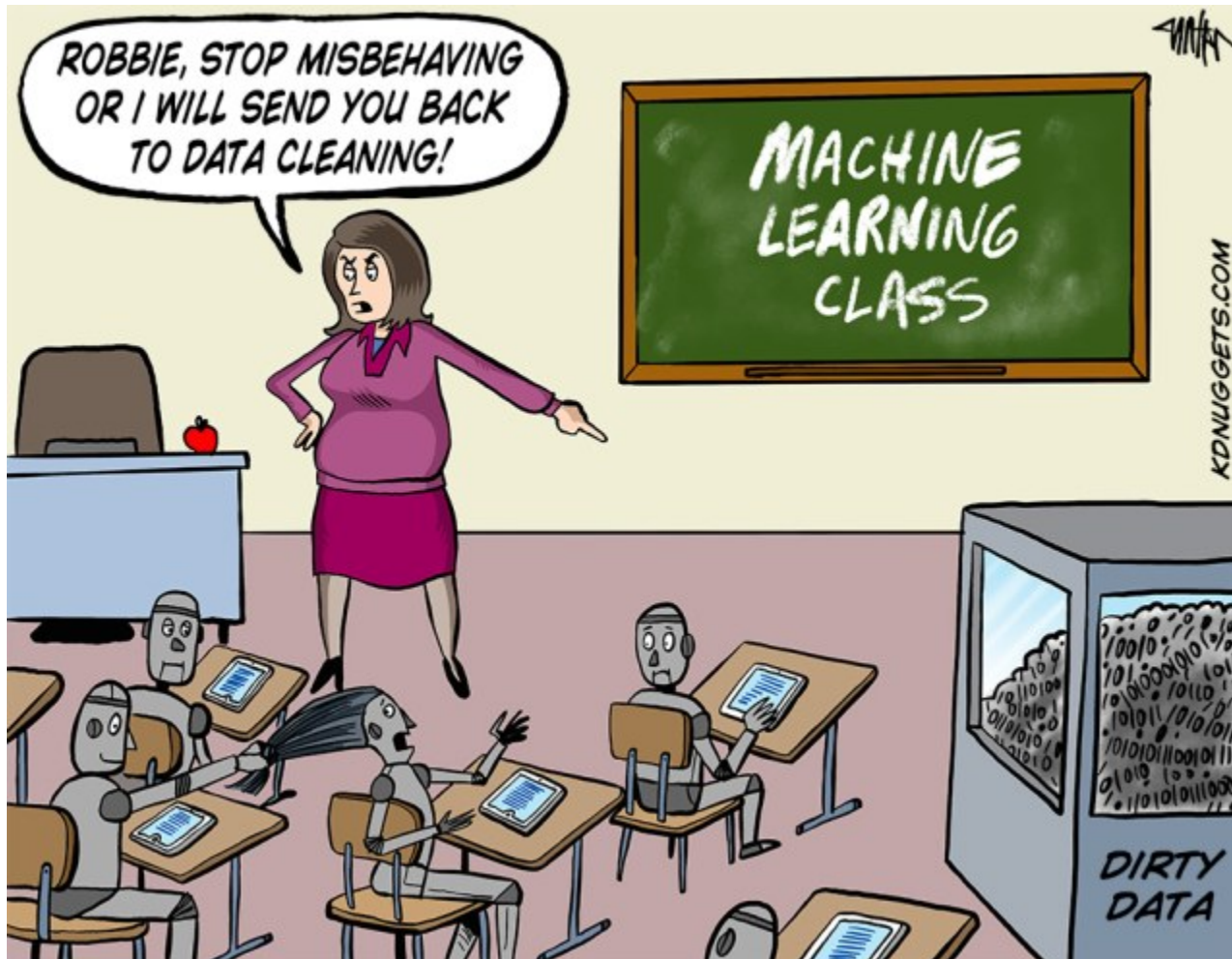
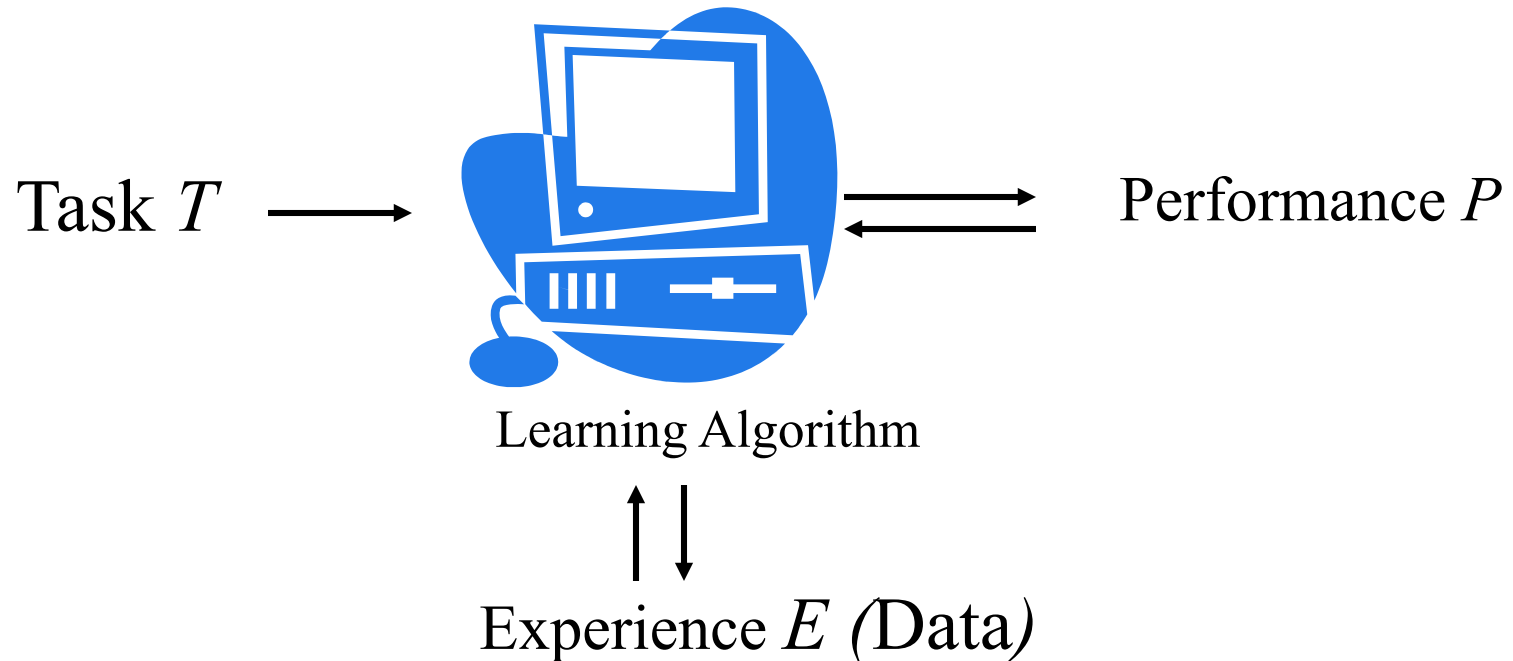


