Outline

- Review Genetic Algorithms
- Why do they work?
  - Introduce the schema theorem
  - Experiments to support theorem
- Putting things together
- Evaluating ML

Genetic Algorithms

- Operate on a population of candidate solutions
- Candidate solutions originally represented as bits
- Key operators are:
  - Selection
  - Mutation
  - Crossover
- Today we will focus on the theory of why GAs work

How do GAs work?

Theory of GAs was first formulated by John Holland in the mid 1970's

Hypothesis: *GAs work by discovering, emphasizing, and recombining good building blocks within individuals in a highly parallel fashion*

First we need to develop some terminology for talking about building blocks

Building block

- A combination of bits that confer higher fitness on individuals when they are present
- Suppose we use a fitness function that interprets the list of bits as a binary number
- For the binary fitness function, having two 1's in the leftmost portion would be a good building block

2^4	2^3	2^2	2^1	2^0	Integer
1	1	0	0	0	24
0	0	1	1	0	6

Schema

- Formalize the notion of a building block
- Template made up of 1's, 0's and *'s which represent wild cards
- `11****` is the schema that represents all 6-bit individuals that begin with two 1's
- `1****0` is the schema that represents all 6-bit individuals that begin with a 1 and end with a 0

Instances

- Individuals that fit a given schema are said to be instances of it
- For example, `110001` and `111000` are both instances of the schema `11****`
- `111000` is also an instance of the schema `1****0`

Order of schema

- Represented as o(S)
- The number of defined bits (non-wildcards)
- o(****1*) = 1
- o(1****0) = 2
- o(1**0*1) = 3

Defining length of schema

- Represented as d(S)
- The distance between the schema's outermost defined bits
- `d(11****) = 1`
- `d(1****0) = 5`

Number of schema one individual can be an instance of

- The individual `11` is an instance of the schema:
  - `**`
  - `*1`
  - `1*`
  - `11`
- What are all of the schema for which 101 is an instance?

Number of schema an individual can be an instance of

- There are two possible values for each location in the individual, the wild card * or the actual bit value
- Assume that the length of an individual is ℓ
- Then any individual is an instance of 2^{ℓ} schema

How a GA processes schema

- Any bit string of length ℓ is an instance of 2^{ℓ} schema
- Therefore, any given population of n strings contains instances of between 2^{ℓ} and $n*2^{\ell}$ different schemas
- GA explicitly evaluates the fitness of n strings in any given generation, but it also implicitly estimates the average fitness of a much larger number of schema

Example of how a GA processes schema

- Consider a random population of size `n`
- On average, half of the strings will be instances of the schema `1*****` and half will be instances of the schema `0*****`
- The evaluation of the approximately `n/2` strings that begin with a 1 gives an estimate of the average fitness of the schema `1*****`
- Schema are NOT explicitly represented or evaluated by GAs, yet the behavior of GAs can be described as if they are actually calculating the average fitness of schema

Recall how a generation of a GA works

```
repeat until n offspring have been created
    select two members of population based on fitness
    with probability pCross, perform crossover
    with probability pMutate, mutate eat bit of the offspring
    Add two children to the next population
```

How selection works

How crossover works

How mutation works

Schema theorem

- Describes the change in number of instances of a schema from one generation to the next
- Provides a lower bound
- Depends on 3 factors, which we will look at in turn

expected number of instances of schema S >=

[1] (expected selection of S) *

[2] probability(S survives crossover) *

[3] probability(S survives mutation)

[1] Expected selection of S

- Using fitness proportionate selection, the probability of being selected is higher for schema with above average fitness
- u(S,t) is the average fitness of instances of schema S in population at time t
- avgFit(t) is the average fitness of entire population at time t
- m(S,t) number of instances of S at time t

u(S,t) / avgFit(t) * m(S,t)

u(S,t) / avgFit(t) * m(S,t)

- Let's consider three schema that are equally represented in the current population, one with below average fitness, one with above average, and one with significantly above average
- Notice that the expected number of selections mirrors their fitness (higher fitness => more selections)

u(S,t)	avgFit(t)	m(S,t)	Expected selections
10 [below avg]	15	9	6
20 [above avg]	15	9	12
50 [way above avg]	15	9	30

[2] Probability that a schema survives crossover

- Probability is higher for schema with shorter defining lengths
- d(S) is the defining length of schema S
- len-1 is the maximum defining length for schema of length len

