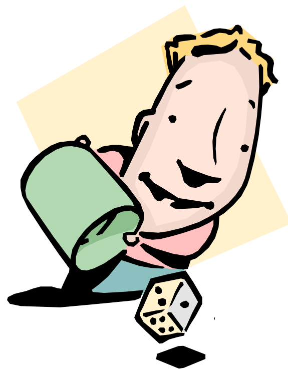
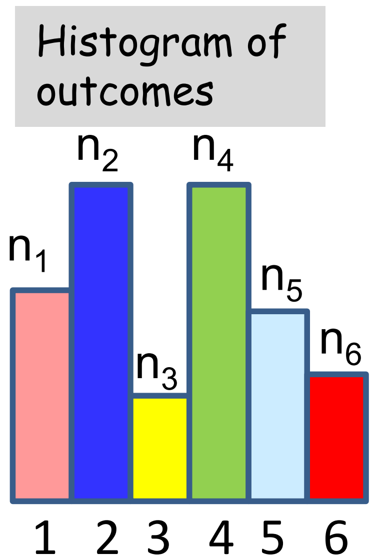
Maximum Likelihood Estimation
and
Expectation Maximization

Bhiksha Raj

Estimating distribution, an example: Multinomials

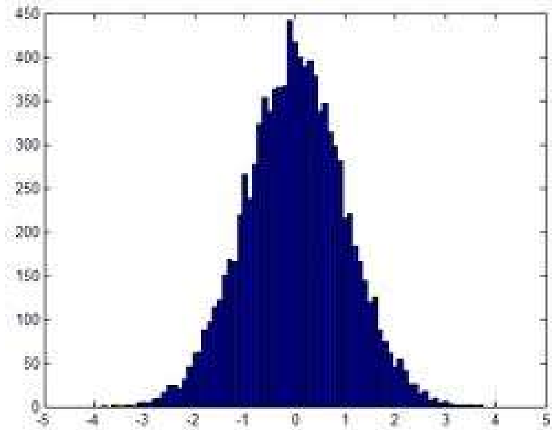


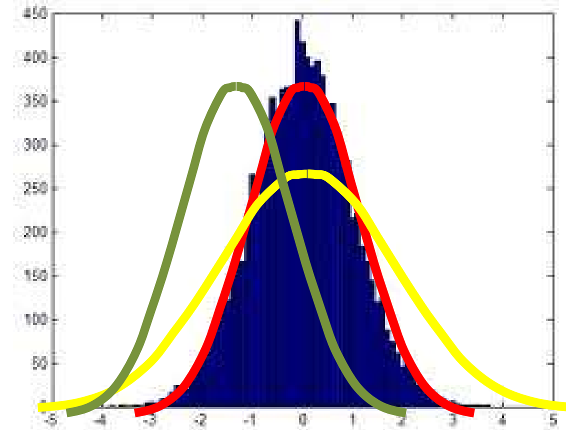
6 3 1 5 4 1 2 4 …



- A dice roller rolls dice and you plot the histogram of outcomes
  - Shown to the right
- The distribution is a multinomial
  - More precisely, a category distribution
  - Parameters to be learned: $p_1, p_2, p_3, p_4, p_5, p_6$
- Estimate the distribution

Estimating distributions: An example





- The left figure shows the histogram of a collection of observations
- We decide to model the distribution as Gaussian
  - Parameters: Mean μ and variance σ^2

- Estimate the parameters

Agenda

- Generative Models
- Fitting models to data
- Where'd the closed forms go?
- Dealing with missing information
- How expectation maximization solves all our problems

The story of generative models

- What are generative models
- How to estimate them
  - *Expectation maximization*

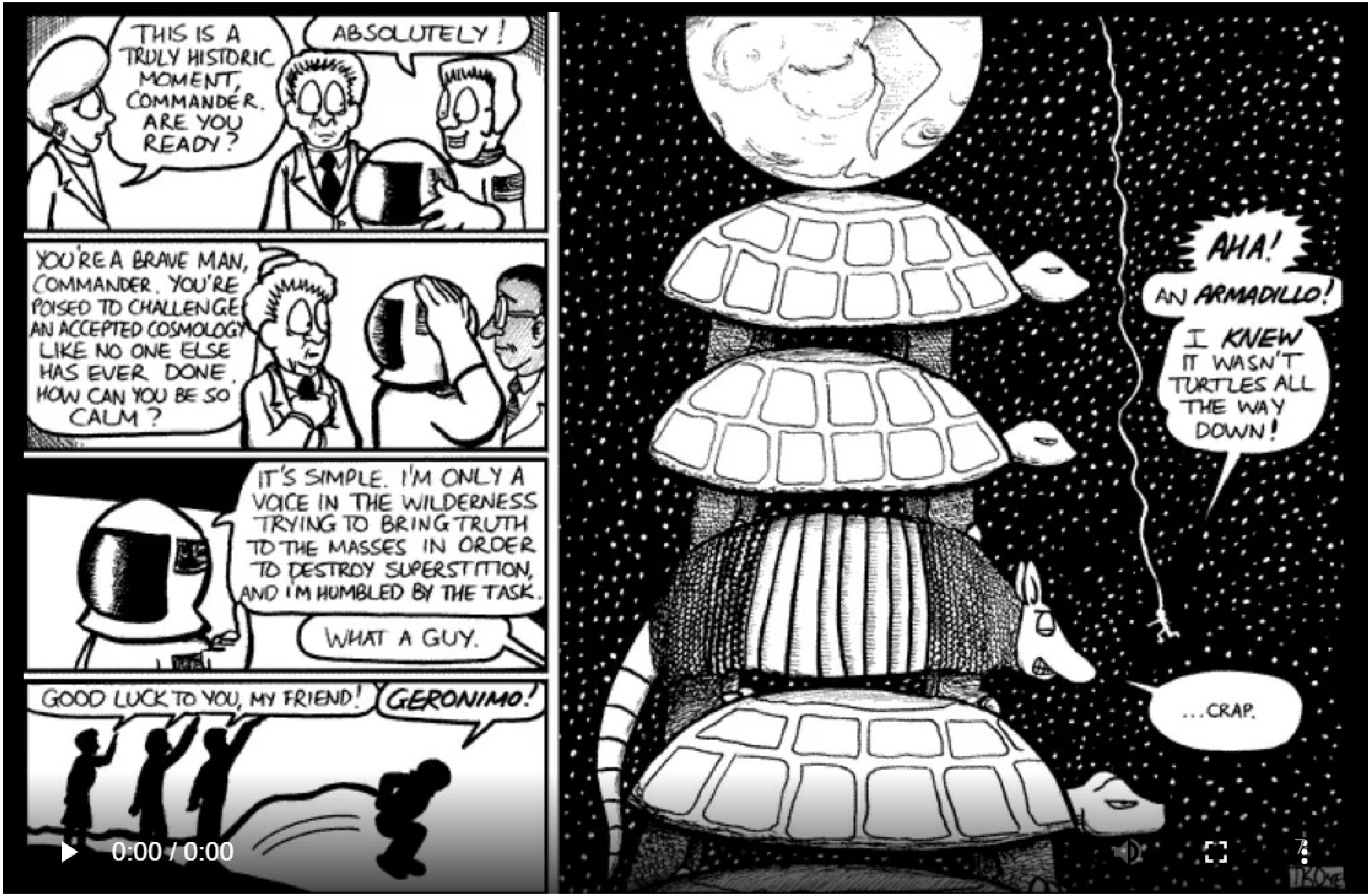
What is a generative model

- A model for the probability distribution of a data x
  - E.g. a multinomial, Gaussian etc.

- Computational equivalent: a model that can be used to "generate" data with a distribution similar to the given data x
  - Typical setting: a box that takes in random seeds and outputs random samples like x

  - Question: how do we generate the random seeds…

Its turtles all the way down (kinda)…



Some "simple" generative models

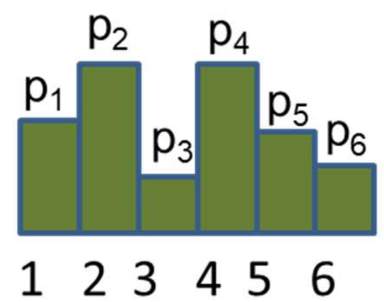
- The multinomial PMF

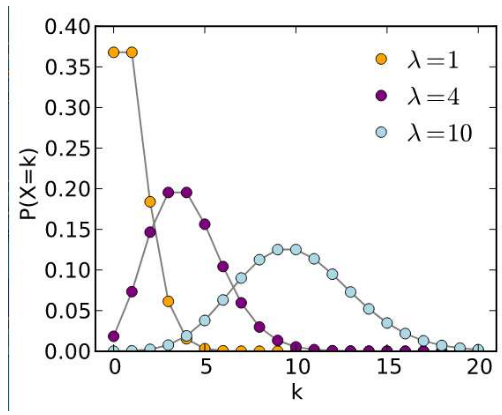
$$P(x = v) \equiv P(v)$$

  - For discrete data
    - v belongs to a discrete set
  - Can be expressed as a table of probabilities if the set of possible vs is finite
  - Else, requires a parametric form, e.g. Poisson

$$P(x = k) = \frac{\lambda^k e^{-\lambda}}{k!} \quad for\; k \geq 0$$

    - λ is the Poisson parameter



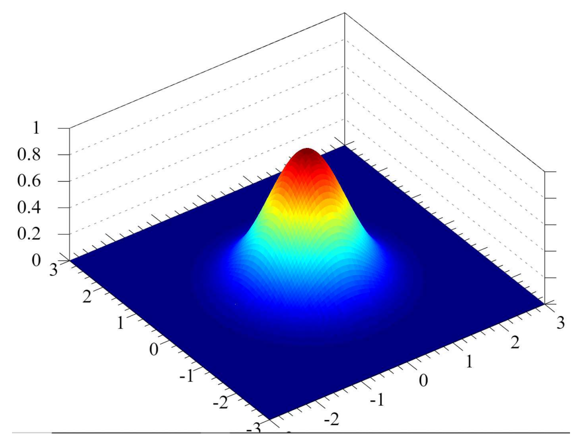


- The Gaussian PDF

$$P(x = v)$$

$$= \frac{1}{\sqrt{2\pi|\Sigma|}^D} \exp(-0.5(x-\mu)^T\Sigma^{-1}(x-\mu))$$

  - For continuous-valued data
  - μ is the mean of the distribution
  - Σ is the Covariance matrix

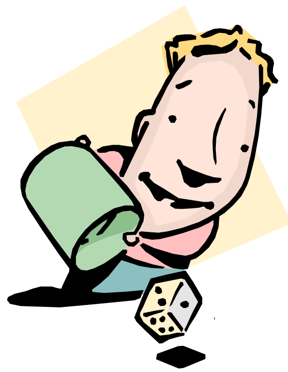


Learning a generative model for data

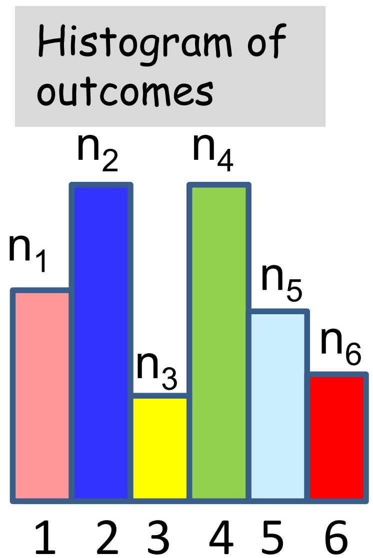
- You are given some set of observed data $X = \{x\}$.
- You choose a model $P(x;\theta)$ for the distribution of x
  - θ are the parameters of the model

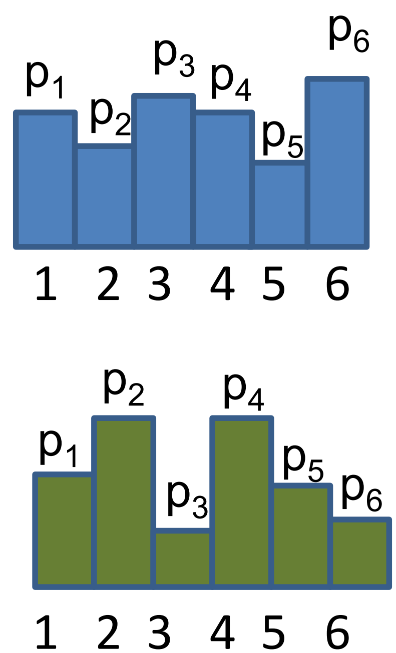
- Estimate the theta such that $P(x;\theta)$ best "fits" the observations $X = \{x\}$
  - Hoping it will also represent data outside the training set.

