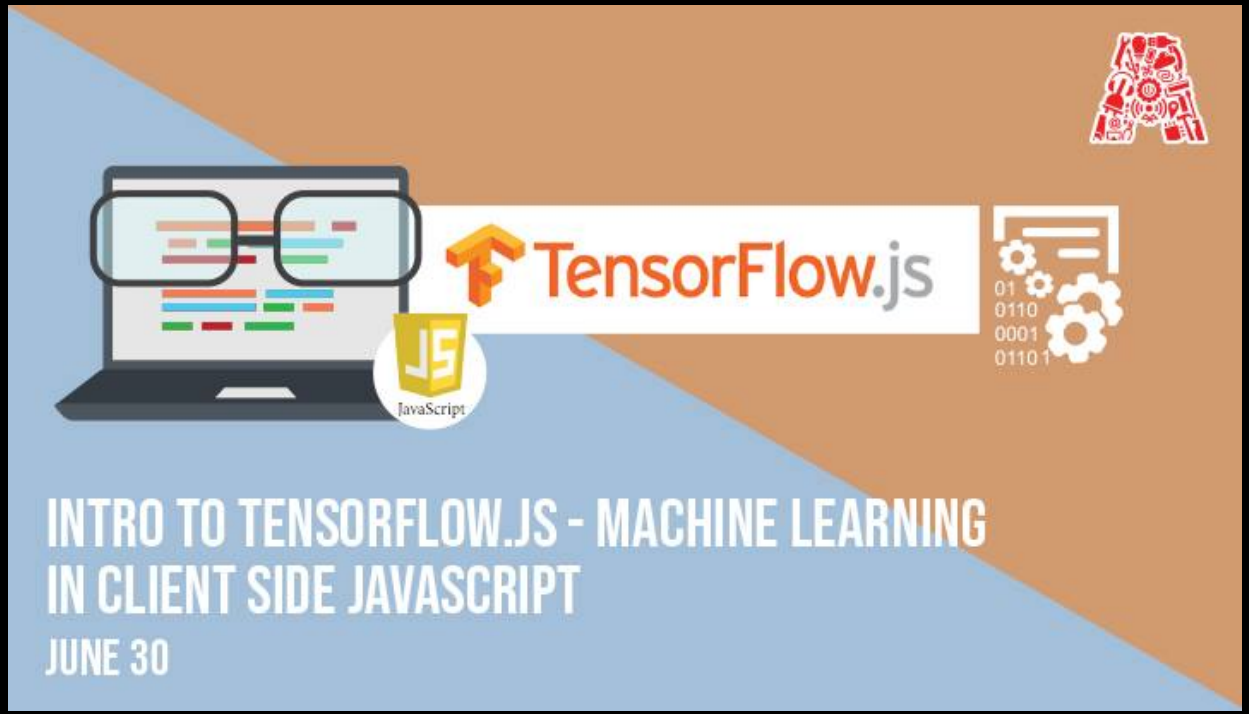


Wifi - In5-Tech [Code - WelcomeToIn5]

About the Assembly

COMMUNITY
INNOVATION
WORKSPACE

- A smart lab based out of In5 since Dec 2014
- Over 200 free workshops done
- ASSEMBLY: HACK - Embedded systems, iOT and hardware
- ASSEMBLY: CODE - Software projects - APIs, frameworks, apps
- Age range: 16-60 - students, professionals, entrepreneurs
- Focus on smart technology and practical applications
- Forum: members.theassembly.ae

تركيب
THEASSEMBLY
MAKE | SMART | THINGS

COMMUNITY
INNOVATION
WORKSPACE

TAG US ON SOCIAL MEDIA!

