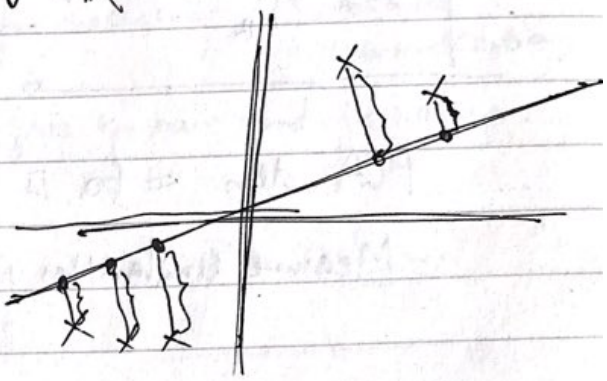
Have $x^{(i)} \in \mathbb{R}^n$. View representation of data in $\{u_1, \dots, u_k\}$ basis.

$y^{(i)} = (u_1^T x^{(i)}, u_2^T x^{(i)}, \dots, u_k^T x^{(i)})$. $y^{(i)} \in \mathbb{R}^k$

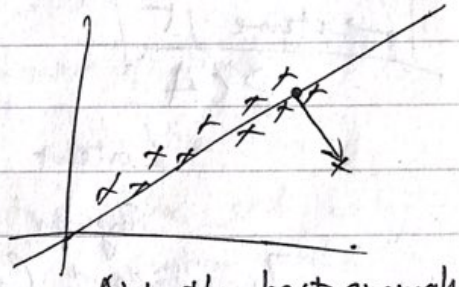


$\min \sum \{\text{distance}\}^2$

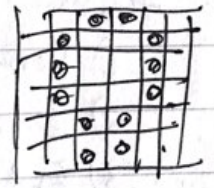
More interpretations:
See Problem Set.

Applications of PCA

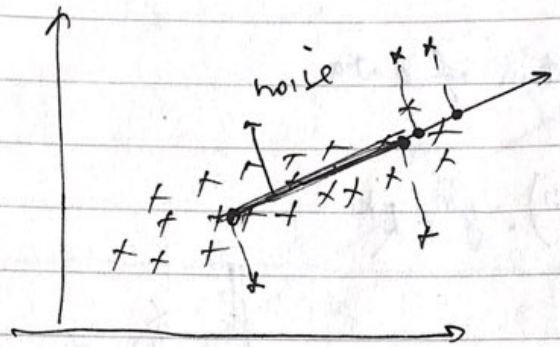
- Visualization
  Krishna - Shenoy's Lab.
- Compression
- Learning
  less prone to overfitting
  over used in industry.
- Anomaly detection
- Matching / distance calculations
  100 x 100 image
  (face detection)
  $x^{(i)} \in \mathbb{R}^{10000}$



Not the best anomaly detection algorithm, but sometimes it is enough.



"Sandy Alex Penland's Lab @MIT"



Face image lies in lower dim. space, because many dim's don't come from face (random noise).

PCA dim. = 50 is Typical.

Measure similarities in subspace.

$x^{(i)} \in \mathbb{R}^{10000}$ → 

$x^{(i)} \in \mathbb{R}^{10000}$ → 

$x^{(i)}$ is linear combinations of eigenfaces

* Often Recommend PCA. (Andrew Ng).
But over-used in industry (more often used):

* Before using PCA, think whether need compression exactly.

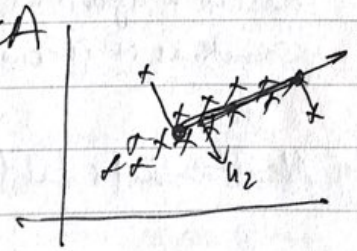
Lecture 15

PCA

- Latent Sematic Indexing (LSI)
- Singular Value Decomposition (SVD) implementation.

Independent Component Analysis (ICA)

PCA



Algorithm:

1) Normalize to zero mean, unit variance
2) $\Sigma = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} x^{(i)T}$
3) Find top k eigenvectors of Σ.

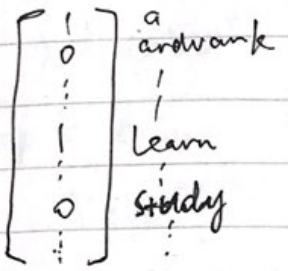
$x^{(i)} \in \mathbb{R}^{10000}$ → 50 PCs is eigenface

100×100

$\Sigma \in \mathbb{R}^{10000 \times 10000}$

?!

PCA: Text data

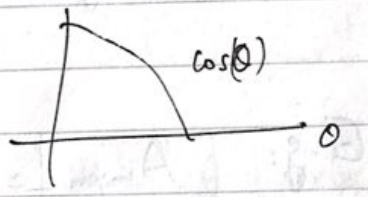


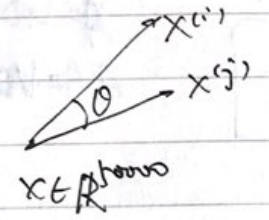
LSI
(Latent Sematic Indexing)
(Usually skip preprocessing: 1 variance normalization)

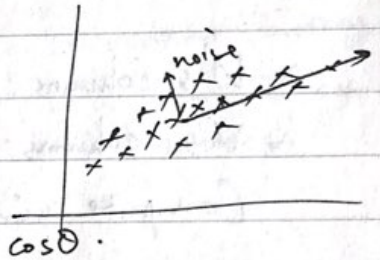
if do that, weights of rare word will scale up.

