CS840a
Machine Learning in Computer Vision
Olga Veksler

Lecture 1
Introduction

Outline

- Course overview
- Introduction to Machine Learning

Course Outline

- Prerequisite
  - First-year course in Calculus
  - Introductory Statistics
  - Linear Algebra
  - Some Computer Vision/Image Processing
- Grading
  - Class participation 10%
  - In class paper presentation 30%
  - Final Project Presentation 20%
  - Written project report + code, 40 %
    - Matlab, C/C++, anything else as long as I can run it

Course Outline: Content

- Lecture (1/2 of the time), paper presentation/discussions/video (1/2 of the time)
- Machine Learning Methods (tentatively)
  - Nearest neighbor
  - Linear classifiers
  - Neural nets
  - SVM
  - Boosting
- Applications in Computer Vision
  - Object detection/recognition
  - Segmentation
  - Tracking
  - Inpainting

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
  - “Machine Learning: a Probabilistic Perspective” by Kevin Patrick Murphy
- Conference papers

Intro: What is Machine Learning?

- How to write a computer program that automatically improves its performance through experience
- Machine learning is useful when it is too difficult to come up with a program to perform a desired task
- Make computer to learn by showing examples (usually with correct answers)
  - “supervised” learning or learning with a teacher
- In practice: computer program (or function) which has a tunable parameters, tune parameters until the desirable behavior on the examples

Different Types of 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 world: e.g. discover clusters
- **Reinforcement Learning**: not covered in this course

Supervised Machine Learning

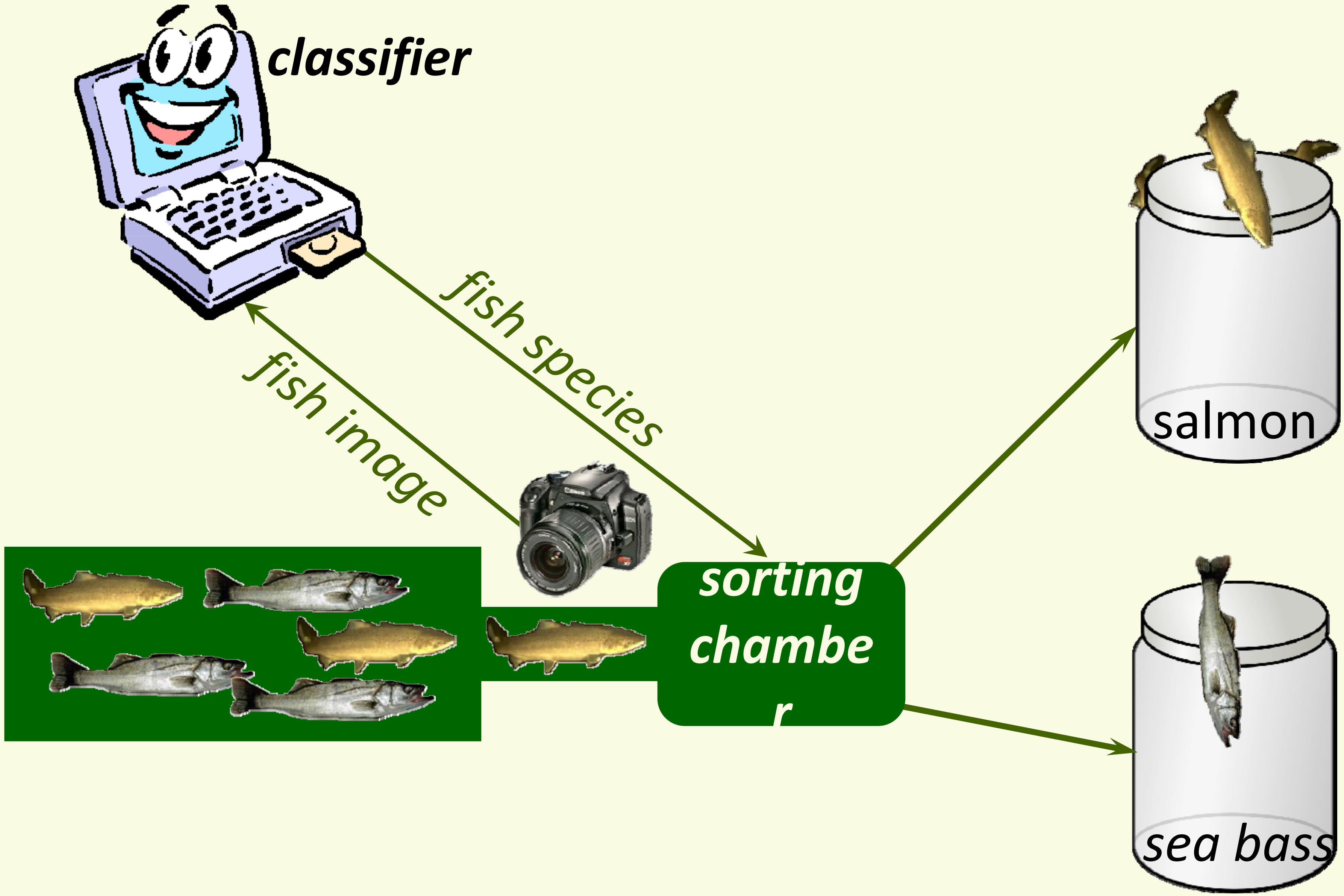
- Training samples (or examples) $\mathbf{x}^1,\mathbf{x}^2,\ldots, \mathbf{x}^n$
- Each $\mathbf{x}^i$ is usually multi-dimensional
  - $\mathbf{x}^i_1, \mathbf{x}^i_2 ,\ldots, \mathbf{x}^i_d$ are called *features*
  - $\mathbf{x}^i$ is also called a *feature vector*
  - example: $\mathbf{x}^1 = \{3,7, 35\}$, $\mathbf{x}^2 = \{5, 9, 47\}$, …
    - how many and which features do we take?
- Have target output for each example $\mathbf{y}^1, \mathbf{y}^2,\ldots\mathbf{y}^n$
  - “teacher” gives target outputs
  - $\mathbf{y}^i$ are usually one-dimensional
  - example: $\mathbf{y}^1 = 1$ (“face”), $\mathbf{y}^2 = 0$ (“not a face”)

