

# AA 274

# Principles of Robotic Autonomy

Information extraction and intro to  
machine learning for robot autonomy



**Stanford**  
University



# Techniques for information extraction

- Aim
  - Learn how to extract information from sensor measurements
- Readings
  - SNS: 4.1.3, 4.6.1 - 4.6.5, 4.7.1 - 4.7.4

# Information extraction

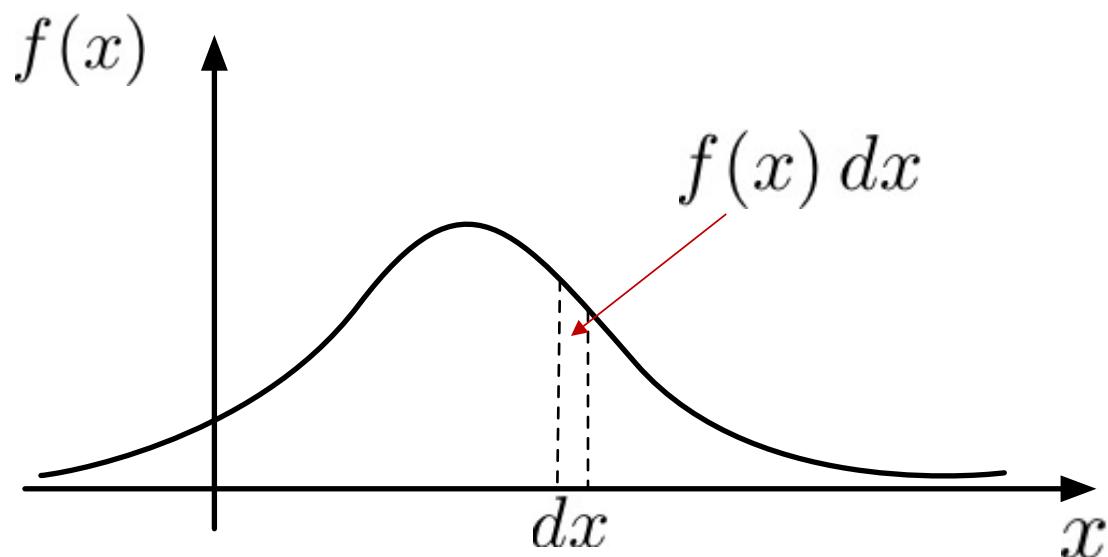
- Next step is to extract *information* from images, such as
  - Geometric primitives (e.g., lines and circles): useful, for example, for robot localization and mapping
  - Object recognition and scene understanding: useful, for example, for localization within a topological map and for high-level reasoning
- Since sensors are noisy, first step is to discuss a mathematical characterization of uncertainty

# Representing uncertainty

- Sensors are imperfect devices with errors of both *systematic* and *random* nature
- **Goal:** mathematical tools to characterize the uncertainty of a sensor, and to scale up such uncertainty to the entire robot system
- Strategy:
  - Quantify uncertainty of a sensor via a *statistical* representation
  - Find a relation that describes how uncertainty *propagates*
- In the following, denote with  $X$  the random variable describing the outcome of a given sensor measurement

# Step #1: statistical representation

- Statistical properties of  $X$  are captured by a probability density function (pdf)  $f(x)$



Mean value

$$\mu = \mathbb{E}[X] := \int_{-\infty}^{\infty} x f(x) dx$$

2/4/19

Probability  $X$  falls within an interval  $[a,b]$

$$\mathbb{P}(a \leq X \leq b) = \int_a^b f(x) dx$$

Normalization

$$\int_{-\infty}^{\infty} f(x) dx = 1$$

Variance

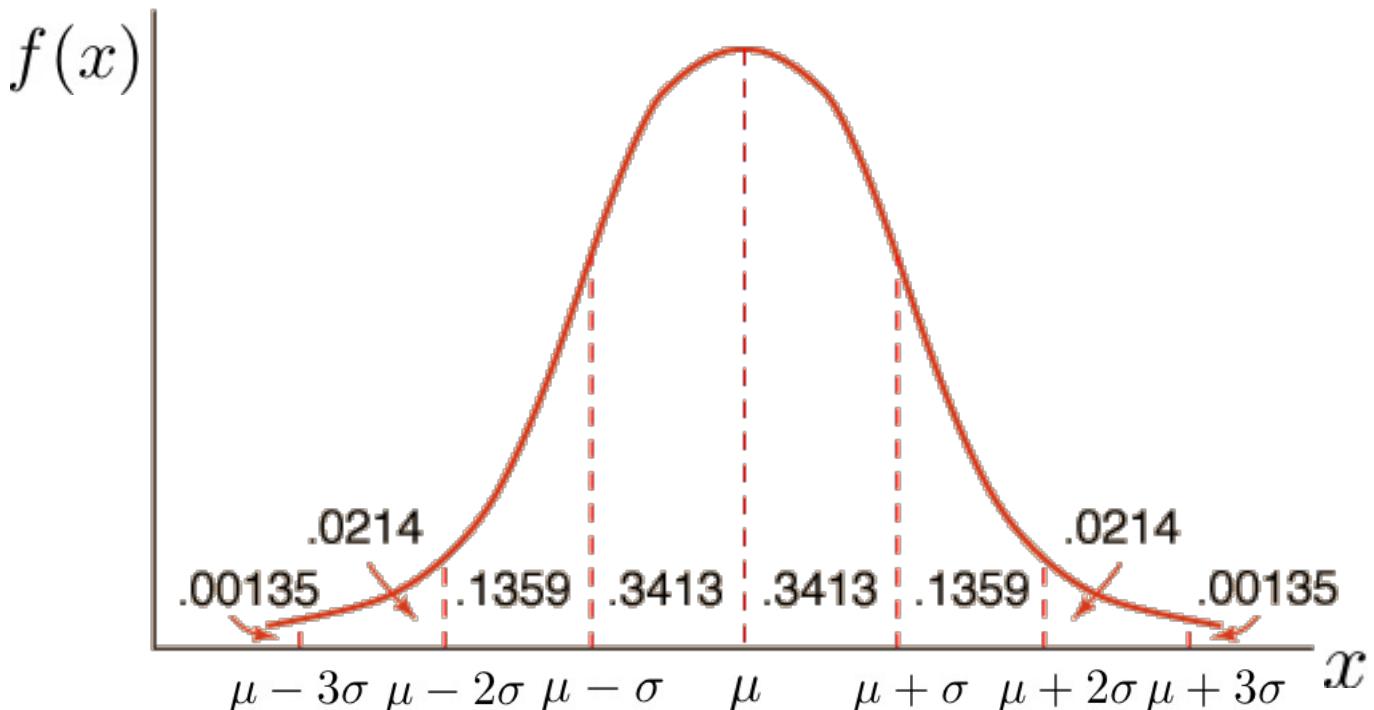
$$\sigma^2 = \text{var}[X] := \int_{-\infty}^{\infty} (x - \mu)^2 f(x) dx$$

AA 274 | Lecture 8

# Gaussian distribution

- A very common pdf to characterize measurement uncertainty is the Gaussian distribution

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)$$



# Independence of random variables

- When combining multiple random variables (e.g., measurements), a critical property is that of *independence*
- Two random variables  $X$  and  $Y$  are independent if and only if

$$f_{X,Y}(x,y) = f_X(x)f_Y(y)$$

- Important properties of independent random variables:
  - Expected value of product of independent random variables  $X$  and  $Y$

$$\mathbb{E}[X \cdot Y] = \mathbb{E}[X] \cdot \mathbb{E}[Y]$$

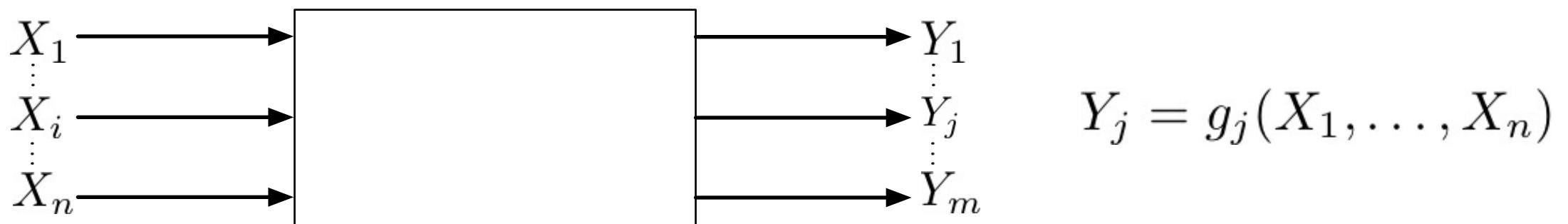
- Variance of the sum of independent random variables  $X$  and  $Y$

$$\text{var}[X + Y] = \text{var}[X] + \text{var}[Y]$$

# Error propagation

- Several input measurements are often combined to determine a number of output quantities

Multi-input multi-output system with  $n$  inputs and  $m$  outputs



- How do input uncertainties (i.e.,  $X_i$ 's uncertainties) propagate to the output?

# Error propagation law: 1D

- Performing a first-order Taylor series expansion at the point  $X = \mu_X$

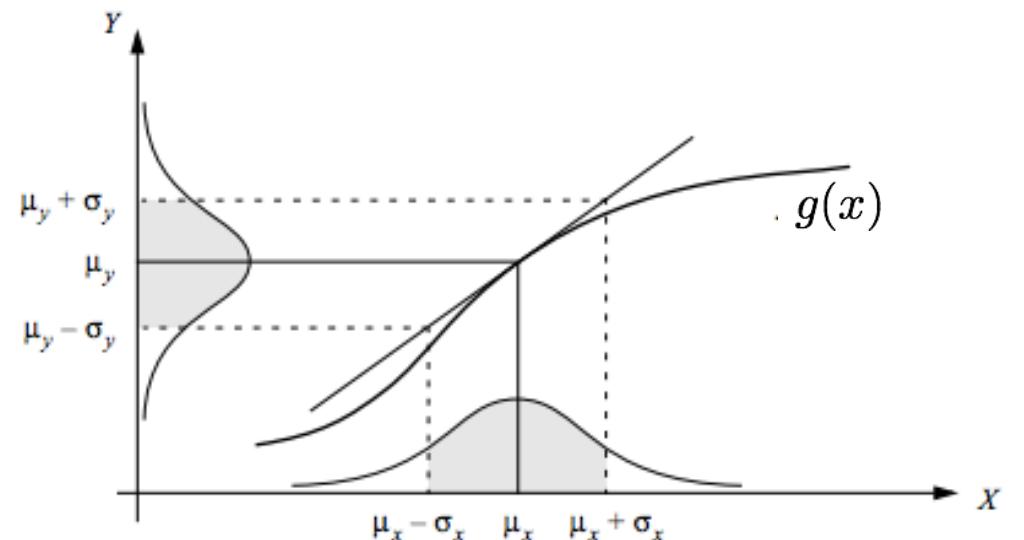
$$Y \approx g(\mu_X) + \frac{\partial g}{\partial X} \Big|_{X=\mu_X} (X - \mu_X)$$

- Moments for  $Y$  variable

$$\mu_Y = g(\mu_X)$$

$$\sigma_Y^2 = \left( \frac{\partial g}{\partial X} \Big|_{X=\mu_X} \right)^2 \sigma_X^2$$

