Machine Learning

Overview and Introduction

What is Learning?

- Improvement at a task
  - Efficiency
  - Coverage
- Knowledge acquisition
  - ML was originally seen as reducing the role of AI programmers
- Discovery of new knowledge
  - Scientific Discovery
  - Mathematical Induction (AM)
- Automatic Programming
- General self organization (60's)

What task?

- Pattern Classification and object recognition
- Improving at Playing Games
- Solving new Problems given old ones
- Controlling Robot bodies
- Learning Language by immersion

Kinds of Learning

- Rote learning and memorization
- Learning by being told
- Learning through practice
- Learning through incremental improvement
- Learning through chunking experience
- Learning through simulating evolution
- Learning through analysis and discovery
- Learning by invention and discovery
- Learning by drawing analogies

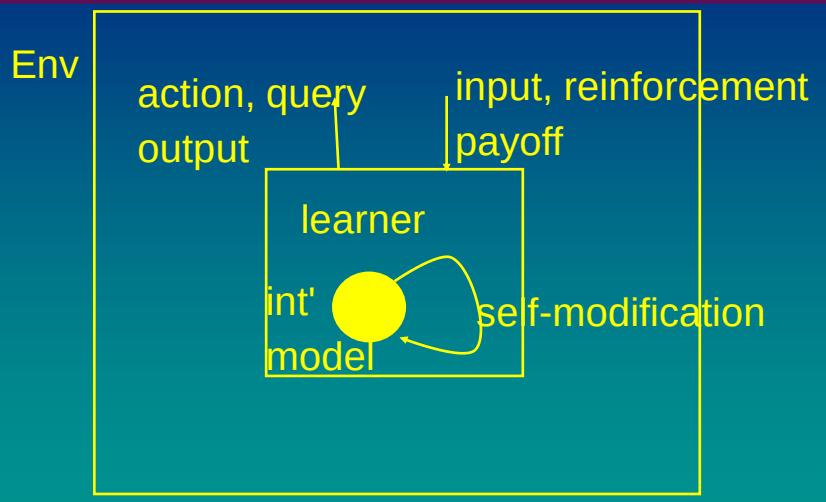
Major Areas of Research in Machine Learning

- Inductive logical learning
- Inductive Classification (Supervised)
- Unsupervised conceptual clustering
- Bio-Mimetic Learning
  - Genetic Algorithms
  - Neural Networks
- Reinforcement Learning
- Discovery and Heuristic Learning
- Computational Learning Theory
- Data-Mining (Industrial ML)
- Game Learning

General Issues

- In general, practical Machine Learning can be seen as **search** through a space of concepts or mechanisms
- What is the space or basis set, e.g. the universe of possibilities?
- How is search conducted?
  - Random, Informed, Strong
- What kind of Generalization?

Learner is in Environment



Learner's Internal Model

- The machine/program is one configuration in a large (infinite?) space of possible machines
  - –Logical functions
  - –Finite State machines
  - –turing machines
  - –Matrices
  - –Polynomials
  - –arbitrary (Lisp) Computer Programs
  - –Neural networks
  - –Decision Trees

What is the universe?

- Much of ML takes place on restrictive enumerable basis sets, rather than involving "general computer programs"
  - WHY?
- Neural Nets, matrices, Polynomials
  - Small changes -> small effects?
- Computer Programs
  - small changes --> Break the program.
- Desire for Universality in Universe.
  - WHY can't we easily learn in the space of universal turing machines?

The Halting Problem

- Is there a Turing machine which can predict whether another Turing machine (given as input) will ever halt?
  - NO!
- So why bother with Machine Learning at all?
  - Cool apps!

Self-Modification Function

- How can the learner change its model?
  - –Adding noise to bits or numbers (mutation)
  - –memorizing states of world (caching as learning)
  - –contructing new program with algorithm
  - –adding/deleting internal states
  - –Searching through space of changes
  - –Thinking!
    - Gradual improvement via training
    - "Kerchunk" AHA! learning
    - evolution learning - "selection" instead of self-modification

Environment

- In the simple situation, the environment is the teacher.
- Learner can receive feedback, read inputs from environment
  - –often in terms of "utility" or payoff function
- learner can query, perform actions - send outputs - to change environment
- Advanced: Environment can contain other agents and learners

Learning Environment

- For most ML Apps Environment is
- A statistically significant sample of the data which must be learned and generalized from
  - A table of samples
    - Numbers, bits, enumerated symbols
    - Signal-processed Samples (e.g. OCR)
  - Arranged into Categories
    - e.g. like lexical-categories
  - Scored
    - like credit-scores

Evaluation

- After learning, does the program Generalize?
- How do you measure learning?
- how do you measure generalization?

