Finding Love: An Algorithmic Approach

Slides by Griffin Kao for CS0040: Introduction to Scientific Computing at Brown University

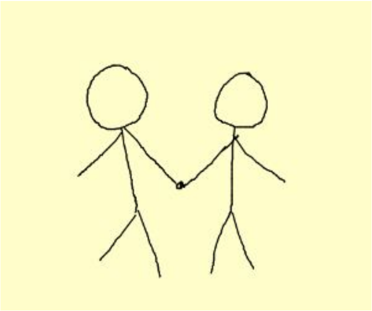
Motivation

“Online Algorithms”

- Up until now, we’ve had all the information we need to solve our problems
    - Deciding an investment plan, like you did in HW 2 is significantly more difficult if we don’t know all the list prices ahead of time
- So what do we do if we have to do well without knowing what the future holds…?

The Dating Problem

- Throughout your life you date a series of people one by one with who you have varying degrees of compatibility
- How do you pick The One? (maximize expected reward)
- Best Case: You pick the one you’re most compatible with 👀
- Worst Case: You end up alone 😢 or even worse with someone you hate 😖



Assumptions

- Quantify compatibility (we'll return to this later!)
- Date n people, one person at a time
- Can't undo: if you ask your ex to take you back, they say f u
- Random order
- Can't predict the future

