That which does not kill us makes us stronger.

Friedrich Nietzsche

Pain is weakness leaving the body

U.S. Marine Corps


salomon

www.lakesinaday.co.uk

LAKES IN A DAY

ULTRA RUN 11TH OCTOBER 2014

CALDBECK TO **CARTMEL**

48 MILES **4000** METRES OF ASCENT

HAGLÖFS

PRESENTED BY open adventure

HOME • DETAILS • HAGLÖFS • ROUTE • KIT • ENTRY LIST • CONTACT US

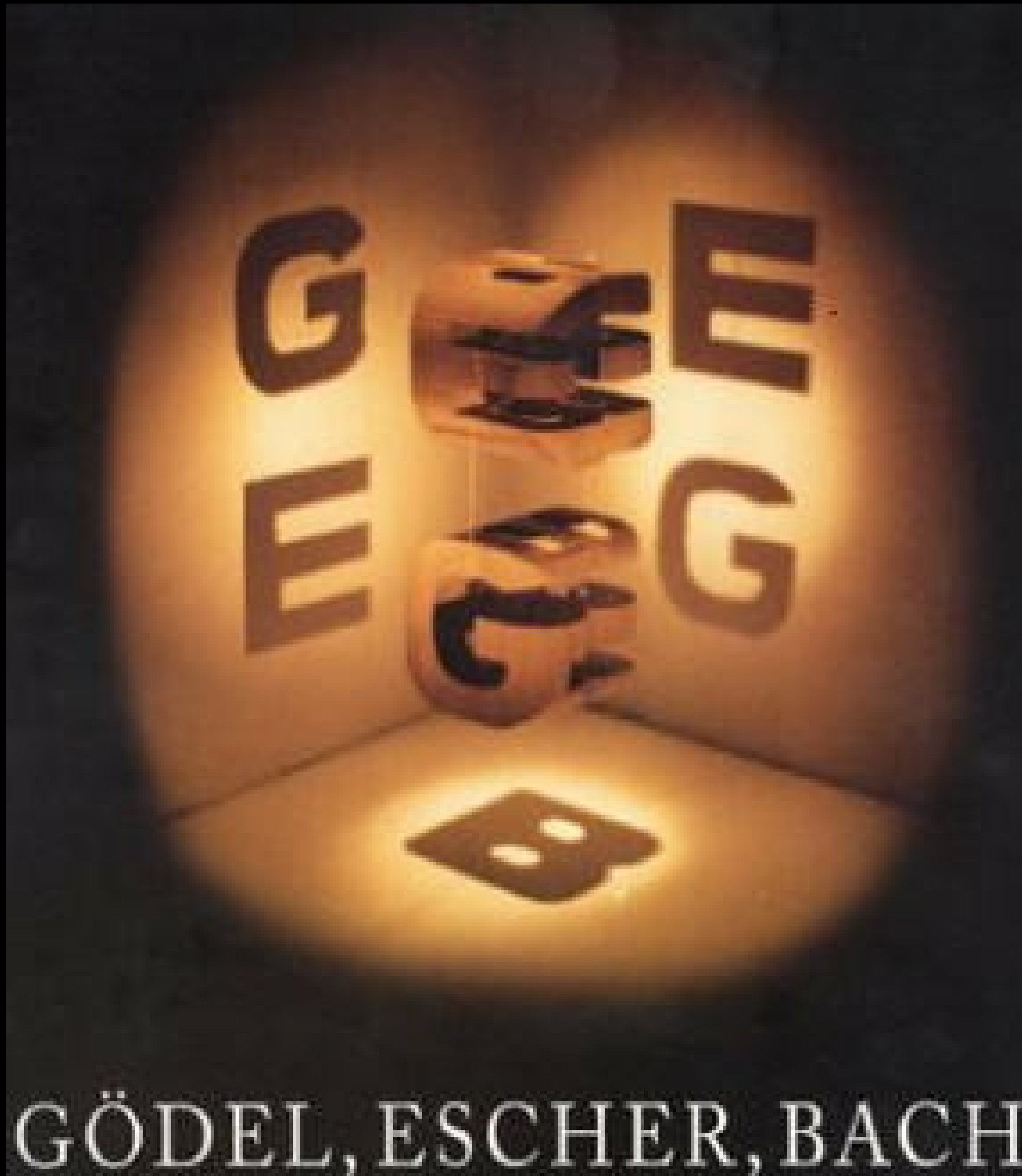
ENTER ONLINE

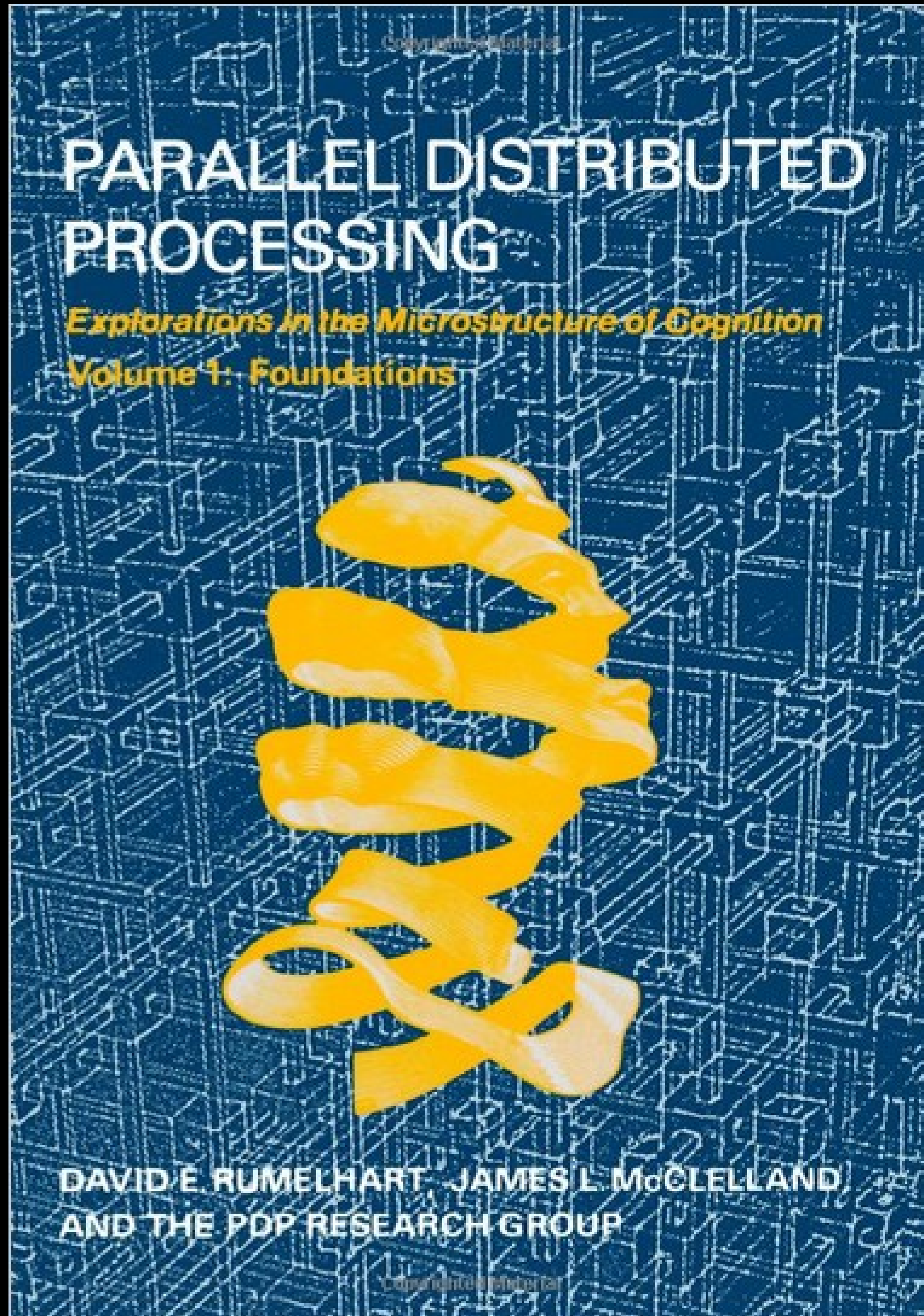

vibram
vibram
Vibram
Vibram

What does that have to do with this talk?

Nothing

Machine Learning


GÖDEL, ESCHER, BACH


PARALLEL DISTRIBUTED
PROCESSING
Explorations in the Microstructure of Cognition
Volume 1: Foundations
DAVID E. RUMELHART, JAMES L. McCLELLAND,
AND THE PDP RESEARCH GROUP

Science Fair Project – shape recognition

Life Plan

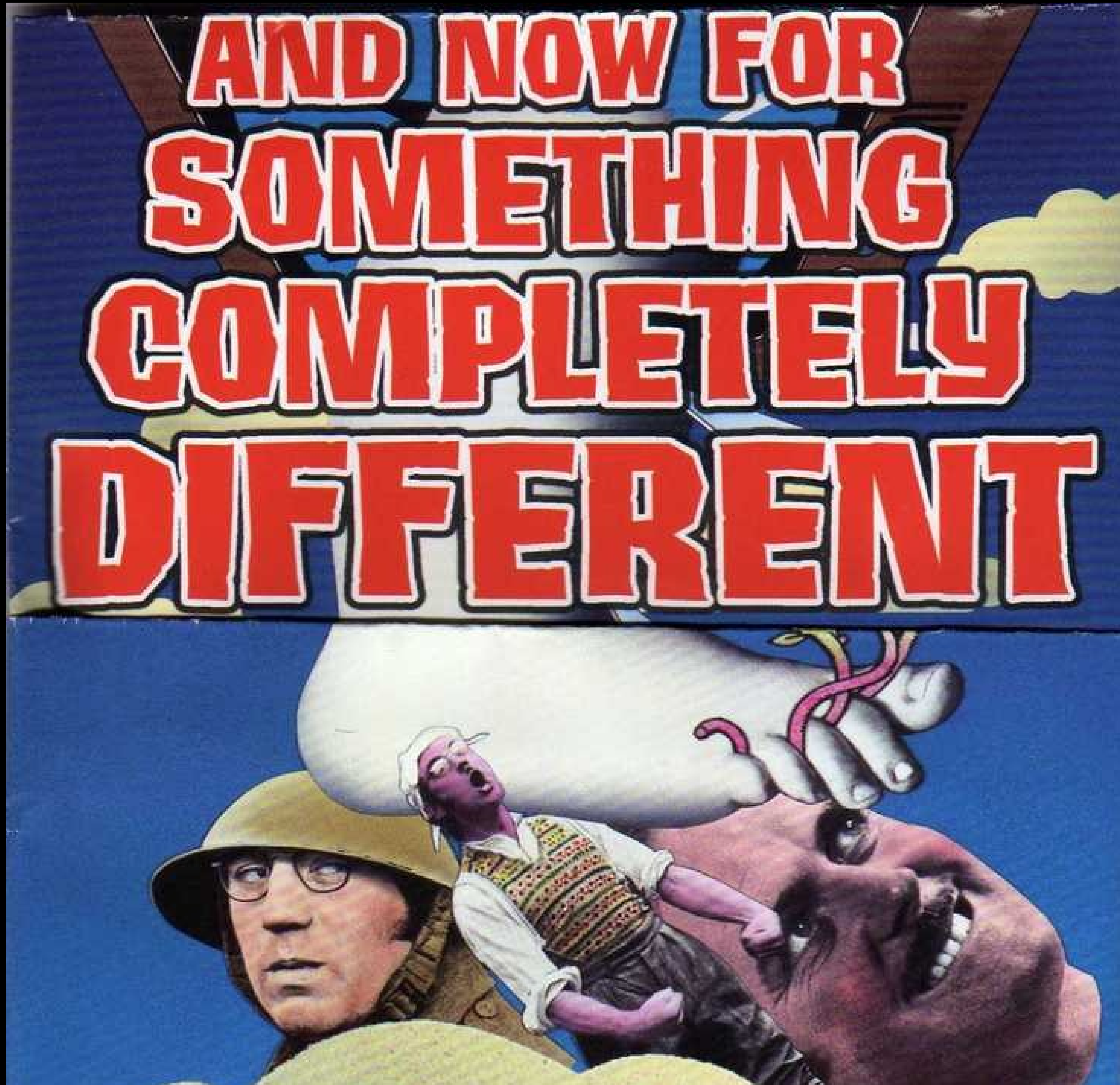
1. Start a company
2. ?
3. Profit

Life Plan

1. Start a company
2. ?
3. Profit
4. Retire to my private island and work on AI

Something must have gone wrong...

