CS 6375 Machine Learning

Professor: Sriraam Natarajan

Email: Sriraam.Natarajan@utdallas.edu

Webpage:
http://utdallas.edu/~sxn177430/Courses/6375ML.html

Class Hours: MW 11:30-12:45

Office Hours W 10:30-11:30 and by appointment

Course Information

- No text book required, slides and reading materials will be provided in elearning page
- There are a few recommended books that are good references
  - Pattern recognition and machine learning by Chris Bishop (Bishop)
  - Machine learning by Tom Mitchell (TM)
  - Machine Learning: A Probabilistic Perspective by Kevin Murphy (Murphy)
- Slides will be posted **after** the classes in time for assignments
- Most of the slides are based on Tom Dietterich and Jude Shavlik's class slides (obtained with their permission)

Course Syllabus

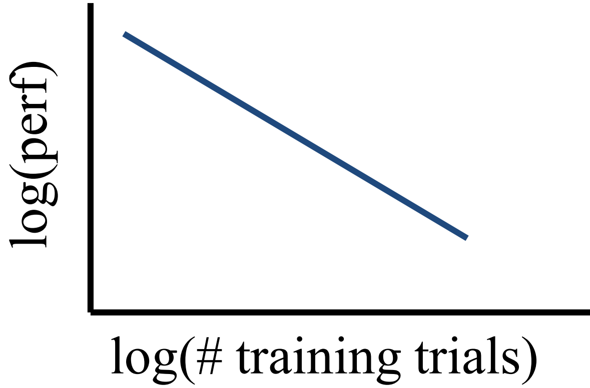
- Linear Classifiers (Naive Bayes, and logistic regression)
- Non-linear classifiers (neural networks, decision trees, support-vector machines, nearest neighbor methods)
- Ensemble Methods (bagging and boosting)
- Computational Learning Theory
- Reinforcement Learning
- Unsupervised Learning

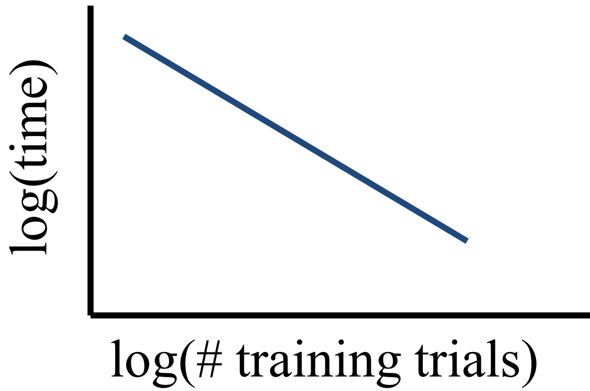
Grading

- Assignments (25%)
- Programming Assignments(25%)
- Midterm (25%)
- Final (25%)

Learning

- Herbert Simon: “Learning is any process by which a system improves performance from experience.”
- A Collaborator: “We need machine learning because we like being lazy. i.e., let the machines learn to do what we do”
- Ray Mooney: Develop systems that are too difficult/expensive to construct manually because they require specific detailed skills or knowledge tuned to a specific task.
- Develop systems that can automatically adapt and customize themselves to individual users.
- Discover new knowledge from large databases.





History of Machine Learning

- 1950s
  - Samuel’s checker player
  - Selfridge’s Pandemonium
- 1960s:
  - Neural networks: Perceptron
  - Pattern recognition
  - Learning in the limit theory
  - Minsky and Papert prove limitations of Perceptron
- 1970s:
  - Symbolic concept induction
  - Winston’s arch learner
  - Expert systems and the knowledge acquisition bottleneck
  - Quinlan’s ID3
  - Michalski’s AQ and soybean diagnosis
  - Scientific discovery with BACON
  - Mathematical discovery with AM

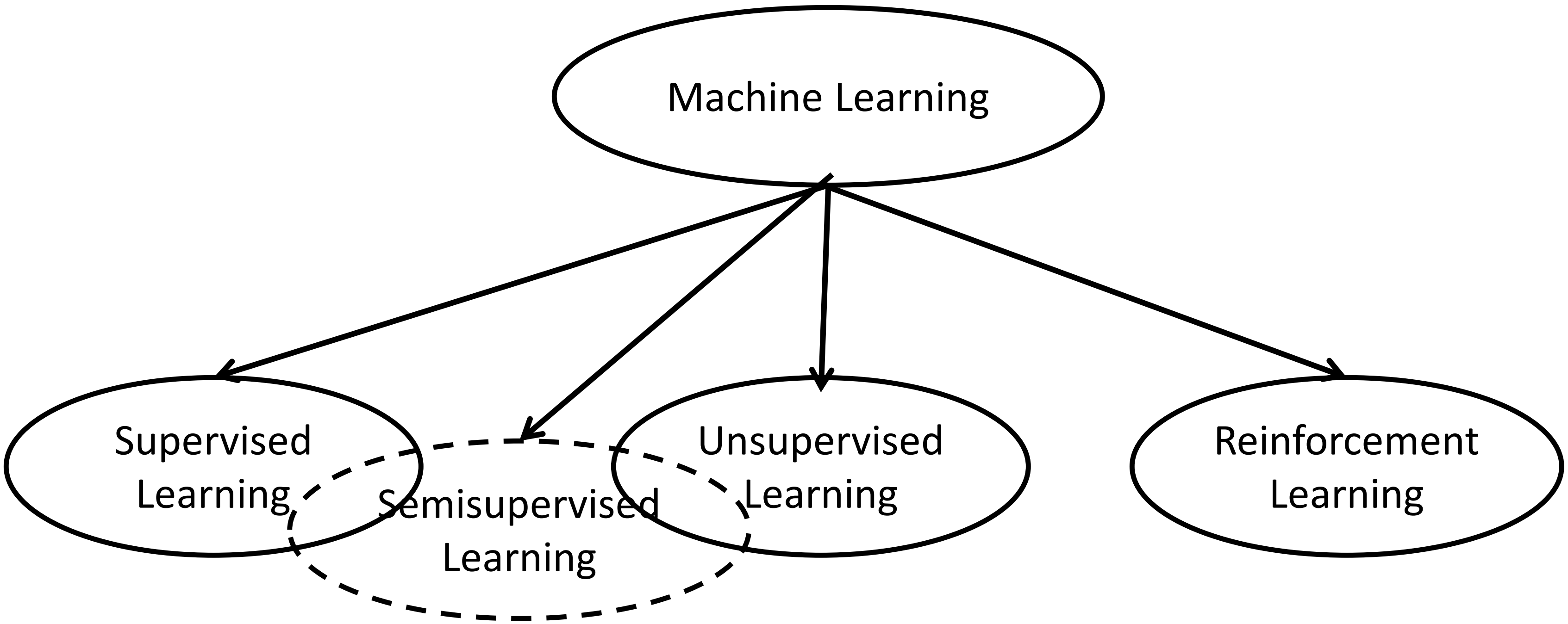
History of Machine Learning (cont.)

