

Lec 13 [Andrew NG]

key ideas

1. Diagnostics for debugging learning algos
2. Error and ablative analyses
3. How to get started on a ML problem: - premature stat. optimisation.

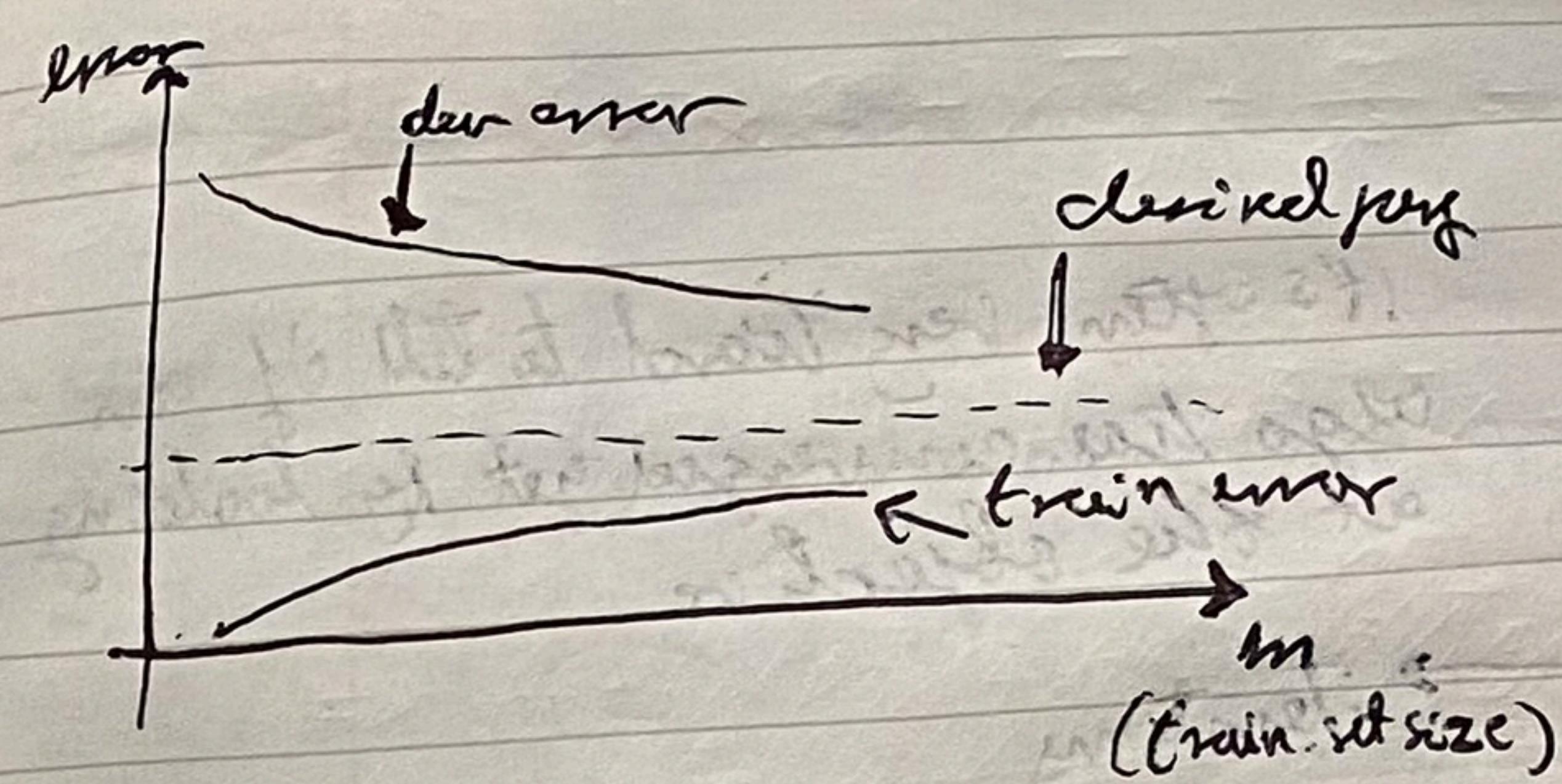
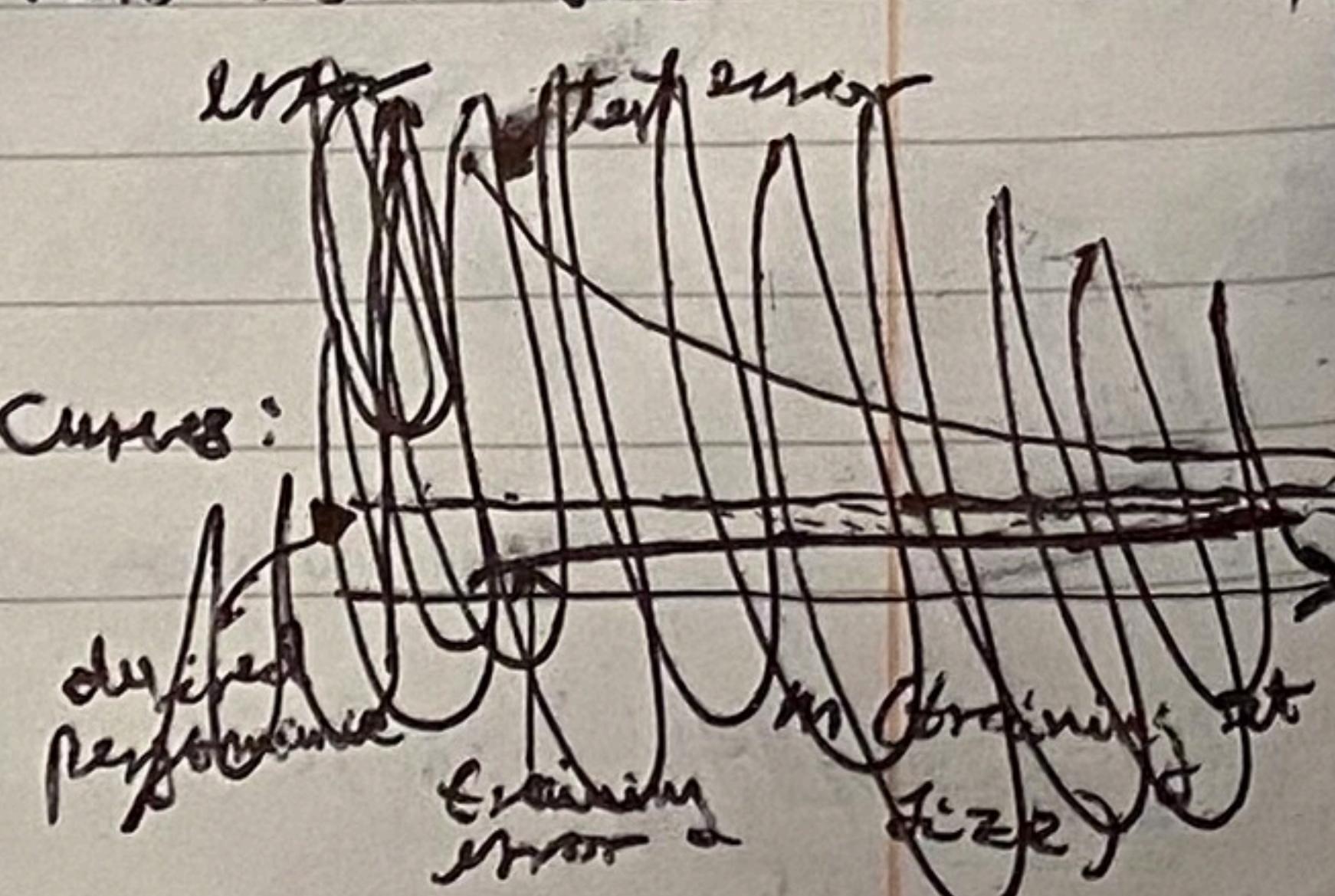
Ex:

- Anti-spam: carefully ~~do~~ choose 100 words to use as features (instead of all ~50k)
- Logistic reg + regularization (= Bayesian logreg) w/ grad-ascent gets 20% **train** error: too high. [$\max \sum_i \log p(y_i | x_i, \theta) - \lambda \|\theta\|^2$]
- What next?
- Common approach: Try improving algo in various ways.
 - Try getting more data & hope.
 - Try smaller set of features
 - Try more features
 - Try changing the features. Eg email header vs body features
 - Run gradient ascent for more iterations
 - or Newton's method
 - or use different λ
 - Try using SVM

Just brainstorming is already ahead of many teams. What diagnostics we use to pick best approach?

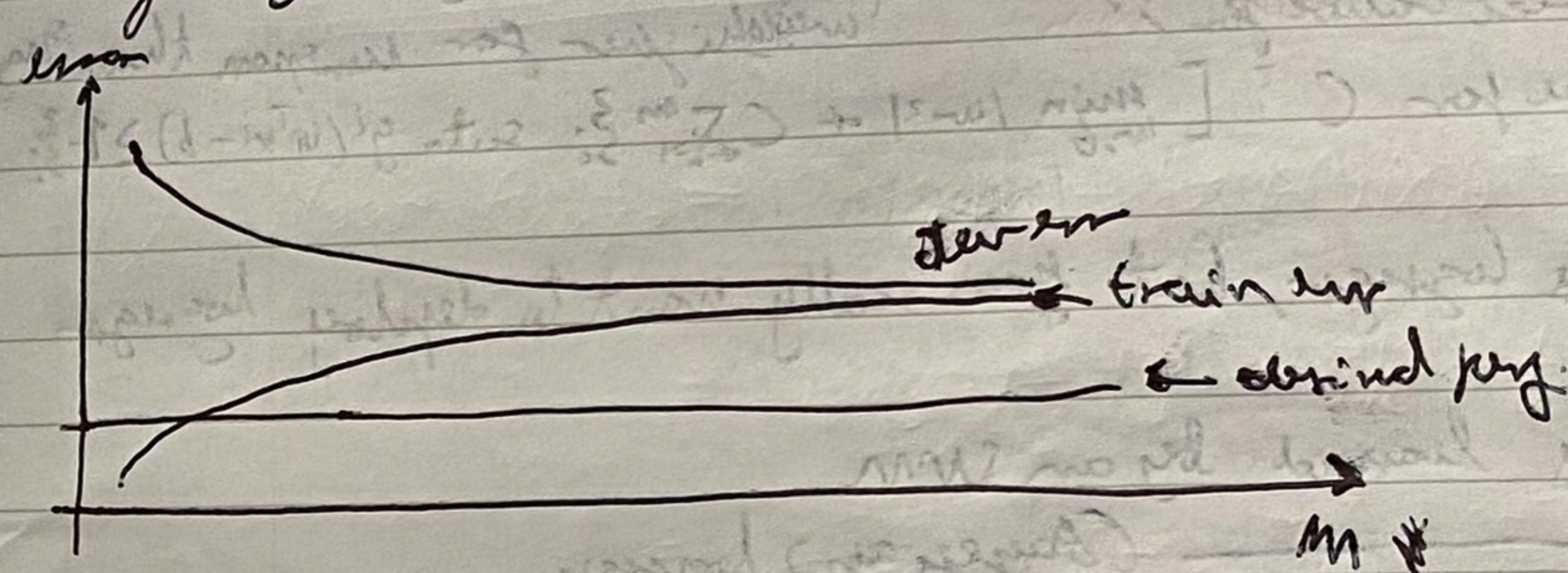
Well, suppose you suspect the problem is either overfitting (high variance) or too few features (high bias). \Rightarrow Diagnostic: Variance: test error $\approx 20\%$. While of bias: test error \approx train error.