Generalization

- Consider a learner with k binary inputs
  - There are 2^k possible inputs
  - Universe of 2^(2^k) boolean functions!
- If environment only provides 2^j bits of feedback, j < k
- there are 2^2^(k-j) possible ways of making valid generalizations

Example: learning binary function from examples

- there are 16 possible boolean functions

P	Q	and				xor	or	=				->			
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Example

- there are 16 possible boolean functions

	P Q	and		xor or	=	->
0	0 0	0	0	0 0		
x	0 1	0	0	1 1		
1	1 0	1	1	1 1		
x	1 1	0	1	0 1		

if we can only establish 2 bits, then there are 4 equivalent functions which can be induced and P itself

Inductive Bias

- Any restriction on the universe, or any "internal" ways to choose between alternatives equally called for by the environment is "inductive bias"
- Conundrum:
  - Bias is necessary for learning
  - Bias limits what can be learned
- Failures of ML claims often by secret biases programmed in by AI researcher

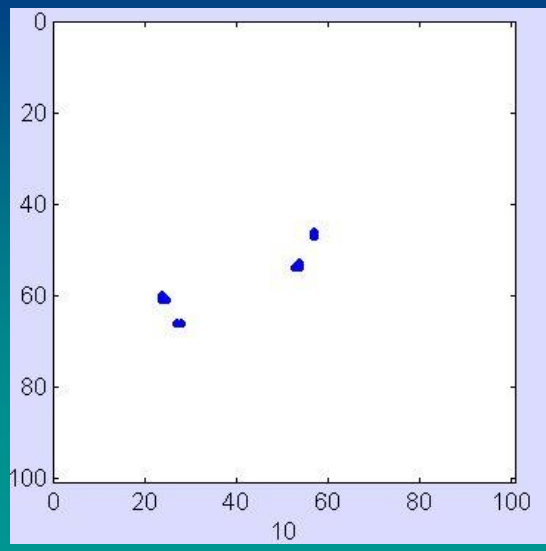
Hill climbing is Simplest ML Example

- Internal Model is 12 weights in matrices
- Self-modification is "Add random noise"
- Environment provides a "fitness" or evaluation or score
- Learner "changes itself" in order to "maximize" score

Simple 12 weight model

- Two 3-2 matrices (6 weights each)
  - A_{Left} and A_{Right}
- Standard Neural Squashing function
  - (x',y')=1/1+e ^ A_{Bit}(x,y,1)
- Infinite random bitstring chooses Left or right Matrix
- Set of all (x,y) States is the output
  - HOW MANY STATE MACHINE?

Score is # pixels in a graph of the set of states

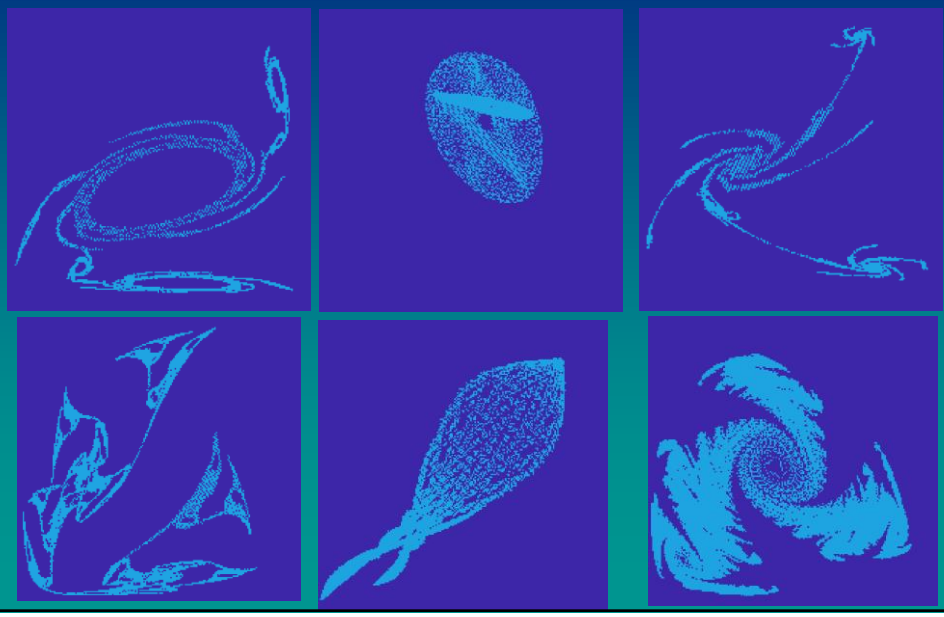


Finding "interesting" networks via hillclimbing

- oldnet = 12 random weights
- Forever do:
  - newnet=oldnet+noise
  - if newnet is "more interesting" than oldnet
  - oldnet=newnet

