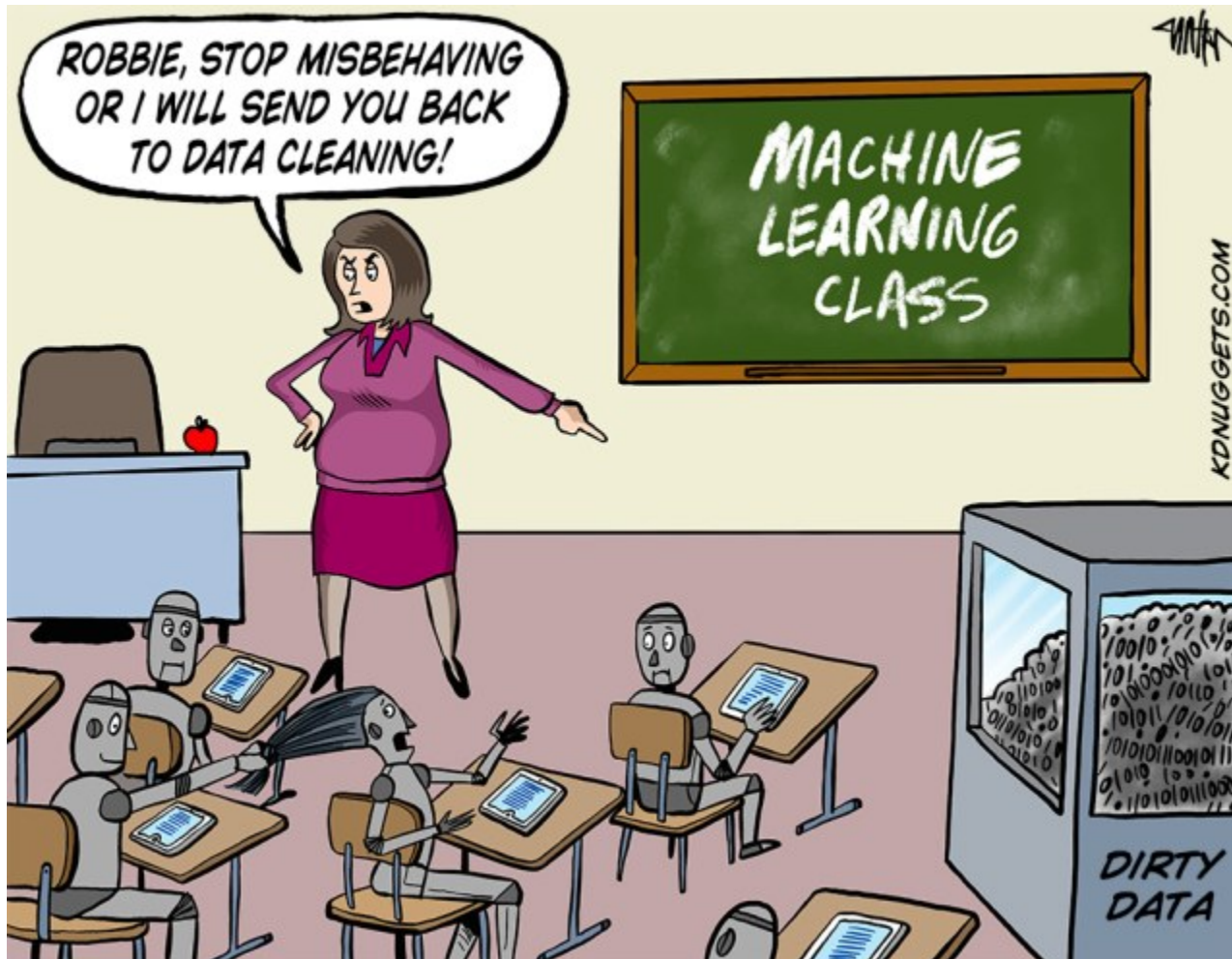
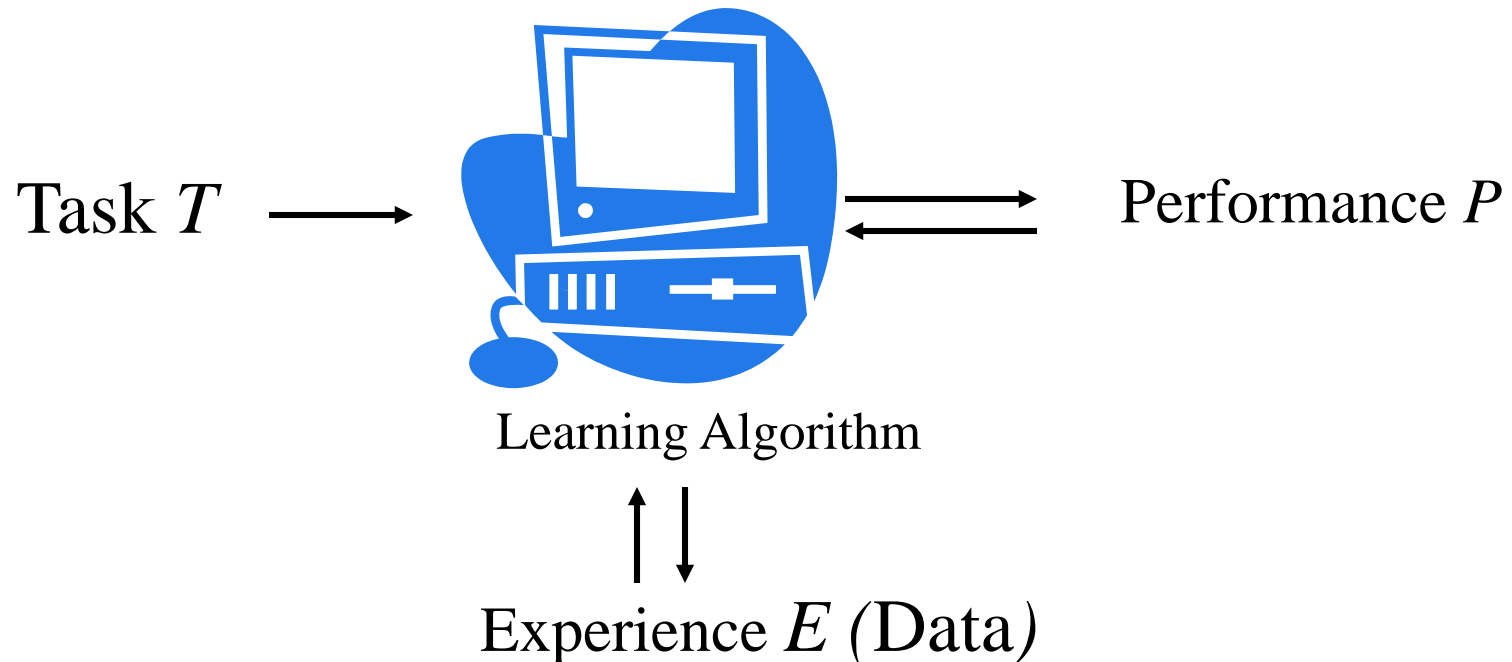