An example: Multinomials



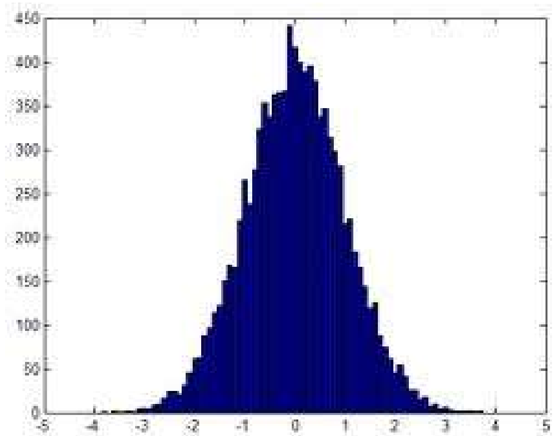
6 3 1 5 4 1 2 4 …

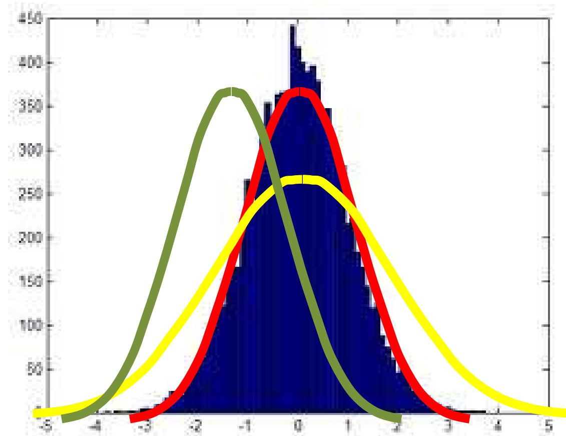




- A dice roller rolls dice and you plot the histogram of outcomes
  - Shown to right
- The distribution is a multinomial
  - Parameters to be learned: $p_1, p_2, p_3, p_4, p_5, p_6$
- Which of the two multinomial PDFs shown to the right is more likely to be the PDF for the dice?
  - Why?

An example





- The left figure shows the histogram of a collection of observations
- We decide to model the distribution as Gaussian
  - Parameters: Mean μ and variance σ^2

- Which of the three Gaussians shown in the right figure is most likely to be the actual PDF of the RV?
  - Why?

Defining “Best Fit”: Maximum likelihood

- The data are generated by draws from the distribution
  - I.e. the generating process draws from the distribution
- Assumption: The world is a boring place
  - The data you have observed are very typical of the process
- Consequent assumption: The distribution has a high probability of generating the observed data
  - Not necessarily true
- Select the distribution that has the *highest* probability of generating the data
  - Should assign lower probability to less frequent observations and vice versa

Maximum Likelihood Estimation: Multinomial

- Probability of generating (n_1, n_2, n_3, n_4, n_5, n_6)

$$P(n_1, n_2, n_3, n_4, n_5, n_6) = Const \prod_i p_i^{n_i}$$

- Find p_1,p_2,p_3,p_4,p_5,p_6 so that the above is maximized
- Alternately maximize

$$\log\left(P(n_1, n_2, n_3, n_4, n_5, n_6)\right) = \log(Const) + \sum_i n_i \log\left(p_i\right)$$

  - Log() is a monotonic function
  - argmax_x f(x) = argmax_x log(f(x))

Maximum Likelihood Estimation: Multinomial

- Probability of generating (n_1, n_2, n_3, n_4, n_5, n_6)

$$P(n_1, n_2, n_3, n_4, n_5, n_6) = Const \prod_i p_i^{n_i}$$

- Find p_1,p_2,p_3,p_4,p_5,p_6 so that the above is maximized
- Alternately maximize

$$\log\left(P(n_1, n_2, n_3, n_4, n_5, n_6)\right) = \log(Const) + \sum_i n_i \log\left(p_i\right)$$

  - Log() is a monotonic function
  - argmax_x f(x) = argmax_x log(f(x))
- Solving for the probabilities gives us
  - Requires constrained optimization to ensure probabilities sum to 1

$$p_i = \frac{n_i}{\sum_j n_j}$$

ITS JUST COUNTING!

Maximum Likelihood: Gaussian

- Given a collection of observations (X_1, X_2,…), estimate mean μ and covariance Θ

$$P(X_1, X_2, \ldots) = \prod_i \frac{1}{\sqrt{(2\pi)^d |\Theta|}} \exp\left(-0.5 (X_i - \mu)^T \Theta^{-1} (X_i - \mu)\right)$$

$$\log\left(P(X_1, X_2, \ldots)\right) = C - 0.5 \sum_i \left(\log\left(|\Theta|\right) + (X_i - \mu)^T \Theta^{-1} (X_i - \mu)\right)$$

Maximum Likelihood: Gaussian

- Given a collection of observations ($X_1, X_2,\ldots$), estimate mean μ and covariance Θ

$$P(X_1, X_2,\ldots) = \prod_i \frac{1}{\sqrt{(2\pi)^d |\Theta|}} \exp\left(-0.5(X_i - \mu)^T \Theta^{-1}(X_i - \mu)\right)$$

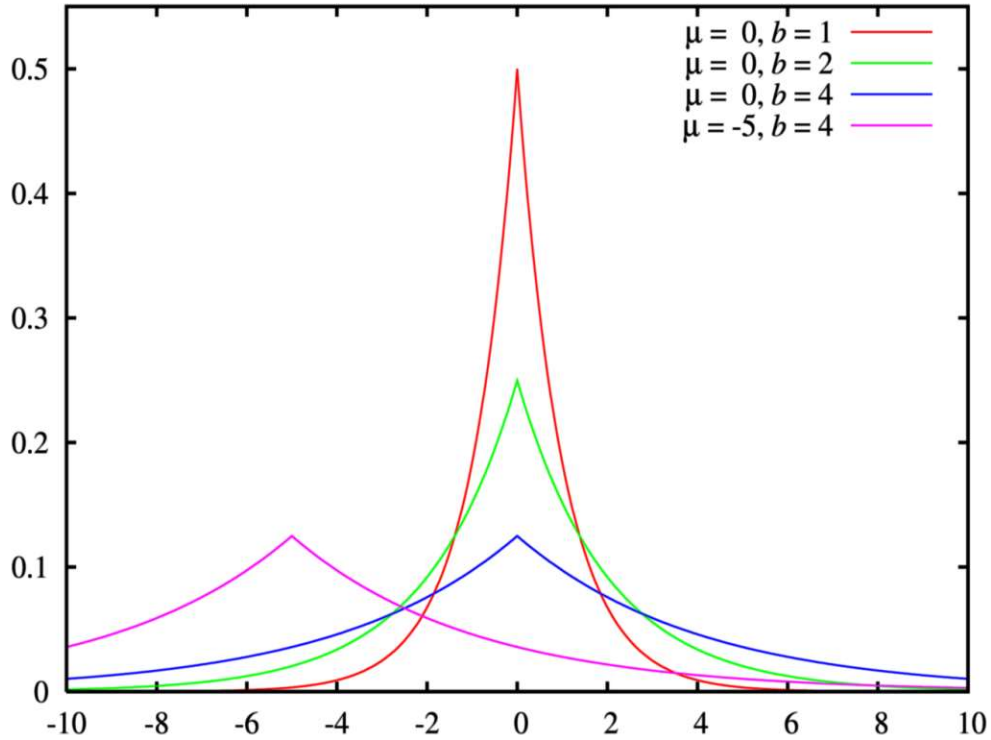
$$\log\left(P(X_1, X_2,\ldots)\right) = C - 0.5\sum_i \left(\log\left(|\Theta|\right) + (X_i - \mu)^T \Theta^{-1}(X_i - \mu)\right)$$

- Maximizing w.r.t μ and Θ gives us

$$\mu = \frac{1}{N}\sum_i X_i \qquad \Theta = \frac{1}{N}\sum_i (X_i - \mu)(X_i - \mu)^T$$

ITS STILL JUST COUNTING!

Laplacian



$$P(x) = L(x;\mu,b) = \frac{1}{2b}\exp\left(-\frac{|x-\mu|}{b}\right)$$

- Parameters: Median μ, scale b ($b > 0$)
  - μ is also the mean, but is better viewed as the median

Maximum Likelihood: Laplacian

- Given a collection of observations (x_1, x_2,…), estimate mean μ and scale b

$$\log\left(P(x_1, x_2, \ldots)\right) = C - N\log(b) - \sum_i \frac{|x_i - \mu|}{b}$$

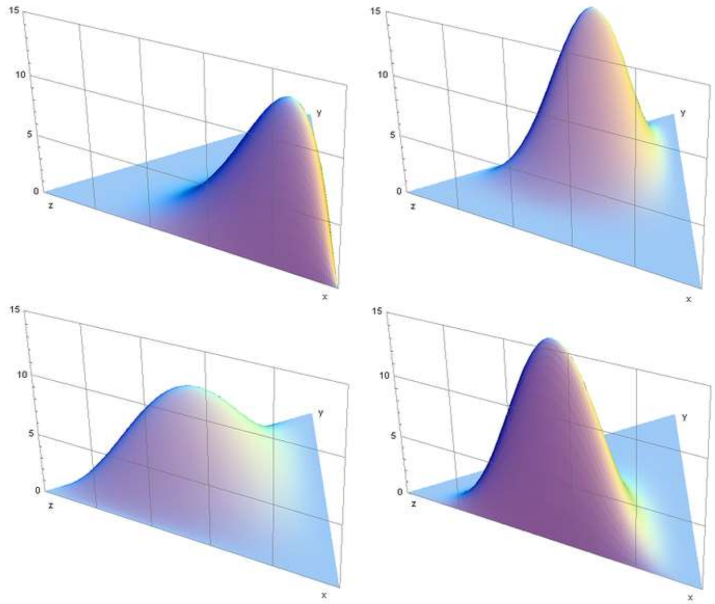
- Maximizing w.r.t μ and b gives us

$$\mu = median(\{x_i\}) \quad b = \frac{1}{N}\sum_i |x_i - \mu|$$

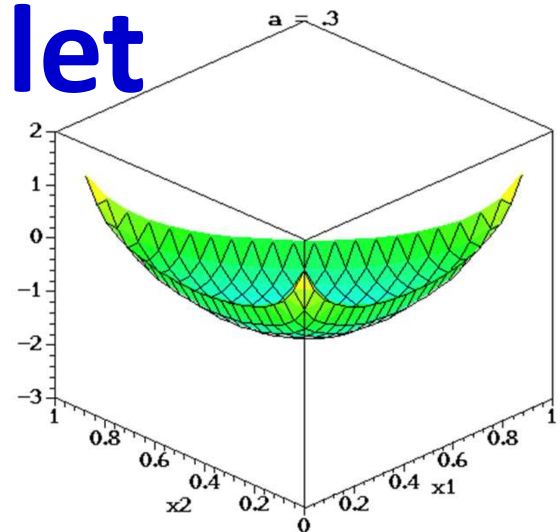
Still just counting

Dirichlet

(from wikipedia)



K=3. Clockwise from top left:
α=(6, 2, 2), (3, 7, 5), (6, 2, 6), (2, 3, 4)



log of the density as we change α from α=(0.3, 0.3, 0.3) to (2.0, 2.0, 2.0), keeping all the individual αi's equal to each other.

$$P(X) = D(X;\alpha) = \frac{\prod_i \Gamma(\alpha_i)}{\Gamma\left(\sum_i \alpha_i\right)} \prod_i x_i^{\alpha_i - 1}$$

- Parameters are αs
  - Determine mode and curvature
- Defined only for probability vectors
  - $X = [x_1\ x_2\ ..\ x_K]$, $\Sigma_i\ x_i = 1$, $x_i >= 0$ for all i

Maximum Likelihood: Dirichlet

- Given a collection of observations (X_1, X_2,…), estimate α

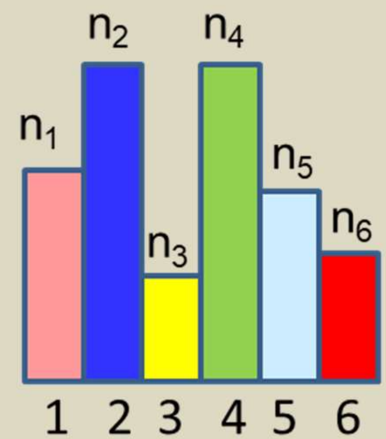
$$\log\left(P(X_1, X_2, \ldots)\right) = \sum_j \sum_i (\alpha_i - 1)\log(X_{j,i}) + N\sum_i \log\left(\Gamma(\alpha_i)\right) - N\log\left(\Gamma\left(\sum_i \alpha_i\right)\right)$$

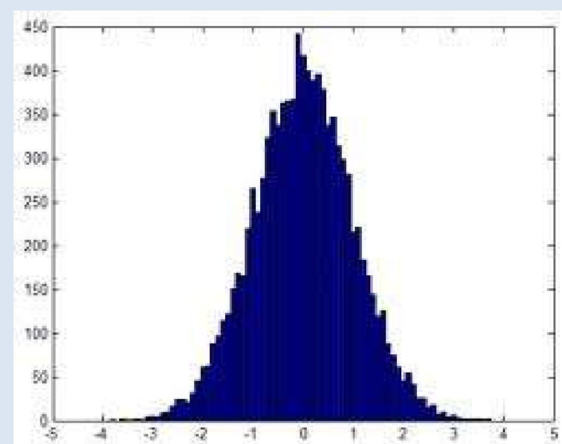
- No closed form solution for αs.
  - Needs gradient ascent

