Deep Learning: a crash course

James G. Shanahan [1,2,3]
Jimi@iSchool.Berkeley.edu

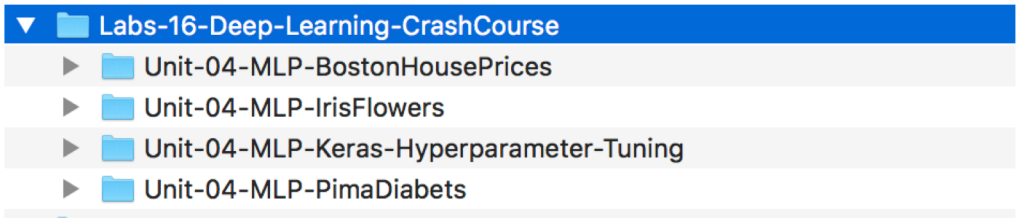
@JimiShan

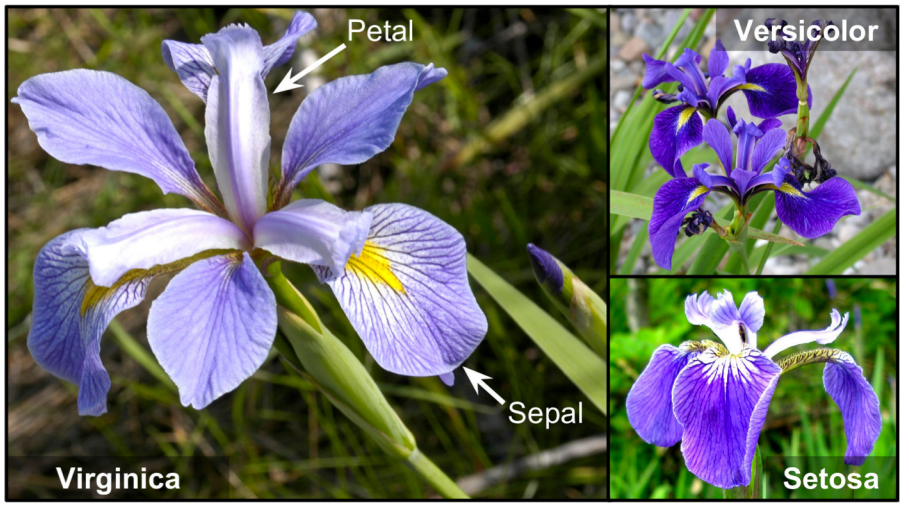
[1]Church and Duncan Group,
[2]Information School, UC Berkeley
[3]School of Informatics, Computing and Engineering, Indiana University

Notebooks for lecture

- **Labs for classification/regression using deep learning**
  - On GitHub: /Labs/Unit-16
    - *Src/Dev/Labs/Unit-16*

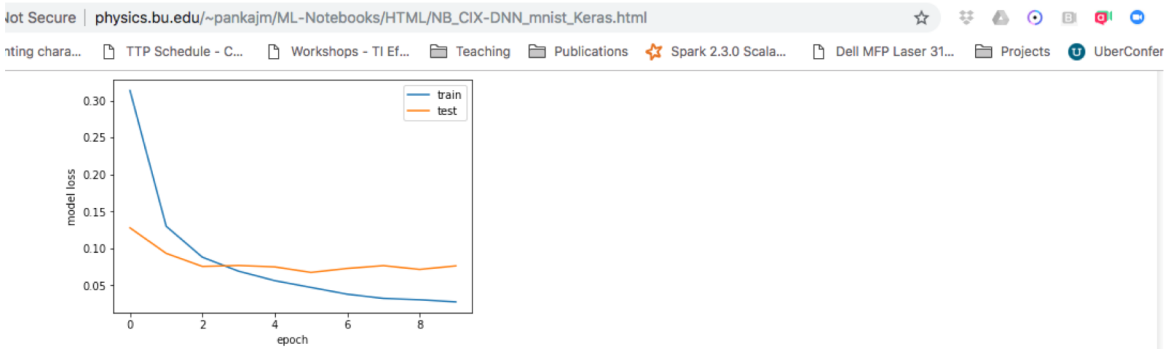
- **QUIZ: Iris Notebook GOOGLE Colab**
  - Iris: Three class classification problem
    - https://drive.google.com/file/d/1AcVGV64AU0-gFM8qW0t0Zy5JspTQi1vk/view?usp=sharing





Notebook: CNN and GridSearch

http://physics.bu.edu/~pankajm/ML-Notebooks/HTML/NB_CIX-DNN_mnist_Keras.html



Step 6: Modify the Hyperparameters to Optimize Performance of the Model

Last, we show how to use the grid search option of scikit-learn to optimize the hyperparameters of our model. An excellent blog on this by Jason Brownlee can be found on https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/.

[7]:
```
from sklearn.model_selection import GridSearchCV
from keras.wrappers.scikit_learn import KerasClassifier

# call Keras scikit wrapper
model_gridsearch = KerasClassifier(build_fn=compile_model,
                                   epochs=1,
                                   batch_size=batch_size,
                                   verbose=1)

# list of allowed optional arguments for the optimizer, see `compile_model()`
optimizer = ['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam']
# define parameter dictionary
param_grid = dict(optimizer=optimizer)
# call scikit grid search module
grid = GridSearchCV(estimator=model_gridsearch, param_grid=param_grid, n_jobs=1, cv=4)
```

Outline

1. **Introduction**
2. **ML and deep learning review**
3. **What is computer vision?**
    1. Computer vision
    2. Convolutional Neural Nets (CNNs)
4. **Deep NN Architectures for CV Tasks**
    1. Backbone networks
5. **Solving edge-based IoT**
6. **Conclusions and Next steps**

∇ the gradient chorus

- **What is the gradient for linear regression?**

- **Chorus**
  - The gradient is the weighted sum of the training data, where the weights are proportional to the error (for each example) !

weight *example*

$$\frac{\partial E}{\partial W} = (O - t)\,\boldsymbol{X}$$

$$W = W - \alpha \times \frac{\partial E}{\partial W}$$

$$\begin{bmatrix} 4.32 \\ 3.98 \\ 7.92 \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \\ 8 \end{bmatrix} - 0.01 \times 2 \times \begin{bmatrix} \mathbf{34} \\ \mathbf{1} \\ \mathbf{4} \end{bmatrix}$$

$$4.32 = 5 - 0.01 \times (2 \times 34) \text{ for } i = 1$$

Bio: James G. Shanahan

- **30 years in Artificial Intelligence**
- **Currently**
  - Principal and Founder, Church and Duncan Data Science Consultancy
    - Clients: Target, Adobe, Akamai, Ancestry, AT&T, Nokia Siemens, SearchMe, …
  - Teaching
    - Founding faculty of UC Berkeley MIDS program; curriculum development
      - Large Scale Machine Learning, Optimization theory
      - Visiting Professor, Department of Economics, University of Gent, Belgium (Deep Learning)
  - Advising: ChartBoost, InMobi, Aylien, InferSystems (aquired by Kochava),
- **Previously**
  - Entrepreneur: Cofounder of Document Souls and RTB Fast, Church and Duncan Group Inc.
  - NativeX: SVP of Data Science, Chief Scientist, and board member (Acquired by MobVista)
  - Founding Chief Scientist, Turn Inc. (Acquired by SingTel's Admobi)
  - Principal Scientist, Clairvoyance Corp (CMU spinoff; sister lab to JRC)
  - Research Scientist, Xerox Research
- **Education:** PhD in AI, University of Bristol, UK; B.Sc. CS, Uni. of Limerick, Ireland

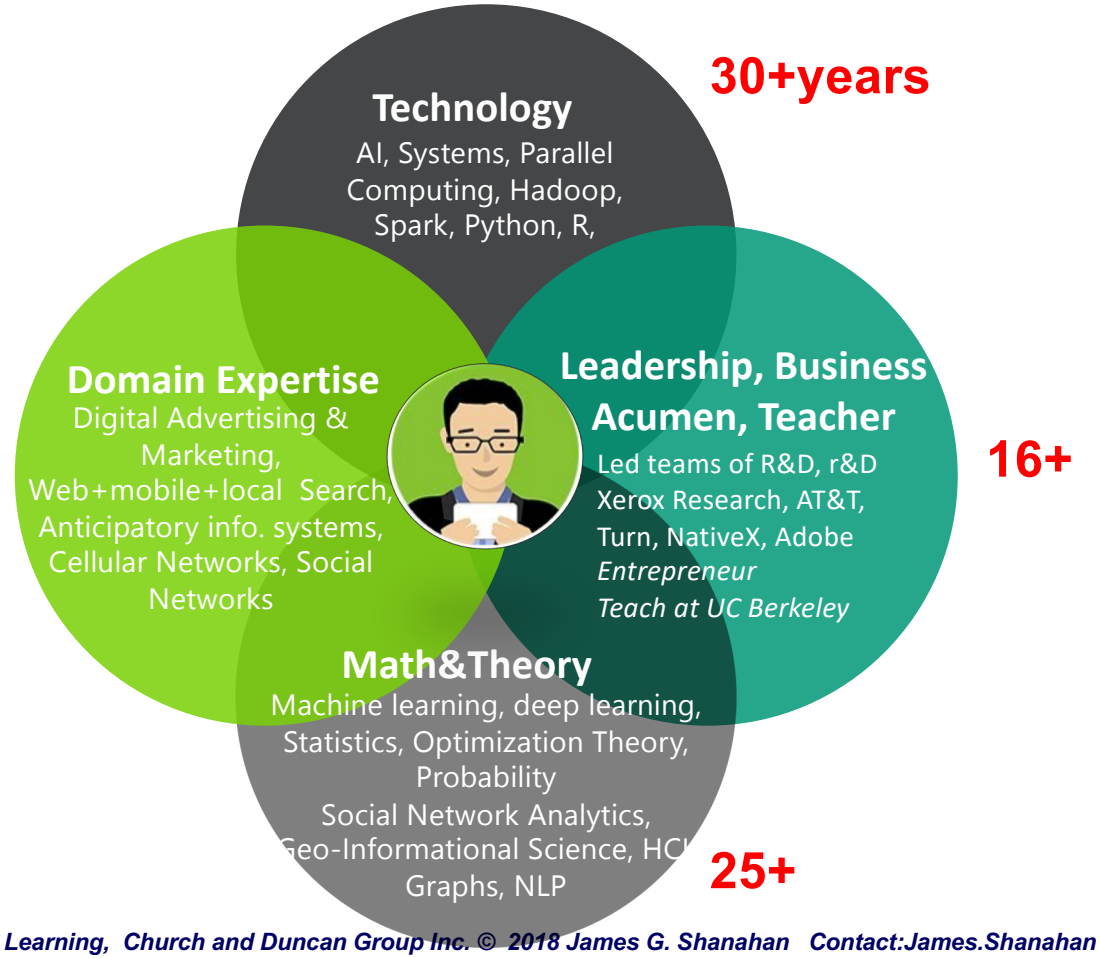
AI is transforming industries

	CONSUMER	HEALTH	FINANCE	RETAIL	GOVERNMENT	ENERGY	TRANSPORT	INDUSTRIAL	OTHER
EXAMPLES	Smart Assistants	Enhanced Diagnostics	Algorithmic Trading	Support	Defense	Oil & Gas Exploration	Automated Cars	Factory Automation	Advertising
	Chatbots	Drug Discovery	Fraud Detection	Experience	Data Insights	Smart Grid	Automated Trucking	Predictive Maintenance	Education
	Search	Patient Care	Research	Marketing	Safety & Security	Operational Improvement	Aerospace	Precision Agriculture	Gaming
	Personalization	Research	Personal Finance	Merchandising	Resident Engagement	Conservation	Shipping	Field Automation	Professional & IT Services
	Augmented Reality	Sensory Aids	Risk Mitigation	Loyalty	Smarter Cities		Search & Rescue		Telco/Media
	Robots			Supply Chain					Sports
				Security					

Artificial intelligence was founded as an academic discipline in 1956, and in the years since has experienced several waves of optimism….

[Intel.com]

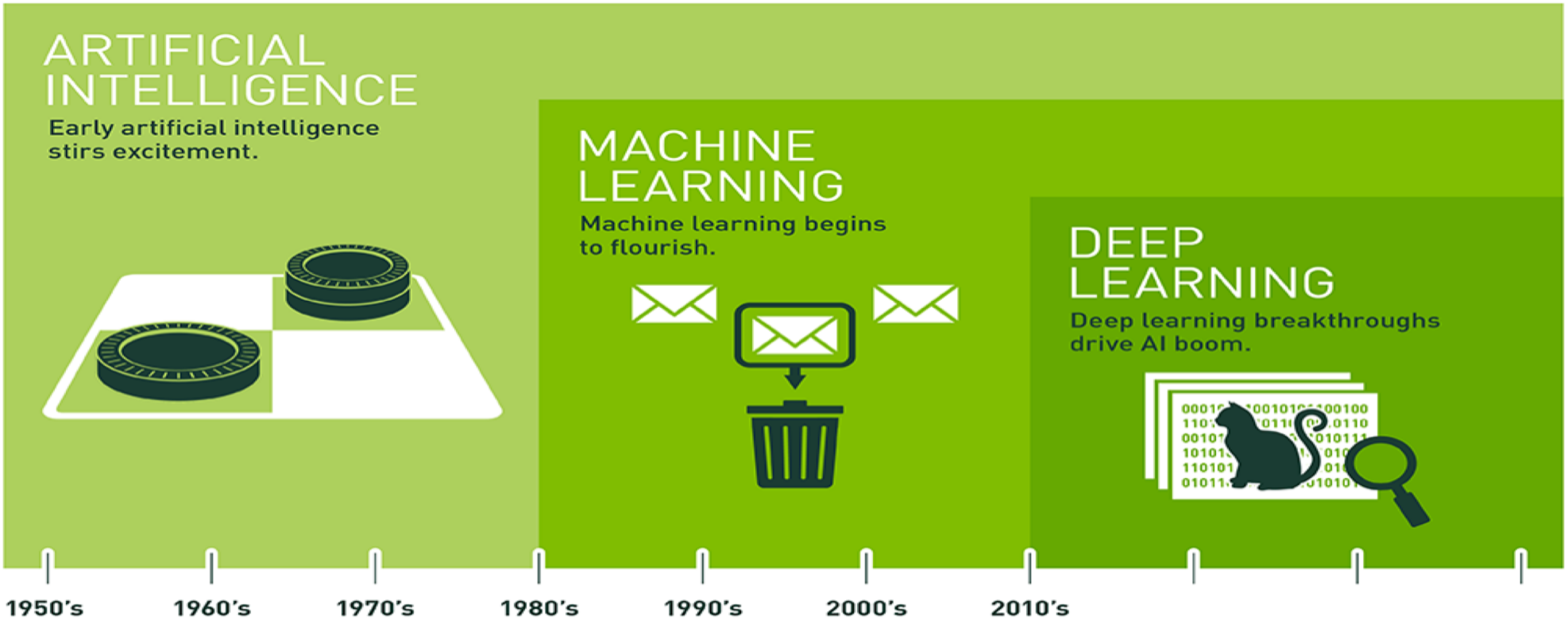
James G. Shanahan 30+ years in artificial intelligence



“The world’s first TRILLIONAIRE is the person who masters AI”

Mark Cuban – Owner, Dallas Mavericks

AI→ML →DL



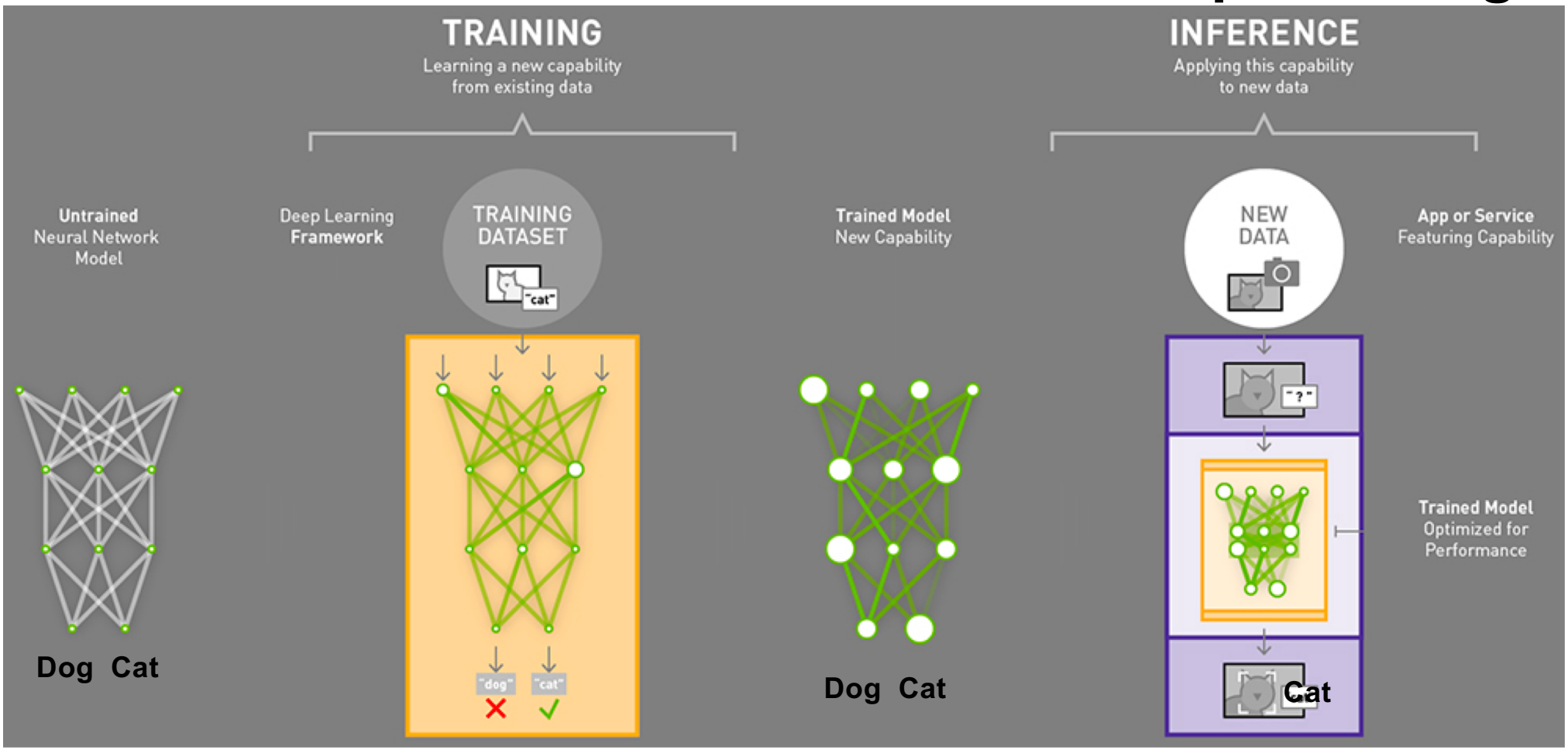
Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

