Lecture II - Intro to Neural Networks

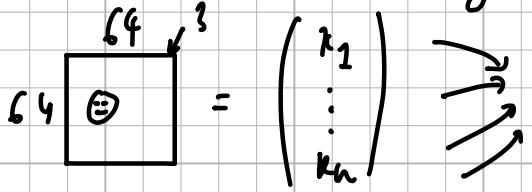
Ⓘ Logistic Regression

Ⓘ Neural Networks

Deep Learning
- Computational power
- Data available
- Algorithms

Ⓘ Logistic Regression

goal 1.0 Find cats in images { 1 → presence of cat; 0 → absence of cat



$\rightarrow \hat{y} = \sigma(\theta^T x) = \sigma(wx+b)$

$(1,1)$ $(1, 12288)$ $(64\times64\times3, 1)$

i) initialize w, b

ii) find the optimal w, b → $\mathcal{L} = -[y \log \hat{y} + (1-y)\log(1-\hat{y})]$

iii) Use $\hat{y} = \sigma(wx+b)$ on product

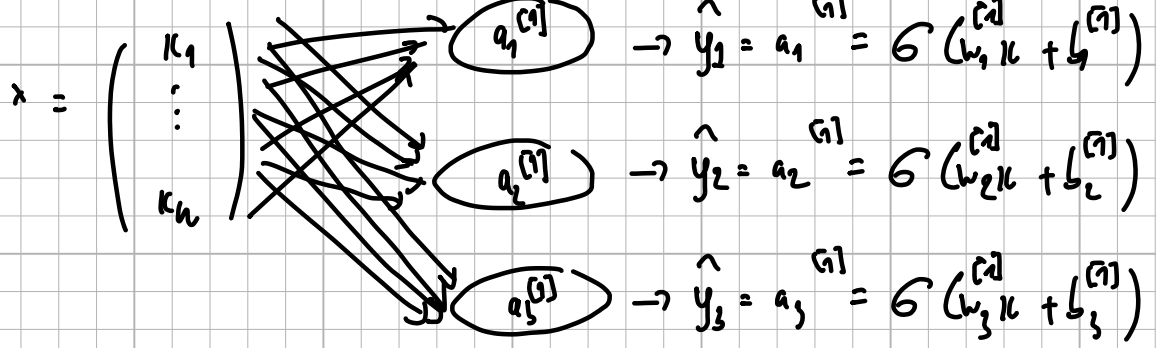
$$\begin{cases} w := w - \alpha \frac{\partial \mathcal{L}}{\partial w} \\ b := b - \alpha \frac{\partial \mathcal{L}}{\partial b} \end{cases}$$

(Eq1) neuron: linear ($a^{[1]}$) + activation ($z^{[1]}$)

(Eq2) model = architecture + parameter

$n = 64\times64\times3+1$ params

goal 2.0 find cat | iguana | lion in images



3n params

$\hat{y}$ $\Rightarrow$ $\begin{pmatrix}1\\1\\0\end{pmatrix}$ p(cat) p(lion) p(iguana)

$\Rightarrow$ robust, 3 neurons don't talk to each

$\mathcal{L}_{sw} = -\sum_{k=1}^{3}\left[y_k \log \hat{y}_k + (1-y_k)\log(1-\hat{y}_k)\right]$

goal 3.0 + constraint

unique animal on an image

$\hat{y} = \begin{pmatrix} x_1 \\ \vdots \\ x_{nx} \end{pmatrix}$ $z_1^{[1]} = \ldots$ $z_2^{[1]} = \ldots$ $z_3^{[1]} = \ldots$

$a_i^{[1]} = e^{z_i^{[1]}} \Big/ \sum_{k=1}^{3} e^{z_k^{[1]}}$

$\rightarrow$ Softmax multi-class network

3n paras $\begin{pmatrix}1\\0\\0\end{pmatrix}\begin{pmatrix}0\\1\\0\end{pmatrix}\begin{pmatrix}0\\0\\1\end{pmatrix}$

Cross-entropy loss

$\mathcal{L}_{ce} = -\sum_{k=1}^{3} y_k \log \hat{y}_k$

? age of cat $\rightarrow$ linear Regression

ReLU - rectified linear units

II Neural Networks

goal image $\Rightarrow$ cat vs no cat (1) (0)

denotes: - neuron are not connected to each other
- cluster of neurons

$\hat{y} = \begin{pmatrix} x_1 \\ \vdots \\ x_{nx} \end{pmatrix}$ Layer 1 Layer 2 $\rightarrow \hat{y}$

$\rightarrow$ fully connected layer $\rightarrow$ paras

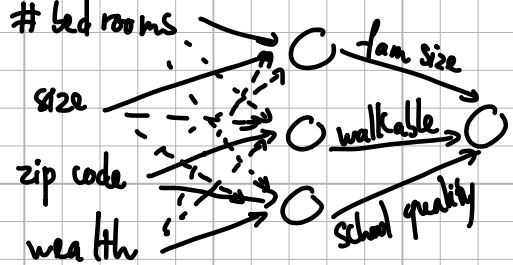
3n+3 | 2·3+2 | 1·2+1

input layer = 1st

hidden layer

output layer = last

House price prediction



end to end learning / black box model

Propagation equations

$(3,1) \rightarrow z^{[1]} = \underset{(3,m)}{W^{[1]}} \underset{(n_0,1)}{a^{[0]}} + \underset{(3,1)}{b^{[1]}}$

$(3,1) \rightarrow a^{[1]} = \sigma(z^{[1]})$

$z^{[2]} = W^{[2]} a^{[1]} + b^{[2]}$

$a^{[2]} = \sigma(z^{[2]})$

$z^{[3]} = W^{[3]} a^{[2]} + b^{[3]}$

$\hat{y} = a^{[3]} = \sigma(z^{[3]})$ $(1,1)$

* What happens for an input batch of m examples?

$$X = \begin{pmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{pmatrix} \quad (n_0, m)$$

Broadcasting (numpy)

$$Z^{[1]} = W^{[1]} X + b^{[1]}$$

$(3,m) = (3,n_0)(n_0,m) + (3,1)$

$$= \begin{bmatrix} | & & | \\ z^{[1](1)} & \cdots & z^{[1](m)} \\ | & & | \end{bmatrix}$$

$b^{[1]} = \begin{pmatrix} | & & | \\ b^{[1]} & \cdots & b^{[1]} \\ | & & | \end{pmatrix}$ (m columns)

Optimizing $W^{[1]} \dots W^{[3]}, b^{[1]} \dots b^{[3]}$

Define loss / cost function

loss: 1 ex. — cost: mean

$$J(\hat{y}, y) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}^{(i)}$$

with $\mathcal{L}^{(i)} = -\left[y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log (1-\hat{y}^{(i)}) \right]$

Backward Propagation

$\forall l = 1 \dots 3$

$$\begin{cases} W^{[l]} = W^{[l]} - \alpha \dfrac{\partial J}{\partial W^{[l]}} \\ b^{[l]} = b^{[l]} - \alpha \dfrac{\partial J}{\partial b^{[l]}} \end{cases}$$

→ Chain rule

$$\frac{\partial J}{\partial W^{[3]}} = \frac{\partial J}{\partial a^{[3]}} \cdot \frac{\partial a^{[3]}}{\partial z^{[3]}} \cdot \frac{\partial z^{[3]}}{\partial W^{[3]}}$$

$$\frac{\partial J}{\partial W^{[2]}} = \frac{\partial J}{\partial z^{[3]}} \cdot \frac{\partial z^{[3]}}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial W^{[2]}}$$

$$\frac{\partial J}{\partial W^{[1]}} = \frac{\partial J}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial a^{[1]}} \cdot \frac{\partial a^{[1]}}{\partial z^{[1]}} \cdot \frac{\partial z^{[1]}}{\partial W^{[1]}}$$

