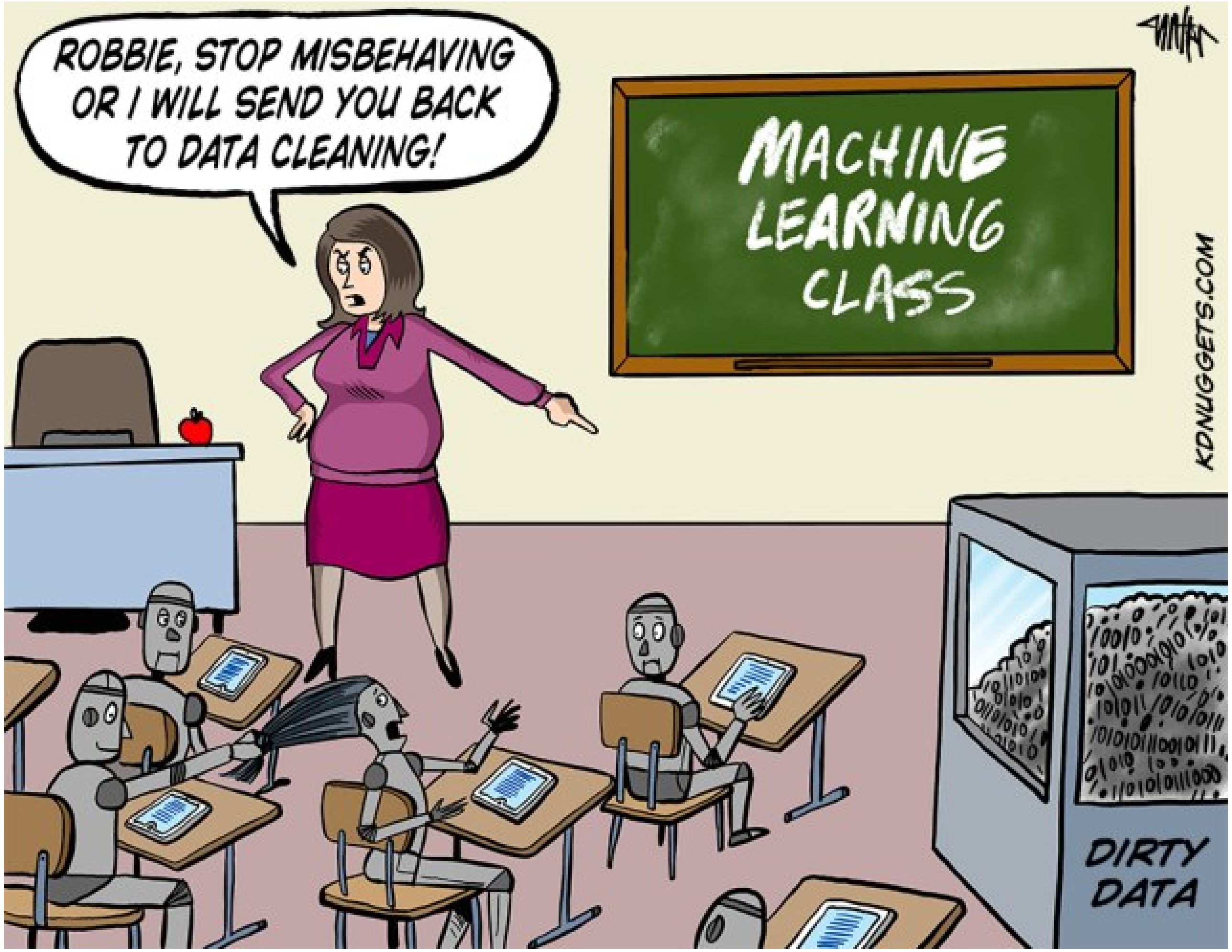
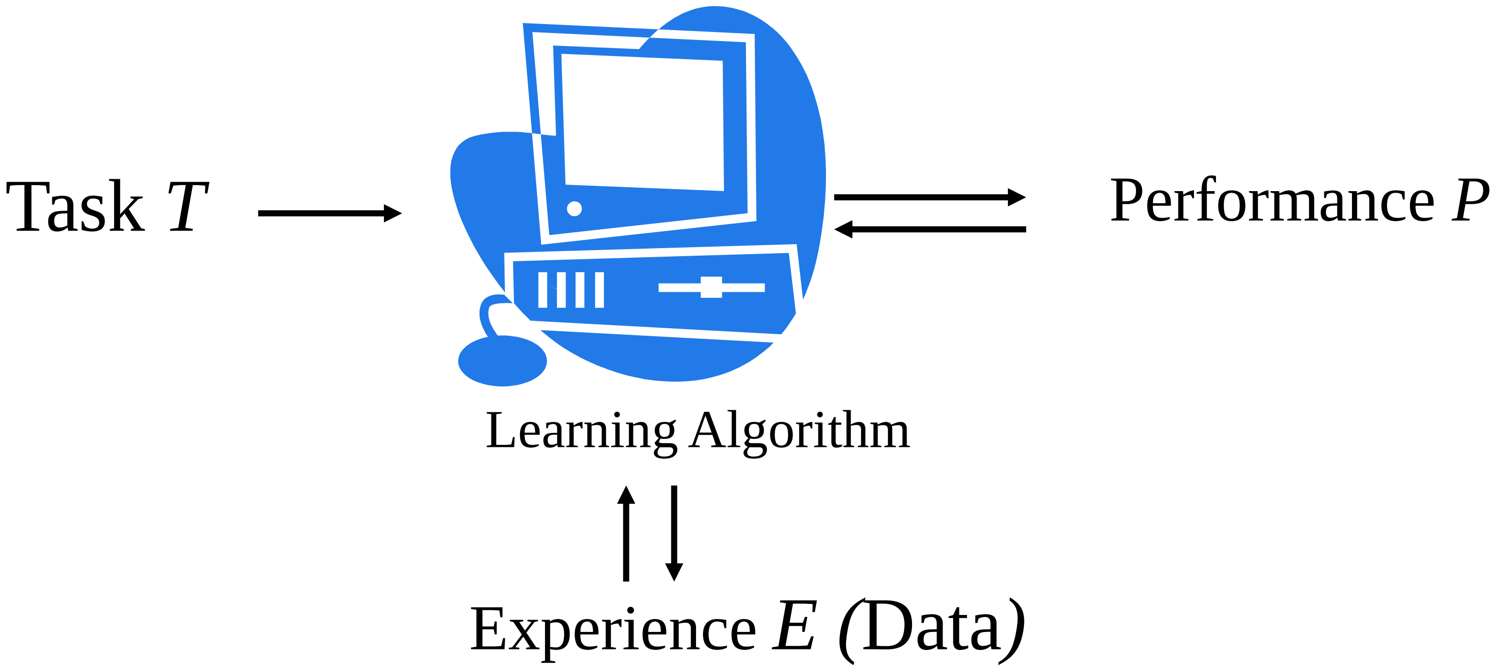
Welcome to the machine learning class