$x^{(i)}, x^{(j)}$ — measure similarity.

$$Sim(x^{(i)}, x^{(j)}) = \cos\theta = \frac{x^{(i)T} x^{(j)}}{\|x^{(i)}\| \cdot \|x^{(j)}\|}$$





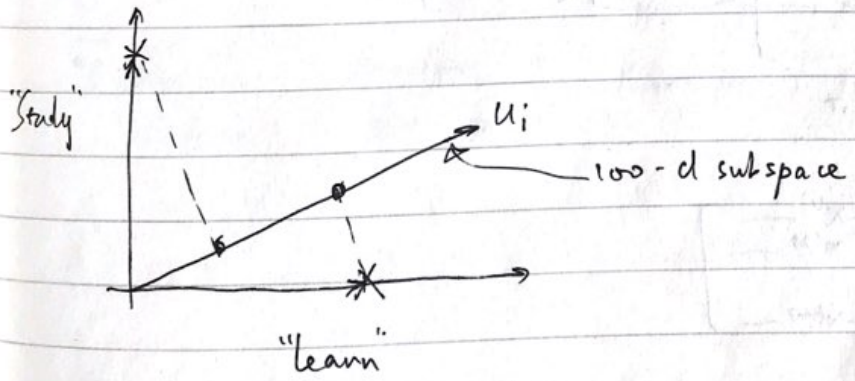


Eg: $x^{(i)} \rightarrow$ "study"
$x^{(j)} \rightarrow$ "learn"

Numerator of $Sim(x^{(i)}, x^{(j)})$

$$x^{(i)T} x^{(j)} = \sum_k x_k^{(i)} x_k^{(j)}$$

$$= \sum_k 1\{\text{Doc } i \text{ and } j \text{ both contain word } k\}.$$



$Sim(x^{(i)}, x^{(j)}) > 0$ after PCA.

Implementation PCA

$x^{(i)} \in \mathbb{R}^{50000}$ $\quad \Sigma \in \mathbb{R}^{50000 \times 50000}$

$A \in \mathbb{R}^{n \times n}$

$\underset{m \times n}{A} = \underset{m \times n}{U} \underset{n \times n}{D} \underset{n \times n}{V^T}$ (SVD)

$D = \text{diagonal} = \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \ddots \\ & & & \sigma_n \end{bmatrix}$ σ_i = sigular values of A.

E.g.: $[A_{m \times n}] = [U_{m \times n}] [D_{n \times n}] [V^T_{n \times n}]$

$A = UDV^T$

$A^TA = VD^TU^TUDV$

* U's columns: Eigenvectors of AA^T ✓
* V's columns: Eigenvectors of A^TA ✓

Compute using "svd" in Matlab/Octave (Very stable and efficient)

$$[A_{m \times n}] = [U_{m \times n}] [D_{n \times n}] [V^T_{n \times n}]$$

$$\Sigma = \sum_{i=1}^{n} x^{(i)} x^{(i)T}.$$

Design Matrix $X = \begin{bmatrix} — x^{(1)} — \\ — x^{(2)} — \\ \vdots \\ — x^{(m)} — \end{bmatrix}$

$$\Sigma = X^T X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix} \begin{bmatrix} — x^{(1)} — \\ — x^{(2)} — \\ \vdots \\ — x^{(m)} — \end{bmatrix}$$

To get top k eigenvectors

$X = UDV^T$

V? U?

Top k columns of V are top k eigenvectors of $X^TX = \Sigma$.

$X \in \mathbb{R}^{m \times 50000}$

$\Sigma \neq XX^T$

A note of caution:

$[X_{m \times n}] = [U_{m \times n}] [D] [V^T]$

$[U_{m \times n}] [D] [V^T]$

dim of matrix "conventions" may be different between diff. SVD commands.

	Model $P(x)$	Not probabilistic
"Subspace"	Factor Analysis	PCA
"Clumps" / "Groups"	Mixture of Gaussians	K-means

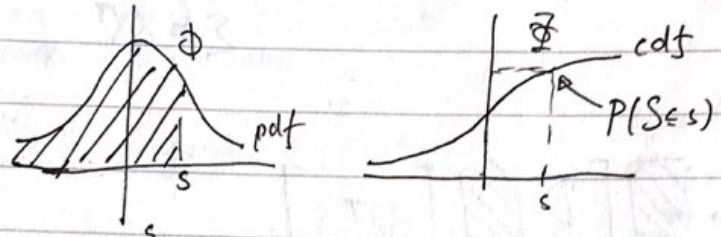
ICA

cdf (cumulative distribution function)

random variable S, density $P_S(s)$

cdf: $F(s) = P(S \leq s)$

E.g.:



$$F(s) = \int_{-\infty}^{s} P_S(t)\,dt$$

$S \rightarrow$ Specify $P_S(s)$ or Specify $F(s)$

$$P_S(s) = F'(s)$$

cdf: sigmoid

Any functions:
- continuous
- $f(-\infty) \approx 0$
- $f(+\infty) = 1$

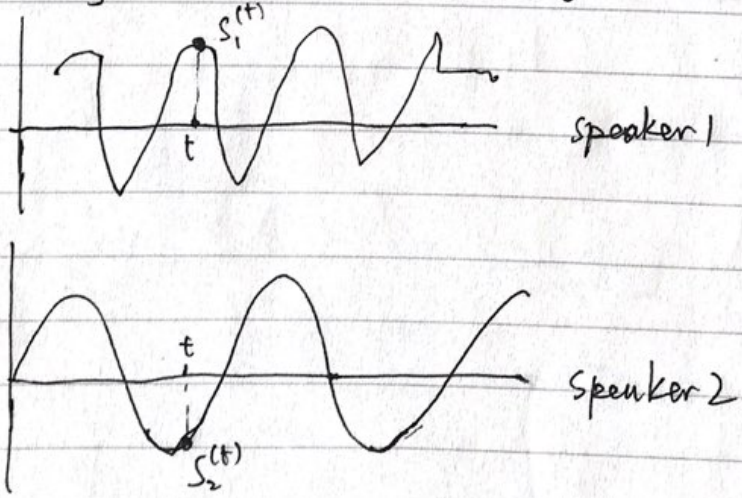
can do.

