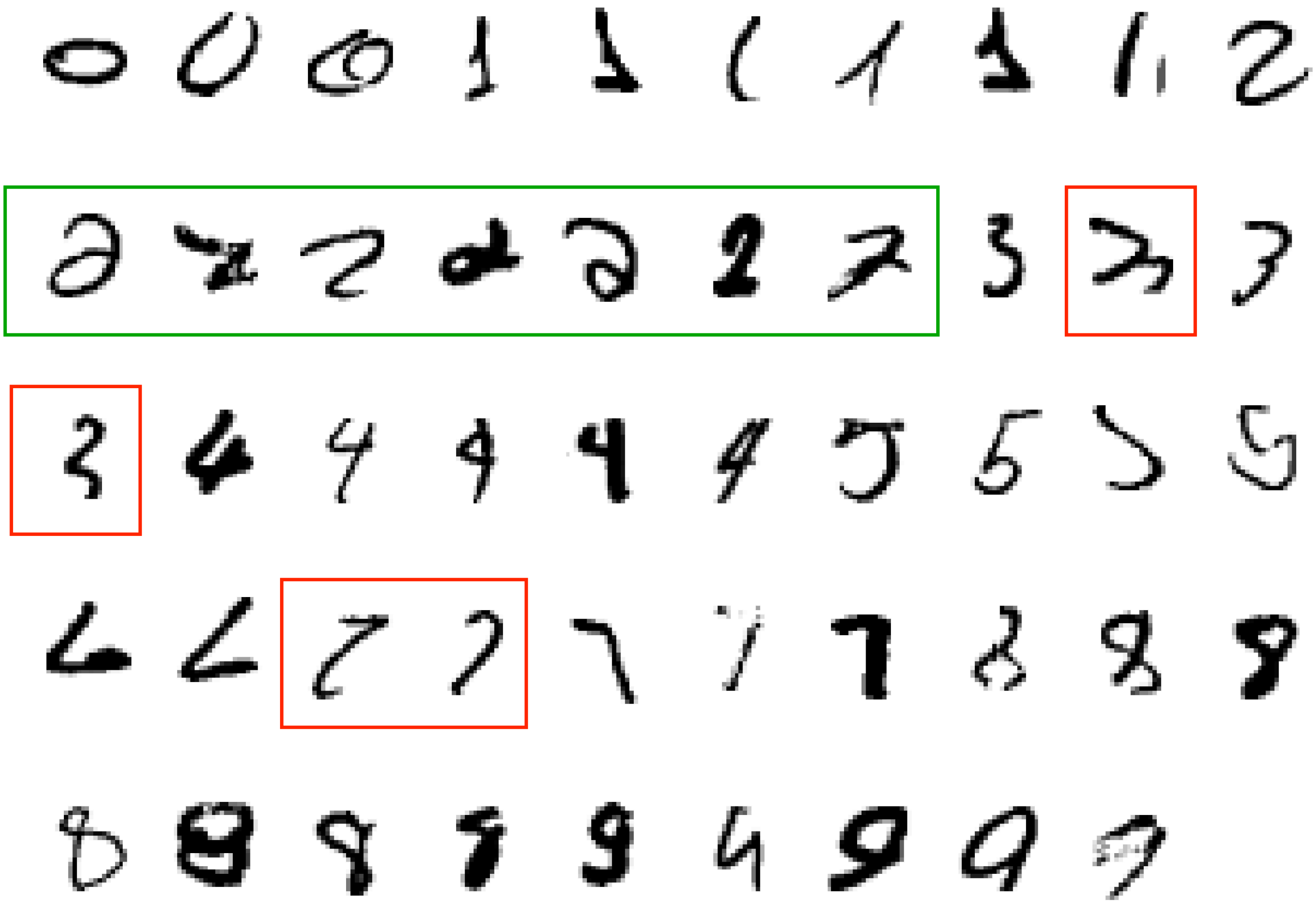
What is Machine Learning?

- How can we solve a specific problem?
  - As computer scientists we write a program that encodes a set of rules that are useful to solve the problem

Figure : How can we make a robot cook?

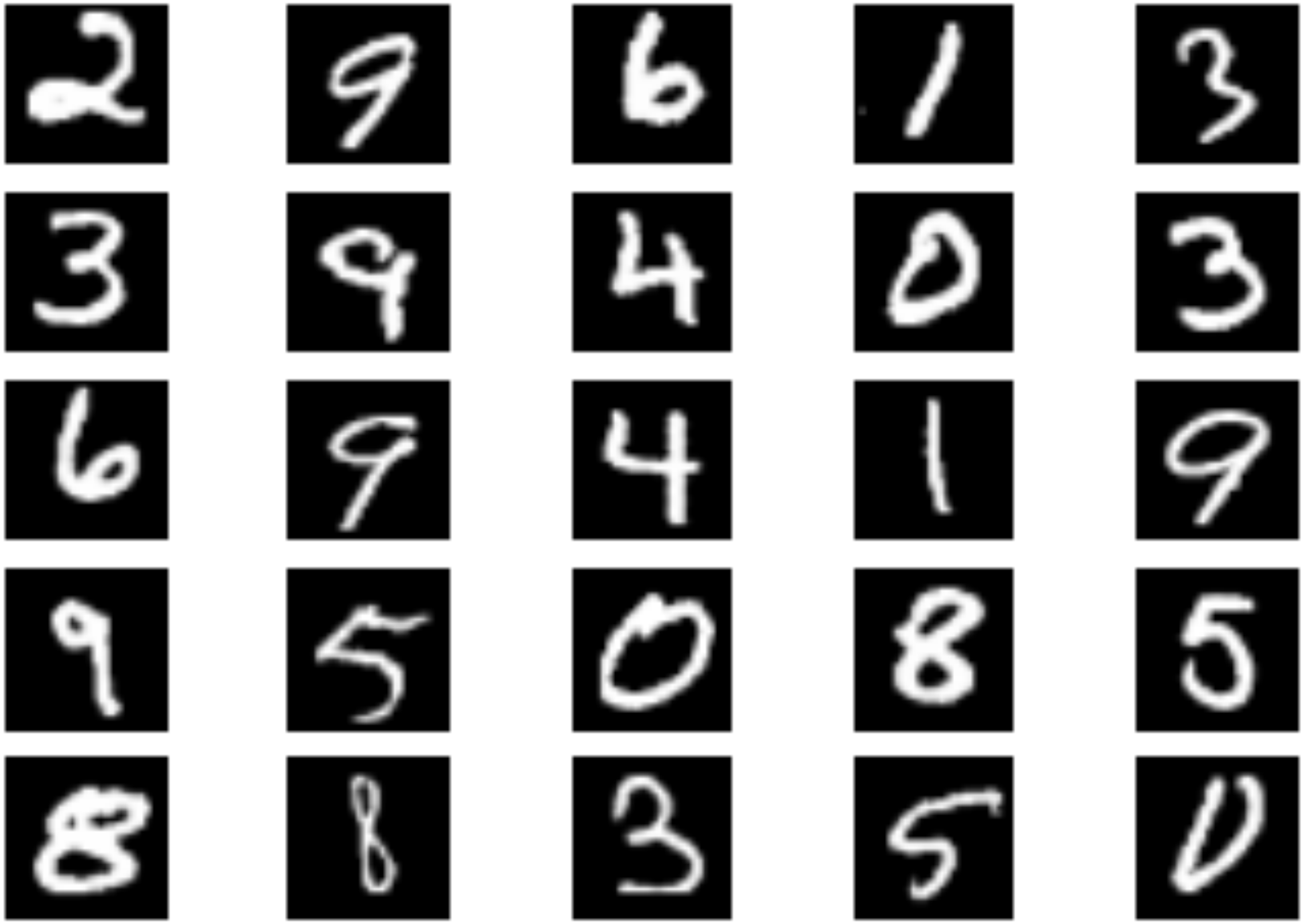
Tasks that requires machine learning: What makes a 2?