Error propagation law



# Error propagation law: general case

- Moments for  $Y$  variable

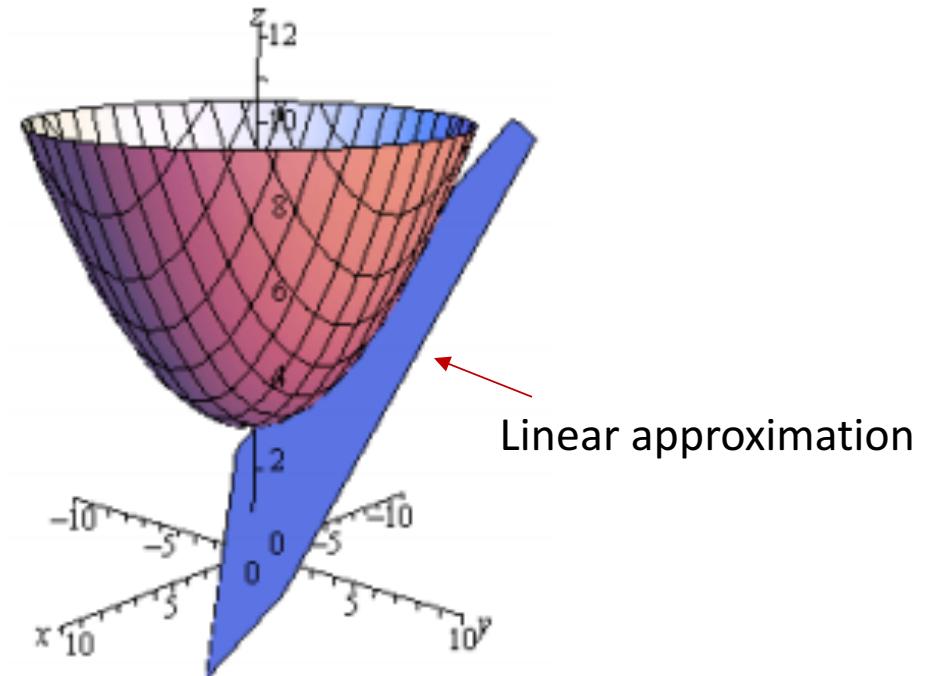
$$\mu_{Y_j} = g_j(\mu_{X_1}, \dots, \mu_{X_n})$$

$$C_Y = G_X C_X G_X^T$$

Error propagation law

where

$$G_X := \begin{bmatrix} \frac{\partial g_1}{\partial X_1} & \dots & \frac{\partial g_1}{\partial X_n} \\ \vdots & \dots & \vdots \\ \frac{\partial g_m}{\partial X_1} & \dots & \frac{\partial g_m}{\partial X_n} \end{bmatrix} \quad C_X := \begin{bmatrix} \sigma_{X_1}^2 & \sigma_{X_1}\sigma_{X_2} & \dots & \sigma_{X_1}\sigma_{X_n} \\ \sigma_{X_2}\sigma_{X_1} & \sigma_{X_2}^2 & \dots & \sigma_{X_2}\sigma_{X_n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{X_n}\sigma_{X_1} & \sigma_{X_n}\sigma_{X_2} & \dots & \sigma_{X_n}^2 \end{bmatrix} \quad C_Y := \begin{bmatrix} \sigma_{Y_1}^2 & \sigma_{Y_1}\sigma_{Y_2} & \dots & \sigma_{Y_1}\sigma_{Y_m} \\ \sigma_{Y_2}\sigma_{Y_1} & \sigma_{Y_2}^2 & \dots & \sigma_{Y_2}\sigma_{Y_m} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{Y_m}\sigma_{Y_1} & \sigma_{Y_m}\sigma_{Y_2} & \dots & \sigma_{Y_m}^2 \end{bmatrix}$$



# Geometric feature extraction

- **Geometric feature extraction:** extract geometric primitives from sensor data (e.g., range data)
- Examples: line, circles, corners, planes, etc.
- We focus on *line extraction* from range data (a quite common task); other geometric feature extraction tasks are conceptually analogous
- The two main problems of line extraction from range data
  1. Which points belong to which line? -> *segmentation*
  2. Given an association of points to a line, how to estimate line parameters? -> *fitting*

## Step #2: line fitting

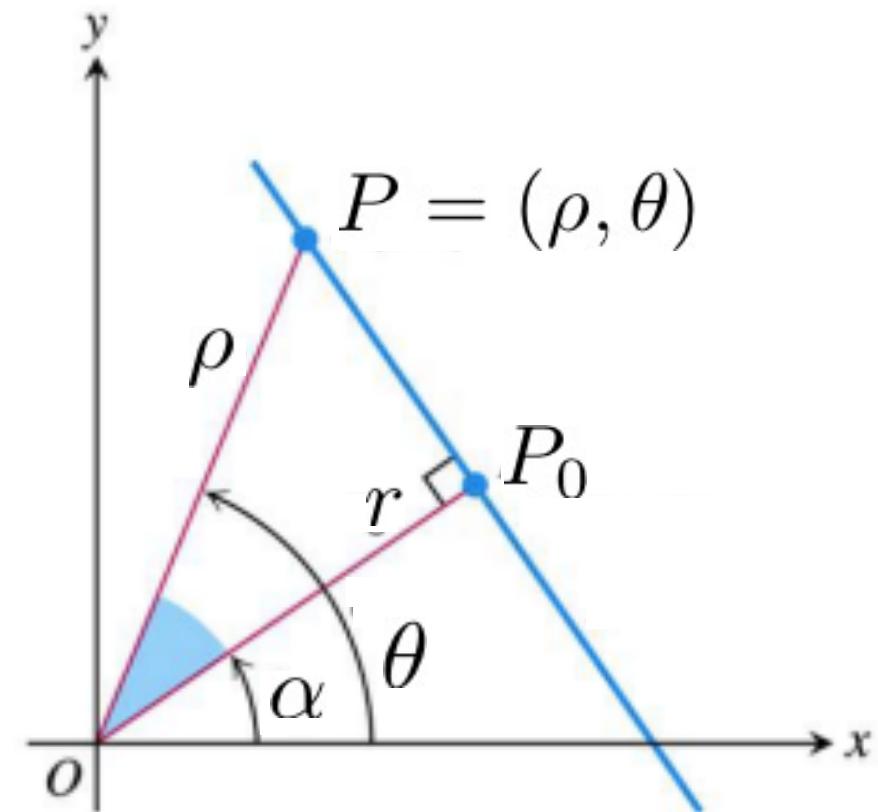
- **Goal:** fit a line to a set of sensor measurements
- It is useful to work in polar coordinates:

$$x = \rho \cos \theta, \quad y = \rho \sin \theta$$

- Equation of a line in polar coordinates
  - Let  $P = (\rho, \theta)$  be an arbitrary point on the line
  - Since  $P, P_0, O$  determine a right triangle

$$\boxed{\rho \cos(\theta - \alpha) = r} \quad \text{or} \quad x \cos \alpha + y \sin \alpha = r$$

- $(r, \alpha)$  are the parameters of the line



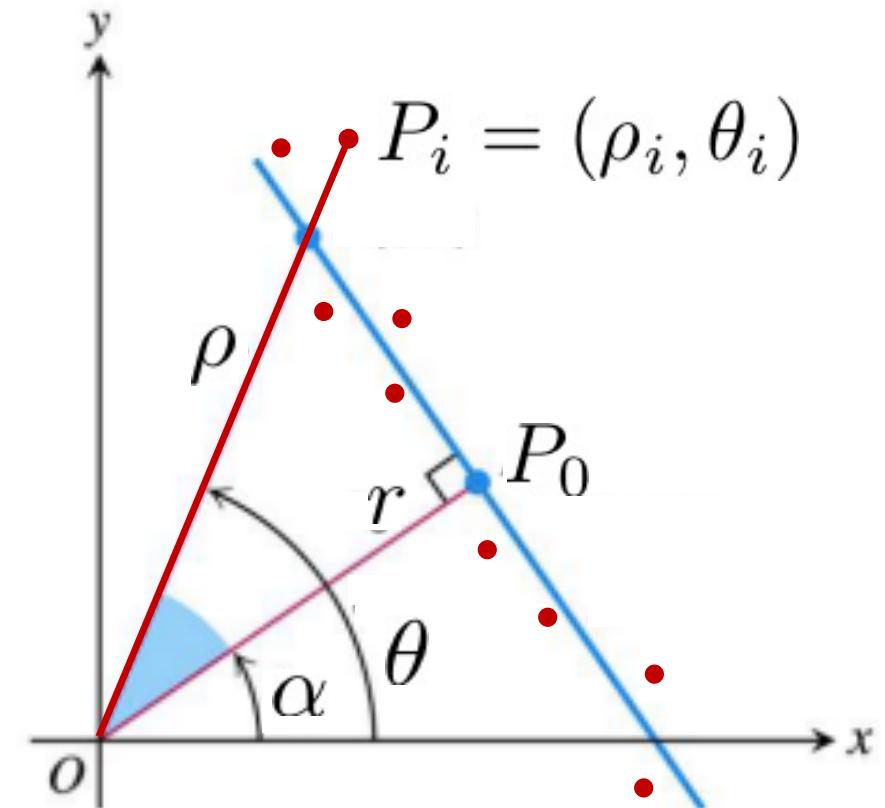
## Step #2: line fitting

- Since there is measurement error, the equation of the line is only *approximately* satisfied

$$\rho_i \cos(\theta_i - \alpha) = r + d_i$$

Error

- Assume  $n$  ranging measurement points represented in polar coordinates as  $(\rho_i, \theta_i)$
- We want to find a line that best “fits” all the measurement points



## Step #2: line fitting

- Consider, first, that all measurements are equally uncertain
- Find line parameters  $(r, \alpha)$  that minimize squared error

$$S(r, \alpha) := \sum_{i=1}^n d_i^2 = \sum_{i=1}^n (\rho_i \cos(\theta_i - \alpha) - r)^2$$

- Unweighted least squares

## Step #2: line fitting

- Consider, now, the case where each measurement has its own, unique uncertainty
- For example, assume that the variance for each range measurement  $\rho_i$  is  $\sigma_i$
- Associate with each measurement a weight, e.g.,  $w_i = 1/\sigma_i^2$
- Then, one minimizes

$$S(r, \alpha) := \sum_{i=1}^n w_i d_i^2 = \sum_{i=1}^n w_i (\rho_i \cos(\theta_i - \alpha) - r)^2$$

- Weighted least squares

## Step #2: line fitting solution

- Assume that the  $n$  ranging measurements are **independent**
- Solution:

$$r^* = \frac{\sum_i w_i \rho_i \cos(\theta_i - \alpha)}{\sum_i w_i} := g_1$$

$$\alpha^* = \frac{1}{2} \text{atan} \left( \frac{\sum_i w_i \rho_i^2 \sin 2\theta_i - \frac{2}{\sum_i w_i} \sum_i \sum_j w_i w_j \rho_i \rho_j \cos \theta_i \sin \theta_j}{\sum_i w_i \rho_i^2 \cos 2\theta_i - \frac{1}{\sum_i w_i} \sum_i \sum_j w_i w_j \rho_i \rho_j \cos(\theta_i + \theta_j)} \right) + \frac{\pi}{2} := g_2$$

- Uncertainty for the line  $l = (r^*, \alpha^*)$  can be characterized as

2x2:

$$C_l = G_X C_X G_X^T$$

where:

$$C_X = \begin{bmatrix} \sigma_{\rho_1}^2 & 0 & \dots \\ 0 & \sigma_{\theta_1}^2 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix} \quad G_X = \begin{bmatrix} \frac{\partial g_1}{\partial \rho_1} & \frac{\partial g_1}{\partial \theta_1} & \dots \\ \frac{\partial g_2}{\partial \rho_1} & \frac{\partial g_2}{\partial \theta_1} & \dots \end{bmatrix}$$

# Step #1: line segmentation

- Several algorithms are available
- We will consider three popular algorithms
  1. Split-and-merge
  2. RANSAC
  3. Hough-Transform

