DAT2 week 9

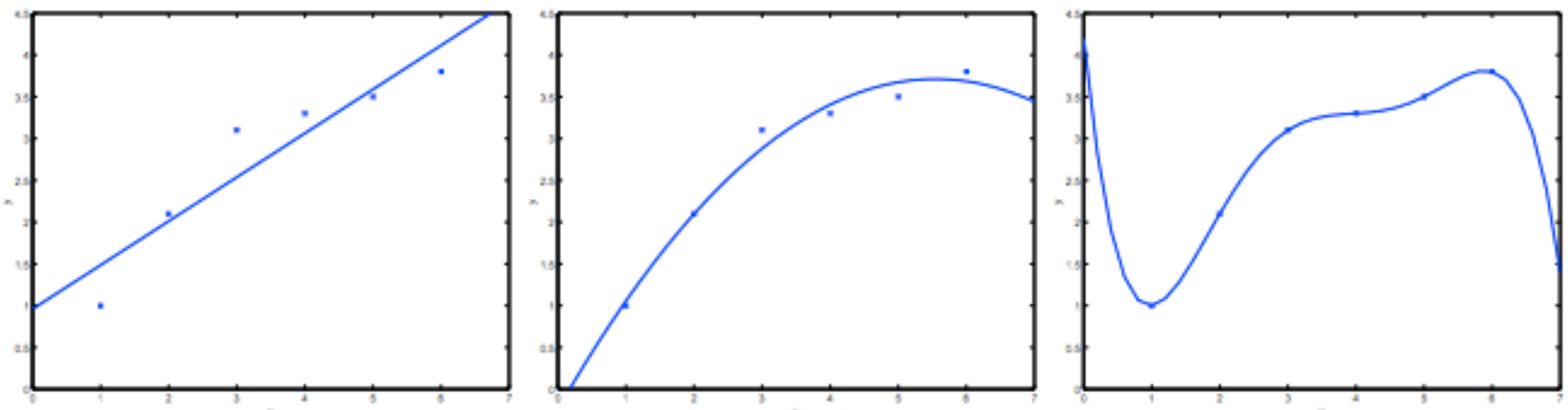
Misrab

Agenda

- Learning Theory
- Reinforcement Learning
- CS
- HMMs

Learning Theory

Bias / Variance: parameters, models



How much data needed?

Which models to even consider?

Learning Theory

Lemma. (The union bound). Let $A_1, A_2, \ldots, A_k$ be k different events (that may not be independent). Then

$$P(A_1 \cup \cdots \cup A_k) \leq P(A_1) + \ldots + P(A_k).$$

Lemma. (Hoeffding inequality) Let $Z_1, \ldots, Z_m$ be m independent and identically distributed (iid) random variables drawn from a Bernoulli(ϕ) distribution. I.e., $P(Z_i = 1) = \phi$, and $P(Z_i = 0) = 1 - \phi$. Let $\hat{\phi} = (1/m)\sum_{i=1}^{m} Z_i$ be the mean of these random variables, and let any $\gamma > 0$ be fixed. Then

$$P(|\phi - \hat{\phi}| > \gamma) \leq 2\exp(-2\gamma^2 m)$$

How many training points (samples) to be within 0.01 of each other with probability 95%?

Learning Theory

Binary classification empirical error

$$\hat{\varepsilon}(h) = \frac{1}{m}\sum_{i=1}^{m} 1\{h(x^{(i)}) \neq y^{(i)}\}.$$

(fraction misclassified)

Generalisation error

$$\varepsilon(h) = P_{(x,y)\sim\mathcal{D}}(h(x) \neq y).$$

Learning Theory

PAC framework
"probably approximately correct"

i.e. training and testing from same distribution

Learning Theory

Minimising training error

$$\hat{\theta} = \arg\min_{\theta} \hat{\varepsilon}(h_\theta).$$

$$\mathcal{H} = \{h_\theta : h_\theta(x) = 1\{\theta^T x \geq 0\}, \theta \in \mathbb{R}^{n+1}\}$$

equivalent to $\hat{h} = \arg\min_{h \in \mathcal{H}} \hat{\varepsilon}(h)$

Learning Theory

Finite $\mathcal{H} = \{h_1, \ldots, h_k\}$

We want guarantees on generalisation error when we minimise empirical error

- theoretical framework
- applied cases e.g. too few data points for cross-validation