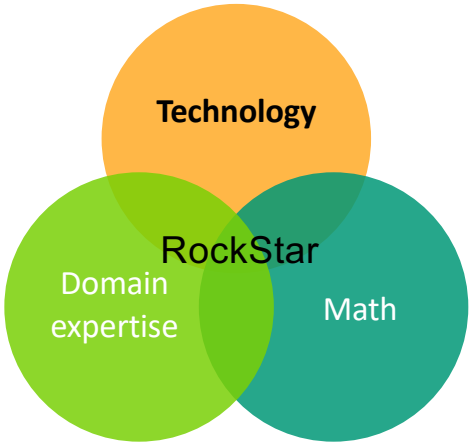
Cats and Dogs classifier

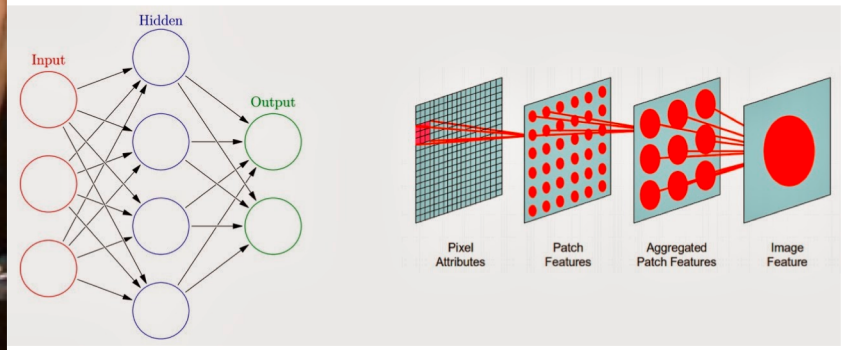
Sample of cats & dogs images from Kaggle Dataset

Deep Learning

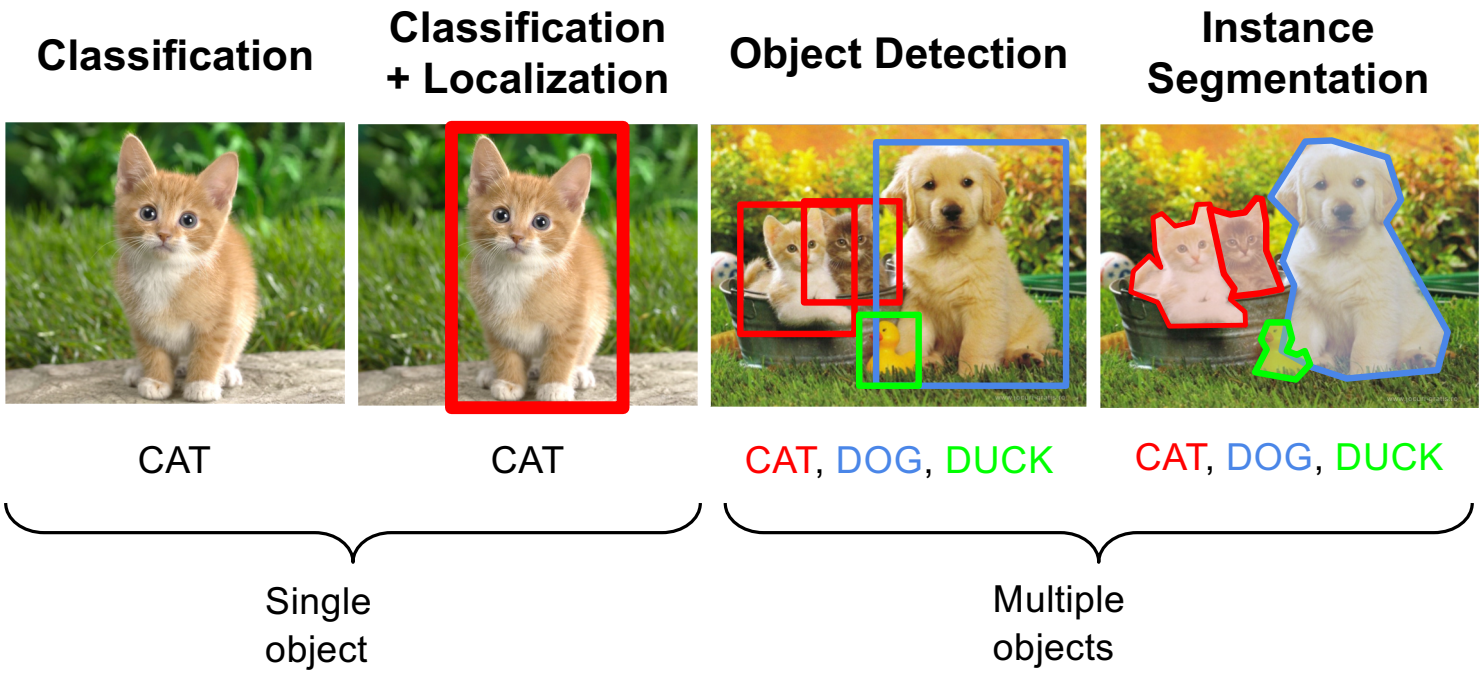


RockStars and Super Models





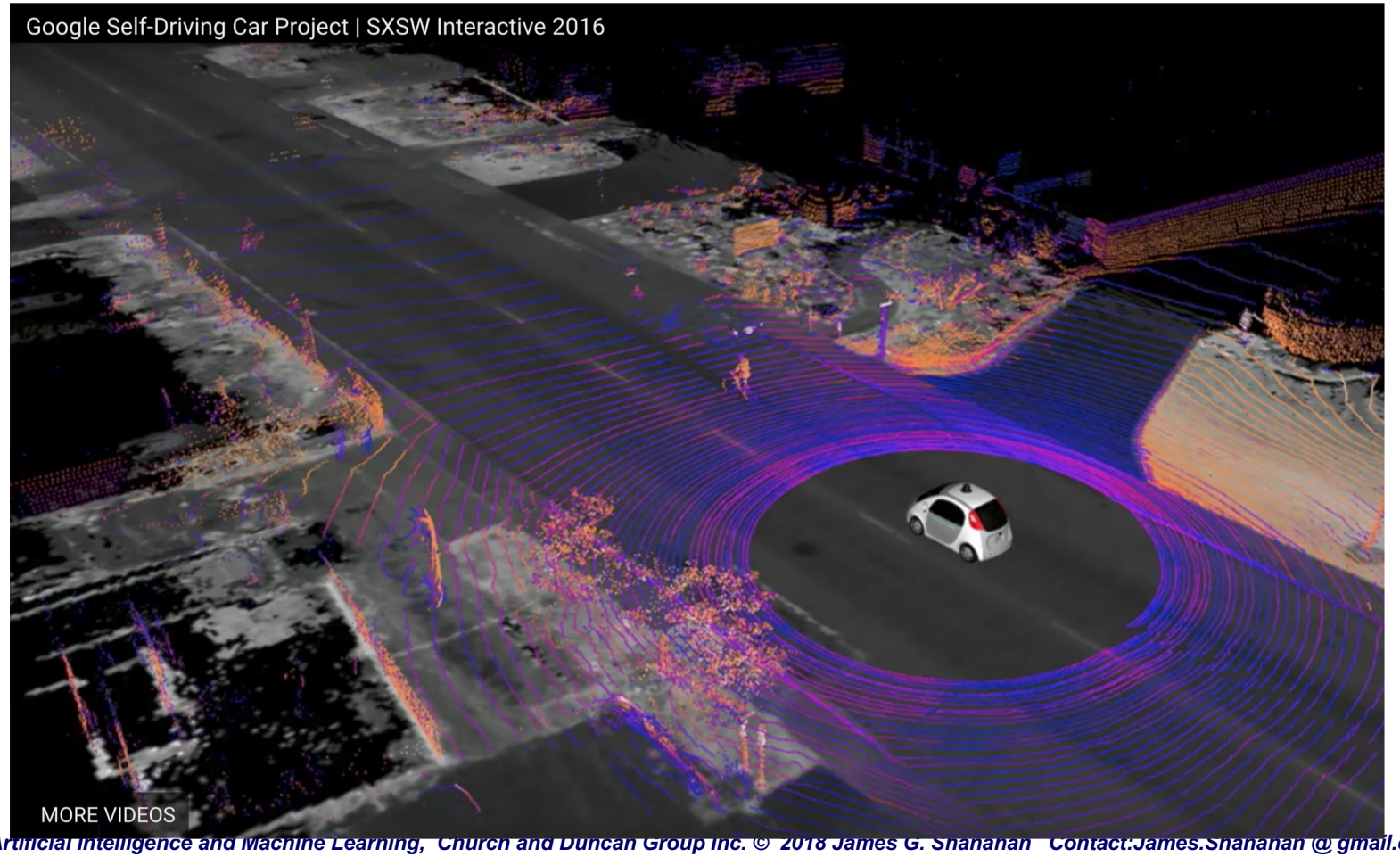
Imagenet Tasks: Computer Vision Tasks



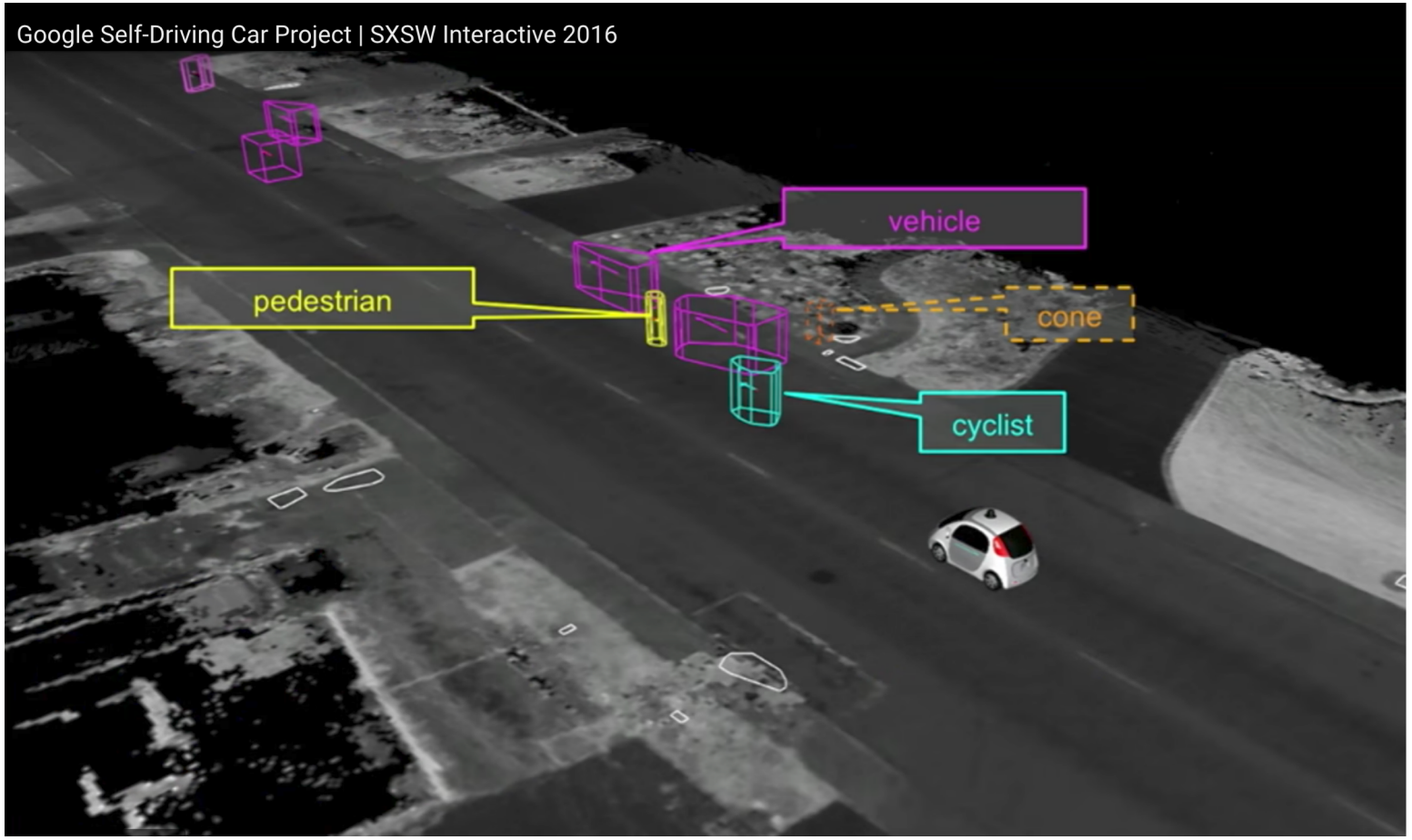
unlock your iPhone with your face

- **With Face ID, Apple has launched a grand experiment in a form of biometric security previously untested at this scale.**

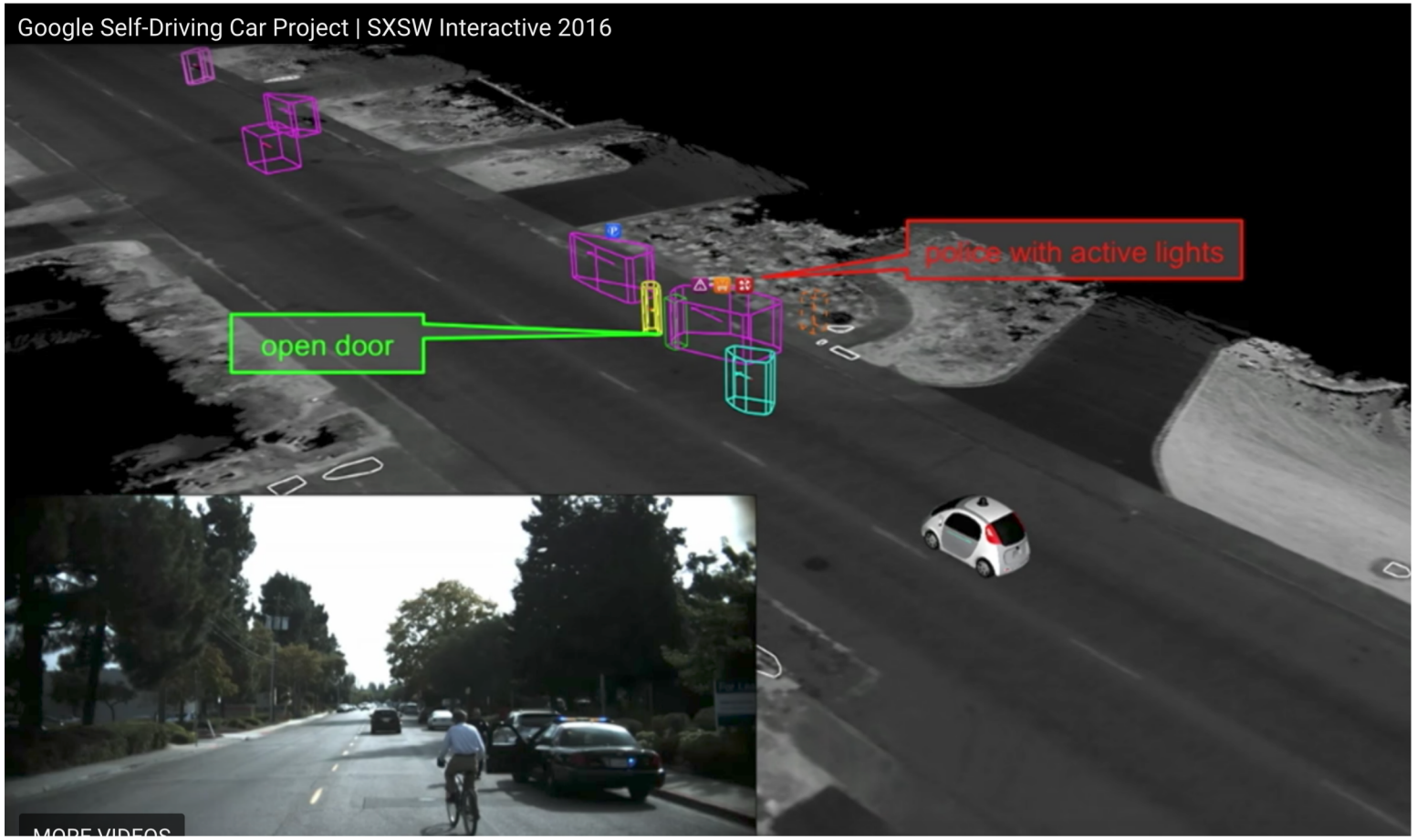
Autonomous driving: Sensor-based data (millions of inputs) Radar, Laser, Map, Camera data