# Welcome to the machine learning class



# What is Machine learning



Machine learning studies algorithms that

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

Task  $T$



Performance  $P$

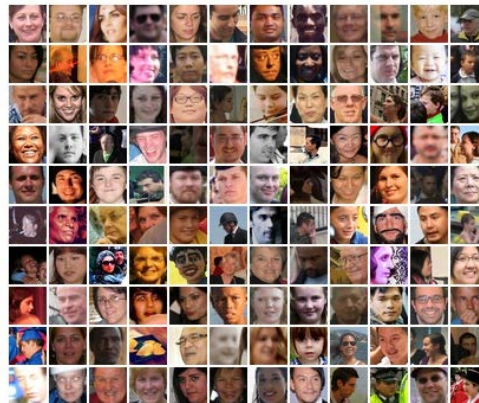
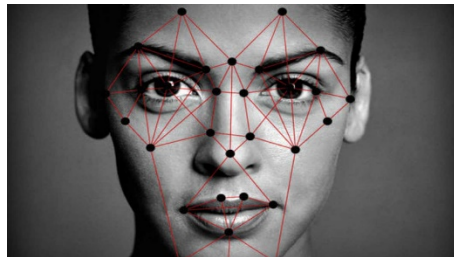
Facial  
recognition

Prediction  
accuracy

Learning Algorithm



Experience  $E$  (Data)