Learning Theory

- Take fixed h_i in H

- Let $Z_j = 1\{h_i(x^{(j)}) \neq y^{(j)}\}.$

- Training error of h_i

$$\hat{\varepsilon}(h_i) = \frac{1}{m}\sum_{j=1}^{m} Z_j.$$

- Hoeffding

$$P(|\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma) \leq 2\exp(-2\gamma^2 m).$$

Learning Theory

Great. But we don't just want this for one h_i, we want it for all of H simultaneously

$$P(|\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma) \leq 2\exp(-2\gamma^2 m).$$

$$P(A_i)$$

$$\begin{aligned} P(\exists\, h \in \mathcal{H}.|\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma) &= P(A_1 \cup \cdots \cup A_k) \\ &\leq \sum_{i=1}^{k} P(A_i) \\ &\leq \sum_{i=1}^{k} 2\exp(-2\gamma^2 m) \\ &= 2k\exp(-2\gamma^2 m) \end{aligned}$$

Learning Theory

$$
\begin{aligned}
P(\exists\, h \in \mathcal{H}. |\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma) &= P(A_1 \cup \cdots \cup A_k) \\
&\leq \sum_{i=1}^{k} P(A_i) \\
&\leq \sum_{i=1}^{k} 2\exp(-2\gamma^2 m) \\
&= 2k\exp(-2\gamma^2 m)
\end{aligned}
$$

If we subtract both sides from 1, we find that

$$
\begin{aligned}
P(\neg\exists\, h \in \mathcal{H}. |\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma) &= P(\forall h \in \mathcal{H}. |\varepsilon(h_i) - \hat{\varepsilon}(h_i)| \leq \gamma) \\
&\geq 1 - 2k\exp(-2\gamma^2 m)
\end{aligned}
$$

Learning Theory

Example

For instance, we can ask the following question: Given γ and some $\delta > 0$, how large must m be before we can guarantee that with probability at least $1 - \delta$, training error will be within γ of generalization error? By setting $\delta = 2k \exp(-2\gamma^2 m)$ and solving for m, [you should convince yourself this is the right thing to do!], we find that if

$$m \geq \frac{1}{2\gamma^2} \log \frac{2k}{\delta},$$

then with probability at least $1 - \delta$, we have that $|\varepsilon(h) - \hat{\varepsilon}(h)| \leq \gamma$ for all $h \in \mathcal{H}$. (Equivalently, this shows that the probability that $|\varepsilon(h) - \hat{\varepsilon}(h)| > \gamma$ for some $h \in \mathcal{H}$ is at most δ.) This bound tells us how many training examples we need in order make a guarantee. The training set size m that a certain method or algorithm requires in order to achieve a certain level of performance is also called the algorithm's **sample complexity**.

Learning Theory

With just a bit more effort, we can get
an explicit guarantee on generalisation error

Theorem. Let $|\mathcal{H}| = k$, and let any m, δ be fixed. Then with probability at least $1 - \delta$, we have that

$$\varepsilon(\hat{h}) \leq \left(\min_{h \in \mathcal{H}} \varepsilon(h)\right) + 2\sqrt{\frac{1}{2m} \log \frac{2k}{\delta}}.$$

lower bias: increasing
H class

variance tradeoff of
increasing H class

Learning Theory

In summary, given a hypothesis class of complexity k, we can figure out how many training examples we need to be within a certain distance from minimising generalisation error when carrying out empirical risk minimisation

Analog for continuous case:
Vapnik-Chervonenki (VC) dimension

Reinforcement Learning

Previously, y had a “correct” value

Now instead, what if the best we can provide is a reward function

e.g. robot motion, gaming…trading?

https://www.youtube.com/watch?v=qv6UVOQ0F44

https://www.youtube.com/watch?v=M8YjvHYbZ9w

https://www.youtube.com/watch?v=rVlhMGQgDkY

Reinforcement Learning

A Markov decision process is a tuple $(S, A, \{P_{sa}\}, \gamma, R)$, where:

- S is a set of **states**. (For example, in autonomous helicopter flight, S might be the set of all possible positions and orientations of the helicopter.)

