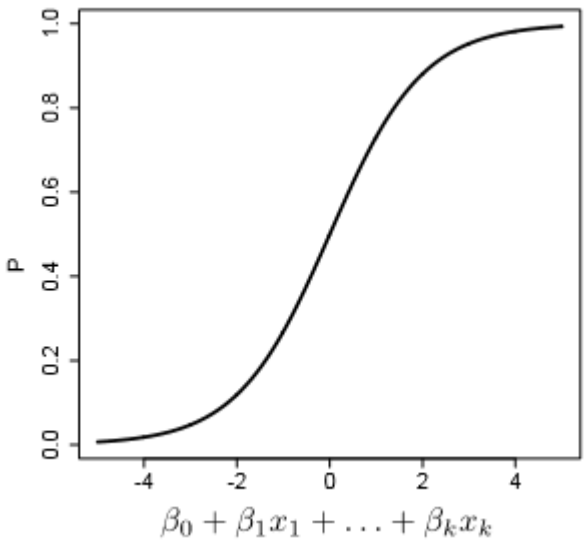
Logistic Regression

- Linear regression provides a way to predict continuous data
- Logistic regression provides a mechanism to *classify* data
  - Discrete or categorical data

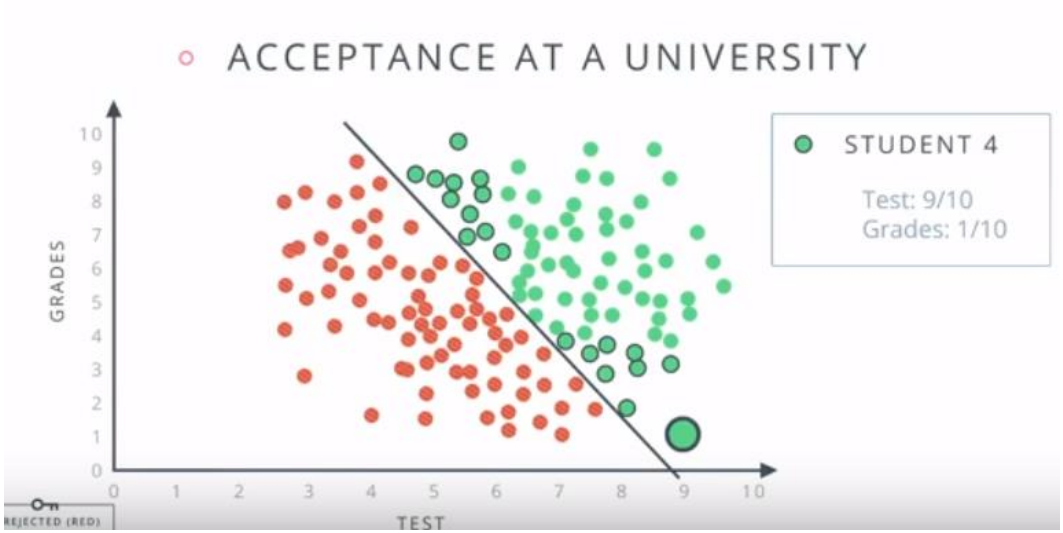
$$P(y = 1) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k)}}$$

The coefficient are selected to predict a probability for a given class

P(digit = 1)
P(image is a bird)
P(patient is a smoker)



Neural Networks (1)



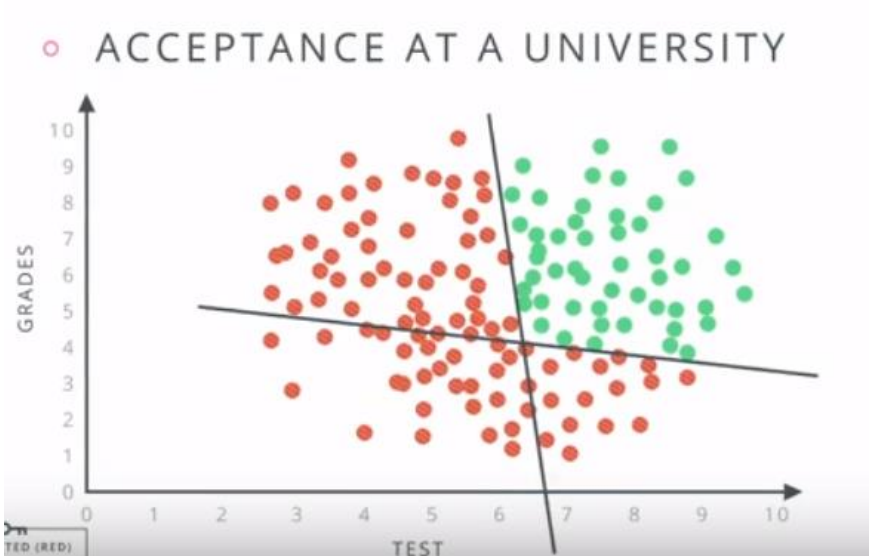
Green = accept students
Should student 4 be accepted?