Cocktail Party

Original sources: n speakers

$S \in \mathbb{R}^n$

$S_j^{(i)}$ = signal from speaker j at time i.



We observe : n microphones

$x^{(i)} = A s^{(i)} \quad x^{(i)} \in \mathbb{R}^n$

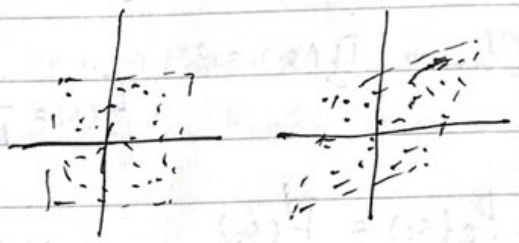
$x_j^{(i)} = \sum_k A_{jk} s_k^{(i)}$

$W = \begin{bmatrix} — & w_1^T & — \\ — & w_2^T & — \\ & \vdots & \\ — & w_n^T & — \end{bmatrix}$

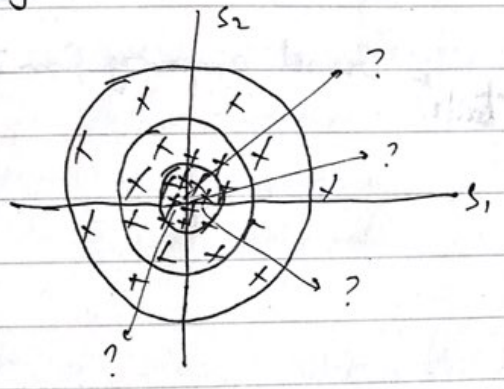
Goal: Find $W = A^{-1}$ so that $s^{(i)} = W x^{(i)}$

If $s_j^{(i)} \sim$ Uniform $[-1, 1]$

Ambiguities { Permutation of speaker / Sign of speaker signal



$s_j^{(i)}$ - non-Gaussian



$s^{(i)} \sim N(0, I)$

If sources is Gaussian, it's actually impossible to do ICA !!

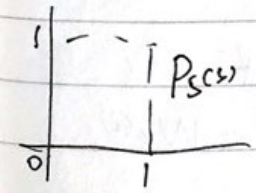
Let $s \in \mathbb{R}^n$.

Density of s: $p_s(s)$

$x = As = W^{-1}s, \quad s = Wx$

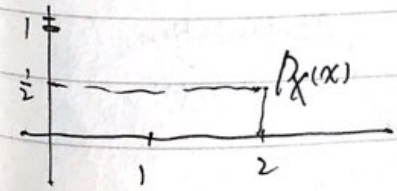
$p_x(x) = p_s(Wx) \cdot |W|$

E.g: $p_s(s) = 1\{0 \le s \le 1\}$ ($s \sim$ Uniform $[0, 1]$).



Let $x = 2s$, $A = 2$, $W = \frac{1}{2}$

$p_x(x) = 1\{0 \le x \le 2\} \cdot \frac{1}{2}$.



ICA model

$P(s) = \prod_{i=1}^{n} P_S(s_i)$ (s_i's independent)

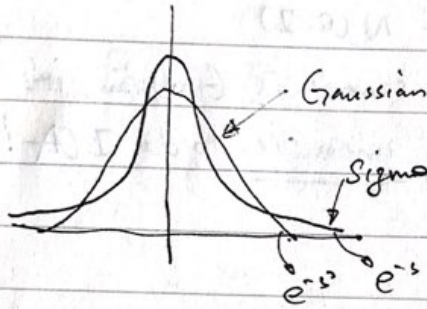
$P(x) = \left[\prod_{i=1}^{n} P_S(w_i^T x)\right] \cdot |W|$ $\quad W = A^{-1} = \begin{bmatrix} — w_1^T — \\ — w_2^T — \\ \vdots \\ — w_n^T — \end{bmatrix}$

$s_i = w_i^T x.$

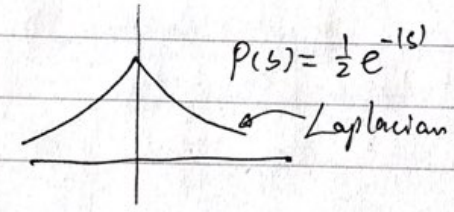
Choose $P_S(s_i) = ?$

$F(s) = \frac{1}{1-e^{-s}}$ ← convience, familiar.

$P_S(s_i) = F'(s_i)$



← Good property for ICA!



Given $\{x^{(1)} \cdots x^{(m)}\}$.

$\ell(W) = \sum_i \log\left[\left(\prod_j P_S(w_j^T x^{(i)})\right) \cdot |W|\right]$ $\quad P_S(s) = F'(s)$

Stochastic gradient ascent:

$W := W + \alpha \nabla_W \ell(W)$

$$\nabla_W \ell(W) = \begin{bmatrix} 1 - 2g(w_1^T x^{(i)}) \\ \vdots \\ 1 - 2g(w_n^T x^{(i)}) \end{bmatrix} x^{(i)T} + (W^T)^{-1}$$

Output

$s^{(i)} = W x^{(i)}$

Permutation

$\pm$ sign

Applications of ICA

① Katie (T.A. of Ng) Photo: electrons on his head.

