# *CS840a*
# *Machine Learning in Computer Vision*
# *Olga Veksler*

## Lecture 1

Introduction

# Outline

- Course overview
- Introduction to Machine Learning

# Course Outline

- Prerequisites
  - Calculus, Statistics, Linear Algebra
  - Some Computer Vision/Image Processing
- Grading
  - Class participation: 10%
  - Four assignments (Matlab): 20%
    - Each assignment is worth 5% of the course mark
    - Assignment grades are 0%, 20%, 40%, 60%, 80%, 100%
  - In class paper presentation 20%
  - Final project: 50%
    - Final Project Presentation 20%
    - Written project report + code, 30 %
    - Matlab, C/C++, anything else as long as I can run it

# Course Outline: Content

- Course Structure
  - Lecture (2/3 of the time)
  - Paper discussion (1/3 of the time)
- Machine Learning Topics (tentatively)
  - Nearest neighbor
  - Linear and generalized linear classifiers
  - SVM
  - Boosting
  - Neural Networks
- Computer Vision Topics
  - Image features
  - Mostly classification/detection/recognition
    - object, action, etc

# Course Outline: Textbook

- No required textbook, but recommended
  - “Pattern Classification” by R.O. Duda, P.E. Hart and D.G. Stork, second edition
  - “Machine Learning” by Tom M. Mitchell
  - “Pattern Recognition and Machine Learning, by C. Bishop
  - “Machine Learning: a Probabilistic Perspective” by Kevin Patrick Murphy
- Journal/Conference papers

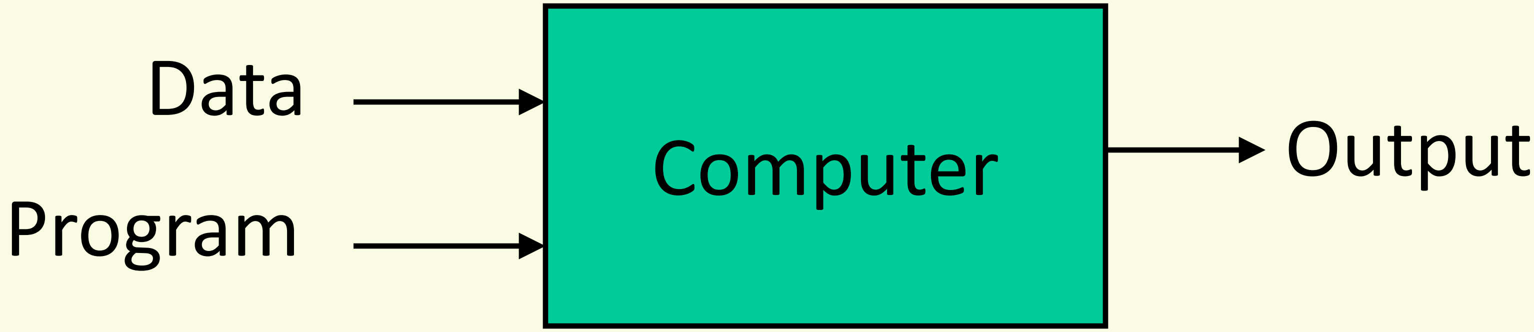
# *Intro: What is Machine Learning?*

- Difficult to come up with explicit program for some tasks
- Classic Example: digit recognition

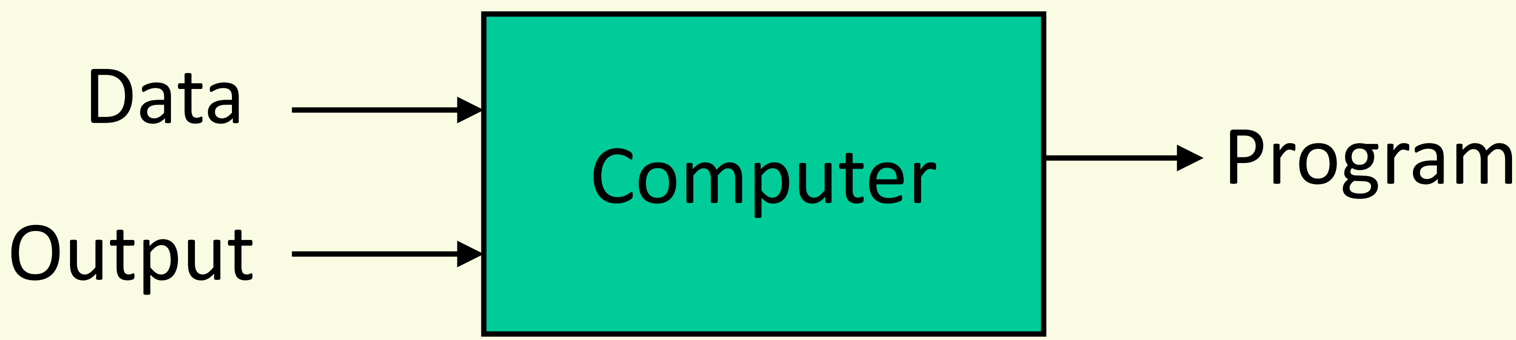
- However, easy to collect images of digits with their correct labels
- Machine Learning Algorithm will take the collected data and produce a program for recognizing digits
  - done right, program will recognize correctly new images it has never seen

# *Intro: What is Machine Learning?*

## Traditional Programming




## Machine Learning



# *Intro: What is Machine Learning?*

- More general definition (Tom Mitchell):
  - Based on experience **E**, improve performance on task **T** as measured by performance measure **P**
- In computer vision
  - **T** is usually classification, **E** is data (images), and **P** is classification error
  - Digit recognition Example
    - **T** = recognize character in the image
    - **P** = percentage of correctly classified images
    - **E** = dataset of human-labeled images of characters

## *Different Types of Machine Learning*

- **Supervised Learning**: given training examples of inputs and corresponding outputs, produce the "correct" outputs for new inputs
- **Unsupervised Learning**: given only inputs as training, find structure in the data
  - e.g. discover "natural" clusters
- **Reinforcement Learning**: not covered in this course

# Supervised Machine Learning

- Training samples (also called examples, feature vectors, etc. )