VBA :(

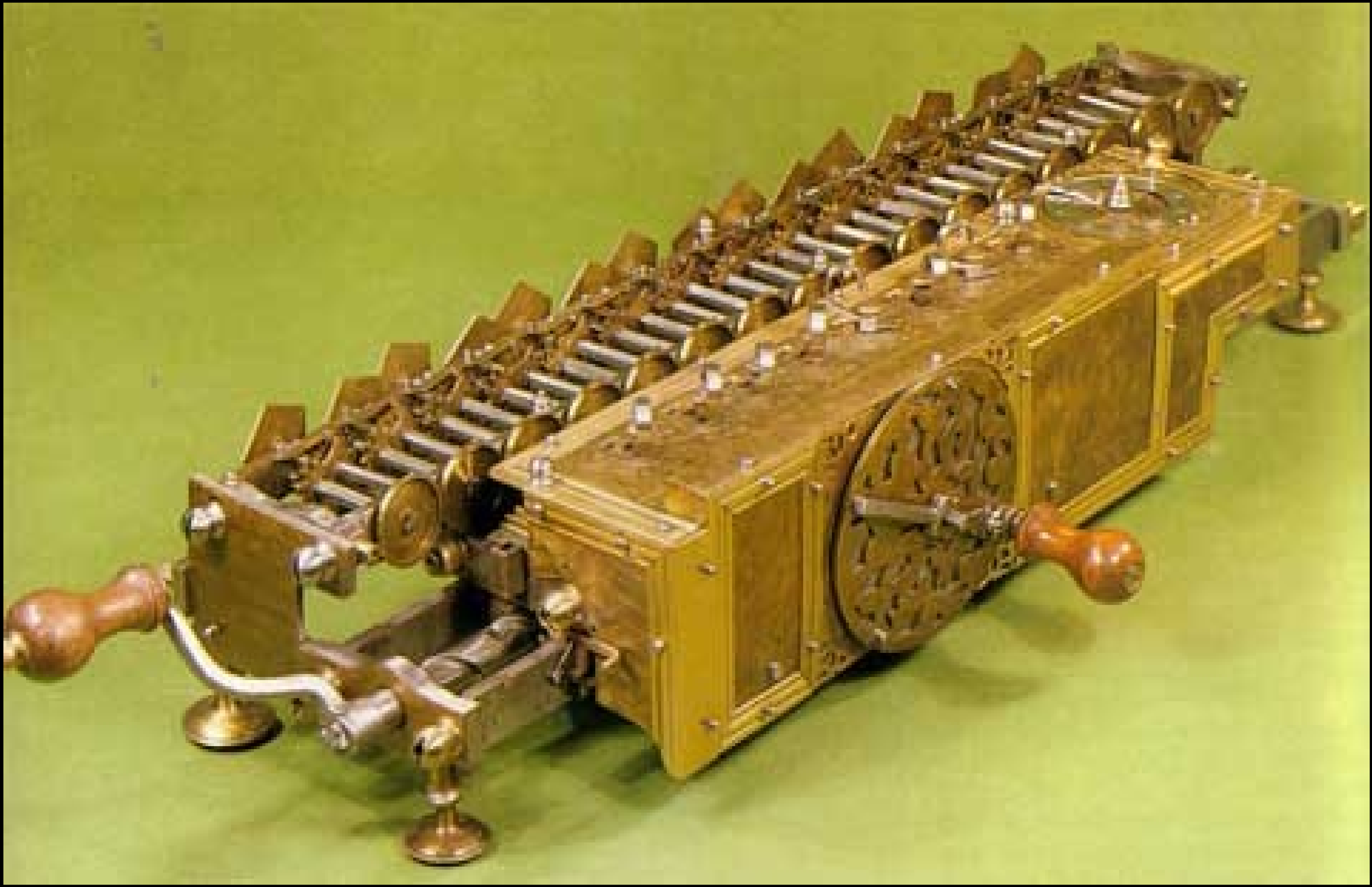

AND NOW FOR
SOMETHING
COMPLETELY
DIFFERENT

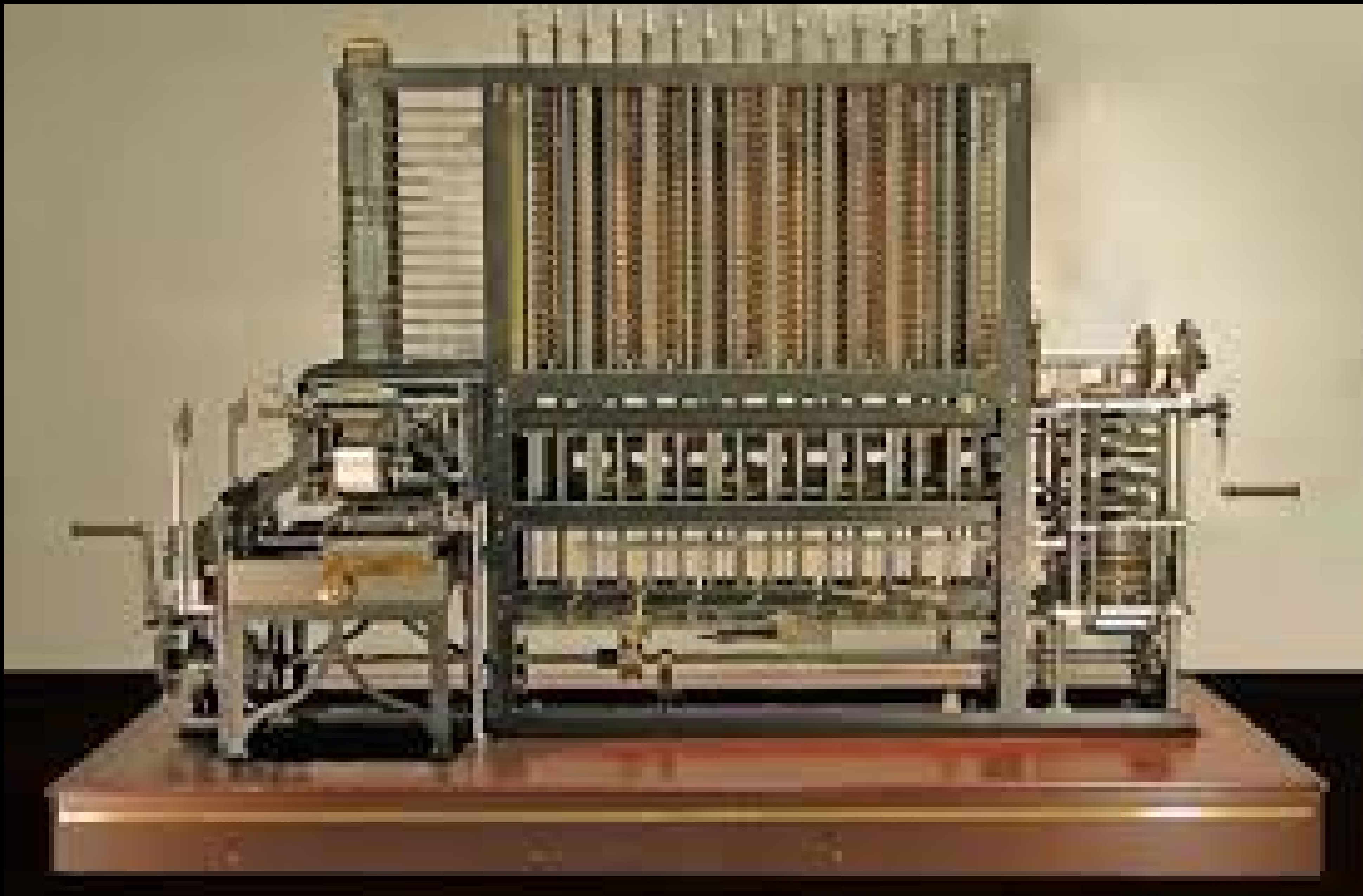
Machine Learning

History

Man has always dreamed of Intelligent Machines


PHONETIC KEMP INC.
UKIYO CAMERA SYSTEMS





VOL. LIX. No. 236.] [October, 1950

MIND

A QUARTERLY REVIEW

OF

PSYCHOLOGY AND PHILOSOPHY

I.—COMPUTING MACHINERY AND INTELLIGENCE

BY A. M. TURING

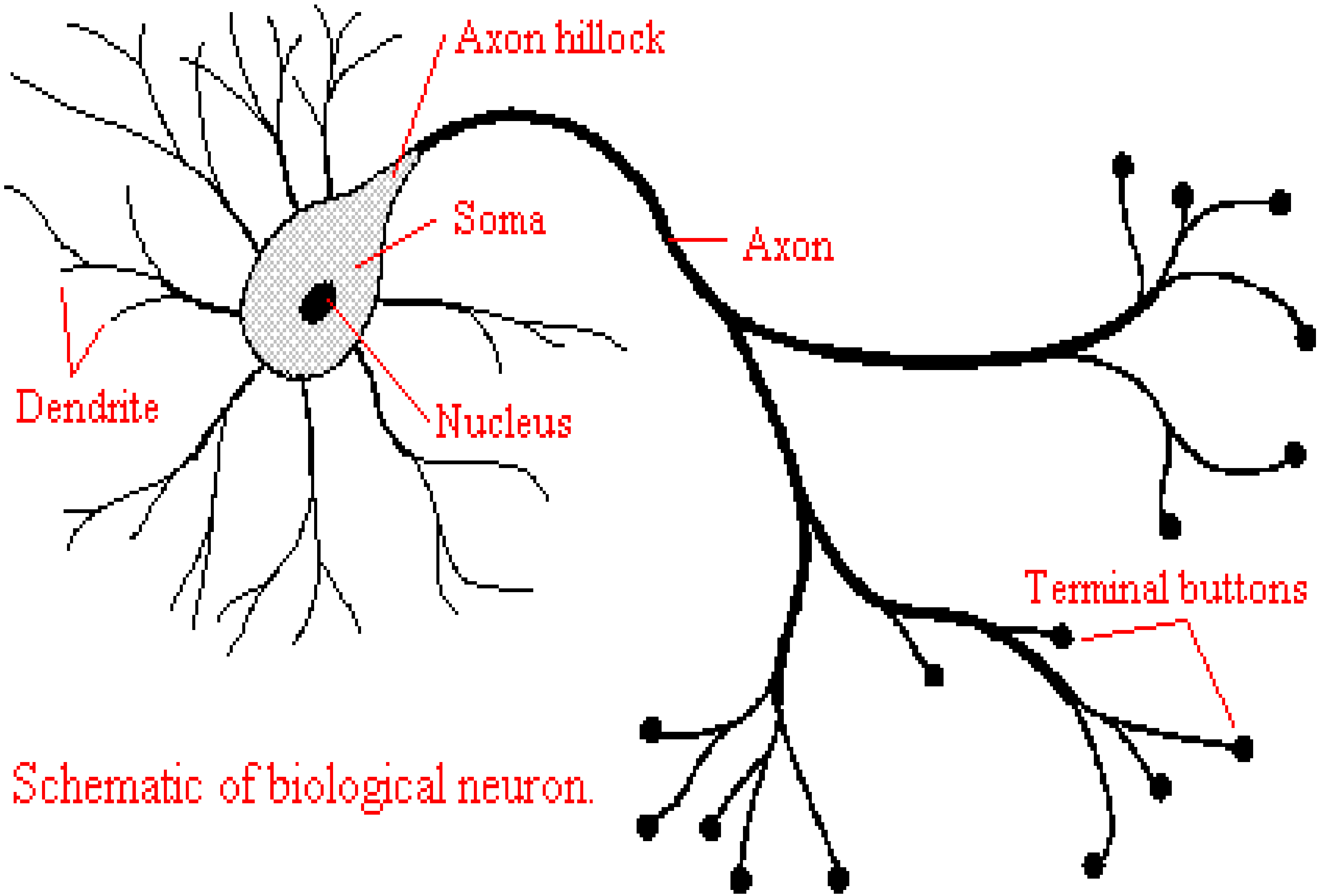
1. *The Imitation Game.*

I PROPOSE to consider the question, 'Can machines think?' This should begin with definitions of the meaning of the terms 'machine' and 'think'. The definitions might be framed so as to reflect so far as possible the normal use of the words, but this attitude is dangerous. If the meaning of the words 'machine' and 'think' are to be found by examining how they are commonly used it is difficult to escape the conclusion that the meaning and the answer to the question, 'Can machines think?' is to be sought in a statistical survey such as a Gallup poll. But this is absurd. Instead of attempting such a definition I shall replace the question by another, which is closely related to it and is expressed

I propose to consider the question, “Can machines think?”

This should begin with definitions of the meaning of the terms “machine” and “think”.