EEG cap: pre-processing for research on brain.

② Aapo

Natural Images (~~Piece~~ Image Piece → Pattern Piece).

hypothesis on early visual processing in human brain

Lecture 16

Reinforcement Learning ← "Between supervised Learning & unsupervised Learning"

- Markov Decision Process (MDP)
- Value Function
- Value iteration
- Policy iteration

"Much less supervision"

"Just like training a dog"

Not one-time decision, but <u>sequential decision process</u>

<u>Credit Assignment Problem</u>: which step cause to bad result?

MDP: Markov Decision Process

$MDP(S, A, \{P_{sa}\}, \gamma, R)$ ... five-tuple ...

① S - set of state (position & orientation)

② A - set of actions (position of control stake)

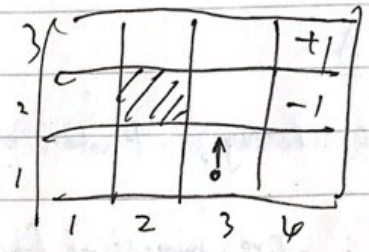
③ P_{sa} - state transition distributions

$$\sum_{s'} P_{sa}(s') = 1. \quad P_{sa} \geq 0.$$

④ γ - discount factor $0 \leq \gamma \leq 1$

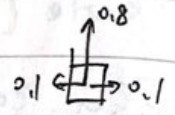
⑤ R - reward function $R: S \mapsto \mathbb{R}$

eg: from AI: A Modern Approach
Russell & Norvig)



① S: 11 states

② A = {N, S, E, W}



③ $P_{(3,1)N}((3,2)) = 0.8$
$P_{(3,1)N}((4,1)) = 0.1$
$P_{(3,1)N}((2,1)) = 0.1$
$P_{(3,1)N}((3,1)) = 0$
⋮ etc.

At state s_0
Choose a_0
Get to $s_1 \sim P_{s_0, a_0}$
Choose a_1
Get to $s_2 \sim P_{s_1, a_1}$
⋮

④ $R((4,3)) = +1$
$R((4,2)) = -1$
$R(s) = -0.02$ for all other states: "Battery charges and time consume"

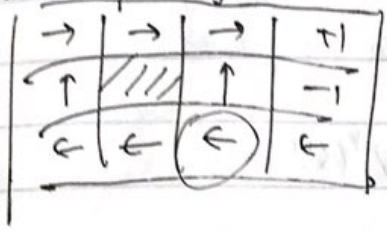
④⑤ Total Payoff:
$R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots$
$0 \le \gamma < 1$ (often 0.99) In economic app. represent interest of bank etc.

Win/lost immediately greater than future.

* Choose actions over time $(a_0, a_1, \ldots)$ to maximize
$E[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots]$
* Policy $\pi: S \to A$

Policy: Simple, only depend on current time
Strategy: Multiple policies, include previous states

Optimal policy



MDP good at:

Subtle tradeoff between different impact factors

(Conservative policy $\overset{\nu}{\longleftrightarrow}$ Active policy)

"low risk" "shorter distance"

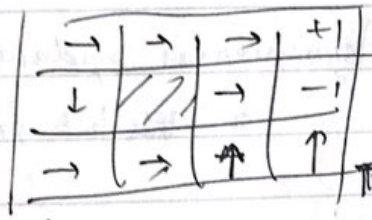
Define V^π, V^*, π^*

For any π, define value function $V^\pi : S \longmapsto \mathbb{R}$

s.t. $V^\pi(s)$ is expected total payoff starting in state s, and executed π.

$$V^\pi(s) = E[R(s_0) + \nu R(s_1) + \cdots \mid \pi, s = s_0].$$

eg:



π

.52	.73	.77	+1
.90		.82	−1
.88	.81	.85	1.00

V^π

$s_0 \hookrightarrow s$, $s_1 \hookrightarrow s'$

$$V^\pi(s) = R(s) + \nu \sum_{s'} P_{s\pi(s)}(s')\, V^\pi(s')$$

Bellman's Equations

$\pi((3,1)) = N$

$V^\pi(s) = ?$

$V^\pi((3,1)) = R((3,1)) +$

$\nu[0.8\, V^\pi((3,2)) + 0.1\, V^\pi((4,1)) + 0.1\, V^\pi((2,1))]$.

Optimal value function:

$$V^*(s) = \max_\pi V^\pi(s)$$

Bellman's eqns:

$$\begin{cases} V^*(s) = R(s) + \max_a \gamma \sum_{s'} P_{sa}(s') V^*(s') \\ \pi^*(s) = \arg\max_a \sum_{s'} P_{sa}(s') V^*(s') \end{cases}$$

Value iteration ("individual" "liberalism")

Initialize $V(s) = 0$ for $\forall s$.

For every s, update

$$V(s) := R(s) + \max_a \gamma \sum_{s'} P_{sa}(s') V(s')$$

This will make $V(s) \to V^*(s)$

Synchronous update

$V := B(V)$

↑ Bellman backup operator

Asynchronous update (a little bit faster)

.86	.90	.93	+1
.82	//	.69	-1
.78	.75	.71	.49

V^*

→	→	→	+1
↑	//	↑	-1
←	←	←	←

π^*

$\pi((3,1)) = W$: $\sum_{s'} P_{sa}(s') V^*(s')$

$= .8 \times .75 + .1 \times .69 + .1 \times .71 = .74$

↑ Stay when hit wall

$\pi((3,1)) = N$: $\sum_{s'} P_{sa}(s') V^*(s')$

$= .8 \times .69 + .1 \times .75 + .1 \times .69 = .676$

Policy iteration ("Centralised" / "Collective")

Initialize π randomly.