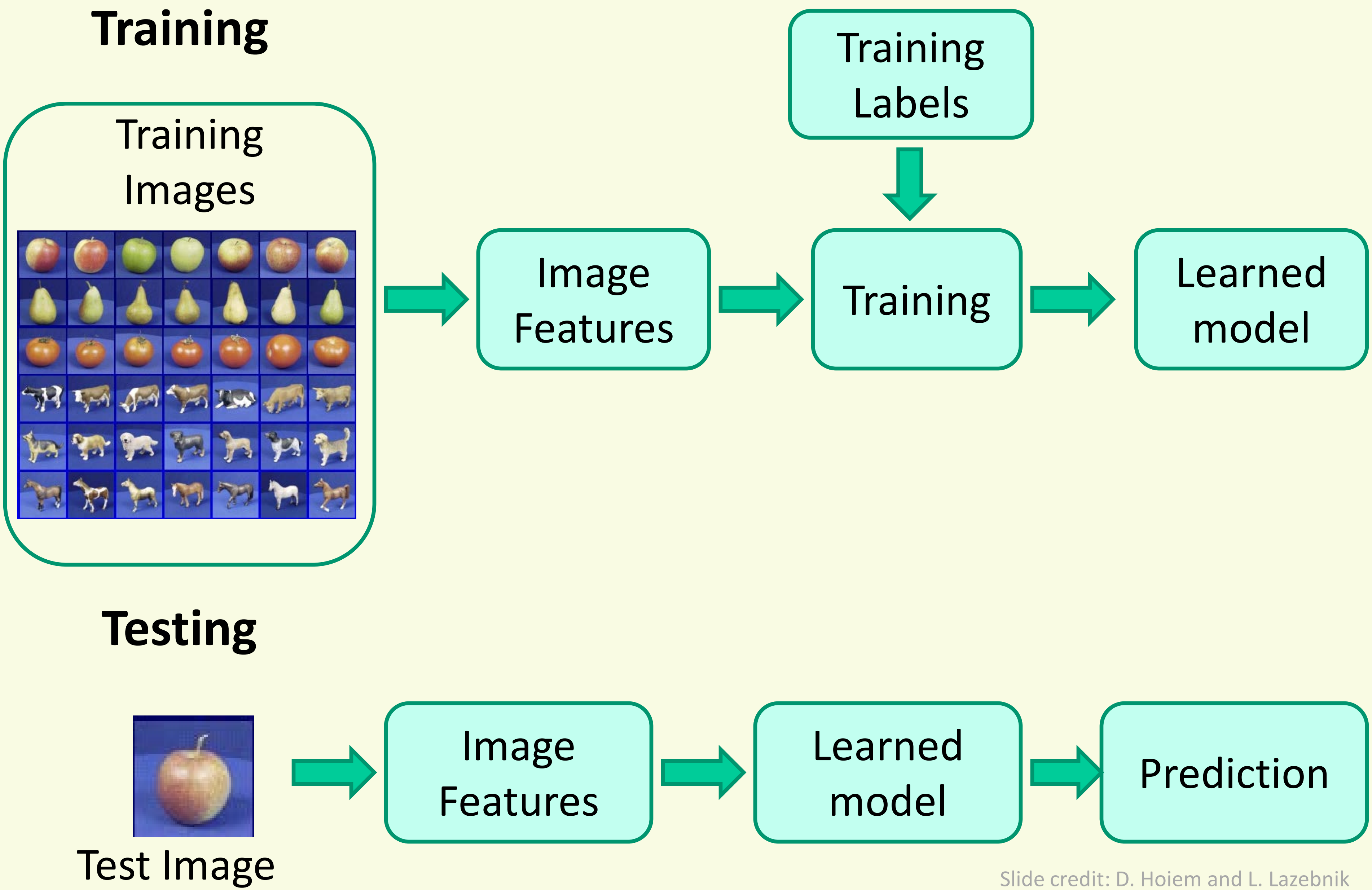
$\mathbf{x}^1 = \begin{bmatrix} 3.3 \\ 5.7 \end{bmatrix}$ $\mathbf{x}^2 = \begin{bmatrix} 6.3 \\ 8.7 \end{bmatrix}$ $\mathbf{x}^3 = \begin{bmatrix} 2.3 \\ 1.7 \end{bmatrix}$ … $\mathbf{x}^n = \begin{bmatrix} 6.4 \\ 7.0 \end{bmatrix}$

*salmon* *sea bass* *salmon* *salmon* *sea bass* *sea bass*

$\mathbf{y}^1=0$ $\mathbf{y}^2=1$ $\mathbf{y}^3=0$ … $\mathbf{y}^n=1$

- Target output (label) for each sample $\mathbf{y}^1$, $\mathbf{y}^2$,…$\mathbf{y}^n$
  - “teacher” gives target outputs
- Training phase: estimate prediction function $\mathbf{y} = \mathbf{f}(\mathbf{x})$ from labeled data
  - $\mathbf{f}$ is also called classifier, learning machine, etc.
- Testing phase: predict label $\mathbf{f}(\mathbf{x})$ for a new (unseen) sample $\mathbf{x}$

# *Training/Testing Phases Illustrated*




Slide credit: D. Hoiem and L. Lazebnik

# *Two Types of Supervised Machine Learning*

- Classification
  - $\mathbf{y}^i$ is finite, typically called a *label* or a *class*
  - example: $\mathbf{y}^i \in$ {baby, child, adult, elder}

- Regression
  - $\mathbf{y}^i$ is continuous, typically called an *output value*
  - Example: $\mathbf{y}^i$ = age $\in [0,130]$

# *More on Training Stage*

- Training stage: estimate prediction function $\mathbf{y} = \mathbf{f}(\mathbf{x})$ from labeled data
- Start with a set of predictor functions or *hypothesis space*
  - hypothesis space $\mathbf{f}(\mathbf{x},\mathbf{w})$ is parameterized by parameters or *weights* $\mathbf{w}$
  - each setting of $\mathbf{w}$ corresponds to a different hypothesis
  - find (or *tune*) weights $\mathbf{w}$ s.t. $f(\mathbf{x}^i,\mathbf{w}) = \mathbf{y}^i$ "as much as possible" for training samples $(\mathbf{x}^i, \mathbf{y}^i)$
    - "as much as possible" needs to be defined
    - usually done by optimization, can be time consuming

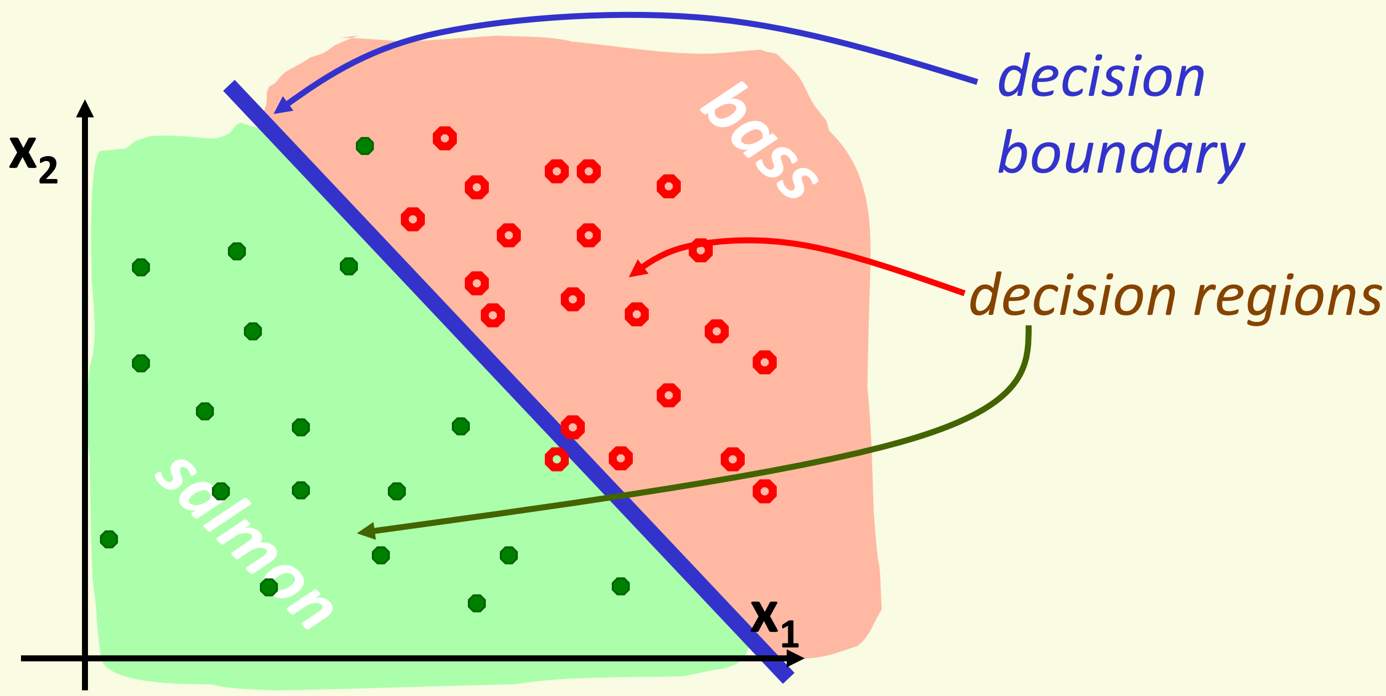
# Training Stage: Linear Classifier

- Linear classifier **f**(**x**,**w**) has a simple functional form
- For 2 class problem

$$\mathbf{f}(\mathbf{x},\mathbf{w}) = \mathrm{sign}(\mathbf{w}^t\mathbf{x}+\mathbf{w_0})$$