Learning algorithms are useful in many tasks

1. Classification: Determine which discrete category the example is

Examples of Classification



Learning algorithms are useful in many tasks

1. Classification: Determine which discrete category the example is
2. Recognizing patterns: Speech Recognition, facial identity, etc

Examples of Recognizing patterns

Figure : Siri: https://www.youtube.com/watch?v=8ciagGASro0

Learning algorithms are useful in other tasks

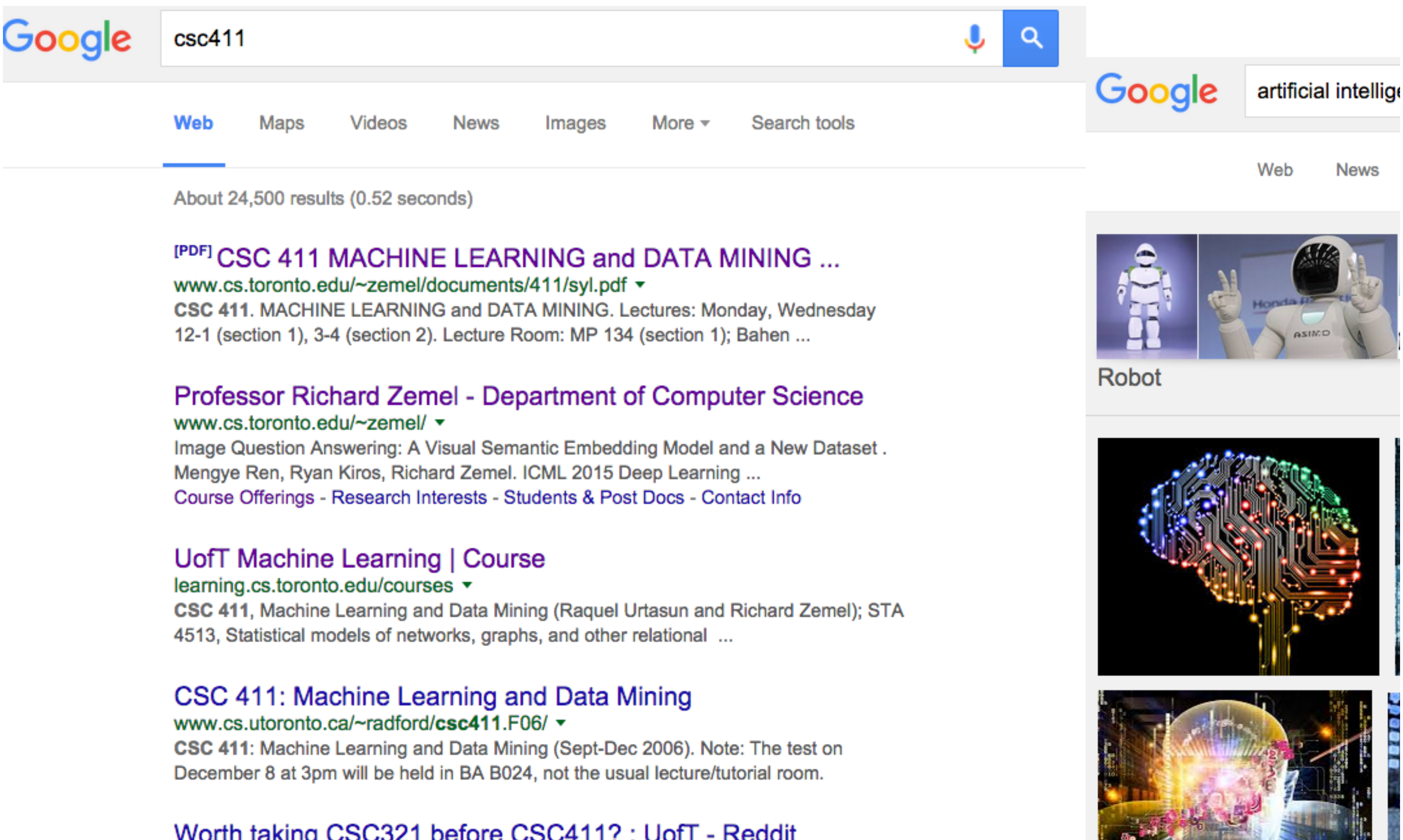
1. Classification: Determine which discrete category the example is
2. Recognizing patterns: Speech Recognition, facial identity, etc
3. Recommender Systems: Noisy data, commercial pay-off (e.g., Amazon, Netflix).