# Welcome to the machine learning class



# What is Machine learning



Machine learning studies algorithms that

- Improve **performance**  $P$
- at some **task**  $T$
- based on **experience**  $E$

Task  $T$



Performance  $P$

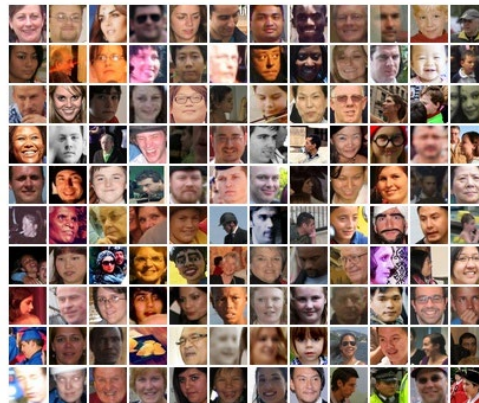
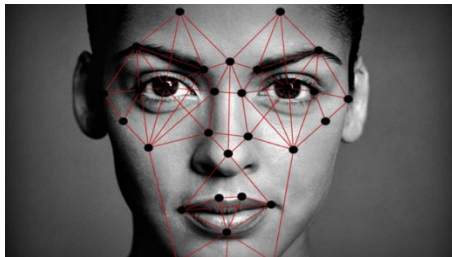
Facial  
recognition

Percentage of correctly  
identified faces

Learning Algorithm



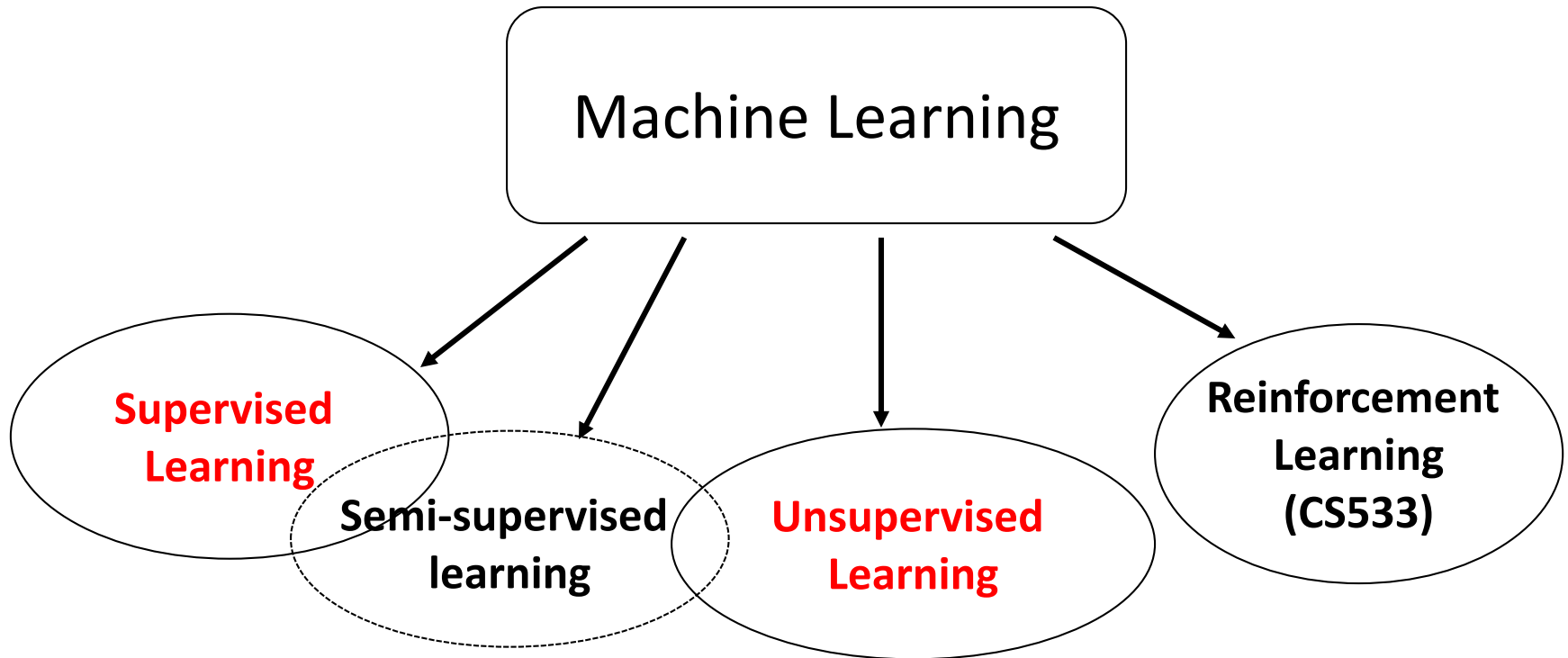
Experience  $E$  (Data)



# Machine learning in Computer Science

- Machine learning is already the preferred approach to
  - Speech recognition
  - Natural language processing
  - Computer vision
  - Robot control
  - Recommender system
  - Precision medicine
  - ....
- This trend is growing with
  - Improved machine learning algorithms
  - Increased data capture, and new sensors
  - Increased computing power
  - Increasing demand for self-customization to user and environment

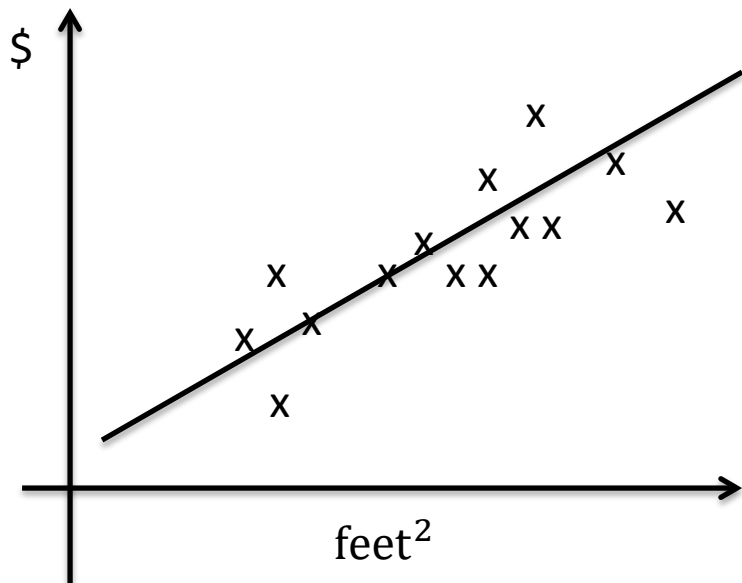
# Basic Topics of Machine Learning



- Deep learning, Active learning, Transfer learning, Learning theory .....
- Many more topics within the broad umbrella of machine learning