Repeat:

Let $V := V^\pi$ (solve Bellman's eqns) — Computationally expensive for large MDP (eg: 10 million). Better choose value iter.

Let $\pi(s) := \arg\max_a \sum_{s'} P_{sa}(s') V(s')$

$V \to V^*$ $\pi \to \pi^*$ (At last)

Also Convex optimization

Proof not hard

What if don't know P_{sa}?

estimate P_{sa} from data

S	A	$\{P_{sa}\}$	γ	R
✓	✓	?	✓	✓

$$P_{sa}(s') = \frac{\#\text{times took action "a" in } s, \text{ got to } s'}{\#\text{times took action "a" in } s}$$

(or $\frac{1}{|S|}$ if "$\frac{0}{0}$") uniform distribution when no experiments

More mix & match algo's in R.L.

Putting it together

Repeat: {

Take actions using π to get experience in MDP

Update estimates of P_{sa}

Solve Bellman's eqns (V^*) using V.I. to get V^*

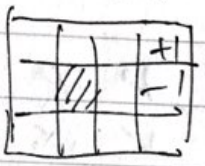
Update $\pi(s) = \arg\max_a \sum_{s'} P_{sa}(s') V^*(s')$

}

Lecture 17

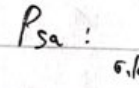
- Continous State MDPs
- Discretization
- Models / Simulators
- Fitted value iteration
- Q function
- Approximate policy iteration

MDP $(S, A, \{P_{sa}\}, \gamma, R)$



S: 11 states

$A = \{N, S, E, W\}$

P_{sa}: 

$R = \{\pm 1$ wt. absorbing states
-0.02 elsewhere

$\gamma = 0.99$

$\pi : S \mapsto A$

$V^{\pi}(s) = E[R(s_0) + \gamma R(s_1) + \dots \mid \pi, s = s_0]$

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

$\pi^*(s) = \arg\max_{\pi} \sum_{s'} P_{sa}(s') V^*(s')$

$V^*(s) = R(s) + \gamma \max_a \sum_{s'} P_{sa}(s') V^*(s')$

V.I.: $V(s) := R(s) + \gamma \max_a \sum_{s'} P_{sa}(s') V(s')$

P.I.: Repeat { Solve for V^{π}. Let $V := V^{\pi}$

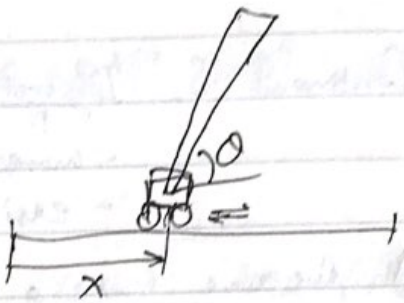
Update $\pi(s) = \arg\max_a \sum_{s'} P_{sa}(s') V(s')$

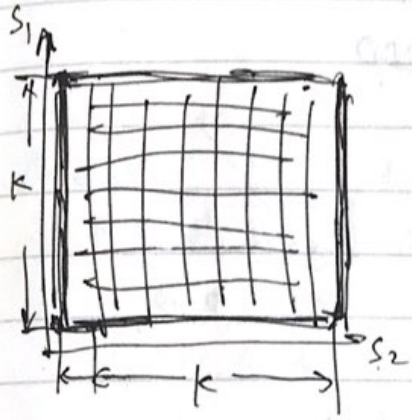
E.g.: Car : $x, y, \theta, \dot{x}, \dot{y}, \dot{\theta}$

Helicopter: $x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}$

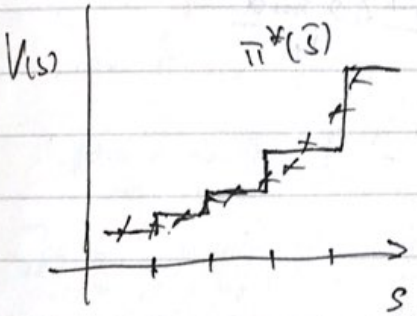
(roll, peck, yaw)

Inverted pendulum: $x, \theta, \dot{x}, \dot{\theta}$





Discretize into discrete states $\bar{s}$
Solve for $V^*(\bar{s})$, $\pi^*(\bar{s})$
execute $\pi^*(\bar{s})$

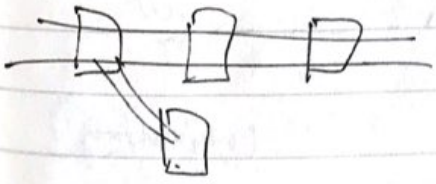


Not really good.
Not smooth between buckets
Not generalize between buckets

Curse of dimensionality.

If $S \in \mathbb{R}^n$ $n \leq 3$
Get k^n states. "folk trick"

Factory Automations.



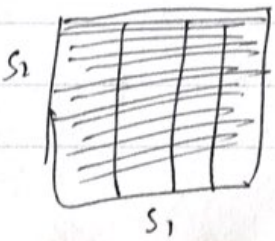
Scheduling orders

N machines * k states/machine
$= k^N$ states.

Board Games (eg. play chess)

N pieces * k position/pieces
$= k^N$ states (positions).

discretization scale poorly for high-dimensional spaces



sensitive difference for diff. dim.
manually HARD

How: Approximate V^*!