# Split-and-merge algorithm

- Most popular line extraction algorithm

---

**Data:** Set  $S$  consisting of all  $N$  points, a distance threshold  $d > 0$

**Result:**  $L$ , a list of sets of points each resembling a line

$L \leftarrow (S), i \leftarrow 1;$

**while**  $i \leq \text{len}(L)$  **do**

fit a line  $(r, \alpha)$  to the set  $L_i$ ;

detect the point  $P \in L_i$  with the maximum distance  $D$  to the line  $(r, \alpha)$ ;

**if**  $D < d$  **then**

$i \leftarrow i + 1$

**else**

split  $L_i$  at  $P$  into  $S_1$  and  $S_2$ ;

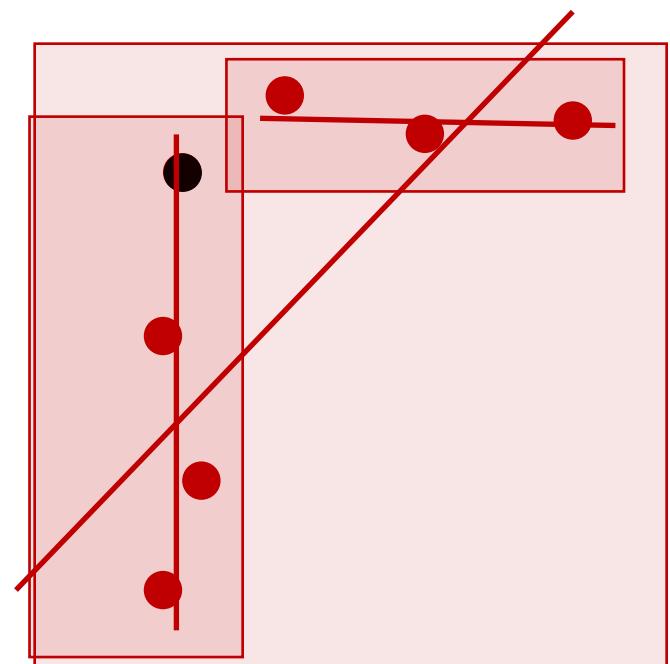
$L_i \leftarrow S_1; L_{i+1} \leftarrow S_2;$

**end**

**end**

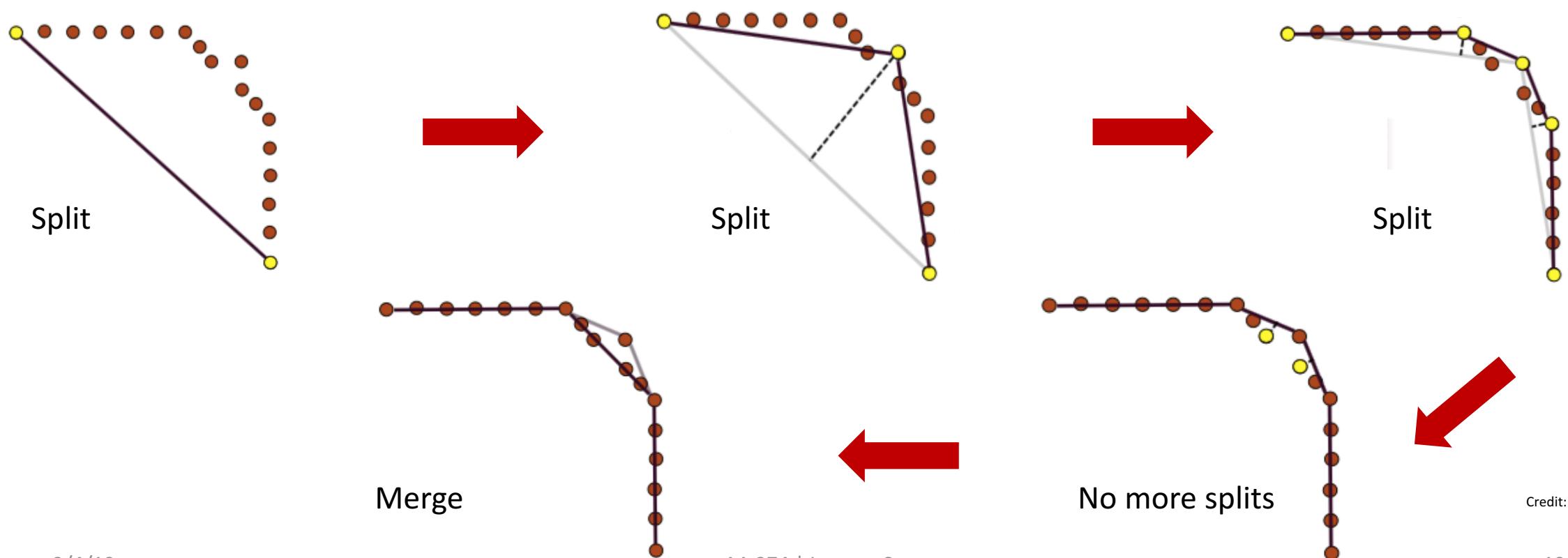
Merge collinear sets in  $L$ ;

---



# Split-and-merge: iterative-end-point-fit variant

- Iterative-end-point-fit: split-and-merge where the line is constructed by simply connecting the first and last points



# RANSAC

- RANSAC: **Random Sample Consensus**
- General method to estimate parameters of a model from a set of observed data in the presence of outliers, where outliers should have no influence on the estimates of the values
- Typical applications in robotics: line extraction from 2D range data, plane extraction from 3D point clouds, feature matching for structure from motion, etc.
- RANSAC is *iterative* and *non-deterministic*: the probability of finding a set free of outliers increases as more iterations are used

# RANSAC

**Data:** Set  $S$  consisting of all  $N$  points

**Result:** Set with maximum number of inliers  
(and corresponding fitting line)

**while**  $i \leq k$  **do**

    randomly select 2 points from  $S$ ;

    fit line  $l_i$  through the 2 points;

    compute distance of all other points to line  $l_i$  ;

    construct *inlier* set, i.e., count number of

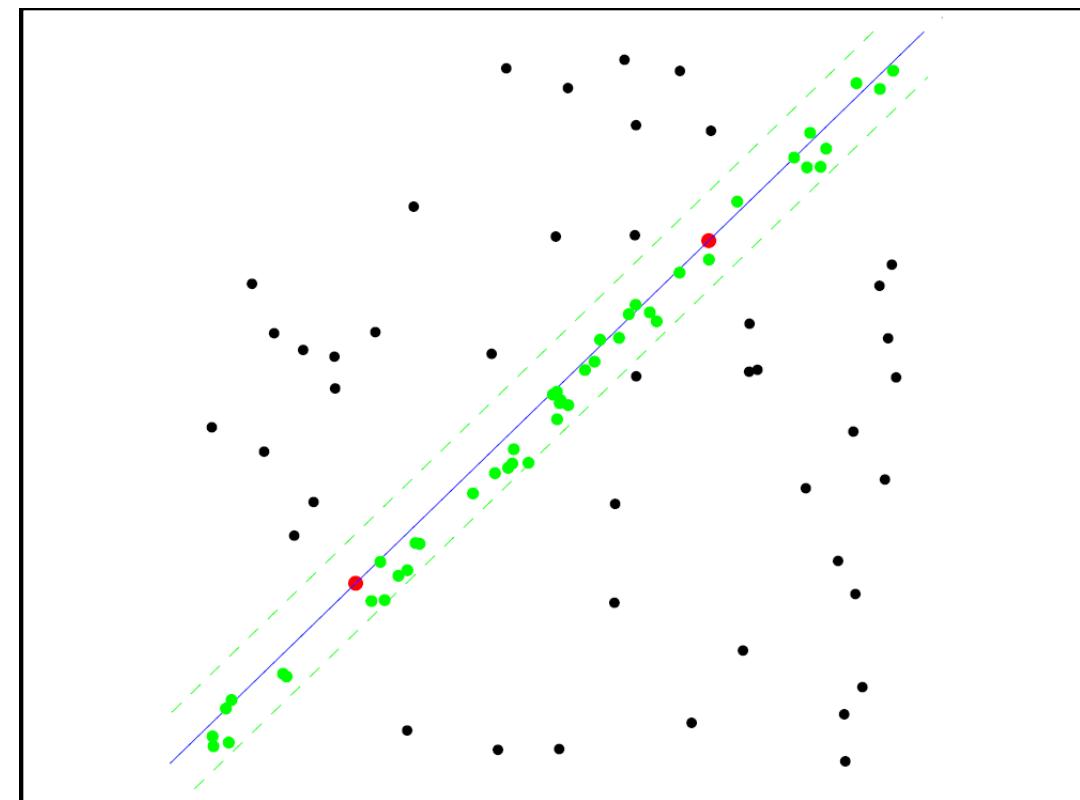
        points with distance to the line less than  $\gamma$ ;

    store line  $l_i$  and associated set of inliers;

$i \leftarrow i + 1$

**end**

Choose set with maximum number of inliers



# RANSAC iterations

- In principle, one would need to check all possible combinations of 2 points in dataset
- If  $|S| = N$ , number of combinations is  $N(N - 1)/2 \rightarrow$  too many
- However, if we have a rough estimate of the percentage of inliers, we do not need to check all combinations...

# RANSAC iterations: statistical characterization

- Let  $w$  be the percentage of inliers in the dataset, i.e.,

$$w = \frac{\text{number of inliers}}{N}$$

- Let  $p$  be the desired probability of finding a set of points free of outliers (typically,  $p = 0.99$ )
- Assumption: 2 points chosen for line estimation are selected independently
  - $P(\text{both points selected are inliers}) = w^2$
  - $P(\text{at least one of the selected points is an outlier}) = 1 - w^2$
  - $P(\text{RANSAC never selects two points that are both inliers}) = (1 - w^2)^k$

# RANSAC iterations: statistical characterization

- Then minimum number of iterations  $\bar{k}$  to find an outlier-free set with probability at least  $p$  is:

$$1 - p = (1 - w^2)^{\bar{k}} \Rightarrow \bar{k} = \frac{\log(1 - p)}{\log(1 - w^2)}$$

- Thus if we know  $w$  (at least approximately), after  $\bar{k}$  iterations RANSAC will find a set free of outliers with probability  $p$
- Note:
  - $\bar{k}$  depends only on  $w$ , not on  $N$ !
  - More advanced versions of RANSAC estimate  $w$  adaptively

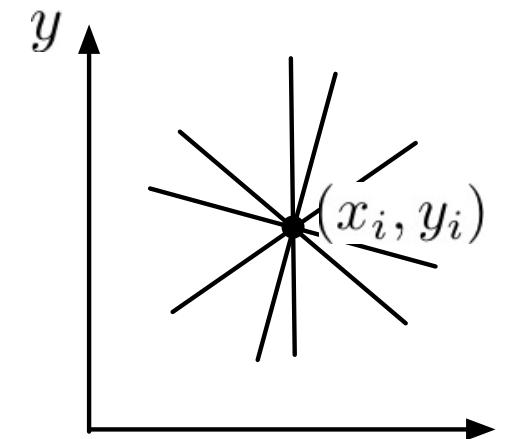
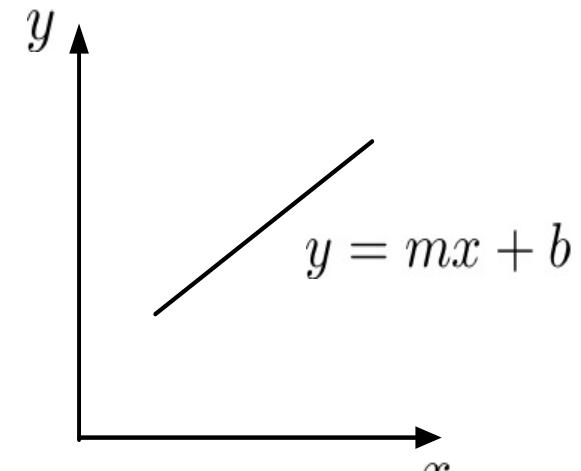
# Hough transform