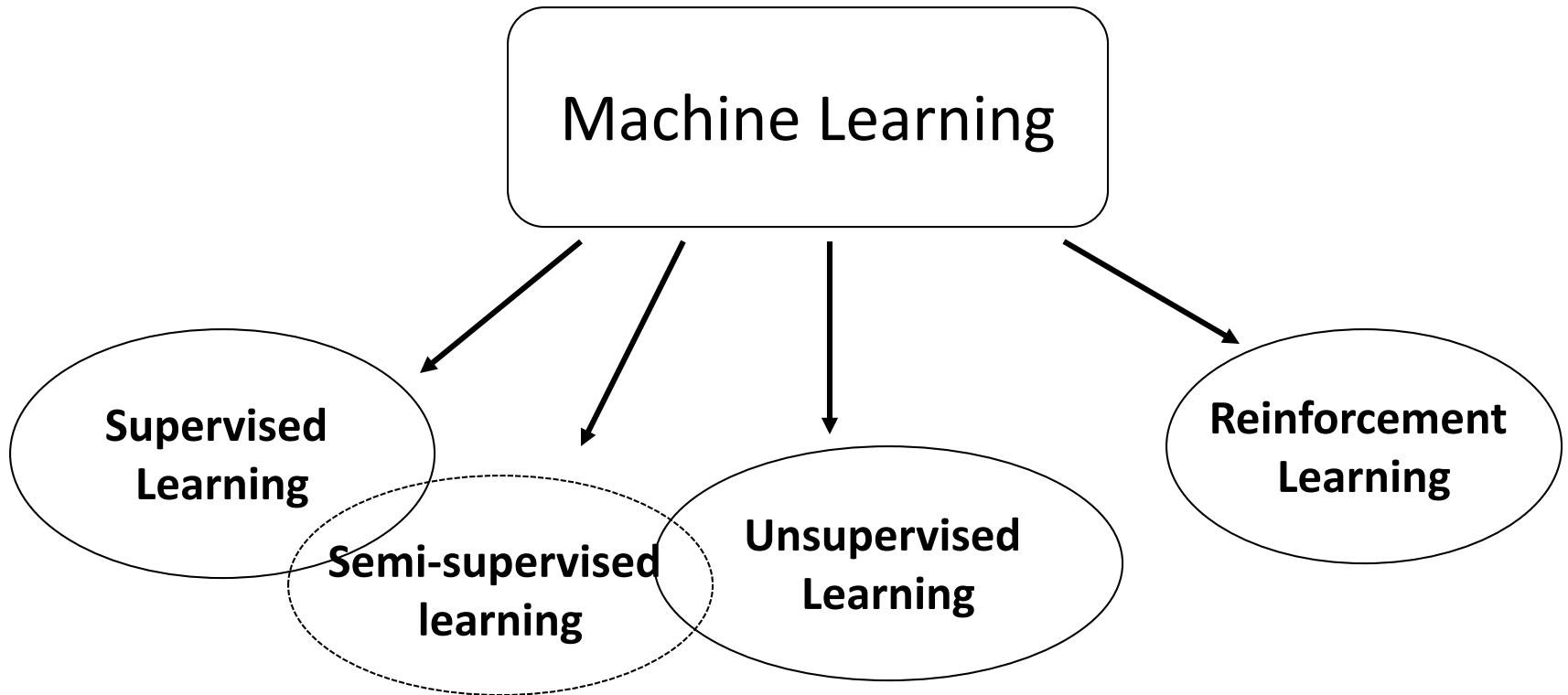


by aXo  
Ha Minh Lu  
15 Sep 2017

# Machine learning in Computer Science

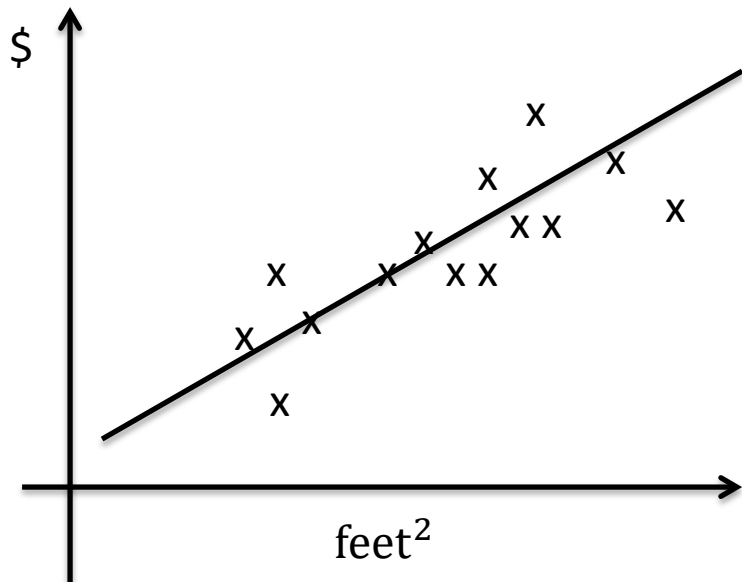
- Machine learning is already the preferred approach to
  - Speech recognition
  - Natural language processing
  - Computer vision
  - Robot control
  - Recommender system
  - Precision medicine
  - ....
- This trend is growing
  - Improved machine learning algorithms
  - Increased data capture, and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

# Topics



# Supervised Learning

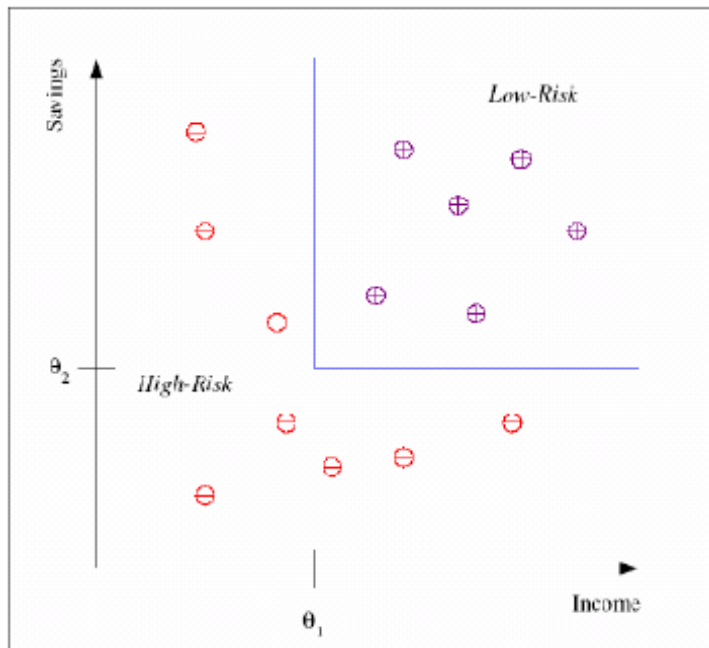
- Learn to predict output from input.
- Output can be
  - **continuous: regression problems**