Lecture 12 - Backprop & Improving Neural Networks

I Log Reg with a NN mindset

II Neural Networks

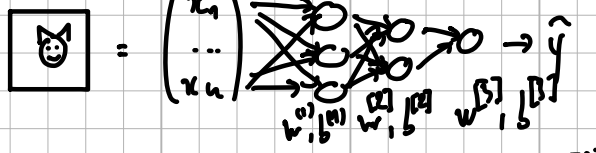
→ Back Propagation

III Improving your NNs

Backpropagation

Cost function: $J(\hat{y}, y) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}^{(i)}(\hat{y}, y)$

with $\mathcal{L}^{(i)} = -\left[y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log (1-\hat{y}^{(i)})\right]$



Update $w^{[l]} = w^{[l]} - \alpha \frac{\partial J}{\partial w^{[l]}}$

$$\frac{\partial \mathcal{L}}{\partial w^{[3]}} = (a^{[3]} - y^{(i)})\, a^{[2]T} = -(y^{(i)} - a^{[3]})\, a^{[2]T} \quad \text{via } \frac{\partial z^{[3]}}{\partial w^{[3]}}$$

$$\frac{\partial J}{\partial w^{[3]}} = -\frac{1}{m} \sum_{i=1}^{m} (y^{(i)} - a^{[3]})\, a^{[2]T}$$

$$\frac{\partial \mathcal{L}}{\partial w^{[2]}} = (a^{[3]} - y) \cdot w^{[3]T} a^{[2]} (1 - a^{[2]})\, a^{[1]T}$$

$= w^{[3]T} * a^{[2]}(1-a^{[2]})(a^{[2]} - y) \cdot a^{[1]T}$

$(1,2)^T$ | $2,1$ | $1,1$ | $1,3$

$(2,5)$ ↓ 5 neurons connect to 2

↳ $\frac{\partial J}{\partial w^{[2]}} = \frac{1}{m} \sum_{i=1}^{m} \frac{\partial \mathcal{L}}{\partial w^{[2]}}$

Recall

$\sigma(x) = 1 - \sigma(-x)$

$\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x + 1}$

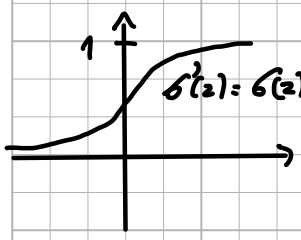
$\sigma'(x) = \sigma(x)(1 - \sigma(x))$

Cache → Computation efficiency

↳ Memory cost

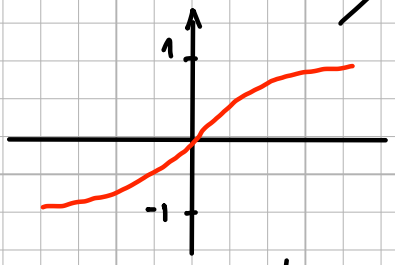
III Improving your NNs

A Activation functions



Sigmoid $\sigma(z) = \frac{1}{1+e^{-z}}$

→ Saturation



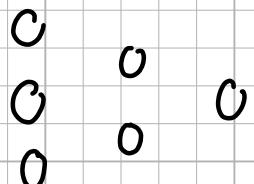
$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$

$\tanh'(z) = 1 - \tanh(z)^2$

$\mathrm{ReLU}(z) = \begin{cases} 0 & \text{if } z \leq 0 \\ 1 & \text{if } z > 0 \end{cases}$

$\mathrm{ReLU}'(z) = 1_{\{z>0\}}$

Why do we need activation functions?



activations: $|c| \quad z \to z$

$\hat{y} = a^{[3]} = z^{[3]} = w^{[3]} a^{[2]} + b^{[3]}$

$= w^{[3]} w^{[2]} (w^{[1]} x + b^{[1]}) + w^{[3]} b^{[2]} + b^{[3]}$

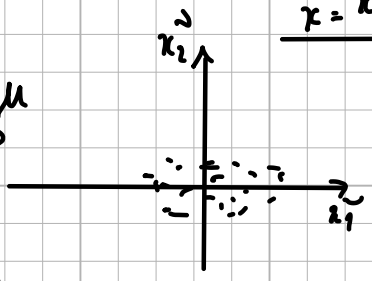
$= Wx + b$

B Initialization methods

Normalizing your input

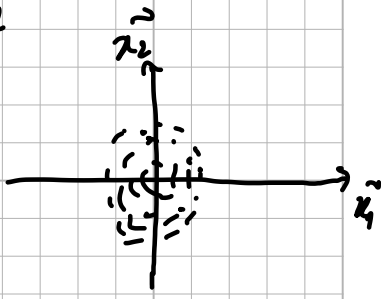
$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$

$x = x - \mu$



$\sigma^2 = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)})^2$

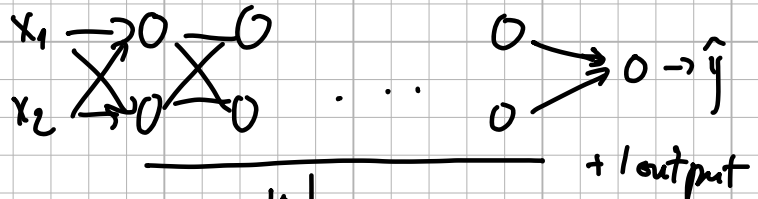
$x = x / \sigma$



$\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} = \mu = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$

$\mu_1 < 1$ $\quad \mu_2 > 1$

Vanishing / Exploding gradients



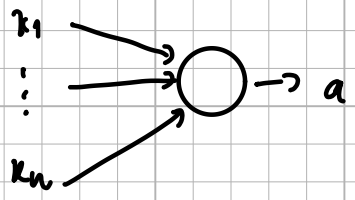
$g: z \to z$

$L = 0$

$\hat{y} = w^{[L]} \cdot a^{[L-1]}$

$= w^{[L]} \cdot w^{[L-1]} \cdots w^{[1]} \cdot x$

Example with 1 neuron :



$a = \sigma(z)$

$z = w_1 x_1 + \dots + w_n x_n$

large n → small w

$w_i \sim \frac{1}{n}$

$w^{[l]} = np.random.randn(shape) * np.sqrt\left(\frac{1}{n^{[l-1]}}\right)$

Sigmoid , ReLU $\frac{2}{n^{[l-1]}}$

Xavier Initialization

$W^{[l]} \sim \sqrt{\frac{1}{n^{[l-1]}}}$ for tanh

He Initialization :

$W^{[l]} \sim \sqrt{\frac{2}{n^{[l]} + n^{[l-1]}}}$

© optimization

$X = (x^{(1)}\ x^{(2)} \dots x^{(m)})$

$Y = (y^{(1)}\ y^{(2)} \dots y^{(m)})$

Batch $X = (X^{\{1\}} \dots X^{\{T\}})$

$(x^{(1)} \dots x^{(1000)})$

Batch $Y = (Y^{\{1\}} \dots Y^{\{m\}})$

Algo

for iteration $t = 1 \dots$

Select batch $(X^{\{t\}}, Y^{\{t\}})$

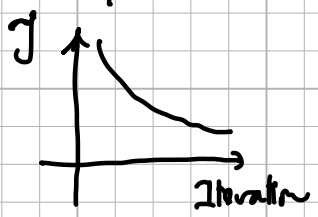
Forward Prop → Cost

Backprop Batch

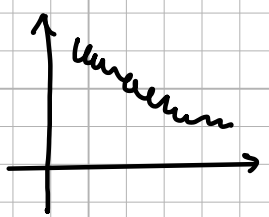
Update $w^{[l]}, b^{[l]}$

$J = \frac{1}{1000} \sum_{i=1}^{1000} \mathcal{L}^{(i)}$

Graph

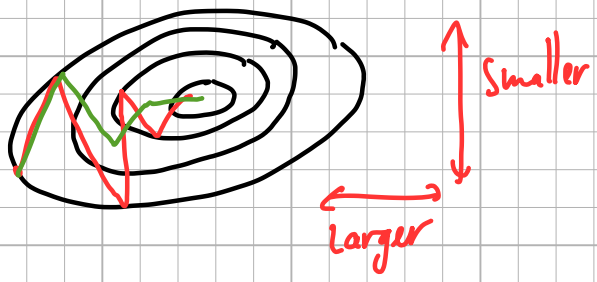


Batch Gradient descent



Mini-batch Gradient descent

GD + Momentum Algo



$$\begin{cases} v = \underset{\text{weights}}{\beta} v + (1-\beta)\frac{\partial \mathcal{L}}{\partial w} \\ w = w - \alpha v \end{cases}$$

