Machine Learning, Offense, and the future of Automation

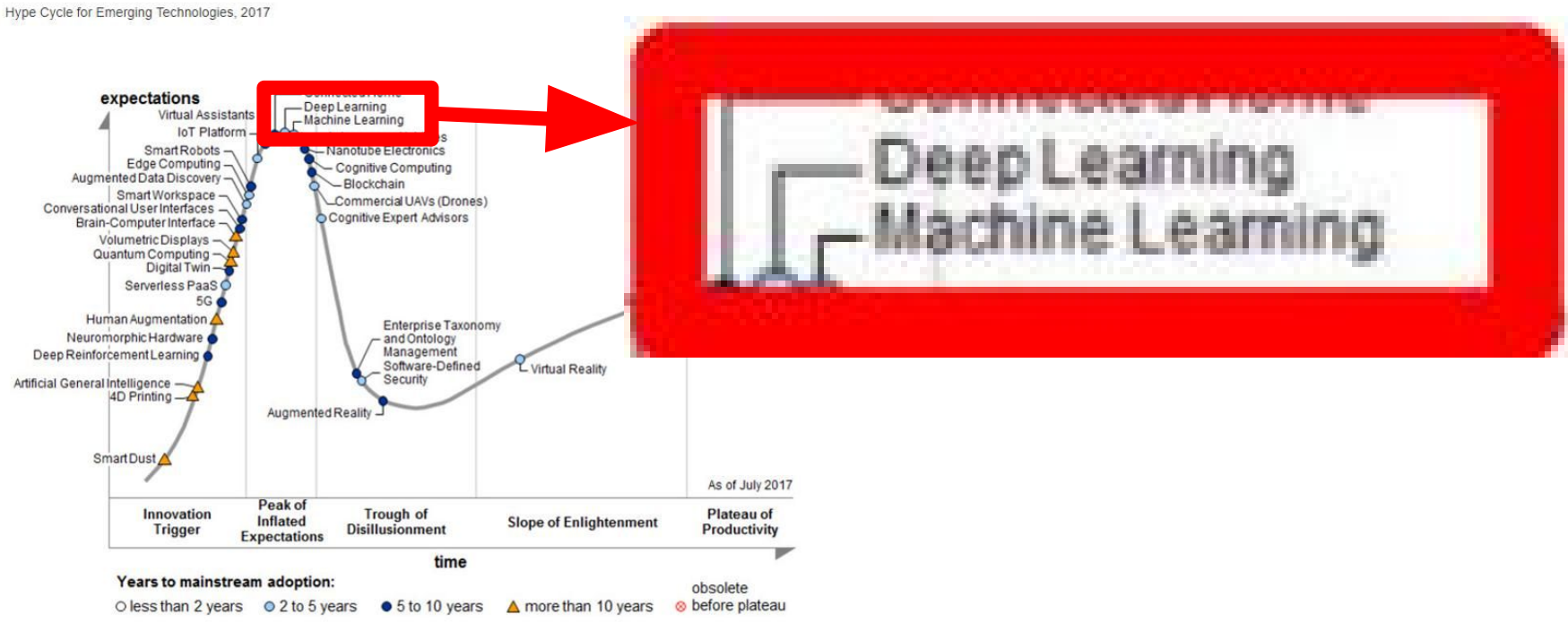
Halvar Flake / Thomas Dullien
ZeroNights 2017 Keynote
Google Project Zero

Google

Warning: This talk is opinions, not facts

- This talk is a keynote
- It reflects what I *think*
- I am wrong about things with quite some regularity

2017: Peak of ML (or “AI”) hype?



Note: PaaS = platform as a service; UAVs = unmanned aerial vehicles

Source: Gartner (July 2017)

From ML scepticism to (careful) believer

- I was a strong ML sceptic for most of the 2000s
  - People threw arbitrary classifiers at high-dimensional noise
  - Too little qualitative understanding of results
- Started digging below the surface in the early 2010's
- Realization: Below all the hype, there is good solid CS, and good maths

- "Converted": I always believed in good math / CS, so I also believe in it when it is called "Machine Learning" or "AI"

Key points of this talk (1/3)

- Not all of the AI hype is bullshit

- Real progress is being made:
  - Algorithmic breakthroughs, orders of magnitude speedups
  - Heterogeneous computing changes the boundary of the possible

Key points of this talk (2/3)

- AI and ML is less magic than you think — mostly college-level applied math + lots of computational power

Key points of this talk (3)

- Biggest potential is for problems with “stable-in-time distributions”
- Most offensive problems fall into this category
- Many defensive problems do, too — but seem not to be focus of ML efforts?

Google

Topics we will cover

1) What are some key algorithmic drivers behind the progress in ML / AI ?

2) Most defensive applications of ML in security are “wrong” — and the most promising area may be “offensive” applications.

3) Where is AI / ML already used offensively? What are promising areas where I expect progress?

Topics we will cover

1) What are some key algorithmic drivers behind the progress in ML / AI ?

2) Most defensive applications of ML in security are “wrong”* — and the most promising area may be “offensive” applications.

3) Where is AI / ML already used offensively? What are promising areas where I expect progress?

* ‘wrong’ does not mean ‘do not do it’, it means ‘not necessarily well-suited for ML’

What are the algorithmic drivers behind the ML hype?

Algorithmic drivers

1) Automated Differentiation (1976, and various times thereafter)
2) Locality-Sensitive Hashing (1998, 2002, 2003)
3) Monte Carlo Tree Search (2006)

Engineering drivers

1) ConvNets / Inception NNs (finding more effective NN structures)
2) Larger data sets to train from (use-once training minimizes empirical error)
3) Heterogeneous computation (GPU, FPGA, ASICS / TPU)
4) ReLU, Leaky ReLU and other tricks

1) Automated Differentiation

- Originally invented by Seppo Linnainmaa in 1970, published in 1976
- Also known as Baur-Strassen method
- Reinvented many times in many contexts
- Also known as “Backpropagation” in neural network circles

- Key Result: **“Cheap Gradient Principle”**

Cheap Gradient Principle

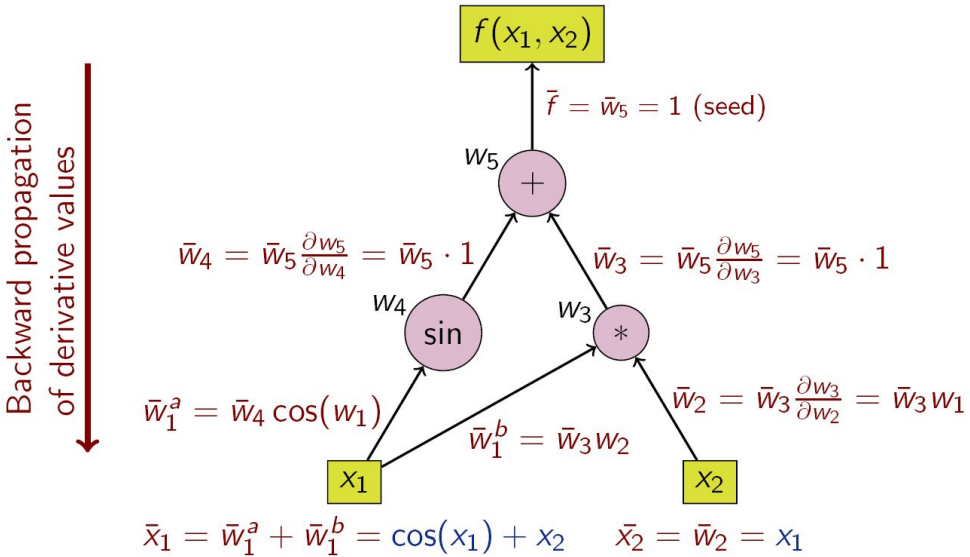
- People think that calculating the gradient of a function $f : \mathbb{R}^n \to \mathbb{R}$ costs roughly $n \times cost(eval(f))$

- Automatic differentiation gives the gradient of $f : \mathbb{R}^n \to \mathbb{R}$ at $O(cost(eval(f))) = c \times cost(eval(f))$ with $c < 30$

- Gradient computations for n = millions is easily doable

Basic Idea of Automatic Differentiation

- Consider the function: $f(x_1, x_2) = \sin(x_1) + x_1 x_2$
- Decompose into intrinsic operations and draw a “computational graph”
- Calculate forward, sweep backward through the graph



Automated Differentiation: FOSS implementations

- Tensorflow, Torch-Autograd etc. — “differentiate a computational graph”
- FADBAD++ — “differentiate your C++ function”, SPII — “minimize your C++ loss function or lambda”
- Autograd — “efficiently differentiate a numpy function”

2) Locality-Sensitive Hashing

- Invented at Altavista in ’98 (Broder), at Google ’99 (Charikar), formalized and extended to other metrics at MIT ’03 (Indyk)

- 2012 ACM Theory & Practice Award

- **“Cheap approximate nearest neighbors in high dimensions”**

Google

Cheap Approximate Nearest Neighbors

- Old solutions to solve NN-problems cost $O(n)$ where n is the index size

- LSH provides approximate NN with $O(\log n)$

- Approximate nearest-neighbor search for high dimensions (thousands) and big-data index sizes (billions) is feasible

Basic Idea of Locality Sensitive Hashing

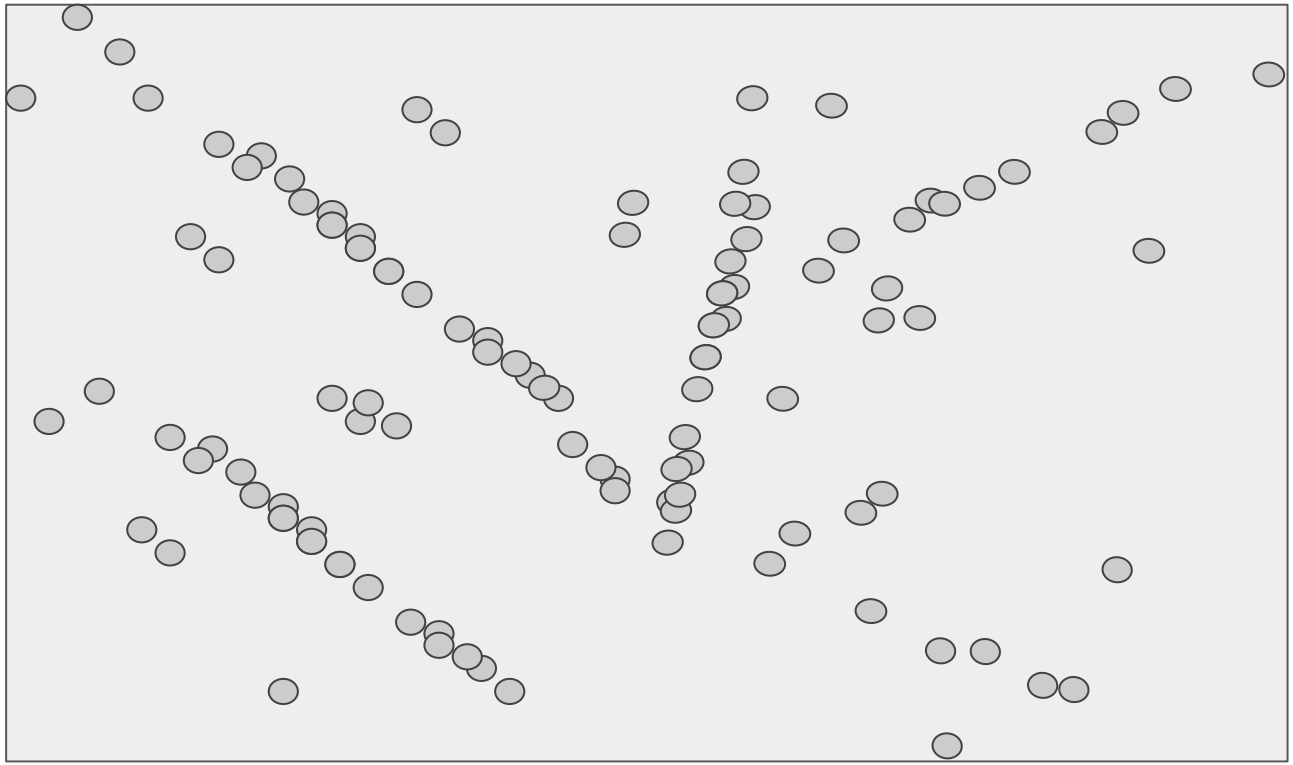
Obtain family of hash functions $\mathcal{H}$ that satisfy (for some distance R, and probability P):