1 - (probCross * d(S) / (len-1))

[3] Probability that a schema survives mutation

- Probability is higher for lower order schema
- Recall that the probability of doing mutation is computed at each bit
- o(S) is the order of schema S

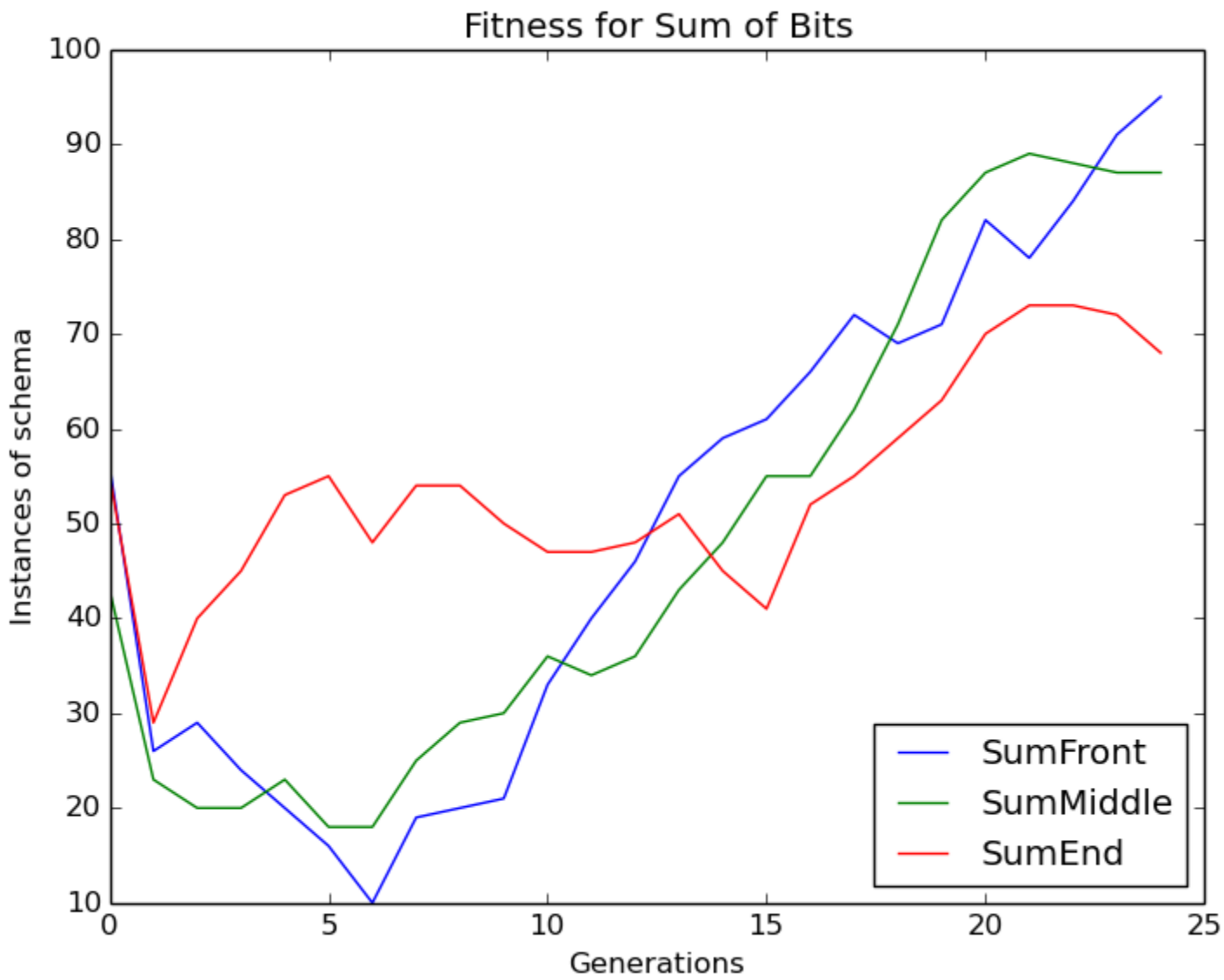
$$(1 - probMutate)^{o(s)}$$

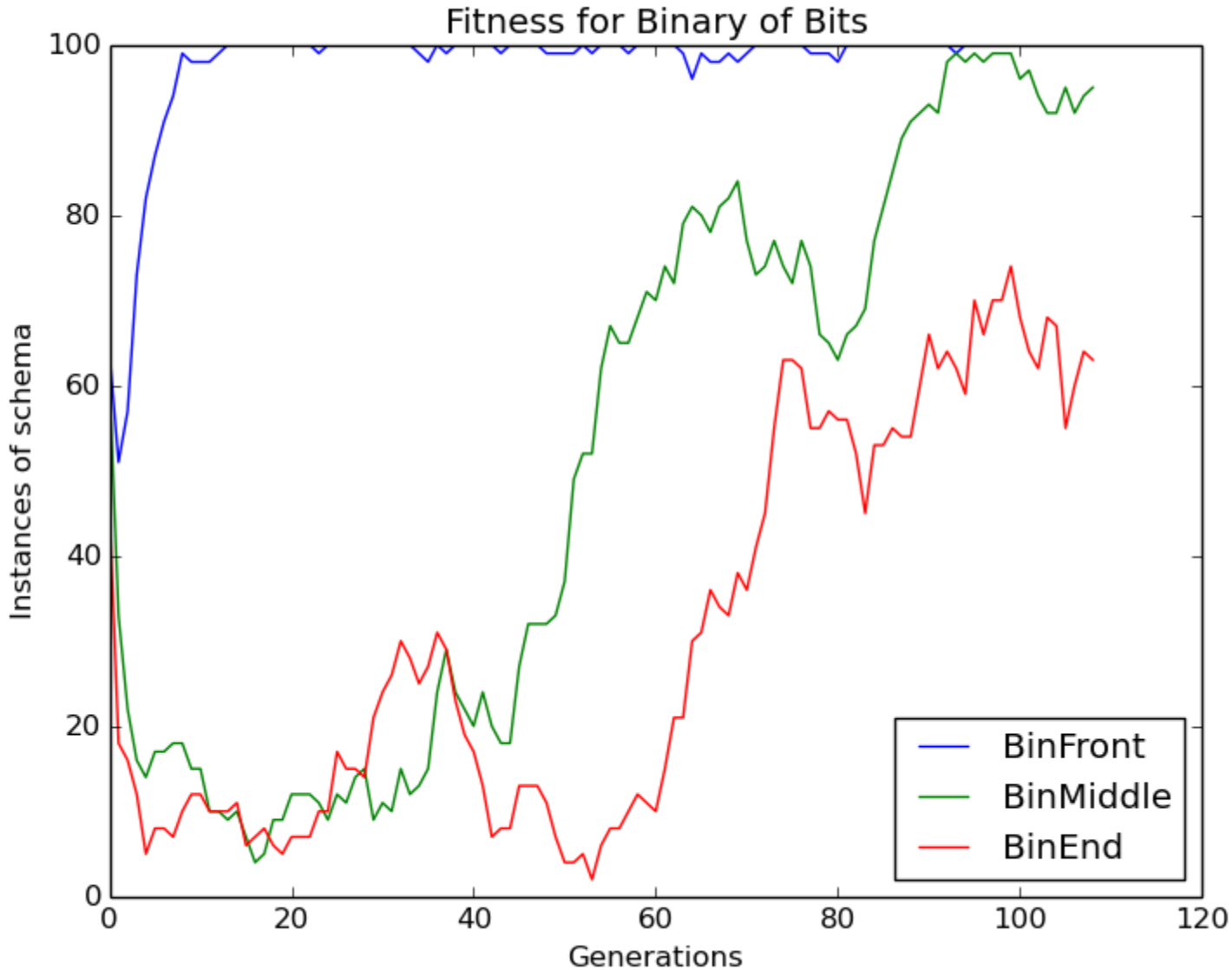
Implications of the Schema Theorem

- Short, low-order schema whose average fitness remains above the mean will receive exponentially increasing numbers of samples over time
- So, if you know something about your problem, when defining candidate solutions, try to create good building blocks by putting key features close to together on the string

Demonstration of Schema Theorem

- Compare two fitness functions
  - Sum of bits
  - Binary of bits
- Look at the number of instances for several schema
  - `11**************************` Front
  - `**************11**************` Middle
  - `**************************11` End
- Based on Schema Theorem, how should we expect the number of instances of the schema Front, Middle and End will change over time for these two fitness functions?