Typically
High variance often gives such learning curves:



Note the large dev-train error gap for this high variance curve

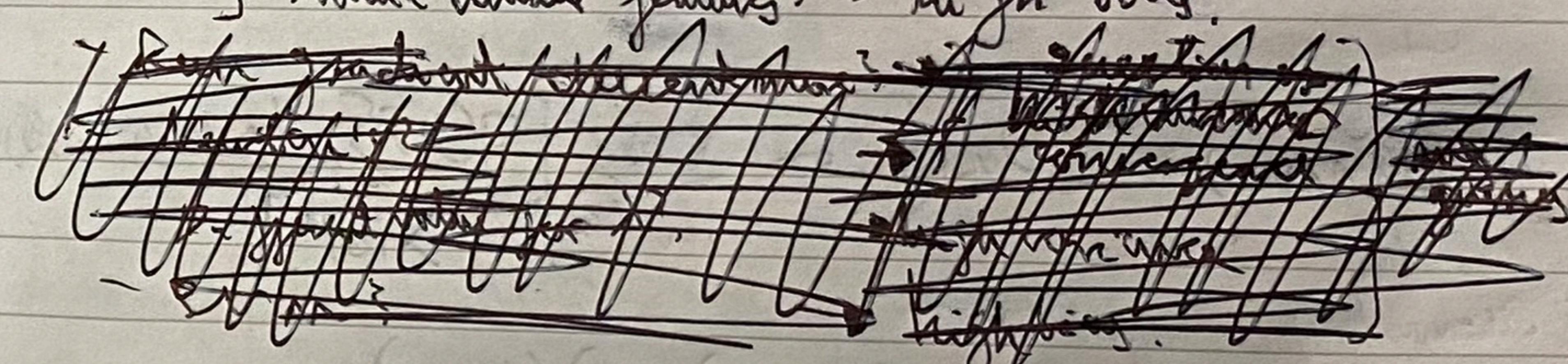
High bias:



Note: small gaps & even train error too high

Fixes to try:

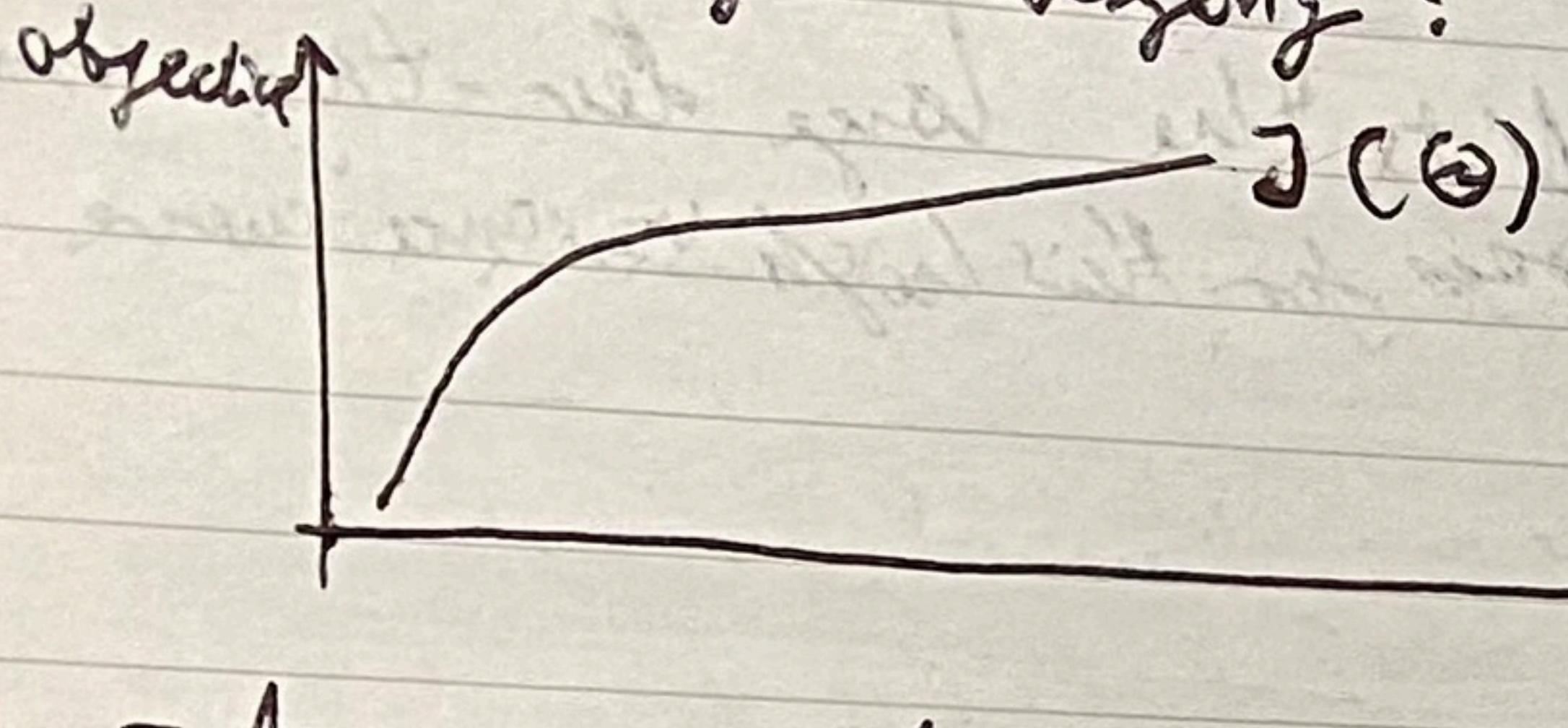
- More training examples? \rightarrow Yes, high variance
- Smaller set of features? \rightarrow ...
- Larger set of features? \rightarrow high bias, yes.
- Try email header features? \rightarrow high bias.



- Bias vs variance is one common diagnostic
- For other problems, it's up to your ingenuity to find your own diagnostic to figure out what's wrong.
- Another example:
 - Logreg gets 2% error on both spam & non-spam.
 - SVM w/ linear kernel gets 10% error on spam, and .01% error on nonspam.
 - But you want to use logreg for compute efficiency etc.
- What to do next?

Other common question:

- is the algo converging?



It's often very hard to tell if an algo has converged yet by looking at the objective

- Are you optimizing the right **gain** function?

$$\text{e.g., what you care about: } a(\theta) = \sum_i w_i \mathbb{I}\{h_\theta(x^i) = y^i\}$$

$$\text{Logistic algo? (correct value for } h_\theta \text{)} \quad \text{weight higher for non-zero than zero} \\ \text{SVM? (correct value for } C \text{)}: \min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \mathbb{I}\{y^i(w^T x^i - b) \geq 1\}$$

An SVM outperforms logistic, but you really want to deploy logistic for your application.

Let θ_{SVM} be parameters learned by an SVM

Let θ_{BLR} be — (Bayesian) logreg

You care about weighted acc $a(\theta) = \max_{\theta} \sum_i w_i \mathbb{I}\{h_\theta(x^i) = y^i\}$, and (as mentioned) $a(\theta_{SVM}) > a(\theta_{BLR})$

Two hypotheses: (1) Grad. descent simply isn't correctly optimizing $J(\theta)$, or (2) $J(\theta)$ [i.e., model] is a bad **gain** function [model ^{approximately} eq. J too large]

Diagnosis: Is $J(\theta_{SVM}) > J(\theta_{BLR})$, or $\sum_i \log p(y^i | x^i, \theta) -> 10^4$

Case 1:

$$a(\theta_{SVM}) > a(\theta_{BLR}) \quad \& \quad J(\theta_{SVM}) > J(\theta_{BLR}).$$

But: BLR was trying to minimize $J(\theta) \Rightarrow$ it fails to converge.

Problem lies in optimization algorithm.

Case 2:

$$a(\theta_{SVM}) > a(\theta_{BLR}) \\ J(\theta_{SVM}) \leq J(\theta_{BLR})$$

This means J is correctly minimized, but that $J(\theta)$ is the wrong way to use as θ_{SVM} does better on weighted accuracy $a(\theta)$.

\Rightarrow problem is with objective function itself.

Fixes to try: (old)

- Run gradient descent for more iterations. \rightarrow fixes optimisation algo.
- Try Newton's method!
- Use a different J : \rightarrow fixes optimisation objective
- Try using an SVM: \rightarrow

Machine Learning Algorithm: Stanford Helicopter Project

1. Build a simulator of ~~helicopter~~ helicopter

2. Choose a cost function. Say $J(\theta) = \|X - X_{desired}\|^2$ (X = helicopter pos.)

3. Run reinforcement learning (RL) algorithm to fly helicopter in sim, so as to try to minimize $\theta_{RL} = \arg \min J(\theta)$

Suppose you do this, and the resulting θ_{RL} gives worse performance than your human pilot. What next?

• Improve simulator? Modify cost function J ? Modify RL algo?

Ok. Well, suppose that: 1. The helicopter sim is accurate. 2. The RL algo correctly controls the heli in sim so as to minimize $J(\theta)$

3. Minimizing $J(\theta)$ corresponds to correct autonomous flight

Then: The learned parameters θ_{RL} should fly well on the actual helicopter.

Diagnosis:

1. If θ_{RL} flies well in sim but not real life \Rightarrow simulator is ~~the prob.~~

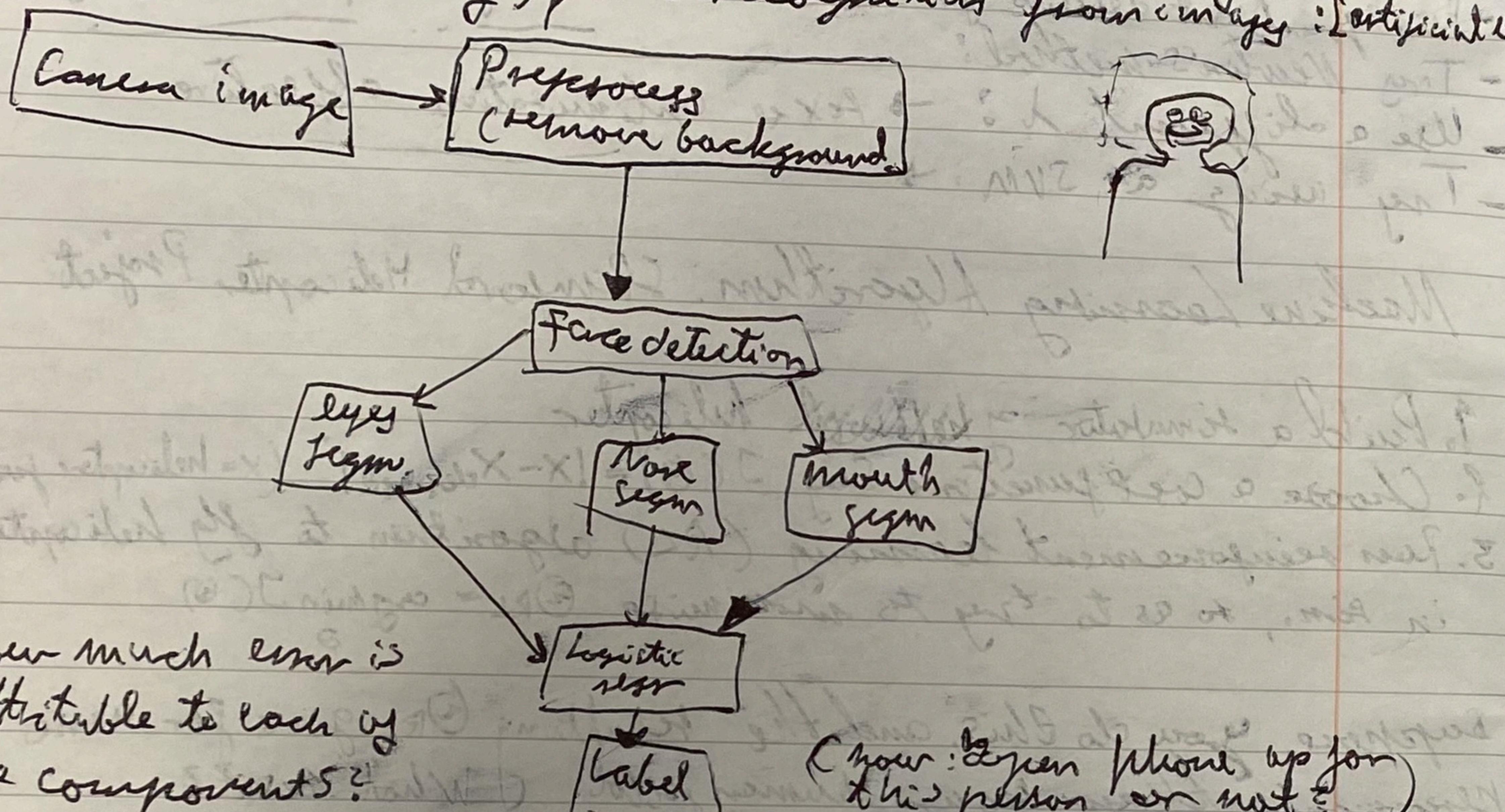
Otherwise:

2. Let θ_{human} be the human control policy. If $J(\theta_{human}) \leq J(\theta_{RL})$, then the problem is in the reinforcement learning algo. (Failing to minimize the cost function J .)

3. If $J(\theta_{human}) > J(\theta_{RL})$, then the problem is in the cost function. (Minimizing it doesn't correspond to good autonomous flight).

Error Analysis: explain difference between where you are and ~~where you want to be~~ rejections

Many applications combine many different learning components into a "pipeline". Eg, face recognition from image: [artificially]



How much error is attributable to each of the components?

Plug in ground-truth for lack component, and see how acc. changes.

Component	Accuracy
Overall sys.	85%
Preprocess	+1% → 86% F1 MA → 86.17%
Face det.	+1% → 91%
Eyes segm.	+1% → 95%
Nose	+1% → 96%
Mouth	+1% → 97%
Log. reg	→ 100%

Conclusion: in this example, most room for improvement in face detection and eyes segm.

Nice way to isolate different factors!

Tips: whenever you find weird/unexpected behaviour → never ignore! it's gold!

Ablative analysis: explain cliff between where you are and something much worse.

→ Remove components one at a time to see how it affects.

Ex. for spam classifier:

Component	Accuracy
Overall sys.	99.9%
Spelling corr.	99.8%
Header host feats.	98.9%
Email header feat.	98.7%
Email text pause feats.	95%
JavaScript pause	99.5%
Feature from image	94%

Cumulative removal (in this case, but in iso. also possible)

Conclusion: The email text pause feats. account for most of the improvement.

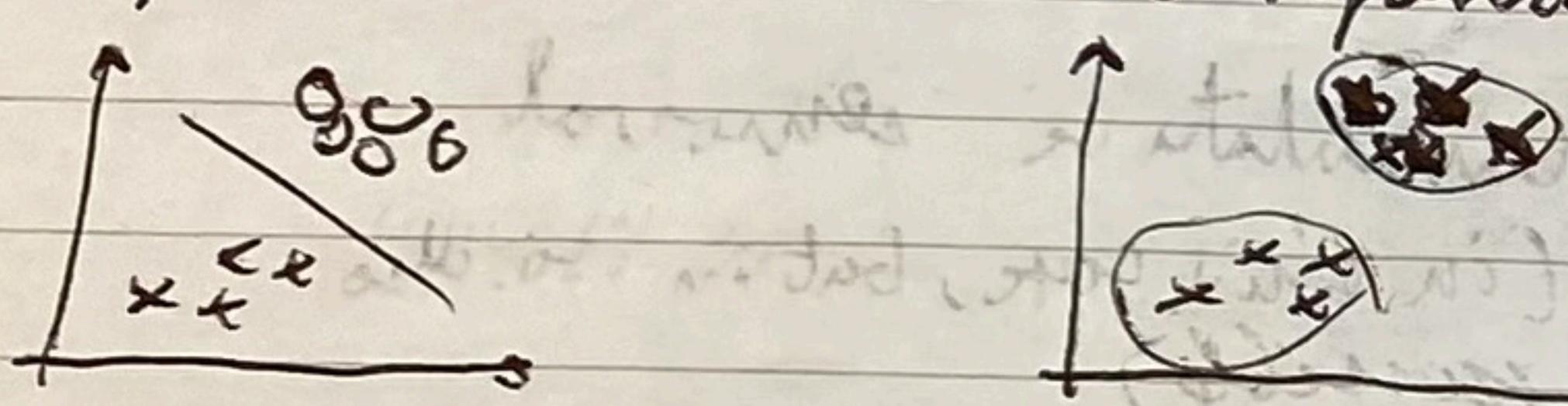
Lecture 14

Topic:

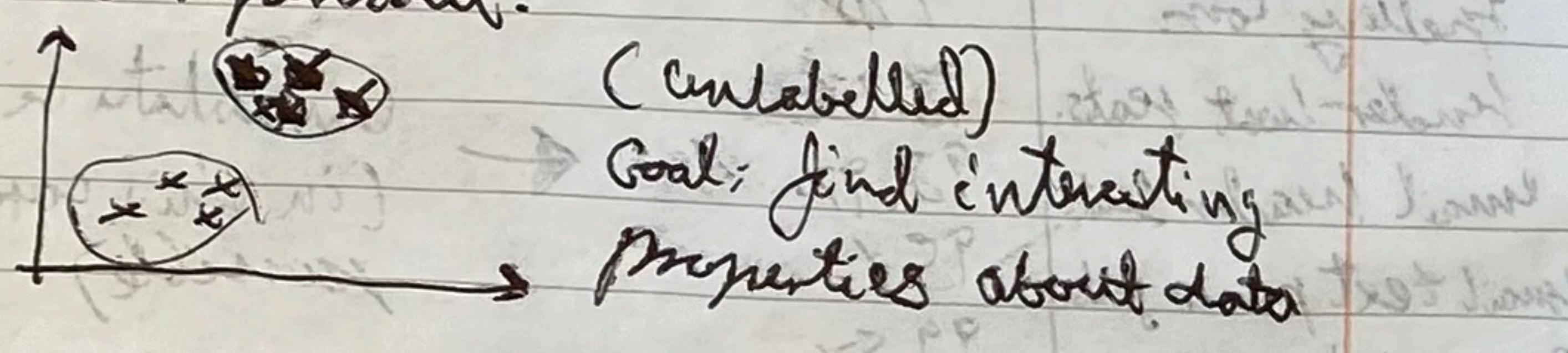
Unsupervised learning

- K-means clustering
- Mixture of Gaussians
- Expectation maximization (EM) algorithm & derivation

Supervised:



Unsupervised:



K-means clustering algo:

Data $\{x^1, \dots, x^m\}$

1. Initialize cluster centroids $\mu_1, \dots, \mu_k \in \mathbb{R}^n$ randomly

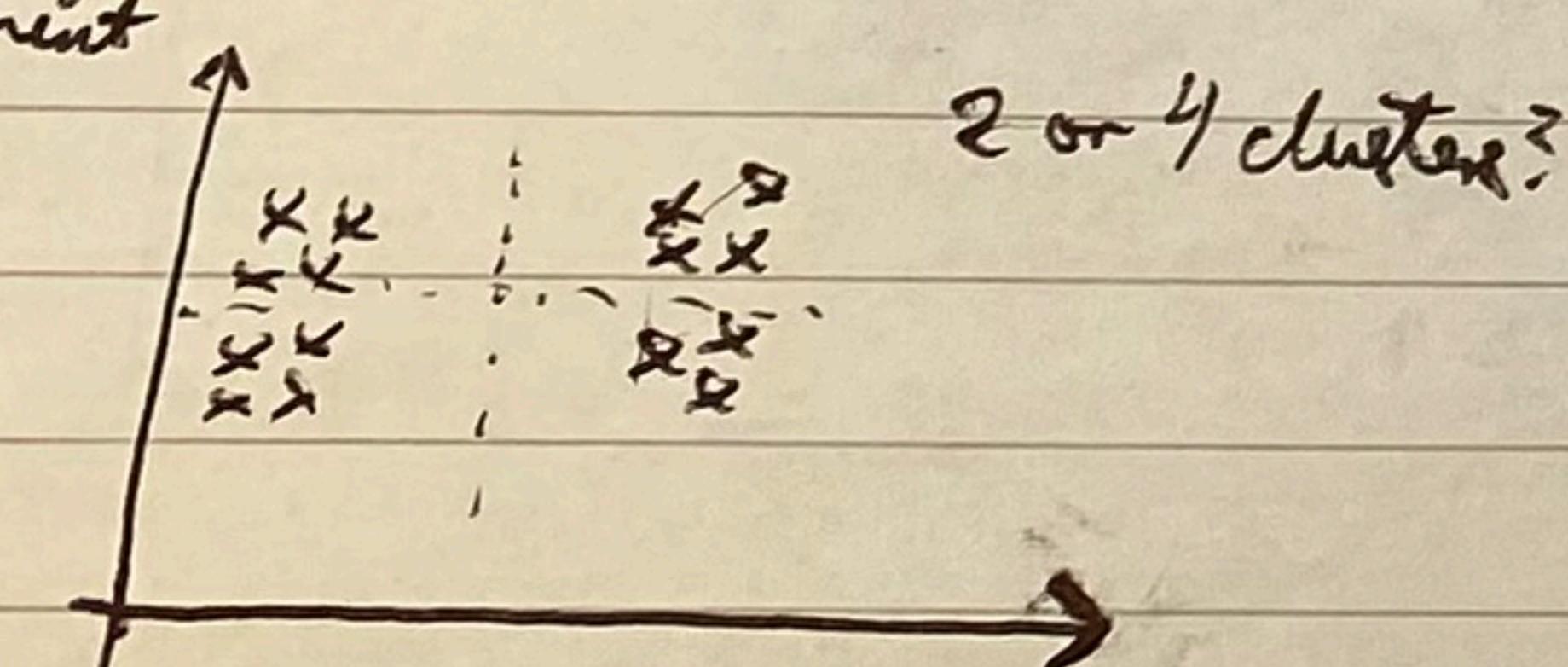
2. Repeat until convergence:

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

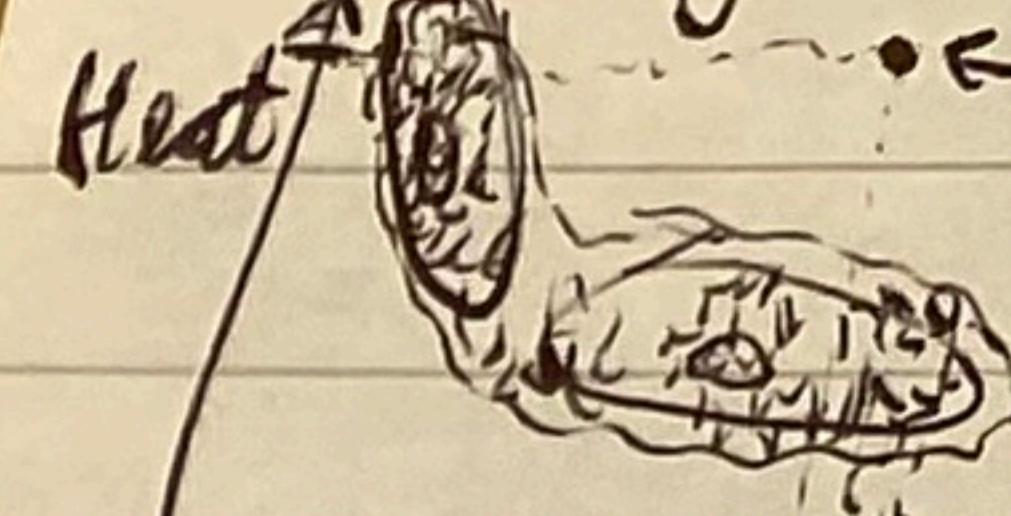
b) For $j = 1, \dots, k$

$$\mu_j := \frac{\sum_{i=1}^m I\{c^i=j\} x^i}{\sum_{i=1}^m I\{c^i=j\}}$$

$J(c, \mu) = \sum_{i=1}^m \|x^i - \mu_{c^i}\|^2$ - This cost function is strictly decreasing



Density estimation



Anomaly detection

Vibration

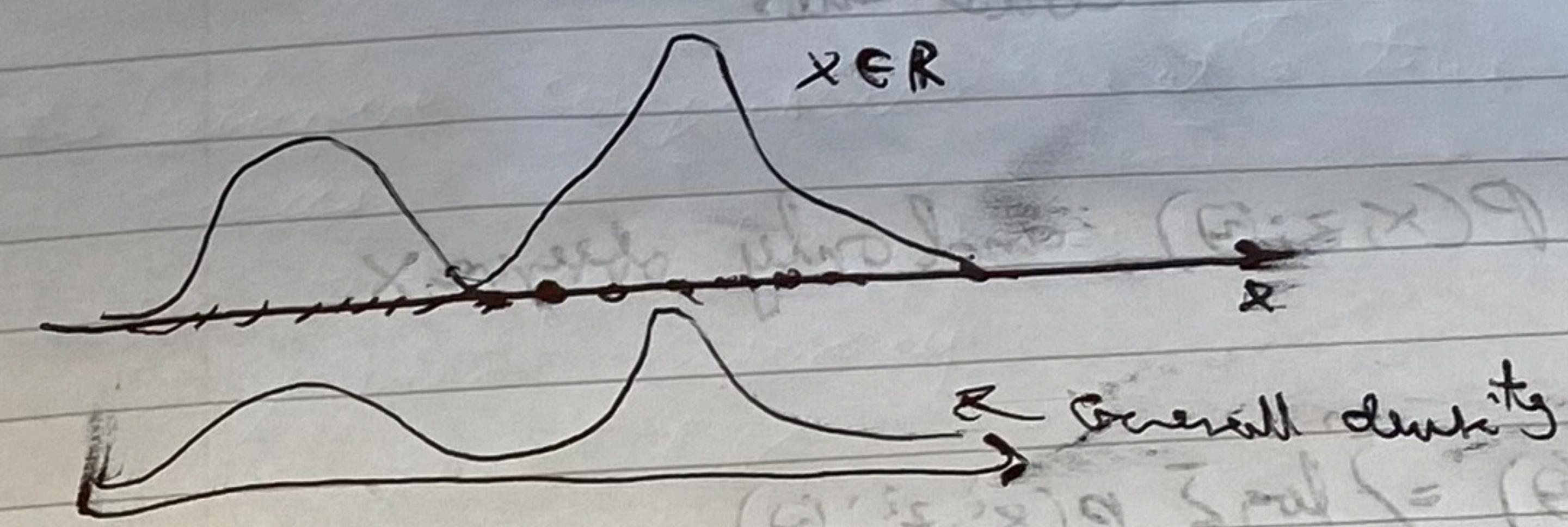
Model $P(x) < \varepsilon \rightarrow$ anomaly!

$\rightarrow \varepsilon \rightarrow$ looks ok

Looks like data comes from two Gaussians

\Rightarrow Let's use the "mixture of Gaussians" method.

1D example



Mixture of Gaussians model:

Suppose there's a latent (hidden/unobserved) random variable z_i , and x^i, z^i are distributed $P(x^i, z^i) = P(x^i|z^i)P(z^i)$, where

$z^i \sim \text{Multinomial}(\phi), z^i \in \{1, \dots, k\}$

$$(x^i | z^i=j) \sim N(\mu_j, \Sigma_j)$$

If we knew z^i , can use MLE: $\ell(\phi, \mu, \Sigma) = \sum_{i=1}^m \log p(x^i, z^i; \phi, \mu, \Sigma)$

$$\Rightarrow \phi_j = \frac{1}{m} \sum_{i=1}^m I\{z^i=j\} \quad \& \quad \mu_j = \frac{\sum_{i=1}^m I\{z^i=j\} x^i}{\sum_{i=1}^m I\{z^i=j\}} \quad \& \quad \Sigma = \dots$$

EM:

E-step: (Guess value of z^i): Set $w_{ij} = p(z^i=j | x^i; \phi, \mu, \Sigma)$
 $= p(x^i | z^i=j)P(z^i=j) / \sum_{l=1}^k p(x^i | z^i=l)P(z^i=l)$

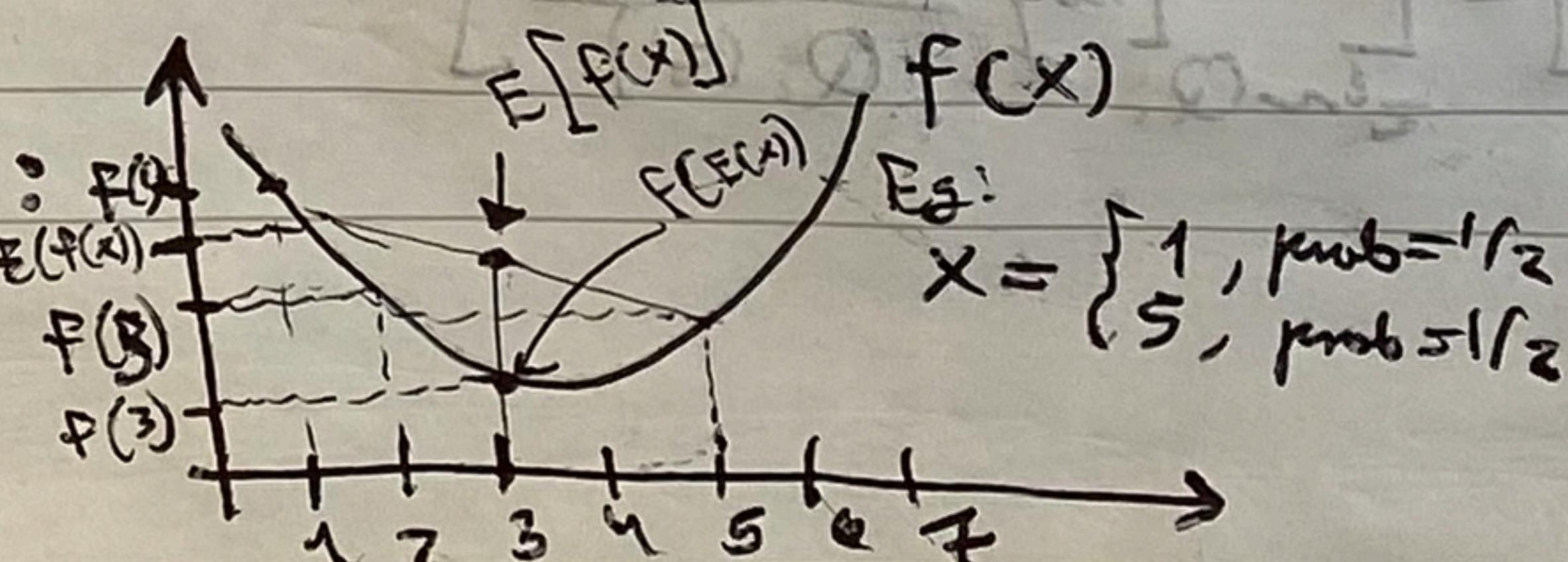
$$\pi(\mu_j, \Sigma_j) = \frac{\det(\Sigma_j)^{-\frac{1}{2}} (\det(\Sigma))^{\frac{1}{2}} \prod_{i=1}^m w_{ij}^{-\frac{1}{2}}}{(2\pi)^{\frac{n}{2}} |\Sigma_j|^{\frac{1}{2}}} \quad \& \quad \phi_j, \Sigma_j \text{ multinomial}(\phi)$$

$$\text{M-step: } \phi_j := \frac{1}{m} \sum_{i=1}^m w_{ij} \quad \& \quad \mu_j = \frac{\sum_{i=1}^m w_{ij} x^i}{\sum_{i=1}^m w_{ij}} \quad \left(\sum_{i=1}^m I\{z^i=j\} \rightarrow w_{ij} = E[I\{z^i=j\}] \right)$$

Jensen's inequality

Let f be a convex function ($f''(x) > 0$) & X be a RV; then $E[f(X)] \leq f(E[X])$.

Example:



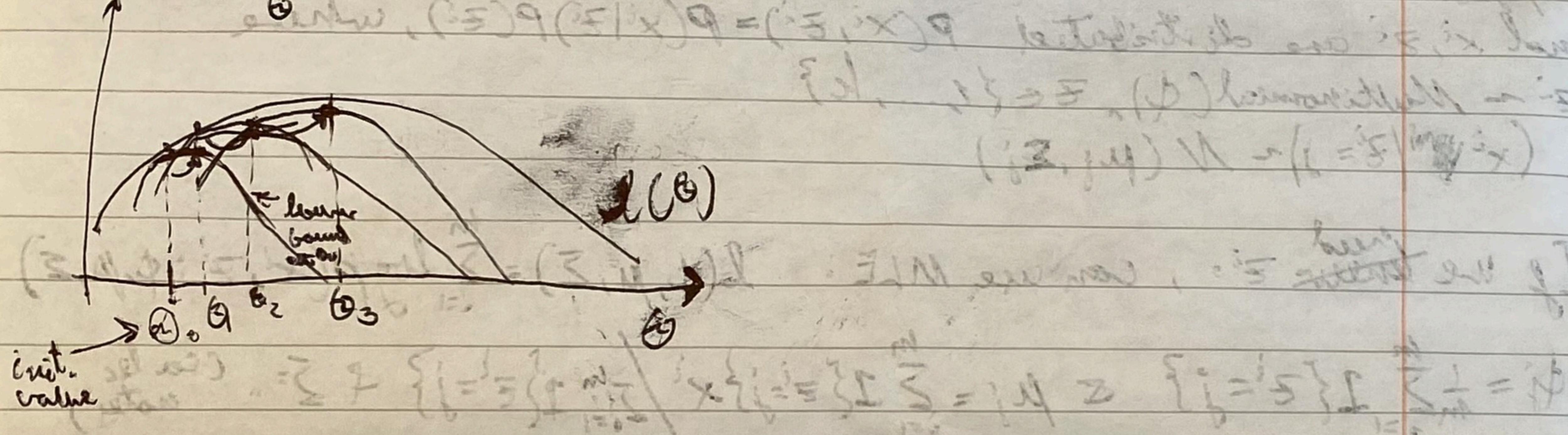
$$\Rightarrow E[f(x)] = \frac{1}{2}[f(1) + f(5)] = 3 > f(3)$$

Further, if $f''(x) > 0$ (f is strictly convex), then
 $E[f(x)] = f(E(x))$ only if x is a constant.