Two Types of Supervised Machine Learning

- Classification
  - $\mathbf{y}^i$ is finite, typically called a *label* or a *class*
  - example: $\mathbf{y}^i \in$ {“sunny”, ”cloudy”, ”raining”}

- Regression
  - $\mathbf{y}^i$ is continuous, typically called an *output value*
  - Example: $\mathbf{y}^i$ = temperature $\in [-60,60]$

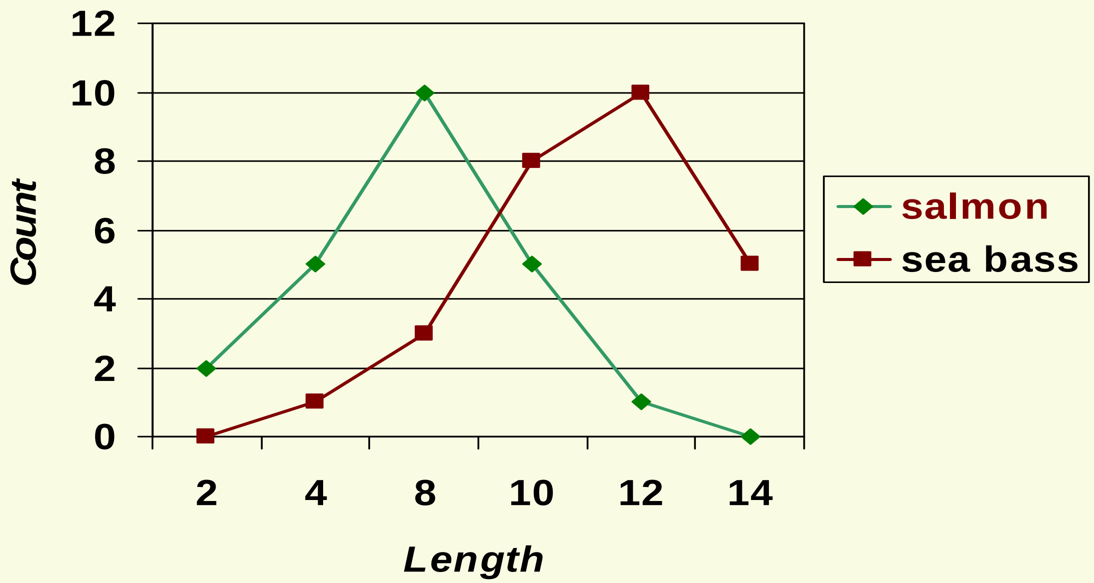
Toy Application: fish sorting



Classifier design

- Notice salmon tends to be shorter than sea bass
- Use *fish length* as the discriminating feature
- Count number of bass and salmon of each length

	2	4	8	10	12	14
bass	0	1	3	8	10	5
salmon	2	5	10	5	1	0



Single Feature (length) Classifier

- Find the best length ***L*** threshold

fish length < L → ***classify as salmon***

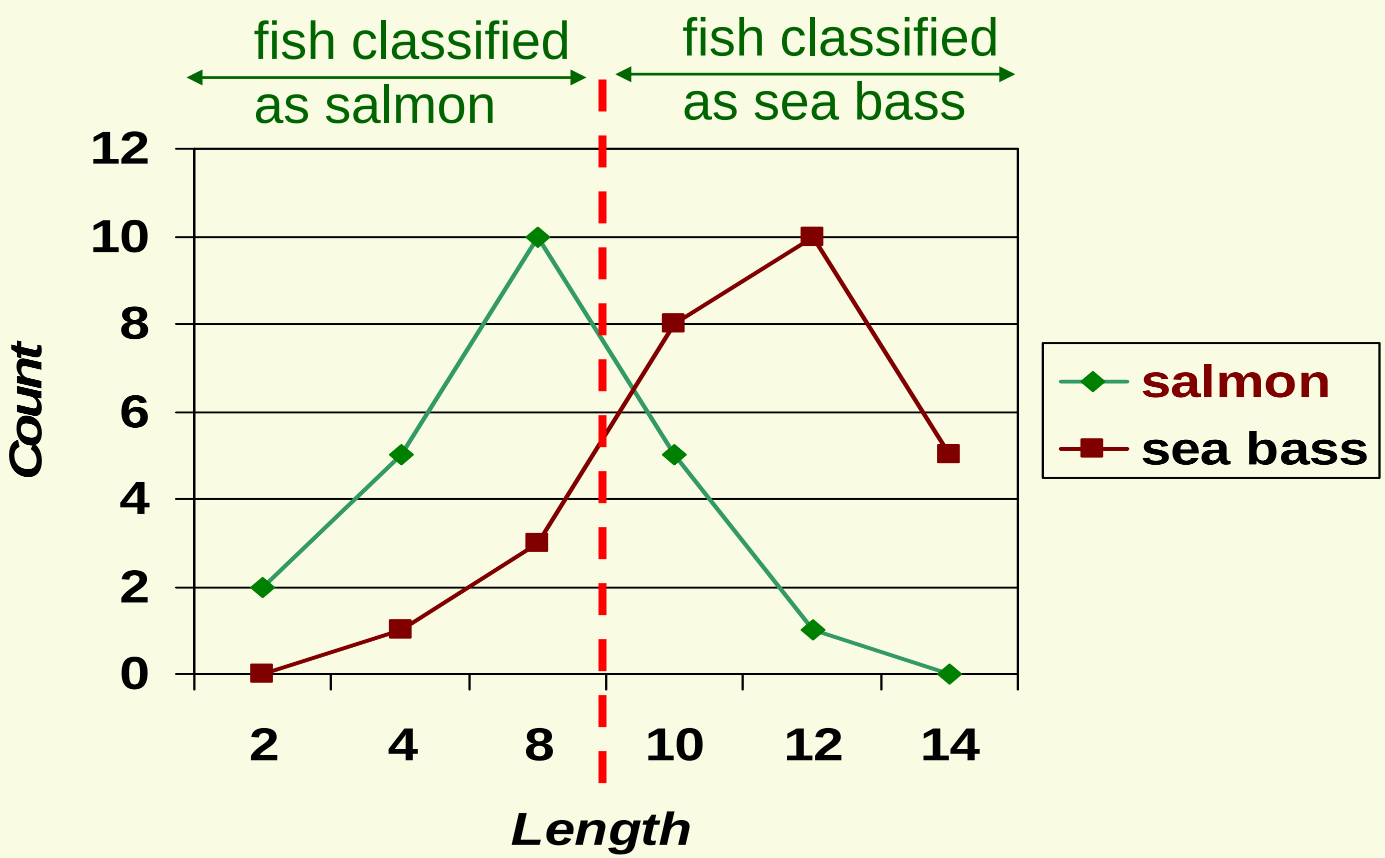
fish length > L → ***classify as sea bass***

