Frontiers of Computer Science

Eric Roberts
CSCI 121
December 11, 2019

Requirements for the CS Major at Reed

Computer Science 382 – Algorithms and Data Structures
Full course for one semester. An introduction to computer science covering the design and analysis of algorithms. The course will focus on various abstract data types and associated algorithms. The course will include implementation of some of these ideas on a computer.
Prerequisite: Computer Science 121 or equivalent, and one of Mathematics 112 or 113. Conference.

Computer Science 387 – Computability and Complexity
Full course for one semester. Introduction to models of computation including finite automata, formal languages, and Turing machines, culminating in universality and undecidability. An introduction to resource-bounded models of computation and algorithmic complexity classes, including NP and PSPACE, and the notions of relative hardness and completeness.
Prerequisite: Computer Science 121 or equivalent, and Mathematics 112 and 113. Lecture-conference. Cross-listed as Mathematics 387.

Designing the Turing Machine

- In his groundbreaking 1936 paper, “On computable numbers, with an application to the Entscheidungsproblem,” Turing described the process of computation in informal terms:

  It may be that some of these changes necessarily involve a change of state of mind. The most general single operation must therefore be taken to be one of the following:
  (a) A possible change of symbol together with a possible change of state of mind.
  (b) A possible change of observed squares, together with a possible change of state of mind.
  The operation actually performed is determined, as has been suggested above, by the state of mind of the computer and the observed symbols.

- These operations and the notion of a “state of mind” form the basis for the Turing machine.

Turing Machine Components

Computation requires:
- Scratch paper
- An unbounded amount of space
- At least two symbols
- A read/write mechanism
- Some form of program control
A Sample Turing Machine

```
0 0 0 1 1 1 1 0 0 0 0
```

1

2

1L1

1L2

1R2

The Halting Problem in Python

- Turing showed that no program can reliably tell whether another program will halt.
- Suppose there were a function called `doesProgramHalt` that takes a filename and determines whether that program halts.
- If that function exists, you could write the following program:

```python
# File: Paradox.py
# This program uses the assumption that doesProgramHalt exists to generate a paradox.

def paradox():
    if doesProgramHalt("Paradox.py"):
        print("The program runs forever.")
        while True:
            pass
    else:
        print("The program halts.")

if __name__ == "+_main_":
    paradox()
```

The Church-Turing Thesis

- The question of what is computable by a Turing machine is important in a search for what is generally computable mostly because no one has ever found a more powerful model.
- Most computer scientists believe what has come to be known as the Church-Turing thesis:

  No method of computation carried out by a mechanical process can be more powerful than a Turing machine.

- This claim remains a conjecture, and it is not clear there is any way to prove it. At the same time, it has so far resisted all efforts to disprove it.

The 3-D Fax Machine

1. Start with a statue of the Happy Buddha.
2. Use laser range scanning to produce a triangle mesh from a single perspective.
3. Merge scans from different perspectives.
5. Use stereolithography to construct a lucite copy.
### Scanning Michelangelo’s David

- Some years ago, my Stanford colleague Marc Levoy spent a year at Stanford’s Overseas Studies campus in Florence, at which he and about 30 students used the technology developed for the 3-D fax machine project to scan the Michelangelo sculptures open to the public.
- The 3-D fax machine technology allowed Marc to construct a model of each statue with sub-millimeter accuracy—accurate enough to tell when the sculptor changed chisels. Mark’s work makes it possible to view statues from any perspective and enables mathematical analysis.

### The Forma Urbis Romae Project

- While in Italy, Professor Levoy and his students also scanned fragments of the *Forma Urbis Romae*, a huge marble map from the 3rd century AD that is now a jigsaw puzzle with 1,186 pieces.
- Scanning the fragments makes it possible to use computational techniques to reassemble the pieces of the puzzle. The first match found by Levoy’s team is shown at the right. Since that time, the Stanford group has been able to determine the placement of missing pieces at a far faster rate than was previously possible.

### Machine-Learning Strategies

- There are three principal strategies for machine learning:
  - **Supervised learning**, in which the program starts with training data that has already been classified by human observers, thereby giving the program enough information to make correct inferences on examples it has not yet seen.
  - **Unsupervised learning**, in which the program receives no information about the examples but instead tries to identify patterns.
  - **Reinforcement learning**, in which the program learns by receiving rewards for good outcomes and punishments for bad ones.
- These strategies can produce either of two results:
  - **Classification**, in which the answer is a category.
  - **Regression**, in which the answer is a continuous value.

### Unsupervised Learning

- The goal of unsupervised learning is to find patterns even in the absence of a training set.
- The most common applications of unsupervised learning involve clustering, which is the process of dividing a data set into some number of independent, closely related clusters.
- As an example, if you have plotted a set of points as shown in the x-y grid at the bottom of the slide, unsupervised learning should be able to identify the three groups based on locality.

### K-Means Clustering Algorithm

- A common strategy for clustering data points is the *k-means algorithm*, which partitions a set into k clusters.
- The *k*-means algorithm requires the following steps:
  1. Choose k data points at random to serve as the initial centers of the clusters. These points are called means.
  2. Assign each point to the closest mean.
  3. Move each mean to the geometric center of its points.
  4. Repeat steps 2 and 3 until the means stop changing.
A Larger Clustering Example ($k = 5$)

1. Choose five random points as the initial means.
2. Assign each point to the closest mean.
3. Move each mean to the geometric center of its points.

Reinforcement Learning

- Reinforcement learning uses positive and negative feedback from previous experience to control future behavior.
- Algorithms that use reinforcement learning algorithms keep track of earlier decisions along with a score indicating how well those decisions worked.
- Successful outcomes increase the score of all decisions made to reach that goal. Symmetrically, failures decrease the score.
- You could, for example, write a program that learns to play Tic-Tac-Toe simply by keeping track of the moves the program makes in the game, rewarding each win by increasing the score for every move that led to the victory and punishing each loss by decreasing those scores.

Neural Networks

- In AI today, the hottest topic is **neural networks**, which are collections of nodes that simulate neurons in the brain.
- Each neuron receives signals from **dendrites**. When the input signals at a neuron reaches a threshold, the neuron fires a signal down an **axon**, which transmits it to other neurons.

Artificial Neural Networks

- In an artificial neural network, the biological neurons are replaced by nodes in a graph, which is organized into layers.
- Typically, a network contains an **input layer** that serves as the source for the data driving the network, an **output layer** that registers the results, and some number of **hidden layers** between the two.

The Operation of an Artificial Neuron

- Each node in a neural network simulates the function of a biological neuron by combining signals from its inputs:
  - The signals appear at the left.
  - Each signal is adjusted by a weight as it enters the neuron.
  - The weighted signals are added together.
  - The node calls a threshold function to compute the output.
  - The signal is then propagated to the next layer.

Learning in a Neural Network

- Reinforcement learning in a neural network is implemented using a technique called **backpropagation**:  
  - The output signals are compared to their target values.
  - The errors are propagated backward to adjust the weights.
  - Weights change according to their influence on the output.
  - This process involves partial derivatives and the chain rule.