Lecture 13 - Debugging ML Models and Error Analysis

claim có 1 cái ít nhất 2

Mình | đối thủ

2 | x → 1, 2, 3, ..., 6

→ Mình thua khi $x > 2$ và đối thủ đoán

↳ 3, 4, 5, 6

→ Mình thua khi $x = 2$ và đối thủ đoán 2 25

Các th đối thủ có thể đoán

1 → x 4 (3 → 6) = 4

2 → x 6 (1 → 6) = 6 → 10 th

→ đối thủ đoán 2 $[1..6] \setminus \{2\}$ → Mình reject → Mình thắng

↳ Mình không reject

Mình | đối thủ

x_1 | x_2

y_1 | y_2

Mình claim $x_1 \geq y_1$

đối thủ claim $x_2 \geq y_2 \geq x_1$

$x_2 \geq y_2 \geq x_1 \geq y_1$

↓

→ optimal để đoán

• 4 cái ít nhất y_1

→ th1: $x_2 \geq y_2 \geq x_1 = y_1$

→ th2: $x_2 \geq y_2 \geq x_1 > y_1$

moves = {1 ... 6}

Optimal để mình thắng

Mình (x_1) 1

Đối thủ (x_2) 2

Mình claim: $x_1 \geq y_1$

Đối thủ claim $x_2 \geq y_2 \geq y_1$

$f(y_1)$ = true nếu (1) thắng

$f(y_2)$ = false với $y_2 \geq y_1$

$f(y_1) = \begin{cases} \text{true}, & \exists\, y_2 \geq y_1 : f(y_2) = \text{false} \\ \text{false}, & \text{other} \end{cases}$

$f[i][y_1] = \begin{cases} \text{true}, & \exists\, y_2 \geq y_1 : f[i][y_2] = \text{false} \\ \text{false}, & \text{other} \end{cases}$

```
isWinning(state){
  moves = [1,2,3,4,5,6]
  for (all move in moves){
        newState = move if >= state else return false
        if (! isWinning(newState)) return True;
  }
  return false
}
```

→

th_1: $x_1 \geq 3$:

$f[1][3]$

	1	2	3	4	5	6

CS 229

PCA - Principal Component Analysis → Unsupervised Learning

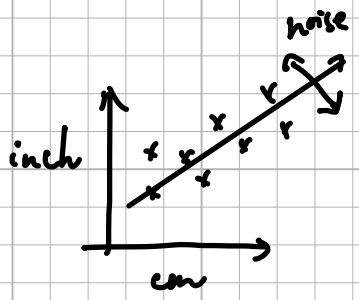
ICA - Independent Component Analysis

- Unlabeled dataset

$$\{x^{(1)}, \dots, x^{(m)}\} \in \mathbb{R}^n$$

Reduce dimension from $n \to k$

$k \ll n$



Pre-processing:

1. $\mu = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$
2. $x^{(i)} \leftarrow x^{(i)} - \mu$

} zero of mean

3. $\sigma_j^2 = \frac{1}{m} \sum_i x_j^{(i)2}$
4. $x_j^{(i)} \leftarrow \frac{x_j^{(i)}}{\sqrt{\sigma_j^2}}$

} standardize variance to 1

$\Rightarrow$ Max $u^T \Sigma u$

$u: \|u\| = 1 \rightarrow u^T u = 1$

$\Rightarrow$ Sol: u is principal eigenvector of Σ

11 ways to derive PCA !

General case: If you wish to project data to k-dim

→ Set $u_1 \to u_k$: top k - eigenvector

Have $x^{(i)} \in \mathbb{R}^n$ (say $n = 1000$)

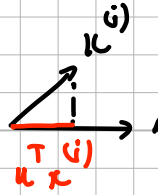
$u_1 \dots u_k$ (say $k = 10$)

$x^{(i)} \to (u_1^T x^{(i)}, \dots, u_{10}^T x^{(i)}) = y^{(i)} \in \mathbb{R}^k$

$x^{(i)} \approx y_1^{(i)} u_1 + \dots + y_k^{(i)} u_k$

- PCA → find direction of green line

→ $\vec{u}$, $\|u\| = 1$



Choose u to max:

$$\max_{u:\|u\|=1} \sum_{i=1}^{m} (x^{(i)T} u)^2$$

$$= \frac{1}{m} \sum_{i=1}^{m} u^T x^{(i)} x^{(i)T} u$$

$$= u^T \left(\frac{1}{m} \sum_{i=1}^{m} x^{(i)} x^{(i)T} \right) u$$

$$\Sigma = \frac{1}{m} \sum_{i=1}^{m} x^{(i)} x^{(i)T}$$

- Application:
  - Visualization

    Project from n-D → 2, 3 - D
  - Compression for ML efficiency
  - Reduce overfitting
  - outlier detection } matching
  - Eigenfaces

Questionable

- Rules of thumb

- Before using PCA, use original data.

	Model $P(x)$ (e.g anomaly detection)	Non-probabilistic (e.g compression, visualization)
"Subspaces"	factor analysis	PCA
"Clusters"	Mixture of Gaussian	K-means

E.M

$P(x, z)$

↑ latent

$P(x)$

ICA ⟸ Independent

- Original sources: $s \in \mathbb{R}^n$ (n speakers)

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

→ We observe

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

$x_j^{(i)}$ = recording of micro j at time i ($j = 1 \dots n$)

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

find $W = A^{-1}$ so that $s^{(i)} = W \cdot x^{(i)}$

$$Wx = \begin{bmatrix} — & w_1^T & — \\ & \dots & \\ — & w_n^T & — \end{bmatrix} \cdot x$$

Lecture 14: Expectation - Maximization Algorithms

+) Clustering (K-means)

Data: $\{x^{(1)} \dots x^{(m)}\}$

1. Initialize cluster centroids

$\mu_1 \dots \mu_k \in \mathbb{R}^n$ randomly

2. Repeat until convergence:

a) Set $c^{(i)} = \arg\min_j \|x^{(i)} - \mu_j\|_2^2$ ("color the points")

b) for $j = 1 \dots k$

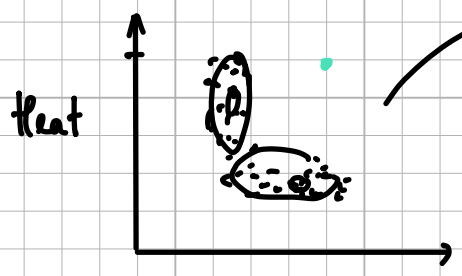
$$\mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\}\, x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}}$$ ("move the cluster to")

$$J(c, \mu) = \sum_{i=1}^{m} \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

(c: assignments, μ: centroid)

=> purpose to choose k ?

+) Density estimation



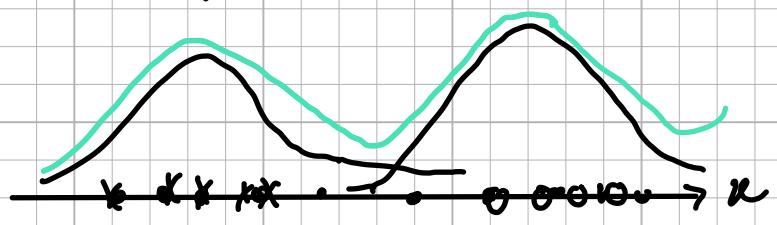
→ the contours of two Gaussians ?