What is Machine learning



Machine learning studies algorithms that

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

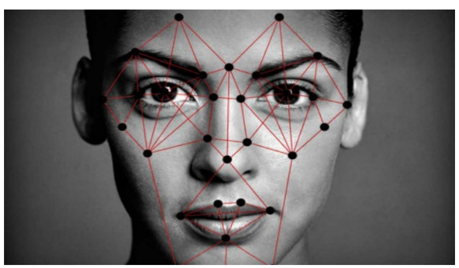
Task T

Performance P

Facial recognition

Learning Algorithm

Prediction accuracy

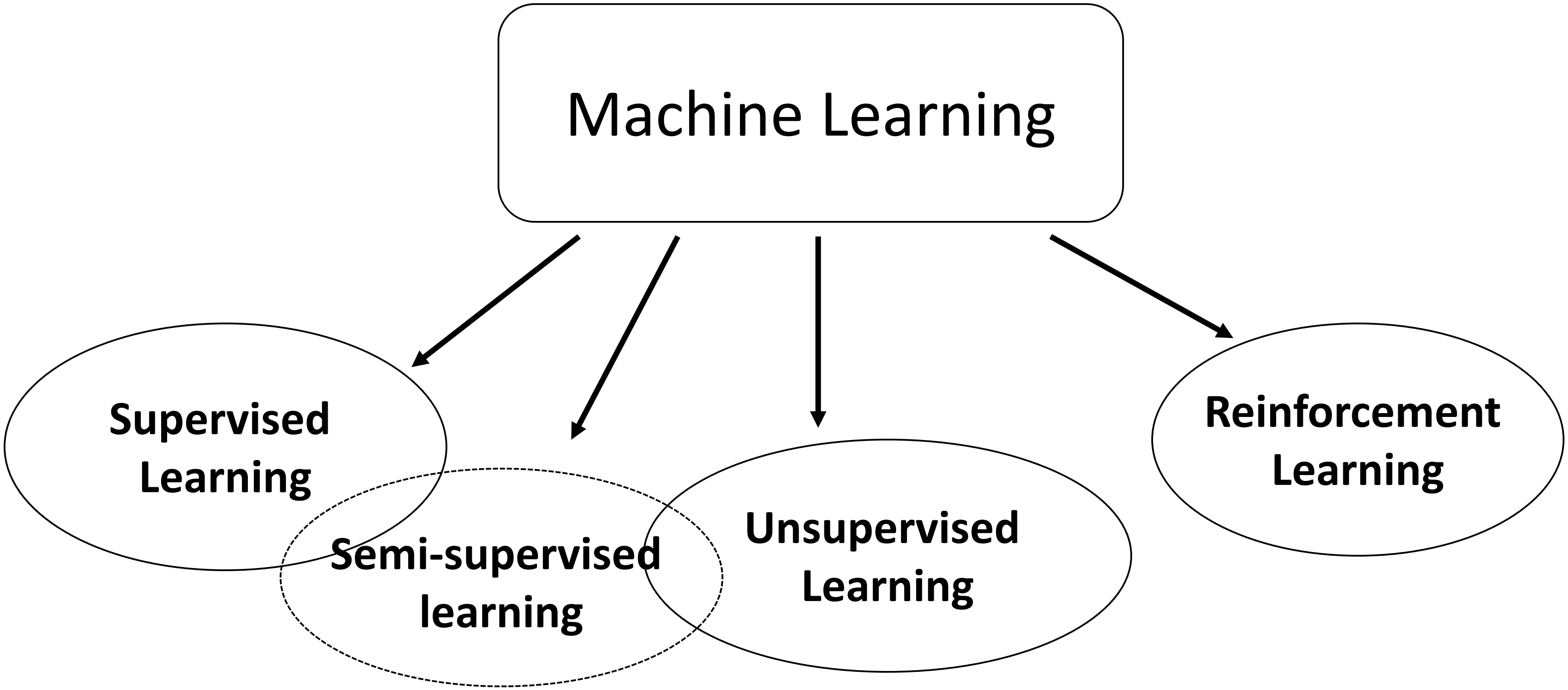


Experience E (Data)

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
  - Improved machine learning algorithms