Examples of Recommendation systems

Learning algorithms are useful in other tasks

1. Classification: Determine which discrete category the example is
2. Recognizing patterns: Speech Recognition, facial identity, etc
3. Recommender Systems: Noisy data, commercial pay-off (e.g., Amazon, Netflix).
4. Information retrieval: Find documents or images with similar content

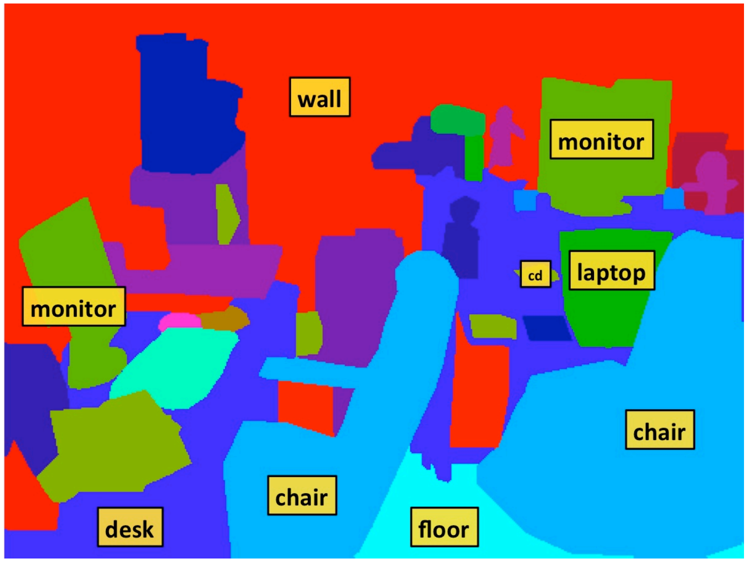
Examples of Information Retrieval



Learning algorithms are useful in other tasks

1. Classification: Determine which discrete category the example is
2. Recognizing patterns: Speech Recognition, facial identity, etc
3. Recommender Systems: Noisy data, commercial pay-off (e.g., Amazon, Netflix).
4. Information retrieval: Find documents or images with similar content
5. Computer vision: detection, segmentation, depth estimation, optical flow, etc

Computer Vision



Learning algorithms are useful in other tasks

1. Classification: Determine which discrete category the example is
2. Recognizing patterns: Speech Recognition, facial identity, etc
3. Recommender Systems: Noisy data, commercial pay-off (e.g., Amazon, Netflix).
4. Information retrieval: Find documents or images with similar content
5. Computer vision: detection, segmentation, depth estimation, optical flow, etc
6. Robotics: perception, planning, etc

Autonomous Driving

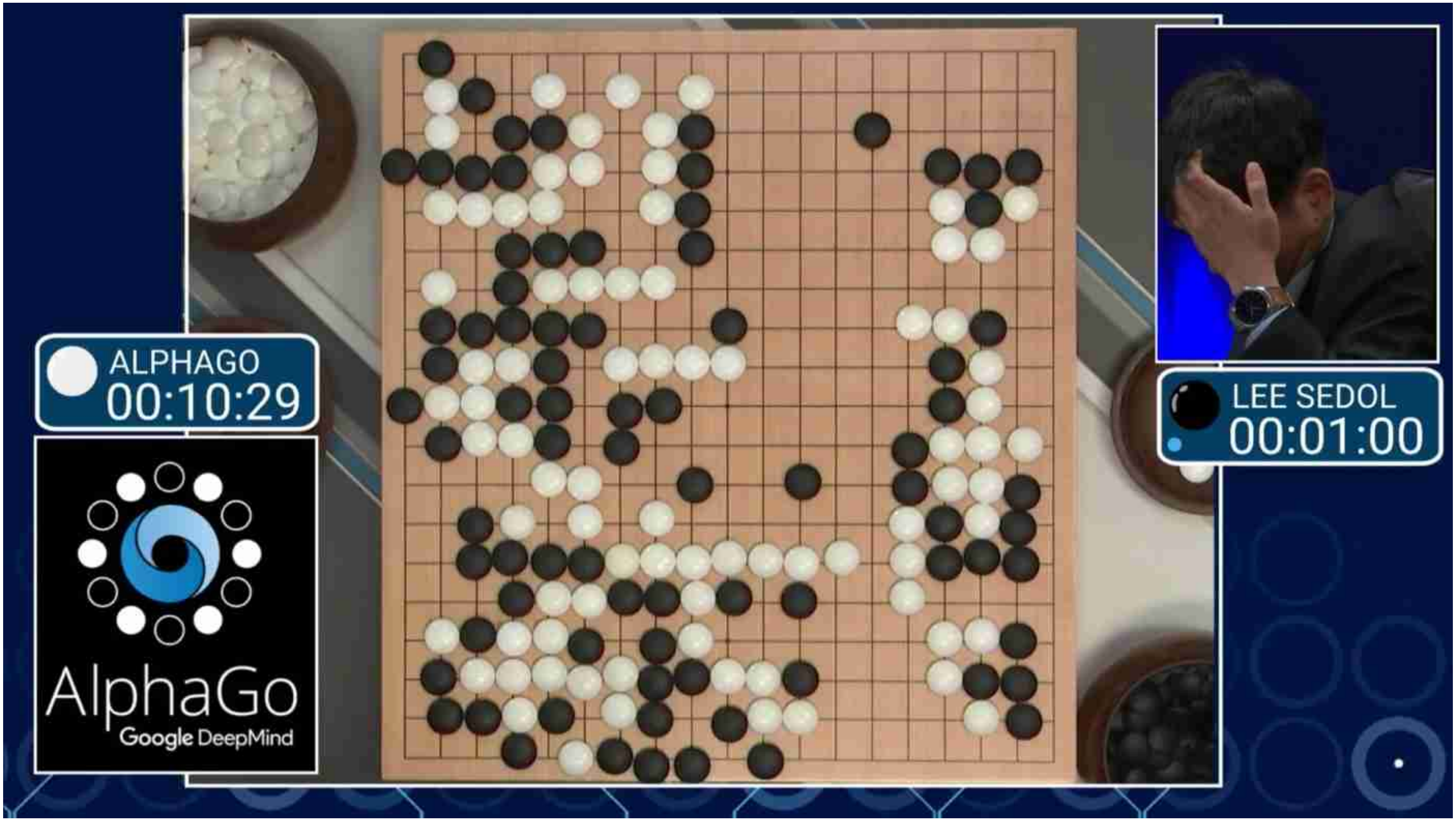
Flying Robots

Figure : Video: https://www.youtube.com/watch?v=YQIMGV5vtd4

Learning algorithms are useful in other tasks

1. Classification: Determine which discrete category the example is
2. Recognizing patterns: Speech Recognition, facial identity, etc
3. Recommender Systems: Noisy data, commercial pay-off (e.g., Amazon, Netflix).
4. Information retrieval: Find documents or images with similar content
5. Computer vision: detection, segmentation, depth estimation, optical flow, etc
6. Robotics: perception, planning, etc
7. Learning to play games