- Several distributions have this property: the ML estimate of their parameters have no closed form solution

Maximum likelihood

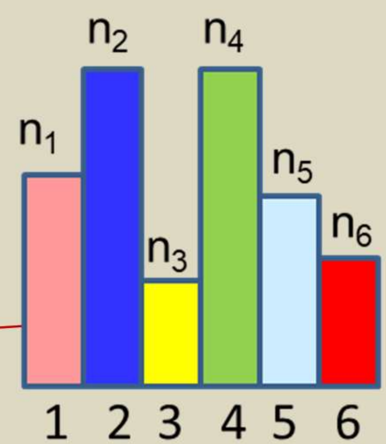
- The maximum likelihood principle:
  - $\underset{\theta}{\operatorname{argmax}} P(X;\theta) = \underset{\theta}{\operatorname{argmax}} \log(P(X;\theta))$



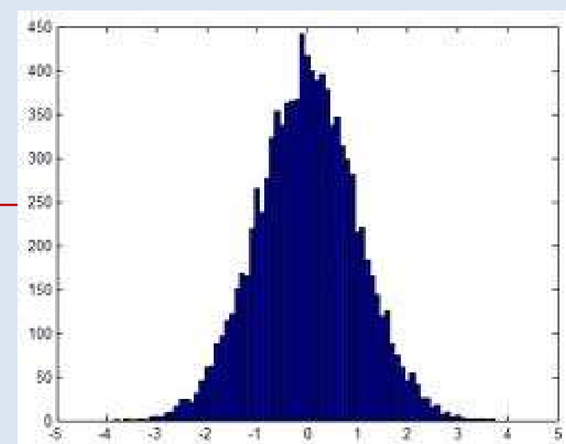


Maximum likelihood

- The maximum likelihood principle:
  - $\underset{\theta}{\operatorname{argmax}}\, P(X;\theta) = \underset{\theta}{\operatorname{argmax}}\, \log(P(X;\theta))$
- For the histogram
  - $\underset{\{p_1,p_2,p_3,p_4,p_5,p_6\}}{\operatorname{argmax}}\, \log(\prod_{x\in X} P(x))$

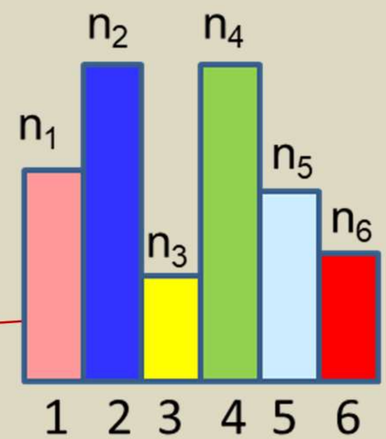


- For the Gaussian
  - $\underset{\mu,\sigma^2}{\operatorname{argmax}}\, \log(\prod_{x\in X} P(x))$



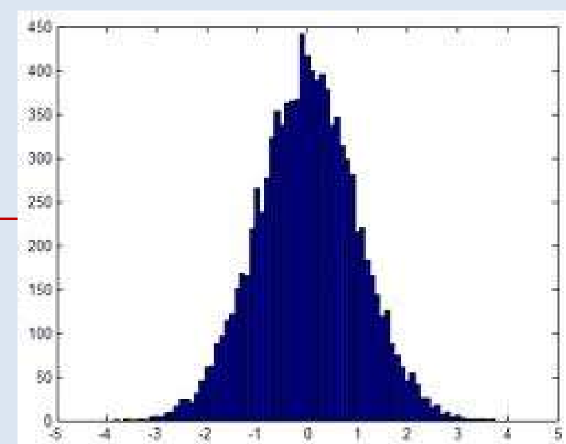
Maximum likelihood

- The maximum likelihood principle:
  - $\underset{\theta}{\operatorname{argmax}} P(X;\theta) = \underset{\theta}{\operatorname{argmax}} \log(P(X;\theta))$
- For the histogram
  - $\underset{\{p_1,p_2,p_3,p_4,p_5,p_6\}}{\operatorname{argmax}} \log(\prod_{x\in X} P(x))$



Can be grouped by value (every instance of i has the same probability)

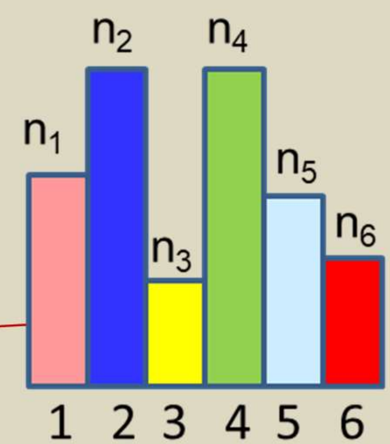
- For the Gaussian
  - $\underset{\mu,\sigma^2}{\operatorname{argmax}} \log(\prod_{x\in X} P(x))$



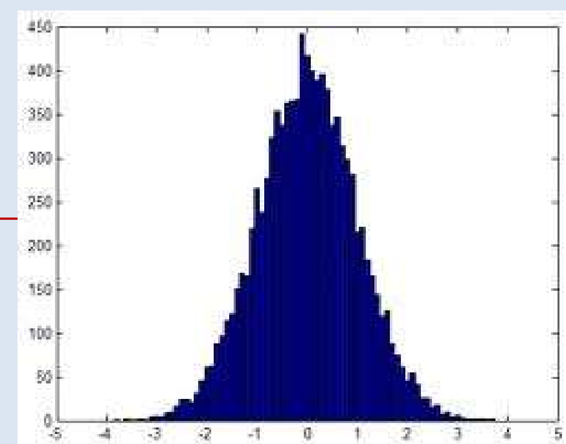
This probability is a Gaussian

Maximum likelihood

- The maximum likelihood principle:
  - $\underset{\theta}{\operatorname{argmax}} P(X;\theta) = \underset{\theta}{\operatorname{argmax}} \log(P(X;\theta))$
- For the histogram
  - $\underset{\{p_1,p_2,p_3,p_4,p_5,p_6\}}{\operatorname{argmax}} \log\left(\prod_i p_i^{n_i}\right)$

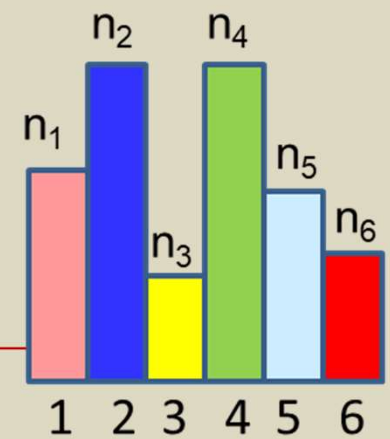


- For the Gaussian
  - $\underset{\mu,\sigma^2}{\operatorname{argmax}} \log\left(\prod_{x\in X} Gaussian(x;\mu,\sigma^2)\right)$



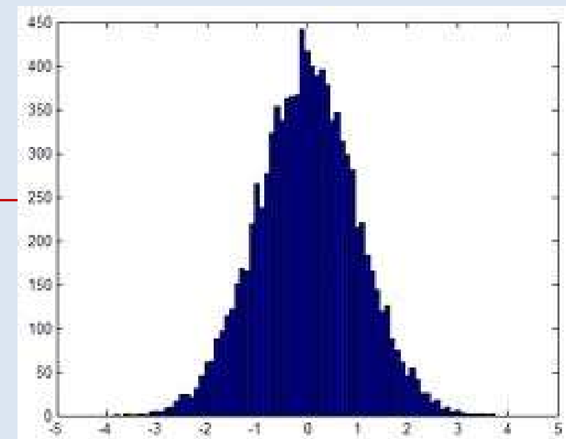
Maximum likelihood

- The maximum likelihood principle:
  - $\underset{\theta}{\operatorname{argmax}} P(X;\theta) = \underset{\theta}{\operatorname{argmax}} \log(P(X;\theta))$
- For the histogram
  - $\underset{\{p_1,p_2,p_3,p_4,p_5,p_6\}}{\operatorname{argmax}} \sum_i n_i \log(p_i)$



$\Rightarrow p_i = \frac{n_i}{N}$ (N is the total number of observations)

- For the Gaussian
  - $\underset{\mu,\sigma^2}{\operatorname{argmax}} \sum_{x\in X} \log Gaussian(x;\mu,\sigma^2)$



$\Rightarrow \mu = \frac{1}{N}\sum_{x\in X} x$; $\sigma^2 = \frac{1}{N}\sum_{x\in X}(x-\mu)^2$

Poll 1: tinyurl.com/mlsp23-20231102-1

- The true model that generates any dataset is always the most likely model (i.e. the one with the highest probability to generate the dataset
  - True
  - False

- Maximum likelihood estimation reduces to simple counting-like solutions in many cases
  - True
  - False

Poll 1

- The true model that generates any dataset is always the most likely model (i.e. the one with the highest probability to generate the dataset
  - True
  - **False**

- Maximum likelihood estimation reduces to simple counting-like solutions in many cases
  - **True**
  - False

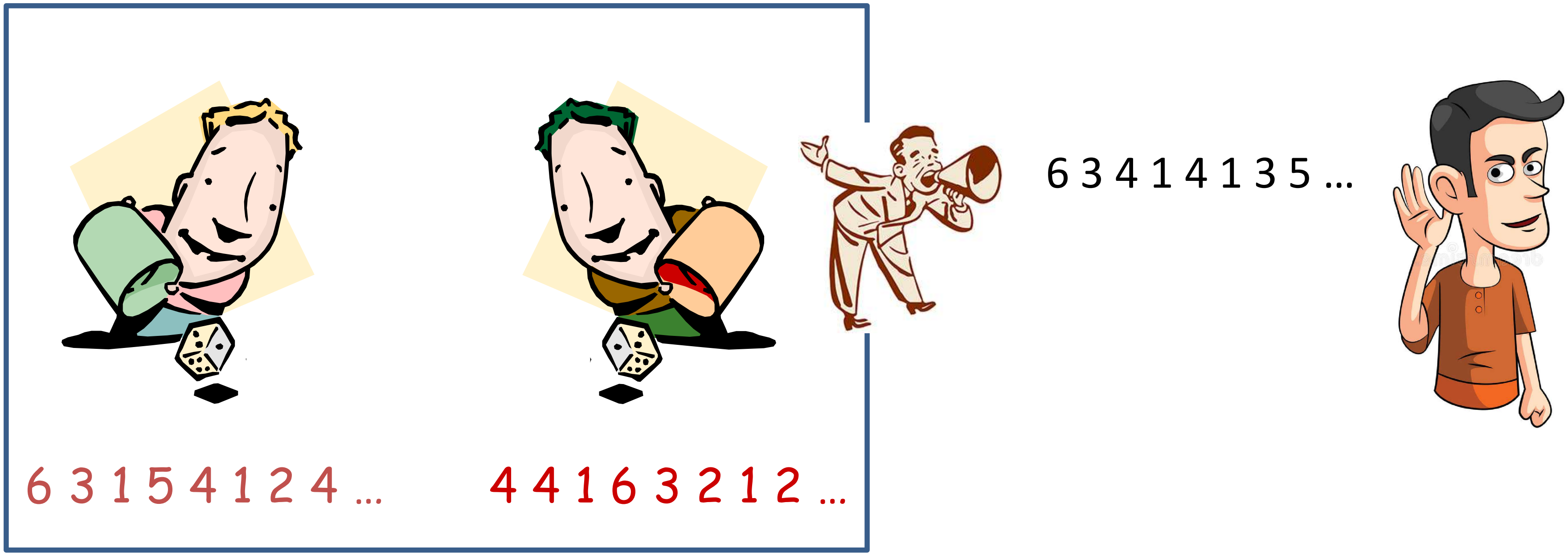
But now for something somewhat different


- Caller rolls a dice and flips a coin
- He calls out the number rolled if the coin shows head
- Otherwise, he calls the number+1
- Can we estimate p(heads) and p(number) for the dice from a collection of outputs

But now for something somewhat different

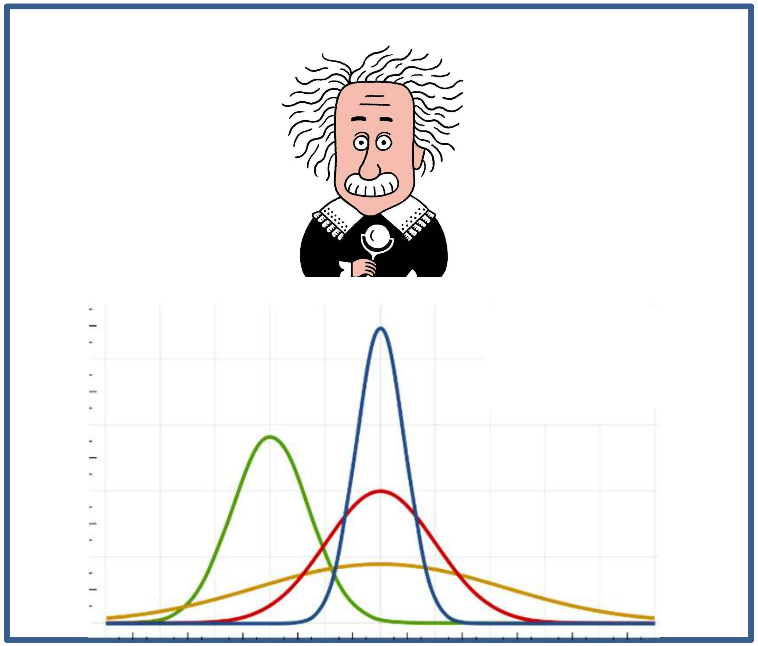
- Roller rolls two dice
- He calls out the sum
- Determine P(dice) from a collection of outputs