- **Key idea:** each point votes for a set of plausible line parameters

- A line has two parameters:  $(m, b)$

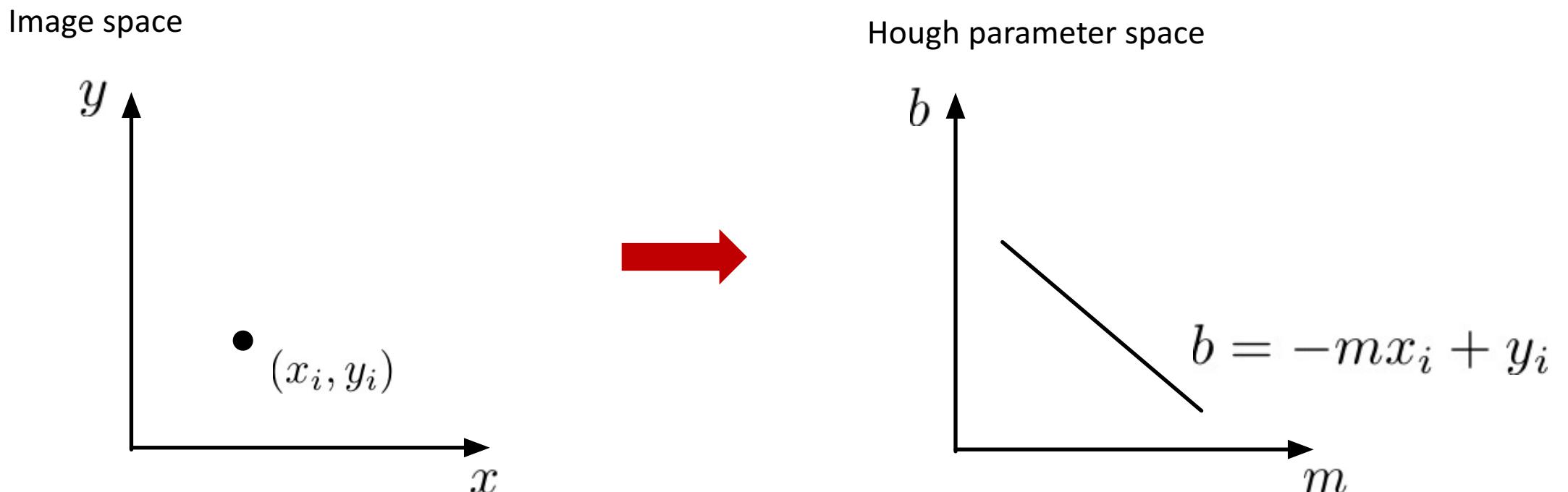
- Given a point  $(x_i, y_i)$ , the lines that could pass through this point are all  $(m, b)$  satisfying

$$y_i = mx_i + b, \quad \text{or} \quad b = -mx_i + y_i$$



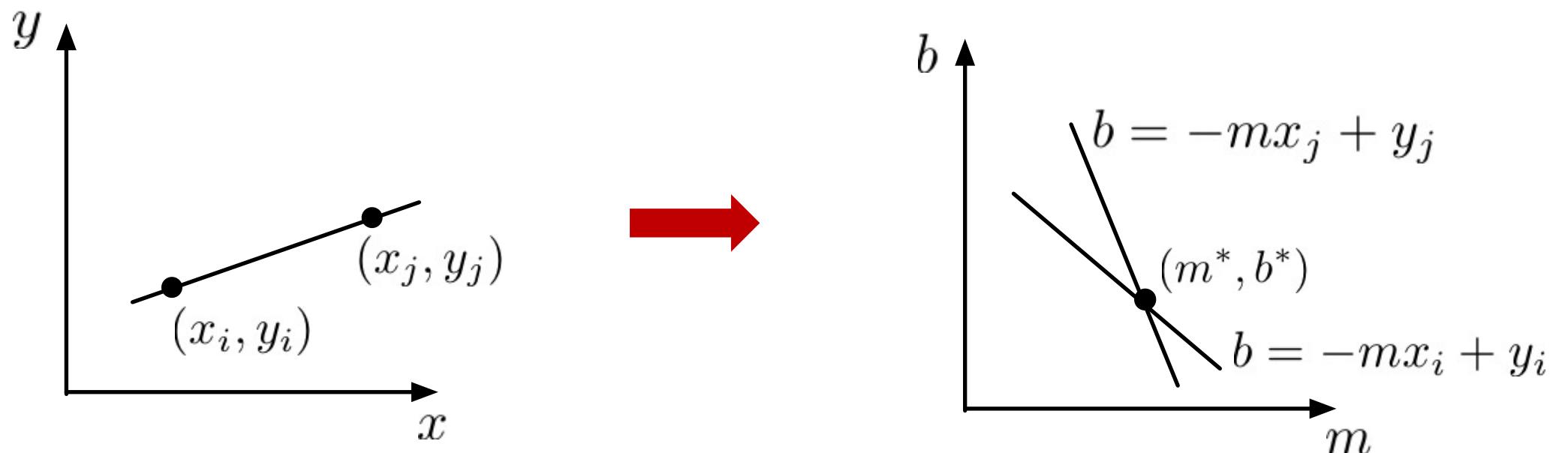
# Hough transform

- A point in image space maps into a line in *Hough space*



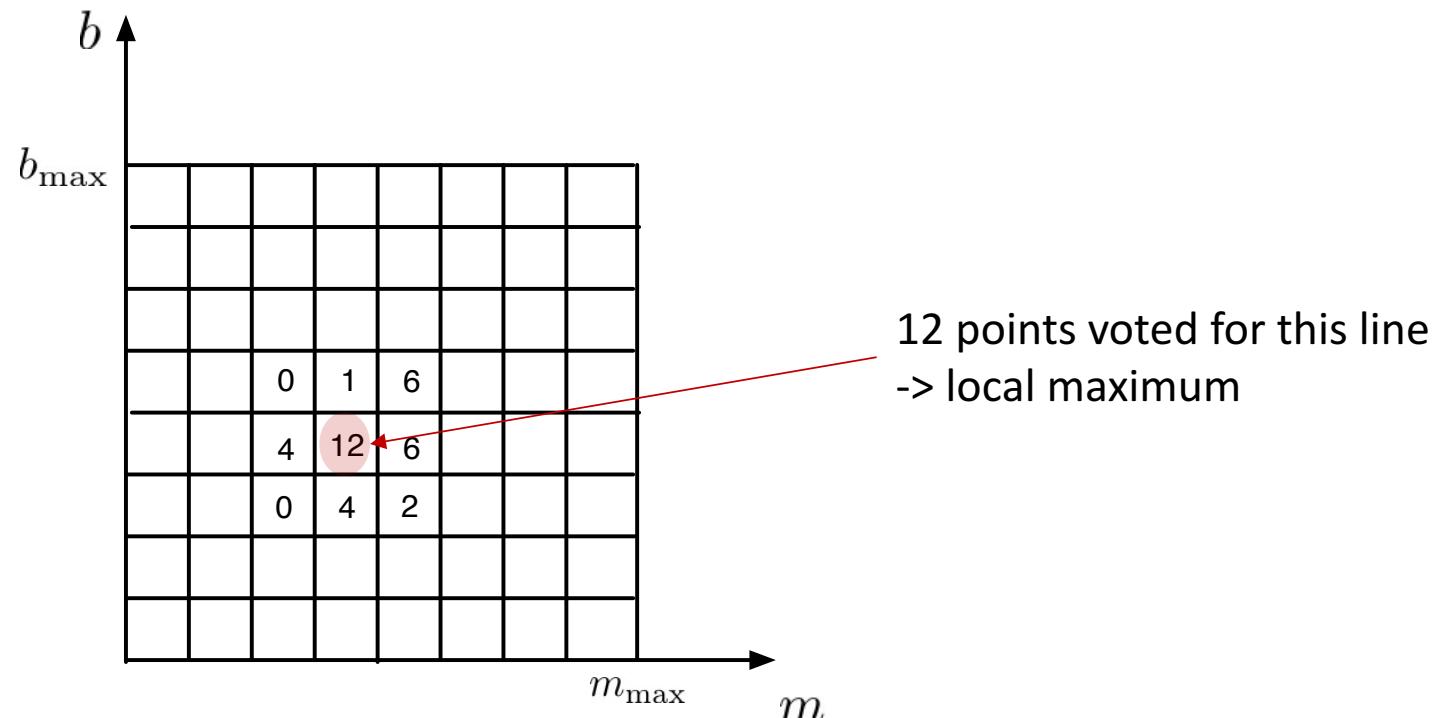
# Hough transform

- **Key fact:** all points on a line in image space yield lines in parameter space which intersect at a *common point*,  $(m^*, b^*)$



# Hough transform algorithm

1. Initialize an accumulator array  $H(m, b)$  to zero
2. For each point  $(x_i, y_i)$ , increment all cells that satisfy  $b = -x_i m + y_i$
3. Local maxima in array  $H(m, b)$  corresponds to lines

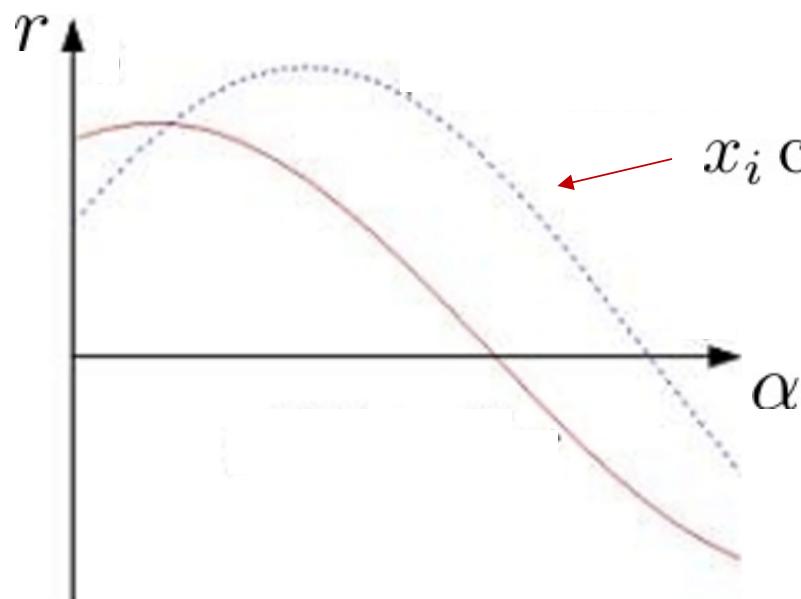


# Hough transform algorithm: polar coordinate representation

- Equation of a line in polar coordinates

$$x \cos \alpha + y \sin \alpha = r$$

- The parameter space transform of a point is a sinusoidal curve



- Avoids infinite slope
- Constant resolution

# Hough transform algorithm, revised

**Data:** Set  $S$  containing  $N$  points

**Result:** Line fitting the points in  $S$

Initialize  $n_\alpha \times n_r$  accumulator  $H$  with zeros;

**foreach**  $(x_i, y_i) \in S$  **do**

**foreach**  $\alpha \in \{\alpha_1, \dots, \alpha_{n_\alpha}\}$  **do**

        compute  $r = x_i \cos \alpha + y_i \sin \alpha$ ;

$H[\alpha, r] \leftarrow H[\alpha, r] + 1$ ;