Fitness for Sum of Bits
Instances of schema
Generations
100
90
80
70
60
50
40
30
20
10
0
5
10
15
20
25
SumFront
SumMiddle
SumEnd


Fitness for Binary of Bits
Instances of schema
Generations
0
20
40
60
80
100
120
BinFront
BinMiddle
BinEnd

Building block hypothesis

- Crossover is believed to be a key source of a GA's success by combining instances of good schema to form instances of even better schema
- This constructive aspect is not captured in the Schema Theorem
- Schema Theorem only accounts for the destructive effects of crossover and mutation

Easy problems for a GA

- Match the building block hypothesis
- We would expect them to be easy for a GA to solve

Schema	Fitness
***0**0*	low
***0**1*	mid
***1**0*	mid
***1**1*	high

Deceptive problems

- Violate the building block hypothesis
- We would expect them to be hard for a GA to solve
- This is similar to the challenge of XOR for networks

Schema	Fitness
***0**0*	mid
***0**1*	low
***1**0*	low
***1**1*	high

Summarizing this module

What we've learned about

Ways to define agent behavior in environments that may be:

- Continuous
- Dynamic
- Partially-observed
- Stochastic

***State**:* now a set of measurements that define what we know about the current environment. Need to map state to **action**

Hypothesis: function that maps state to action

Examples?

Ways to ***learn*** a hypothesis defined by a set of **parameters** (i.e. **weights**)

- **Supervised (or imitation learning):** given the "correct" action for each state it encounters
  - **Supervised:** Given action is optimal
  - **Imitation:** Given action is what a human did
- **Reinforcement learning:** Need to *infer* correct actions from environmental rewards (that may not be immediate!)

Both approaches need a way to find the best parameters, typically iteratively

- ***Local search*** over a **continuous** domain
  - Stochastic gradient descent is a form of hill climbing to minimize a **loss**
  - Genetic algorithms are similar to stochastic beam search to maximize a fitness

Where does backprop fit in to this idea?

What we've learned about

Ways to define agent behavior in environments that may be:

- Continuous
- Dynamic
- Partially-observed
- Stochastic

***State**:* now a set of measurements that define what we know about the current environment. Need to map state to **action**

Hypothesis: function that maps state to action

Examples?

- *Percepton*
- *Neural network*
- *Convolutional neural network*

Ways to ***learn*** a hypothesis defined by a set of **parameters** (i.e. **weights**)

- **Supervised (or imitation learning):** given the "correct" action for each state it encounters
  - **Supervised:** Given action is optimal
  - **Imitation:** Given action is what a human did
- **Reinforcement learning:** Need to *infer* correct actions from environmental rewards (that may not be immediate!)

Both approaches need a way to find the best parameters, typically iteratively

- ***Local search*** over a **continuous** domain
  - Stochastic gradient descent is a form of hill climbing to minimize a **loss**
  - Genetic algorithms are similar to stochastic beam search to maximize a fitness

Where does backprop fit in to this idea?

