# Day 15: ML Intro

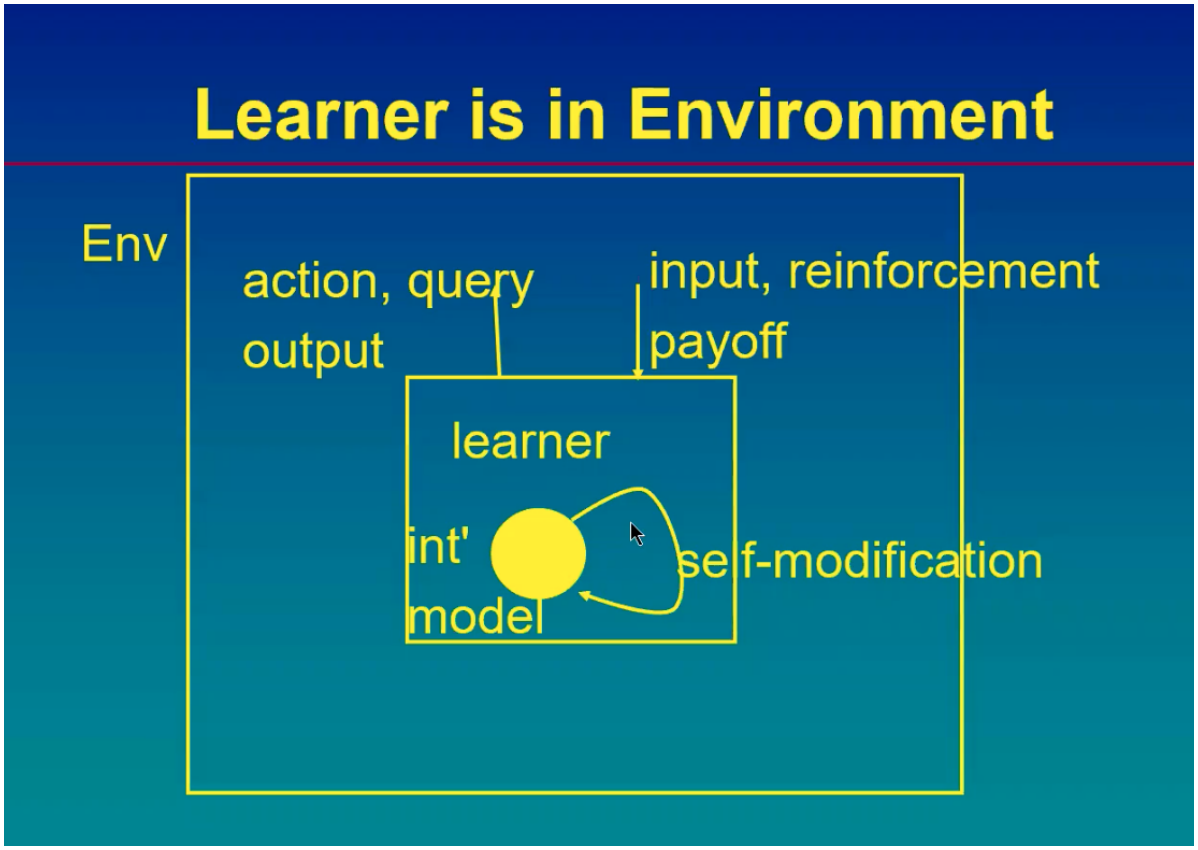
## What is ML?

- ML is a big part of AI, especially now
  - Deep Learning is very popular, but not the only thing
- What is learning?
  - Knowledge acquisition
    - Reduces the role of the programmer
  - Improvement at a task
  - Discovery of new knowledge
    - Scientific discovery
    - Mathematical induction
  - Originally conceived as automatic programming, but that doesn't work in practice
  - Self-organization
    - Evolution models, etc.
- What task does it focus on?
  - Pattern classification/object recognition
  - Playing games even better
  - Transfer learning: can you use knowledge of an old problem to solve a new one (unsolved)
  - How to control a robot (unsolved)
  - Immersion language learning
- There are many different types of learning, examples: Learning through...
  - Rote learning/memorization

    - instruction
    - practice
    - incremental improvement
    - "chunking" experience
    - simulated evolution
    - analysis and discovery
    - analogies
- Major areas of research:
    - Inductive logical learning
    - Inductive classification (supervised learning)
    - Unsupervised conceptual clustering
    - Bio-mimetic learning (genetic algorithms and neural networks)
    - Reinforcement learning
    - Discovery and heuristic learning
    - Computational learning theory
    - Data mining (very industrial)
    - Game learning

# ML Principles

- Big questions of any ML:
    - In practice, ML can generally be seen as a search through a set of concepts/mechanisms
    - What is the problem space? (ex. Automata, algorithms, etc.)
    - What sort of search do we do?
    - What kind of generalizations can it do?
- ML Diagram




- Not every ML system has all these parts
- Sometimes more than one learner

- What goes inside the learner?
  - Each program is one configuration out of a large/infinite set of possible programs.
  - Types of programs include:
    - Logical functions
    - Finite state machines
    - Turing machines
    - Matricies
    - Polynomials
    - Full computer programs
    - Neural Networks