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

# Supervised Learning

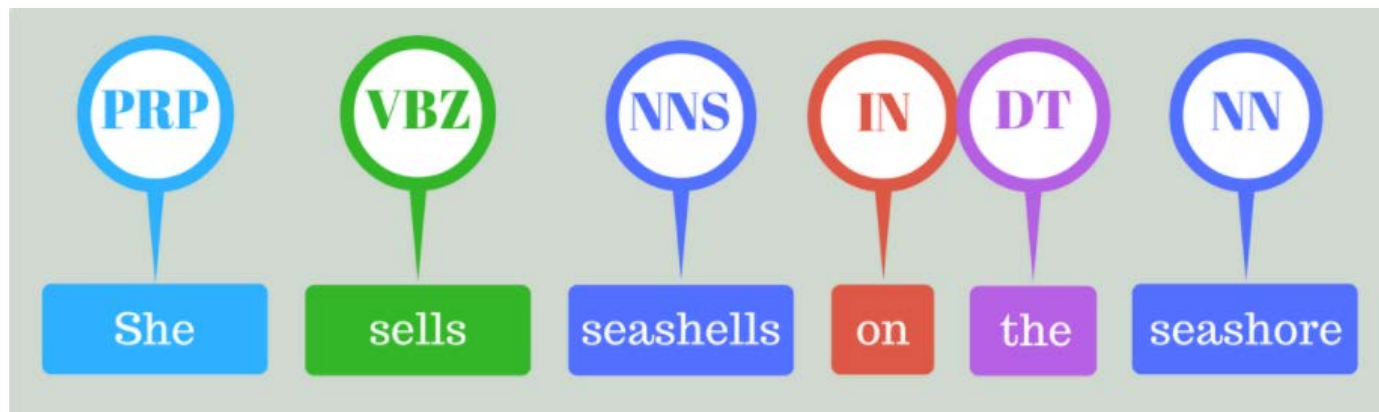
- Learn to predict output from input.
- Output can be
  - continuous: regression problems
  - **Discrete: classification problems**



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

# Supervised Learning

- Learn to predict output from input.
- Output can be
  - continuous: regression problems
  - Discrete: classification problems
  - **Structured: structured prediction problems**

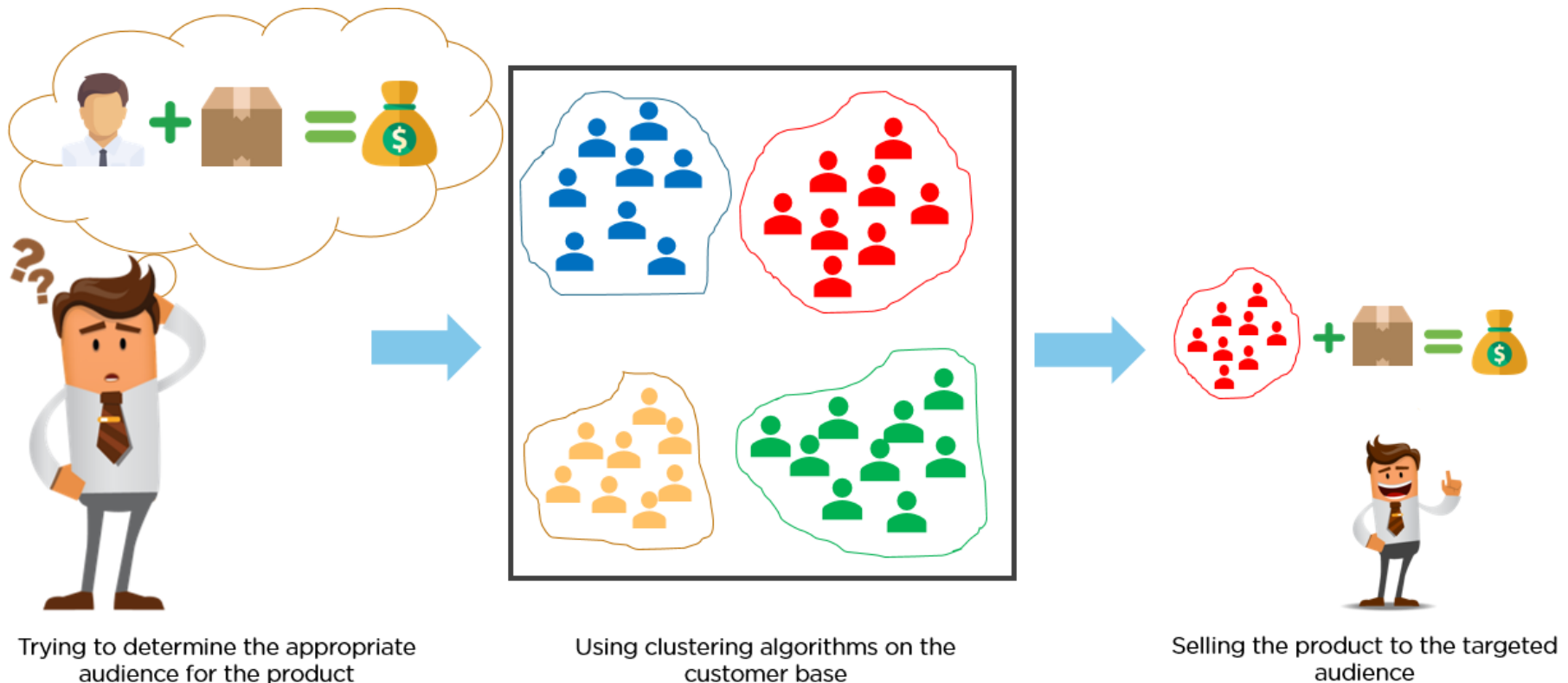


Example: part of speech tagging



# Unsupervised Learning

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



# Unsupervised Learning

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



Image Segmentation

# Unsupervised Learning

- Given a collection of examples (objects), discover self-similar groups within the data – **clustering**
- Learn the underlying distribution that generates the data we observe – **density estimation**
  - 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 ...
- 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 533 (spring) if you want to learn about this.