$$R, (1+\epsilon)R, P_1, P_1 - \epsilon$$

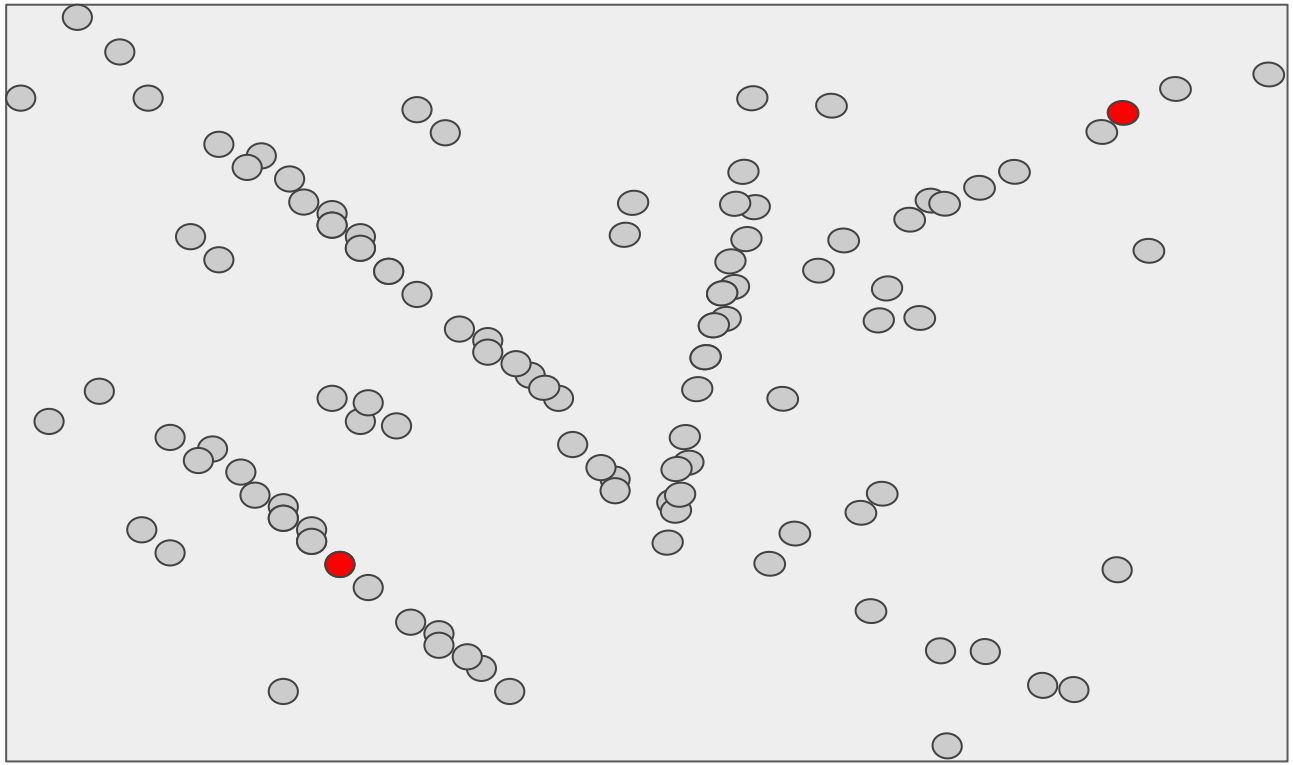
$$\begin{aligned} ||x - x'|| &< R \Rightarrow P[h(x) = h(x')] > P_1 \\ ||x - x'|| &> R \Rightarrow P[h(x) = h(x')] < P_1 - \epsilon \end{aligned}$$

“Similar elements have a higher probability to be hashed to the same value than dissimilar items”