Step 1 of 3: Classify things



Step 2 of 3: Reason, plan and …



Step 3 of 3: Execute

Segmentation

Atari Games … all of them

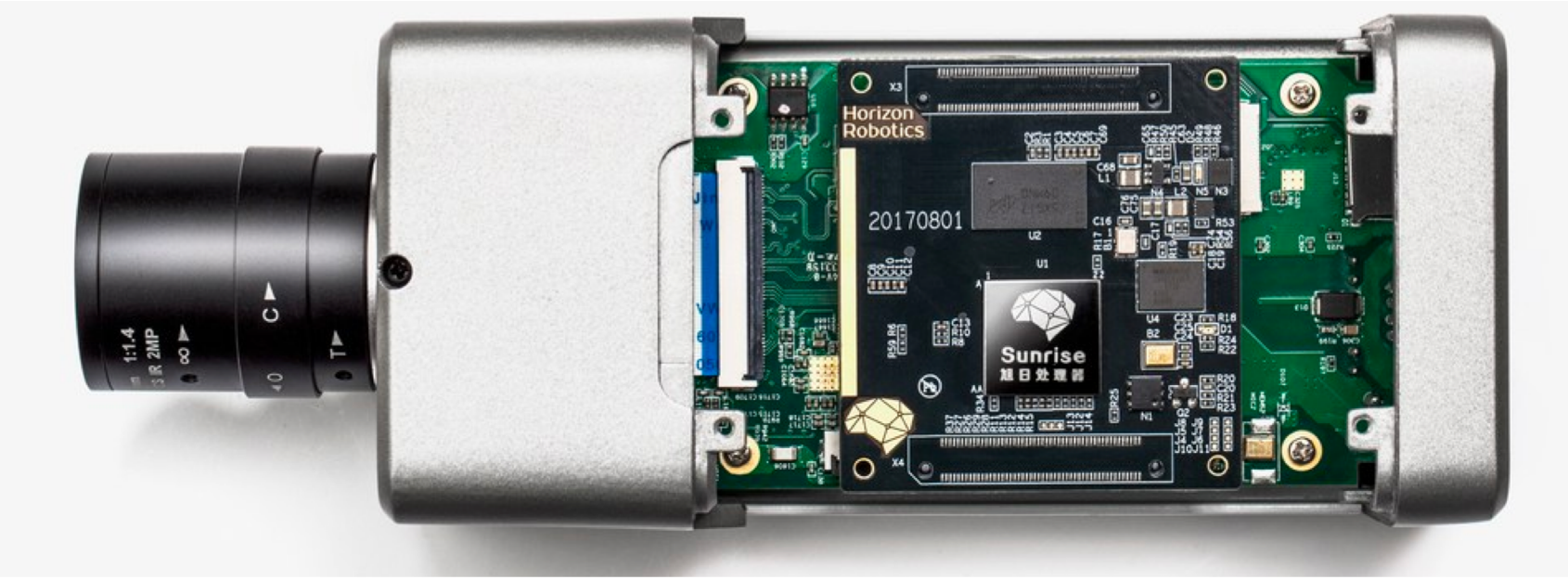
What does $500M look like?

- **Goal:**
  - System that can play any game
  - Only use information human has
    - Raw pixel data
  - **Computer vision + Reinforcement Learning**
- **Success:**
  - Google's Deep Mind team
  - Used "Deep" Neural Network
  - Can learn any Atari 2600 game
- **Can we get more general?**

- **Video:** https://youtu.be/V1eYniJ0Rnk
- **Code:** https://sites.google.com/a/deepmind.com/dqn/

IoT devices will know what they are seeing

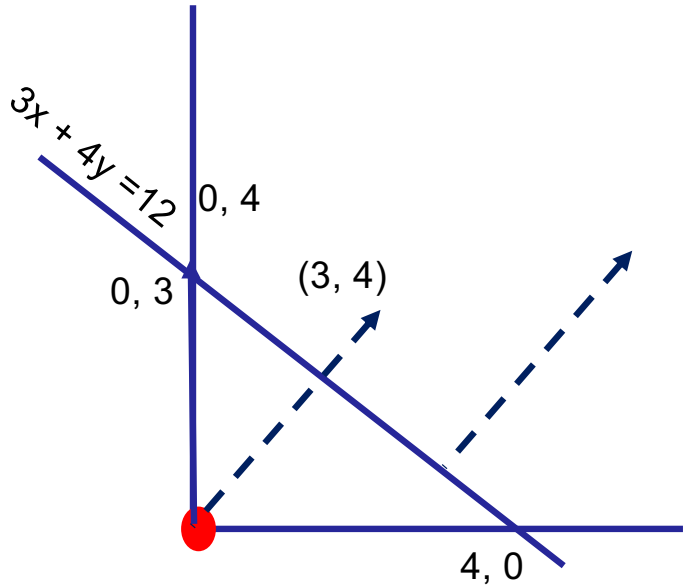
- A new breed of chips tuned for artificial intelligence is arriving to help cameras/devices around stores, sidewalks, and homes make sense of what they see.
- Cloud based → Edge-based processing



Outline

1. **Introduction**
2. **ML and deep learning review**
3. **What is computer vision?**
    1. Computer vision
    2. Convolutional Neural Nets (CNNs)
4. **Deep NN Architectures for CV Tasks**
    1. Backbone Networks
5. **Solving edge-based IoT**
6. **Conclusions and Next steps**

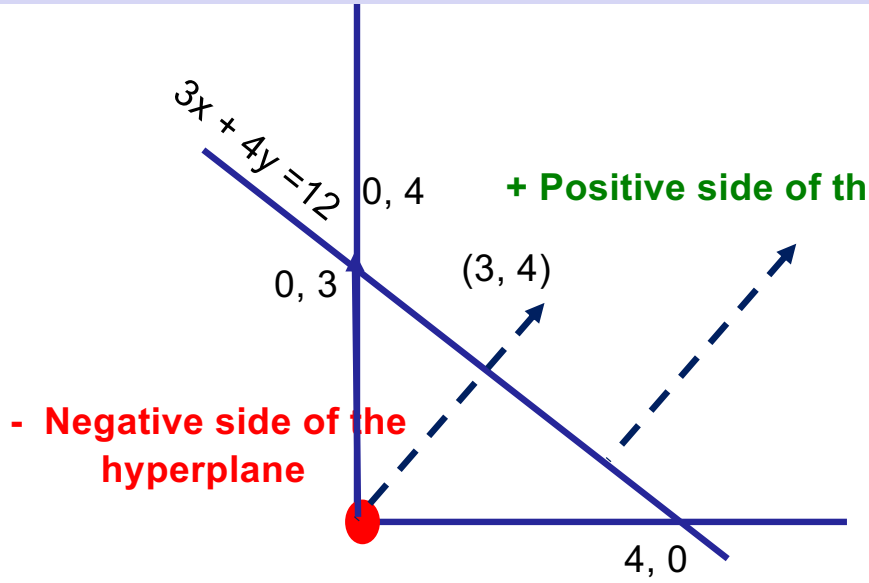
HyperPlane 3x + 4y =12



To see if a point is on the line we plug in its values into the equation and see if evaluates to 0 (ZERO)

HyperPlane 3x + 4y – 12 = 0

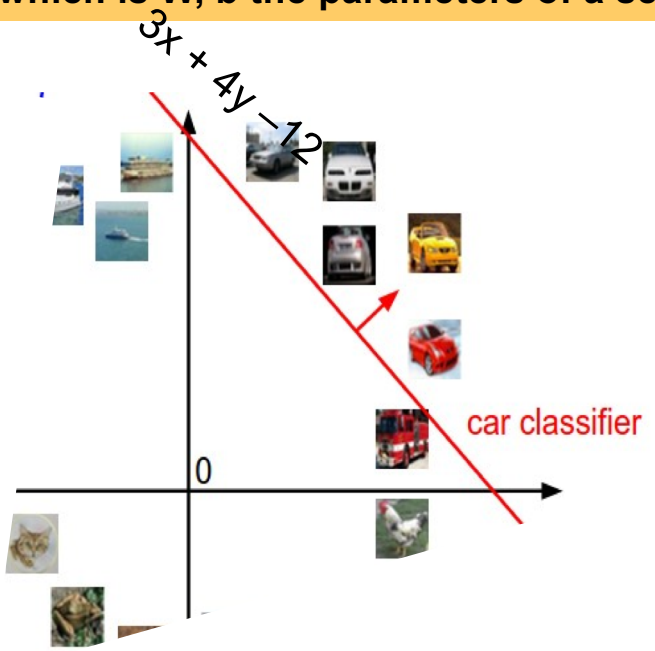
Hyperplane is characterized by the Vector that normal to Hyperpane (3, 4) and the bias term (in this case –12)
You can place this vector anywhere on the hyperplane.
Vector points to the positive side of the hyperplane [see next slide for more details]



To see if a point is on the line we plug in the values into the equation and see if evaluates to 0 (ZERO)

Interpreting a Linear Classifier: 2 halfspaces

Learn θ which is W, b the parameters of a separating hyperplane

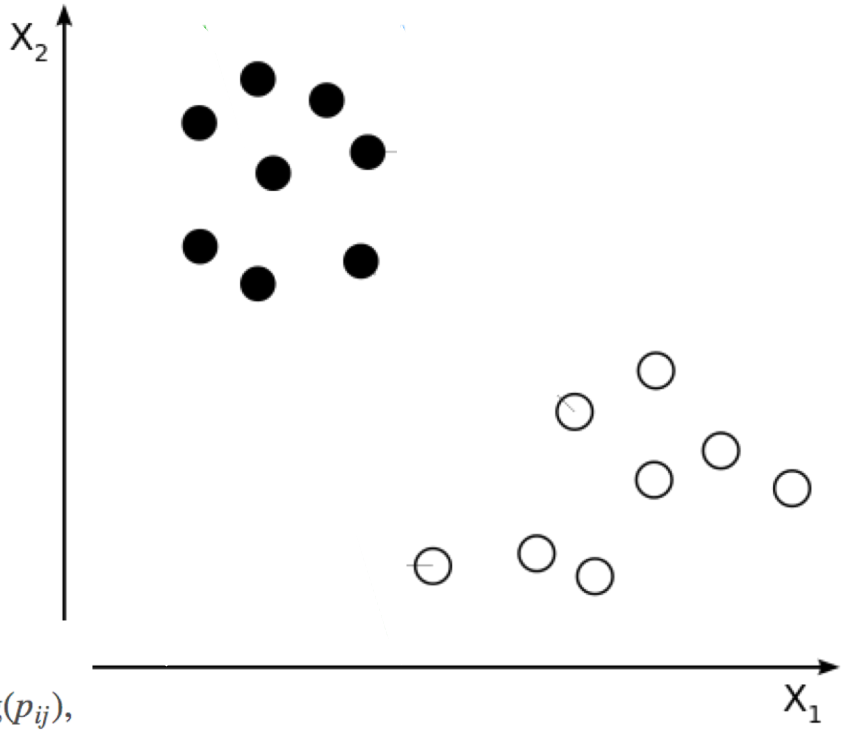


3x + 4y –12; represent as W=[3,4, b –12]

$$f(x_i, W, b) = Wx_i + b$$

[32x32x3]
array of numbers 0...1
(3072 numbers total)

Binary Linear Classifier: split space into 2 pure halfspaces



- Data lives in n-space;
- Partition training data such that one class lives in one halfspace and the other in the other halfspace
  - 2 pure halfspaces
- Accuracy of H_3 is 16/16; 100%

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij}),$$

H_3 separates them with the maximum margin.

$$logloss=-1N\sum i=1N\sum j=1Myijlog(pij),$$

Binary Linear Classifier: split space into 2 pure halfspaces

Submissions are evaluated using the multi-class logarithmic loss. Each image has been labeled with one true class. For each image, you must submit a set of predicted probabilities (one for every image). The formula is then,

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij}),$$

where N is the number of images in the test set, M is the number of image class labels, log is the natural logarithm, y_{ij} is 1 if observation i belongs to class j and 0 otherwise, and p_{ij} is the predicted probability that observation i belongs to class j.

The submitted probabilities for a given image are not required to sum to one because they are rescaled prior to being scored (each row is divided by the row sum). In order to avoid the extremes of the log function, predicted probabilities are replaced with $max(min(p, 1-10^{-15}), 10^{-15})$.

$logloss = -\frac{1}{N}\sum_{i=}^{l}$

H_3 separates them with the maximum margin.

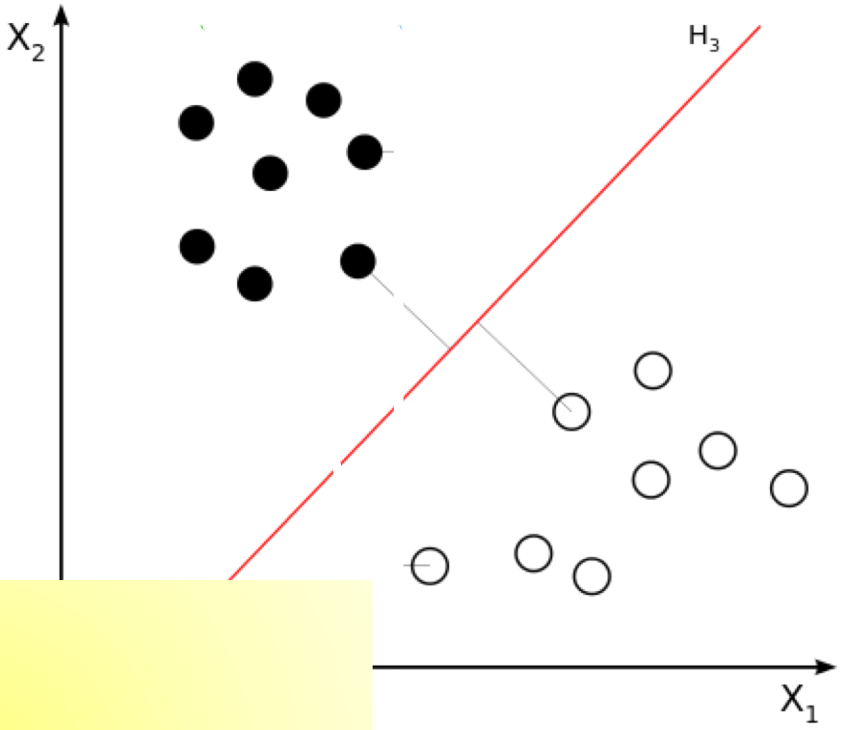
$gloss=-1N\sum i=1N\sum j=1Myij\log(pij),$

such that
halfspace
ner

6; 100%

Binary Linear Classifier: split space into 2 pure halfspaces

Perpendicular distance from a point to a hyperplane tell me the class of the test case



- Data lives in n-space;
- Partition training data such that one class lives in one halfspace and the other in the other halfspace
  - 2 pure halfspaces
- Accuracy of H_3 is 16/16; 100%

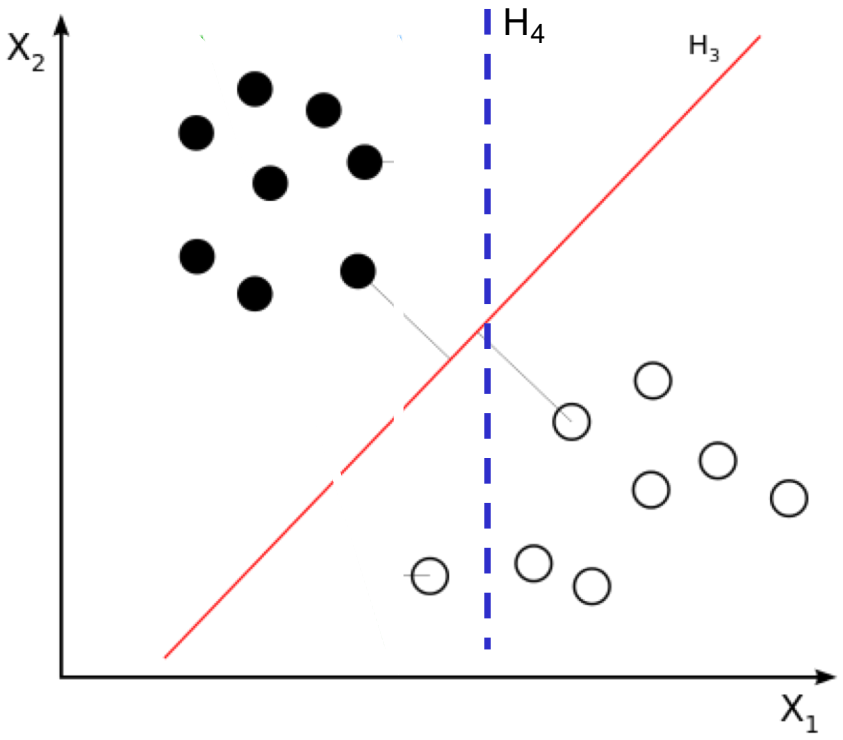
$$J = \frac{1}{N}\sum_{i=1}^{N} L(X_i^T W, y_i)$$

Where

- $L(X_i^T W, y_i) = 0$ if $heaviside(X_i^T W) == y_i$,
- $L(X_i^T W, y_i) = 1$ otherwise
- $where\ \ heaviside(X_i^T W) = \begin{cases} -1 & X_i^T W < 0 \\ 1 & X_i^T W \geq 0 \end{cases}$

H_3 separates them with the maximum margin.