# Supervised Learning

- Simply put, learn from supervised training examples to predict certain clearly-defined output ( $y$ ) based on some inputs ( $x$ )
- Output can be
  - **continuous: regression problems**



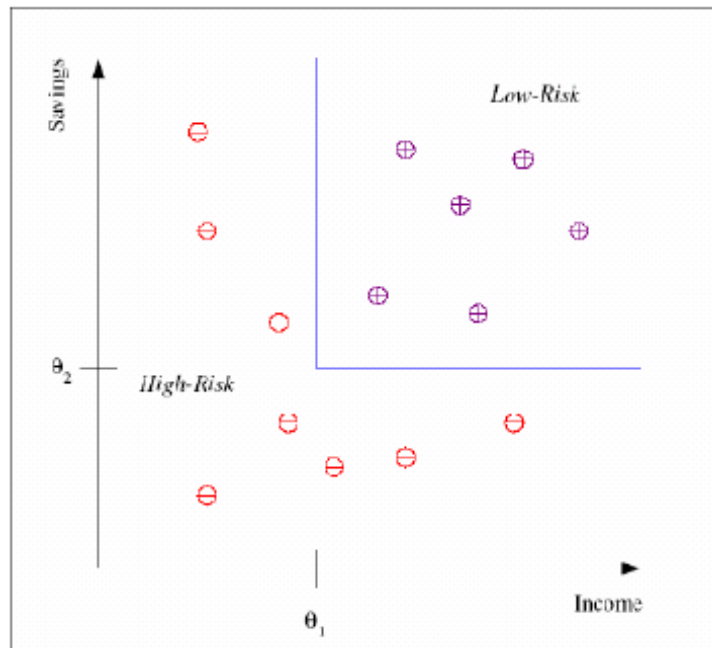
**Example:** Predicting the price of a house based on its square footage

$x \in R$ : square footage

$y \in R$ : price

# Supervised Learning

- Simply put, learn from supervised training examples to predict certain clearly-defined output ( $y$ ) based on some inputs ( $x$ )
- Output can be
  - continuous: regression problems
  - **Discrete: classification problems**



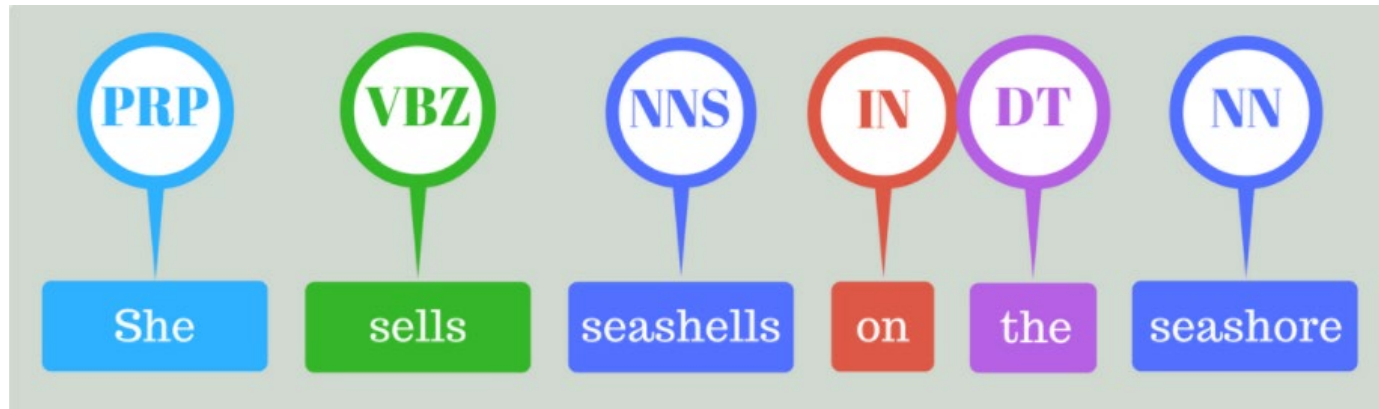
**Example:** classify a loan applicant as either high risk or low risk based on income and saving amount.

$x \in R^2$ : [income, saving]  
 $y \in \{\text{high-risk, low-risk}\}$

# Supervised Learning

- Output can be
  - continuous: regression problems
  - Discrete: classification problems
  - **Structured: structured prediction problems**

Example: part of speech tagging



Input  $\mathbf{x}$  is a sentence, output  $\mathbf{y}$  is a sequence of tags

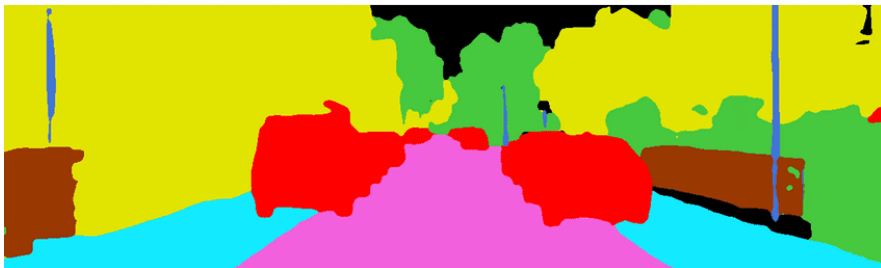


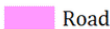
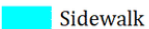
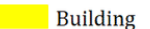
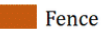

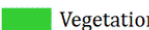
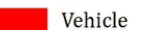
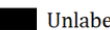
# Supervised Learning

- Output can be
  - continuous: regression problems
  - Discrete: classification problems
  - **Structured: structured prediction problems**