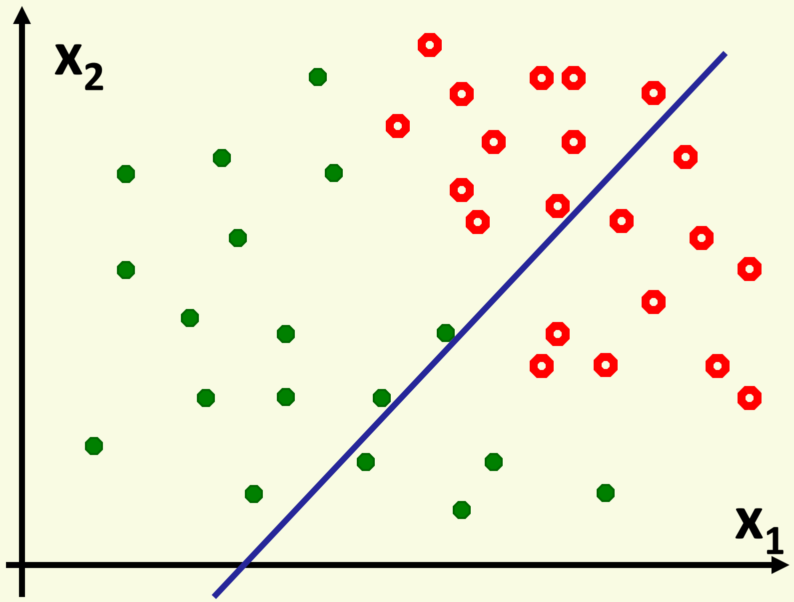
- If samples are 2 dimensional

$$\mathbf{f}(\mathbf{x},\mathbf{w}) = \mathrm{sign}(\mathbf{w_0}+\mathbf{w_1}\mathbf{x_1}+\mathbf{w_2}\mathbf{x_2})$$



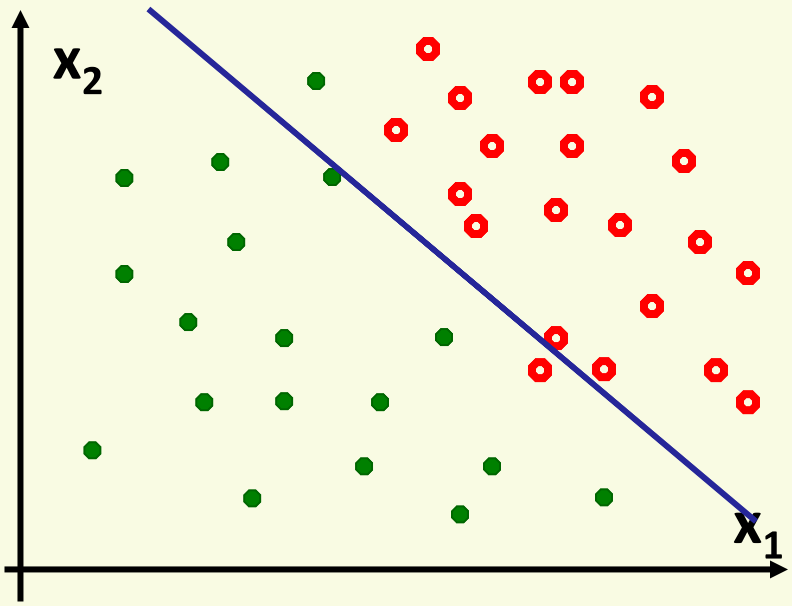
# Training Stage: Linear Classifier

bad setting of **w**



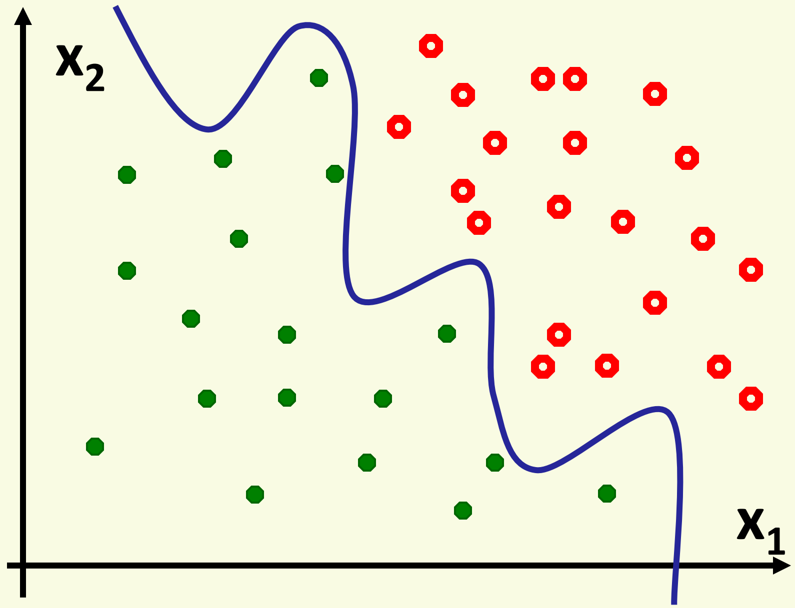

classification error 38%

best setting of **w**




classification error 4%

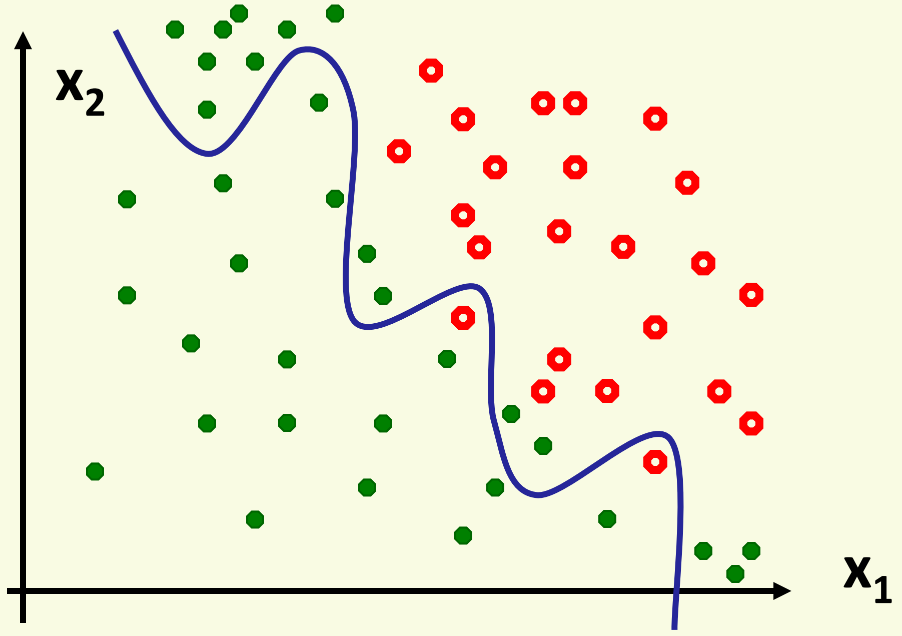
# Training Stage: More Complex Classifier



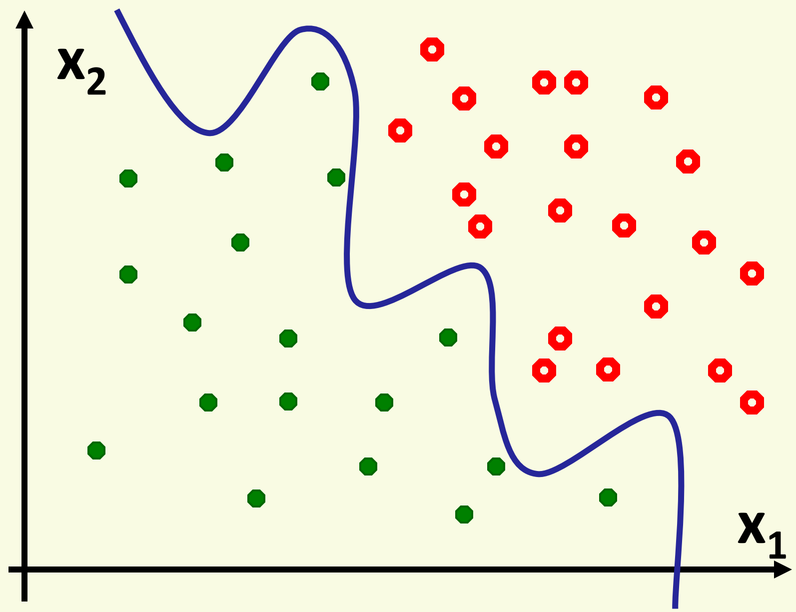

- for example if **f**(**x**) is a polynomial of high degree
- 0% classification error