Playing Games: Alpha Go

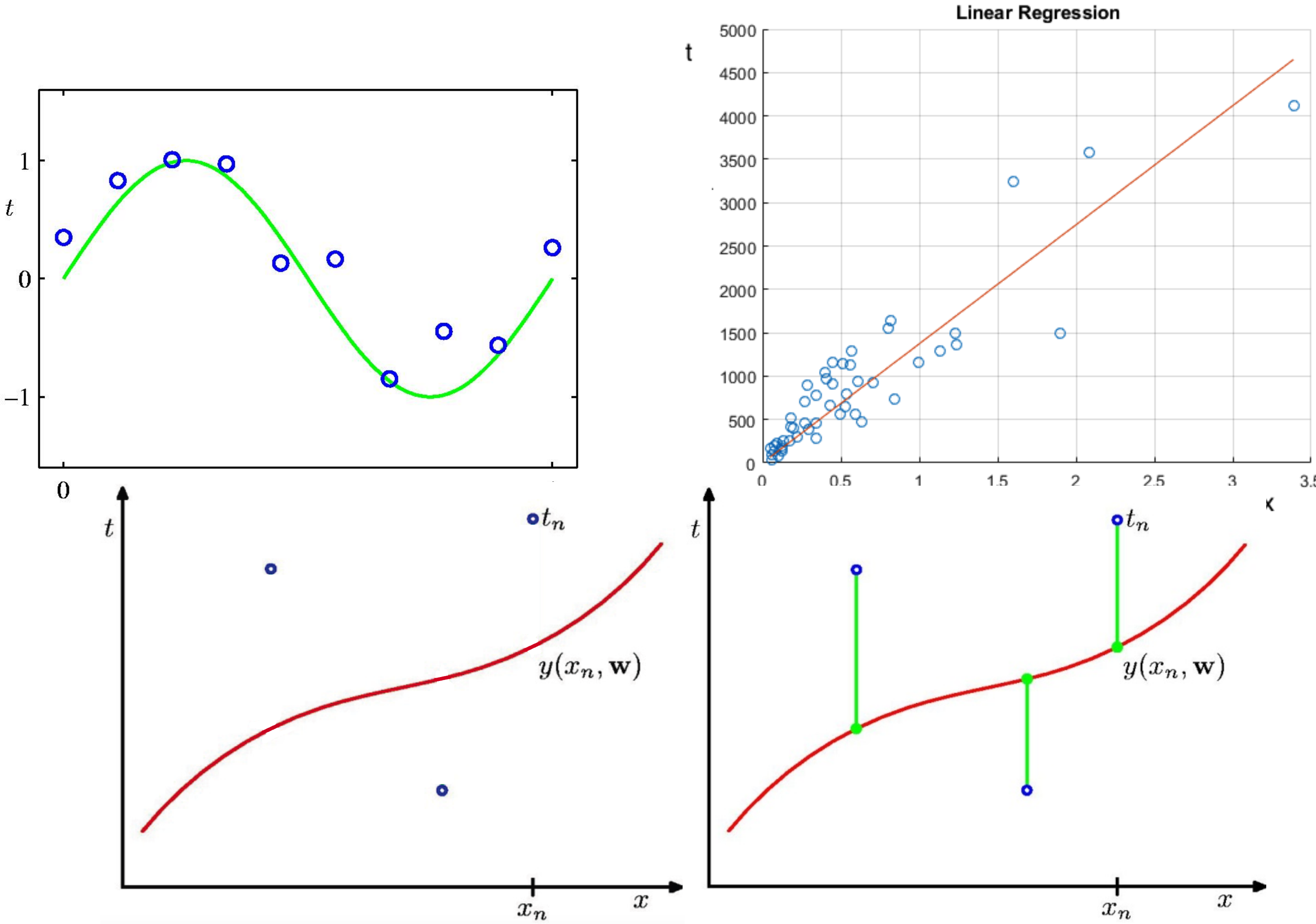


Learning algorithms are useful in other tasks

1. Classification: Determine which discrete category the example is
2. Recognizing patterns: Speech Recognition, facial identity, etc
3. Recommender Systems: Noisy data, commercial pay-off (e.g., Amazon, Netflix).
4. Information retrieval: Find documents or images with similar content
5. Computer vision: detection, segmentation, depth estimation, optical flow, etc
6. Robotics: perception, planning, etc
7. Learning to play games
8. Recognizing anomalies: Unusual sequences of credit card transactions, panic situation at an airport
9. Spam filtering, fraud detection: The enemy adapts so we must adapt too
10. Many more!

Types of learning tasks

- Supervised: correct output known for each training example
  - Learn to predict output when given an input vector
    - Classification: 1-of-N output (speech recognition, object recognition, medical diagnosis)
    - Regression: real-valued output (predicting market prices, customer rating)
- Unsupervised learning
  - Create an internal representation of the input, capturing regularities/structure in data
  - Examples: form clusters; extract features
    - How do we know if a representation is good?
- Reinforcement learning
  - Learn action to maximize payoff
    - Not much information in a payoff signal
    - Payoff is often delayed