Example: semantic segmentation

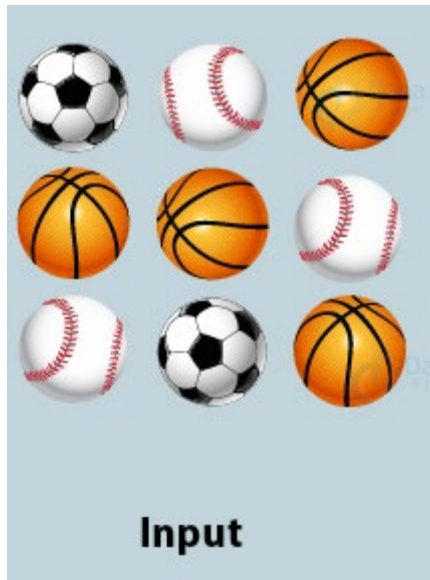


 Road	 Sidewalk	 Building	 Fence
 Pole	 Vegetation	 Vehicle	 Unlabel

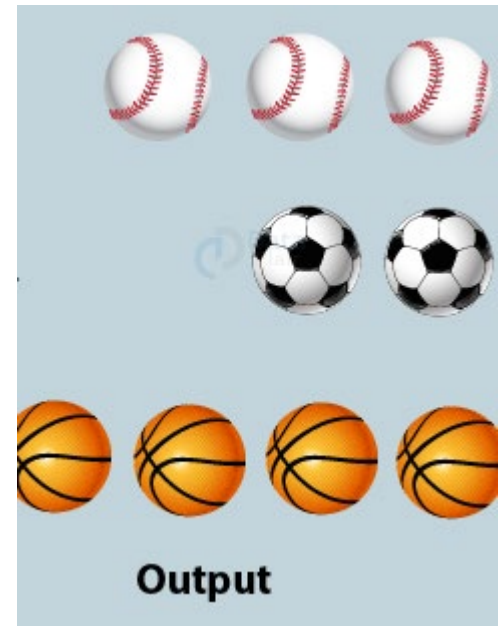
Input  $\mathbf{x}$  is an natural image  
output  $\mathbf{y}$  is a segmentation of  
the image into semantic classes

# Unsupervised Learning: clustering

- Given a collection of examples (objects), discover self-similar groups within the data