- A is a set of **actions**. (For example, the set of all possible directions in which you can push the helicopter's control sticks.)

- P_{sa} are the state transition probabilities. For each state $s \in S$ and action $a \in A$, P_{sa} is a distribution over the state space. We'll say more about this later, but briefly, P_{sa} gives the distribution over what states we will transition to if we take action a in state s.

- $\gamma \in [0, 1)$ is called the **discount factor**.

- $R : S \times A \mapsto \mathbb{R}$ is the **reward function**. (Rewards are sometimes also written as a function of a state S only, in which case we would have $R : S \mapsto \mathbb{R}$).

Reinforcement Learning

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_3 \xrightarrow{a_3} \dots$$

$$s_1 \sim P_{s_0 a_0}.$$

Upon visiting the sequence of states $s_0, s_1, \dots$ with actions $a_0, a_1, \dots$, our total payoff is given by

$$R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \cdots .$$

Or, when we are writing rewards as a function of the states only, this becomes

$$R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots .$$

Reinforcement Learning

We want to maximise expected discounted reward

Economic interpretation of gamma?

$$\mathrm{E}\left[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots\right]$$

Reinforcement Learning

Policy to choose actions

$$\pi : S \mapsto A$$

$$a = \pi(s)$$

Value function based on policy

$$V^{\pi}(s) = \mathrm{E}\left[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots \,\middle|\, s_0 = s, \pi\right].$$

Reinforcement Learning

$$V^{\pi}(s) = \mathrm{E}\left[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots \,\middle|\, s_0 = s, \pi\right].$$

Bellman equation

$$V^{\pi}(s) = R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^{\pi}(s').$$

Can you draw this (perhaps as a decision tree?)