# Test Classifier on New Data

- The goal is for classifier to perform well on new data
- Test "wiggly" classifier on new data: 25% error



# *Overfitting*




- Have only a limited amount of data for training
- Overfitting:
  - Complex model may have too many parameters to fit reliably with a limited amount of training data
  - Complex model may adapt too closely to the random “noise” of the training data

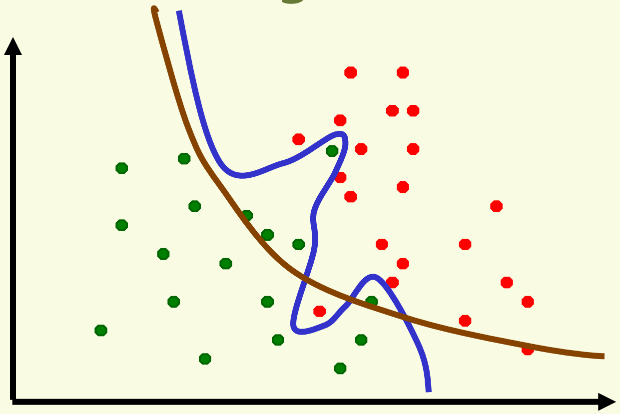
# Overfitting: Extreme Example

- 2 class problem: face and non-face images
- Memorize (i.e. store) all the "face" images
- For a new image, see if it is one of the stored faces
  - if yes, output "face" as the classification result
  - If no, output "non-face"
  - also called "rote learning"
- **problem**: new "face" images are different from stored "face" examples
  - zero error on stored data, 50% error on test (new) data
  - decision boundary is very unsmooth
- Rote learning is memorization without generalization

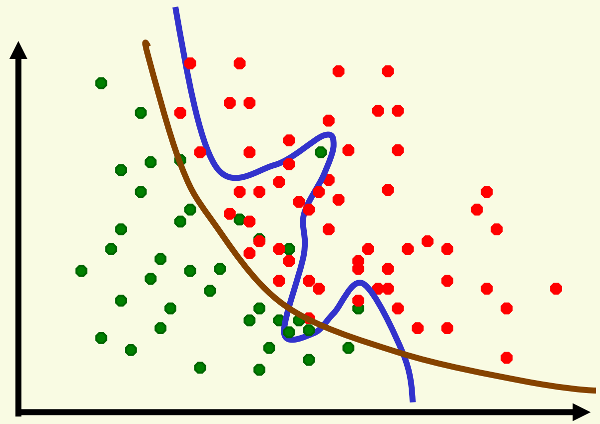
*slide is modified from Y. LeCun*

# Generalization

training data



new data



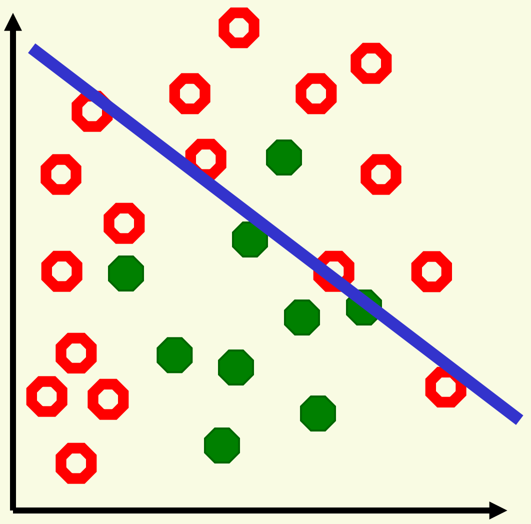
- The ability to produce correct outputs on previously unseen examples is called **generalization**
- Big question of learning theory: how to get good generalization with a limited number of examples
- Intuitive idea: favor simpler classifiers
  - William of Occam (1284-1347): “entities are not to be multiplied without necessity”
- Simpler decision boundary may not fit ideally to the training data but tends to generalize better to new data

# *Training and Testing*

- How to diagnose overfitting?
- Divide all labeled samples $\mathbf{x}^1, \mathbf{x}^2, \ldots \mathbf{x}^n$ into *training* set and *test* set
- There are 2 phases, training and testing
  - Training phase is for "teaching" machine
    - tuning weights **w**
    - classification error on the training data is called training error
  - Testing phase is for evaluating how well machine works on unseen examples
    - classification error on the test data is called test error

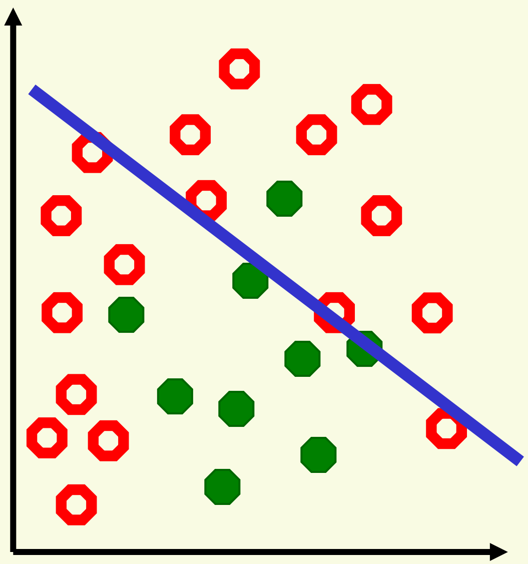
# *Underfitting*

- Can also underfit data, i.e. too simple decision boundary
  - chosen model is not expressive enough
- No linear decision boundary can well separate the samples
- Training error is too high
  - test error is, of course, also high



