Machine Learning

CSE 473

Machine Learning Outline

- Machine learning:
  - What & why?
  - Bias
- Supervised learning
  - Classifiers
  - A supervised learning technique in depth
    - Induction of Decision Trees
- Ensembles of classifiers
- Overfitting

Why Machine Learning

- Flood of data
  - WalMart – 25 Terabytes
  - WWW – 1,000 Terabytes
- Speed of computer *vs.* %#@! of programming
  - Highly complex systems (telephone switching systems)
  - Productivity = 1 line code @ day @ programmer
- Desire for customization
  - A browser that browses by itself?
- Hallmark of Intelligence
  - How do children learn language?

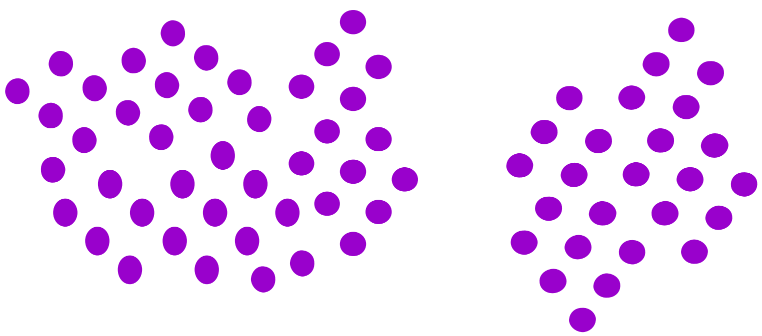
Applications of ML

- Credit card fraud
- Product placement / consumer behavior
- Recommender systems
- Speech recognition

Most mature & successful area of AI

Examples of Learning

- Baby touches stove, gets burned, next time…
- Medical student is shown cases of people with disease X, learns which symptoms…
- How many groups of dots?



What *is* Machine Learning??

Defining a Learning Problem

A program is said to **learn** from experience ***E*** with respect to task ***T*** and performance measure ***P***, if it's performance at tasks in T, as measured by P, improves with experience E.

- Task T:
  Playing Othello
- Performance Measure P:
  Percent of games won against opponents
- Experience E:
  Playing practice games against itself

Issues

- What feedback (experience) is available?
- How should these features be represented?
- What kind of knowledge is being increased?
- How is that knowledge represented?
- What prior information is available?
- What is the right learning algorithm?
- How avoid overfitting?

Choosing the Training Experience

- Credit assignment problem:
  Direct training examples:
    - E.g. individual checker boards + correct move for each
    - Supervised learning
  Indirect training examples :
    - E.g. complete sequence of moves and final result
    - Reinforcement learning
- Which examples:
  Random, teacher chooses, learner chooses

Choosing the Target Function

- What type of knowledge will be learned?
- How will the knowledge be used by the performance program?
- E.g. checkers program

  Assume it knows legal moves

  Needs to choose best move

  So learn function: F: Boards -> Moves
  - hard to learn

  Alternative: F: Boards -> R

Note similarity to choice of problem space

The Ideal Evaluation Function

- V(b) = **100** if b is a final, won board
- V(b) = **-100** if b is a final, lost board
- V(b) = **0** if b is a final, drawn board
- Otherwise, if b is not final

  V(b) = **V(s)** where s is best, reachable final board

Nonoperational…
Want operational approximation of V: $\hat{V}$

How Represent Target Function

- x_1 = number of black pieces on the board
- x_2 = number of red pieces on the board
- x_3 = number of black kings on the board
- x_4 = number of red kings on the board
- x_5 = num of black pieces threatened by red
- x_6 = num of red pieces threatened by black

$$\hat{V}(b) = a + bx_1 + cx_2 + dx_3 + ex_4 + fx_5 + gx_6$$

Now just need to learn 7 numbers!

Example: Othello

- Task T:
  Playing othello
- Performance Measure P:
  Percent of games won against opponents
- Experience E:
  Playing practice games against itself
- Target Function
  V: board -> R
- Representation of approx. of target function

$$\hat{V}(b) = a + bx1 + cx2 + dx3 + ex4 + fx5 + gx6$$

Target Function

- Profound Formulation:

  Can express any type of inductive learning as approximating a function
- E.g., Checkers

  V: boards -> evaluation
- E.g., Handwriting recognition

  V: image -> word
- E.g., Mushrooms

  V: mushroom-attributes -> {E, P}

More Examples

- **Given:** Training examples $\langle \mathbf{x}, f(\mathbf{x}) \rangle$ for some unknown function f.
- **Find:** A good approximation to f.

Example Applications

- **Credit risk assessment**

  $\mathbf{x}$: Properties of customer and proposed purchase.

  $f(\mathbf{x})$: Approve purchase or not.

- **Disease diagnosis**

  $\mathbf{x}$: Properties of patient (symptoms, lab tests)

  $f(\mathbf{x})$: Disease (or maybe, recommended therapy)

- **Face recognition**

  $\mathbf{x}$: Bitmap picture of person's face

  $f(\mathbf{x})$: Name of the person.

More Examples

- Collaborative Filtering
  Eg, when you look at book B in Amazon
  It says "Buy B and also book C together & save!"

- Automatic Steering

Supervised Learning

- **Inductive learning** or "**Prediction**":
  Given examples of a function *(X, F(X))*
  Predict function *F(X)* for new examples *X*

- Classification
  F(X) = Discrete
- Regression
  F(X) = Continuous
- Probability estimation
  F(X) = Probability*(X)*:

Task
Performance Measure
Experience

Why is Learning Possible?