Perhaps there is a better distibution

Is a single line sufficient?

Images from udacity.com

Neural Networks (2)



Use two lines

Find the two lines using gradient descent

This is called a ***Neural Network***

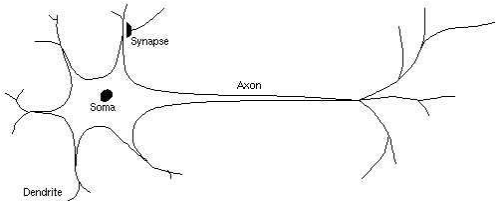
With a neural network we split the problem into separate questions

1) Is the point of interest above the horizontal line?
2) Is the point of interest to the right of the vertical line?
3) Is the answer to both previous questions yes?

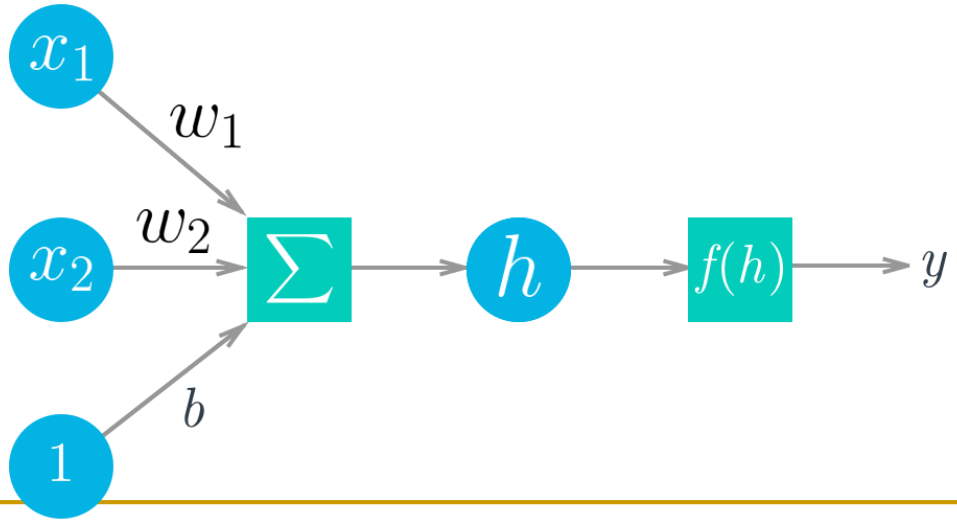
Images from udacity.com

Neural Networks (3)