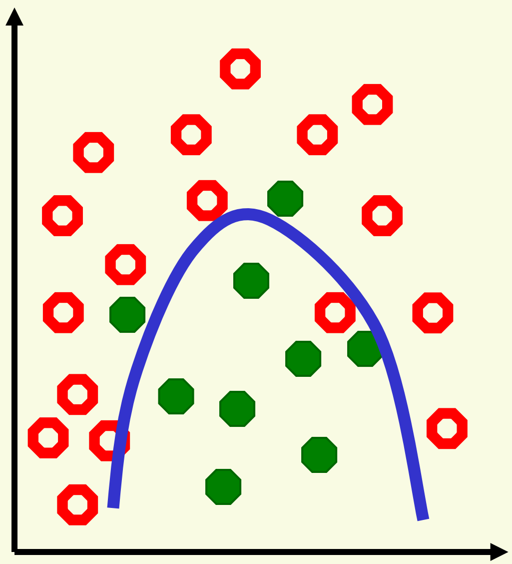
# *Underfitting → Overfitting*

## *underfitting*



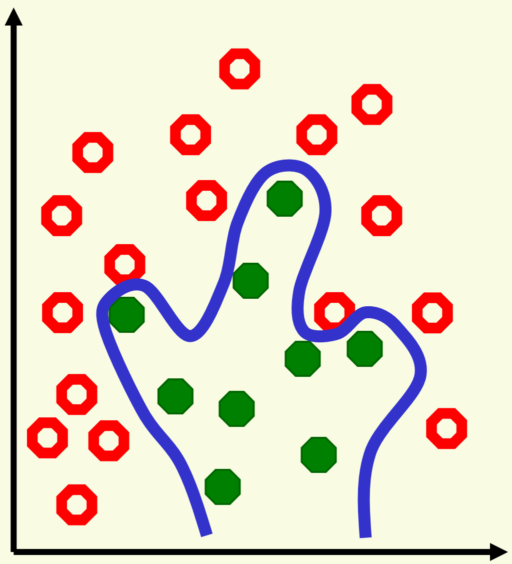
- high training error
- high test error

## *“just right”*



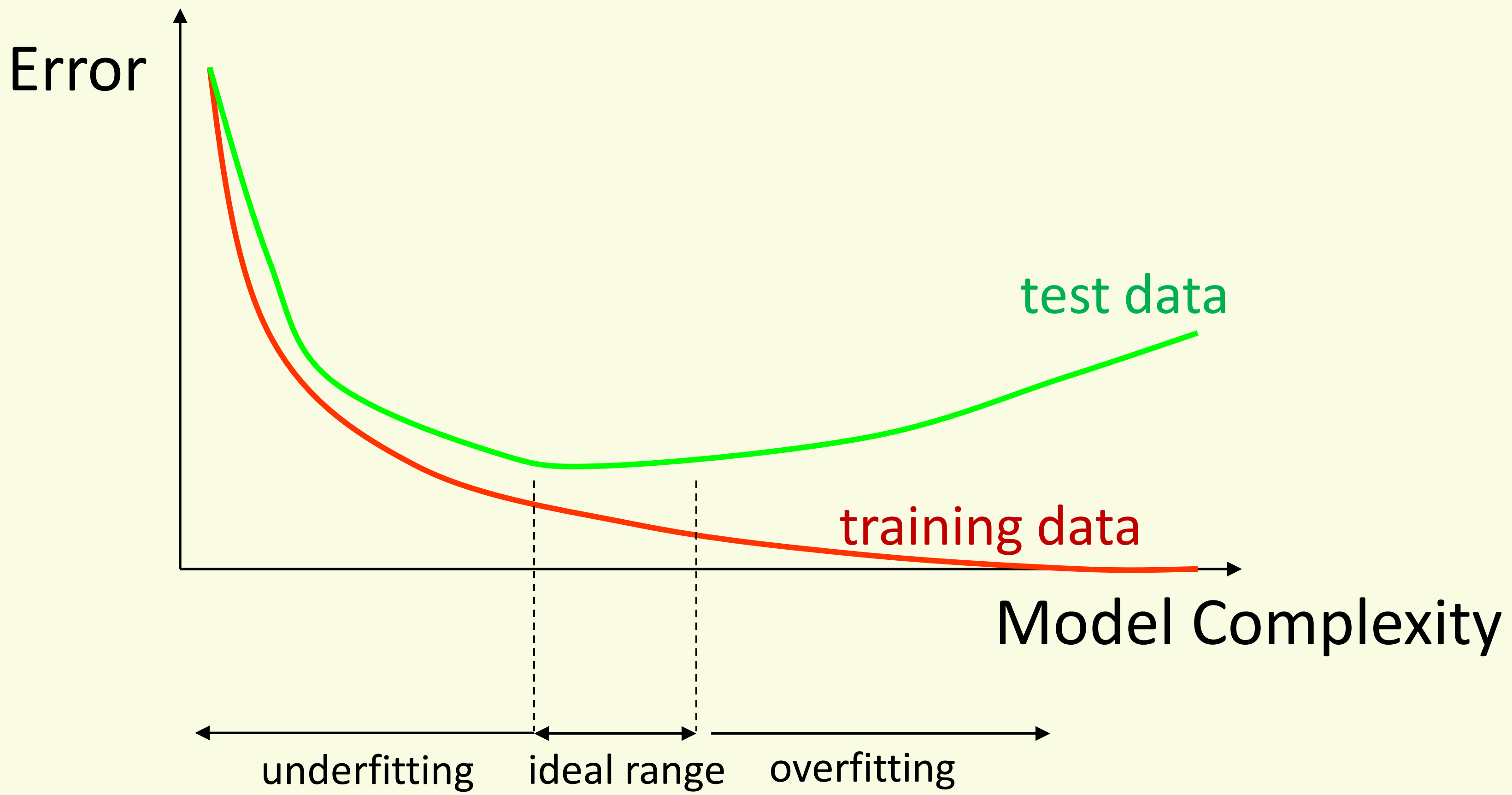
- low training error
- low test error

## *overfitting*



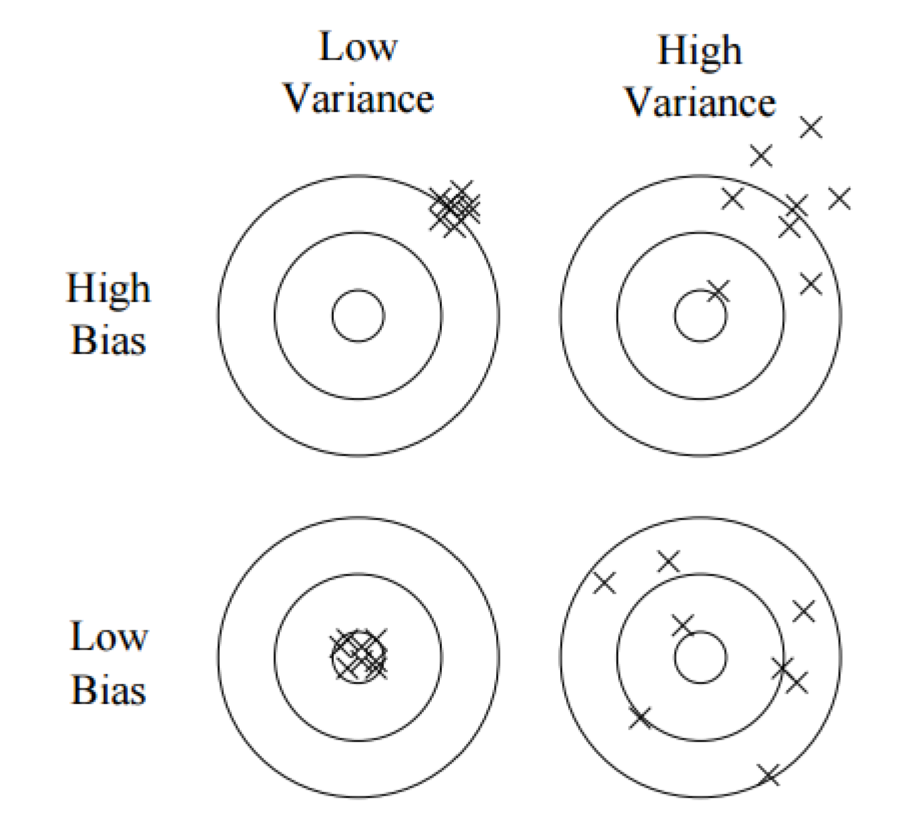
- low training error
- high test error

# *How Overfitting affects Prediction*



# Bias/Variance

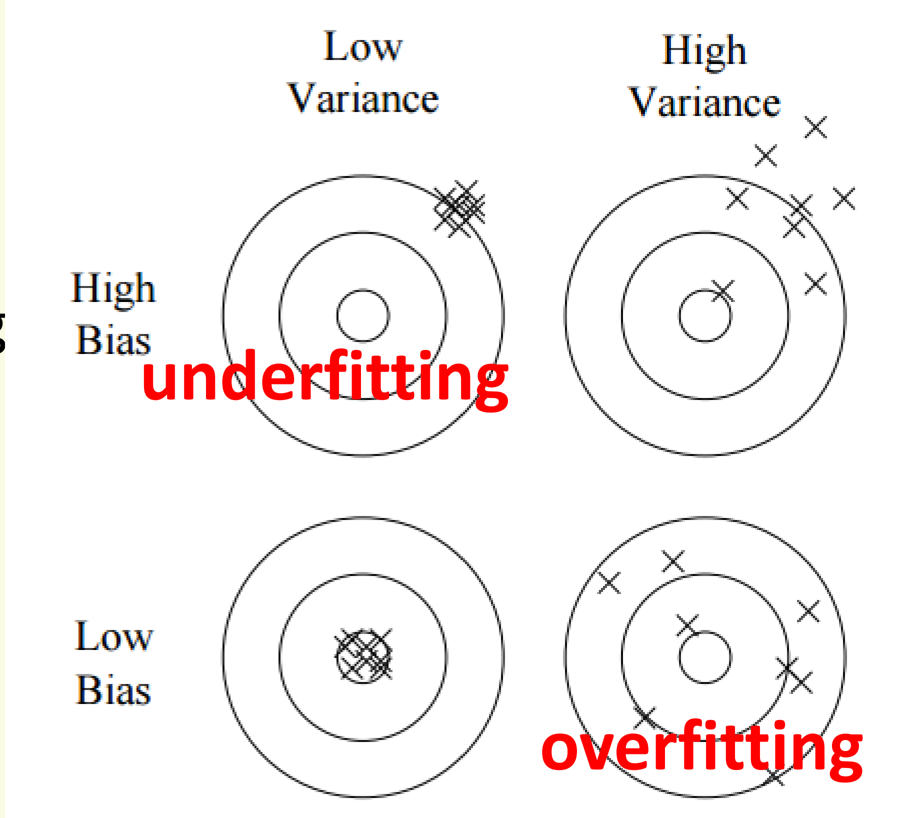
- High bias, informally, is the tendency to consistently learn the same wrong thing on different sets of training data
- High variance, informally, is the tendency to learn the wrong thing irrespective of the training data
- Dart throwing illustration




