Machine Learning - Introduction

Note: These slides have been borrowed and/or adapted from the Machine Learning courses offered by

Pedro Domingos at the University of Washington:
http://www.cs.washington.edu/education/courses/cse546/10wi

Raymond Mooney at the University of Texas at Austin:
http://www.cs.utexas.edu/~mooney/cs391L

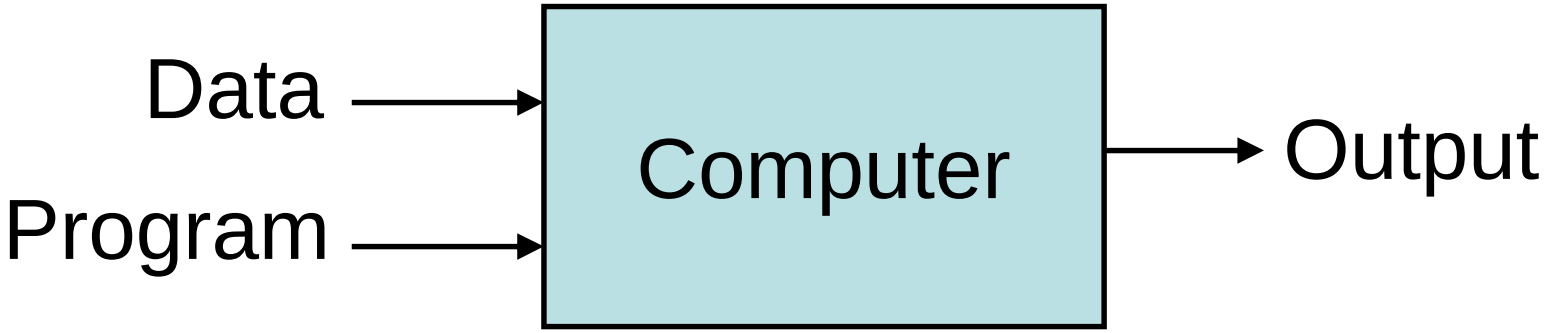
A Few Quotes

- “A breakthrough in machine learning would be worth ten Microsofts” (Bill Gates, Chairman, Microsoft)
- “Machine learning is the next Internet”
  (Tony Tether, Director, DARPA)
- Machine learning is the hot new thing”
  (John Hennessy, President, Stanford)
- “Web rankings today are mostly a matter of machine learning” (Prabhakar Raghavan, Dir. Research, Yahoo)
- “Machine learning is going to result in a real revolution”
  (Greg Papadopoulos, CTO, Sun)
- “Machine learning is today’s discontinuity”
  (Jerry Yang, CEO, Yahoo)

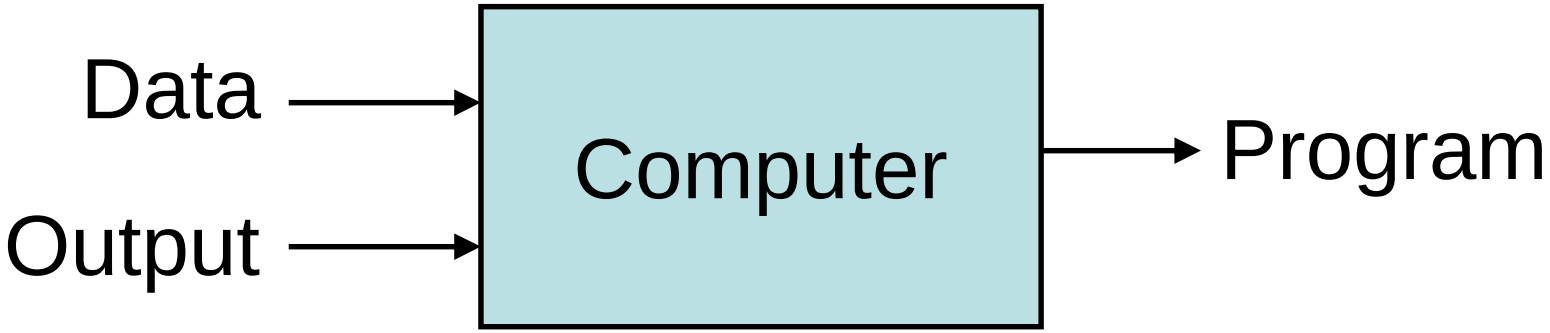
So What Is Machine Learning?