Continous S, Discrete A

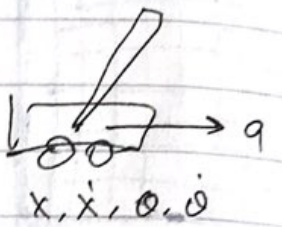
↑
- more rational
- easier

car: 6 states vs. 2 actions
helicopter: 12 states vs 4 actions
Inv.P.: 4 states vs 1 action

① Assume have a model/simulator of MDP.

$S \rightarrow$ [Model] $\rightarrow s' \sim P_{sa}$.

↑ a



Eg: physics simulator

$$s = \begin{pmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{pmatrix}$$

$$\dot{s} = \begin{pmatrix} \dot{x} \\ \alpha - L\beta\cos\theta/M \\ \dot{\theta} \\ \beta \end{pmatrix}, \text{ where } \begin{cases} \alpha = \dfrac{a + L\dot{\theta}^2 \sin\theta}{M} \\ \beta = \ldots \\ \text{etc.} \end{cases}$$

$$s_{t+1} = s_t + f(\Delta t) \begin{bmatrix} \dot{x} \\ \alpha - L\beta\cos\theta/M \\ \dot{\theta} \\ \beta \end{bmatrix}.$$

$\Delta t = 0.1$ second

② Learn a model

experiments: $s_0^{(i)} \xrightarrow[\text{Choose}]{a_0^{(i)}} s_1^{(i)} \xrightarrow[\text{Choose}]{a_1^{(i)}} s_2^{(i)} \rightarrow \cdots \rightarrow s_T^{(i)}$

("trajectory")

(Repeat i times).

Use learning algo. to estimate s_{T+1} as function of s_t, a_t.

[Inv. P: $s_t \in \mathbb{R}^4$, $a_t \in \mathbb{R}^4$. run L.R. 4 times]

E.g: $S_{t+1} = A S_t + B a_t$ (linear), $A \in \mathbb{R}^{4\times 4}$ $B = \mathbb{R}^4$

$$\arg\min_{A,B} \sum_{i=1}^{m} \sum_{t=0}^{T-1} \left\| S_{t+1}^{(i)} - (A S_t^{(i)} + B a_t^{(i)}) \right\|^2$$

LR ... LWLR ... kernel L.R.

(effective method in practice)

Model: $S_{t+1} = A S_t + B a_t$ (deterministic)

or $S_{t+1} = A S_t + B a_t + \varepsilon_t$, $\varepsilon_t \sim N(0, \Sigma)$ (stochastic)

↑ sample from Gaussian distribution

$S_t \rightarrow$ [Model] $\rightarrow S_{t+1}$.

↑ a_t

Inv. P: LWLR. Nonlinear Algo.

Approximate V^*!

Just like LR:

Chose feature $\phi(s)$ for state s

Could $\phi(s) = s$

Approximate: $V(s) = \theta^T \phi(s)$ (θ: parameters)

"$h_\theta(x) = \theta^T \phi(s)$"

E.g: Inv. P:

$$\phi(s) = \begin{bmatrix} 1 \\ x \\ \dot{x}^2 \\ \dot{x} \\ \theta \cdot x \\ \dot{\theta}^2 \\ \vdots \end{bmatrix}$$

V. I:

$$V(s) := R(s) + \gamma \max_a \sum_{s'} P_{sa}(s') V(s')$$

↖ Replace by integral

$$= R(s) + \gamma \max_a E_{s' \sim P_{sa}}[V(s')].$$

Fitted value iteration

Sample $\{s^{(1)}, s^{(2)}, \ldots, s^{(m)}\} \subseteq S$ randomly.

Initialize $\theta := 0$ (Just like L.R.)

Repeat {

For $i = 1 \ldots m$ {

For each action $a \in A$ {

Sample $s'_1 \ldots s'_k \sim P_{s^{(i)}a}$ (using model)

Let $q(a) = \frac{1}{k} \sum_{j=1}^{k} [R(s^{(i)}) + \gamma V(s'_j)]$

// estimate for $R(s^{(i)}) + \gamma E_{s' \sim P_{s^{(i)}a}}[V(s')]$

}

Set $y^{(i)} = \max_a q(a)$

// estimate for $R(s^{(i)}) + \gamma \max_a E_{s' \sim P_{s^{(i)}a}}[V(s')]$

}

// Want $V(s^{(i)}) \approx y^{(i)}$

$\theta := \arg\min_\theta \frac{1}{2} \sum_{i=1}^{m} (\theta^T \phi(s^{(i)}) - y^{(i)})^2$

}

helicopter control: 10Hz ~ 100Hz

~100 samples

Runs quite well for 6~20 dim. if choose features properly.

$MDP = \{S, A, \{P_{sa}\}, \gamma, R\}$

Inv.P.: R^n R model γ: 0.99

$R(s) = \begin{cases} -1 & \text{fallen } |\theta| > 30^\circ \\ 0 & \text{otherwise} \end{cases}$

All above,

Assume: we have a stochastic simulator

If we have a deterministic simulator:

Set $k = 1$. don't need sample (simplify F.V.I.)

Find V^* → Compute → π^*

for L.R.: easy to choose feature.

For approx. value func.: harder/tricker.

discrete state s:

$$\pi^*(s) = \arg\max_a E_{s' \sim P_{sa}}[V^*(s')] = \arg\max_a \sum_{s'} P_{sa}(s') V^*(s')$$

continous state:

~~At~~ specific state $s = (x, \theta, \dot{x}, \dot{\theta})$

$$a = \arg\max_a E_{s' \sim P_{sa}}[V^*(s')]$$

next iter: s ← Replace

If deterministic simulator:

$s_{t+1} = f(s_t, a_t)$

$s' = f(s, a)$

$$\arg\max_a V^*(f(s,a))$$

If stochastic simulator:

$$s_{t+1} = f(s_t, a_t) + \varepsilon_t$$

ε_t: Gaussian noise

E.g.: $s_{t+1} = As_t + Ba_t + \varepsilon_t$

$$E_{s' \sim P_{sa}}[V^*(s')] \approx V^*(E[s'])$$

$$= V^*(f(s,a))$$

$$\hookrightarrow \theta^T \phi(s')$$

$s' = f(s,a)$

θ from F.V.I.