Alan Turing, 1950



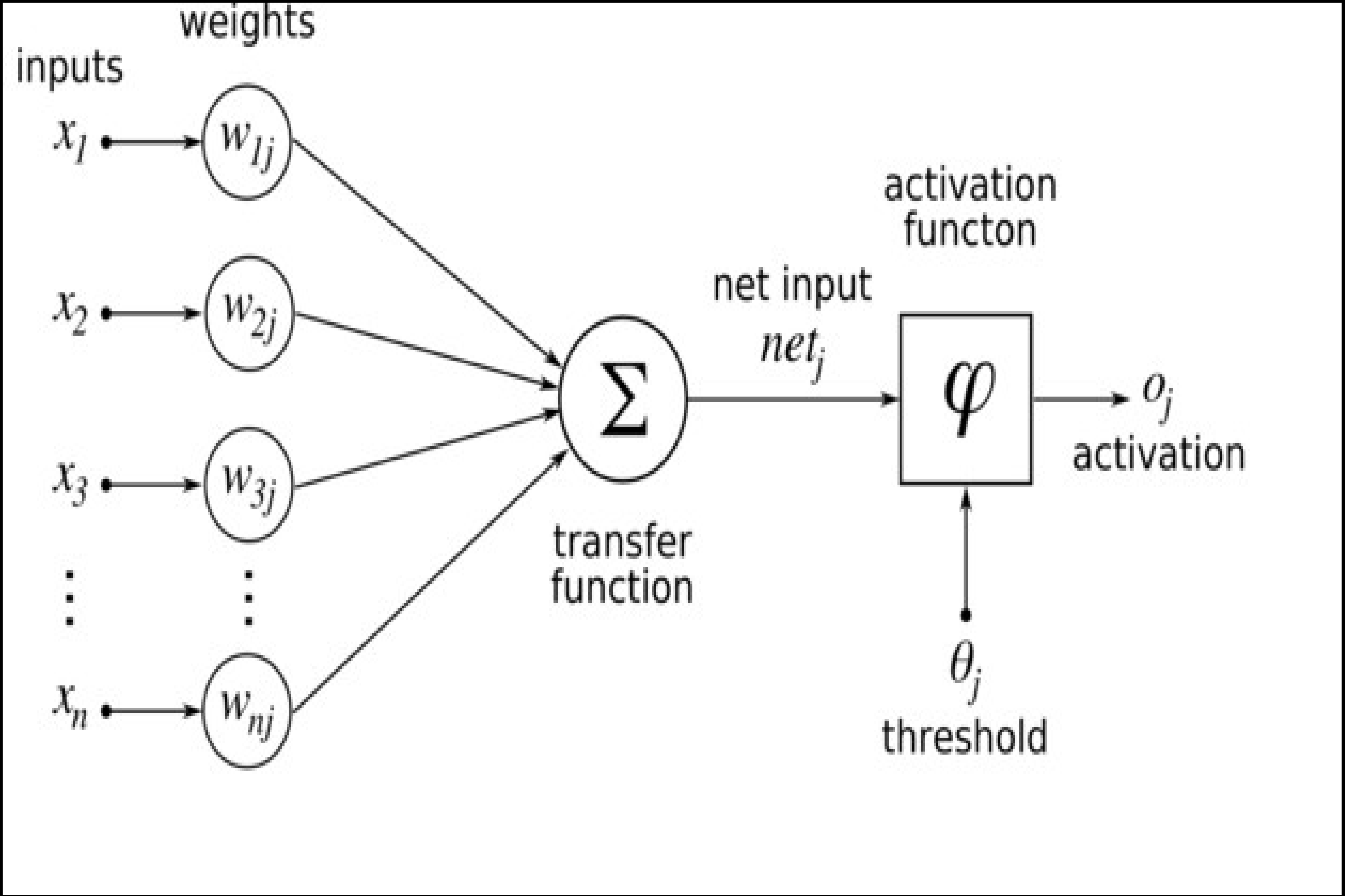
Schematic of biological neuron.

A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY

WARREN S. MCCULLOCH and WALTER H. PITTS

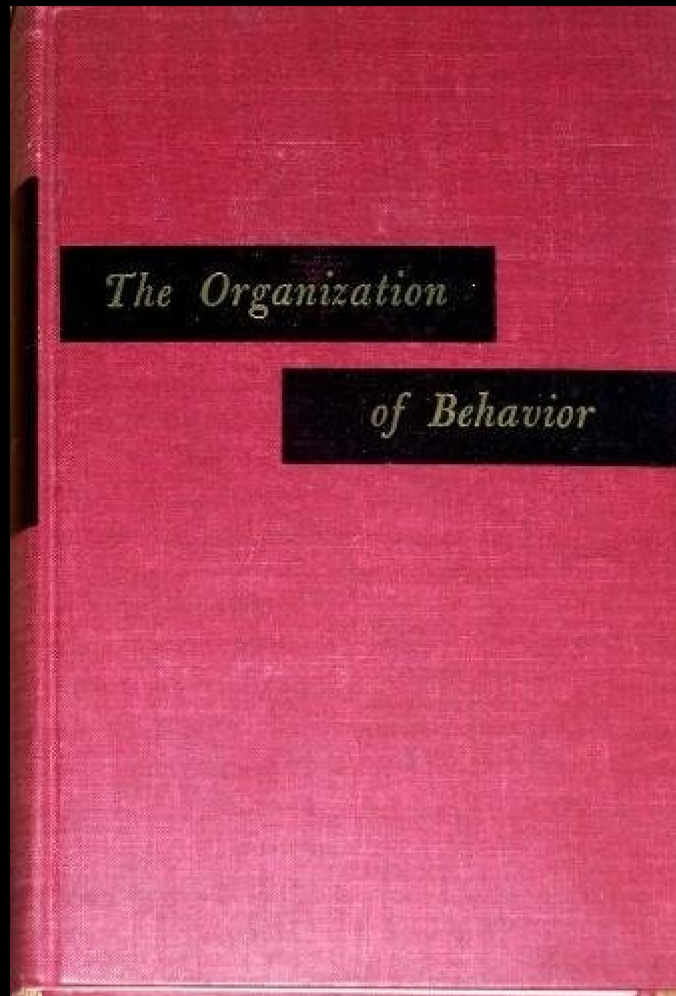
Because of the "all-or-none" character of nervous activity, neural events and the relations among them can be treated by means of propositional logic. It is found that the behavior of every net can be described in these terms, with the addition of more complicated logical means for nets containing circles; and that for any logical expression satisfying certain conditions, one can find a net behaving in the fashion it describes. It is shown that many particular choices among possible neurophysiological assumptions are equivalent, in the sense that for every net behaving under one assumption, there exists another net which behaves under the other and gives the same results, although perhaps not in the same time. Various applications of the calculus are discussed.

McCulloch & Pitts, 1943


weights
inputs
x_1
w_{1j}
x_2
w_{2j}
x_3
w_{3j}
x_n
w_{nj}
Σ
transfer
function
net input
net_j
activation
functon
φ
o_j
activation
θ_j
threshold

Slow, but lots of them

- 200 billion neurons (100-400 billion stars in the galaxy)
- 200,000,000,000,000
- 125 trillion synapses
- 125,000,000,000,000,000,000



Donald Hebb, 1949

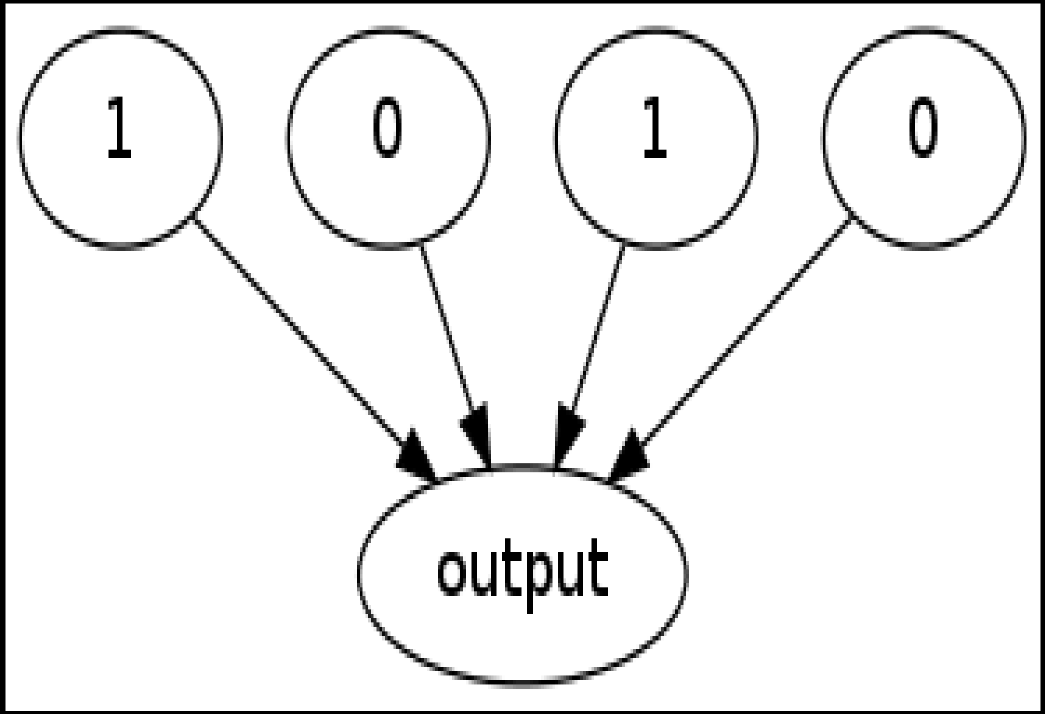
"The general idea is an old one, that any two cells or systems of cells that are repeatedly active at the same time will tend to become 'associated', so that activity in one facilitates activity in the other."

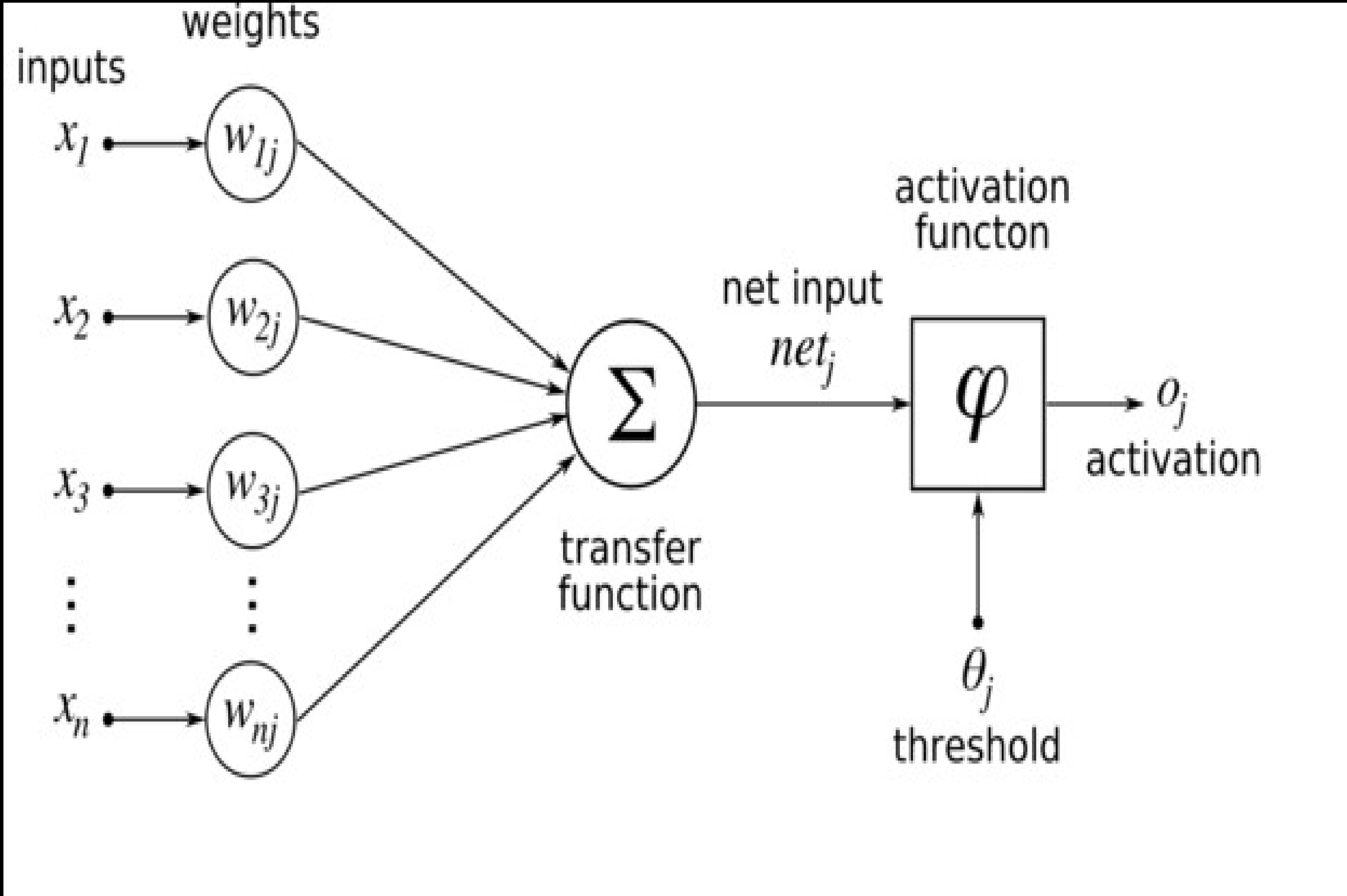
"When one cell repeatedly assists in firing another, the axon of the first cell develops synaptic knobs (or enlarges them if they already exist) in contact with the soma of the second cell."