Reinforcement Learning

We want to find the optimal value

$$V^*(s) = \max_{\pi} V^{\pi}(s).$$

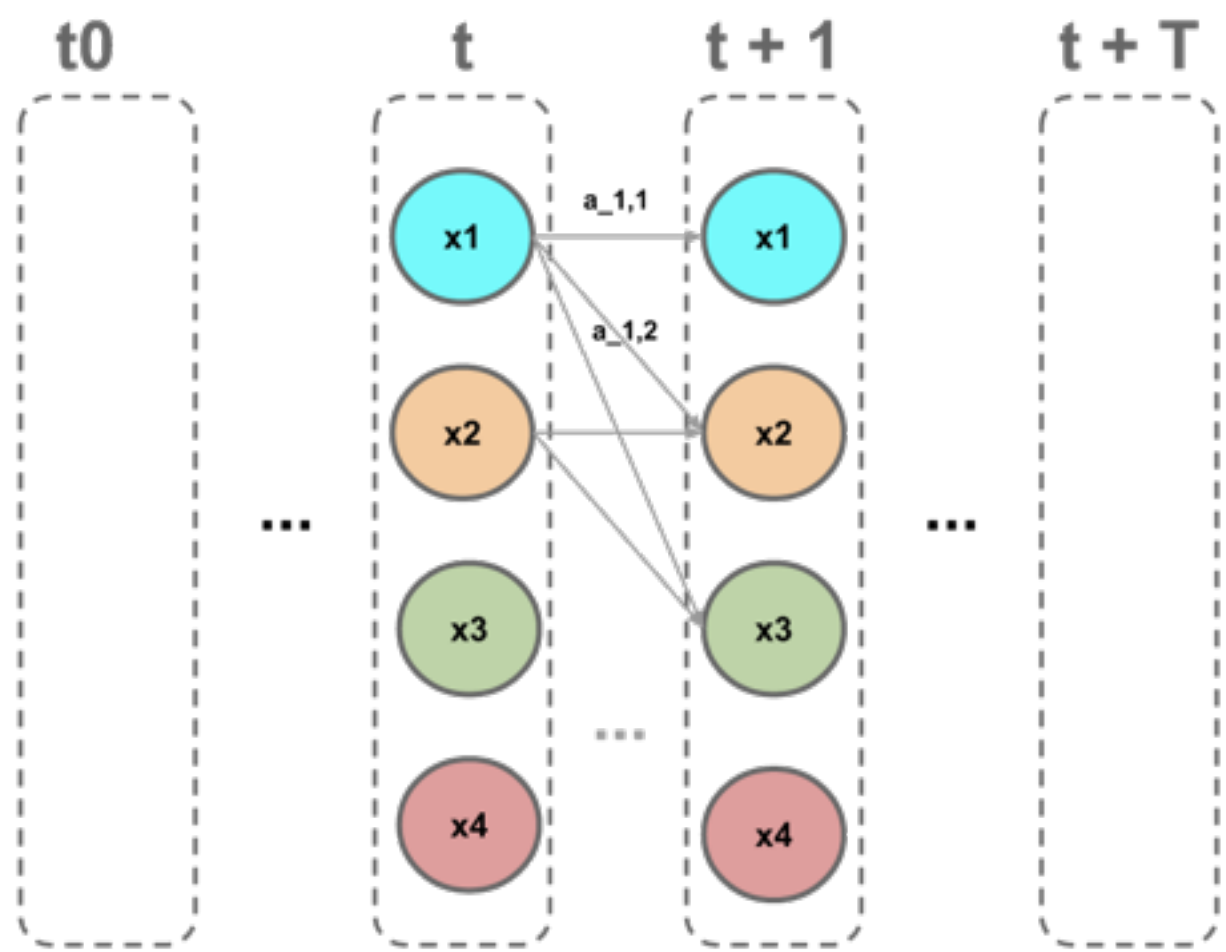
or

$$V^*(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} P_{sa}(s') V^*(s').$$

max over possible actions

Reinforcement Learning

Exponential paths to explore



Hansel and Grettel!

Reinforcement Learning

Recommended for Dynamic Programming and Value Iteration:

https://en.wikipedia.org/wiki/Memoization

https://www.youtube.com/watch?v=oefOCk3koZo

https://www.youtube.com/watch?v=ip4iSMRW5X4

Reinforcement Learning

Extensions:

Learning P_sa

Continuous space (discretisation)

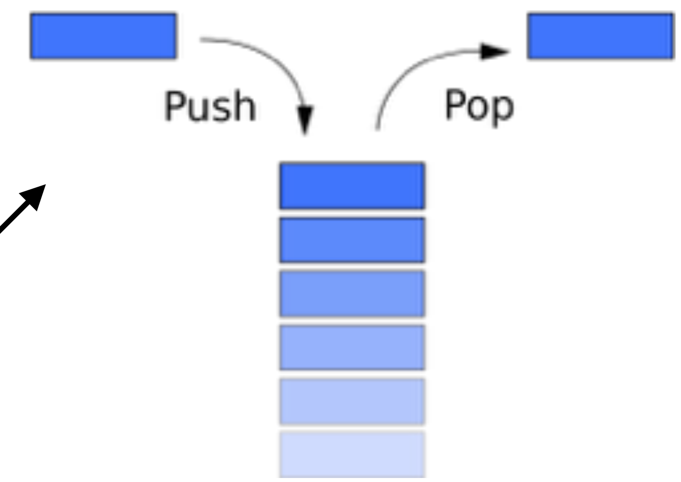
Computer Science

Scope and the call stack

```
def f():
  x = 2
  return x + 4


x = 3
print f()

# what is printed?
```