Problem domain: Self driving cars

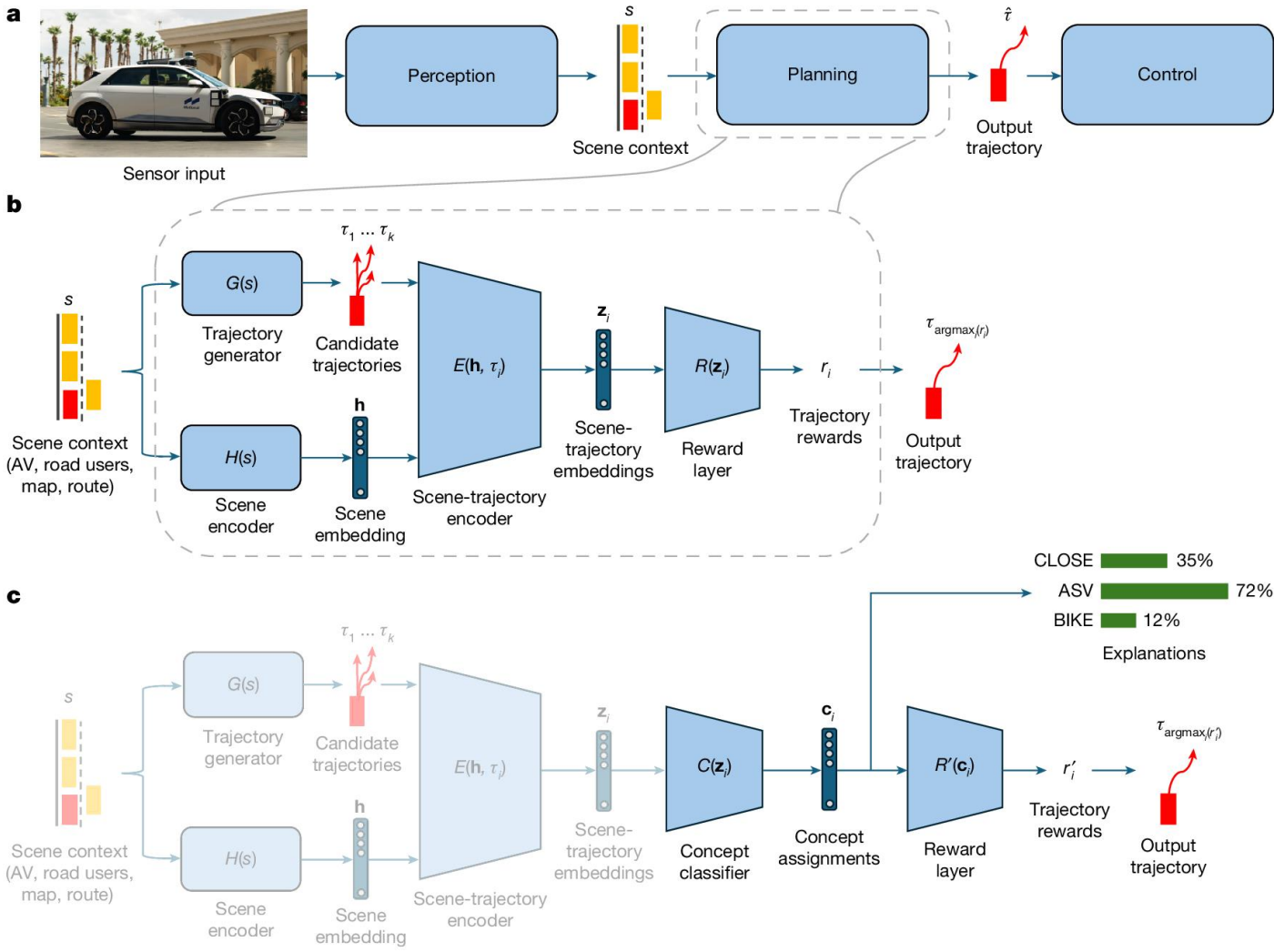
I want a car that can drive from Swarthmore to Center City

How *might* I begin to build a system to do this?

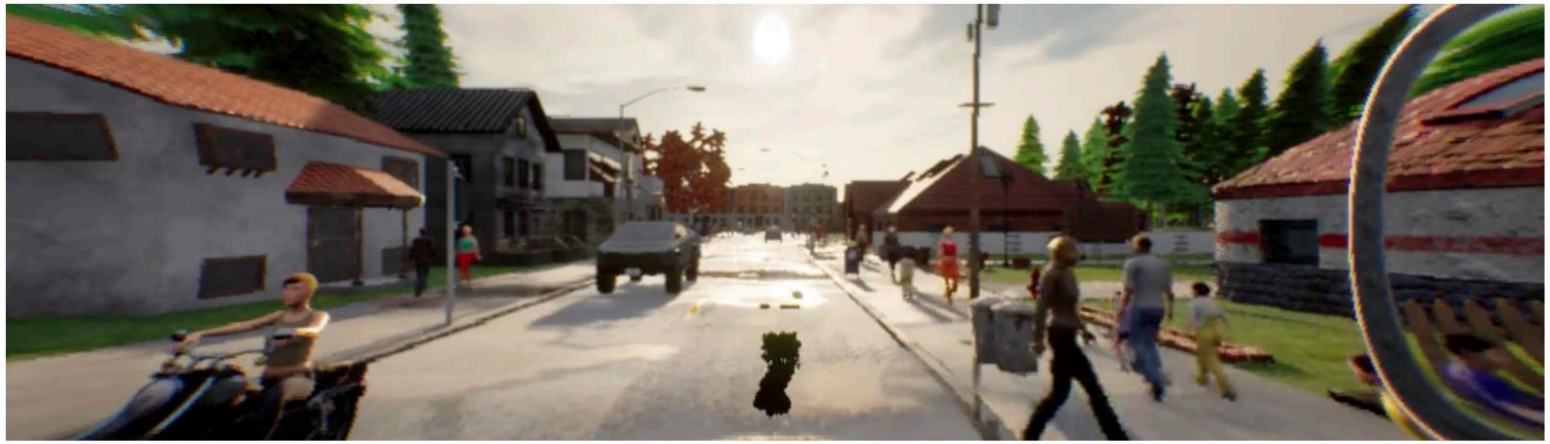
Questions to consider:

- How can a break this into subproblems that I can solve with an appropriate level of abstraction for each?
- What approach might I apply to each subproblem?
- How might these work together?
- What made learning to play an Atari game difficult?

Example system



Simulation



(a) Reflections from after-rain puddles in fornt of the ego-vehicle are incorrectly rendered as black pixels.



(b) Reflections are correctly rendered if the puddle is not directly in front of the ego-vehicle.

Figure 6: **Rendering issue of CARLA 0.9.11 running on Ubuntu with OpenGL.**

Assessing ML systems



XKCD- Machine Learning

The promise of AI vs the reality of deploying AI

“AI is the new electricity. Just as electricity transformed almost everything 100 years ago, today I actually have a hard time thinking of an industry that I don’t think AI will transform in the next several years.”

Andrew Ng

“[S]hould we be more cautious given AI’s susceptibility to bias, its potential errors in edge cases, its lack of transparency in decision making? … What should we require of an AI system in order to trust it enough to let it work autonomously?”

Melanie Mitchell

Claims about deep learning systems:

- Learn in a way similar to humans
- Learn how to perform tasks on their own
- Enable computers to teach themselves

Michell questions these claims

Mitchell raises the following points

The only assurance that we have about learning success is that the networks are very accurate on the test/validation sets

- Deep learning systems cannot explain their decisions
- If there are statistical correlations present in the data, even if they are irrelevant to the task, networks may learn these rather than the intended mapping
- There may be bias built in to the training set
- Often fail on long-tail events
- These networks may be vulnerable to adversarial attacks

ML systems cannot explain their decisions

ML systems may make decisions based on irrelevant statistical correlations present in data

ML systems mimic bias present in training data

ML systems fail to respond correctly to long-tail events

ML systems are vulnerable to adversarial attacks

Assessing convolutional networks

- Something very different from human perception is going on with convNets
- Brains evolved to control bodies
- In animals, perception is central to survival

One possible conclusion:

A system cannot learn to perceive in isolation, it depends on embodiment

How computers currently learn

- Largely trained via supervised learning
- Often have a fixed set of possible output categories
- Typically require labeled data
- A huge amount of effort is needed to collect and prepare this data
- A human must construct an appropriate deep network architecture
- A human must tune the hyperparameters of the deep learning system
- Many epochs of training are needed to successfully learn

How similar is this to how humans learn?

How children learn

- More open-ended learners
- Don’t require labeled data
- Explore on their own

Video of a toddler at play

Supervised learning is not a viable path to general-purpose AI

Yoshua Bengio, professor at University of Montreal, won Turing Award for his work on deep learning: “we can’t realistically label everything in the world”

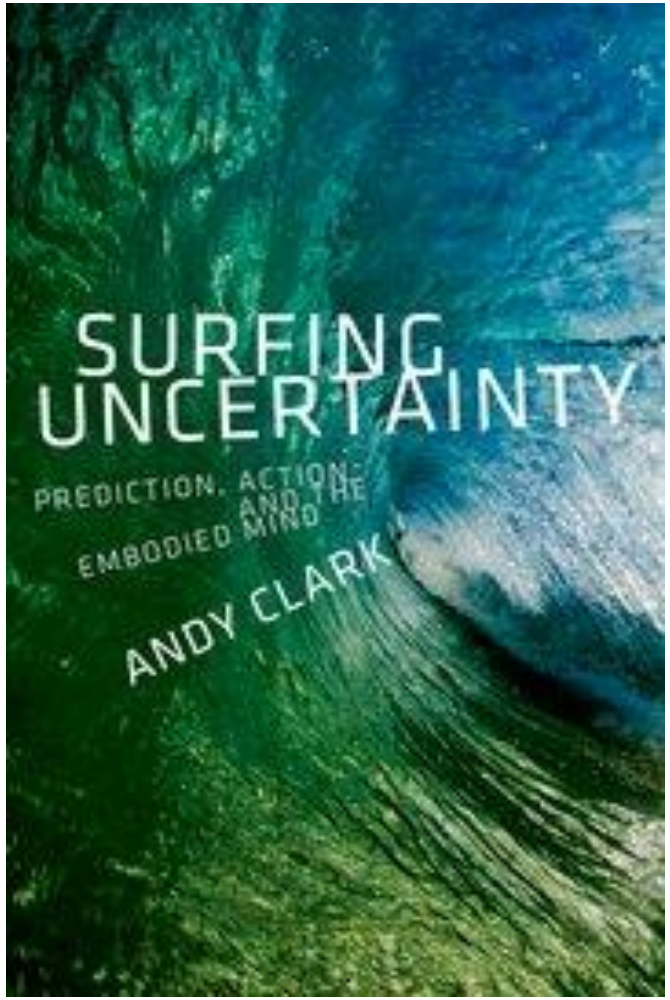
Andrew Ng, professor at Stanford University: “requiring so much data is a major limitation of deep learning”

