Machine Learning for Vision


ITCH


ONE
WAY
macy's

Classification:

Does this image contain a building? [yes/no]

Classification:

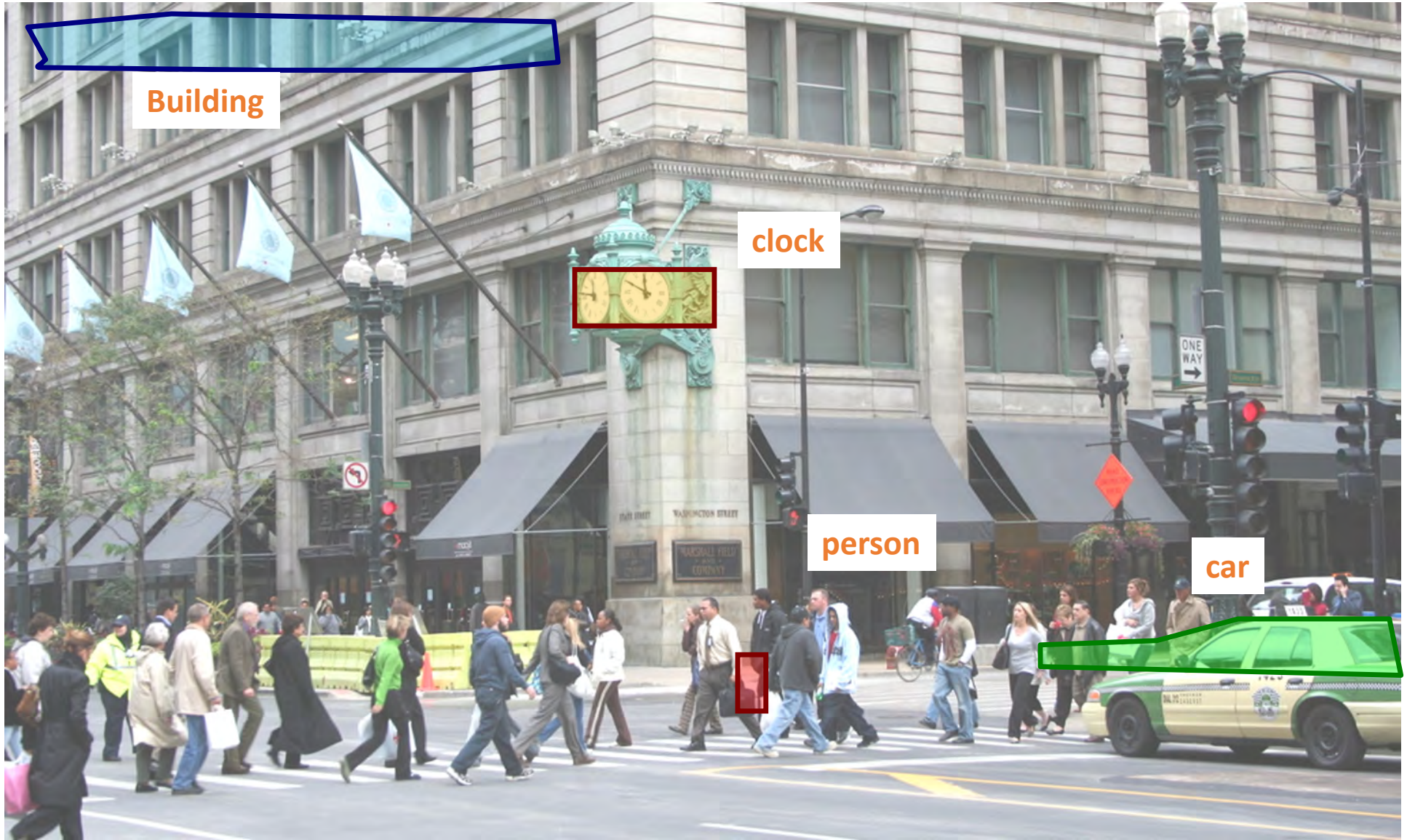
Is this an beach?

Detection:

Does this image contain a car? [where?]

Detection:

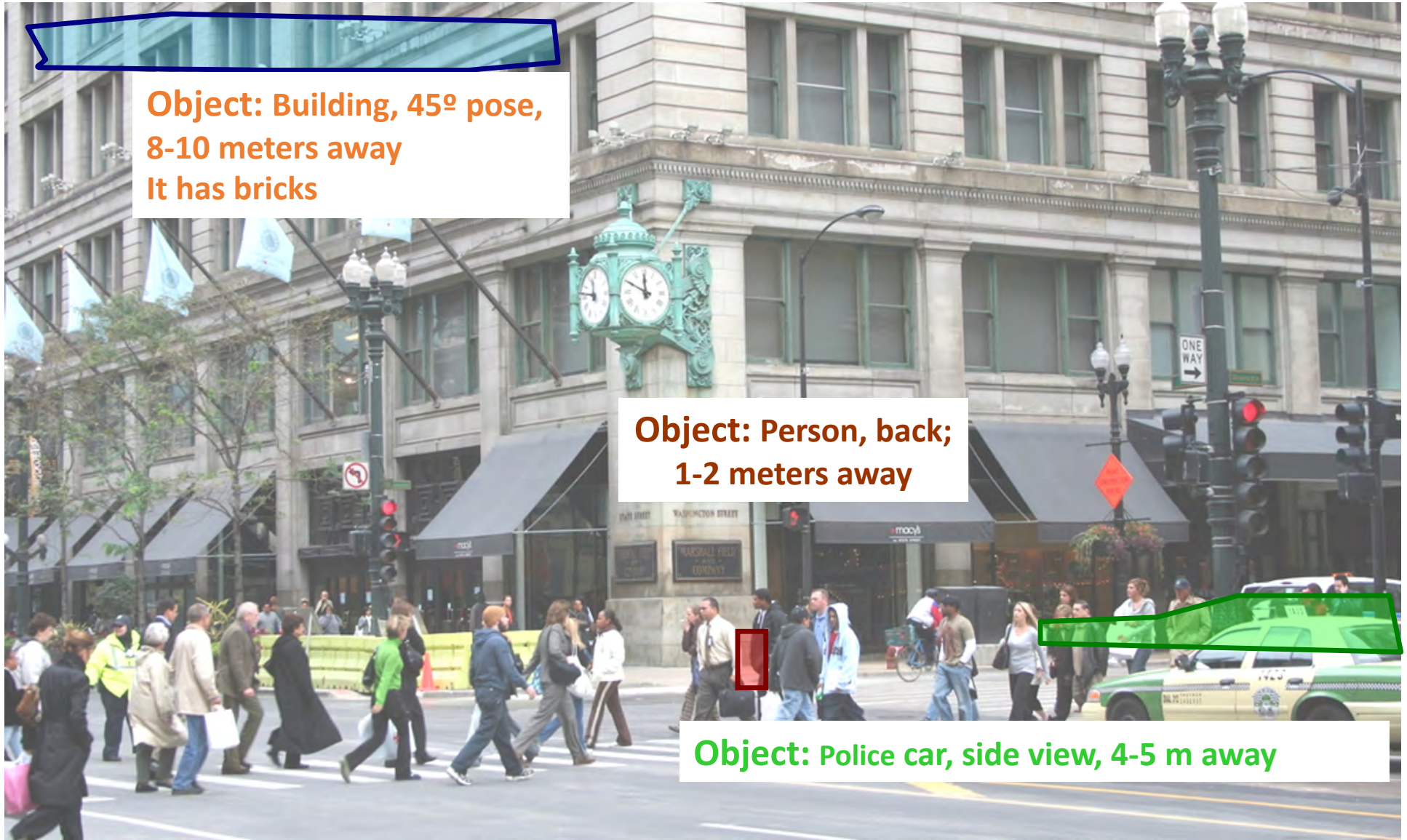
Which object does this image contain? [where?]



Detection:

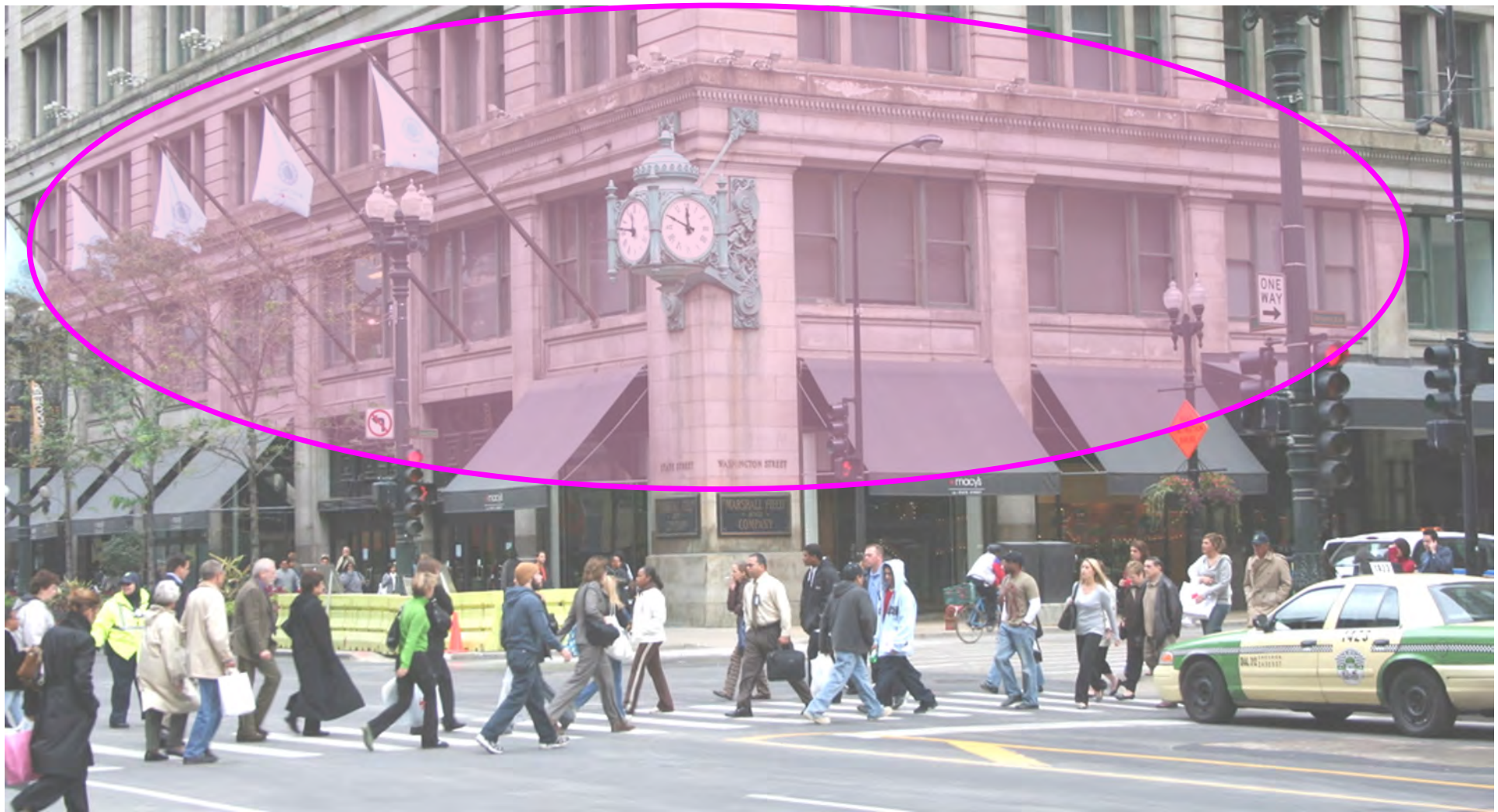
Accurate localization (segmentation)

Detection: Estimating object semantic & geometric attributes



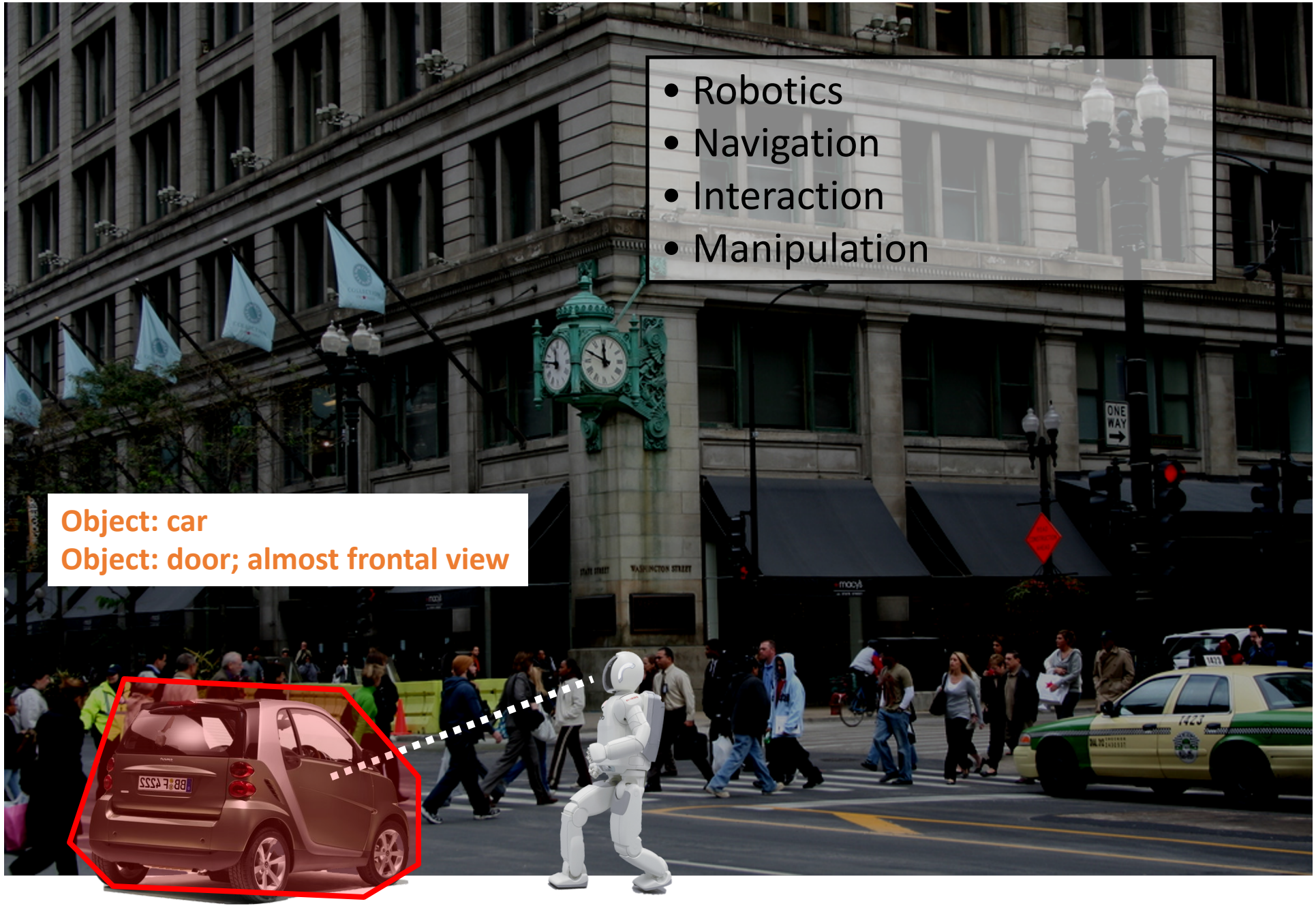
Categorization vs Single instance recognition

Which building is this? *Marshall Field* building in Chicago



Categorization vs Single instance recognition

Where is the crunchy nut?


• Robotics
• Navigation
• Interaction
• Manipulation
Object: car
Object: door; almost frontal view
ONE WAY

Activity or Event recognition

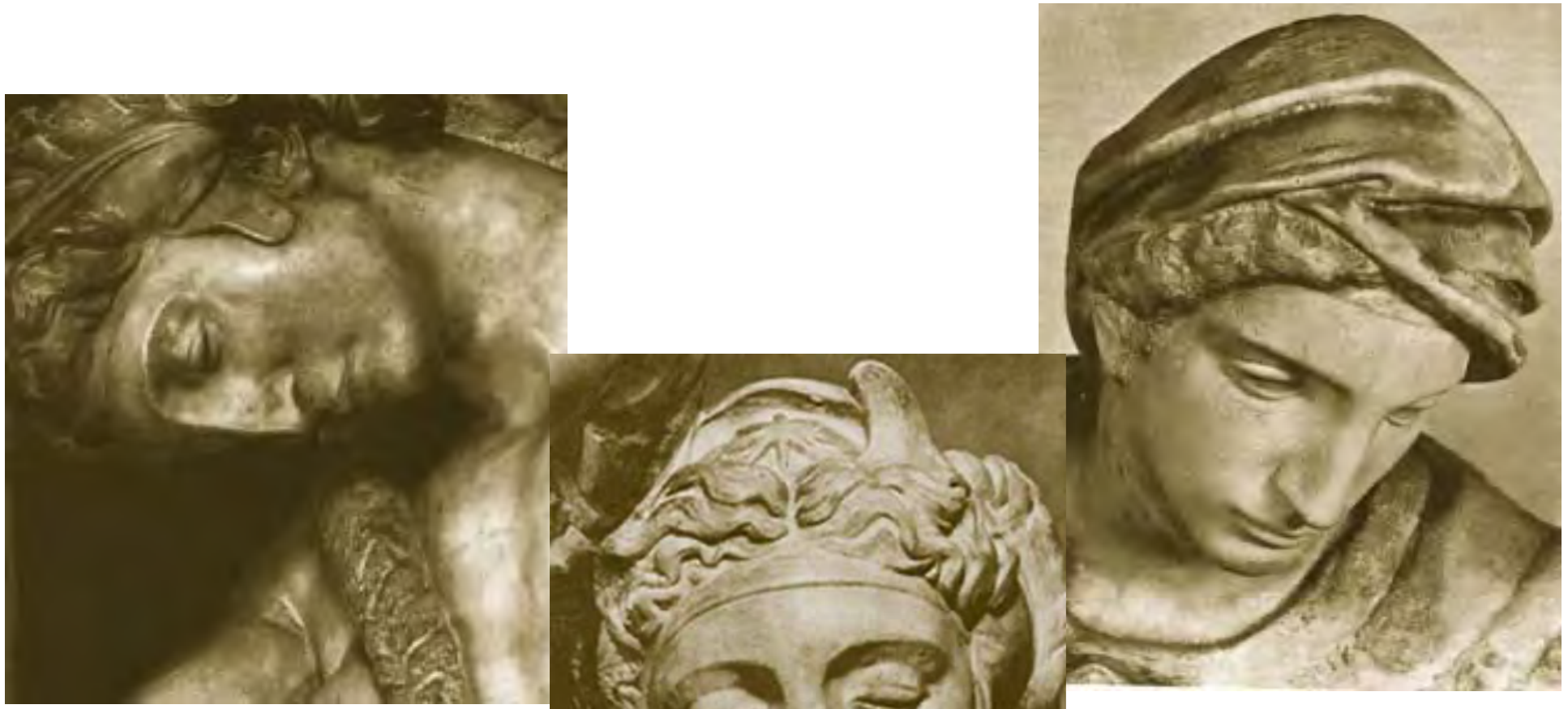
What are these people doing?

Visual Recognition

- Design algorithms that are capable to
  - Classify images or videos
  - Detect and localize objects
  - Estimate semantic and geometrical attributes
  - Classify human activities and events

Why is this challenging?

Challenges: viewpoint variation



Michelangelo 1475-1564

slide credit: Fei-Fei, Fergus & Torralba

Challenges: illumination

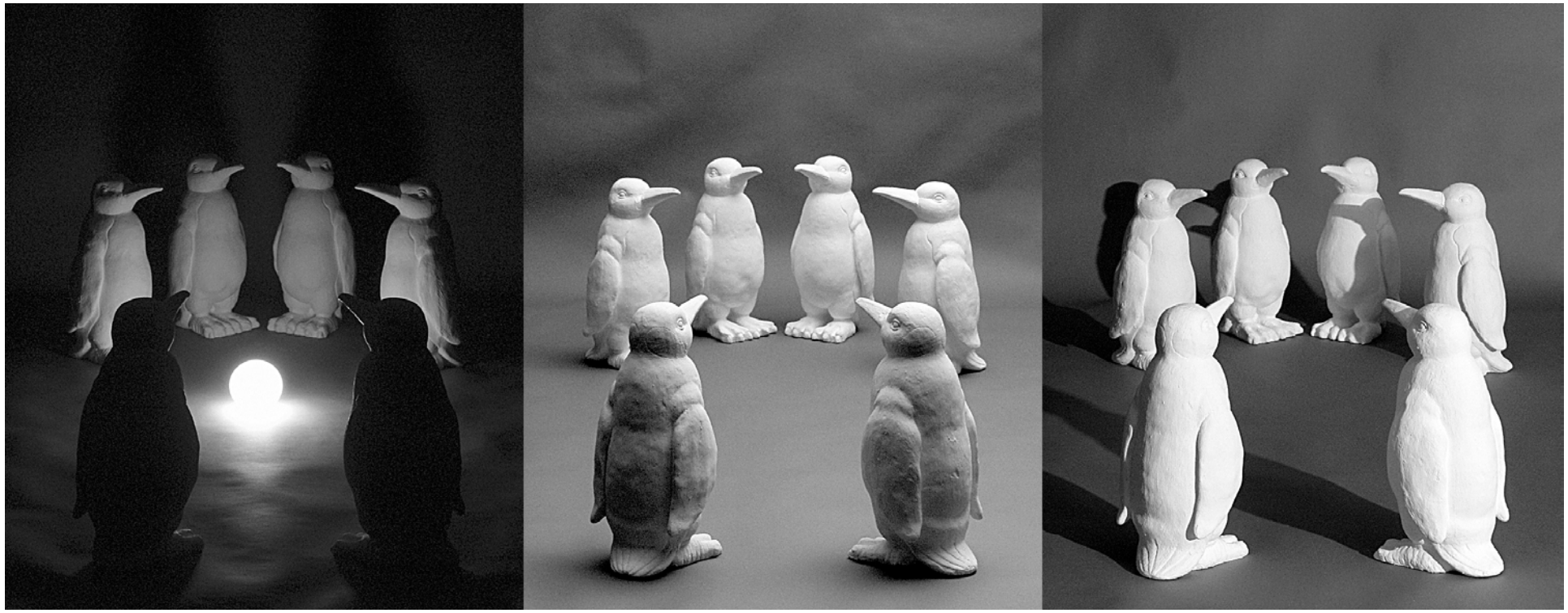


image credit: J. Koenderink

Challenges: scale

slide credit: Fei-Fei, Fergus & Torralba

Challenges: deformation

Challenges: occlusion

Magritte, 1957

slide credit: Fei-Fei, Fergus & Torralba

Challenges: background clutter

Kilmeny Niland. 1995

Challenges: intra-class variation

Some early works on object categorization

- Turk and Pentland, 1991
- Belhumeur, Hespanha, & Kriegman, 1997
- Schneiderman & Kanade 2004
- Viola and Jones, 2000

- Amit and Geman, 1999
- LeCun et al. 1998
- Belongie and Malik, 2002

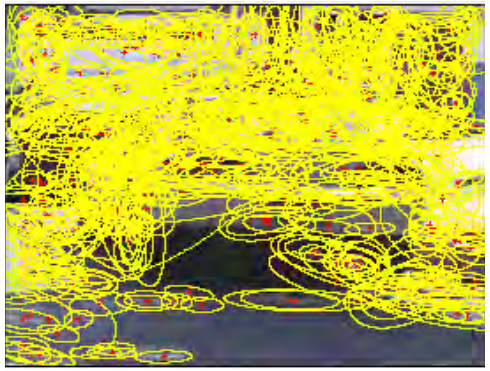
- Schneiderman & Kanade, 2004
- Argawal and Roth, 2002
- Poggio et al. 1993

Basic properties

- Representation
  - How to represent an object category; which classification scheme?
- Learning
  - How to learn the classifier, given training data
- Recognition
  - How the classifier is to be used on novel data

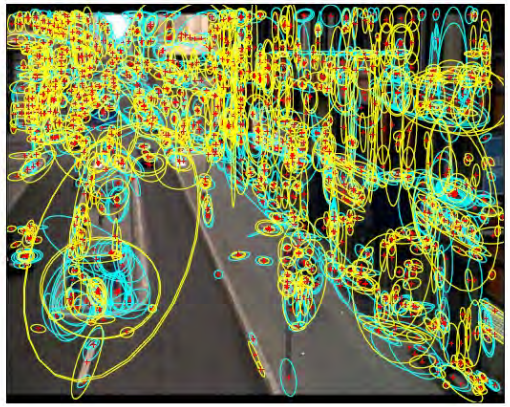
Representation

- Building blocks: Sampling strategies



Interest operators

Dense, uniformly



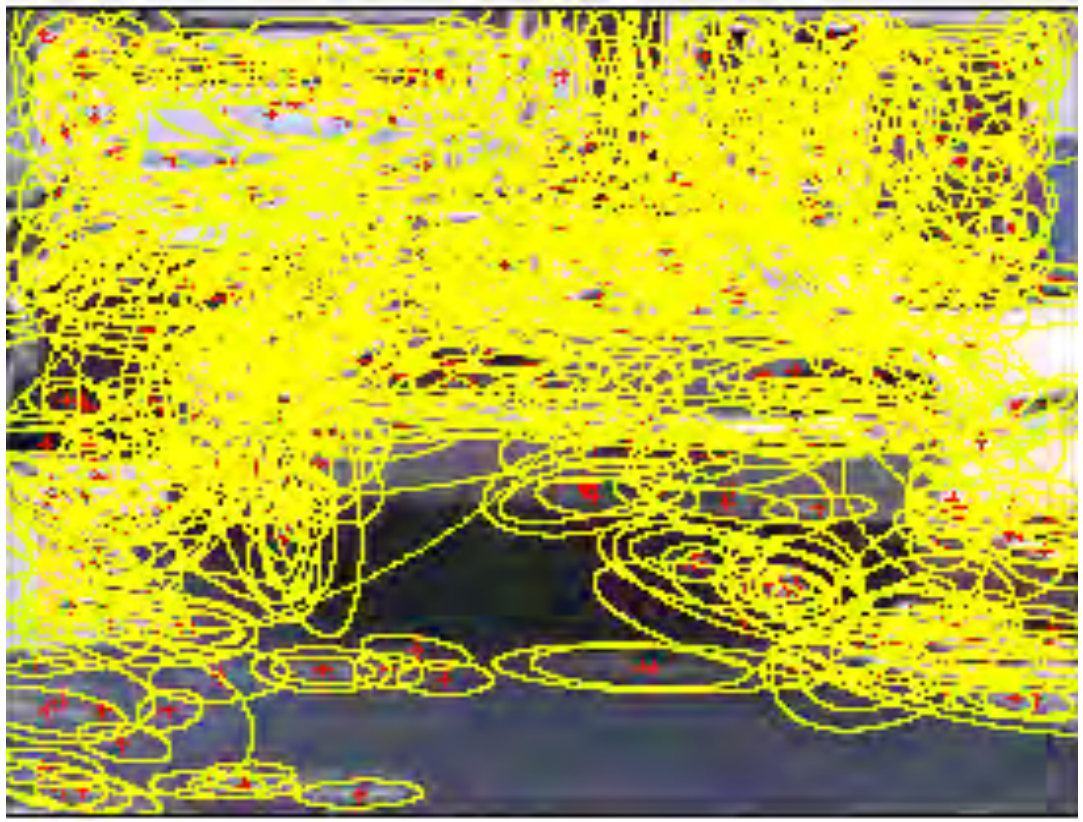
Multiple interest operators

Randomly

Image credits: F-F. Li, E. Nowak, J. Sivic

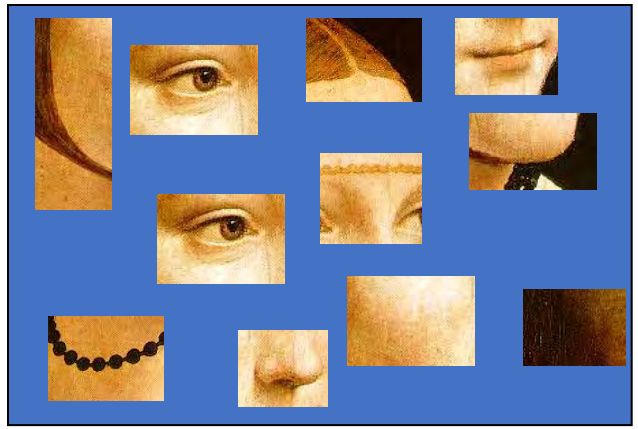
Representation

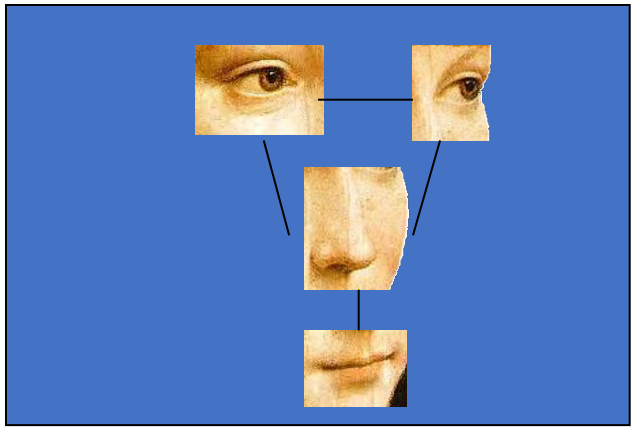
- Building blocks: Choice of descriptors
  [SIFT, HOG, codewords….]

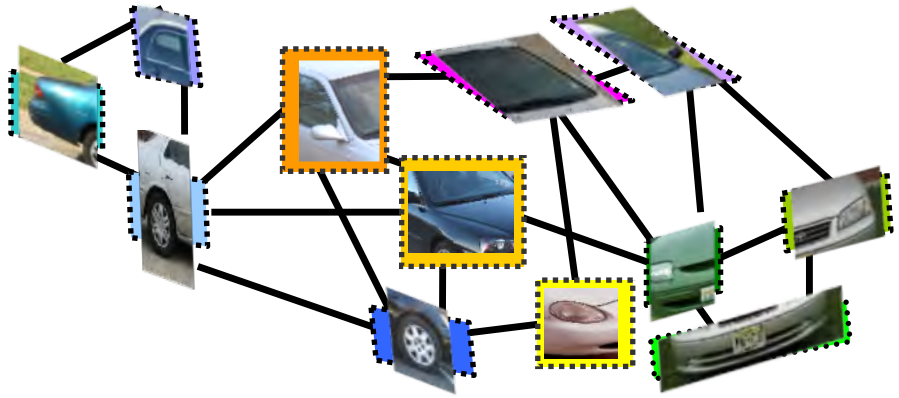


Representation

- Appearance only or location and appearance

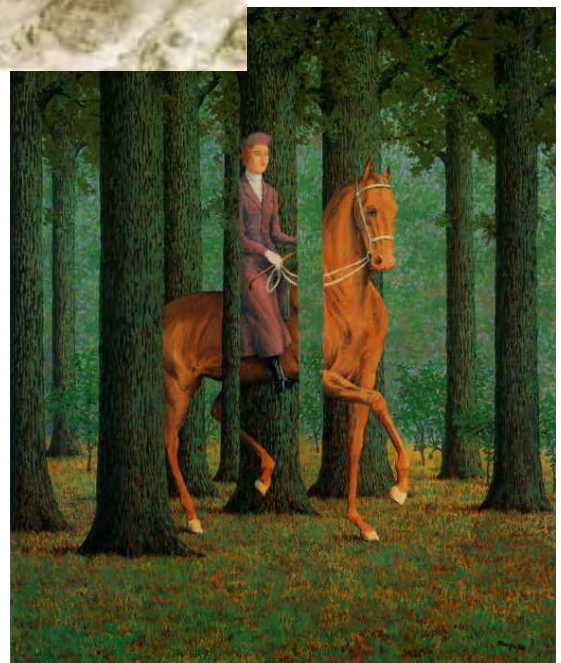






Representation

- Invariances
  - View point
  - Illumination
  - Occlusion
  - Scale
  - Deformation
  - Clutter
  - etc.



Representation

- To handle intra-class variability, it is convenient to describe an object categories using probabilistic models
- Object models: Generative vs Discriminative vs hybrid

Object categorization: the statistical viewpoint

$$p(zebra \mid image)$$

vs.

$$p(no\ zebra \mid image)$$

- Bayes rule: $P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$

$$\frac{p(zebra \mid image)}{p(no\ zebra \mid image)}$$

Object categorization: the statistical viewpoint

$$p(zebra \mid image)$$

vs.

$$p(no\ zebra \mid image)$$

- Bayes rule: $P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$

$$\underbrace{\frac{p(zebra \mid image)}{p(no\ zebra \mid image)}}_{\text{posterior ratio}} = \underbrace{\frac{p(image \mid zebra)}{p(image \mid no\ zebra)}}_{\text{likelihood ratio}} \cdot \underbrace{\frac{p(zebra)}{p(no\ zebra)}}_{\text{prior ratio}}$$

Object categorization: the statistical viewpoint

- **Discriminative methods model posterior**
- **Generative methods model likelihood and prior**

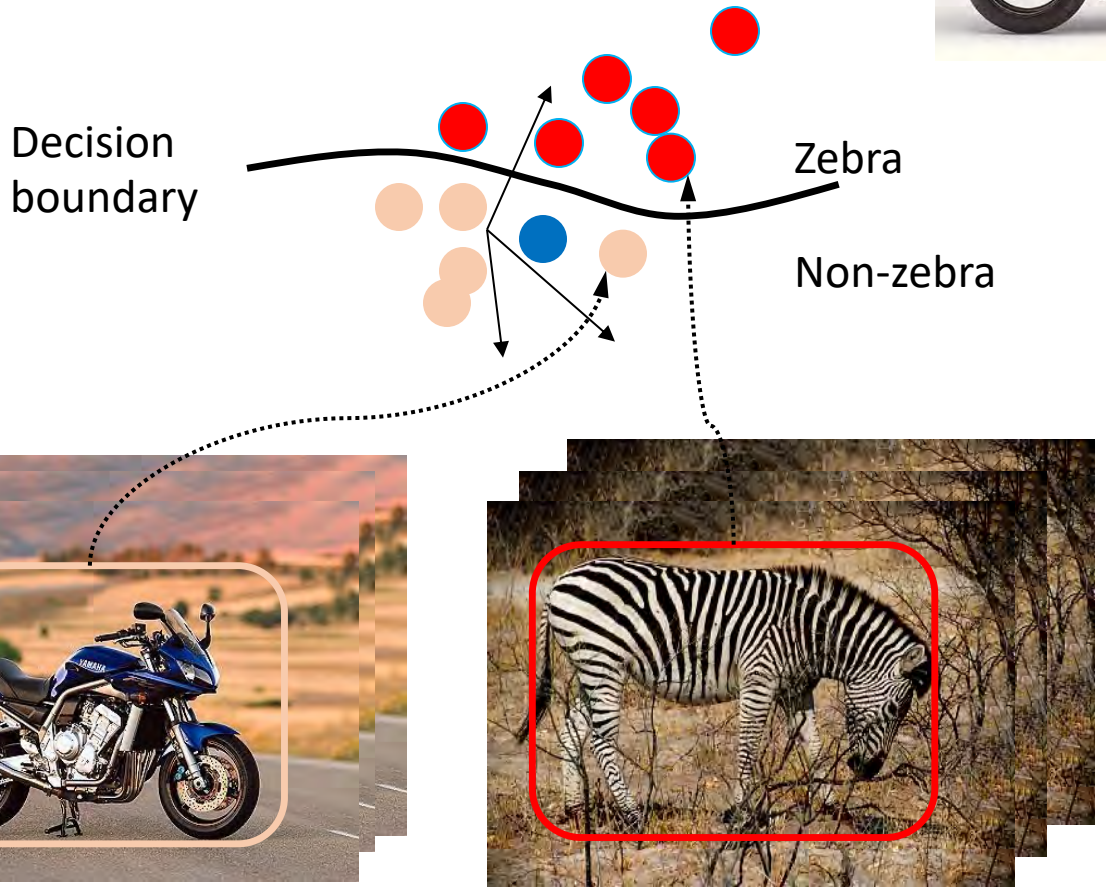
- Bayes rule:

$$\underbrace{\frac{p(zebra \mid image)}{p(no\ zebra \mid image)}}_{\text{posterior ratio}} = \underbrace{\frac{p(image \mid zebra)}{p(image \mid no\ zebra)}}_{\text{likelihood ratio}} \cdot \underbrace{\frac{p(zebra)}{p(no\ zebra)}}_{\text{prior ratio}}$$

Discriminative models

- Modeling the posterior ratio:

$$\frac{p(zebra \mid image)}{p(no\ zebra \mid image)}$$



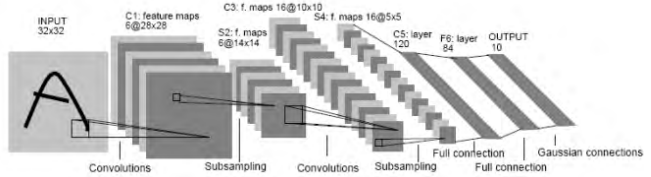
Discriminative models

Nearest neighbor

10^6 examples

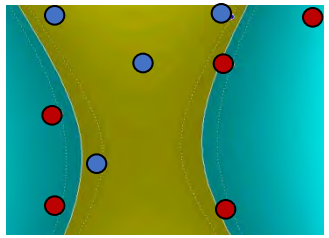
Shakhnarovich, Viola, Darrell 2003
Berg, Berg, Malik 2005...

Neural networks



LeCun, Bottou, Bengio, Haffner 1998
Rowley, Baluja, Kanade 1998
…

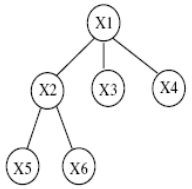
Support Vector Machines



Guyon, Vapnik, Heisele, Serre, Poggio…

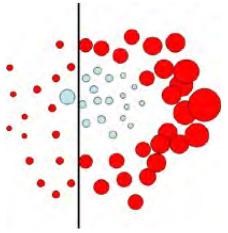
Latent SVM

Structural SVM



Felzenszwalb 00
Ramanan 03…

Boosting



Viola, Jones 2001,
Torralba et al. 2004,
Opelt et al. 2006,…

Courtesy of Vittorio Ferrari
Slide credit: Kristen Grauman

Slide adapted from Antonio Torralba

Generative models

- Naïve Bayes classifier
  - Csurka Bray, Dance & Fan, 2004
- Hierarchical Bayesian topic models (e.g. pLSA and LDA)
  - Object categorization: Sivic et al. 2005, Sudderth et al. 2005
  - Natural scene categorization: Fei-Fei et al. 2005
- 2D Part based models
  - Constellation models: Weber et al 2000; Fergus et al 200
  - Star models: ISM (Leibe et al 05)
- 3D part based models:
  - multi-aspects: Sun, et al, 2009

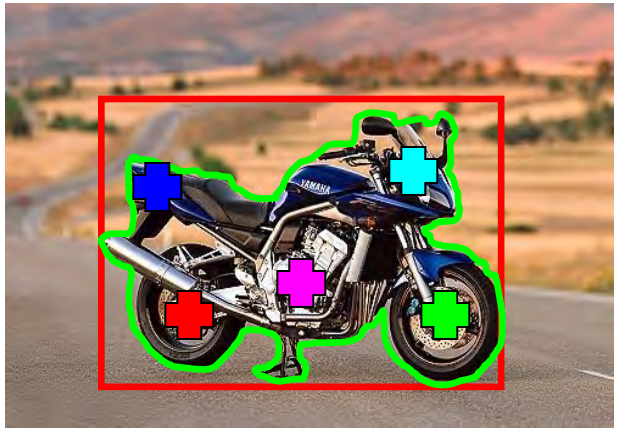
Learning

- Learning parameters: What are you maximizing? Likelihood (Gen.) or performances on train/validation set (Disc.)

Learning

- Learning parameters: What are you maximizing? Likelihood (Gen.) or performances on train/validation set (Disc.)
- Level of supervision
  - Manual segmentation; bounding box; image labels; noisy labels

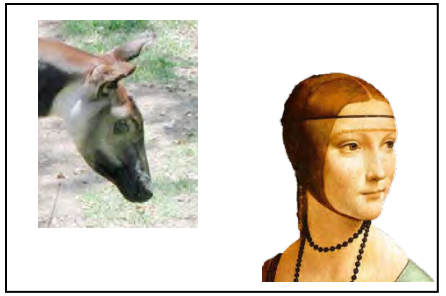
- Batch/incremental
- Priors



Learning

- Learning parameters: What are you maximizing? Likelihood (Gen.) or performances on train/validation set (Disc.)
- Level of supervision
  - Manual segmentation; bounding box; image labels; noisy labels

- Batch/incremental
- Priors
- Training images:
  - Issue of overfitting
  - Negative images for discriminative methods



Basic properties

- Representation
  - How to represent an object category; which classification scheme?

- Learning
  - How to learn the classifier, given training data

- Recognition
  - How the classifier is to be used on novel data

Recognition

- Recognition task: classification, detection, etc..