- 1980s:
  - Advanced decision tree and rule learning
  - Explanation-based Learning (EBL)
  - Learning and planning and problem solving
  - Utility problem
  - Analogy
  - Cognitive architectures
  - Resurgence of neural networks (connectionism, backpropagation)
  - Valiant’s PAC Learning Theory
  - Focus on experimental methodology
- 1990s
  - Data mining
  - Adaptive software agents and web applications
  - Text learning
  - Reinforcement learning (RL)
  - Inductive Logic Programming (ILP)
  - Ensembles: Bagging, Boosting, and Stacking
  - Bayes Net learning

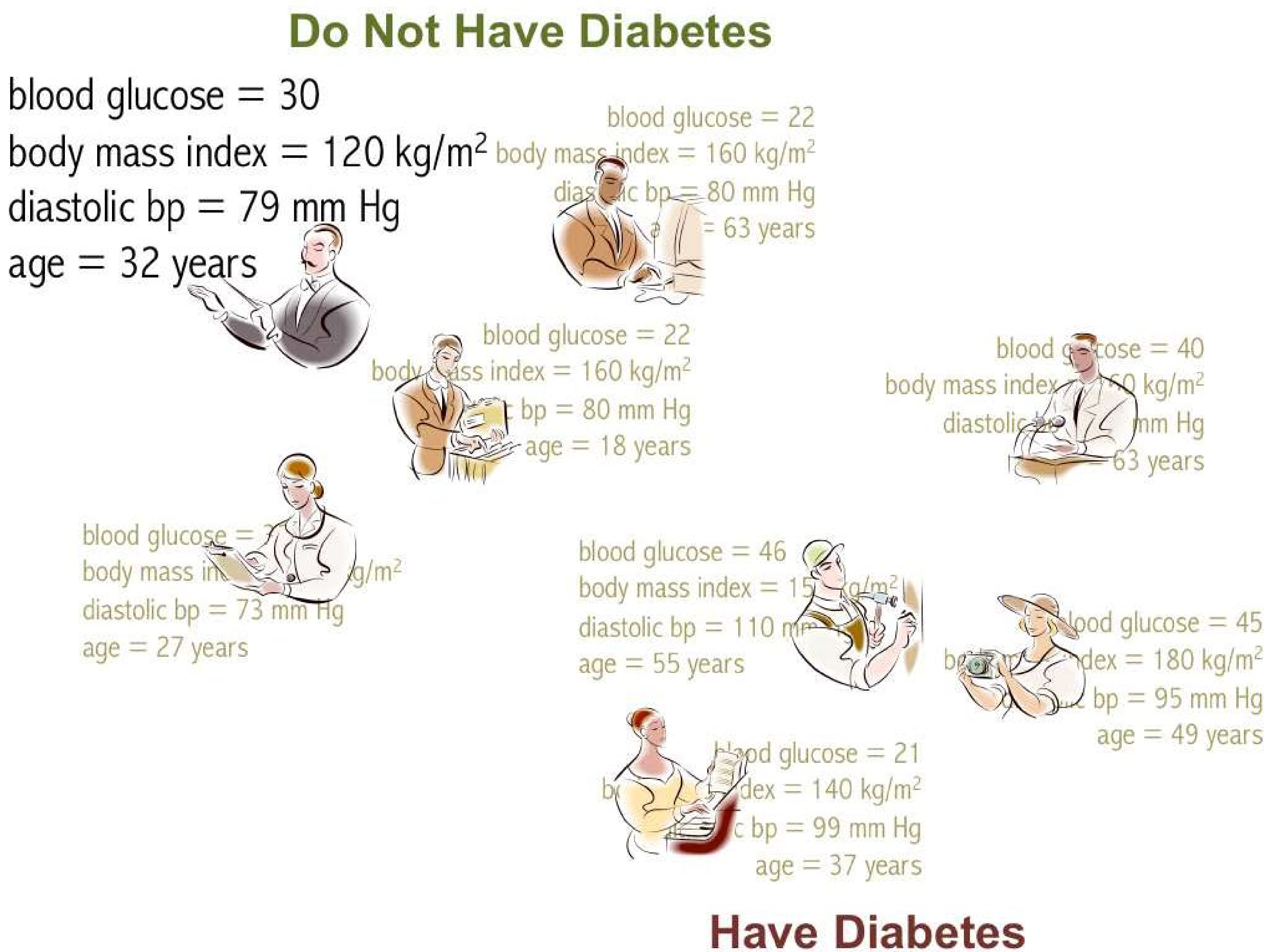
History of Machine Learning (cont.)