  - Increased data capture, and new sensors
  - Increased computing power
  - Increasing demand for self-customization to user and environment

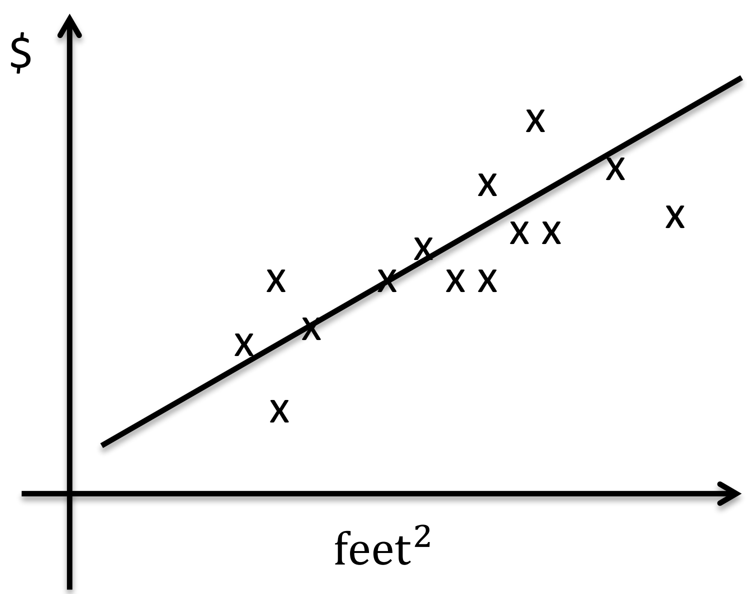
Topics



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

Supervised Learning

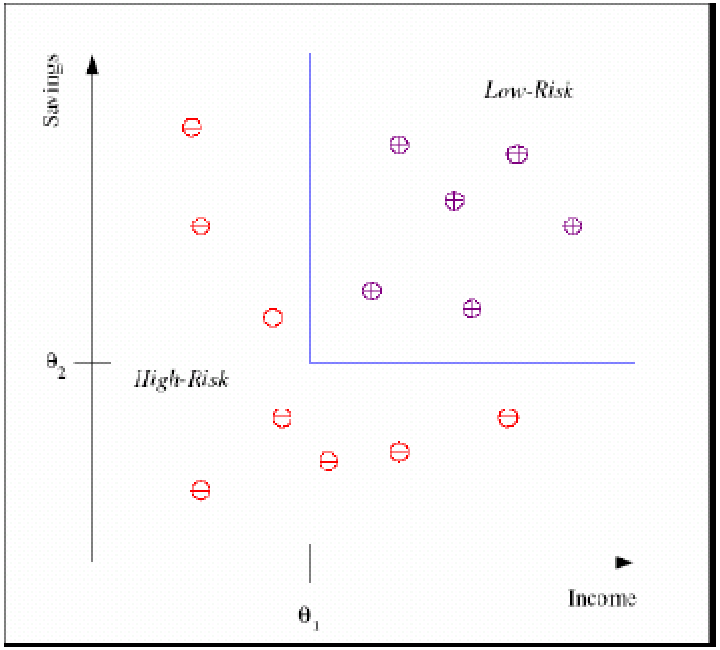
- Learn to predict output from input.
- Output can be
  - **continuous: regression problems**