Your friendly neighborhood gamblers



- Two gamblers shoot dice in a closed room
  - The dice are differently loaded for the two of them
- A crazy crier randomly select one of the them and calls out his number
  - But doesn't mention whose number he chose
- You only see the numbers
  - But do not know which of them rolled the number
- **How to determine the probability distributions of the two dice?**

Your friendly Gaussian gambler…

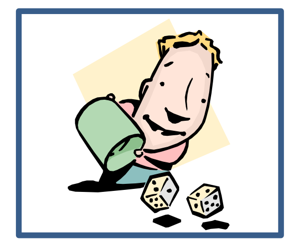


6.1 1.2 -2.1 3.4 0.9 -2.1 -0.8 …

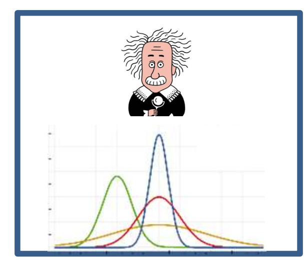
- Your friendly neighborhood Gaussian gambler has a collection of Gaussian generators
- In each trial he randomly selects a Gaussian, and draws a number from it
- He calls out that number
- From only the numbers he calls out, can you estimate all of the Gaussians?

The challenge


- In each of these problems there was some information missing



- If this information were available, estimation would've been trivial



Let's Look at Missing Information

Missing Information
about **Underlying Data**

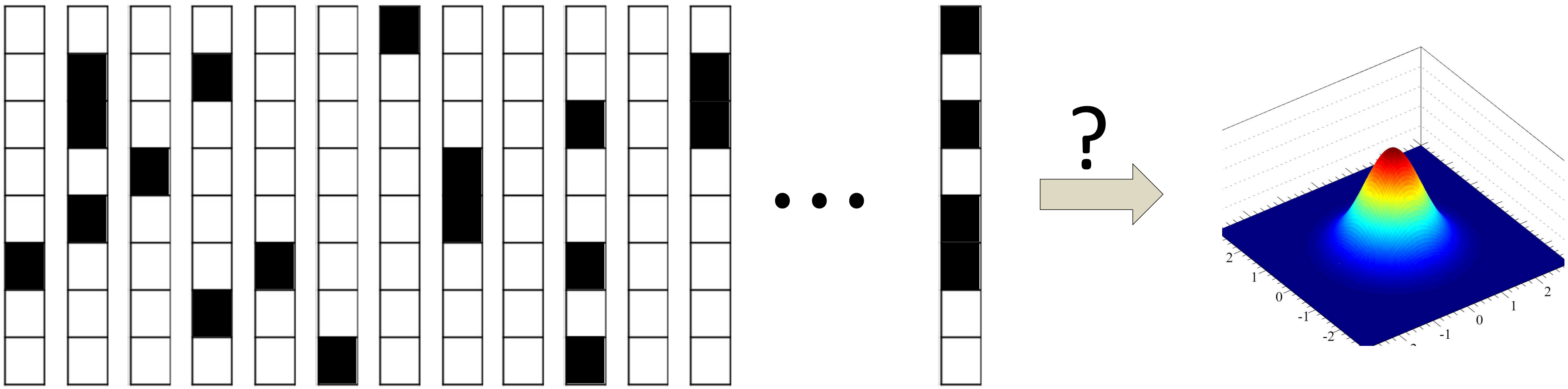
Missing Information
about **Underlying Process**

Let's Look at Missing Information

Missing Information
about **Underlying Data**

Missing Information
about **Underlying Process**

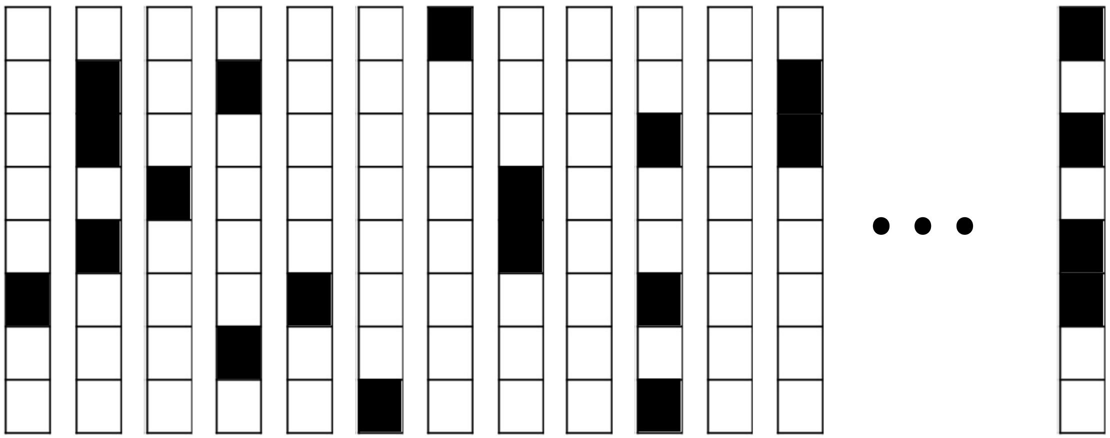
Examples of incomplete data: missing data



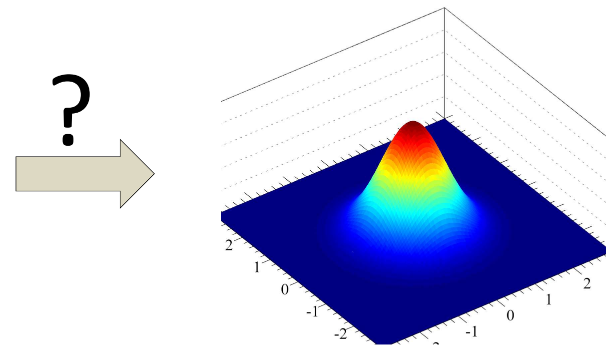
Blacked-out components are missing from data

- Objective: Estimate a Gaussian distribution from a collection of vectors
- Problem: Several of the vector components are missing
- Must estimate the mean and covariance of the Gaussian with these incomplete data
  - What would be a good way of doing this?

Maximum likelihood estimation with incomplete data



$P(x) = Gaussian(x; \mu, \Sigma)$



Blacked-out components are missing from data

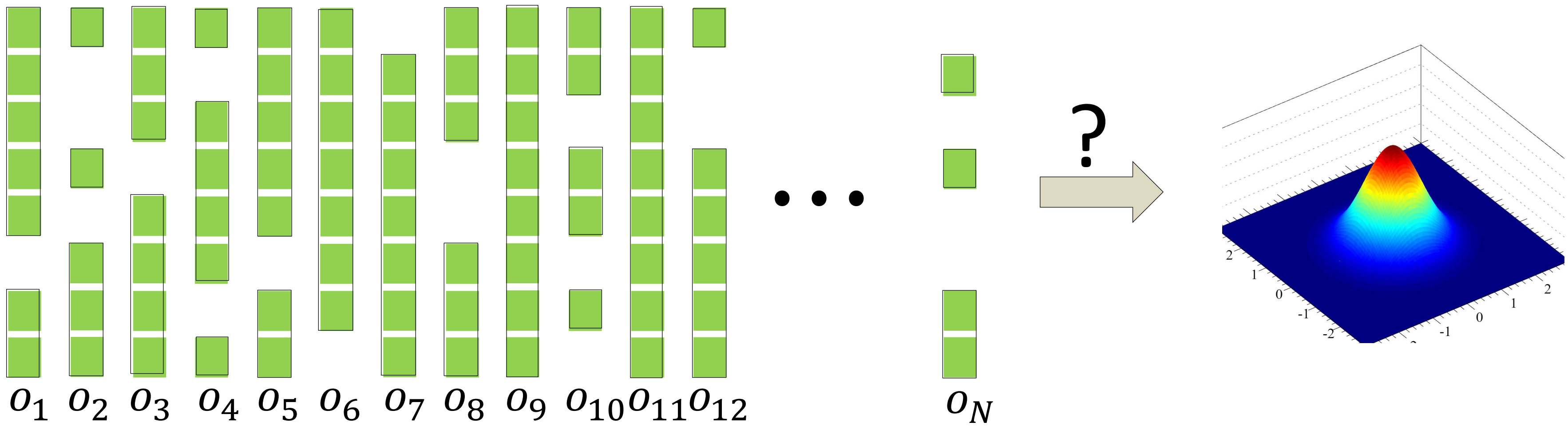
- Original problem: Estimate the Gaussian given a collection $X = \{x\}$ of *complete* vectors

$$\underset{\mu,\Sigma}{\operatorname{argmax}} \log(P(X)) \quad \text{where X is the entire data}$$

$$= \underset{\mu,\Sigma}{\operatorname{argmax}} \sum_{x \in X} \log P(x) \quad \text{where P() is a Gaussian}$$

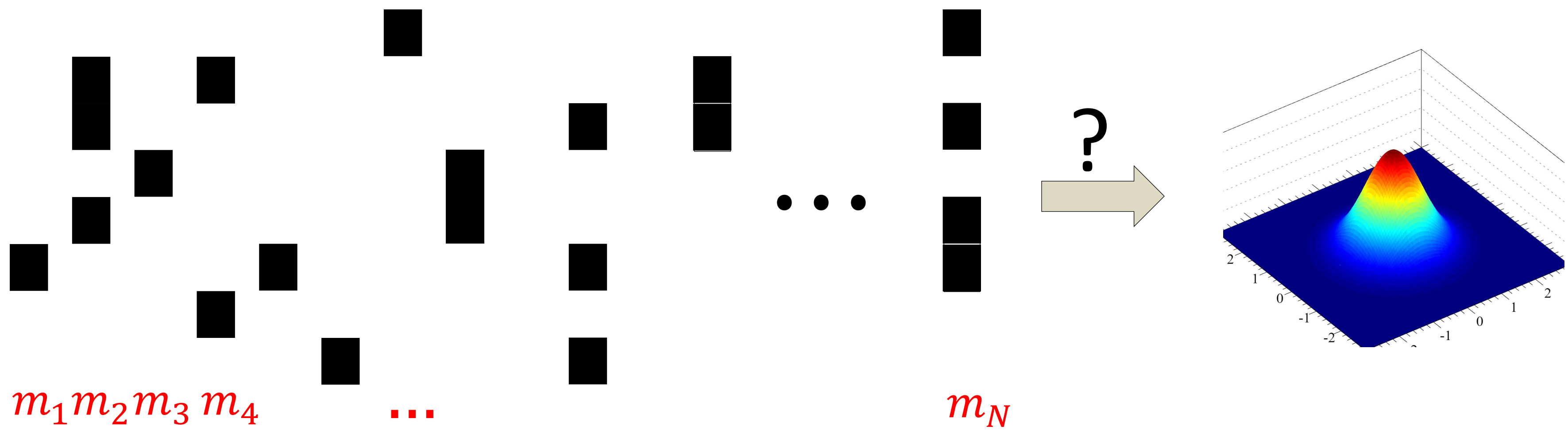
- Unfortunately, many components of each vector are missing in our data

Maximum likelihood estimation with incomplete data



- These are the actual data we have: A set $O = \{o_1, \ldots, o_N\}$ of *incomplete* vectors
  - Comprising only the *observed* components of the data