- 2000s
  - Support vector machines
  - Kernel methods
  - Graphical models
  - Statistical relational learning
  - Transfer learning
  - Sequence labeling
  - Collective classification and structured outputs
  - Computer Systems Applications
    - Compilers
    - Debugging
    - Graphics
    - Security (intrusion, virus, and worm detection)
  - E mail management
  - Personalized assistants that learn
  - Learning in robotics and vision

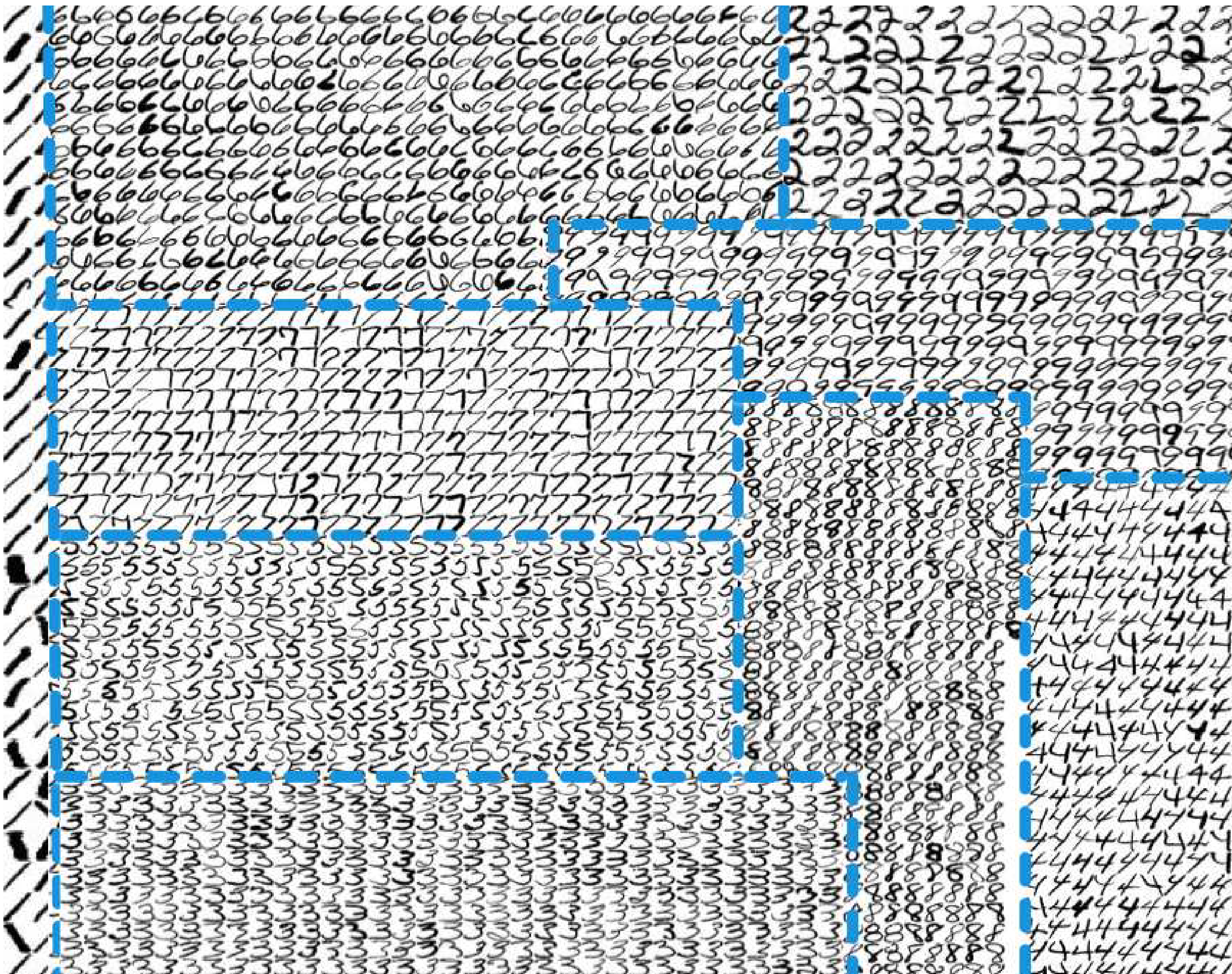
Machine Learning



Supervised Learning – Medical Diagnosis

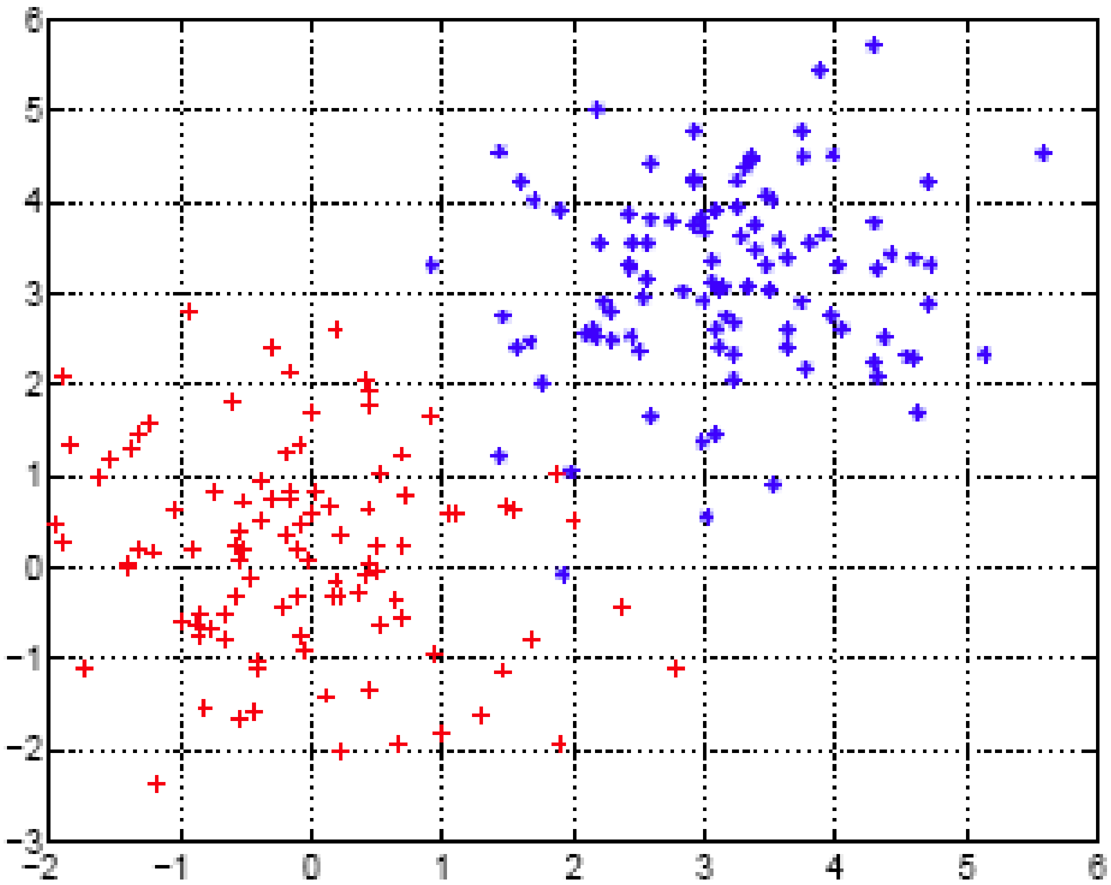