Choose action:

$\arg\max_a V^*(f(s,a))$ ignoring the noise (compute quickly)

effective in practice (mean 0 Gaussian noise in model is proper)

Lecture 18

State-action rewards

Finite horizon MDPs

Linear Dynamical Systems

- Models
- Linear Quadratic Regulation (LQR) control
- Riccati Equations

MDP: $(S, A, \{P_{sa}\}, \gamma, R)$ $\quad 0 \le \gamma < 1$

$R: S \mapsto \mathbb{R}$.

$$V(s) := R(s) + \max_a \gamma \sum_{s'} P_{sa}(s') V(s')$$

$$\pi^*(s) = \arg\max_a \gamma \sum_{s'} P_{sa}(s') V(s')$$

State-action Rewards

$R: S \times A \mapsto \mathbb{R}$.

$s_0, a_0 \to s_1, a_1 \to s_2, a_2 \to \cdots$

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

$$V^*(s) = \max_a R(s,a) + \gamma \sum_{s'} P_{sa}(s') V^*(s')$$

V.I: $V(s) := \max_a R(s,a) + \gamma \sum_{s'} P_{sa}(s') V^*(s')$

$$\pi^*(s) = \arg\max_a R(s,a) + \gamma \sum_{s'} P_{sa}(s') V^*(s')$$

action "stay"

driving over rail/road, grass/rock.

Finite horizon MDPs

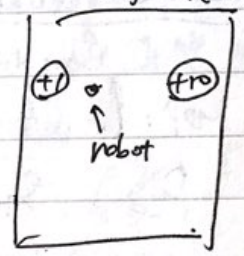
$(S, A, \{P_{sa}^{(t)}\}, T, R)$ T: horizon time.

Payoff: $R^{(0)}(s_0, a_0) + R^{(1)}(s_1, a_1) + \cdots + R^{(T)}(s_T, a_T)$

Optimal policy: non-stationary (depend on time)

$s_{t+1} \sim P_{s_t a_t}^{(t)}$

Spacecraft: fuel (time)
traffic: weather (time)
factory: workers (time).
...



depend on the remained

$$V_t^*(s) = E\left[R^{(t)}(s_t, a_t) + \cdots + R^{(T)}(s_T, a_T) \mid \pi^*, s_t = s\right]$$

$$V_T^*(s) = \max_a R^{(T)}(s, a)$$

$$V_t^*(s) = \max_a R^{(t)}(s, a) + \sum_{s'} P_{sa}^{(t)}(s') V_{t+1}^*(s')$$

$$\pi_t^*(s) = \arg\max_a R^{(t)}(s, a) + \sum_{s'} P_{sa}^{(t)}(s') V_{t+1}^*(s')$$

Dynamic Programming (Recursion)

$\{V_T^*, V_{T-1}^*, V_{T-2}^*, \ldots, V_0^*$
$\pi_T^*, \ldots, \pi_0^*$

directly compute, not iter. to converge.

$\gamma^{(T)} \approx 0.0000 1 \rightarrow T$.

Commonly use discounting OR finite horizon.

① Do discounting can ensure that the value function is finite.

② Use finite horizon is sufficient to ensure the value function is finite.

Both use state-action rewards and finite horizon MDPs (strong assumption)

But it is reasonable for many systems.

Then we have a very elegant algo. to solve even very large MDPs.

Linear Quadratic Regulation (LQR)

Continous state space, maybe continous action.

MDP: $(S, A, \{P_{sa}\}, T, R)$ $S \in \mathbb{R}^n$ $A \in \mathbb{R}^d$

P_{sa}: $s_{t+1} = A_t s_t + B_t a_t + w_t$ $w_t \sim N(0, \Sigma_w)$

↳ not very important

$A_t \in \mathbb{R}^{n \times n}$ $B_t \in \mathbb{R}^{n \times d}$

Assume the dynamic is linear.

$R^{(t)}(s_t, a_t) = -(s_t^T U_t s_t + a_t^T V_t a_t)$

$U_t \in \mathbb{R}^{n \times n}$ $V_t \in \mathbb{R}^{d \times d}$

$U_t \geq 0$ $V_t \geq 0$ (positive semi-definite, psd)

$s_t^T U_t s_t \geq 0$, $a_t^T V_t a_t \geq 0$.

$\Rightarrow R^{(t)}(s_t, a_t) \leq 0$.

Helicopter: Want $s_t \approx 0$.

Choose $U_t = I$. $V_t = I$.