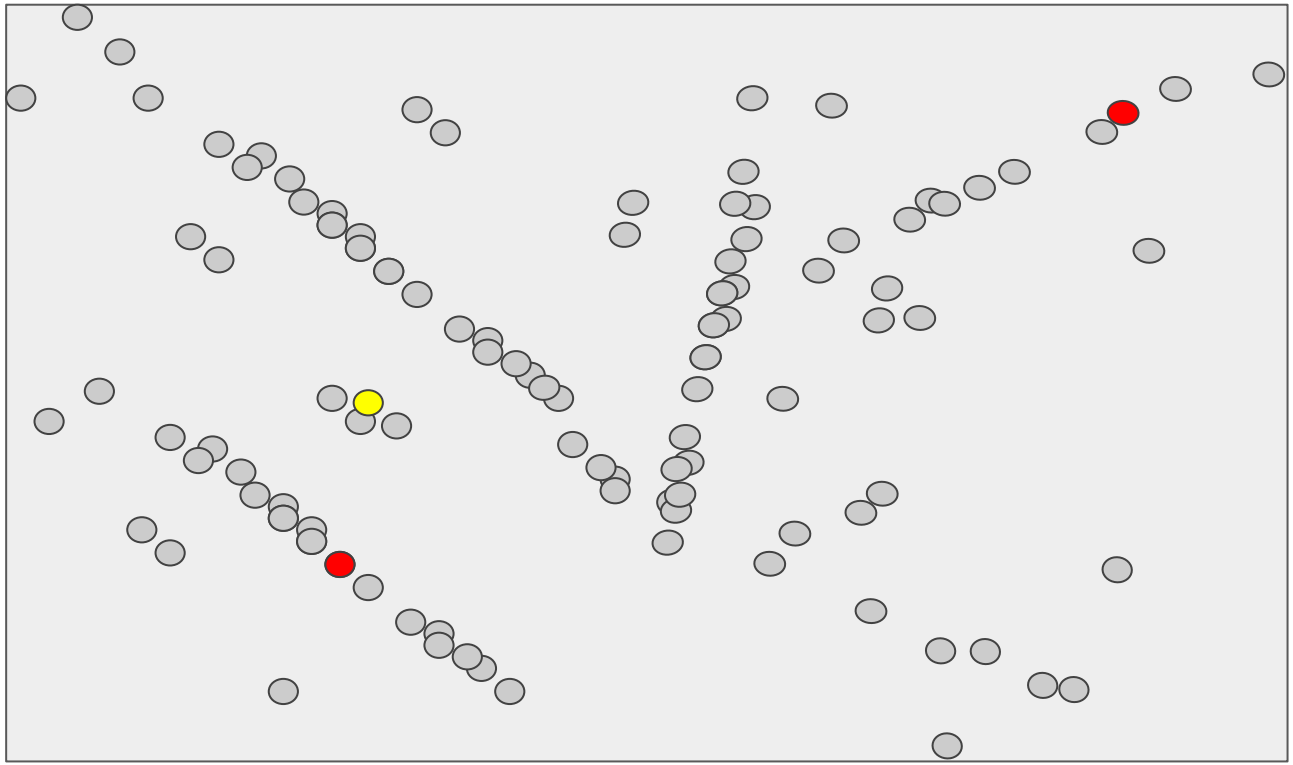
FOSS example: ANNoy



Pick two random points

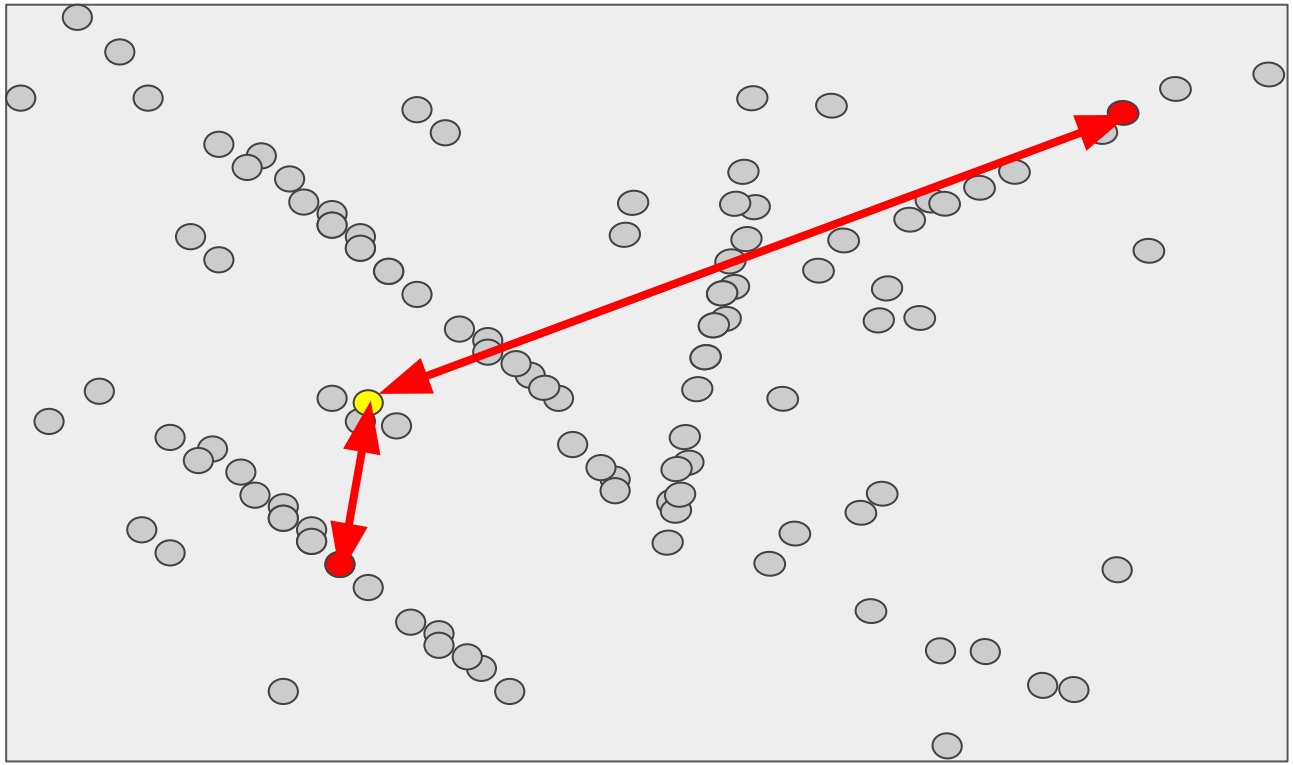


Sample next point

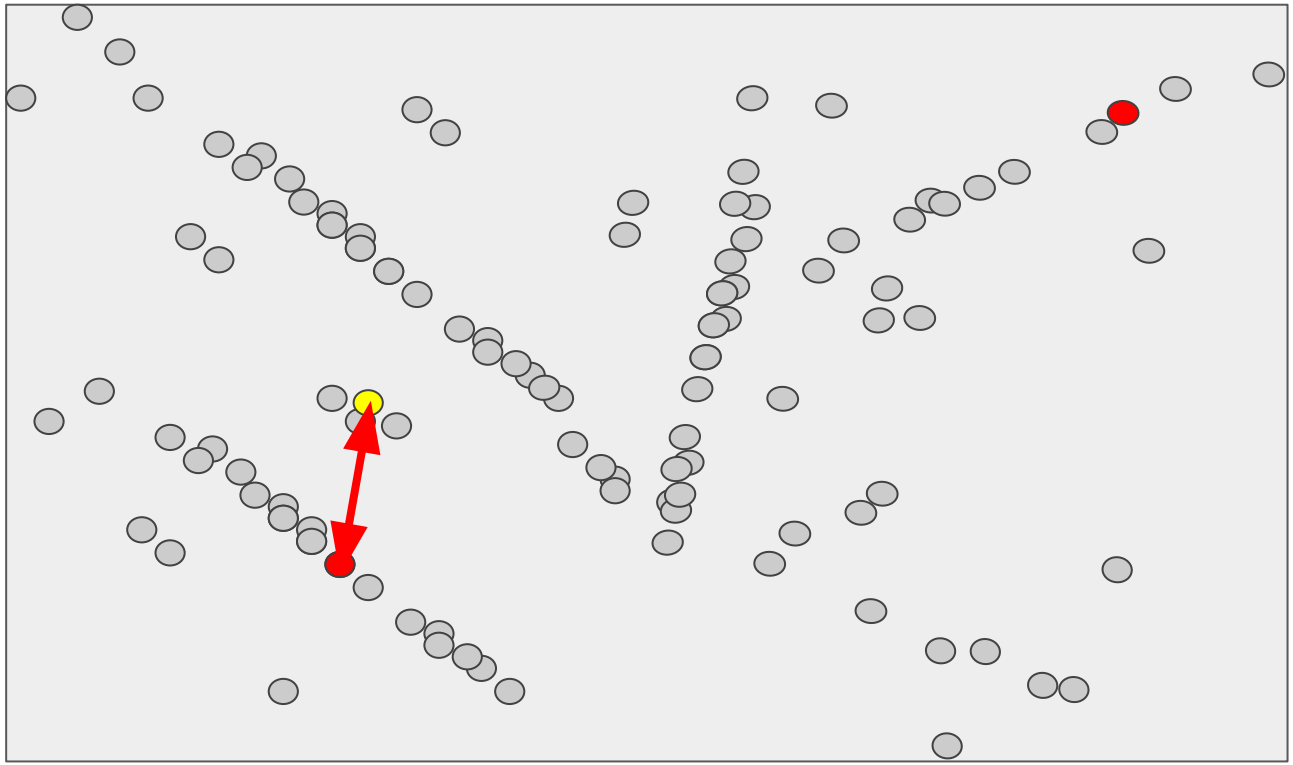


Google

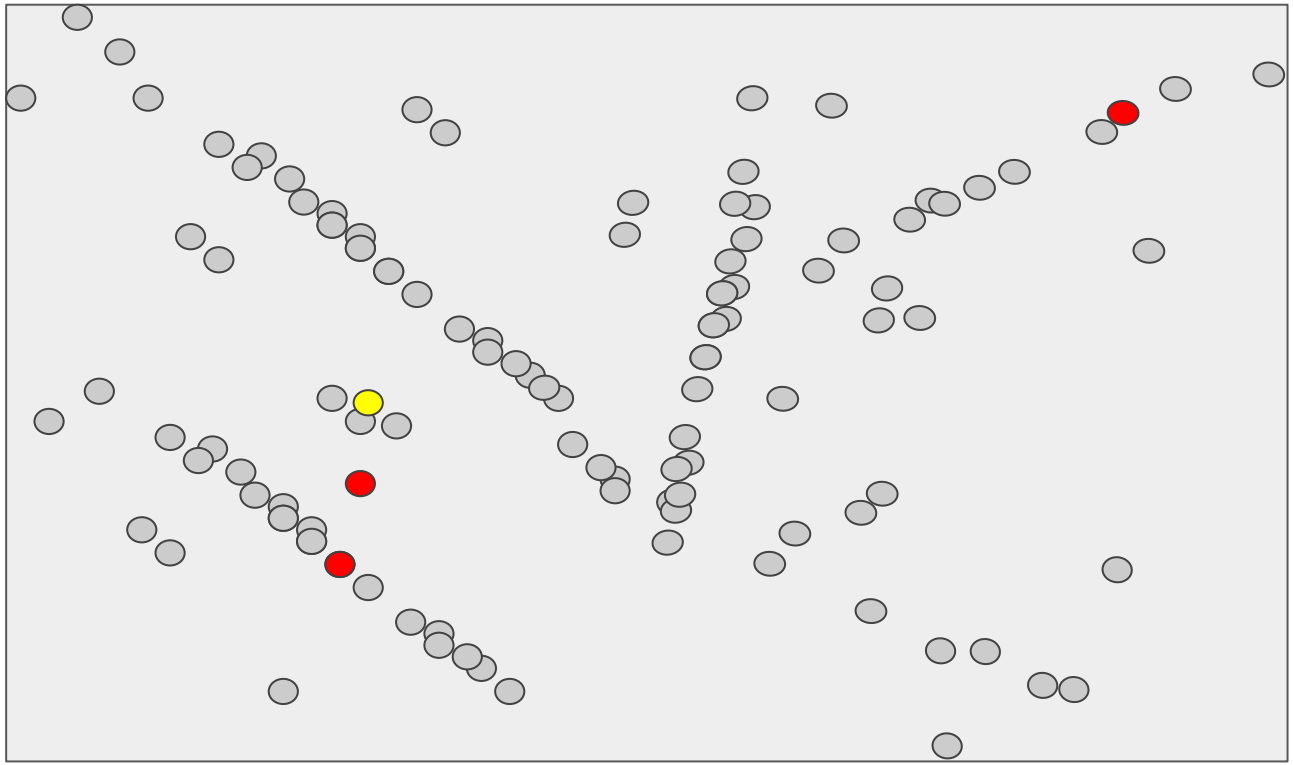
Is it closer to point A or point B ?