ML does lots of different things

- Regression
- Clustering
- Classification

Single Layer Perceptrons




weights
inputs
x_1
w_{1j}
x_2
w_{2j}
x_3
w_{3j}
x_n
w_{nj}
activation
functon
net input
net_j
Σ
φ
o_j
activation
transfer
function
θ_j
threshold

AND		
In		*Out*
F	F	F
F	T	F
T	F	F
T	T	T

OR		
In		*Out*
F	F	F
F	T	T
T	F	T
T	T	T

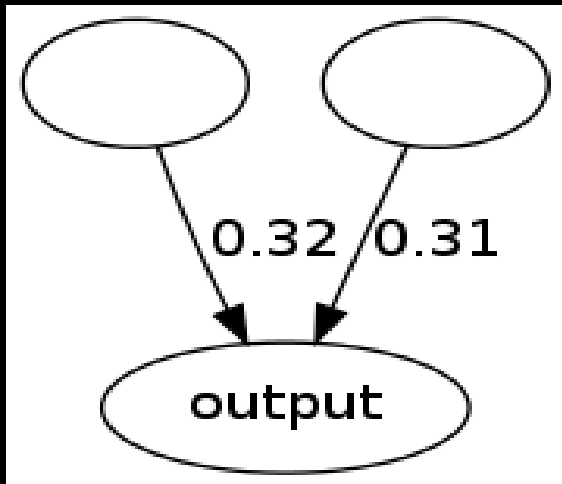
XOR		
In		*Out*
F	F	F
F	T	T
T	F	T
T	T	F

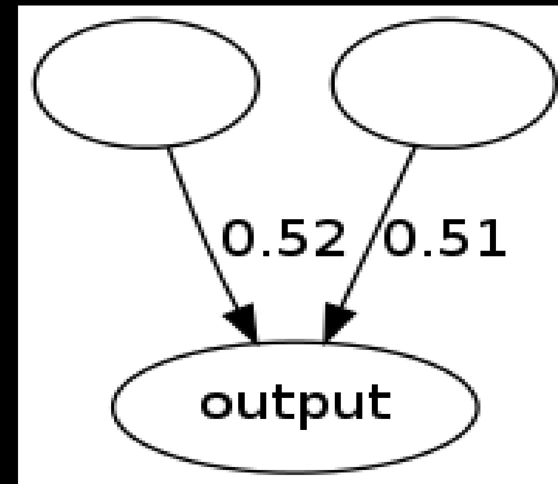
```python
class ActivationFunction (object):
    def __init__ (self, threshold = 0.5):
        self.threshold = threshold

    def __call__ (self, x):
        return 1 if x > 0.5 else 0


class SingleLayerPerceptron (object):
    def __init__ (self, n_inputs, activation_function = ActivationFunction ()):
        self.weights = np.random.rand (n_inputs)
        self.activation_function = activation_function

    def evaluate (self, inputs):
        assert len (inputs) == len (self.weights)

        sum_inputs = sum (inputs * self.weights)
        return self.activation_function (sum_inputs)
```

```python
def train (self, input_set, target_set, learning_rate = 0.1):
    assert len (input_set) == len (target_set)

    sum_squared_error = 0.0
    for inputs, target in zip (input_set, target_set):
        estimate = self.evaluate (inputs)
        error = target - estimate

        for i in range (len (inputs)):
            self.weights [i] += learning_rate * error * inputs [i]

        sum_squared_error += error ** 2

    return sum_squared_error
```

```python
inputs = [
    [ 0, 0],
    [ 0, 1],
    [ 1, 0],
    [ 1, 1],
    ]
outputs = [
        0,
        1,
        1,
        1
    ]

p = SingleLayerPerceptron (len (inputs [0]))
for i in range (50):
    err = p.train (inputs, outputs)
    print "SSQ = %.6f " % err

    for inp, output in zip (inputs, outputs):
        estimate = p.evaluate (inp)
        print inp, " => ", estimate, " expected: ", output
```

```
[0, 0]  =>  0  expected:  0
[0, 1]  =>  0  expected:  1
[1, 0]  =>  0  expected:  1
[1, 1]  =>  1  expected:  1
SSQ = 2.000000
[0, 0]  =>  0  expected:  0
[0, 1]  =>  1  expected:  1
[1, 0]  =>  1  expected:  1
[1, 1]  =>  1  expected:  1
SSQ = 0.000000
```





		Initial Weights	
		0.32	0.31
Inputs		**Weighted input**	**Activation**
0	0	0.00	0
0	1	0.31	0
1	0	0.32	0
1	1	0.63	1

		Trained Weights	
		0.52	0.51
Inputs		**Weighted input**	**Activation**
0	0	0.00	0
0	1	0.51	1
1	0	0.52	1
1	1	1.03	1

Remember XOR?

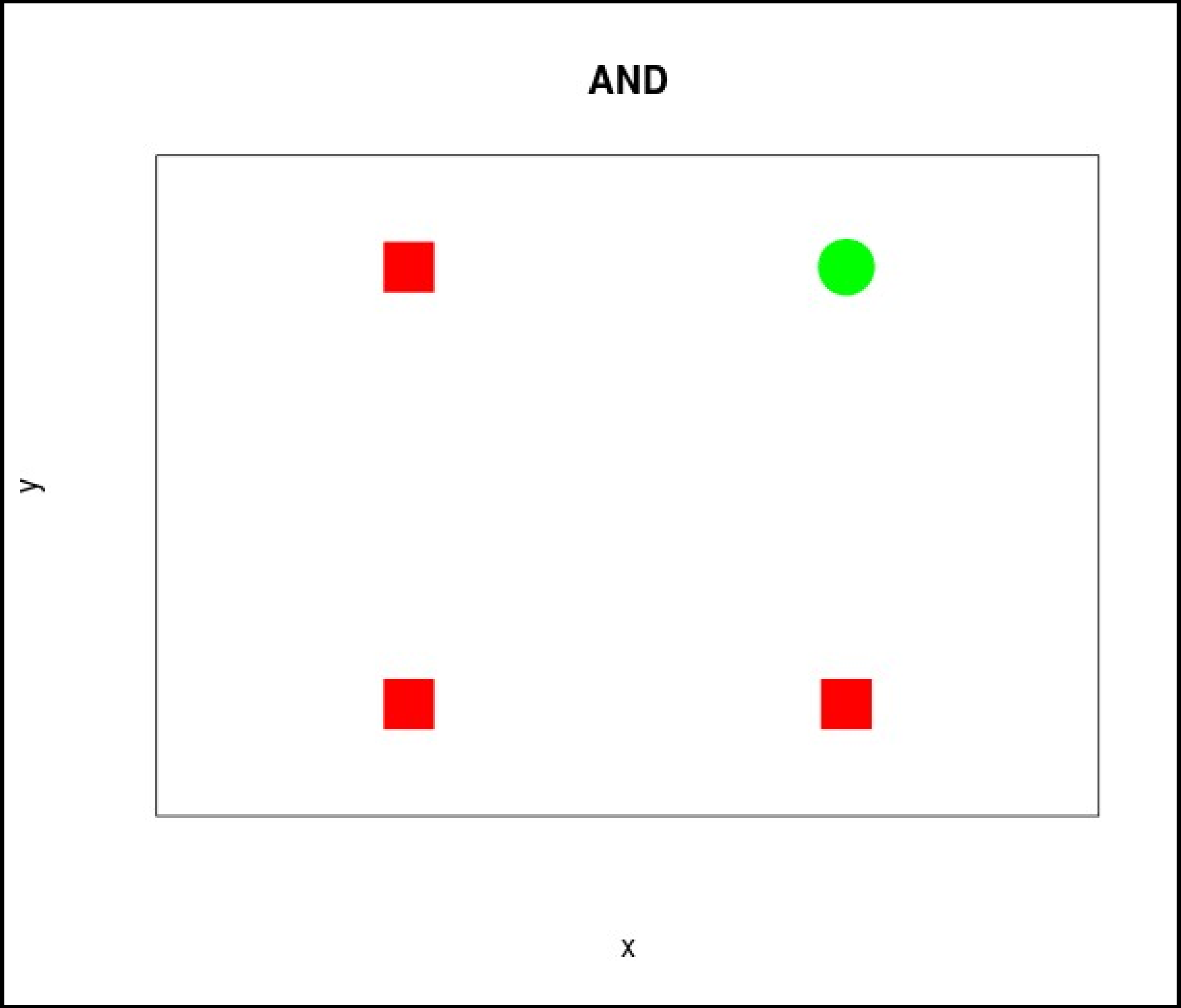
XOR is special

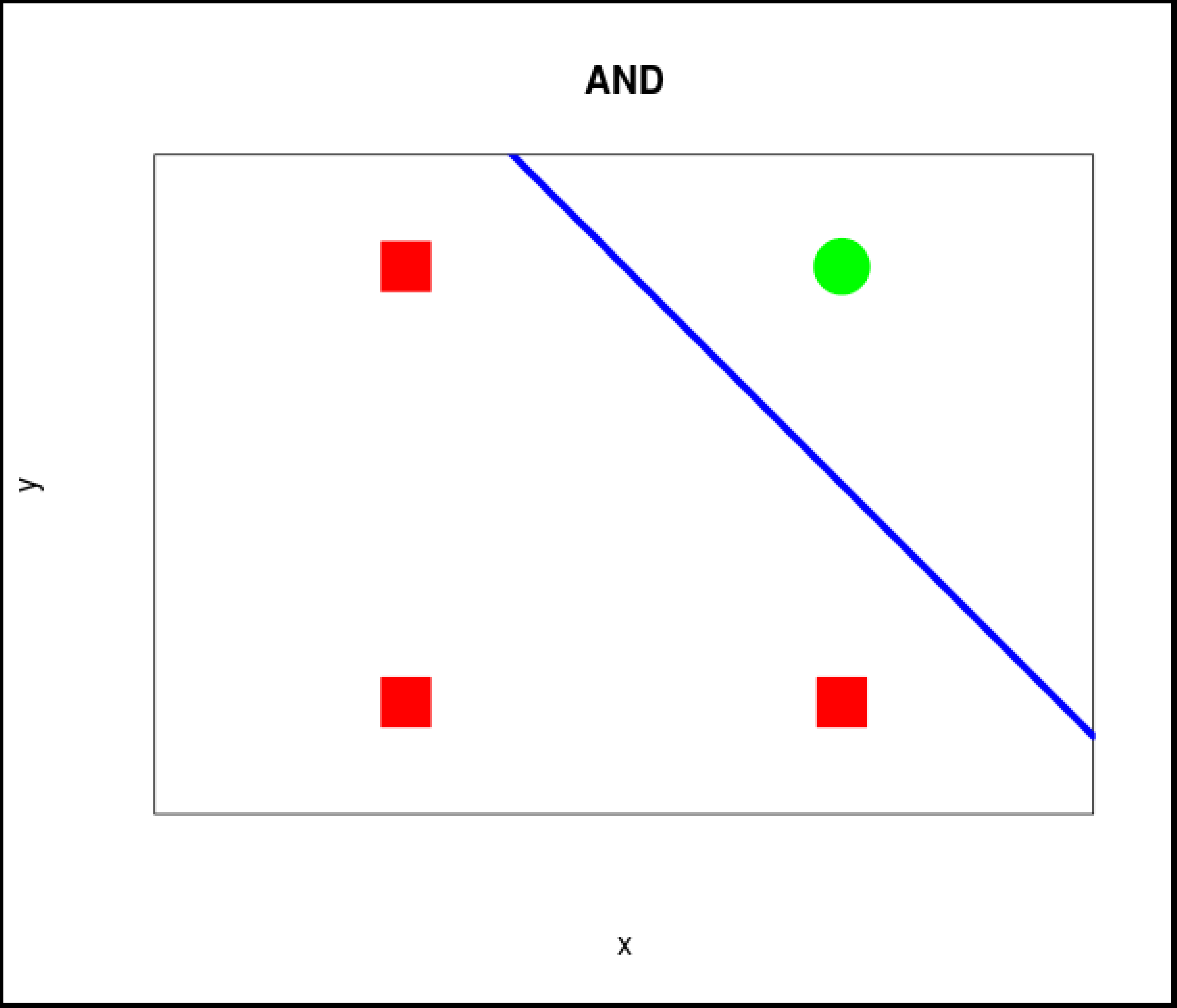
```
inputs = [
    [ 0, 0],
    [ 0, 1],
    [ 1, 0],
    [ 1, 1]
    ]
outputs = [
        0,
        1,
        1,
        0
    ]

p = SingleLayerPerceptron (len (inputs [0]))
for i in range (100):
    err = p.train (inputs, outputs)
    print "%d,%.6f " % (i,err)
```