FACEBOOK - The Assembly (@MakeSmartThings)
TWITTER - @MakeSmartThings
INSTAGRAM - @MakeSmartThings
YOUTUBE - The Assembly

تركيب
THEASSEMBLY
MAKE | SMART | THINGS

Overview

COMMUNITY
INNOVATION
WORKSPACE

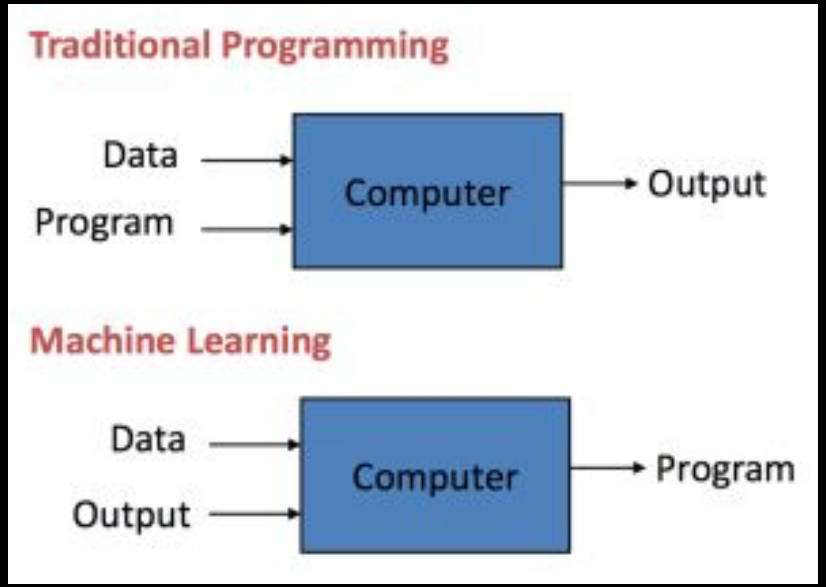
1. Concepts of machine learning
2. What is TensorFlow.js?
3. Practical applications of TensorFlow.js
4. Sample app for regression using TensorFlow.js
5. TensorFlow Playground
6. Introduction to Convolutional Neural Networks
7. Sample app for recognizing handwritten digits

Getting started

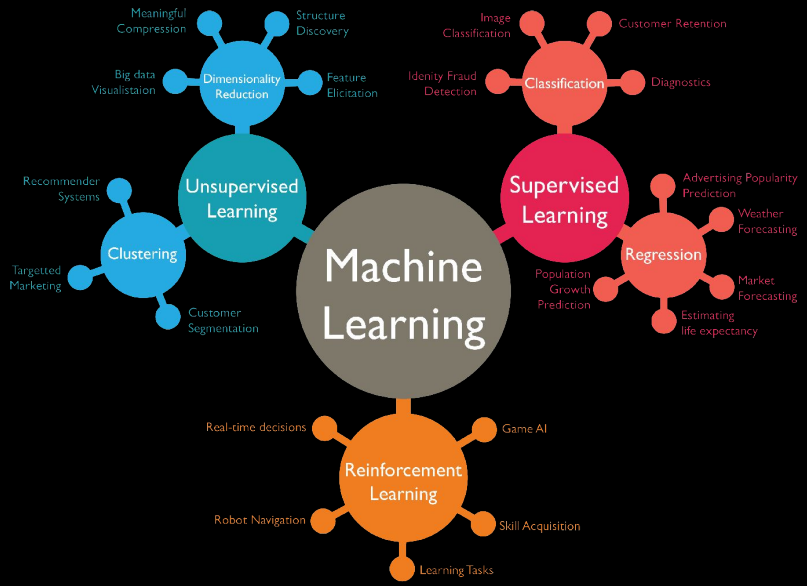
COMMUNITY
INNOVATION
WORKSPACE

1. Install: Visual Studio Code, Node.js
2. All the sample code we've created will be at this location: https://github.com/The-Assembly/IntroToTensorFlowJS
3. Ask for help from the Assembly team if you get stuck with anything!