Binary Linear Classifier: split space into 2 pure halfspaces



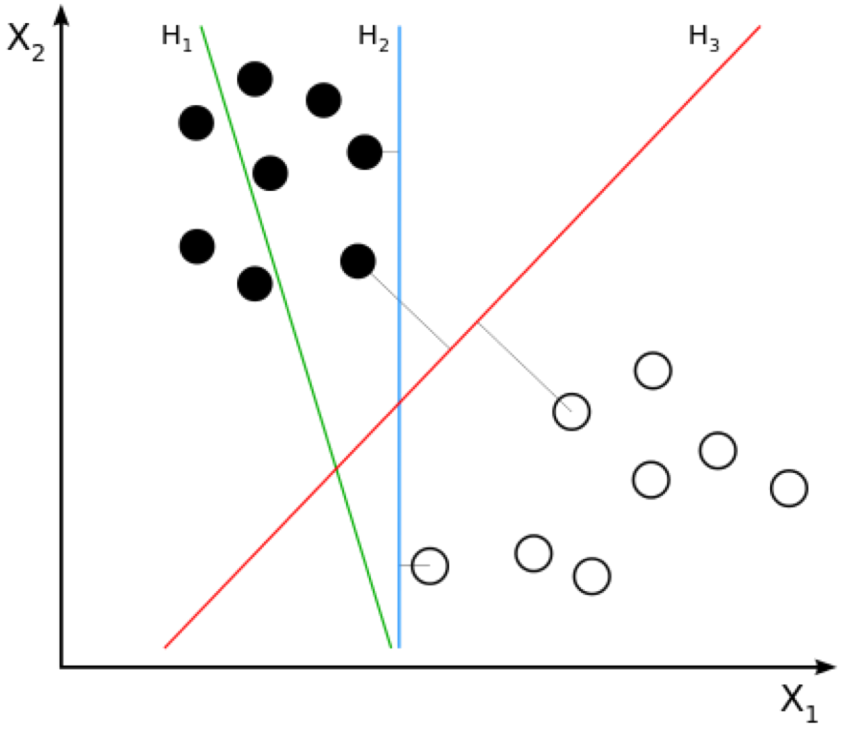
- Data lives in n-space;
- Partition training data such that one class lives in one halfspace and the other in the other halfspace
  - 2 pure halfspaces

- Accuracy of H_3 is 16/16; 100%
- Accuracy of H_4 is 15/16; 93.75%

- Which hyperplane do you prefer?
  - H_3

H_1 does not separate the classes.
H_2 does, but only with a small margin.
H_3 separates them with the maximum margin.

Binary Linear Classifier: many possibilities



- Data lives in n-space;
- Partition training data such that one class lives in one halfspace and the other in the other halfspace
  - 2 pure halfspaces

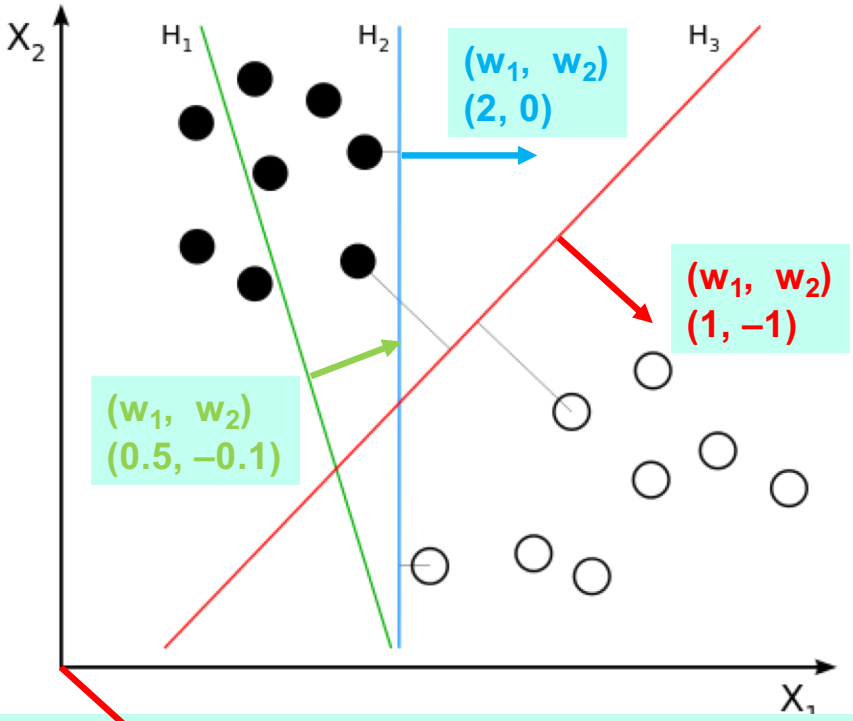
- Accuracy of H_1 is 11/16; 68.75%
- Accuracy of H_2 is 16/16; 100%
- Accuracy of H_3 is 16/16; 100%

- Which hyperplane do you prefer?
  - H_3 or H_2 (if accuracy is used only)

H_1 does not separate the classes.
H_2 does, but only with a small margin.
H_3 separates them with the maximum margin.

Binary Linear Classifier: many possibilities

$2x_1 + 0\ X_2 + C$



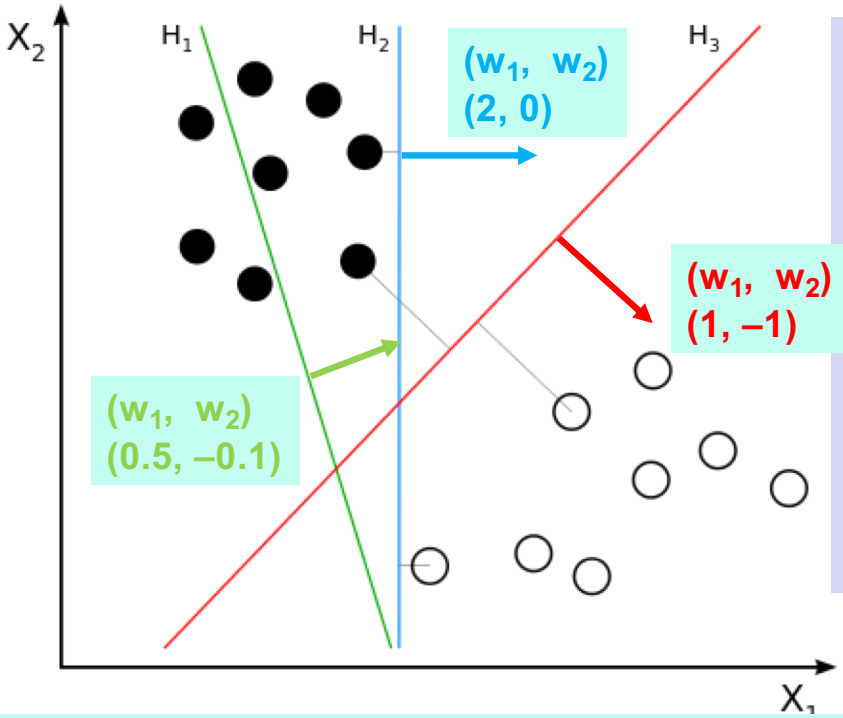
- Data lives in n-space;
- Partition training data such that one class lives in one halfspace and the other in the other halfspace
- 2 pure halfspaces

- Accuracy of H_1 is 11/16; 68.75%
- Accuracy of H_2 is 16/16; 100%
- Accuracy of H_3 is 16/16; 100%

- Which hyperplane do you prefer?
  - H_3 or H_2 (if accuracy is used only)

Can use accuracy as measure of quality
Each hyperplane is determined by the vector that is normal to it (assume no bias term for now), i.e., w_1, w_2

Binary Linear Classifier: many possibilities



- Data lives in n-space;
- Partition training data such that one class lives in one halfspace and the other in the other halfspace
- 2 pure halfspaces

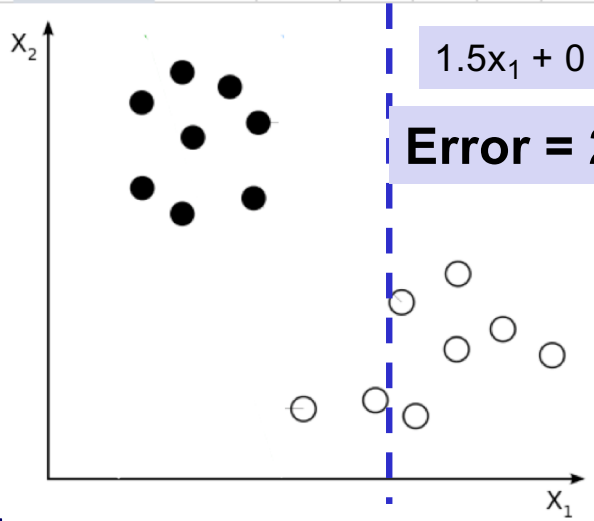
- Accuracy of H_1 is 11/16; 68.75% (31%)
- Accuracy of H_2 is 16/16; 100% (0%)
- Accuracy of H_3 is 16/16; 100% (0%)

- Which hyperplane do you prefer?
  - H_3 or H_2 (if accuracy is used only)

Can use accuracy as measure of quality OR
OR Error rate = number of mistakes / number of examples
Each hyperplane is determined by the vector that is normal to it (assume no bias term for now), i.e., w_1, w_2

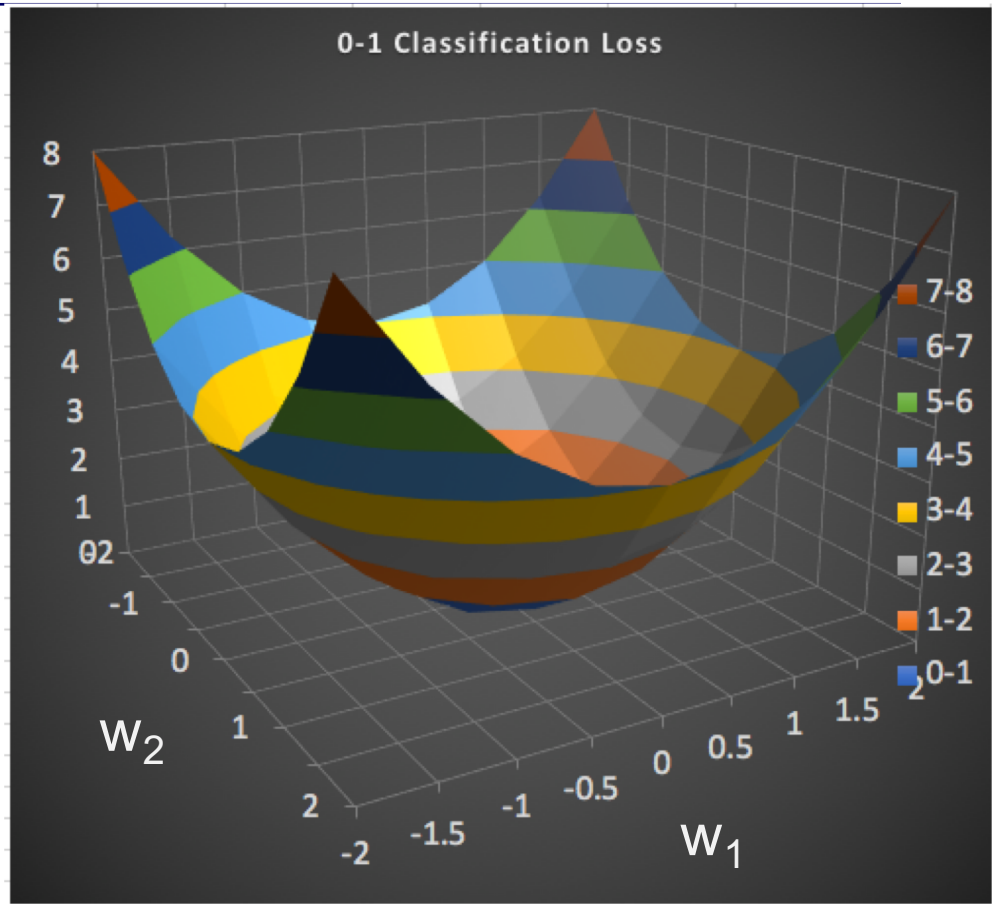
Grid search: w_1 and w_2 → 0-1 Loss for each combo [w_1, w_2]

	A	B	C	D	E	F	G	H	I	J	K
42			W1 -->								
43	w2	0-1L Loss	-2	-1.5	-1	-0.5	0	0.5	1	1.5	2
44		-2	8	6.25	5	4.25	4	4.25	5	6.25	8
45		-1.5	6.25	4.5	3.25	2.5	2.25	2.5	3.25	4.5	6.25
46		-1	5	3.25	2	1.25	1	1.25	2	3.25	5
47		-0.5	4.25	2.5	1.25	0.5	0.25	0.5	1.25	2.5	4.25
48		0	4	2.25	1	0.25	0	0.25	1	2	4
49		0.5	4.25	2.5	1.25	0.5	0.25	0.5	1.25	2.5	4.25
50		1	5	3.25	2	1.25	1	1.25	2	3.25	5
51		1.5	6.25	4.5	3.25	2.5	2.25	2.5	3.25	4.5	6.25
52		2	8	6.25	5	4.25	4	4.25	5	6.25	8



$1.5x_1 + 0\ X_2 + C$

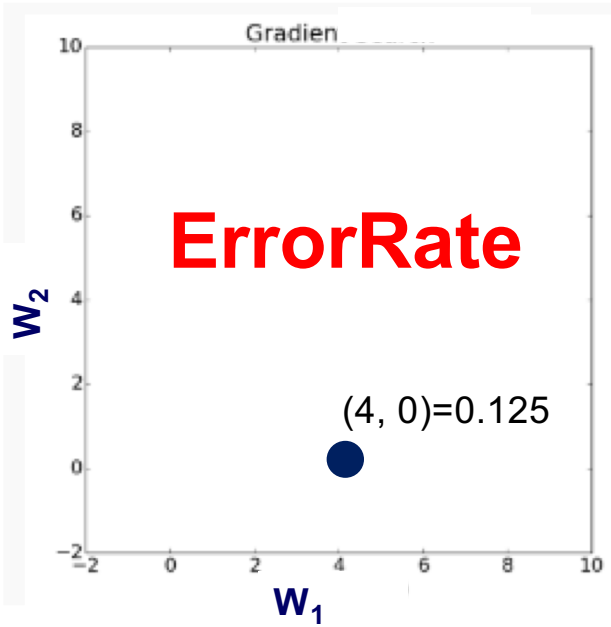
Error = 2/16

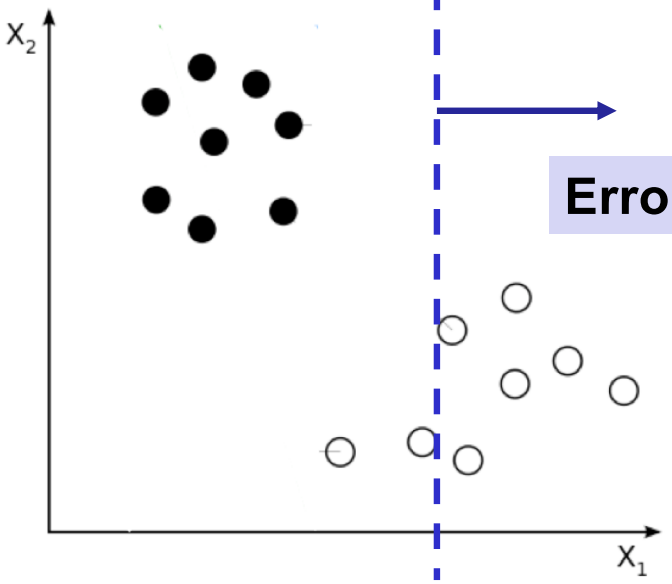


0-1 Loss Function for linear classification models

Model Space
F(W) aka F(θ)

Problem Space
F(X)





Error = 2/16=0.125

$$ErrorRate = \frac{1}{N}\sum_{i=1}^{N} L(X_i^T W, y_i)$$

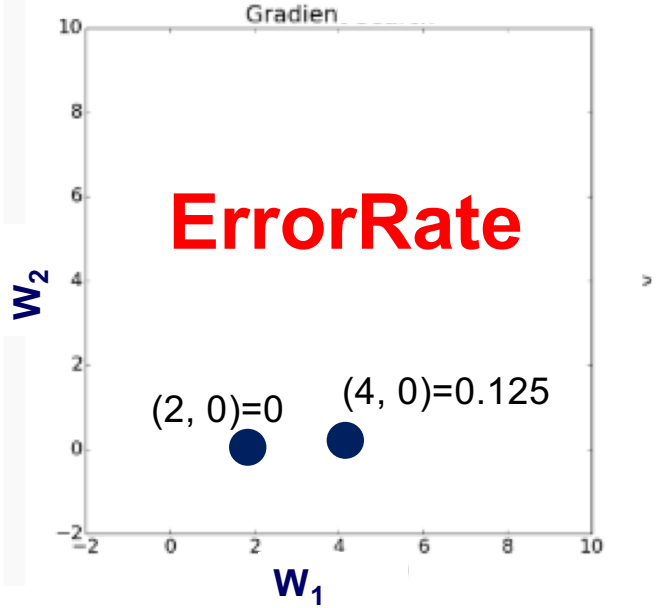
Where

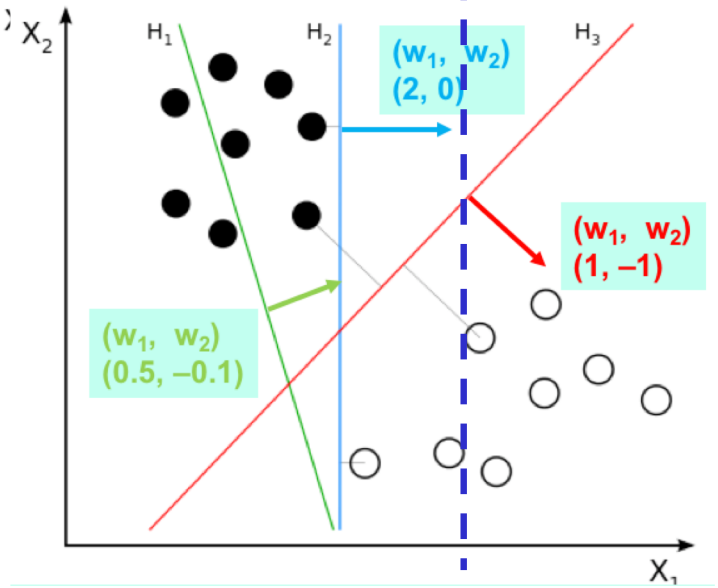
- $L(X_i^T W, y_i) = 0$ if $heaviside(X_i^T W) == y_i$,
- $L(X_i^T W, y_i) = 1$ otherwise
- $where\ heaviside(X_i^T W) = \begin{cases} -1 & X_i^T W < 0 \\ 1 & X_i^T W \geq 0 \end{cases}$