- For example, at ***L*** = 5, misclassified:
  - 1 sea bass
  - 16 salmon

	2	4	8	10	12	14
bass	0	1	3	8	10	5
salmon	2	5	10	5	1	0

- Classification error (total error) $\frac{17}{50}$ = ***34%***

Single Feature (length) Classifier



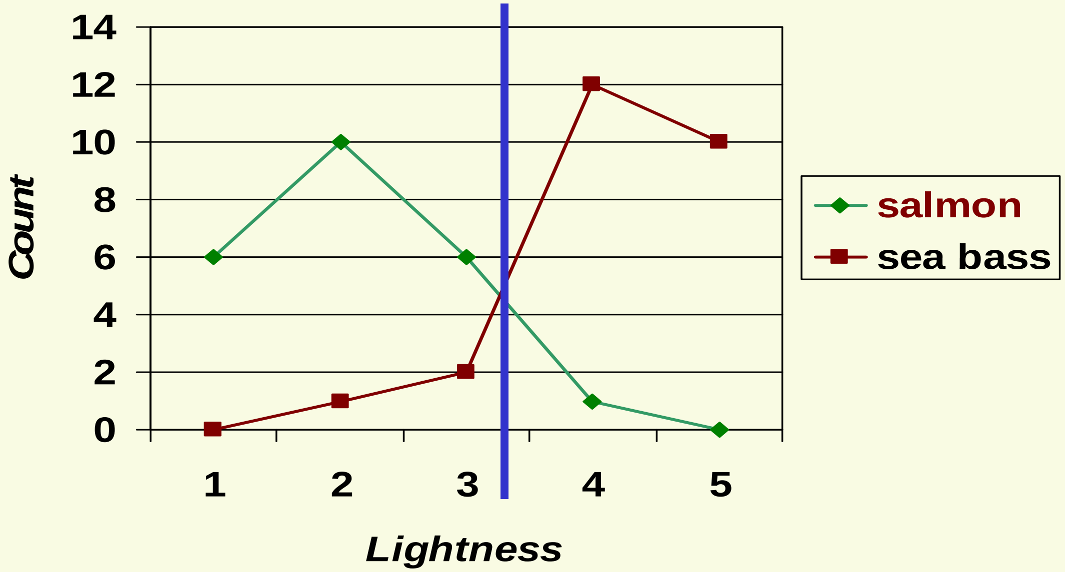
- After searching through all possible thresholds ***L***, the best ***L***= 9, and still 20% of fish is misclassified

Next Step

- Lesson learned:
  - Length is a poor feature alone!
- What to do?
  - Try another feature
  - Salmon tends to be lighter
  - Try average fish lightness

Single Feature (lightness) Classifier

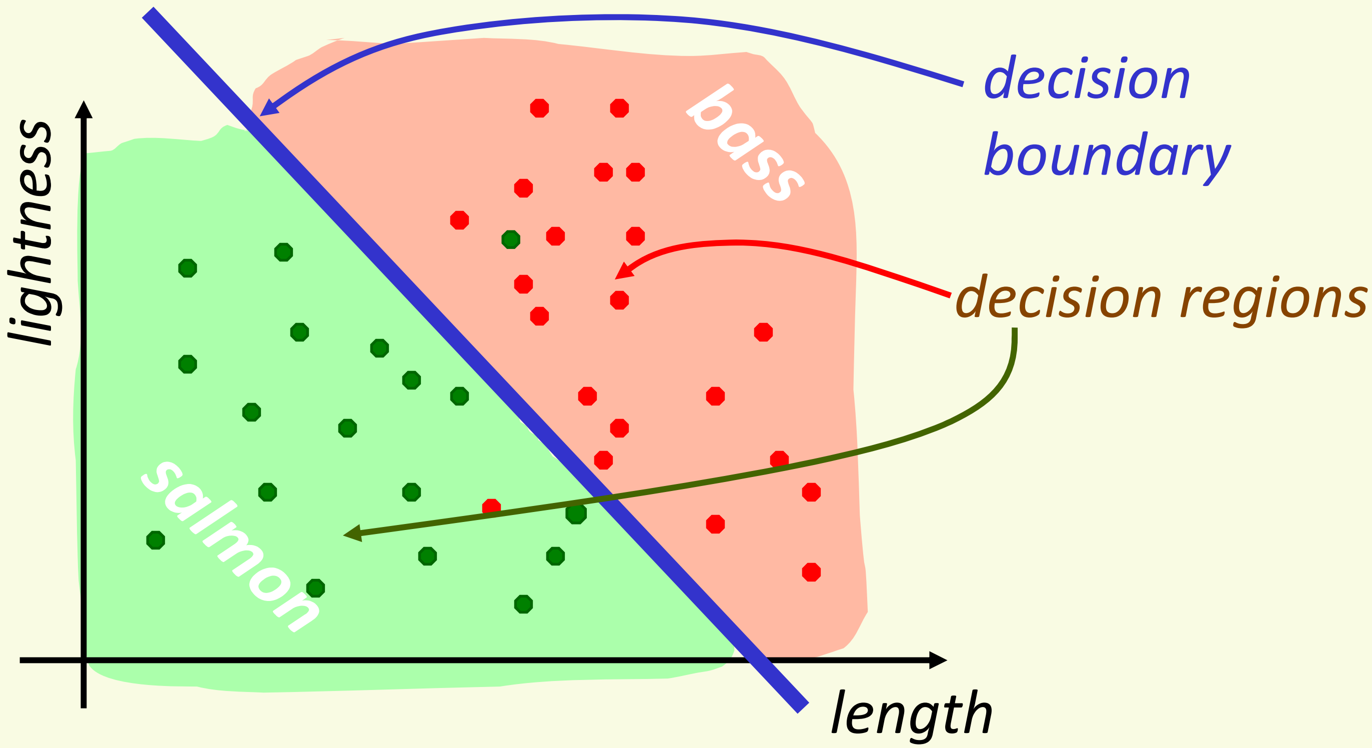
	1	2	3	4	5
bass	0	1	2	10	12
salmon	6	10	6	1	0