Artificial Neurons

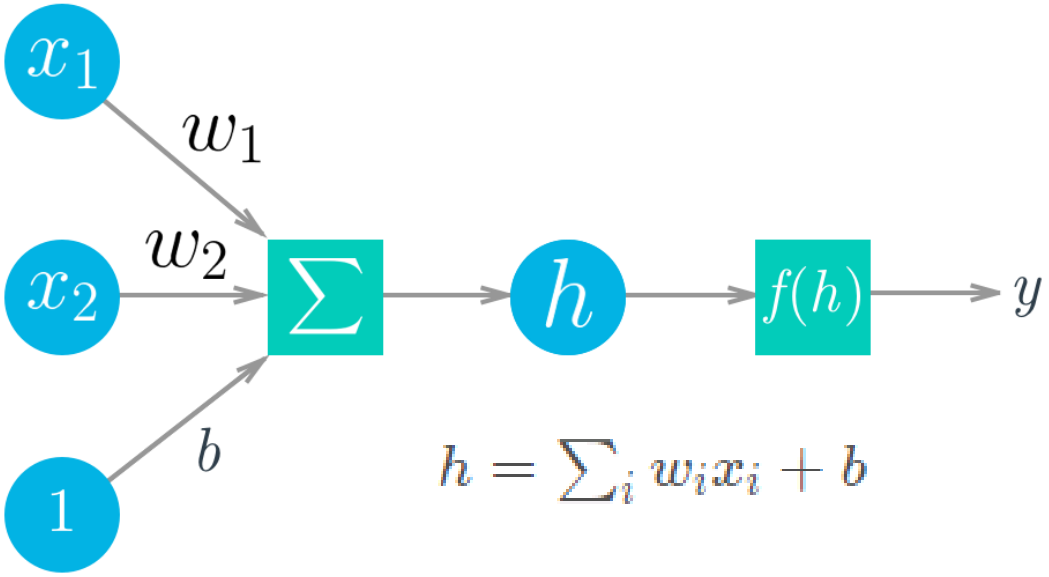


- Basic unit of a neural network
- Loosely model after a biological neuron
- Neurons can be combined to form complex networks

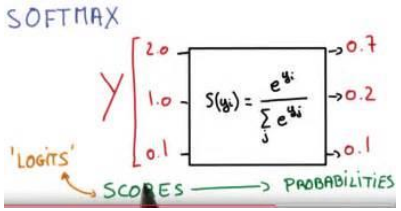


See https://sebastianraschka.com/faq/docs/logisticregr-neuralnet.html

Weights and Activation

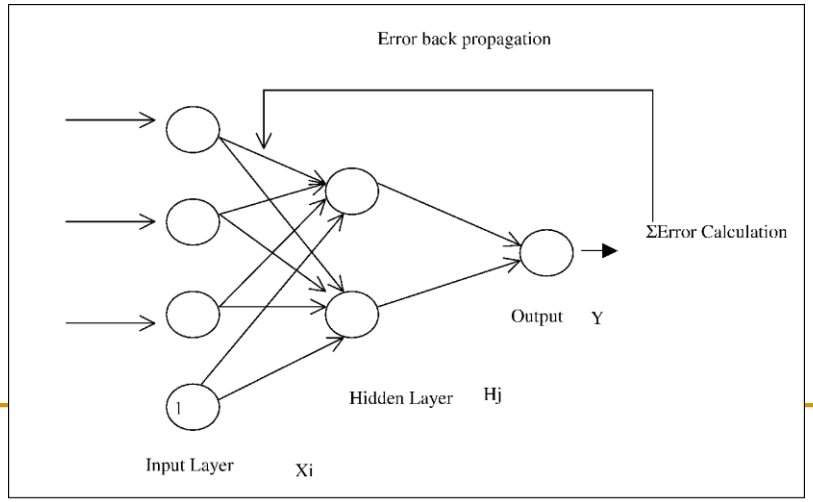


- Linear combination of weights, inputs, and bias, denoted h, are passed to an activation function
- The activation function can be any function
  - Common choices are:
    - sigmoid $1/(1+e^{-x})$
    - softmax – convert score to probabilities



Gradient Descent

- The weights in a neural network are trained using a known set of data usually referred to as the training set
- Gradient Descent is used to minimize the error in a network
- Backpropagation – the error is propagated back to adjust the weights



http://i.stack.imgur.com/pOR6t.png

Notes: The weight connecting node *i* in the input layer to node *j* in the hidden layer is denoted by *Wji*, and the weight connecting node *i* to the output node is represented by *Vi*

Gradient Descent Recipe

- Compute network output
  - $h = \Sigma (w_i x_i)$
  - y^ = f(h) # f is the activation function
- Compute error
  - y – y^ # expected value – output
- Compute error term
  - (y – y^)*f'(h) # derivative activation function
- Compute $\Delta\omega_i$ (change in weights)
  - $\omega_i = \eta*(y-y^)*x$ # η is the learning rate

TensorFlow

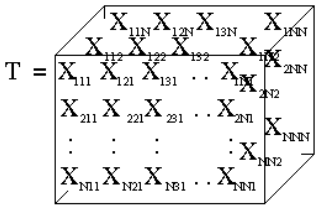
- Open source machine learning framework
- Released approx. 1 year ago
- Current version is 1.2
  - APIs in Python, Java, C, Go

A software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them.

https://www.tensorflow.org/

So …what's a tensor?