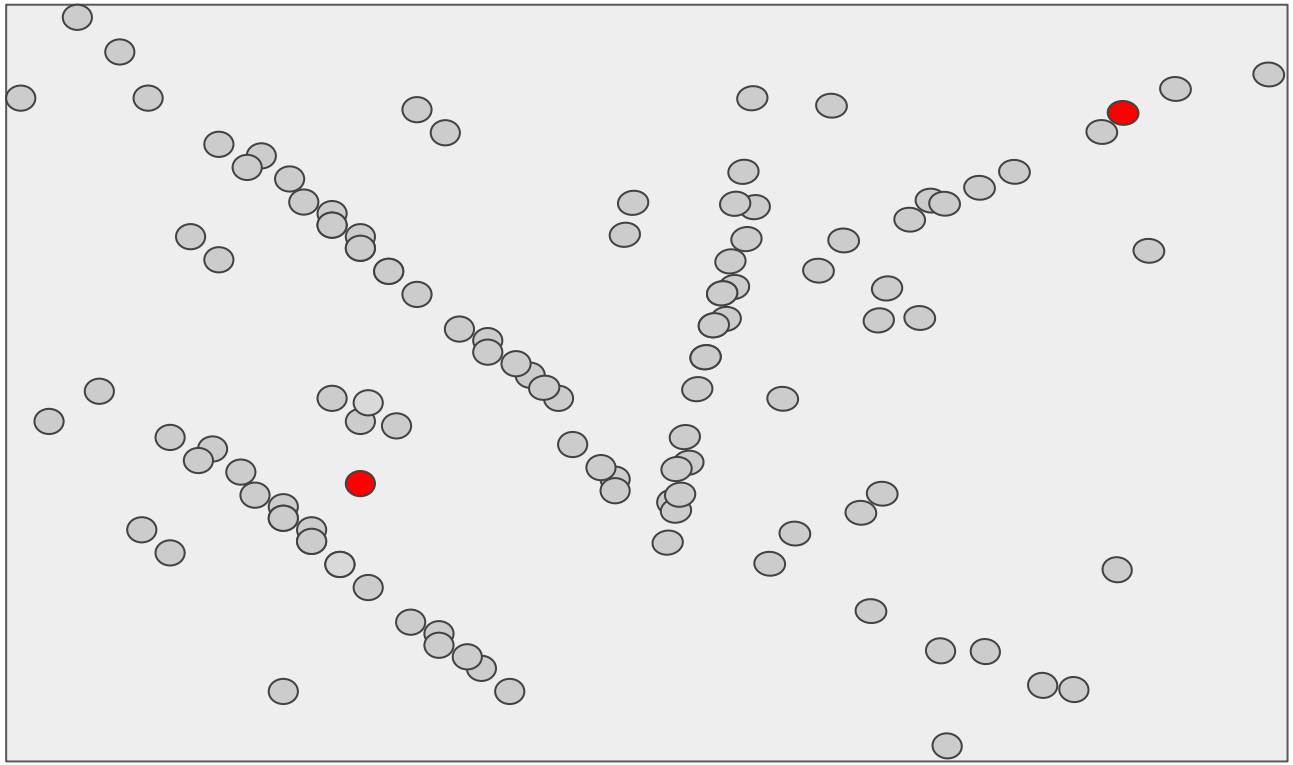
It is closer to point A



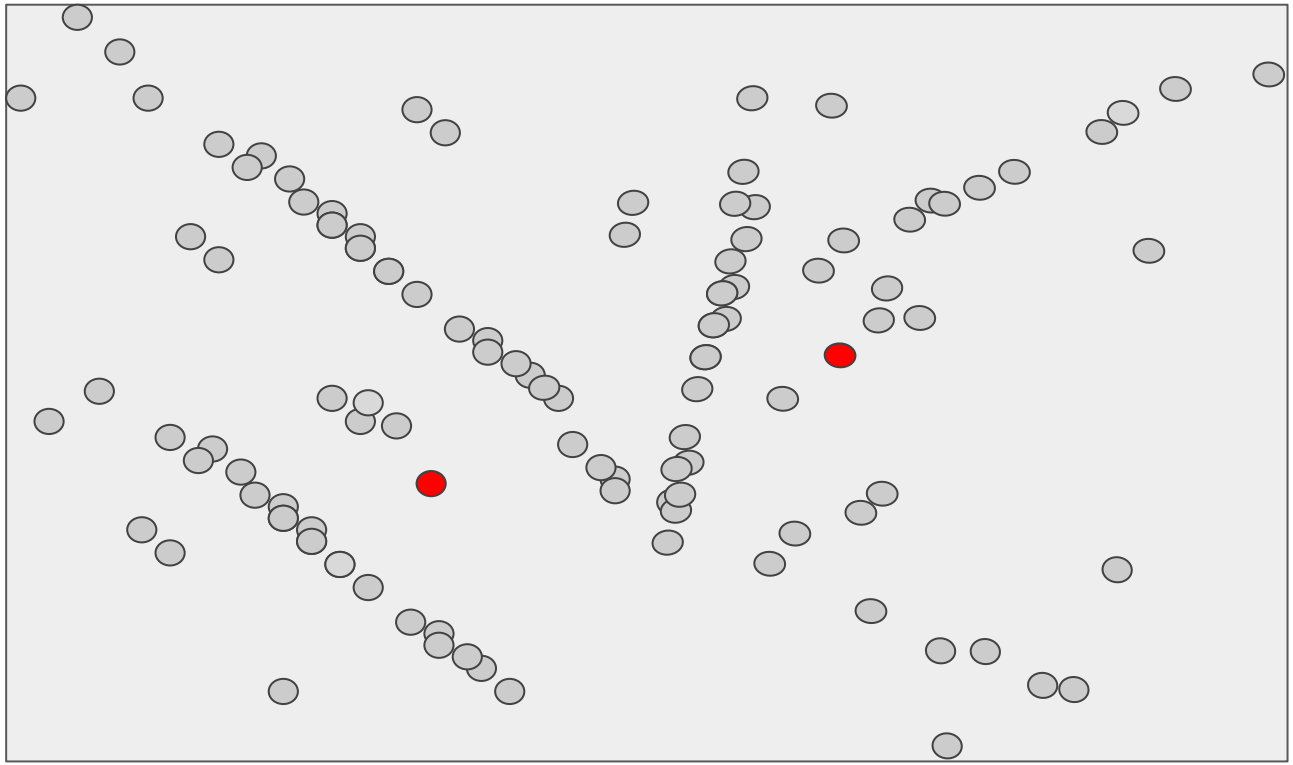
Create synthetic A' point by averaging



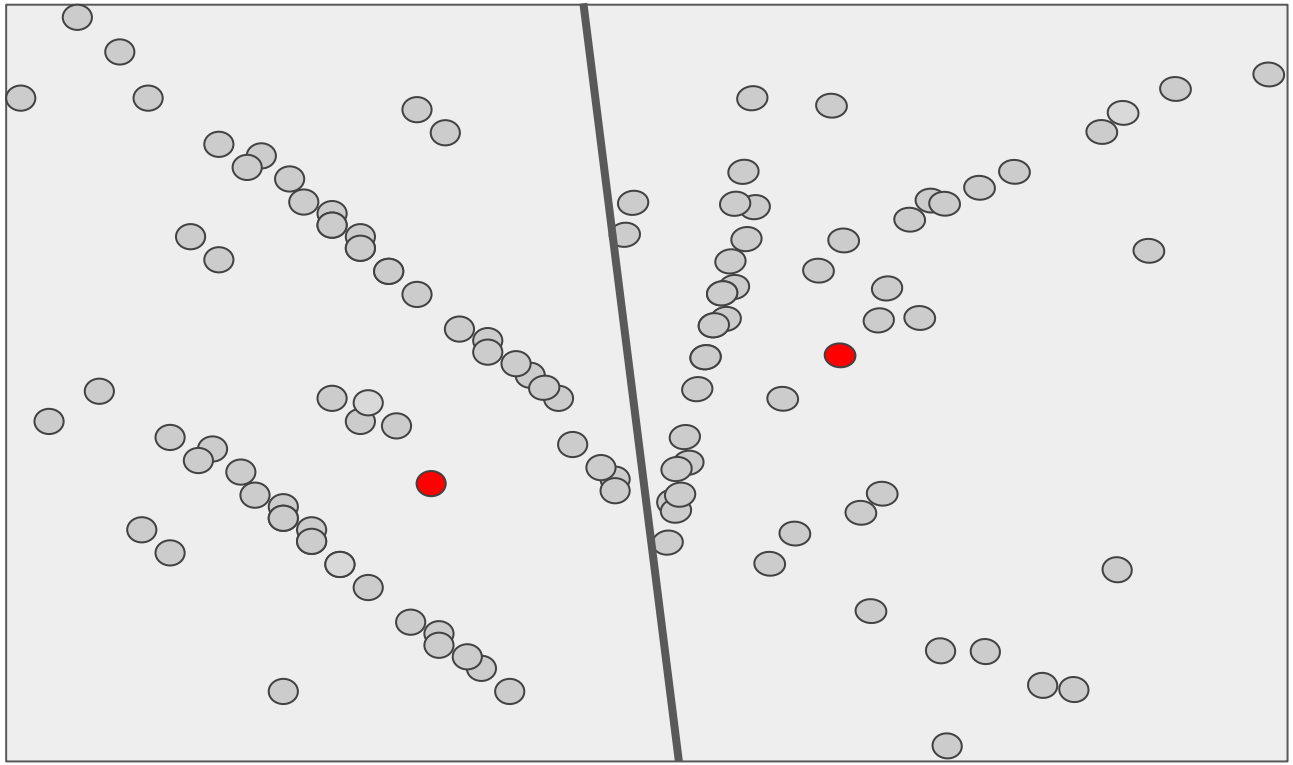
Repeat with the new A'



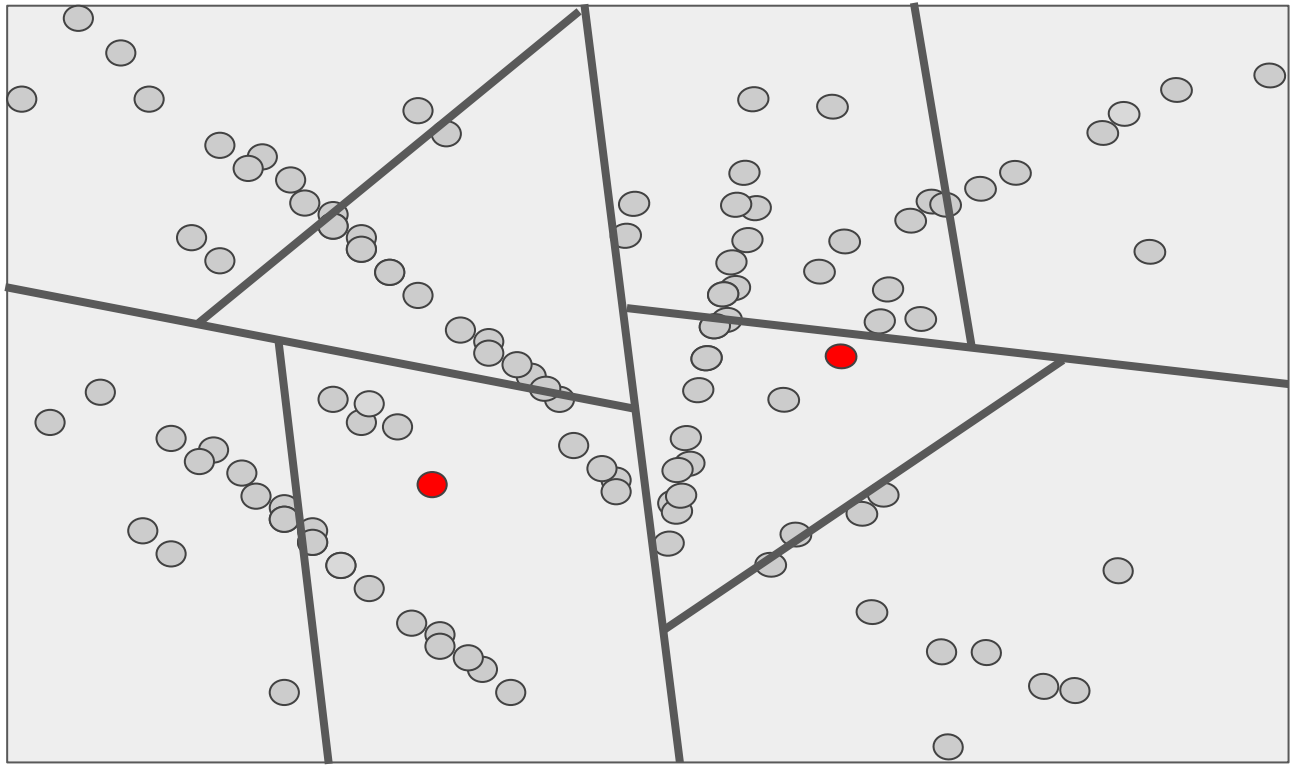
Result: Two “centroids”