- Now fish are classified best at lightness threshold of 3.5 with classification error of 8%

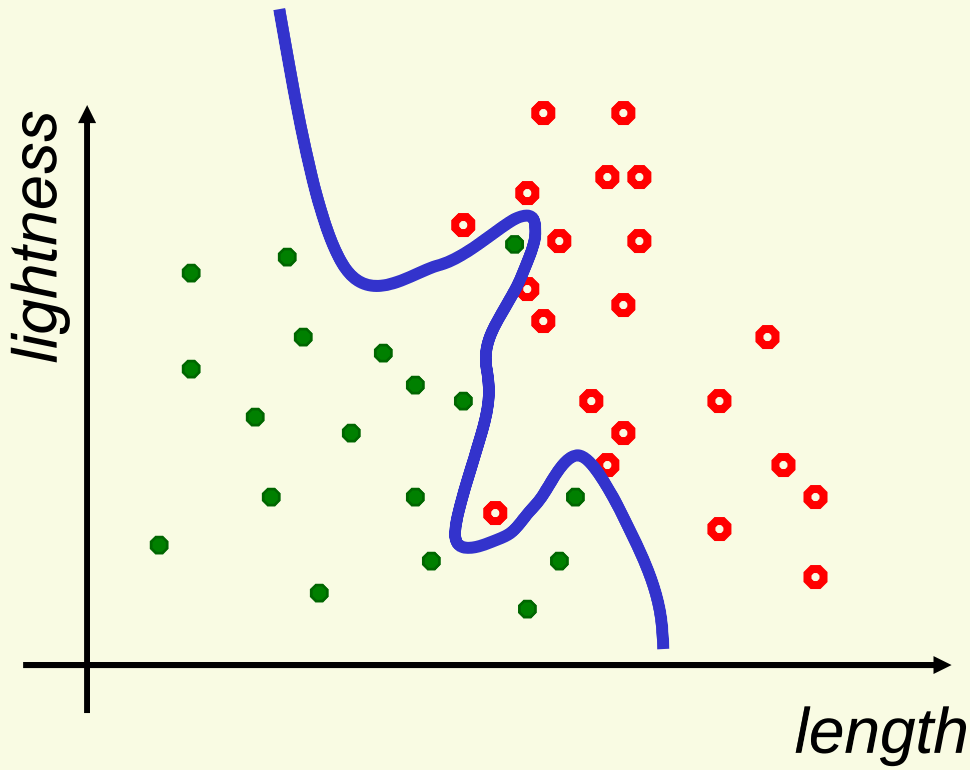
Can do better by feature combining

- Use both length and lightness features
- Feature vector [length,lightness]



