Fast Neural Network…

a no brainer!

Agenda

- What types of problems does a neural network solve?
- What exactly is a neural network?
- How does a neural network actually work?
- How to Parallelize a Neural Network
- Demonstrate a Neural Network in Action

Introduction - Riccardo Terrell

- Originally from Italy, currently - Living/working in Washington DC ~10 years
- +/- 19 years in professional programming
  - C++/VB → Java → .Net C# → Scala → Haskell → C# & F# → ??
- DC F# User Group - Organizer
- Working @ statmuse
- *Polyglot programmer – I believe in the art of finding the right tool for the job*

@trikace *rickyterrell.com* *tericcardo@gmail.com*

Goals

Neural Networks ease complex problem solving…

Neural Networks are based on simple linear algebra!

Neural Networks can be parallelized!!

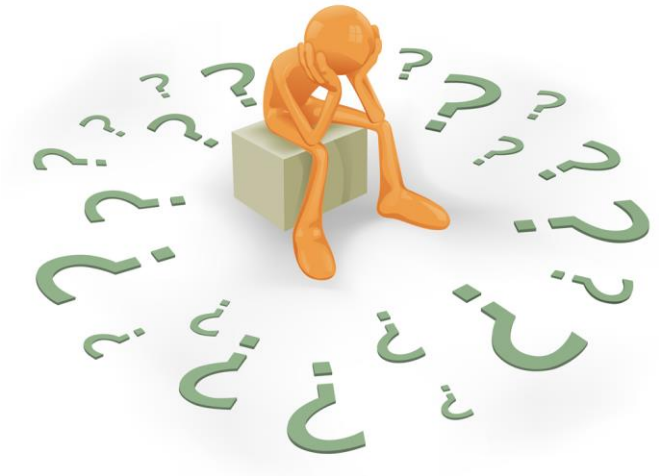
…to be very fast


What about you?

MOTIVATION

Machine Learning Algorithms

- *Linear Regression*
- *Logistic Regression*
- *Decision Tree*
- *SVM*
- *Naive Bayes*
- *KNN*
- *K-Means*
- *Random Forest*
- *Dimensionality Reduction Algorithms*
- *Gradient Boost*
- *Adaboost*



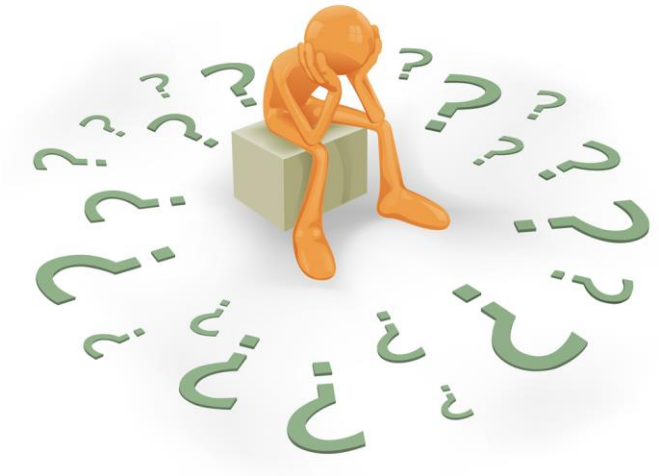
Supervised Learning

Unsupervised Learning

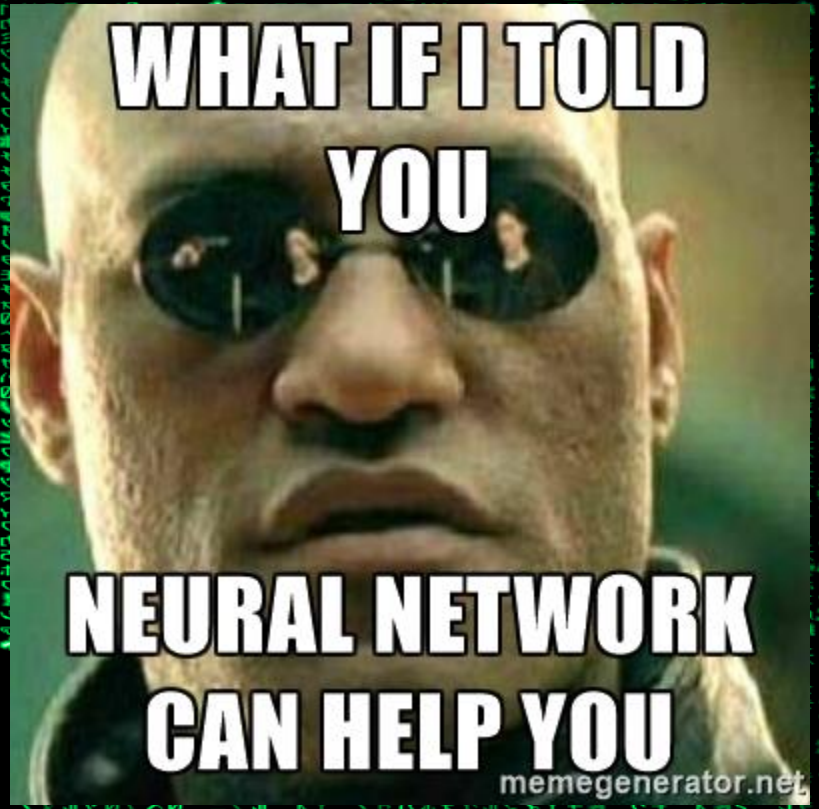
dataaspirant.wordpress.com

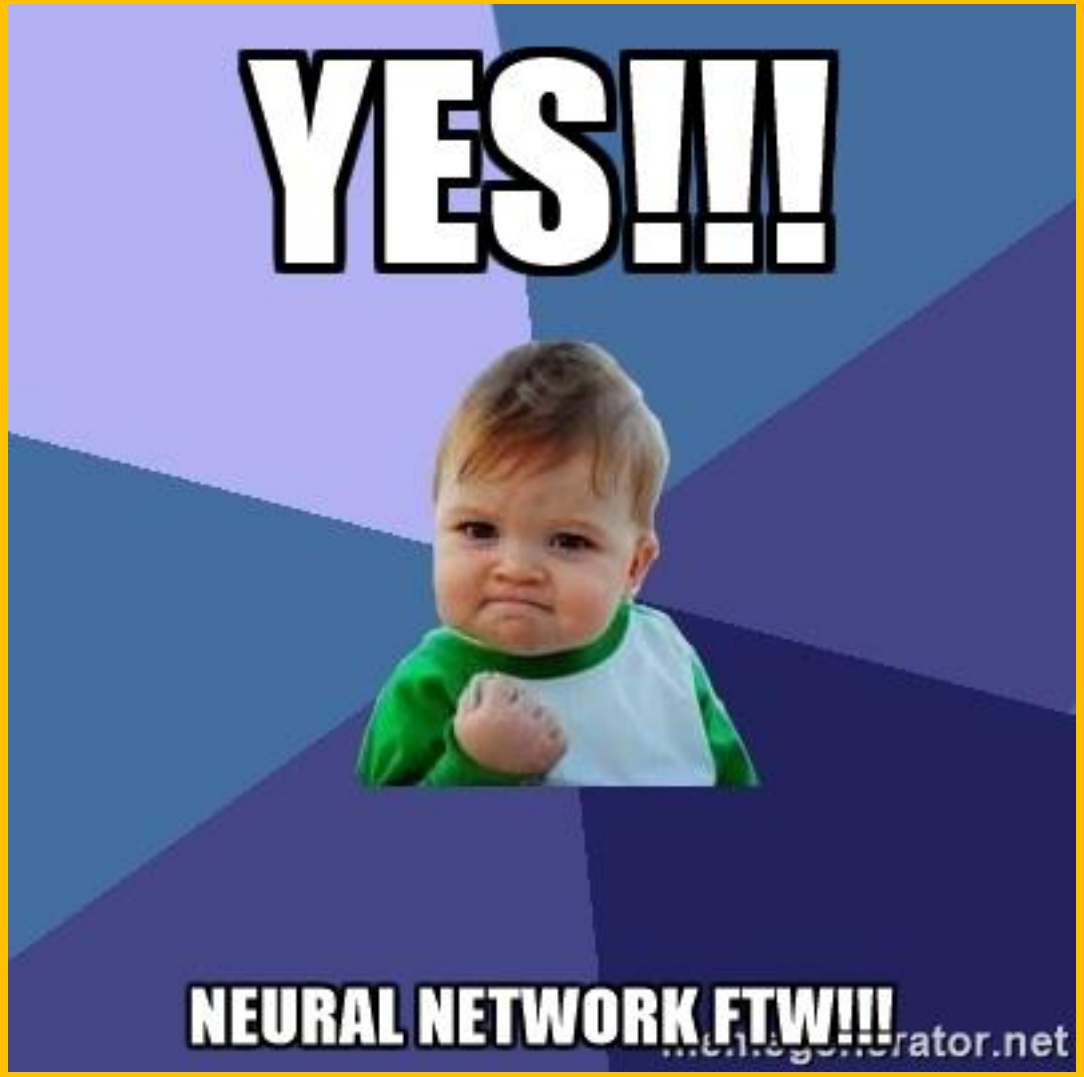
Unsupervised & Supervised Learning

- unsupervised learning:
  - gaussian mixture models
  - latent semantic analysis (better pca for text)
  - self organizing hiddens
  - smarter interfaces (API to deal with reduction chaining)
  - multi-processor, gpgpu



- supervised learning:
  - support vector machines (non-linear classification models)
  - stochastic models (naïve bayes, etc.)
  - neural networks
  - simplified interface (system chooses appropriate model by interrogating label)
  - probabilistic graphical models
  - multi-processor, gpgpu