Version Space for linear classification models

Model Space
F(W) aka F(θ)

Problem Space
F(X)





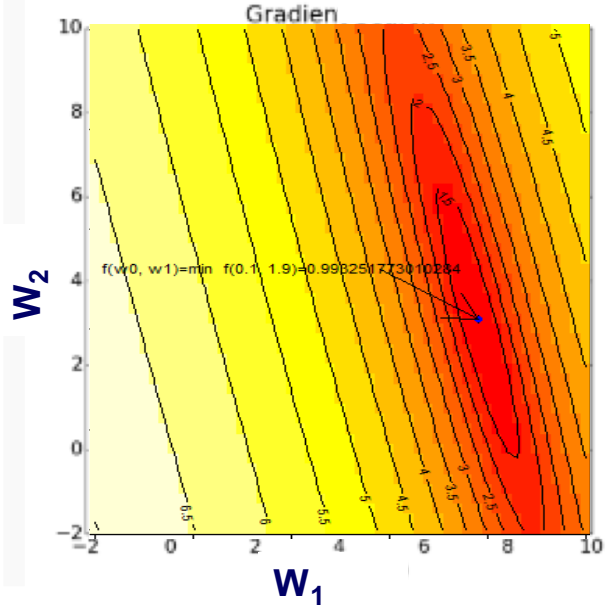
$$ErrorRate = \frac{1}{N}\sum_{i=1}^{N} L(X_i^T W, y_i)$$

Where

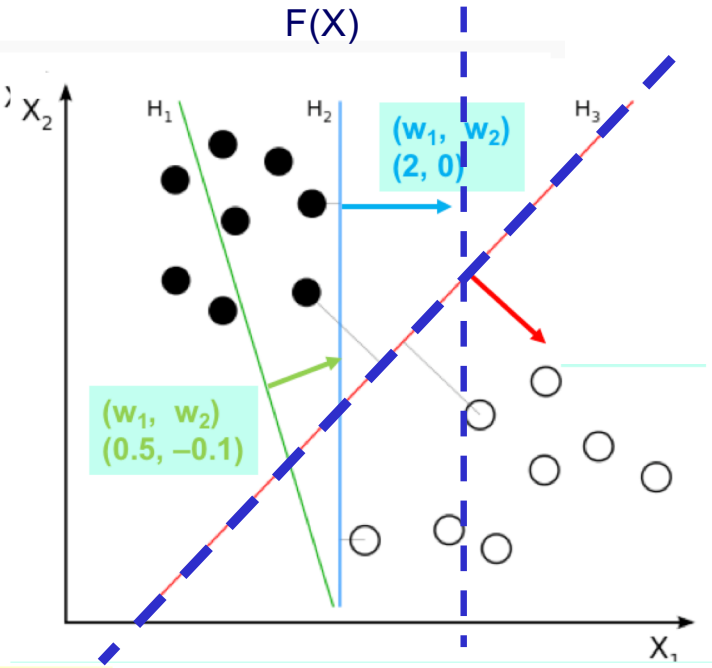
- $L(X_i^T W, y_i) = 0$ if $heaviside(X_i^T W) == y_i$,
- $L(X_i^T W, y_i) = 1$ otherwise
- $where\ heaviside(X_i^T W) = \begin{cases} -1 & X_i^T W < 0 \\ 1 & X_i^T W \geq 0 \end{cases}$

Error surface for linear classification models

Model Space
F(W) aka F(θ)



Problem Space
F(X)



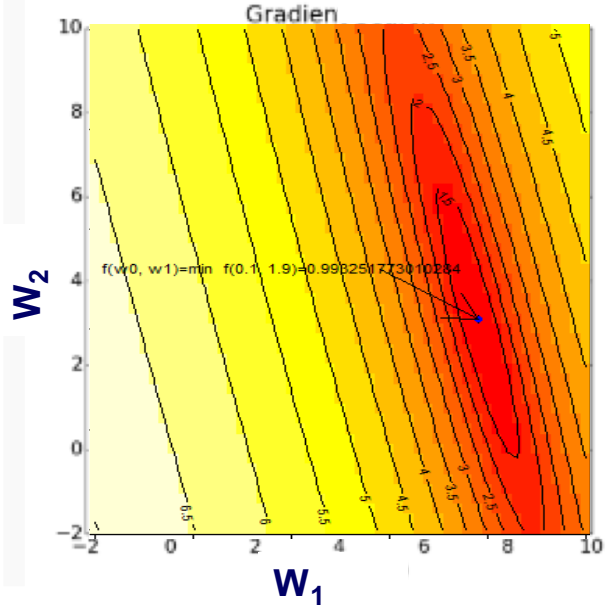
$$ErrorRate = \frac{1}{N}\sum_{i=1}^{N} L(X_i^T W, y_i)$$

Where

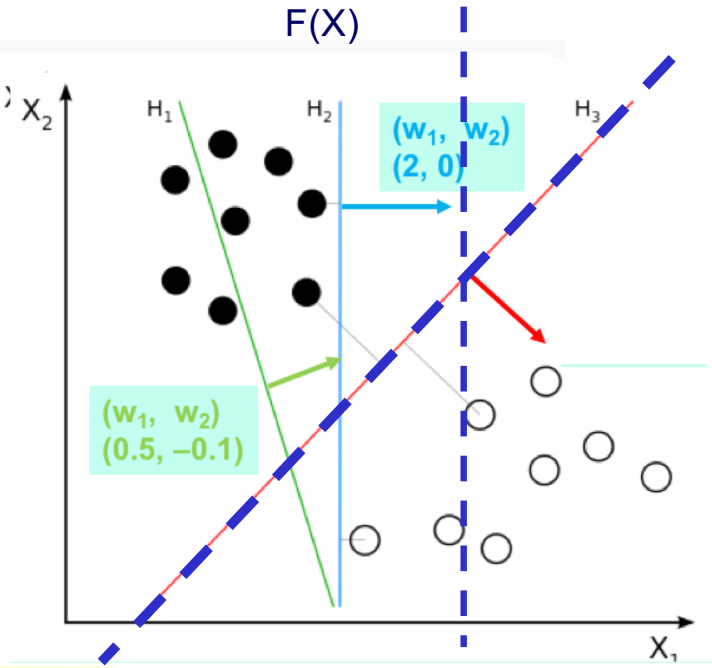
- $L(X_i^T W, y_i) = 0$ if $heaviside(X_i^T W) == y_i$,
- $L(X_i^T W, y_i) = 1$ otherwise
- $where\ heaviside(X_i^T W) = \begin{cases} -1 & X_i^T W < 0 \\ 1 & X_i^T W \geq 0 \end{cases}$

Error surface for linear classification models

Model Space
F(W) aka F(θ)



Problem Space
F(X)



$$ErrorRate = \frac{1}{N}\sum_{i=1}^{N} L(X_i^T W, y_i)$$

Where

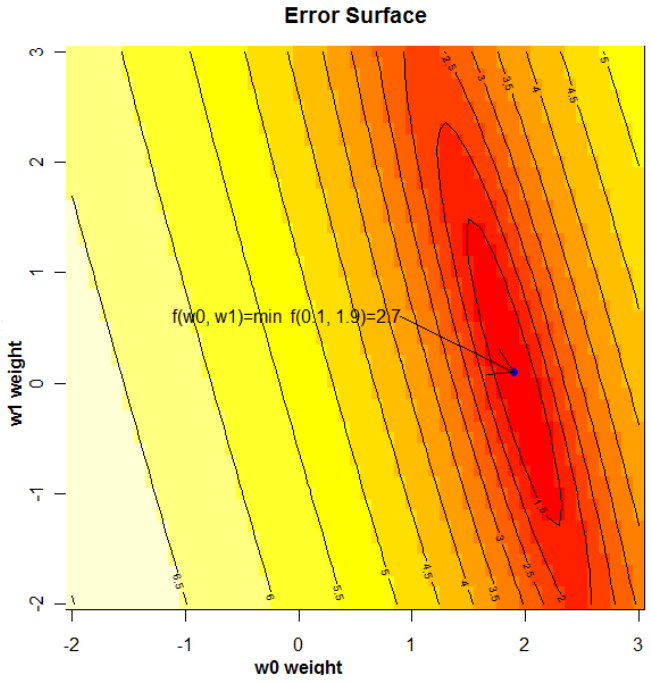
- $L(X_i^T W, y_i) = 0$ if $heaviside(X_i^T W) == y_i$,
- $L(X_i^T W, y_i) = 1$ otherwise
- $where\ heaviside(X_i^T W) = \begin{cases} -1 & X_i^T W < 0 \\ 1 & X_i^T W \geq 0 \end{cases}$

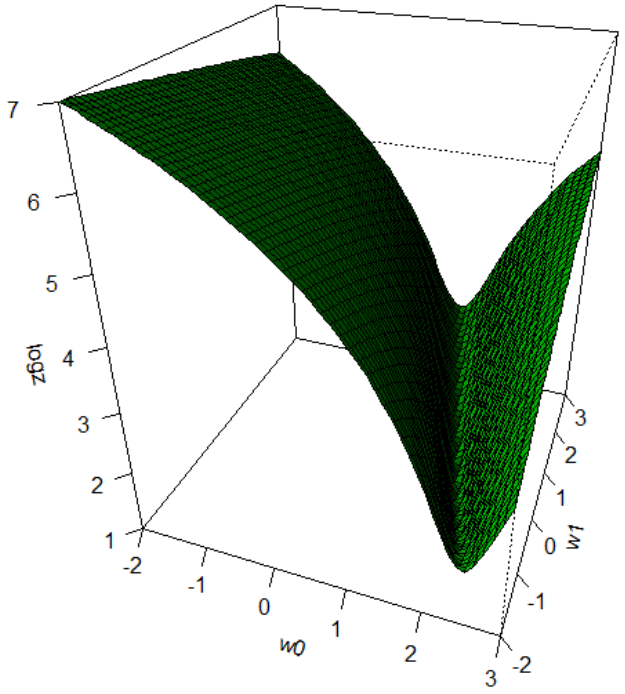
Search the Error surface using Calculus (gradient descent) but must be smooth if possible: Tensorflow, Caffe, Torch, Theano, etc

Brute Force Search of Weights

Enumerate all possible coefficient combinations; in our case (w_1, w_2)

- Select the weight combination that minimizes the error rate

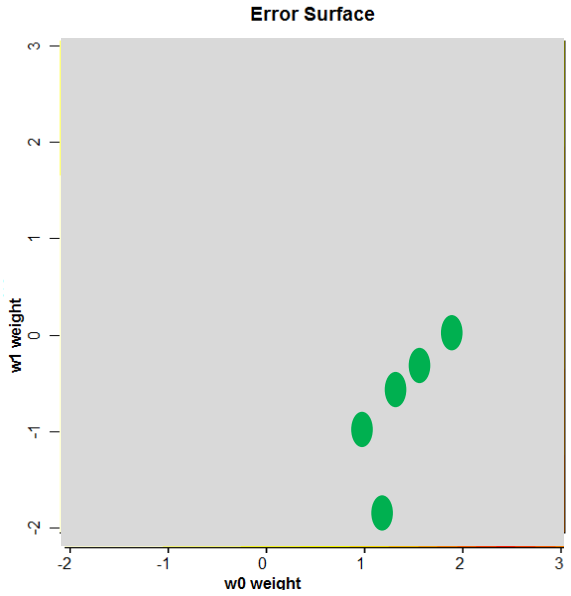


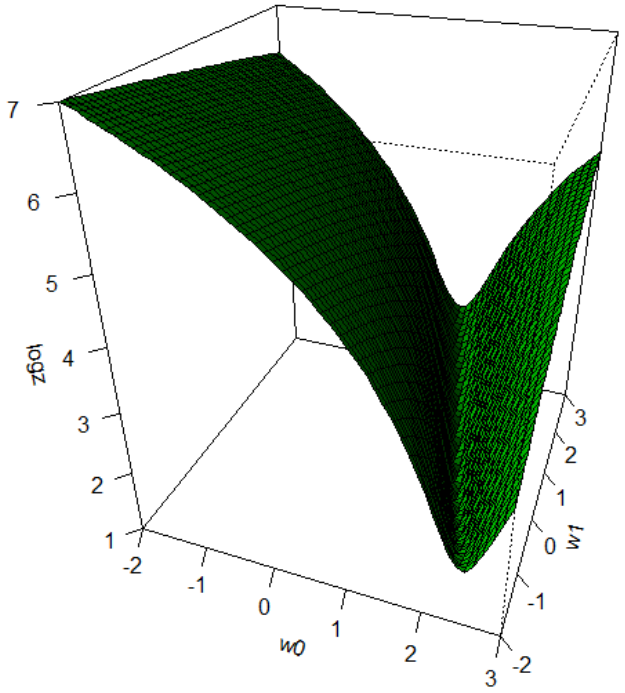


HeatMap with isolines overlayed

Brute Force Search of Weights

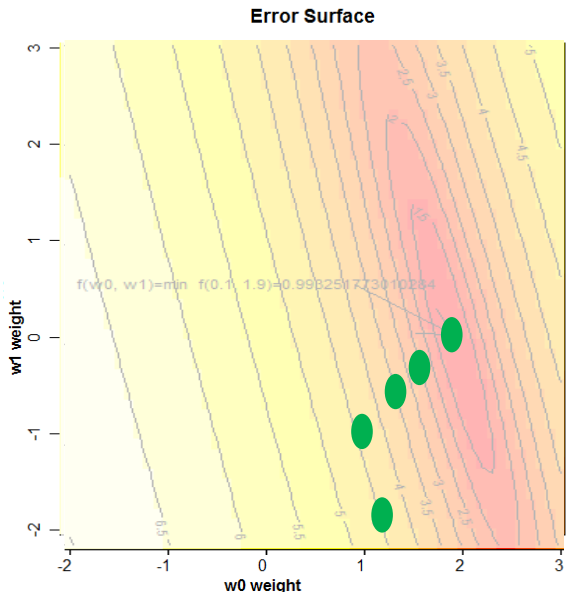
- Very inefficient; at best we can only approximate the surface
- Not scaleable
- Avoid this approach….. Use gradient descent instead

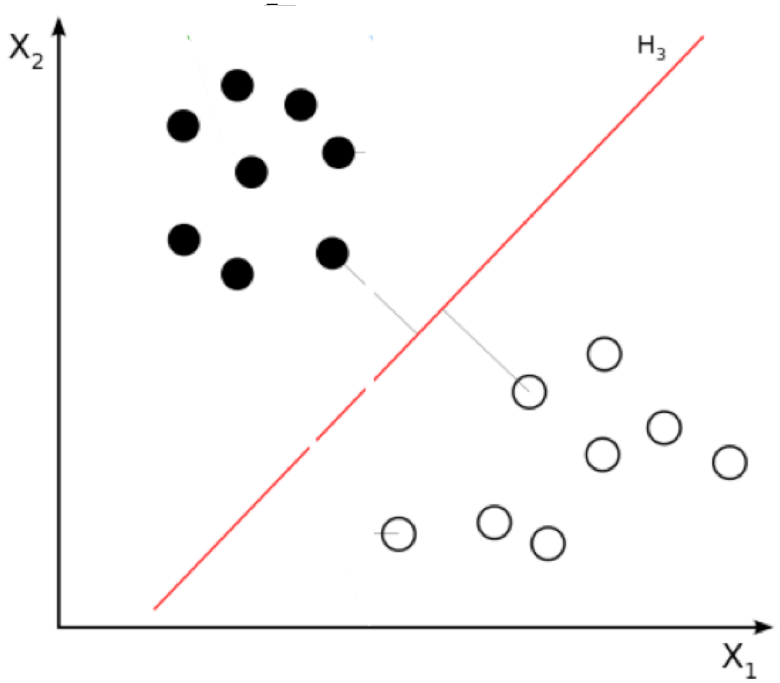




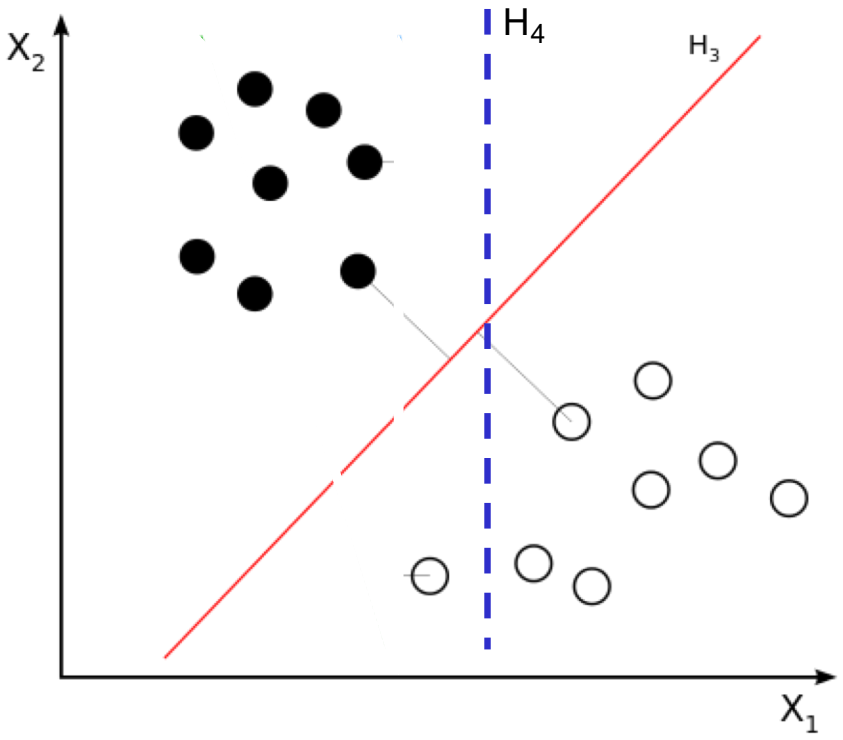
Brute Force Search of Weights vs. Gradient descent

- **Very inefficient; at best we can only approximate the surface**
  - Not scaleable
- **Avoid this approach….. Use gradient descent instead with a better error measure**





Binary Linear Classifier: split space into 2 pure halfspaces



- Data lives in n-space;
- Partition training data such that one class lives in one halfspace and the other in the other halfspace
  - 2 pure halfspaces

- Accuracy of H_3 is 16/16; 100%
- Accuracy of H_4 is 15/16; 93.75%

- Which hyperplane do you prefer?
  - H_3

H_1 does not separate the classes.
H_2 does, but only with a small margin.
H_3 separates them with the maximum margin.