- Automating automation
- Getting computers to program themselves
- Writing software is the bottleneck
- Let the data do the work instead!

Traditional Programming



Machine Learning



Magic?

No, more like gardening

- **Seeds** = Algorithms
- **Nutrients** = Data
- **Gardener** = You
- **Plants** = Programs

What is Learning?

- Herbert Simon: “Learning is any process by which a system improves performance from experience.”

Defining the Learning Task

Improve on task, T, with respect to
performance metric, P, based on experience, E.

T: Playing chess
P: Percentage of games won against an arbitrary opponent
E: Playing practice games against itself

T: Recognizing hand-written words
P: Percentage of words correctly classified
E: Database of human-labeled images of handwritten words

T: Driving on four-lane highways using vision sensors
P: Average distance traveled before a human-judged error
E: A sequence of images and steering commands recorded while observing a human driver.

T: Categorize email messages as spam or legitimate.
P: Percentage of email messages correctly classified.
E: Database of emails, some with human-given labels

Why Study Machine Learning?

Engineering Better Computing Systems

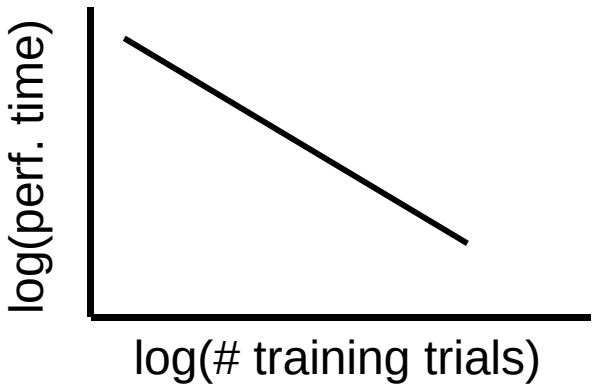
- Develop systems that are too difficult/expensive to construct manually because they require specific detailed skills or knowledge tuned to a specific task (***knowledge engineering bottleneck***).
- Develop systems that can automatically adapt and customize themselves to individual users.
  - Personalized news or mail filter
  - Personalized tutoring
- Discover new knowledge from large databases (***data mining***).
  - Market basket analysis (e.g. diapers and beer)
  - Medical text mining (e.g. migraines to calcium channel blockers to magnesium)

Why Study Machine Learning?

Cognitive Science

- Computational studies of learning may help us understand learning in humans and other biological organisms.

- Power law of practice



Why Study Machine Learning?

The Time is Ripe

- Many basic effective and efficient algorithms available.
- Large amounts of on-line data available.
- Large amounts of computational resources available.

Sample Applications

- Web search
- Computational biology
- Finance
- E-commerce
- Space exploration
- Robotics
- Information extraction
- Social networks
- Debugging
- [Your favorite area]

ML in a Nutshell

- Tens of thousands of machine learning algorithms
- Hundreds new every year
- Every machine learning algorithm has three components:
  - **Representation**
  - **Evaluation**
  - **Optimization**

Representation

- Decision trees
- Sets of rules / Logic programs
- Instances
- Graphical models (Bayes/Markov nets)
- Neural networks
- Support vector machines
- Model ensembles
- Etc.

Evaluation

- Accuracy
- Precision and recall
- Squared error
- Likelihood
- Posterior probability
- Cost / Utility
- Margin
- Entropy
- K-L divergence
- Etc.

Optimization

- Combinatorial optimization
  - E.g.: Greedy search
- Convex optimization
  - E.g.: Gradient descent
- Constrained optimization
  - E.g.: Linear programming

Types of Learning

- **Supervised (inductive) learning**
  - Training data includes desired outputs
- **Unsupervised learning**
  - Training data does not include desired outputs
- **Semi-supervised learning**
  - Training data includes a few desired outputs
- **Reinforcement learning**
  - Rewards from sequence of actions

Inductive Learning

- **Given** examples of a function *(X, F(X))*
- **Predict** function *F(X)* for new examples *X*
  - Discrete *F(X)*: Classification
  - Continuous *F(X)*: Regression
  - *F(X)* = Probability(*X*): Probability estimation

ML in Practice

- Understanding domain, prior knowledge, and goals
- Data integration, selection, cleaning, pre-processing, etc.
- Learning models
- Interpreting results
- Consolidating and deploying discovered knowledge
- Loop

Inductive Learning

Supervised Learning

- **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.