- Classification error 4%

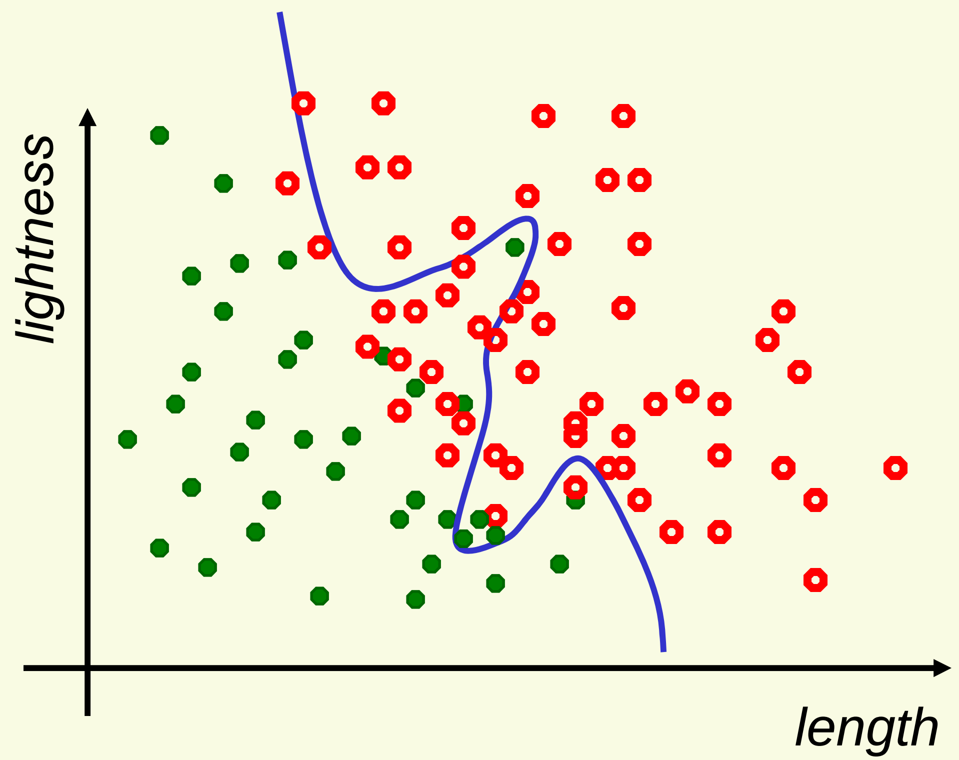
Even Better Decision Boundary



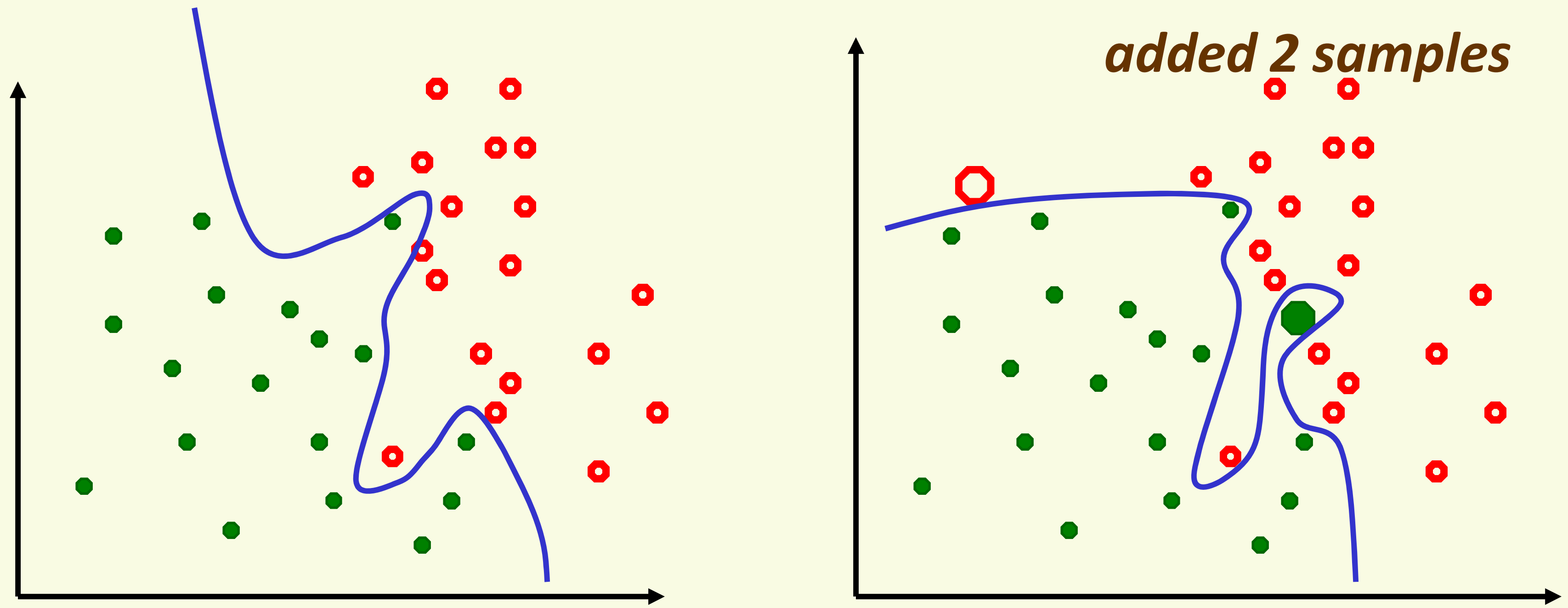
- Decision boundary (wiggly) with 0% error

Test Classifier on New Data

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

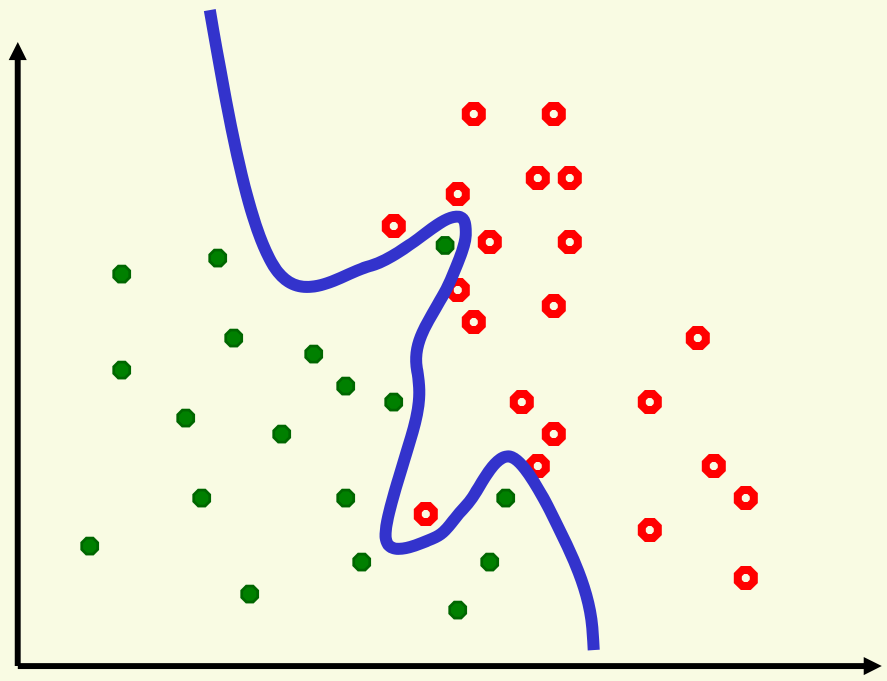


What Went Wrong?



- We always have only a limited amount of data, not all possible data
- We should make sure the decision boundary does not adapt too closely to the particulars of the data we have at hand, but rather grasps the “big picture”

Overfitting



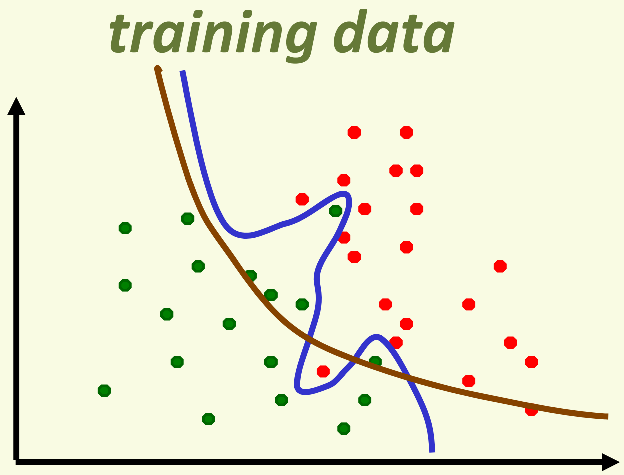
- Complicated boundaries ***overfit*** the data, they are too tuned to the particular training data at hand
- Therefore complicated boundaries tend to not ***generalize*** well to the new data
- We usually refer to the new data as "test" data