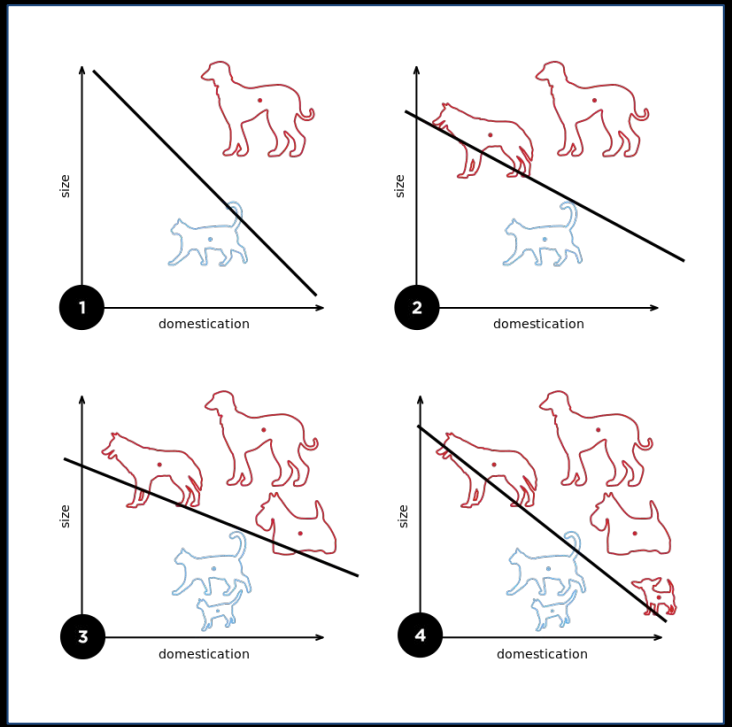
Machine Learning



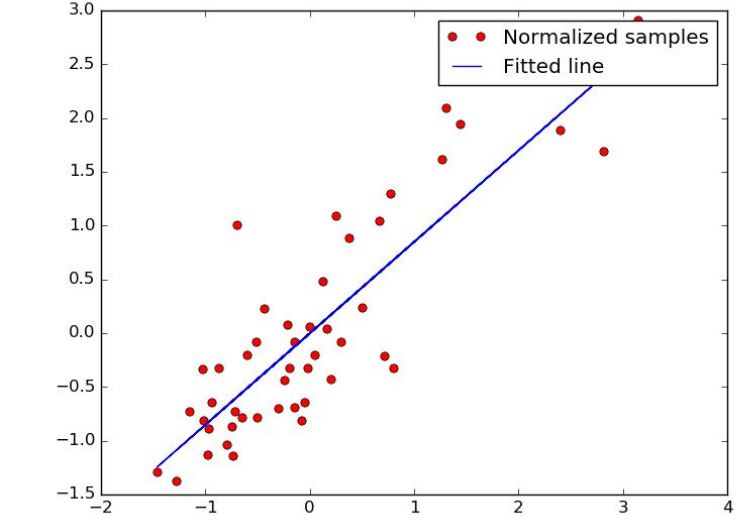
Machine Learning



Classification



Regression

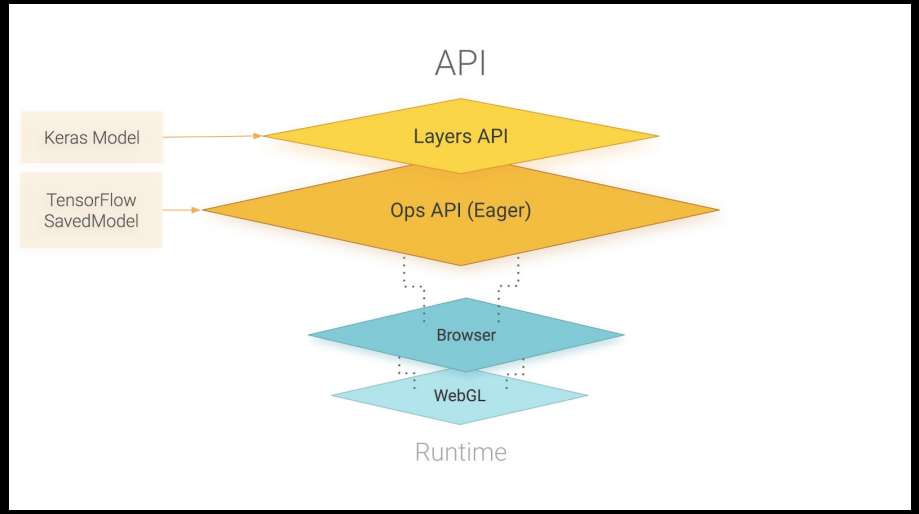

Normalized samples
Fitted line
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2
−1
0
1
2
3
4