- **Automatic Steering**

  $\mathbf{x}$: Bitmap picture of road surface in front of car.

  $f(\mathbf{x})$: Degrees to turn the steering wheel.

Appropriate Applications for Supervised Learning

- **Situations where there is no human expert**

  $\mathbf{x}$: Bond graph for a new molecule.

  $f(\mathbf{x})$: Predicted binding strength to AIDS protease molecule.

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

  $\mathbf{x}$: Bitmap picture of hand-written character

  $f(\mathbf{x})$: Ascii code of the character

- **Situations where the desired function is changing frequently**

  $\mathbf{x}$: Description of stock prices and trades for last 10 days.

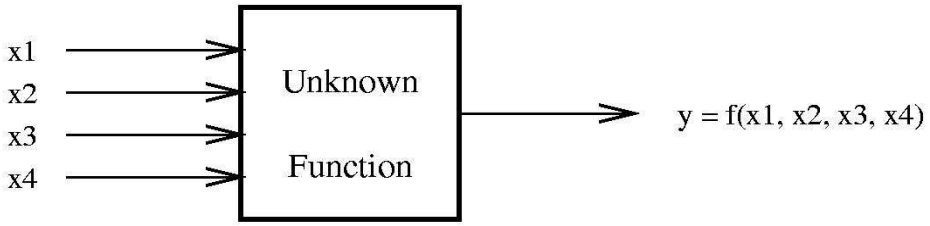
  $f(\mathbf{x})$: Recommended stock transactions

- **Situations where each user needs a customized function f**

  $\mathbf{x}$: Incoming email message.

  $f(\mathbf{x})$: Importance score for presenting to user (or deleting without presenting).

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.

Hypothesis Space (3)

- m-**of-**n **rules**. There are 32 possible rules (includes simple conjunctions and clauses).

	Counterexample			
variables	1-of	2-of	3-of	4-of
$\{x_1\}$	3	–	–	–
$\{x_2\}$	2	–	–	–
$\{x_3\}$	1	–	–	–
$\{x_4\}$	7	–	–	–
$\{x_1, x_2\}$	3	3	–	–
$\{x_1, x_3\}$	4	3	–	–
$\{x_1, x_4\}$	6	3	–	–
$\{x_2, x_3\}$	2	3	–	–
$\{x_2, x_4\}$	2	3	–	–
$\{x_3, x_4\}$	4	4	–	–
$\{x_1, x_2, x_3\}$	1	3	3	–
$\{x_1, x_2, x_4\}$	2	3	3	–
$\{x_1, x_3, x_4\}$	1	***	3	–
$\{x_2, x_3, x_4\}$	1	5	3	–
$\{x_1, x_2, x_3, x_4\}$	1	5	3	3

Two Views of Learning

- **Learning is the removal of our remaining uncertainty**. Suppose we *knew* that the unknown function was an m-of-n boolean function, then we could use the training examples to infer which function it is.
- **Learning requires guessing a good, small hypothesis class**. We can start with a very small class and enlarge it until it contains an hypothesis that fits the data.

We could be wrong!

- **Our prior knowledge might be wrong**
- **Our guess of the hypothesis class could be wrong**
  The smaller the hypothesis class, the more likely we are wrong.

Example: $x_4 \wedge Oneof\{x_1, x_3\} \Rightarrow y$ is also consistent with the training data.

Example: $x_4 \wedge \neg x_2 \Rightarrow y$ is also consistent with the training data.

If either of these is the unknown function, then we will make errors when we are given new x values.

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.

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 Hypothesis Spaces

- **Size**. Does the hypothesis space have a **fixed size** or **variable size**?
  Fixed-size spaces are easier to understand, but variable-size spaces are generally more useful. Variable-size spaces introduce the problem of overfitting.

- **Randomness**. Is each hypothesis **deterministic** or **stochastic**?
  This affects how we evaluate hypotheses. With a deterministic hypothesis, a training example is either *consistent* (correctly predicted) or *inconsistent* (incorrectly predicted). With a stochastic hypothesis, a training example is *more likely* or *less likely*.

- **Parameterization**. Is each hypothesis described by a set of **symbolic** (discrete) choices or is it described by a set of **continuous** parameters? If both are required, we say the hypothesis space has a **mixed** parameterization.
  Discrete parameters must be found by combinatorial search methods; continuous parameters can be found by numerical search methods.

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.