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

Supervised Learning

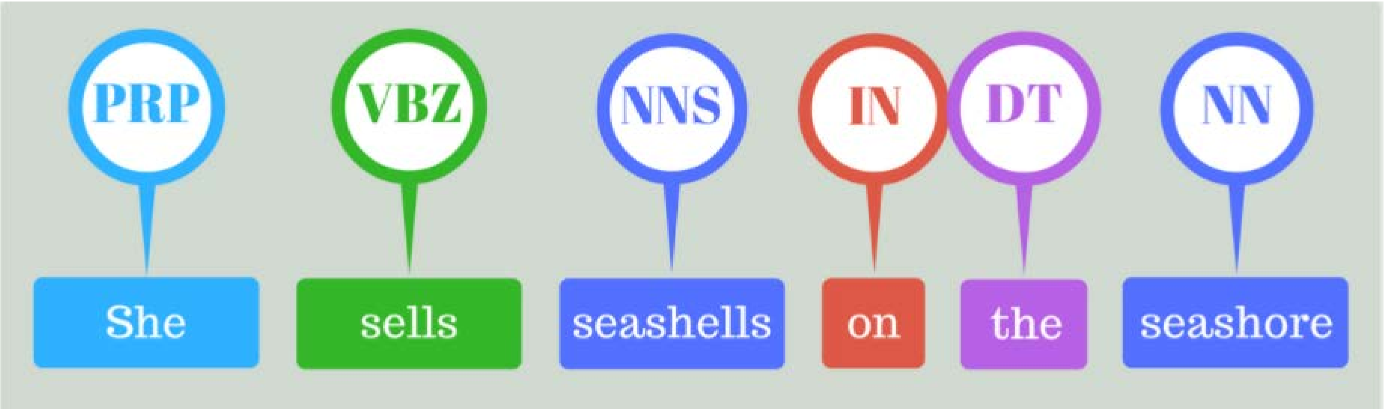
- Learn to predict output from input.
- Output can be
  - continuous: regression problems
  - **Discrete: classification problems**



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

Supervised Learning

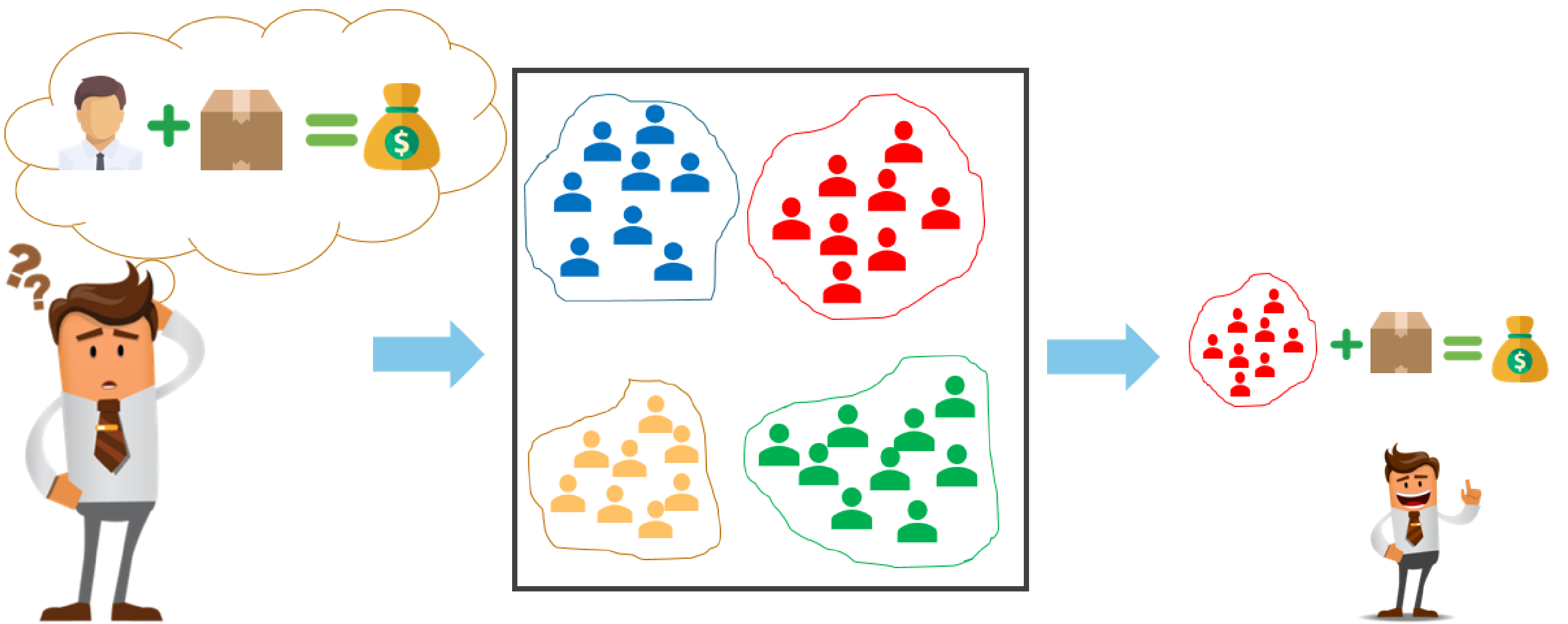
- Learn to predict output from input.
- Output can be
  – continuous: regression problems
  – Discrete: classification problems
  – **Structured: structured prediction problems**