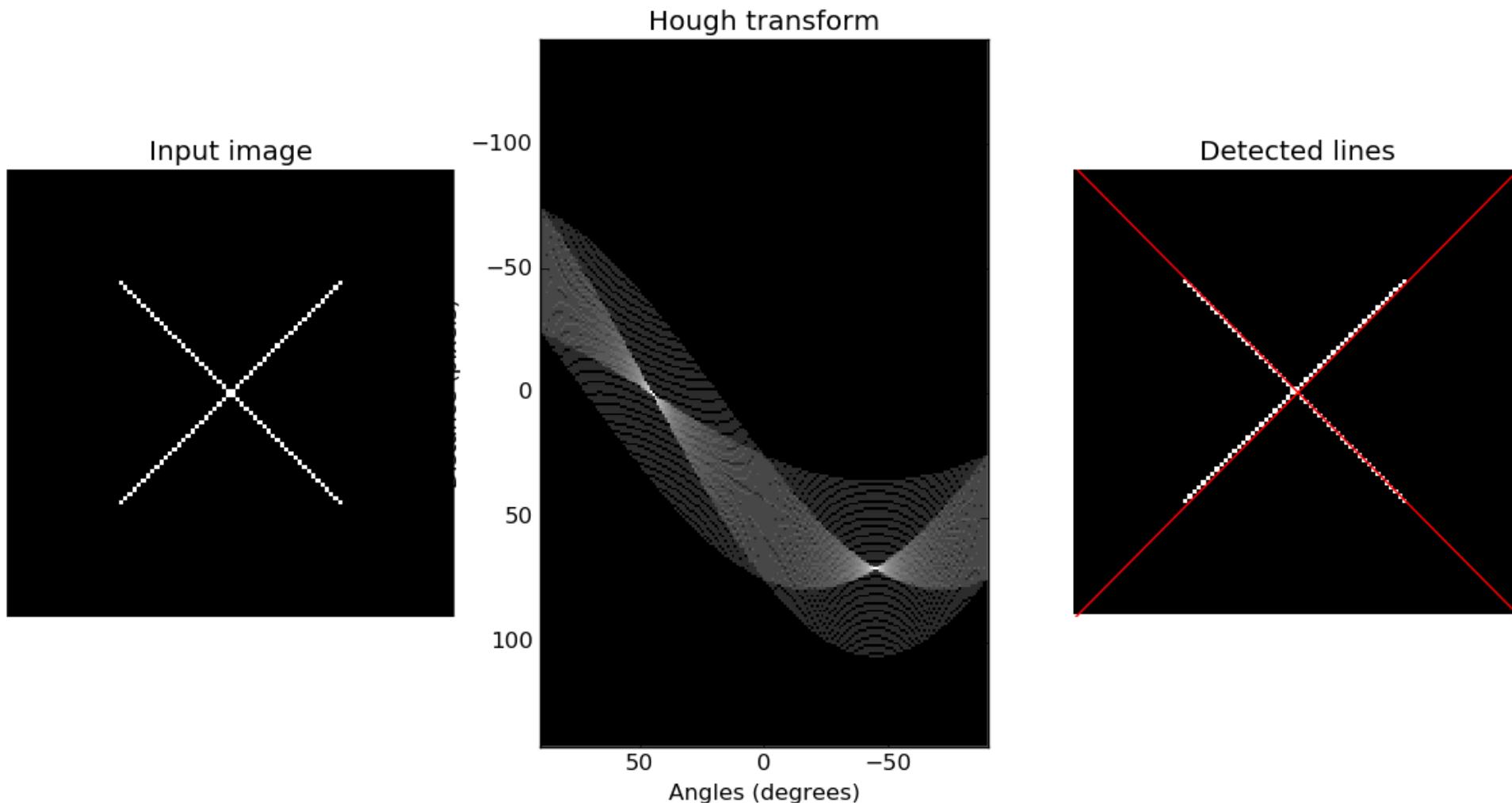
**end**

**end**

Choose  $(\alpha^*, r^*)$  that corresponds to largest count in  $H$ ;

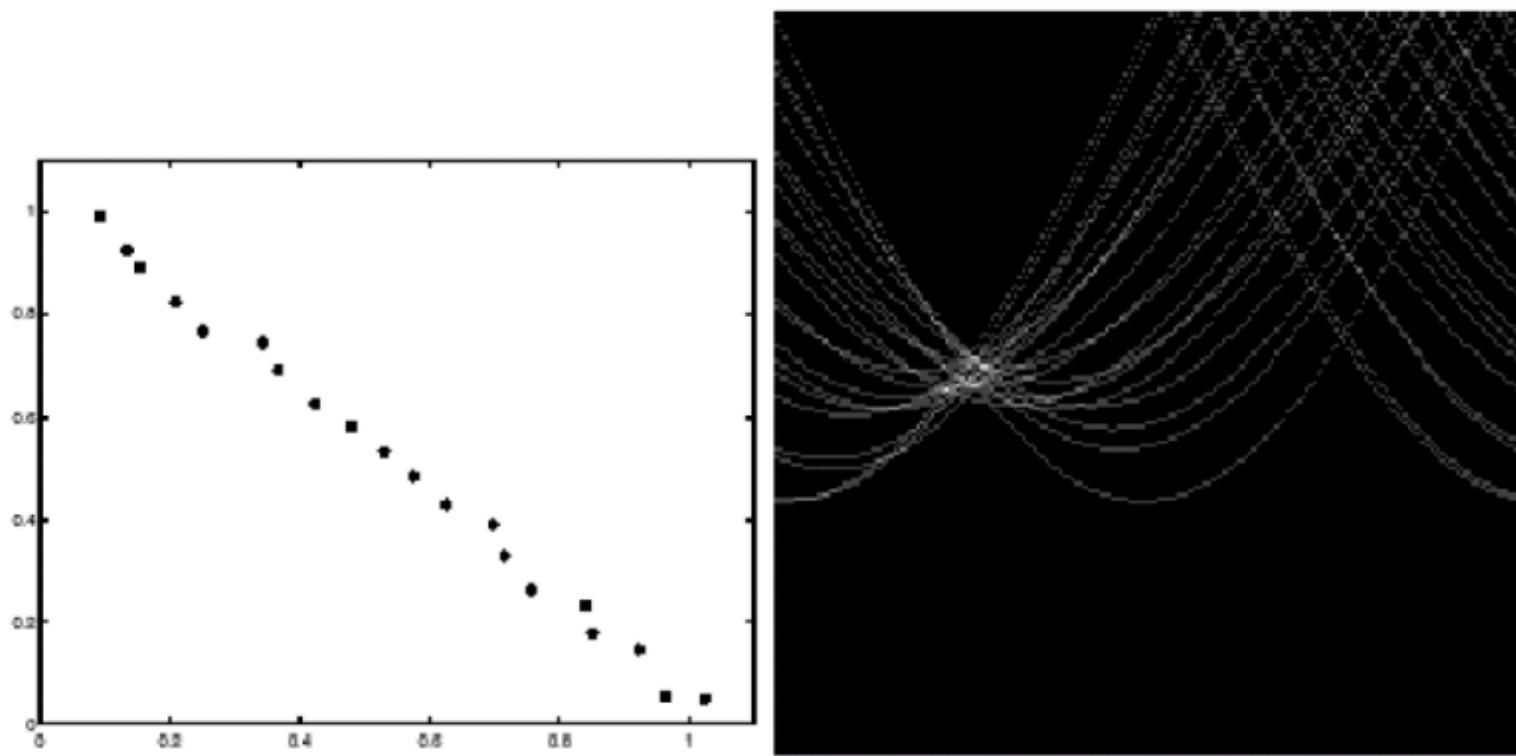
Return line defined by  $(\alpha^*, r^*)$

# Hough transform: example



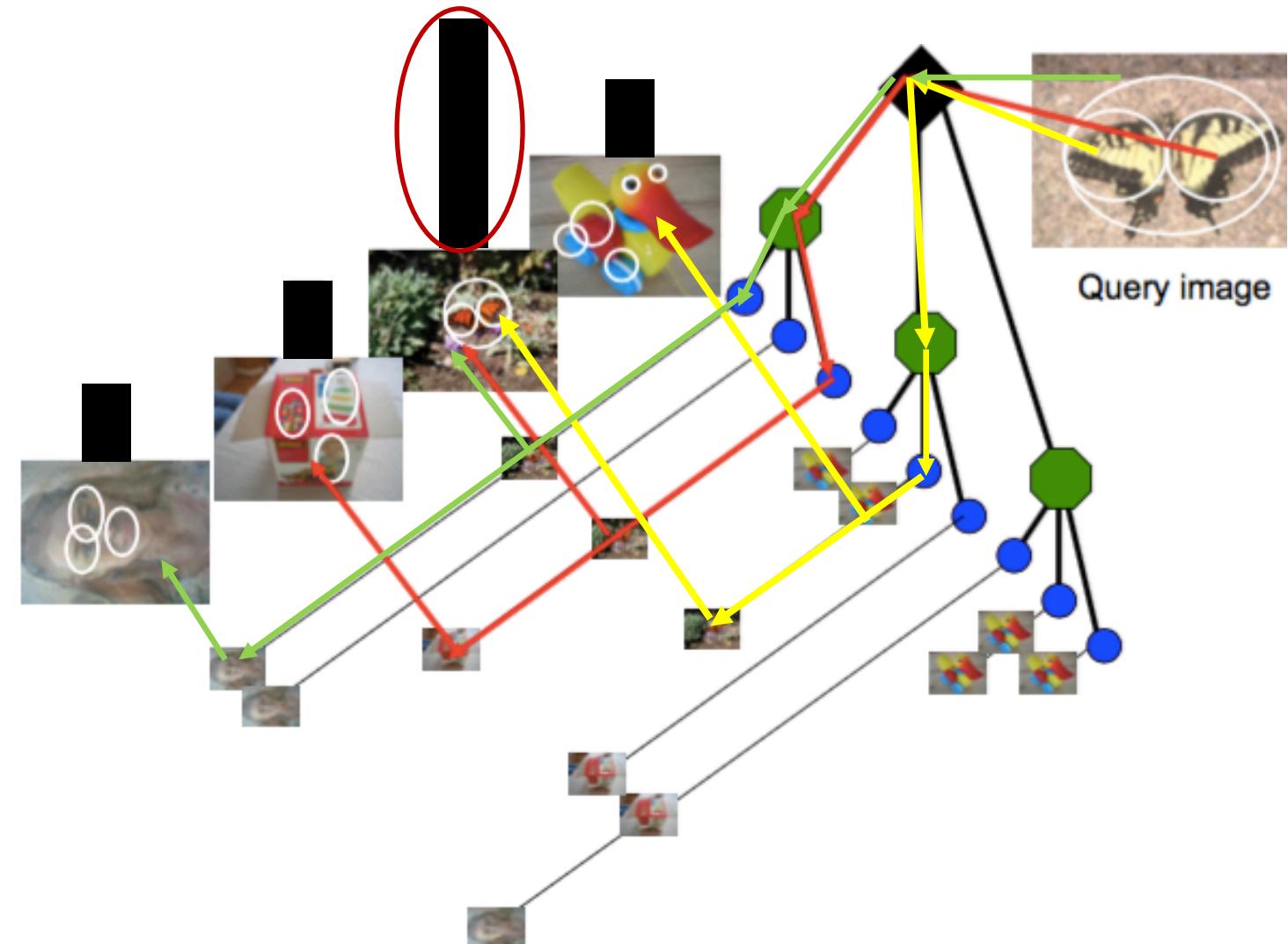
# Hough transform: example

- With noise, peaks may be hard to detect



# Object recognition and scene understanding

- Object recognition:  
capability of naming  
discrete objects in the  
world
- Popular approach: *bag  
of visual words*
- In this class, we will  
focus on neural  
network methods



# Intro to ML

- Aim
  - Present and motivate modern ML techniques for vision applications
- Readings
  - CS231n (CNNs for Visual Recognition) class notes: <http://cs231n.github.io/> [vast majority of figures/slides borrowed from these notes]

# Let's zoom out

- What is a convolutional neural network (CNN)?
- What is a “deep” neural network?
- What is a neural network?
- What is machine learning?