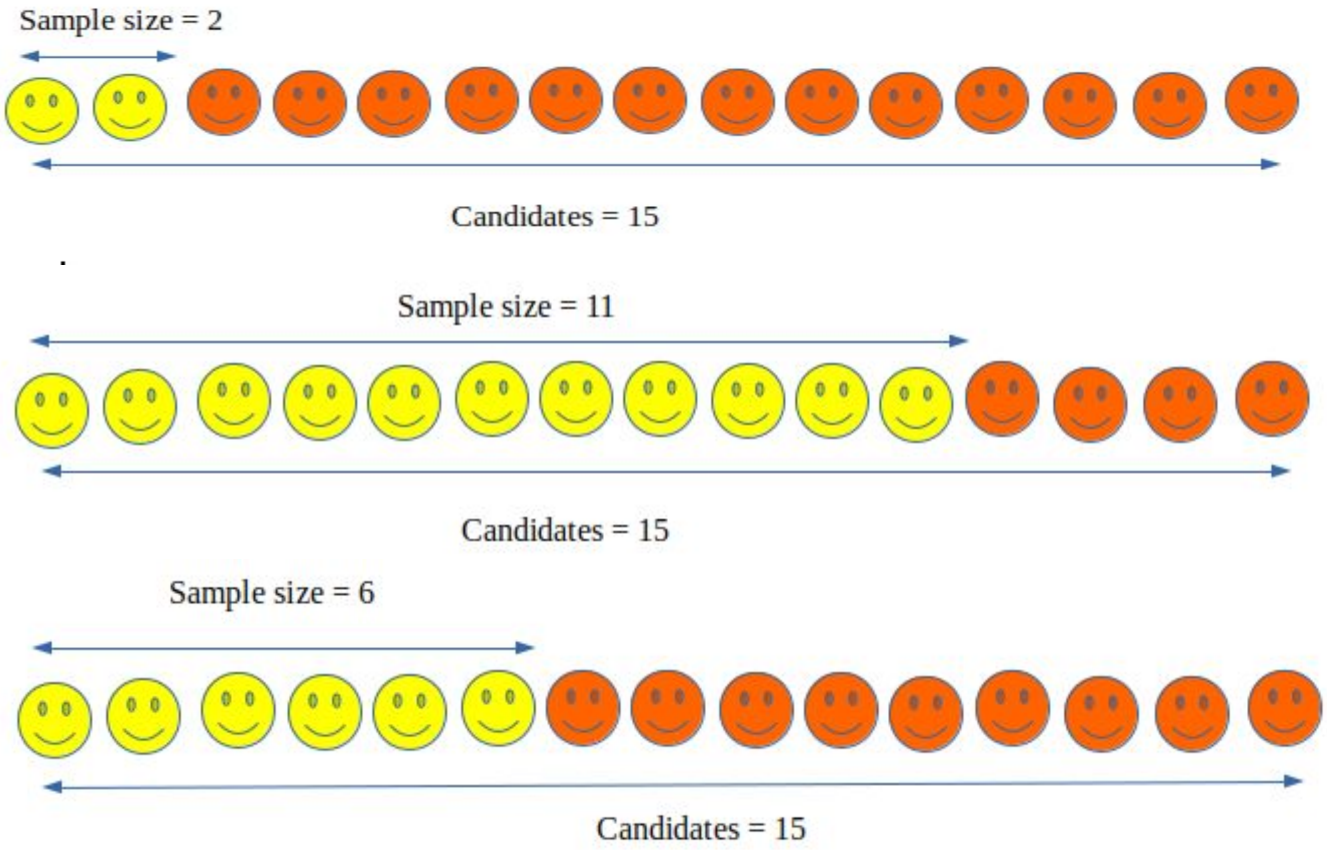
Strategy

- Date and reject the first r-1 people
- Let M be the best of those
- Then choose the next one better than M

Problems

- What if you stop dating too early and miss The One later on?
- What if you pass up on The One and your current partner ends up walking away after 25 years of marriage (with half your multibillion dollar Amazon fortune)???
- When should you stop and select the best SO? Or just stop and just pick the next one since you're super desperate!

How do we pick r?


Sample size = 2
Candidates = 15
Sample size = 11
Candidates = 15
Sample size = 6
Candidates = 15

Probability of Success

For arbitrary cutoff r

$$
\begin{aligned}
P(r) &= \sum_{i=1}^{n} P\left(\text{applicant } i \text{ is selected} \cap \text{applicant } i \text{ is the best}\right) \\
&= \sum_{i=1}^{n} P\left(\text{applicant } i \text{ is selected} | \text{applicant } i \text{ is the best}\right) \cdot P\left(\text{applicant } i \text{ is the best}\right) \\
&= \left[\sum_{i=1}^{r-1} 0 + \sum_{i=r}^{n} P\left(\begin{array}{l}\text{the best of the first } i-1 \text{ applicants} \\ \text{is in the first } r-1 \text{ applicants}\end{array} \middle| \text{applicant } i \text{ is the best}\right)\right] \cdot \frac{1}{n} \\
&= \left[\sum_{i=r}^{n} \frac{r-1}{i-1}\right] \cdot \frac{1}{n} \quad = \quad \frac{r-1}{n} \sum_{i=r}^{n} \frac{1}{i-1}.
\end{aligned}
$$

(Taken from Wikipedia)

Choose r to Maximize P

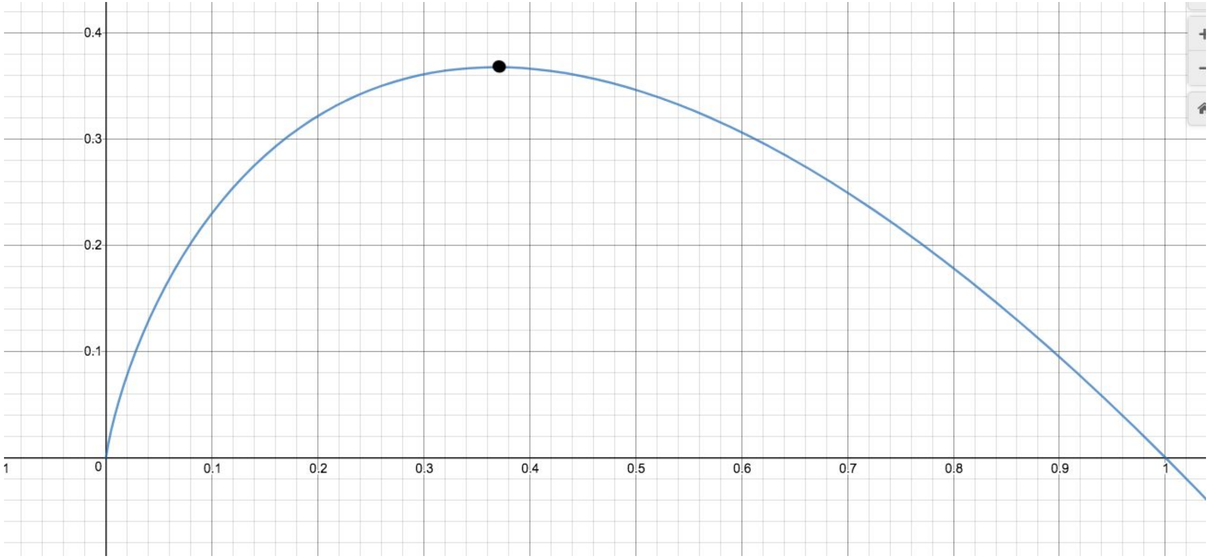
- Let n tend towards infinity, x = lim (r-1)/n, t = (i-1)/n, dt = 1/n
- We can approximate the sum with an integral (basically scaling from n to 1):