Example: part of speech tagging

Unsupervised Learning

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



Trying to determine the appropriate audience for the product

Using clustering algorithms on the customer base

Selling the product to the targeted audience

Unsupervised Learning

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

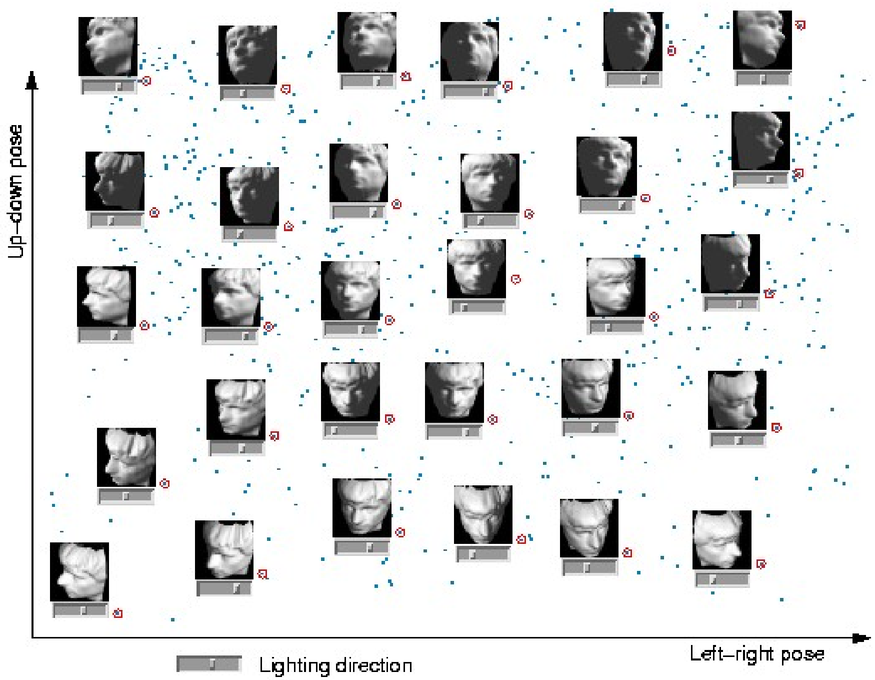
Image Segmentation

Unsupervised Learning

- 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

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



Reinforcement Learning

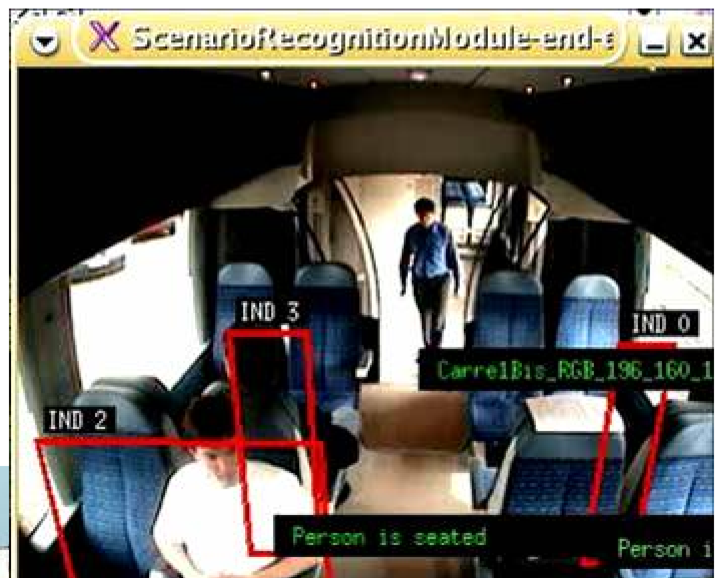
- Learn to act
- An agent
  - Observes the environment
  - Takes action
  - With each action, receives rewards/punishments
  - Goal: learn a policy that optimizes rewards
- No examples of optimal outputs are given
- Not covered in this class. Take 533 (spring) if you want to learn about this.

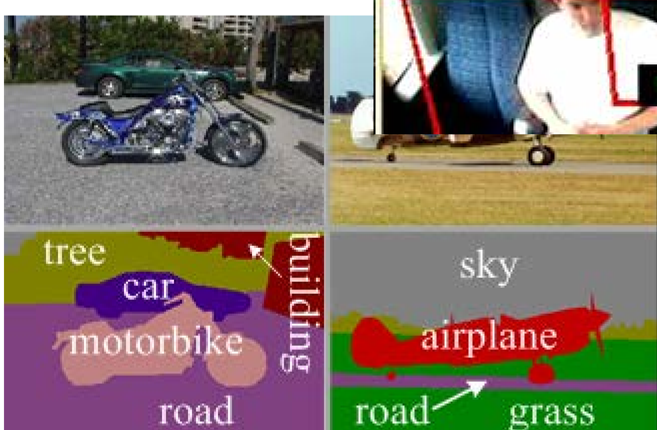
When do we need computer to learn?

Do we need learning to do tax return?