# When do we need computer to learn?

©2001 Shannon Burns

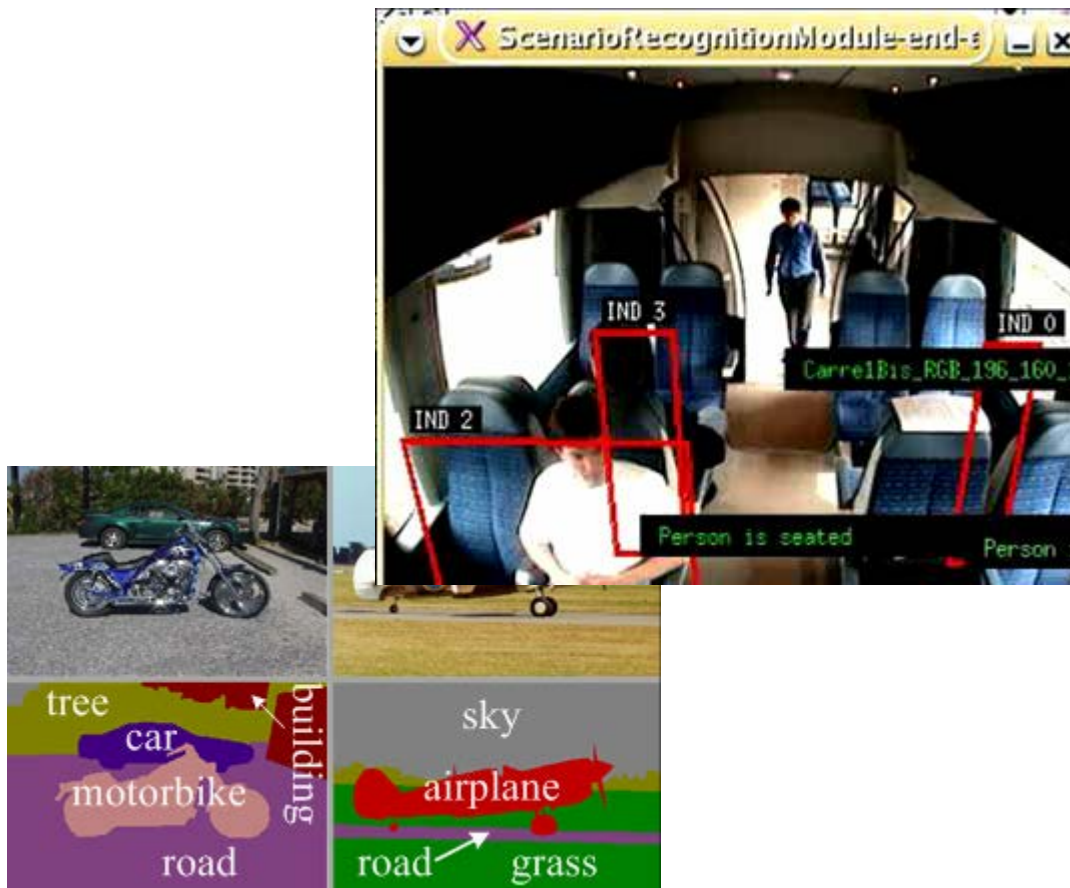
www.shannonburns.com



*Do we need learning to do tax return?*

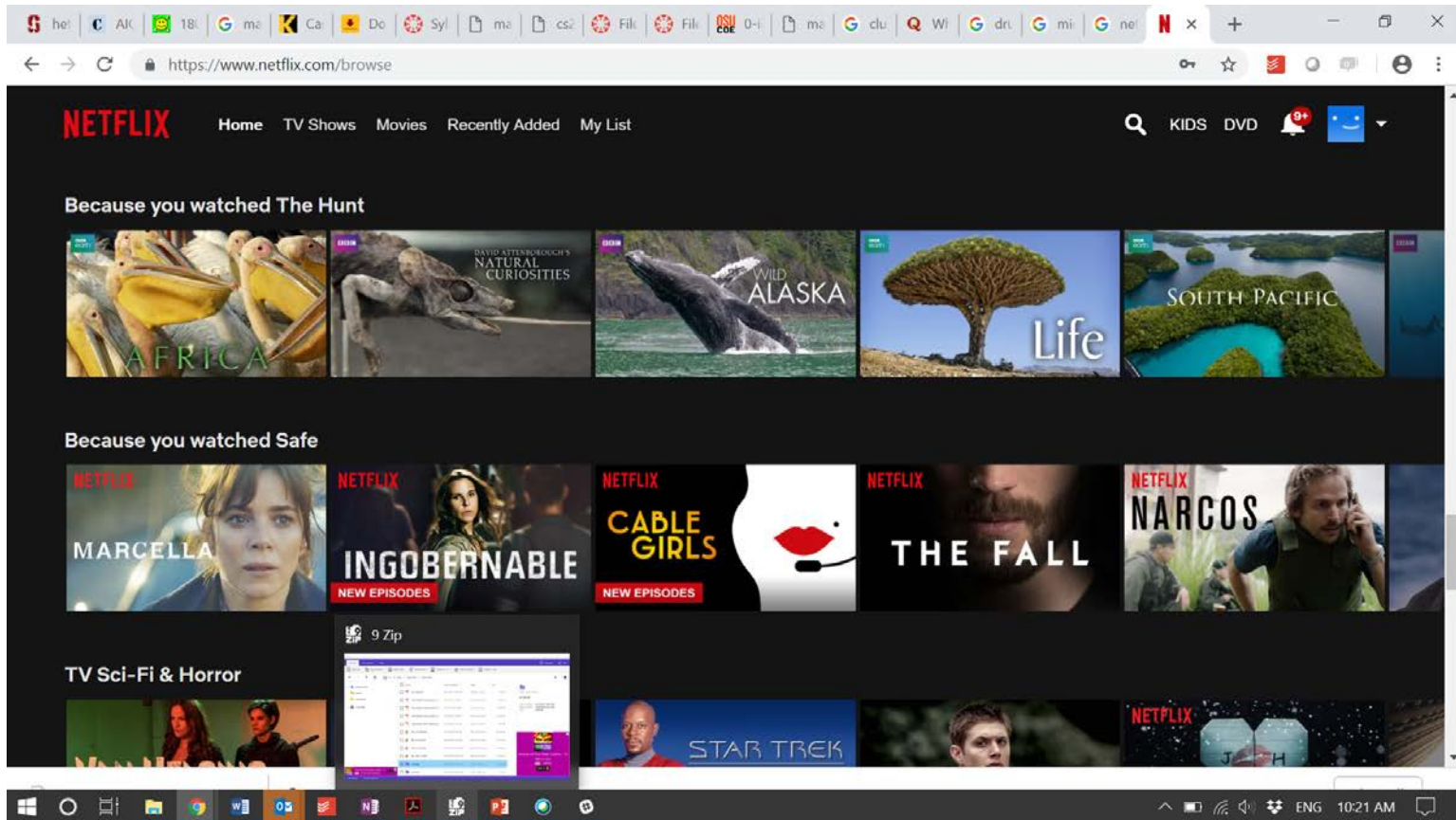
# Appropriate Applications for Supervised Learning

- 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