$$P(x) = x \int_x^1 \frac{1}{t}\, dt = -x \ln(x)$$

- So P'(x) = -ln(x) - 1 by product rule, set = 0 and solve for x
- What do we get?

Solution: Optimal Stopping

x = 1/e !!!!!!! 🤪😆😃👏 (which gives us r = n/e)



Reject the first 37% of the people you date :)

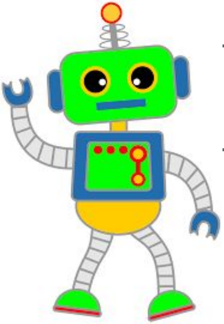
Pseudocode

```
Def doDating(numSuitors, life):
	r = numSuitors/e
	best = life.suitors[1]
	for i in 1 to r-1:
		if compatibility(life.suitors[i]) > compatibility(best):
			best = life.suitors[i]
	for i in r to numSuitors:
		if compatibility(life.suitors[i]) > compatibility(best):
			return life.suitors[i]
	return life.suitors[numSuitors]
```

1/e Other Applications

- Options Trading
    - Imagine you’re holding an American option and can buy/sell underlying asset before expiry date
    - Value of underlying asset (i.e. stock price) follows geometric Brownian motion => random!
    - When do you exercise option??

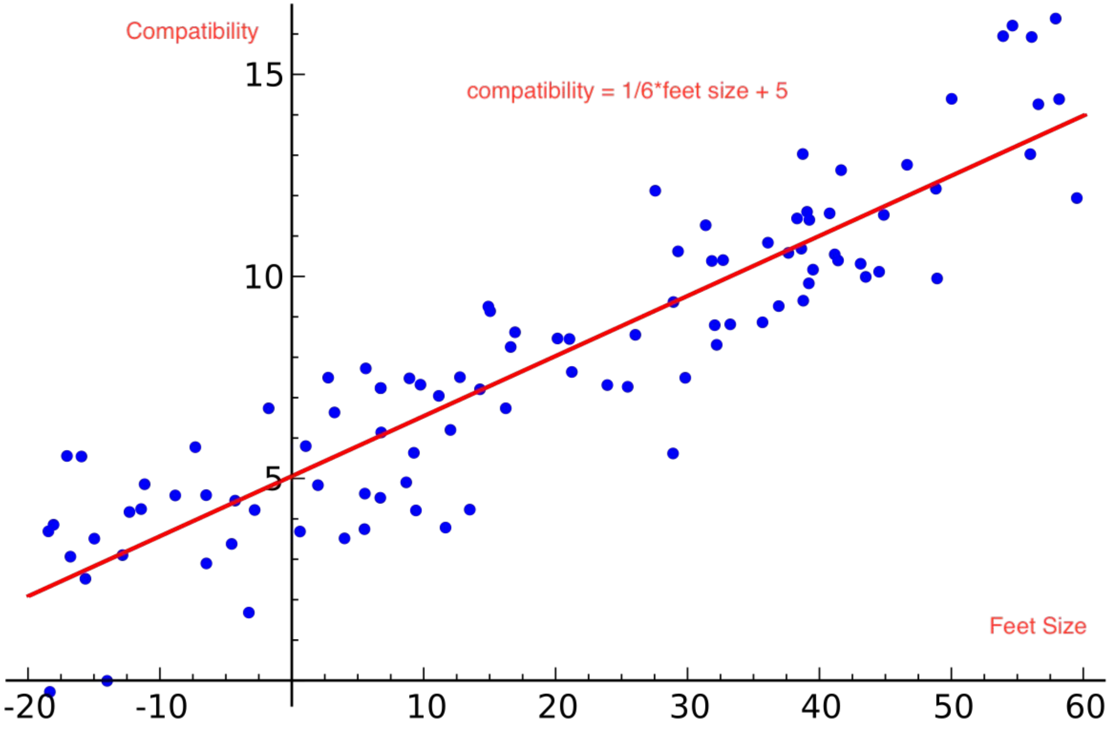
- Reinforcement Learning in Machine Learning
    - Model that learns by interacting with environment
    - Perform action then observe and store result
    - When do you choose to perform random action vs. perform best action according to what you’ve learned so far?
    - Explore w probability 1/e to maximize information gain!