(https://www.tensorflow.org/get_started/get_started)



- Central unit of data in TensorFlow
- Multidimensional data array
  - Rank – number of dimensions

```
3 # a rank 0 tensor; this is a scalar with shape []
[1. ,2., 3.] # a rank 1 tensor; this is a vector with shape [3]
[[1., 2., 3.], [4., 5., 6.]] # a rank 2 tensor; a matrix with shape [2, 3]
[[[1., 2., 3.]], [[7., 8., 9.]]] # a rank 3 tensor with shape [2, 1, 3]
```

$$\begin{bmatrix} w_{11} & w_{21} & w_{31} & w_{41} & w_{51} & w_{61} & w_{71} & w_{81} & w_{91} \\ w_{12} & w_{22} & w_{32} & w_{42} & w_{52} & w_{62} & w_{72} & w_{82} & w_{92} \\ w_{13} & w_{23} & w_{33} & w_{43} & w_{53} & w_{63} & w_{73} & w_{83} & w_{93} \\ w_{14} & w_{24} & w_{34} & w_{44} & w_{54} & w_{64} & w_{74} & w_{84} & w_{94} \\ w_{15} & w_{25} & w_{35} & w_{45} & w_{55} & w_{65} & w_{75} & w_{85} & w_{95} \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \\ i_5 \\ i_6 \\ i_7 \\ i_8 \\ i_9 \end{bmatrix} = \begin{bmatrix} total\ input\ to\ H_1 \\ total\ input\ to\ H_2 \\ total\ input\ to\ H_3 \\ total\ input\ to\ H_4 \\ total\ input\ to\ H_5 \end{bmatrix}$$

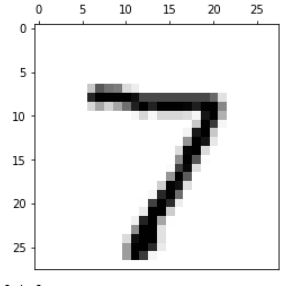
BREAK

Getting Started with TensorFlow

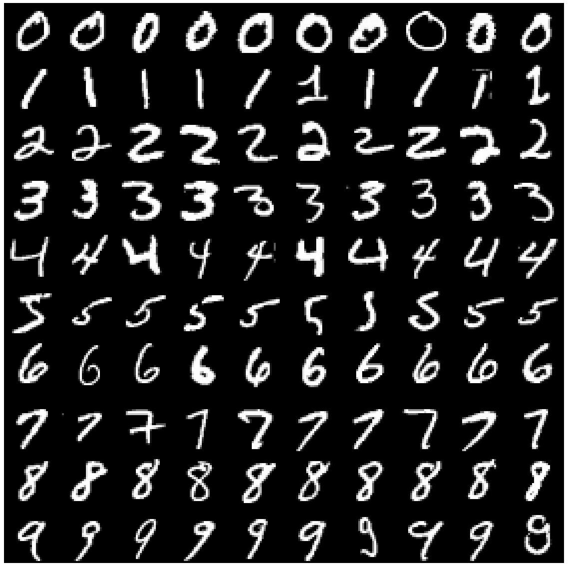
- Go to
  - https://github.com/pcomitz/reforge_ml
- Open the interactive notebook
  - `TensorFlowGetting Started.ipynb`