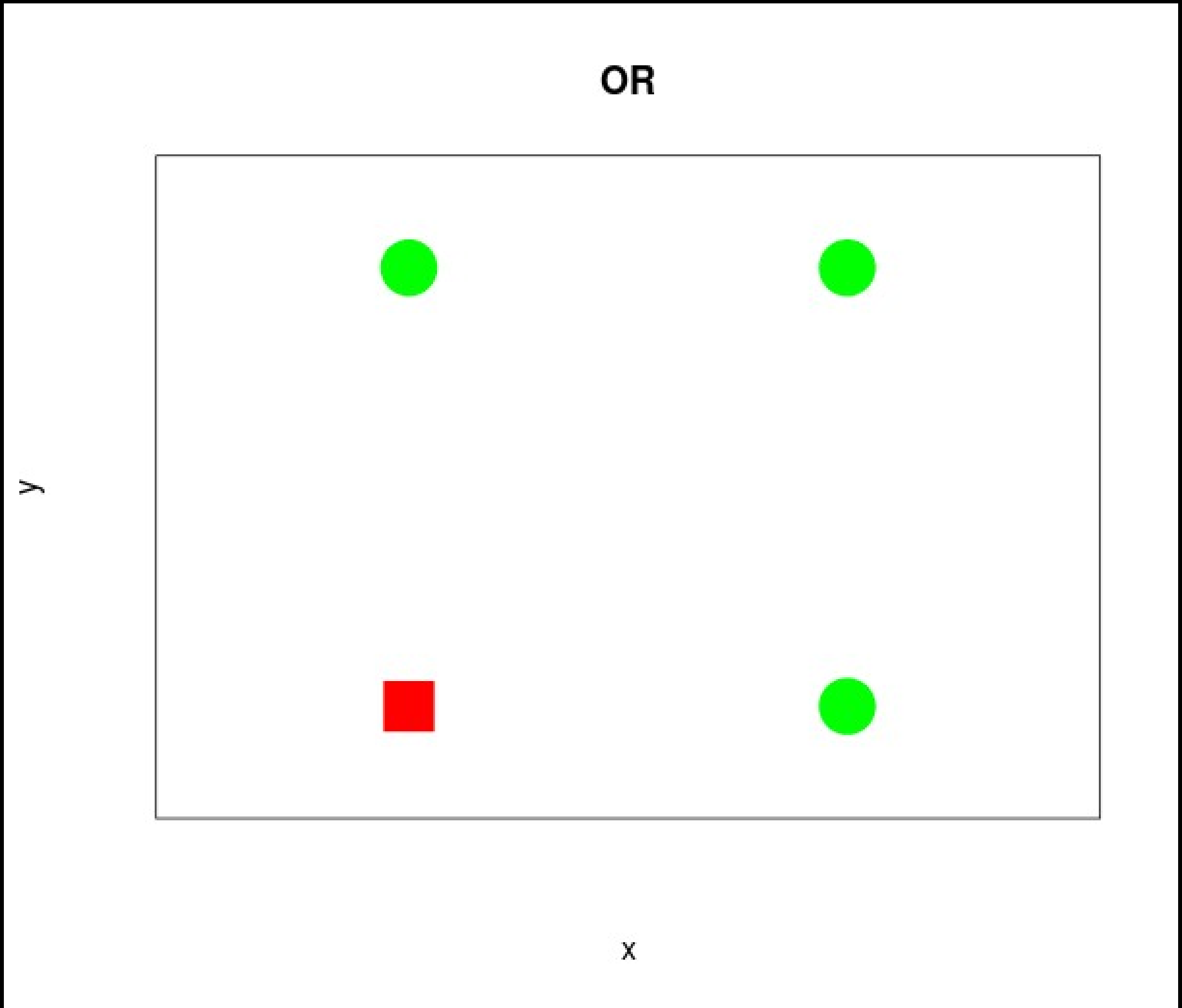
```
0,1.000000
1,1.000000
2,1.000000
3,1.000000
4,1.000000
5,3.000000
6,3.000000
7,3.000000
8,3.000000
9,3.000000
10,3.000000
11,3.000000
12,3.000000
13,3.000000
14,3.000000
15,3.000000
16,3.000000
17,3.000000
18,3.000000
19,3.000000
20,3.000000
21,3.000000
```

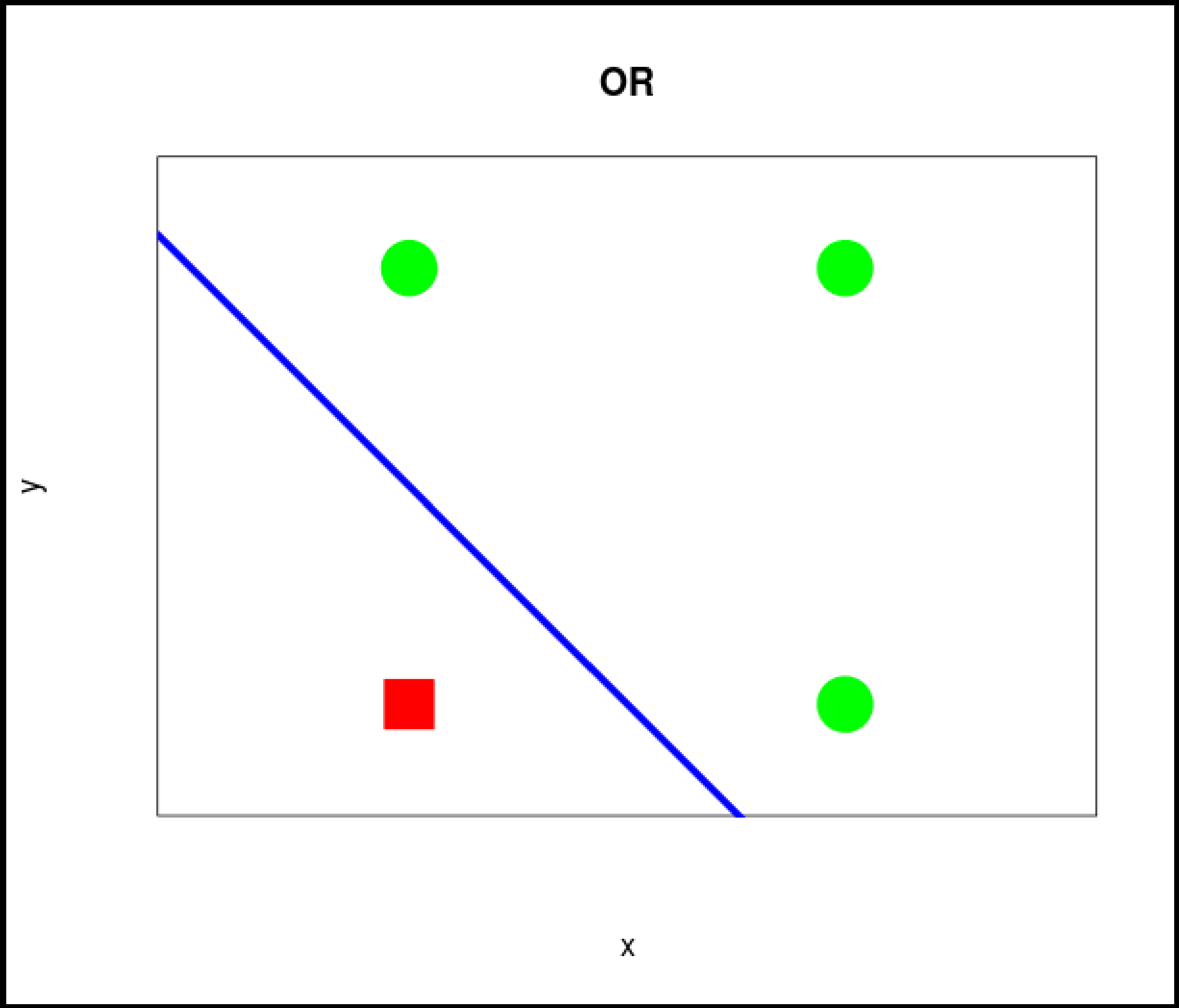
```
0,1.000000
1,1.000000
2,1.000000
3,1.000000
4,1.000000
5,3.000000
6,3.000000
7,3.000000
8,3.000000
9,3.000000
10,3.000000
11,3.000000
12,3.000000
13,3.000000
14,3.000000
15,3.000000
16,3.000000
17,3.000000
18,3.000000
19,3.000000
20,3.000000
21,3.000000
```

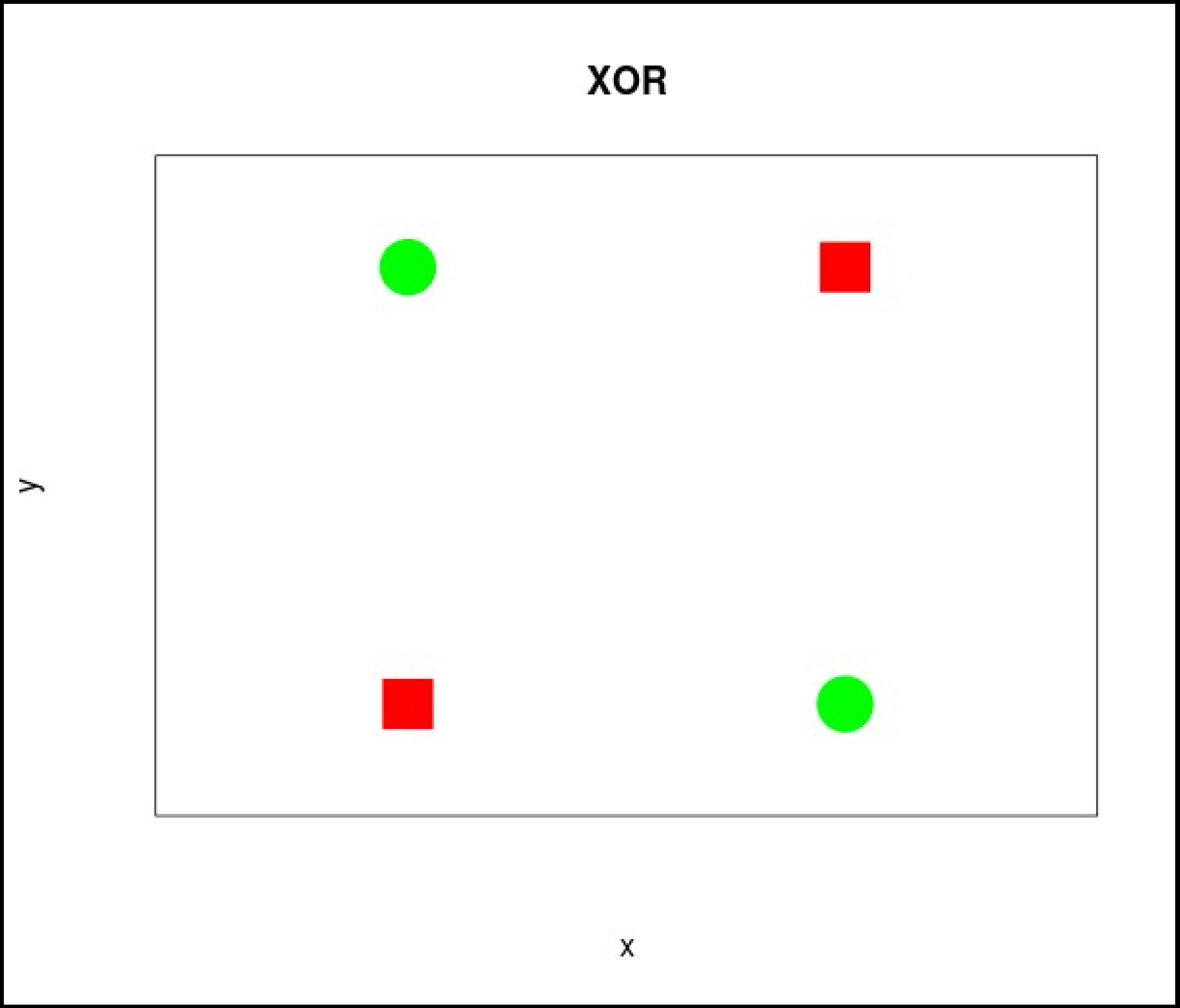
```
79,3.000000
80,3.000000
81,3.000000
82,3.000000
83,3.000000
84,3.000000
85,3.000000
86,3.000000
87,3.000000
88,3.000000
89,3.000000
90,3.000000
91,3.000000
92,3.000000
93,3.000000
94,3.000000
95,3.000000
96,3.000000
97,3.000000
98,3.000000
99,3.000000
```