Appropriate Applications for Supervised Learning

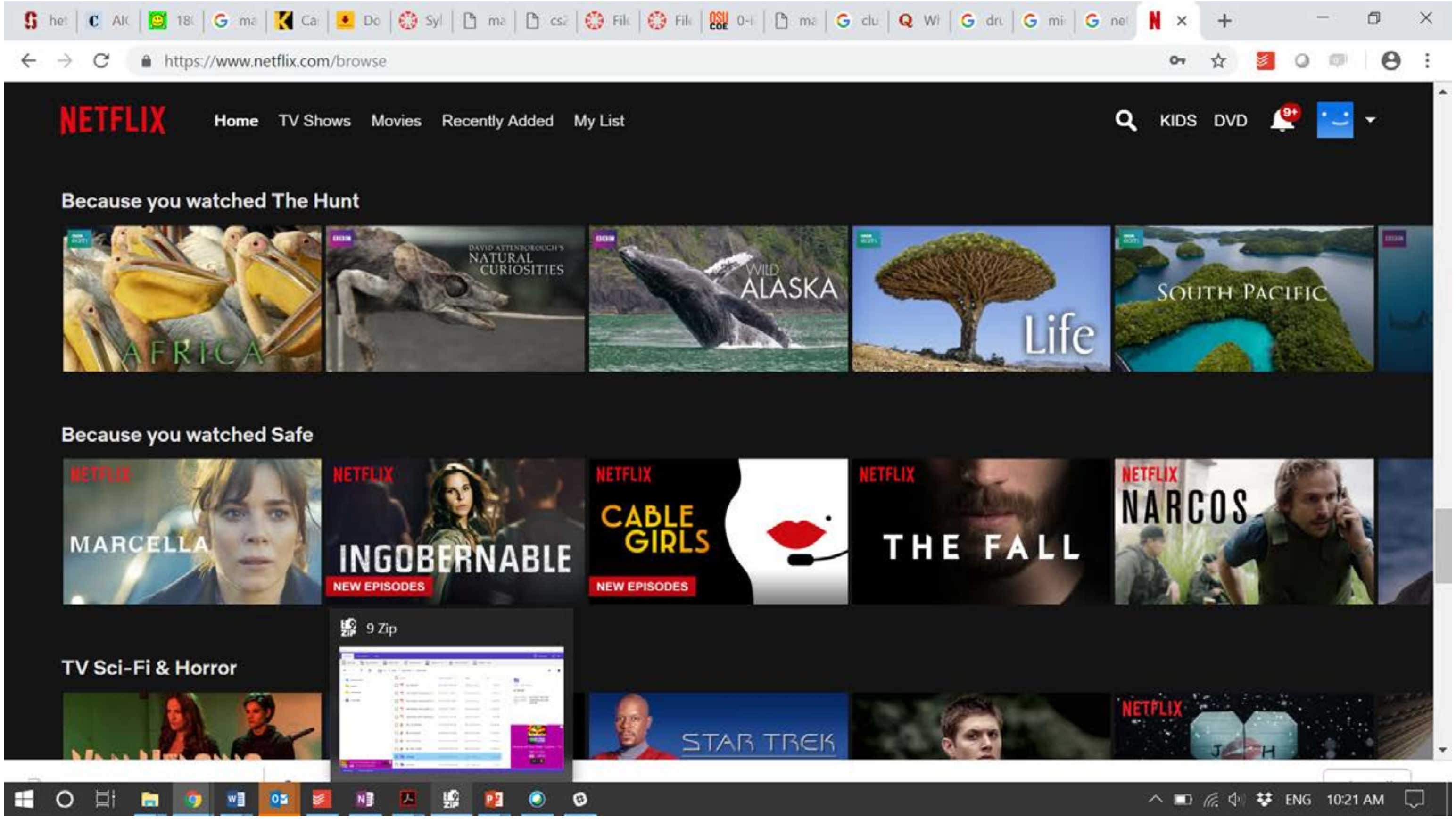
- Situations where humans can perform the task but can't describe how they do it





Appropriate Applications for Supervised Learning

- Situations where the desired function is different for each individual



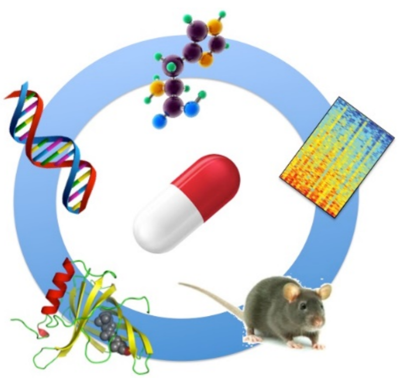
Appropriate Applications for Supervised Learning

Situations where the desired function is changing frequently

Finance Chart Demonstration

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 ith 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 <u>previously unseen $\mathbf{x}$</u>
  - **Generalization**