Anomaly detection

→ model $p(x)$

$p(x) < \varepsilon$

+) 1-D example ($x \in \mathbb{R}$)



→ Gaussian Discriminant analysis

→ EM algo !!! (expectation maximation)

+) Mixture of Gaussian model

Suppose there's a latent (hidden/unobserved) random variable z, and $x^{(i)}$, $z^{(i)}$ are distributed

$$p(x^{(i)}, z^{(i)}) = p(x^{(i)} \mid z^{(i)}) \cdot p(z^{(i)})$$

where $z^{(i)} \sim \text{Multinomial}(\phi)$

$$x^{(i)} \mid z^{(i)} = j \sim \mathcal{N}(\mu_j, \Sigma_j)$$

• If we knew the $z^{(i)}$, can use MLE

$$\ell(\phi, \mu, \Sigma) = \sum_{i=1}^{m} \log p(x^{(i)}, z^{(i)}; \phi, \mu, \Sigma)$$

$$\phi_j = \frac{1}{m} \sum_{i=1}^{m} 1\{z^{(i)} = j\}$$

$$\mu_j = \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{z^{(i)} = j\}}$$

+) E-step (Guess value of $z^{(i)}$s)

Set $w_j^{(i)} = p(z^{(i)} = j \mid x^{(i)}; \phi, \mu, \Sigma)$

← →

$$= \frac{p(x^{(i)} \mid z^{(i)} = j)\, p(z^{(i)} = j)}{\sum_{l=1}^{k} p(x^{(i)} \mid z^{(i)} = l)\, p(z^{(i)} = l)} \rightsquigarrow \phi_j$$

$\mathcal{N}(\mu_j, \Sigma_j)$

+) M-Step

$$\phi_j := \frac{1}{m} \sum_{i=1}^{m} w_j^{(i)}$$

Σ_j : lecture notes

$$\mu_j = \frac{\sum_{i=1}^{m} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_j^{(i)}} \rightsquigarrow E\left[1\{z^{(i)} = j\}\right]$$

$$p(x, z; \theta) \rightarrow \operatorname{argmax}_{\theta} \prod_{i=1}^{m} p(x^{(i)}; \theta)$$

- $w_j^{(i)}$ is how much $x^{(i)}$ is assigned to the j Gaussian

+) Jensen's inequality

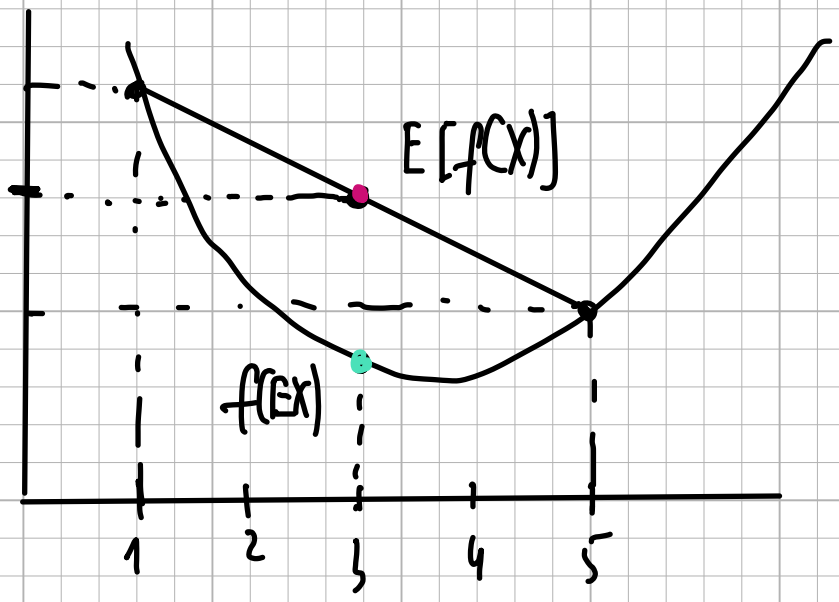
Let f be a convex function (e.g. $f''(x) \geq 0$)

x be a random variable

Then $f(EX) \leq E(f(X))$

$$X = \begin{cases} 1 & \text{with prob } 1/2 \\ 5 & \text{with prob } 1/2 \end{cases} \qquad E[X] = EX = 3$$

Example:



Đẳng "=" xảy ra khi f is strictly convex

$\Leftrightarrow X$ is constant

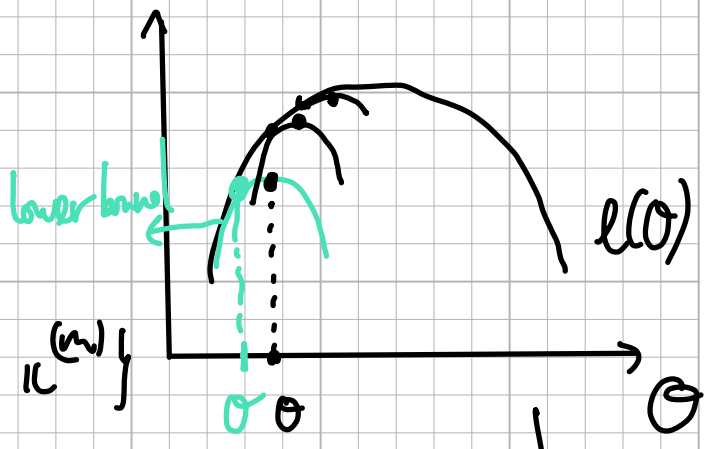
($X = EX$ with prob 1)

- Concave function

Have model for

$P(x, z; \theta)$

only observe $x \{x^{(1)} \dots x^{(m)}\}$



$$\ell(\theta) = \sum_{i=1}^{m} \log P(x^{(i)}; \theta)$$

$$= \sum_{i=1}^{m} \log \sum_{z^{(i)}} P(x^{(i)}, z^{(i)}; \theta)$$

Converge to local optimum

Want $\arg\max_{\theta} \ell(\theta)$

$$\max_{\theta} \sum_i \log P(x^{(i)}; \theta)$$

$$= \sum_i \log \sum_{z^{(i)}} P(x^{(i)}, z^{(i)}; \theta)$$

$$= \sum_i \log \sum_{z^{(i)}} Q_i(z^{(i)}) \frac{P(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})}$$

where $Q(z^{(i)})$ is a prob distribution $\sum_{z^{(i)}} Q(z^{(i)}) = 1$

↑ how to come up?

$$= \sum_i \log E_{z^{(i)} \sim Q_i} \left[\frac{P(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})} \right]$$

$$\geq \sum_i E_{z^{(i)} \sim Q_i} \left[\log \frac{P(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})} \right]$$

$$= \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \cdot \log \frac{P(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})}$$

On a given iteration of EM with parameters θ, we want: $Q_i(z^{(i)}) = p(z^{(i)} | x^{(i)}; \theta)$ so that

$$\frac{P(x^{(i)}, z^{(i)})}{Q_i(z^{(i)})} = \text{constant}$$

Set $Q_i(z^{(i)}) \propto p(x^{(i)}, z^{(i)}; \theta)$

$$Q_i(z^{(i)}) = \frac{p(x^{(i)}, z^{(i)}; \theta)}{\sum_{z^{(i)}} p(x^{(i)}, z^{(i)}; \theta)} = \dots = p(z^{(i)} \mid x^{(i)}; \theta)$$

- E-step: $w_j^{(i)}$

  Set $Q_i(z^{(i)}) := p(z^{(i)} \mid x^{(i)}; \theta)$

- M-step:

$$\theta := \arg\max_{\theta} \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})}$$

Lecture 15: EM Algorithm & Factor Analysis

Recap: $p(x^{(i)}, z^{(i)}) = p(x^{(i)} \mid z^{(i)}) \cdot p(z^{(i)})$

$z^{(i)} \sim \text{Multinomial}(\phi)$

$(p(z^{(i)} = j) = \phi_j)$

$x^{(i)} \mid z^{(i)} = j \sim \mathcal{N}(\mu_j, \Sigma_j)$

↖ latent

E-step:

$w_j^{(i)} = Q_i(z^{(i)} = j) = p(z^{(i)} = j \mid x^{(i)}; \phi, \mu, \Sigma)$

"$p(z^{(i)} = j)$"

M-step:

$$\max_{\phi,\mu,\Sigma} \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \phi, \mu, \Sigma)}{Q_i(z^{(i)})}$$

$$= \sum_i \sum_j w_j^{(i)} \log \frac{\frac{1}{(2\pi)^{n/2} |\Sigma_j|^{1/2}} \exp\left(-\frac{1}{2}(x^{(i)} - \mu_j)^T \Sigma_j^{-1} (x^{(i)} - \mu_j)\right) \cdot \phi_j}{w_j^{(i)}}$$