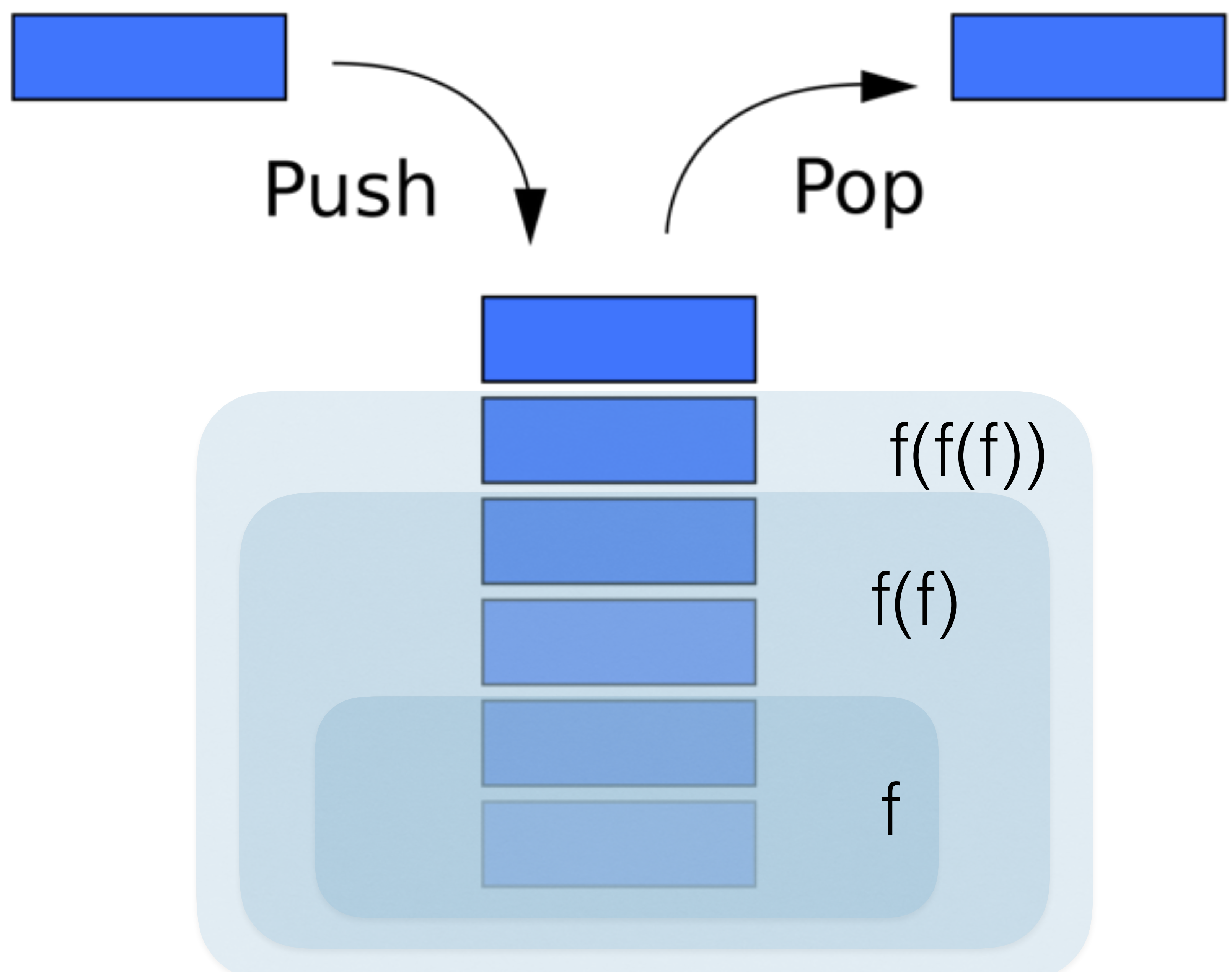



```
def f():
  x = 2
  return x + 4
```