TensorFlow.js

COMMUNITY
INNOVATION
WORKSPACE

1. Tensors are n-dimensional containers for numbers
2. TensorFlow
   a. **C++** library, with a popular **Python** wrapper
   b. Created in 2011, open sourced in 2015
3. TensorFlow.js is NOT a wrapper - built from ground up
   a. Started as Deeplearn.js - created in August 2017 by **Nikhil Thorat** and **Daniel Smilkov**
   b. Uses **Javascript** and **WebGL** (for acceleration)

TensorFlow.js



https://js.tensorflow.org/tutorials/core-concepts.html

Regression Code Samples

COMMUNITY
INNOVATION
WORKSPACE

GitHub for method 1, using Layers API:
https://github.com/The-Assembly/IntroToTensorFlowJS/tree/master/Linear_Regression/Using_layers

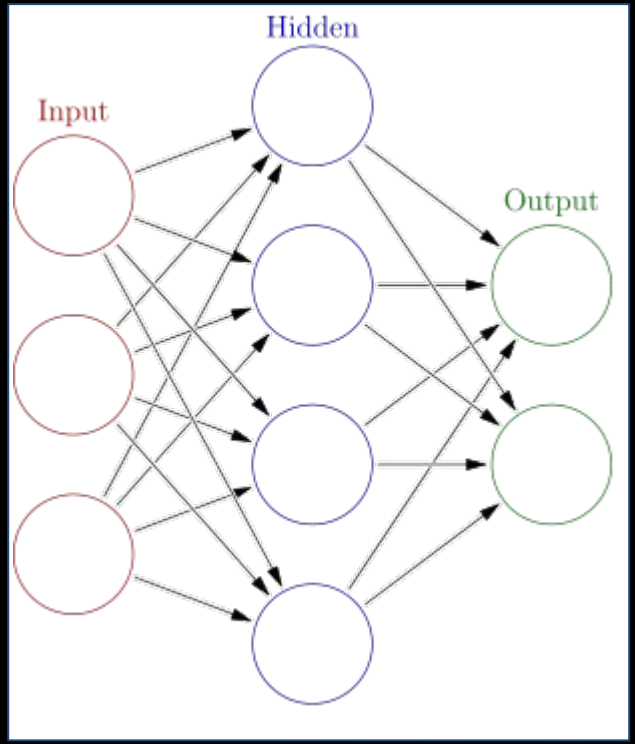
Regression Code Samples

COMMUNITY
INNOVATION
WORKSPACE

GitHub for method 2, using Core API:
https://github.com/The-Assembly/IntroToTensorFlowJS/tree/master/Linear_Regression/Using_core

One layer isn’t enough

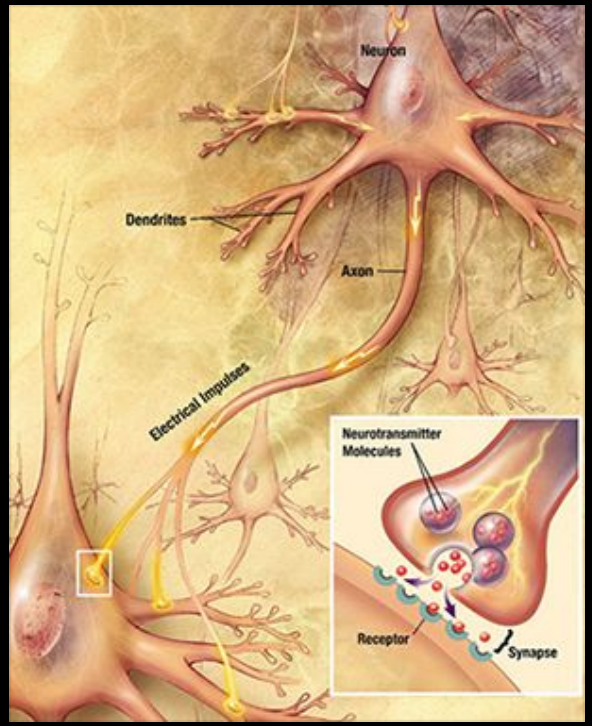
The XOR problem
https://medium.com/@jayeshbahire/the-xor-problem-in-neural-networks-50006411840b