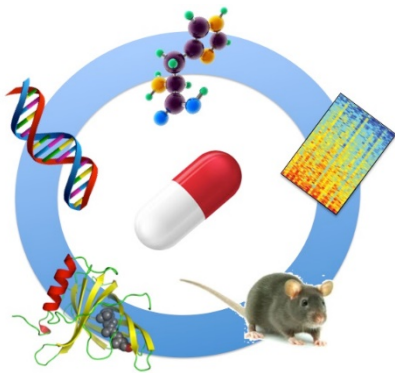
Situations where the desired function is changing frequently





# 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

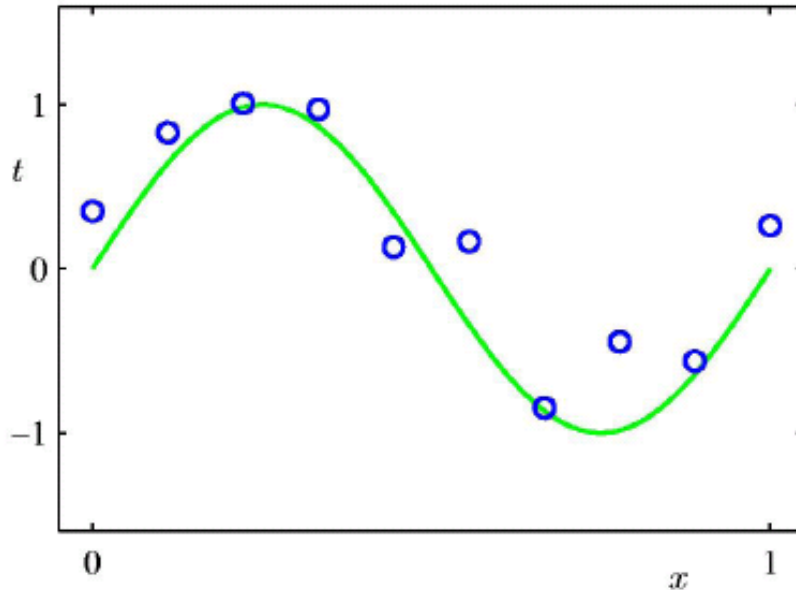
# Supervised learning (basic setup)