Split space

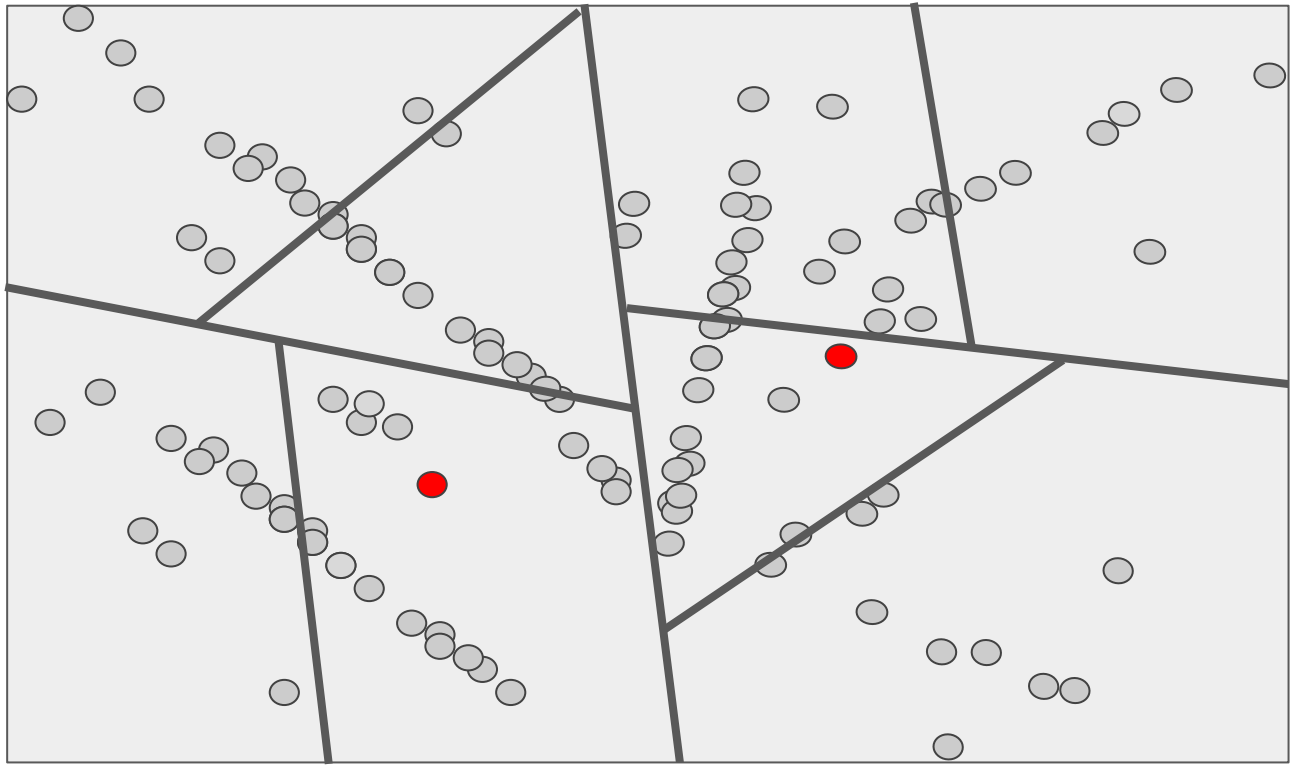


Repeat recursively



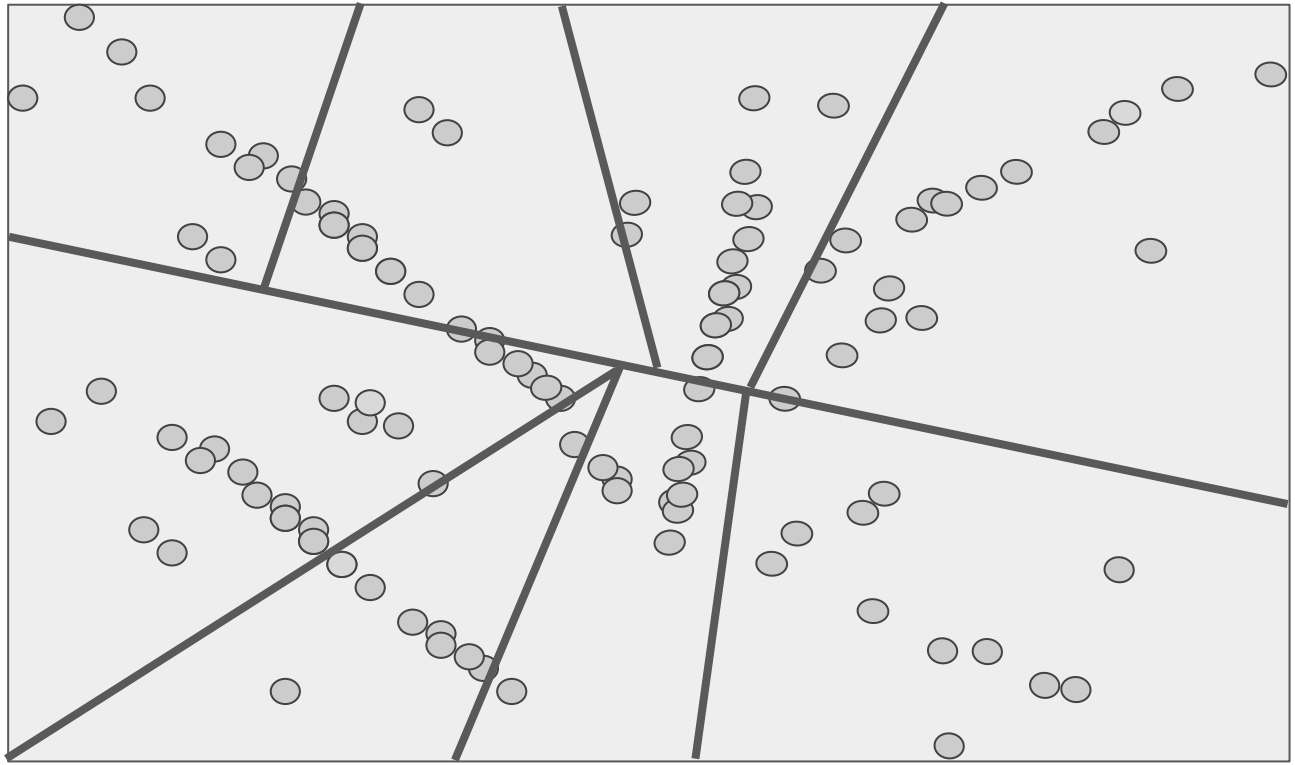
Google

Space “tiled” into a tree, quick search for “right” tile



Google

Use N different trees tiles the space N different ways



Example: ANNoy

- Each tree is a “hash function” that satisfies the requirements

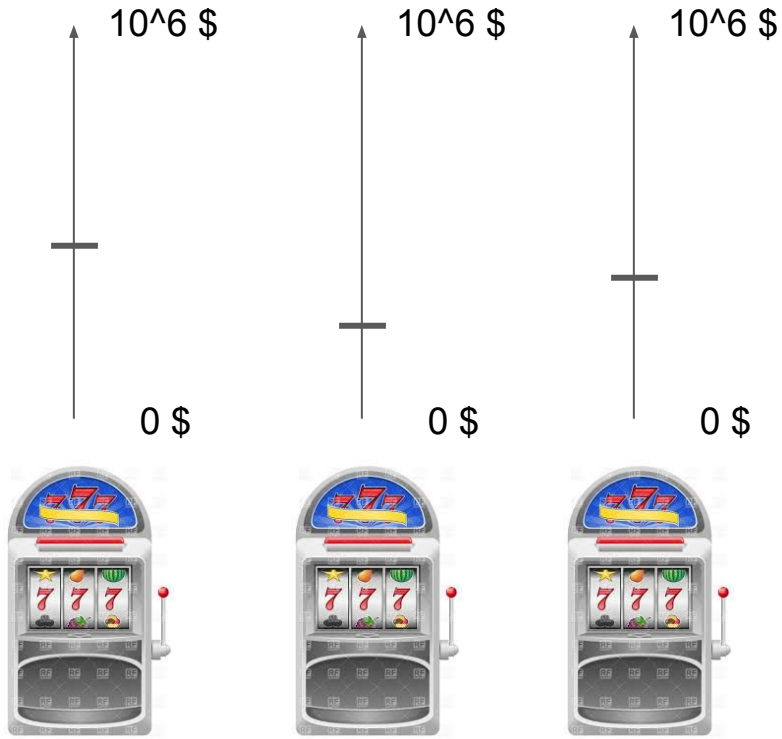
- Points that are close together have high probability to be in the same bucket

3) Monte Carlo Tree Search

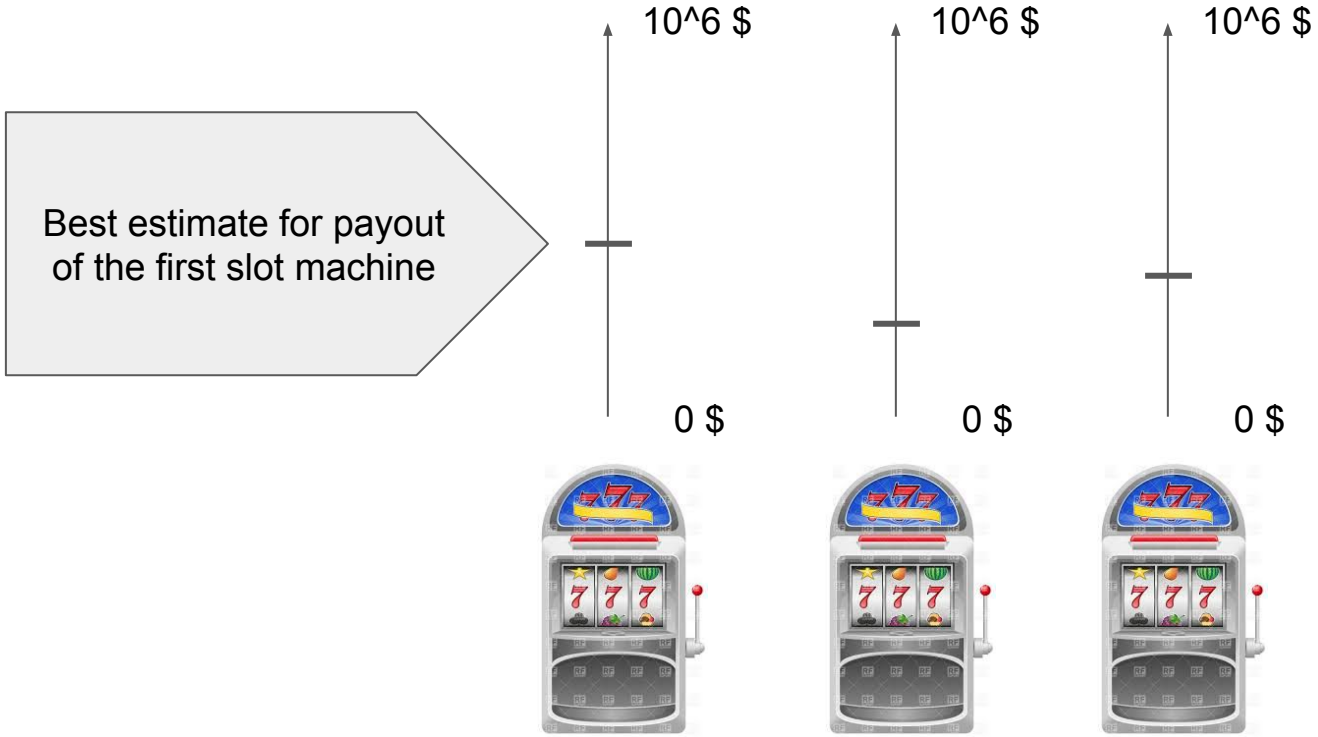
- Generalisation of upper-confidence-bounds (UCB1) algorithm applied to trees
- Invented in 2006
- Algorithm capable of “generic game-playing”
- Clever way to prune large search spaces by “focusing on promising areas first” while still exploring other areas

Basic Idea of MCTS: UCB1 algorithm

- K slot machines with unknown (but fixed) payout distributions
- How to best choose which slot machine to play?
- Assume we have N = n1, n2, n3 samples for the machines already

