More Than A Program

- Usually, we think a program is something written by an experienced person.
- Often, the program isn’t complete without “experience” of its own.
- **Example**: If I had to write a compression program, I could do ok. But, using the character-count statistics from the Gettysburg Address makes it possible to produce the most compact code.
Machine Learning

• *Machine learning* is the idea of writing programs that use data (experience) to create better programs than people can write directly.

• *Supervised learning* is a specific problem in which the experience is in the form of labeled examples and the learning system needs to learn to label novel examples.

Classification: Idea

• In its simplest form, a learner’s job is to produce a *classifier*.

• A classifier takes objects as input and assigns each one to a *class*.

• Most simply, objects are represented as *vectors of features* and classes are 0/1.
Future Killer?

Example Classifier

- **Input**: A high school student
- **Output**: Will the student drop out of college?
- **Vector of Features**: Score on SATs, grades in Math/English/Science, age, parent’s income, years at current address, height
- Such a classifier *might* be useful as a tool for admissions or financial aid.
Learning: The Problem

- **Input**: A *training set* consisting of *labeled instances*, each of which is a feature vector and a desired class (1 = yes, 0 = no).
- **Output**: A classifier, which we hope will accurately assign new feature vectors to classes.
- A *learning algorithm* is a program that addresses this problem.

Informal to Formal

- Note that problem is not really defined at this point! Whether it does well on new examples isn’t directly measurable.
- One way to make it formal: find classifier with minimum number of mistakes on the training set.
- Usually not enough to create good classifiers: *overfitting*.
Learning Algorithms

• Decision trees
• Boosting
• Nearest neighbors
• Support Vector Machines
• Neural networks
• Naive Bayesian classifiers
• etc.

Some Applications

• Autonomous driving (camera ⇒ steering wheel)
• Loan applications (age, income ⇒ default?)
• Handwriting recognition (scanner ⇒ digits)
• Speech recognition (audio ⇒ phonemes)
Decision Trees

- Decision trees are another kind of classifier.
- Branch depending on test on attribute value.
- Can build such a tree incrementally and automatically using examples.

ML & Robotics

- Machine learning is almost always grounded in measurable data.
- Computer programs that can collect and use their own data can be quite powerful.
- Essentially, that’s what learning robots are.
- International conference at Rutgers on Epigenetic Robotics!
While True

- What sorts of program would *purposely* have an infinite loop?
- Think about a software-controlled thermostat. It might have a program that looks something like:

```python
def thermostat(low, high):
    while True:
        t = currentTemp()
        if t >= high:
            runAC(4)
        elif t < low:
            runHeat(2)
thermostat(68, 75)
```

Loop Forever

- operating systems
- user interfaces
- video games
- process controllers
- robots
Robot Basics

- From a software standpoint, modern robots are just computers.
- Typically, they have less memory and processing power than a standard computer.
- Sensors and effectors under software control.

Standard Robots

- Industrial manufacturing robots.
- Research /hobby robots.
- Demonstration robots.
- Home robots.
- Planetary rovers.
- Movie robots.
Manufacturing

- Often arms, little else.
- Part sorting.
- Painting.
- Repeatable actions.

Research / Hobby

- Pioneer
- Handy Board / Lego
- Segbot
- Stanley
Space Exploration

• Sojourner
• Deep Space Agent

Home Robots

• Roomba.
• Mowers.
• Moppers.
• Big in Japan.
• Nursebots.
• Emergency rescue bots, Aibo.
Demonstration Robots

- Honda: Asimo.
- Toyota: lip robot.
- Sony: QRio.

Sensors and Effectors

**Sensors:**
- bump
- infrared
- vision
- light
- sonar
- sound

**Effectors:**
- motors
- lights
- sounds
- graphical display
- laser
Simple Learning

- Words: “hello”, “don’t do that”, “sit”, “stand up”, “lie down”, “shake paw”

Example Code

```python
act[0] = 0
act[1] = 0
actions = ["lay6", "sit2", "sit4", "stand2", "stand9"]
lastact = 0
while True:
    cmd = Voice()
    if cmd == "sit":
        doAction(actions[act[0]])
        lastact = 0
```

```python
elif cmd == "stand":
    doAction(actions[act[1]])
    lastact = 1
elif cmd == "good Aibo":
    doAction("happy")
elif cmd == "bad dog":
    doAction("sad sound")
act[lastact] = (act[lastact] + 1) % 4
```
Trainer: In Words

• For each recognized voice command, there is an associated action program.
• When a voice command is recognized, the corresponding action is taken.
• On “Good Aibo”, nothing needs to change.
• On “Don’t do that”, the most recent command needs a different action program. It is incremented to the next on the list.

Impressive Accomplishment

Honda’s Asimo

• development began in 1999, building on 13 years of engineering experience.
• claimed “most advanced humanoid robot ever created”
• walks 1mph
And Yet…

Asimo is programmed/controlled by people:

- structure of the walk programmed in
- reactions to perturbations programmed in
- directed by technicians and puppeteers during the performance
- no camera-control loop
- static stability

Compare To Kids

Molly

- development began in 1999
- “just an average kid”
- walks 2.5mph even on unfamiliar terrain
- very hard to control
- dynamically stable (sometimes)
Crawl Before Walk

Impressive accomplishment:

- Fastest reported walk/crawl on an Aibo
- Gait pattern optimized automatically

Human “Crawling”

Perhaps our programming isn’t for crawling at all, but for the desire for movement!
My Research

How can we create smarter machines?

- Programming
  - tell them exactly what to do
  - “give a man a fish...”
- Programming by Demonstration (supervised learning)
  - show them exactly what to do
  - “teach a man to fish...”
- Programming by Motivation (reinforcement learning)
  - tell them what to want to do
  - “give a man a taste for fish...”

Find The Ball Task

Learn:

- which way to turn
- to minimize time
- to see goal (ball)
- from camera input
- given experience.
In Other Words...

- It “wants” to see the pink ball.
- Utility values from seeing the ball and the cost of movement come from the *reward function*.
- It gathers experience about how its behavior changes the state of the world.
- We call this knowledge its *transition model*.
- It selects actions that it predicts will result in maximum reward (seeing the ball soon).
- This computation is often called *planning*.

---

**Exploration/Exploitation**

Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth;

- In RL: A system cannot make two decisions and be one decision maker---it can only observe the effects of the actions it *actually chooses to make*. Do the right thing *and* learn.
• We’re building artificial decision makers as follows:
  • We define perceptions, actions, and rewards (including shaping rewards to aid learning).
  • Learner explores its environments to discover:
    • What actions do
    • Which situations lead to reward
  • Learner uses this knowledge via “planning” to make decisions that lead to maximum reward.