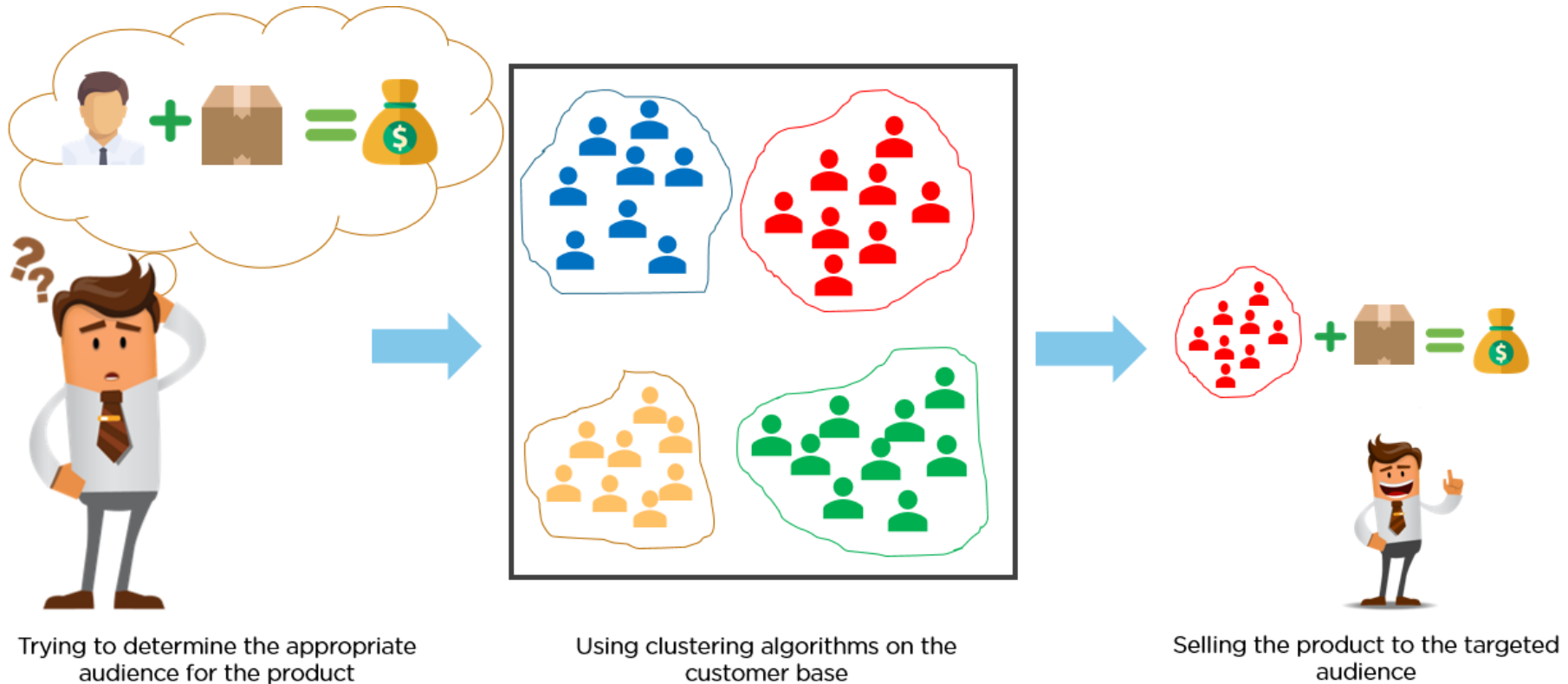


clustering



# Unsupervised Learning: clustering

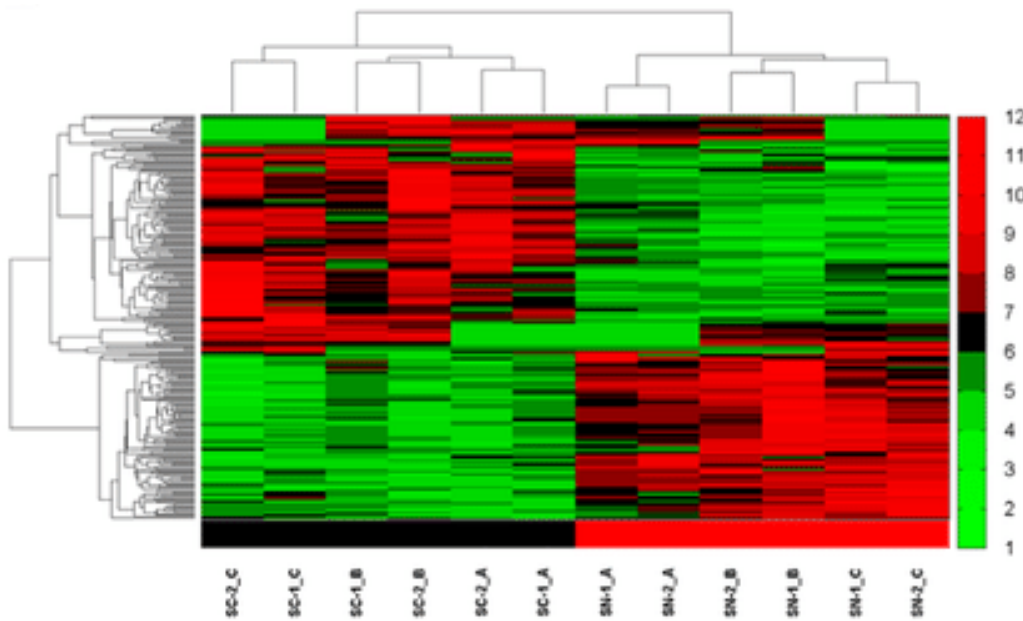
- Given a collection of examples (objects), discover self-similar groups within the data



Tremendous commercial usage

# Unsupervised Learning: clustering

- Given a collection of examples (objects), discover self-similar groups within the data



Clustering the gene/protein expression data

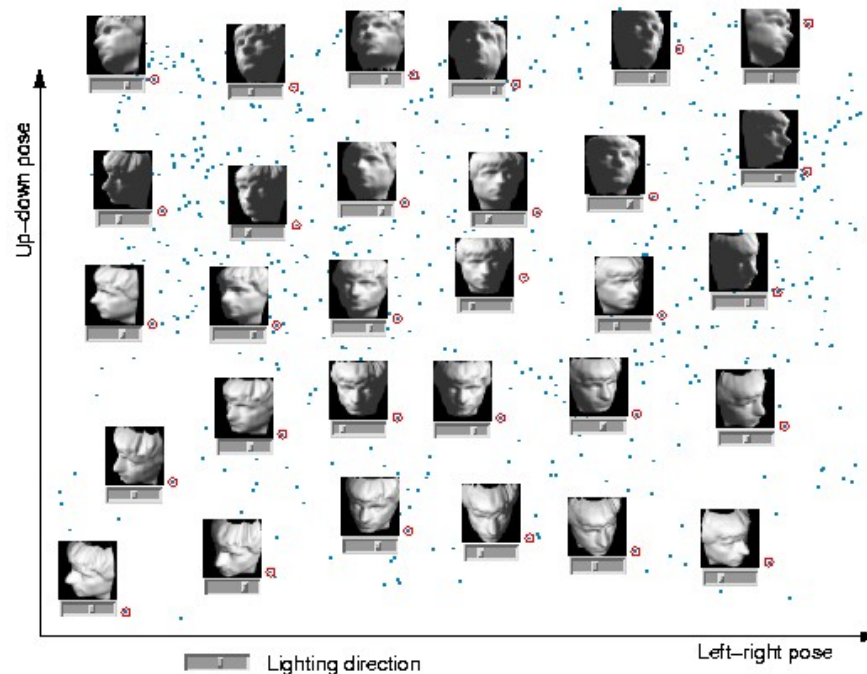
Great tool for scientific discovery

# Unsupervised Learning: density estimation

- Learn the underlying distribution or model that generates the data we observe – **density estimation or generative models**
  - So that we can recognize when something comes from a different distribution – anomaly detection
  - So that we can generate new data from the same distribution – synthetic voice, image and text generation ...

# Unsupervised Learning: dimension reduction

- Represent high dimensional data using a low-dimensional representation for compression or visualization – **dimension reduction**



# Reinforcement Learning

- Learn to act
- An agent
  - Observes the environment
  - Takes action
  - With each action, receives rewards/punishments
  - Goal: learn a policy that optimizes rewards
- No examples of optimal outputs are given
- Not covered in this class. Take AI533 if you want to learn about this.

# A little thought exercise

What type of learning is appropriate for the following problems.

1. Given a set of students and their schedules, assign students into study groups such that each study group has the most scheduling flexibility
2. Given a set of endangered species and map out where each of the species can live based on past observations of presence/absence of different species in different habitats
3. Given a neighborhood served by a power grid, figure out the best strategy for the timing of flexible power usage (e.g., charging of electrical car)