WHAT IF I TOLD YOU
NEURAL NETWORK CAN HELP YOU
memegenerator.net

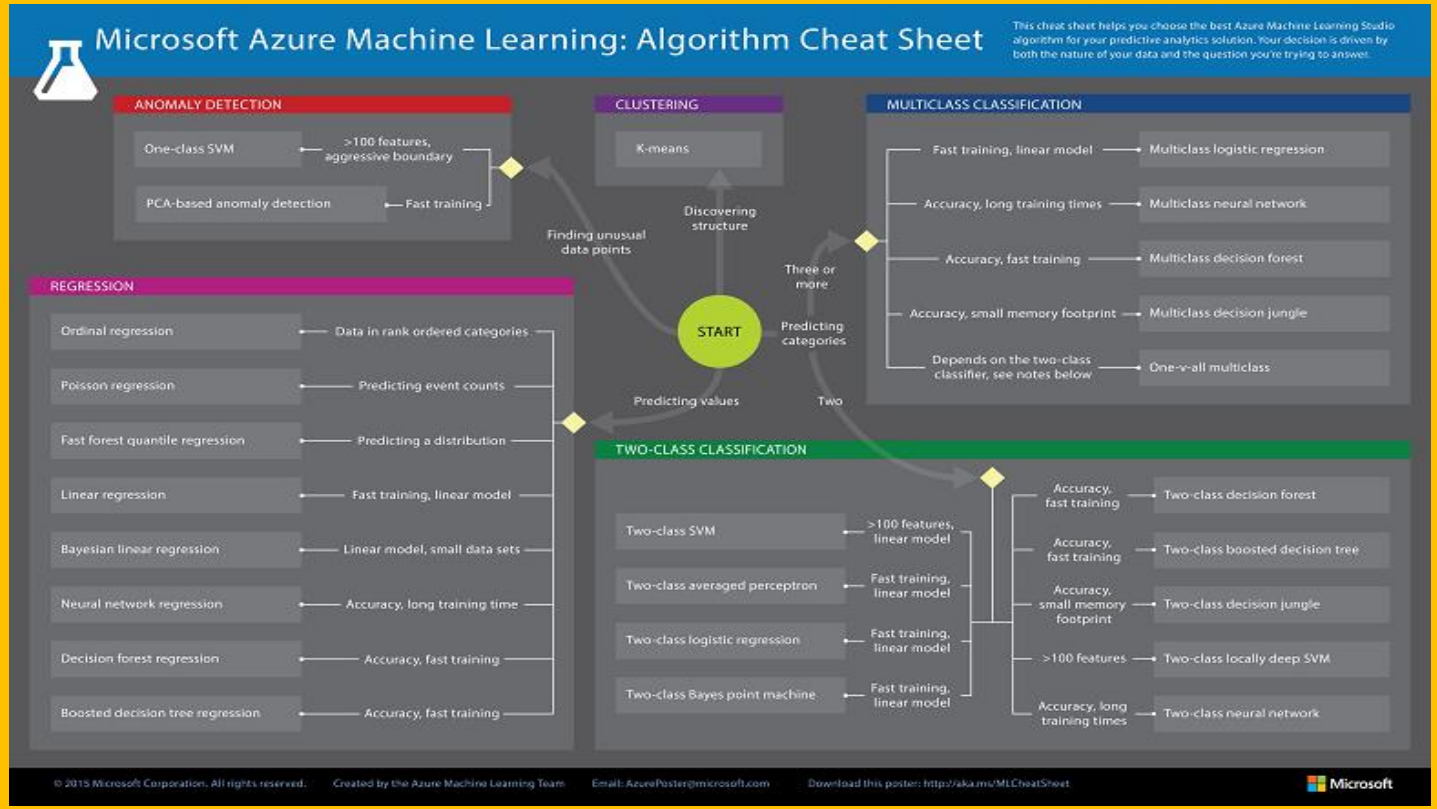

YES!!!
NEURAL NETWORK FTW!!!

Neural Networks are *multipurpose*

A Neural network is an alternative to:

1. *Linear regression:*
2. *Logistic regression*
3. *Naive Bayes*
4. *Decision trees*
5. *Adaptive boosting*
6. *Support vector machines*

Microsoft Azure ML Cheat Sheet



https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-cheat-sheet

What kind of problems does a Neural Network solve?

Problems NN solve

- *OCR (optical character recognition)*
- *Natural language processing*
- *Machine translation*
- *Biology & medicine*
- *Robotics (Autonomous Systems)*
- *Pattern Recognition Time Series Prediction*
- *Signal Processing*
- *Control*
- *Soft Sensors*
- *Anomaly Detection*

Recommendation systems

- what do people like?

Recommendation systems

- how are things alike?

NN Prediction Example

Grade	GPA	Age	Gender	Sport
A	3.5	16	M	Baseball
C	2.0	12	M	Football
F	2.1	13	F	Soccer
B	3.5	17	M	Baseball
D	2.0	18	F	Soccer
A	3.8	15	M	Baseball
D	2.3	14	F	Baseball
B	3.3	17	M	????

Training data

features
gpa, age, gender, sport

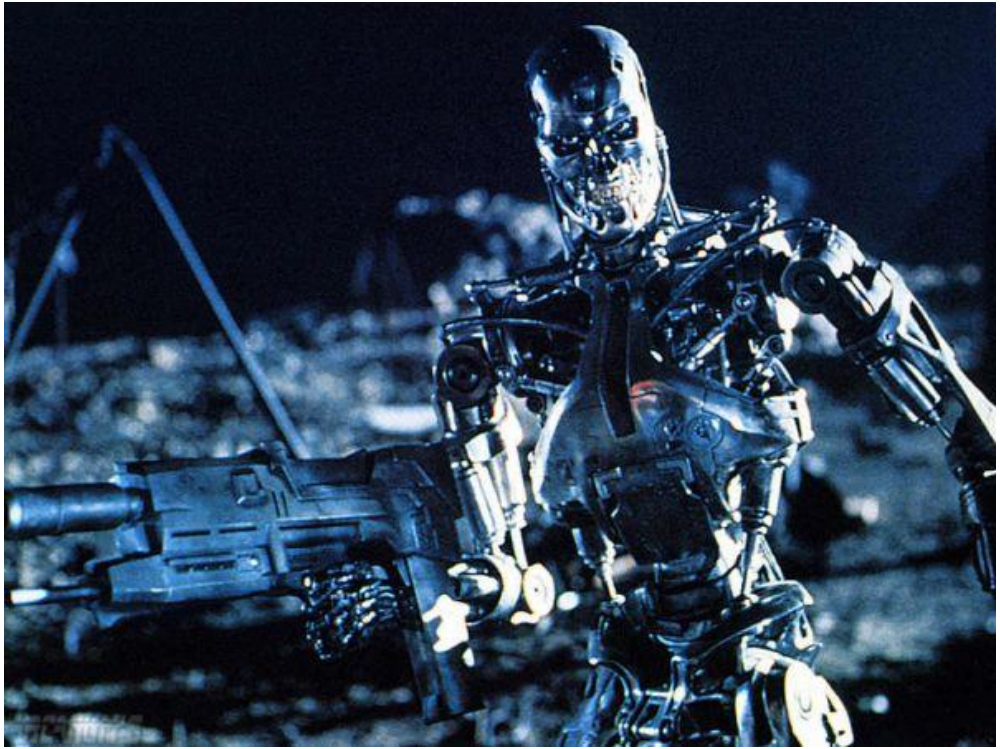
values (x)
[B, 3.3, 17, M, Baseball]

Independent variables/predictors/ attributes/regressors/x-values

"The thing to classify (predict)"/ dependent variable/y

What is a Neural Network

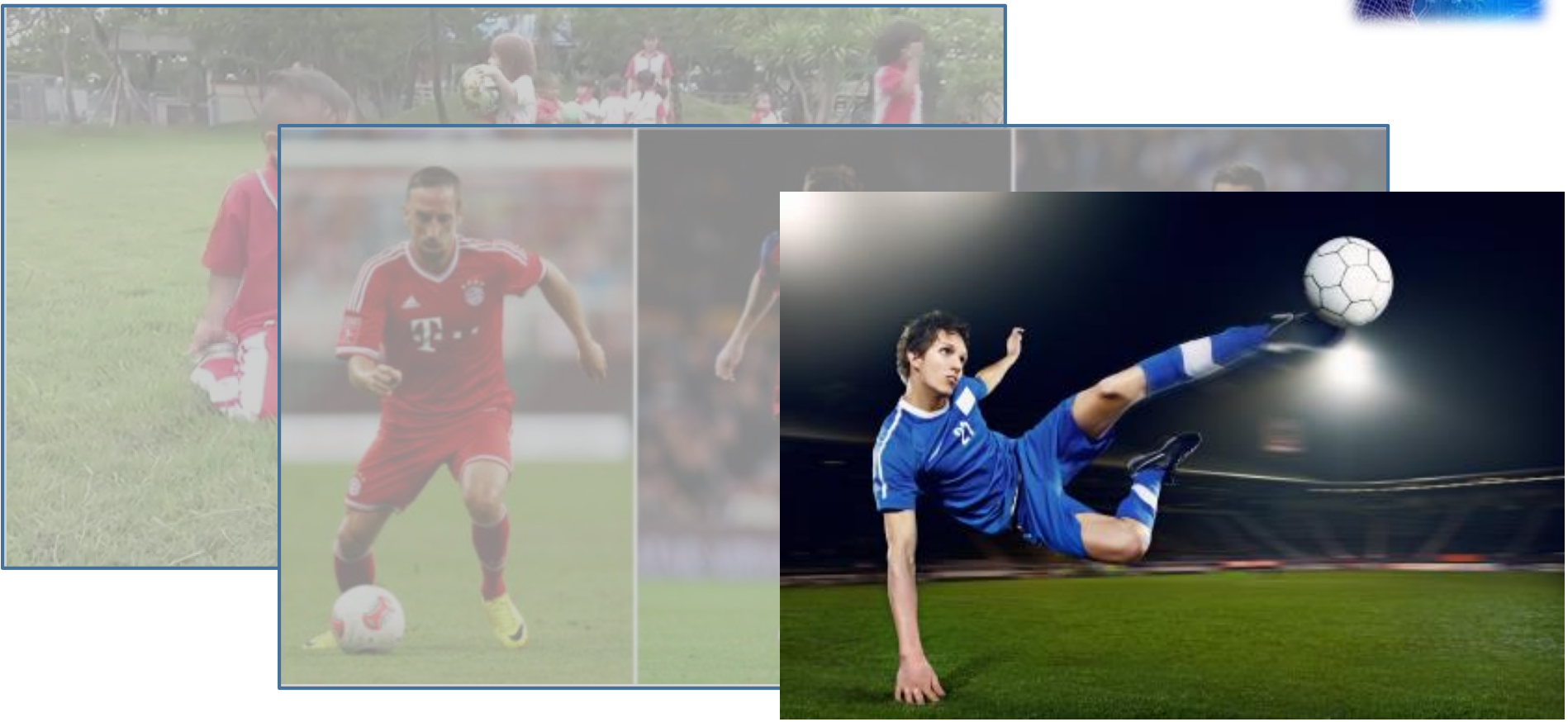
What is this Neural Network?