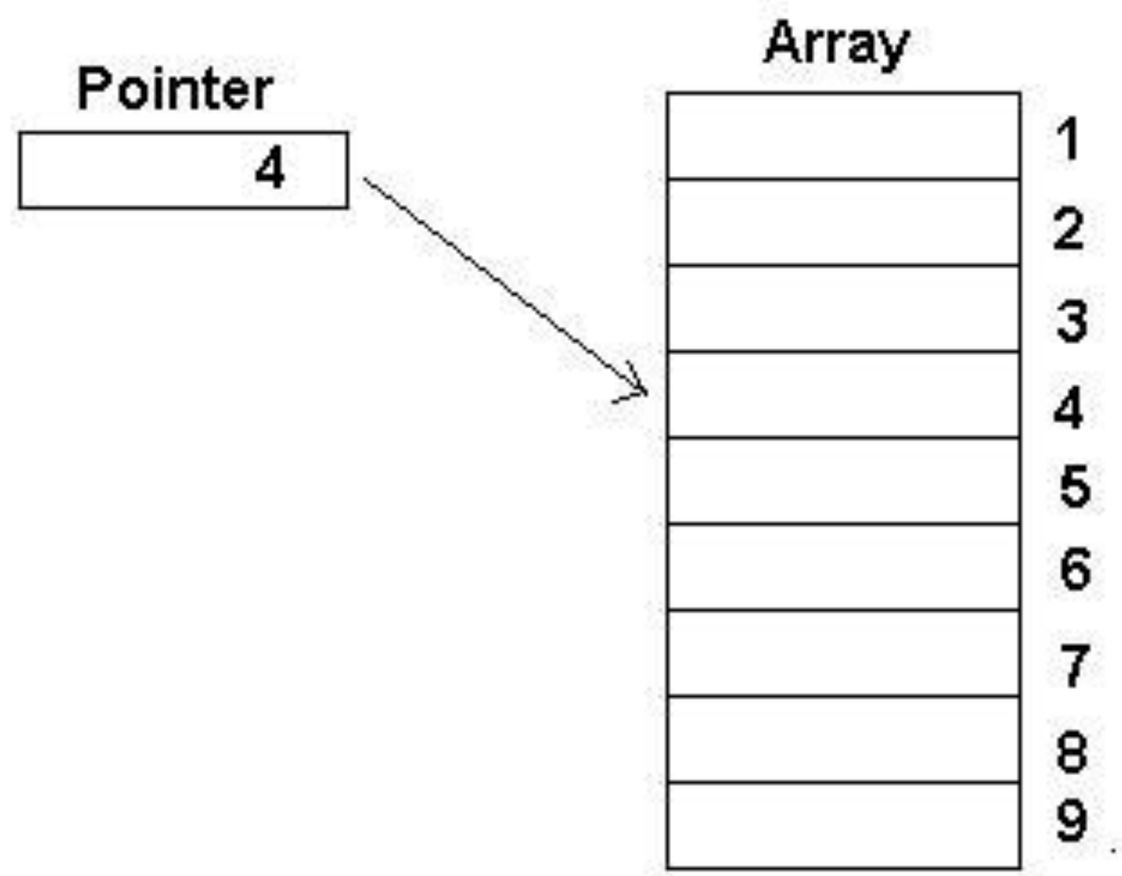
```
x = 3
print f()

# what is printed?
```

Recursion: function calling itself



Pointers!



How pythons can bite!

```
arr = [ { … } , { … }, { … } ]
results = []

for …
  x = arr[i]
  x.moo = “foo”

  results.append(x)
```

What’s wrong?

…

results = [x, x, x, x]
x —> the last item

```
arr = [ { … } , { … }, { … } ]
results = []

for …
  x = copy.deepcopy(arr[i])
  x.moo = "foo"

  results.append(x)
```

Practice!

```
int *moo  // pointer
int moo // actual value

int moo = 3
&moo // address of value

int *moo
*moo = 3 // set value!
```

http://www.gdsw.at/languages/c/programming-bbrown/c_0771.htm

Hidden Markov Models

Sequence of words over time based on audio

Other examples?

Easy case: we observe the actual state

S = { rain, sun, cloud }

Observed over time:

z1 = sun, z2 = cloud, z3 = sun…

$$P(z_t|z_{t-1}, z_{t-2}, ..., z_1) = P(z_t|z_{t-1})$$

Simplifying Assumptions

Limited horizon: $$P(z_t|z_{t-1}, z_{t-2}, ..., z_1) = P(z_t|z_{t-1})$$

Stationary process: $$P(z_t|z_{t-1}) = P(z_2|z_1);\ t \in 2...T$$

Can you explain this in a picture?

State transition matrix

$$A = \begin{array}{c|cccc} & s_0 & s_{sun} & s_{cloud} & s_{rain} \\ \hline s_0 & 0 & .33 & .33 & .33 \\ s_{sun} & 0 & .8 & .1 & .1 \\ s_{cloud} & 0 & .2 & .6 & .2 \\ s_{rain} & 0 & .1 & .2 & .7 \end{array}$$

What's the prob of a given sequence?

$$
\begin{aligned}
P(\vec{z}) &= P(z_t, z_{t-1}, ..., z_1; A) \\
&= P(z_t, z_{t-1}, ..., z_1, z_0; A) \\
&= P(z_t|z_{t-1}, z_{t-2}, ..., z_1; A)P(z_{t-1}|z_{t-2}, ..., z_1; A)...P(z_1|z_0; A) \\
&= P(z_t|z_{t-1}; A)P(z_{t-1}|z_{t-2}; A)...P(z_2|z_1; A)P(z_1|z_0; A)
\end{aligned}
$$

$$
\begin{aligned}
&= \prod_{t=1}^{T} P(z_t|z_{t-1}; A) \\
&= \prod_{t=1}^{T} A_{z_{t-1}\, z_t}
\end{aligned}
$$