slide credit Pedro Domingos

# More on Overfitting/Underfitting

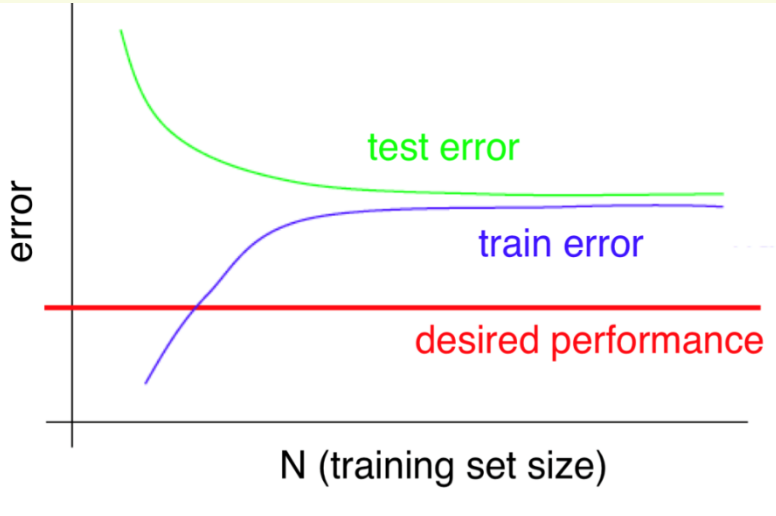
- Underfitting
  - fitted model has large deviation from true values
  - but different sets of training data give models that are similar
- Overfitting
  - fitted model has small deviation from true values
  - different sets of training data give models that are not similar



# *Learning Curve*

- To diagnose overfitting/underfitting, useful to look at training/test error vs. number of samples called *learning curve*

**underfitting**




**overfitting**


*slide is modified from Andrew Ng*

# *Fixing Underfitting/Overfitting*

- Underfitting
  - add more features (if underfitting)
  - use more complex **f**(**x**,**w**)
- Overfitting
  - remove features
  - collect more training data
  - use less complex **f**(**x**,**w**)

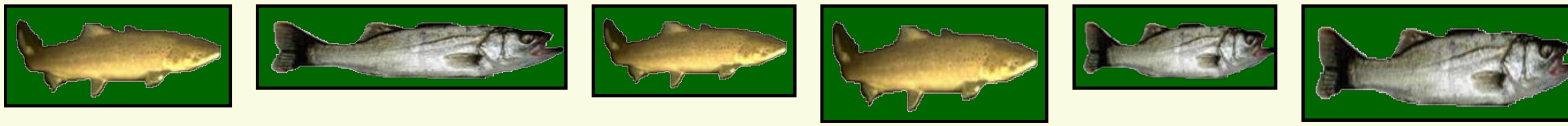
# *Sketch of Supervised Machine Learning*

- Chose a hypothesis space **f**(**x**,**w**)
  - **w** are tunable weights
  - **x** is the input sample
  - tune **w** so that **f**(**x**,**w**) gives the correct label for training samples **x**
- Which hypothesis space **f**(**x**,**w**) to choose?
  - has to be expressive enough to model our problem well, i.e. to avoid ***underfitting***
  - yet not to complicated to avoid ***overfitting***

# *Classification System Design Overview*

- Collect and label data by hand

*salmon* *sea bass* *salmon* *salmon* *sea bass* *sea bass*



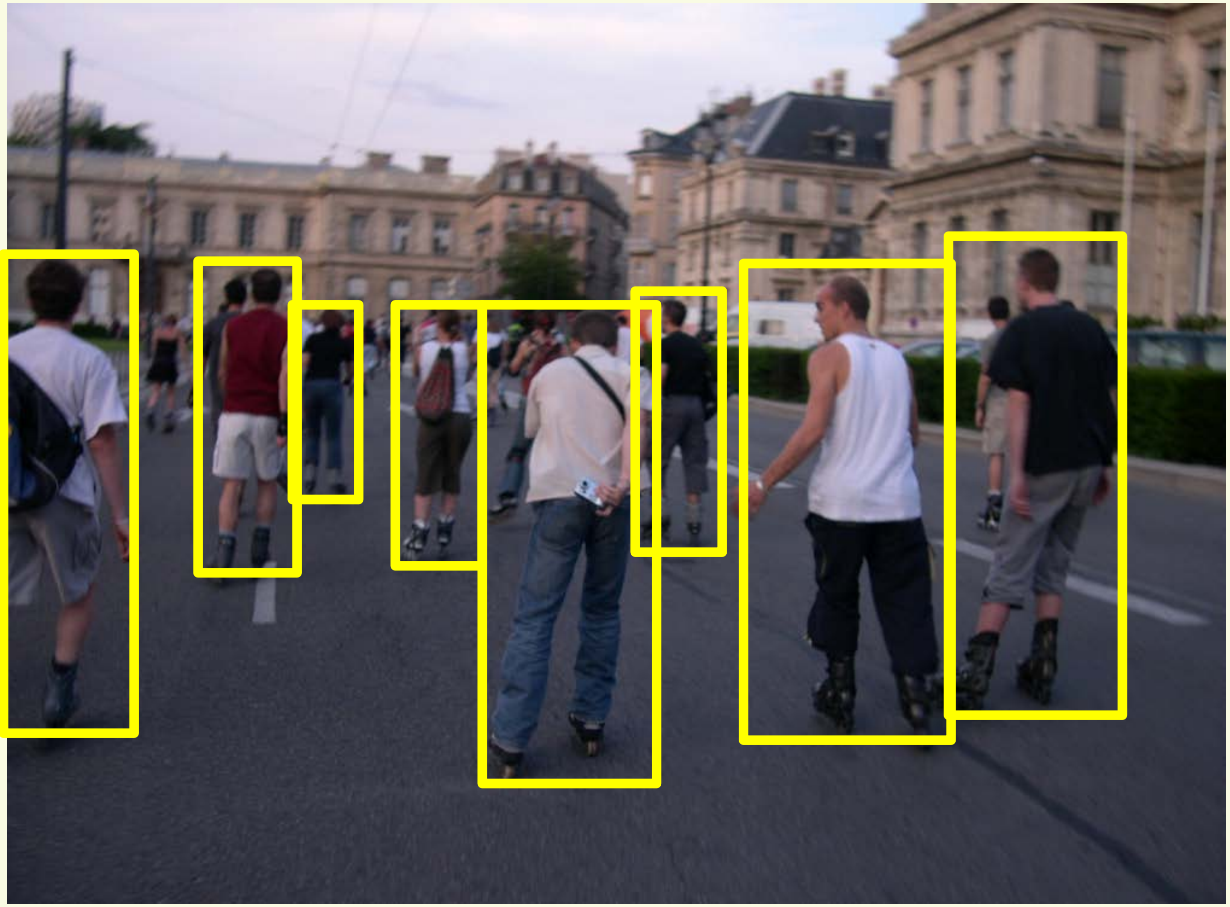
- Split data into training and test sets
- Preprocess data (i.e. segmenting fish from background)