# Itinerary

- Stats/ML review
- Neural network basics
- Convolutional neural networks
- Robotic applications

# Itinerary

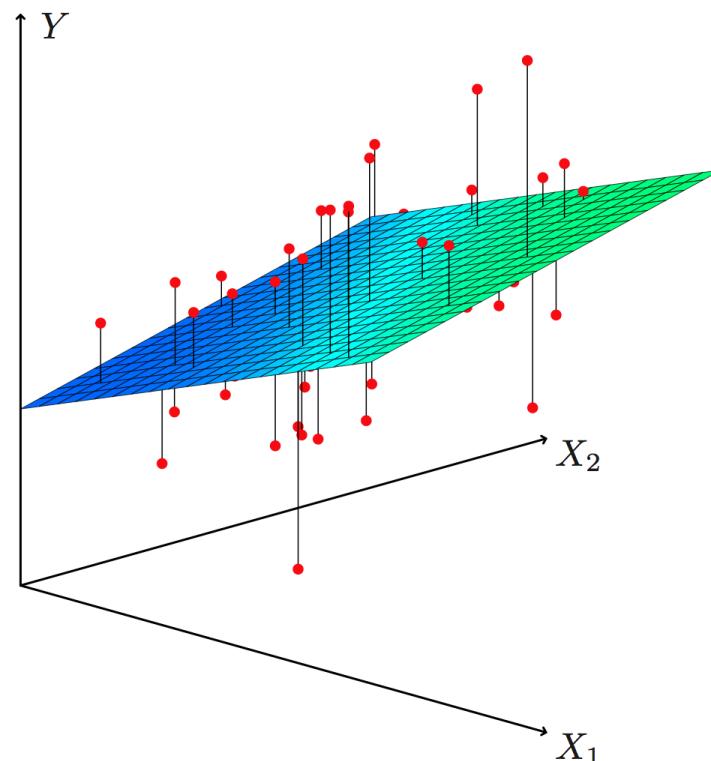
- Stats/ML review
- Neural network basics
- Convolutional neural networks
- Robotic applications

# Machine learning

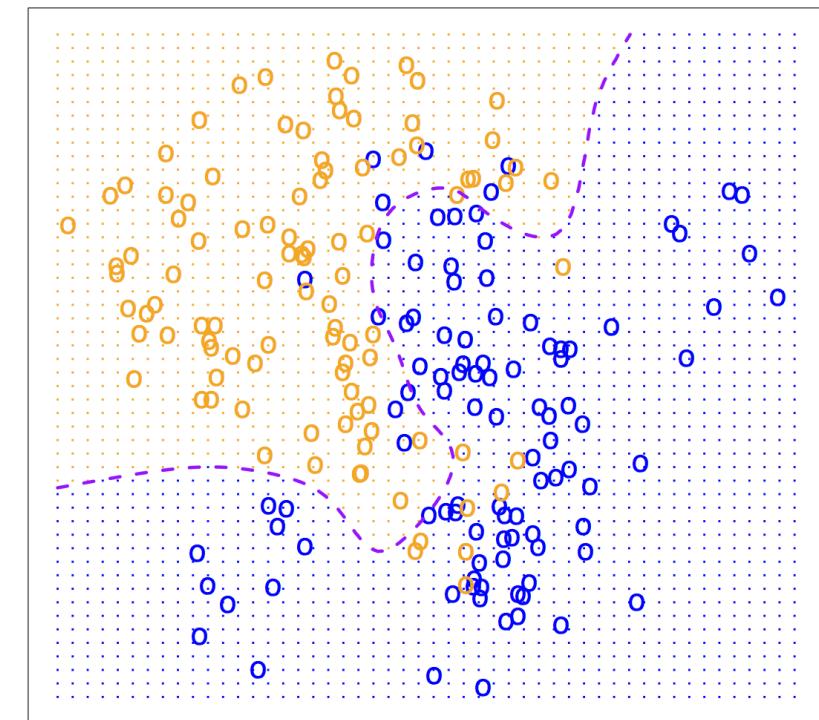
- Supervised learning (classification, regression)
  - Given  $(x^1, y^1), \dots, (x^n, y^n)$  choose a function  $f(x) = y$   
 $x_i$  = data point  
 $y_i$  = class/value
- Unsupervised learning (clustering, dimensionality reduction)
  - Given  $(x^1, x^2, \dots, x^n)$  find patterns in the data

# Supervised learning

- Regression



- Classification



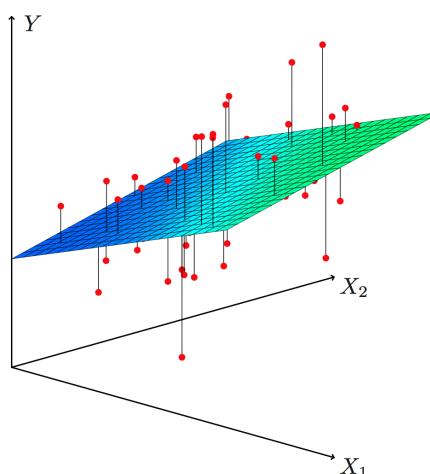
# Loss functions

In selecting  $f(x) \approx y$  we need a quality metric, i.e., a loss function to minimize

- Regression

$$\ell^2 \text{ loss} : \sum_i |f(x^i) - y^i|^2$$

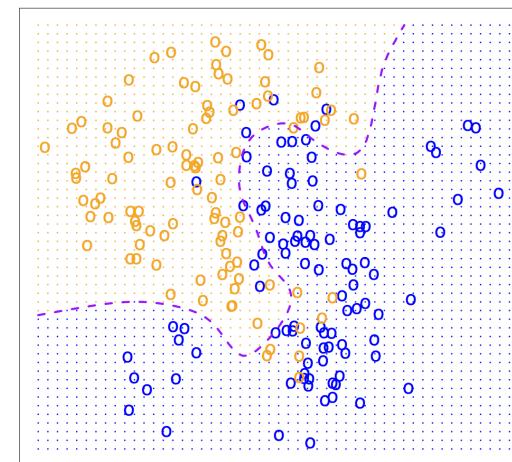
$$\ell^1 \text{ loss} : \sum_i |f(x^i) - y^i|$$



- Classification

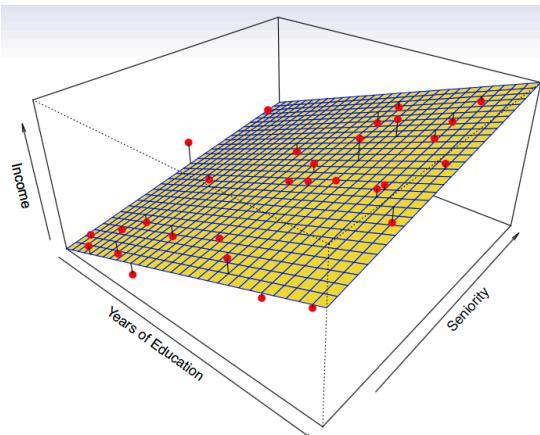
$$0 - 1 \text{ loss} : \sum_i \mathbf{1}\{f(x^i) \neq y^i\}$$

$$\text{Cross entropy loss} : - \sum_i (y^i)^T \log(f(x^i))$$



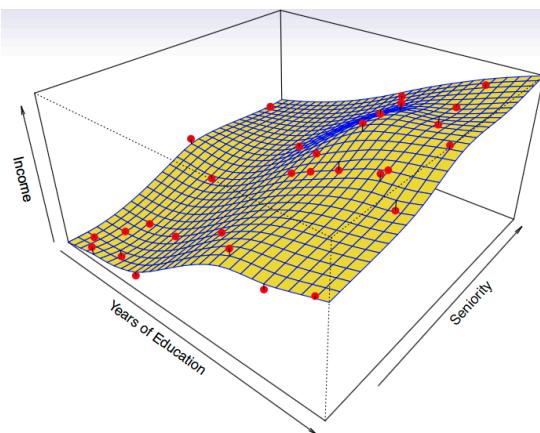
# Learning models

Parametric  
models

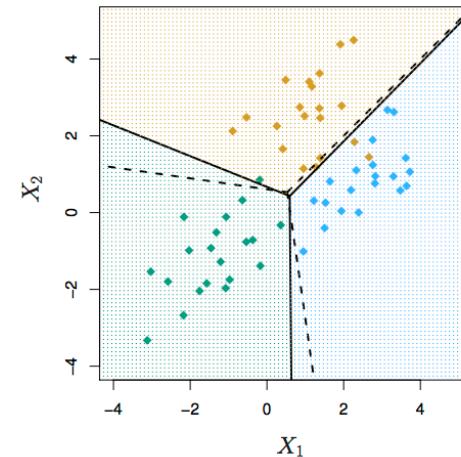


Linear regression

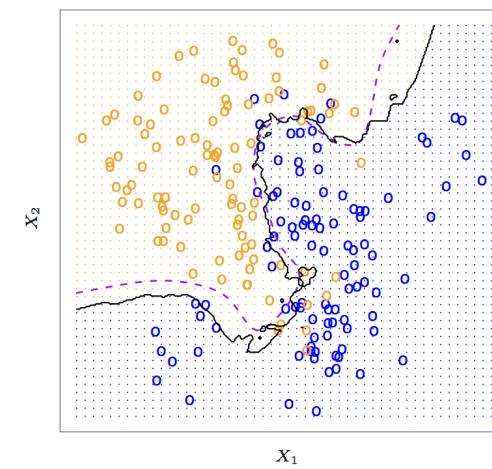
Non-parametric  
models



Spline fitting



Linear classifier



k-Nearest Neighbors

# Machine learning as optimization

We'll focus on parametric models today – how can we choose the best (loss minimizing) parameters to fit our training data?\*

Analytical solution

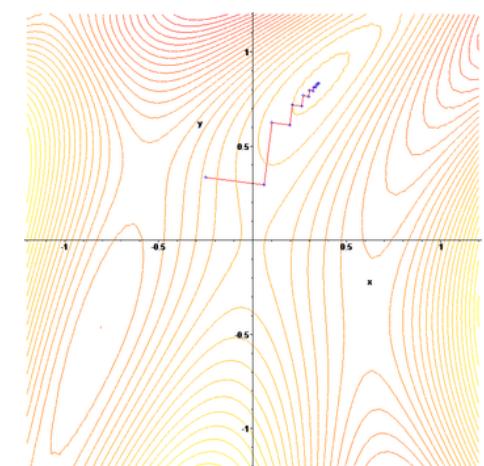
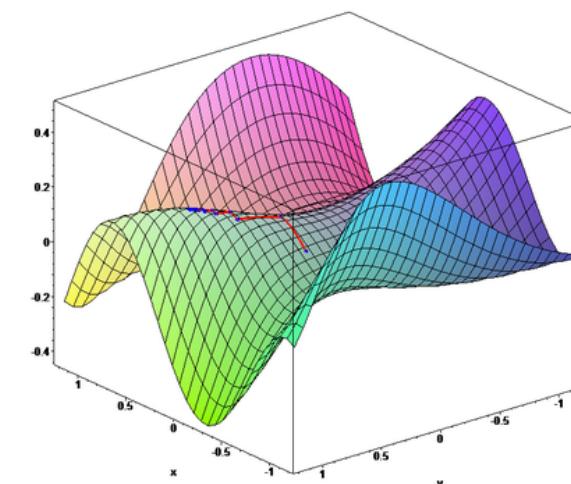
$$\begin{bmatrix} y_1^1 & y_2^1 \\ y_1^2 & y_2^2 \\ \vdots & \\ y_1^n & y_2^n \end{bmatrix} \approx \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_k^1 \\ x_1^2 & x_2^2 & \cdots & x_k^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^n & x_2^n & \cdots & x_k^n \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} \\ a_{11} & a_{12} \\ \vdots & \\ a_{k1} & a_{k2} \end{bmatrix}$$

$$f_A(x) = xA, \quad \ell^2 \text{ loss}$$

$$\hat{A} = (X^T X)^{-1} X^T Y$$

(example: linear least squares)