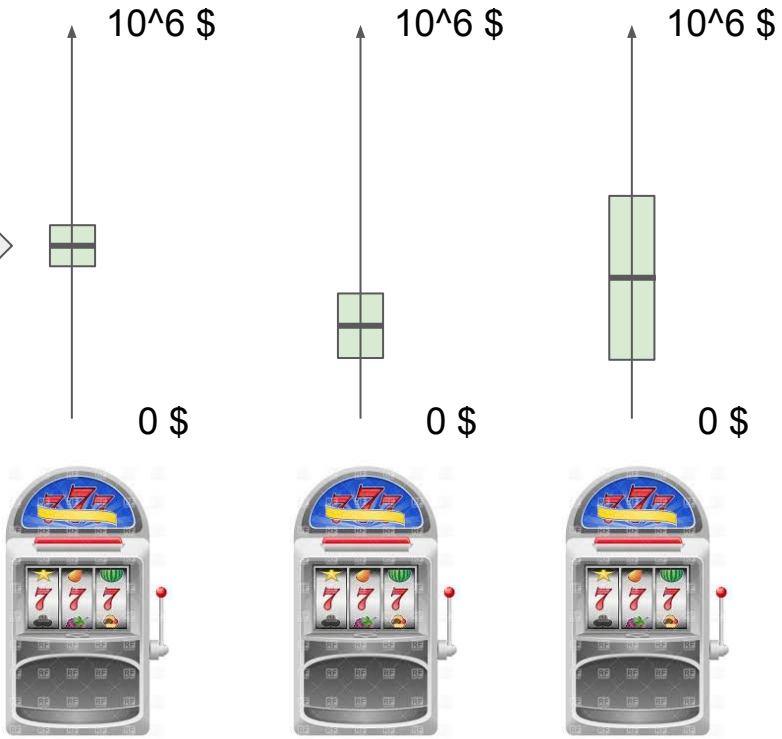


Basic Idea of MCTS: UCB1 algorithm



Basic Idea of MCTS: UCB1 algorithm

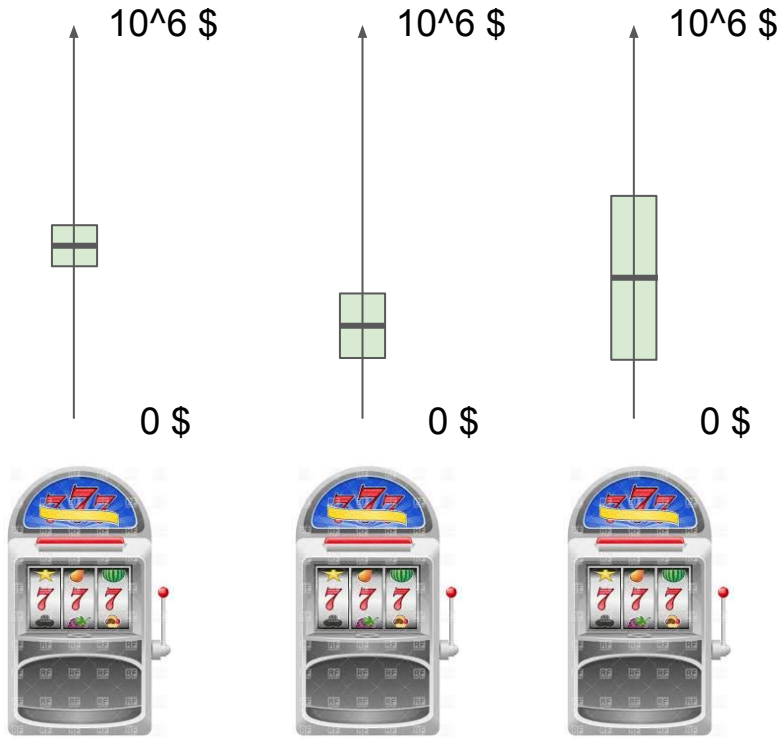
Uncertainty of estimate depends on number of samples



$$estimate \pm \sqrt{\frac{2 \ln N}{n_j}}$$

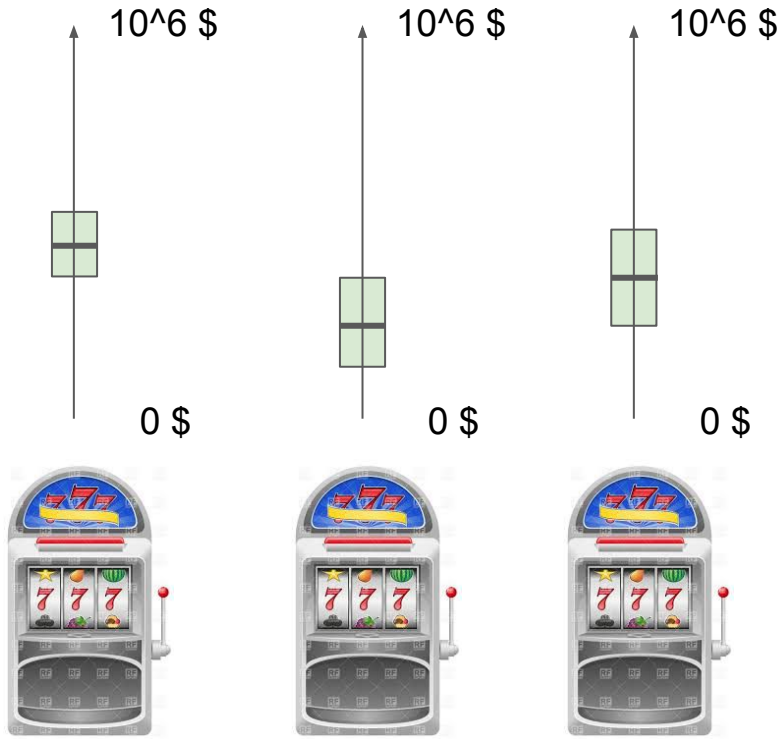
Basic Idea of MCTS: UCB1 algorithm

- Play slot machine with highest upper bound when factoring in uncertainty
- Shrinks uncertainty about that machine
- (Slightly) grows uncertainty about the others



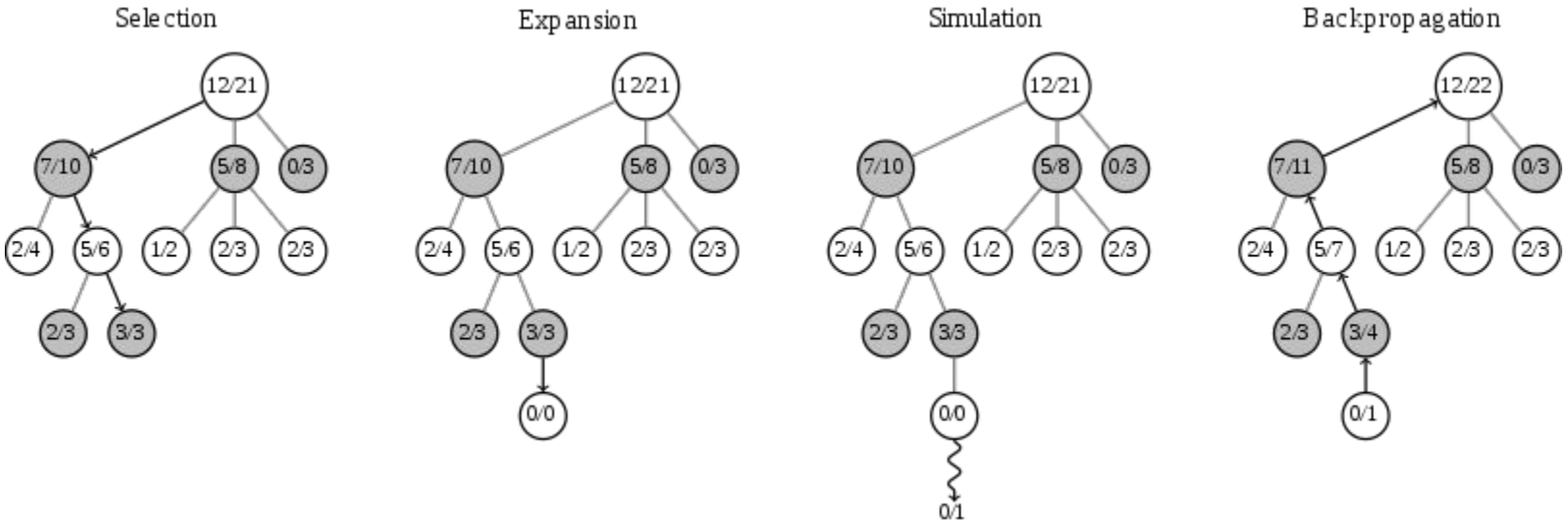
Basic Idea of MCTS: UCB1 algorithm

- Play slot machine with highest upper bound when factoring in uncertainty
- Shrinks uncertainty about that machine
- (Slightly) grows uncertainty about the others