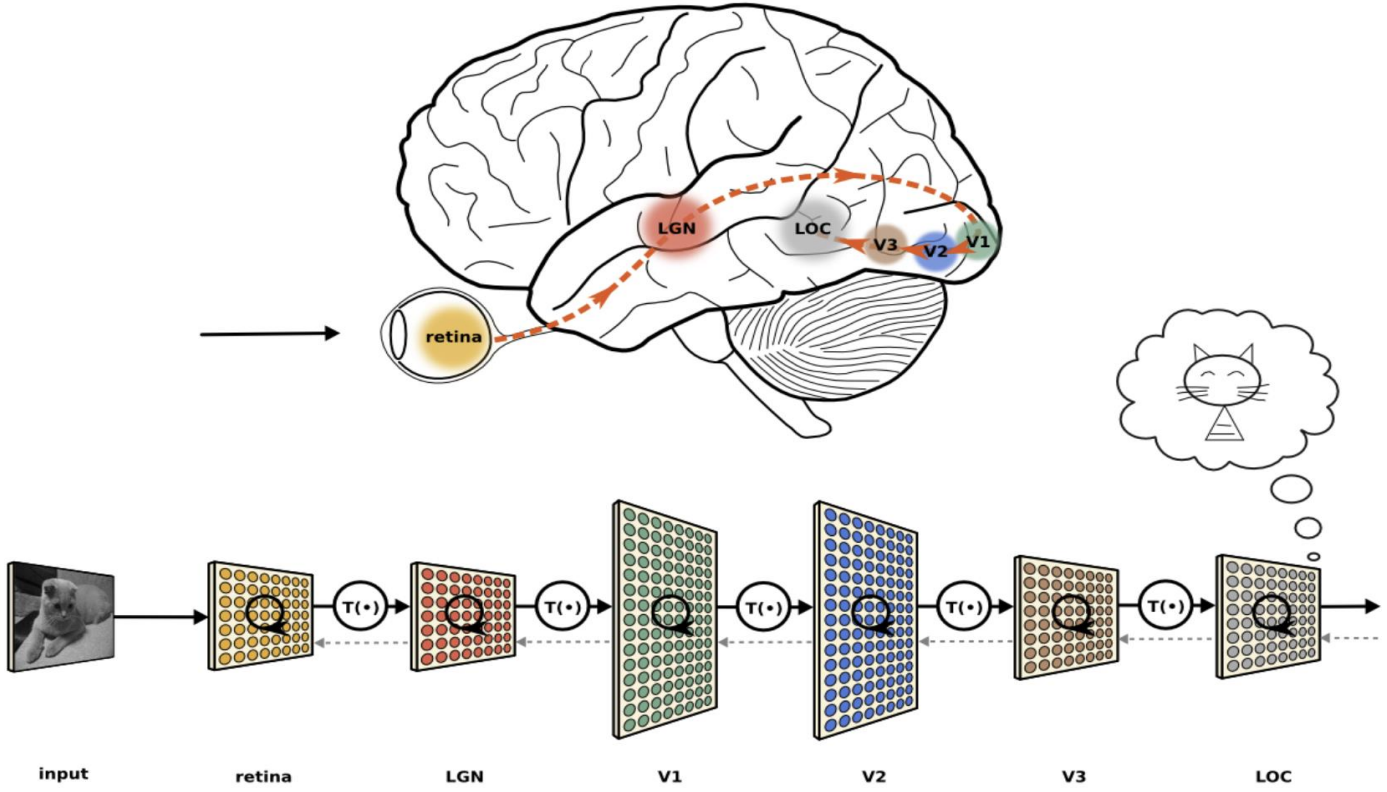
Is about learning/growing

Is about learning/growing

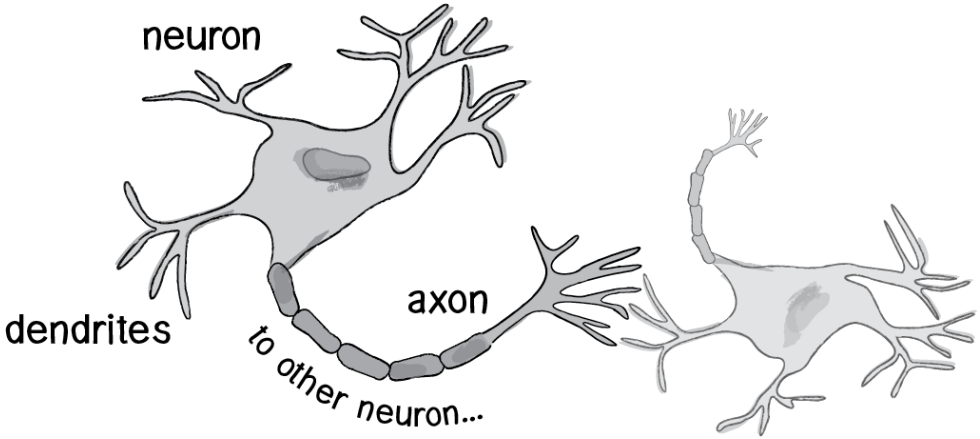
Is about learning/growing



What is this?


LGN
LOC
V3
V2
V1
retina
T(•)
T(•)
T(•)
T(•)
T(•)
input
retina
LGN
V1
V2
V3
LOC

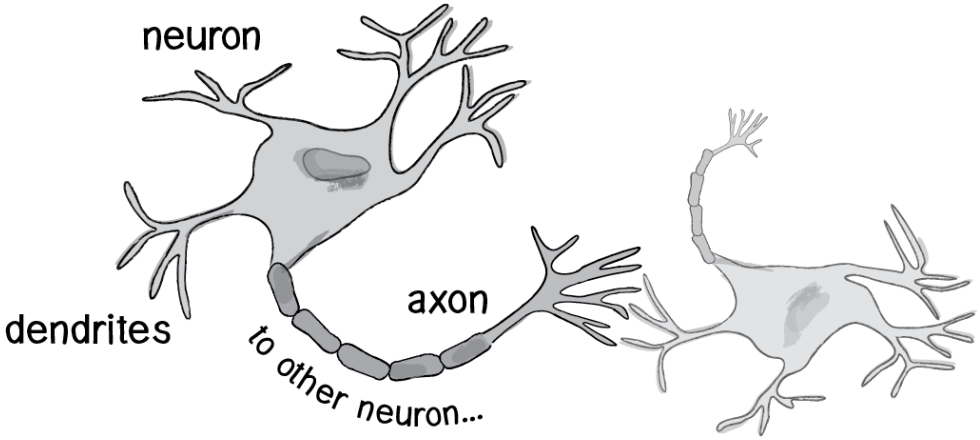
What is Neural Network

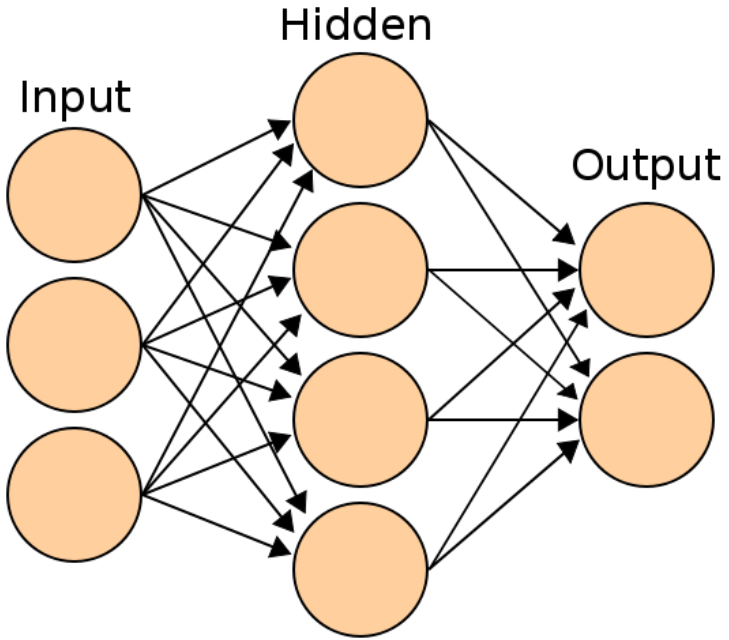


Four parts of a typical nerve cell :

- **DENDRITES**: Accepts the inputs
- **SOMA** : Process the inputs
- **AXON** : Turns the processed inputs into outputs.
- **SYNAPSES** : The electrochemical contact between the neurons.