# When do we need computer to learn?

©2001 Shannon Burns

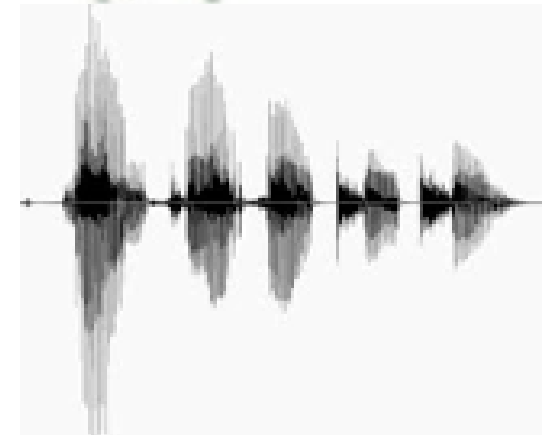
www.shannonburns.com



*Do we need learning to do tax return?*

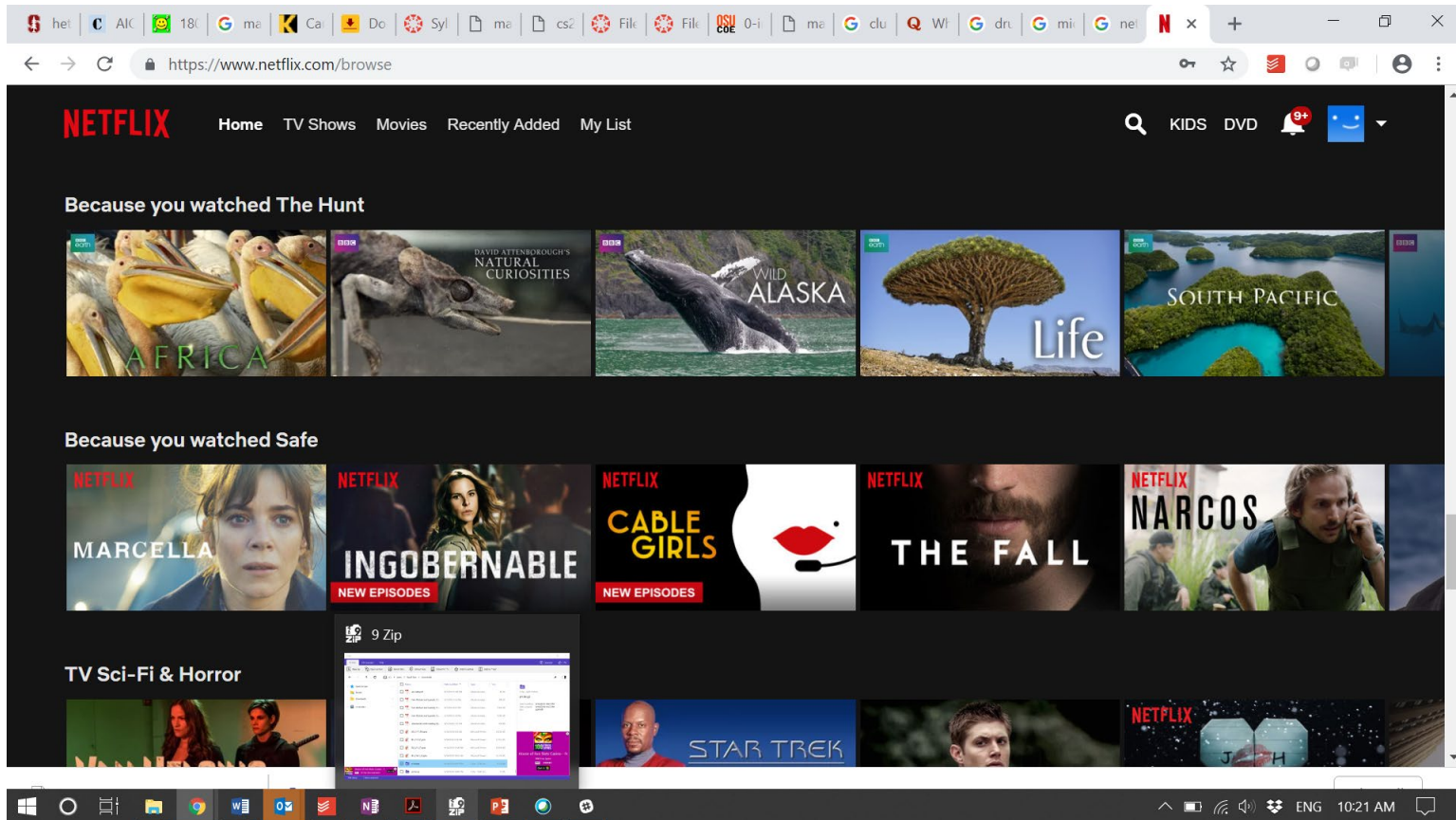
# Appropriate Applications for Supervised Learning

- Situations where humans can perform the task but can't describe how they do it



# Appropriate Applications for Supervised Learning

- Situations where the desired function is different for each individual



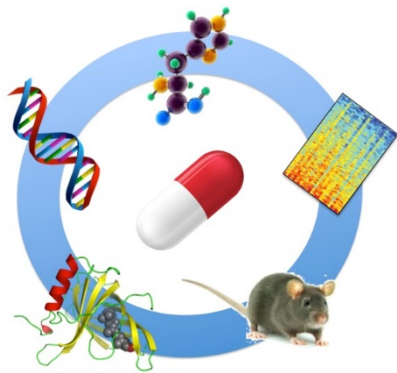
# Appropriate Applications for Supervised Learning

Situations where the desired function is changing frequently



# Appropriate Applications for Supervised Learning

- Situations where human experts do not have sufficient knowledge and need help



## Drug discovery

Based on the molecular structure  
to predict the effectiveness of drug



## Material discovery

Use chemical elements of a crystal  
to predict material properties

# Supervised learning (basic setup)

- Given: a set of **training examples**

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \dots, (\mathbf{x}_N, y_N)$$

- $\mathbf{x}_i$ : the input of the  $i$ -th example, we assume data is vectorized, i.e.,  $\mathbf{x}_i \in R^d$ , a  $d$ -dimensional vector
  - $y_i$  is its corresponding output (continuous or discrete)
  - $N$ : the total number of training examples
  - We assume there is some underlying function  $f$  that maps from  $\mathbf{x}$  to  $y$  – our **target function**
- Goal: find a **good approximation of  $f$**  so that an accurate prediction can be made for previously unseen  $\mathbf{x}$ 
    - **Generalization**

# Key Components of Machine learning

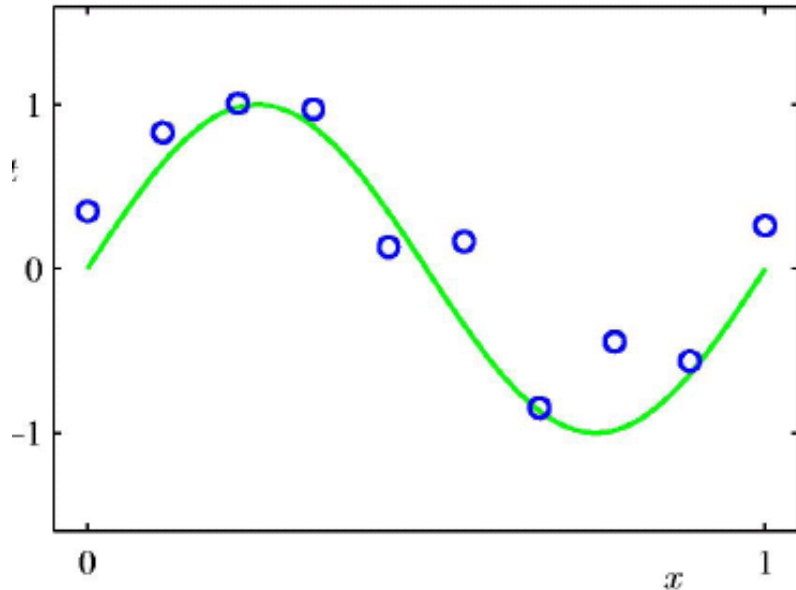
- Data representation
  - How do we represent our input? What features do we use? Dimension reduction? Normalization?
  - **Design of the features** --- less critical for some applications with deep learning
- Function representation
  - How do we represent this function  $f$  we are trying to learn? Linear, polynomial, tree, neural networks, graphical model, set of rules ....
  - **Design of the hypothesis space**
- Objective for learning
  - What is our goal of learning? How to quantify it? Accuracy? Precision/recall? likelihood, cost? avoid overfitting via regularization?
  - **Designing of the loss function (and evaluation criterion)**
- Optimization
  - How do we optimize the objective? Search? Combinatorial optimization? Convex optimization? Constrained optimization? Backpropagation?
  - **Design of the optimization algorithm**

# A Practitioner's ML Loop (for supervised learning)

- Set up a supervised learning problem
- Data collection of input-output pairs where we know the output is right
- Representation: Choose how to represent the data
- Modelling: Choose a hypothesis space / model class
- Learning/Training/Estimation: Find the best member of the chosen hypothesis class via optimization
- Model Selection: Try different models and evaluate them on held out data (more on this later)
- If happy, stop. Else repeat/refine one or more of the above.