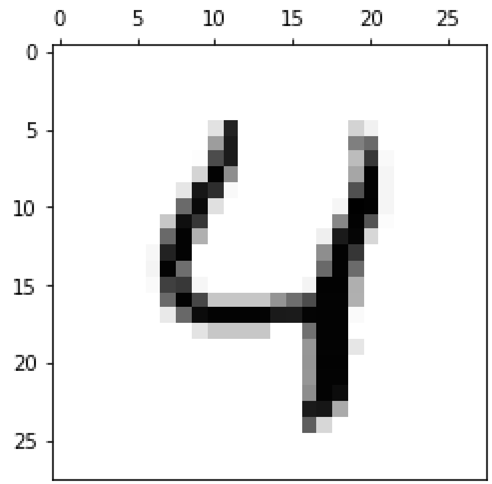
MNIST Classifier

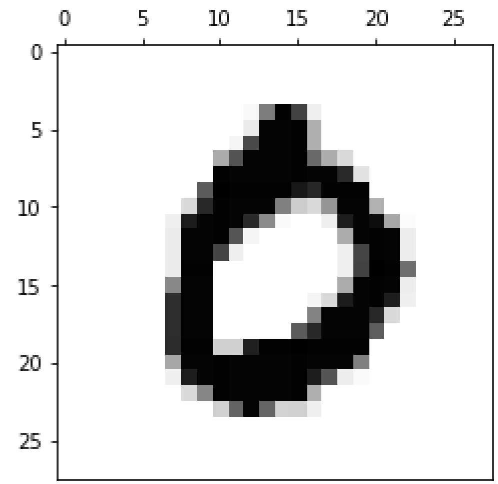
- MNIST is Database of handwritten digits 0 - 9
- Subset of larger data set available from National Institute of Standards and Technology
- Training set : 55000 images
- Test Set: 10000 images
- Each image 28 x 28 = 784 pixels
- Each pixel has a greyscale value of 0 – 255