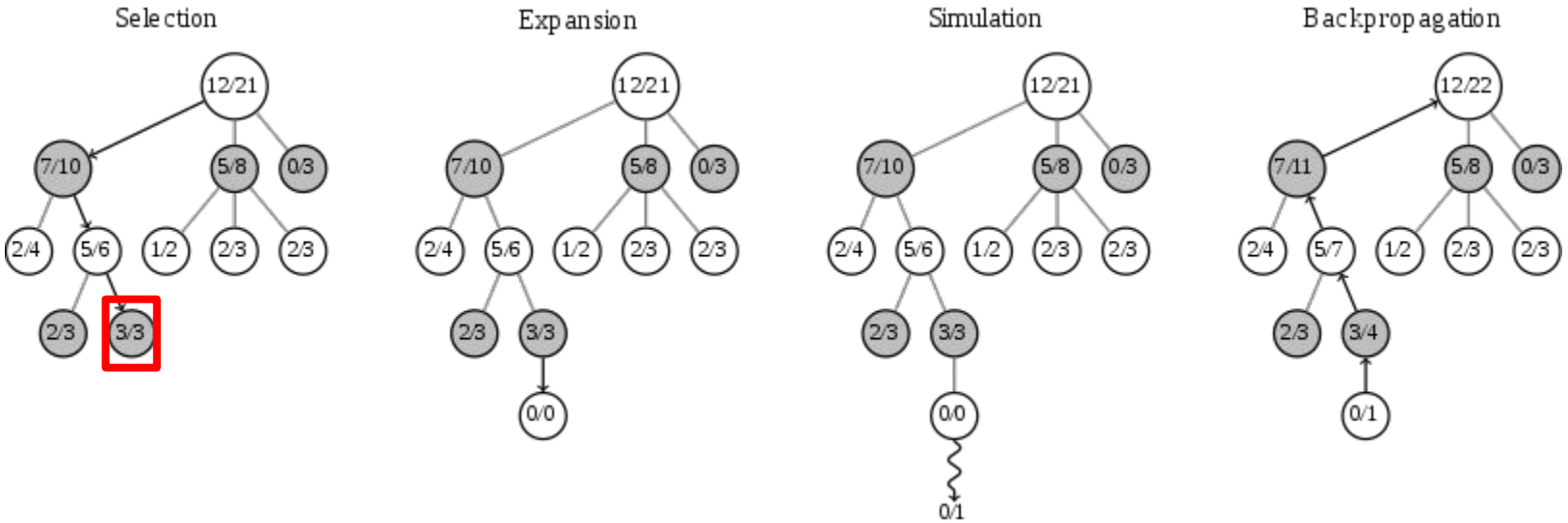
Monte Carlo Tree Search

- Apply UCB1 algorithm to trees



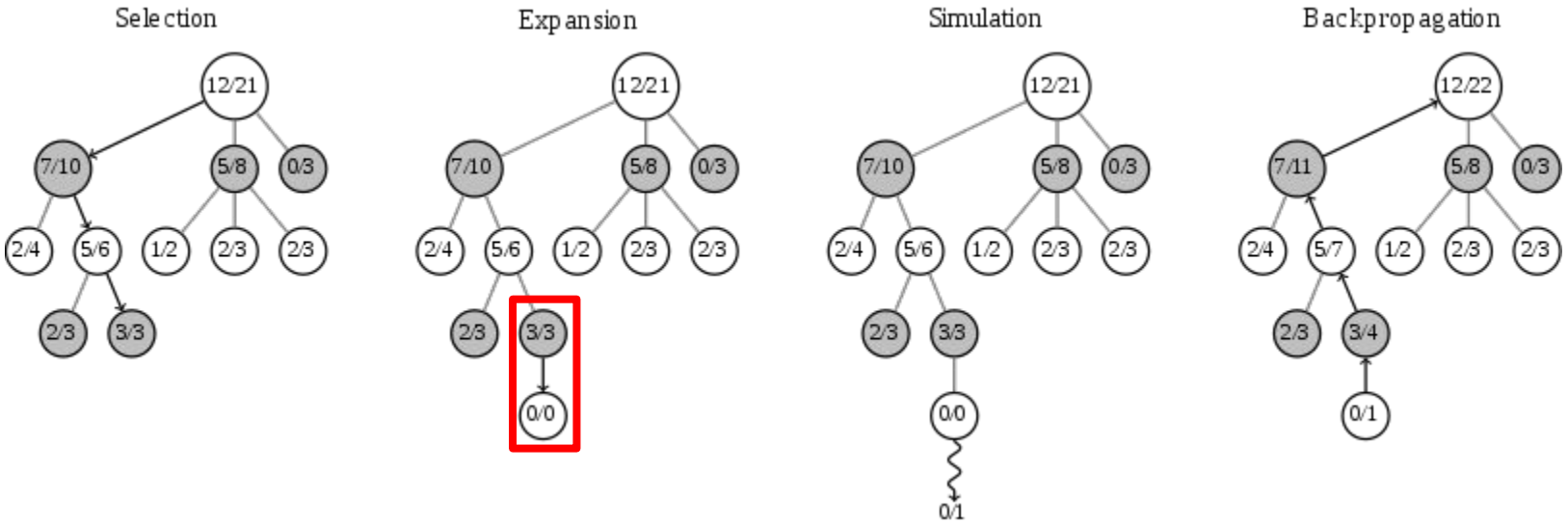
Monte Carlo Tree Search

- Apply UCB1 algorithm to trees



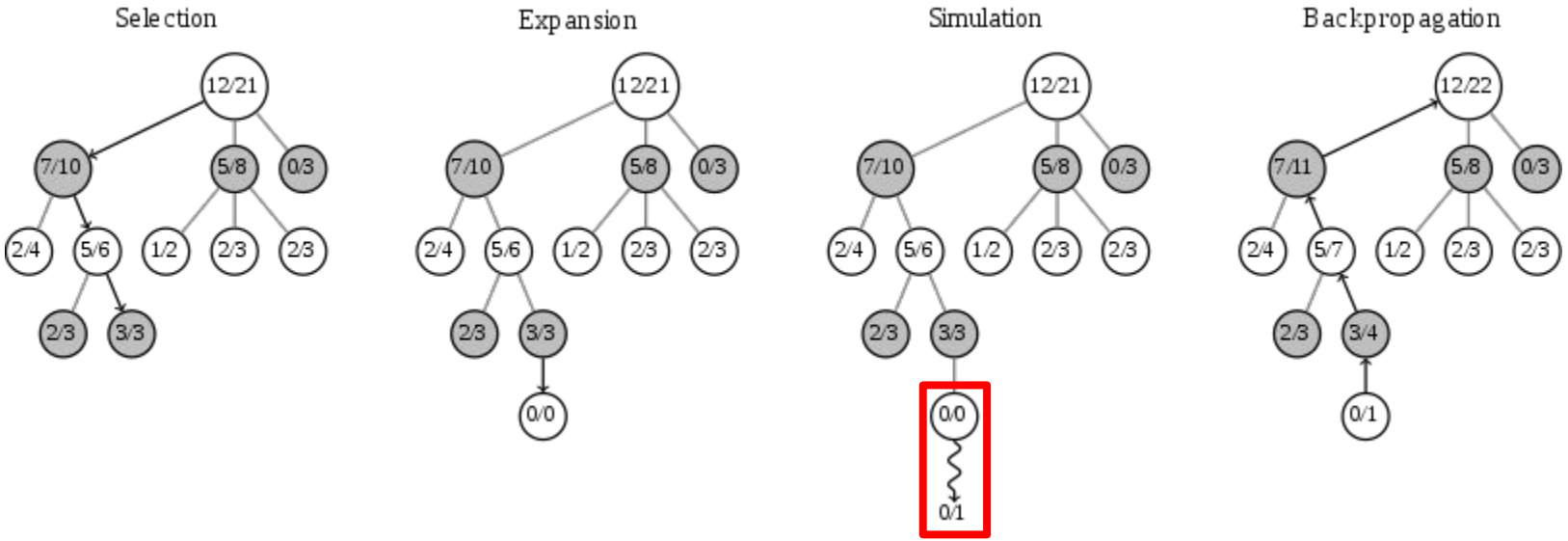
Monte Carlo Tree Search

- Apply UCB1 algorithm to trees



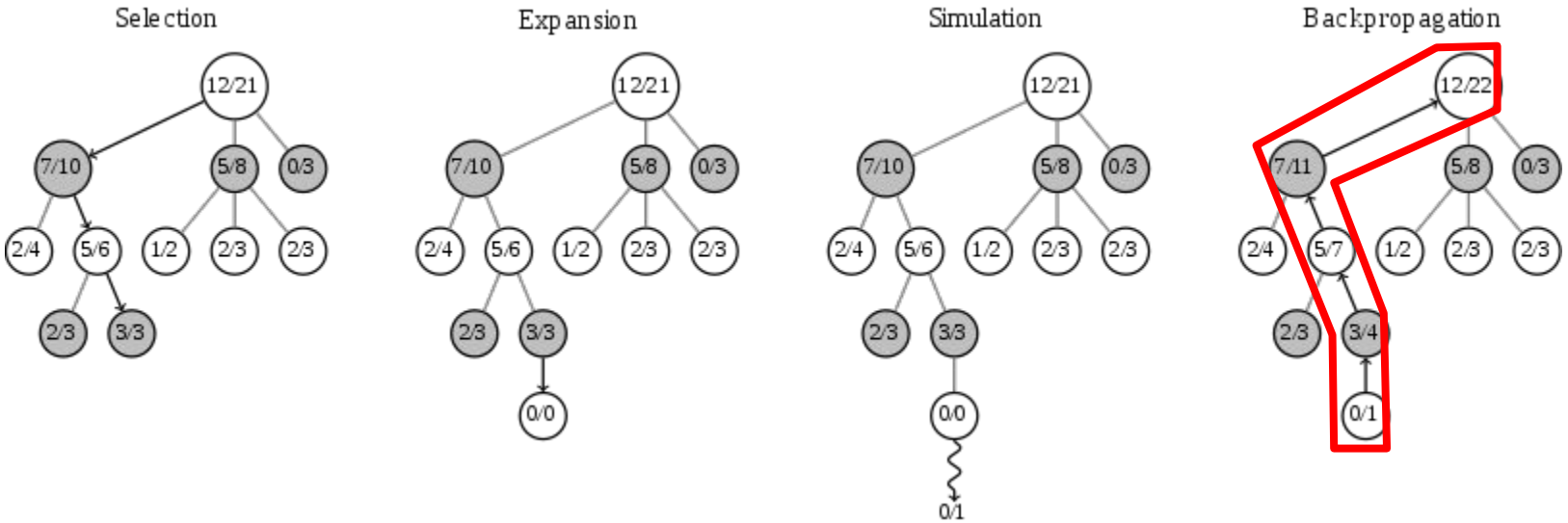
Monte Carlo Tree Search

- Apply UCB1 algorithm to trees



Monte Carlo Tree Search

- Apply UCB1 algorithm to trees



Google

Most practical advances in AI are combinations of these technologies:

1) Image Similarity Search: Cheap Gradients + LSH
    a) ConvNet trained with cheap gradients to extract high-dimensional vector
    b) Similarity search using LSH

2) AlphaGo Zero: Cheap Gradients / Deep Nets + MCTS
    a) ConvNet to predict valuation of positions and good next moves
    b) MCTS using the ConvNet

… etc ...

Most defensive applications of ML in security are “wrong” — why?*

* or at least not the ideal place to apply ML

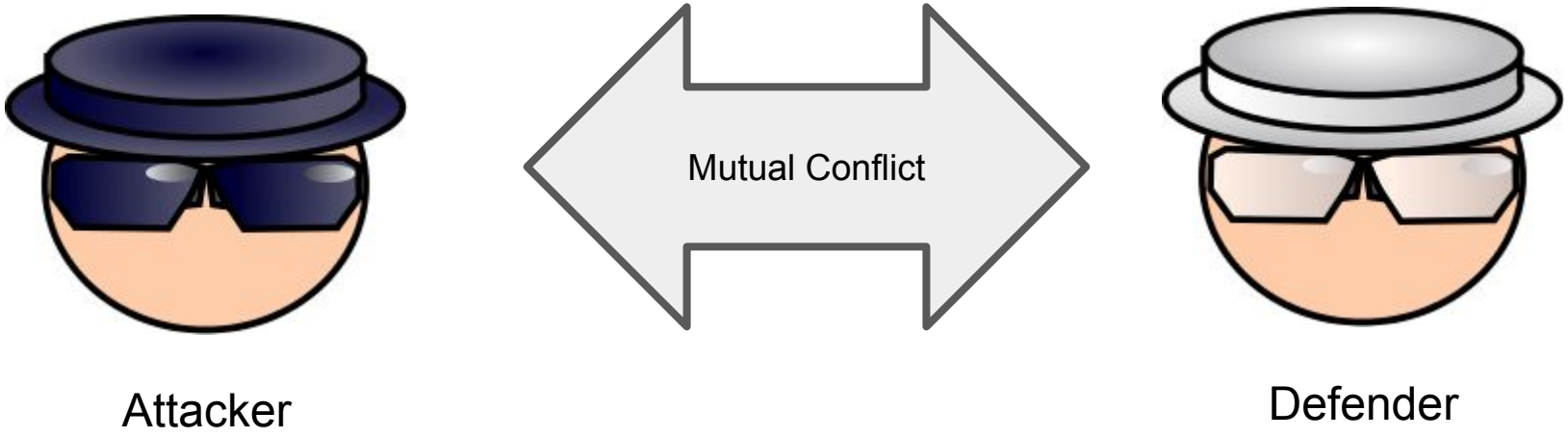
What does ML / AI actually do?

- Machine learning estimates probabilities for a probability distribution it approximates from training data

Golden rule: **“Data you are going to work on needs to come from approximately the same distribution as the data you are training on.”**

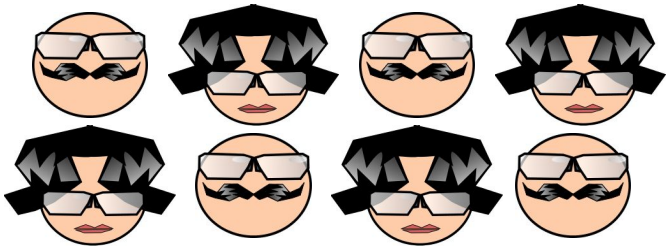
Structure of IT security

Incorrect view of IT security:



Better (simplified) view of IT security

Attacker



“The world”

Defender

Attacker

Defender

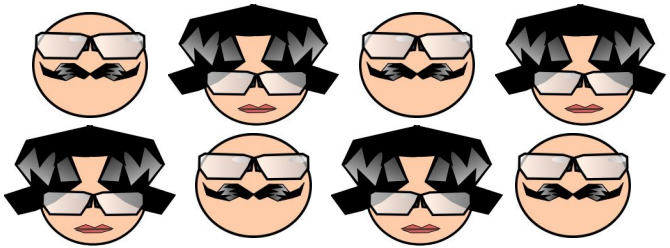
Phish/deceive users

Traverse target network

Find bugs & exploit

Persist

...



“The world”

Attacker

Phish/deceive users

Traverse target network

Find bugs & exploit

Persist

...