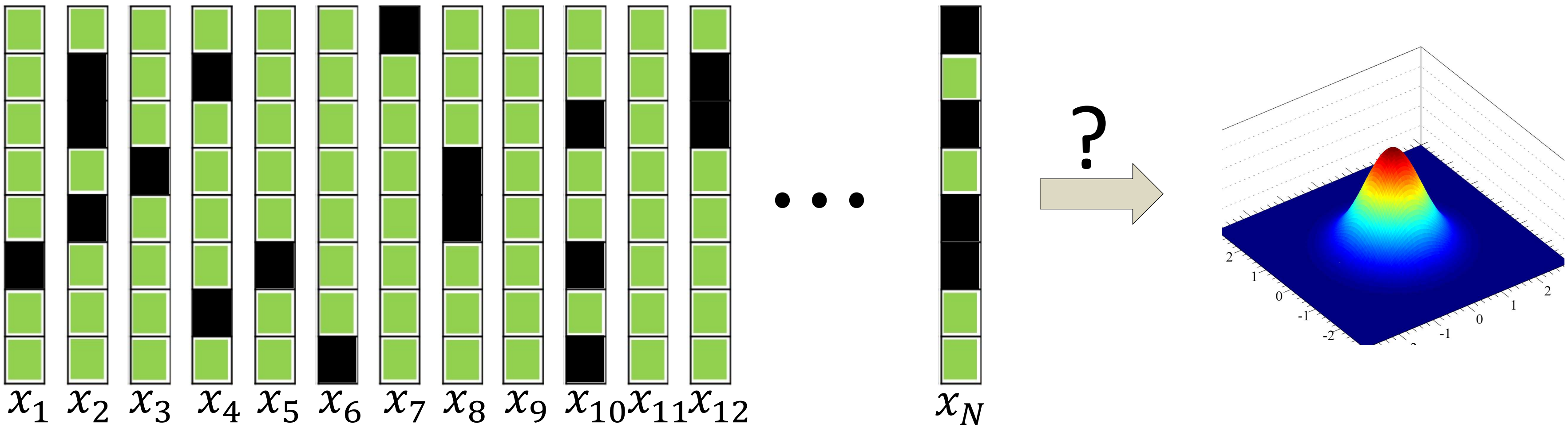
Maximum likelihood estimation with incomplete data



- These are the actual data we have: A set $O = \{o_1, \ldots, o_N\}$ of *incomplete* vectors
  - Comprising only the *observed* components of the data

- We are *missing* the data $M = \{m_1, \ldots, m_N\}$
  - Comprising the *missing* components of the data

Maximum likelihood estimation with incomplete data

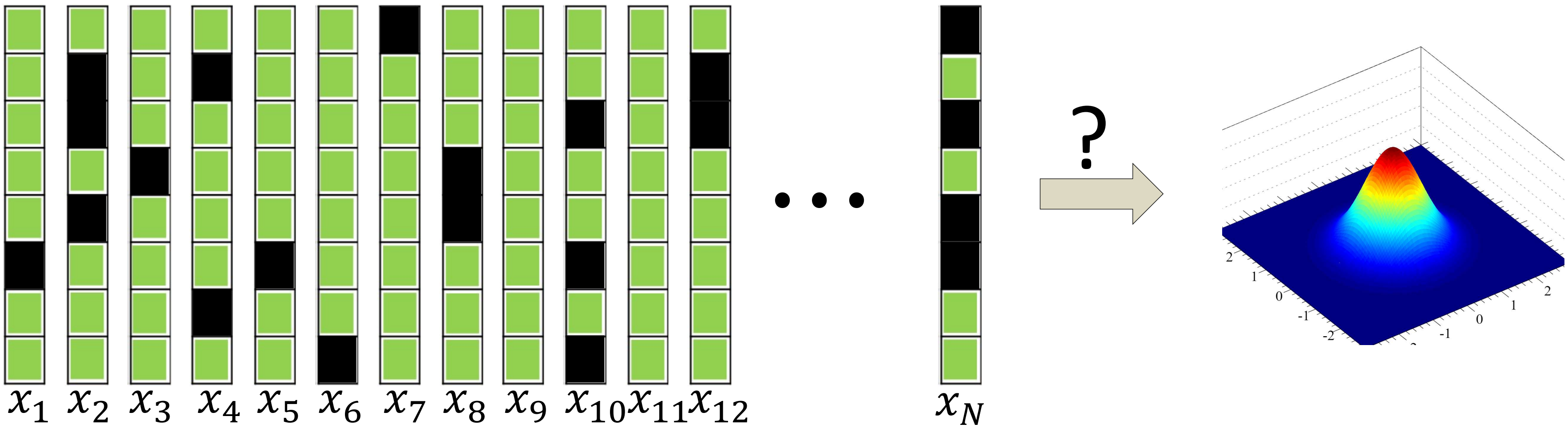


- These are the actual data we have: A set $O = \{o_1, \ldots, o_N\}$ of *incomplete* vectors
  - Comprising only the *observed* components of the data
- We are *missing* the data $M = \{m_1, \ldots, m_N\}$
  - Comprising the *missing* components of the data
- The *complete* data includes both the observed and missing components

$$X = \{x_1, \ldots, x_N\}, \qquad x_i = (o_i, m_i)$$

  - Keep in mind that at the complete data are *not* available (the missing components are missing)

Maximum likelihood estimation with incomplete data



- Maximum likelihood estimation: Maximize the likelihood of the *observed* data
  - That is all we really have

$$\underset{\mu,\Sigma}{\operatorname{argmax}} \log(P(O)) = \underset{\mu,\Sigma}{\operatorname{argmax}} \sum_{o \in O} \log P(o)$$

- Unfortunately, the Gaussian is defined on the *complete* vector :
  - $P(x) = Gaussian(x; \mu, \Sigma)$
  - In order to compute $P(o)$ we must *derive* it from $P(x)$

The log likelihood of incomplete data

- The probability of any vector x with observed and missing parts o and m

$$P(x) = P(o, m)$$

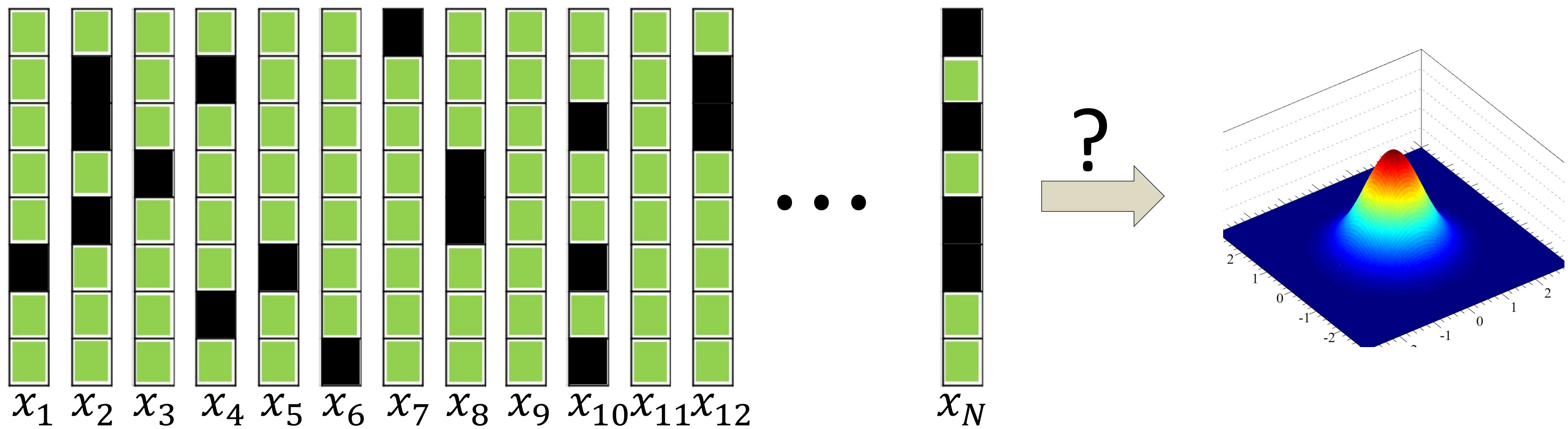
- Compute the probability of the observed components by marginalizing out the missing components

$$P(o) = \int_{-\infty}^{\infty} P(x)dm = \int_{-\infty}^{\infty} P(o, m)dm$$

- The log probability of the *entire observed training data:*

$$\sum_{o \in O} \log \int_{-\infty}^{\infty} P(o, m)dm$$

Maximum likelihood estimation with incomplete data



- Maximum likelihood estimation: Maximize the likelihood of the *observed* data

$$\underset{\mu,\Sigma}{\operatorname{argmax}} \log(P(O)) = \underset{\mu,\Sigma}{\operatorname{argmax}} \sum_{o \in O} \log \int_{-\infty}^{\infty} P(o,m)dm$$

- This requires the maximization of the log of an integral!
  - No closed form
  - Challenging on a good day, impossible on a bad one

Let’s Look at Missing Information

Missing Information
about **Underlying Data**

Missing Information
about **Underlying Process**

Let's Look at Missing Information

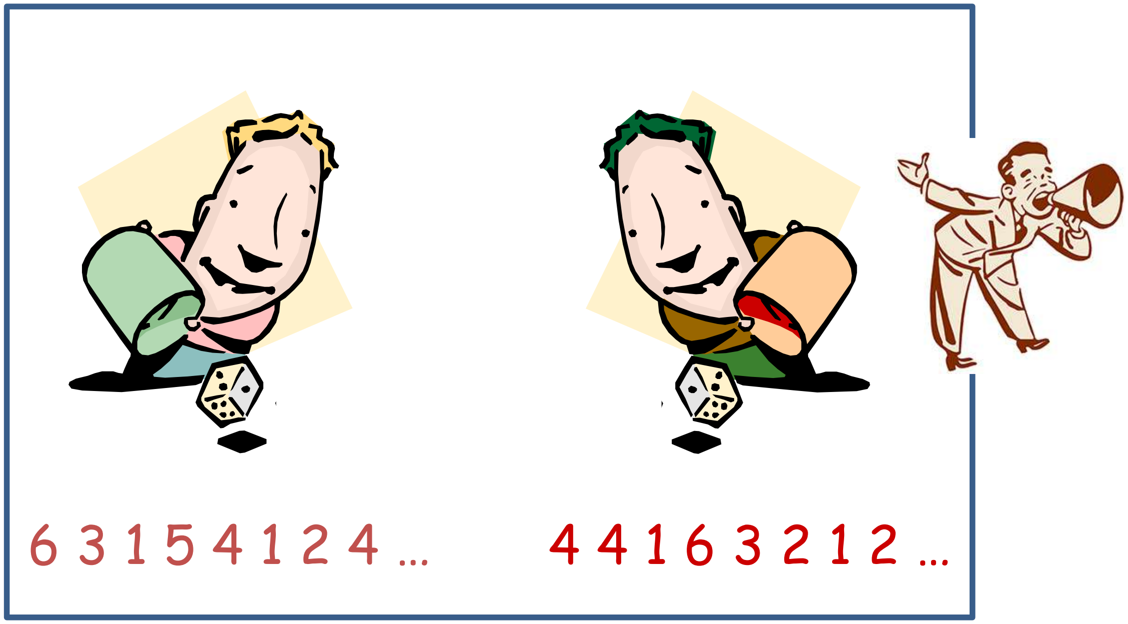
Missing Information
about **Underlying Data**

Missing Information
about **Underlying Process**

Shooting Dice

General Mixtures

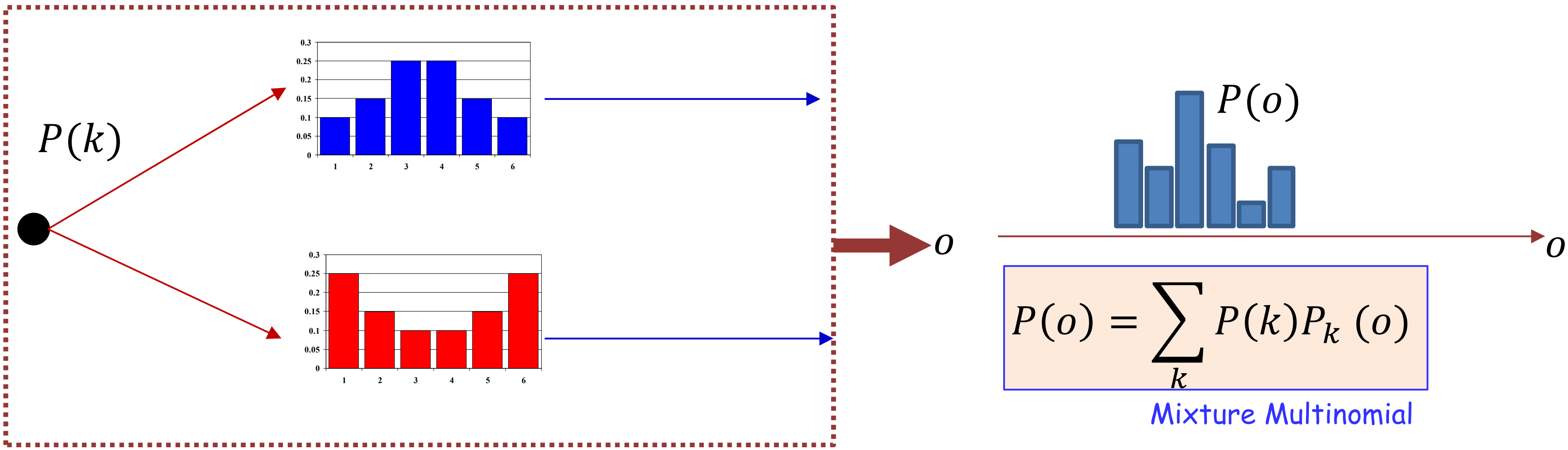
Our dice rolling gamblers



- Two persons shoot loaded dice repeatedly
  - The dice are differently loaded for the two of them
- We observe the series of outcomes for both persons