    - We define interesting as having more pixels on.

Sample Attractors



In Reality

- Few ML systems grant autonomy to learners and vary environments
- Most focus on careful construction of algorithms, for desirable application
- Coming Soon:
- Induction of rules for pattern classification
  - Environment specified in terms of features and classes
  - Learner must induce "compact" description and generalize

two main types of learning tasks

- classification or categorization (Supervised)
  - Learn from instances with features in multiple dimensions
  - finite set of classes
  - Generalize to unseen instances
- Clustering or Unsupervised Learning
  - given a bunch of data in multiple dimensions, determine the classes by clustering similar instances
  - Generalize to unseen instances

Should you get contact lenses?

Age	myopia	astigmatic	tears	category
1	1	2	2	hard
1	2	2	2	hard
2	1	2	2	hard
3	1	2	2	hard
1	1	1	1	none
1	1	2	1	none
3	1	2	1	none
3	2	1	1	none
3	2	2	1	none
3	2	2	2	none
1	1	1	2	soft
1	2	1	2	soft
2	1	1	2	soft
2	2	1	2	soft
3	2	1	2	soft

Introduction to Clustering

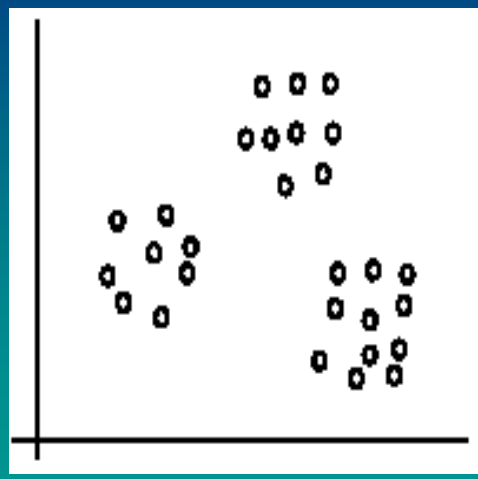
- Supervised learning: discover patterns in the data that relate data attributes with a target (class) attribute.
  - These patterns are then utilized to predict the values of the target attribute in future data instances.
- Unsupervised learning: The data have no target attribute.
  - We want to explore the data to find some intrinsic structures in them.

Clustering

- Clustering is a technique for finding similarity groups in data, called **clusters**. I.e.,
  - it groups data instances that are similar to (near) each other in one cluster and data instances that are very different (far away) from each other into different clusters.
- Clustering is often called an unsupervised learning task as no class values denoting an *a priori* grouping of the data instances are given, which is the case in supervised learning.
- Due to historical reasons, clustering is often considered synonymous with unsupervised learning.

An illustration

- The data set has three natural groups of data points, i.e., 3 natural clusters.



K-means clustering

- Let the set of data points (or instances) D be

  $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$,

  where $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ir})$ is a vector in a real-valued space $X \subseteq R^r$, and r is the number of attributes (dimensions) in the data.
- The *k*-means algorithm partitions the given data into *k* clusters.
  - Each cluster has a cluster **center**, called **centroid**.
  - *k* is specified by the user

K-means algorithm

- Given *k*, the *k-means* algorithm works as follows:
  1)Randomly choose *k* data points (seeds) to be the initial centroids, cluster centers
  2)Assign each data point to the closest centroid
  3)Re-compute the centroids using the current cluster memberships.
  4)If a convergence criterion is not met, go to 2).