What is Neural Network





- A neuron: many-inputs / one-output unit
- output can be *excited* or *not excited*
- incoming signals from other neurons determine if the neuron shall excite ("fire")
- Output subject to attenuation in the *synapses,* which are junction parts of the neuron

Mathematical Background

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 2 & 5 \end{bmatrix}$$

Matrix Operations

$$d/dx * x^2 = 2x$$

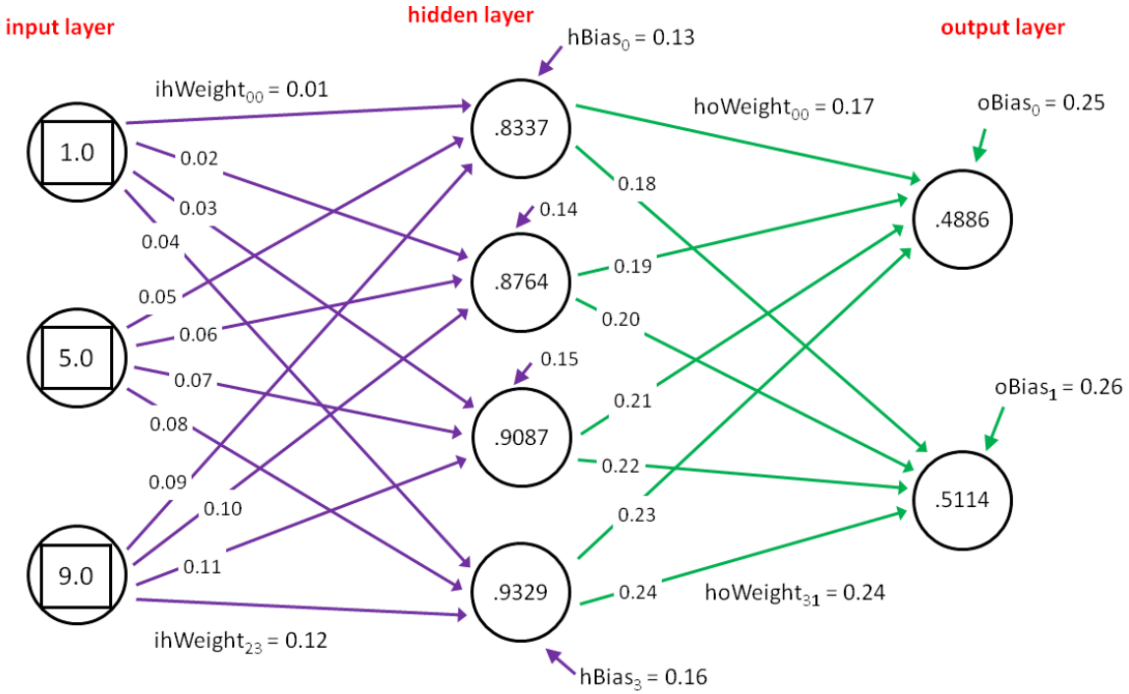
Equation for derivatives and Integrals

Some vocabulary

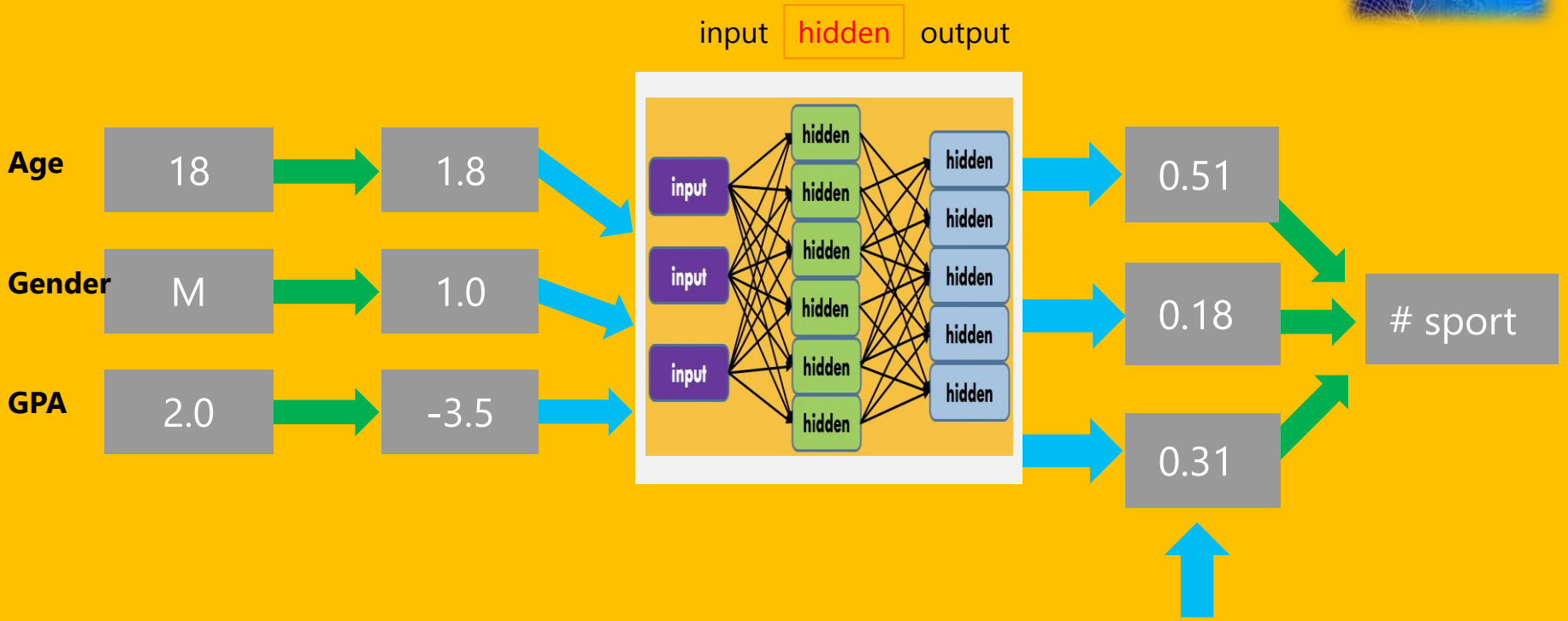
Biological Terminology	**Artificial Neural Network Terminology**
Neuron	Node/Unit/Cell/Neurode
Synapse	Connection/Edge/Link
Synaptic Efficiency	Connection Strength/Weight
Firing frequency	Node output

How does a Neural Network work?

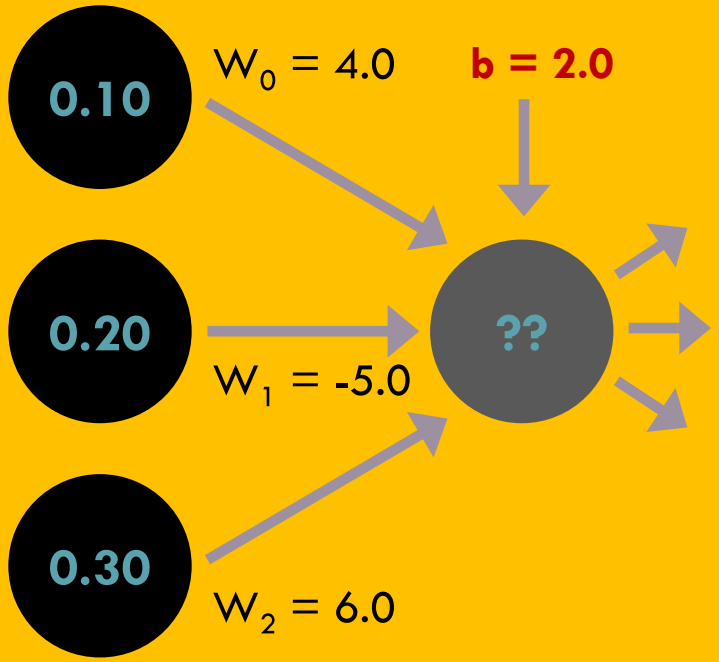
Neural Network Classification



What's going on?



The perceptron



a. (0.1)(4.0) + (0.2)(-5.0) + (0.3)(6.0) = 1.2

b. 1.2 + 2.0 = 3.2

c. Activation Function (3.2) = 0.73

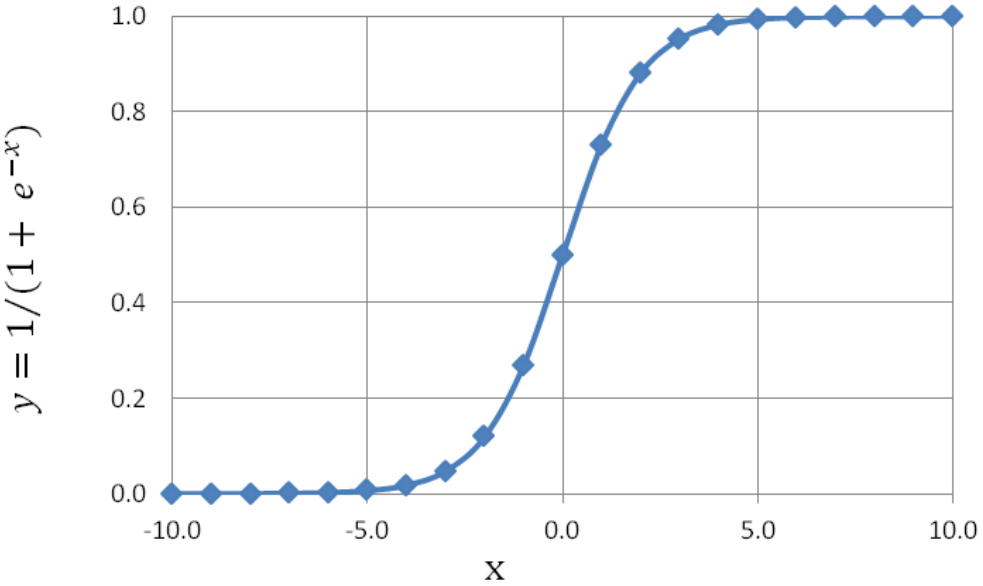
d. Update weight

```
for i = 0 to layers.Length - 2 do
    Parallel.For(0, layers[i].Length, (fun j ->
      let mutable num = 0.0f;
      for (int z = 0; z < layers.[i + 1].Length; ++z)
        num <- num + layers.[i + 1][z]  * layrer.[i].[j, z]
      layers.[i].[j, z] <- sigmoid layers[i][j].output * num
    ))

for i = 0 to layers.Length - 1 do
    Parallel.For(0, layers.[i + 1].Length, (fun j ->
      for z = 0 to layers.[i].[i].Length -1 do
        layers.[i].[i, z] <- layers.[i].[i, z] +
                learningRate * layers.[i + 1].[j].output ))
```

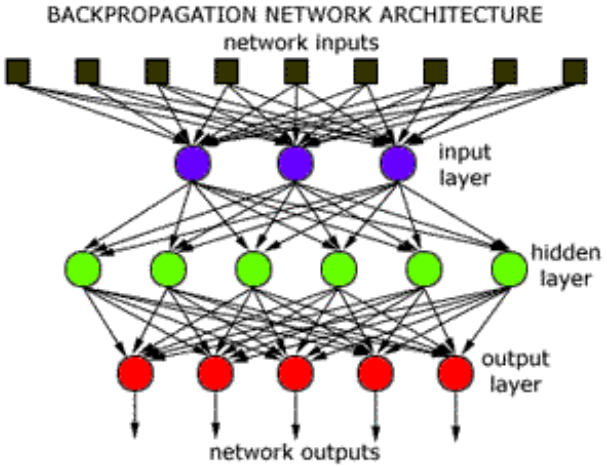
Activation functions

- Logistic sigmoid
  - Output between [0, 1]
  - y = 1.0 / (1.0 + e–x)
- Hyperbolic tangent
  - Output between [-1, +1]
  - y = tanh(x) = (ex – e-x) / (ex + e-x)
- Heaviside step
  - Output either 0 or 1
  - if (x < 0) then y = 0 else if (x >= 0) then y
- Softmax
  - Outputs between [0, 1] and sum to 1.0
  - y = (e-xi) / Σ (e-xj)



Neural Network Training

- Back-propagation
  - Fastest technique.
  - Does not work with Heaviside activation.
  - Requires “learning rate” and “momentum.”
- Genetic algorithm
  - Slowest technique.
  - Generally most effective.
  - Requires “population size,” “mutation rate,” “max generations,” “selection probability.”
- Particle swarm optimization
  - Good compromise.
  - Requires “number particles,” “max iterations,” “cognitive weight,” “social weight.”