Experience alone never justifies any conclusion about any unseen instance.

Learning occurs when PREJUDICE meets DATA!

Bias

- The nice word for prejudice is "bias".
- What kind of hypotheses will you consider?

  What is allowable ***range*** of functions you use when approximating?
- What kind of hypotheses do you prefer?

Some Typical Bias
The world is simple

Occam's razor

"It is needless to do more when less will suffice"

– William of Occam,

died 1349 of the Black plague

MDL – Minimum description length

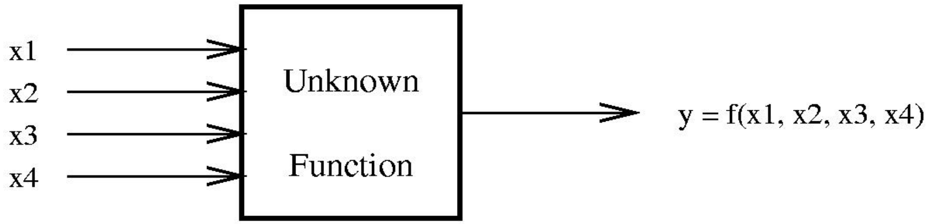
Concepts can be approximated by

... conjunctions of predicates

... by linear functions

... by short decision trees

A Learning Problem



Example	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Hypothesis Spaces

- **Complete Ignorance.** There are $2^{16} = 65536$ possible boolean functions over four input features. We can't figure out which one is correct until we've seen every possible input-output pair. After 7 examples, we still have 2^9 possibilities.

x_1	x_2	x_3	x_4	y
0	0	0	0	?
0	0	0	1	?
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	?
1	0	0	0	?
1	0	0	1	1
1	0	1	0	?
1	0	1	1	?
1	1	0	0	0
1	1	0	1	?
1	1	1	0	?
1	1	1	1	?

Hypothesis Spaces (2)

- **Simple Rules.** There are only 16 simple conjunctive rules.

Rule	Counterexample
$\Rightarrow y$	1
$x_1 \Rightarrow y$	3
$x_2 \Rightarrow y$	2
$x_3 \Rightarrow y$	1
$x_4 \Rightarrow y$	7
$x_1 \wedge x_2 \Rightarrow y$	3
$x_1 \wedge x_3 \Rightarrow y$	3
$x_1 \wedge x_4 \Rightarrow y$	3
$x_2 \wedge x_3 \Rightarrow y$	3
$x_2 \wedge x_4 \Rightarrow y$	3
$x_3 \wedge x_4 \Rightarrow y$	4
$x_1 \wedge x_2 \wedge x_3 \Rightarrow y$	3
$x_1 \wedge x_2 \wedge x_4 \Rightarrow y$	3
$x_1 \wedge x_3 \wedge x_4 \Rightarrow y$	3
$x_2 \wedge x_3 \wedge x_4 \Rightarrow y$	3
$x_1 \wedge x_2 \wedge x_3 \wedge x_4 \Rightarrow y$	3

No simple rule explains the data. The same is true for simple clauses.

Terminology

- **Training example.** An example of the form $\langle \mathbf{x}, f(\mathbf{x})\rangle$.
- **Target function (target concept).** The true function f.
- **Hypothesis**. A proposed function h believed to be similar to f.
- **Concept**. A boolean function. Examples for which $f(\mathbf{x}) = 1$ are called **positive examples** or **positive instances** of the concept. Examples for which $f(\mathbf{x}) = 0$ are called **negative examples** or **negative instances.**
- **Classifier**. A discrete-valued function. The possible values $f(\mathbf{x}) \in \{1, \ldots, K\}$ are called the **classes** or **class labels**.
- **Hypothesis Space**. The space of all hypotheses that can, in principle, be output by a learning algorithm.
- **Version Space**. The space of all hypotheses in the hypothesis space that have not yet been ruled out by a training example.

Two Strategies for ML

- Restriction bias: use prior knowledge to specify a restricted hypothesis space.

    Version space algorithm over conjunctions.

- Preference bias: use a broad hypothesis space, but impose an ordering on the hypotheses.

    Decision trees.

Key Issues in Machine Learning

- **What are good hypothesis spaces?**
  Which spaces have been useful in practical applications and why?
- **What algorithms can work with these spaces?**
  Are there general design principles for machine learning algorithms?
- **How can we optimize accuracy on future data points?**
  This is sometimes called the "problem of overfitting".
- **How can we have confidence in the results?**
  How much training data is required to find accurate hypotheses? (the *statistical question*)
- **Are some learning problems computationally intractable?**
  (the *computational question*)
- **How can we formulate application problems as machine learning problems?** (the *engineering question*)

A Framework for Learning Algorithms

- **Search Procedure**.

  Direction Computation: solve for the hypothesis directly.

  Local Search: start with an initial hypothesis, make small improvements until a local optimum.

  Constructive Search: start with an empty hypothesis, gradually add structure to it until local optimum.

- **Timing**.

  Eager: Analyze the training data and construct an explicit hypothesis.

  Lazy: Store the training data and wait until a test data point is presented, then construct an ad hoc hypothesis to classify that one data point.

- **Online vs. Batch**. (for eager algorithms)

  Online: Analyze each training example as it is presented.

  Batch: Collect training examples, analyze them, output an hypothesis.