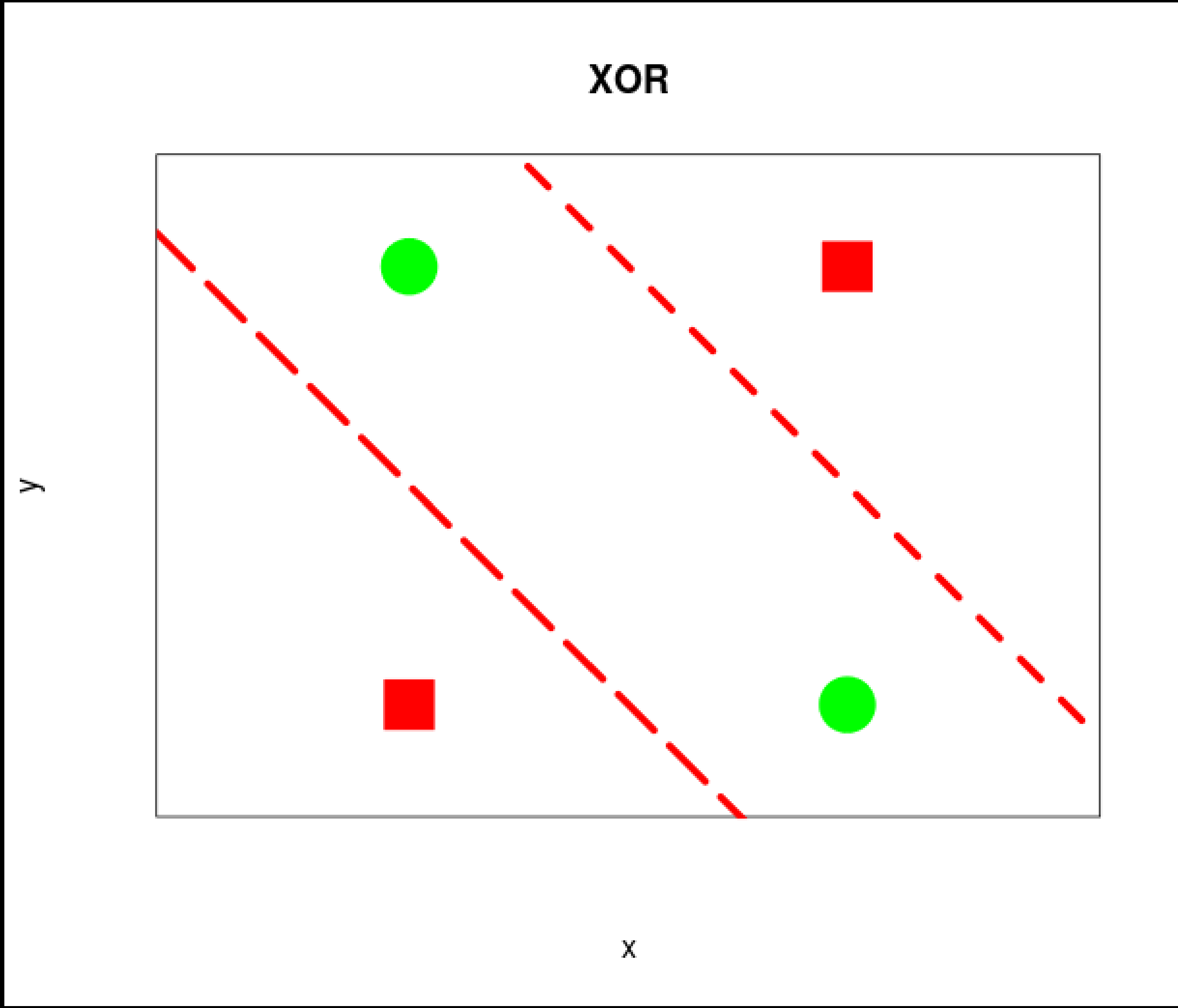

AND
y
x


AND
y
x


OR
y
x

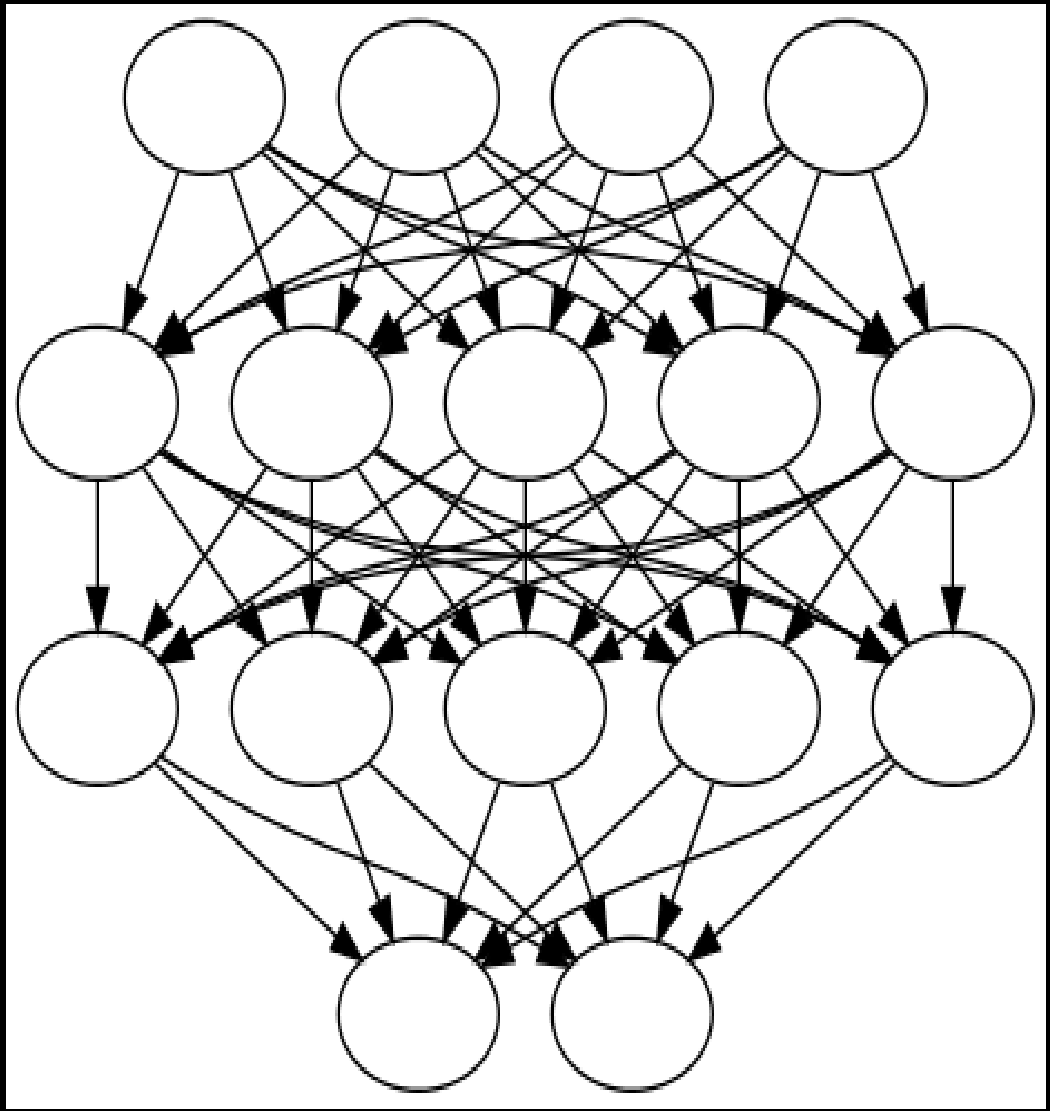

OR
y
x


XOR
y
x


XOR
y
x

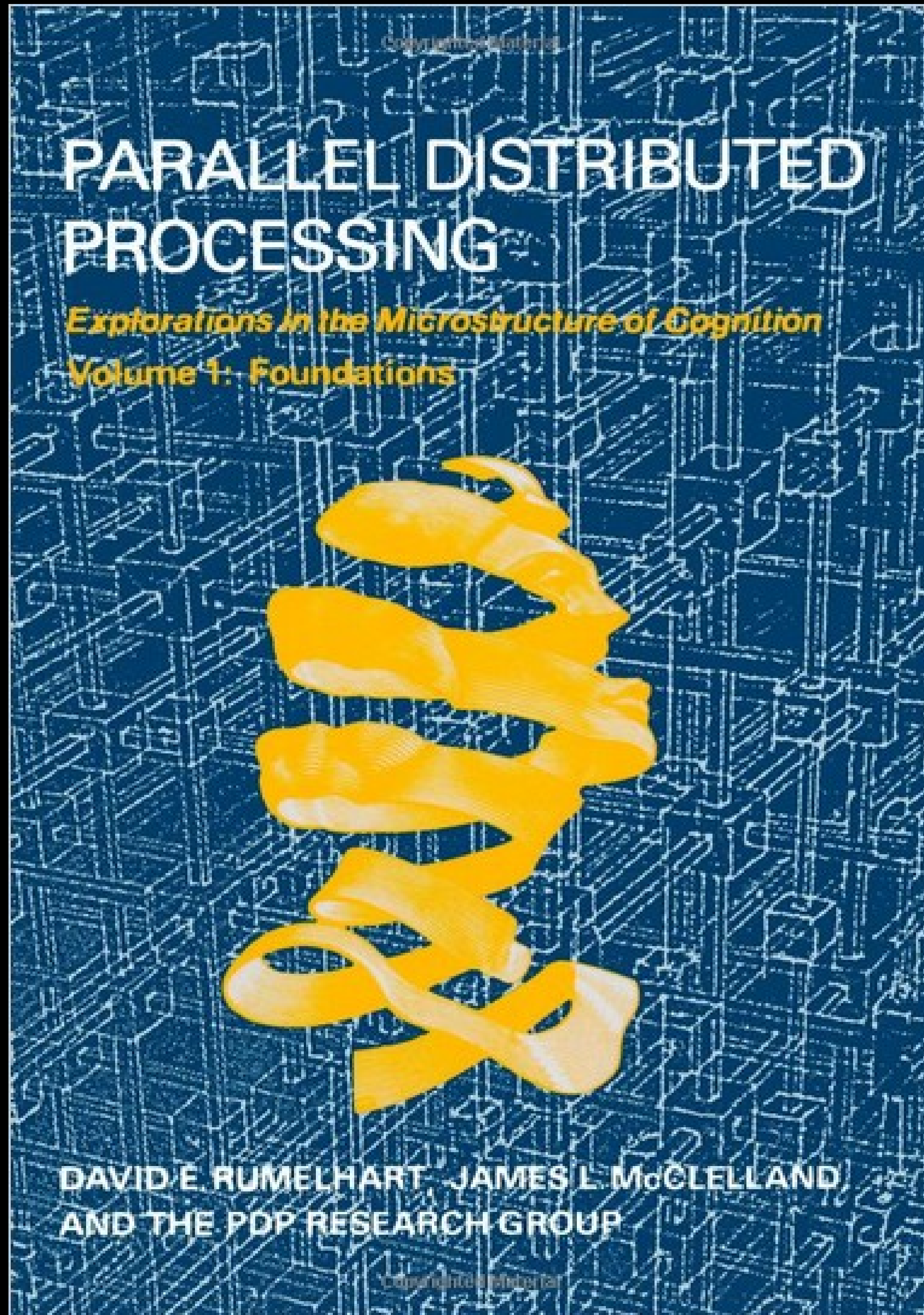
Multi Layer Perceptrons

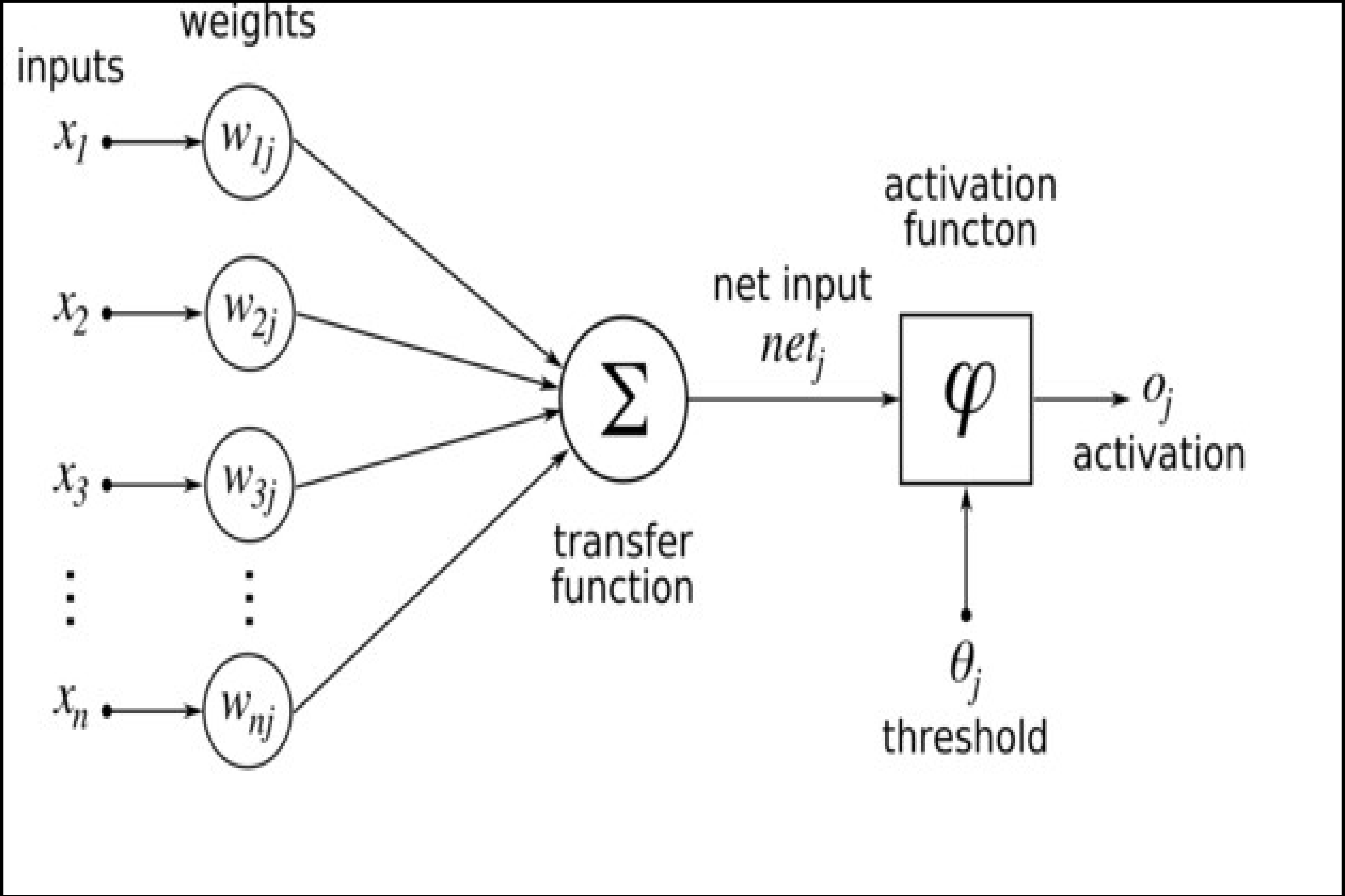
Multi Layer Perceptrons



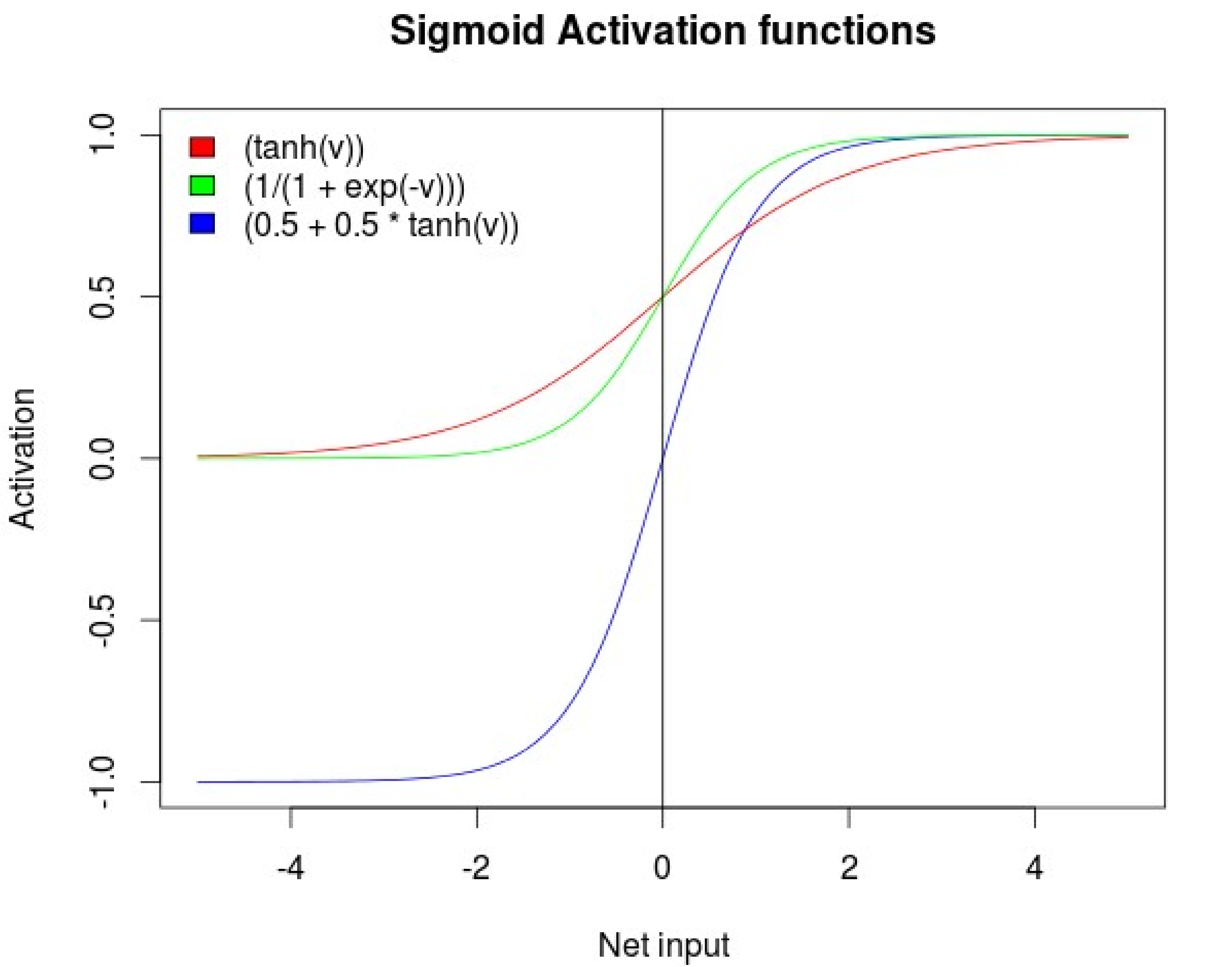
Multi Layer Perceptrons

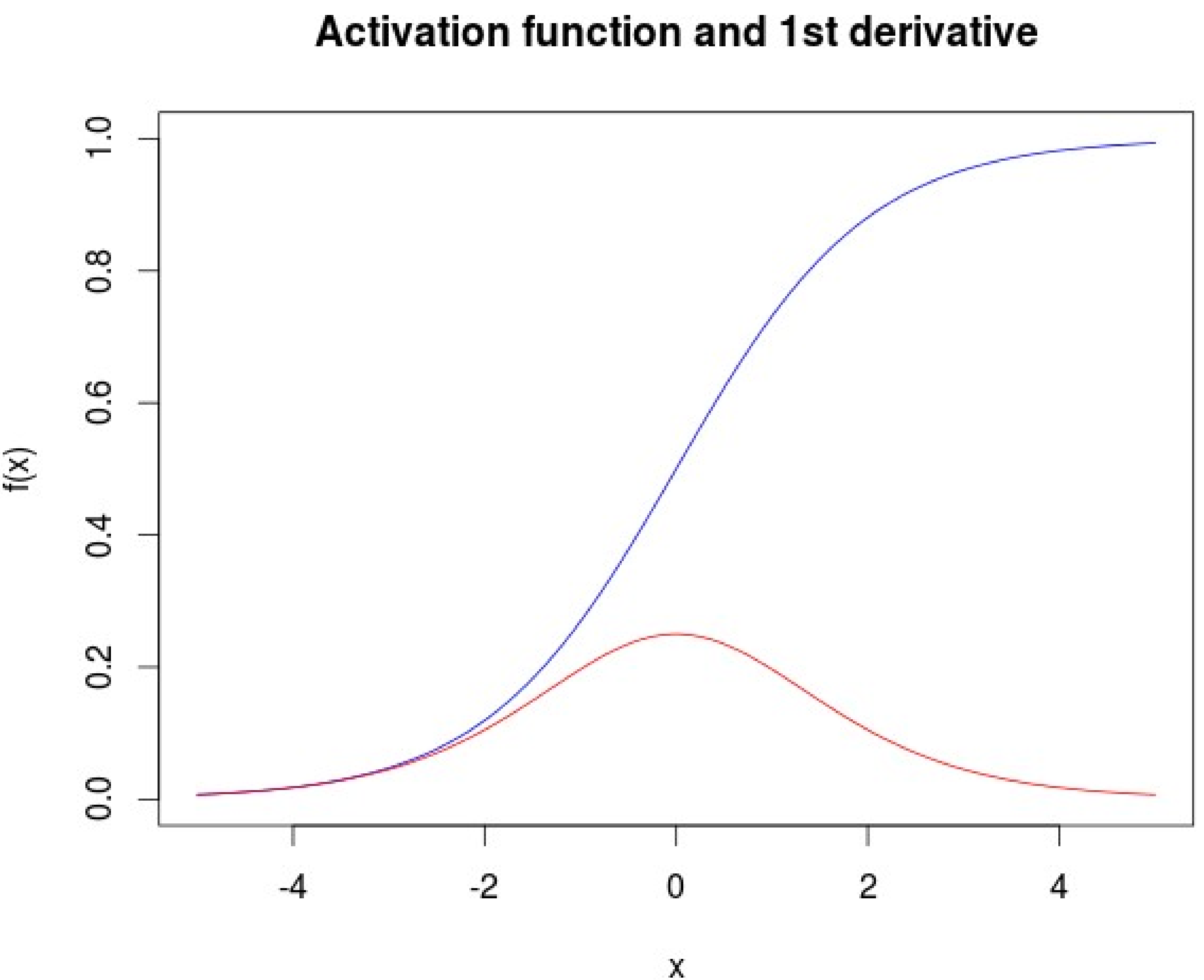
$$\Delta w_{ij} = ???$$

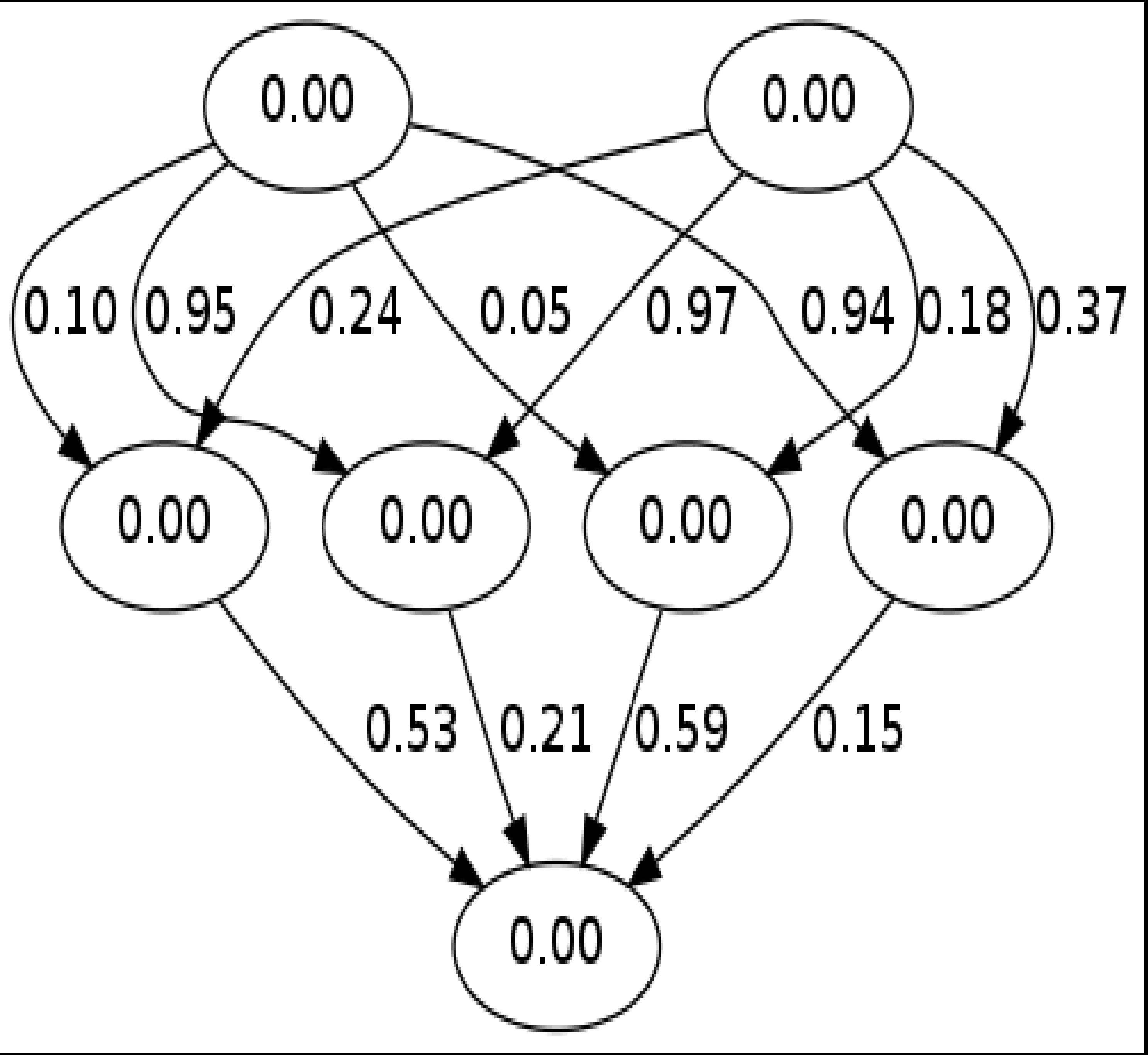

PARALLEL DISTRIBUTED PROCESSING
Explorations in the Microstructure of Cognition
Volume 1: Foundations
DAVID E. RUMELHART, JAMES L. McCLELLAND,
AND THE PDP RESEARCH GROUP



weights
inputs
x_1
x_2
x_3
x_n
w_{1j}
w_{2j}
w_{3j}
w_{nj}
Σ
transfer function
net input
net_j
activation functon
φ
o_j
activation
θ_j
threshold


Sigmoid Activation functions
(tanh(v))
(1/(1 + exp(-v)))
(0.5 + 0.5 * tanh(v))
Activation
1.0
0.5
0.0
-0.5
-1.0
-4
-2
0
2
4
Net input