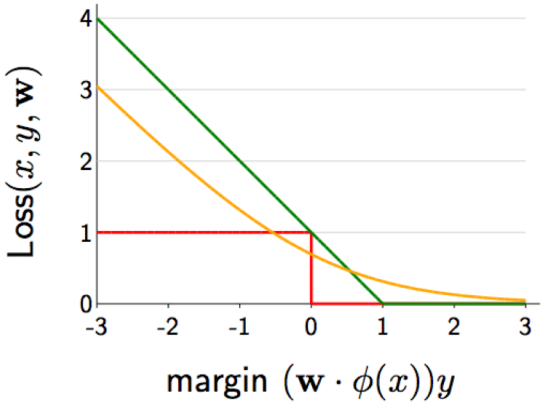
0-1 Loss versus Perceptron (aka hinge) loss surface

$$Loss_{0-1} = \frac{1}{N}\sum_{i=1}^{N} L(X_i^T W, y_i)$$

Where

- $L(X_i^T W, y_i) = 0$ if $heaviside(X_i^T W) == y_i$,
- $L(X_i^T W, y_i) = 1$ otherwise
- $where\ heaviside(X_i^T W) = \begin{cases} -1 & X_i^T W < 0 \\ 1 & X_i^T W \geq 0 \end{cases}$

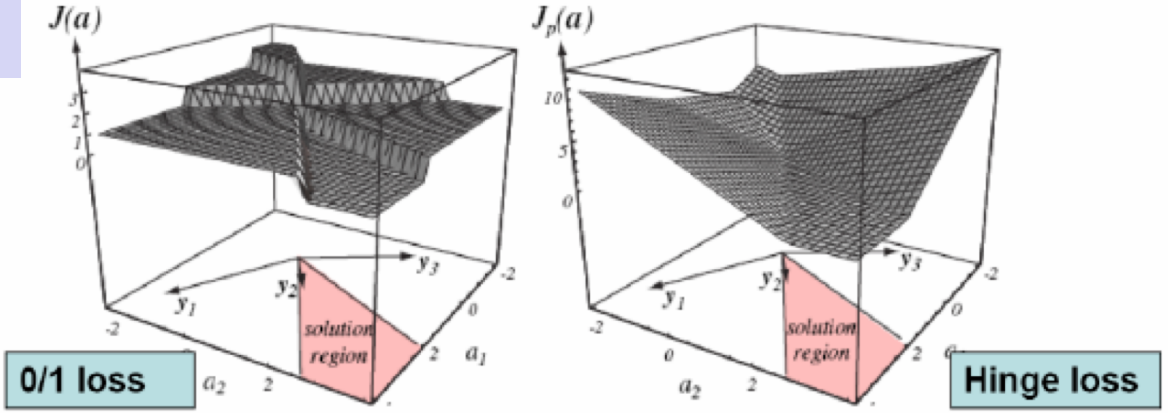
1. $Loss_{P\_Hinge}(W) = \frac{1}{N}\sum_{i=1}^{N} Max(0, -X_i^T W y_i)$
2. $Loss_{SVM\_Hinge}(W) = \frac{1}{N}\sum_{i=1}^{N} Max(0, 1 - X_i^T W y_i)$
3. $Loss_{Logistic}(W) = \frac{1}{N}\sum_{i=1}^{N} \log(1 + e^{(-X_i^T W y_i)})$



- Instead we will consider the "**hinge loss**":

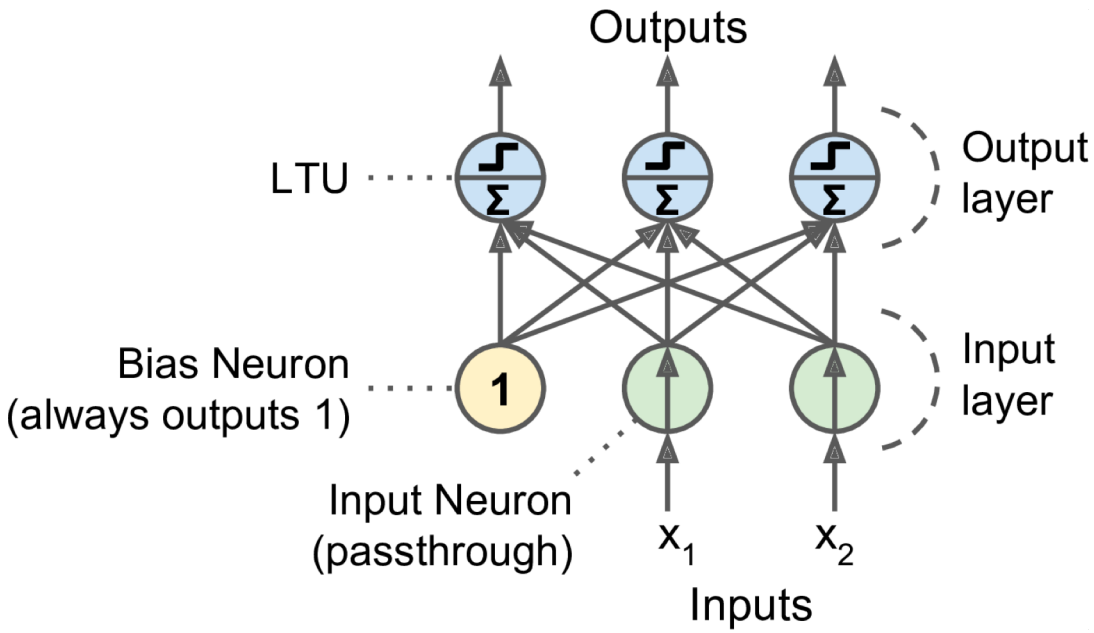
$$J_p(w) = \frac{1}{N}\sum_{i-1}^{N} \max(0, -y_i w \cdot x_i)$$

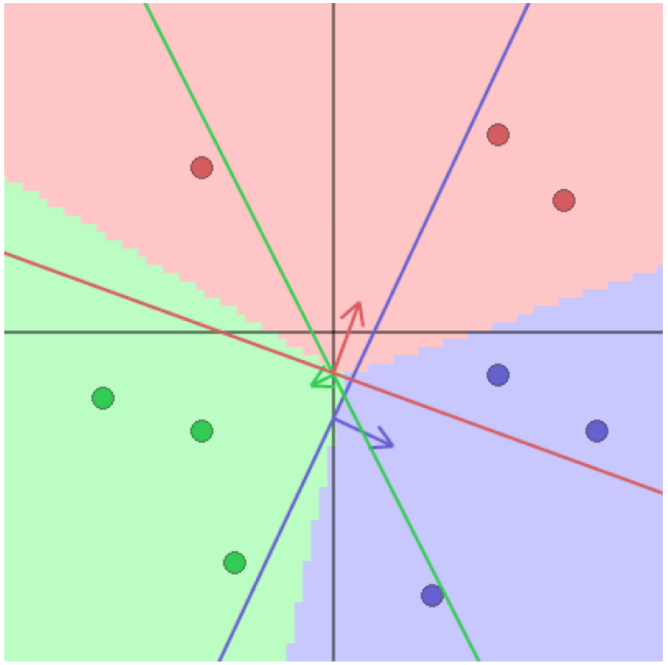
- The term $\max(0, -y_i w \cdot x_i)$ is 0 when y_i is predicted correctly otherwise it is equal to the "confidence" in the mis-prediction
  Has a nice gradient leading to the solution region



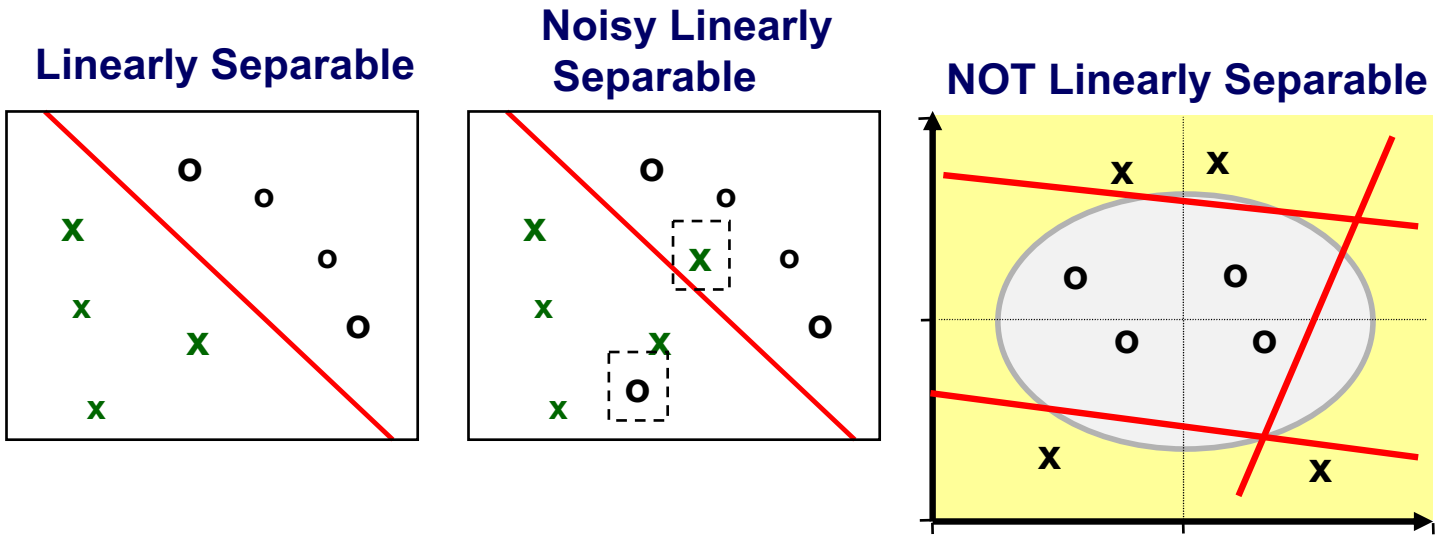
A 2-3 Multinomial Perceptron

- **Classification rule is similar to logistic regression**
  - For each class calculate the perpendicular distance
  - $y' = \text{argmax}_y (X^T w_y)$. #class with largest perdDist



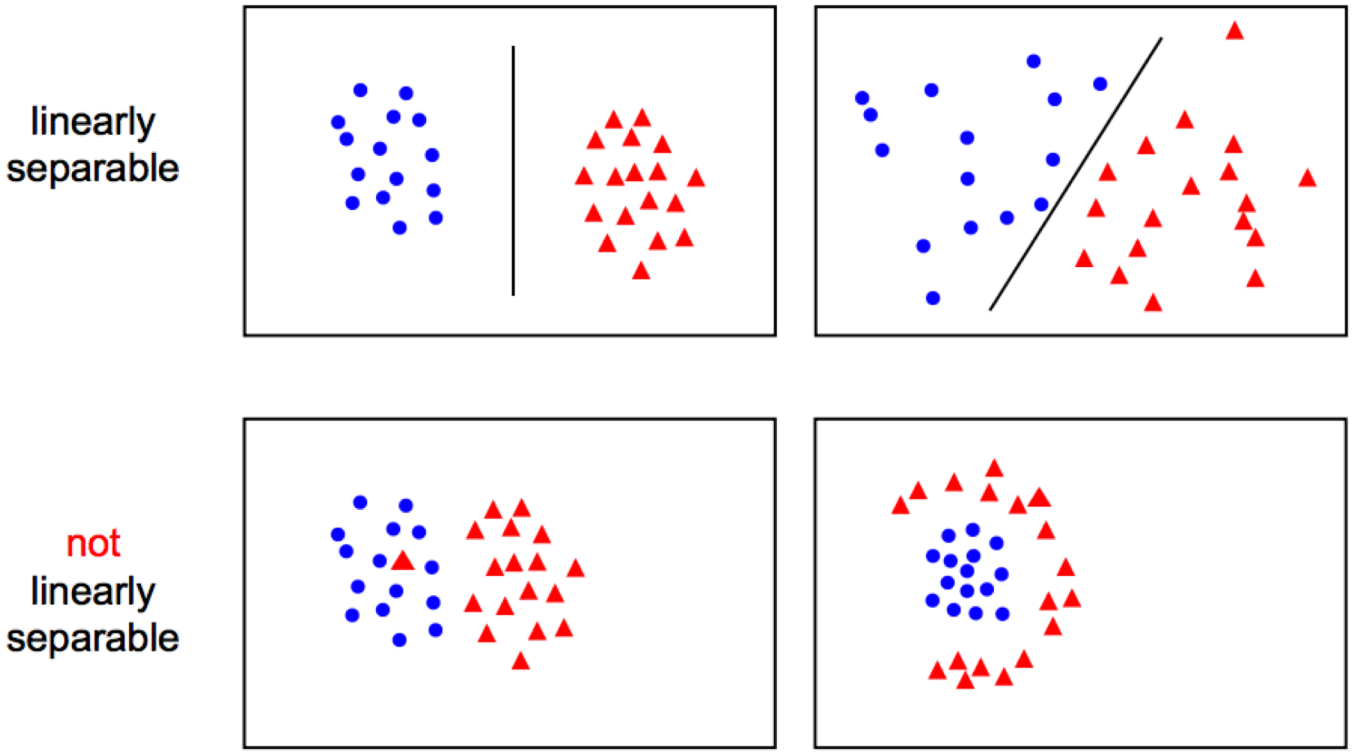


Non-Linearly Separable Data



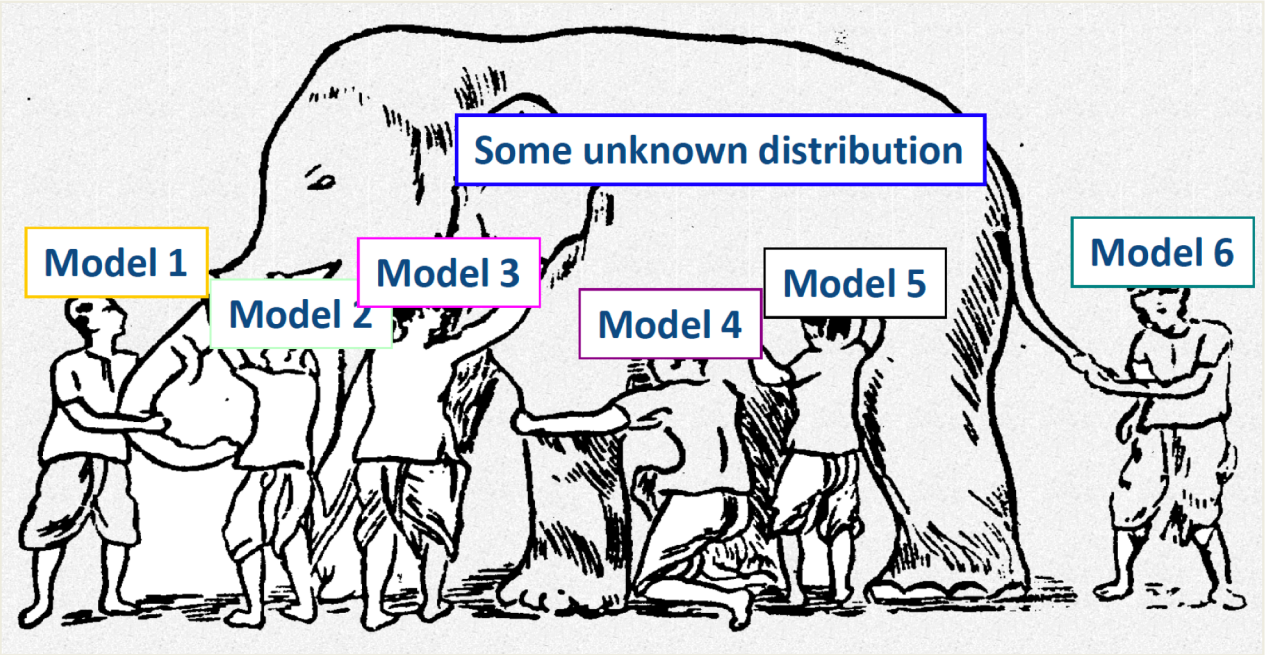
linearly separable data:
Use Hard/Soft SVMs;
Noisy linearly separable data:
Use Soft SVMs;
Text is generally linearly separable but is sometimes noisy
Non-Linearly Separable Data: Use kernels

Need more high powered models: e.g., deep learning



Strength in numbers

The main idea behind the ensemble methodology is to **combine several individual base classifiers** in order to have a classifier that outperforms everyone of them