Recognition

– Recognition task

– Search strategy: Sliding Windows

Viola, Jones 2001,

• Simple

• Computational complexity

- BSW by Lampert et al 08

- Also, Alexe, et al 10

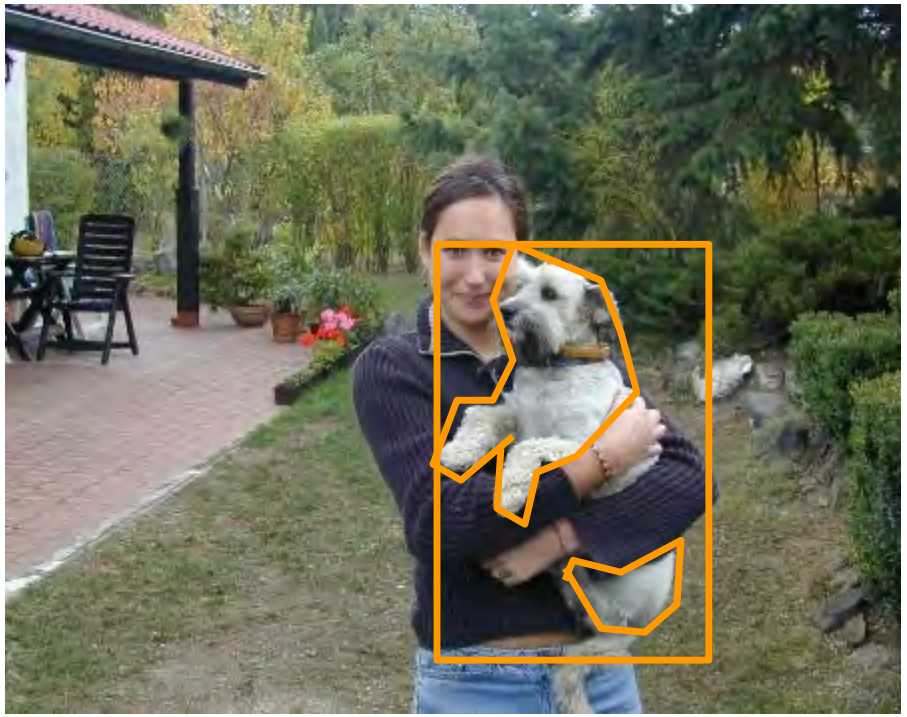
Recognition

– Recognition task

– Search strategy: Sliding Windows

Viola, Jones 2001,

- Simple
- Computational complexity
  - BSW by Lampert et al 08
  - Also, Alexe, et al 10
- Localization
  - Objects are not boxes



Recognition

– Recognition task

– Search strategy: Sliding Windows

Viola, Jones 2001,

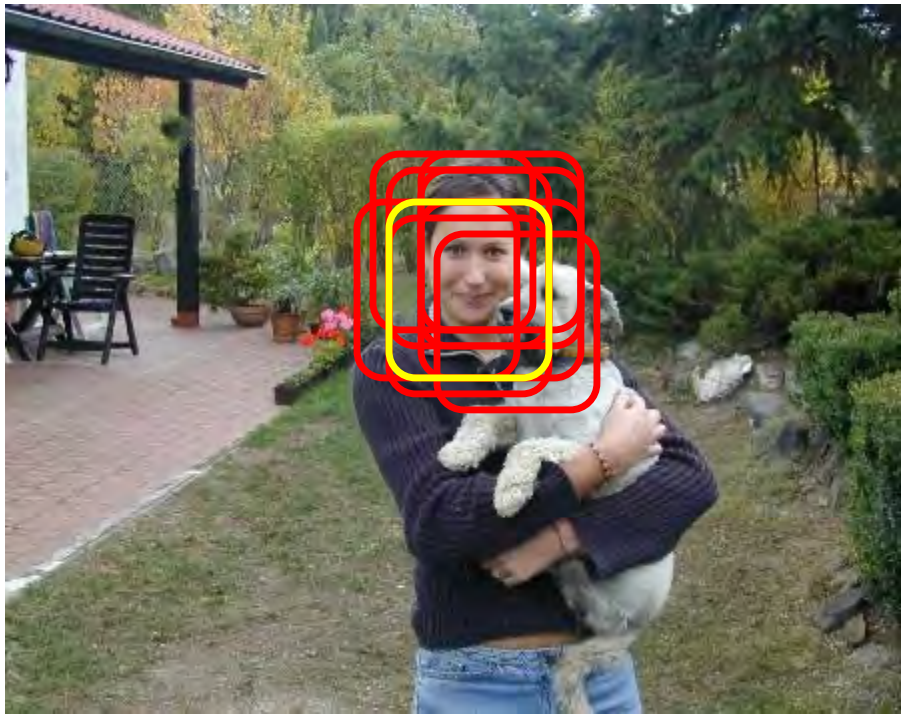
- Simple
- Computational complexity

  - BSW by Lampert et al 08

  - Also, Alexe, et al 10

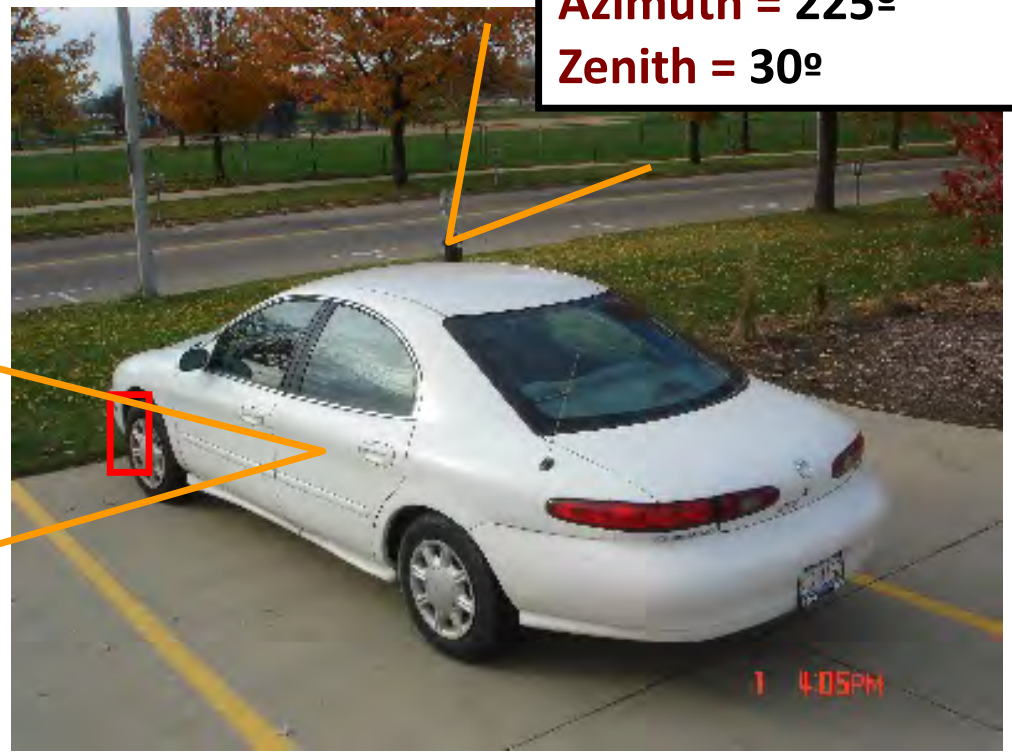
- Localization
  - Objects are not boxes
  - Prone to false positive

    Non max suppression:
    Canny ’86
    ….
    Desai et al , 2009



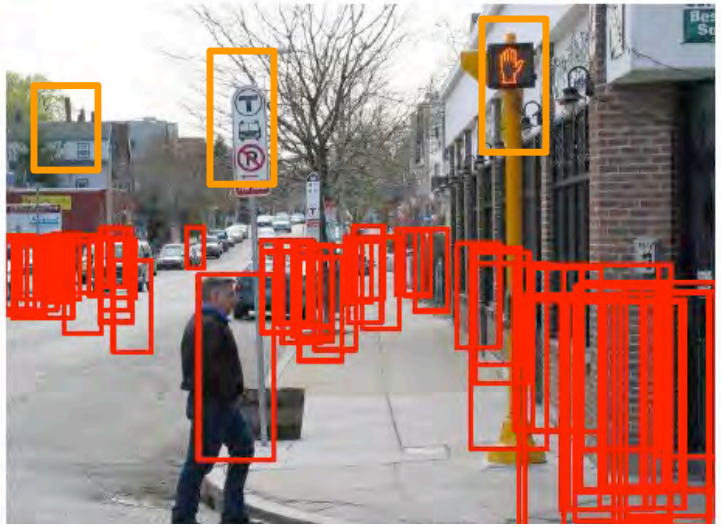
Recognition

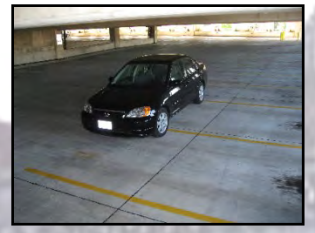
- Recognition task
- Search strategy
- Attributes



Recognition

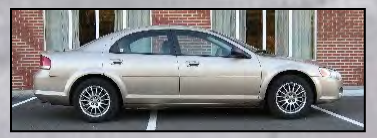
- Recognition task
- Search strategy
- Attributes
- Context







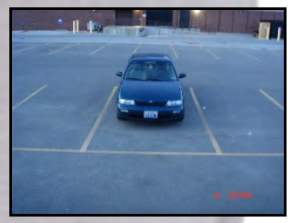










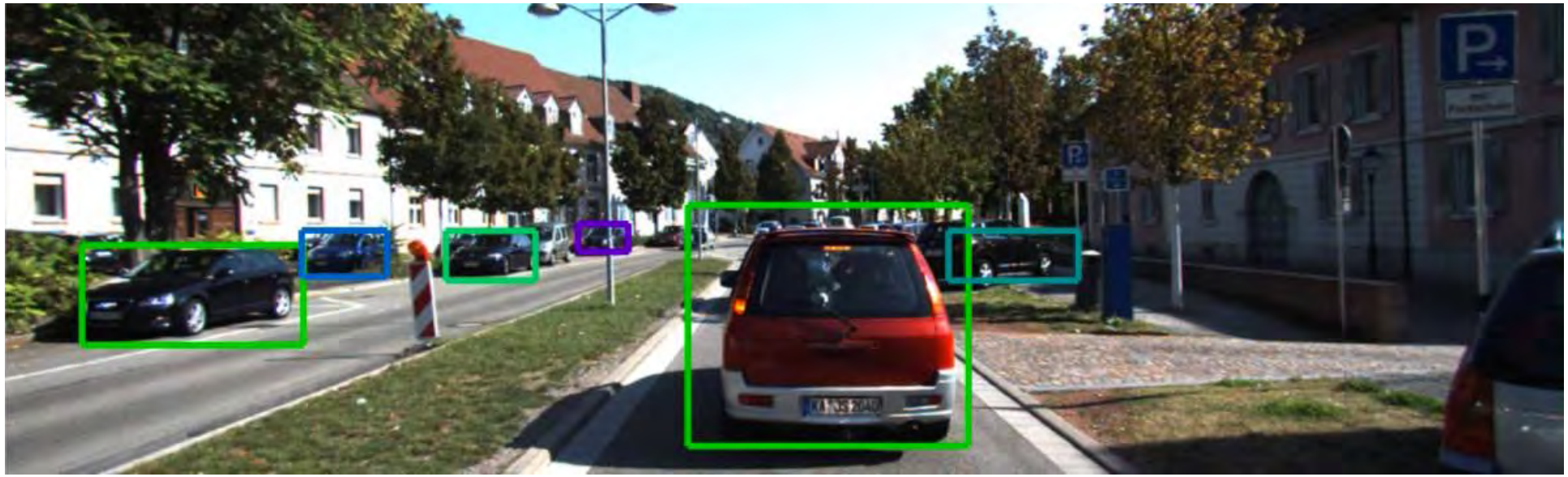



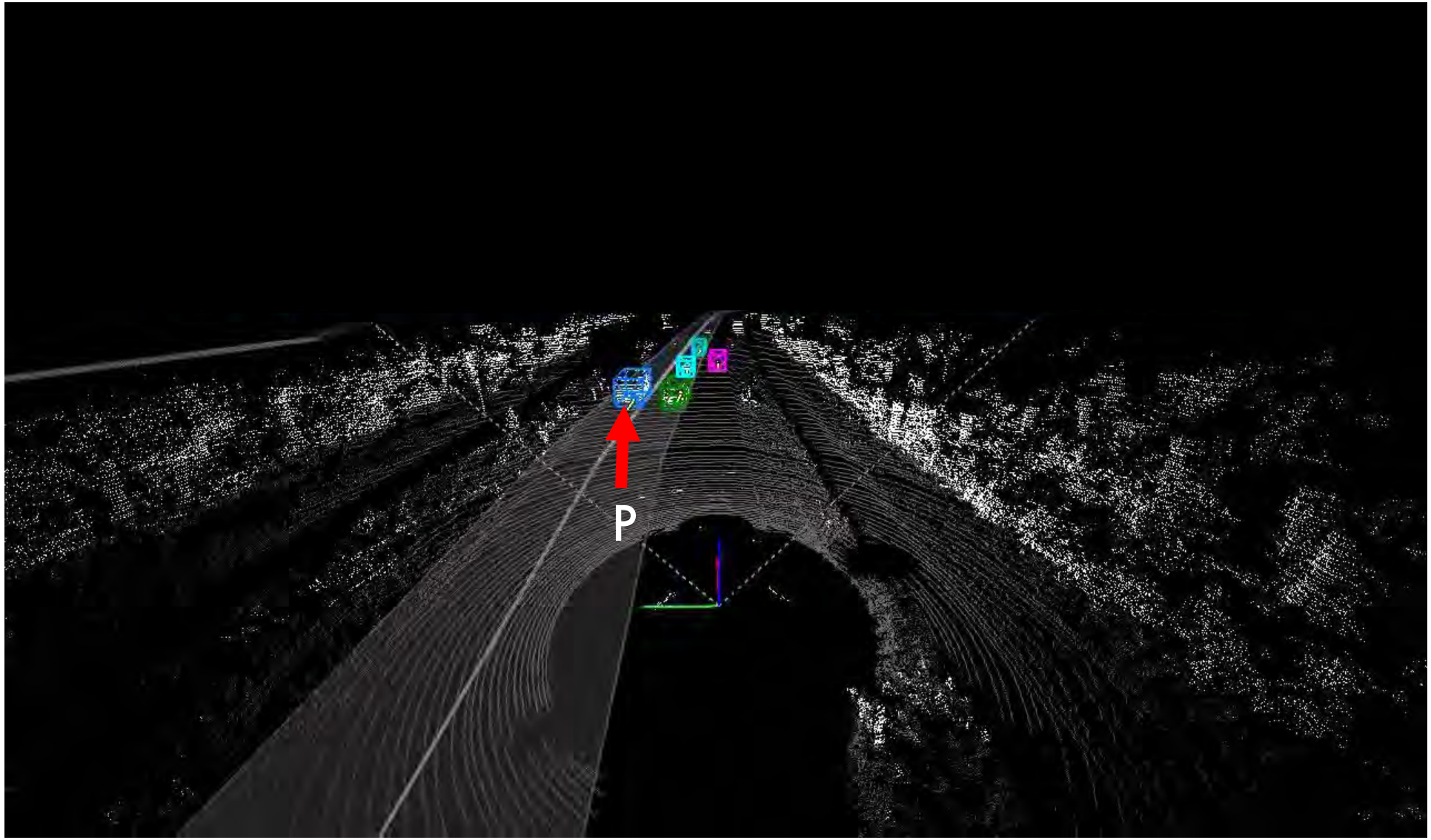
History

- AI Winter

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

ROB 599 - Final Project
Perception



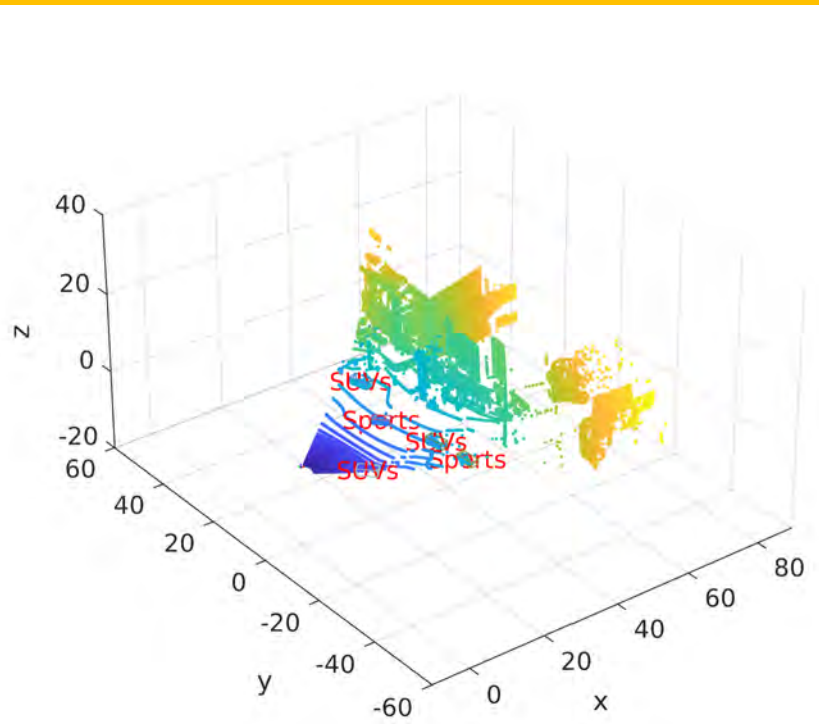

P


grand
theft
auto

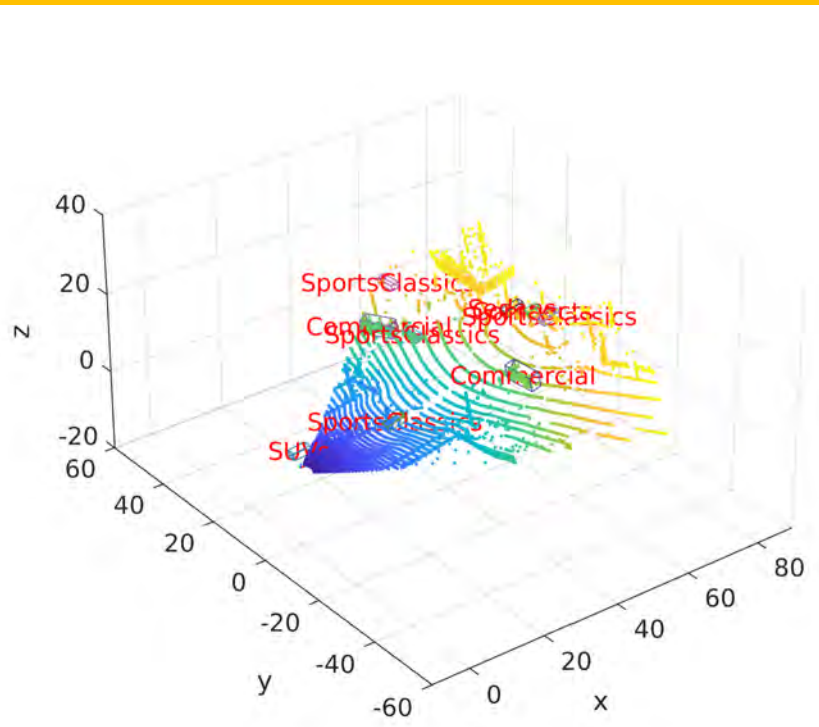

OPIUM NIGHTS
HOTEL

Video

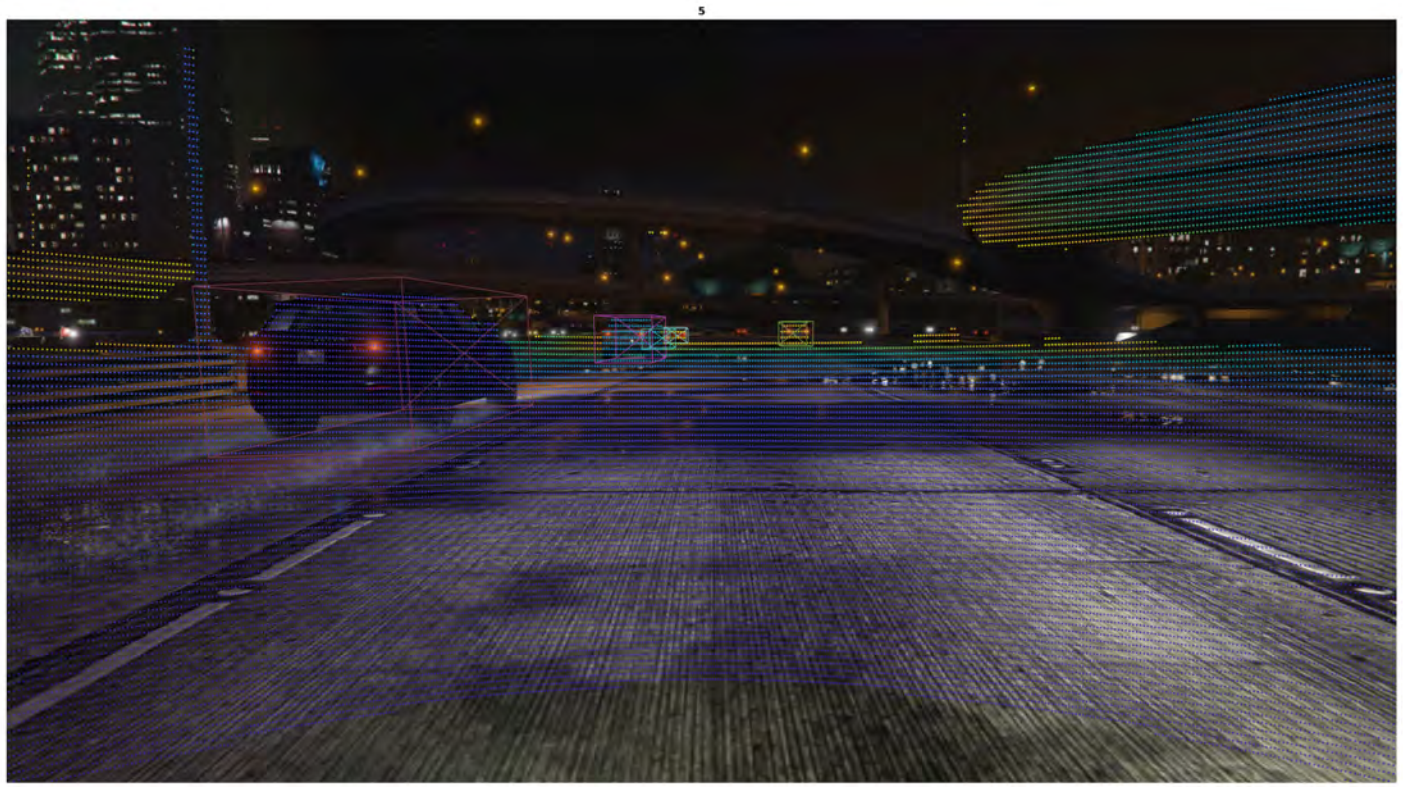
Dataset

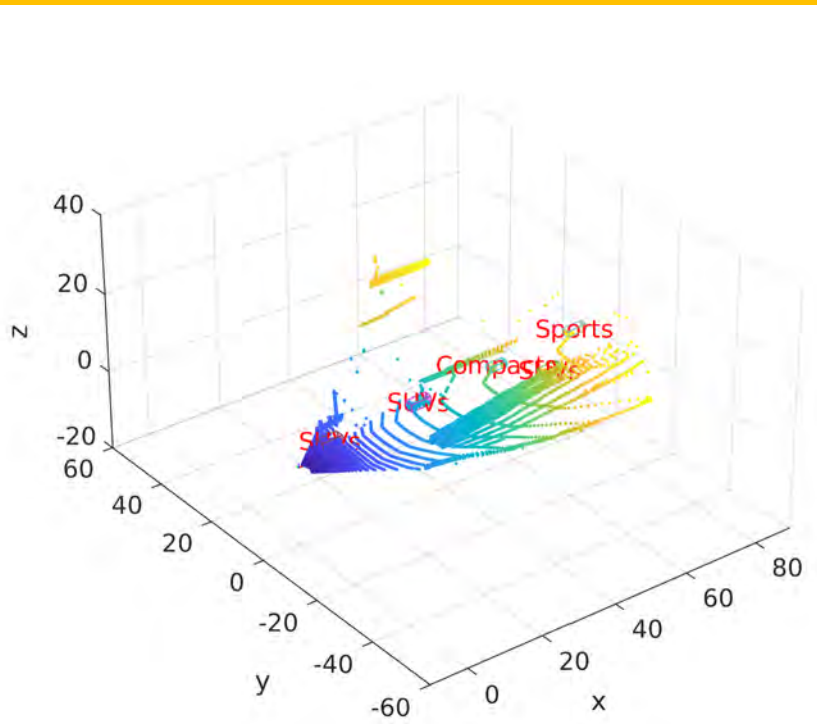

40
20
0
z
-20
60
40
20
0
-20
-40
-60
y
0
20
40
60
80
x
SUVs
Sports
SUVs
Sports
SUVs

Dataset

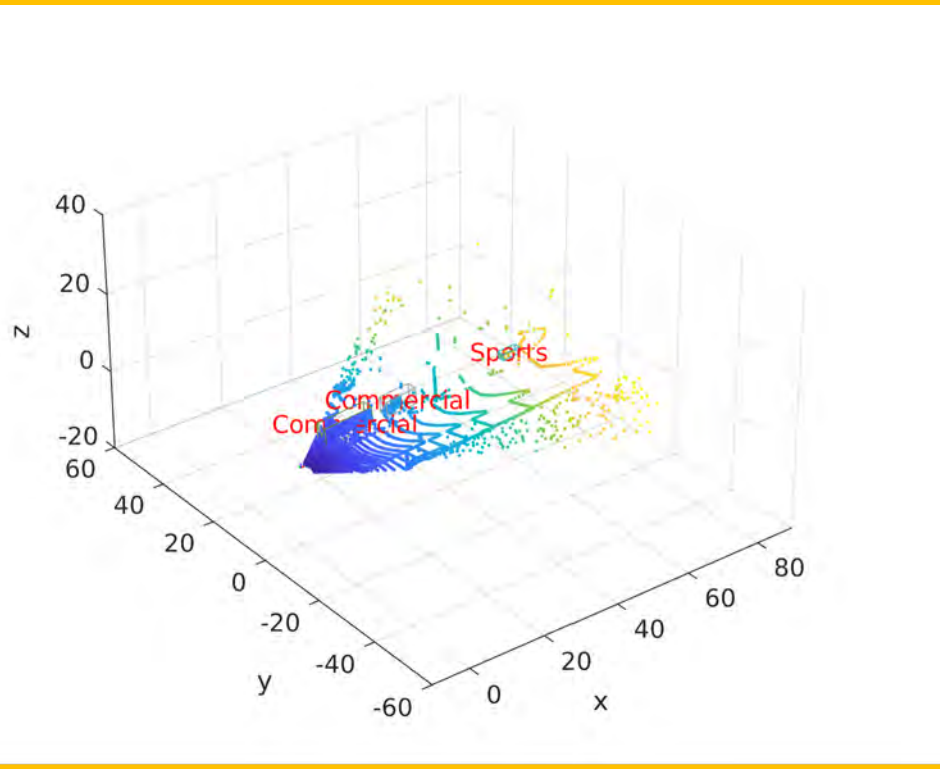

40
20
0
-20
z
60
40
20
0
-20
-40
-60
y
0
20
40
60
80
x
SportsClassic
Commercial
SportsClassics
SportsClassics
Commercial
SportsClassics

Dataset

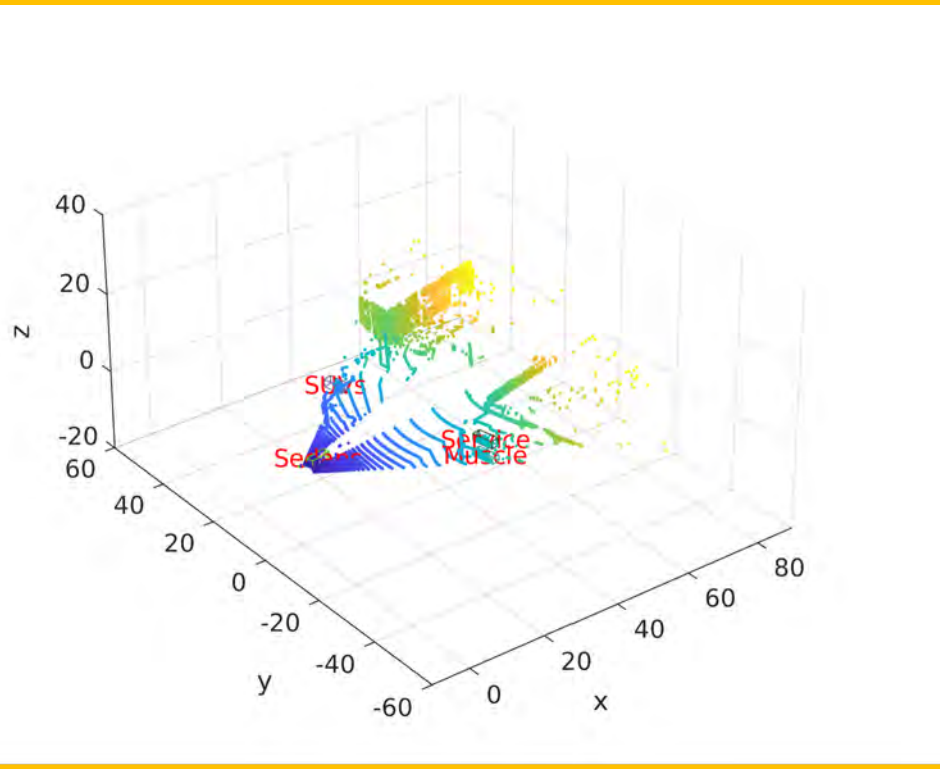



40
20
0
-20
z
60
40
20
0
-20
-40
-60
y
0
20
40
60
80
x
Sports
Compact
SUVs
SUVs

Dataset


40
20
0
-20
z
60
40
20
0
-20
-40
-60
y
0
20
40
60
80
x
Sport's
Commercial
Commercial

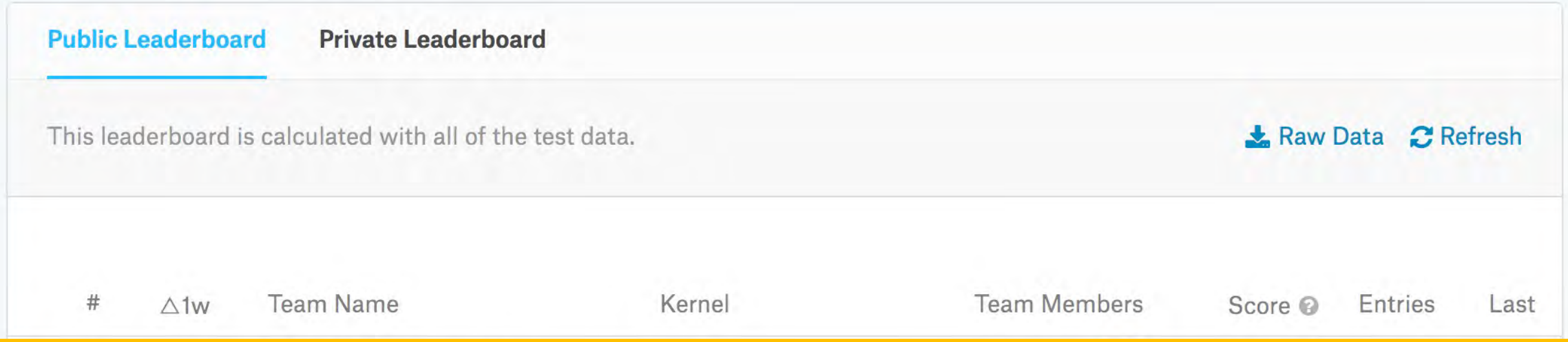
Dataset



Task

- Classify the scene: Car, Motorcycle, or Nothing
- You will get full credit for successfully completing this task

Leaderboard

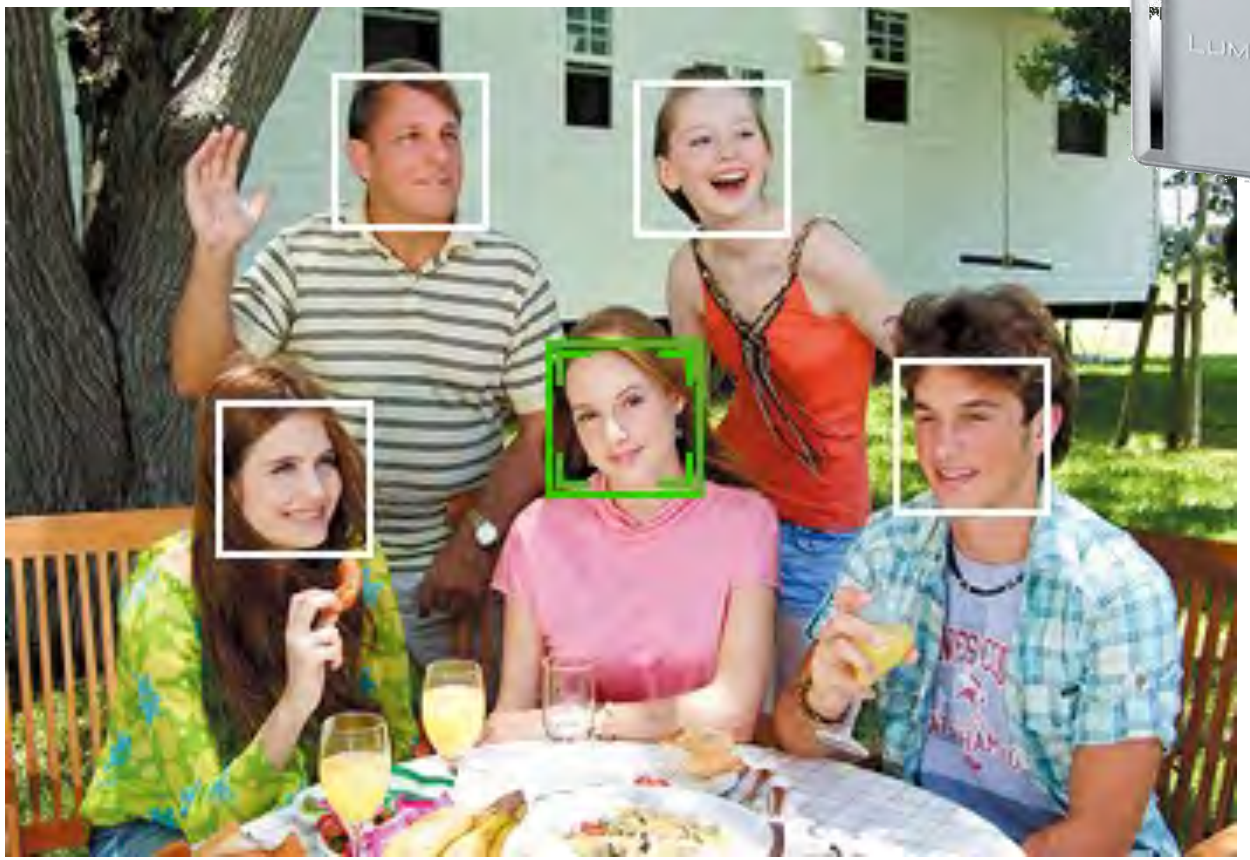


Dataset

- You will get
  - training and testing snapshots
  - Sample code to read and visualize snapshots in MATLAB and Python
- Each snapshot contains
  - RGB image (*_image.jpg) with size 1052 x 1914
  - LiDAR points (*_cloud.bin)
  - Camera matrix (*_proj.bin)

Face Recognition

- Digital photography



Face Recognition

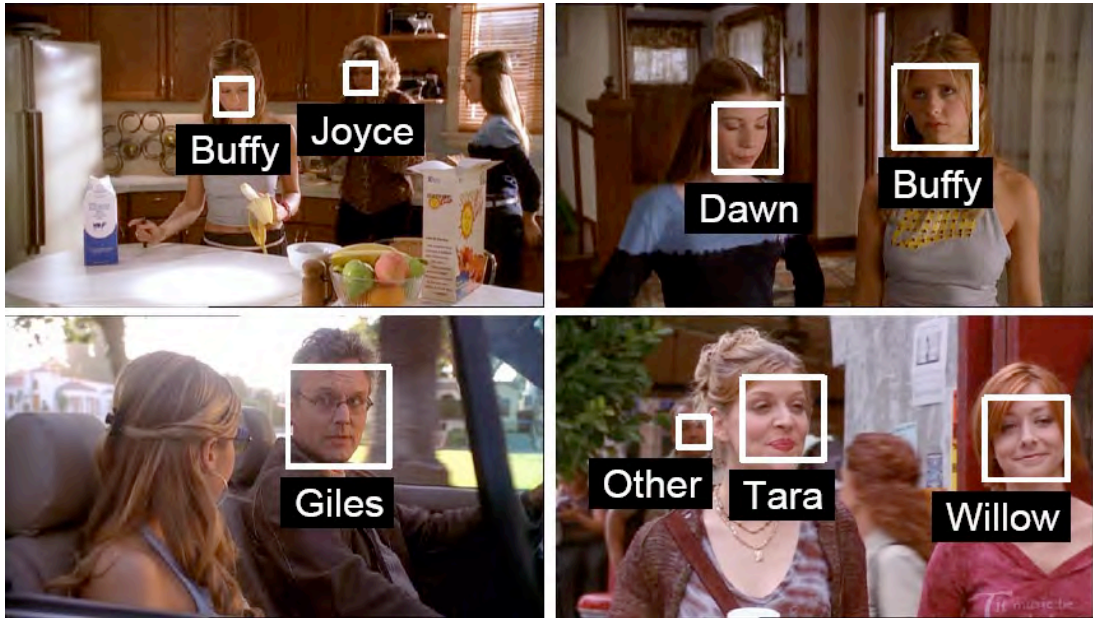
- Digital photography
- Surveillance

Face Recognition

- Digital photography
- Surveillance
- Album organization

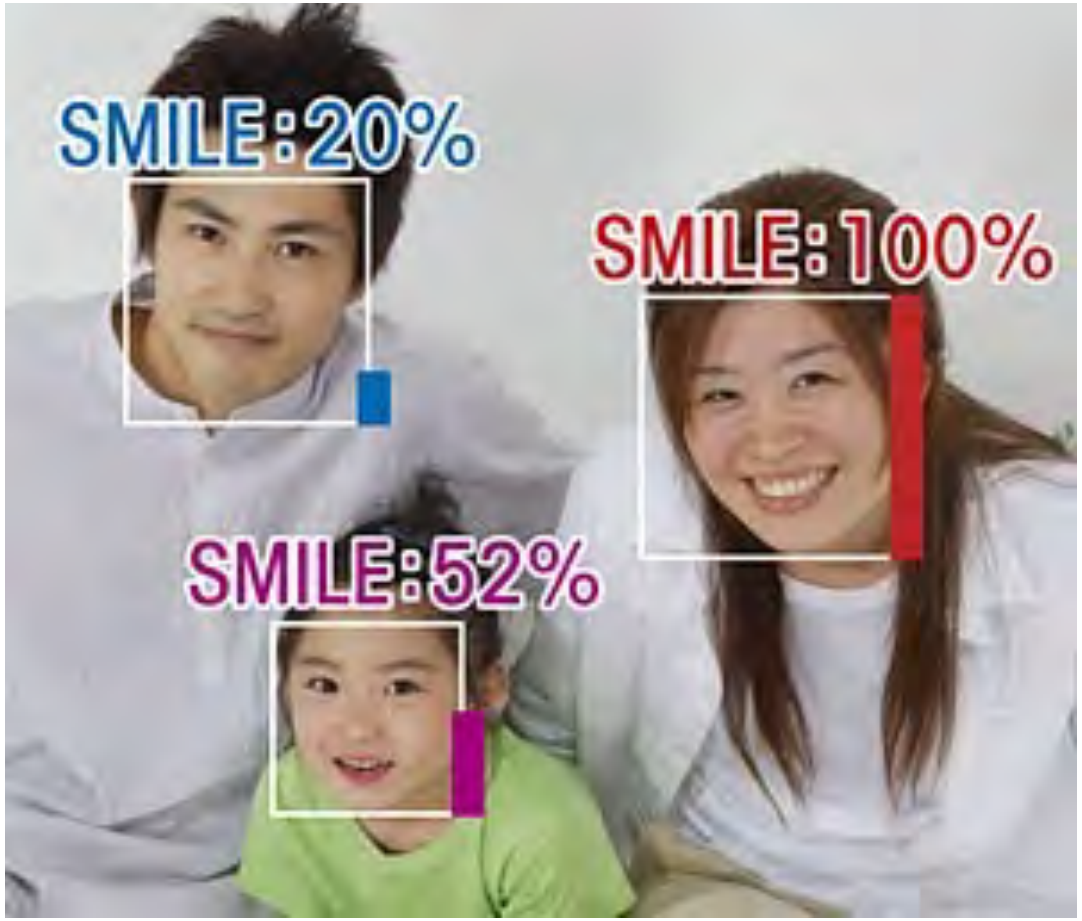
Face Recognition

- Digital photography
- Surveillance
- Album organization
- Person tracking/id.