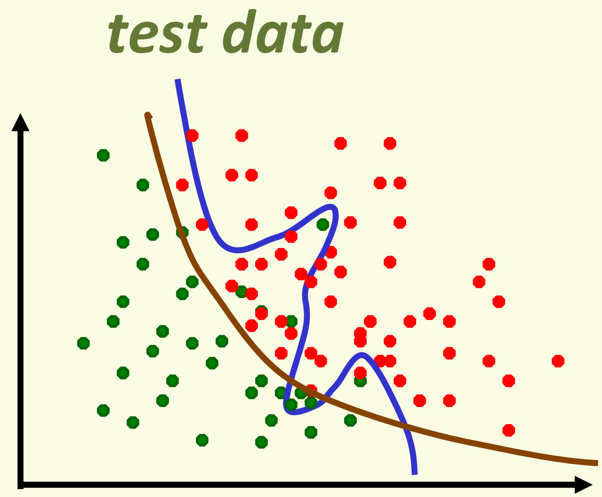
Overfitting: Extreme Example

- Say we have 2 classes: 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
- Rote learning is memorization without generalization

slide is modified from Y. LeCun

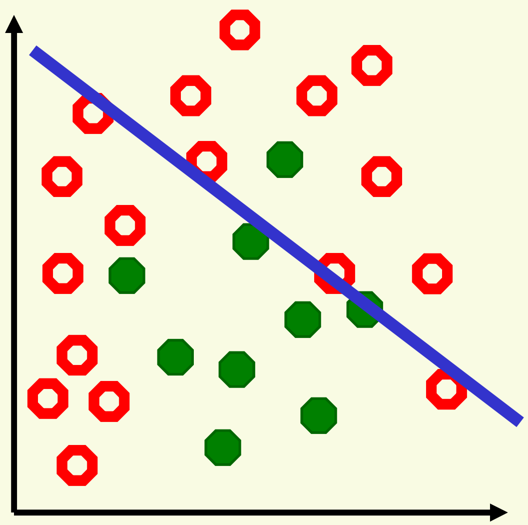
Generalization





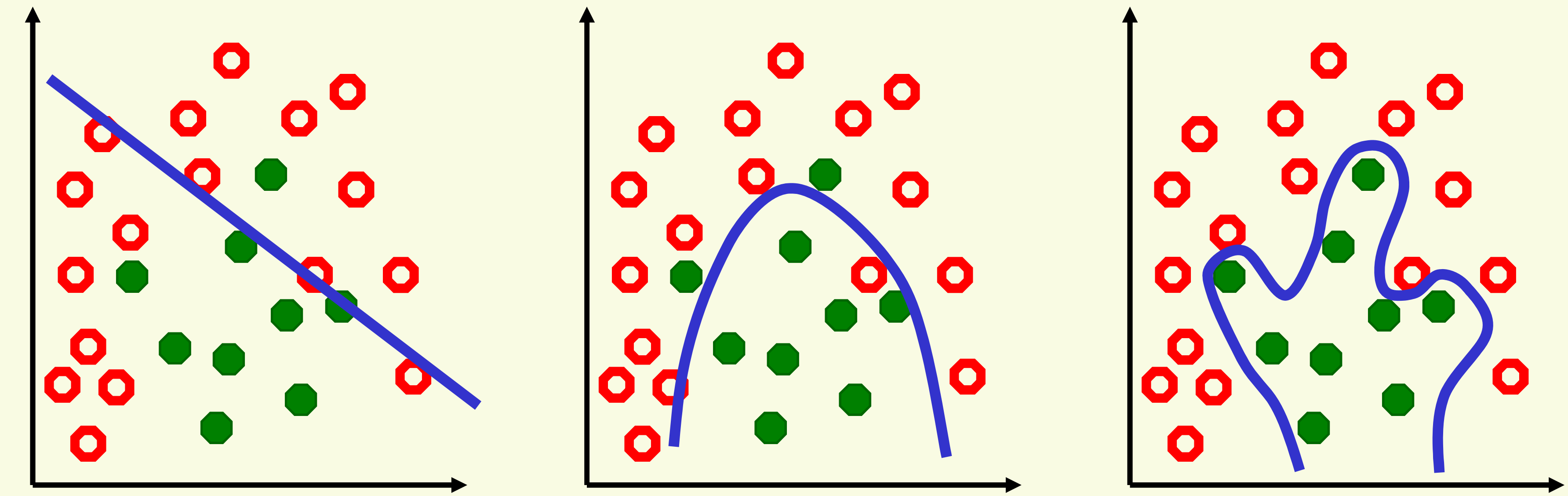
- The ability to produce correct outputs on previously unseen examples is called **generalization**
- The big question of learning theory: how to get good generalization with a limited number of examples
- Intuitive idea: favor simpler classifiers
  - William of Occam (1284-1347): "entities are not to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

Underfitting



- Can also underfit data, i.e. too simple decision boundary
  - chosen model is not expressive enough
- There is no way to fit a linear decision boundary so that the training examples are well separated
- Training error is too high
  - test error is, of course, also high

Underfitting → Overfitting



underfitting *"just right"* *overfitting*

Sketch of Supervised Machine Learning

- Chose a *learning machine* **f**(**x**,**w**)
  - **w** are tunable weights
  - **x** is the input sample
  - **f**(**x**,**w**) should output the correct class of sample **x**
  - use labeled samples to tune weights **w** so that **f**(**x**,**w**) give the correct label for sample **x**
- Which function **f**(**x**,**w**) do we choose?
  - has to be expressive enough to model our problem well, i.e. to avoid ***underfitting***
  - yet not to complicated to avoid ***overfitting***

Training and Testing

- There are 2 phases, training and testing
  - Divide all labeled samples $\mathbf{x}^1, \mathbf{x}^2, \ldots \mathbf{x}^n$ into 2 sets, *training* set and *test* set
  - Training phase is for "teaching" our machine (finding optimal weights $\mathbf{w}$)
  - Testing phase is for evaluating how well our machine works on unseen examples

Training Phase

- Find the weights **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
- How do we find parameters **w** to ensure $\mathbf{f}(\mathbf{x}^i,\mathbf{w}) = \mathbf{y}^i$ for most training samples $(\mathbf{x}^i,\mathbf{y}^i)$?
  - This step is usually done by optimization, can be quite time consuming

Testing Phase

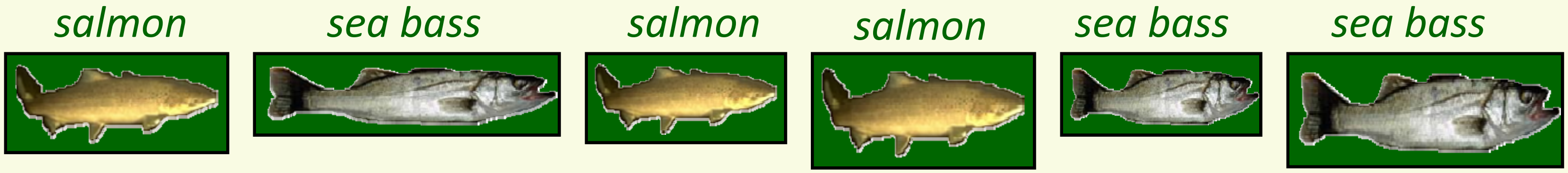
- The goal is to design machine which performs well on unseen examples
- Evaluate the performance of the trained machine **f**(**x**,**w**) on the test samples (unseen labeled samples)
- Testing the machine on unseen labeled examples lets us approximate how well it will perform in practice
- If testing results are poor, may have to go back to the training phase and redesign **f**(**x**,**w**)