# A toy example: regression



The true underlying function:

$$y = \sin(2\pi x) + \epsilon$$

where  $\epsilon$  is some added observation noise (Gaussian)

- Green line shows **the true underlying function** (without the noise)
- **Training examples** are shown as blue circles (with added Gaussian noise)
- Goal of Learning: make **accurate** prediction of the  $y$  value for some **new values** of  $x$

# Polynomial curve fitting

- There are infinite # of functions that will fit the training data perfectly.
- In order to learn, we have to make some assumptions about our function, specifically by assuming the function belongs to a limited class
  - We call this our *Hypothesis Space*
  - E.g., all M-th order polynomial functions

$$y(x, \mathbf{w}) = w_0 + w_1x + w_2x^2 + \dots + w_Mx^M$$

- $\mathbf{w} = (w_0, w_1, \dots, w_M)$  represents the unknown parameters that we wish to learn from the training data

# Polynomial curve fitting

- Learning here means to find a good set of parameters to minimize a **loss function**, which measures how well the function fit the training examples
- For example:

Given a set of training examples

$$(x_1, y_1), (x_2, y_2) \dots (x_N, y_N)$$

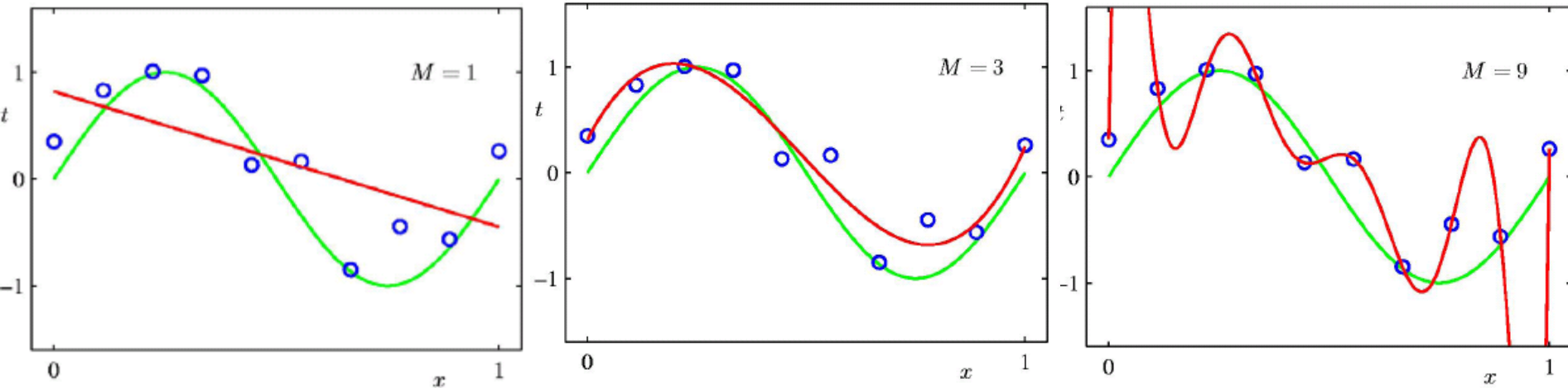
A commonly used loss function is:

$$L(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^n (y(x_i, \mathbf{w}) - y_i)^2$$

Sum of Squared Errors

- Learning is then formulated as an optimization problem to find the optimal  $\mathbf{w}$  that minimize the loss function

# Important Issue: Model Selection



- The red line shows the function learned with different  $M$  values
- Which  $M$  should we choose? – this is a **model selection** problem
- Can we use  $L(\mathbf{w})$  that we define in previous slides as a criterion to choose  $M$ ?

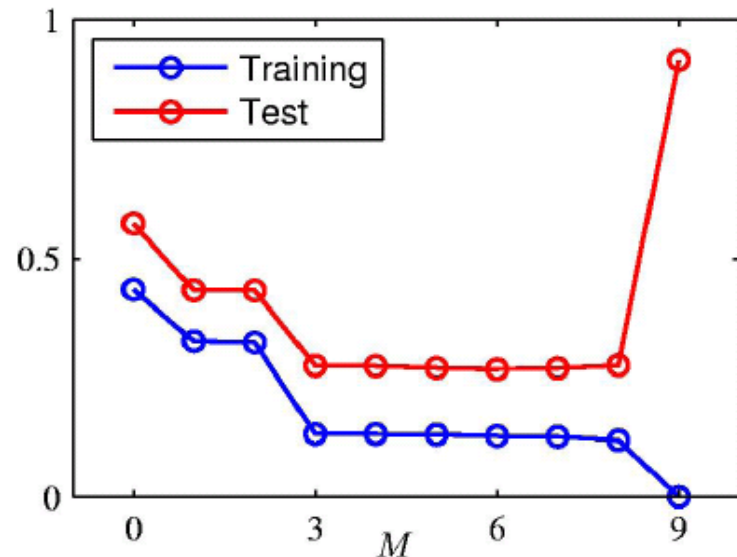
$$L(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N (y(x_i, \mathbf{w}) - y_i)^2$$