Face Recognition

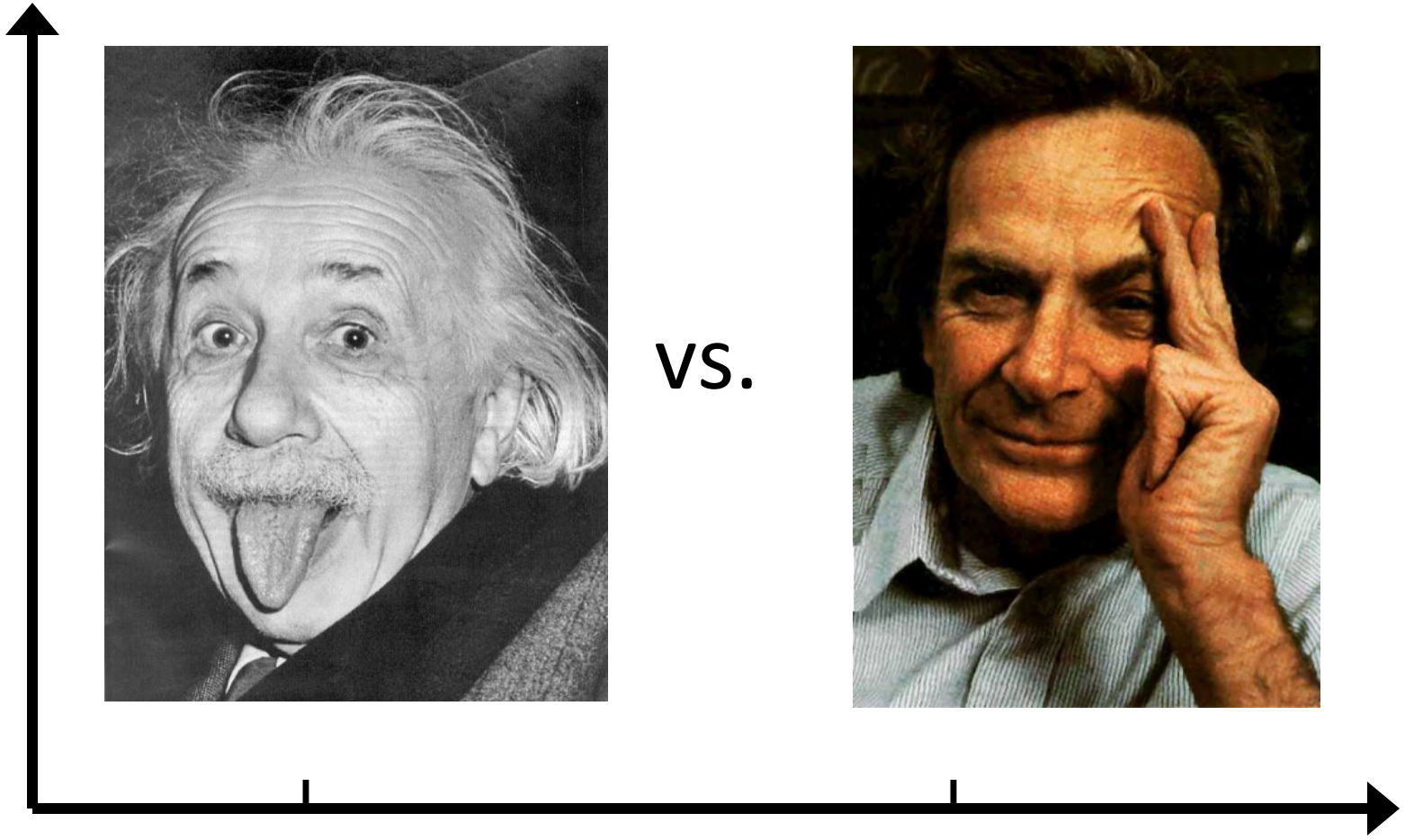
- Digital photography
- Surveillance
- Album organization
- Person tracking/id.
- Emotions and expressions



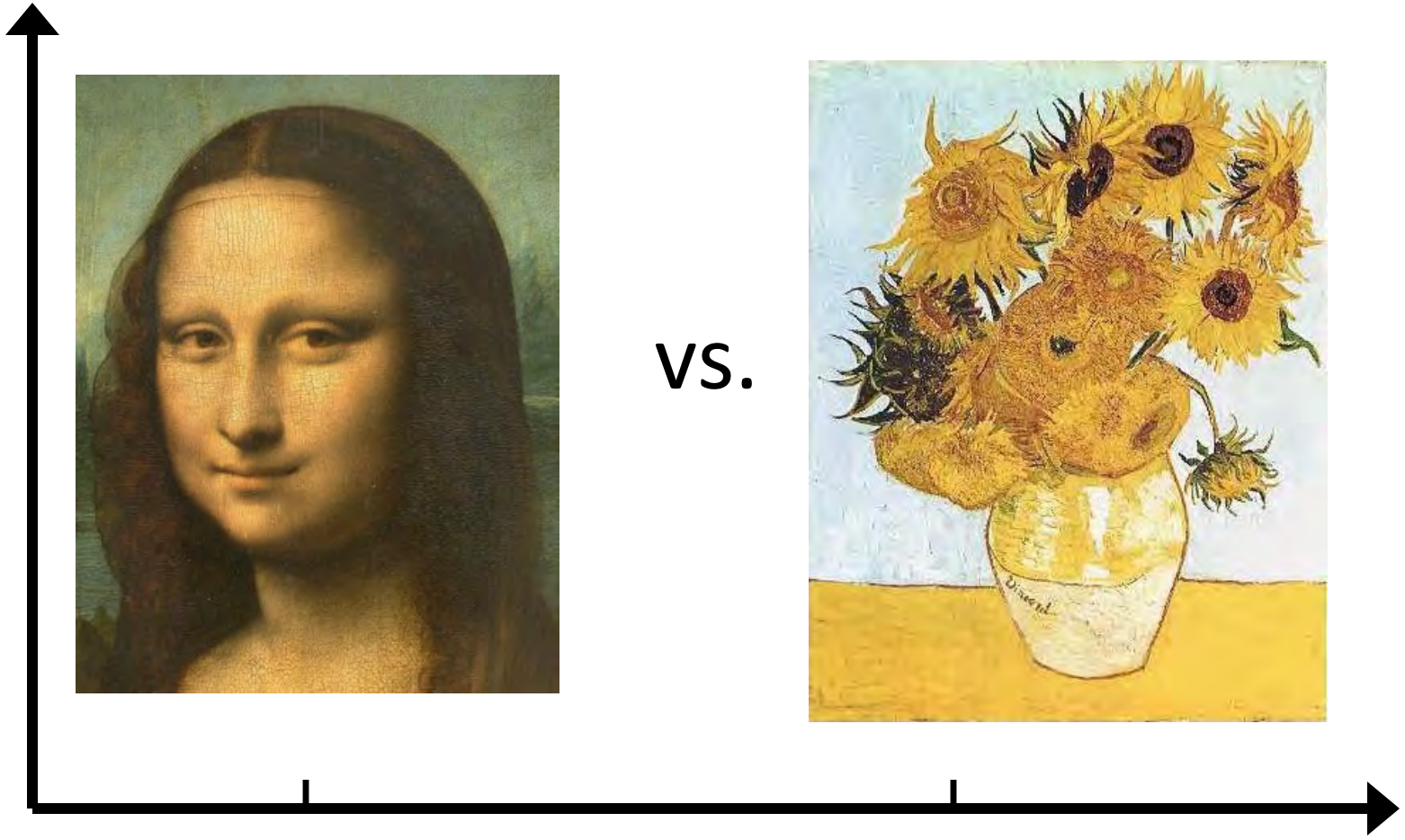
Face Recognition

- Digital photography
- Surveillance
- Album organization
- Person tracking/id.
- Emotions and expressions
- Security/warfare
- Tele-conferencing
- Etc.

What's 'recognition'?



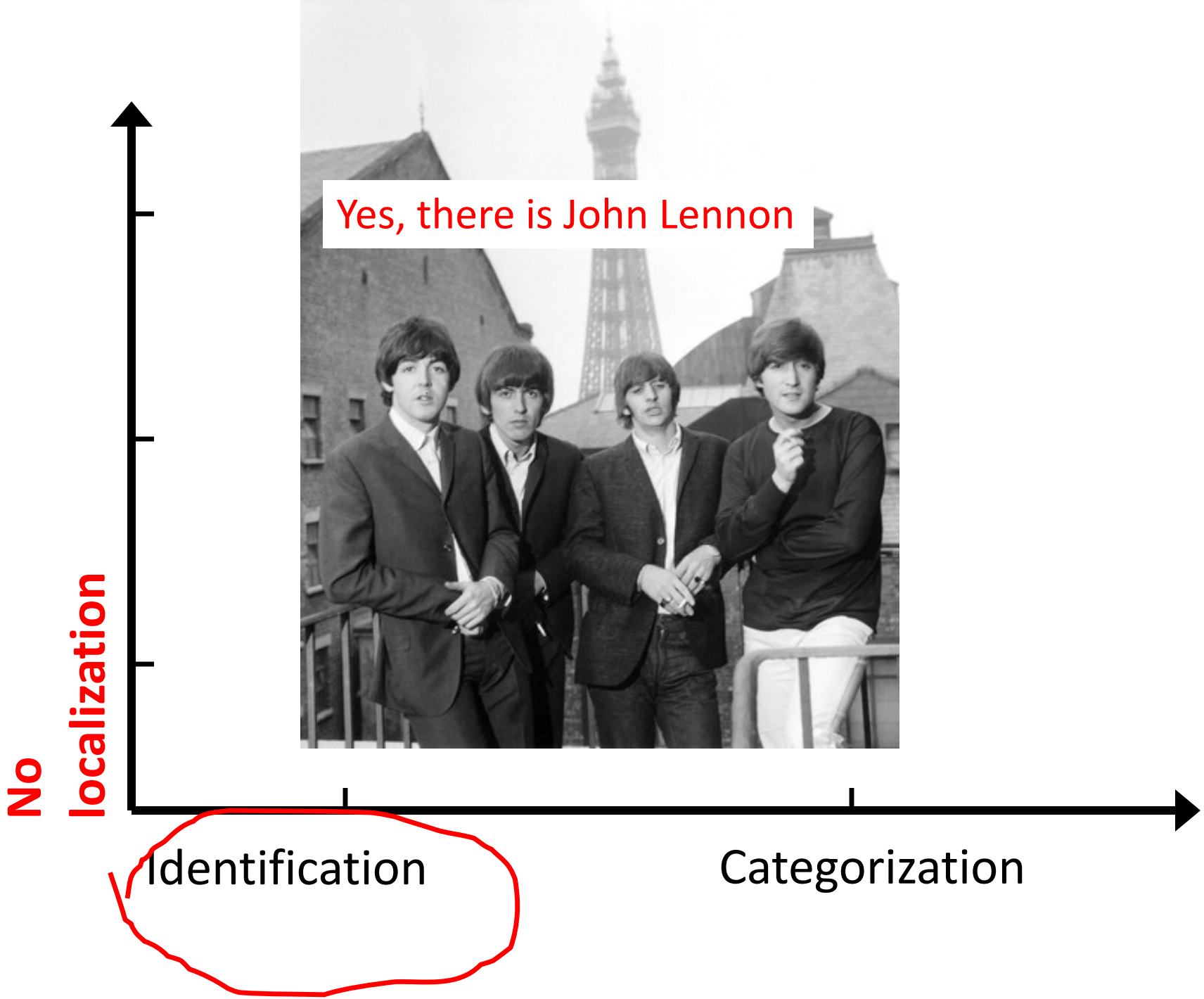
What's 'recognition'?

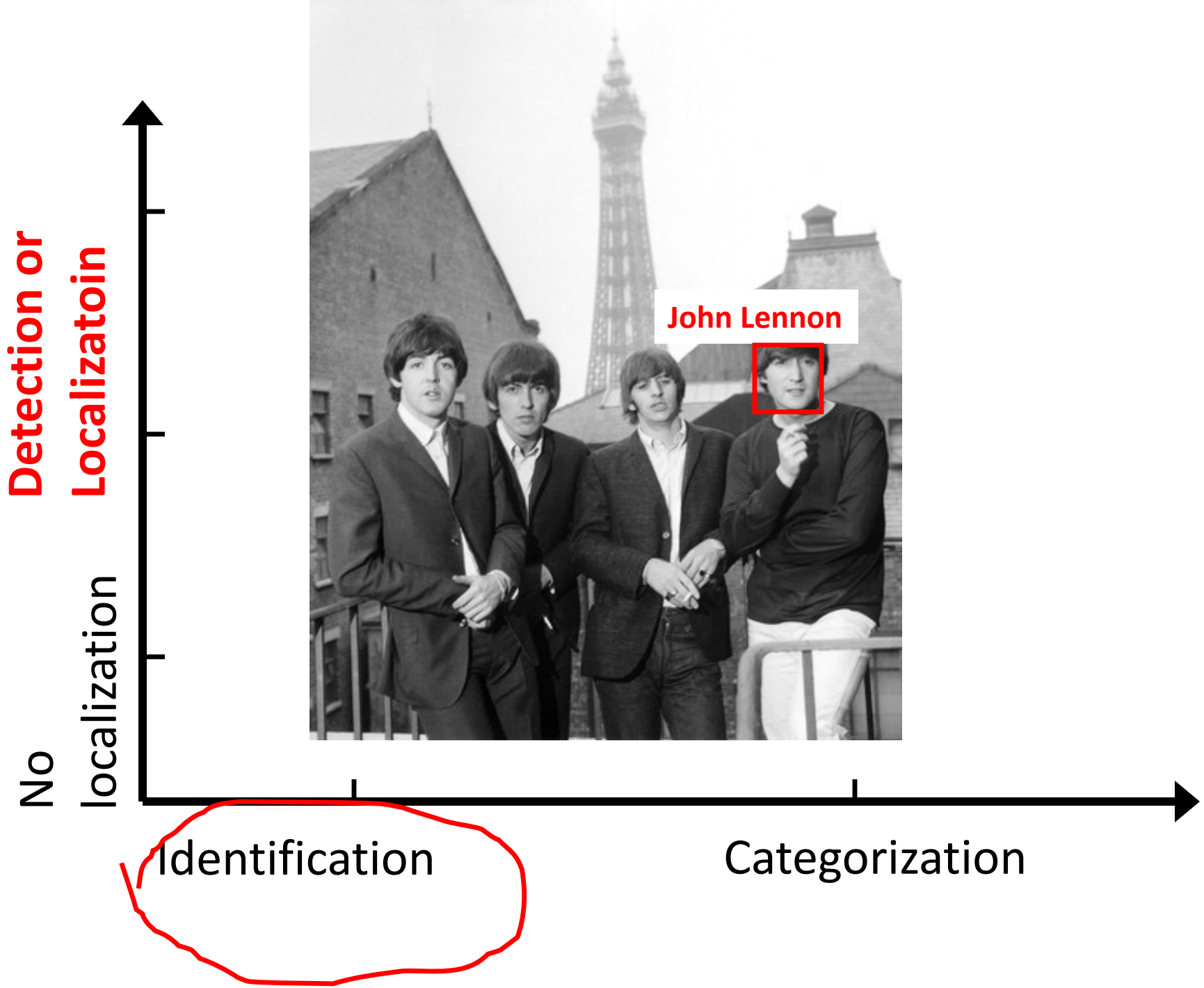


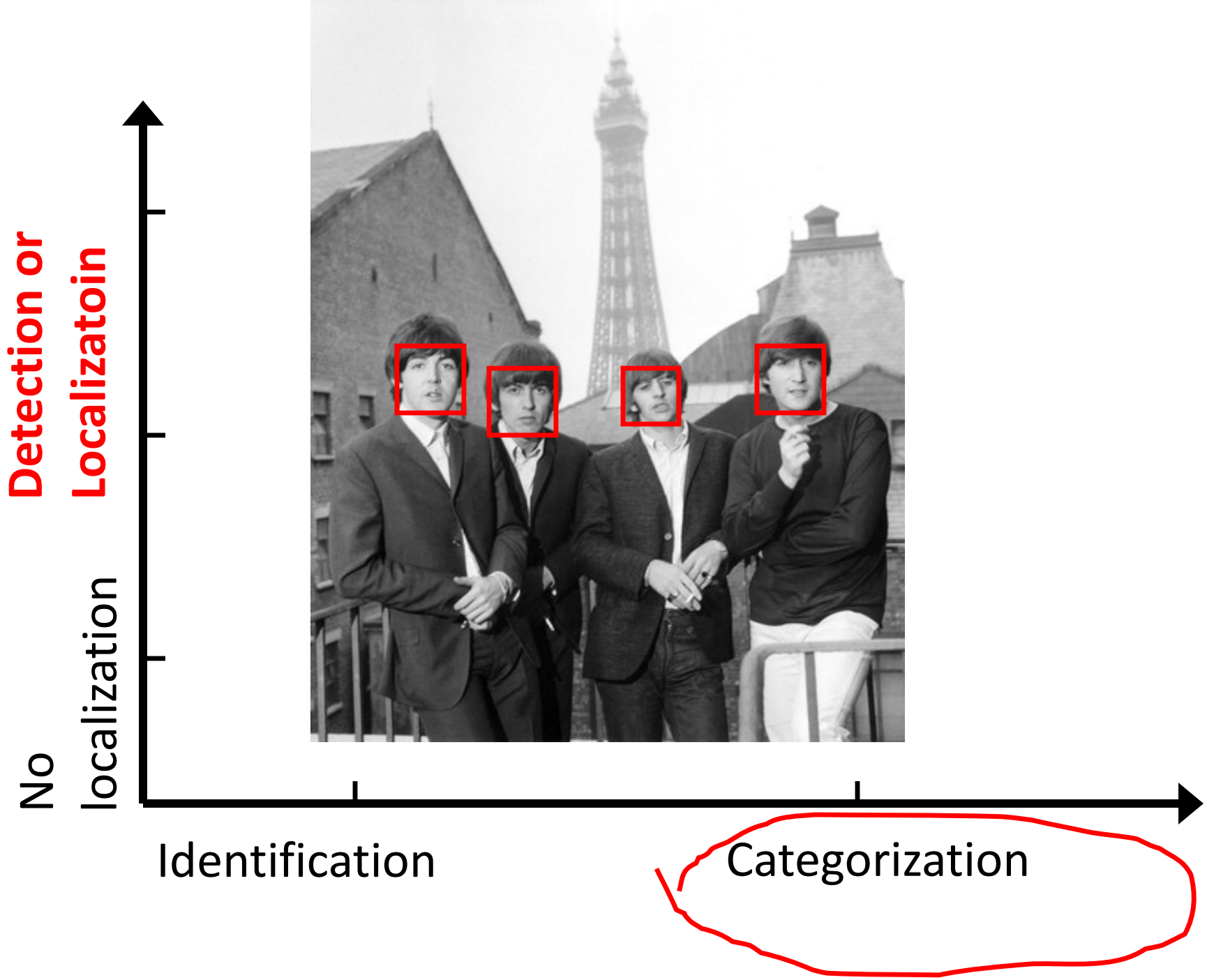
No localization

Identification

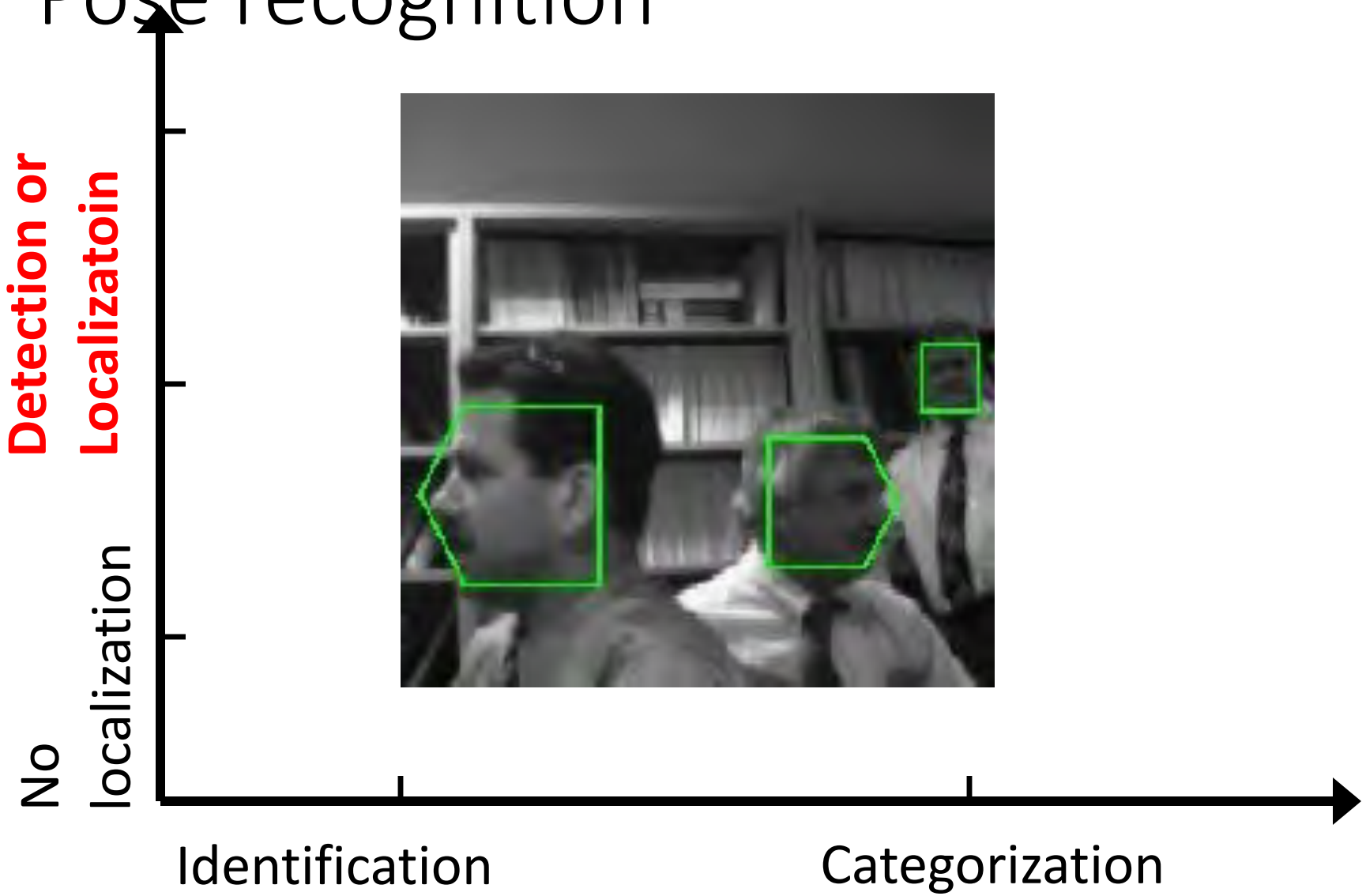
Categorization


Yes, there is John Lennon
No localization
Identification
Categorization

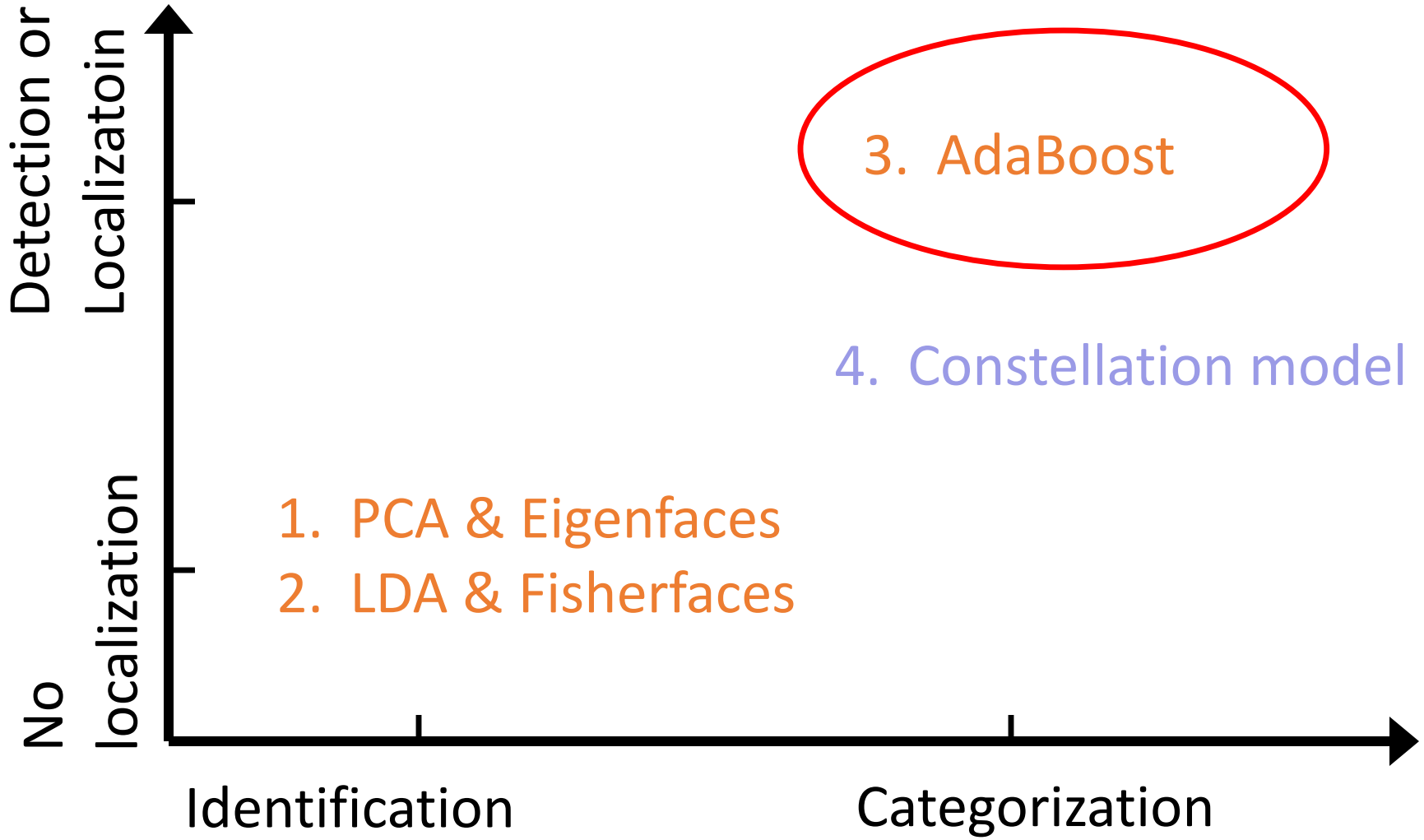

Detection or
Localizatoin
No
localization
John Lennon
Identification
Categorization


Detection or
Localizatoin
No
localization
Identification
Categorization

Pose recognition



Face Recognition methods



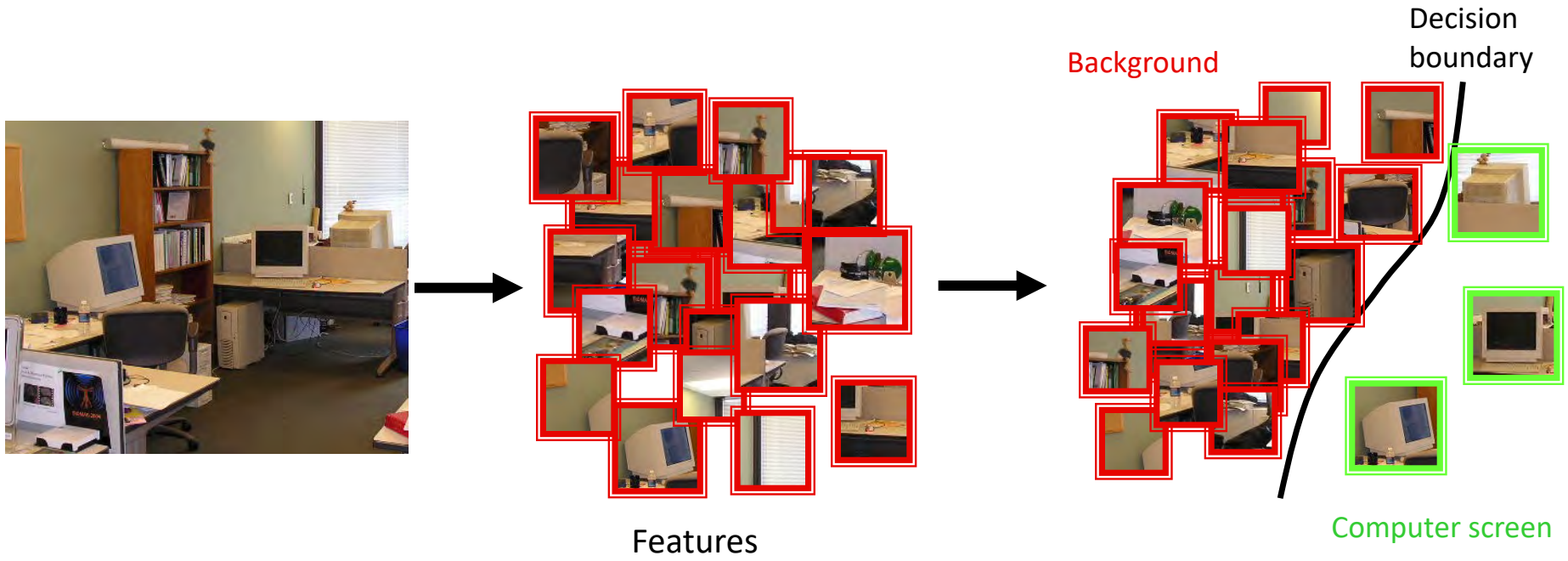
Robust Face Detection Using AdaBoost

- Brief intro on (Ada)Boosting
- Viola & Jones, 2001

Reference:
P. Viola and M. Jones (2001) Robust Real-time Object Detection, IJCV.

Boosting

Designing a strong classifier from a set of weak classifier



Boosting

- Defines a classifier using an additive model:

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \ldots$$

Strong classifier

Features vector

Weight

Weak classifier

Boosting

- Defines a classifier using an additive model:

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \ldots$$

Strong classifier

Features vector

Weight

Weak classifier

- We need to define a family of weak classifiers

$f_k(x)$ form a family of weak classifiers

Why boosting?

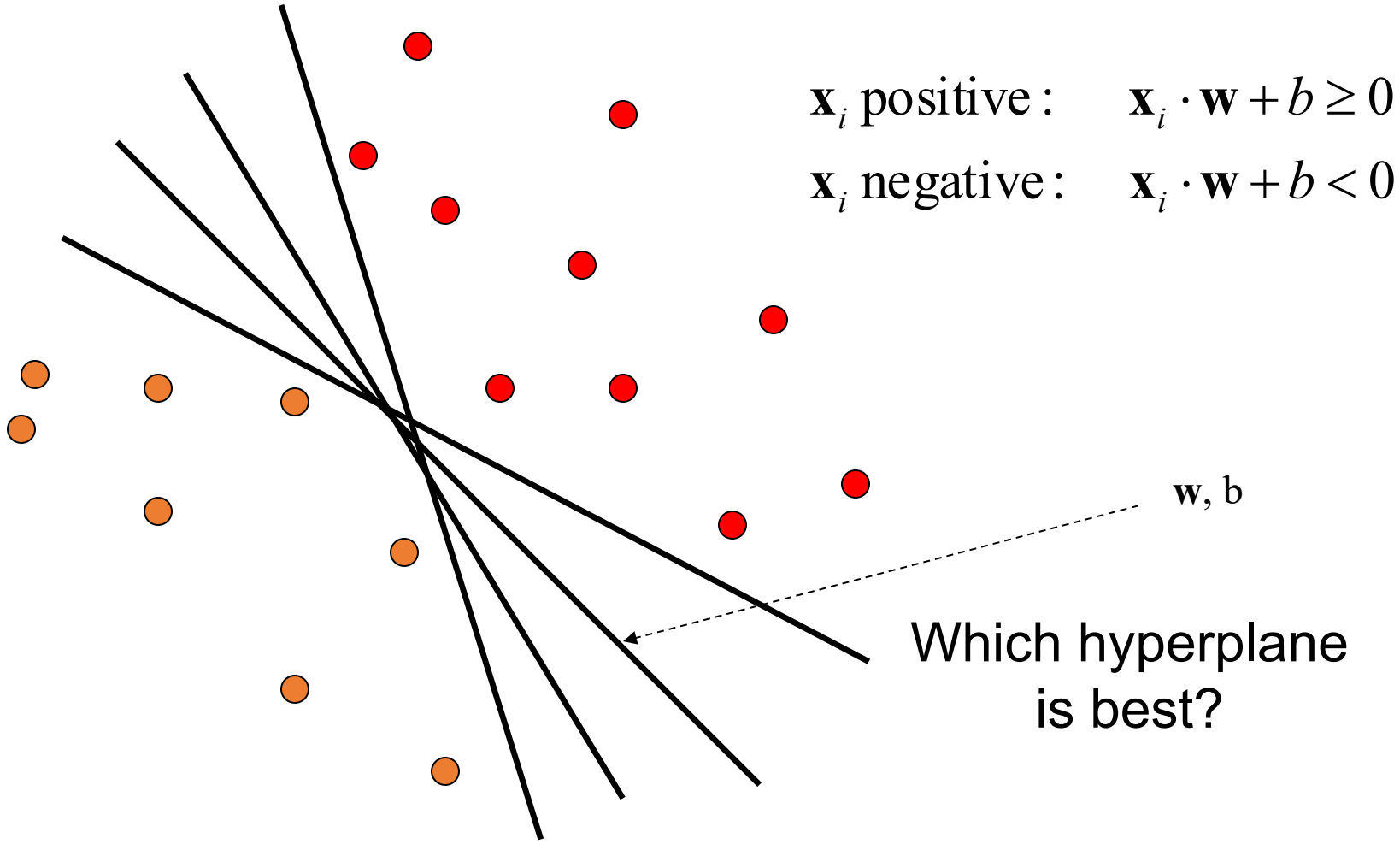
- A simple algorithm for learning robust classifiers
  - Freund & Shapire, 1995
  - Friedman, Hastie, Tibshhirani, 1998

- Provides efficient algorithm for sparse visual feature selection
  - *Tieu & Viola, 2000*
  - *Viola & Jones, 2003*

- Easy to implement, not requires external optimization tools.

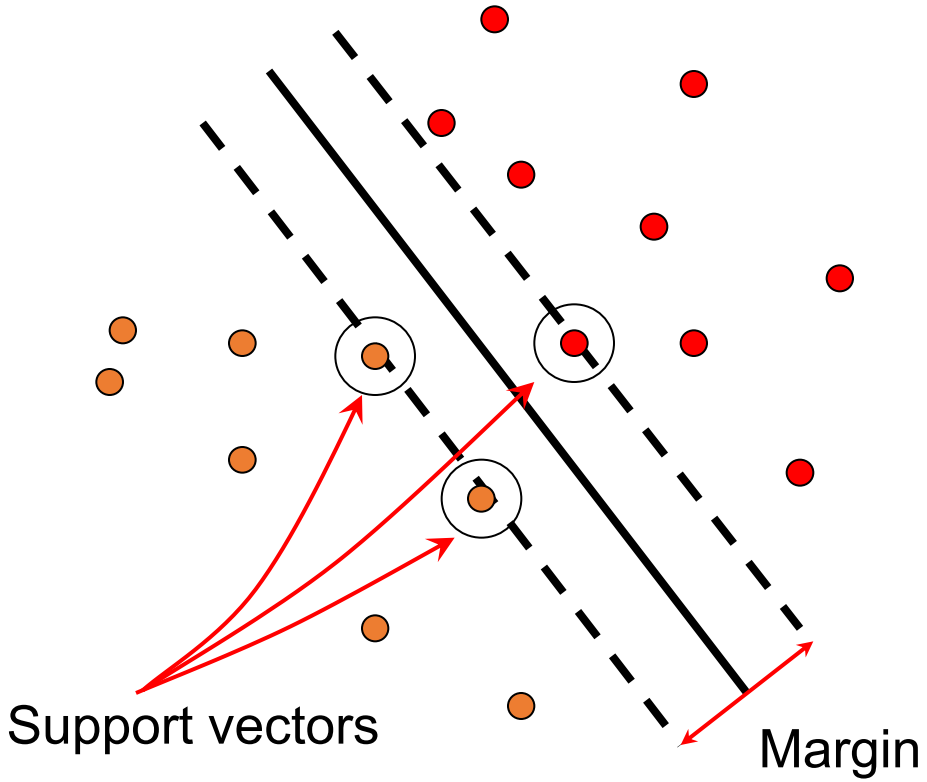
Linear classifiers

- Find linear function (*hyperplane*) to separate positive and negative examples



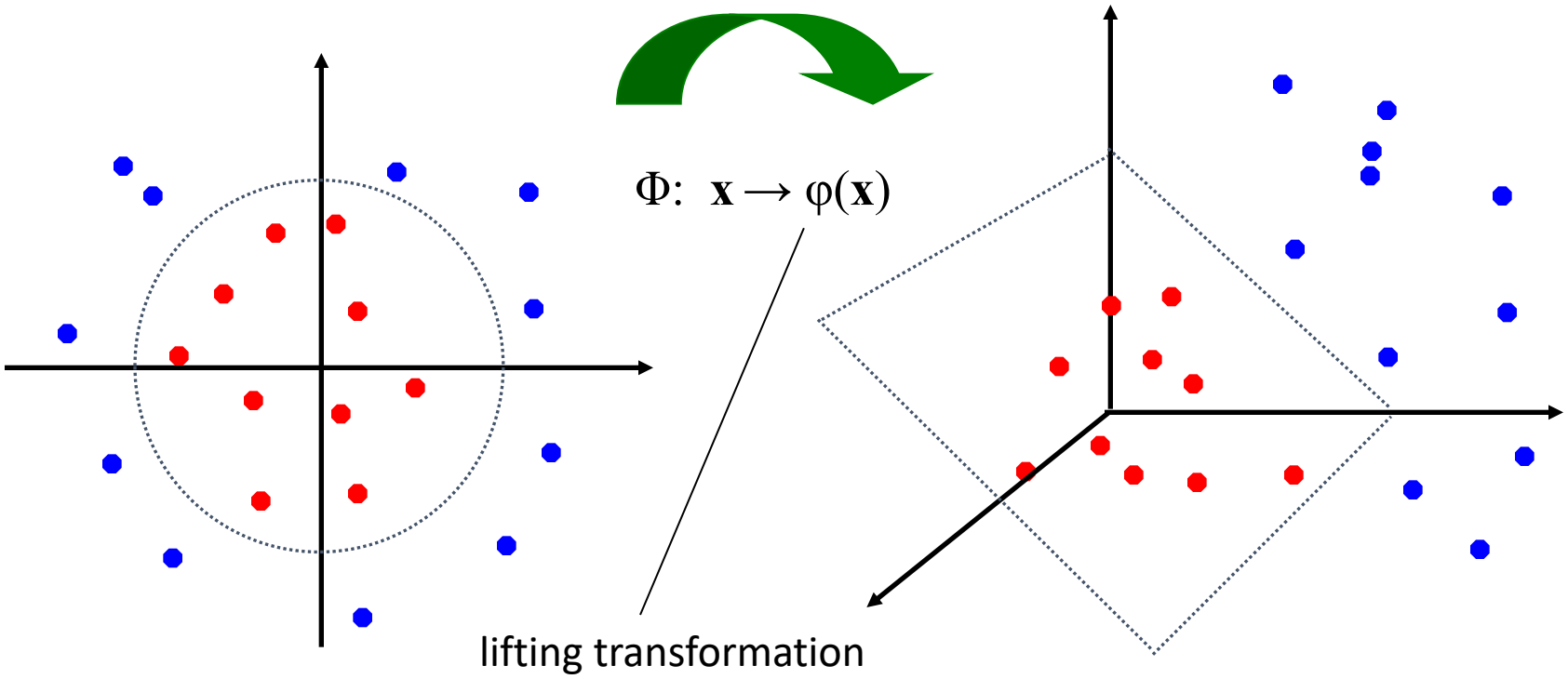
Support vector machines

- Find hyperplane that maximizes the *margin* between the positive and negative examples



Nonlinear SVMs

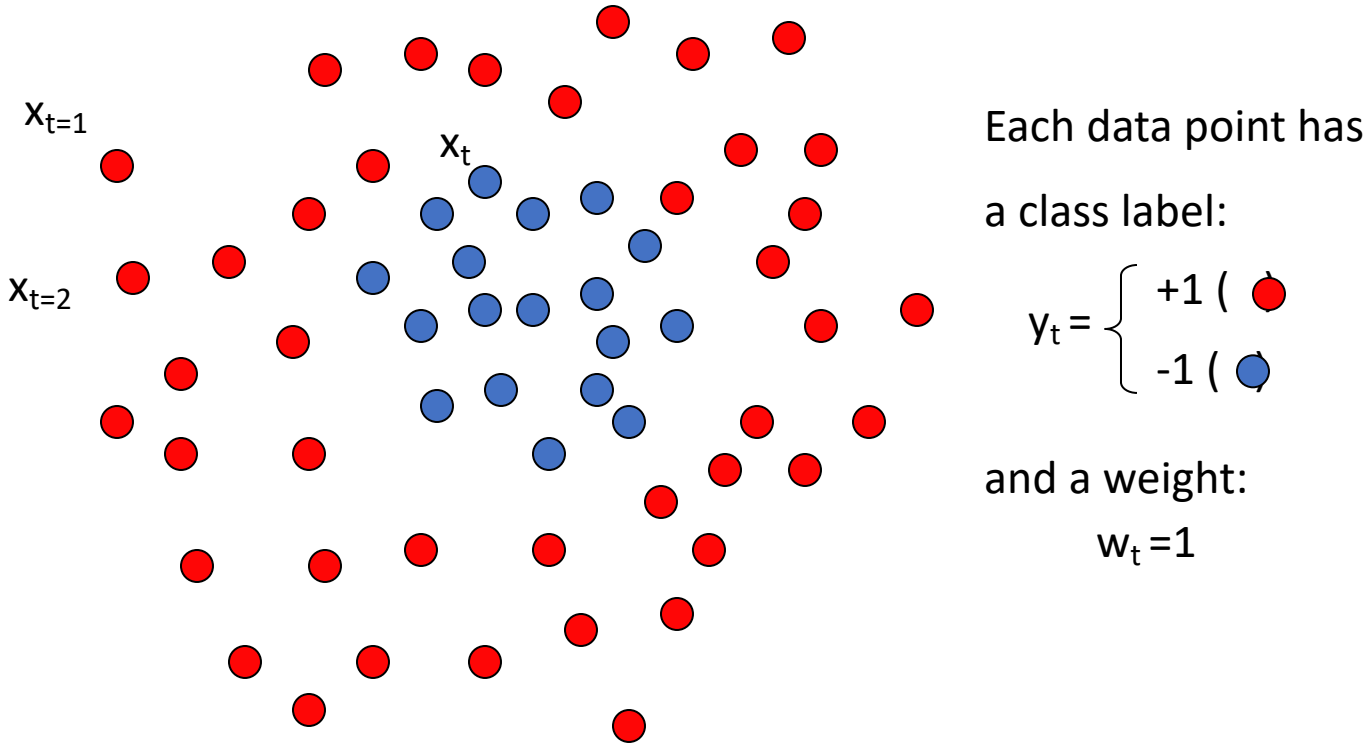
- Map original input space into some higher-dimensional feature space where the training set is separable:



Slide credit: Andrew Moore

Boosting

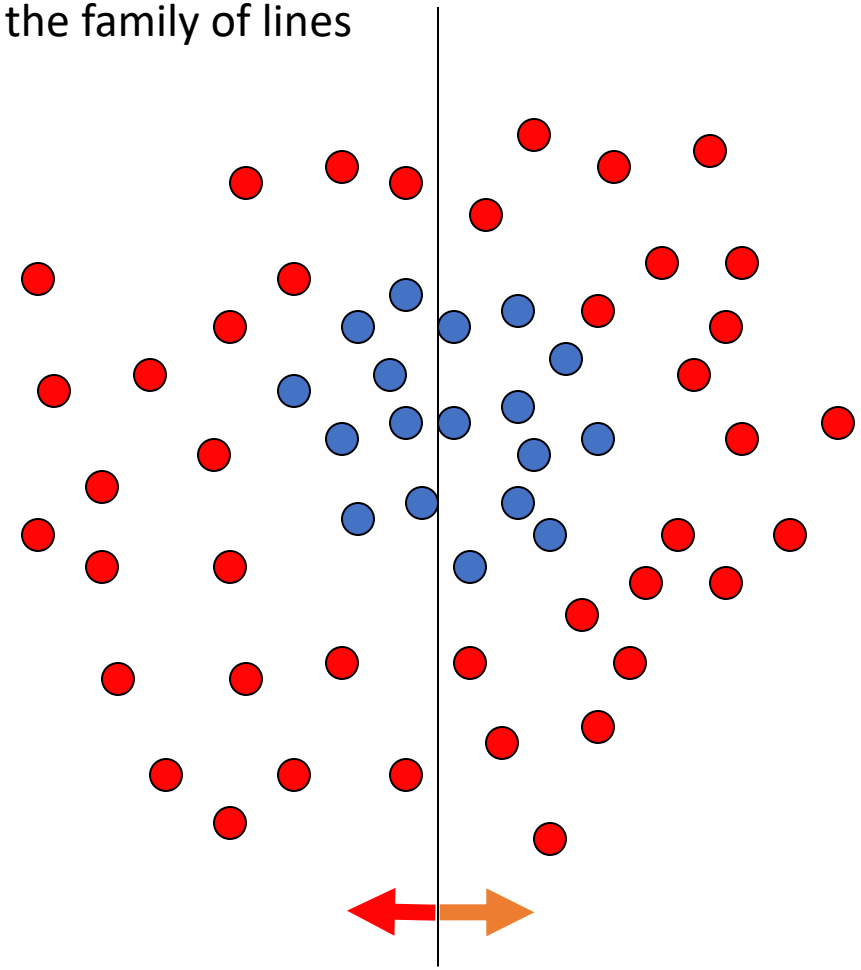
Y. Freund and R. Schapire, A short introduction to boosting, *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, September, 1999.