Defender

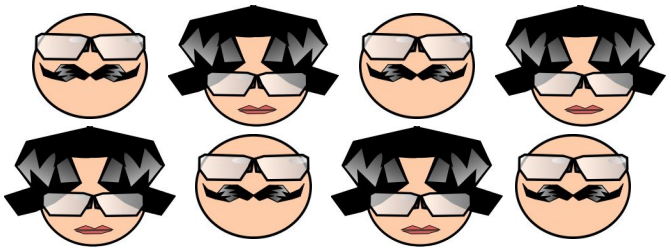
Educate

Make network defensible

Provide safe defaults

Auth & Access Control

...



“The world”

Attacker

Detect

Evade

Defender

Phish/deceive users

Traverse target network

Find bugs & exploit

Persist

...

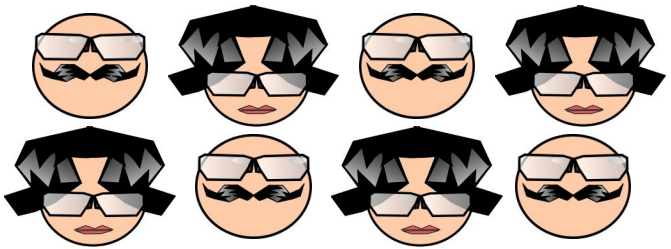
Educate

Make network defensible

Provide safe defaults

Auth & Access Control

...



“The world”

Attacker

Detect

Evade

Defender

Phish/deceive users

Traverse target network

Find bugs & exploit

Persist

...

Educate

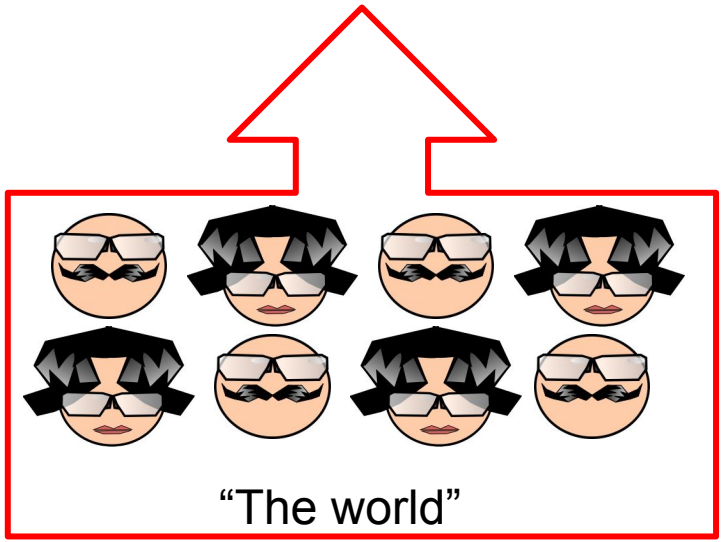
Make network defensible

Provide safe defaults

Auth & Access Control

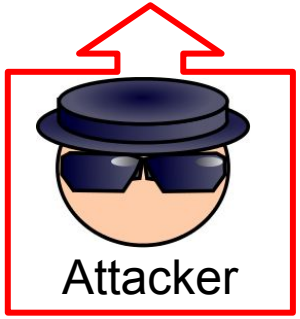
...

Does not change adversarially



"The world"

Changes adversarially



Phish/deceive users

Traverse target network

Find bugs & exploit

Persist

...

Detect

Evade

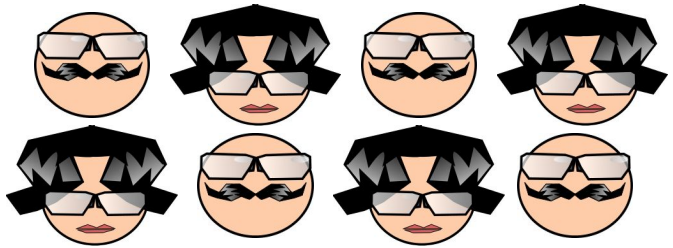
Educate

Make network defensible

Provide safe defaults

Auth & Access Control

...



"The world"

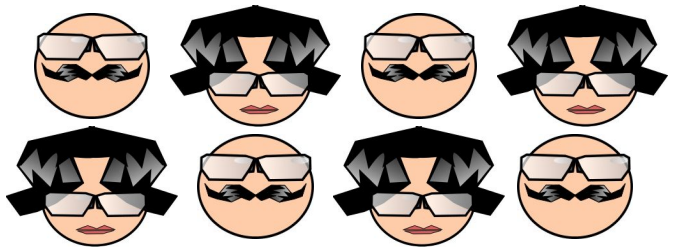
Changes adversarially

Attacker

Detect

Probably the worst place to apply machine learning.

At least the only place that explicitly violates the underlying assumptions of machine learning.



“The world”

Defender

Examples of sketchy ML-in-security applications:

1) Detecting malicious software
2) Detecting malicious behavior
3) Authentication

All of these violate the principle that their training data will resemble their real-world deployment target.

Offensive uses of ML/AI

Where is it already used ?

What are promising future areas ?

Offense is mostly a cottage industry

- Highly artisanal
- Most practitioners have extremely limited tooling
- Work-intensive, lots of unpredictability in development

- Surprisingly averse to adapting new tools, or maintaining them
- **But:** Some tools have made major inroads already, and they can be classified as “machine learning”

Brute force as crucial component of AI/ML

“People underestimate how much of the real-world ML / AI successes consisted of brute-force tree search. There is always a brute deductive component.” — Anonymous smart person

- Deep Blue was enhanced with 480 Chess ASICs for brute-force tree search
- AlphaGo leverages TPUs now (specialized deep learning CPUs), game used 1920 CPUs + 280 GPUs

People do not view brute force as “intelligence”, but it is an important component.

Google

Claim: AFL is already a superhuman ML bugfinder

Wait, what ?

AFL has nothing to do with ML !

Google

Claim: AFL is already a superhuman ML bugfinder

1) More CVEs than any individual ever
2) Cleverness in AFL:
    a) Brute force cleverly combined with …
        i) genetic algorithm
        ii) carefully balanced exploration / exploitation tradeoff
        iii) “curiosity bonus” - new behavior jumps to front of queue
3) Fuzzing corpora from one target can be used against a different target for the same file format / protocol with some benefits
4) Almost all CGC competitors with good results had to run AFL internally

“It works in practice, but does it work in theory?”

- Used to think “innovation comes from theoretical breakthrough”
- Lesson I learnt repeatedly:
  - Innovation comes from something working all of a sudden
  - Theory comes when we try to understand “why?”
- AFL may be one of these situations

Offensive uses of ML / AI: Recognition tasks

Recognition / similarity search tasks:

- Embedding of API patterns into vectorspace & similarity search:
  - “Vulnerability Extrapolation” (link - WOOT’11)
- Unsupervised clustering + LSH for finding vulnerable libraries in firmware:
  - “Scalable Graph-based Bug Search for Firmware Images” (CCS’16).
- Supervised embeddings + LSH for doing the same:
  - “Neural Network-based Graph Embedding for Cross-Platform Binary Code Similarity Detection” (link - CCS’17)

Offensive uses of ML / AI: Exploitation tasks

Heap layout manipulation has distinct similarities with board games.

- Black-box randomized tree search for heap layout:
  - “Heap Layout Optimization for Exploitation” (link - Blackhat EU 2017)

Future offensive uses of ML / AI

Anywhere where we have a “stable distribution”:

- **Bug detection** -- programmers do not change quickly
- **Exploitation** -- heap implementations and their interaction with a program do not change quickly
- **Phishing / user deception** -- learning how to generate a good phishing email looks like a classical ML problem
- **Autonomous lateral movement** -- most Windows-centric organisation have a somewhat similar structure, and winning moves transfer between targets
- **Password cracking** -- good models of passwords prioritize high-probability words

Bug detection

1. Analysis and ranking of source / binary updates for security relevance
   a. Vendors ship updates full of changes
   b. Which ones are security-relevant, and can they be backported (semi-)automatically?

2. Inferring the intent of a piece of code
   a. When auditing, I often look for “does this code do what the programmer wants it to do”
   b. Inferring intent is difficult, but once we get there, this sort of analysis can be automated

3. Finding cut & paste vulnerabilities, repetitions etc.
   a. Bugs are quite often duplicated elsewhere, and similarity search is now easy

Google

Improved Fuzzing

1. Better / more conscious exploration / exploitation tradeoffs
    a. AFL is good, but where is the limit? How much can it be improved?

2. GAN's for generating better input sample corpora
    a. GAN's are specialized networks that learn to "mimic" a target distribution
    b. Successful at faking human faces etc. - should be successful at faking input to applications by example

Exploitation

1. Exploration of alternate heap layouts after a fuzz-crash
   a. Mostly a matter of brute force?

2. Learning how to manipulate heap layouts for one target
   a. Many targets are attacked recurrently
   b. Learning heap manipulation will transfer between similar bugs

3. Automatic discovery of exploit primitives (ideally with return to stability)
   a. Attackers need to combine several “weird machine instructions”
   b. Discovery of these instructions is still a mostly manual task
   c. Large-scale brute-force exploration will likely yield situations that humans overlook
   d. Time-scale compression if combined with automated patch diffing?

Phishing / User deception

Internet giants are good at machine learning because they have used ML for years to maximize click-through rates.

Phishers also want to maximize click-through rates.

Deception may be “killer application” for generative models.

Defensive uses of ML/AI

Is deception the killer defensive application of ML ?

Deception in 1960’s espionage

- If a spy was identified, the information source could be “poisoned” by providing false information
- This does not work in cyber - volume of exfiltrated information is too high
- No defender can generate enough garbage information to feed to an attacker
- Do advances in machine learning change this?

Fake employee directory?

GAN-generated photos after training on celebrity images

(note that the GAN learns that women’s faces have no pores :-/)

Fake source code?

```
static void do_command(struct seq_file *m, void *v)
{
  int column = 32 << (cmd[2] & 0x80);
  if (state)
    cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
  else
    seq = 1;
  for (i = 0; i < 16; i++) {
    if (k & (1 << 1))
      pipe = (in_use & UMXTHREAD_UNCCA) +
        ((count & 0x00000000fffffff8) & 0x000000f) << 8;
    if (count == 0)
      sub(pid, ppc_md.kexec_handle, 0x20000000);
    pipe_set_bytes(i, 0);
  }
  /* Free our user pages pointer to place camera if all dash */
  subsystem_info = &of_changes[PAGE_SIZE];
  rek_controls(offset, idx, &soffset);
  /* Now we want to deliberately put it to device */
  control_check_polarity(&context, val, 0);
  for (i = 0; i < COUNTER; i++)
    seq_puts(s, "policy ");
}
```

Fake networks? Emails? Blueprints?

Deception in 1960’s espionage

- If a spy was identified, the information source could be “poisoned” by providing false information
- This does not work in cyber - volume of exfiltrated information is too high
- No defender can generate enough garbage information to feed to an attacker
- **Do advances in machine learning change this?**
- **Or are we targeting the wrong distribution again?**

Summary (1)

- Not all of the AI hype is bullshit

- Real progress is being made:
  - Algorithmic breakthroughs, orders of magnitude speedups
  - Heterogeneous computing changes the boundary of the possible

Summary (2)

- AI and ML is less magic than you think - mostly college-level applied math + lots of computational power

Summary (3)

- Biggest potential is for problems with “stable-in-time distributions”
- Most offensive problems fall into this category
- Many defensive problems do, too - but seem not to be focus of ML efforts?

Google

Enjoy the rest of the conference :-)

Attacker

Defender