Have a model for $P(x, z; \theta)$, and only observe x .
 $\{x^1, \dots, x^m\}$

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

Want: $\arg\max_{\theta} L(\theta)$ (maximizes likelihood) to estimate θ
 $\text{curv. } P(z|x) = P(z, x)/P(x)$ $\text{likeli. } P(z|x) = P(x|z)$



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

$$= \sum_i \log \left[Q_i(z^i) \frac{P(x^i, z^i; \theta)}{Q_i(z^i)} \right], \text{ where } Q_i(z^i) \text{ is a pdf}$$

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

$$\text{Jensen's inequality} \quad \leq \sum_i E_{z^i \sim Q_i} \left[\log \frac{P(x^i, z^i; \theta)}{Q_i(z^i)} \right] = \sum_i \sum_{z^i} Q_i(z^i) \log \frac{P(x^i, z^i; \theta)}{Q_i(z^i)}$$

On a given iteration of EM (with parameters θ), we want:

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

for this to hold, we need $\frac{P(x^i, z^i)}{Q_i(z^i)} = \text{constant}$.

Set $Q_i(z^i) \propto P(x^i, z^i; \theta)$, $Q_i(z^i) = P(x^i, z^i; \theta)$

$$\sum_{z^i} Q_i = 1, Q_i(z^i) = \frac{P(x^i, z^i; \theta)}{\sum_{z^i} P(x^i, z^i; \theta)} = \dots = p(z^i | x^i; \theta)$$

E-step: set $Q_i(z^i) = p(z^i | x^i; \theta)$

$$M\text{-step: } \theta_t = \arg\max_{\theta} \sum_i \sum_{z^i} Q_i(z^i) \log \frac{P(x^i, z^i; \theta)}{Q_i(z^i)}$$

Lecture 15

Outline:

- EM Convergence
- Gaussian properties
- Factor analysis
- Gaussian marginals & conditions
- EM steps

$$P(x^i, z^i) = P(x^i | z^i) P(z^i), \quad z^i \sim \text{Multinomial}(\phi) \\ [P(z^i=j) = \phi_j]$$

$$x^i | z^i = j \sim N(\mu_j, \Sigma_j)$$

$$\text{E-step: } Q_i(z^i=j) = P(z^i=j | x^i; \phi, \mu, \Sigma) = w_j \\ \text{if } P(z^i=j) \text{ is } \sum_{j=1}^J w_j$$

$$\text{M-step: } \max_{\phi, \mu, \Sigma} \sum_i \sum_{z^i} Q_i(z^i) \log \frac{P(x^i, z^i; \phi, \mu, \Sigma)}{Q_i(z^i)} \\ = \sum_i \sum_j w_j \log \frac{(2\pi)^{n/2} |\Sigma_j|^{1/2}}{(2\pi)^{m/2} |\Sigma|^{1/2}} \exp \left[-\frac{1}{2} (x^i - \mu_j) \Sigma_j^{-1} (x^i - \mu_j) \right] \delta_j \\ \rightarrow P(x^i | z^i, \phi, \mu, \Sigma)$$

$$\nabla_{\mu_j} (\dots) = 0 \Rightarrow \mu_j = \frac{\sum_i w_j x^i}{\sum_i w_j} \quad \text{weight of } x^i \text{ @ Gaussian } j$$

$$\nabla_{\phi} (\dots) = 0 \Rightarrow \phi_j = \frac{\sum_i w_j}{\sum_i w_j}$$

Note: Equivalent interp:

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

We know $J(\theta) \geq J(\theta, Q)$ for any θ, Q (Jensen's inequality)

E-step: Maximise J wrt Q

M-step: Maximise J wrt θ

Recap:

$$\text{E-step: } Q_i(z^i) := P(z^i | x^i; \theta)$$

$$\text{M-step: } \theta := \arg \max_{\theta} \sum_i \sum_{z^i} Q_i(z^i) \log \frac{P(x^i, z^i; \theta)}{Q_i(z^i)}$$

$$P(x^i, z^i) = P(x^i | z^i) P(z^i), \quad z^i \sim \text{Multinomial}(\phi) \\ [P(z^i=j) = \phi_j]$$

$$x^i | z^i = j \sim N(\mu_j, \Sigma_j)$$

$$\text{E-step: } Q_i(z^i=j) = P(z^i=j | x^i; \phi, \mu, \Sigma) = w_j \\ \text{if } P(z^i=j) \text{ is } \sum_{j=1}^J w_j$$

$$\text{M-step: } \max_{\phi, \mu, \Sigma} \sum_i \sum_{z^i} Q_i(z^i) \log \frac{P(x^i, z^i; \phi, \mu, \Sigma)}{Q_i(z^i)} \\ = \sum_i \sum_j w_j \log \frac{(2\pi)^{n/2} |\Sigma_j|^{1/2}}{(2\pi)^{m/2} |\Sigma|^{1/2}} \exp \left[-\frac{1}{2} (x^i - \mu_j) \Sigma_j^{-1} (x^i - \mu_j) \right] \delta_j \\ \rightarrow P(x^i | z^i, \phi, \mu, \Sigma)$$

$$\nabla_{\mu_j} (\dots) = 0 \Rightarrow \mu_j = \frac{\sum_i w_j x^i}{\sum_i w_j} \quad \text{weight of } x^i \text{ @ Gaussian } j$$

$$\nabla_{\phi} (\dots) = 0 \Rightarrow \phi_j = \frac{\sum_i w_j}{\sum_i w_j}$$

Note: Equivalent interp:

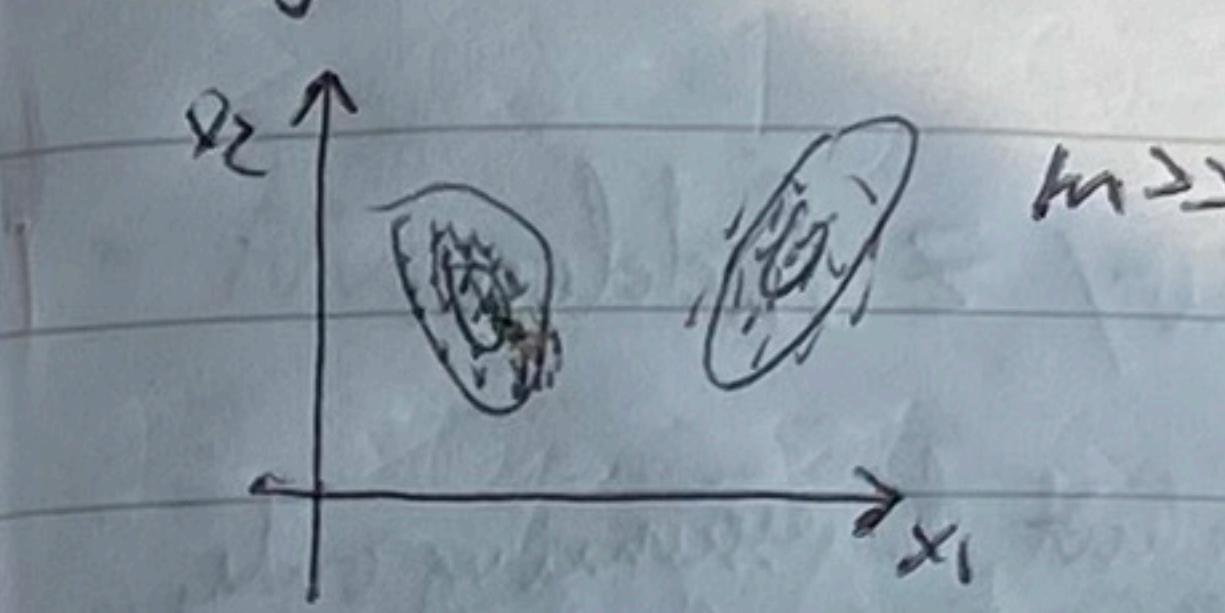
$$J(\theta, Q) = \sum_i \sum_{z^i} Q_i(z^i) \log \frac{P(x^i, z^i; \theta)}{Q_i(z^i)}$$

We know $J(\theta) \geq J(\theta, Q)$ for any θ, Q (Jensen's inequality)

Factor Analysis Algorithm

Mixture of Gaussians:

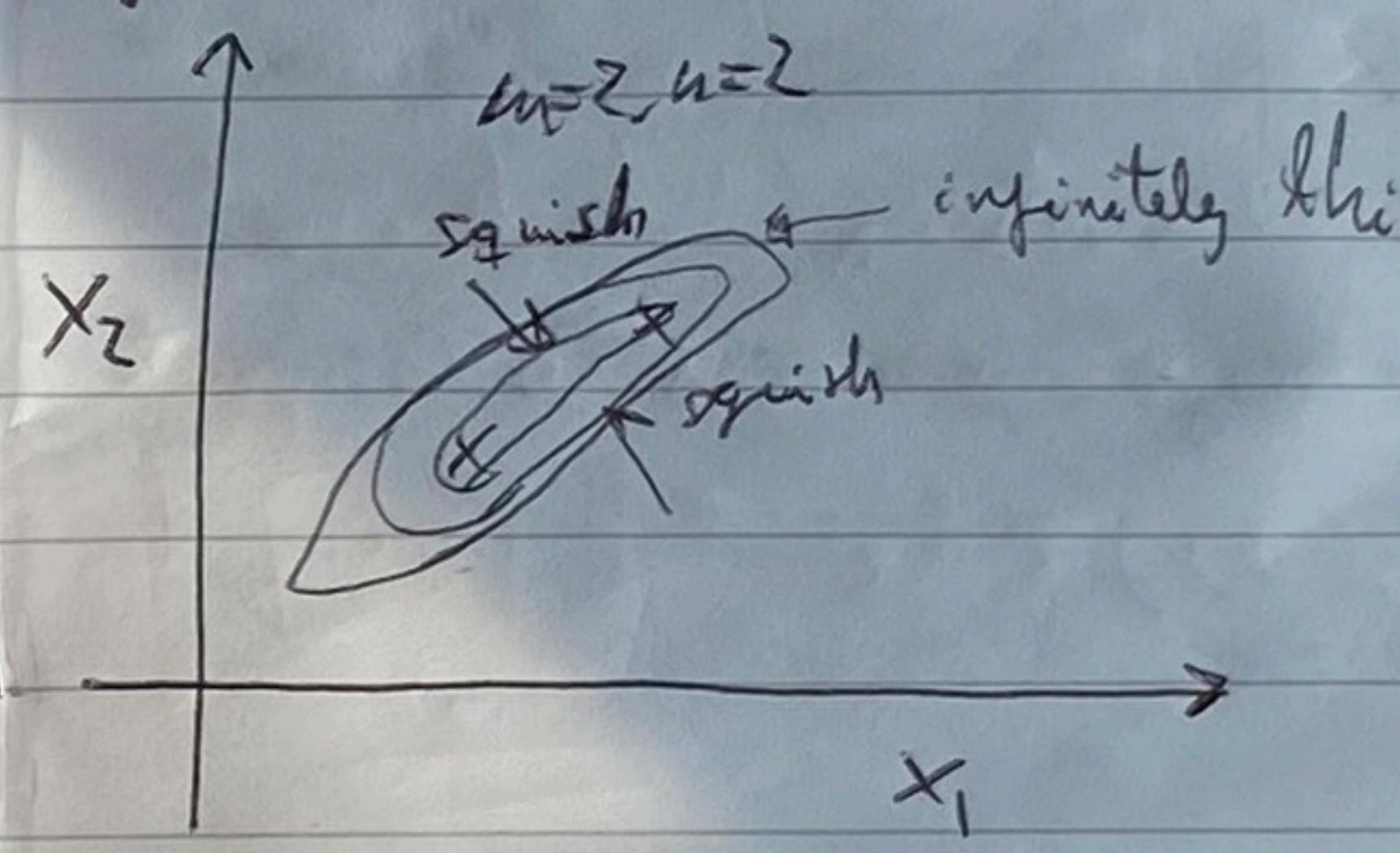
say $n=2, m=100$



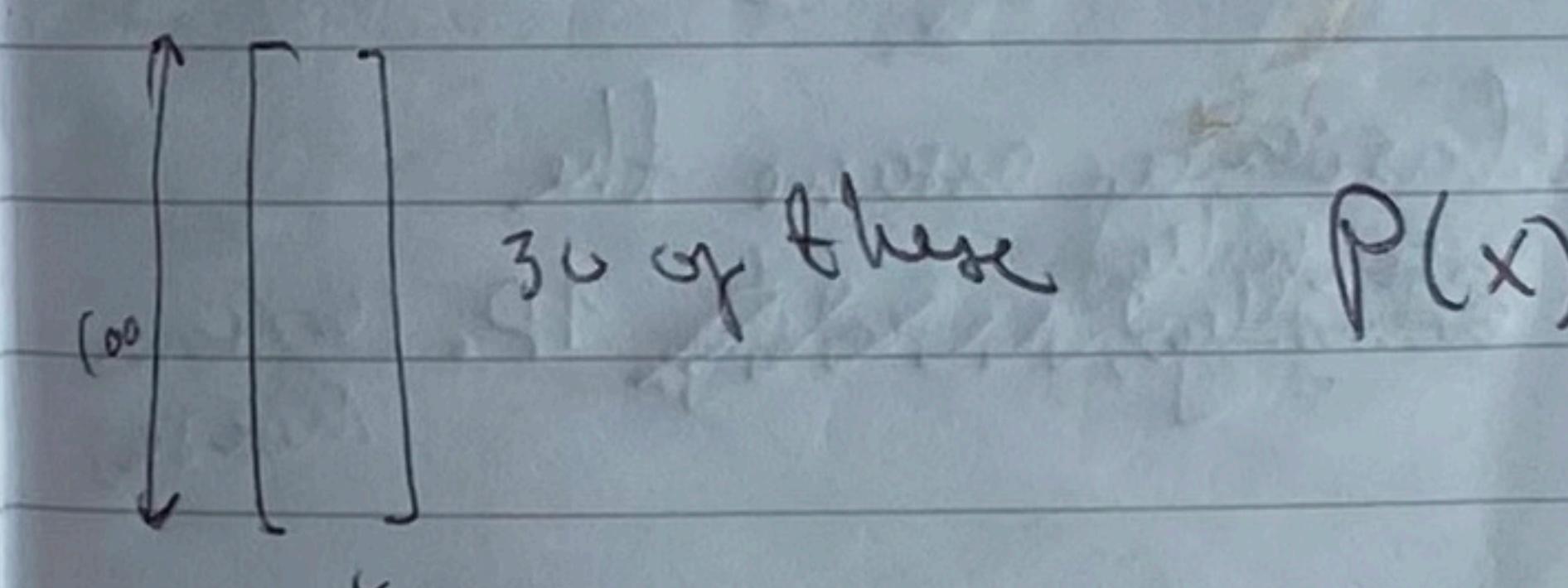
Model as single Gaussian: $X \sim N(\mu, \Sigma) \Rightarrow \text{MLE: } \mu = \frac{1}{m} \sum_i x^i$

$\Sigma = \frac{1}{m} \sum_i (x^i - \mu)(x^i - \mu)^T$

If $m \leq n$, then Σ will be singular. But $N(\mu, \Sigma) \propto \exp[-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu)]$



100 psychological attributes, 30 people



Option 1: constrain Σ to be diagonal: $\Sigma = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$

$\Leftrightarrow \leftrightarrow \oplus \rightarrow$ or $\leftarrow \oplus \rightarrow$. but not $\circlearrowleft \oplus \rightarrow$

$\Rightarrow \text{MLE gives } \sigma_j^2 = \frac{1}{m} \sum_{i=1}^m (x_j^i - \mu_j)^2 \text{ for all } m \text{ rows } (j=1, 2, \dots, n)$

But assuming everything is uncorrelated isn't exactly smart...
Real world usually.

Option 2: constrain Σ to be $\Sigma = \sigma^2 I \Leftrightarrow$ or

~~MLE~~: $\hat{\sigma}^2 = \frac{1}{mn} \sum_{i,j} (x_{ij} - \mu_i)^2$. Very bad simplistic model.

Factor Analysis model: $P(X, z) = P(x|z)P(z)$ w/ z hidden.

$z \sim N(0, I)$, $z \in \mathbb{R}^d$, $d < n$. e.g.: $\begin{matrix} d=3 \\ n=100 \\ m=30 \end{matrix}$

Assume $x = \mu + \Lambda z + \epsilon$, where $\epsilon \sim N(0, \Sigma)$

Parameters: $\mu \in \mathbb{R}^n$, $\Lambda \in \mathbb{R}^{n \times d}$, $\Sigma \in \mathbb{R}^{n \times n}$ & diagonal.

Equivalently, $x|z \sim N(\mu + \Lambda z, \Sigma)$.

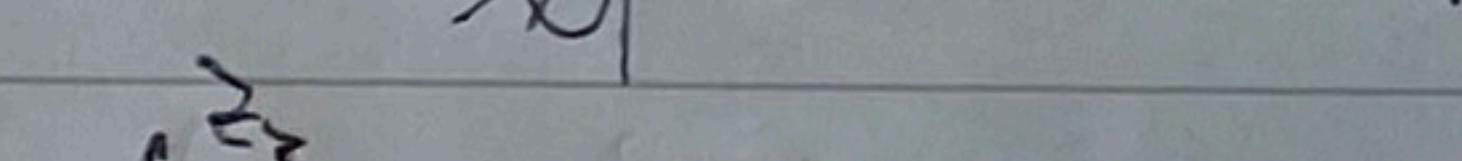
Examples:

$z \in \mathbb{R}^1$, $x \in \mathbb{R}^m \Rightarrow d=1$ & $m=7$



Say $\Lambda = \begin{bmatrix} z \\ 1 \end{bmatrix}$, $\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow \Lambda z + \mu \in \mathbb{R}^2$. And $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$

\Rightarrow Here we basically assume the data is ^{essentially} 1D (w/some noise)

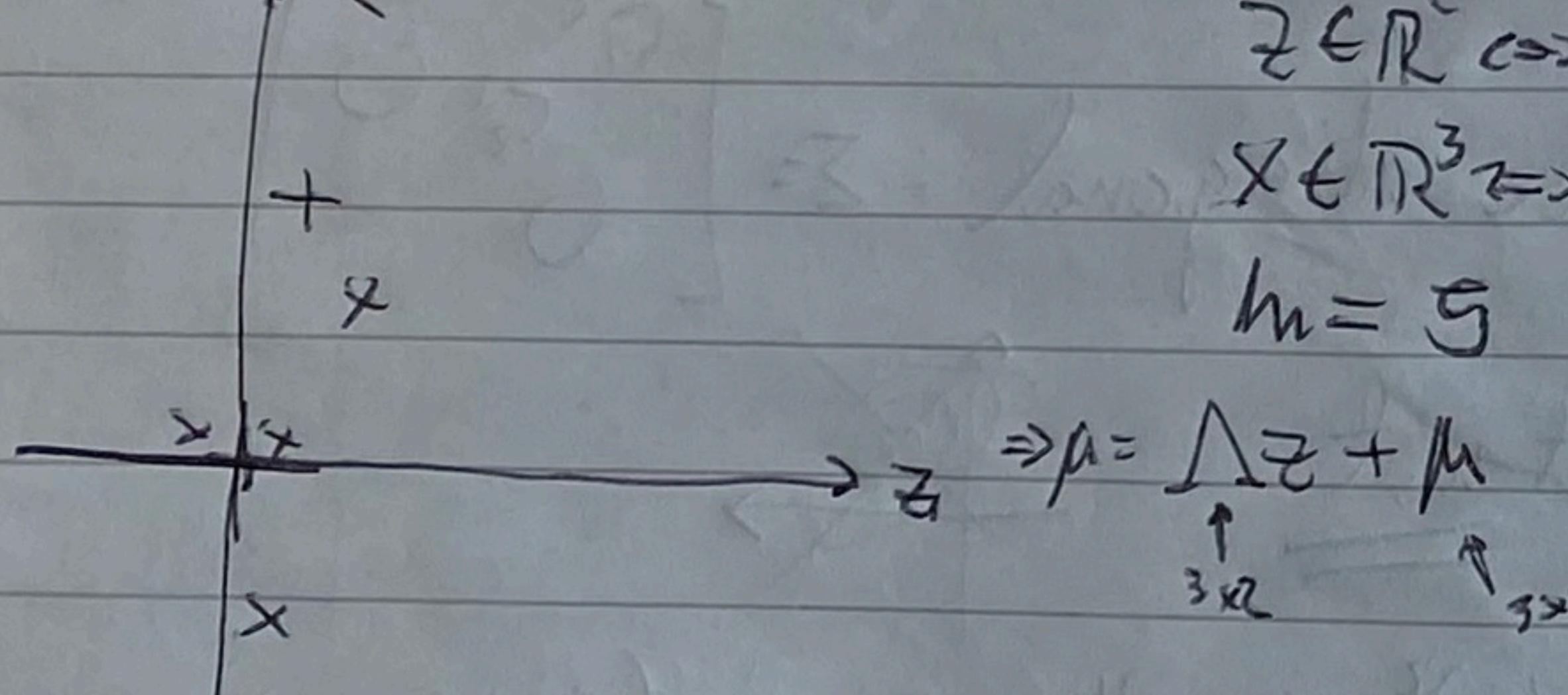


$z \in \mathbb{R}^2 \Leftrightarrow d=2$

$x \in \mathbb{R}^3 \Leftrightarrow m=3$

$n=5$

Here we're assuming that the data is really just 2D (w/some noise)



$\mu = \Lambda z + \mu$

Σ_{zz}

Σ_{zx}

Σ_{xz}

Σ_{xx}

Multivariate Gaussian

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \stackrel{r}{\uparrow} \quad x_1 \in \mathbb{R}^r, x_2 \in \mathbb{R}^s \Rightarrow x \in \mathbb{R}^{r+s}$$

$$x \sim N(\mu, \Sigma) \Rightarrow \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \quad \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} \in \mathbb{R}^{r+s \times r+s}$$

Marginal: $p(x_1) = ?$ Well, $p(x) = p(x_1, x_2)$ and $\int p(x_1, x_2) dx_2 = p(x_1)$.
 $\int \frac{\exp\left(\frac{(x_1 - \mu_1)}{\sqrt{2\pi}}\right) \left(\frac{(x_2 - \mu_2)}{\sqrt{2\pi}}\right)^T \left[\begin{matrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{matrix}\right]}{(2\pi)^{rc} |\Sigma|^{1/2}} = p(x_1, x_2)$, then it turns out

that $x_1 \sim N(\mu_1, \Sigma_{11})$
~~Because~~

Conditional: $p(x_1|x_2) = ?$ Well, $p(x_1, x_2)/p(x_2)$ and $x_1|x_2 \sim N(\mu_{12}, \Sigma_{12})$