Neural Networks

COMMUNITY INNOVATION WORKSPACE

Built on how your brain works - networks of cells called **neurons** that ‘activate’ each other



Convolution Neural Networks

COMMUNITY
INNOVATION
WORKSPACE

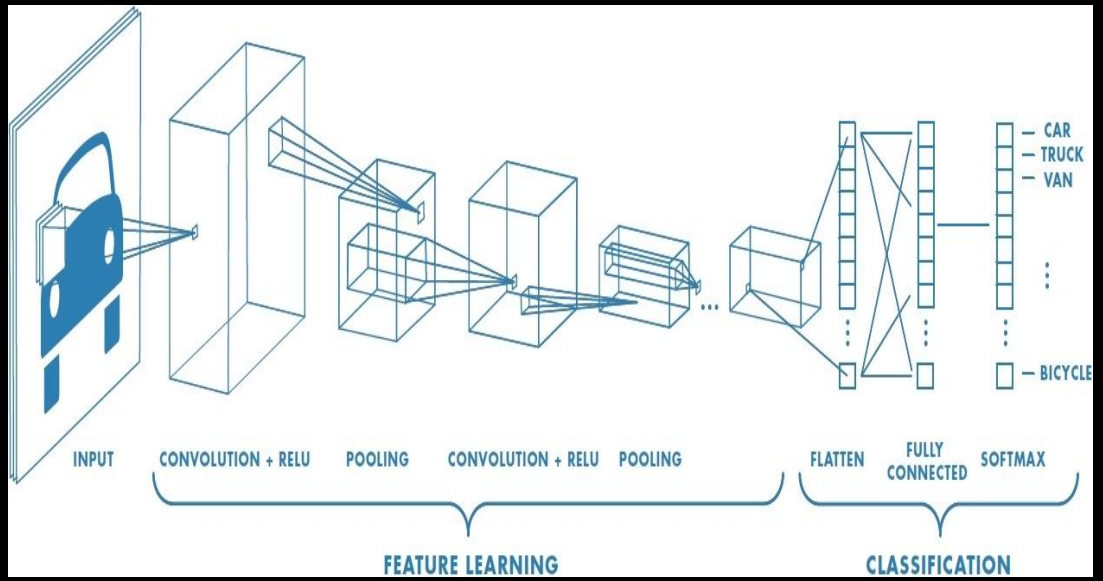
1. https://medium.freecodecamp.org/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050

Convolution Neural Networks

COMMUNITY INNOVATION WORKSPACE

Major Components of a CNN

1. Convolution Layer
2. Non-linear layer
3. Pooling layer
4. Full Connected layer
5. Output layer

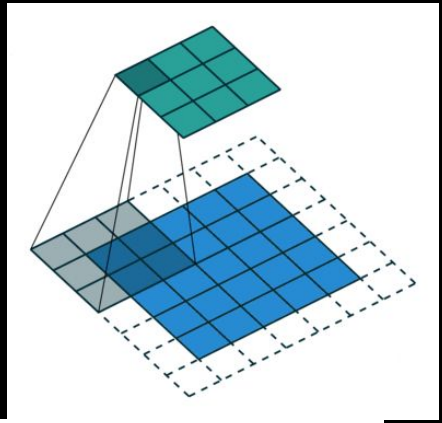


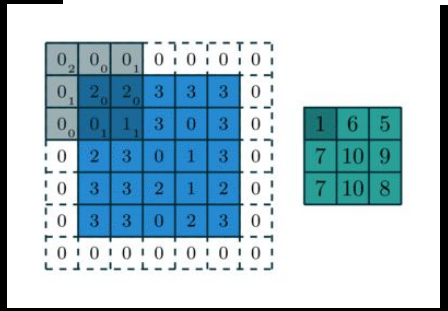
Convolution Neural Networks

COMMUNITY INNOVATION WORKSPACE

Parameters of Convolution Layer:

1. Input Shape: The shape of the data that will flow into the first layer of the model.
2. Kernel Size: The size of the sliding convolutional filter windows to be applied to the input data.
3. Filters:The number of filter windows of size kernelSize to apply to the input data.
4. Strides:The "step size" of the sliding window.
5.



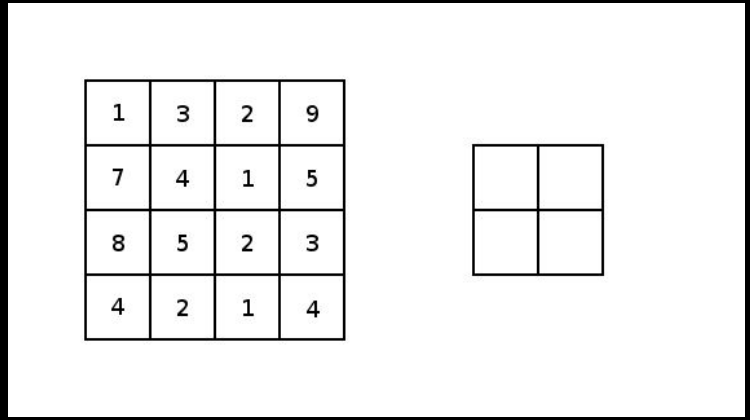


CNN - Max Pooling

Apply a filter and take the max value from that receptive field. After we know we know the specific feature, the exact location of feature is not as important as location of other features.

Pool Size = The size of the sliding pooling windows to be applied to the input data.

Strides = The "step size" of the sliding pooling window



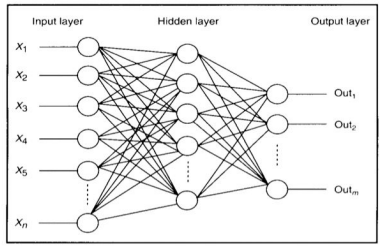
CNN - Fully Connected Layer

COMMUNITY INNOVATION WORKSPACE

Outputs the probability of the image being in any of the N classes. For example in case of MNIST dataset, it will be a 10 x 1 vector. This works by taking output of previous layer and checking the correlation which each of the class.

Output
The neural network outputs a class label

1



TensorFlow Playground

COMMUNITY
INNOVATION
WORKSPACE

http://playground.tensorflow.org

https://www.tensorflow.org/versions/r0.9/tutorials/mnist/beginners/index.html

https://cloud.google.com/blog/big-data/2016/07/understanding-neural-networks-with-tensorflow-playground

TensorFlow.js Models

Method 1:

```
# bash
tensorflowjs_converter --input_format keras \
                path/to/my_model.h5 \
                path/to/tfjs_target_dir
```

Method 2:

```
# Python
import tensorflowjs as tfjs

def train(...):
    model = keras.models.Sequential()
    model.compile(...)
    model.fit(...)
    tfjs.converters.save_keras_model(model, tfjs_target_dir)
```

Digit recognition sample app

COMMUNITY INNOVATION WORKSPACE

GitHub:
https://github.com/The-Assembly/IntroToTensorFlowJS/tree/master/DigitRecognizer

This version loads a pre-trained model output from Keras and converted to the TensorFlow.JS format.

Digit recognition app 2

COMMUNITY INNOVATION WORKSPACE

GitHub: https://github.com/The-Assembly/IntroToTensorFlowJS/tree/master/DigitRecognizer_Browser

This version trains the model in the browser and then outputs the same results as previous sample