Supervised Learning – Digit Recognition

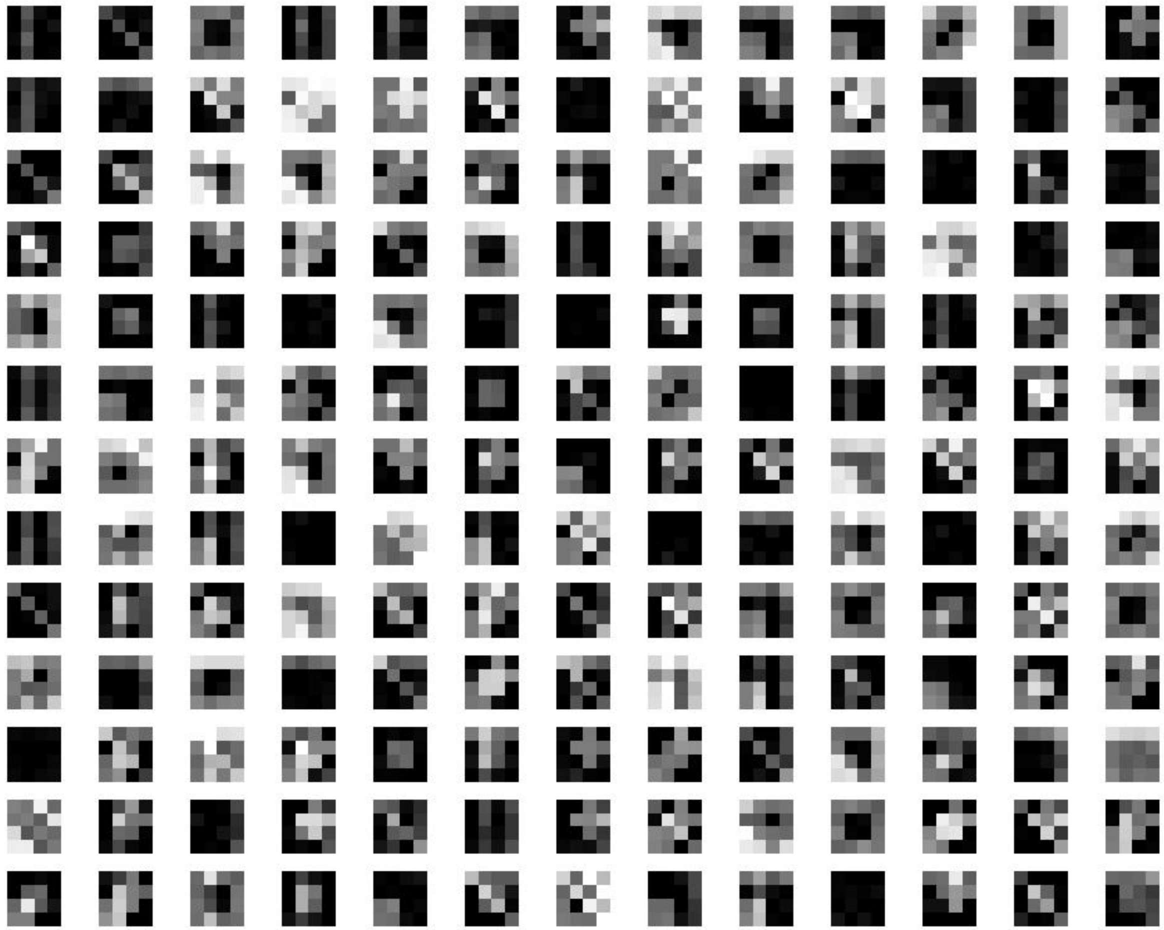


Unsupervised Learning

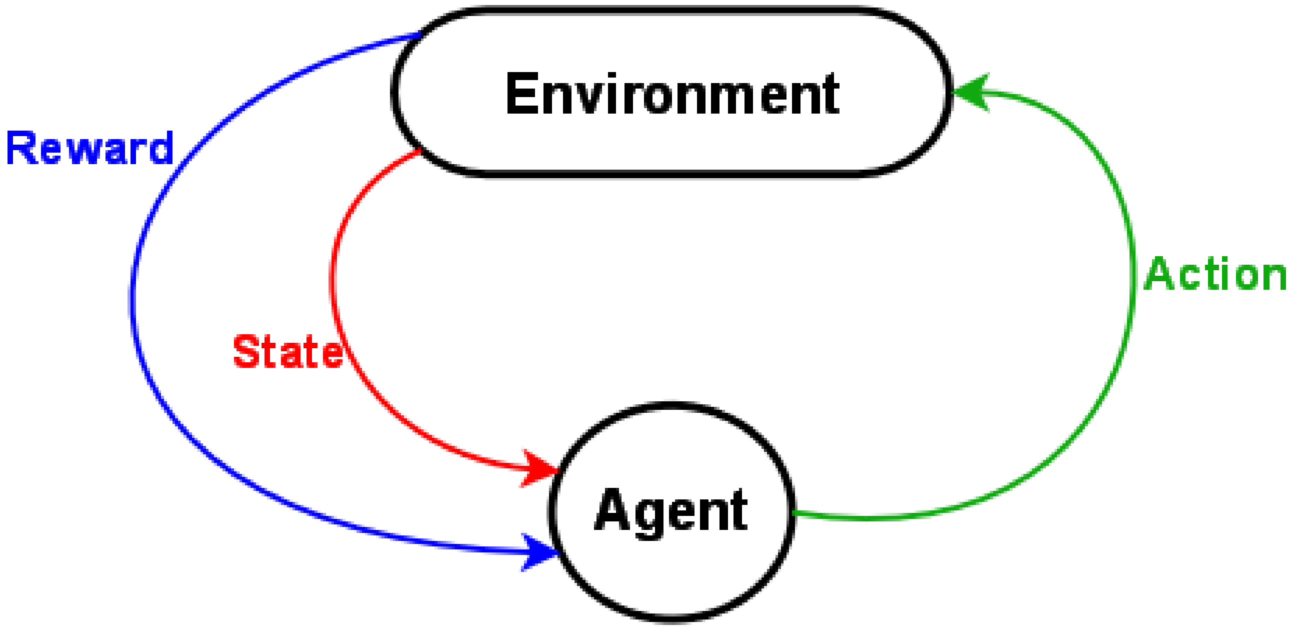
- Also, called clustering



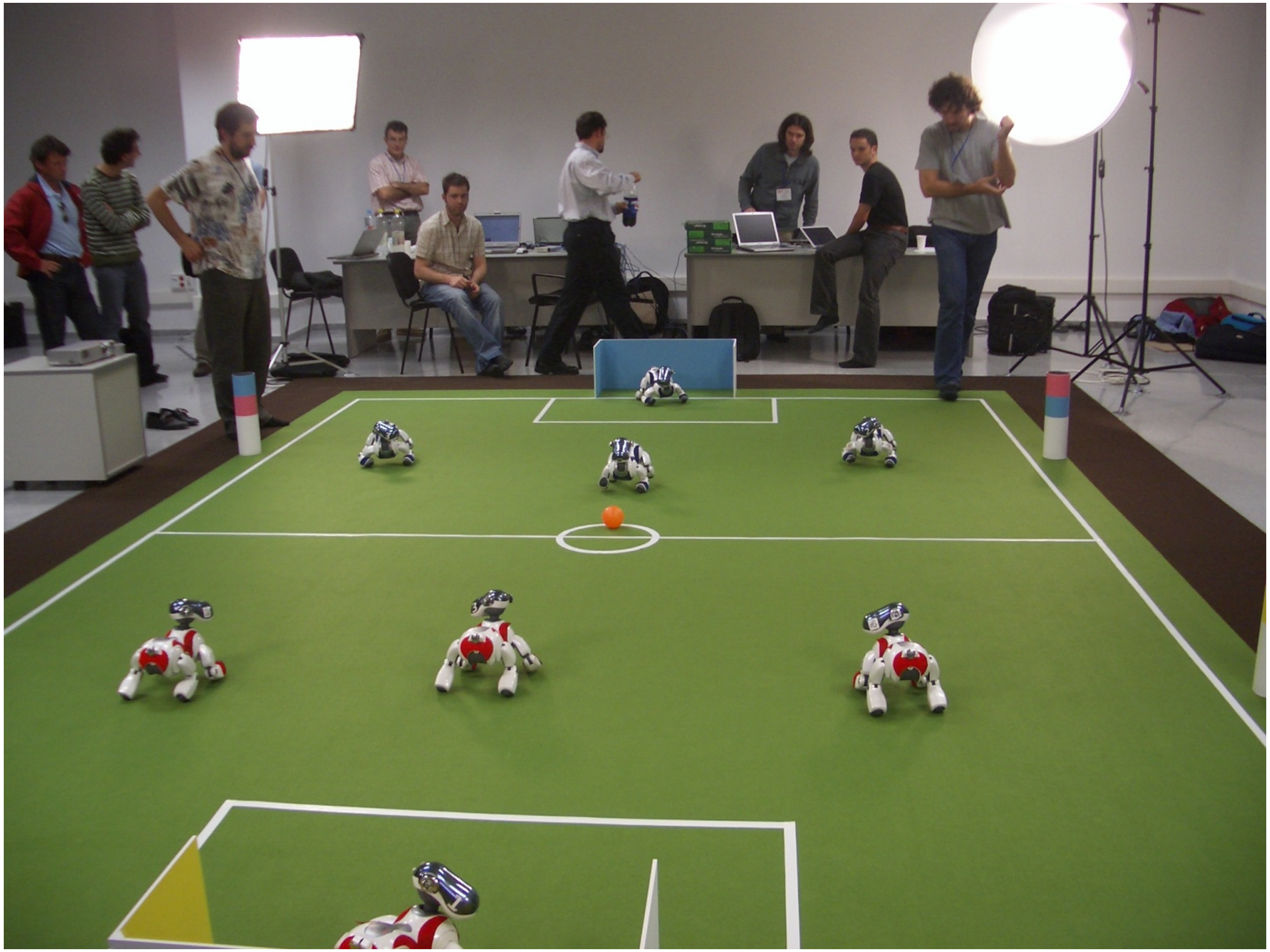
Unsupervised Learning – Object Recognition