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

$$\Sigma_{1|x_2} = \Sigma_{11} - \Sigma_{12} \Sigma_{22}^{-1} \Sigma_{21}$$

Now we're ready to derive the EM algo

1. Derive $p(x|z)$:

$$(x|z) \sim N(\mu_{x,z}, \Sigma) \cdot [z \sim N(0, I)] \stackrel{\text{recall}}{\Rightarrow} E[z] = 0 \quad E[x] = \mu$$

$$\Rightarrow \mu_{x,z} = \begin{bmatrix} 0 \\ \mu \end{bmatrix} \stackrel{d}{\uparrow} \quad \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} = \begin{bmatrix} E[(z-0)(z-0)^T] & E[(z-0)(x-\mu)^T] \\ E[(x-\mu)(z-0)^T] & E[(x-\mu)(x-\mu)^T] \end{bmatrix}$$

$$\Rightarrow \Sigma_{zz} = E[(x-\mu)(x-\mu)^T] = E[((\Lambda z + \mu + \epsilon)(\Lambda z + \mu + \epsilon)^T)]$$

$$= E[\Lambda z z^T \Lambda^T + \Lambda z \epsilon^T + \epsilon z^T \Lambda^T + \epsilon \epsilon^T] = E[\Lambda z z^T \Lambda^T + \epsilon \epsilon^T]$$

$$= \cancel{\Lambda z z^T \Lambda^T} \underbrace{\Lambda E[z z^T]}_{=\Sigma} + \epsilon \epsilon^T = \Lambda \Lambda^T + \Sigma$$

$$\Rightarrow \Sigma = \begin{bmatrix} \Sigma & \Lambda^T \\ \Lambda & \Lambda \Lambda^T + \Sigma \end{bmatrix}$$

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

E-step:

~~compute:~~
 $Q_i(z^i) = P(z^i | x^i; \Theta)$

$$z^i | x^i \sim N(\mu_{z^i|x^i}, \Sigma_{z^i|x^i}), \text{ where } \mu_{z^i|x^i} = \bar{\mu} + \Lambda^T (\Lambda \Lambda^T + \Sigma)^{-1} x^i$$

$$\Sigma_{z^i|x^i} = \Sigma - \Lambda (\Lambda \Lambda^T + \Sigma)^{-1} \Lambda$$

M-step:

We know that $Q_i(z^i) = \frac{\exp(-\frac{1}{2}(z^i - \mu_{z^i|x^i})^T \Sigma_{z^i|x^i}^{-1} (z^i - \mu_{z^i|x^i}))}{(2\pi)^{n/2} |\Sigma|^{1/2}}$. (in the actual derivation (which isn't covered fully here), one needs to compute terms like $\int_{\mathbb{R}^n} Q_i(z^i) z^i dz^i = \int_{\mathbb{R}^n} \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp(-\frac{1}{2}(z^i - \mu_{z^i|x^i})^T \Sigma_{z^i|x^i}^{-1} (z^i - \mu_{z^i|x^i})) z^i dz^i = E_{z^i \sim Q_i}[z^i]$)

$$= \mu_{z^i|x^i}$$

$$\Theta := \underset{\Theta}{\operatorname{argmax}} \sum_i \int_{\mathbb{R}^n} Q_i(z^i) \log \frac{P(x^i, z^i)}{Q_i(z^i)} dz^i = \sum_i E_{z^i \sim Q_i} \left[\log \frac{P(x^i, z^i)}{Q_i(z^i)} \right]$$

This whole thing will be quad, and when doing $\nabla_{\mu_{z^i|x^i}}$ you'll only face quadratic complexity.

Independent Components Analysis (Lec 16)

- ICA model

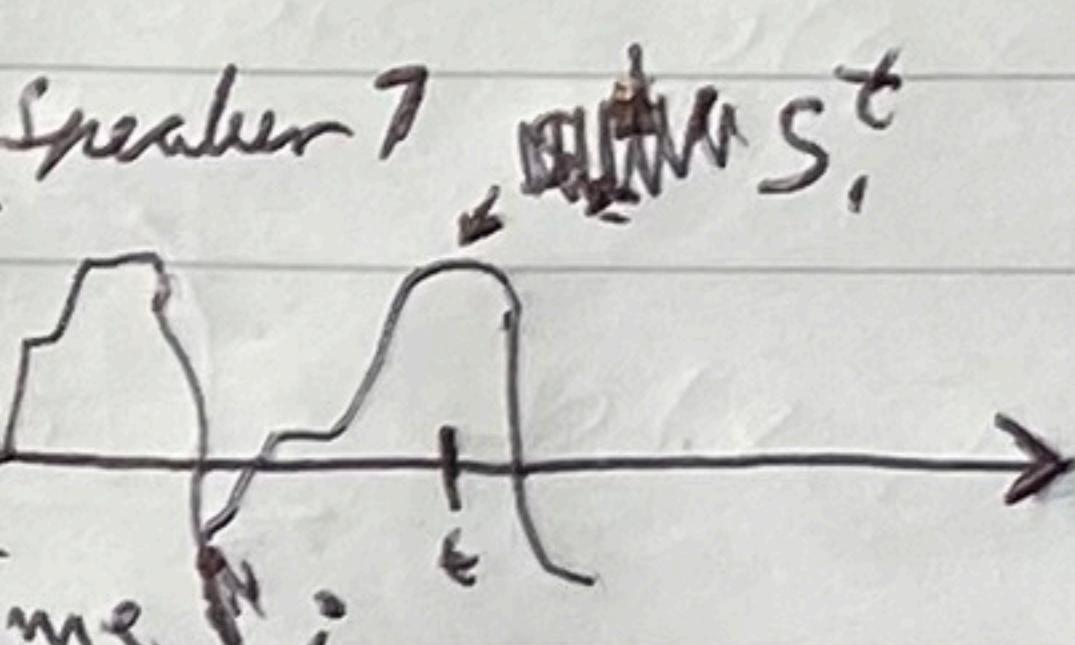
Reinforcement learning

- MDP (Markov decision process)

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

$$s_j^i = \text{Speaker } j \text{ at time } t^i$$

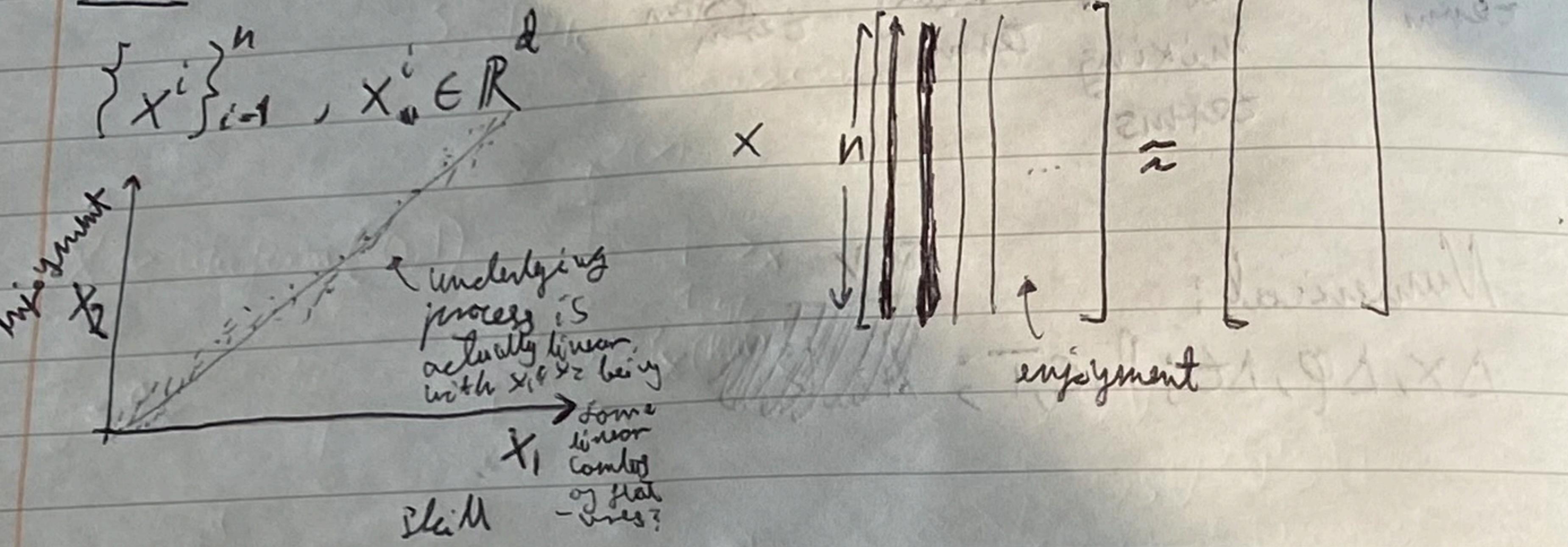
$$x = As^i, x \in \mathbb{R}^n$$



Goal: Find $w = A^{-1}$, so $s^i = w x^i \mid w = \begin{bmatrix} w_1^T \\ \vdots \\ w_n^T \end{bmatrix}$

Principal & Independent CA (lec 15.5)

PCA

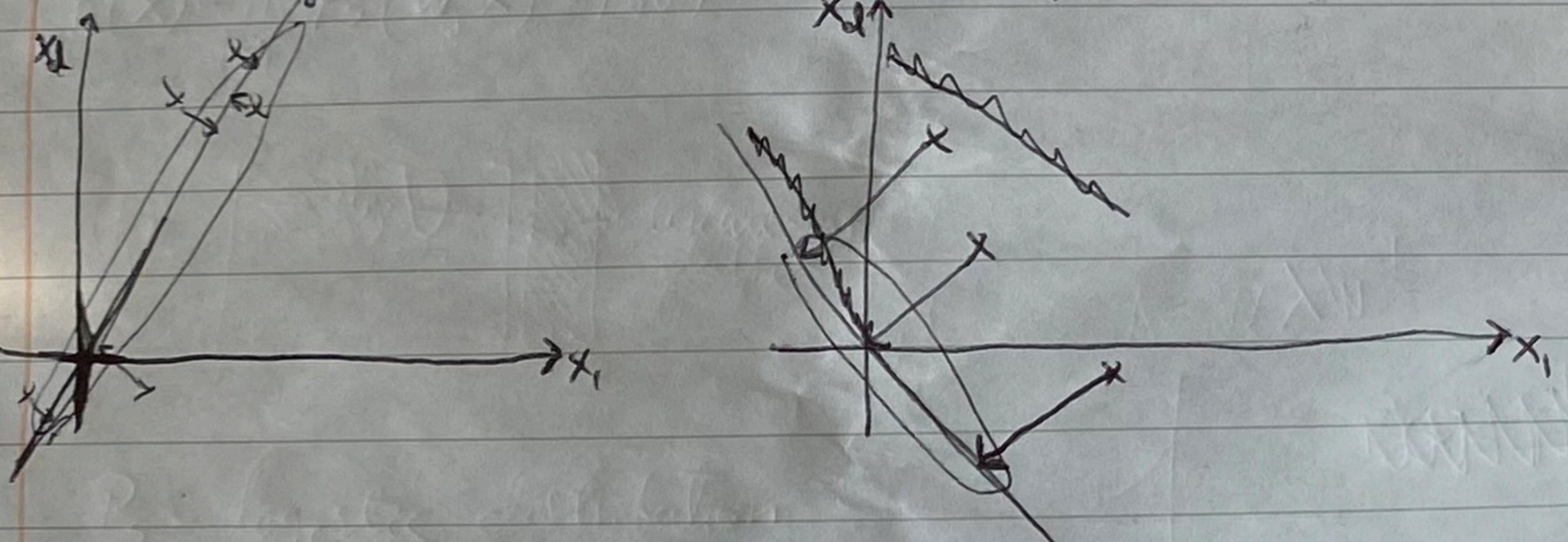


Standardize

$$x_j^i \leftarrow \frac{x_j^i - \mu_j}{\sigma_j} \quad \text{mean & std. of } j\text{-th column.}$$