- **How to determine the probability distributions of the two dice?**

Examples of incomplete data: missing information in multinomial mixtures

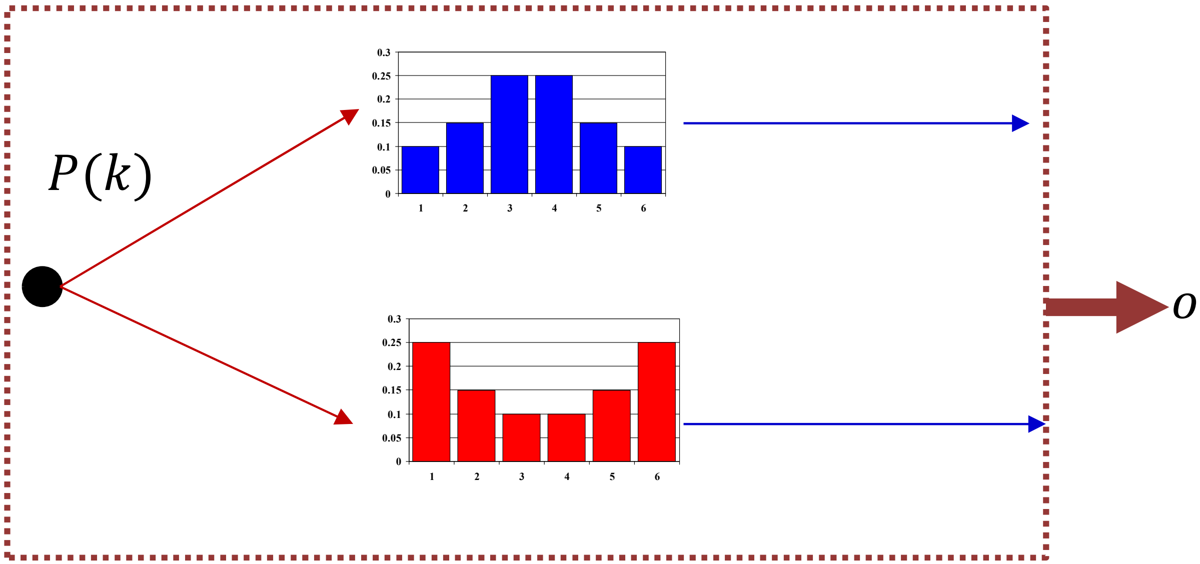


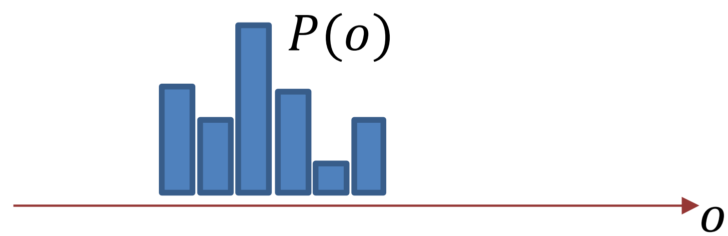
$$P(o) = \sum_k P(k)P_k(o)$$

Mixture Multinomial

- The generative model characterizes the data as the outcome of a two-level process
  - In the first step the process chooses a Multinomial from a collection
  - In the second, it draws the observation o from the chosen multinomial
  - The overall model is a *mixture Multinomial*
- Objective: Learn the parameters of all the multinomials from training data
  - The probabilities of the individual outcomes
  - And also the probability with which each multinomial is selected for the draw

Examples of incomplete data:
missing information in multinomial mixtures





$$P(o) = \sum_k P(k) P_k(o)$$

Mixture Multinomial

- Note, the process actually draws *two* variables for each observation, k and o.

- The probability of a particular draw is actually the joint probability of both variables

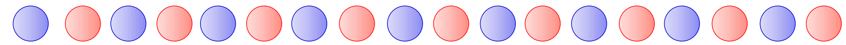
$$P(k, o) = P(k)P(o|k) = P(k)P_k(o)$$

- To compute the probability of obtaining any observation o, we are *marginalizing out* the multinomial index variable

$$P(o) = \sum_k P(k, o) = \sum_k P(k)P_k(o)$$

The *complete* data needed to precisely learn the model

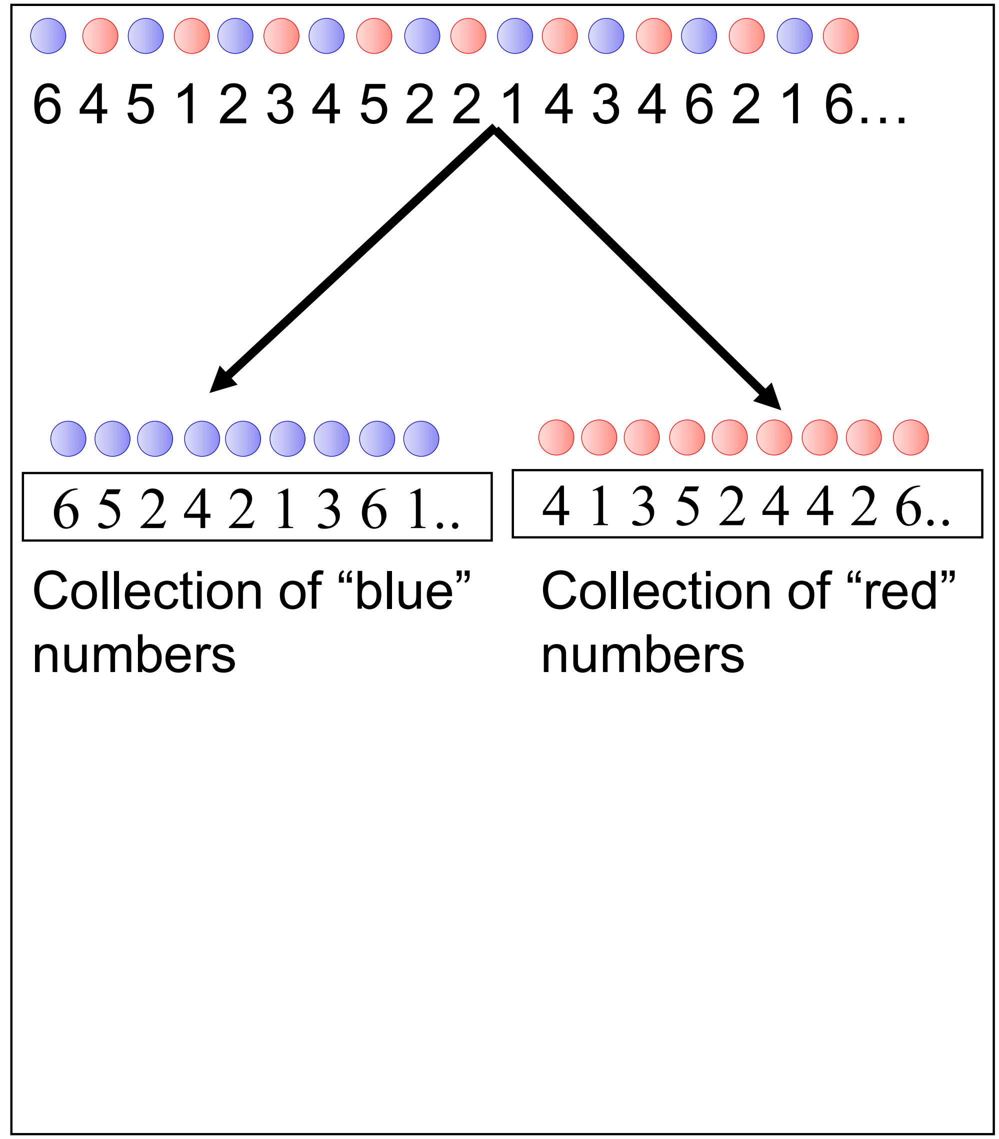
- **Ideal training data:** Each number comes with information about which dice rolled it
  - As indicated by the colors, we know who rolled what number



6 4 5 1 2 3 4 5 2 2 1 4 3 4 6 2 1 6…

Estimating probabilities with complete data

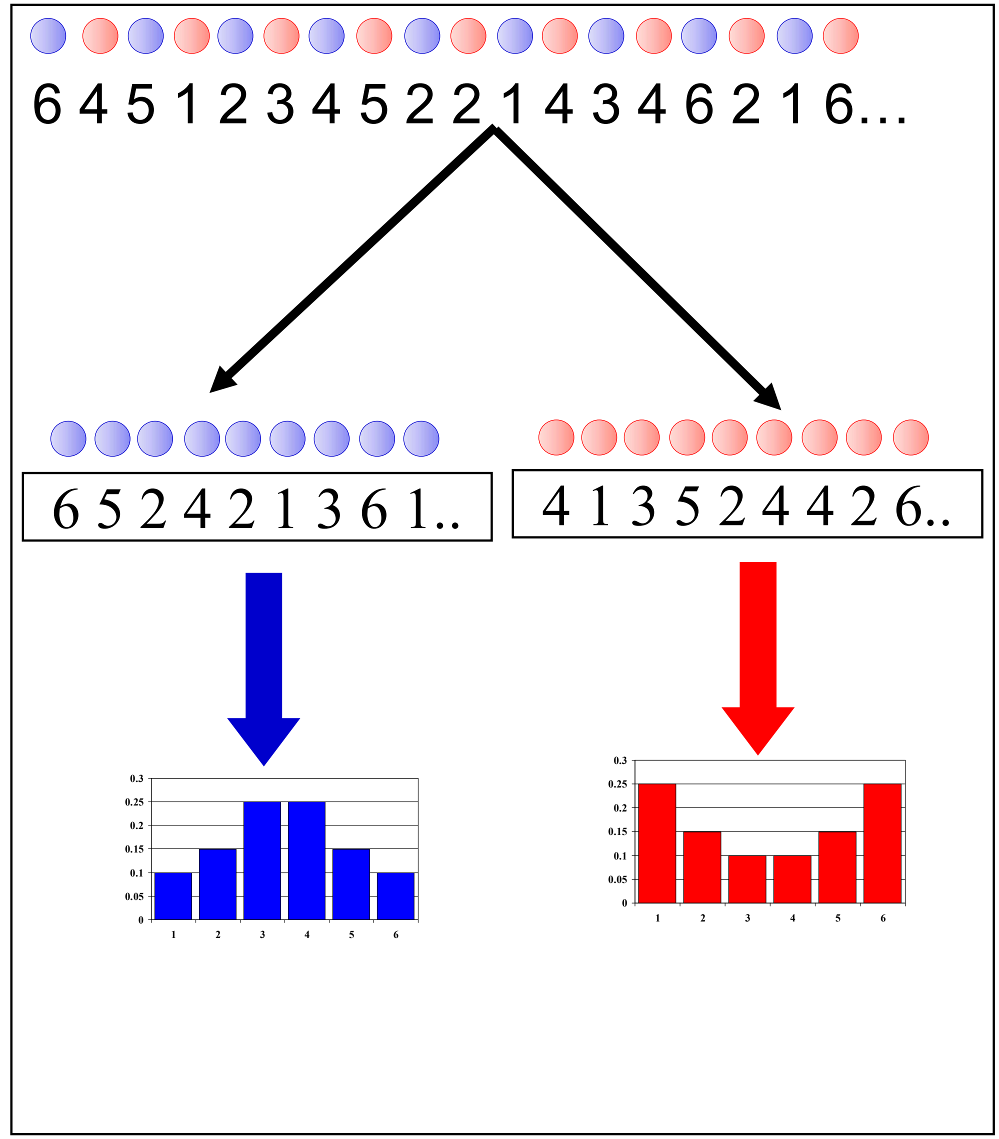
- **Ideal training data:** Each number comes with information about which dice rolled it
  - As indicated by the colors, we know who rolled what number
- Segregate numbers by "color"



Estimating probabilities with complete data

- **Ideal training data:** Each number comes with information about which dice rolled it
  - As indicated by the colors, we know who rolled what number
- Segregate numbers by "color"
- Estimate individual distributions from the separated counts

$$P(number) = \frac{\text{no. of times number was rolled}}{\text{total number of observed rolls}}$$

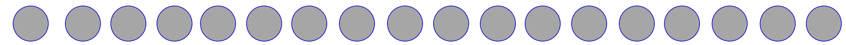


The problem

- We are not given information about which dice rolled what number
  - Our data are *incomplete*

- What we want :
  $(o_1, k_1), (o_2, k_2), (o_3, k_3)$ …

- What we have: o_1, o_2, o_3 …



6 4 5 1 2 3 4 5 2 2 1 4 3 4 6 2 1 6…

ML estimation with only *observed data*

- The maximum likelihood estimation problem:
  - Given *observed data* $\mathrm{O} = \{o_1, o_2, o_3 \ldots\}$,
  - estimate $P_k(o)$ – the parameters of all the multinoials

$$\underset{\{P_k(o),\forall k\}}{\operatorname{argmax}} \log(P(O)) = \underset{\{P_k(o),\forall k\}}{\operatorname{argmax}} \sum_{o \in O} \log P(o)$$