Generalization and Overfitting

- *Generalization* is the ability to produce correct output on previously unseen examples
  - i.e. low error on unseen examples
  - Good generalization is the main goal of ML
- Low training error does not necessarily imply low test error
  - we have seen that it is easy to produce **f**(**x**,**w**) which is perfect on training samples (rote “learning”)
- *Overfitting*
  - when the machine performs well on training data but poorly on test data

Classification System Design Overview

- Collect and label data by hand

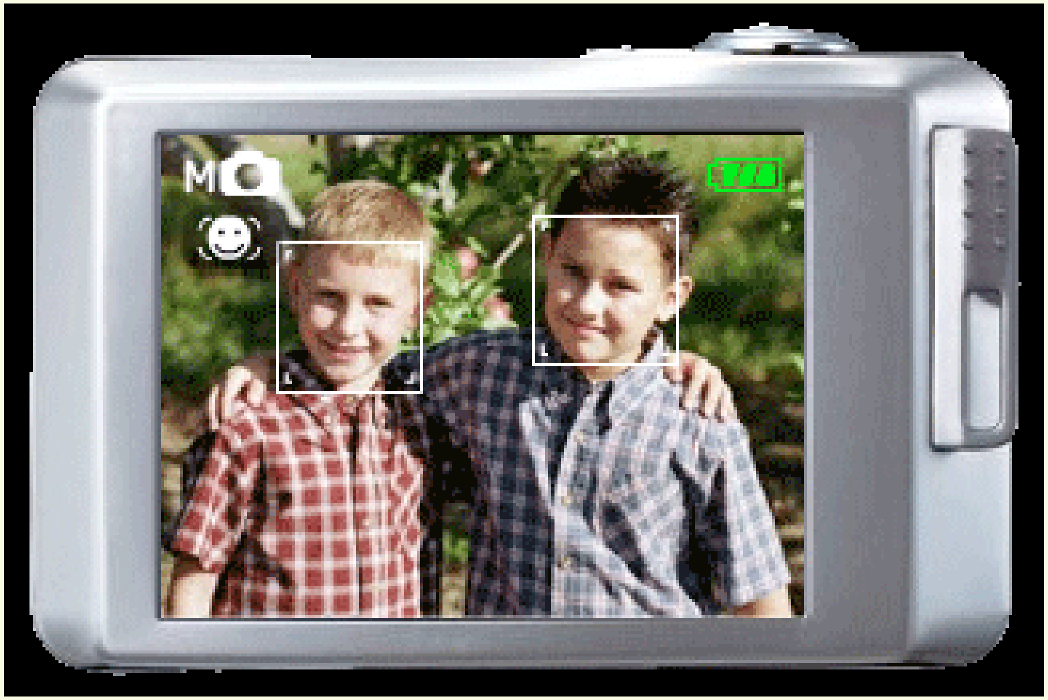


- Split data into training and test sets
- Preprocess by segmenting fish from background

- 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 two steps in this course

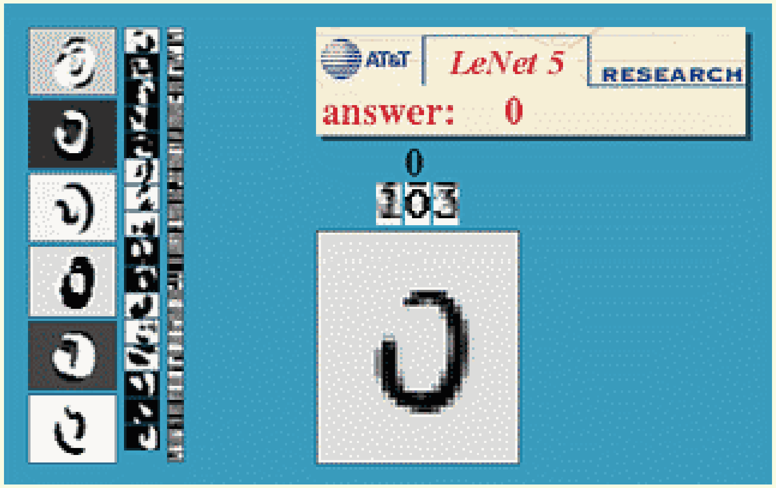
Application: Face Detection



- Objects – image patches
- Classes – "face" and "not face"

Optical character recognition (OCR)