        - Decision Trees
        - Anything else! (although more complex can make learning harder)
- What is the set of possible programs?
    - Much ML uses something more restrictive than general computer programs
        - A small change in a neural net produces a small change in the output, but a small change in a program can easily break it entirely
            - Also, the halting problem makes it easy to get infinite loops in generated programs (and other issues)
        - Doing ML against a full program is obviously desirable though (universality)
- How can the learner change itself? (Self-modification function)
    - Add noise to bits and/or numbers (mutation)
    - Memorizing states (caching as learning)
    - Write a new program with a different algorithm
    - Add/delete internal states
    - Search through a space of possible changes
    - "Thinking": unclear what this means
- What's the environment?
    - At the simplest, it's just the teacher
    - The learner can receive feedback and inputs from the environment (commonly, a payoff/reward function)
    - The learner can query, perform actions, and send outputs to change the environment
    - The environment can also contain other agents (and other learners), ex. for evolution or game theory
    - For most ML apps, the environment is a table of statistically significant sample data

        - The goal is to learn from and generalize the table
        - Table may be of samples, arranged into categories, or scored (with an evaluation function)
- Evaluation: how do you measure learning?
    - For a learner with $k$ bits of input, there are $2^k$ possible inputs and $2^{2^k}$ possible boolean functions
    - If the environment provides $2^j$ bits of feedback (where $j < k$), then there are $2^{2^{k-j}}$ possible generalizations
    - Ex: there are 16 possible 2-bit functions. If you can provide two of the possibilities (ex. $f(0,0) = 0; f(1,0) = 1$) then there are only 4 possibilities.
- Inductive Bias
    - Any restriction on the universe of possible algorithms, or ways to choose between equally-valid solutions (ex. prefer the shortest output)
    - Vital for learning, but limits what can be learned
    - Many failures of AI are improper inductive biases
- Hill Climbing: simple ML
    - Internal model is 12 weights in matricies
    - Self-modifies by adding random noise
    - Environment provides fitness evaluation/score
    - Learner changes itself to maximize the score
    - *He went through this example very quickly, I don't think it was particularly important*.
- ML in Reality
    - Uncommon to have fully autonomous learners
    - Focus on constructing algorithms
    - Induction of rules for pattern classification

            - Environment specified in terms of features and classes
            - Learner must induce simpler description and generalize
            - Training/testing sets
- Two main types of learning tasks:
    - Classification/Categorization ("Supervised" learning)
        - Many-dimensioned data divides into a finite set of classes
        - Goal is to generalize to unseen instances
    - Clustering ("Unsupervised" learning)
        - Many-dimensioned data, cluster similar instances to determine classes
        - Then generalize to unseen instances
- UC Irvine has a big repo of test datasets for ML

# Clustering/Unsupervised Learning

- Technique for finding similar groups in data ("clusters")
- Doesn't require pre-assigning classes to data
- Most famous algorithm: k-means clustering:
    - Let $D = \{x_1, x_2, ..., x_n\}$ be the set of all data points, where each $x_i = (x_{i1}, x_{i2}, ..., x_{ir})$ is a vector in the real-valued $r$-dimensional space $X \subset \mathbb{R}$
    - The algorithm splits the data into $k$ clusters, each with a center (called a *centroid*).
        - $k$ is specified by the user
    - Algorithm:
        1. Randomly choose $k$ data points (*seeds*) to be the initial centroids
        2. Assign each data point to the closest centroids
        3. Re-compute the centroids using the current cluster memberships

        - I *think* this is done by setting the centroid of each cluster to be the mean of all its data points (meaning that the centroid probably won't be a data point).
    4. If it hasn't converged, go to Step 2.
        - This is a heuristic
        - Possibilities:
            - No (or a minimum of) re-assignment of data points between clusters,
            - No (or a minimum of) change of centroids, or
            - A minimum decrease in the **sum of squared error (SSE)**:

$$SSE = \sum_{j=1}^{k} \sum_{x \in C_j} = \mathrm{dist}(x, m_j)^2$$

              Where $C_j$ is the $j$th cluster, $m_j$ is the centroid (mean of all data points) of cluster $C_j$ and $dist(x, m_j)$ is the distance between $x$ and $m_j$.

- Strengths:
    - Simple: easy to understand and implement
    - Efficient: linear time ($t$ and $k$ are constants): $O\left(tkn\right)$, where:
        - $n$ is the number of data points
        - $k$ is the number of clusters
        - $t$ is the number of iterations
- Weaknesses:
    - Terminates at local optimum
    - Need to know how many clusters you want ($k$)
    - Very sensitive to outliers

    - Sensitive to initial seeds (because sensitive to local optimums)
    - Can't deal with complex-shaped clusters (only does ellipsoid)
- Still most popular clustering algorithm despite weaknesses (because nothing is perfect and the trade-offs are good)
- No evidence that any other algorithm is better in the general case (may be different on a case-by-case basis)
- Outstanding question of if this is Machine Learning or statistics (or if there's even a difference)