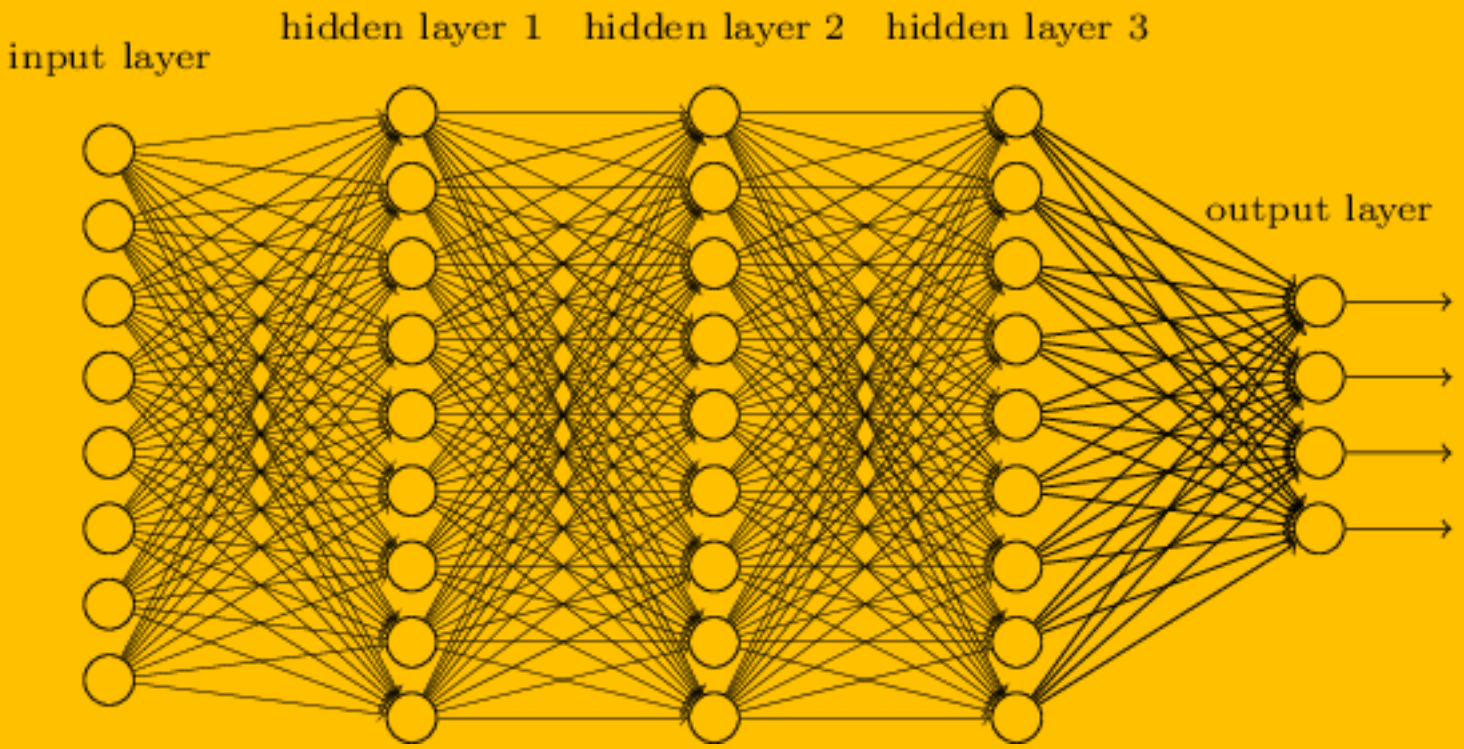
NN Tips

- Ensure you have a good weight initialization
- Choose the right number of layers
- Choose the right number of neurons for each layer
- Avoid overfitting (dropout)

	Similar dataset	Different dataset
little data	Use linear classifier on top layer	*Problems!* *Try liner classifier from different stages*
lot of data	Fine-tune a few layers	Fine-tune a large number of layers

Why a Parallel Neural Network?

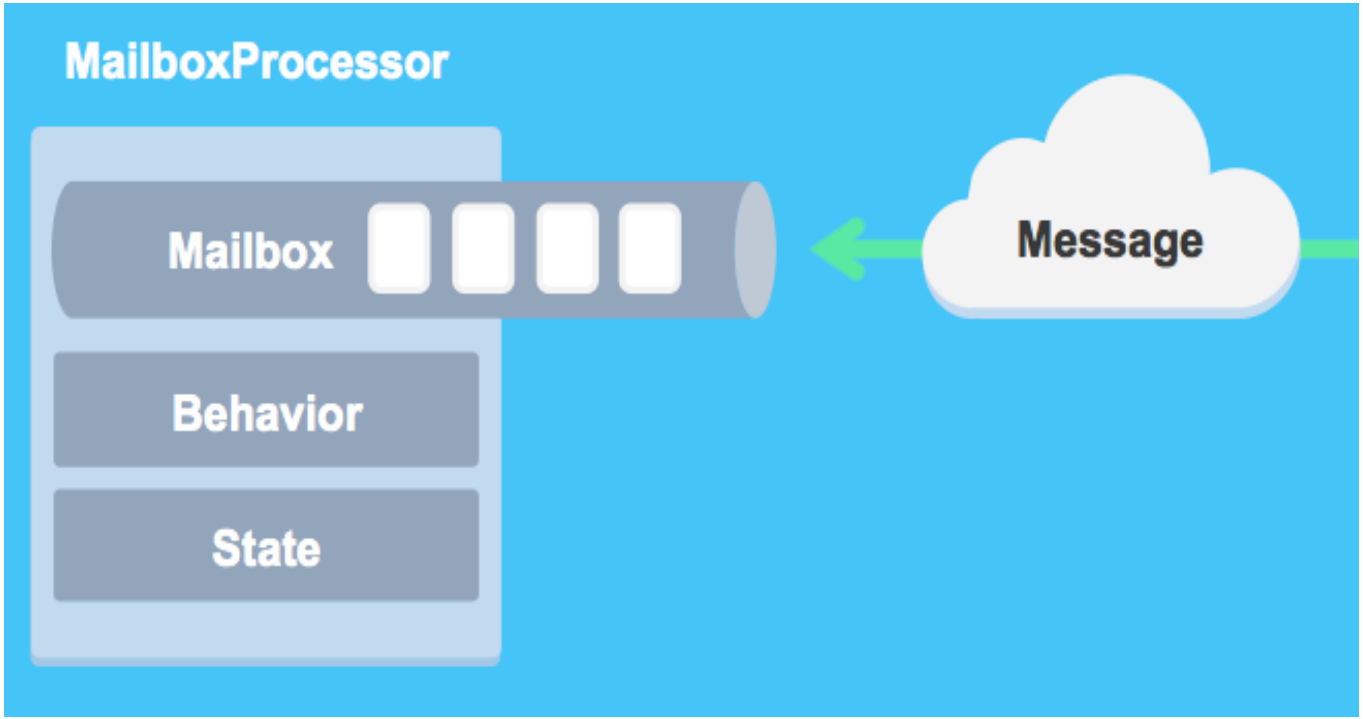
Why a Parallel Neural Network?



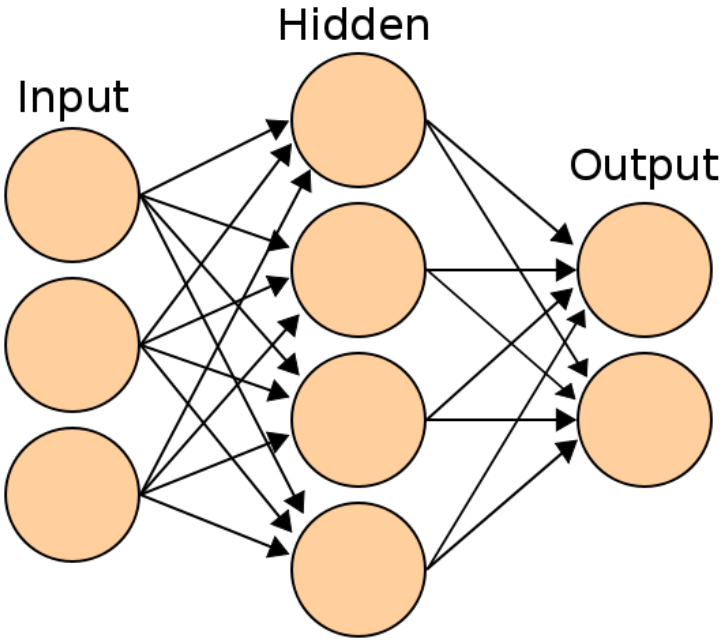
Group of Nodes connected that *communicate* each other

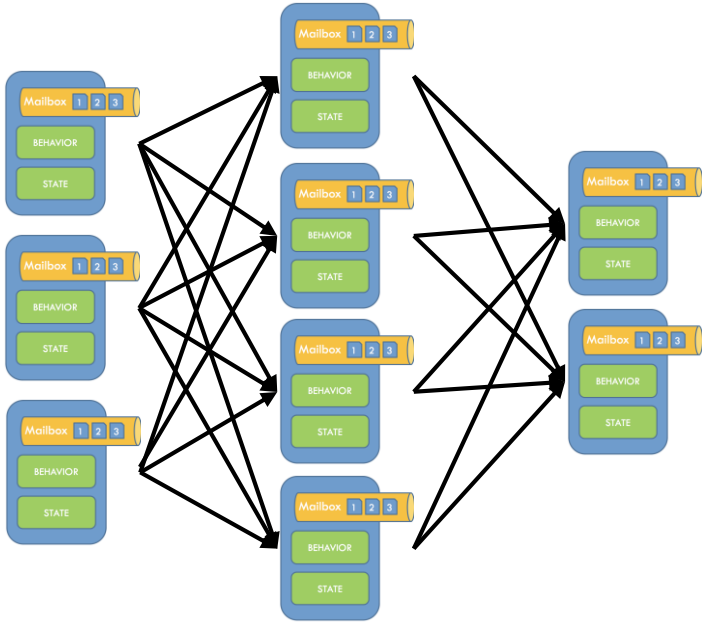
It sounds familiar?