- It is a sequential procedure:

Toy example

Weak learners from the family of lines



Each data point has

a class label:

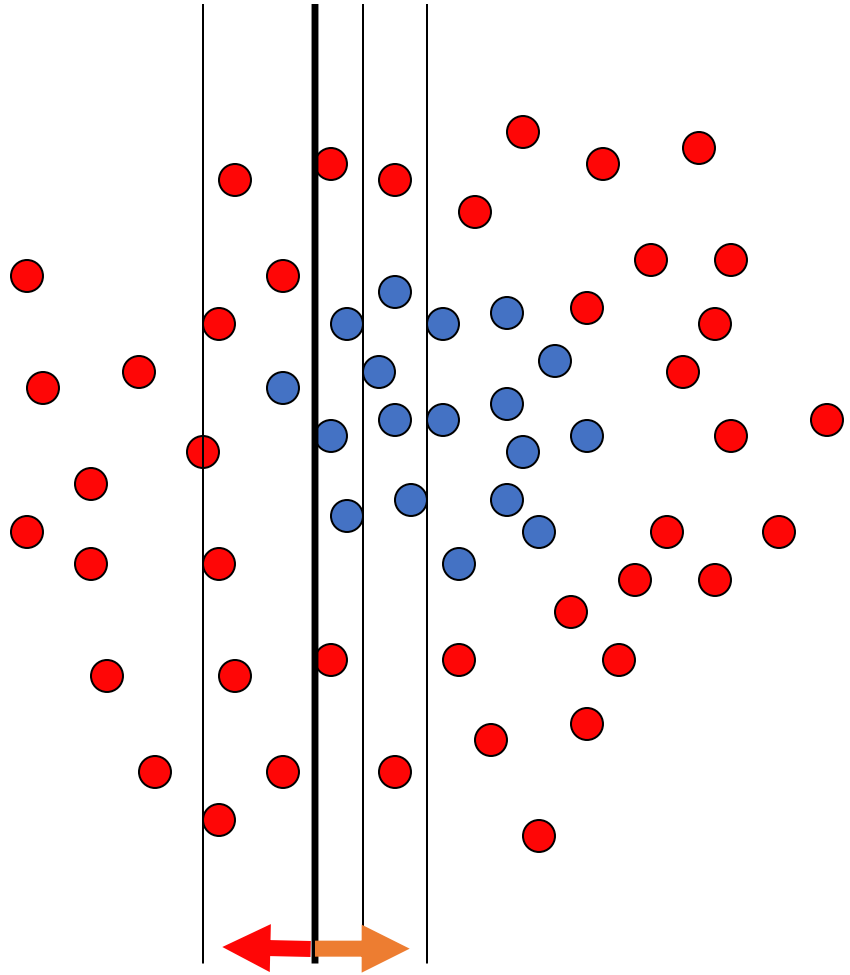
$$y_t = \begin{cases} +1 & (\\ -1 & (\end{cases}$$

and a weight:

$w_t = 1$

h => p(error) = 0.5 it is at chance

Toy example



This one seems to be the best

Each data point has

a class label:

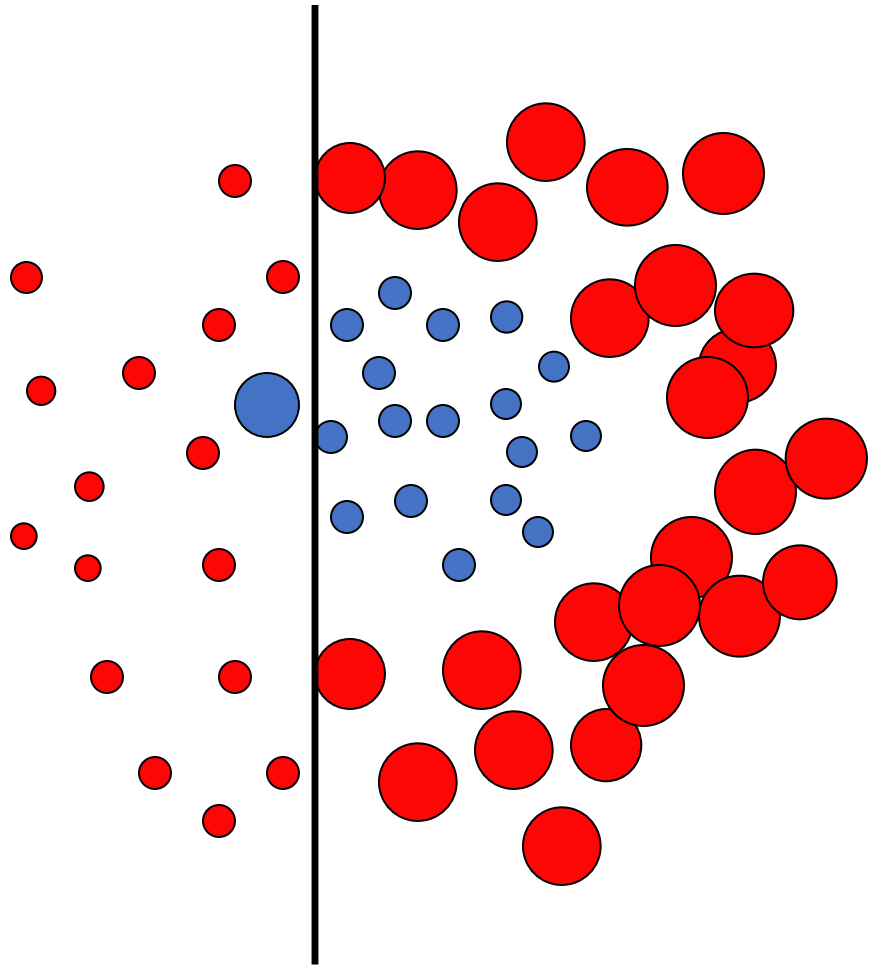
$$y_t = \begin{cases} +1 \ (\ ) \\ -1 \ (\ ) \end{cases}$$

and a weight:

$w_t = 1$

This is a '**weak classifier**': It performs slightly better than chance.

Toy example



Each data point has

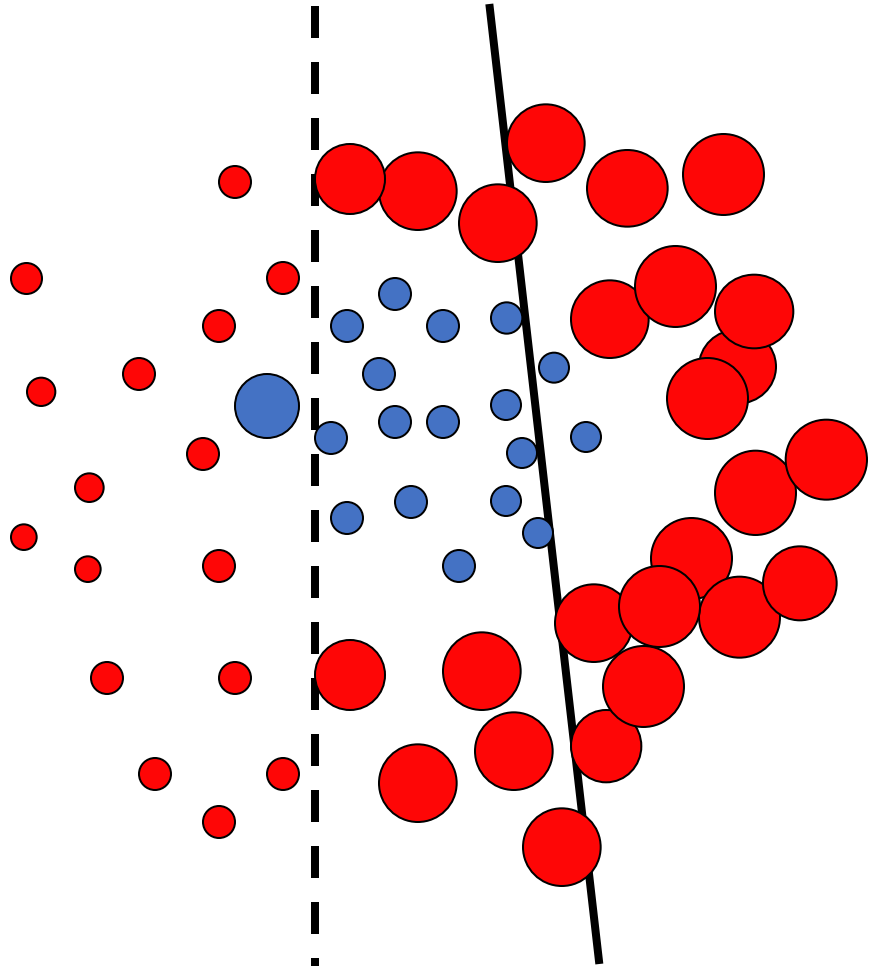
a class label:

$$y_t = \begin{cases} +1 & (\ ) \\ -1 & (\ ) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

Toy example



Each data point has

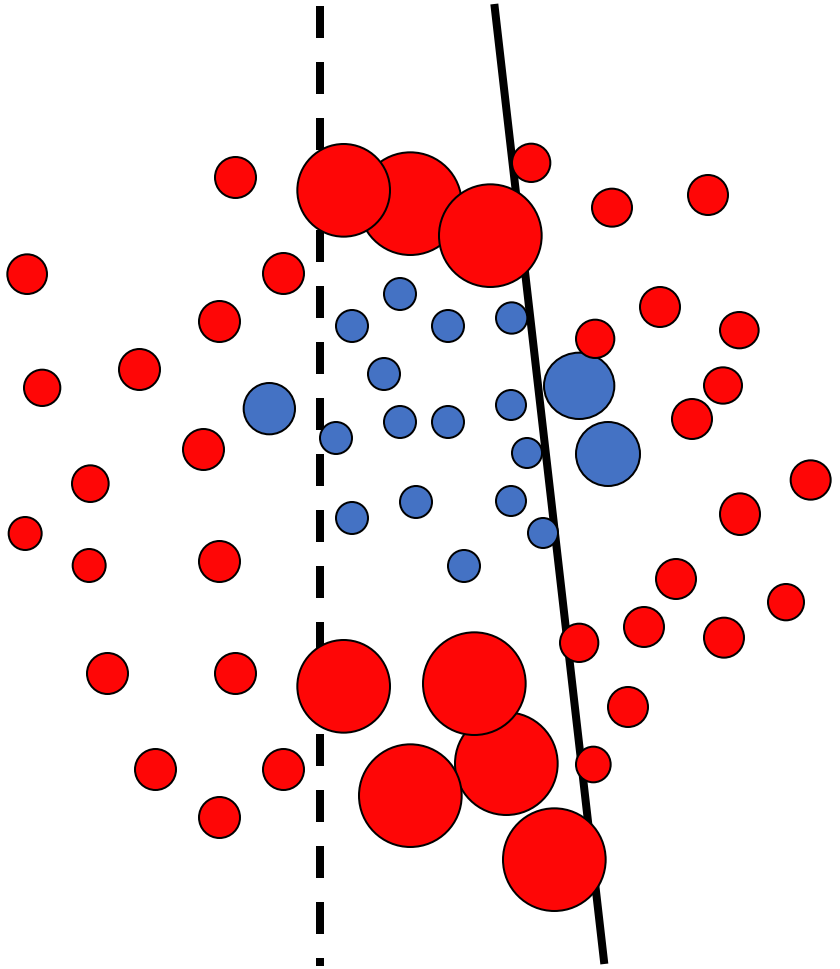
a class label:

$$y_t = \begin{cases} +1 \text{ (} \bullet \text{)} \\ -1 \text{ (} \bullet \text{)} \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

Toy example



Each data point has

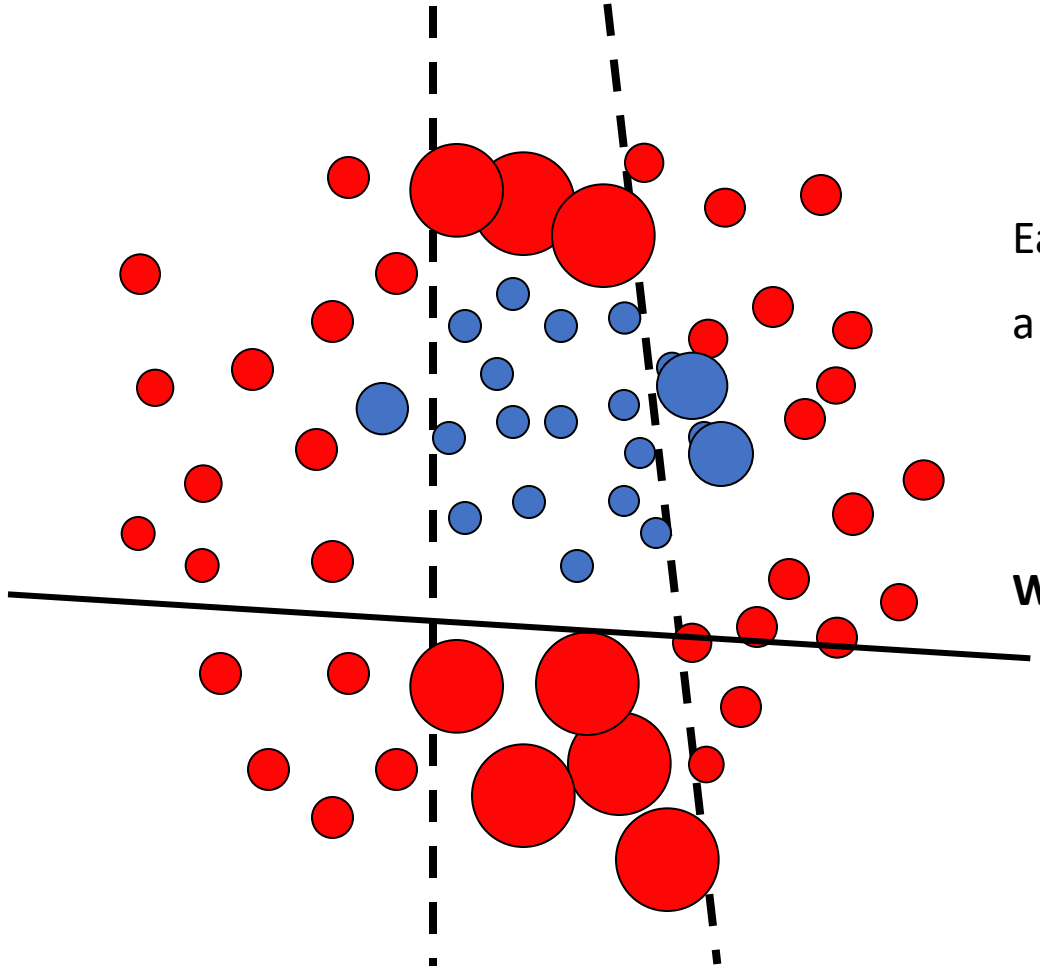
a class label:

$$y_t = \begin{cases} +1 & (\ ) \\ -1 & (\ ) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

Toy example



Each data point has

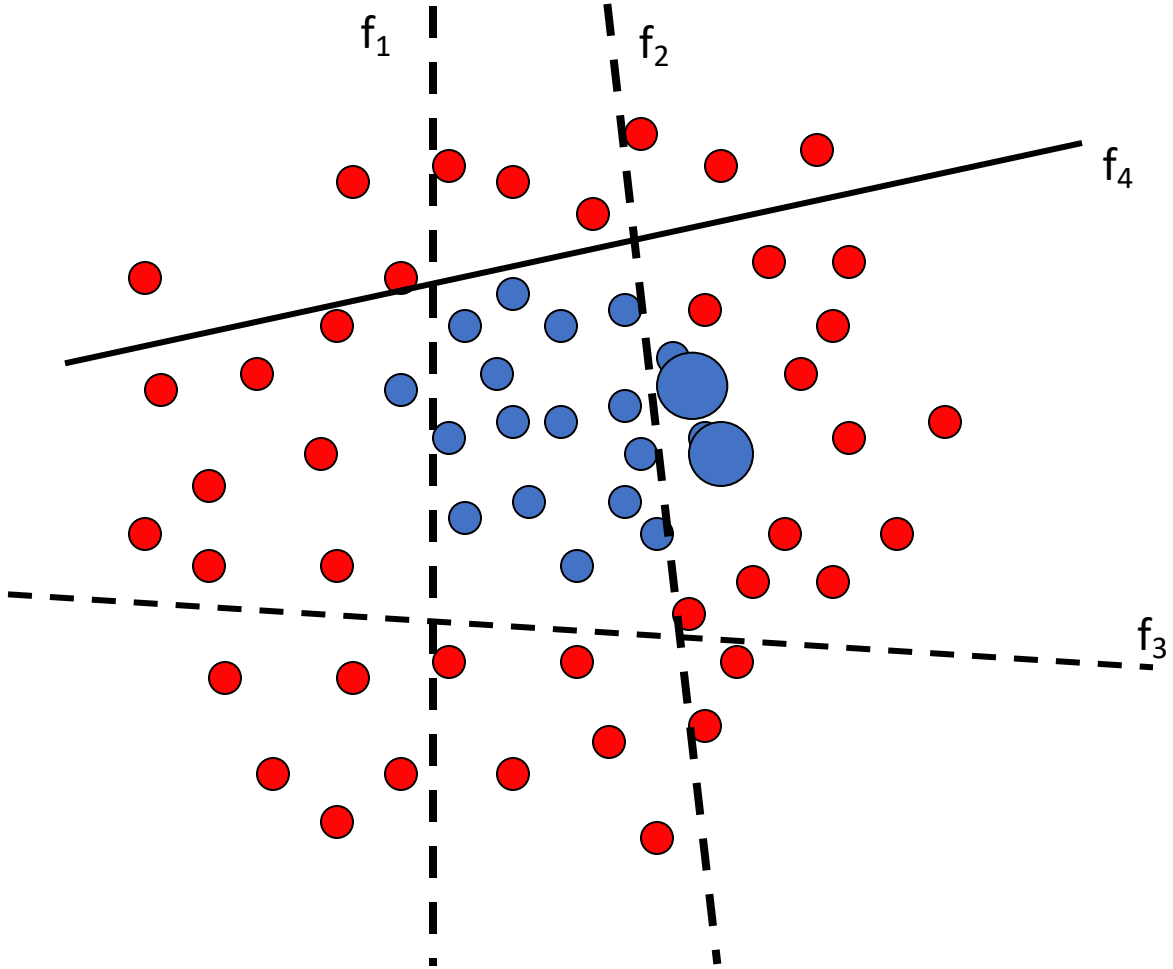
a class label:

$$y_t = \begin{cases} +1 & (\\ -1 & (\end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

Toy example



The strong (non- linear) classifier is built as the combination of all the weak (linear) classifiers.

Boosting - mathematics

- Weak learners

value of rectangle feature

$$h_j(x) = \begin{cases} 1 & \text{if } f_j(x) > \theta_j \\ 0 & \text{otherwise} \end{cases}$$

threshold

- Final strong classifier

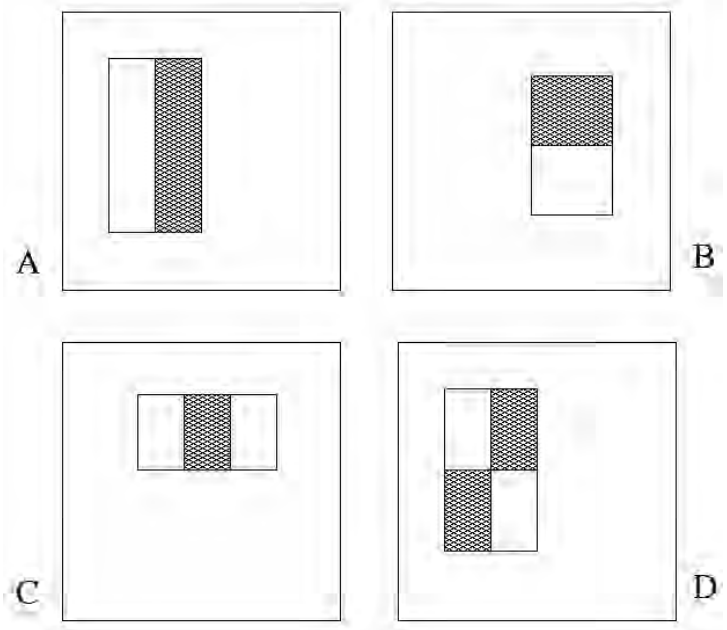
$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

Weak classifier

- 4 kind of Rectangle filters

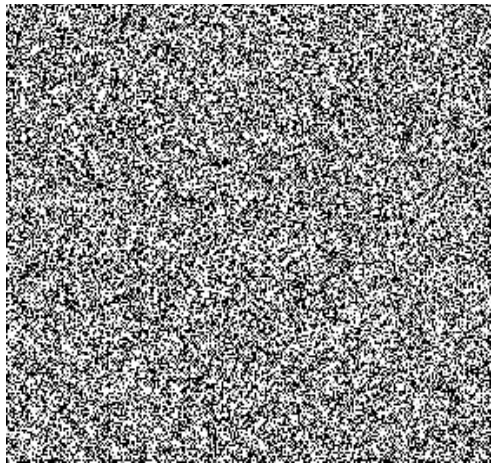
- *Value =*

∑ (pixels in white area) –
∑ (pixels in black area)



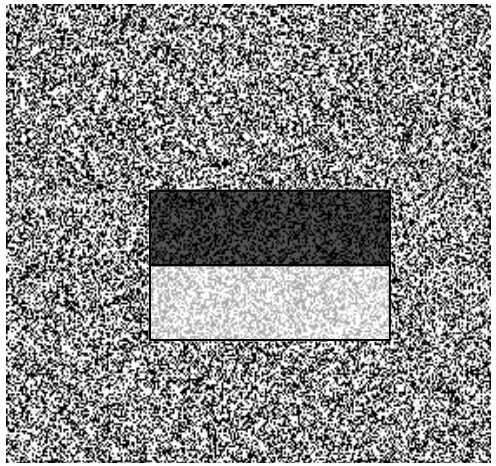
Credit slide: S. Lazebnik

Weak classifier



Source

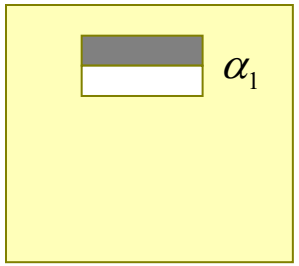
Result



Credit slide: S. Lazebnik

Weak classifier

1. Evaluate each rectangle filter on each example



$(x_1, 1)$ $(x_2, 1)$ $(x_3, 0)$ $(x_4, 0)$ $(x_5, 0)$ $(x_6, 0)$

$\cdots\cdots$ (x_n, y_n)

0.8 0.7 0.2 0.3 0.8 0.1

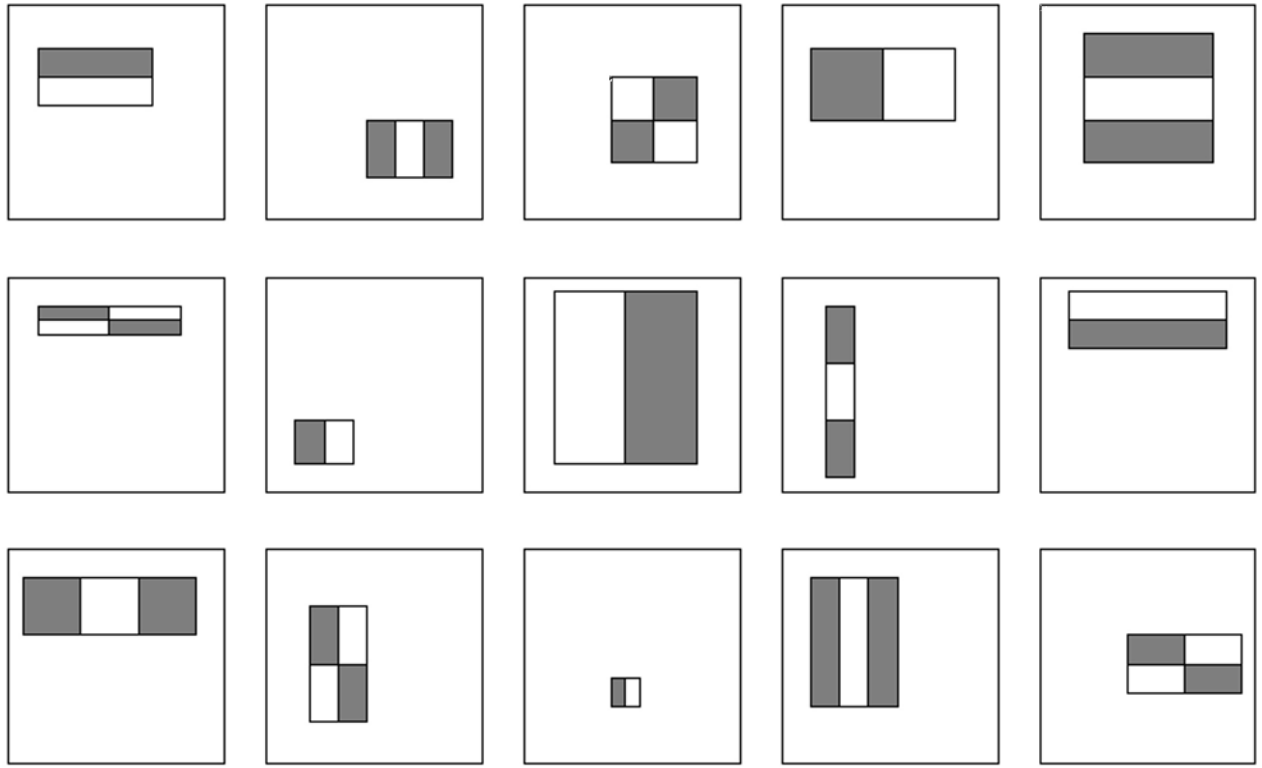
$$h_j(x) = \begin{cases} 1 & \text{if } f_j(x) > \theta_j \\ 0 & \text{otherwise} \end{cases}$$

threshold

Weak classifier

Weak classifier

- For a 24x24 detection region,



Feature selection

Non discriminative features

Discriminative features

The Viola/Jones Face Detector

P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*. CVPR 2001.

- A “paradigmatic” method for real-time object detection
- Training is slow, but detection is very fast
- Key ideas
  - *Integral images* for fast feature evaluation
  - *Boosting* for feature selection
  - *Attentional cascade* for fast rejection of non-face windows

Credit slide: S. Lazebnik

The implemented system

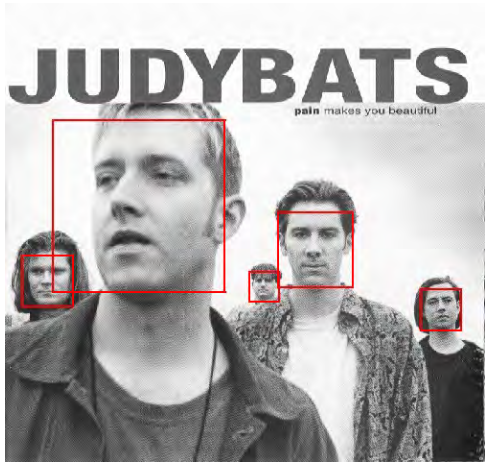
- Training Data
  - 5000 faces
    - All frontal, rescaled to 24x24 pixels
  - 300 million non-faces
    - 9500 non-face images
  - Faces are normalized
    - Scale, translation
- Many variations
  - Across individuals
  - Illumination
  - Pose

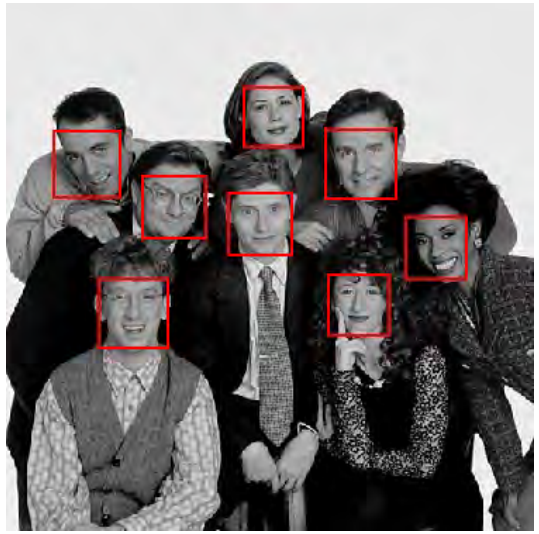
(Most slides from Paul Viola)

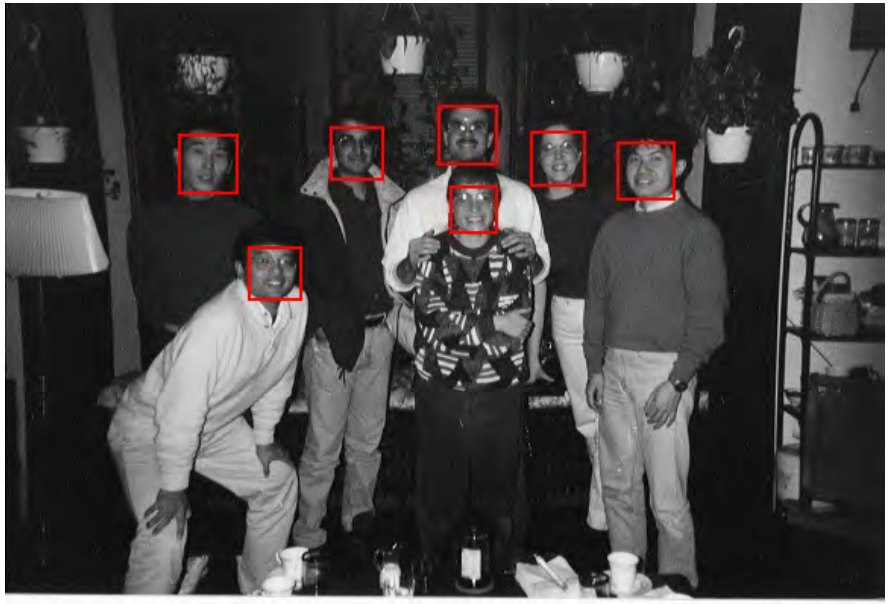
System performance

- Training time: “weeks” on 466 MHz Sun workstation
- 38 layers, total of 6061 features
- Average of 10 features evaluated per window on test set
- “On a 700 Mhz Pentium III processor, the face detector can process a 384 by 288 pixel image in about .067 seconds”
  - 15 Hz
  - 15 times faster than previous detector of comparable accuracy (Rowley et al., 1998)

Output of Face Detector on Test Images





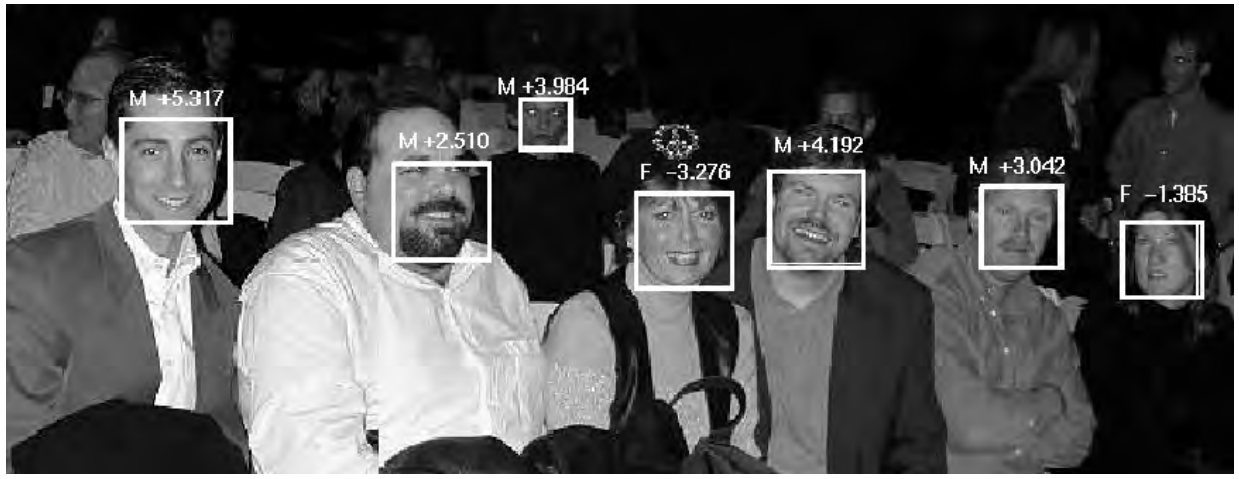


Other detection tasks

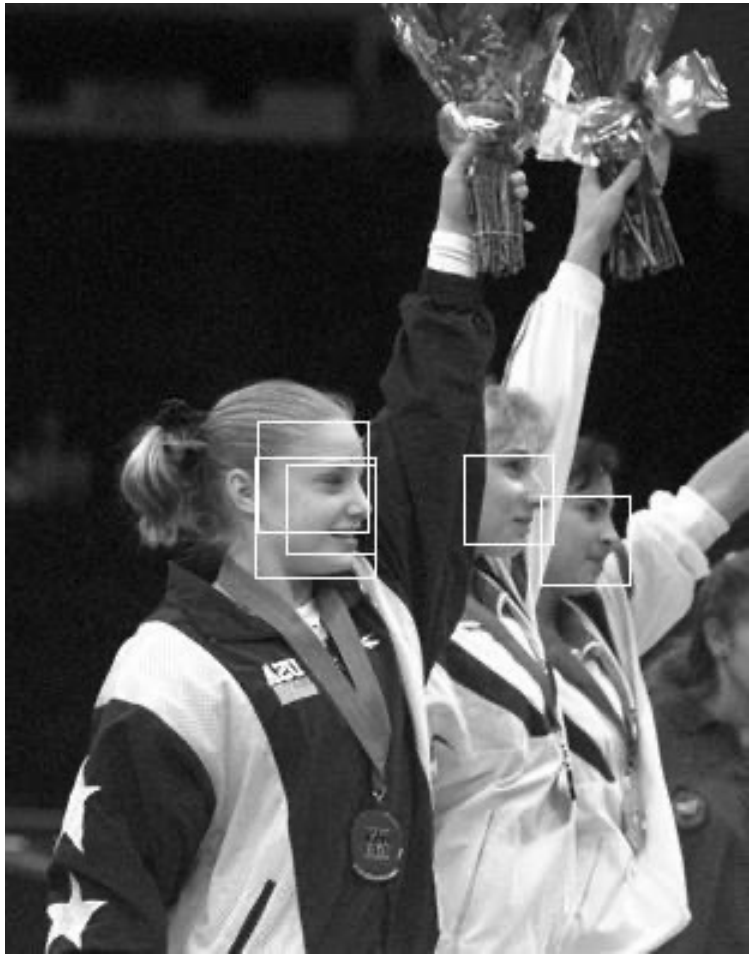
Facial Feature Localization

Profile Detection

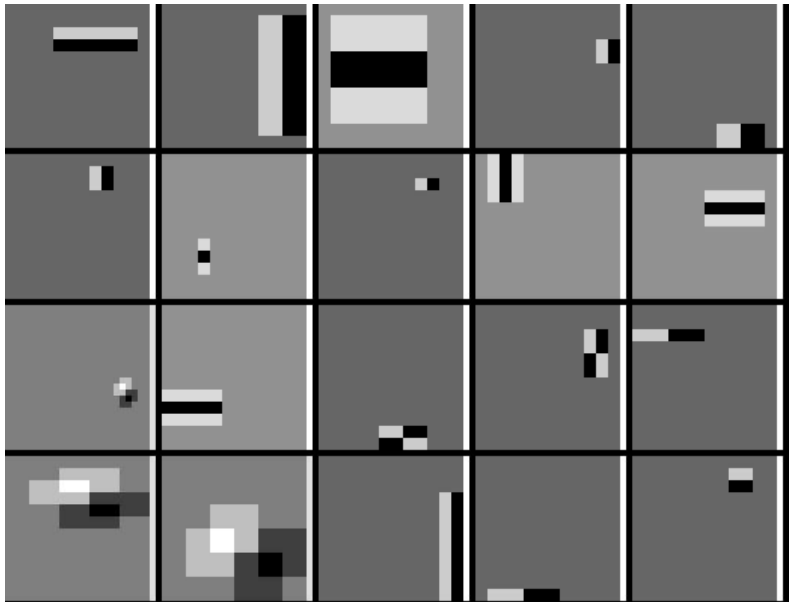
Male vs. female



Profile Detection



Profile Features



Usual Challenges:

Variability due to:

- View point
- Illumination
- Occlusions

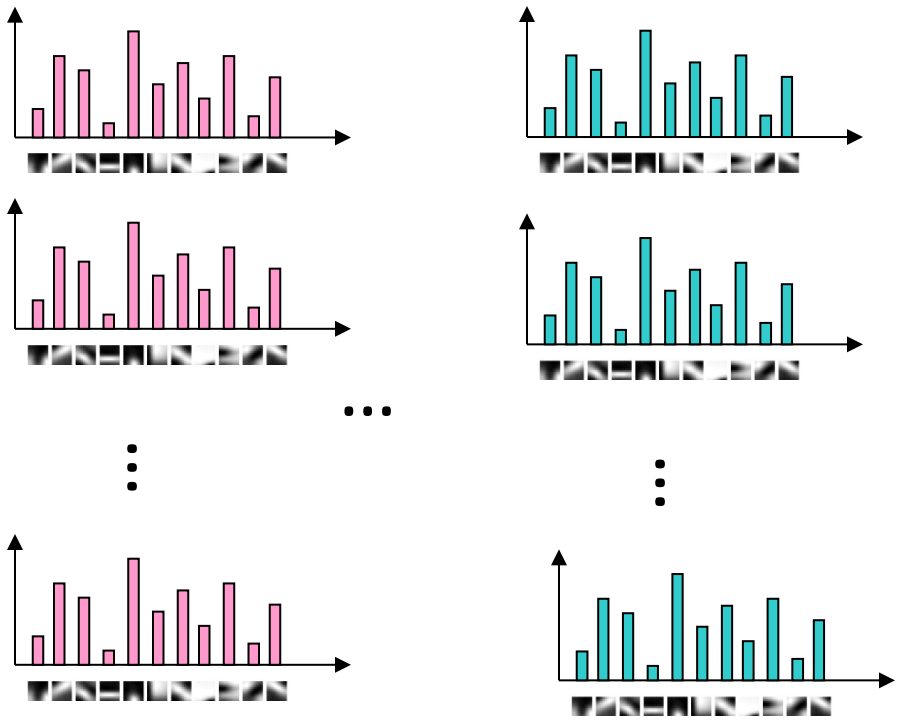
Learning and Recognition

1. Discriminative method:
   - NN
   -Neural Networks

2. Generative method

Discriminative classifiers

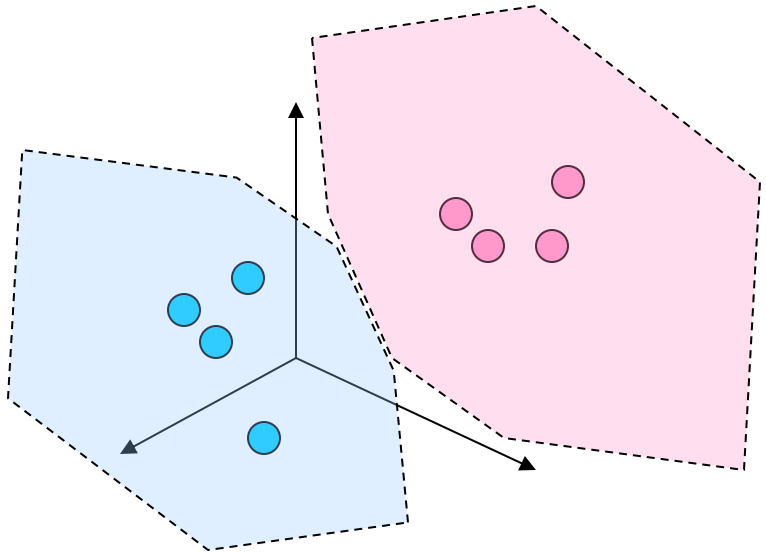
category models



Class 1

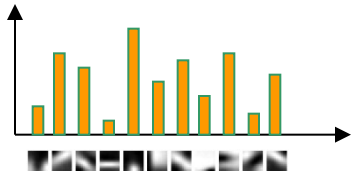
Class N

Model space



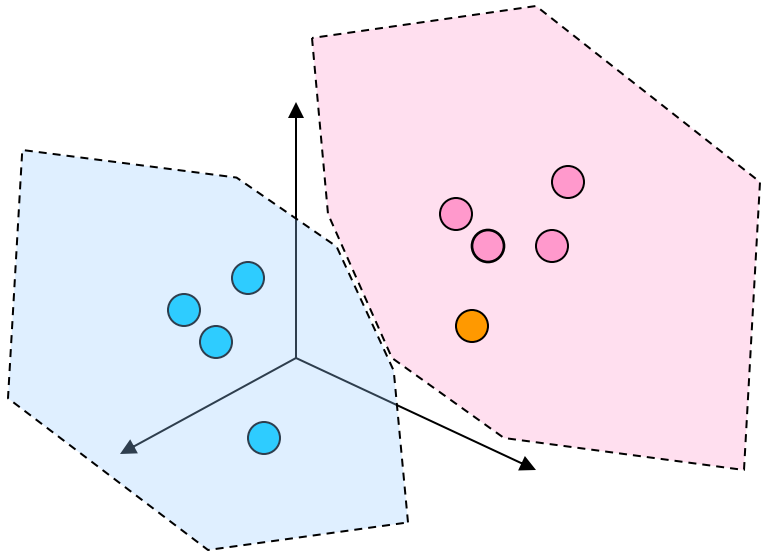
Discriminative classifiers

Query image



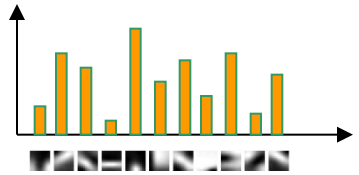
Winning class: pink

Model space

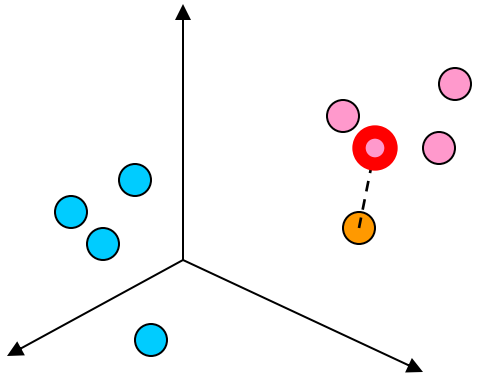


Nearest Neighbors classifier

Query image



Model space

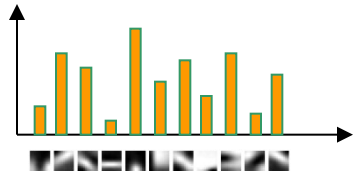


Winning class: pink

- Assign label of nearest training data point to each test data point

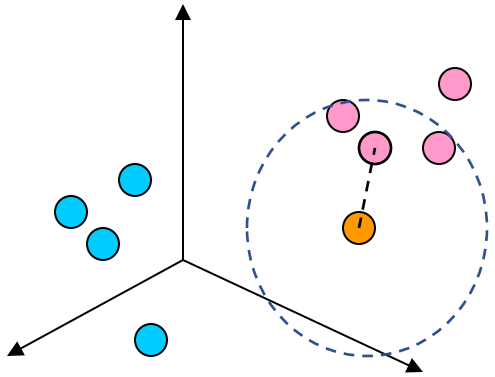
K- Nearest Neighbors classifier

Query image



Winning class: pink

Model space



- For a new point, find the k closest points from training data
- Labels of the k points “vote” to classify
- Works well provided there is lots of data and the distance function is good

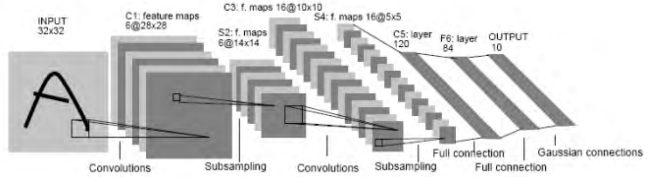
Discriminative models

Nearest neighbor

10^6 examples

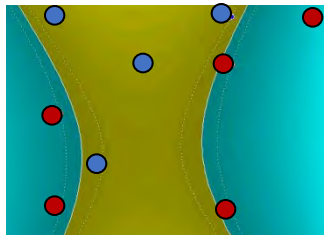
Shakhnarovich, Viola, Darrell 2003
Berg, Berg, Malik 2005...