Numerical optimization



(example: gradient descent)

\* we'll come back to worrying about test data

# Stochastic optimization (quick aside)

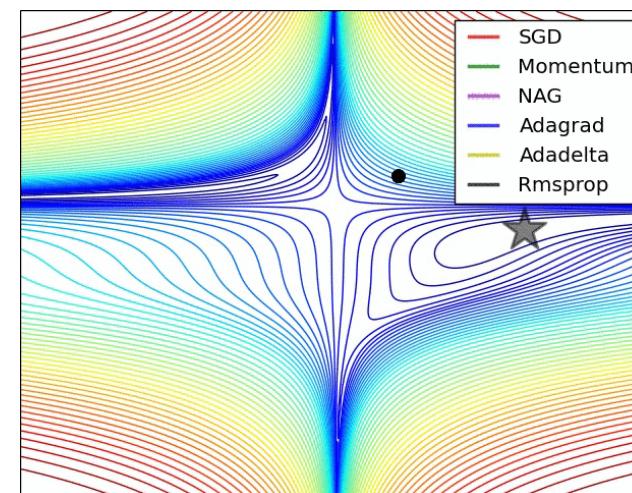
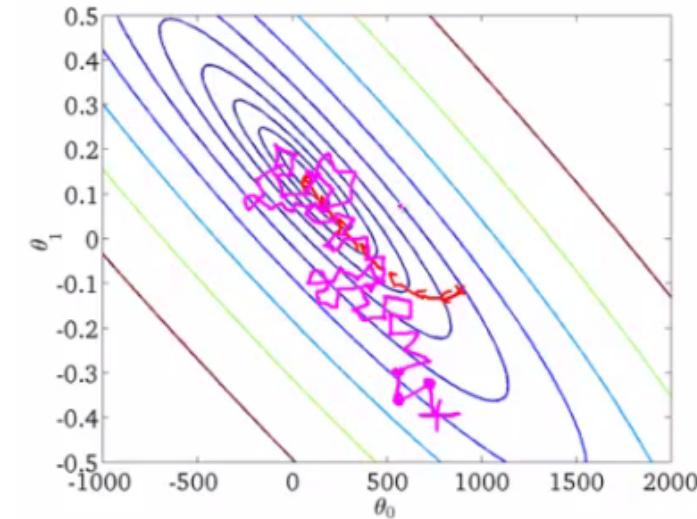
Our loss function is defined over the entire training dataset:

$$L = \frac{1}{n} \sum_{i=1}^n |f(x^i) - y^i|^2 = \frac{1}{n} \sum_{i=1}^n L_i$$

Computing  $\nabla L$  could be very computationally intensive. We approximate:

$$\nabla L \approx \frac{1}{|S|} \sum_{i \in S \subset \{1, \dots, n\}} \nabla L_i$$

Stochastic  
gradient  
descent



Other  
variants

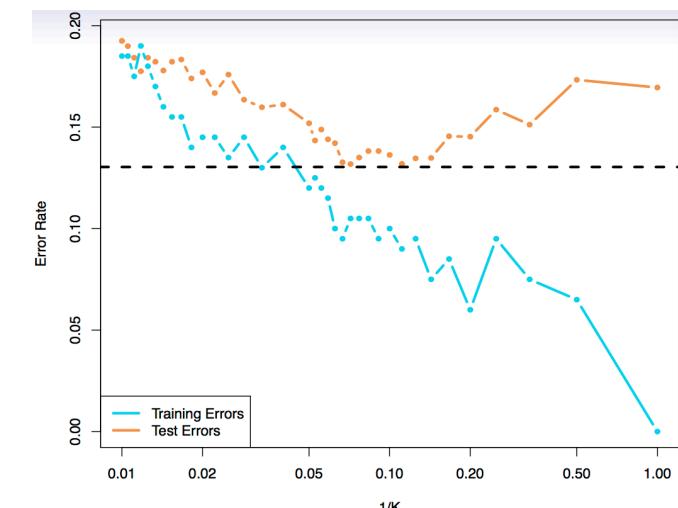
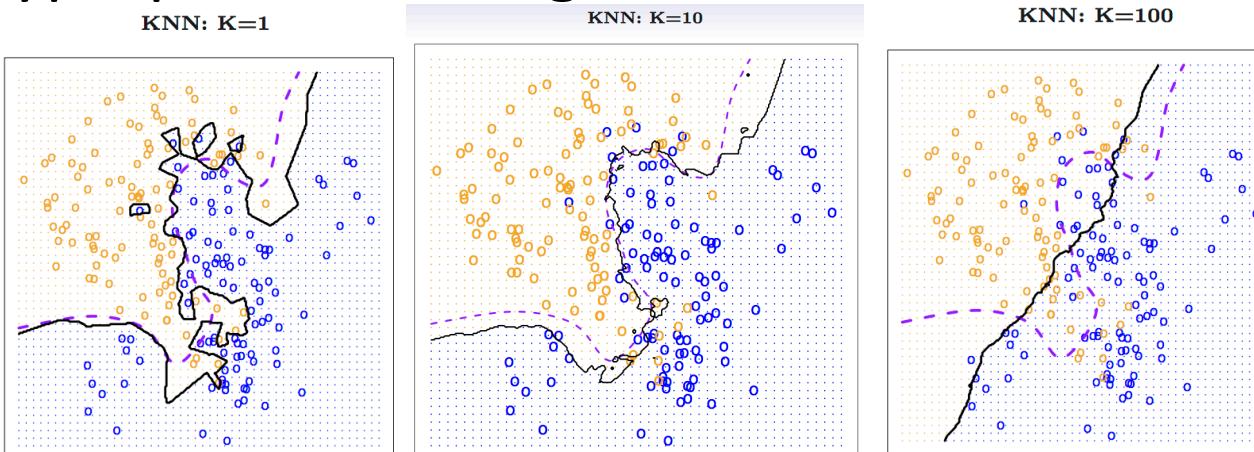
# Regularization

To avoid overfitting on the training data, we may add additional terms to the loss function to penalize “model complexity.”

$\ell^2$  regularization:  $\|A\|_2$   
often corresponds to a Gaussian prior on parameters A

$\ell^1$  regularization:  $\|A\|_1$   
often encourages sparsity in A (easier to interpret/explain)

Hyperparameter regularization:



# Linear classifiers



[32x32x3]  
array of numbers 0...1

$$f(x, W) = \boxed{W} \boxed{x}$$

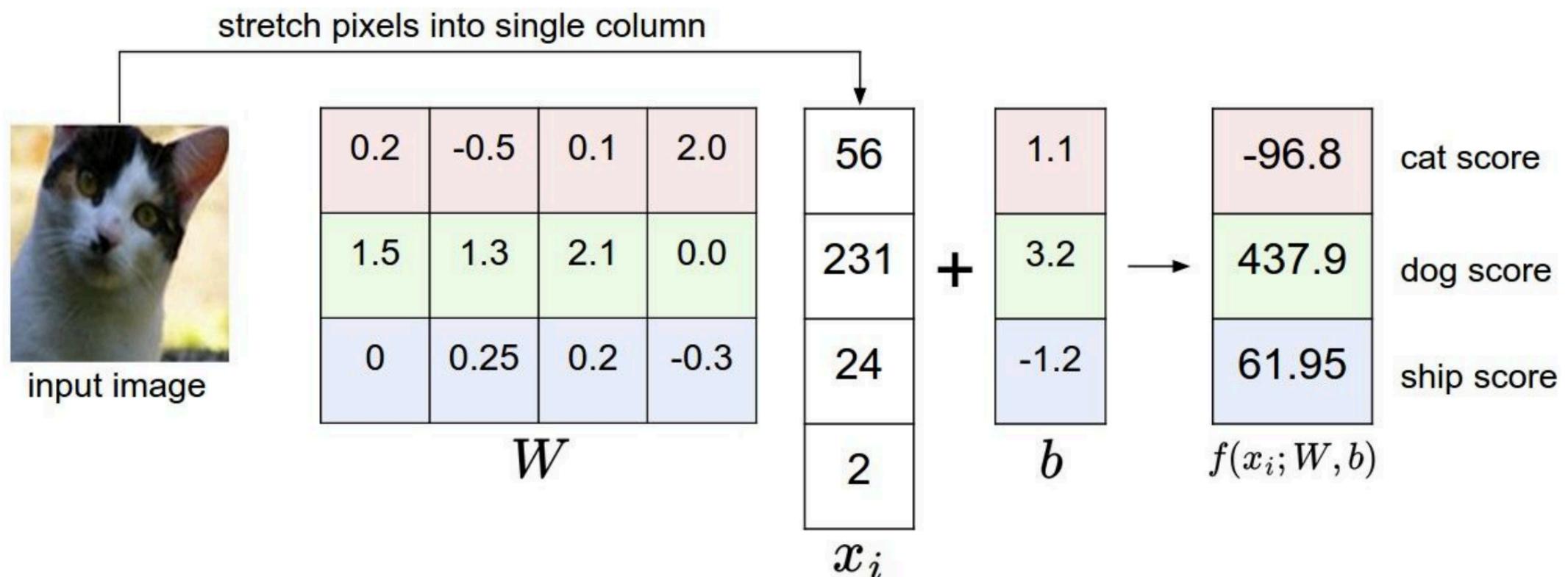
10x1                    10x3072

3072x1  
(+b) 10x1  
10 numbers,  
indicating class  
scores

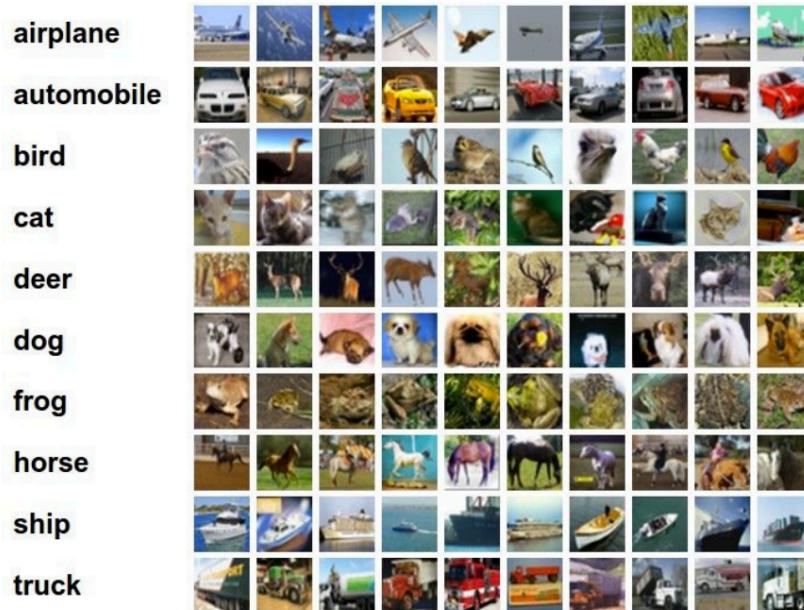
parameters, or “weights”

# Linear classifiers

Example with an image with 4 pixels, and 3 classes (**cat/dog/ship**)



# Linear classifiers – Interpretation



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

Example trained weights  
of a linear classifier  
trained on CIFAR-10:



Each row of  $W$  can be thought of as a “template” for nearest neighbor classification