Neural Network Agent based

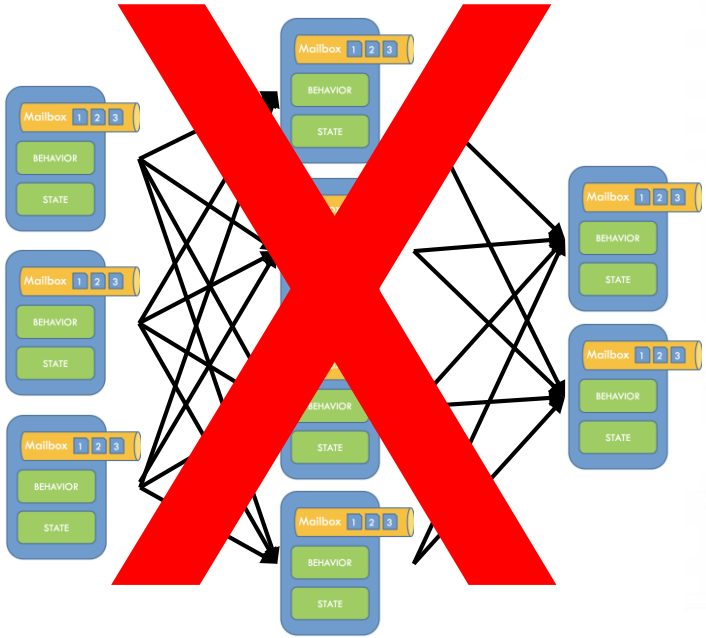


Neural Network Agent based





Neural Network Agent based



Neural Network is Functional

```
(inputs : Vector<float>)  ->  (output : Vector<float>)
```

Neural Network in code

```
let appendValue value (vec : Vector<float>) =
    vector [ yield! value :: (vec |> Vector.toList) ]


let appendBias : Vector<float> -> Vector<float> = appendValue 1.0


let layer activationF (weights : Matrix<float>)
            (inputs : Vector<float>) : Vector<float> =
    (weights * inputs) |> Vector.map  activationF |> appendBias
```

Neural Network in code

```
let sigmoid x = 1.0 / (1.0 + exp(-x))
```

```
let lm1 = matrix [ [-2.1; 0.4; 0.8; 0.9];
                   [-1.0; 0.1; 0.3; 1.3];
                   [-1.5; 0.7; 0.5; 0.8];
                   [-1.8; 0.3; 0.4; 0.9]; ]
let lm2 = matrix [ [-1.5; 0.5; 0.5; 0.5; 0.5];
                   [-2.5; 0.5; 0.5; 0.5; 0.5]; ]
let lm3 = matrix [ [-0.5; 0.2; 0.1];
                   [-1.5; 0.3; 0.7]; ]
```

```
let threeLayerNetwork : (Vector<float> -> Vector<float>) =
    layer sigmoid lm1 >> layer sigmoid lm2 >> layer sigmoid lm3

let run (input : Vector<float>) =
    (threeLayerNetwork <| prependForBias input).[ 1 .. ]
```

```
>> : ('a -> 'b) -> ('b -> 'c) -> 'a -> 'c
```

Neural Network in code

```
let networkComp
    (mxs : (Matrix<float> * (float -> float)) list) =
     mxs |> List.fold(fun acc (ws,f)-> acc >> layer f ws) id


let execute v =
       [ (lm1, sigmoid); (lm2, sigmoid); (lm3, sigmoid) ]
       |> networkComp
       |> run v

execute (vector [1.0; 2.0; 3.0])
```

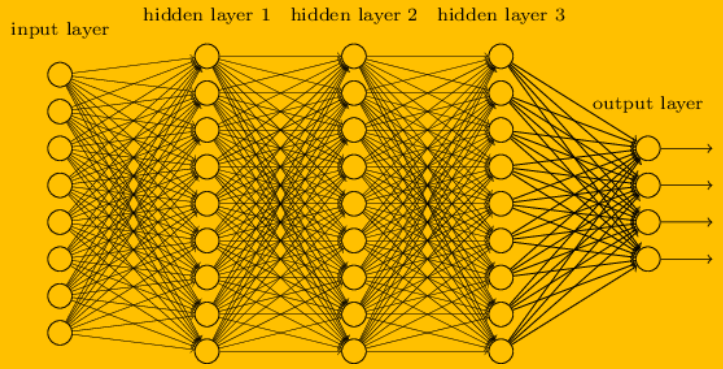
RESULT ➔ Vector<float> = seq [0.5392375632; 0.3009608905]

```
let adjustWeights (neuron:Neuron) (value:float) =
        neuron.output * (1.0 - neuron.output) |> fun derivative ->
        neuron.error <- value
        for i in [0..neuron.weights.Length-1] do
                neuron.weights.[i].Value <- neuron.weights.[i].Value +
                        (neuron.error * learnRate * neuron.weights.[i].output)
        neuron.bias <- neuron.bias + neuron.error * learnRate * derivative


let training (network:Network) =
    let adjustWeights (delta:float) =
        for output in network.outputs do adjustWeights output delta
        for neuron in network.neurons do adjustWeights neuron(feedback output neuron)

    let mutable error = 0.0
    for pattern in network.Patterns do
        for i in [0..pattern.TeachingSignal.Length - 1] do
            let output = (networkActivate network pattern).[i]
            let delta = pattern.trainingSignal.[i] - output
            adjustWeights delta
            error <- error + Math.Pow(delta, 2.0)
    { network with Error = error }
```

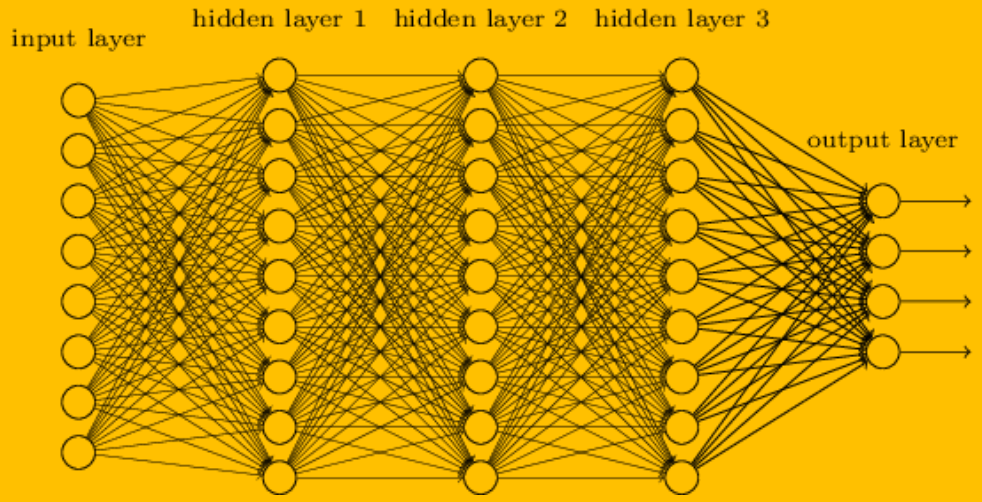
Why a Parallel Neural Network?

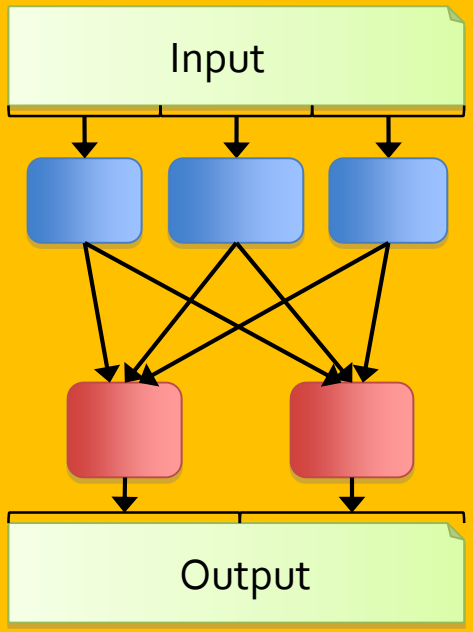


Typical structure of an NN:

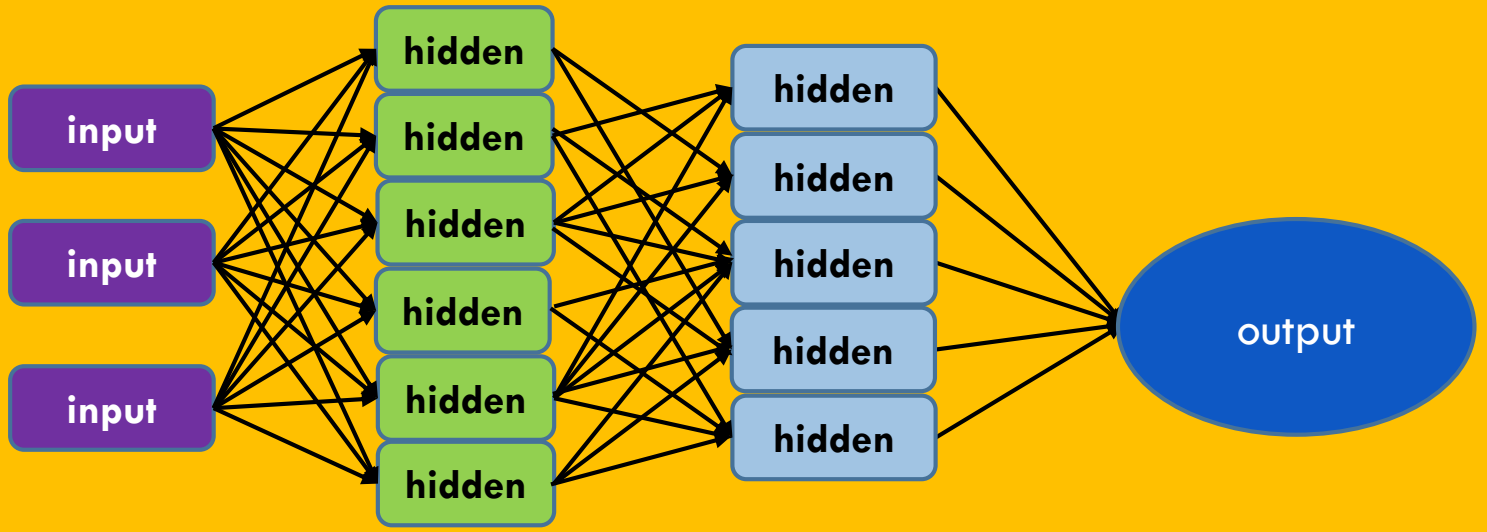
- For each training session
  - For each layer (forward and backward)
    - For each neuron in the layer
      - For all weights of the neuron
        - Compute

Why a Parallel Neural Network?





Parallel NN as Map Reduce



Parallel NN as Map Reduce

```
let mapF  M (map:'value -> seq<'key * 'value>)
        (inputs:seq<'value>) =
    inputs
    |> PSeq.withExecutionMode ForceParallelism
    |> PSeq.withDegreeOfParallelism M
    |> PSeq.collect (map)
    |> PSeq.groupBy (fst)
    |> PSeq.toList
```

```
let reduceF R (reduce:'key -> seq<'value> -> 'reducedValues)
            (inputs:('key * seq<'key * 'value>) seq) =
        inputs
        |> PSeq.withExecutionMode ForceParallelism
        |> PSeq.withDegreeOfParallelism R
        |> PSeq.map (fun (key, items) ->
            items |> Seq.map (snd)
            |> reduce key)
        |> PSeq.toList
```

Parallel NN as Map Reduce

```
let mapF  M (map:'value -> seq<'key * 'value>)
      (inputs:seq<'value>) =
    inputs
    |> PSeq.withExecutionMode ForceParallelism
    |> PSeq.withDegreeOfParallelism M
    |> PSeq.collect (map)
    |> PSeq.groupBy (fst)
    |> PSeq.toList
```

```
let reduceF R (reduce:'key -> seq<'value> -> 'reducedValues)
          (inputs:('key * seq<'key * 'value>) seq) =
      inputs
      |> PSeq.withExecutionMode ForceParallelism
      |> PSeq.withDegreeOfParallelism R
      |> PSeq.map (fun (key, items) ->
         items
         |> Seq.map (snd)
         |> reduce key)
      |> PSeq.toList
```

```
let mapReduce =
      (map:'value -> seq<'key * 'value>)
      (reduce:'key -> seq<'value> -> 'reducedValues)
      M R =
          (mapF M map >> reduceF R reduce)
```

Parallel NN as Map Reduce

```
let threeLayerNetwork : (Vector<float> -> Vector<float>) =
    layer sigmoid lm1 >> layer sigmoid lm2 >> layer sigmoid lm3
```

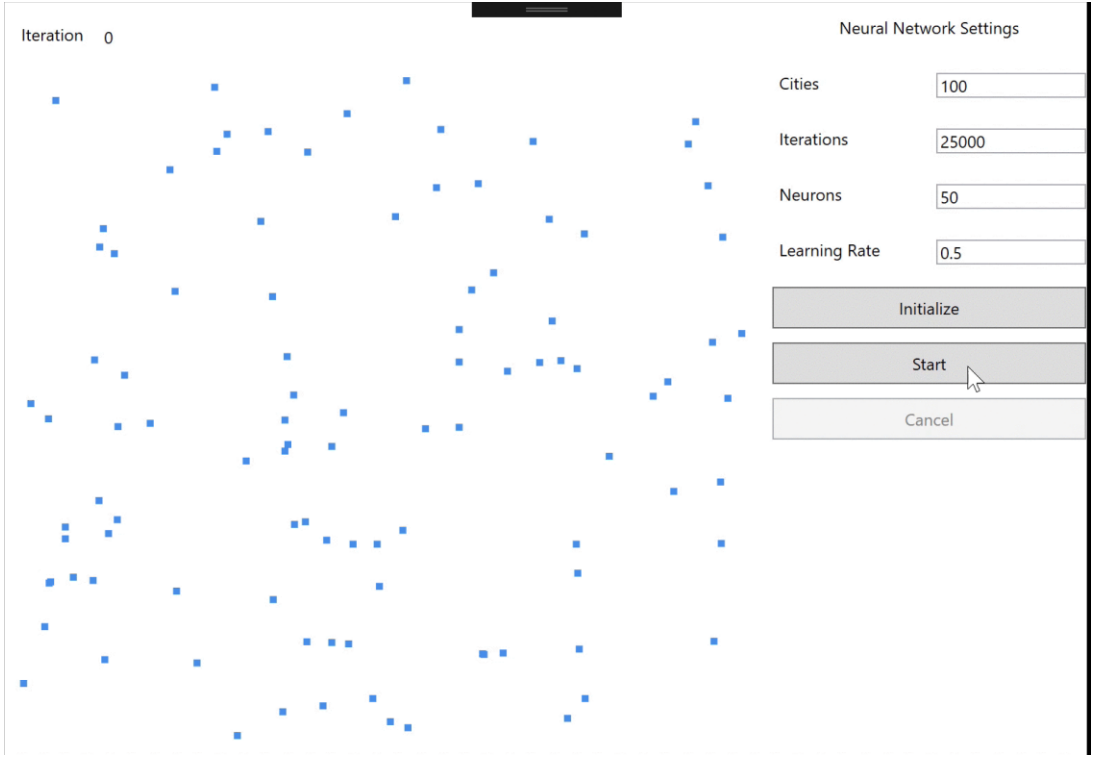
```
let mapReduce
        (map:'value -> seq<'key * 'value>)
        (reduce:'key -> seq<'value> -> 'reducedValues)
        M R =
            (mapF M map >> reduceF R reduce)
```

F# Advent Calendar in English 2016

Update (11/28/2016): Dear writers, this year we are going to make an e-book from #FsAdvent posts and to raise money for a good cause (read more here). If you do not allow to use your post, please let me know.

https://sergeytihon.wordpress.com

The Traveling ~~Salesman~~ Santa Problem



Elastic Network

is a type of neural network which utilizes unsupervised learning algorithms for clusterization problems and treats neural networks as a ring of nodes.

The learning process keeps changing the shape of the ring, where each shape represents a possible solution

The Traveling ~~Salesman~~ Santa Problem

Parallelized neural network system for solving Euclidean traveling salesman problem

Bihter Avşar[a], Danial Esmaeili Aliabadi[a,*]

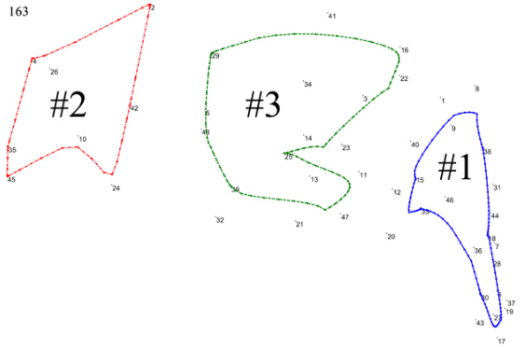
[a] *Sabanci University, Faculty of Engineering and Natural Science, Istanbul, Turkey.*

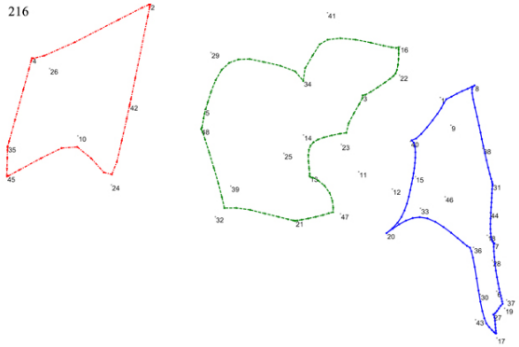
Abstract

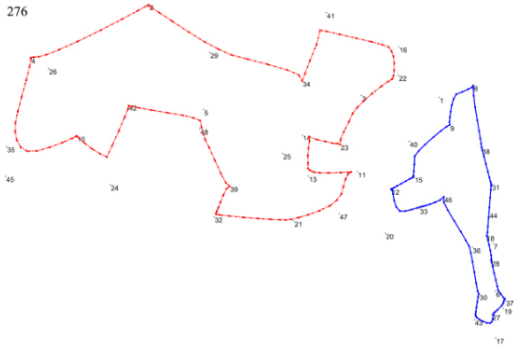
We investigate a parallelized divide-and-conquer approach based on a self-organizing map (SOM) in order to solve the Euclidean Traveling Salesman Problem (TSP). Our approach consists of dividing cities into municipalities, evolving the most appropriate solution from each municipality so as to find the best overall solution and, finally, joining neighborhood municipalities by using a blend operator to identify the final solution. We evaluate performance of parallelized approach over standard TSP test problems (TSPLIB) to show that our approach gives a better answer in terms of quality and time rather than the sequential evolutionary SOM.