Compute this:

$$P(z_1 = s_{sun}, z_2 = s_{cloud}, z_3 = s_{rain}, z_4 = s_{rain}, z_5 = s_{cloud})$$

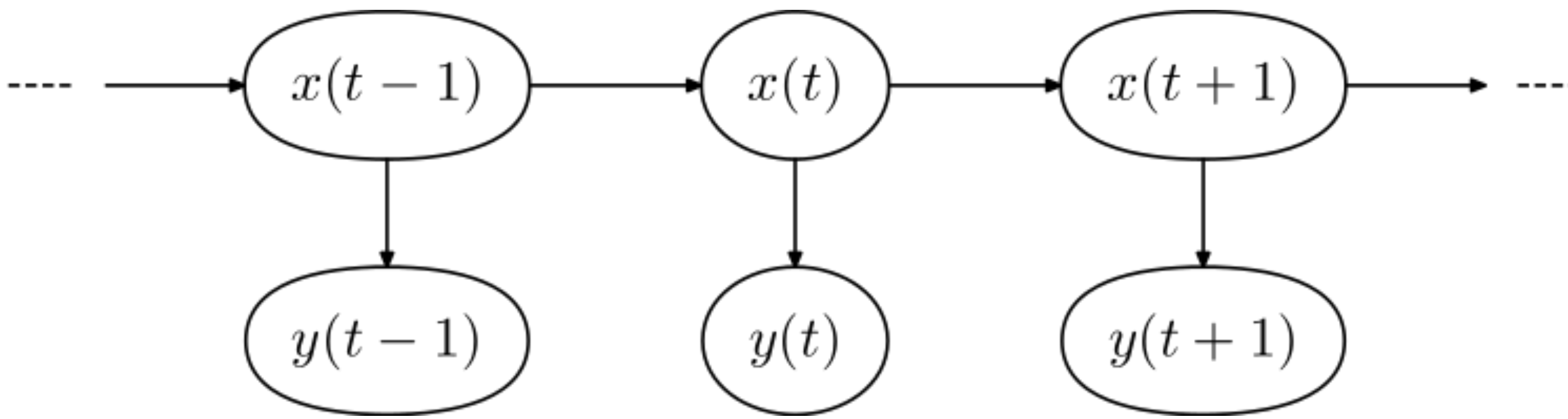
$$A = \begin{array}{c|cccc} & s_0 & s_{sun} & s_{cloud} & s_{rain} \\ \hline s_0 & 0 & .33 & .33 & .33 \\ s_{sun} & 0 & .8 & .1 & .1 \\ s_{cloud} & 0 & .2 & .6 & .2 \\ s_{rain} & 0 & .1 & .2 & .7 \end{array}$$

If we had a sequence $\mathbf{z}$, we could use maximum likelihood to figure out $\mathbf{A}$

Examples?

Hidden Markov Models

We don't observe the sequence directly



e.g. words vs audio waves

New matrix **B** that also tells us $P(y_t = i \mid x_t = j)$

use observe sequence **z**

$$
\begin{aligned}
P(\vec{x};A,B) &= \sum_{\vec{z}} P(\vec{x},\vec{z};A,B) \\
&= \sum_{\vec{z}} P(\vec{x}|\vec{z};A,B)P(\vec{z};A,B)
\end{aligned}
$$

use HMM assumptions

$$
\begin{aligned}
P(\vec{x};A,B) &= \sum_{\vec{z}} P(\vec{x}|\vec{z};A,B)P(\vec{z};A,B) \\
&= \sum_{\vec{z}} (\prod_{t=1}^{T} P(x_t|z_t;B)) (\prod_{t=1}^{T} P(z_t|z_{t-1};A)) \\
&= \sum_{\vec{z}} (\prod_{t=1}^{T} B_{z_t\, x_t}) (\prod_{t=1}^{T} A_{z_{t-1}\, z_t})
\end{aligned}
$$

This is relatively advanced. Even sklearn has outsourced it:

https://github.com/hmmlearn/hmmlearn