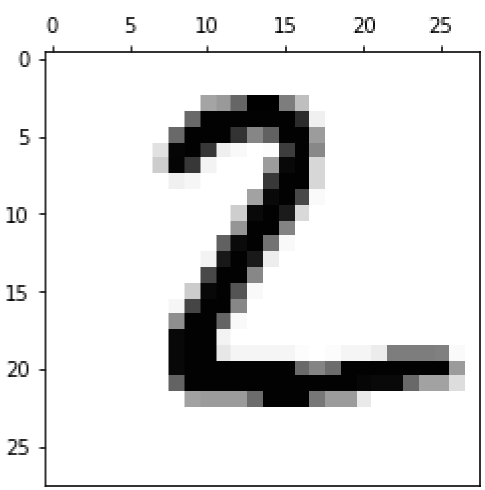


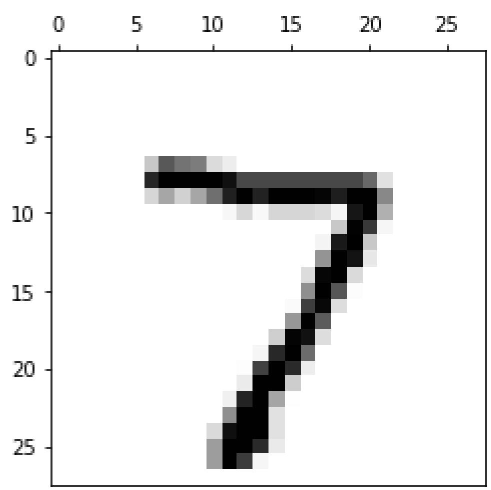
MNIST image data

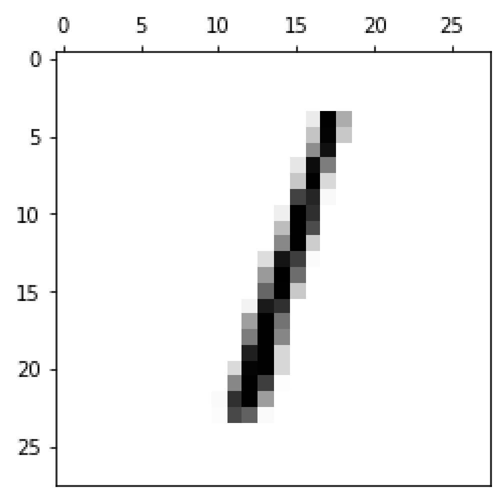












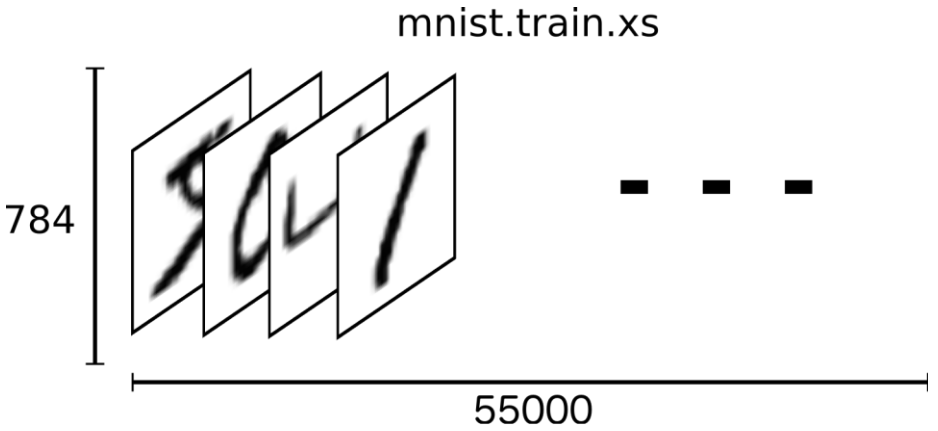
MNIST detail

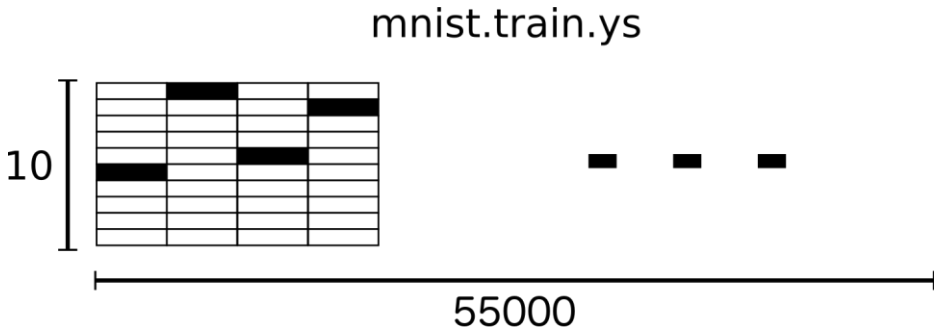
784 features

```
show the pixels
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   50  224 0   0   0   0   0   0   0   70  29  0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   121 231 0   0   0   0   0   0   0   148 168 0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   4   195 231 0   0   0   0   0   0   0   96  210 11  0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   69  252 134 0   0   0   0   0   0   0   114 252 21  0   0   0   0   0   0
0   0   0   0   0   0   0   0   45  236 217 12  0   0   0   0   0   0   0   192 252 21  0   0   0   0   0   0
0   0   0   0   0   0   0   0   168 247 53  0   0   0   0   0   0   0   18  255 253 21  0   0   0   0   0   0
0   0   0   0   0   0   0   84  242 211 0   0   0   0   0   0   0   0   141 253 189 5   0   0   0   0   0   0
0   0   0   0   0   0   0   169 252 106 0   0   0   0   0   0   0   32  232 250 66  0   0   0   0   0   0   0
0   0   0   0   0   0   15  225 252 0   0   0   0   0   0   0   0   134 252 211 0   0   0   0   0   0   0   0
0   0   0   0   0   0   22  252 164 0   0   0   0   0   0   0   0   169 252 167 0   0   0   0   0   0   0   0