Neural networks



LeCun, Bottou, Bengio, Haffner 1998
Rowley, Baluja, Kanade 1998
…

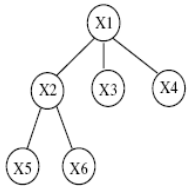
Support Vector Machines



Guyon, Vapnik, Heisele, Serre, Poggio…

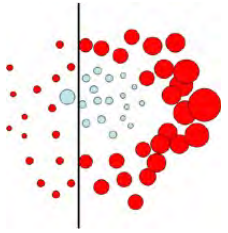
Latent SVM

Structural SVM



Felzenszwalb 00
Ramanan 03…

Boosting



Viola, Jones 2001,
Torralba et al. 2004,
Opelt et al. 2006,…

Courtesy of Vittorio Ferrari
Slide credit: Kristen Grauman

Slide adapted from Antonio Torralba

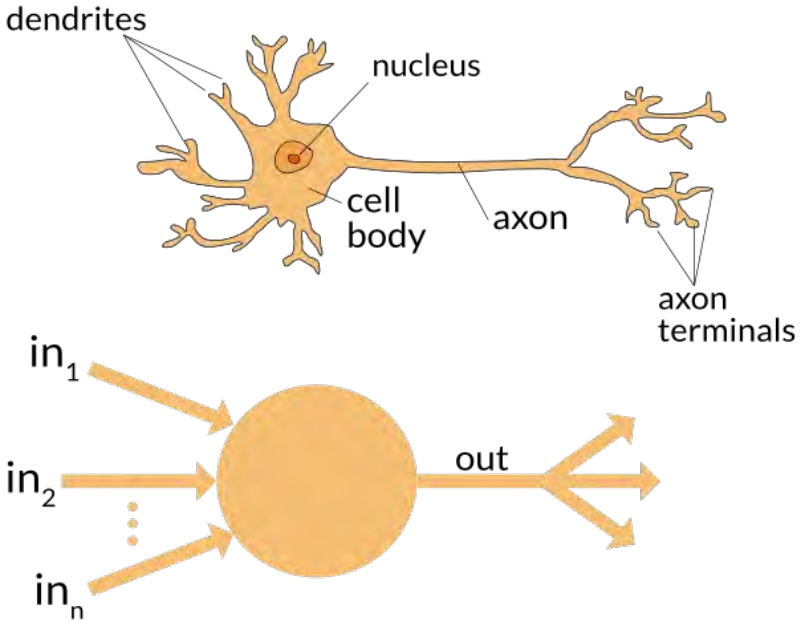
Learning and Recognition

1. Discriminative method:
   - NN
   - Neural Network
2. Generative method:

Neural Network

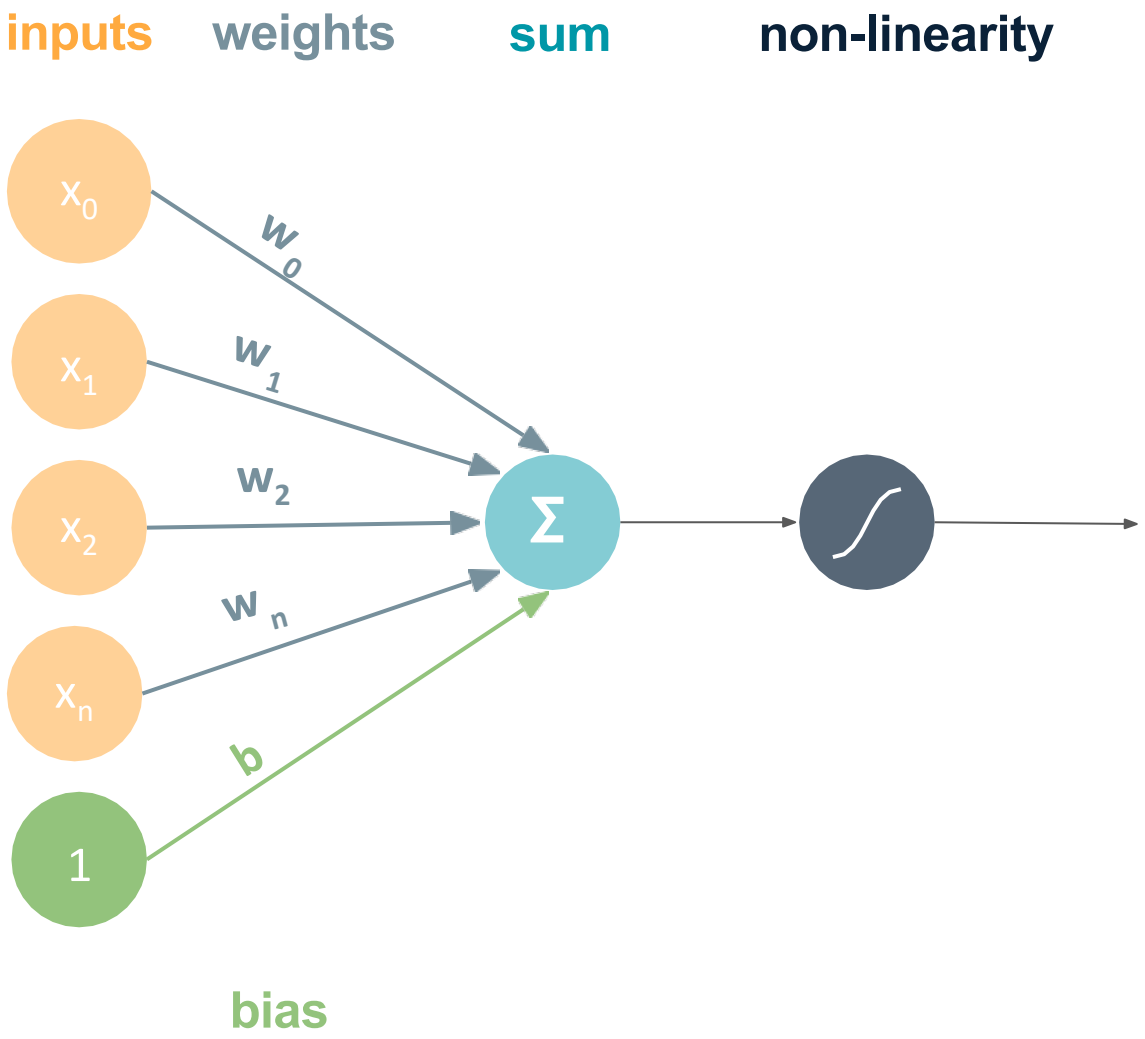
The Perceptron

1. Invented in 1954 by Frank Rosenblatt
2. Inspired by neurobiology



(Stolen from N. Locascio)

The Perceptron

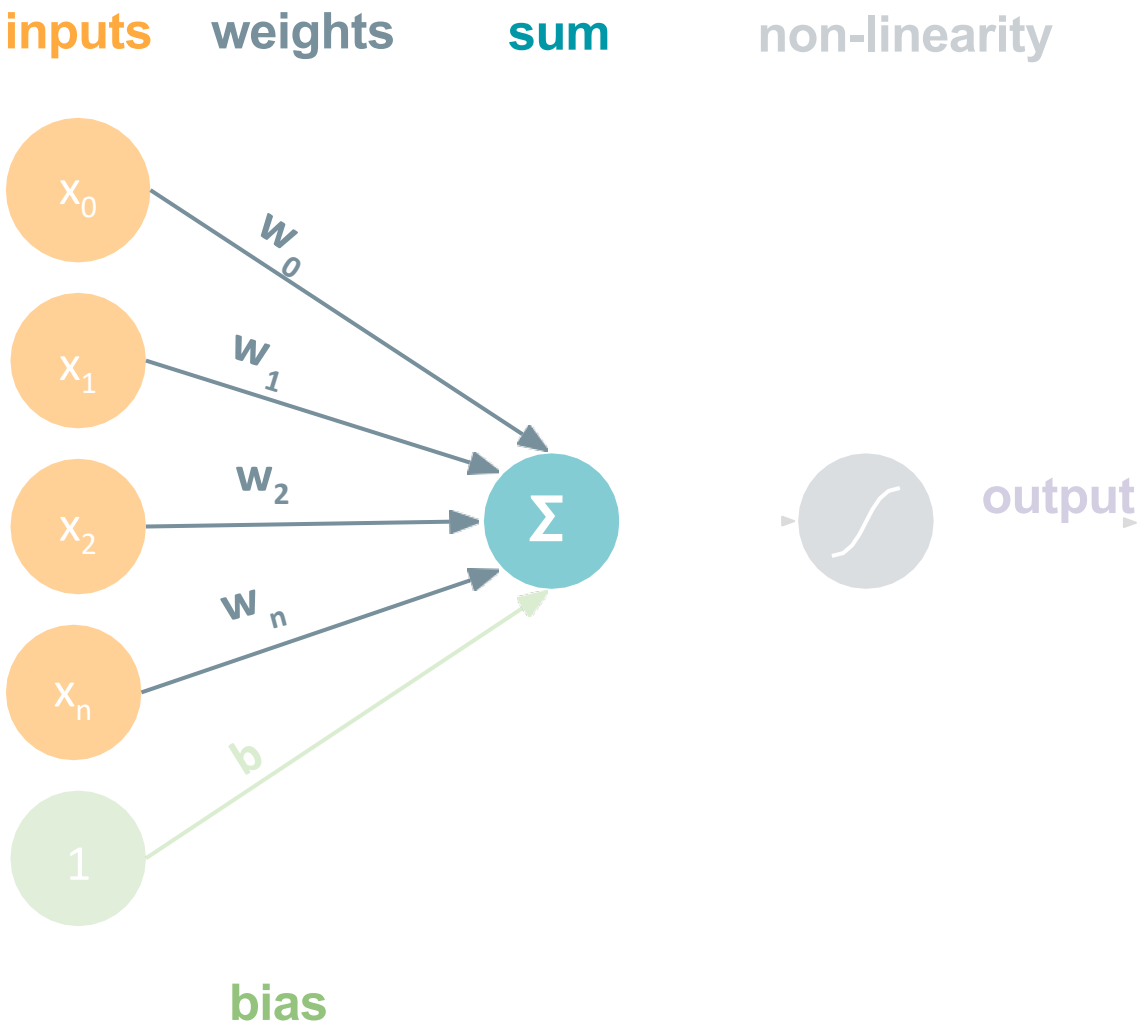


Perceptron Forward Pass

$$output =$$

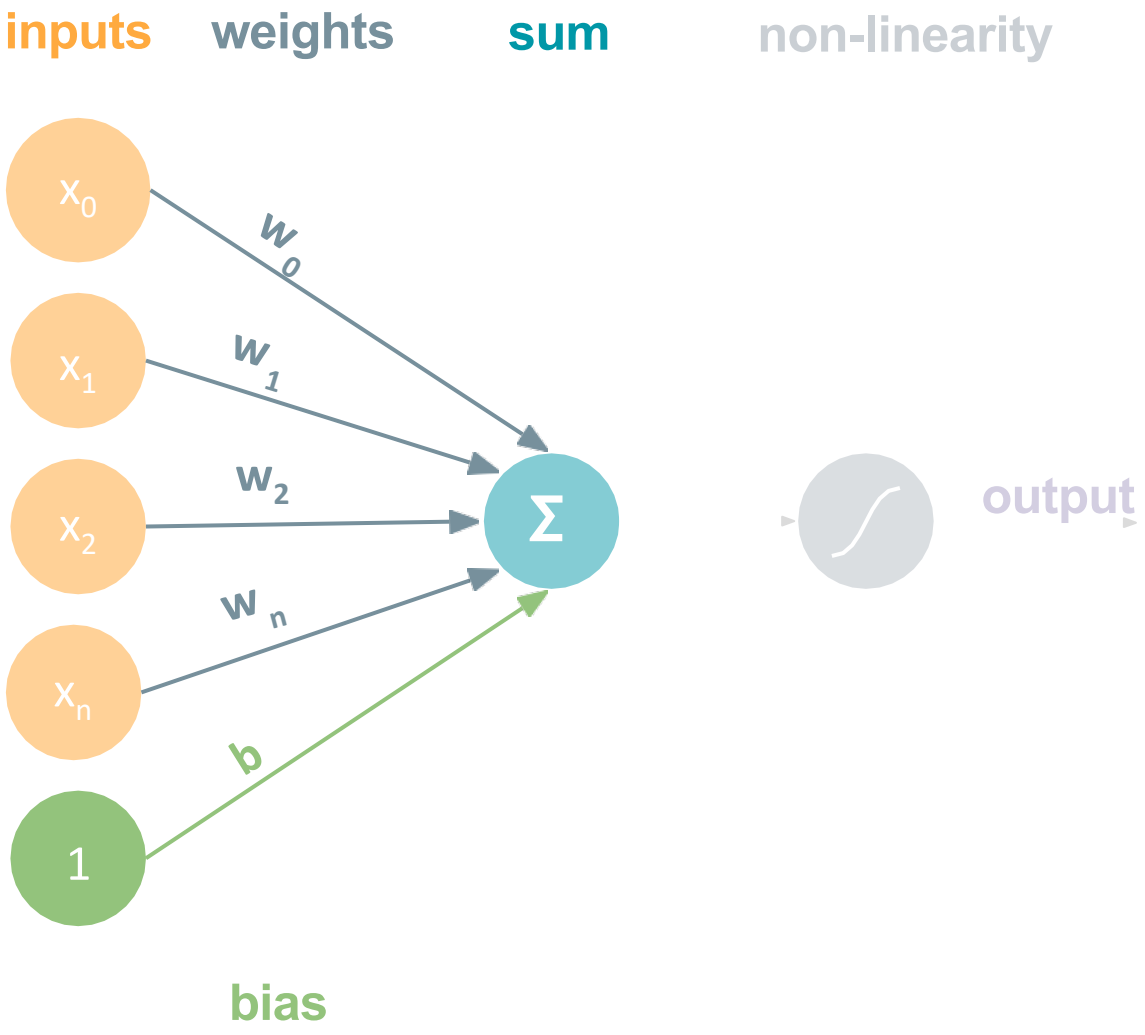
Perceptron Forward Pass

$$output = \sum_{i=0}^{N} x_i * w_i$$



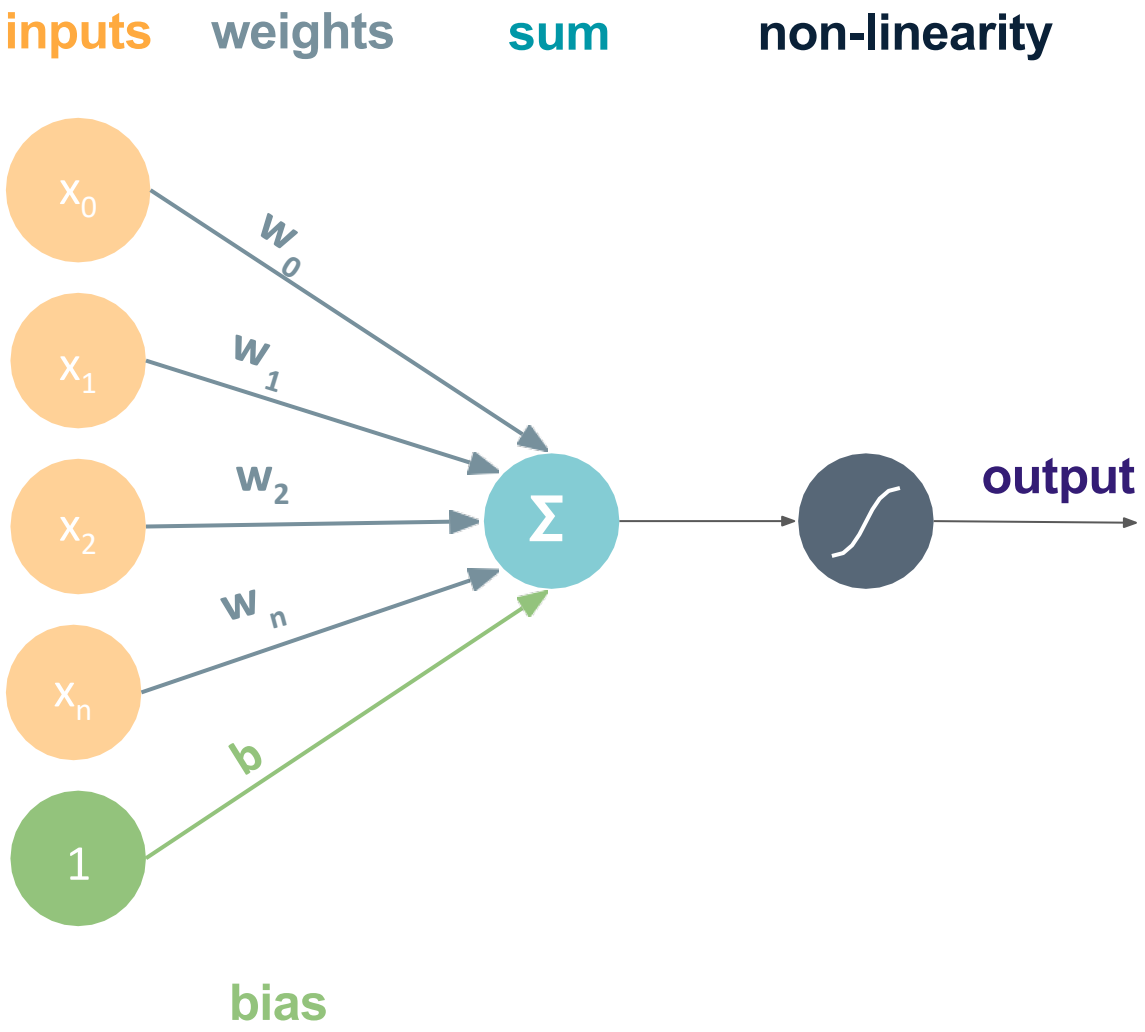
Perceptron Forward Pass

$$output = (\sum_{i=0}^{N} x_i * w_i) + b$$



Perceptron Forward Pass

$$output = g((\sum_{i=0}^{N} x_i * w_i) + b)$$

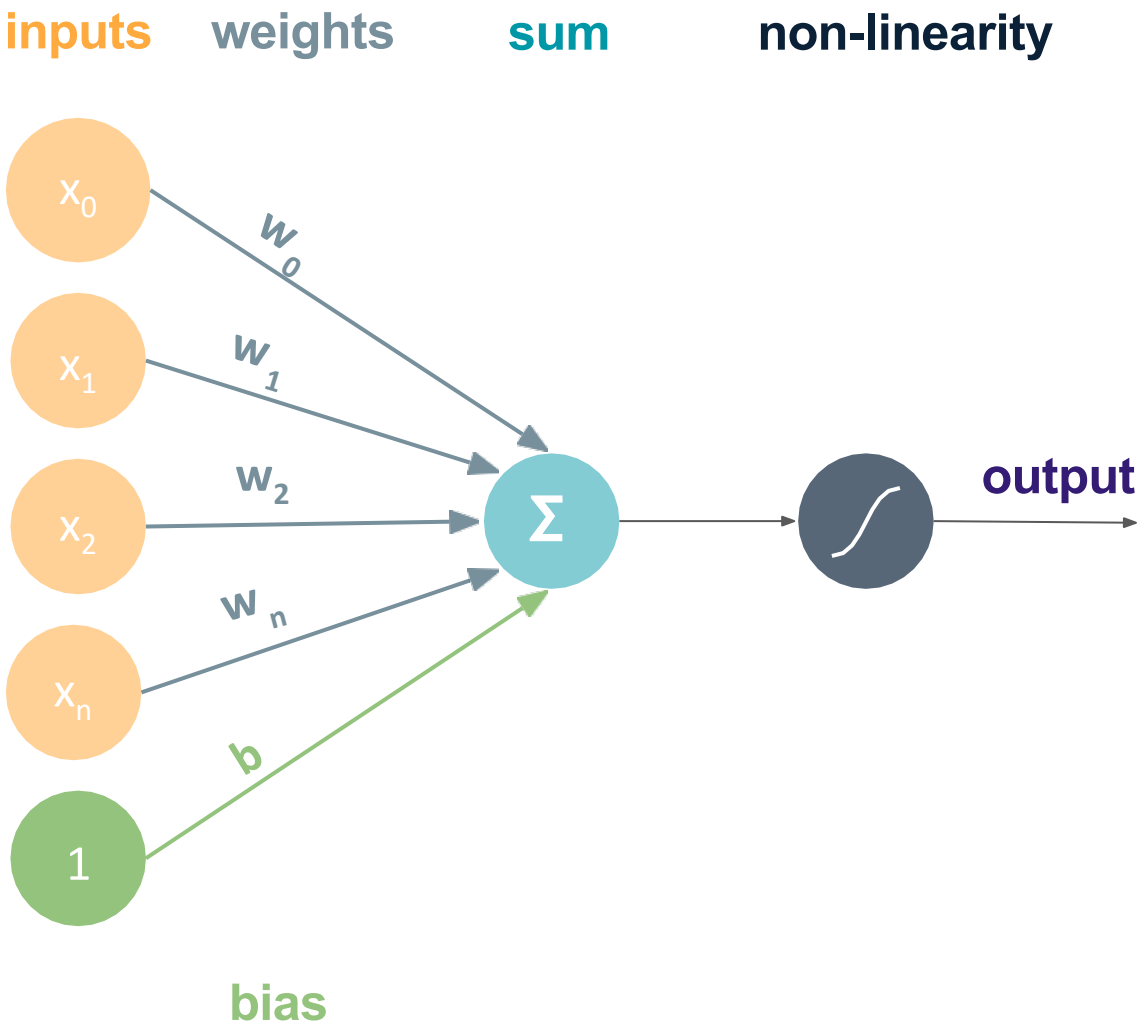


Perceptron Forward Pass

$$output = g(XW + b)$$

$$X = x_0, x_1, \ldots x_n$$

$$W = w_0, w_1, \ldots w_n$$



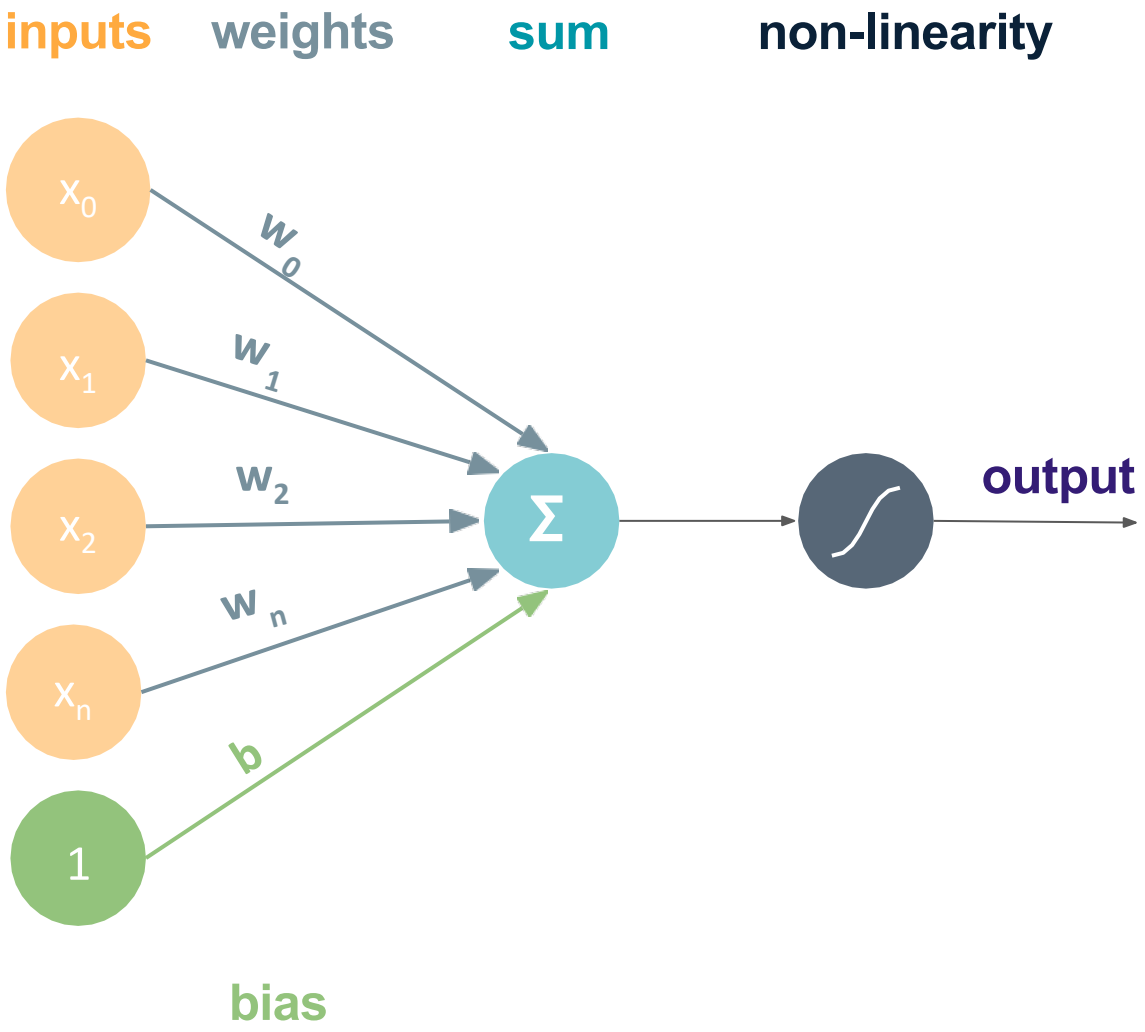
Perceptron Forward Pass

Activation Function

$$output = g(XW + b)$$

$$X = x_0, x_1, \ldots x_n$$

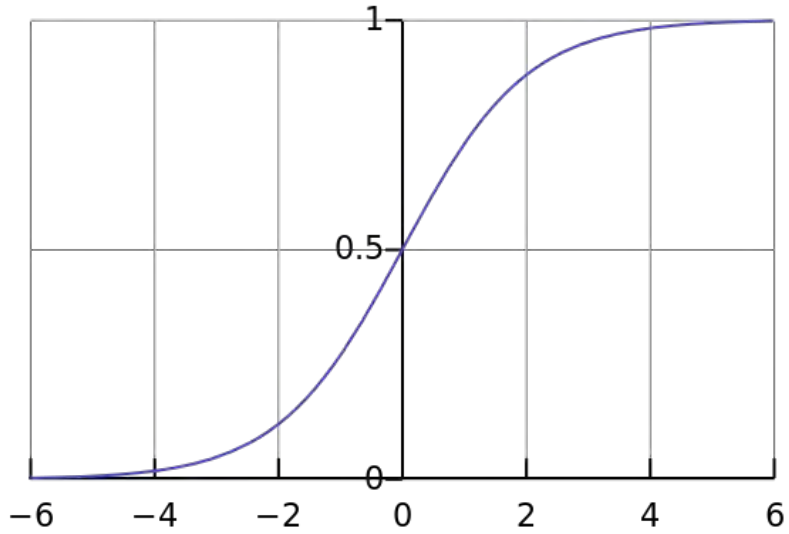
$$W = w_0, w_1, \ldots w_n$$

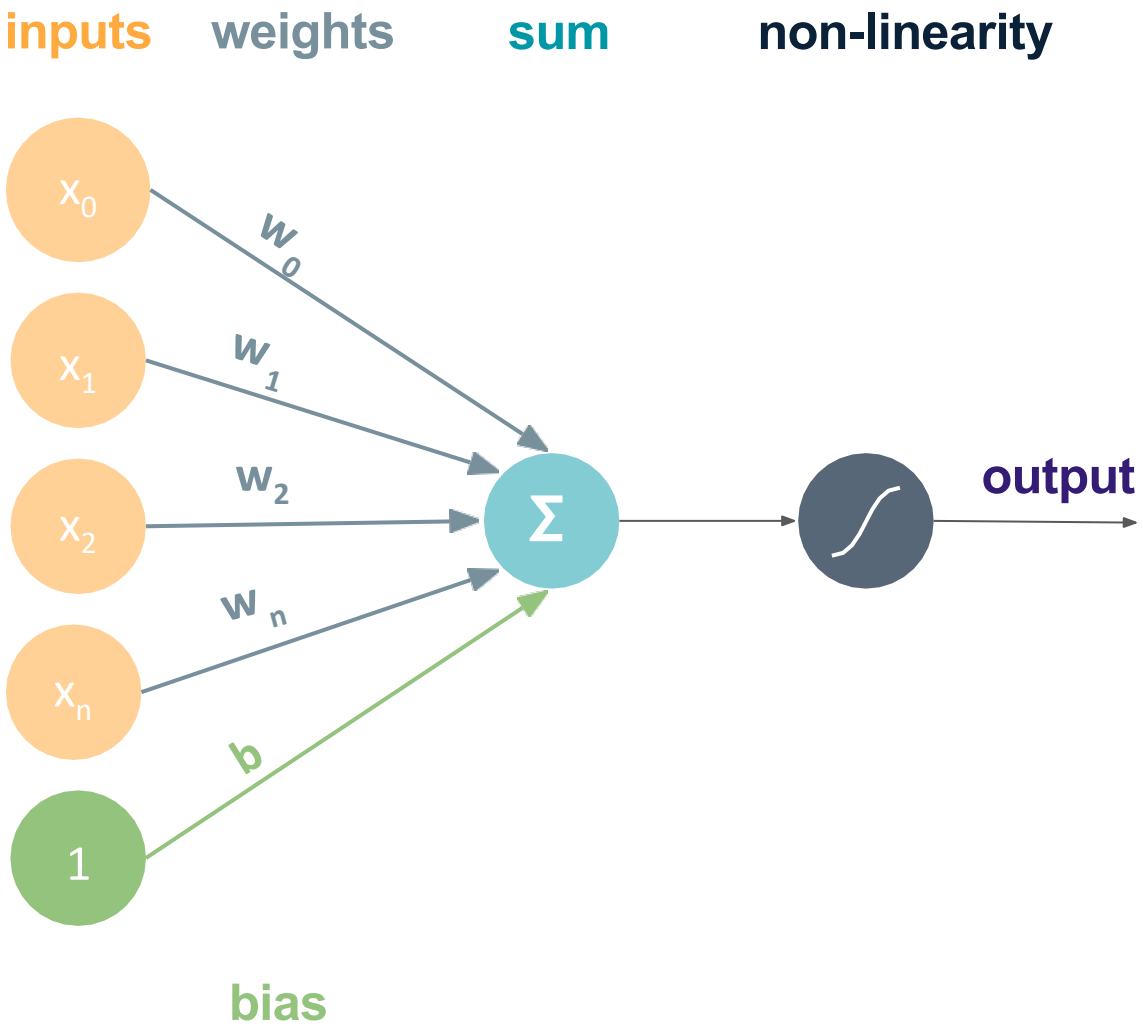


Sigmoid Activation

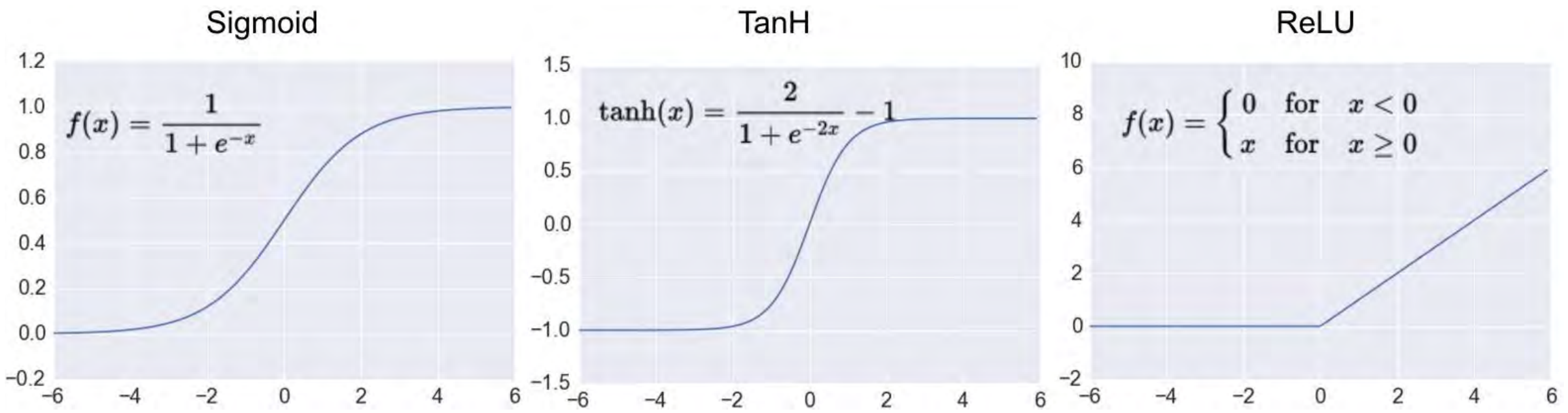
$$output = g(XW + b)$$

$$g(a) = \sigma(a) = \frac{1}{1 + e^{-a}}$$





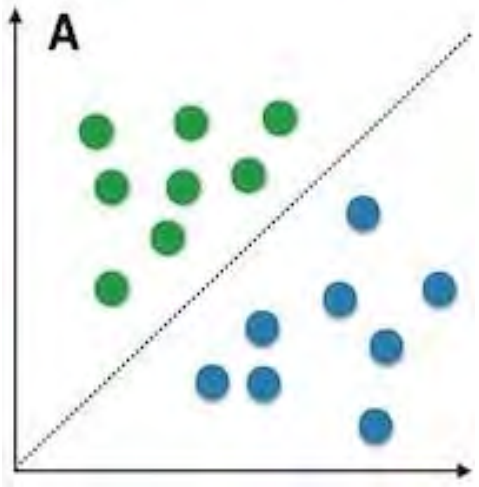
Common Activation Functions

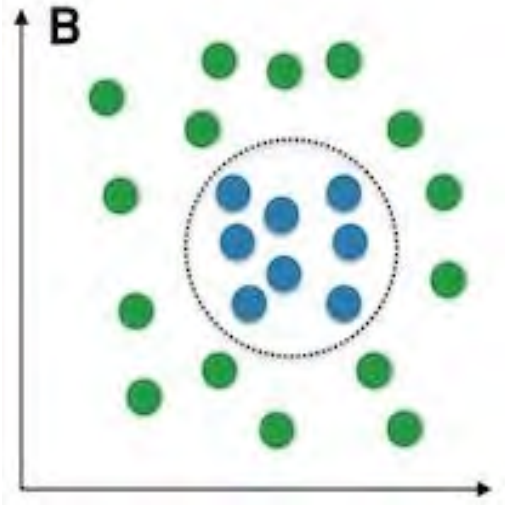


(Stolen from N. Locascio)

Importance of Activation Functions

- Activation functions add non-linearity to our network's function
- Most real-world problems + data are **non-linear**

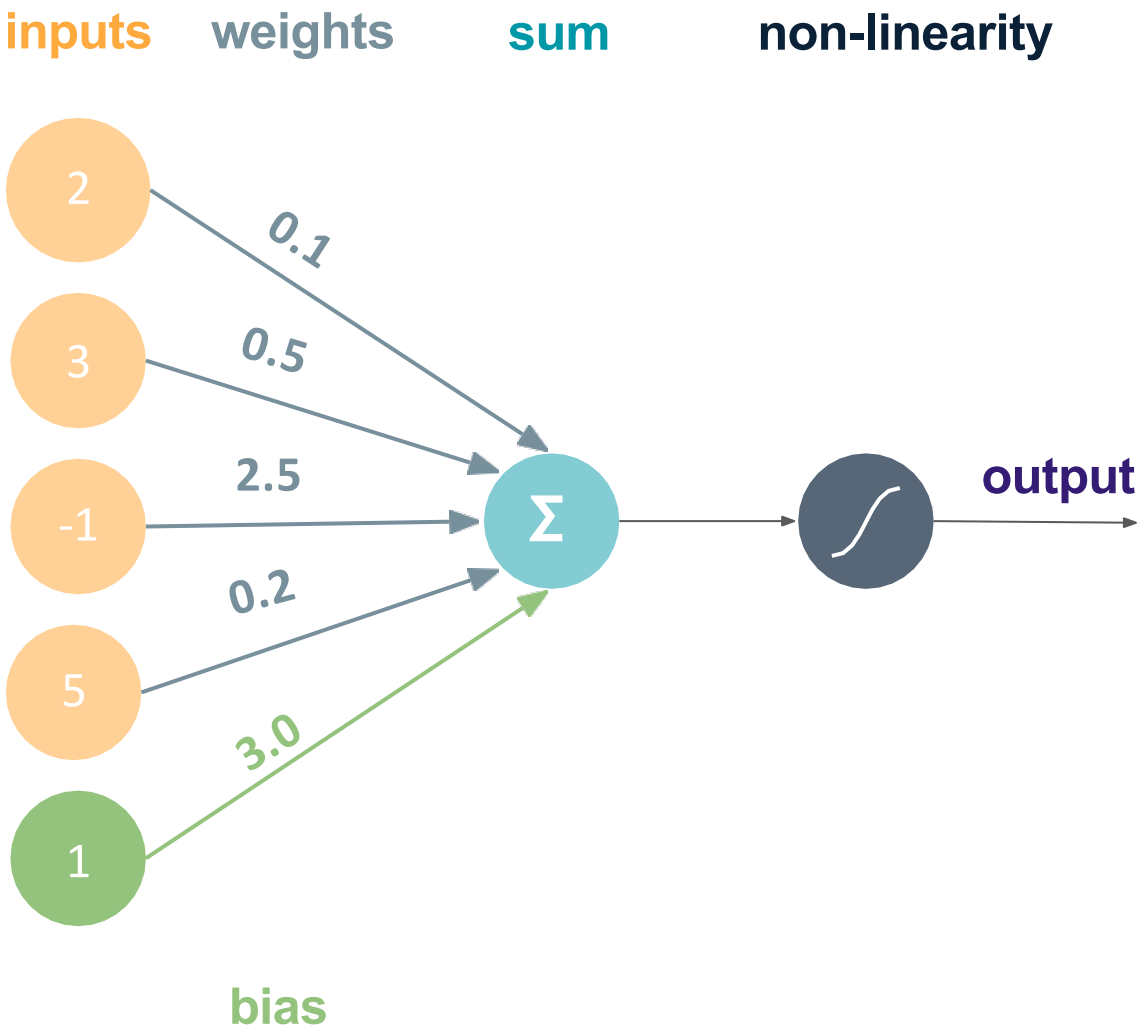




(Stolen from N. Locascio)

Perceptron Forward Pass

$$output = g(XW + b)$$



Perceptron Forward Pass

$$output = g($$

(2*0.1) +

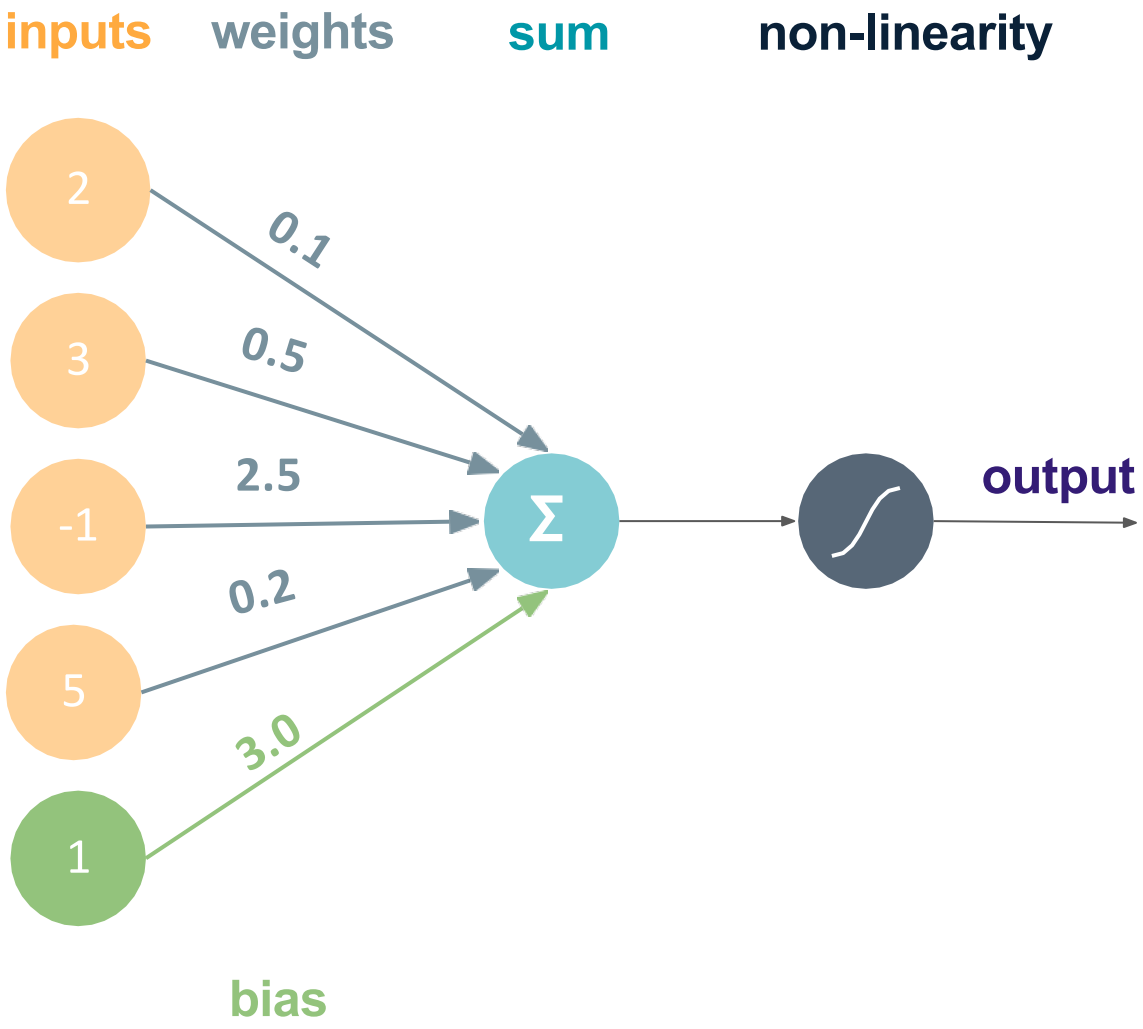
(3*0.5) +

(-1*2.5) +

(5*0.2) +

(1*3.0)