Reinforcement Learning



Reinforcement Learning – Robocup Soccer

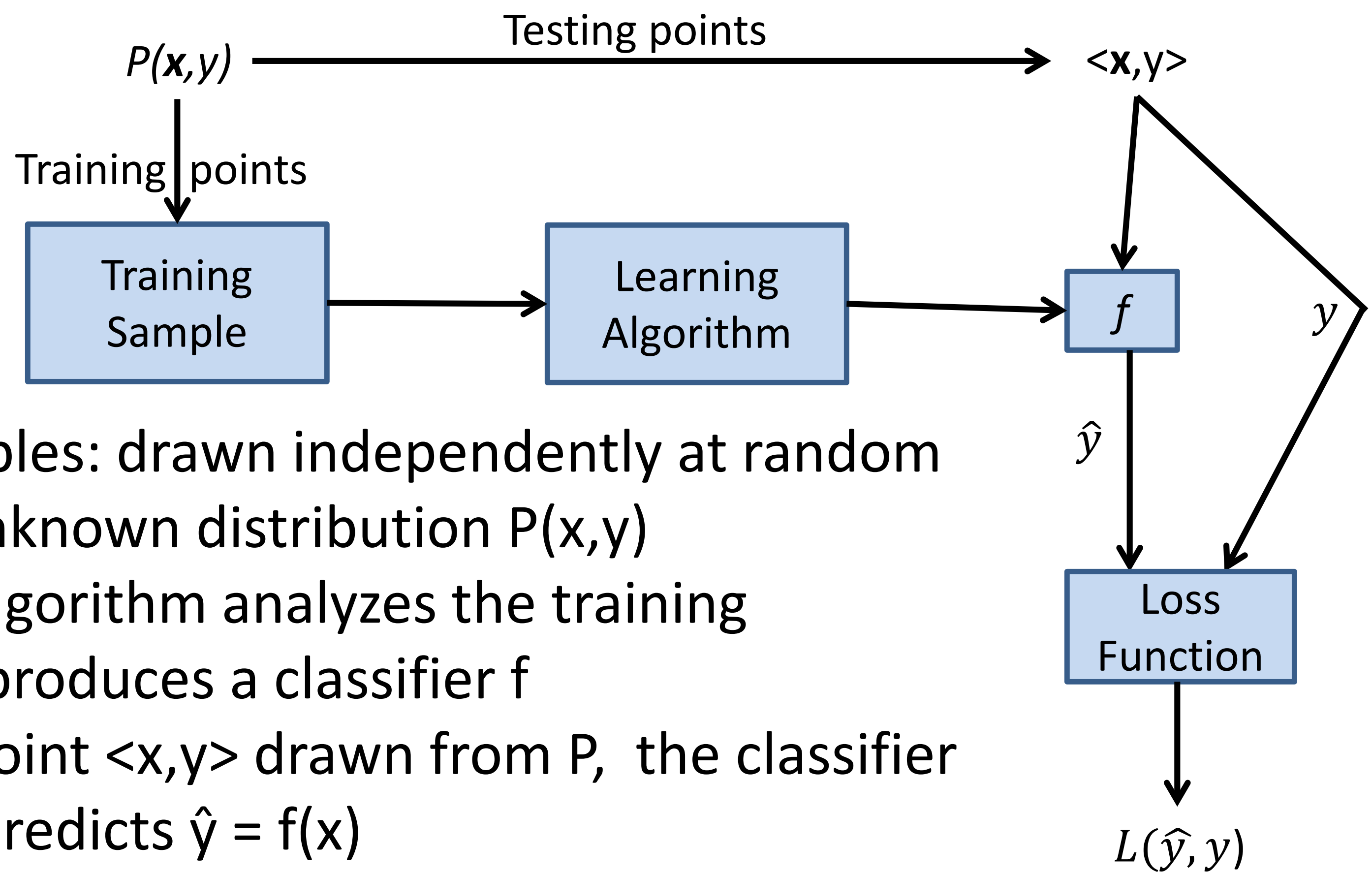


Supervised Learning

- **Given**: Training examples *<x,f(x)>* for some unknown function *f*.
- **Find**: A good approximation to *f*.