Sum of Squared Errors

# Over-fitting

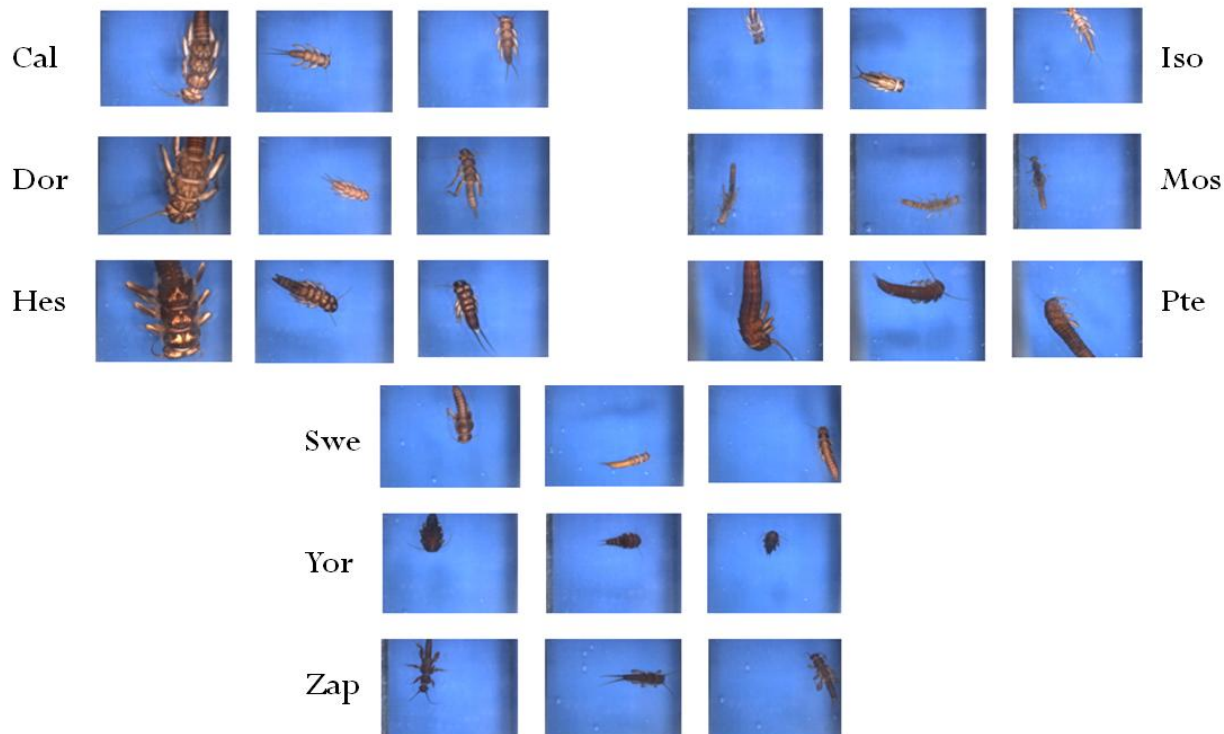
- As  $M$  increases, SSE on the training data decreases monotonically
- However, the SSE on test data starts to increase after a while
  - Why? Is this a fluke or generally true?

It turns out this is generally the case – caused by **over-fitting**



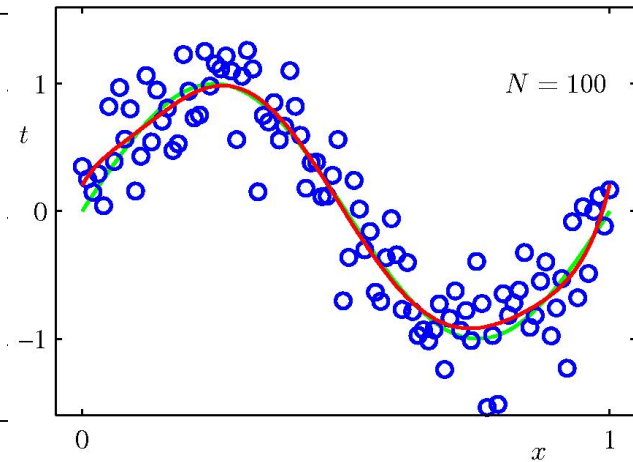
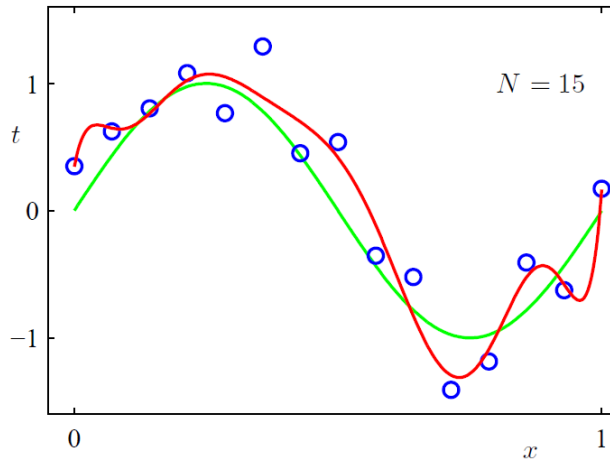
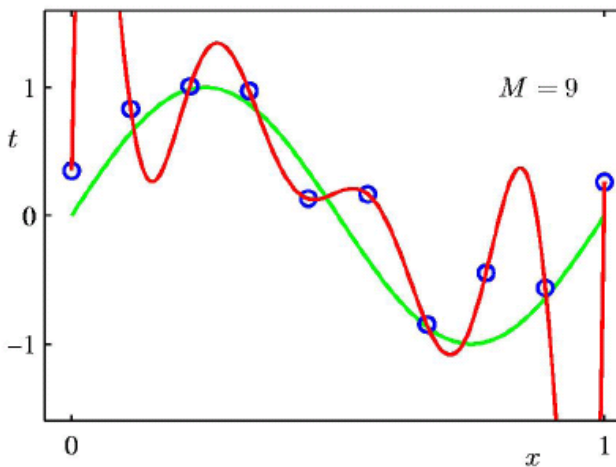
# Over-fitting

- Over-fitting refers to the phenomenon when the learner adjusts to some random signals in the training data that is not relevant to the target function
- Real story from bugID project



# Overfitting

- Over-fitting happens when
  - There is too little data (or some systematic bias in the data )
  - There are too many parameters



How do we deal with this issue? A core theme for many lectures to come.

# Key Issues in Machine Learning

- What are good **hypothesis spaces**?
  - Linear functions? Polynomials?
  - which spaces have been useful in practical applications?
- How to select among different hypothesis spaces?
  - The **Model selection** problem
  - Trade-off between over-fitting and under-fitting
- How can we optimize accuracy on future data points?
  - This is called the **Generalization Error** – error on unseen data pts
  - Related to the issue of “overfitting”, i.e., the model fitting to the peculiarities rather than the generalities of the data
- What level of confidence should we have in the results? (A statistical question)
  - How much training data is required to find an accurate hypotheses with high probability? This is the topic of learning theory
- Are some learning problems computationally intractable? (A computational question)
  - Some learning problems are provably hard
  - Heuristic / greedy approaches are often used when this is the case
- How can we formulate application problems as machine learning problems? (the engineering question)



# Road map for the next few weeks

- Linear regression
  - linear models for continuous target variables
- Linear classification models
  - Logistic regression
  - Naïve bayes
  - Perceptron
  - Linear support vector machines

} Maximum likelihood estimation  
with probabilistic objectives

} Optimizing convex  
Loss functions
- Nonlinear classification models
  - Kernel SVM
  - Decision trees
  - Neural networks