K-means algorithm – (cont …)

```
Algorithm k-means(k, D)
1  Choose k data points as the initial centroids (cluster centers)
2  repeat
3     for each data point x ∈ D do
4        compute the distance from x to each centroid;
5        assign x to the closest centroid        // a centroid represents a cluster
6     endfor
7     re-compute the centroids using the current cluster memberships
8  until the stopping criterion is met
```

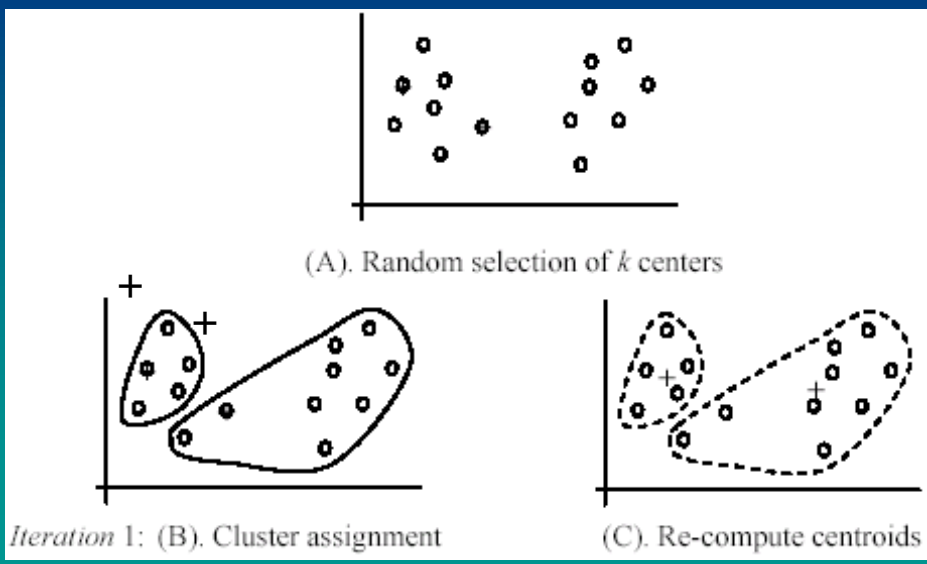
Stopping/convergence criterion

1. no (or minimum) re-assignments of data points to different clusters,
2. no (or minimum) change of centroids, or
3. minimum decrease in the **sum of squared error** (SSE),

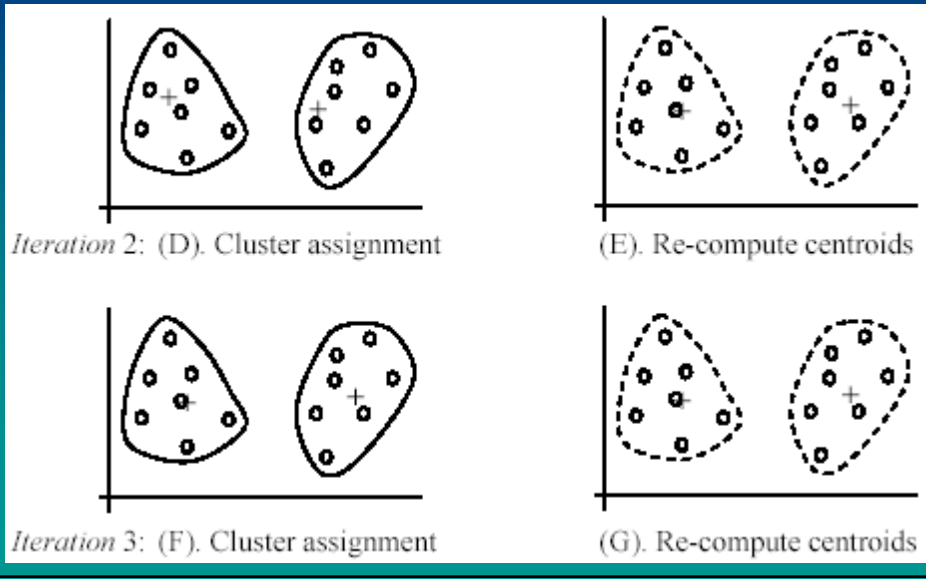
$$SSE = \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} dist(\mathbf{x}, \mathbf{m}_j)^2$$

– C_i is the jth cluster, $\mathbf{m}_j$ is the centroid of cluster C_j (the mean vector of all the data points in C_j), and $dist(\mathbf{x}, \mathbf{m}_j)$ is the distance between data point $\mathbf{x}$ and centroid $\mathbf{m}_j$.