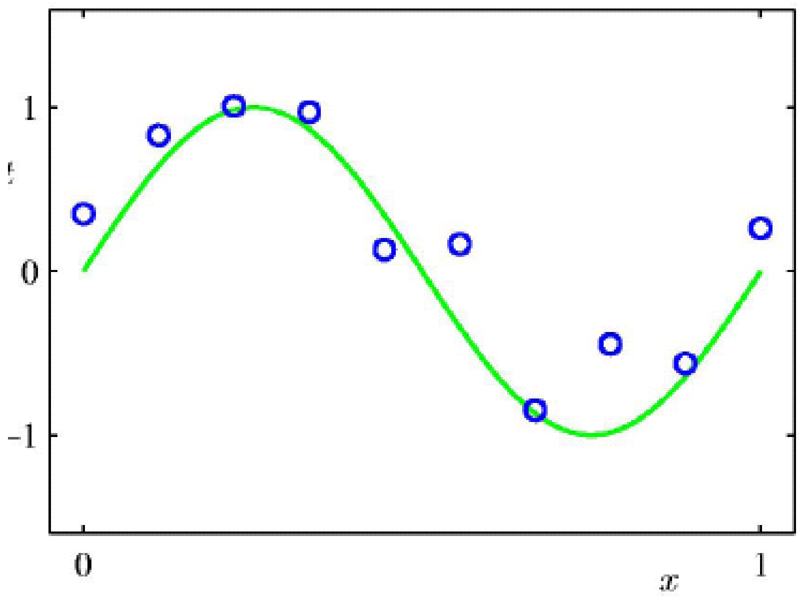
Key Components of Machine learning

- Representation
  - How do we represent this function f we are trying to learn? Linear, polynomial, tree, neural networks, graphical model, set of rules ….
- Objective for training
  - What is our goal of learning? How do we quantify it?

  Accuracy, Precision and recall, likelihood, cost …
- Optimization
  - How do we optimize the objective?

  Search? Combinatorial optimization? Convex optimization? Constrained optimization?

A toy example: regression



The true underline function:

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

where ϵ 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 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

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M$$

  - **w =** (w_0, w_1 ,…, 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

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

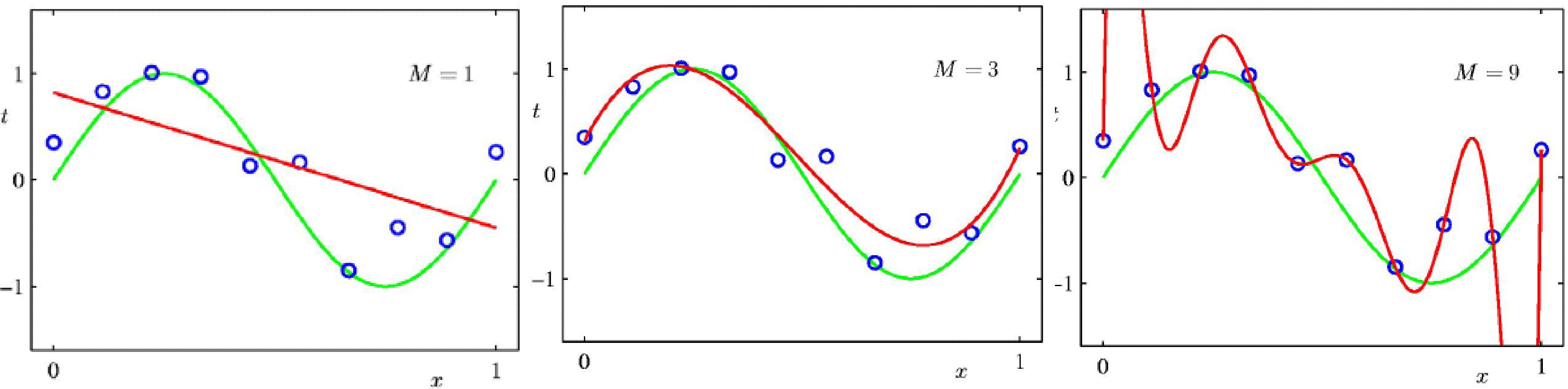
A commonly used loss function is:

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

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(**w**) that we define in previous slides as a criterion to choose M?

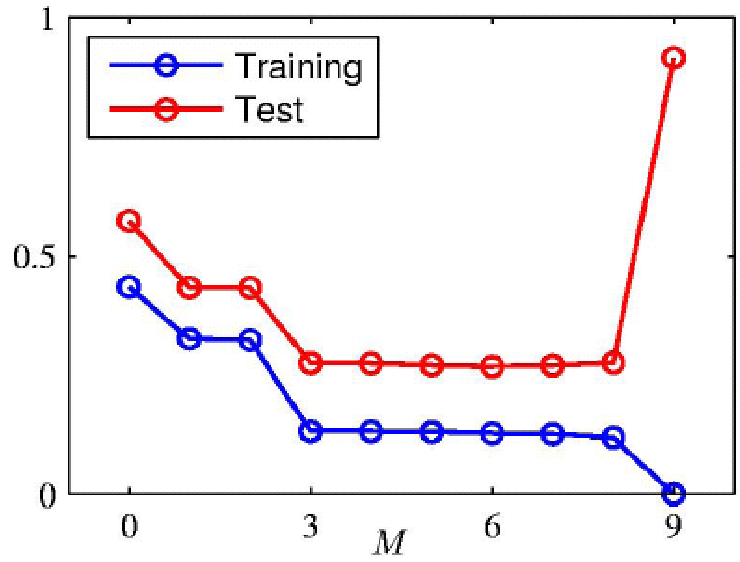
$$L(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(y(x_i, \mathbf{w}) - y_i)^2$$