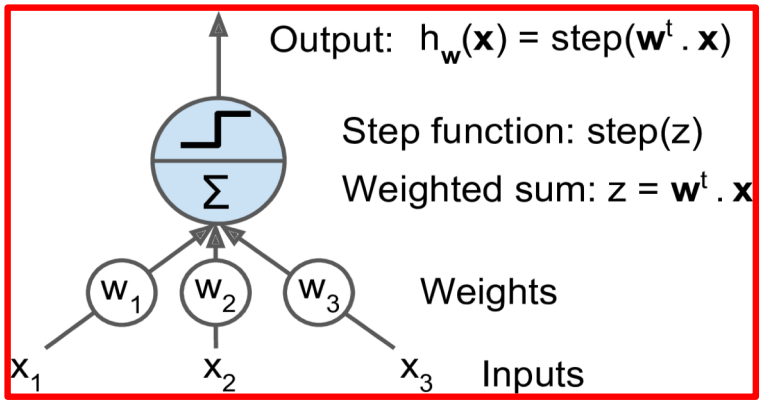


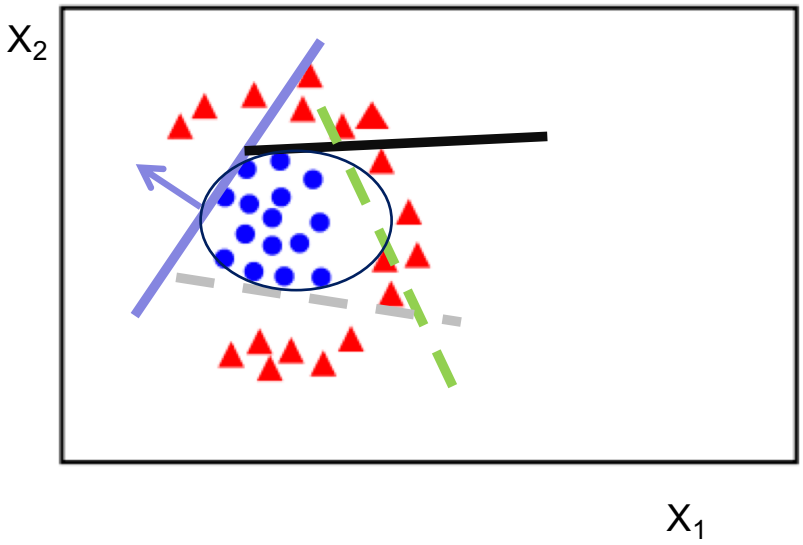
"The Blind Men and the Elephant", Godfrey Saxe's

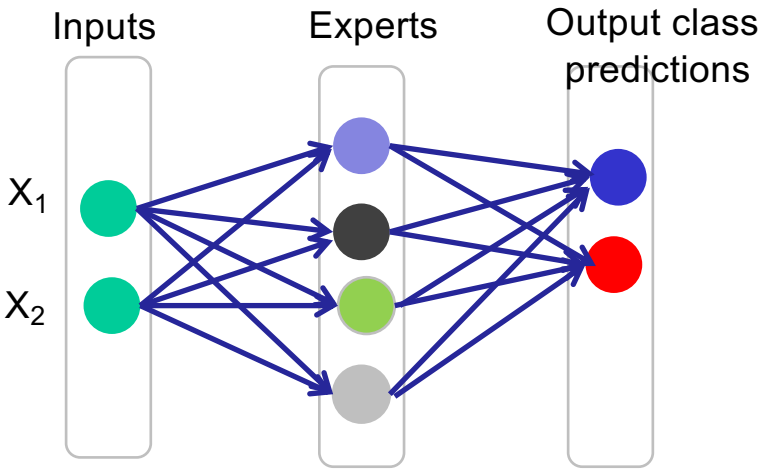
Single model versus an ensemble

$$f(x) = \mathrm{sgn}(X_i^T W)$$

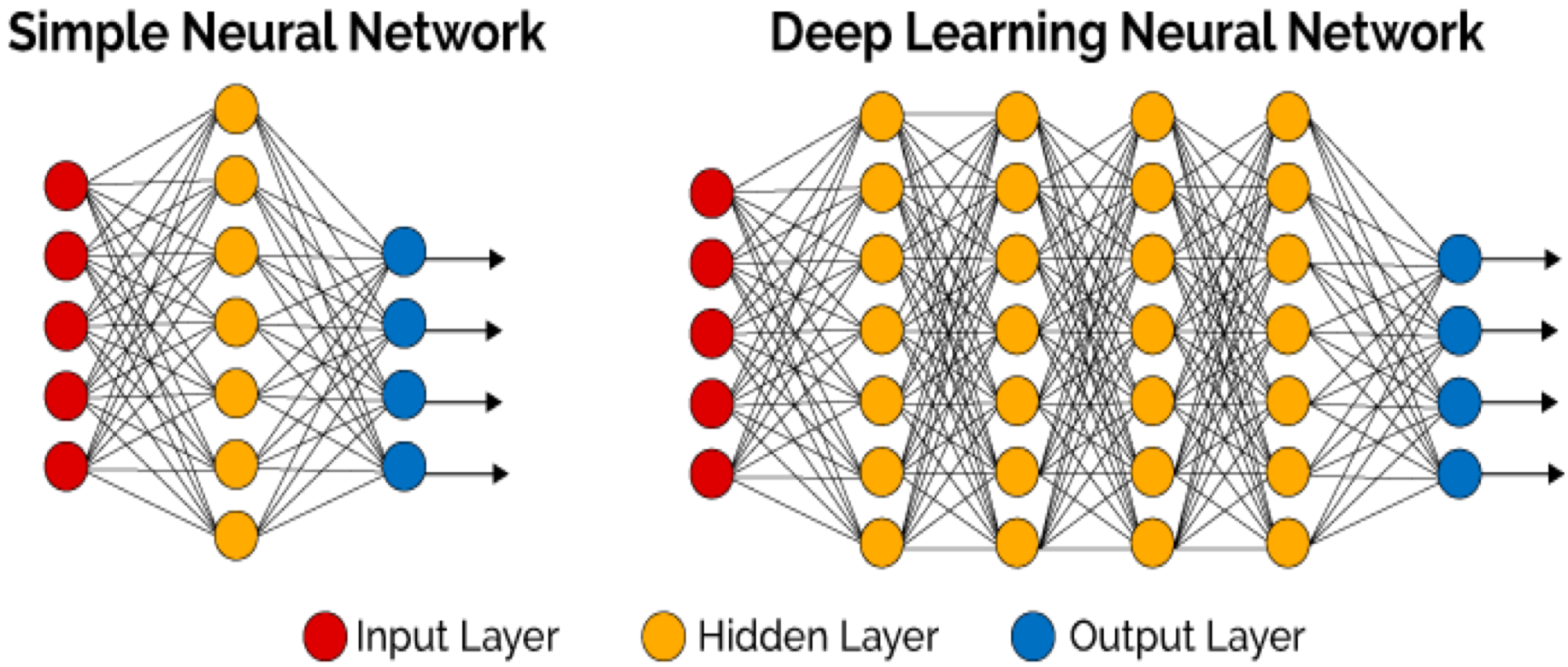
$$where\ sgn(X_i^T W) = \begin{cases} -1 & X_i^T W < 0 \\ 1 & X_i^T W = 0 \\ 1 & X_i^T W > 0 \end{cases}$$







Deeper neural networks

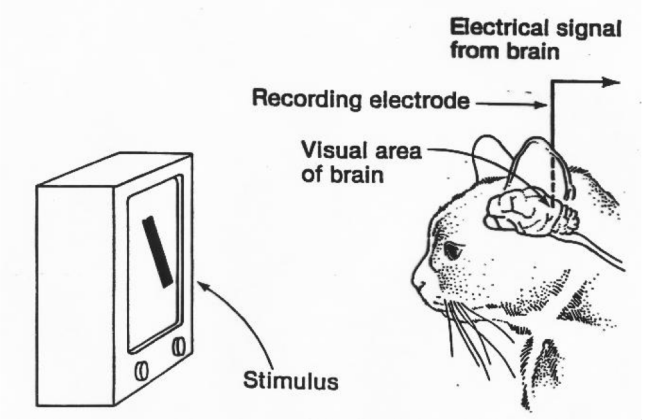


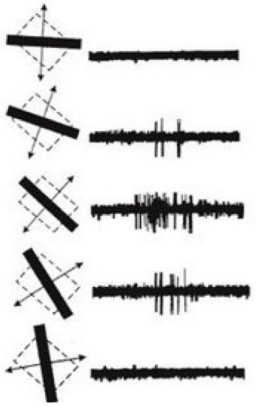
1 hidden layers in the network

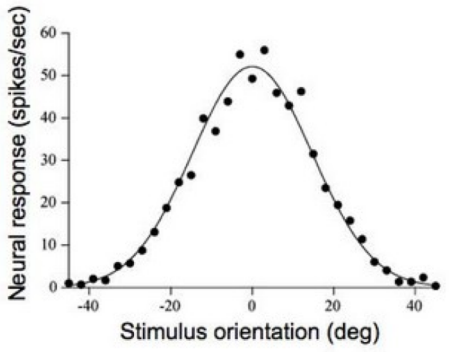
4 hidden layers in the network

Gained insights on the structure of the visual cortex

Hubel & Wiesel,
1959
RECEPTIVE FIELDS OF
SINGLE NEURONES IN
THE CAT'S STRIATE CORTEX
1962
RECEPTIVE FIELDS,
BINOCULAR INTERACTION
AND FUNCTIONAL
ARCHITECTURE IN
THE CAT'S VISUAL CORTEX
(and later in monkeys
1968...

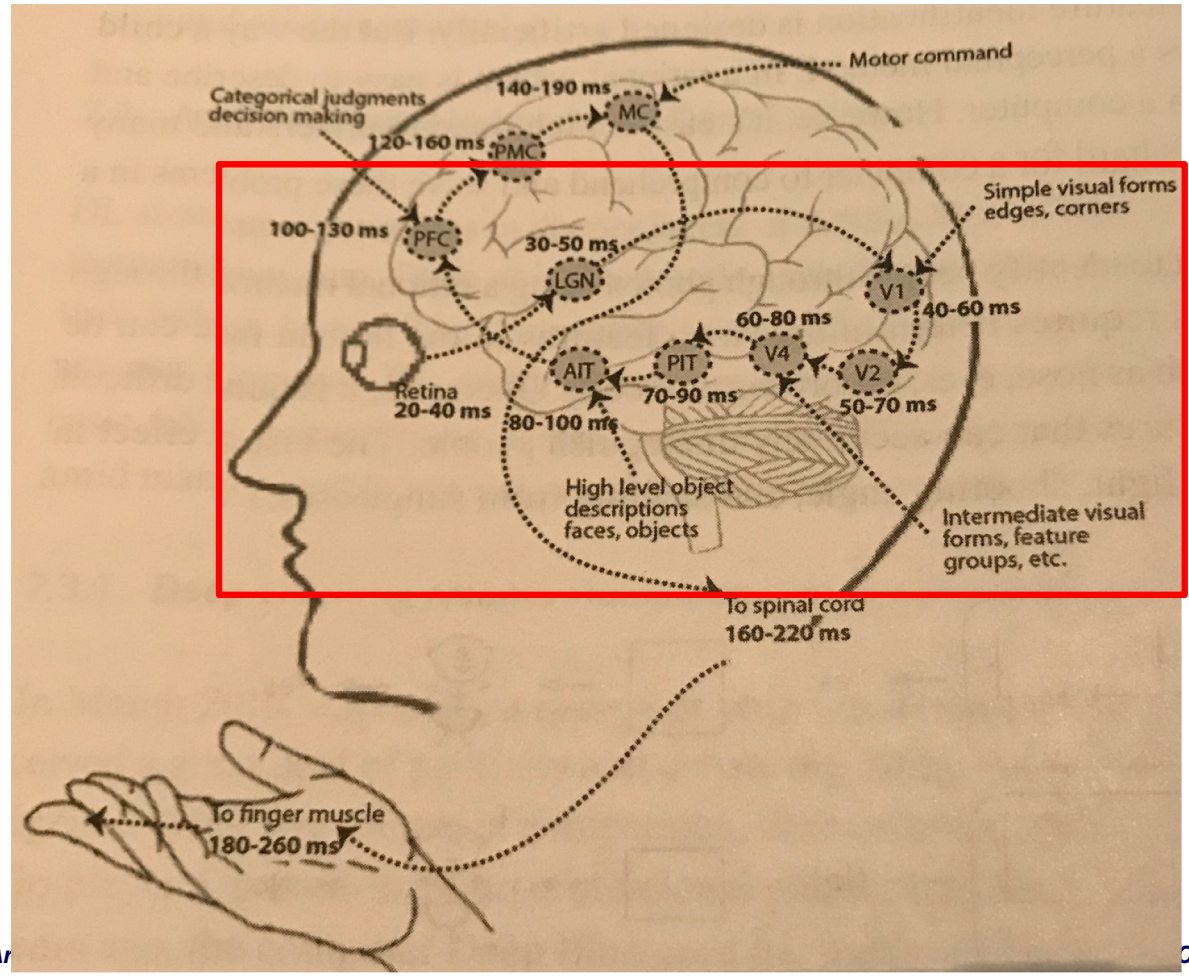






One neuron in the V1 area of the cat brain; orientation

Brain: hierarchy of neurons

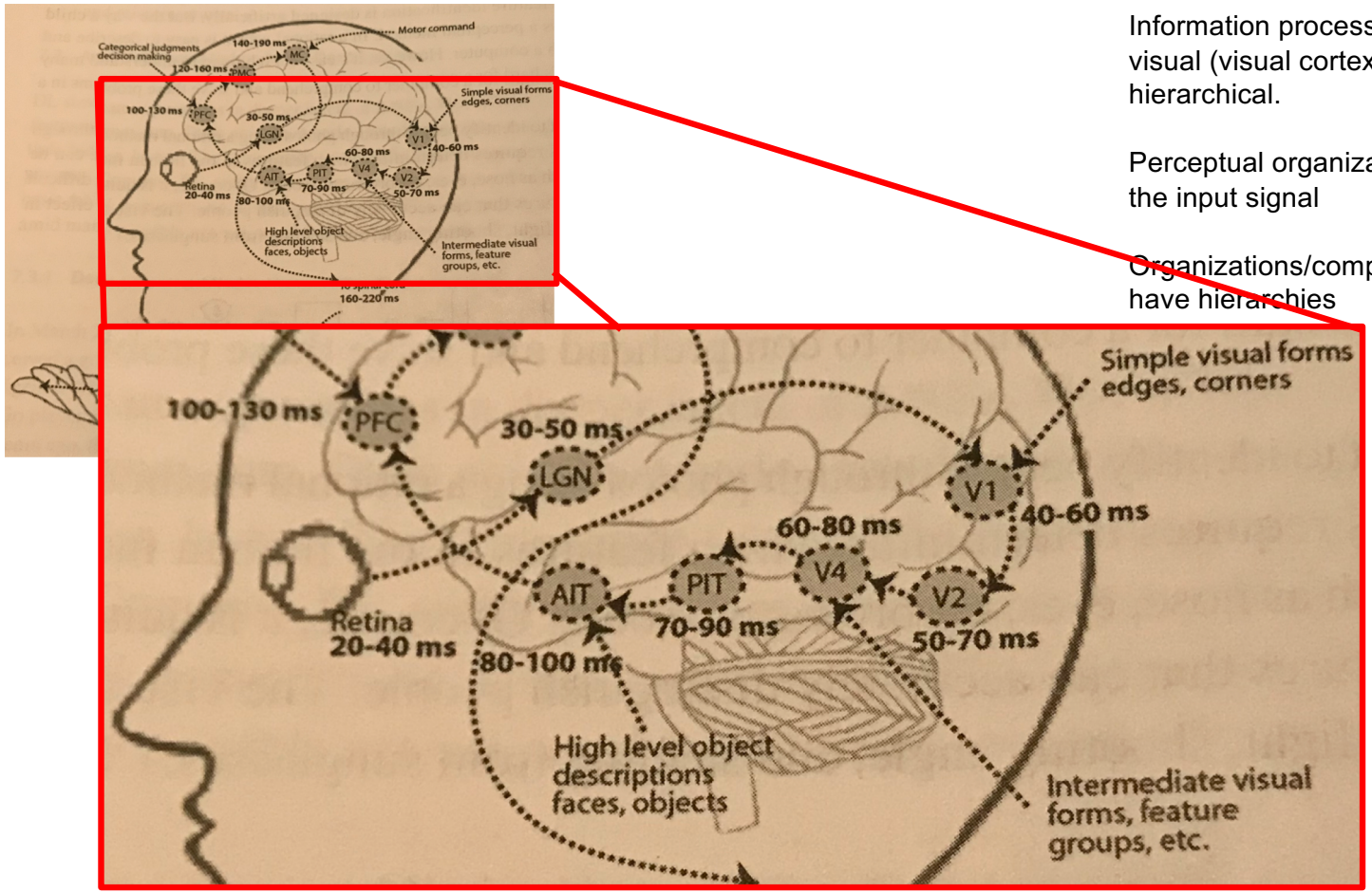


Information processing of visual (visual cortex) is hierarchical.

Perceptual organization of the input signal

Organizations/companies have hierarchies

Brain: hierarchy of neurons



Information processing of visual (visual cortex) is hierarchical.