$w_j^{(i)}$: "strength" with which $x^{(i)}$ is assigned to Gaussian j

$$\nabla_{\mu_j}(\ldots) \overset{\text{set}}{=} 0 \Rightarrow \mu_j = \frac{\sum_i w_j^{(i)} x^{(i)}}{\sum_i w_j^{(i)}}$$

$$\nabla_{\phi}(\ldots) \overset{\text{set}}{=} 0, \quad \nabla_{\Sigma_j}(\ldots) \overset{\text{set}}{=} 0$$

optimal value $\Rightarrow$

$$\phi_j = \frac{\sum_i w_j^{(i)}}{\sum_i \sum_j w_j^{(i)}}$$

+) $$J(\theta, Q) = \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \cdot \log \frac{p(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})}$$

we know $\ell(\theta) \geq J(\theta, Q)$ for any θ, Q

E-step: Maximize J wrt Q

M-step: Maximize J wrt θ

+) Mixture of Gaussian

Say $n = 2$, $m = 100$

$m \approx n$, or $m \ll n$

Say $m = 50$, $n = 100$

↓ 50 days ↓ 100 sensors

$x = \begin{bmatrix} \\ \\ \end{bmatrix} \in \mathbb{R}^{100}$ → anomaly

$P(x) \leq \varepsilon$

Model as single Gaussian

$x \sim N(\mu, \Sigma)$

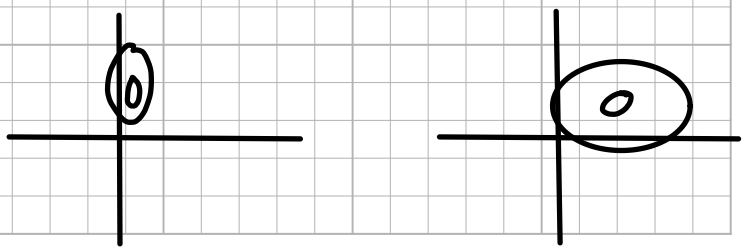
MLE: $\mu = \frac{1}{m} \sum_{i} x^{(i)}$

Maximum likelihood estimate $\Sigma = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)} - \mu)(x^{(i)} - \mu)^T$

If $m \leq n$ ⇒ Σ is singular / non invertable

<u>Option 1</u>: Constrain Σ to be diagonal

$\Sigma = \begin{bmatrix} \sigma_1^2 & & & 0 \\ & \sigma_2^2 & & \\ & & \ddots & \\ 0 & & & \sigma_n^2 \end{bmatrix}$ → axes-alligned contours



MLE: $\sigma_j^2 = \frac{1}{m}\sum_i (x_j^{(i)} - \mu_j)^2$ $\Rightarrow$ Assume uncorrelated

n parameters

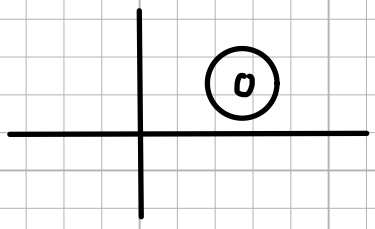
between any 2s

Option 2

Constrain $\Sigma = \begin{bmatrix} \sigma^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma^2 \end{bmatrix}$

1 parameters $\Rightarrow$ circular Contours

MLE $\sigma^2 = \frac{1}{m}\frac{1}{n}\sum_i\sum_j (x_j^{(i)} - \mu_j)^2$



+) $P(x, z) = P(x|z) \cdot P(z)$

z is hidden

$d = 3$

$n = 100$

$m = 50$

$z \sim \mathcal{N}(0, 1)$

Factor analysis model

$z \in \mathbb{R}^d$ $(d < n)$

$x = \mu + \Lambda z + \varepsilon$ "psi" $\rightarrow$ Equivalently

where $\varepsilon \sim \mathcal{N}(0, \Psi)$ $x|z \sim \mathcal{N}(\mu + \Lambda z, \Psi)$

- Parameters

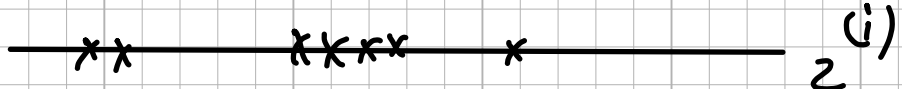
$\mu \in \mathbb{R}^n$, $\Lambda \in \mathbb{R}^{n \times d}$, $\Psi \in \mathbb{R}^{n \times n}$ diagonal

$\hookrightarrow$ noise between sensors independent

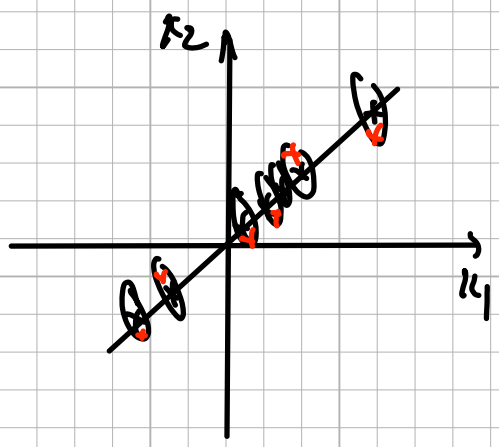
Examples

$d=1 \rightarrow$ lies on 1D subspace with units
$n=2 \rightarrow$ 2D data
$m=7$

1. $z \in \mathbb{R}^1$, $x \in \mathbb{R}^2$ $\quad z \sim \mathcal{N}(0,1)$

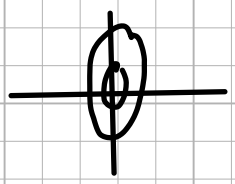


Say $\lambda = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$, $\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$



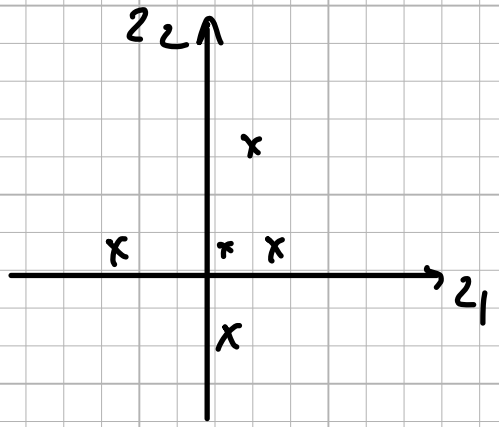
$\lambda z + \mu \in \mathbb{R}^2$

$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$



$x = \lambda z + \mu + \varepsilon$

2. $z \in \mathbb{R}^2$, $x \in \mathbb{R}^1$, $m=5$, $d=2$, $n=3$



z

$\lambda z + \mu$

λ: 3×2, μ: 3×1

+) Multivariate Gaussian

$x = \begin{bmatrix} x_1 \\ \hline x_2 \end{bmatrix} \begin{matrix} \updownarrow r \\ \updownarrow s \end{matrix}$

$x_1 \in \mathbb{R}^r$, $x_2 \in \mathbb{R}^s$

$x \in \mathbb{R}^{r+s}$

$x \sim N(\mu, \Sigma)$ $\quad \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \begin{matrix} \updownarrow r \\ \updownarrow s \end{matrix}$

Marginal $P(x_i) = ?$ $\quad \Sigma = \begin{matrix} r\updownarrow \\ s\updownarrow \end{matrix} \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}$ Cov

$P(x) = P(x_1, x_2)$

$= \int_{x_2} P(x_1, x_2)\, dx_2 \cdot P(x_1)$

$P(x_1)$ given $x_1 \sim \mu(\mu_1, \Sigma_{11})$

Conditional $P(x_1 | x_2) = ?$

$x_1 | x_2 \sim N(\mu_{1|2}, \Sigma_{1|2})$

$\mu_{1|2} = \mu_1 + \Sigma_{12} \Sigma_{22}^{-1} (x_2 - \mu_2)$

$\Sigma_{12} = \Sigma_{11} - \Sigma_{12} \Sigma_{22}^{-1} \Sigma_{21}$

$\frac{P(x_1, x_2)}{P(x_2)}$

Step 1: Derive $P(x, z)$

$\begin{pmatrix} z \\ x \end{pmatrix} \sim N(\mu_{x,z}, \Sigma)$

$z \sim N(0, I)$ $\quad Ez = 0$

$x = \mu + \Lambda z + \varepsilon$ $\quad Ex = E(\mu + \Lambda z + \varepsilon)$