Compatibility Models

- What makes someone compatible?
- Assumptions:
  - You intuitively know how much you like someone
  - You already have dated some people => training data
- Let's use linear regression!

Linear Regression



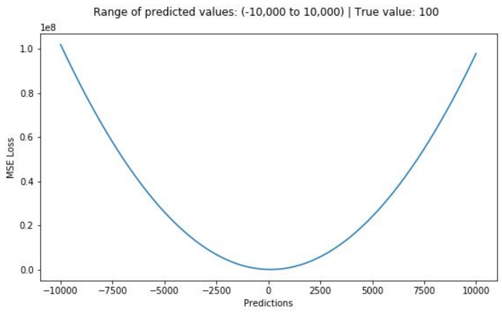
Training the Model

- Each sample in the data we have for training is a person: a bunch of attributes and a measure of how much you liked the person (compatibility) 👅
- We have a weight that corresponds to each attribute (i.e. multiply the attribute by the weight when calculating compatibility)
- For each sample, use our current weights to calculate a prediction for the compatibility and see how different it is from the actual value (how much you actually liked the person)
- Then use that difference to update the weights

Stochastic Gradient Descent

- Mean squared error:

$$MSE = \frac{\sum_{i=1}^{n} (y_i - y_i^p)^2}{n}$$



- Our goal is the bottom of the bowl, the minimum error
- To get there, we can calculate the derivative (the slope) and move in that direction
- Update our weights:
  - Derivative = 2 * (prediction - label)
  - Weights = weights - learning_rate*derivative*example attribute

Let’s Try an Example

- The current weight for foot size is 2, bias of 1
- You date someone who had a footsize of 1 ft and scored them at 4 for how much you liked them
- (2*1 ft) + 1 = 3 is compatibility our model would predict
- SGD:
    - Delta = 2*(prediction - label) = 2*(3 - 4) = -2
    - Ft sz weight = ft sz weight - learning_rate*delta*example attribute = 2 - 0.25*-2*1 = 2.5
- We’ve moved closer to the optimal weight of 3! 🔥💯

Pseudocode

```
Def train(data):
    initialize weights to the number of attributes
    for sample in data:
        calculate derivative
        weights = weights - learning_rate*derivative*sample_attributes

Def compatibility(person):
    return person*weights + bias
```

Issues 🤮

- Requires a lot of data that you might not have in real life (ground truth scarcity)
- Can choose wrong attributes to represent people
- There are likely better models to use (if you're interested, look into unsupervised learning or reinforcement learning methods!)


Conclusion

CS (and math) are cool

Other CS classes you can take in the fall:

- CS15: Intro to OOP and CS
- CS17: CS An Integrated Intro
- CS19: An Accelerated Intro
- CS13: UI/UX
- CS10: Data Fluency for All
- CS141: Artificial Intelligence (email Prof. Konidaris for an override)