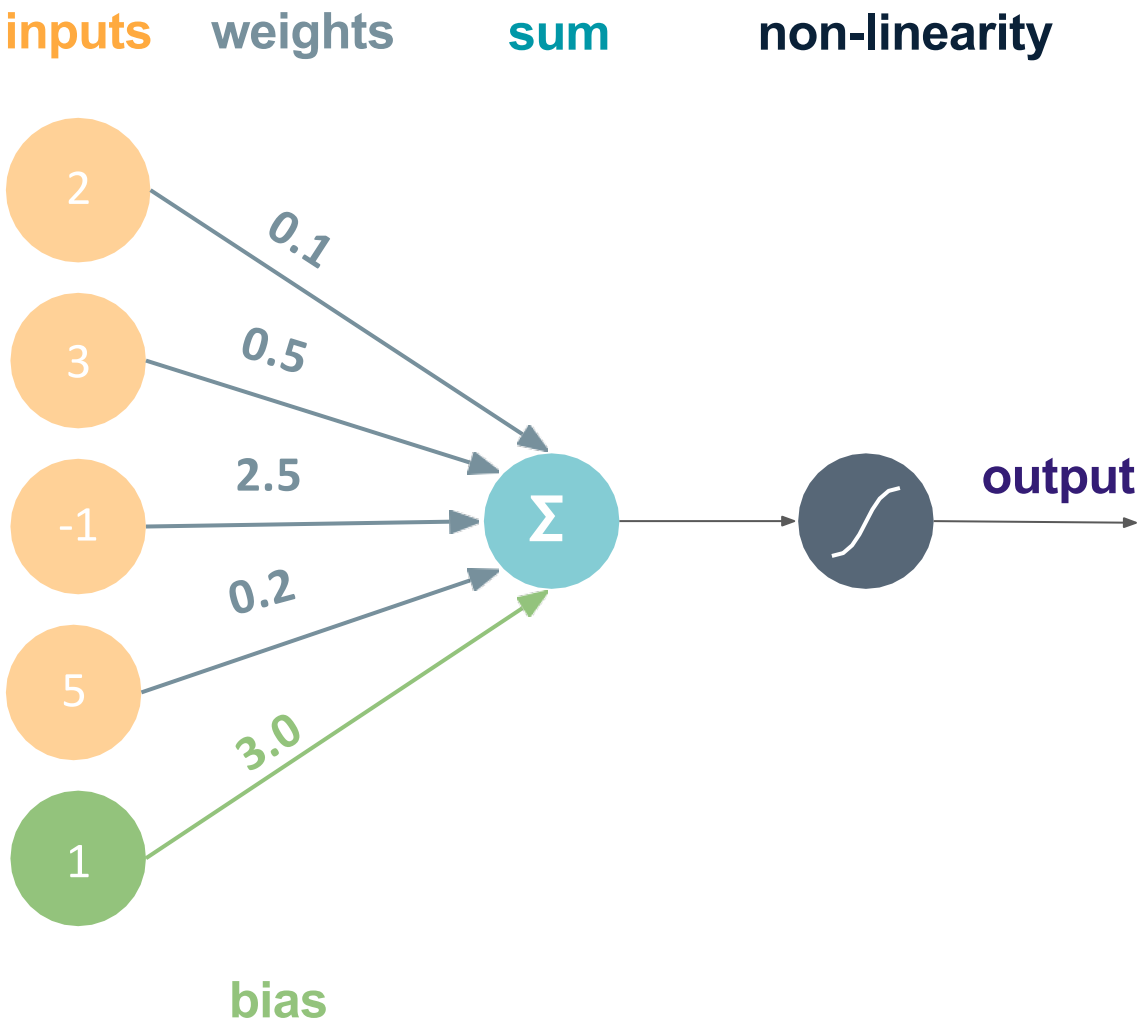
)



Perceptron Forward Pass

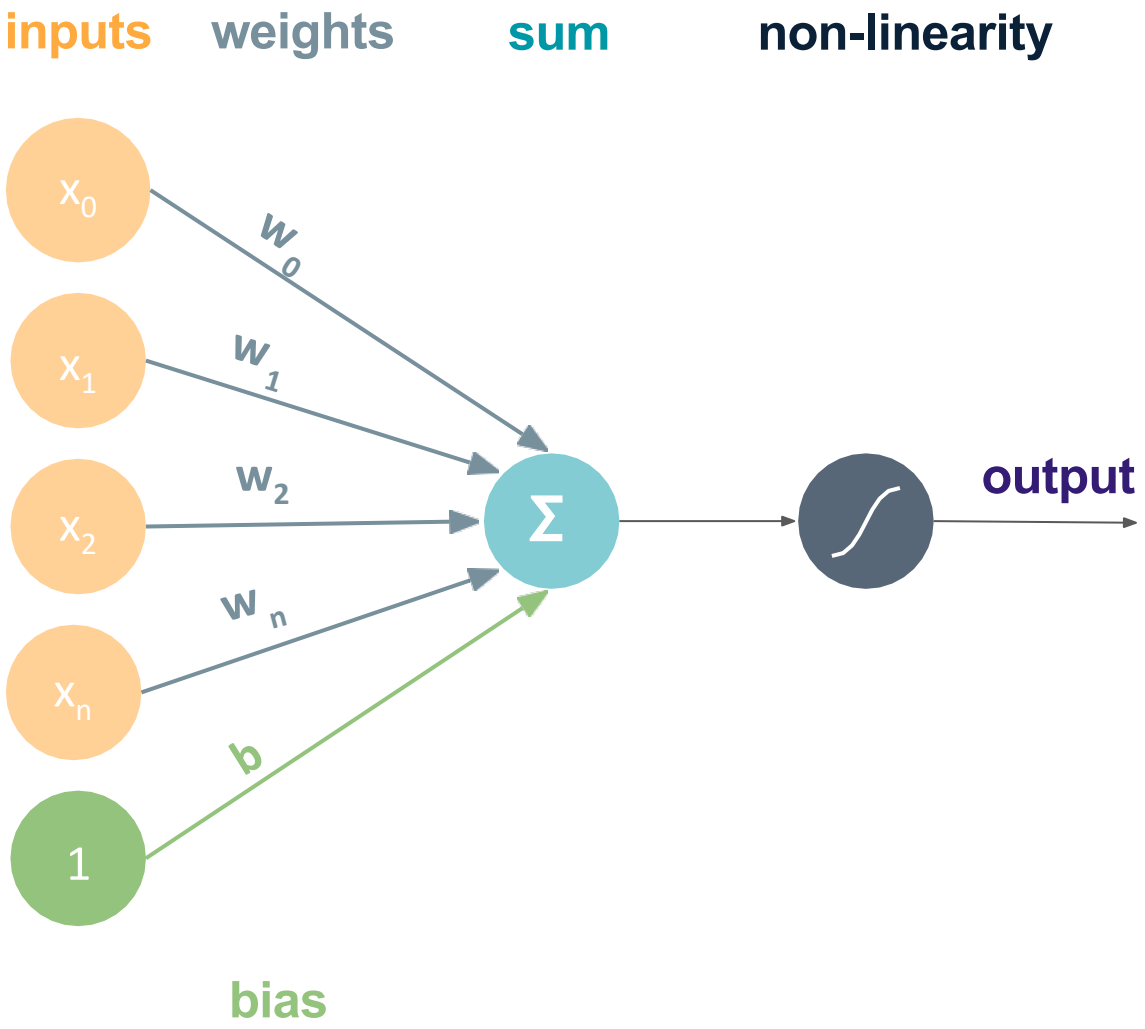
$$output = g(3.2) = \sigma(3.2)$$

$$= \frac{1}{(1 + e^{-3.2})} = 0.96$$

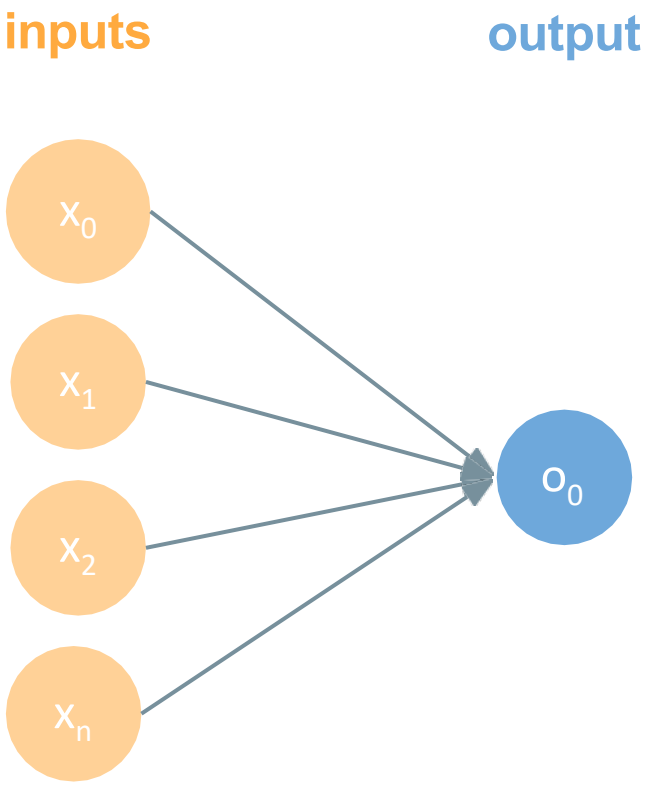


How do we build neural networks with perceptrons?

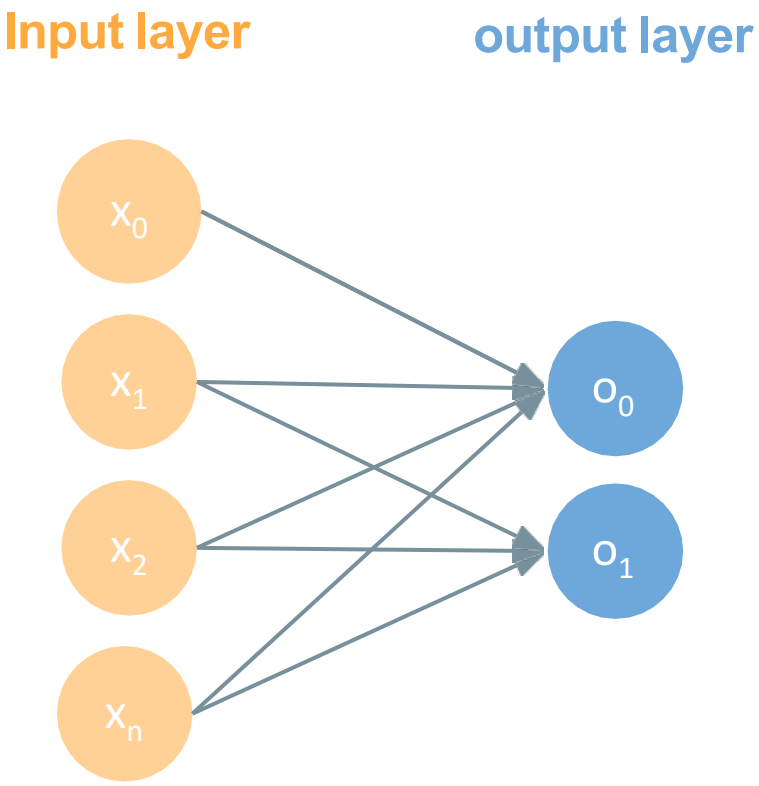
Perceptron Diagram Simplified



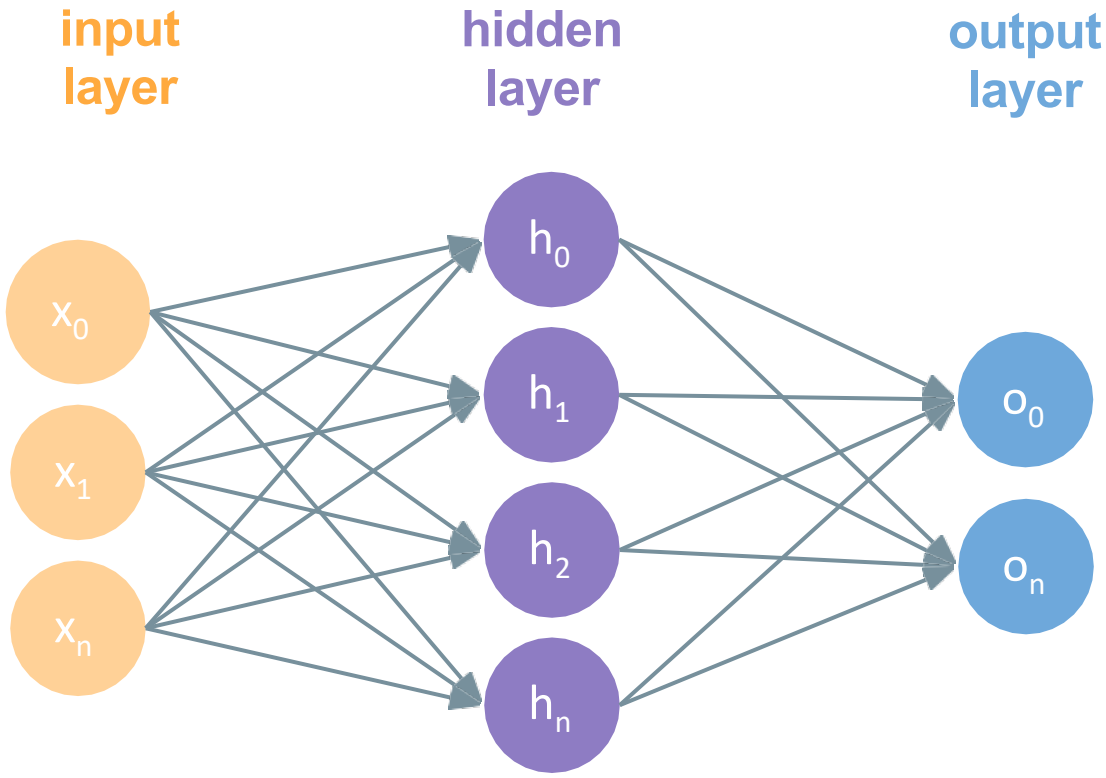
Perceptron Diagram Simplified



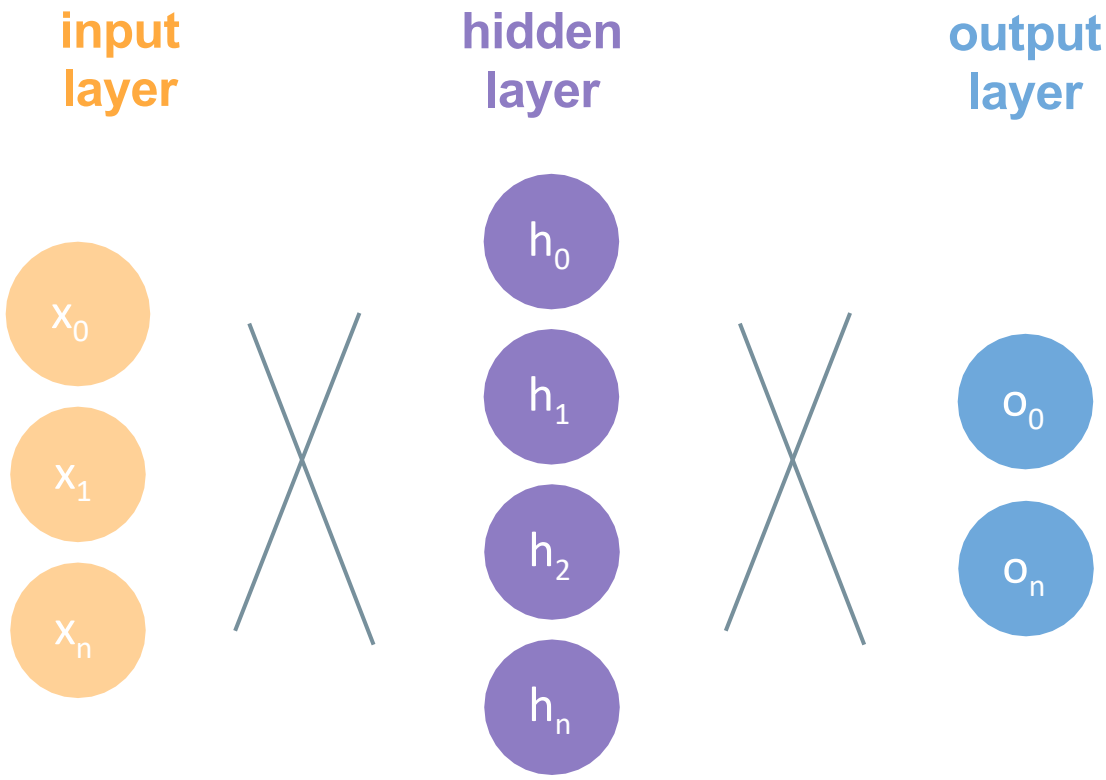
Multi-Output Perceptron



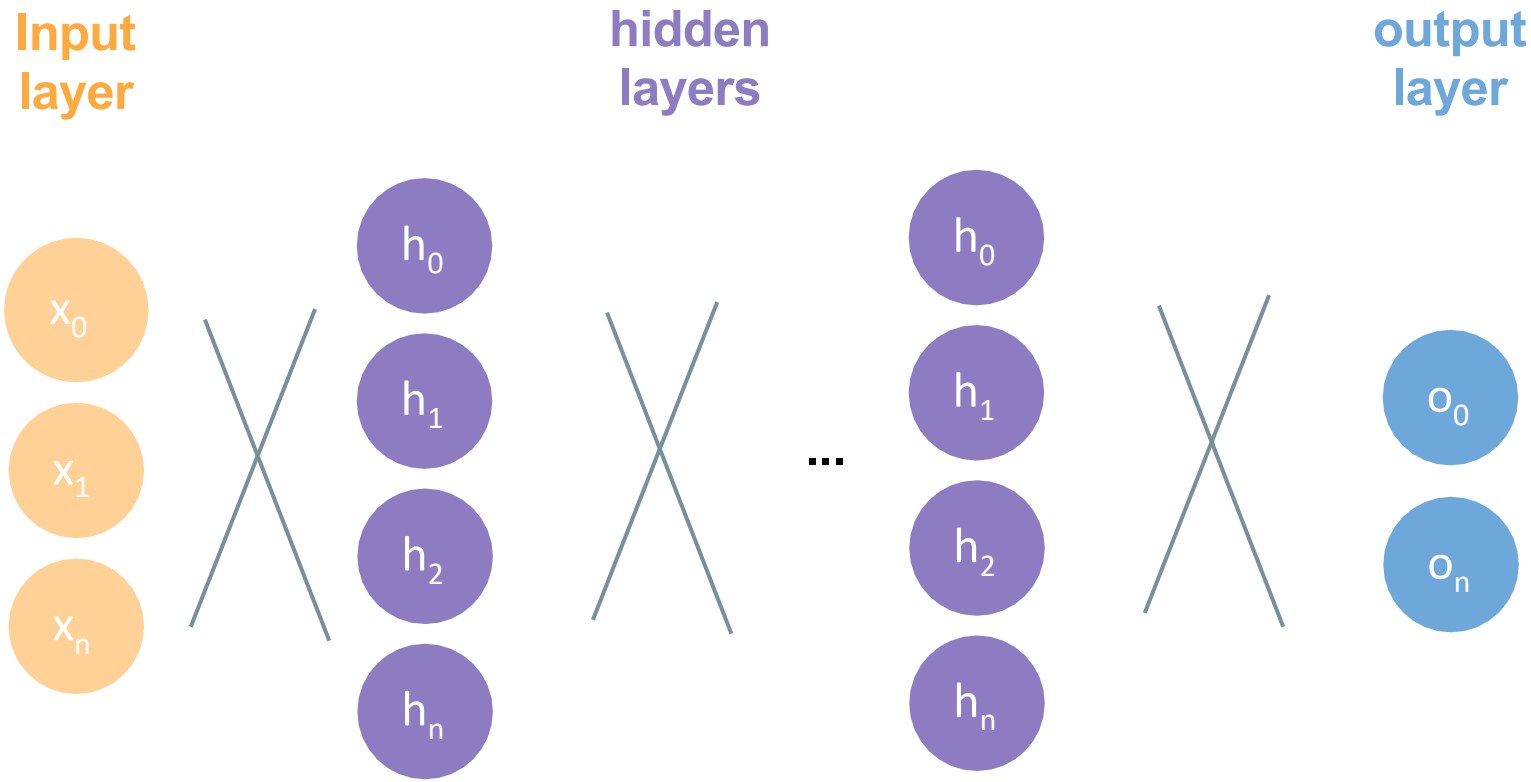
Multi-Layer Perceptron (MLP)



Multi-Layer Perceptron (MLP)



Deep Neural Network



Applying Neural Networks

Example Problem: Will my Flight be Delayed?

Example Problem: Will my Flight be Delayed?

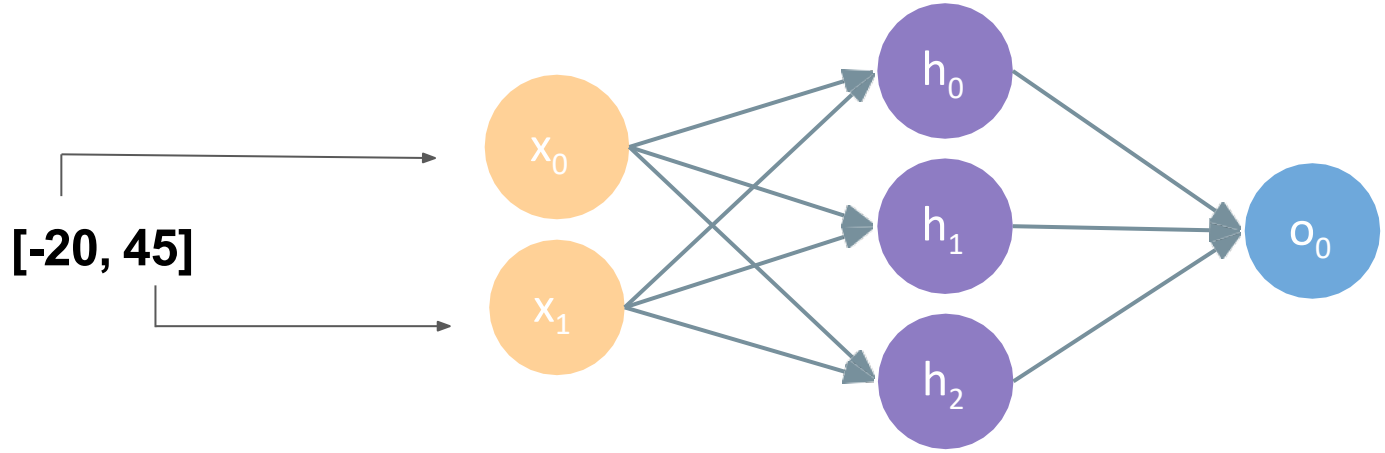
Temperature: -20 F

Wind Speed: 45 mph

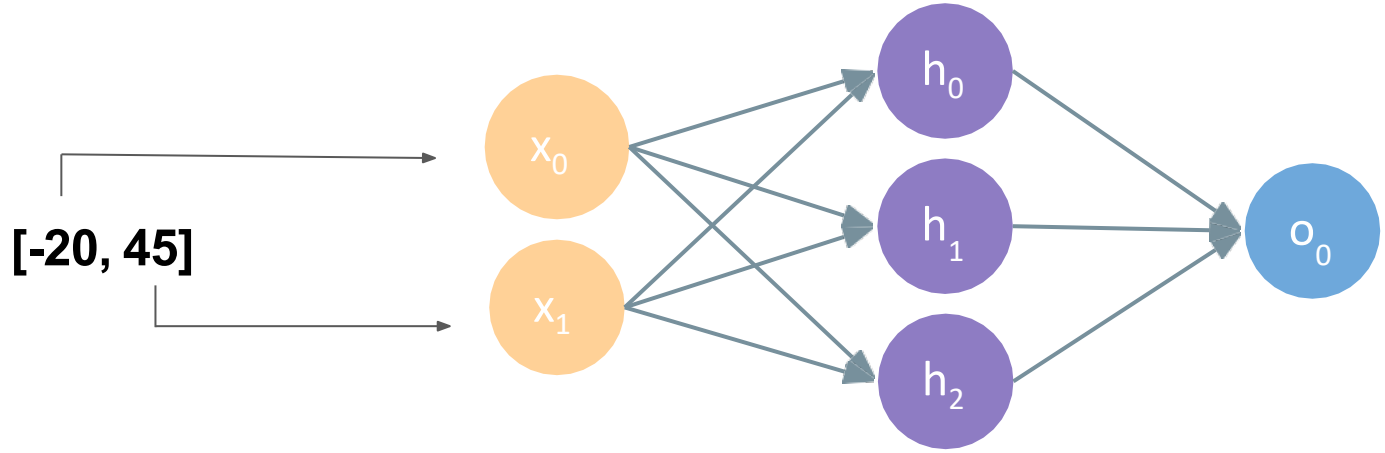
Example Problem: Will my Flight be Delayed?

[-20, 45]

Example Problem: Will my Flight be Delayed?

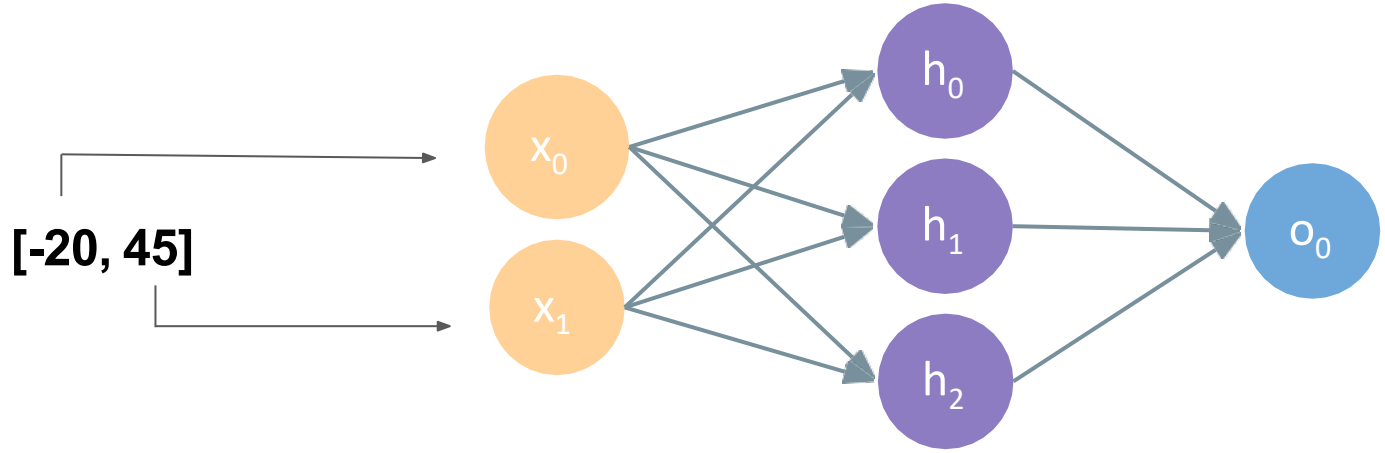


Example Problem: Will my Flight be Delayed?



Predicted: 0.05

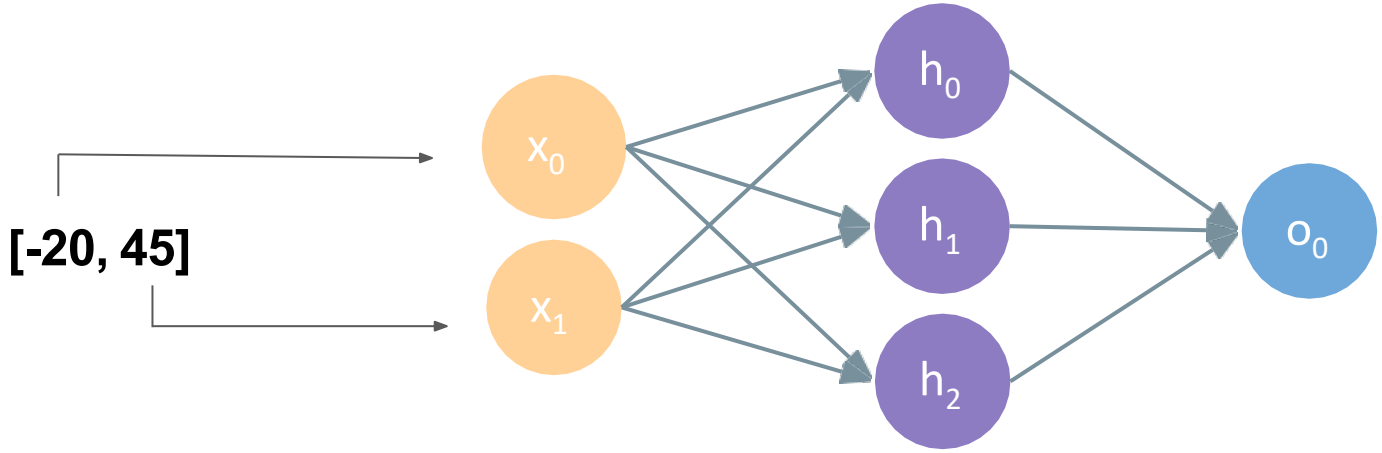
Example Problem: Will my Flight be Delayed?



Predicted: 0.05

Actual: 1

Quantifying Loss



Predicted: 0.05

Actual: 1

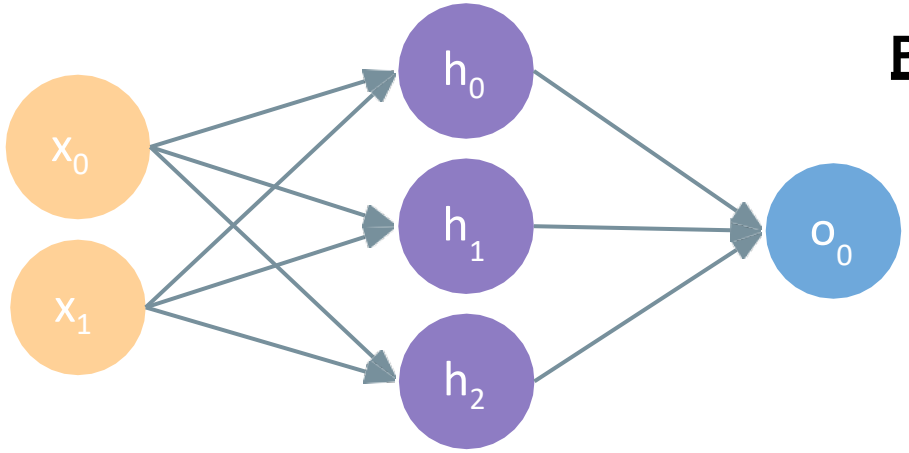
$$loss(f(x^{(i)};\theta), y^{(i)}))$$

Predicted Actual

Total Loss

Input

[
[-20, 45],
[80, 0],
[4, 15],
[45, 60],
]



Predicted

[
0.05
0.02
0.96
0.35
]

Actual

[
1
0
1
1
]

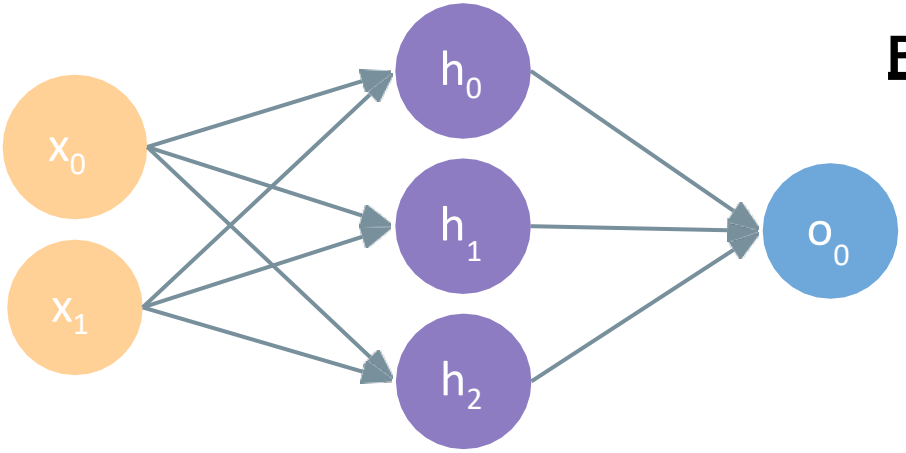
$$\text{total loss} := J(\theta) = \frac{1}{N}\sum_{i}^{N} loss(f(x^{(i)};\theta), y^{(i)}))$$

Predicted Actual

Total Loss

Input

[
[-20, 45],
[80, 0],
[4, 15],
[45, 60],
]



Predicted	**Actual**
[	[
0.05	1
0.02	0
0.96	1
0.35	1
]	]

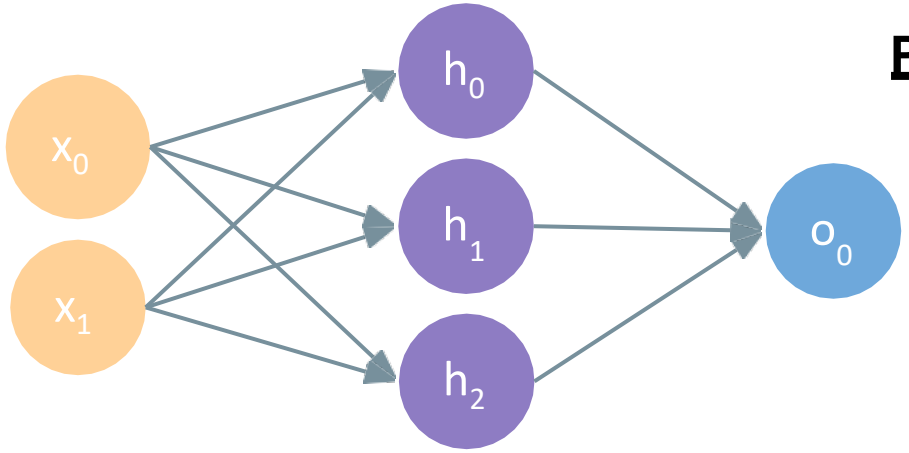
$$\text{total loss} := J(\theta) = \frac{1}{N}\sum_{i}^{N} loss(f(x^{(i)};\theta), y^{(i)}))$$

Predicted Actual

Binary Cross Entropy Loss

Input

[
[-20, 45],
[80, 0],
[4, 15],
[45, 60],
]



Predicted	**Actual**
[	[
0.05	1
0.02	0
0.96	1
0.35	1
]	]

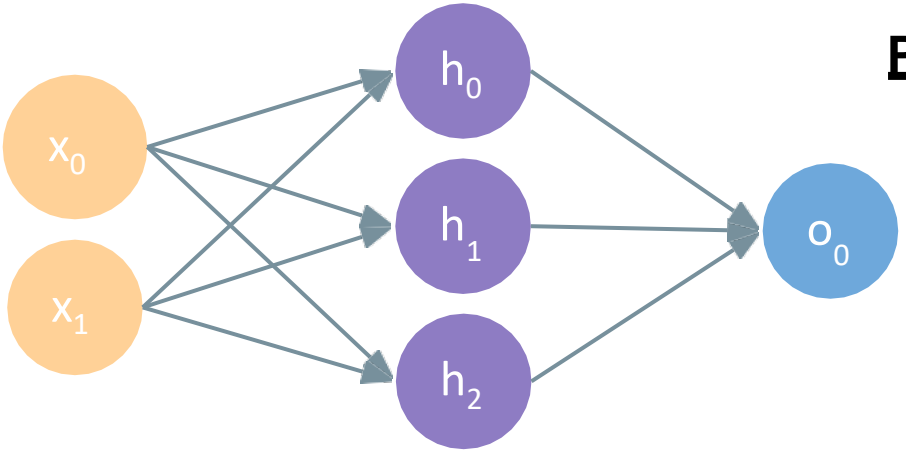
$$\text{cross\_entropy}(\theta) = \frac{1}{N}\sum_{i}^{N} y^{(i)} log(f(x^{(i)};\theta)) + (1 - y^{(i)}) log(1 - f(x^{(i)};\theta)))$$

Actual Predicted Actual Predicted

Mean Squared Error (MSE) Loss

Input

[
[-20, 45],
[80, 0],
[4, 15],
[45, 60],
]



Predicted	**Actual**
[	[
10	40
45	42
100	110
15	55
]	]

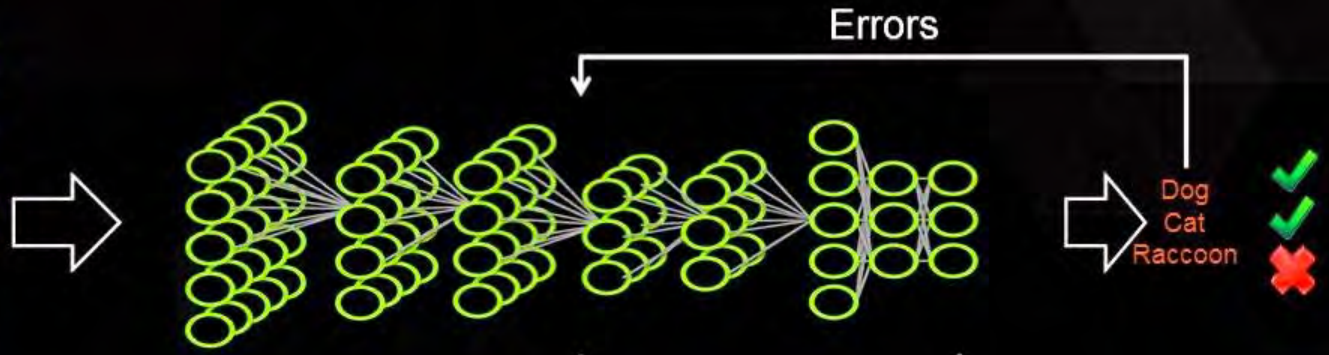
$$\mathrm{MSE}(\theta) = \frac{1}{N}\sum_{i}^{N}(f(x^{(i)};\theta) - y^{(i)})^2$$

Predicted Actual

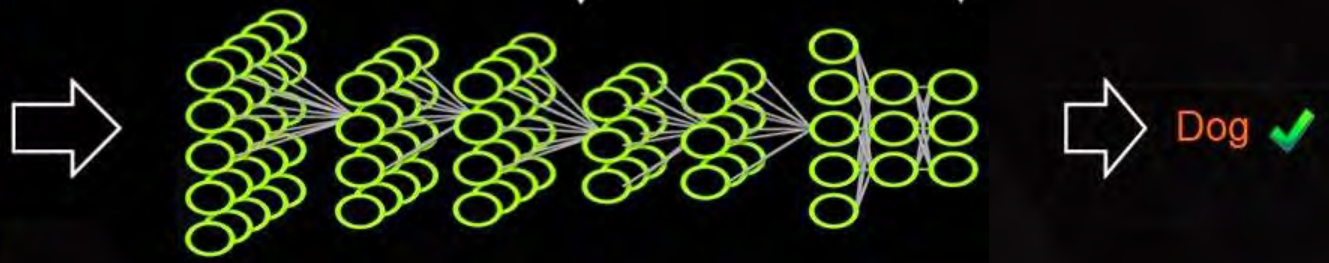
Evaluating Machine Learning Approaches

DEEP LEARNING APPROACH

Train:



Deploy:



Evaluation

- Accuracy

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

- Error Rate

$$\text{Error rate} = \frac{\text{Number of wrong predictions}}{\text{Total number of predictions}}$$