ML estimation with only *observed data*

- The maximum likelihood estimation problem:
  - Given *observed data* $\mathrm{O} = \{o_1, o_2, o_3 \ldots\}$,
  - estimate $P_k(o)$ – the parameters of all the multinoials

$$\underset{\{P_k(o),\forall k\}}{\operatorname{argmax}} \log(P(O)) = \underset{\{P_k(o),\forall k\}}{\operatorname{argmax}} \sum_{o\in O} \log P(o)$$

- The probability of an individual vector:

$$P(o) = \sum_{k} P(k)P_k(o)$$

ML estimation with only *observed data*

- The maximum likelihood estimation problem:
  - Given *observed data* $\mathrm{O} = \{o_1, o_2, o_3 \ldots\}$,
  - estimate $P_k(o)$ – the parameters of all the multinoials

$$\underset{\{P_k(o),\forall k\}}{\operatorname{argmax}} \log(P(O)) = \underset{\{P_k(o),\forall k\}}{\operatorname{argmax}} \sum_{o \in O} \log P(o)$$

- The probability of an individual vector:

$$P(o) = \sum_k P(k) P_k(o)$$

- The maximum likelihood estimation again

$$\underset{\{P_k(o)\forall k\}}{\operatorname{argmax}} \sum_{o \in O} \log \sum_k P(k) P_k(o)$$

ML estimation with only *observed data*

- The maximum likelihood estimation problem:
  - Given *observed data* $\mathrm{O} = \{o_1, o_2, o_3 \dots\}$,
  - estimate $P_k(o)$ – the parameters of all the multinoials

$$\underset{\{P_k(o),\forall k\}}{\operatorname{argmax}} \log(P(O)) = \underset{\{P_k(o),\forall k\}}{\operatorname{argmax}} \sum_{o \in O} \log P(o)$$

- The probability of an individual vector:

$$P(o) = \sum_k P(k) P_k(o)$$

- The maximum likelihood estimation again

$$\underset{\{P_k(o)\forall k\}}{\operatorname{argmax}} \sum_{o \in O} \underbrace{\log \sum_k P(k) P_k(o)}$$

- This includes the log of a sum, which defies direct optimization

Let's Look at Missing Information

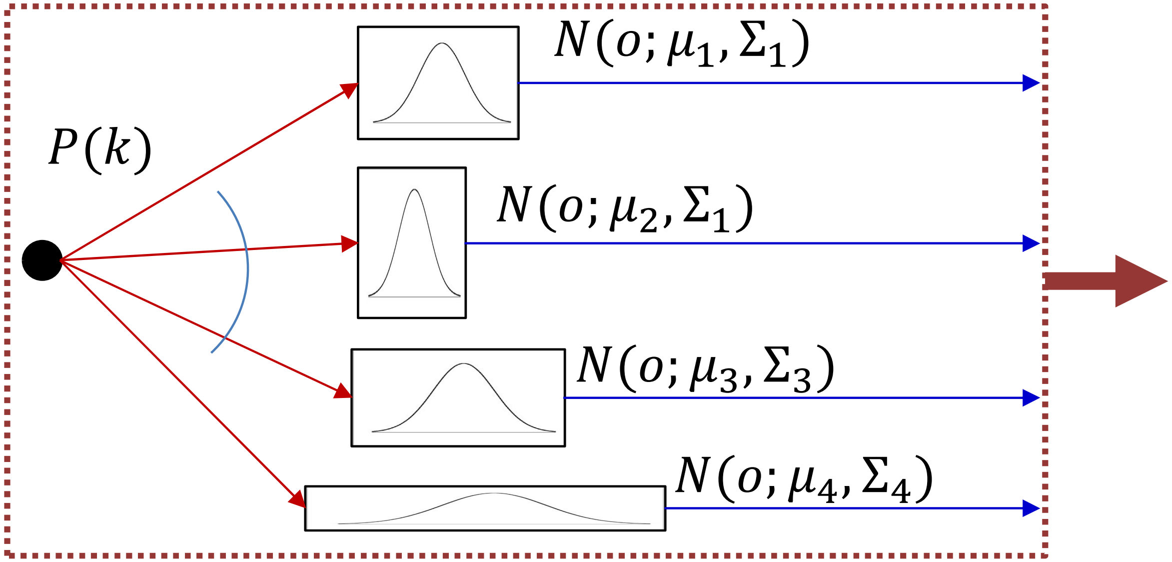
Missing Information
about **Underlying Data**

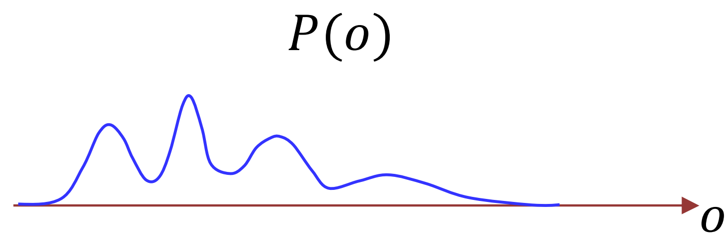
Missing Information
about **Underlying Process**

Shooting Dice

General Mixtures

Examples of incomplete data: missing information in Gaussian mixtures



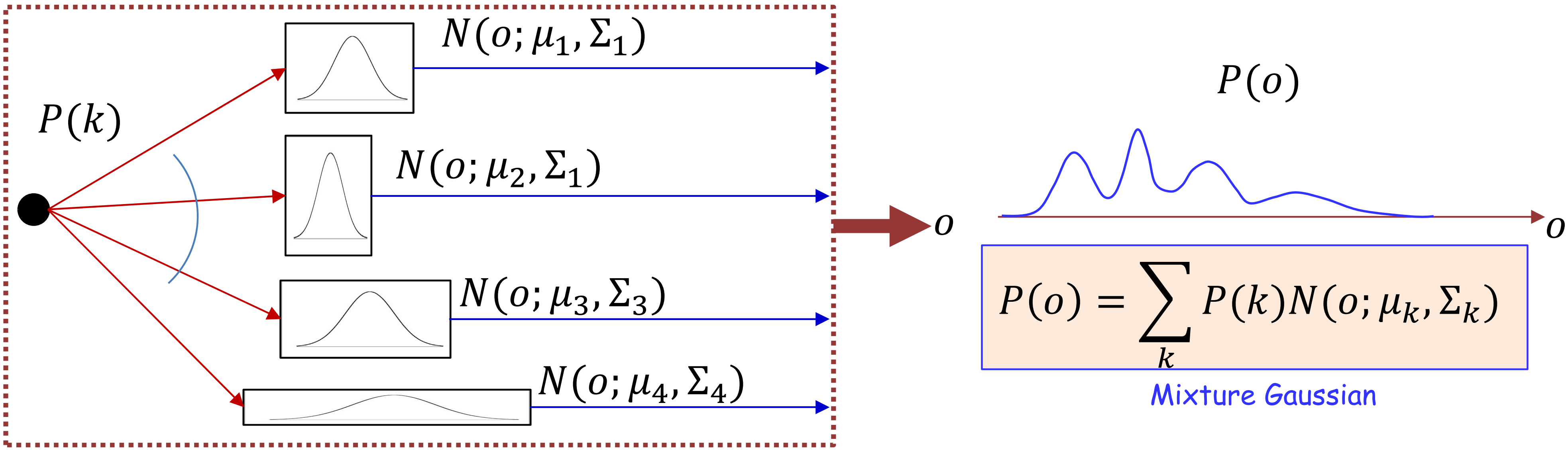


$$P(o) = \sum_k P(k)N(o;\mu_k,\Sigma_k)$$

Mixture Gaussian

- The generative model characterizes the data as the outcome of a two-level process
  - In the first step the process chooses a Gaussian from a collection
  - In the second, it draws the vector o from the chosen Gaussian
  - The overall model is a *mixture Gaussian*
- Objective: Learn the parameters of all the Gaussians from training data
  - Learn the means and variances of the individual Gaussians
    - And also the probability with which each Gaussian is selected for the draw

The Gaussian Mixture generative model



- Note, the process actually draws *two* variables for each observation, k and o.

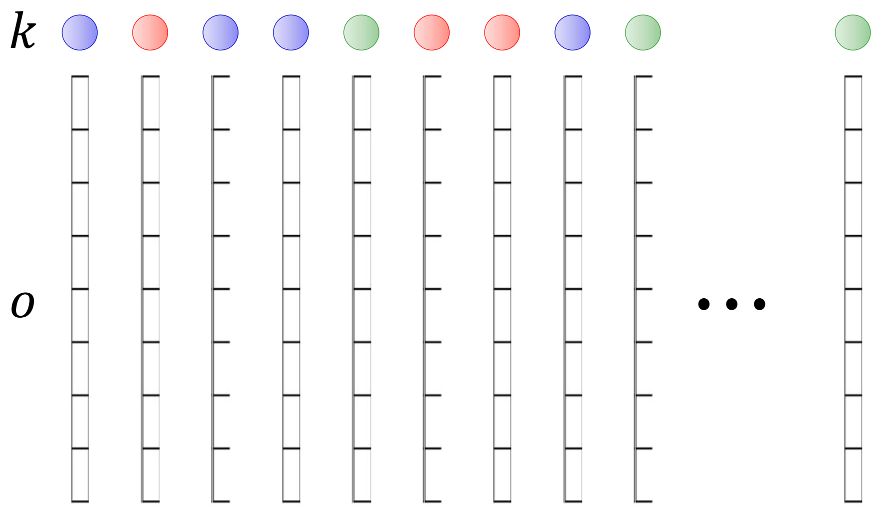
- The probability of a particular draw is actually the joint probability of both variables

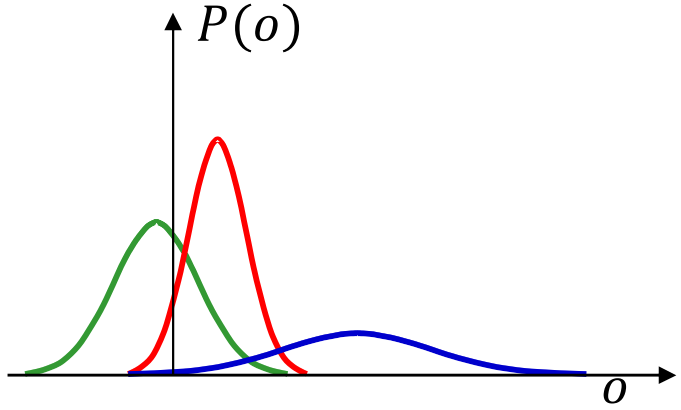
$$P(k,o) = P(k)P(o|k) = P(k)N(o;\mu_k,\Sigma_k)$$

- To compute the probability of obtaining any observation o, we are *marginalizing out* the Gaussian index variable

$$P(o) = \sum_k P(k,o) = \sum_k P(k)N(o;\mu_k,\Sigma_k)$$

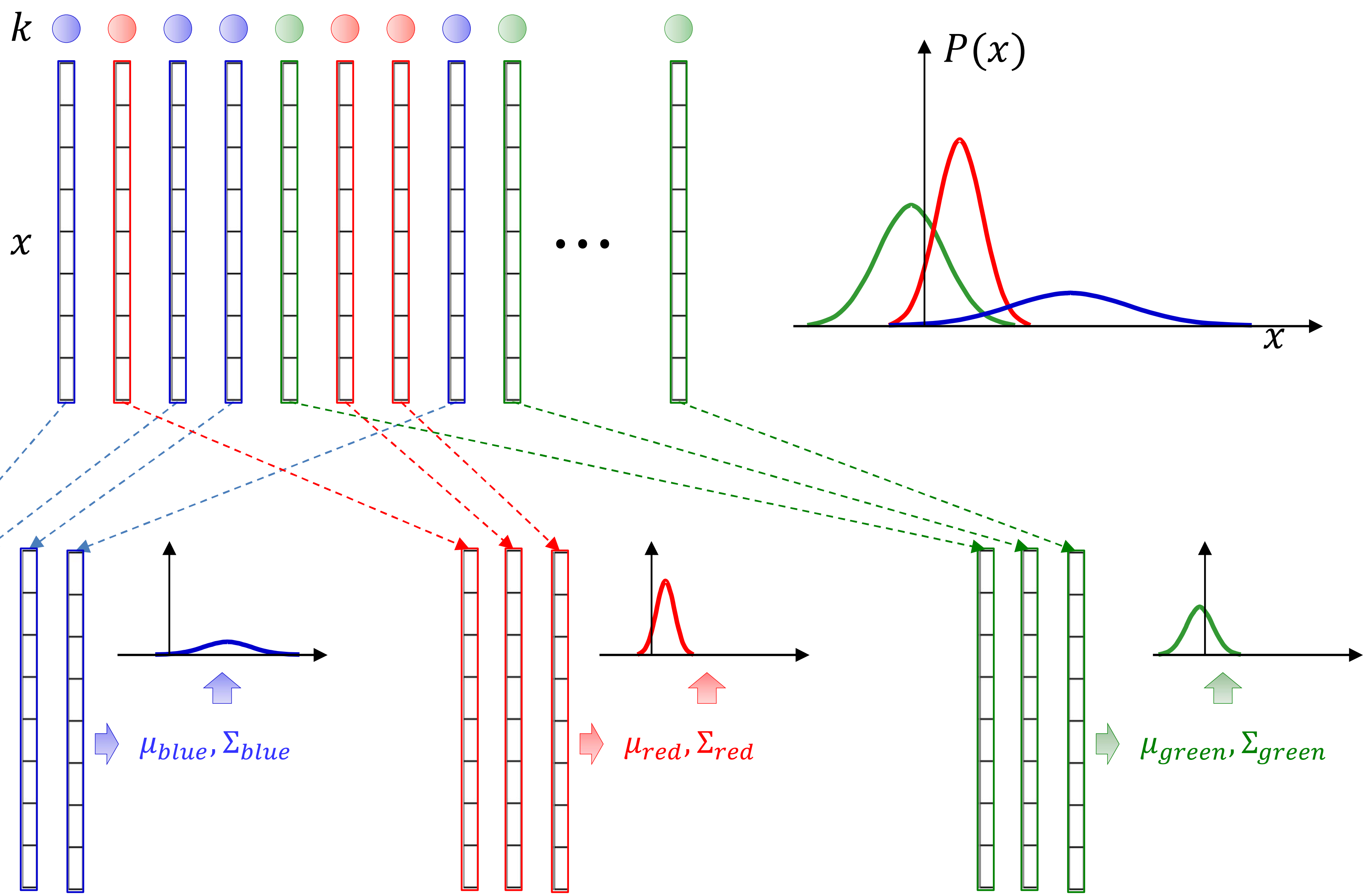
The *complete* data needed to precisely learn the model