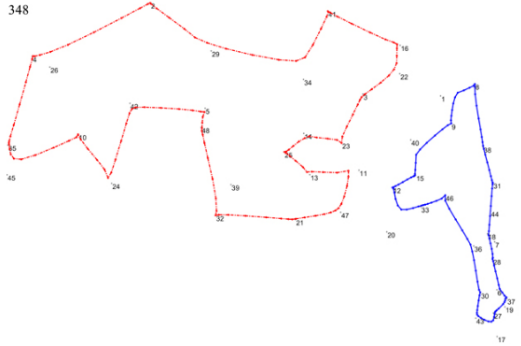
Keywords: Euclidean Traveling Salesman Problem, Artificial Neural Network, Parallelization, Self-organized map, TSPLIB

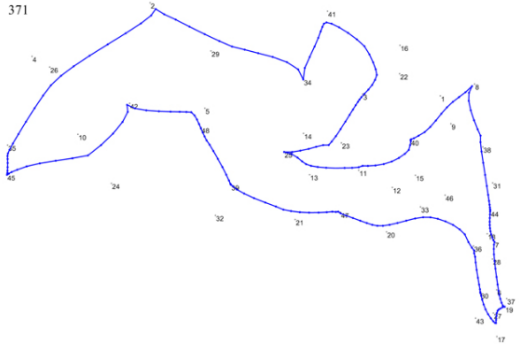
The Traveling ~~Salesman~~ Santa Problem

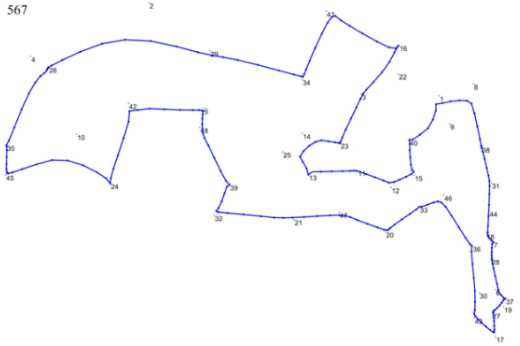


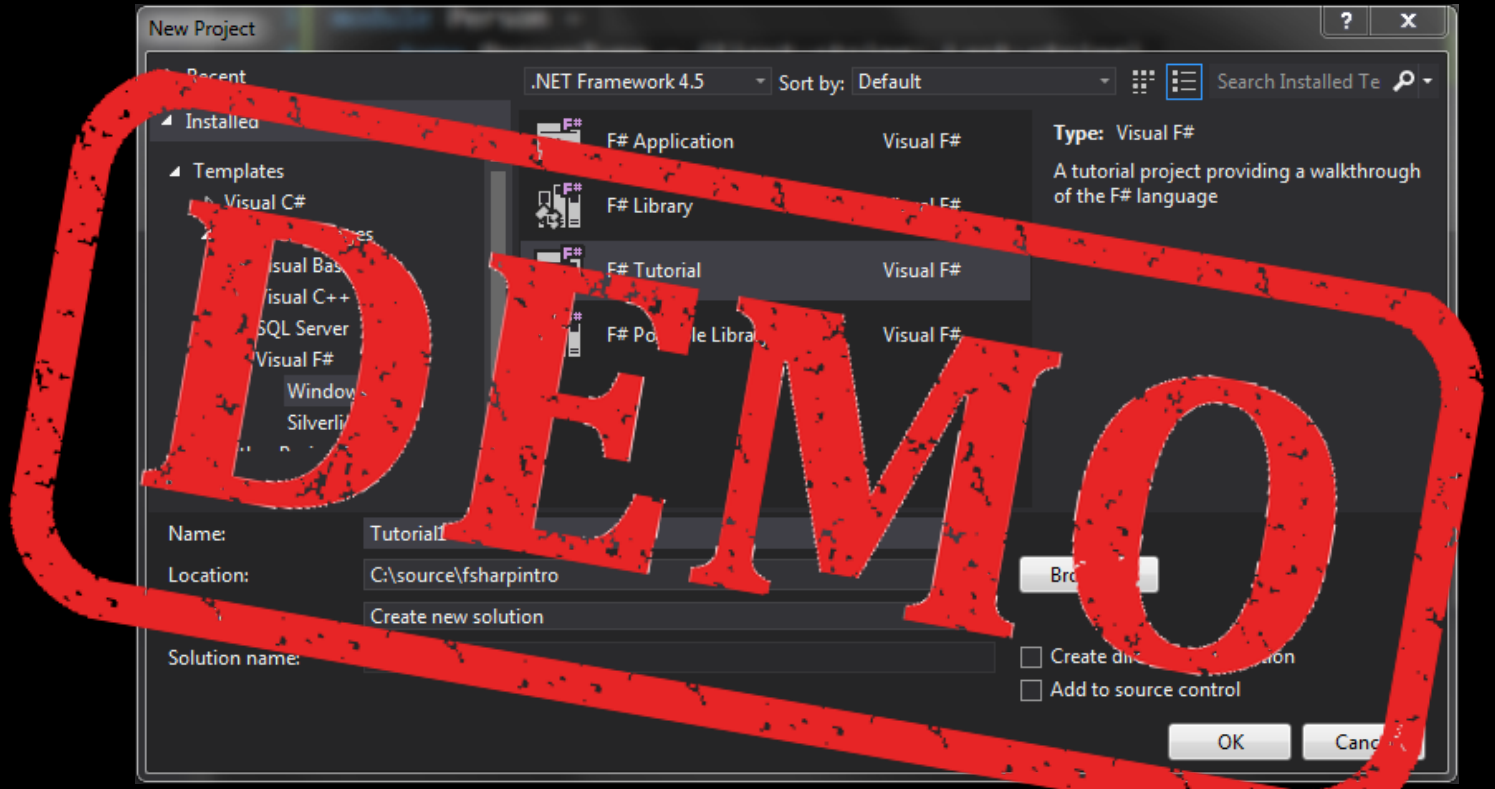











New Project
.NET Framework 4.5
Sort by:
Default
Search Installed Te
Installed
Templates
Visual C#
Visual C++
SQL Server
Visual F#
F# Application
Visual F#
F# Library
F# Tutorial
Visual F#
Type: Visual F#
A tutorial project providing a walkthrough of the F# language
Name:
Location:
C:\source\fsharpintro
Create new solution
Solution name:
Add to source control
OK
DEMO

Summary

NN can be easily parallelized… each domain has possibilities for tailored approaches to increase speed

Neural Networks ease complex problem solving…

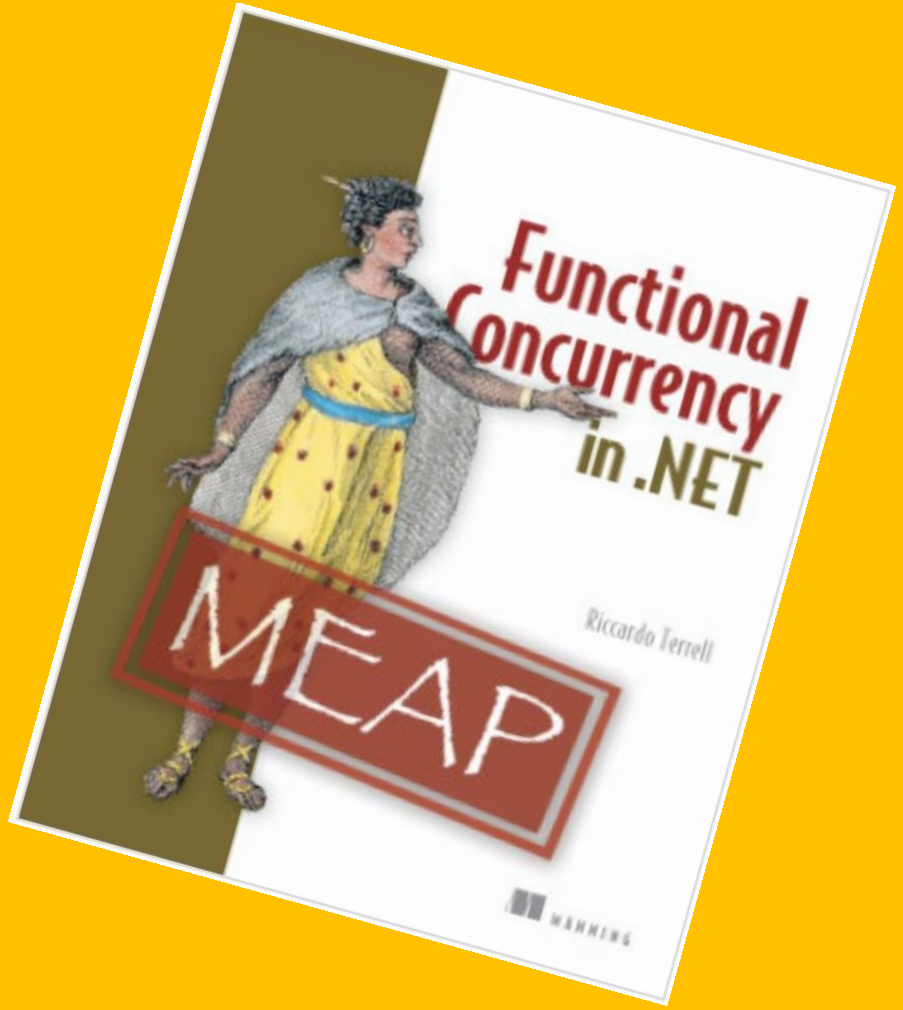
Neural Networks are based on simple linear algebra!

Neural Networks can be parallelized!!

…to be very fast

Resources

- ftp://ftp.sas.com/pub/neural/FAQ.html#questions
- Machine Learning Projects for .NET Developers
- Mastering .NET Machine Learning
- Evelina Gabasova blog
- James McCaffrey blog



Use code coupon

ctwlambda17

for 39% off

https://www.manning.com/books/functional-concurrency-in-dotnet


That's all Folks!

The tools we use have a profound (and devious!) influence on our thinking habits, and, therefore, on our thinking abilities.

-- Edsger Dijkstra

How to reach me

https://github.com/rikace/ml-workshop

meetup.com/DC-fsharp

@DCFsharp @TRikace

tericcardo@gmail.com