- Given: a set of **training examples**  
 $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \dots, (\mathbf{x}_n, y_n)$   
정리해보자면, supervised는 input에 label이 붙어서 제공된다.  
(input, label) 이 pair가 given.  
그럼 어떤 새로운 input에 대해서 개의 label을 예측해내는거지
  - $\mathbf{x}_i$ : the input of the  $i$ -th example ( $\mathbf{x}_i \in R^d$ , i.e., a vector)
  - $y_i$  is its corresponding output (continuous or discrete)
  - We assume there is some underlying function  $f$  that maps from  $\mathbf{x}$  to  $y$  – our **target function** 이 target function을 찾아내는게 supervised learning
- Goal: find a good approximation of  $f$  so that an accurate prediction can be made for previously unseen  $\mathbf{x}$

# Key Components of Machine learning

- Representation
  - How do we represent this function  $f$  we are trying to learn? Linear, polynomial, tree, neural networks, set of rules ....
- Objective 목적 목표
  - What is our goal of learning? How do we quantify it? Accuracy, Precision and recall, likelihood, cost ...
- Optimization
  - How do we optimize the objective?  
Combinatorial optimization? Convex optimization?  
Constrained optimization?

# A toy example: regression



The true underlying function:

$$t = \sin(2\pi x) + \varepsilon$$

where  $\varepsilon$  is Gaussian noise

- 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  $t$  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 fix the representation of our function by focusing on a limited set of possible functions
  - 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
- Learning here means to find a good set of parameters  $\mathbf{w}$  to minimize some **loss function (objective)**

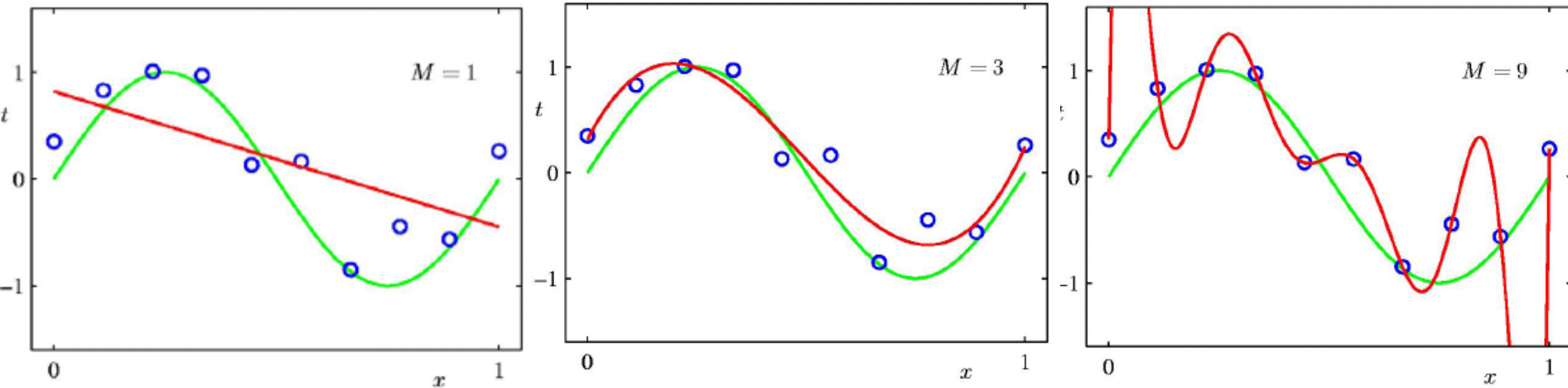
$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (y(x_n, \mathbf{w}) - t_n)^2$$

Sum-of-squares error

This optimization problem can be solved easily.

We will not focus on solving this at this point, will revisit this later.

# 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  $E(\mathbf{w})$  that we define in previous slides as a criterion to choose  $M$ ?

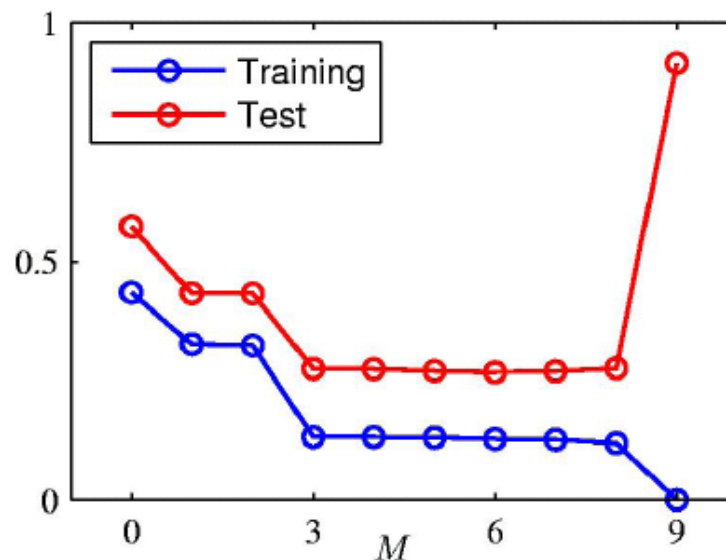
$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (y(x_n, \mathbf{w}) - t_n)^2$$