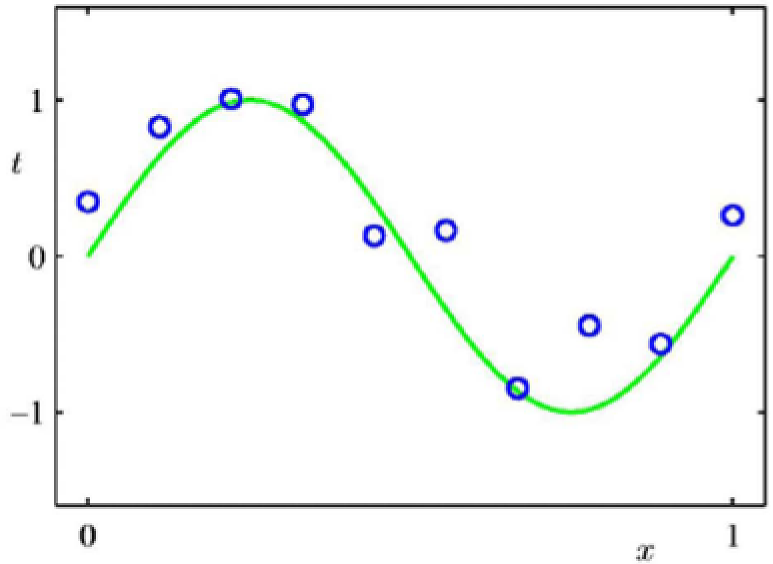
- Situations where there is no human expert
  – x: bond graph of a new molecule
  – f(x): predicted binding strength to AIDS protease molecule
- Situations where humans can perform the task but can't describe how they do it
  – x: picture of a hand-written character
  – f(x): ascii code of the character
- Situations where the desired function is changing frequently
  – x: description of stock prices and trades for last 10 days
  – f(x): recommended stock transactions
- Situations where each user needs a customized function f
  – x: incoming email message
  – f(x): importance score for presenting to the user (or deleting without presenting)

Supervised Learning



- Training examples: drawn independently at random according to unknown distribution P(x,y)
- The learning algorithm analyzes the training examples and produces a classifier f
- Given a new point <x,y> drawn from P, the classifier is given x and predicts ŷ = f(x)
- The loss L(ŷ,y) is then measured
- Goal of the learning algorithm: Find the f that minimizes the expected loss $E_{P(x,y)}[L(f(x), y)]$

Example – Curve fitting



The underline function:

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

where ε is Gaussian noise

- Input: The Blue circles
    - The circles are generated using the green curve
- Goal: Make ***<u>accurate</u>*** predictions on new **unseen** points such that some notion of error is minimized
    - Predict values of *t* given values of *x*

Curve fitting - Continued

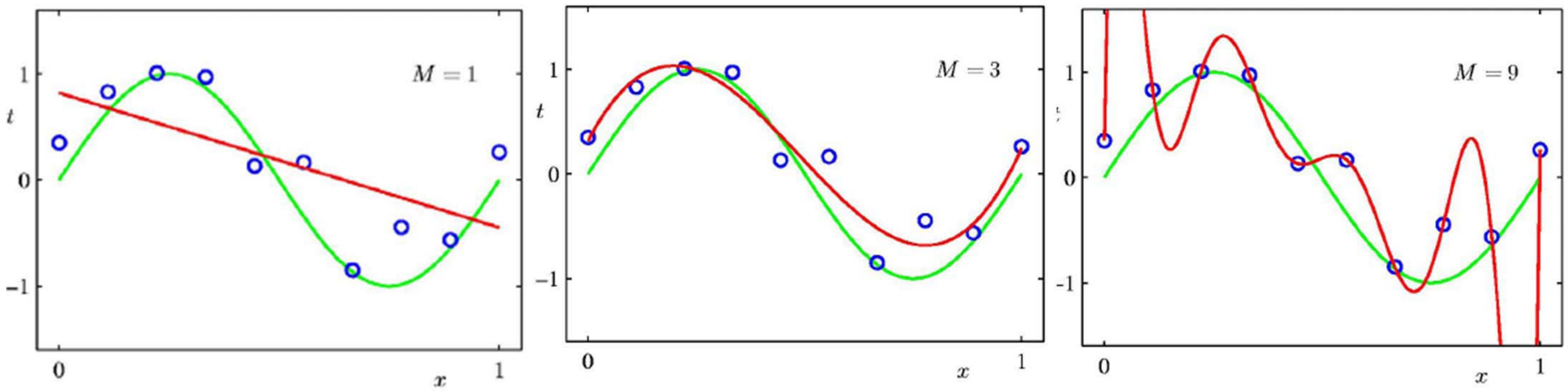
- There are ***infinitely*** many possible functions that can fit this data.
- Hence, we need to focus on a smaller number of functions possible
  - This is called **hypothesis** space
- Ex., we can restrict the order of polynomial (nth order)

$$f(x,w) = w_0 + w_1 x + w_2 x^2 + \ldots + w_n x^n$$

- We wish to learn the parameters $\langle w_0, w_1, \ldots, w_n \rangle$
- These parameters must be learned such that some loss function is minimized
  - For example, squared error $E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(y(x_n,\mathbf{w}) - t_n)^2$

Let us not worry about how it is solved yet ☺ These can be solved easily by optimization techniques.