$= \mu$

$$\mu_{x2} = \begin{bmatrix} 0 \\ \mu \end{bmatrix} \begin{matrix} \updownarrow d \\ \updownarrow n \end{matrix} \qquad \Sigma_{22} = E(x - E[x])(x - E[x])^T$$

$$\Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} \qquad = E[\Lambda z z^T \Lambda^T] + E[\epsilon \epsilon^T]$$

$$\begin{matrix} \hookrightarrow \\ d \end{matrix} \quad \begin{matrix} \hookrightarrow \\ n \end{matrix} \qquad = \Lambda E(zz^T) \Lambda^T + \Psi$$

$$= \Lambda\Lambda^T + \Psi$$

$$= \begin{bmatrix} I & \Lambda^T \\ \Lambda & \Lambda\Lambda^T + \Psi \end{bmatrix}$$

$$\begin{bmatrix} z \\ x \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \mu \end{bmatrix}, \begin{bmatrix} I & \Lambda^T \\ \Lambda\Lambda^T + \Psi \end{bmatrix} \right)$$

$P(x^{(i)})$

- E - Step:

$$Q_i(z^{(i)}) = P(z^{(i)} \mid x^{(i)}; \theta)$$

$$z^{(i)} \mid x^{(i)} \sim \mathcal{N}(\mu_{z^{(i)} \mid x^{(i)}}, \Sigma_{z^{(i)} \mid x^{(i)}})$$

where

$$\mu_{z^{(i)} \mid x^{(i)}} = \vec{0} + \Lambda^T (\Lambda\Lambda^T + \Psi)^{-1} (x^{(i)} - \mu)$$

$$\Sigma_{z^{(i)} \mid x^{(i)}} = I - \Lambda^T (\Lambda\Lambda^T + \Psi)^{-1} \Lambda$$

- M- Step:

$Q_i(z^{(i)})$

$$\theta := \arg\max_\theta \sum_i \int_{z^{(i)}} Q_i(z^{(i)}) \cdot \log \frac{p(x^{(i)}, z^{(i)})}{Q_i(z^{(i)})} \, dz^{(i)}$$

$$= \sum_i E_{z^{(i)} \sim Q_i} \left[\log \frac{p(x^{(i)}, z^{(i)})}{Q_i(z^{(i)})} \right]$$

quadratic function

$\nabla_\mu = 0$

Lecture 16: Independent Component Analysis & RL

ICA Problem:

Sources: $s \in \mathbb{R}^n$

$s_j^{(i)}$ = speaker j at time i

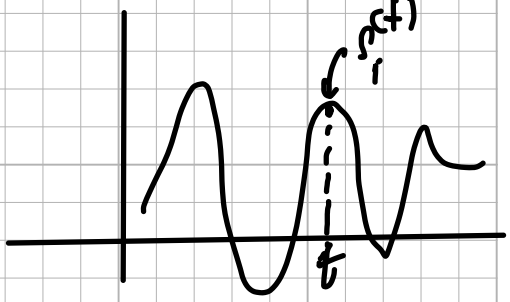
$x^{(i)} = A s^{(i)}$, $x \in \mathbb{R}^n$

Goal: Find $W = A^{-1}$, so

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

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

Speaker



↓

"non Gaussian"

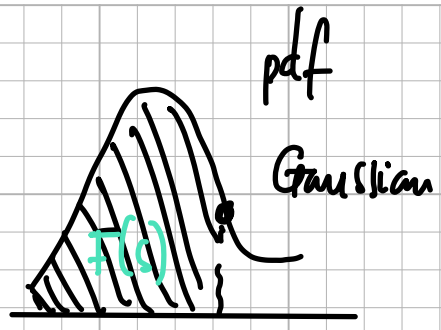
$p_S(s)$

Cumulative DF: constant

$F(s) = P(S \le s)$

↑ random variable

e.g.



$p_S(s) = F'(s)$

Note: Choose $F(s)$

$p_S(s) = 1\{0 \le s \le 1\}$ $(S \sim \text{Uniform}(0,1))$

Density of S: $p_S(s)$

say $X = 2S$ $(A = 2, w = \frac{1}{2})$

$X = AS = w^{-1}S$

$X \sim \text{Uniform}(0,2)$

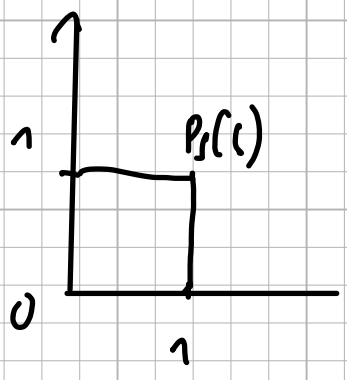
$S = wX$

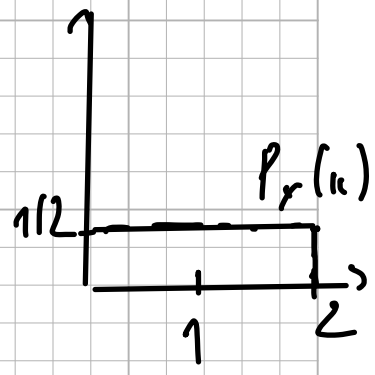
Reciprocal of w

$p_X(x) = p_S(wx)|w|$

↑ Density of x

↓ how much it stretch out





$p_X(x) = \frac{1}{2}\, 1\{0 \le x \le 2\}$

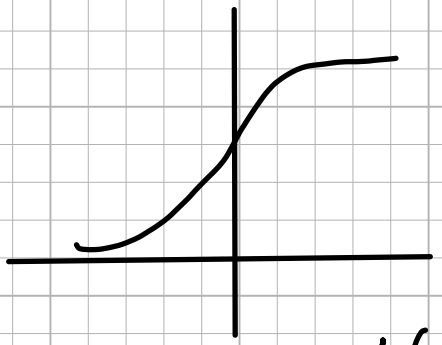
$p_S(s) = ?$

$F(s) = P(S \le s)$

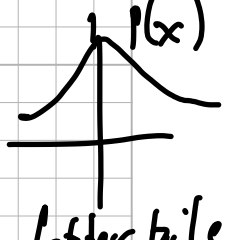
↑ chance of person voice

sigmoid function

$F(s) = \frac{1}{1+e^{-s}}$



$p_S(s) = F'(s)$



→ Laplace distribution also works

$$P_S(s) = \prod_{i=1}^{w} P_s(s_i)$$

(w speakers independent)

$$P_x(x) = P_s(w_x)|w|$$

$$= \left(\prod_{j=1}^{w} P_s(w_j^T x) \right) |w|$$

$$g(z) = \frac{1}{1+e^{-z}}$$

MLE:

$$\ell(w) = \sum_{i=1}^{m} \log \left[\left(\prod_j g(w_j^T x^{(i)}) \right) |w| \right]$$

Stochastic gradient descent:

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

→ Recover $s = wx$

+) Reinforcement Learning

→ Cost function, Reward function

$R(s) = +1$ for win

-1 for lose

0 for tie

reward ↑ state

Credit assignment problem

+) Markov Decision Process (MDP)

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

S = Set of states

A = Set of actions

P_{sa} - State transition probabilities

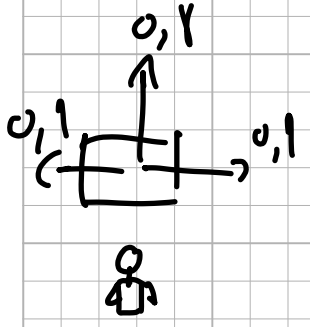
$\left(\sum_{s_i} P_{sa}(s_i) = 1 \right)$