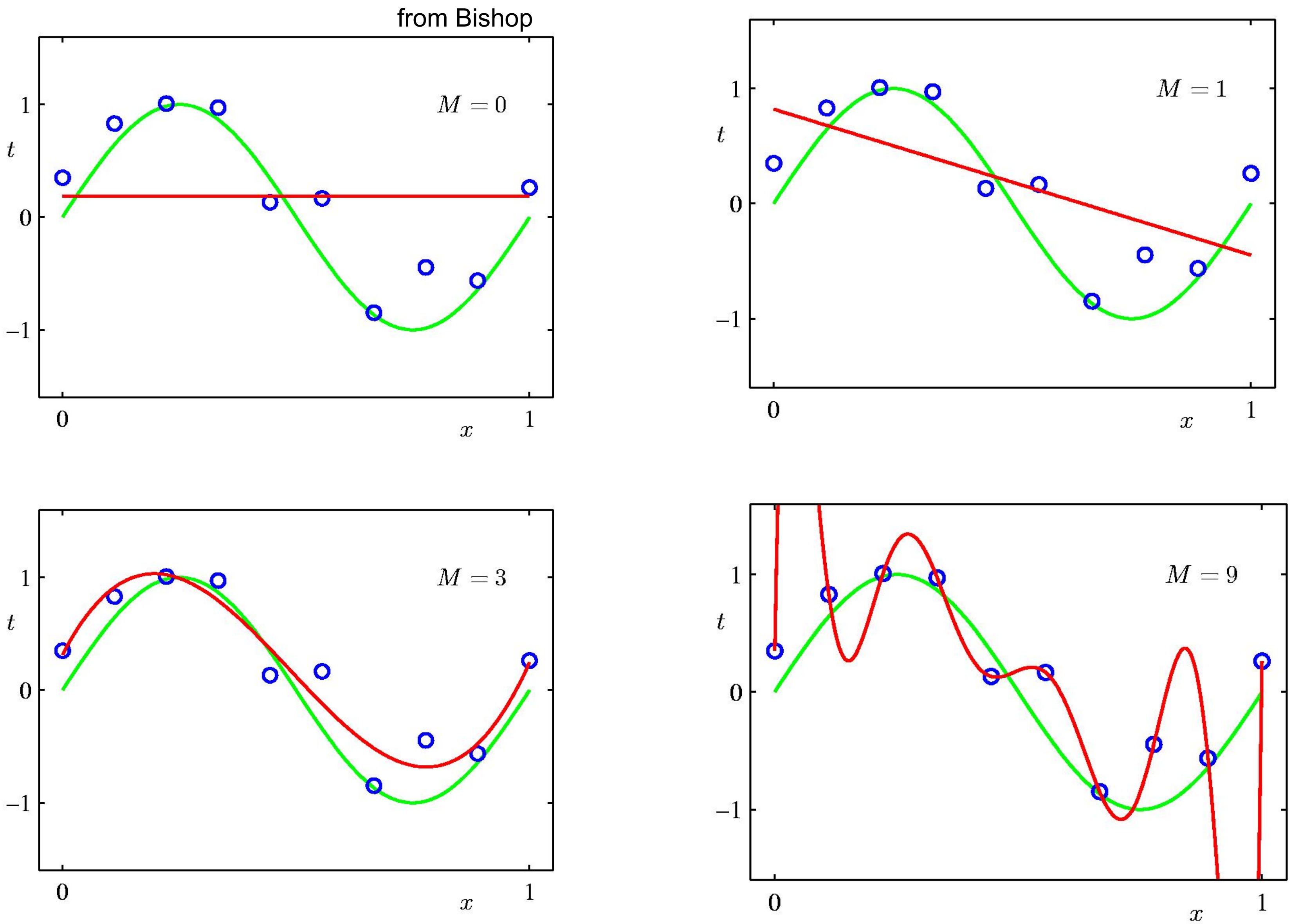


- Define a model

$$y(x) = \text{function}(x, \mathbf{w})$$

Linear: $y(x) = w_0 + w_1 x$

Which Fit is Best?

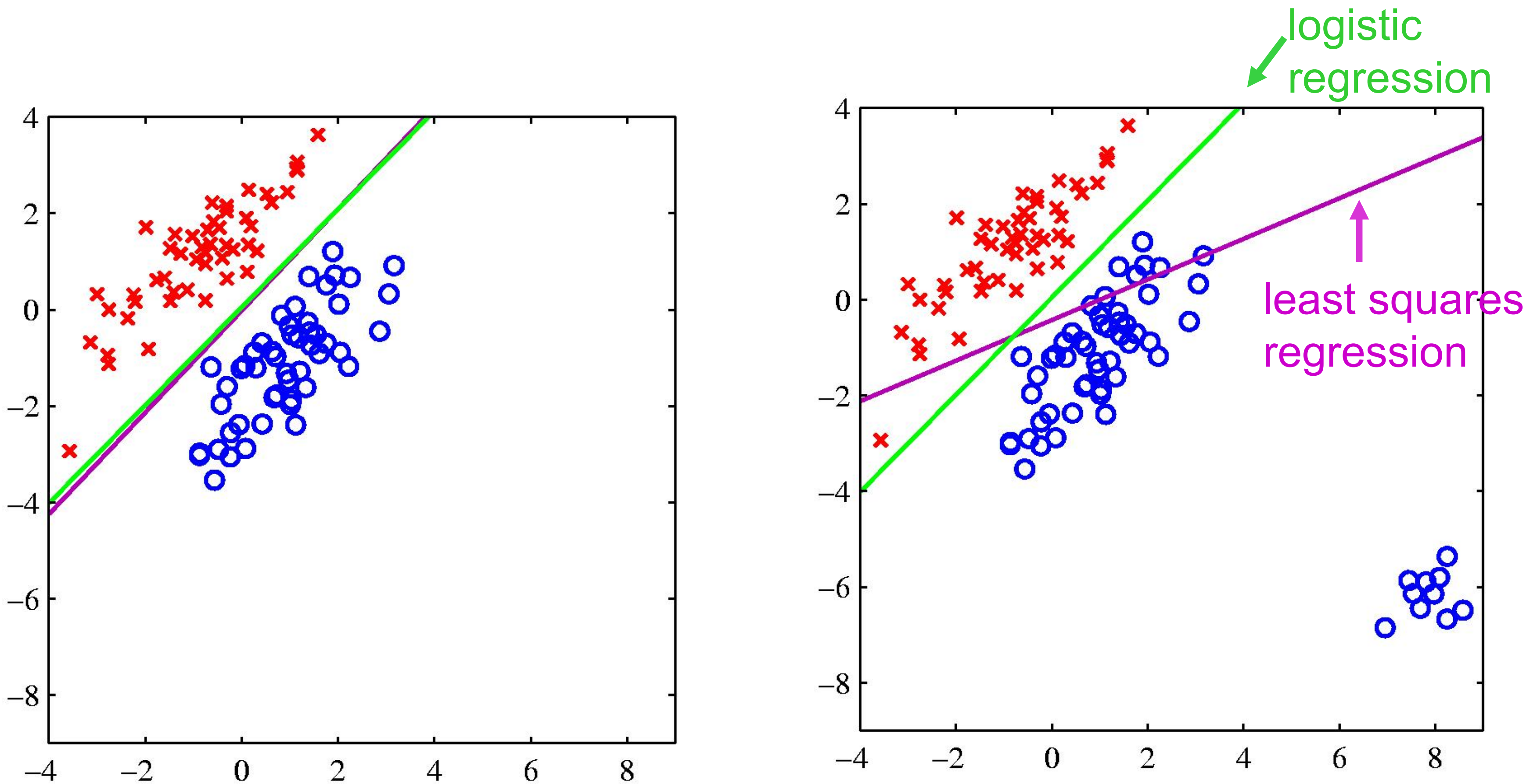


Cross-Validation (with Pictures)

Train your model:

- Leave-one-out cross-validation:
- k-fold cross-validation:

Logistic Regression vs Least Squares Regression



If the right answer is 1 and the model says 1.5, it loses, so it changes the boundary to avoid being "too correct" (tilts away from outliers)

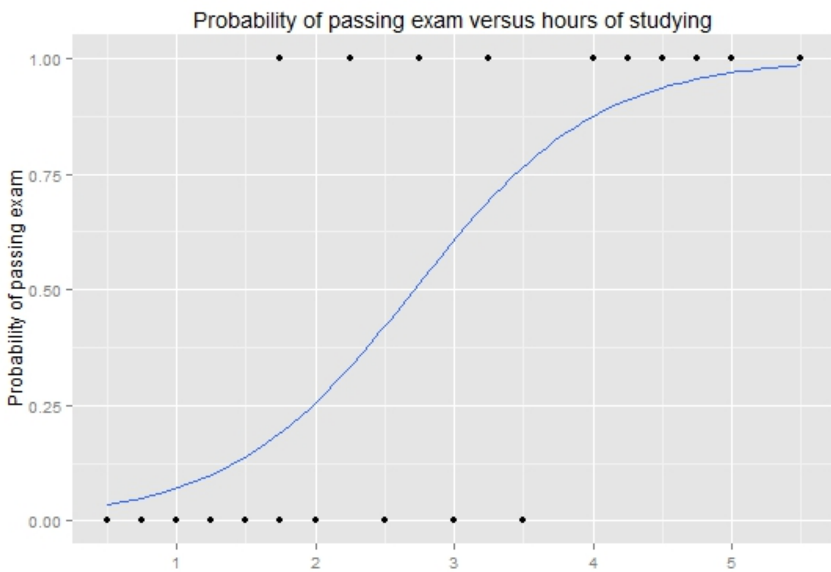
33

Example

- **Problem**: Given the number of hours a student spent learning, will (s)he pass the exam?
- Training data (top row: $x^{(i)}$, bottom row: $t^{(i)}$)

Hours	0.50	0.75	1.00	1.25	1.50	1.75	1.75	2.00	2.25	2.50	2.75	3.00	3.25	3.50	4.00	4.25	4.50	4.75	5.00	5.50
Pass	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1

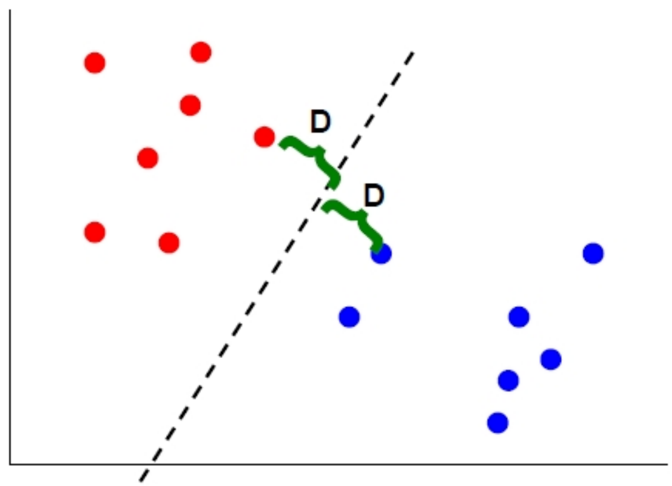
- Learn **w** for our model, i.e., logistic regression (coming up)
- Make predictions:



Hours of study	Probability of passing exam
1	0.07
2	0.26
3	0.61
4	0.87
5	0.97

Max Margin Classification

- Instead of fitting all the points, focus on the boundary points
- Aim: learn a boundary that leads to the largest margin (buffer) from points on both sides



- Why: intuition; theoretical support; and works well in practice
- Subset of vectors that support (determine boundary) are called the support vectors

Metrics

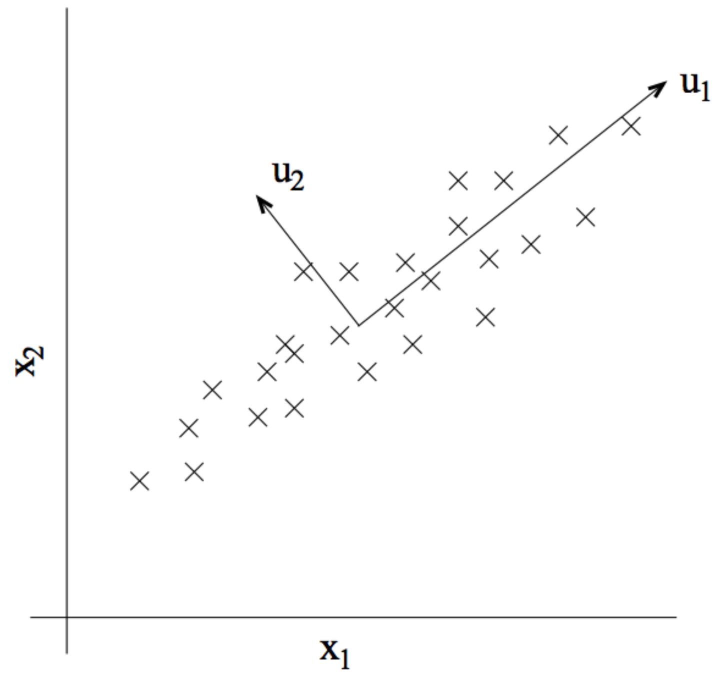
How to evaluate how good my classifier is? How is it doing on dog vs no-dog?

PCA: Common Tool

- Handles high-dimensional data
  - If data has thousands of dimensions, can be difficult for a classifier to deal with
- Often can be described by much lower dimensional representation
- Useful for:
  - Visualization
  - Preprocessing
  - Modeling – prior for new data
  - Compression