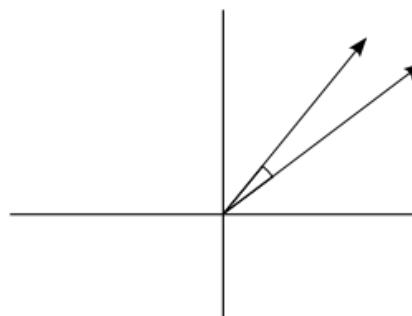
# Dot products as a measure of similarity

$$w^T x = \|w\| \|x\| \cos(\theta)$$

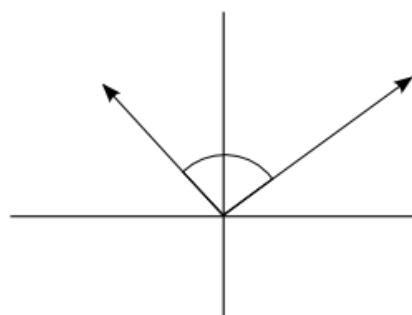
$$\Rightarrow \cos(\theta) = \frac{w^T x}{\|w\| \|x\|}$$

("cosine similarity")

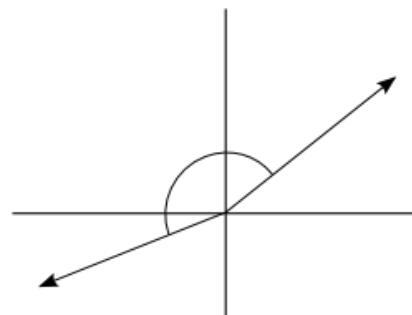
Can also interpret as a projection:  
component of  $x$  along  $w$ ;  
“how much”  $w$  is in  $x$  ...



$$\cos(\theta) \approx 1$$



$$\cos(\theta) \approx 0$$



$$\cos(\theta) \approx -1$$

# Softmax regression

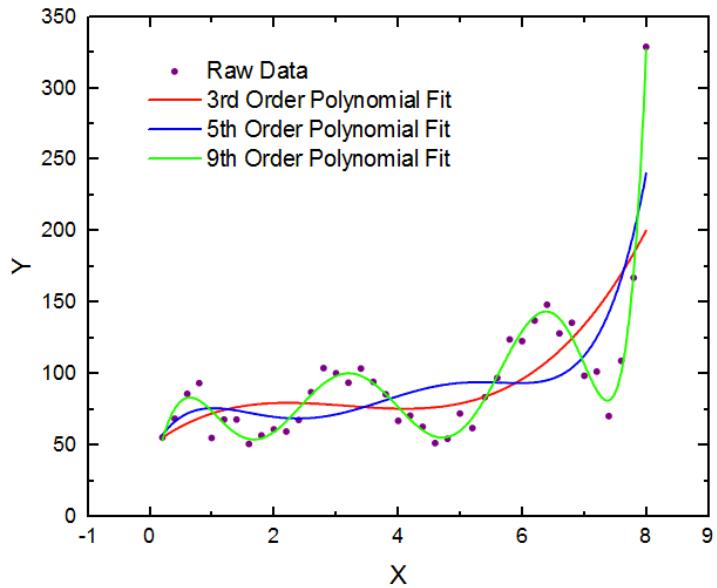
Our class scores can be turned into a probability vector over classes using the softmax function:

$$\sigma(\mathbf{z}) = \begin{bmatrix} \frac{e^{z_1}}{\sum_k e^{z_k}} \\ \vdots \\ \frac{e^{z_m}}{\sum_k e^{z_k}} \end{bmatrix}$$

$$p(y^i = j | x^i) = \frac{e^{x^i W_j + b_j}}{\sum_k e^{x^i W_k + b_k}}$$

Cross-entropy loss computed against “one-hot” class vector [0, ..., 1, ..., 0]

# Generalizing linear models



$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} \approx \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^m \\ 1 & x_2 & x_2^2 & \dots & x_2^m \\ 1 & x_3 & x_3^2 & \dots & x_3^m \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^m \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix}$$

Nonlinearity via basis functions

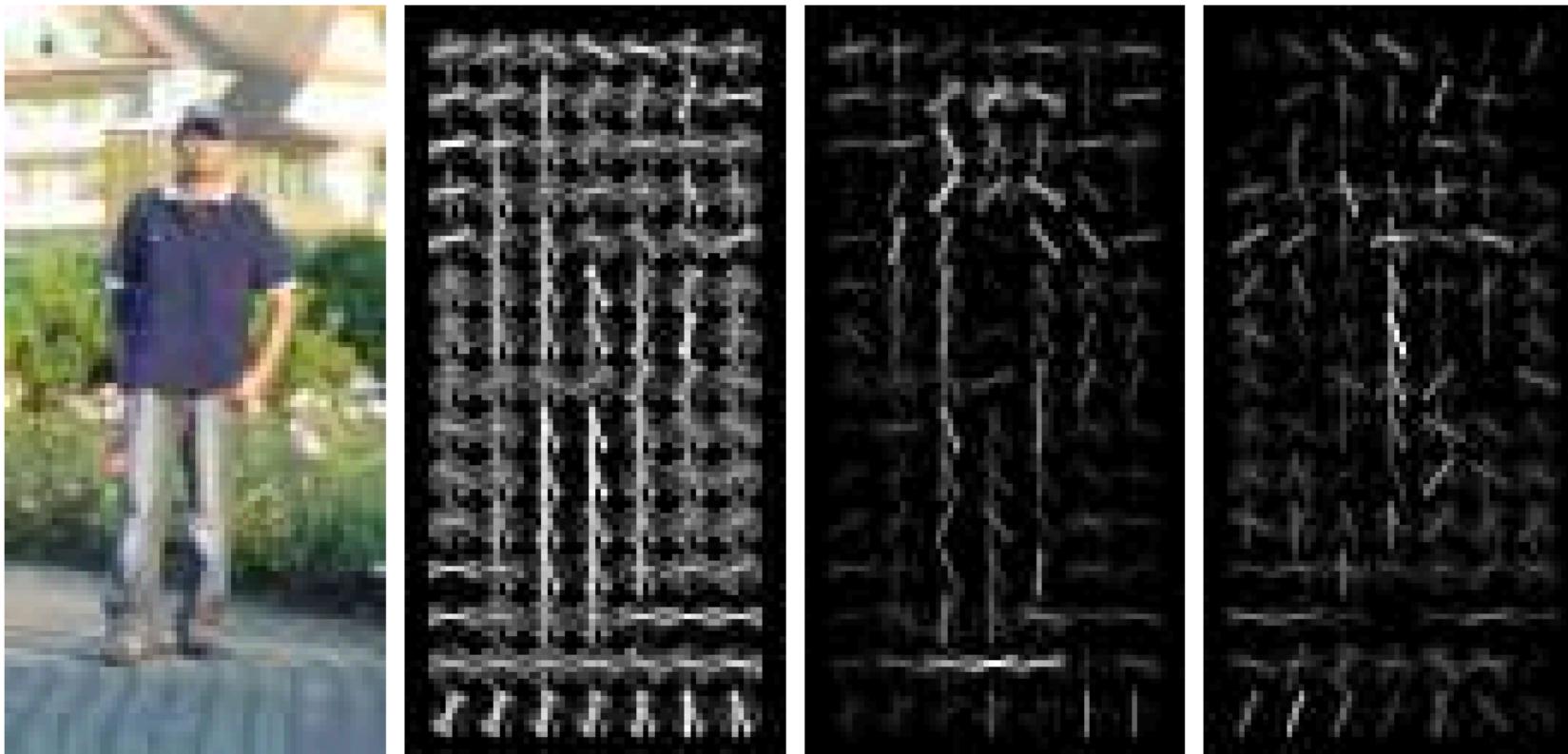
Linear regression/classification  
can be very powerful when  
empowered by the right features



Eigenfaces

# Histogram of Oriented Gradients (HOG)

- Very successful feature set for shape classification/regression



Left to right:

- Test image
- Visualization of HOG descriptor
- HOG descriptor scaled by positive weights
- HOG descriptor scaled by negative weights

(Dalal and Triggs 2005)

# Feature extraction

Human  
Ingenuity

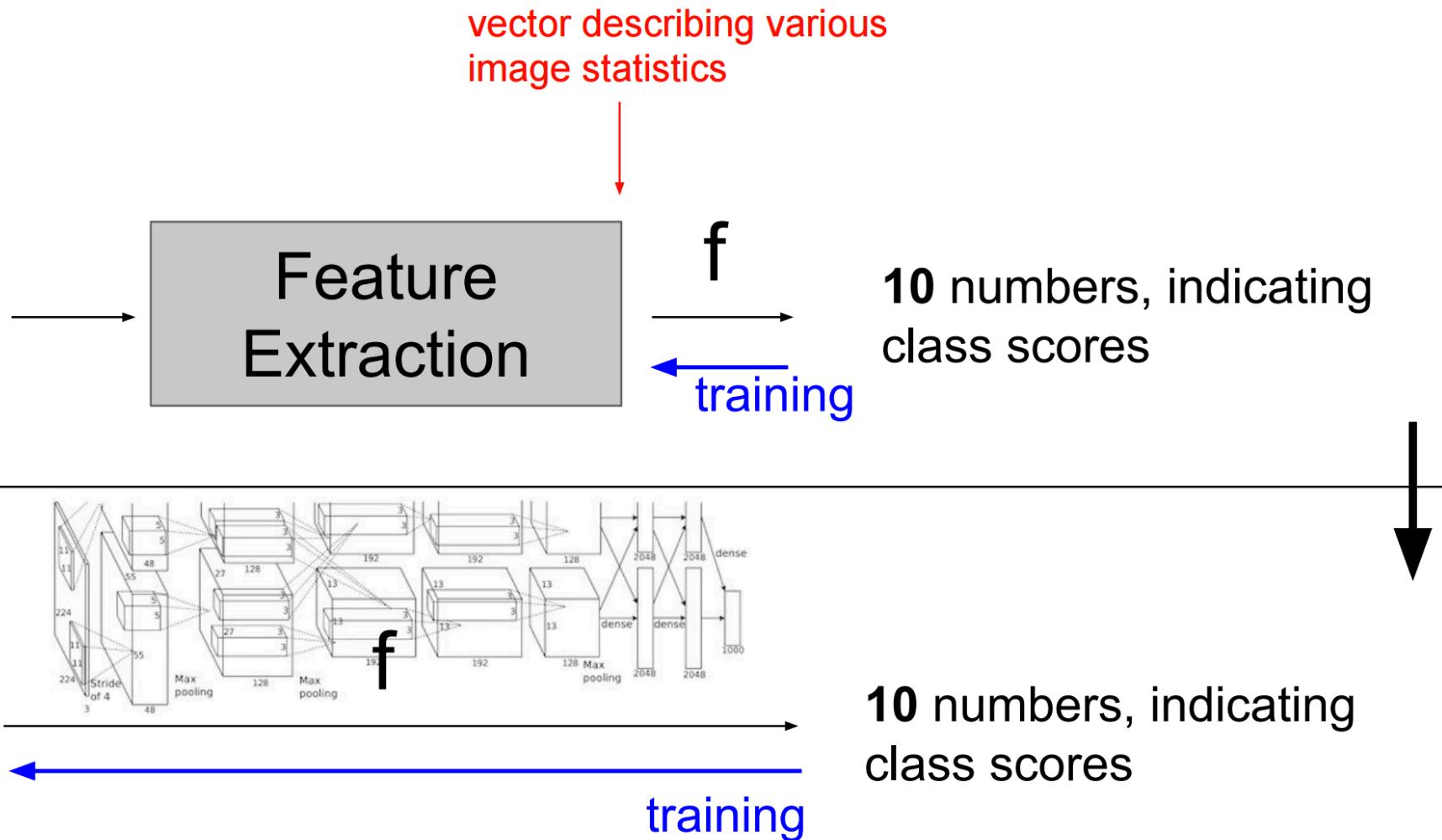


[32x32x3]

Gradient  
Descent



[32x32x3]



# Next time