To achieve general AI, almost all learning will have to be unsupervised, but how will this work?

Prediction may be the key

Andy Clark (Philosopher of Mind) argues in his book *Surfing Uncertainty: Prediction, Action, and the Embodied Mind* that

“To deal rapidly and fluently with an uncertain and noisy world, brains like ours have become masters of prediction.”



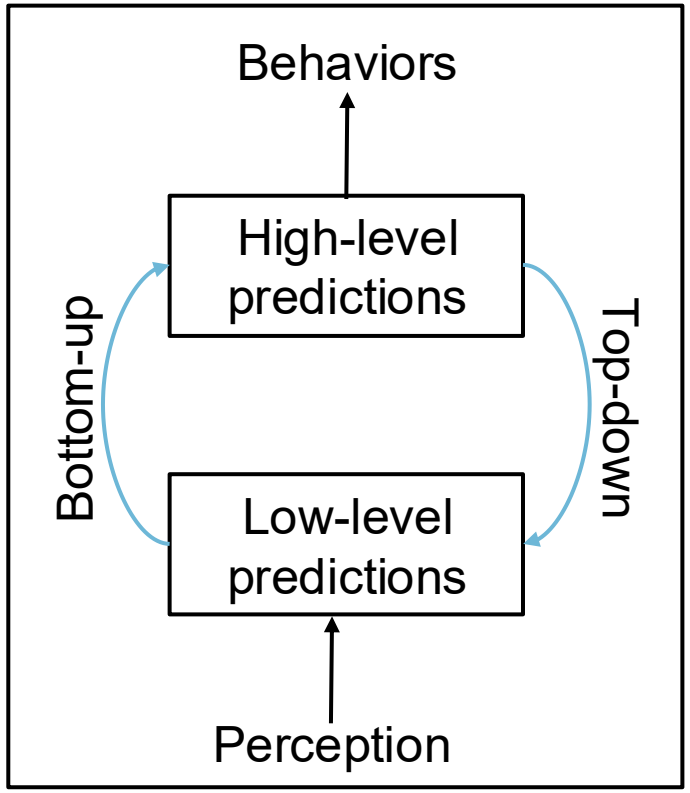
Learning about the world through prediction

- Choose an action to explore
- Predict the perceptual outcome
- Execute the action
- Observe the actual outcome
- Use the error between the predicted and actual outcome to improve your subsequent predictions

This is the essence of ***self-supervised learning***--no external teacher required--and every single moment is another opportunity to learn

Hierarchical predictions

- The world is highly structured and displays regularity at many different time scales
- The brain is likely learning about the world at many time scales simultaneously
- Low-level predictions operate in a bottom-up way to generate high-level predictions
- Once high-level predictions start to form, they can inform and shape low-level predictions through top-down expectations



Using prediction to act

- Intelligent systems don’t just passively process sensory stimulus
- Instead they proactively set expectations for what is likely to be experienced next
- Thus they are always poised to act based on their past experiences

One perspective on intelligence

- The brain is an embodied prediction machine
- It gains knowledge through self-supervised learning
- Acting and perceiving are tightly intertwined
  - We act to gain more predictive knowledge (bottom-up)
  - We predict to better deal with the noise and uncertainty of perception (top-down)