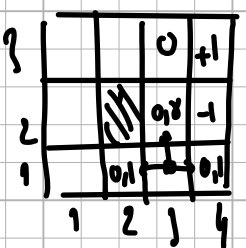
γ - discount factor

$\gamma \in [0, 1)$

R - reward function

- MDP - Robot navigating this simple maze, a obstacle





11 States

Actions {N, S, E, W}

$P_{(3,2)N}((3,2)) = 0,8$

$R((4,1)) = +1$

$R((4,2)) = -1$

$0,1 = P_{(3,2)}((2,1)) = P_{(3,2)}((4,1))$

$P(S) = -0,02$ for all other states

$P_{(3,2)}(3,1) = 0$

- Total payoff

$R(S_0) + \gamma R(S_1) + \gamma^2 R(S_2) + \dots$

$\gamma = 0,99$

S_0

Choose action a_0

Get to $S_1 \sim P_{S_0 a_0}$

Choose action a_1

Get to $S_2 \sim P_{S_1 a_1}$

⋮

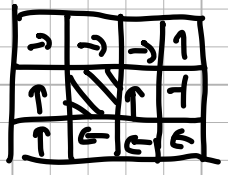
- Goal: Choose actions over time to maximize

$E[R(S_0) + \gamma R(S_1) + \gamma^2 R(S_2) + \dots]$

Policy $\pi: S \to A$ (Controller)

when in state S, take action $\pi(S)$

optimal policy:



Lecture 17: MDP & Value/Policy Iteration

+) Define: V^{π}, V^{*}, π^{*}

for a policy π, $V^{\pi}: S \to \mathbb{R}$ is

s.t $V^{\pi}(s)$ is the expected total payoff for starting in state s and executing π

$$V^{\pi}(s) = E[\underbrace{R(S_0)}_{\text{Immediate reward}} + \gamma R(S_1) + \dots \mid \pi, S_0 = s]$$

"Value function for policy π"

π

3	→	→	→	+1
2	↓		→	-1
1	→	→	↑	↑
	1	2	3	4

V^{π}

.52	.75	.77	+1
.90		.82	-1
.88	.77	.85	.1.20

+) Bellman's Equations:

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

s s'

s_0 s_1

$$V^{\pi}(s) = E[R(s) + \gamma V^{\pi}(s')]$$

$s = s_0$, $s' = s_1$

$$s' \sim P_{s\pi(s)}$$

In state s, take action $a = \pi(s)$, $s' \sim P_{sa}$

Given π, get a linear system of equations in terms of $V^\pi(s)$

$$V^\pi((1,1)) = R((1,1)) + \gamma\left(0.8\, V^\pi((1,2)) + 0.1\, V^\pi((2,1)) + 0.1\, V^\pi((1,1))\right)$$

- 11 states → 11 vectors, linear solver

+) V^* is the optimal value function

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

Bellman equations: $V^*(s) = R(s) + \max_a \gamma \sum_{s'} P_{sa}(s')\, V^*(s')$ (expected future reward)

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

↑ optimal policy

+) Practice w/ confusing notation:

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

for every π, s

Strategy:
1) find V^*
2) use argmax func to find π^*

+) Value Iteration:

Initialize $V(s) := 0$ for every s

For every s, update

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

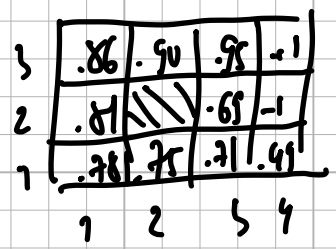
"Synchronous update" vs. "Asynchronous update"

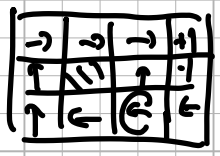
Bellman backup operator

$V = B(V)$

Value Iteration cause V to converge to V^*

V^*





"$\leftarrow$" $W_{\leftarrow} = 0{,}8 \cdot 0{,}75 + 0{,}1 \cdot 0{,}69 + 0{,}1 \cdot 0{,}49$
$= 0{,}740$ ↑ better action

"π" $W_{\pi} = 0{,}8 \cdot 0{,}69 + 0{,}1 \cdot 0{,}75 + 0{,}1 \cdot 0{,}49$
$= 0{,}690$

+) Policy iteration

Initialize π randomly

Repeat:

Set $V := V^{\pi}$ (i.e., solve Bellman's equation to get V^{π})

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

V^{π} - linear system of equations → small problems: faster
expensive

+) What if we don't know P_{sa}?

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

(or $\frac{1}{|S|}$ if above is "0") $\frac{+1}{+|}$ (laplace)

+) Putting it together:

Repeat: {

E-greedy
0.9 chance wrt Π
0.1 chance randomly

take actions wrt Π to get experience in MDP

Update estimates of P_{sa}, (and possibly R)

Solve Bellman's equations using value iteration to get V

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

}

+) Exploration vs. Exploitation

"Boltzmann exploration"

"intrinsic motivation"

☐ ML
☐ DL
☑ Control
☑ Data mining

Lecture 18: Continuous State MDP & Model Simulation

- Discretization
- Models Simulation
- Fitted value iteration

+ Recap:

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

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

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

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

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

Value iteration

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

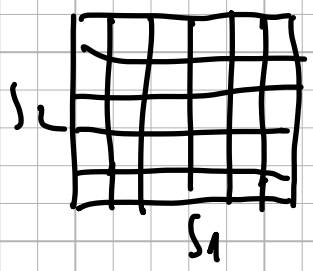
Car, Helicopter:

		roll, pitch, yaw ↑
Kinematic model	(x, y, z)	$x, y, z, \phi, \theta, \psi$
Dynamic model	$x, y, \dot{\theta}$	$x, y, z, \dot{\phi}, \dot{\theta}, \dot{\psi}$

free flying: $x, \theta, \dot{x}, \dot{\theta}$ to keep balanced

State space: R^{N}

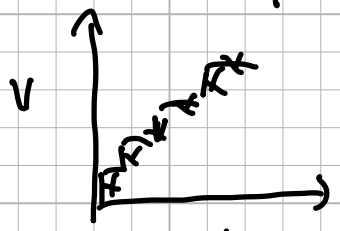
+) Discretization



1-D x, y : state inverted pendulum

Problems:

- Naive representation for V^* (and π^*)



↓ not very good

- Curse of dimensionality

$S = \mathbb{R}^N$, and discretize each dimension into K values

get K^N discrete space

+) Approximate V^* directly, w/o resorting to discretization

$$y \approx \theta^T \phi(x), \quad \phi(x) = \text{feature of } x = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ \vdots \end{bmatrix}$$

$$V^*(s) \approx \theta^T \phi(s)$$

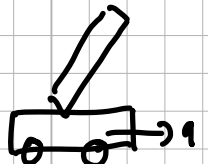
+) Model (simulator) of MDP:

$$s \rightarrow \boxed{\text{Model}} \rightarrow s' \sim P_{sa}$$

a (≈ discretize)

+) How to get model?

- Physics simulator



$$s = \begin{pmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{pmatrix} \quad \dot{s} = \begin{pmatrix} \dot{x} \\ \alpha - L\beta\cos\theta/m \\ \dot{\theta} \\ \beta \end{pmatrix}$$

$s' = s + (\Delta t)\cdot \dot{s}$

- Learn from data

Helicopter $s_0^{(1)} \xrightarrow{a_0^{(1)}} s_1^{(1)} \xrightarrow{a_1^{(1)}} s_2^{(1)} \xrightarrow{a_2^{(1)}} \dots \to s_T^{(1)}$

Human pilot

$\vdots$

$s_0^{(m)} \to \dots$

Apply supervised learning to estimate

s_{t+1} as function of s_t, a_t

(s') (s, a)

Eg: Linear regression version (Low-speed Helicopter)

$$s_{t+1} = \underbrace{A}_{\mathbb{R}^{n \times n}} s_t + B a_t$$

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

Model

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

or

$s_{t+1} = A s_t + B a_t + \varepsilon_t$ where $\varepsilon_t \sim \mathcal{N}(0, \sigma^2 I)$

(stochastic)

"Model-free RL" "Model-based"

Fitted value iteration:

Choose feature $\phi(s)$ of state s

$$V(s) = \theta^T \phi(s)$$

$\phi(s)$ = non linear func...