Sum-of-squares error

# Over-fitting

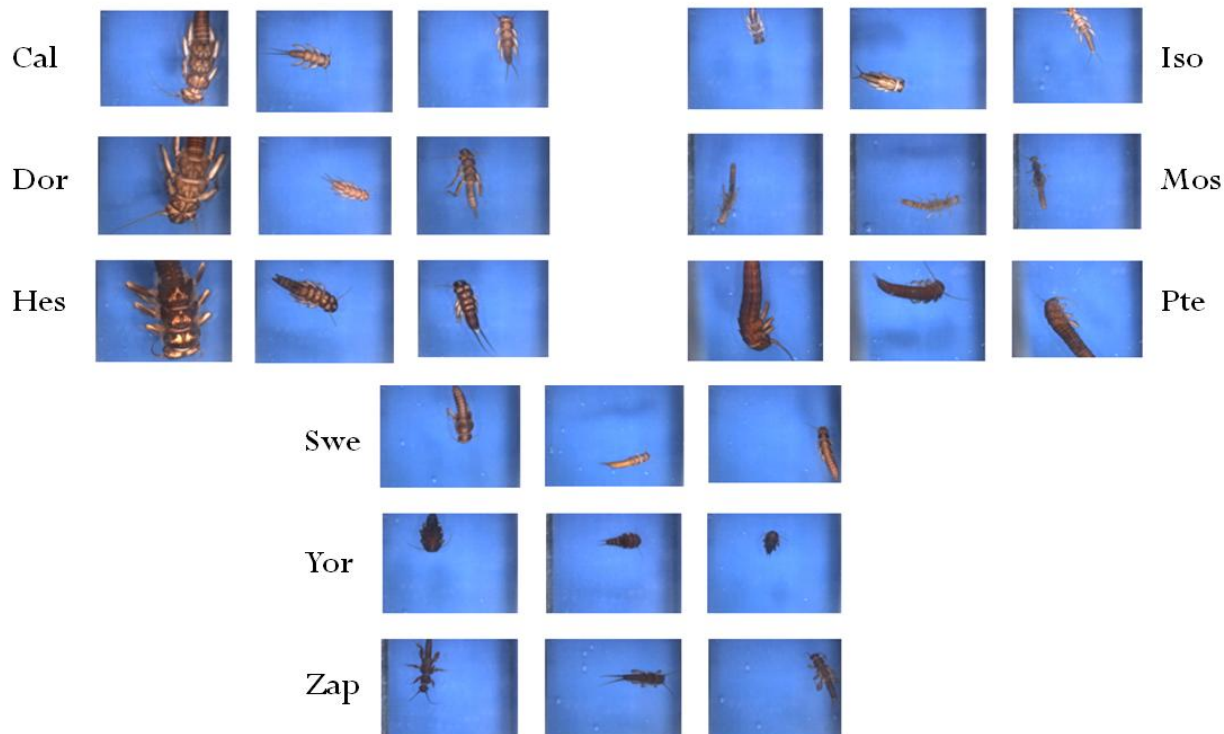
- As  $M$  increases, SSE on the training data decreases monotonically
- However, the SSE on test data starts to increase after a while
  - Why? Is this a fluke or generally true?

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



# Over-fitting

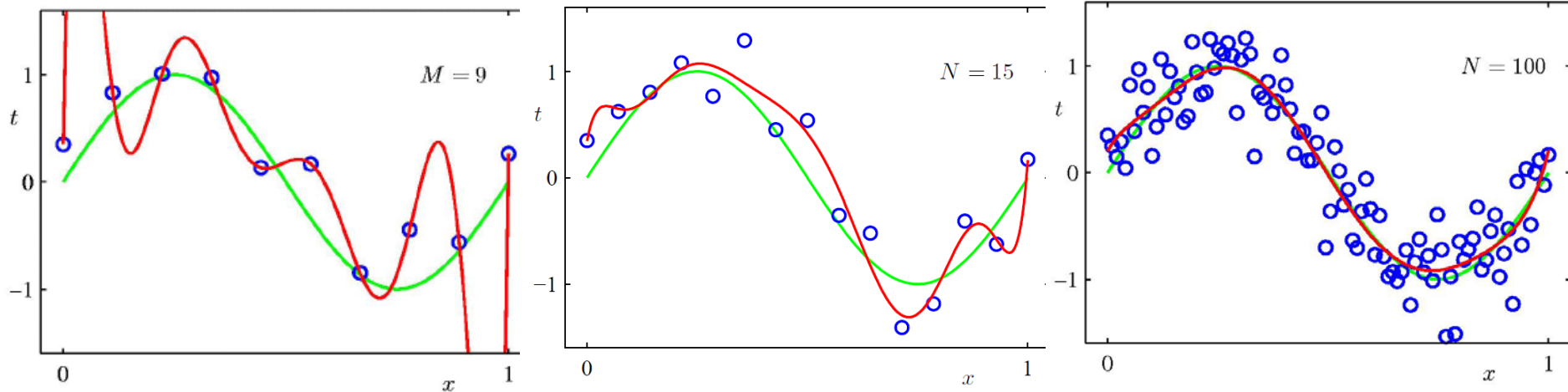
- Over-fitting refers to the phenomenon when the learner adjusts to some random signals in the training data that is not relevant to the target function
- Real story from bugID project





# 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.

# Some 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 often 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 function
- Nonlinear classification models
  - Kernel SVM
  - Decision trees
  - Neural networks