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

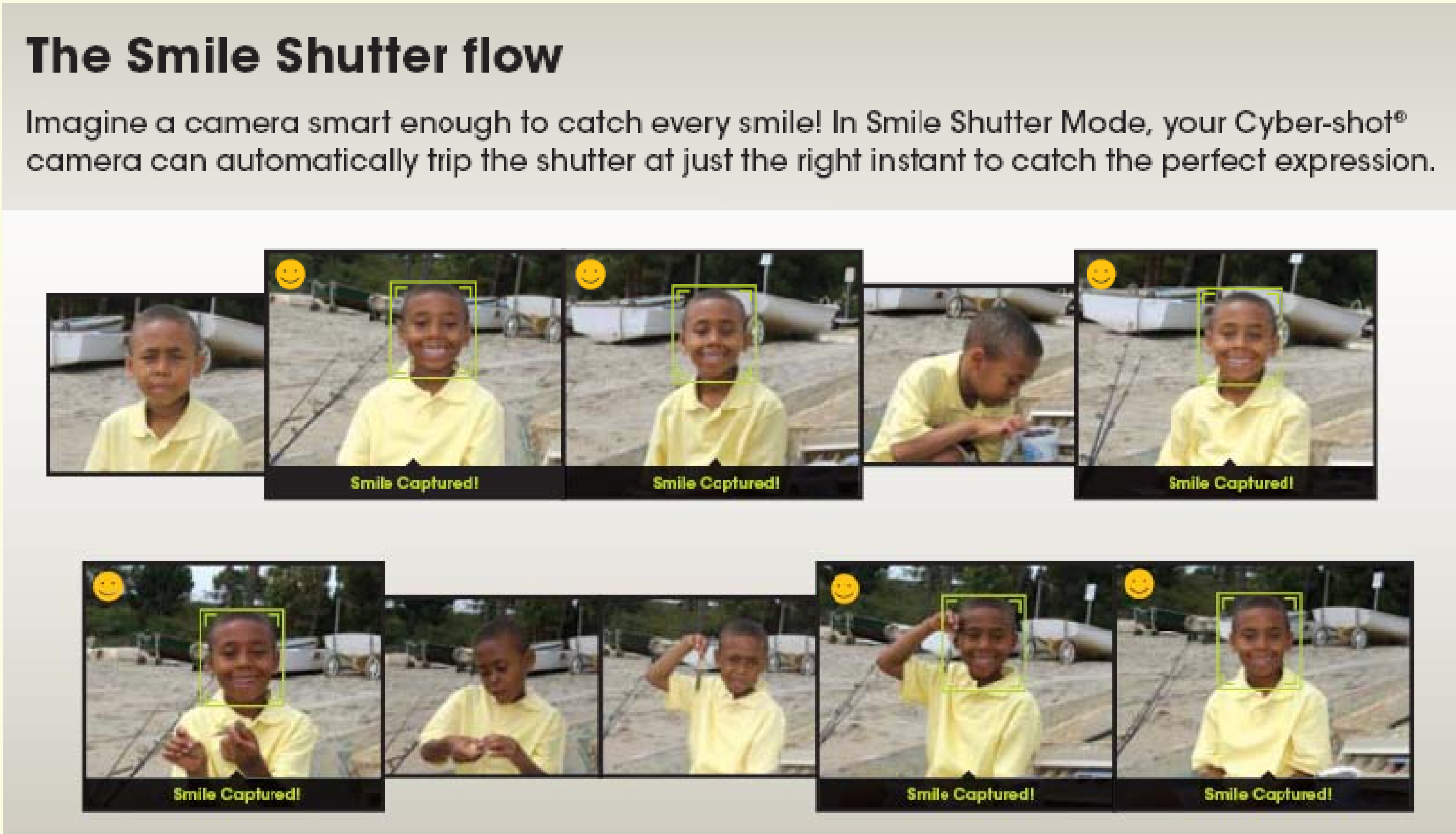


Digit recognition, AT&T labs
http://www.research.att.com/~yann/

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

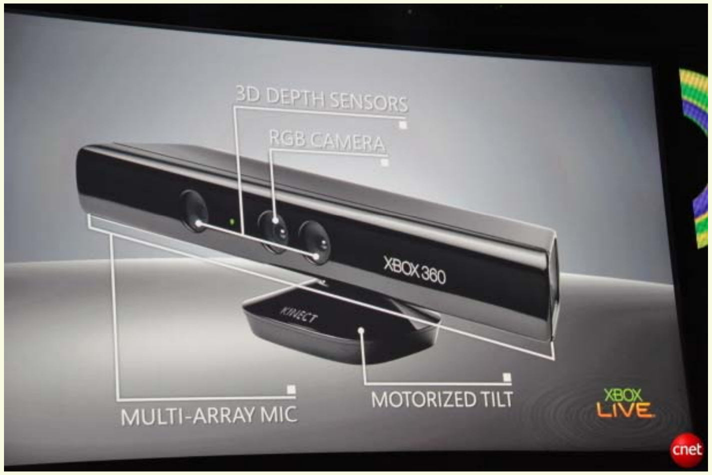
[Point & Find](), [Nokia]()

[Google Goggles]()

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



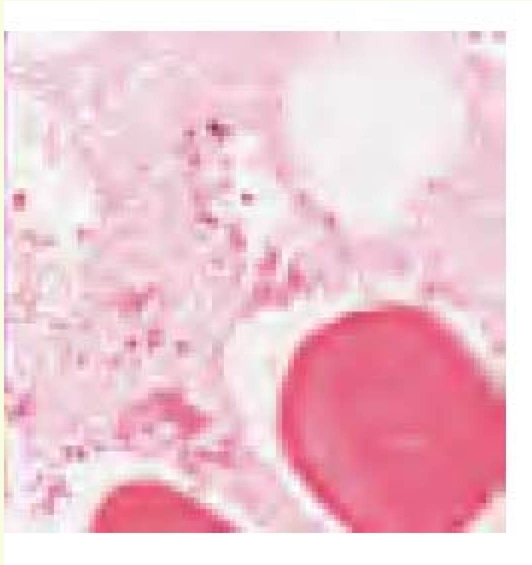
Slide Credit: D. Hoiem

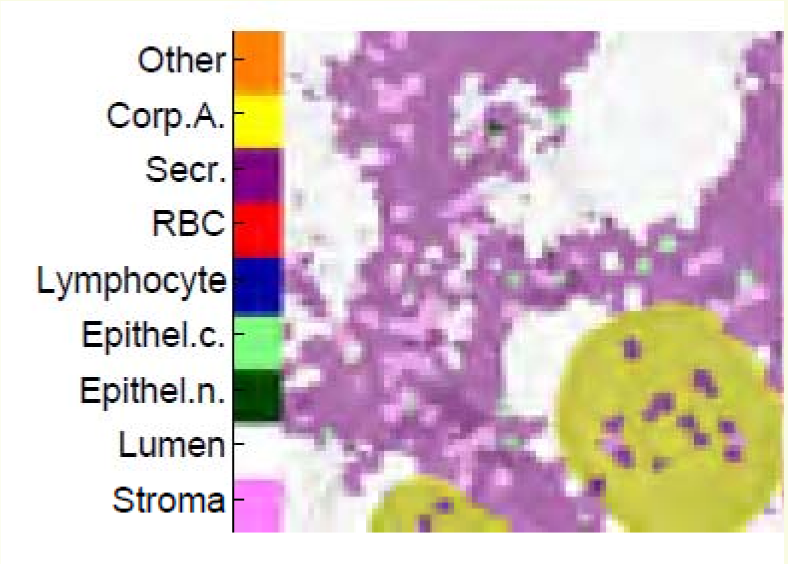
Application: Scene Classification



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

Application: Medical Image Processing





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