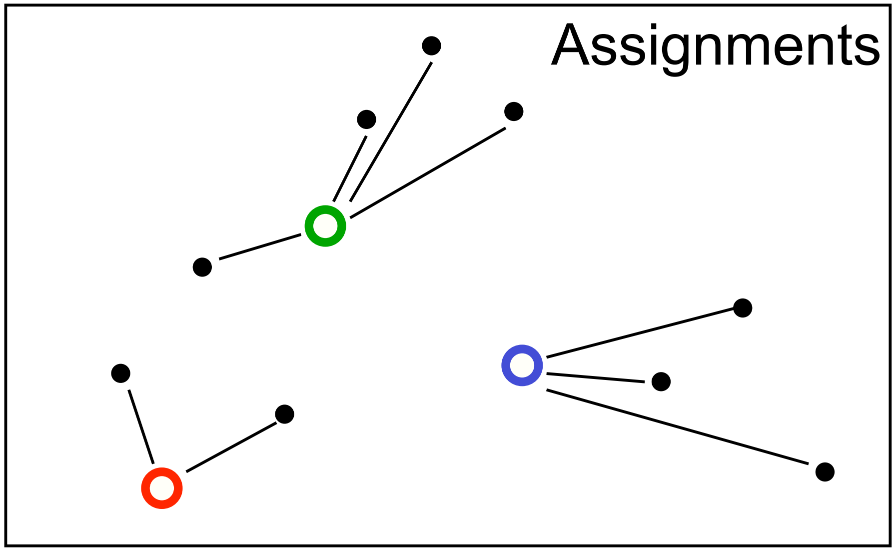
PCA: Intuition

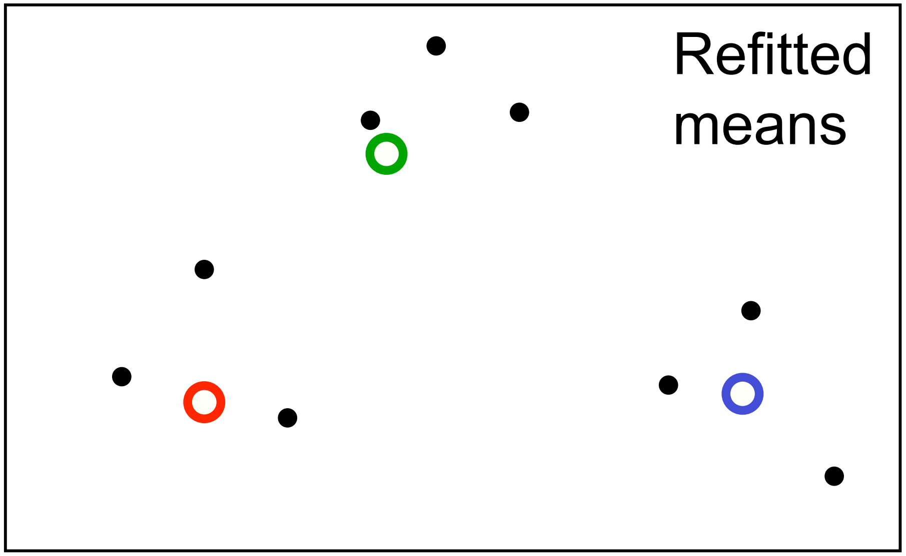
- As in the previous lecture, training data has N vectors, $\{\mathbf{x}_n\}_{n=1}^N$, of dimensionality D, so $\mathbf{x}_i \in \mathbb{R}^D$
- Aim to reduce dimensionality:
  - linearly project to a much lower dimensional space, $M << D$:
$$\mathbf{x} \approx U\mathbf{z} + \mathbf{a}$$
    where U is a $D \times M$ matrix and $\mathbf{z}$ a M-dimensional vector
- Search for orthogonal directions in space with the highest variance
  - project data onto this subspace
- Structure of data vectors is encoded in sample covariance



K-means

- Initialization: randomly initialize cluster centers
- The algorithm iteratively alternates between two steps:
  - Assignment step: Assign each data point to the closest cluster
  - Refitting step: Move each cluster center to the center of gravity of the data assigned to it