- Extract possibly discriminating features
  - length, lightness, width, number of fins,etc.
- Classifier design
  - Choose model for classifier
  - Train classifier on training data
- Test classifier on test data

we mostly look at these steps in the course

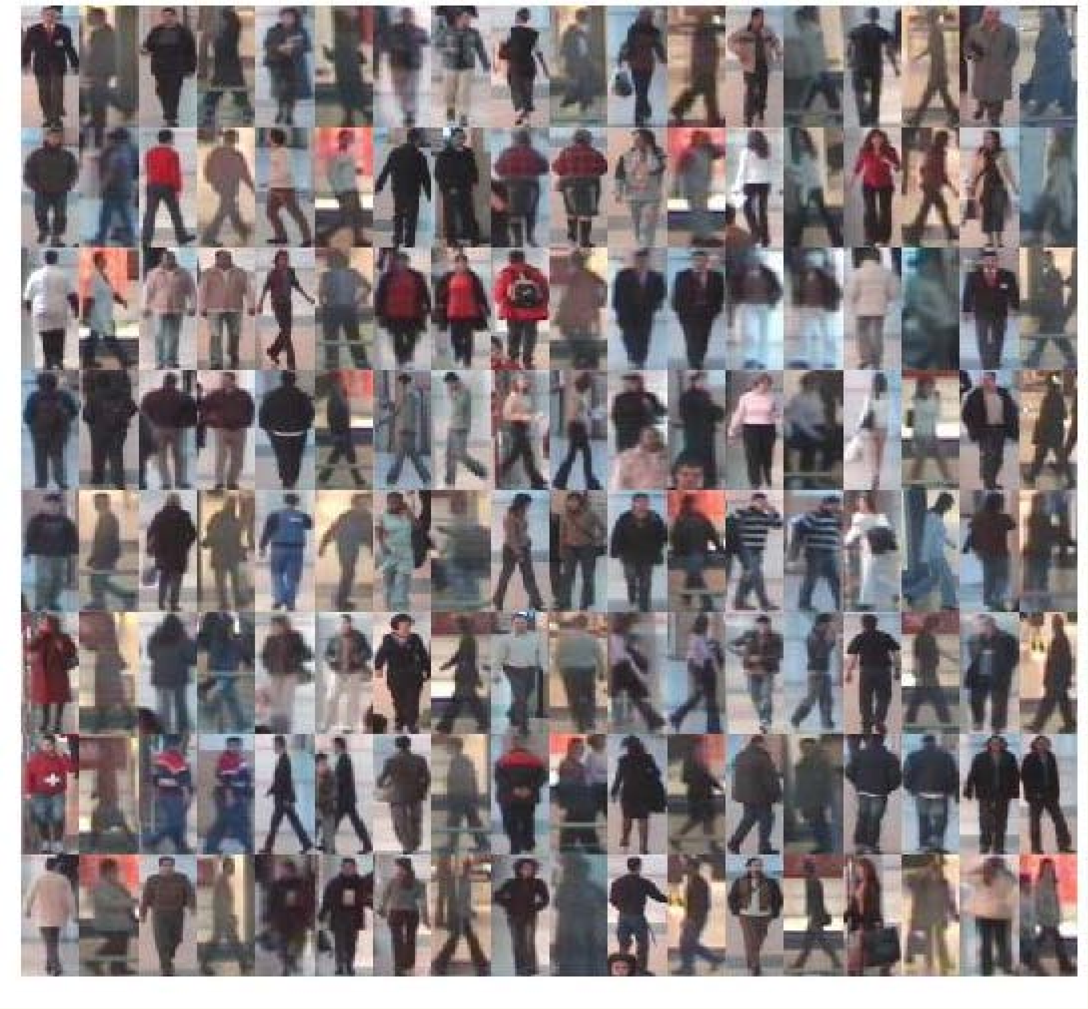
# Sliding Window Approach

- Objects of interest can appear at different scale and location in the image
- Example: Human Detection



# Sliding Window Approach

- Train on examples of the same scale



# Sliding Window Approach

- Apply the trained classifier to different locations
  - handles different locations

# *Sliding Window Approach*

- Shrink image, apply the trained classifier to different locations
  - handles different scales

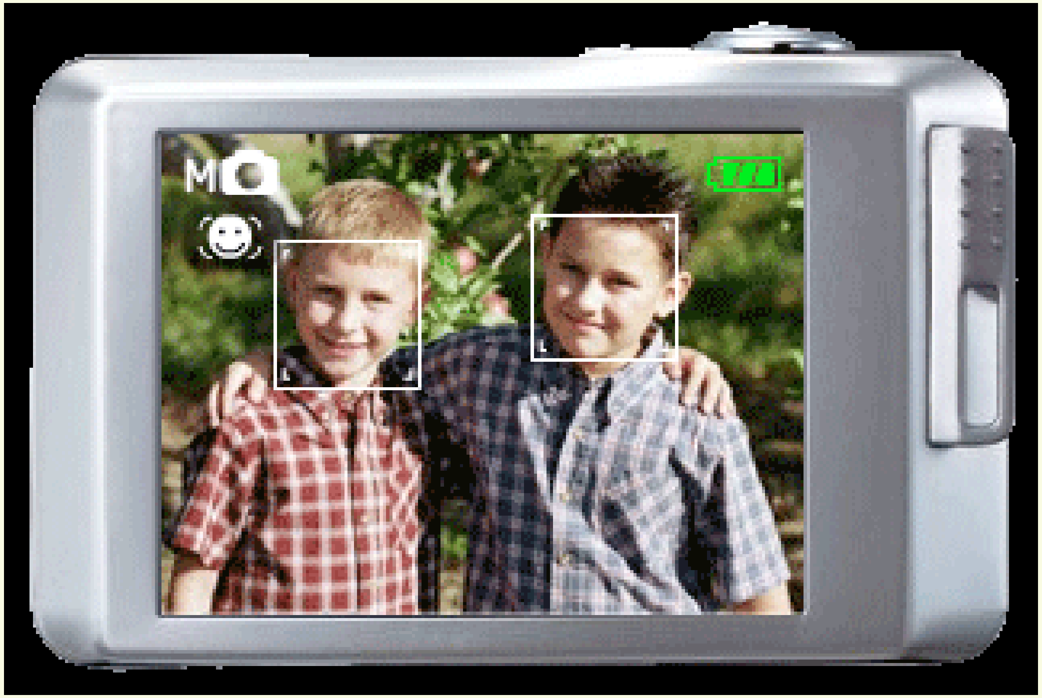
# Sliding Window Approach

- Shrink more
  - also can enlarge image, if needed

# Sliding Window Approach

- Can also apply to different window sizes
  - shrink/enlarge windows to be the same size as training data

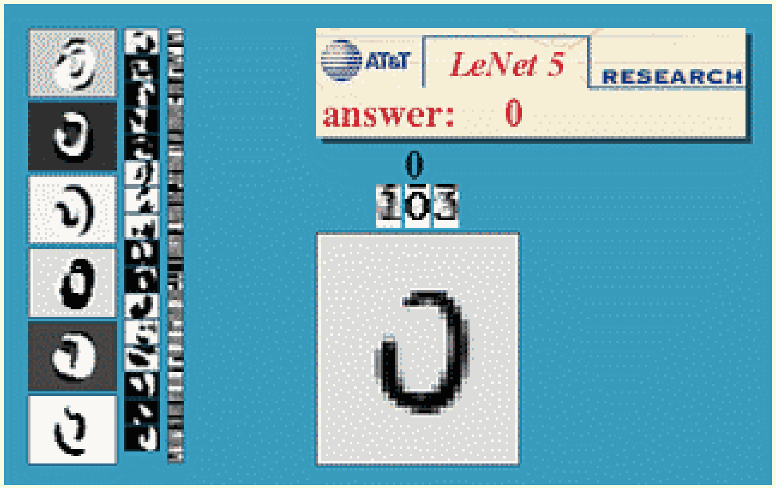
# *Application: Face Detection*




- Objects – image patches
- Classes – “face” and “not face”