Perceptual organization of the input signal

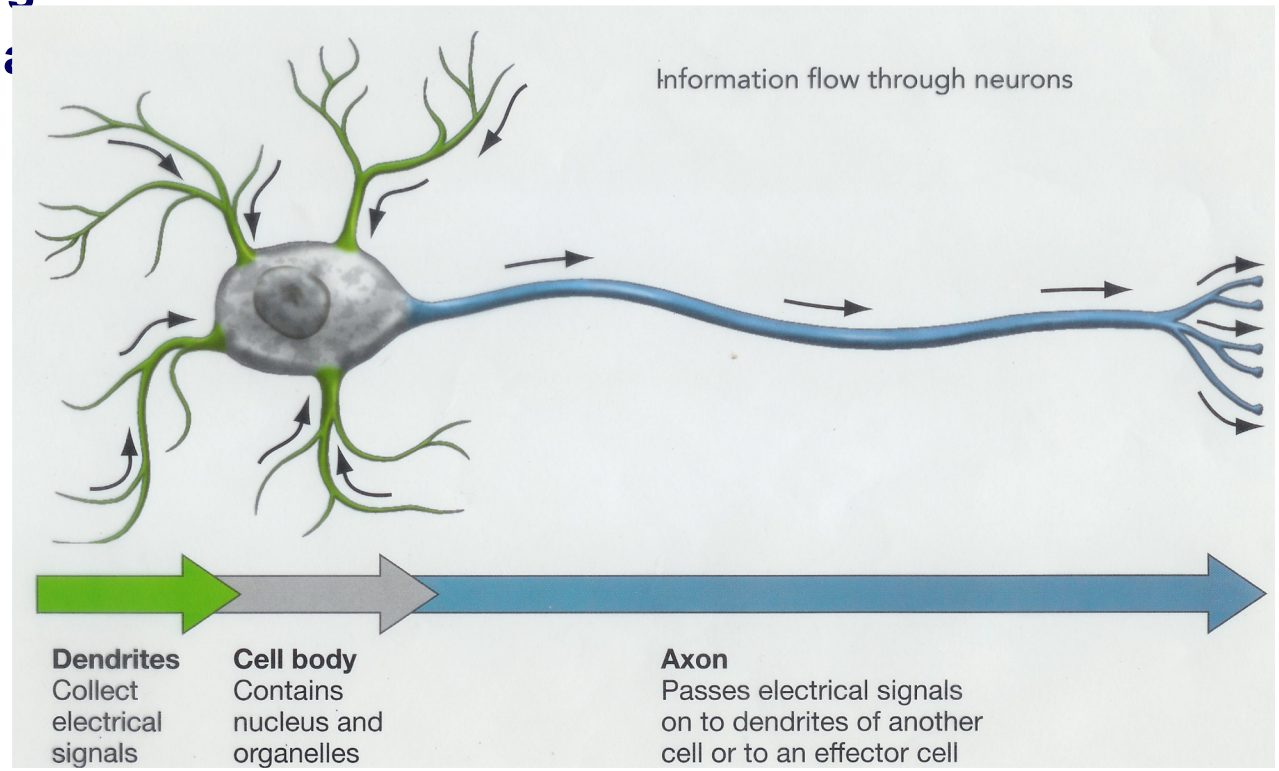
Organizations/companies have hierarchies

Outline

1. **Introduction**
2. **ML and deep learning review**
    1. Perceptron → MLP
    2. Keras
    3. BackPropagation
3. **What is computer vision?**
    1. Computer vision
    2. Convolutional Neural Nets (CNNs)
4. **Deep NN Architectures for CV Tasks**
    1. Backbone networks
5. **Solving edge-based IoT**
6. **Conclusions and Next steps**

Information processing in neurons

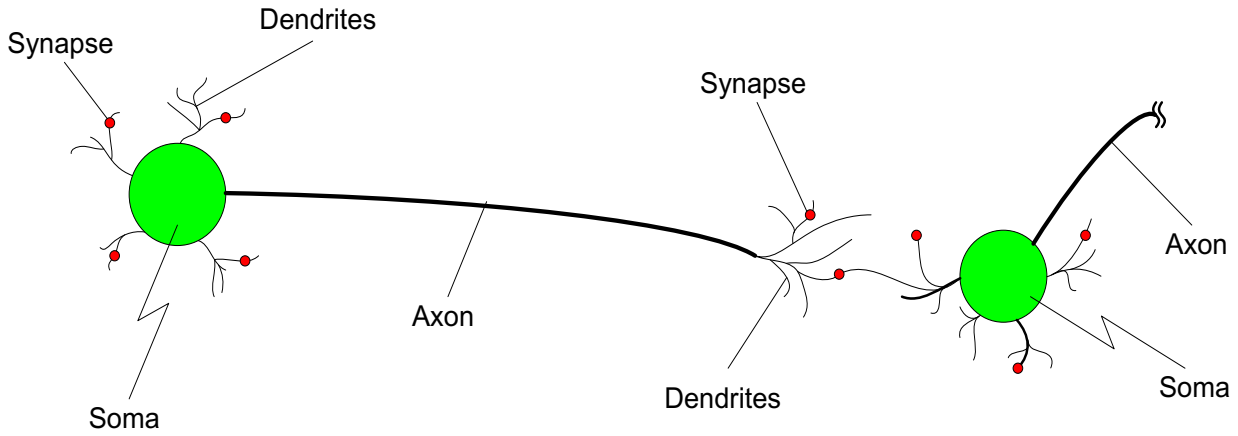
- **Biological "basis" for neural networks**
- **Funda**



Biological “basis” for neural networks

- The brain is what it is because of the structural and functional properties of interconnected neurons.
- The mammalian brain contains between 100 million and 100 billion neurons, depending on the species. Each mammalian neuron consists of a cell body, dendrites, and an axon.
- The **neuron** is the basic working unit of the **brain**, a specialized cell designed to transmit information to other nerve cells, muscle, or gland cells.
- **Neurons** are cells within the nervous system that transmit information to other nerve cells, muscle, or gland cells.
  - Most neurons have a cell body, an axon, and dendrites.

Parallels between Biological NN and ANN

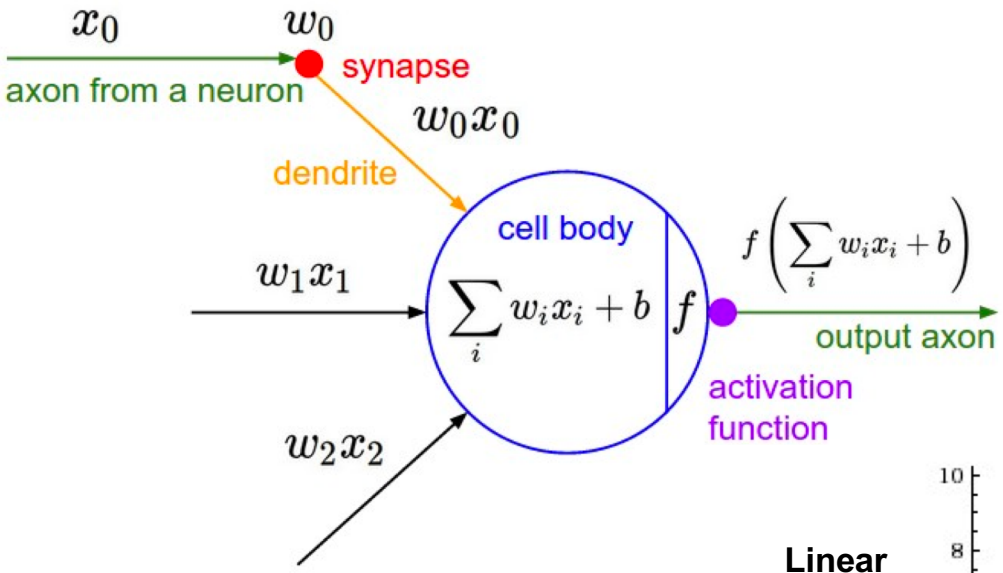


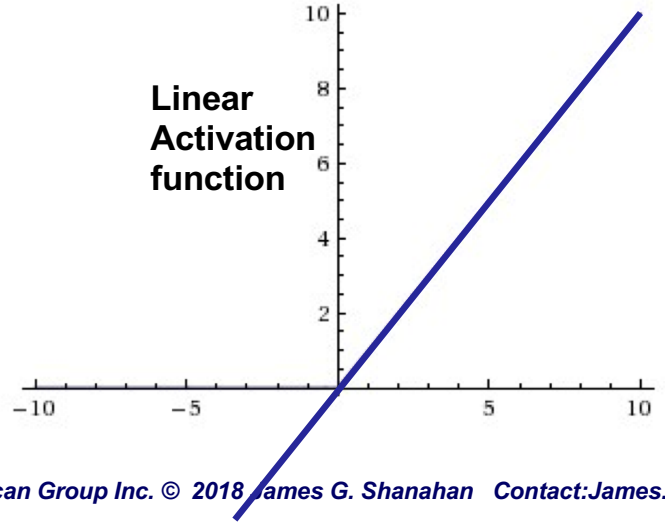
- **Two interconnected brain cells**

Biological	versus	Artificial	NNs
Soma		Node	
Dendrites		Input	
Axon		Output	
Synapse		Weight	
Slow		Fast	
Many neurons (10^9)		Few neurons (~100s)	

- Electrical inputs are passed through networks of neurons, and results in an output
- The dendrites of the neuron collect the inputs, and when it reaches its firing threshold (action potential), the neuron fires (depolarizes).
- 100,000,000,000 neurons in the human

Activation functions





Brain cells, aka neurons

- Brain cells, aka neurons (100 billion)
- Synapses (100 trillion connections)
-

- Parents are the Frontal lobes for kids (leadership, decision)
- Last organ to mature (late 20s)
-

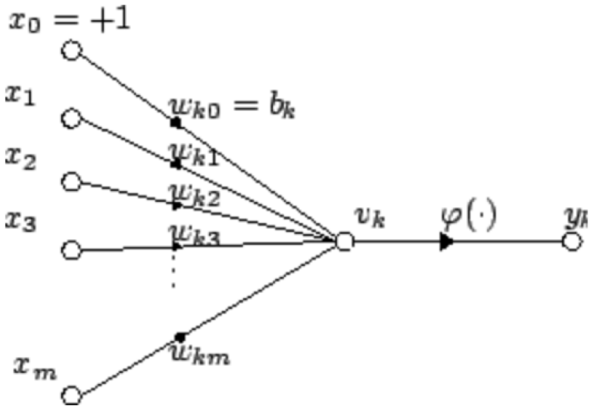
Artificial Neuron

For a given artificial neuron, let there be $m + 1$ inputs with signals x_0 through x_m and weights w_0 through w_m. Usually, the x_0 input is assigned the value +1, which makes it a *bias* input with $w_{k0} = b_k$. This leaves only m actual inputs to the neuron: from x_1 to x_m.

The output of the kth neuron is:

$$y_k = \varphi \left(\sum_{j=0}^{m} w_{kj} x_j \right)$$

Where φ (phi) is the transfer function.



The output is analogous to the axon of a biological neuron, and its value propagates to the input of the next layer, through a synapse. It may also exit the system, possibly as part of an output vector.

Perceptron Roller-coaster

"Scruffies" Rise/Fall/Rise/Fall/Rise

http://www.nature.com/polopoly_fs/7.14689.1389093731!/image/deep-learning-graphic.jpg_gen/derivatives/landscape_400/deep-learning-graphic.jpg

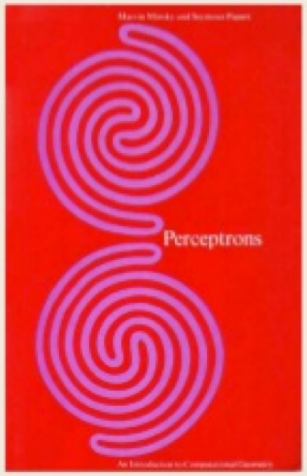
http://www.rutherfordjournal.org/article040101.html

http://opticalengineering.spiedigitallibrary.org/article.aspx?articleid=1714547

1957 Rosenblatt Perceptron

"The embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence."

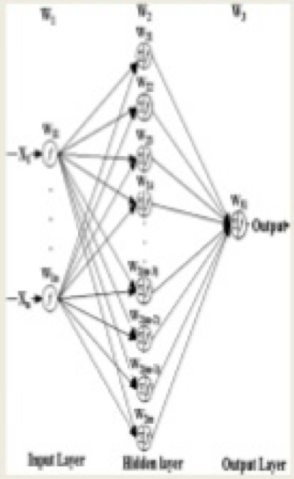
https://en.wikipedia.org/wiki/Perceptron



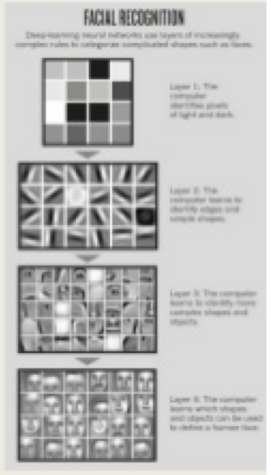
1969 Minsky & Papert
Can't do XOR!

https://constructingkids.files.wordpress.com/2013/05/minsky-papert-71-csolomon-x640.jpg

http://www.i-programmer.info/images/stories/BabBag/AI/book.jpg



Backpropagation
1986 Rumelhart
(1963 Bryson
1974 Werbos)

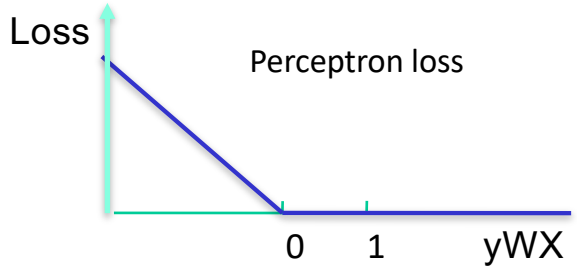


Deep Learning
2007 Hinton
(1989 LeCun
1992 Schmidhuber)

Artificial Neurons

- **1943: The McCulloch & Pitts (perceptron) neuron**
  - Weighted sum of inputs
  - Activation Function
    - "Link" function in GLM/regression (e.g., sigmoid or hyperbolic tangent link for logistic regression)
- **1957: Perceptron learning algo (Rosenblatt 1957)**
  - Learning rate and learning (update) rule
  - Classification problems
  - Neural network as a network of logistic regressions

Penalizes incorrectly classified examples x proportionally to the size of |w'x|



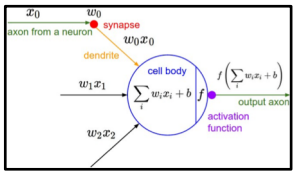
Perceptron: No loss function

The **Mark I Perceptron** machine was the first implementation of the perceptron algorithm.

The machine was connected to a camera that used 20×20 cadmium sulfide photocells to produce a 400-pixel image.

recognized
letters of the alphabet

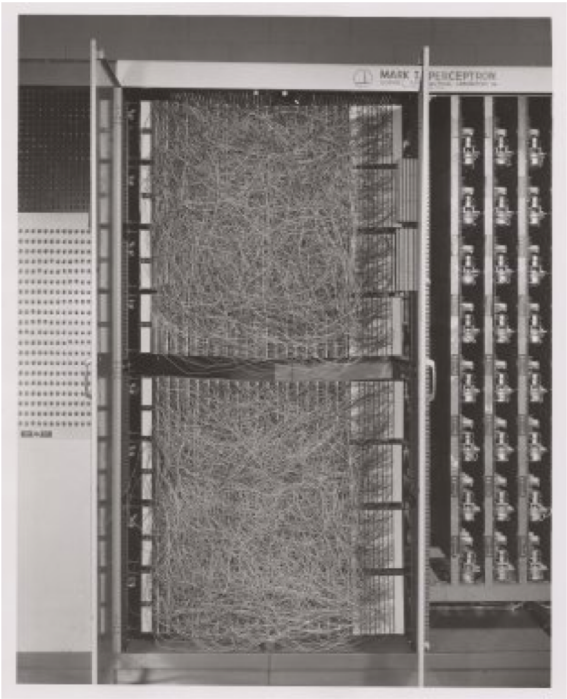
$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$



update rule:

$$w_i(t+1) = w_i(t) + \alpha(d_j - y_j(t))x_{j,i},$$

Frank Rosenblatt, ~1957: Perceptron



No loss function; could not use gradient descent
Linear models only

The perceptron was intended to be a machine, rather than a program, and while its first implementation was in software for the IBM 704, it was subsequently implemented in custom-built hardware as the "Mark 1 perceptron".

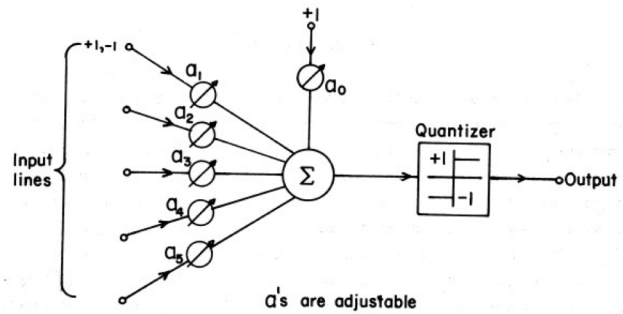
63

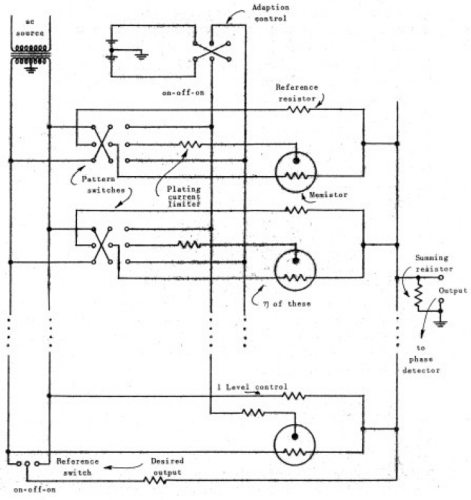
Perceptron machine: updates performed by electric motors

- The perceptron algorithm was invented in 1957 at the Cornell Aeronautical Laboratory by Frank Rosenblatt,[3] funded by the United States Office of Naval Research.[4]
- The perceptron was intended to be a machine, rather than a program, and while its first implementation was in software for the IBM 704, it was subsequently implemented in custom-built hardware as the "Mark 1 perceptron".
- This machine was designed for image recognition: it had an array of 400 photocells, randomly connected to the "neurons". Weights were encoded in potentiometers, and weight updates during learning were performed by electric motors

A bit of history

Hardware based
Stacked Perceptron
MLP





Widrow and Hoff, ~1960: Adaline/Madaline

65