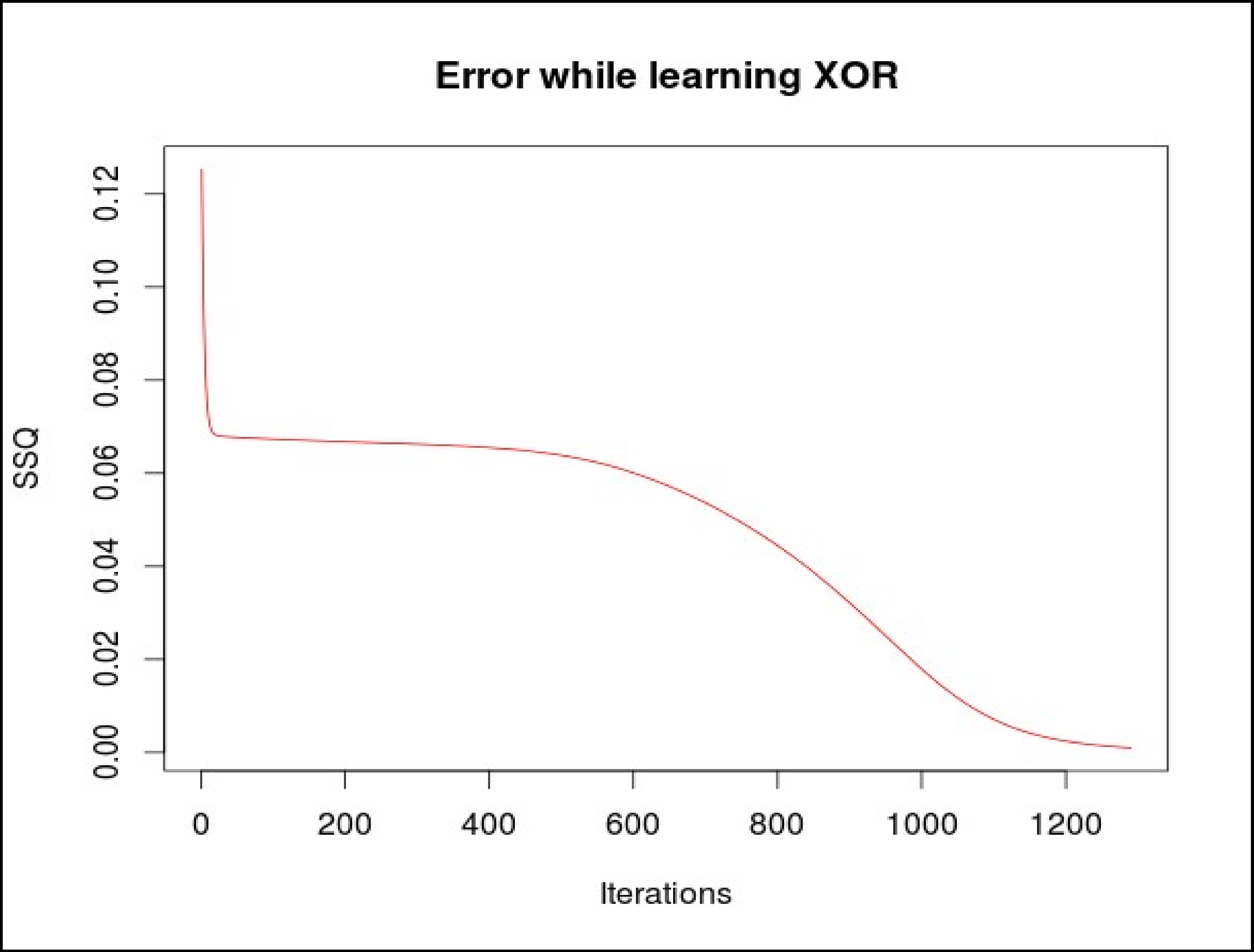

Activation function and 1st derivative
f(x)
0.0
0.2
0.4
0.6
0.8
1.0
-4
-2
0
2
4
x



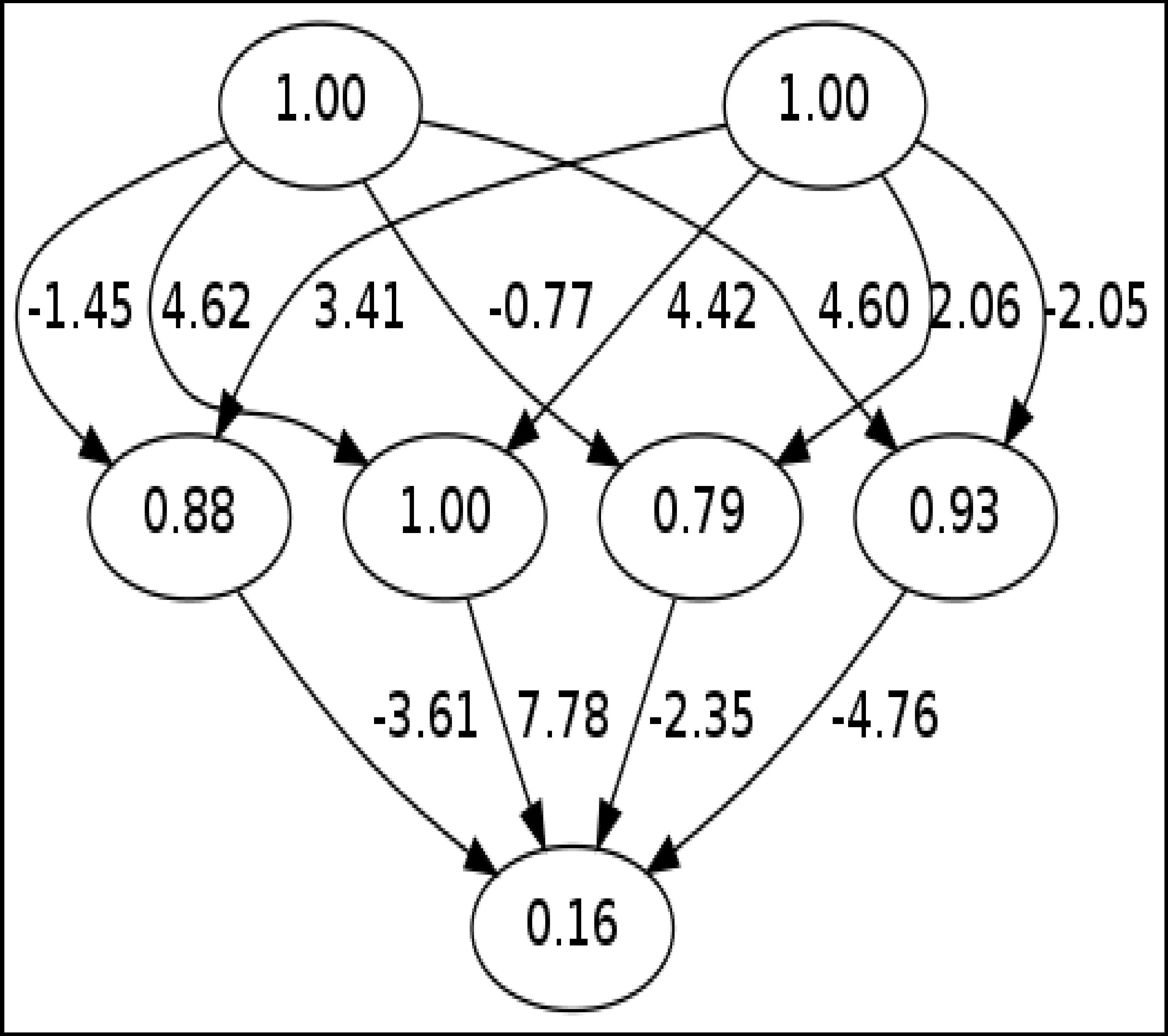
0.00
0.00
0.10
0.95
0.24
0.05
0.97
0.94
0.18
0.37
0.00
0.00
0.00
0.00
0.53
0.21
0.59
0.15
0.00

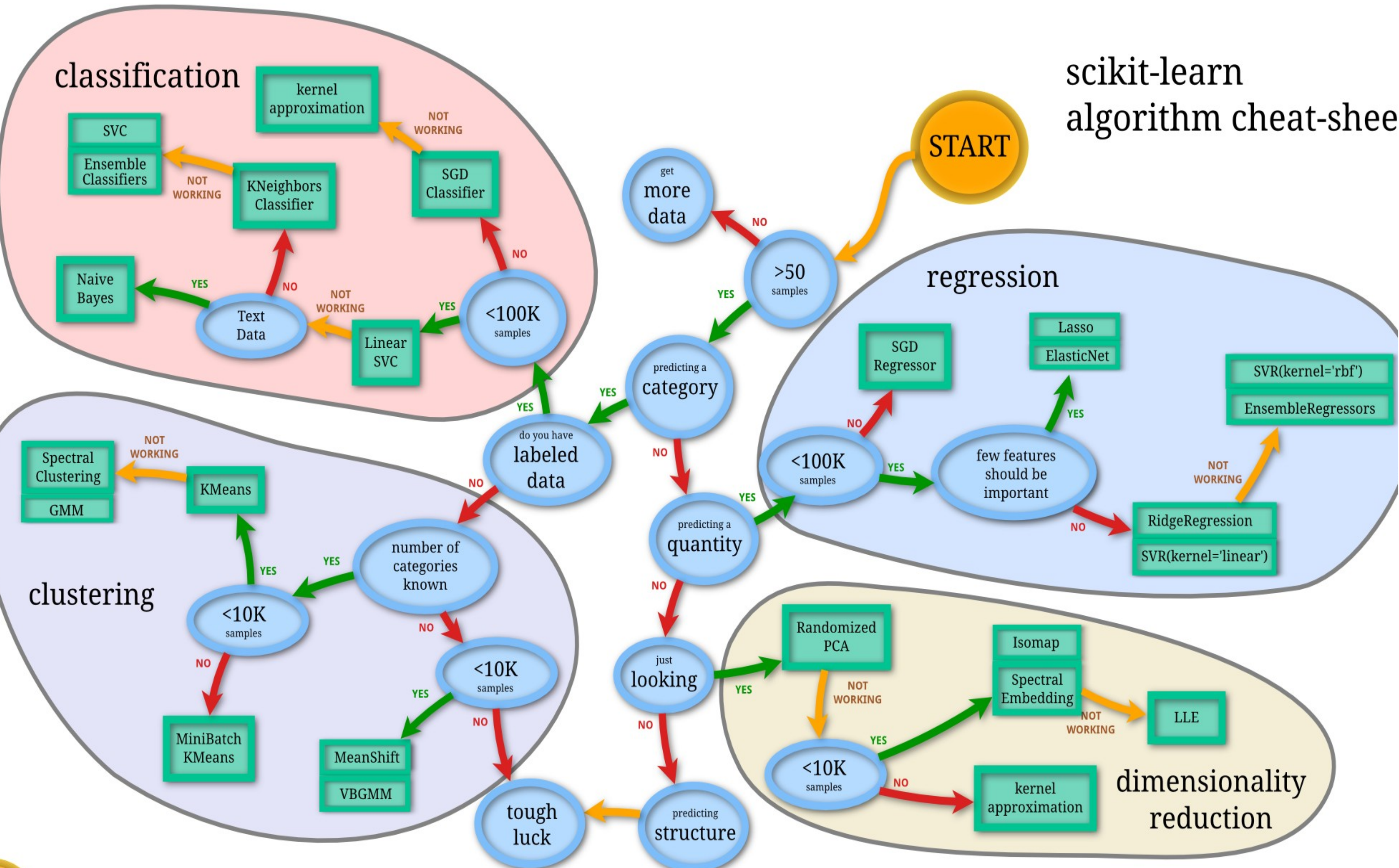
```
[0, 0]  =>  0.676824 expected:  [0]
[0, 1]  =>  0.702282 expected:  [1]
[1, 0]  =>  0.698220 expected:  [1]
[1, 1]  =>  0.718801 expected:  [0]
```

```
SSQ = 0.125263
SSQ = 0.111737
SSQ = 0.101154
SSQ = 0.092994
SSQ = 0.086770
SSQ = 0.082062
SSQ = 0.078519
SSQ = 0.075865
SSQ = 0.073882
SSQ = 0.072402
SSQ = 0.071297
SSQ = 0.070472
SSQ = 0.069856
SSQ = 0.069394
SSQ = 0.069048
SSQ = 0.068786
SSQ = 0.068588
SSQ = 0.068437
SSQ = 0.068321
SSQ = 0.068231
SSQ = 0.068160
SSQ = 0.068104
```


Error while learning XOR
SSQ
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0
200
400
600
800
1000
1200
Iterations

```
finished after 1289 iterations
[0, 0]  =>  0.187037 expected:  [0]
[0, 1]  =>  0.826948 expected:  [1]
[1, 0]  =>  0.826635 expected:  [1]
[1, 1]  =>  0.162049 expected:  [0]
```


1.00
1.00
-1.45
4.62
3.41
-0.77
4.42
4.60
2.06
-2.05
0.88
1.00
0.79
0.93
-3.61
7.78
-2.35
-4.76
0.16


scikit-learn
algorithm cheat-shee
START
>50 samples
get more data
NO
YES
predicting a category
do you have labeled data
<100K samples
Linear SVC
Text Data
Naive Bayes
KNeighbors Classifier
SVC
Ensemble Classifiers
SGD Classifier
kernel approximation
NOT WORKING
classification
predicting a quantity
<100K samples
SGD Regressor
few features should be important
Lasso
ElasticNet
RidgeRegression
SVR(kernel='linear')
SVR(kernel='rbf')
EnsembleRegressors
regression
number of categories known
<10K samples
KMeans
Spectral Clustering
GMM
MiniBatch KMeans
MeanShift
VBGMM
clustering
tough luck
predicting structure
just looking
Randomized PCA
Isomap
Spectral Embedding
LLE
kernel approximation
dimensionality reduction


FINAL EGAD
"That's all Folks!"

bk@xk7.com
@georgebernhard