0   0   0   0   0   0   9   204 209 18  0   0   0   0   0   0   22  253 253 107 0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   169 252 199 85  85  85  85  129 164 195 252 252 106 0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   41  170 245 252 252 252 252 232 231 251 252 252 9   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   49  84  84  84  84  0   0   161 252 252 0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   127 252 252 45  0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   128 253 253 0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   127 252 252 0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   135 252 244 0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   232 236 111 0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   179 66  0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

MNIST Classifier

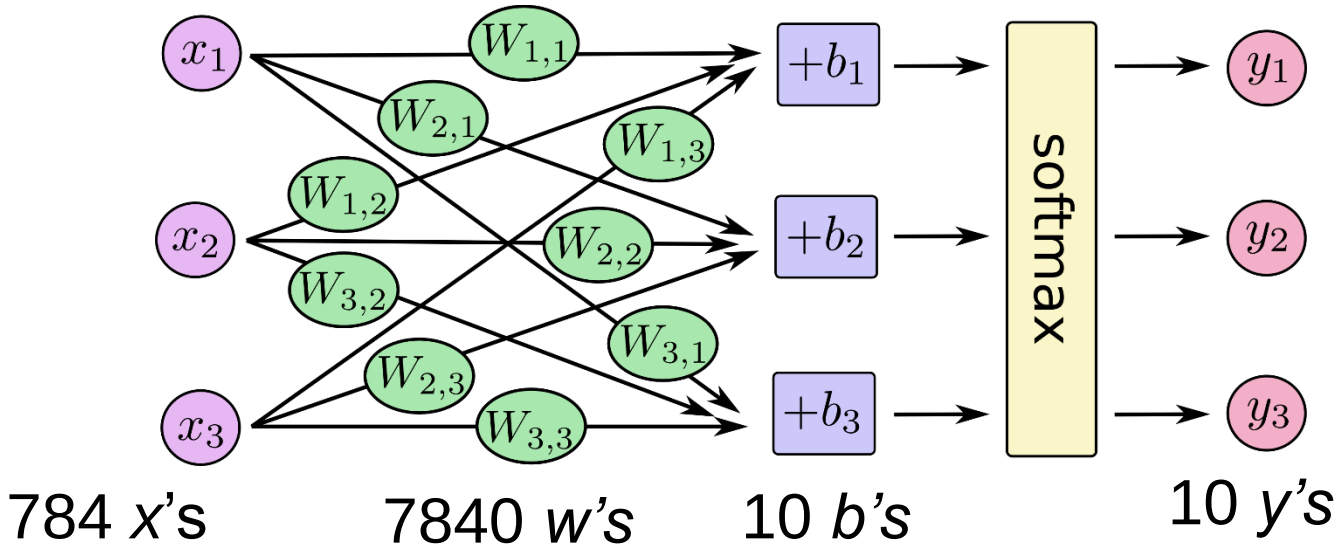
- Load the images
- Training set
  - 55000 inputs
  - 784 features per input
    - 28 x 28 pixels
  - 10 labels
    - (0 1 2 3 4 5 6 7 8 9)
- The classifier output is a label for each input





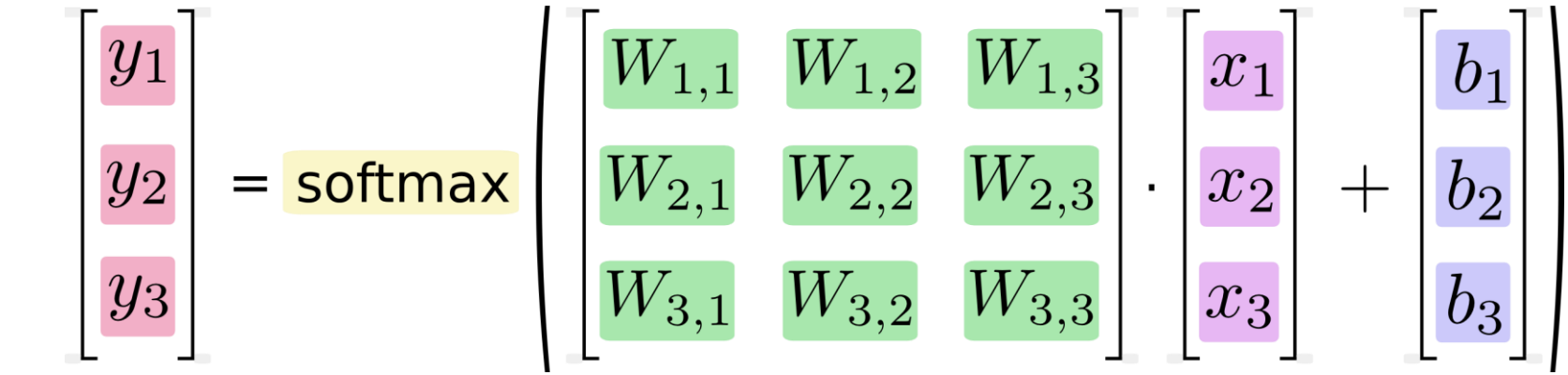
MNIST Classifier (2)

- There is a weight for each input pixel – for each label or class.
  - 784 input features * 10 classes = 7840 weights