Sum of Squared Errors

Over-fitting

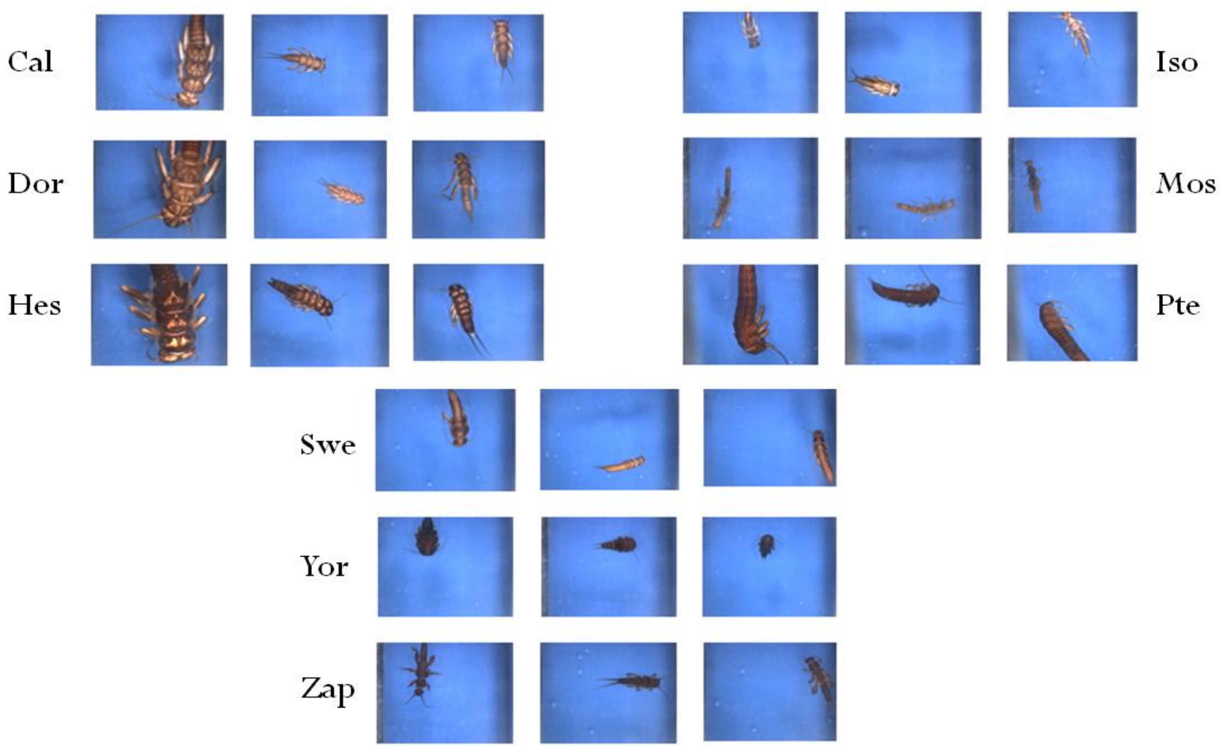
- As M increases, SSE on the training data decreases monotonically
- However, the SSE on test data starts to increase after a while
  - Why? Is this a fluke or generally true?

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



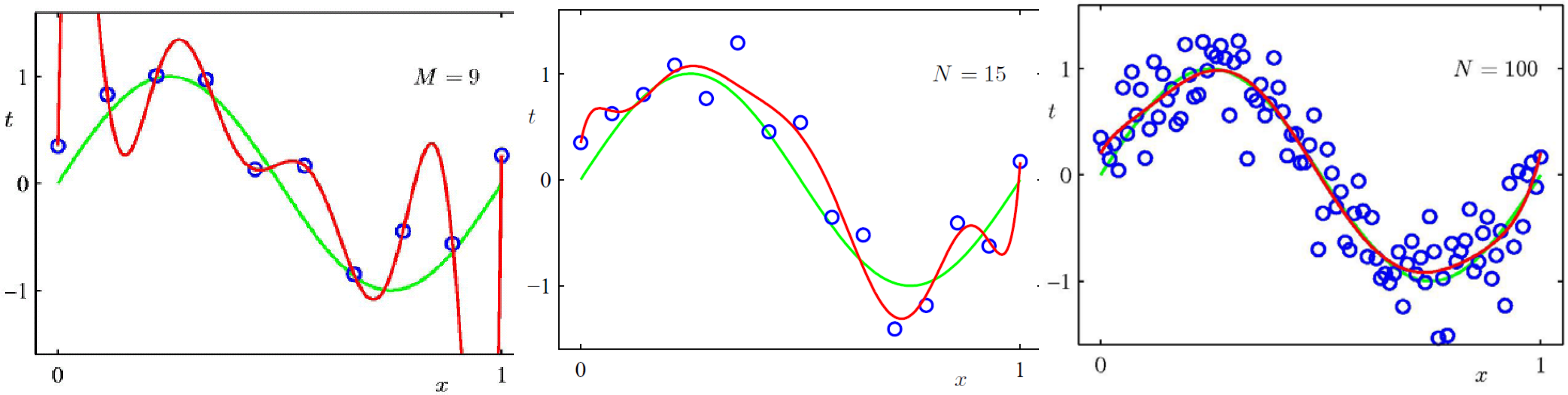
Over-fitting

- 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.

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 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
- Nonlinear classification models
  - Kernel SVM
  - Decision trees
  - Neural networks

Maximum likelihood estimation with probabilistic objectives

Optimizing convex Loss functions