# *Optical character recognition (OCR)*

- Objects – images or image patches
- Classes – digits 0, 1, …,9



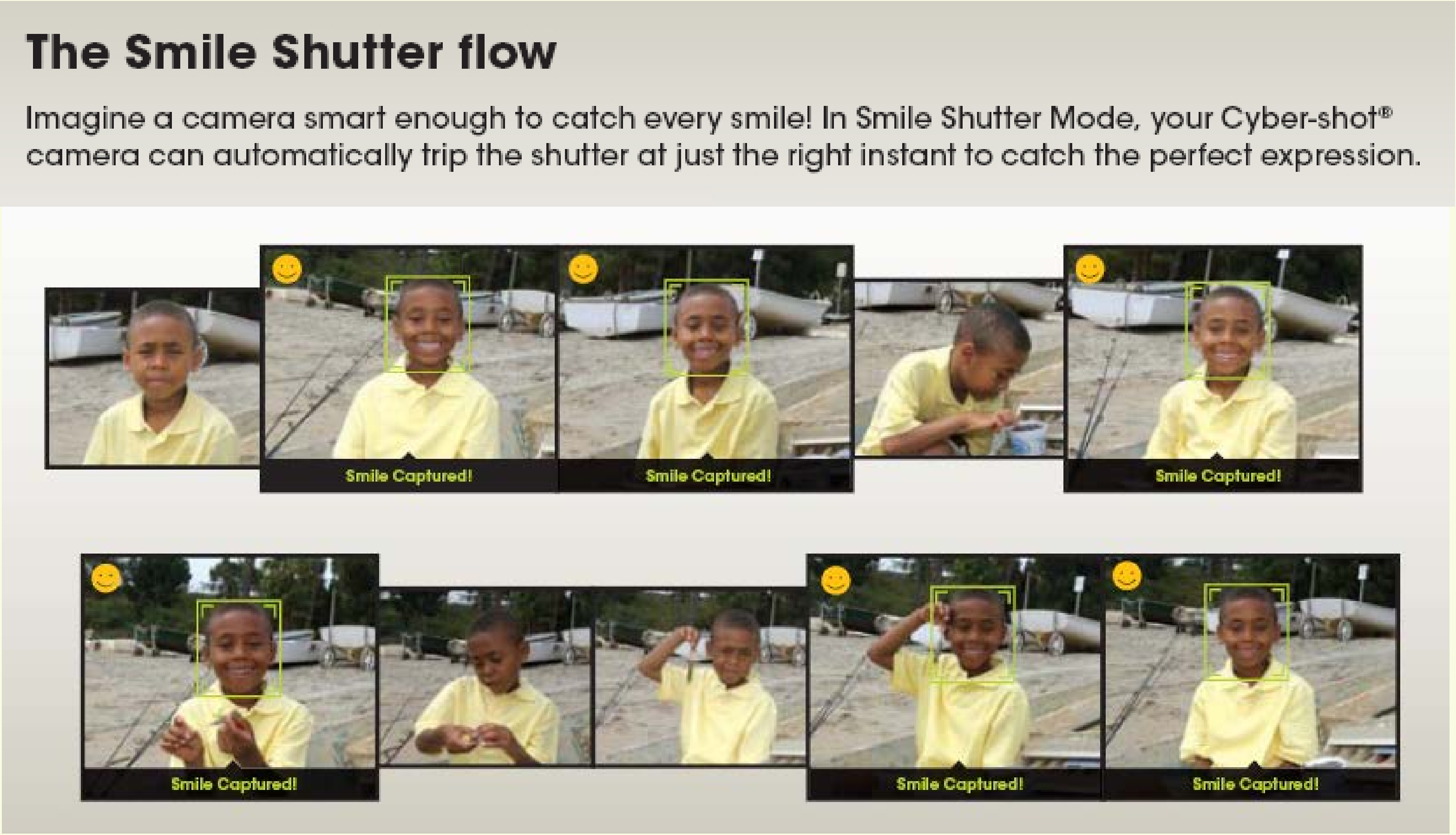

Digit recognition, AT&T labs
http://www.research.att.com/~yann/


License plate readers
http://en.wikipedia.org/wiki/Automatic_number_plate_recognition



Slide Credit: D. Hoiem

# Smile detection




[Sony Cyber-shot® T70 Digital Still Camera](#)



Slide Credit: D. Hoiem

# *Object recognition in mobile phones*

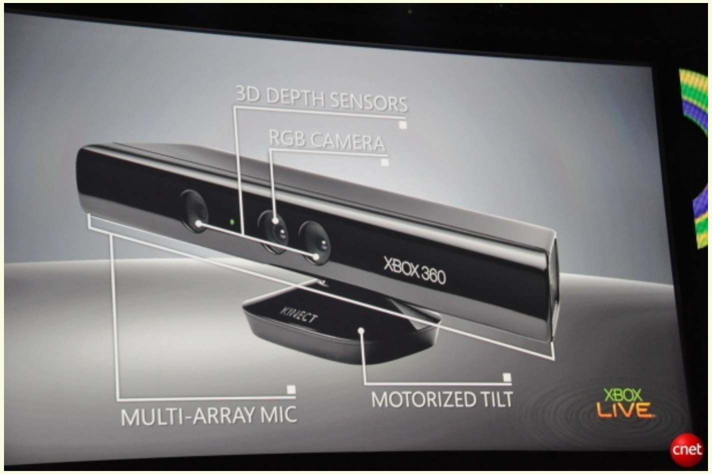

[Point & Find](), [Nokia]()



Slide Credit: D. Hoiem

# *Interactive Games: Kinect*

- Object Recognition: http://www.youtube.com/watch?feature=iv&v=fQ59dXOo63o
- Mario: http://www.youtube.com/watch?v=8CTJL5lUjHg
- 3D: http://www.youtube.com/watch?v=7QrnwoO1-8A
- Robot: http://www.youtube.com/watch?v=w8BmgtMKFbY





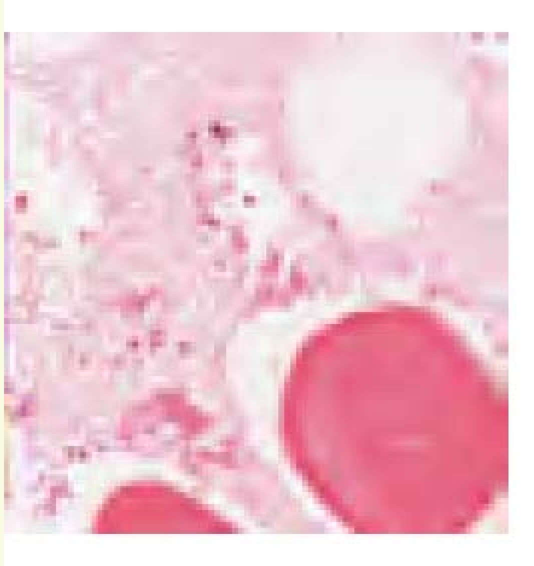


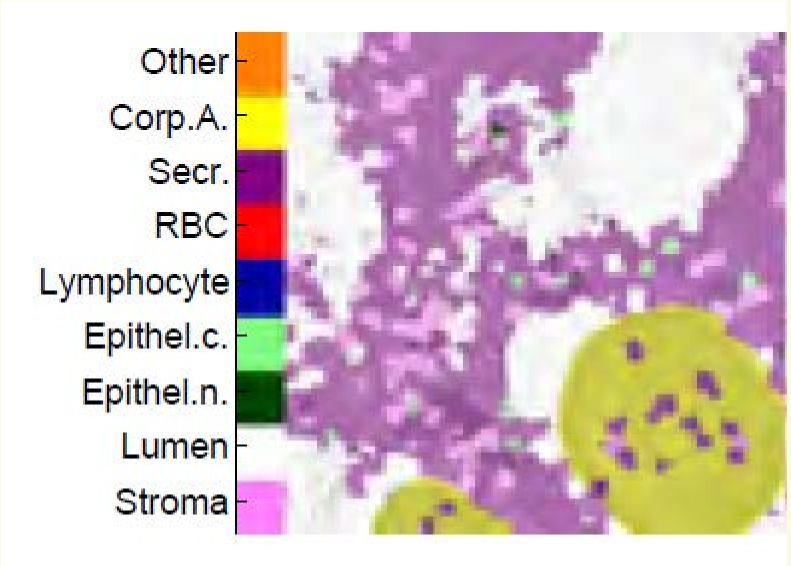
Slide Credit: D. Hoiem

# *Application: Scene Classification*

- Objects – images
- Classes – “mountain”, “lake”, “field”…

# *Application: Medical Image Processing*






- Objects – pixels
- Classes – different tissue types, stroma, lument, etc.