$R(s_t, a_t) = -s_t^T s_t - a_t^T a_t$

$= -\|s_t\|^2 - \|a_t\|^2$

$A = A_1 = A_2 = A_3 = \cdots$ pretend MDP is

$B = B_1 = B_2 = B_3 = \cdots$ stationary for now.

$$s_{t+1} = A s_t + B a_t$$

$s_0^{(i)} \xrightarrow{a_0^{(i)}} s_1^{(i)} \xrightarrow{a_1^{(i)}} s_2^{(i)} \xrightarrow{a_2^{(i)}} \cdots \cdots s_T^{(i)} \quad (i = 1, \ldots, m)$

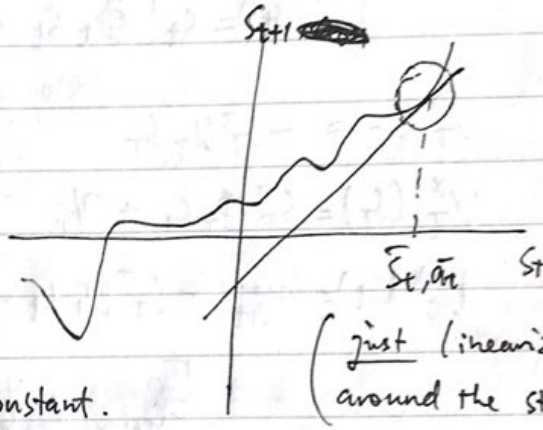
$$\arg\min_{A,B} \frac{1}{2} \sum_{i=1}^{m} \| s_{t+1} - (A s_t + B a_t) \|^2 .$$

Download packets to similate IndP. on Intern

Linearize non-linear model.

$s_{t+1} = f(s_t, a_t)$.

$$\begin{bmatrix} x_{t+1} \\ \dot{x}_{t+1} \\ \theta_{t+1} \\ \dot{\theta}_{t+1} \end{bmatrix} = f\left(\begin{bmatrix} x_t \\ \dot{x}_t \\ \theta_t \\ \dot{\theta}_t \end{bmatrix}, a \right)$$



$s_{t+1} = f(s_t)$ pretend no action.

$s_{t+1} \approx f(\bar{s}_t)(s_t - \bar{s}_t) + f(\bar{s}_t)$

$\bar{s}_t$ = constant.

(just (linearize around the state)

$s_{t+1} = A s_t + B a_t$

$s_{t+1} = A s_t + B a_t + \underline{c}$ $\quad s = \begin{pmatrix} 1 \\ x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{pmatrix}$

$s = \vec{0}$ (expected state near 0)

linearized near 0 state

$$s_{t+1} \approx f(\bar{s}_t, \bar{a}_t) + (\nabla_s f(\bar{s}_t, \bar{a}_t))^T (s_t - \bar{s}_t) + (\nabla_a f(\bar{s}_t, \bar{a}_t))^T (a_t - \bar{a}_t)$$

$\bar{s}_t, \bar{a}_t$ constant.

Goal: $\max \; R^{(0)}(s_0, a_0) + R^{(1)}(s_1, a_1) + \cdots + R^{(T)}(s_T, a_T)$

$$V_T^*(s_T) = \max_{a_T} R^{(T)}(s_T, a_T)$$

$$= \max_{a_T} - s_T^T U_T s_T - a_T^T V_T a_T$$

$$= \max_{a^T} - s_T^T U_T s_T \qquad (\text{because } a_T^T V_T a_T \geq 0)$$

$a_T = 0$?

$$\pi_T^*(s_T) = \arg\max_{a_T} R^{(T)}(s_T, a_T) = 0.$$

$V^*_{t+1} \rightsquigarrow V^*_t$. D.P.

$$V_t^*(s_t) = \max_{a_t} R^{(t)}(s_t, a_t) + \sum_{s_{t+1}} P_{s_t a_t}(s_{t+1}) V^*_{t+1}(s_{t+1})$$

$$= \max_{a_t} R^{(t)}(s_t, a_t) + E_{s_{t+1} \sim P_{s_t, a_t}}\left[V^*_{t+1}(s_{t+1})\right]$$

Suppose $\boxed{V^*_{t+1}(s_{t+1}) = s_{t+1}^T \Phi_{t+1} s_{t+1} + \psi_{t+1}}$ | $\Phi_{t+1} \in \mathbb{R}^{n \times n}$, $\psi_{t+1} \in \mathbb{R}$.

Can show $V_t^*(s_t) = s_t^T \Phi_t s_t + \psi_t$ for some Φ_t, ψ_t.

$V_T^*(s_T) = -s_T^T U_T s_T$. so $\Phi_T = -U_T$ $\psi_T = 0$.

$V_T^*(s_T) = s_T^T \Phi_T s_T + \psi_T$ (Recursion ...)

$$V_t^*(s_t) = \max_{a_t} -s_t^T U_t s_t - a_t^T V_t a_t.$$

$$+ E_{s_{t+1} \sim N(\underbrace{A_t s_t + B_t a_t}_{P_{s_t a_t}}, \Sigma_w)}\left[\underbrace{s_{t+1}^T \Phi_{t+1} s_{t+1} + \psi_{t+1}}_{V^*_{t+1}(s_{t+1})}\right]$$

Simplifies to quadratic function of a_t

$$\Rightarrow a_t = \underbrace{(B_t^T \Phi_{t+1} B_t - V_t)^{-1} B_t^T \Phi_{t+1} A_t}_{L_t} \cdot s_t$$

$$\pi_t^*(s_t) = \arg\max_{a_t} R^{(t)}(s_t, a_t) + E_{s_{t+1} \sim P_{s_t a_t}}\left[V^*_{t+1}(s_{t+1})\right]$$

$$= L_t \cdot s_t \qquad \text{(NOT approx.)}$$

→ plug back into D.P. recursion.

$$V_t^*(s_t) = s_t^T \Phi_t s_t + \psi_t$$

where $\Phi_t = A_t^T(\Phi_{t+1} - \Phi_{t+1} B_t (B_t^T \Phi_{t+1} B_t - V_t)^{-1} B_t^T \Phi_{t+1}) A_t - U_t$

$\psi_t = -\mathrm{tr}\,\Sigma_w \Phi_{t+1} + \psi_{t+1}$

↑ Discrete time Riccati equation