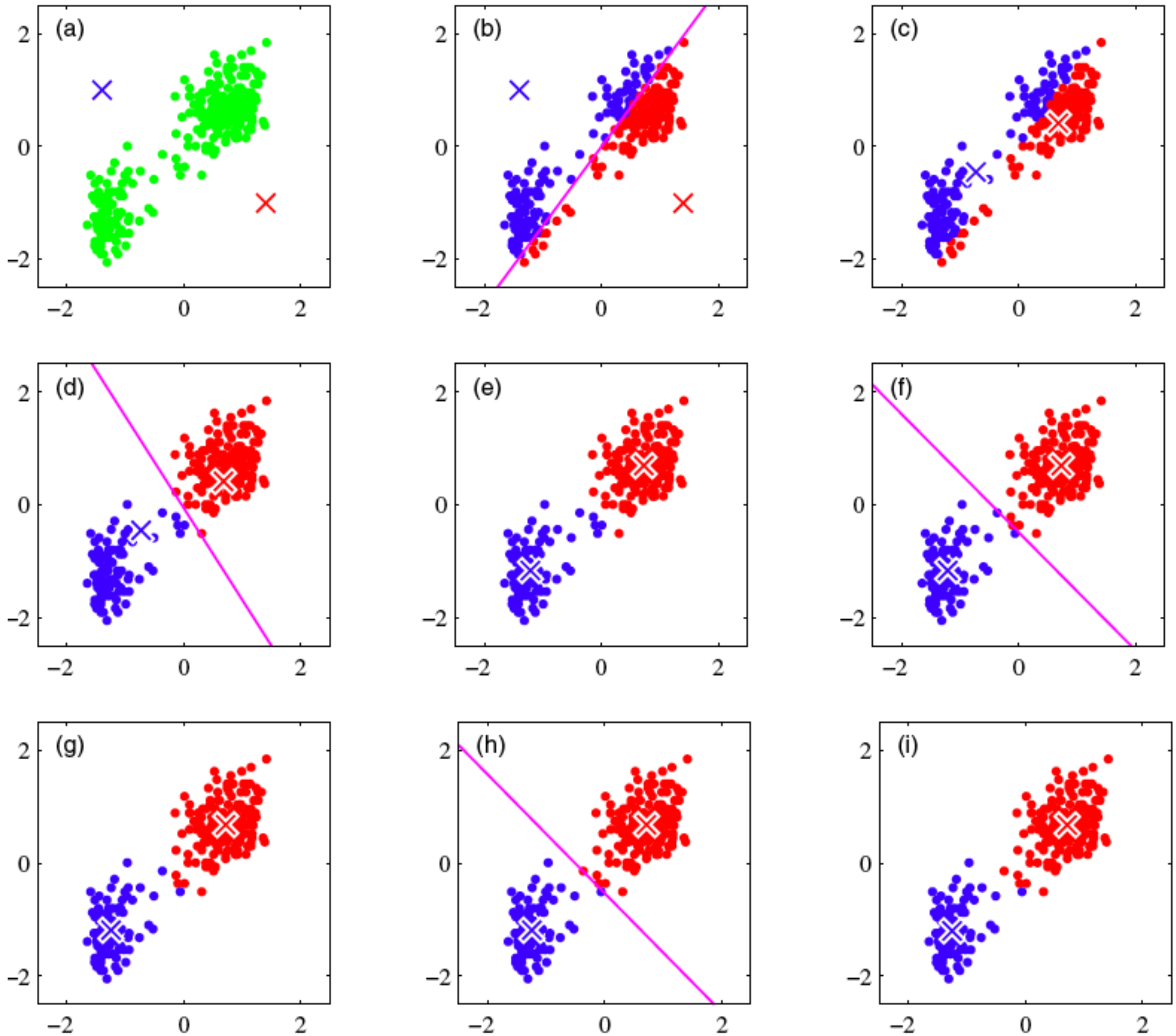


Figure from Bishop

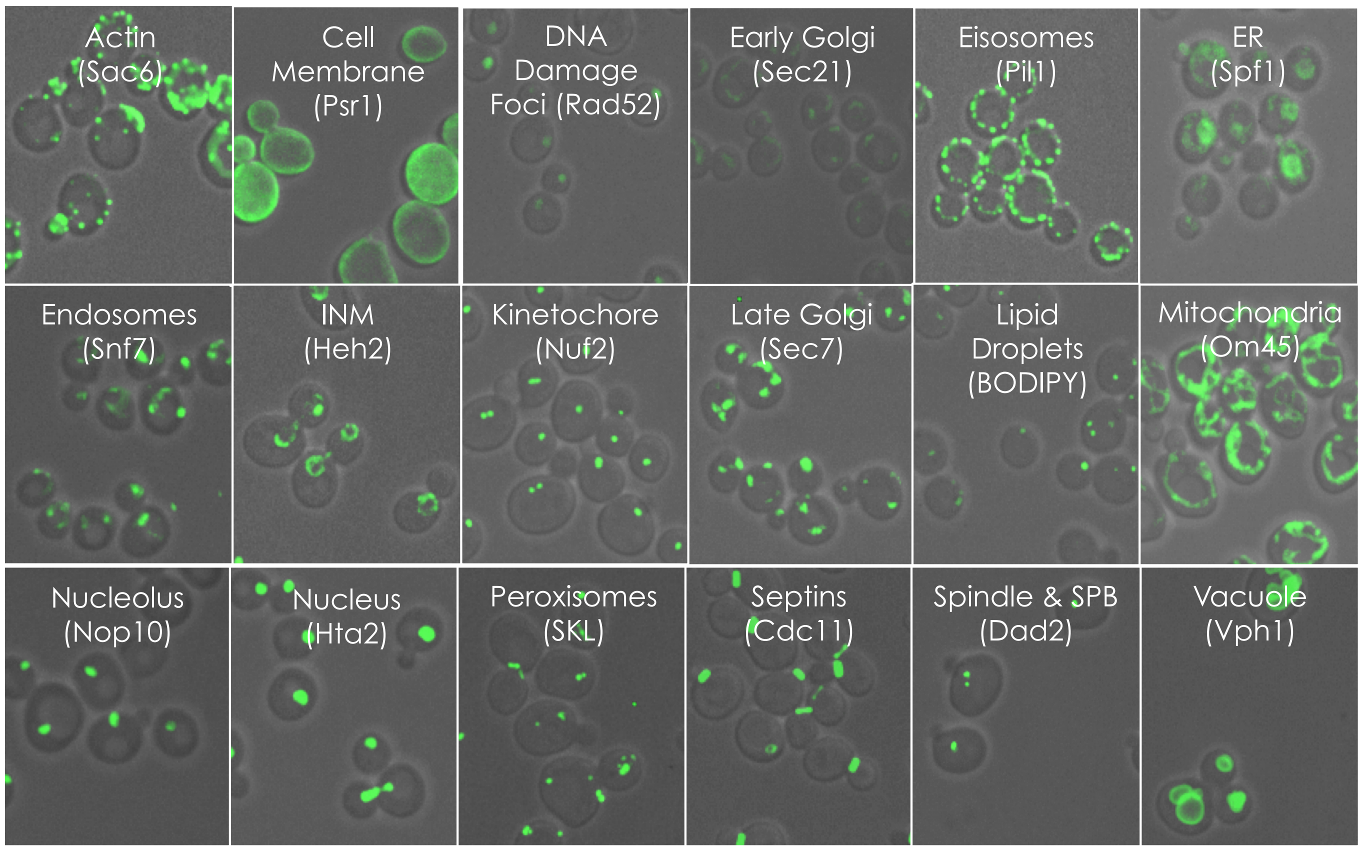
Simple demo: http://syskall.com/kmeans.js/

Molecular Genetics
UNIVERSITY OF TORONTO

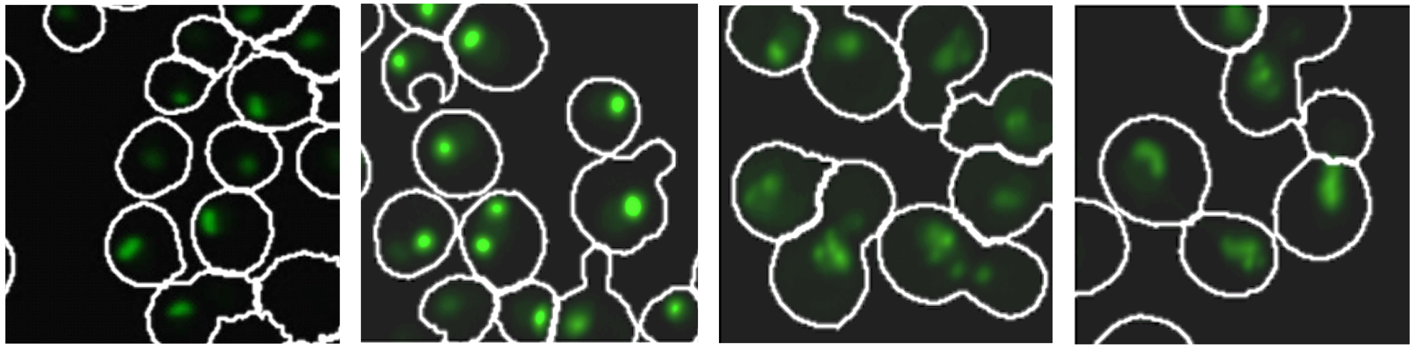
Understanding gene function using ML in yeast

~6000 gene deletions

✖



Organelle morphologies can change with gene deletions



Wild-type Large and bright Fragmented Elongated

Neural Network