- 10 bias values – one for each class



MNIST Classifier (3)

- The model is:
  - `y = tf.nn.softmax(tf.matmul(x,W)+b)`



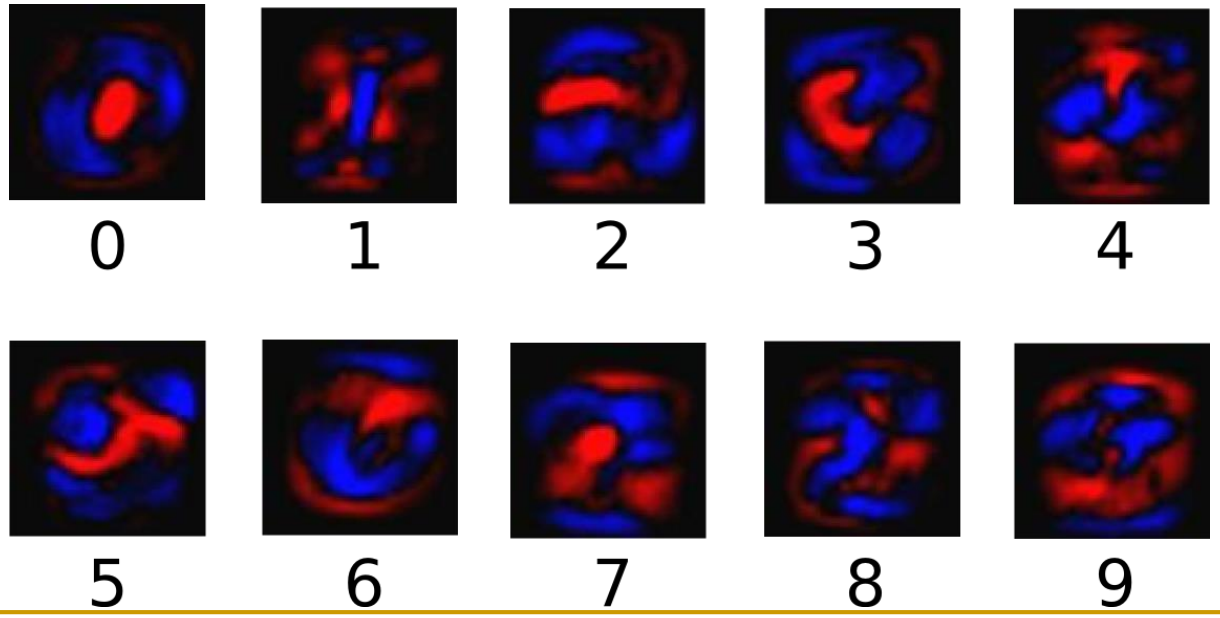
SoftMax in TensorFlow

$$\text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

SoftMax turns scores into a probability vector that sums to 1

Training the Classifier

- Use and stochastic gradient descent (sgd) and backpropagation to train the weights matrix

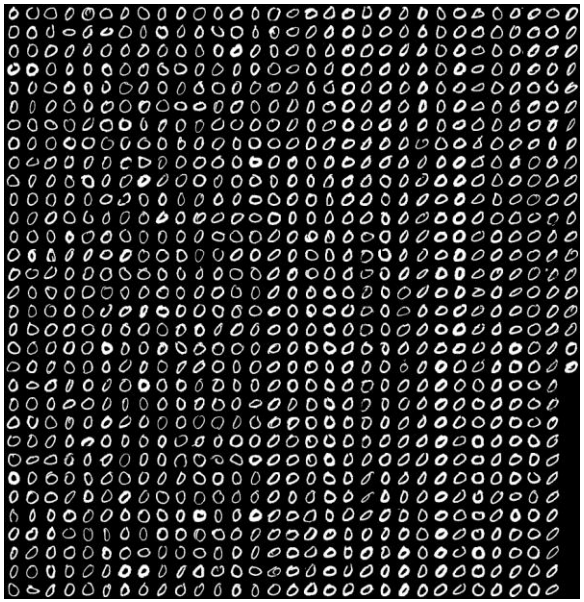


Classifier Notebook

- On Github
- https://github.com/pcomitz/reforge_ml/blob/master/MNIST_for_beginners.ipynb

Summary

- We have barely scratched the surface
  - Significant study required – but many excellent resources available
- Machine learning is a paradigm shift in computer programming
  - Learn from data rather than encoded rules
- Frameworks like TensorFlow are designed to run at scale
  - Thousands (millions) of complex inputs



Additional Resources

- Neural Networks and Deep Learning (by Michael Nielson)
- Deep Learning Book (Goodfellow, Bengio, Courville)
- TensorFlow Playground
- Udacity Machine Learning Course (free)
- Stanford CS231n CIFAR classifier in browser
- Wikipedia Machine Learning Portal
- Artificial Intelligence: a Modern Approach
- Free AI Course at edx
- AI Courses at Coursera


That's all Folks!