An example



An example (cont …)



Core Code

- (defun distance (v1 v2)
- (sqrt (loop for x in v1 as y in v2 sum (square (- x y)))))
- (defun square (x)(* x x))

- (defun averagev (vectors) (loop for v in (transpose vectors) collect (/ (apply #'+ v) (length vectors))))

- (defun knn (point vectors k)
- (subseq (sort (copy-tree vectors) #'< :key #'(lambda (v) (distance v point))) 0 k))

Core Lisp Code

```
(defun kmcluster (vectors medians)
  (let ((nearestpoint (loop for v in vectors append (knn v medians 1))))
   (loop for point in medians collect
        (loop for class in nearestpoint as v in vectors when (equal point class) collect v))))

(defun kmeans (vectors k)
  (let ((medians (choose vectors k)))
    (loop with newmedians = (mapcar #'averagev (kmcluster vectors medians))
          do (print (setf medians newmedians))
          until (< (apply #'+ (mapcar #'distance medians newmedians)) .1))
    medians))
```

Dataset for clustering

percapita/literacy/infmortality/lifeexpectancy

```
(setf clustest '(
                (Brazil     10326   90      23.6    75.4)
                (Germany    39650   99      4.08    79.4)
                (Mozambq    830     38.7    95.9    42.1)
                (Australia  43163   99      4.57    81.2)
                (China       5300   90.9    23      73)
                (Argentina  13308   97.2    13.4    75.3)
                (England     34105  99      5.01    79.4)
                (SouthAfric 10600   82.4    44.8    49.3)
                (Zambia     1000    68      92.7    42.4)
                (Namibia    5249    85      42.3    52.9)
                (Georgia    4200    100     17.36   71)
                (Pakistan   3320    49.9    67.5    65.5)
                (India      2972    61      55      64.7)
                (Turkey     12888   88.7    27.5    71.8)
                (Sweden     34735   99      3.2     80.9)
                (Lithuania  19730   99.6    8.5     73)
                (Greece     36983   96      5.34    79.5)
                (Italy      26760   98.5    5.94    80)
                (Japan      34099   99      3.2     82.6)))
```

Strengths of k-means

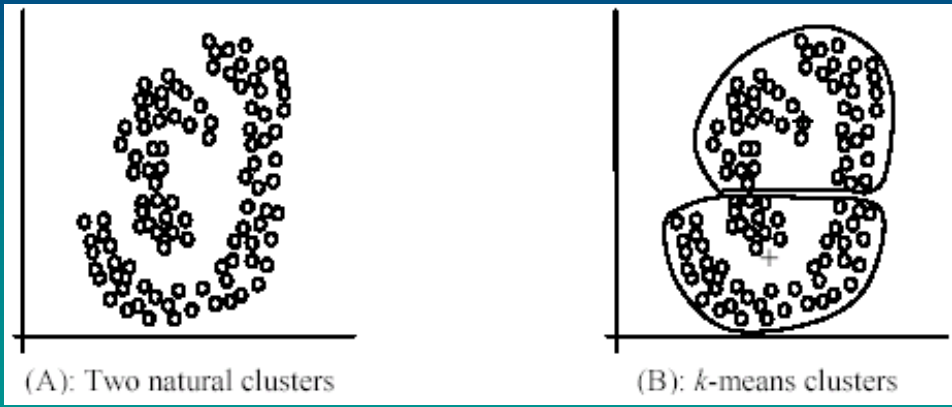
- Strengths:
  - Simple: easy to understand and to implement
  - Efficient: Time complexity: $O(tkn)$,
    where n is the number of data points,
    k is the number of clusters, and
    t is the number of iterations.
  - Since both k and t are small. k-means is considered a linear algorithm.
- K-means is the most popular clustering algorithm.
- Note that: it terminates at a local optimum.

Weaknesses of k-means

- The user needs to specify k.
- The algorithm is sensitive to **outliers**
  - Outliers are data points that are very far away from other data points.
  - Outliers could be errors in the data recording or some special data points with very different values.
- Sensitive to Initial Seeds
- Cant deal with complex shaped clusters

Weaknesses of k-means (cont …)

- The *k*-means algorithm is not suitable for discovering clusters that are not hyper-ellipsoids (or hyper-spheres).



(A): Two natural clusters

(B): *k*-means clusters

K-means summary

- Despite weaknesses, *k*-means is still the most popular algorithm due to its simplicity, efficiency and
  - other clustering algorithms have their own lists of weaknesses.
- No clear evidence that any other clustering algorithm performs better in general
  - although they may be more suitable for some specific types of data or applications.
- Is this really Machine Learning, or just a statistical method?