$$\mu_j = \frac{1}{n} \sum_{i=1}^n x_j^i$$

$$\sigma_j^2 = \frac{1}{n} \sum_{i=1}^n (x_j^i - \mu_j)^2$$



$u \in \mathbb{R}^d$ is unit length

$$\text{Proj}(u) \vec{x} = \langle \vec{x}, u \rangle \vec{u}$$

$$\frac{u^T u}{u^T u} = 1$$

$$\Rightarrow \text{Want to find } u \text{ s.t. } \frac{1}{n} \sum_{i=1}^n \|\text{Proj}(u) x_i\|_2^2 = \frac{1}{n} \sum_{i=1}^n \|\langle x_i, u \rangle u\|_2^2 = \frac{1}{n} \sum_{i=1}^n \langle x_i, u \rangle^2$$

$$\langle x_i, u \rangle^2 = u^T \left(\frac{1}{n} \sum_{i=1}^n x_i x_i^T \right) u \text{ is maximal.}$$

= Sample covariance matrix (since x has been standardized).

$$u = \underset{u}{\operatorname{argmax}} \ u^T \left[\frac{1}{n} \sum_{i=1}^n x_i x_i^T \right] u. \quad \underset{u}{\operatorname{argmax}} \ u^T A u \Rightarrow \text{eigenvector of largest eigenvalue of } A.$$

Now, we find k s.t. $\frac{\sum_{i=1}^k \lambda_i}{\sum_{i=1}^d \lambda_i} = \text{eg } 95\% \Leftrightarrow \text{retained } 95\% \text{ of variance in data.}$
 sorted in decreasing order.
 These are all positive due to $x^T x$ being sorted.

SVD-wise: If X is square & symmetric, then X has an orthogonal eigenvector basis with real eigenvalues. \Rightarrow

Eigendecomp on $X^T X \Leftrightarrow$ SVD on X

equivalent

The Two intuitions of PCA: (1) Hyperplanes that capture the most variance and (2) Hyperplanes that allow you to project the data on with least L2 errors.

So far, we've seen 4 algos in unsupervised learning:

clustering	non-probabilistic probabilistic	classification
clustering	k-means	GMM
subspace analysis	PCA	Factor analysis
vector analysis	EM	Regression problem

ICA

(lectured)-party: d speakers & d microphones, $S \in \mathbb{R}^d$ is one instantaneous recording, Problem $X \in \mathbb{R}^{d \times d}$ & $X = AS$, $S \overset{\text{time}}{\sim} \text{speaker}$ & $X \overset{\text{time}}{\sim} \text{microphone}$
 mixing matrix

(S₁)

(S₂)

S, microphone

x₁

x₂

S₂, time

S₁, time

$$x^i = As^i$$

$$W = A^{-1} \text{[unmixing matrix]}$$

$$s^i \leftarrow W x^i$$

x₁, time

x₂, time

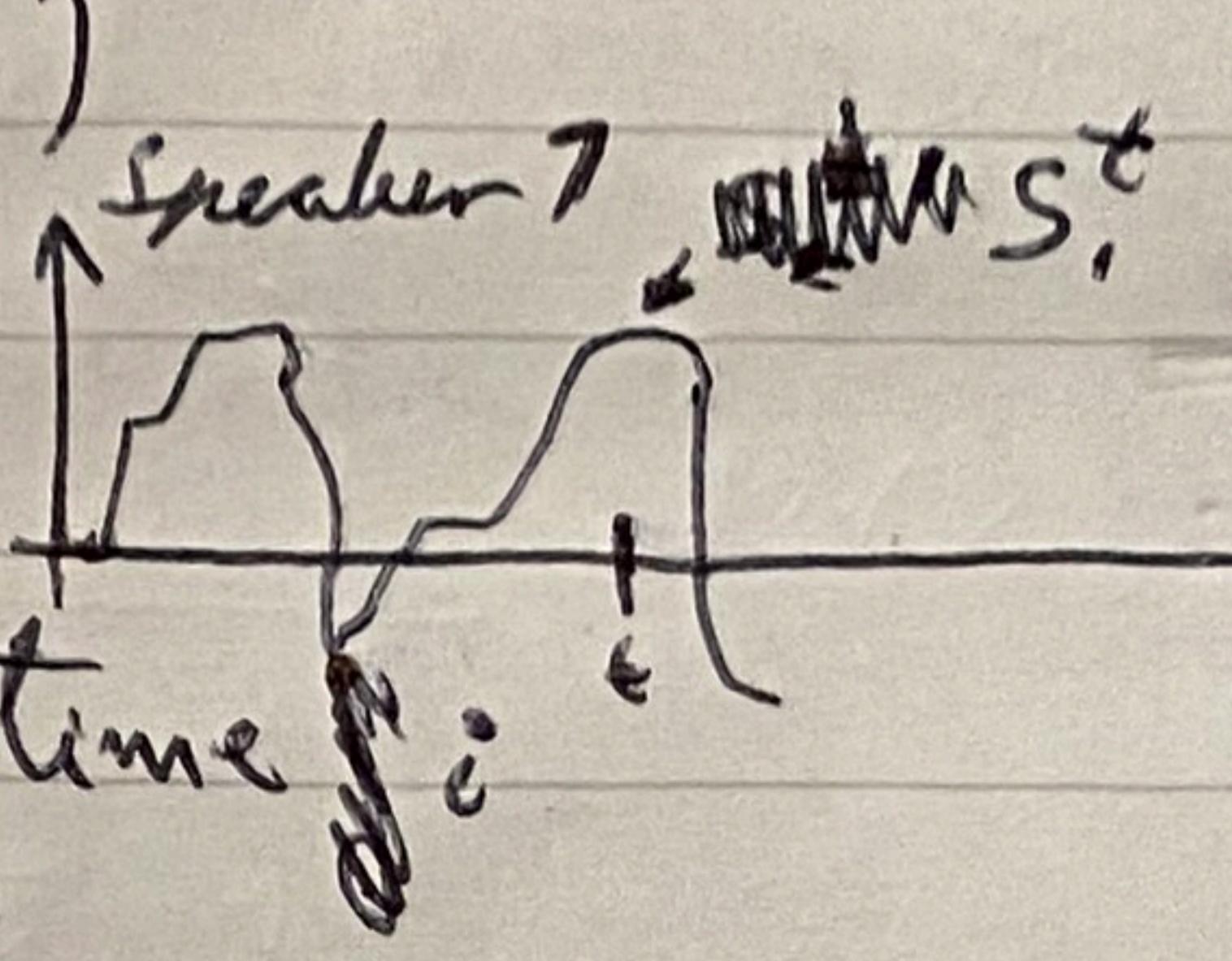
x_i = (x₁ⁱ, x₂ⁱ) $\in \mathbb{R}^2$, $d=2$

Independent Components Analysis (Lec 16)

- ICA model

Reinforcement learning

- MDP (Markov decision process)



$$\boxed{1} \quad \text{Sources: } S \in \mathbb{R}^n$$

$$S^i_j := \text{Speaker } j \text{ at time } i$$

$$x^i = A s^i - x \in \mathbb{R}^n$$

$$\text{Goal: Find } w = A^{-1}, \text{ so } S^i = w x^i \mid w = \begin{bmatrix} w_1^T \\ \vdots \\ w_n^T \end{bmatrix}$$

$P(S|X)$ 3 Definitions

1. CDF Cumulative distribution function

$$F(s) = P(S^i = s) = \int_{-\infty}^s p_{S^i}(s') ds' \Leftrightarrow p_{S^i}(s) = \frac{d}{ds} F(s)$$

Plug in Gaussian density

2. And:

$$p_{S^i}(x) = \frac{1}{\sqrt{|W|}} e^{-\frac{\|w\|^2}{2}} \cdot |W|^{-\frac{1}{2}}$$

Jacobian Determinant of w .
needed to preserve norm.

3. Finally: density of

$$P_S(s) = ? \quad \text{Well, } F(s) = P(S^i \leq s) = (1 + e^{-s})^{-1} = g(s) \text{ works well.}$$

$$\text{So } p_{S^i}(s) = \frac{d}{ds} F(s).$$

$$p(s) = \prod_{i=1}^n p_{S^i}(s_i), \text{ assume the } n \text{ speakers speak independently}$$

$$\text{Now, } p_X(x) = p_{S^i}(w x) \mid w \mid = \prod_{i=1}^n p_{S^i}(w_i^T x) \mid w \mid$$

$$\Rightarrow \text{MLE: } l(w) = \sum_{i=1}^n \log \left(\prod_{j=1}^n p_{S^i}(w_j^T x^i) \mid w \mid \right). \quad (\text{Can use stochastic gradient descent here.})$$

stock grad ascent:

$$\nabla_w l(w) = \begin{bmatrix} 1 - 2g(w^T x^1) \\ \vdots \\ 1 - 2g(w^T x^n) \end{bmatrix} \xrightarrow{\text{sigmoid}} x^{i^T} + (w^T)^{-1}, g(s) = (1 + e^{-s})^{-1}$$

→ This'll result in a good algorithm for unmixing the sources.

Reinforcement Learning

Chess playing program:
 $R(S)$ state reward function
 +1 for win, -1 for loss, 0 for draw

Credit assignment problem

Markov Decision Process (MDP):

$(S, A, \{P_{sa}\}, \gamma, R)$
 states set of actions state transition probabilities, $\sum_s P_{sa}(s') = 1$
 discount factor $\gamma \in [0, 1]$

$s \rightarrow a$ if reward
 Thus MDP has:

-1 per 11 states

Actions: {N, S, E, W}

$$P_{(3,2)} N((3,2)) = 0.8$$

$$P_{(3,1)} N((4,1)) = 0.1$$

$$P_{(3,1)} N((2,1)) = 0.1$$

$$P_{(3,1)} N(\text{rest...}) = 0.$$

small negative to punish lingering & waiting

$$R((9,3)) = +1; R((4,2)) = -1; R(S) = -0.02 \text{ for all other } S.$$

80

Choose action a_0
Get to $S_1 \sim P_{s,a_0}$

Choose action a_1
Get to $S_2 \sim P_{s,a_1}$

\vdots
 $\gamma = 0.99 \leftarrow$
 Gives smaller weight to rewards in distant future. The "time value of money"
 Real reason to use it though is because it turns out reinf. learning algos just converge faster.

Tot payoff

$$R(S_0) + \gamma R(S_1) + \gamma^2 R(S_2) + \dots = R_{\text{tot}}$$

Goal is to choose actions over time that maximize R_{tot} .

Policy $\pi: S \mapsto A$ ← This typically comes out of reinf. learning algos

Optimal policy:

5	↑	→	→	+1
2	↑	↑	↑	-1
1	↑	←	←	←

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