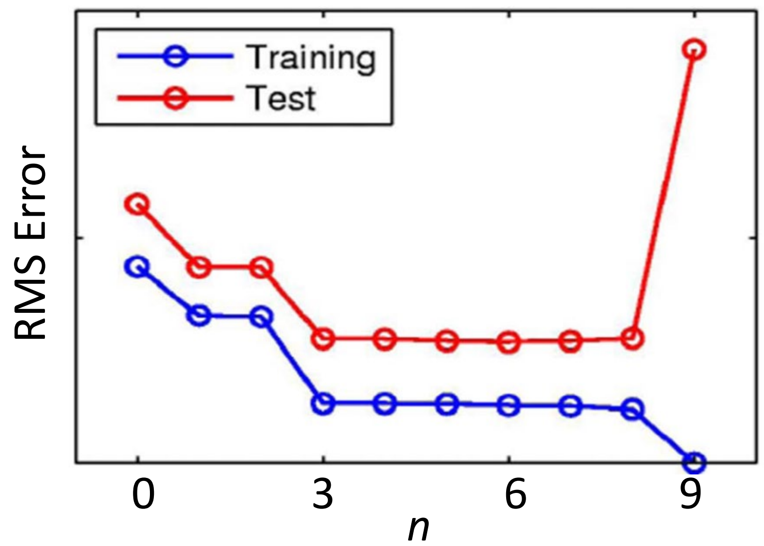
Which Model to choose?



- Red curve is the one with different values for *n*
- Which *n* should we choose – Model selection
- Can we use the error *E(**w**)* as the criterion?

Overfitting

- As *n* increases, training error decreases monotonically
- As *n* increases test error can increase
- Test error can decrease at first, but increases
- Overfitting can occur
  - When the model is too complex and trivially fits the data (i.e., too many parameters)
  - When the data is not enough to estimate the parameters
  - Model captures the noise (or the chance)



Terminology

Terminology

- **Training Example:** <**x**,y>
  - **x:** feature vector (describes the attributes of something)
  - **y:** label (continous values for regression problems: [1,2,…,k] for classification problems)
- **Training set** A set of training examples drawn randomly from P(**x**,y)
  - **Key Assumption:** Independent and identically distributed. i.e., all the examples are drawn from the same distribution but are drawn independent of one another
- **Target function** True mapping from **x** to y
- **Hypothesis:** A function h considered by the learning algorithm to be similar to the target function
- **Test set:** A set of examples drawn from P(**x**,y) to evaluate the "goodness of h"
- **Hypothesis Space:** The space of all hypotheses that can in principle be considered and returned by the learning algorithm

Key approaches

- Directly learn a mapping y = f(**x**)
  - No uncertainty is captured
- Learn the joint distribution i.e., learn p(y**,x)**
  - Captures uncertainty about both the attributes **x** and the target y
- Learn the conditional distribution i.e., learn p(y|**x**)
  - p(**x**,y) = p(y|x)p(x)
  - Hence this avoids modeling the distribution of **x**
  - In general, this is akin to assuming an uniform distribution over **x**
  - Can also be considered as saying "I do not care about **x** but only P(y|**x**)
- Once we learn p, how do we choose y? This is called as decision-theory

Example Training data

x0	x1	x2	x3	y
0	0	1	1	0
1	0	1	0	1
1	1	0	0	0
0	0	1	1	0
0	1	0	0	1
1	0	1	1	1
1	1	0	1	0

Homework Q: Is there a function that is a conjunction of literals that fits the data perfectly?

Two Views

- View1: Learning is the removal of our remaining uncertainty
  - Suppose we know that the function is a conjunction. Then we could use the training example to determine what this function is
- View2: Learning requires guessing a good small hypothesis class
  - We could start with a small hypothesis class and enlarge it till it contains an hypothesis that fits the data
  - For instance, we could say it is a conjunction of 2 literals, then expand to 3 and then say it is may be a disjunction of conjunctions and so on
- But the problem is that we could be wrong
  - Our prior knowledge might be wrong
  - Our guess of hypothesis class itself is wrong – smaller the class higher is the chance of being wrong

Key Questions

- What are good hypothesis spaces?
- What algorithms work on these spaces?
- How can we generalize to unseen points (i.e., avoid overfitting)?
- How can we trust our results?
- Are some problems computationally intractable?
- How can we formulate practical problems as Machine learning ones?

Learning Algorithms

- Search
  - Direct Computation
  - Local Search: Start with an initial hypothesis, make gradual improvements till a local maximum is reached
  - Consructive search: start with an empty hypothesis and grow it till local maximum
- Timing
  - Eager: Analyze training data and construct hypothesis
  - Lazy: Store the training data and wait till a test point is presented, then construct an hypothesis to classify the test point
- Online vs Batch (for eager)
  - Online: Analyze each training point as they arrive
  - Batch: Collect all examples, analyze them together

Next

- Learn a classifier (directly the function f)
- Learn a conditional distribution: Logistic Regression
- Learn the joint distribution: Linear discriminant analysis

- **Key question: How many of you know some basics of probability, maximum likelihood, density estimation etc?**