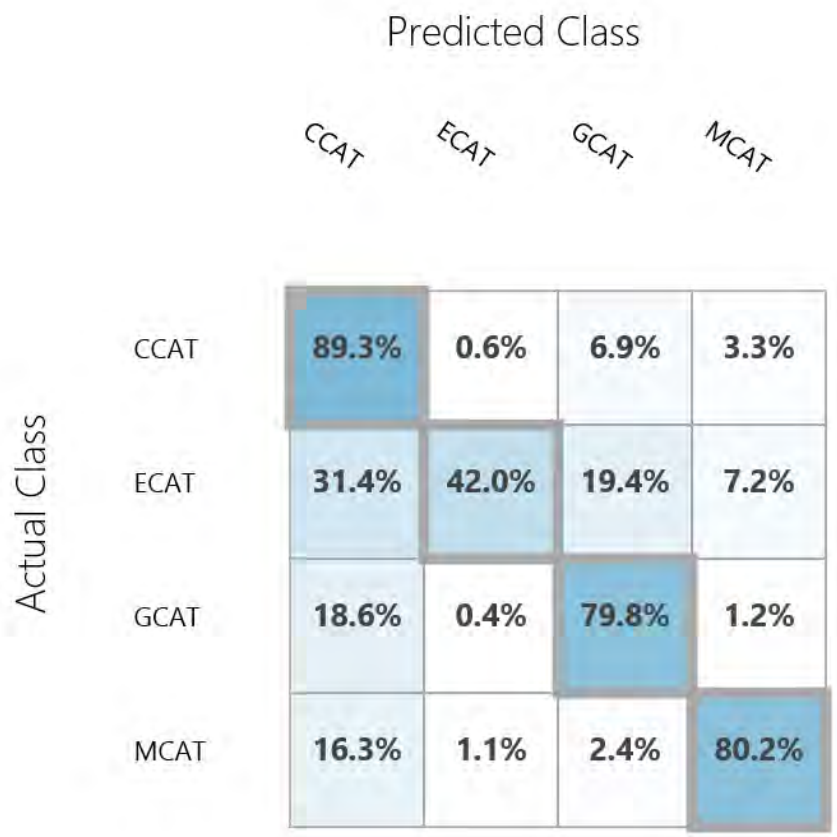
Evaluation

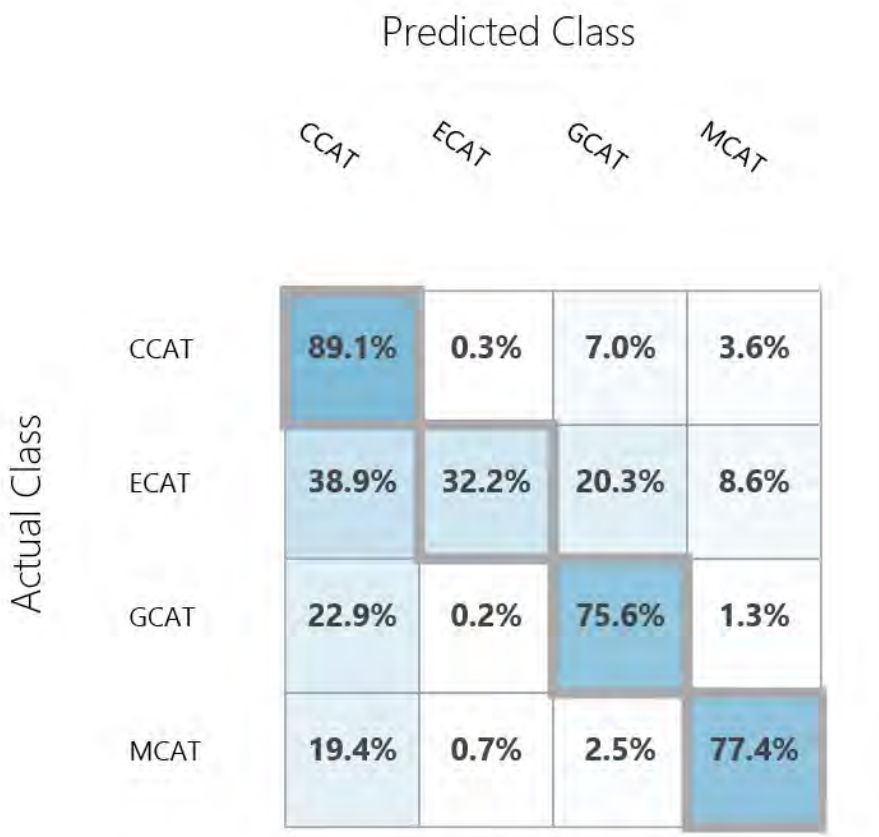
- Confusion Matrix

n=192	**Predicted: 0**	**Predicted: 1**
Actual: 0	118	12
Actual: 1	47	15

Evaluation

- Confusion Matrix



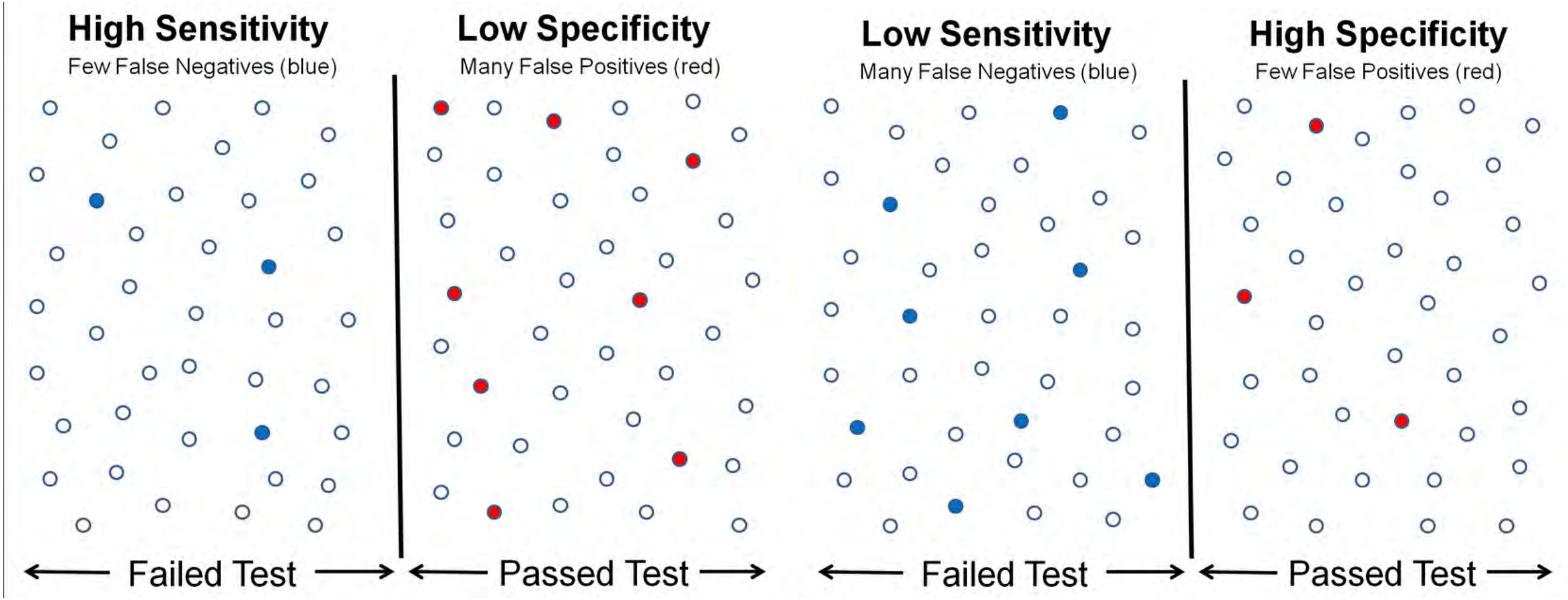


Evaluation

- Confusion Matrix
- Classifier P(dog) = 1 P(cat) = 0 “accuracy” = 90%

N = 100	Dog	Cat
Dog	90	10
Cat	0	0

Sensitivity vs. Specificity



Receiver Operating Characteristic (ROC) Curve

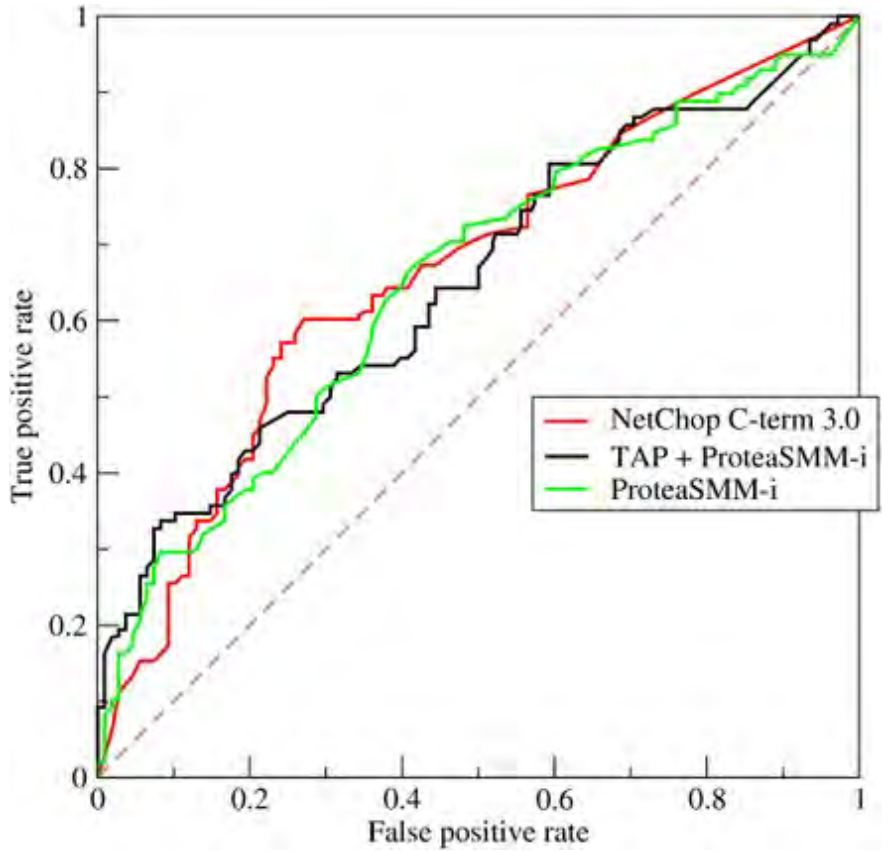
		Disease	
		+	-
Test	+	True Positive (TP)	False Positive (FP)
	-	False Negative (FN)	True Negative (TN)
		All with disease= TP + FN	All without disease= FP + TN

Receiver Operating Characteristic (ROC) Curve

- (number of) positive samples (P)
- (number of) negative samples (N)
- (number of) true positive (TP)
- (number of) true negative (TN)

$$TPR = TP/P = TP/(TP + FN)$$

$$FPR = FP/N = FP/(FP + TN)$$



Training Set / Test Set

20 data points

i1,i2,i3,i4
i5,i6,i7,i8
i9,i10,i11,i12,
i13,i14,i15,i16
i17,i18,i19,i20

Training Set / Test Set

20 data points

i1,i2,i3,i4
i5,i6,i7,i8
i9,i10,i11,i12,
i13,i14,i15,i16
i17,i18,i19,i20

Training Set
Test Set

k-fold cross validation

20 data points

i1,i2,i3,i4
i5,i6,i7,i8
i9,i10,i11,i12,
i13,i14,i15,i16
i17,i18,i19,i20

Training Set
Test Set

Train and Evaluate

k-fold cross validation

20 data points

i1,i2,i3,i4
i5,i6,i7,i8
i9,i10,i11,i12,
i13,i14,i15,i16
i17,i18,i19,i20

Training Set
Test Set

Train and Evaluate

k-fold cross validation

20 data points

i1,i2,i3,i4
i5,i6,i7,i8
i9,i10,i11,i12,
i13,i14,i15,i16
i17,i18,i19,i20

Training Set
Test Set

k-fold cross validation

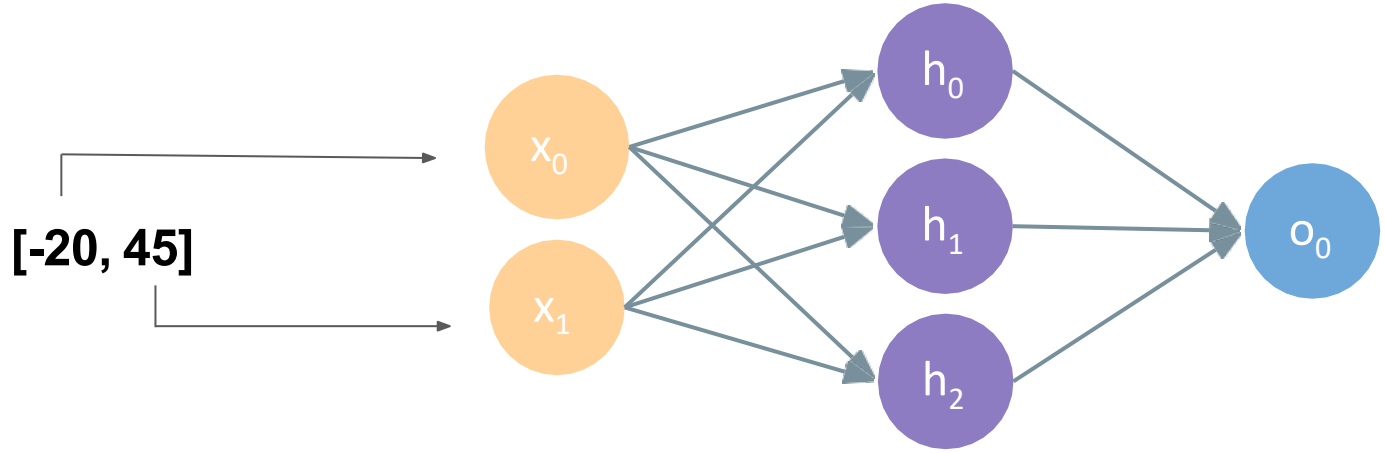
- Solution: k-fold crossvalidation maximizes the use of the data
- Divide data randomly into k folds (subsets) of equal size.
- Train the model on k−1 folds, use one fold for testing.
- Repeat this process k times so that all folds are used for testing.
- Compute the average performance on the k test sets

Training Data

Testing Data

Generalization vs. Overfitting

Example Problem: Will my Flight be Delayed?



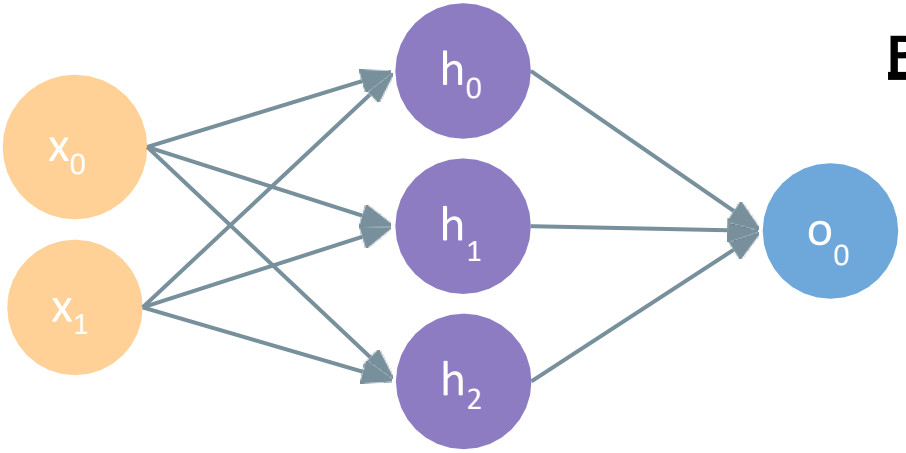
Predicted: 0.05

Actual: 1

Total Loss

Input

[
[-20, 45],
[80, 0],
[4, 15],
[45, 60],
]



Predicted	**Actual**
[	[
0.05	1
0.02	0
0.96	1
0.35	1
]	]

$$\text{total loss} := J(\theta) = \frac{1}{N}\sum_{i}^{N} loss(f(x^{(i)};\theta), y^{(i)}))$$

Predicted Actual

Training Neural Networks

Training Neural Networks: Objective

$$arg_\theta \min \frac{1}{N} \sum_{i}^{N} \text{loss}(f(x^{(i)};\theta), y^{(i)})$$

(Stolen from N. Locascio)

Training Neural Networks: Objective

$$arg_\theta \min \underbrace{\frac{1}{N}\sum_i^N \text{loss}(f(x^{(i)};\theta), y^{(i)})}_{J(\theta)}$$

$J(\theta)$

loss function

(Stolen from N. Locascio)

Training Neural Networks: Objective

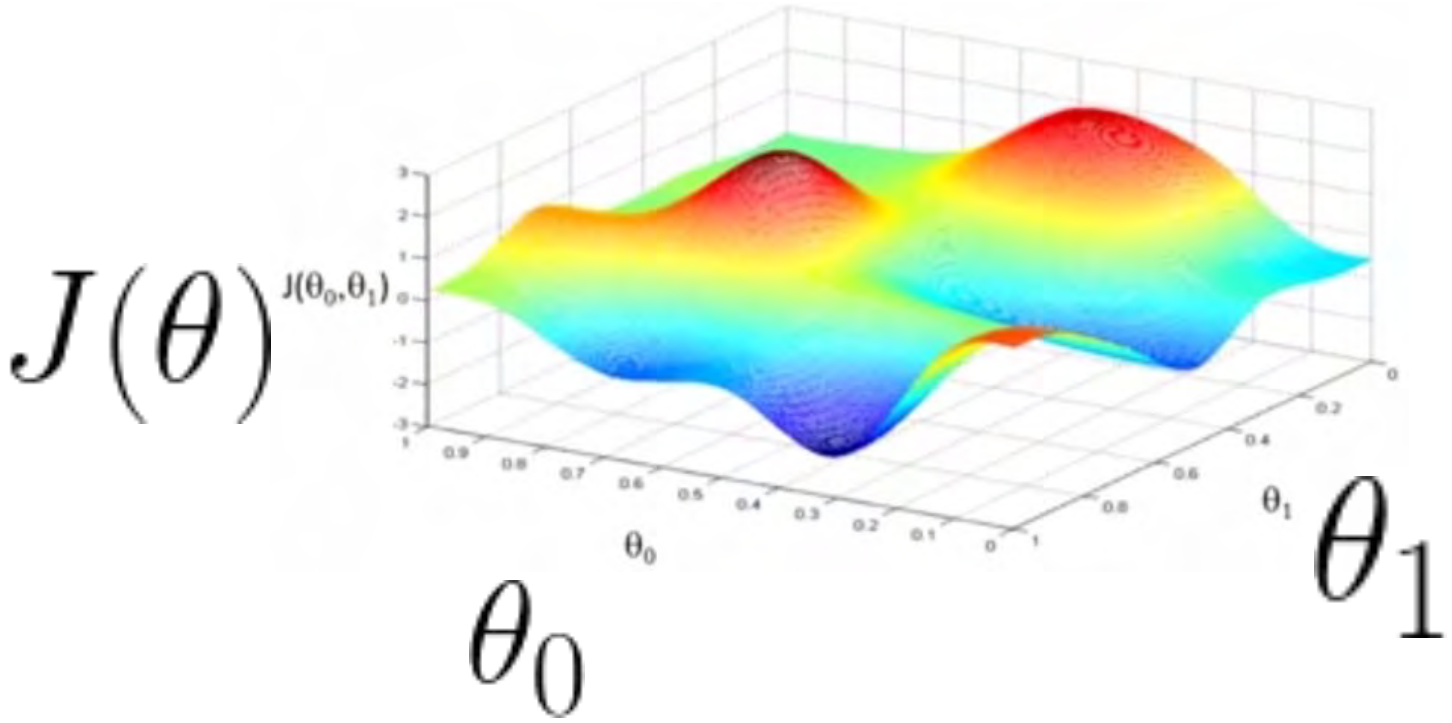
$$arg_\theta \min \underbrace{\frac{1}{N}\sum_{i}^{N} \text{loss}(f(x^{(i)};\theta), y^{(i)})}$$

$$J(\theta)$$

$$\theta = W_1, W_2 ... W_n$$

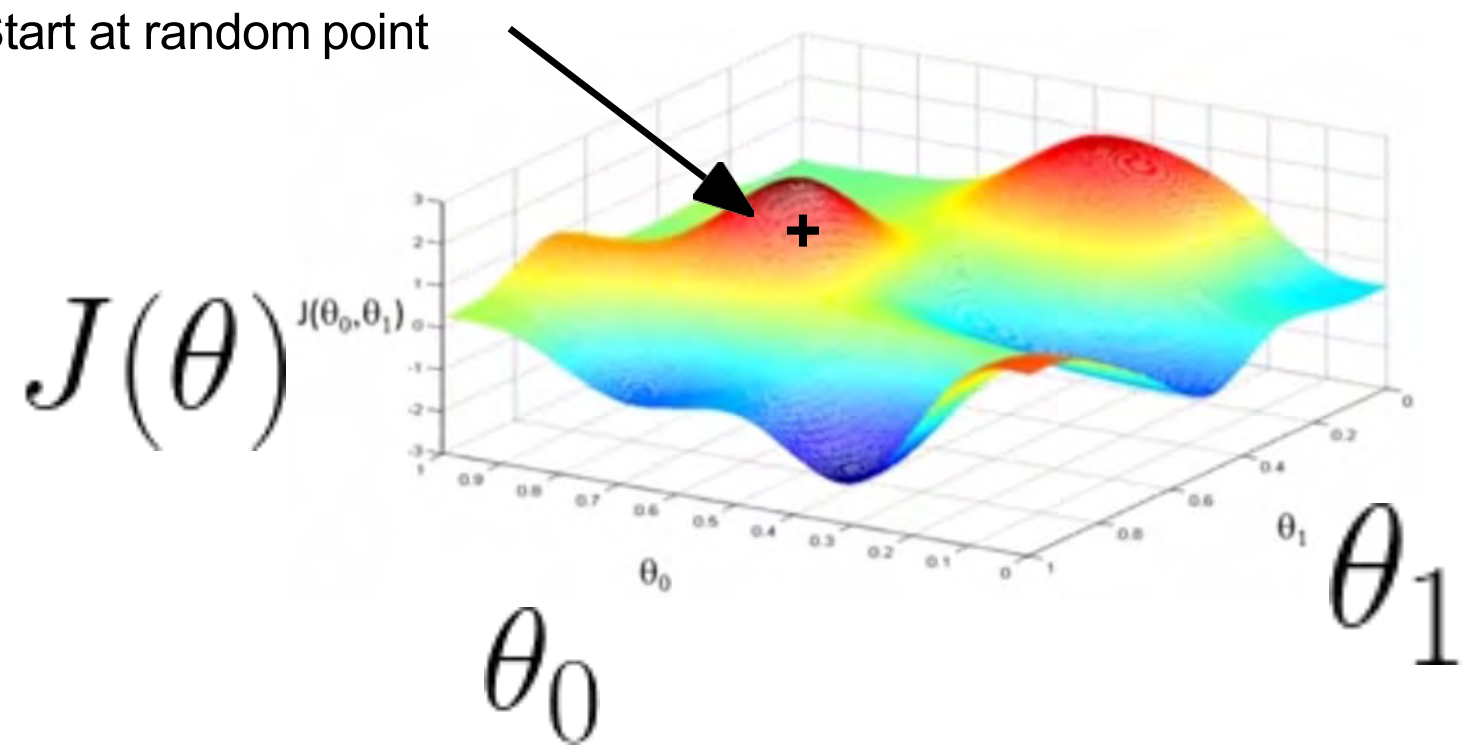
(Stolen from N. Locascio)

Loss is a function of the model's parameters



(Stolen from N. Locascio)

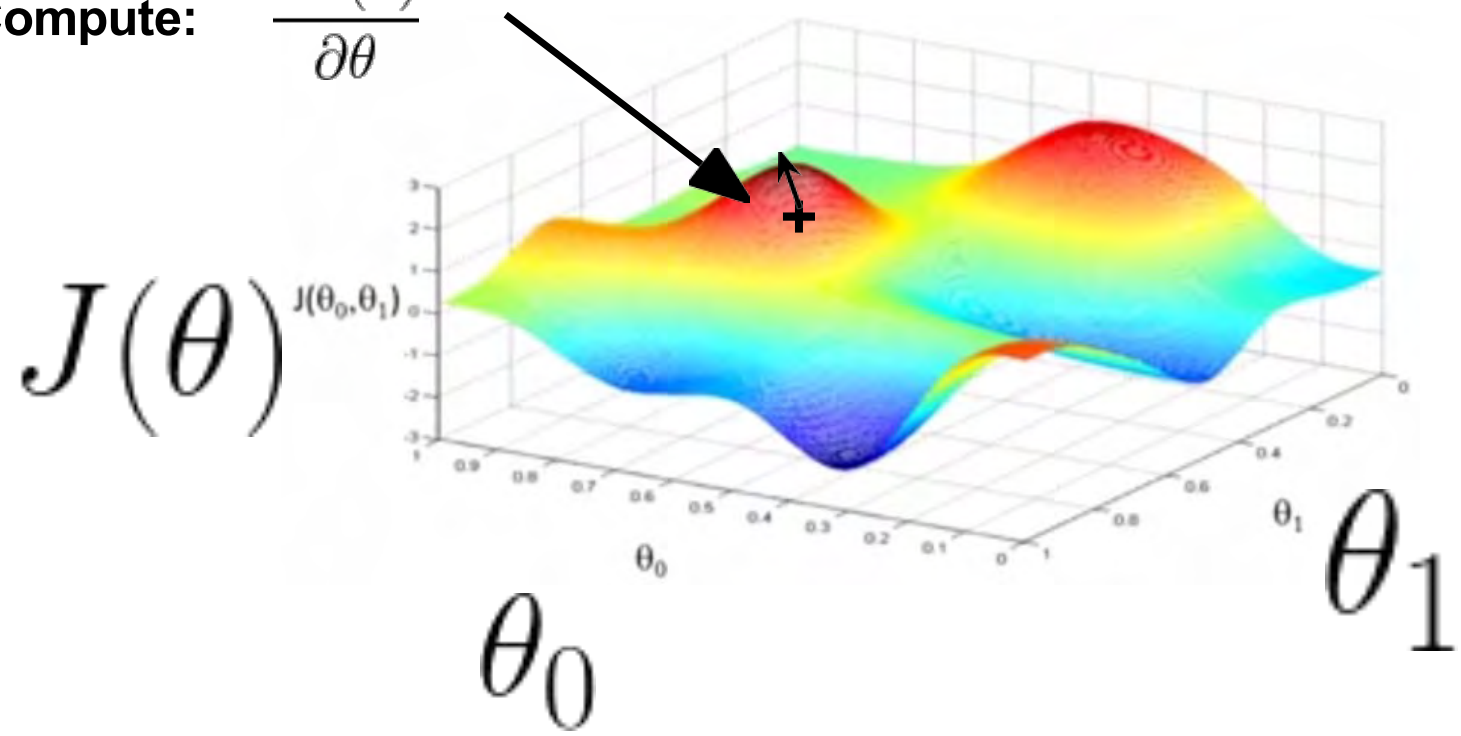
How to minimize loss?



(Stolen from N. Locascio)

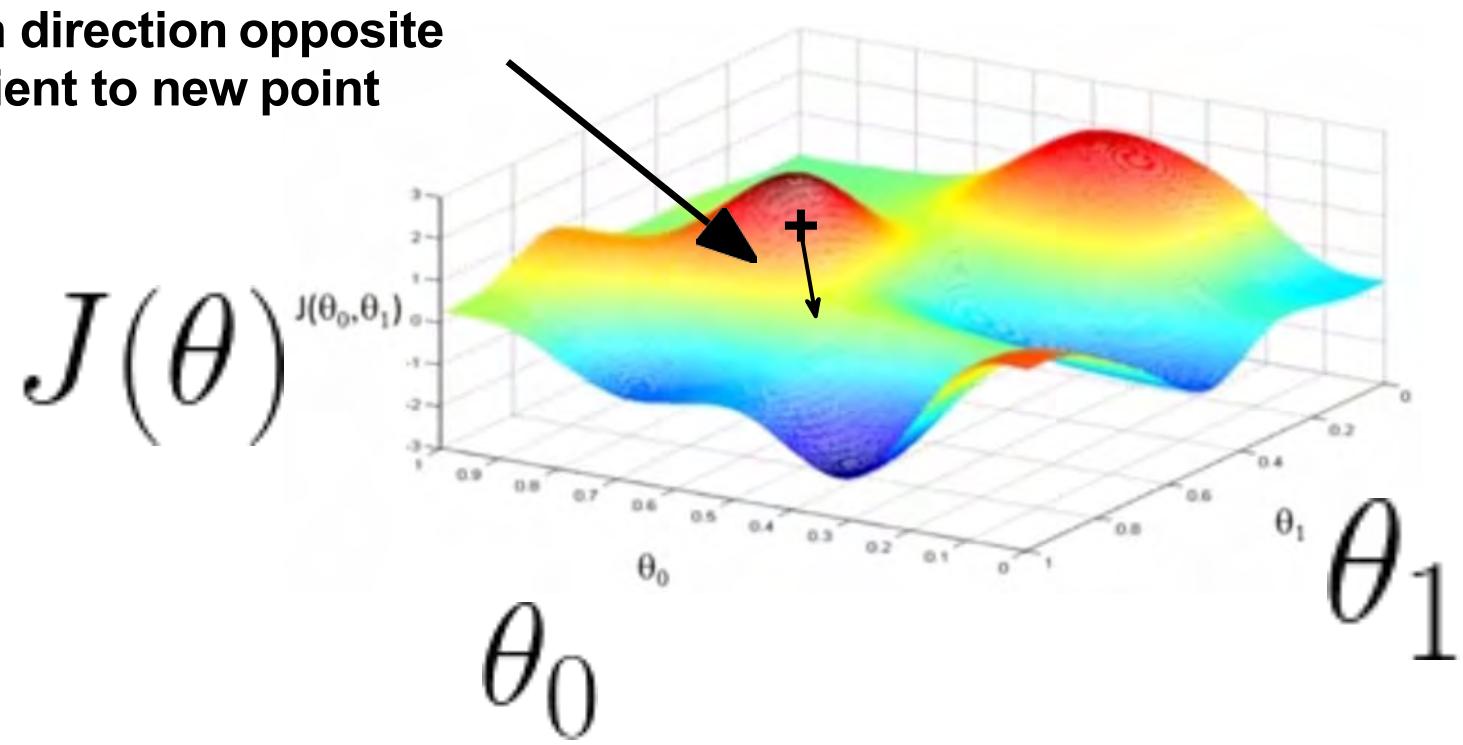
How to minimize loss?

Compute: $\frac{\partial J(\theta)}{\partial \theta}$



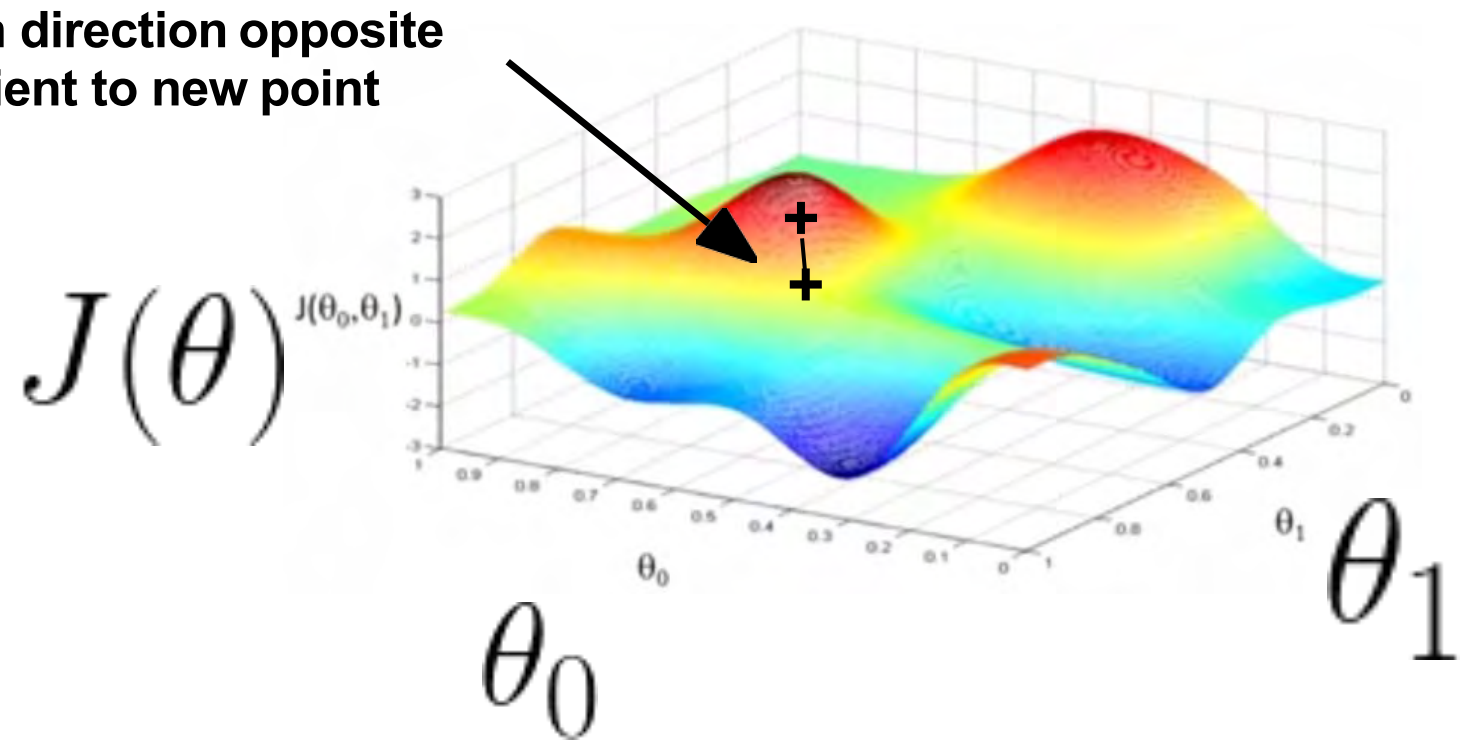
(Stolen from N. Locascio)

How to minimize loss?



(Stolen from N. Locascio)

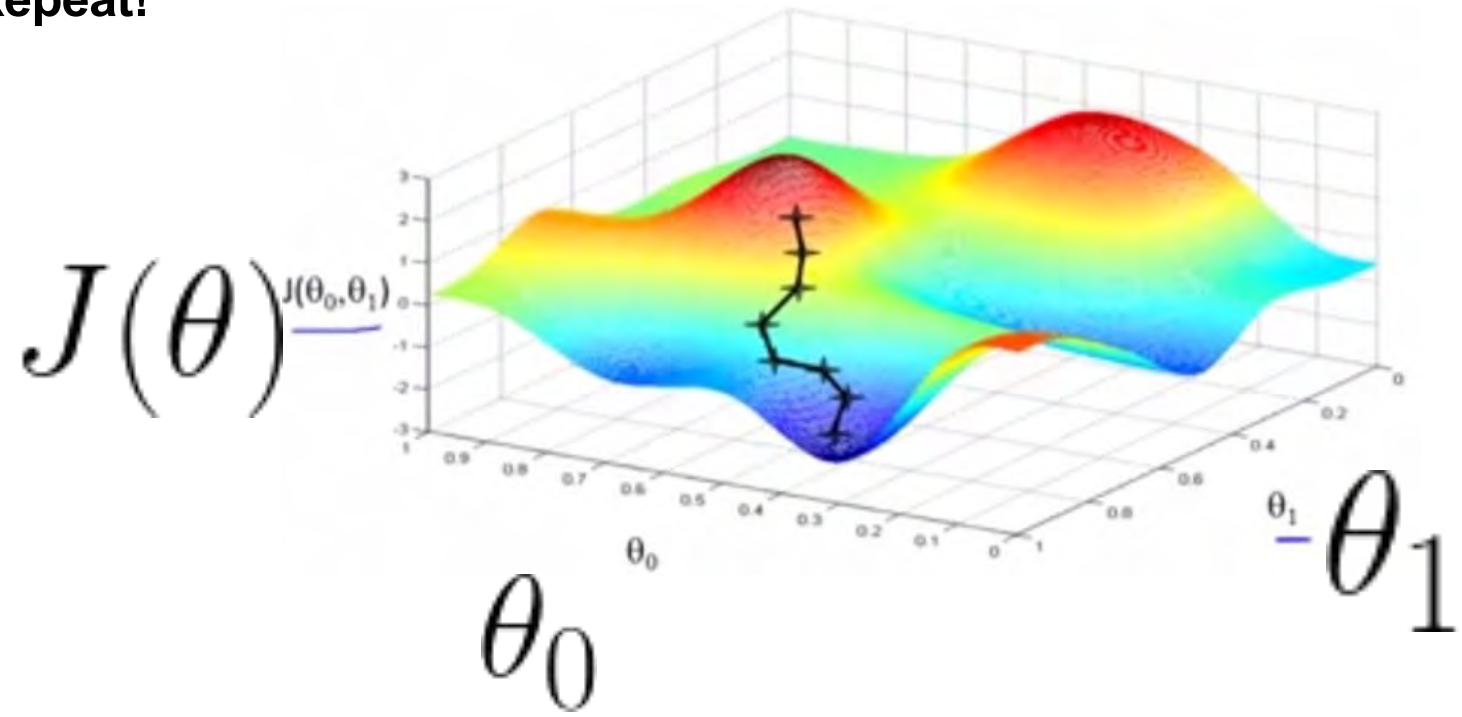
How to minimize loss?



(Stolen from N. Locascio)

How to minimize loss?

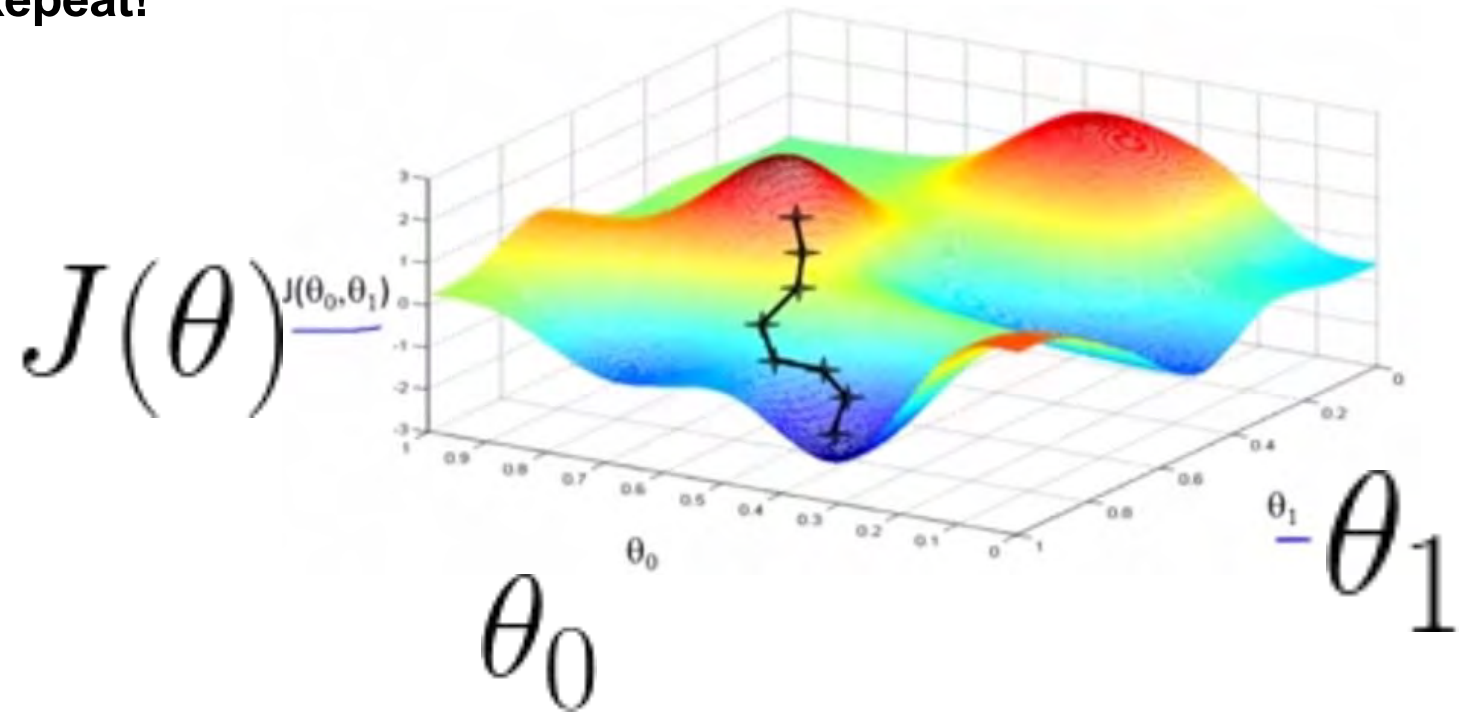
Repeat!



(Stolen from N. Locascio)

This is called Stochastic Gradient Descent (SGD)

Repeat!

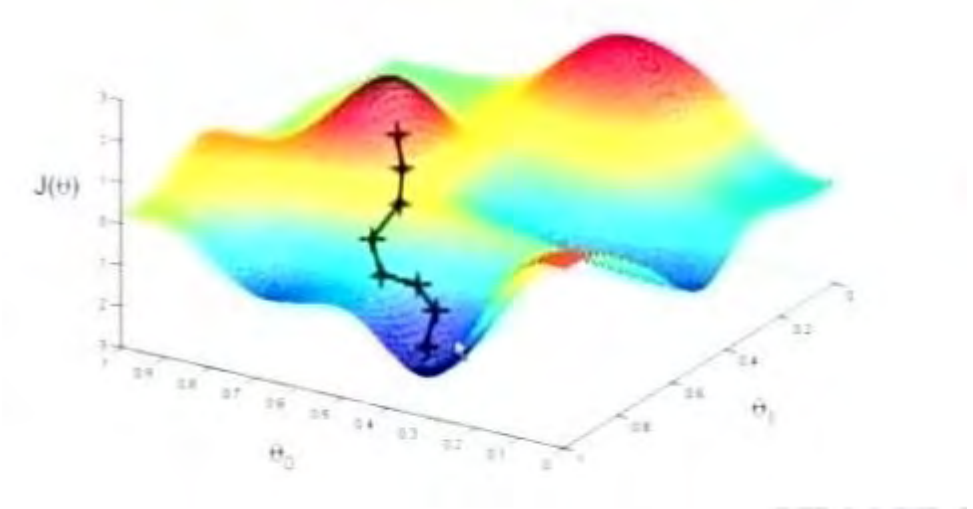


(Stolen from N. Locascio)

Stochastic Gradient Descent (SGD)

- Initialize θ randomly
- For N Epochs
  - For each training example (x, y):
    - Compute Loss Gradient: $\frac{\partial J(\theta)}{\partial \theta}$
    - Update θ with update rule:

$$\theta := \theta - \eta \frac{\partial J(\theta)}{\partial \theta}$$

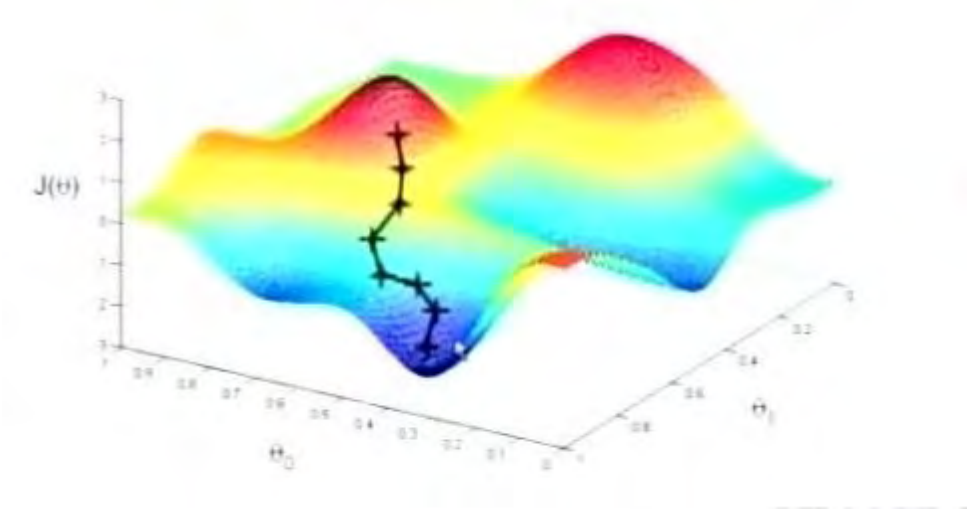


(Stolen from N. Locascio)

Stochastic Gradient Descent (SGD)

- Initialize θ randomly
- For N Epochs
  - For each training example (x, y):
    - Compute Loss Gradient: $\frac{\partial J(\theta)}{\partial \theta}$
    - Update θ with update rule:

$$\theta := \theta - \eta \frac{\partial J(\theta)}{\partial \theta}$$

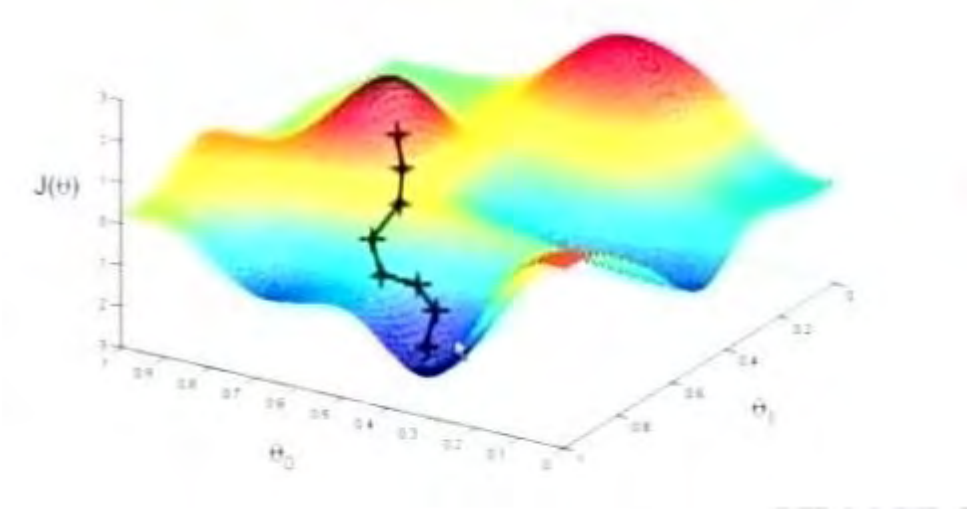


(Stolen from N. Locascio)

Stochastic Gradient Descent (SGD)

- Initialize θ randomly
- For N Epochs
  - For each training example (x, y):
    - Compute Loss Gradient: $\frac{\partial J(\theta)}{\partial \theta}$
    - Update θ with update rule:

$$\theta := \theta - \eta \frac{\partial J(\theta)}{\partial \theta}$$



- How to Compute Gradient?