Value Iteration

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

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

$s \to V(s)$

Fitted Value Iteration

Sample $\{s^{(1)}, \dots s^{(m)}\} \subseteq S$ randomly

Initialize $\theta := 0$

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

Repeat {

For $i = 1 \dots m$ {

For each action $a \in A$ { (using model)

Sample $s_1', s_2' \dots s_k' \sim P_{s^{(i)}, a}$

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

}

}

Fitted VI:

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

i.e. $\theta^T \phi(s^{(i)}) \approx y^{(i)}$

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

- Fitted VI gives approximation to V^*

  Implicitly defines π^*

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

  Used $s'_1, \ldots, s'_k \sim P_{sa}$ to approximate expectation

  Say model is

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

  (e.g. $A s_t + B a_t + \varepsilon_t$)

  for deployment (run time),

  Set $\varepsilon := 0$ and $K := 1$

  When in state s,

  Pick action

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

  ↑ simulation w/o noise

  ↓

  $s' \sim P_{sa}$, with deterministic simulator

Lecture 15: Reward Model & Linear Dynamical System

Outline: - State-action rewards

- finite horizon MDP

- Linear dynamical systems

+ Model

+ LQR (Linear-quadratic regulation)

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

Value iteration:

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

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

+) State-action rewards

$R: S \times A \longmapsto \mathbb{R}$

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

Bellman's eqns (immediate reward / future rewards)

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

Value iteration $V(s) := RHS$

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

+) finite horizon MDP.

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

Horizon time T (No γ)

Payoff $E[R(s_0, a_0) + R(s_1, a_1) \dots R(s_T, a_T)]$

$\pi^*_t(s)$: Non-stationary policy

↳ changes over time

$\pi^*(s)$: Stationary policy

Non-stationary state transitions

$s_{t+1} \sim P^{(t)}_{s_t a_t}$ $\quad R^{(t)}(s,a)$ at certain state, certain time

- Changing dynamics
- weather forecasts
- Industrial automation

...

optimal value function :

$V_t^*(s) = E[R(s_t, a_t) + \cdots + R(s_T, a_T) \mid \pi^*, s_0 = s]$

expected total payoff starting in state s at time t, execute π^*

Value iteration (Dynamic Programming)

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

$$\pi_t^*(s) = \arg\max_a$$

non-stationary

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

$\rightarrow V_T^*, V_{T-1}^*, \ldots V_0^*$

+) Linear Quadratic Regulation (LQR)

only relatively small set

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

$S = \mathbb{R}^N$, $A = \mathbb{R}^d$

$P_{sa}: s_{t+1} = A^{(t)} s_t + B^{(t)} a_t + w_t$ $\quad w \sim \mathcal{N}(0, \Sigma_w)$

A: matrix, B: matrix, w_t: noise

$A \in \mathbb{R}^{N \times N}$, $B \in \mathbb{R}^{N \times d}$

less important

x, y, θ

$\dot{x}, \dot{y}, \dot{\theta}$

$$R(s,a) = -\left(s^{T(t)} U s + a^{T(t)} V a\right)$$

where $U \in \mathbb{R}^{n \times n}$, $V \in \mathbb{R}^{d \times d}$

$U, V \succeq 0$ (positive-semi-definite)

implies $s^T U s \geq 0$, $a^T V a \geq 0$

Want $s \approx 0$

Choose $U = I$, $V = I$

$$R(s,a) = -\left(\|s\|^2 + \|a\|^2\right)$$

Where to get A, B?

$$s_0^{(1)} \to s_1^{(1)} \to \dots \to s_T^{(1)}$$

$$\vdots$$

$$s_0^{(m)} \xrightarrow{a_0^{(m)}} \qquad s_{t+1} \approx A s_t + B a_t$$

$$\min_{A,B} \sum_{i=1}^{m} \sum_{t=0}^{t-1} \| s_{t+1} - (A s_t + B a_t) \|^2$$

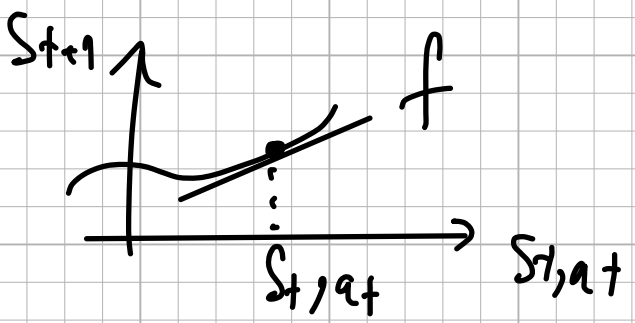
Method 1: learn from data

Method 2: linearize non-linear model

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

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

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



$$s_{t+1} \approx f'(\bar{s}_t)(s_t - \bar{s}_t) + f(\bar{s}_t)$$

$\bar{s}_t$ ~ a const

linearize f around $(\bar{s}_t, \bar{a}_t)$:

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

↳ $s_{t+1} \approx A s_t + B a_t$

$$s_t = \begin{pmatrix} 1 \\ x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{pmatrix}$$

Problem:

$$s_{t+1} = As_t + Ba_t + \underbrace{\omega_t}_{\text{noise}}$$

$$R(s,a) = -(s^T U s + a^T V a)$$

Total payoff: $R(s_0, a_0) + \dots + R(s_T, a_T)$

+) Dynamic programming

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

$$= \max_{a_t} -s^T U s_t - \underbrace{a_T V a}_{\geq 0}$$

$$= -s_T^T U s_T$$

$$\pi_T^*(s_T) = \vec{0}$$

Suppose $V_{t+1}^*(s_{t+1}) = s_{t+1}^T \underbrace{\Phi}_{\in \mathbb{R}^{n \times n}} s_{t+1} + \underbrace{\Psi}_{\in \mathbb{R}} t+1$

$$V_{t+1}^* \to V_t^*$$

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

$N(As_t + Ba_t, \varepsilon_t)$

The best action is linear func

$$V_t^{\pi}(s_t) = s_t^T \Phi_t s_t + \Psi_t$$

where **Lecture Notes !!**

+) To summarize:

Initialize $\Phi_T = -U$, $\Psi_T = 0$

Recursive calculate

Φ_t, Ψ_t using Φ_{t+1}, Ψ_{t+1} (for $t = T-1 \dots 0$)

Calculate L_t

$$\pi^*(s_t) = L_t s_t$$

V^* depends on Σ_w

π^*, L_t don't depend on Σ_w

Lecture 20: RL Debugging and Diagnostics

+) Debugging: Improve model / simulator?

Diagnostics: Modify reward func R?

Modify RL algorithm!

+) Policy search algorithm

("direct policy search")

$V^* \rightarrow \pi^*$

Try to find a good π directly

New definition:

A stochastic policy is a func

$\pi: S \times A \mapsto \mathbb{R}$ when $\pi(s,a)$ is the prob of taking action a in state s

$$\sum_a \pi(s,a) = 1$$

→ Inverted pendulum

$$\pi_\theta(s, \text{"R"}) = \frac{1}{1+e^{-\theta^T s}}$$

$$\pi_\theta(s, \text{"L"}) = 1 - \frac{1}{1+e^{-\theta^T s}}$$

$$s = \begin{pmatrix} 1 \\ x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{pmatrix}$$

Goal: Find θ so that when we execute $\pi_\theta(s,a)$, we maximize

$$\max_{\theta} E[R(s_0, a_0) + \dots + R(s_T, a_T) \mid \pi_\theta]$$

Note: Here, s_0 is a fixed initial state

+) Reinforce algorithm (Let $T = 1$)

Loop {

Sample $s_0, a_0, \dots s_T, a_T$

Compute pay off $= R(s_0) + \dots + R(s_T)$

Update

$$\theta = \theta + \alpha \left[\frac{\nabla_\theta \pi_\theta (s_0, a_0)}{\pi_\theta (s_0, a_0)} + \dots \right] \cdot \text{payoff}$$

}

+) POMDP

↳ partially observable

- at each step get a partial (and potentially noisy) measurement of the state. Have to take action "a" using that

$a = \theta^T s +$ gaussian noise

observation

$$y = \begin{pmatrix} x \\ \phi \end{pmatrix} + \text{noise}$$

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

+) low-level control task

$\int s$ n^{th} or v^{th} sampler