(Stolen from N. Locascio)

Calculating the Gradient: Backpropagation

(Stolen from N. Locascio)

Calculating the Gradient: Backpropagation

$$\frac{\partial J(\theta)}{\partial W_2} = \frac{\partial J(\theta)}{\partial o_0} * \frac{\partial J(o_0)}{\partial W_2}$$

(Stolen from N. Locascio)

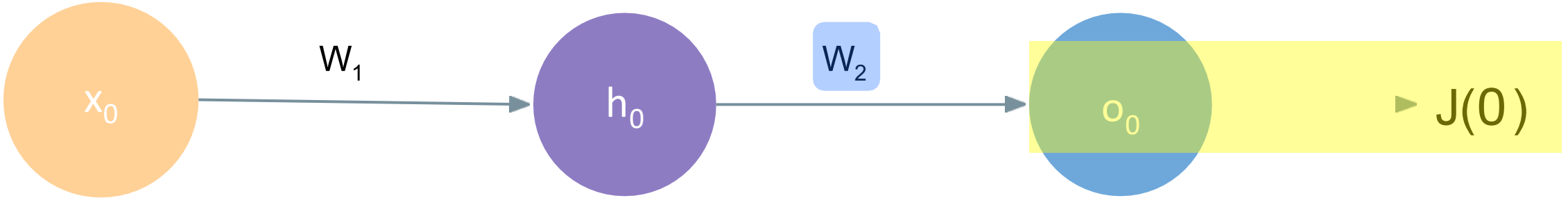
Calculating the Gradient: Backpropagation

Apply the chain rule

$$\frac{\partial J(\theta)}{\partial W_2} = \frac{\partial J(\theta)}{\partial o_0} * \frac{\partial J(o_0)}{\partial W_2}$$

(Stolen from N. Locascio)

Calculating the Gradient: Backpropagation

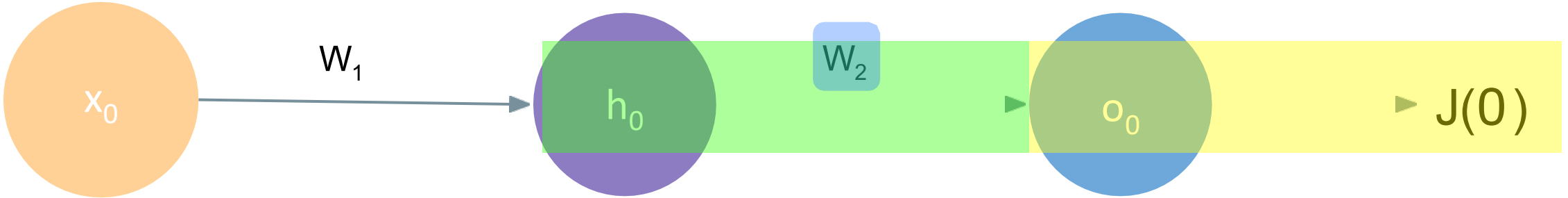


Apply the chain rule

$$\frac{\partial J(\theta)}{\partial W_2} = \frac{\partial J(\theta)}{\partial o_0} * \frac{\partial J(o_0)}{\partial W_2}$$

(Stolen from N. Locascio)

Calculating the Gradient: Backpropagation



Apply the chain rule

$$\frac{\partial J(\theta)}{\partial W_2} = \frac{\partial J(\theta)}{\partial o_0} * \frac{\partial o_0}{\partial W_2}$$

(Stolen from N. Locascio)

Calculating the Gradient: Backpropagation

$$\frac{\partial J(\theta)}{\partial W_1} =$$

(Stolen from N. Locascio)

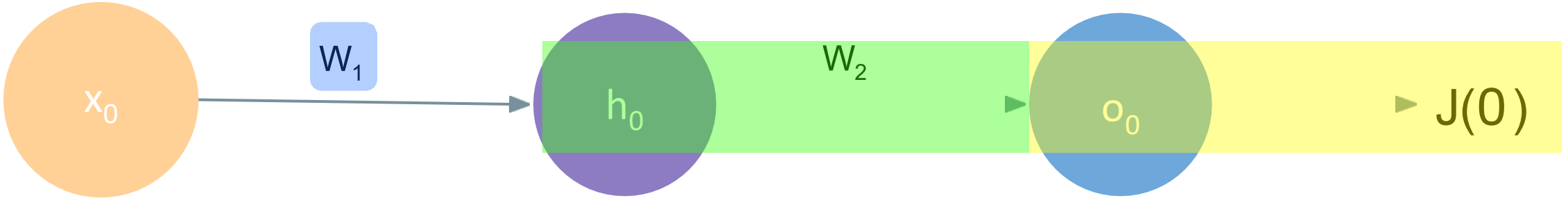
Calculating the Gradient: Backpropagation

Apply the chain rule

$$\frac{\partial J(\theta)}{\partial W_1} =$$

(Stolen from N. Locascio)

Calculating the Gradient: Backpropagation

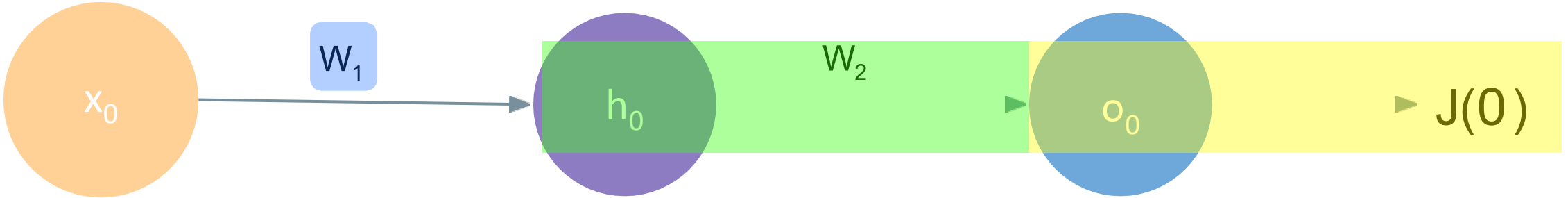


Apply the chain rule

$$\frac{\partial J(\theta)}{\partial W_1} = \frac{\partial J(\theta)}{\partial o_0} * \frac{\partial o_0}{\partial h_0} * \frac{\partial h_0}{\partial W_1}$$

(Stolen from N. Locascio)

Calculating the Gradient: Backpropagation



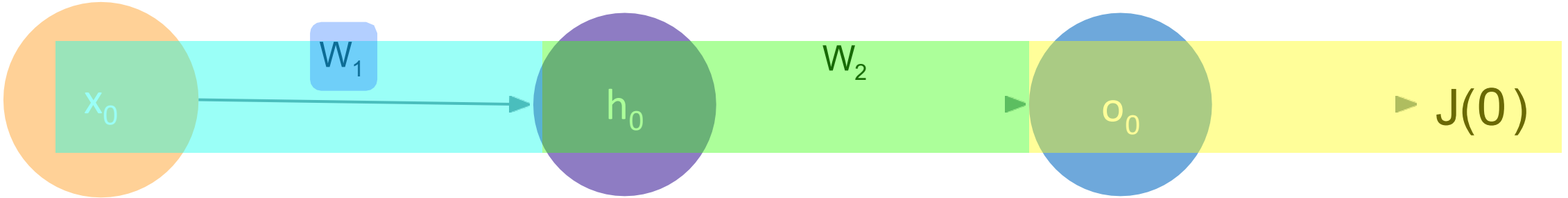
Apply the chain rule

Apply the chain rule

$$\frac{\partial J(\theta)}{\partial W_1} = \frac{\partial J(\theta)}{\partial o_0} * \frac{\partial o_0}{\partial h_0} * \frac{\partial h_0}{\partial W_1}$$

(Stolen from N. Locascio)

Calculating the Gradient: Backpropagation



Apply the chain rule

Apply the chain rule

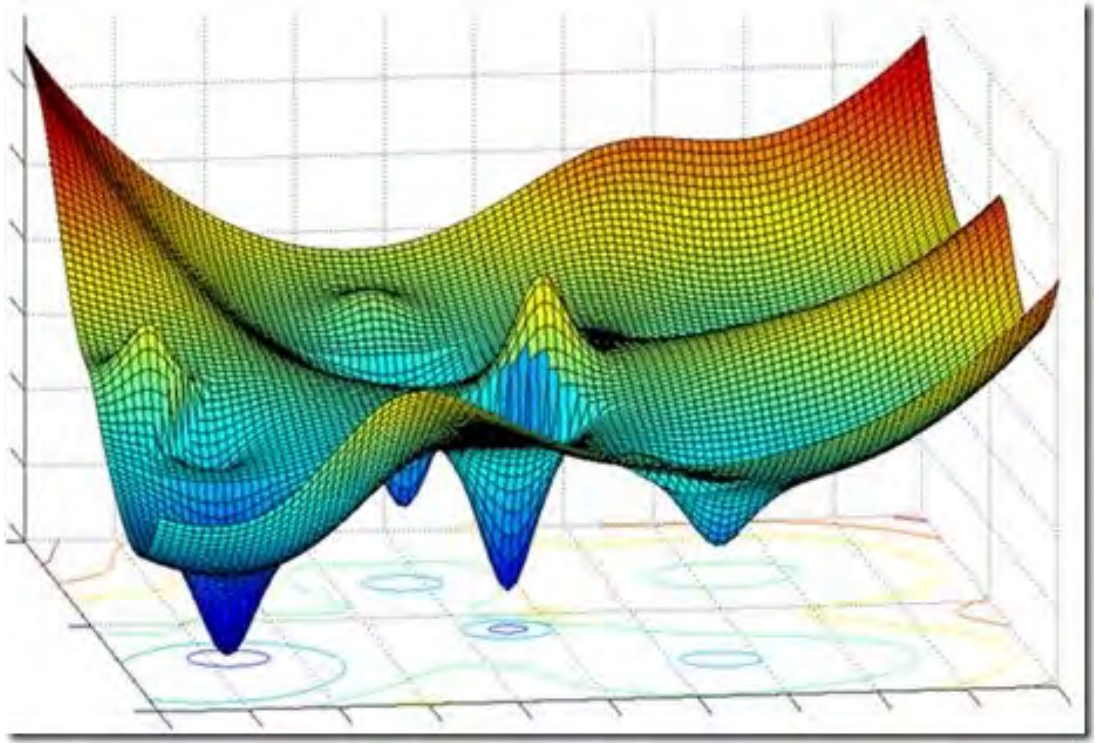
$$\frac{\partial J(\theta)}{\partial W_1} = \frac{\partial J(\theta)}{\partial o_0} * \frac{\partial o_0}{\partial h_0} * \frac{\partial h_0}{\partial W_1}$$

(Stolen from N. Locascio)

Training Neural Networks In Practice

(Stolen from N. Locascio)

Loss function can be difficult to optimize



(Stolen from N. Locascio)

Loss function can be difficult to optimize

Update Rule: $$\theta := \theta - \eta \frac{\partial J(\theta)}{\partial \theta}$$

(Stolen from N. Locascio)

Loss function can be difficult to optimize

How to Choose Learning Rate?

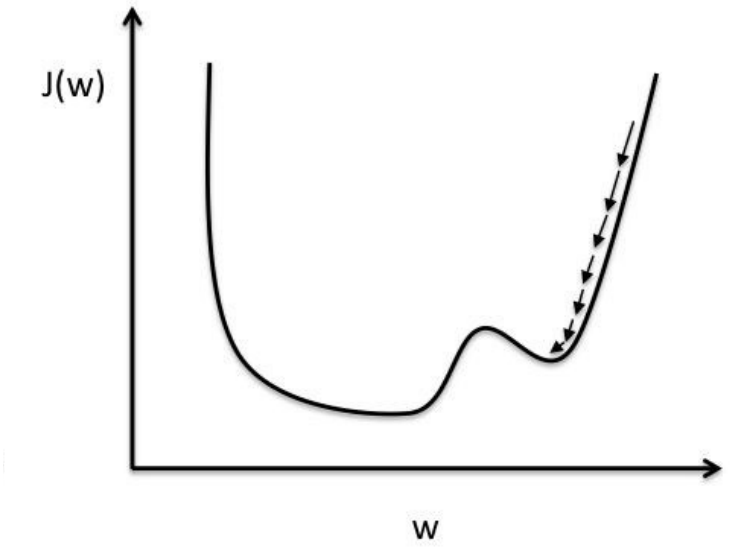
Update Rule:

$$\theta := \theta - \eta \frac{\partial J(\theta)}{\partial \theta}$$

(Stolen from N. Locascio)

Learning Rate & Optimization

- Small Learning Rate

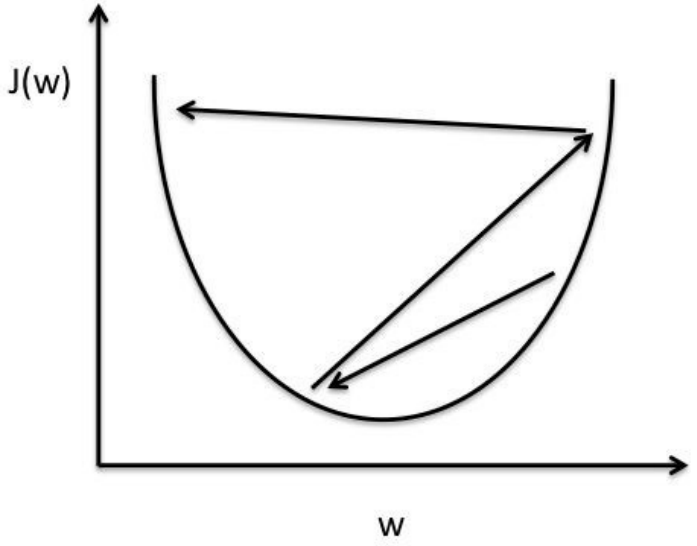


Small learning rate: Many iterations until convergence and trapping in local minima.

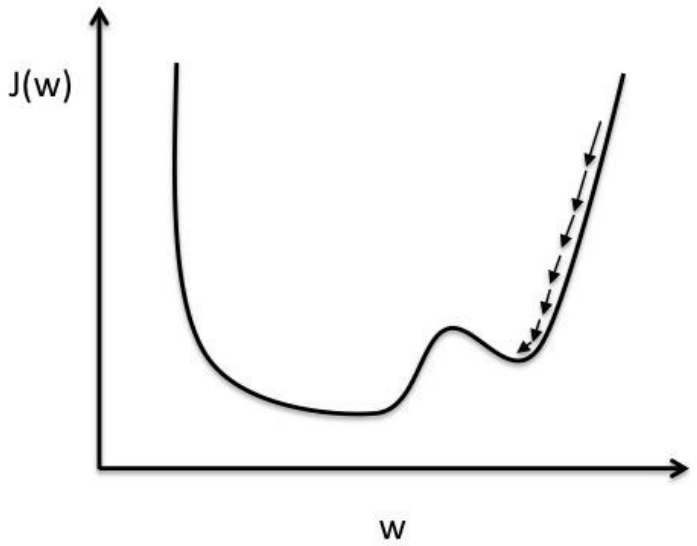
(Stolen from N. Locascio)

Learning Rate & Optimization

- Large learning rate



Large learning rate: Overshooting.



Small learning rate: Many iterations until convergence and trapping in local minima.

(Stolen from N. Locascio)

How to deal with this?

1. Try lots of different learning rates to see what is ‘just right’

(Stolen from N. Locascio)

How to deal with this?

1. Try lots of different learning rates to see what is ‘just right’
2. Do something smarter

(Stolen from N. Locascio)

How to deal with this?

1. Try lots of different learning rates to see what is ‘just right’
2. **Do something smarter : Adaptive Learning Rate**

(Stolen from N. Locascio)

Adaptive Learning Rate

- Learning rate is no longer fixed
- Can be made larger or smaller depending on:
  - how large gradient is
  - how fast learning is happening
  - size of particular weights
  - etc

(Stolen from N. Locascio)

Adaptive Learning Rate Algorithms

- ADAM
- Momentum
- NAG
- Adagrad
- Adadelta
- RMSProp

For details: check out http://sebastianruder.com/optimizing-gradient-descent/

(Stolen from N. Locascio)

Mini Batches

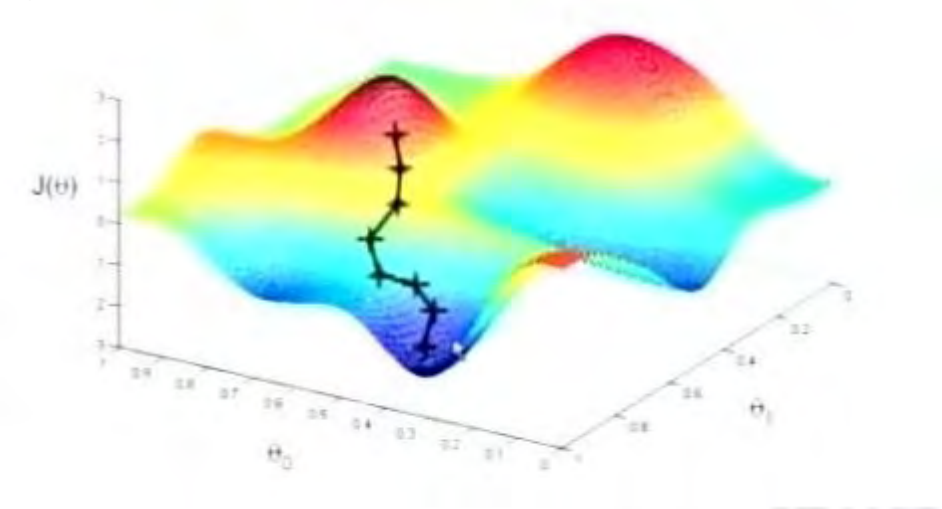
(Stolen from N. Locascio)

Why is it Stochastic Gradient Descent?

- Initialize θ randomly
- For N Epochs
  - For each training example (x, y):
    - Compute Loss Gradient: $\frac{\partial J(\theta)}{\partial \theta}$
    - Update θ with update rule:

$$\theta := \theta - \eta \frac{\partial J(\theta)}{\partial \theta}$$

Only an estimate of true gradient!



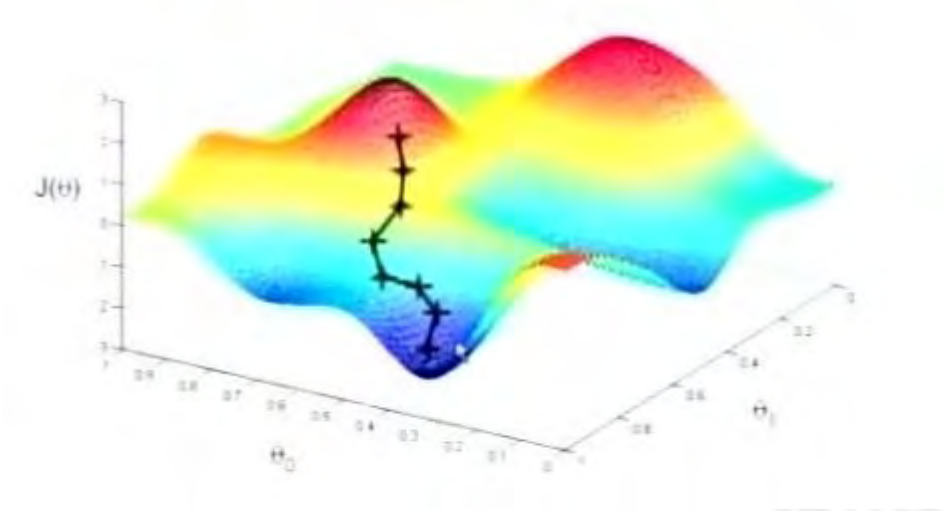
(Stolen from N. Locascio)

Minibatches Reduce Gradient Variance

- Initialize θ randomly
- For N Epochs
  - For each training **batch {(x0, y0), … , (x_B, y_B)}:**
    - Compute Loss Gradient: $\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{B}\sum_{i}^{B}\frac{\partial J_i(\theta)}{\partial \theta}$
    - Update θ with update rule:

$$\theta := \theta - \eta\frac{\partial J(\theta)}{\partial \theta}$$

More accurate estimate!



(Stolen from N. Locascio)

Advantages of Minibatches

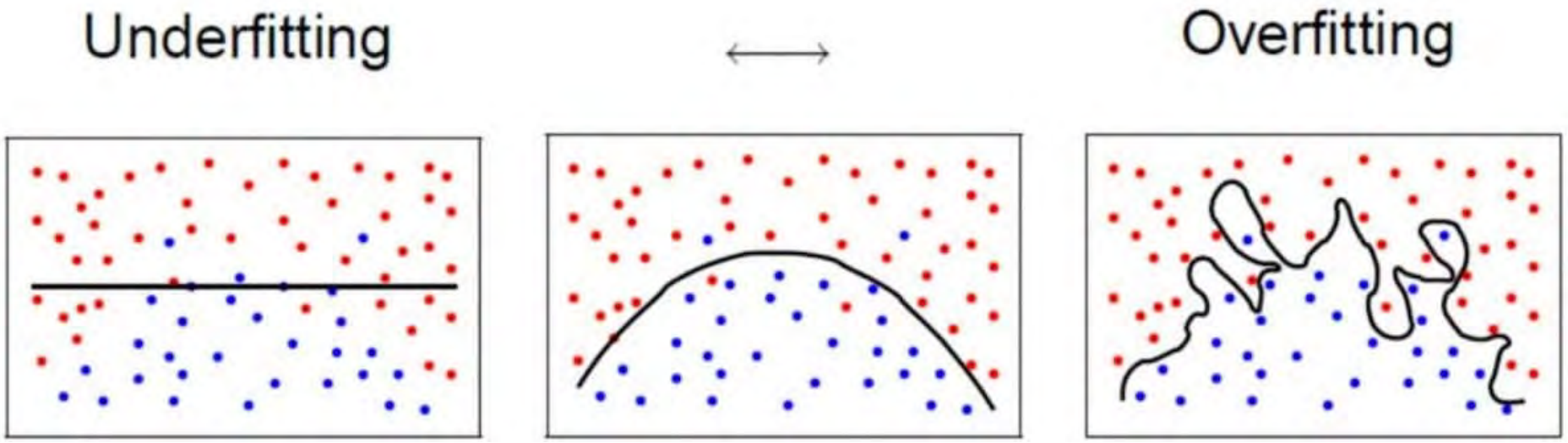
- More accurate estimation of gradient
  - Smoother convergence
  - Allows for larger learningrates
- Minibatches lead to fasttraining!
  - Can parallelize computation + achieve significant speed increases on GPU's

(Stolen from N. Locascio)

Training Neural Networks In Practice: Fighting Overfitting

(Stolen from N. Locascio)

The Problem of Overfitting



(Stolen from N. Locascio)

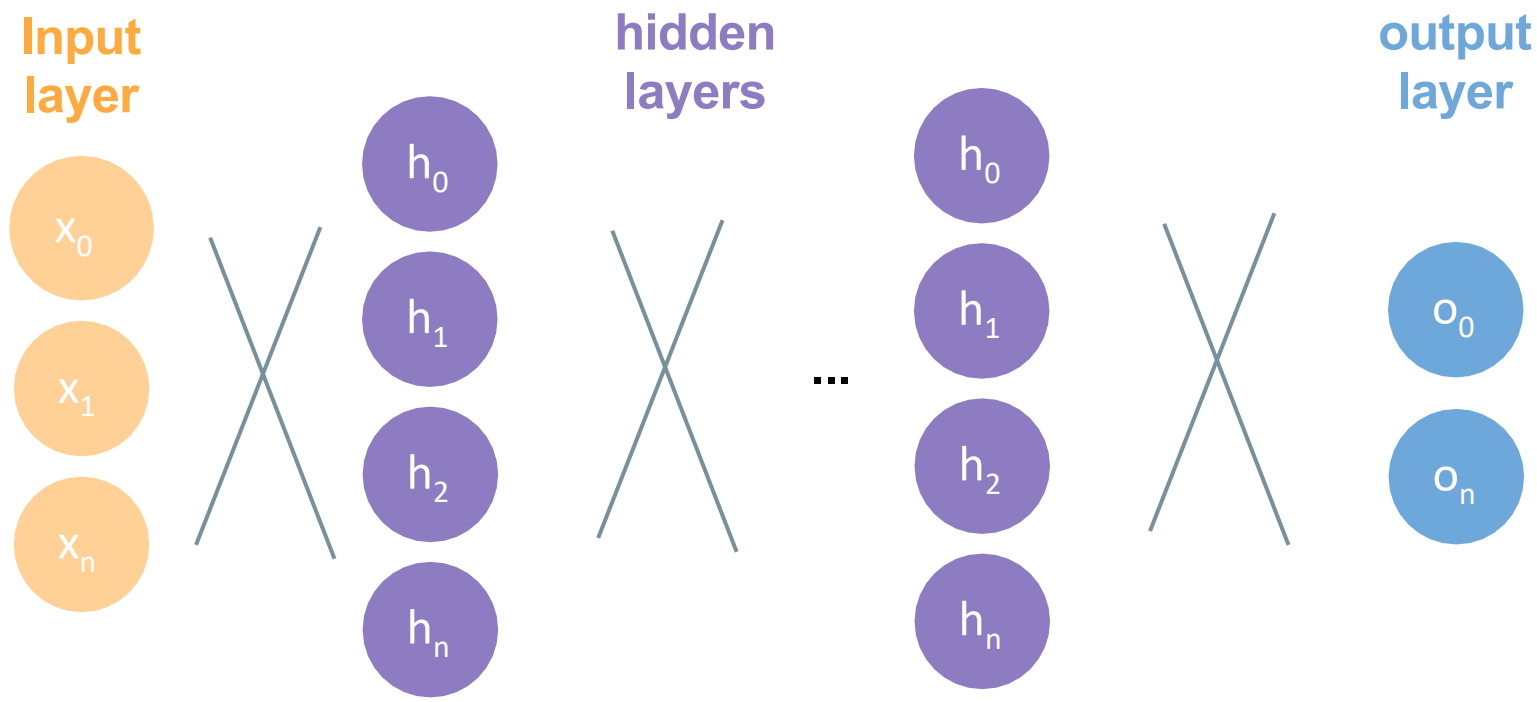
Regularization Techniques

1. Dropout
2. Early Stopping
3. Weight Regularization
4. ...many more

(Stolen from N. Locascio)

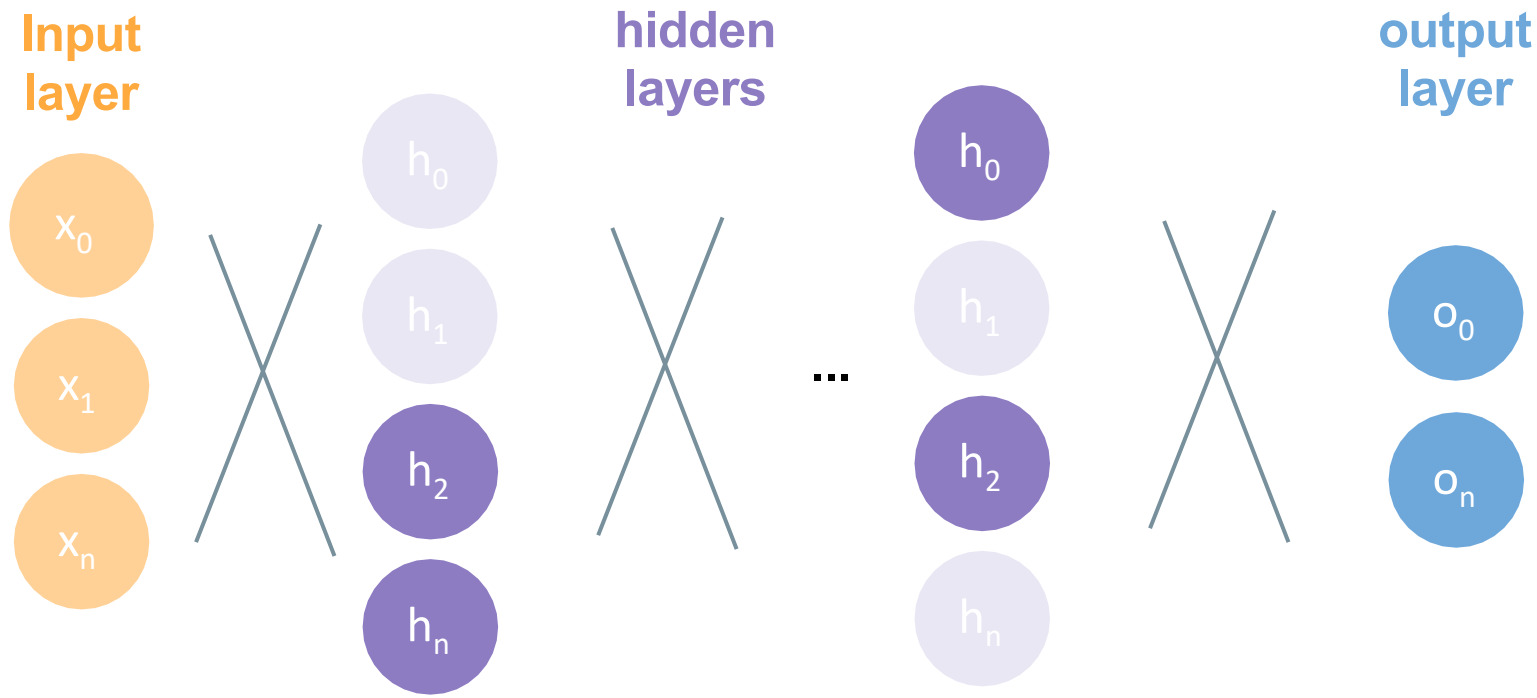
Regularization I: Dropout

- During training, randomly set some activations to 0



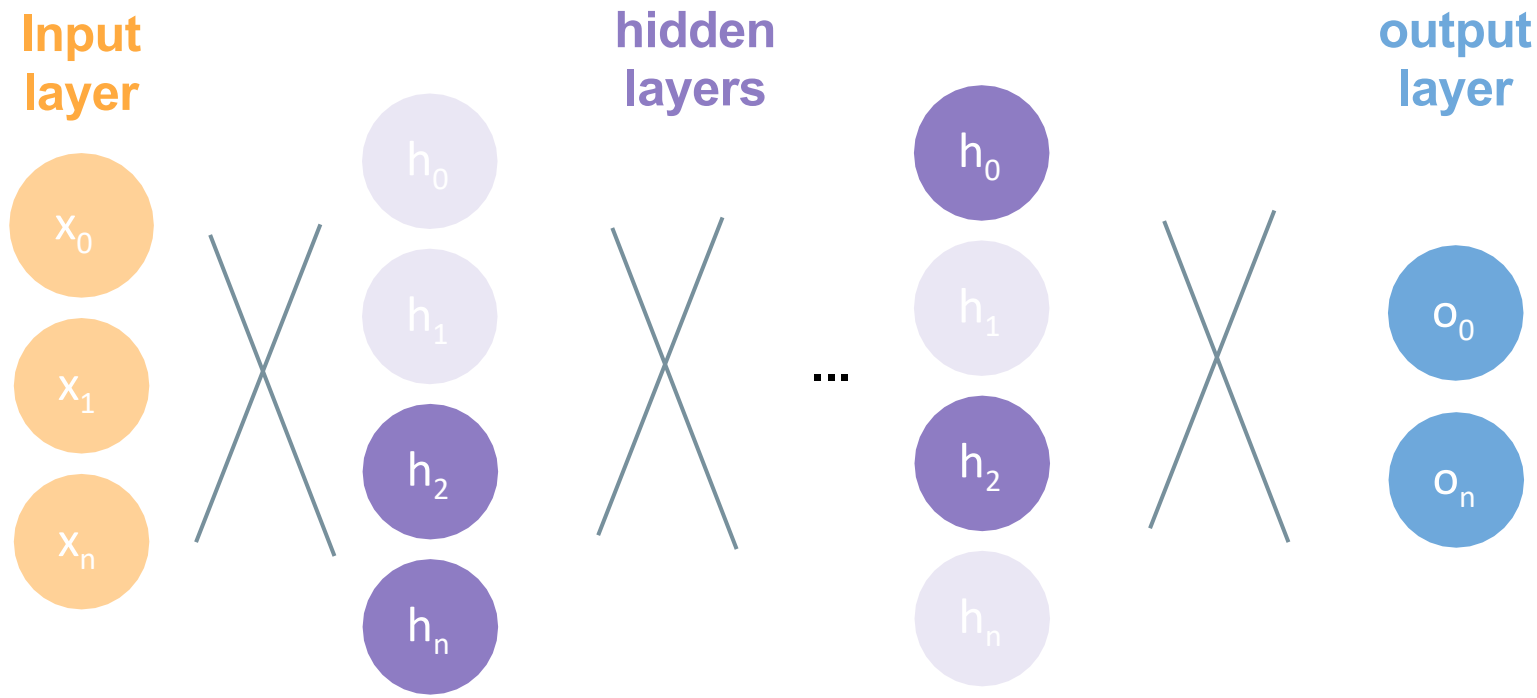
Regularization I: Dropout

- During training, randomly set some activations to 0



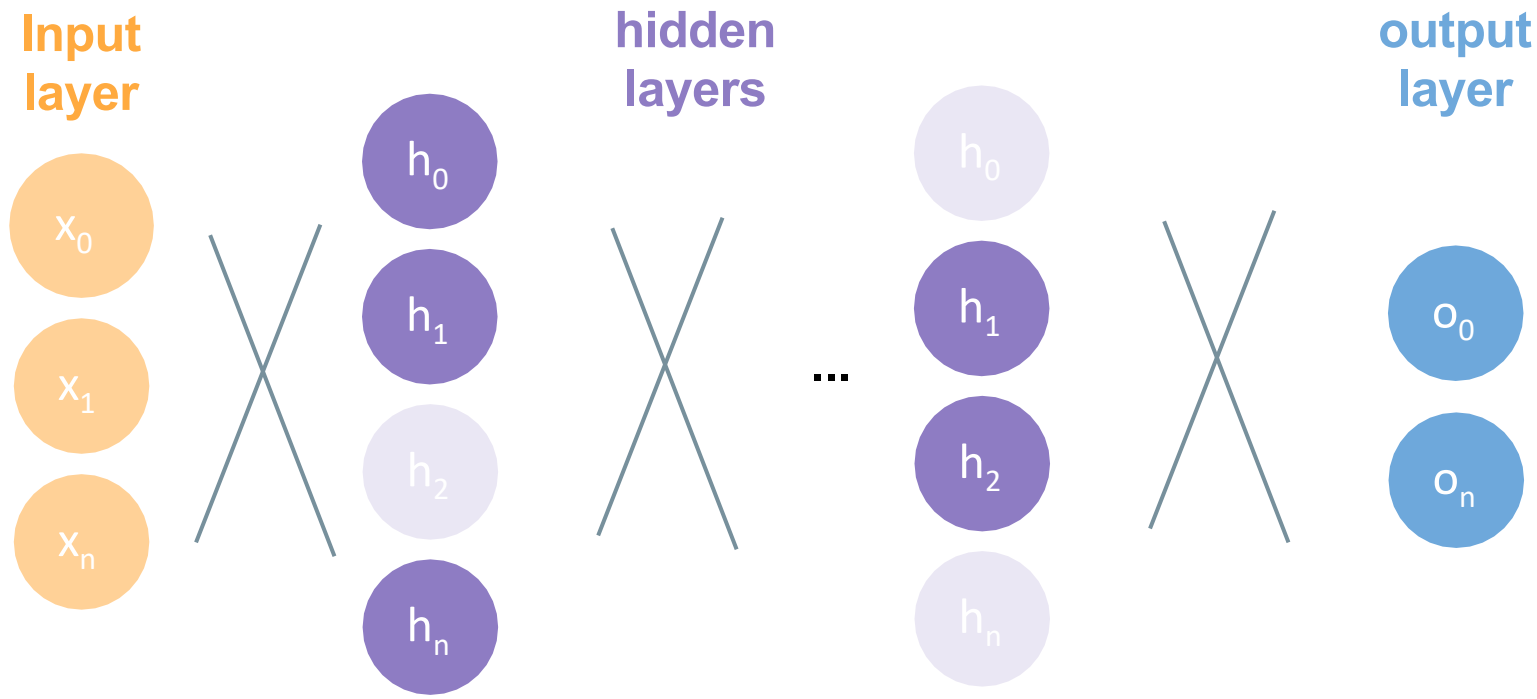
Regularization I: Dropout

- During training, randomly set some activations to 0
  - Typically ‘drop’ 50% of activations in layer
  - Forces network to not rely on any 1 node



Regularization I: Dropout

- During training, randomly set some activations to 0
  - Typically ‘drop’ 50% of activations in layer
  - Forces network to not rely on any 1 node



(Stolen from N. Locascio)

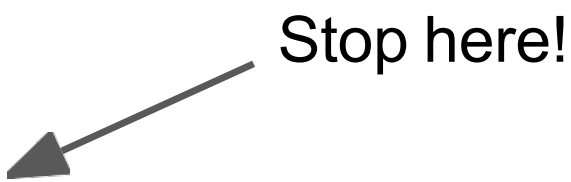
Regularization II: Early Stopping

- Don’t give the network time to overfit
- ...
- **Epoch 15:** Train: 85% Validation: 80%
- **Epoch 16:** Train: 87% Validation: 82%
- **Epoch 17:** Train: 90% Validation: 85%
- **Epoch 18:** Train: 95% Validation: 83%
- **Epoch 19:** Train: 97% Validation: 78%
- **Epoch 20:** Train: 98% Validation: 75%

(Stolen from N. Locascio)

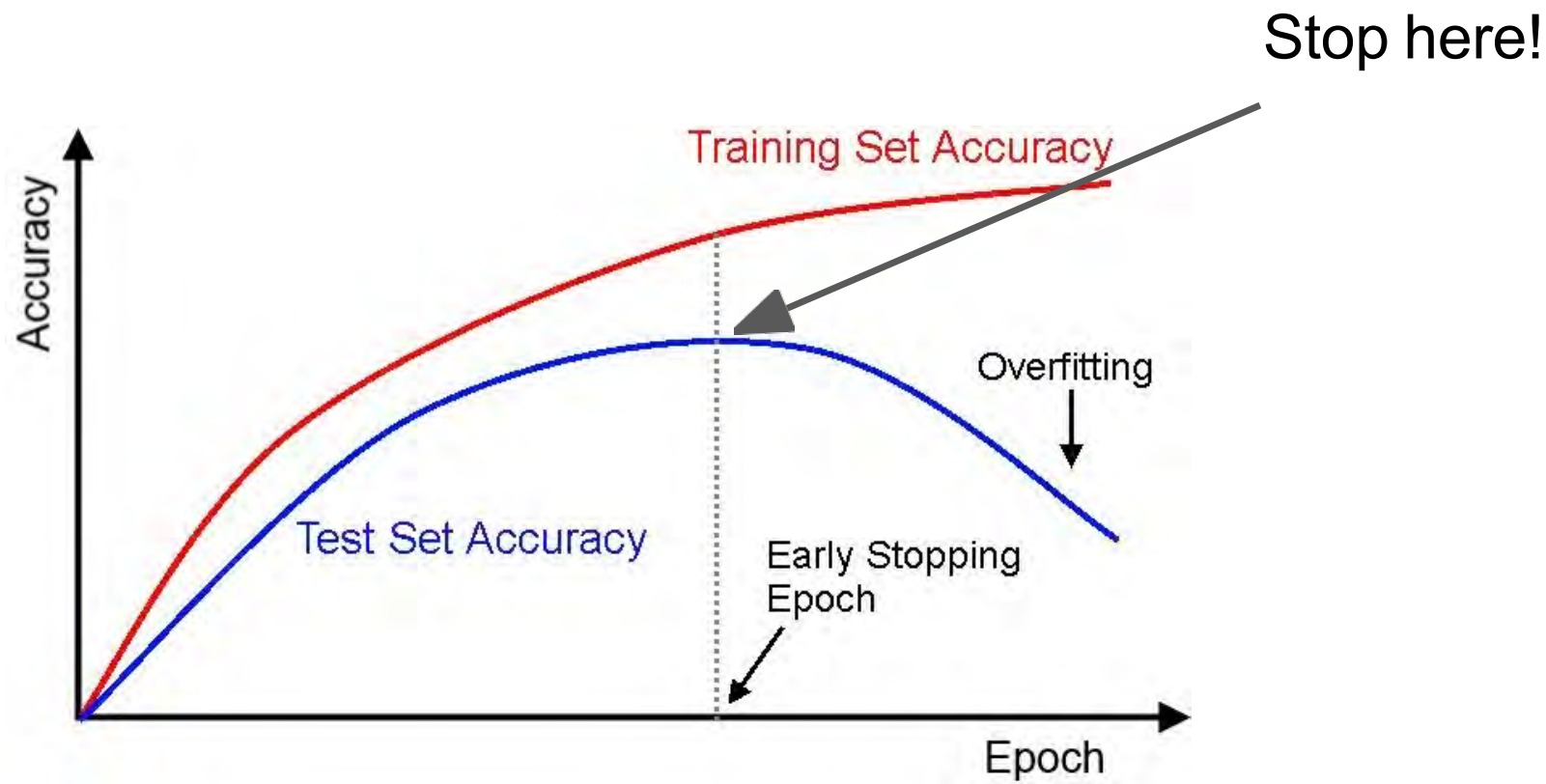
Regularization II: Early Stopping

- Don’t give the network time to overfit
- ...
- **Epoch 15:** Train: 85% Validation: 80%
- **Epoch 16:** Train: 87% Validation: 82%
- **Epoch 17: Train: 90% Validation: 85%**
- **Epoch 18:** Train: 95% Validation: 83%
- **Epoch 19:** Train: 97% Validation: 78%
- **Epoch 20:** Train: 98% Validation: 75%



(Stolen from N. Locascio)

Regularization II: Early Stopping



(Stolen from N. Locascio)

Regularization III: Weight Regularization

- Large weights typically mean model is overfitting
- Add the size of the weights to our loss function
- Perform well on task + keep weights small

(Stolen from N. Locascio)

Regularization III: Weight Regularization

- Large weights typically mean model is overfitting
- Add the size of the weights to our loss function
- Perform well on task + keep weights small

$$\arg_\theta \min \frac{1}{T} \sum_t \text{loss}(f(x^{(t)}; \theta), y^{(t)}))$$

(Stolen from N. Locascio)

Regularization III: Weight Regularization

- Large weights typically mean model is overfitting
- Add the size of the weights to our loss function
- Perform well on task + keep weights small

$$\arg_\theta \min \frac{1}{T} \sum_t \text{loss}(f(x^{(t)}; \theta), y^{(t)})) \; + \lambda \sum_i (\theta_i)^2$$

(Stolen from N. Locascio)

Regularization III: Weight Regularization

- Large weights typically mean model is overfitting
- Add the size of the weights to our loss function
- Perform well on task + keep weights small

$$\arg_\theta \min \underbrace{\frac{1}{T}\sum_t \text{loss}(f(x^{(t)};\theta), y^{(t)})) \; + \lambda \sum_i (\theta_i)^2}_{J(\theta)}$$

(Stolen from N. Locascio)

Next lecture: Convolutional Neural Networks and Driving Related Tasks