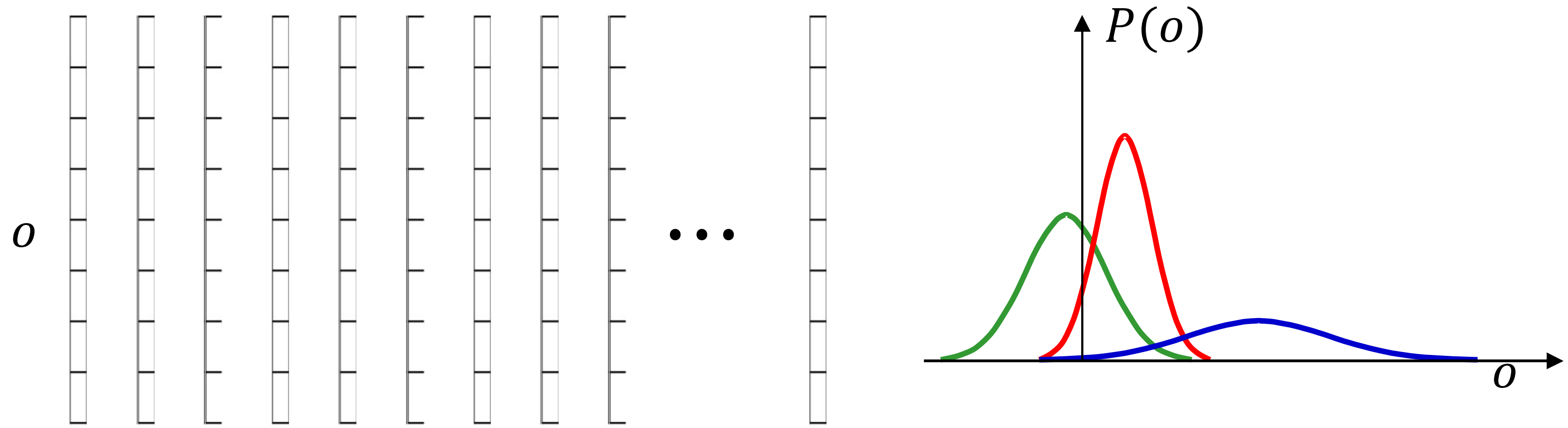


- Ideal data: Each training instance includes both the data vector o and the Gaussian k it was drawn from
  - In order to estimate the parameters of any Gaussian, you only need to segregate the training instances from that Gaussian, and compute the mean and variance from them

Learning a GMM with “complete” data



The GMM problem of incomplete data: missing information



- Problem : We are not given the actual Gaussian for each observation
  - Our data are incomplete

- What we want : $(o_1, k_1), (o_2, k_2), (o_3, k_3)$ ...
- What we have: o_1, o_2, o_3 ...

ML estimation with only *observed data*

- The maximum likelihood estimation problem:
  - Given *observed data* $\mathrm{O} = \{o_1, o_2, o_3 \ldots\}$,
  - estimate $\{(\mu_k, \Sigma_k), \forall k\}$ – the parameters of all the Gaussians

$$\underset{\{(\mu_k,\Sigma_k),\forall k\}}{\operatorname{argmax}} \log(P(O)) = \underset{\{(\mu_k,\Sigma_k),\forall k\}}{\operatorname{argmax}} \sum_{o \in O} \log P(o)$$

ML estimation with only *observed data*

- The maximum likelihood estimation problem:
  - Given *observed data* $O = \{o_1, o_2, o_3 \ldots\}$,
  - estimate $\{(\mu_k, \Sigma_k), \forall k\}$ – the parameters of all the Gaussians

$$\underset{\{(\mu_k,\Sigma_k),\forall k\}}{\operatorname{argmax}} \log(P(O)) = \underset{\{(\mu_k,\Sigma_k),\forall k\}}{\operatorname{argmax}} \sum_{o \in O} \log P(o)$$

- The probability of an individual vector:

$$P(o) = \sum_k P(k) N(o; \mu_k, \Sigma_k)$$

ML estimation with only *observed data*

- The maximum likelihood estimation problem:
  - Given *observed data* $O = \{o_1, o_2, o_3 \dots\}$,
  - estimate $\{(\mu_k, \Sigma_k), \forall k\}$ – the parameters of all the Gaussians

$$\underset{\{(\mu_k,\Sigma_k),\forall k\}}{\operatorname{argmax}} \log(P(O)) = \underset{\{(\mu_k,\Sigma_k),\forall k\}}{\operatorname{argmax}} \sum_{o \in O} \log P(o)$$

- The probability of an individual vector:

$$P(o) = \sum_k P(k) N(o; \mu_k, \Sigma_k)$$

- The maximum likelihood estimation again

$$\underset{\{(\mu_k,\sigma_k^2),\forall k\}}{\operatorname{argmax}} \sum_{o \in O} \log \sum_k P(k) N(o; \mu_k, \Sigma_k)$$

ML estimation with only *observed data*

- The maximum likelihood estimation problem:
  - Given *observed data* $O = \{o_1, o_2, o_3 \dots\}$,
  - estimate $\{(\mu_k, \Sigma_k), \forall k\}$ – the parameters of all the Gaussians

$$\underset{\{(\mu_k,\Sigma_k),\forall k\}}{\operatorname{argmax}} \log(P(O)) = \underset{\{(\mu_k,\Sigma_k),\forall k\}}{\operatorname{argmax}} \sum_{o \in O} \log P(o)$$

- The probability of an individual vector:

$$P(o) = \sum_k P(k) N(o; \mu_k, \Sigma_k)$$

- The maximum likelihood estimation again

$$\underset{\{(\mu_k,\sigma_k^2),\forall k\}}{\operatorname{argmax}} \sum_{o \in O} \underbrace{\log \sum_k P(k) N(o; \mu_k, \Sigma_k)}$$

- This includes the log of a sum, which defies direct optimization

The general form of the problem

- The "presence" of missing data or variables requires them to be marginalized out of your probability
  - By summation or integration

- This results in a maximum likelihood estimate of the form

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \sum_{o} \log \sum_{h} P\,(h, o; \theta)$$

  - The inner summation may also be an integral in some problems
  - Explicitly introducing θ in the RHS to show that the probability is computed by a model with parameter θ which must be estimated

- The log of a sum (or integral) makes estimation challenging
  - No closed form solution
  - Need efficient iterative algorithms

Poll 2: tinyurl.com/mlsp23-20231102-2

- Select all that are true
  - MLE with missing data yields problems that have closed-form solutions
  - There are no closed form solutions due to the integral inside the log, but if there were a sum inside the log, there would be a closed form solution
  - MLE with missing data maximizes the likelihood of the observations only
  - The likelihood of the observed data is derived from the complete data by marginalizing out the missing data

Poll 2

- Select all that are true
  - MLE with missing data yields problems that have closed-form solutions
  - There are no closed form solutions due to the integral inside the log, but if there were a sum inside the log, there would be a closed form solution
  - **MLE with missing data maximizes the likelihood of the observations only**
  - **The likelihood of the observed data is derived from the complete data by marginalizing out the missing data**

The general form of the problem

- The "presence" of missing data or variables requires them to be marginalized out of your probability
  - By summation or integration

Can we get an approximation to this that is more tractable?
(i.e without a summation or integral within the log)

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \sum_{o} \log \sum_{h} P(h, o)$$

  - The inner summation may also be an integral in some problems

- The log of a sum (or integral) makes estimation challenging
  - No closed form solution
  - Need efficient iterative algorithms

The variational lower bound

- We can rewrite

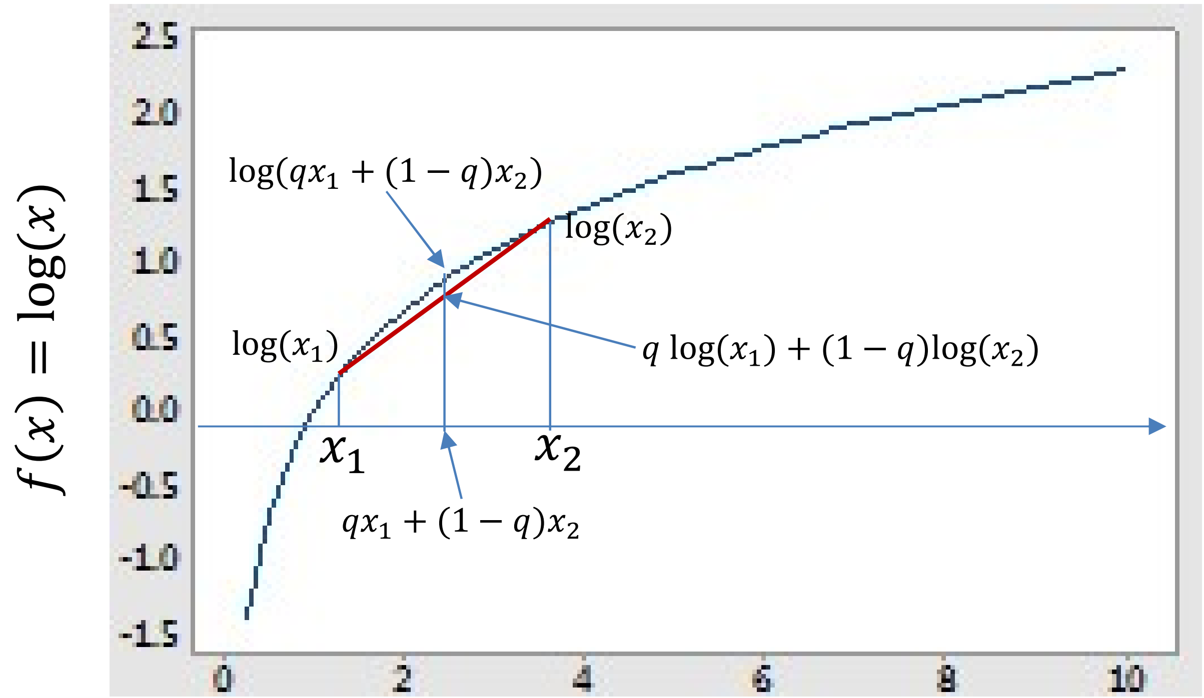
$$\log P(o) = \log \sum_h P\,(h, o) = \log \sum_h Q(h)\ \frac{P(h,o)}{Q(h)}$$

  - Where $Q(h)$ is some function such that $Q(h) \geq 0$ and $\sum_h Q(h) = 1$
    - I.e. a probability distribution

- The logarithm is a concave function, therefore

$$\log \sum_h Q(h)\ \frac{P(h,o)}{Q(h)} \geq \sum_h Q(h)\ \log \frac{P(h,o)}{Q(h)}$$

The logarithm is a concave function



- For any $x_1$and x_2, for any $0 \leq q \leq 1$,

$$\log(qx_1 + (1-q)x_2) \geq q\log(x_1) + (1-q)\log(x_2)$$

- More generally for any set of $\{x_i\}$, and any weights $\{q_i\}$ s.t. $q_i \geq 0$ and $\sum_i q_i = 1$

$$\log\left(\sum_i q_i x_i\right) \geq \sum_i q_i \log(x_i)$$

The variational lower bound

- By the concavity of the log function

$$\log \sum_h Q(h) \, \frac{P(h,o)}{Q(h)} \geq \sum_h Q(h) \log \frac{P(h,o)}{Q(h)}$$

  – For any $Q(h) \geq 0$ and $\sum_h Q(h) = 1$
  – Note, the LHS is exactly equal to $\log P(o)$

- This is the *variational lower bound* on $\log P(o)$
  – Also called the Evidence Lower BOund, or ELBO

Or more explicitly

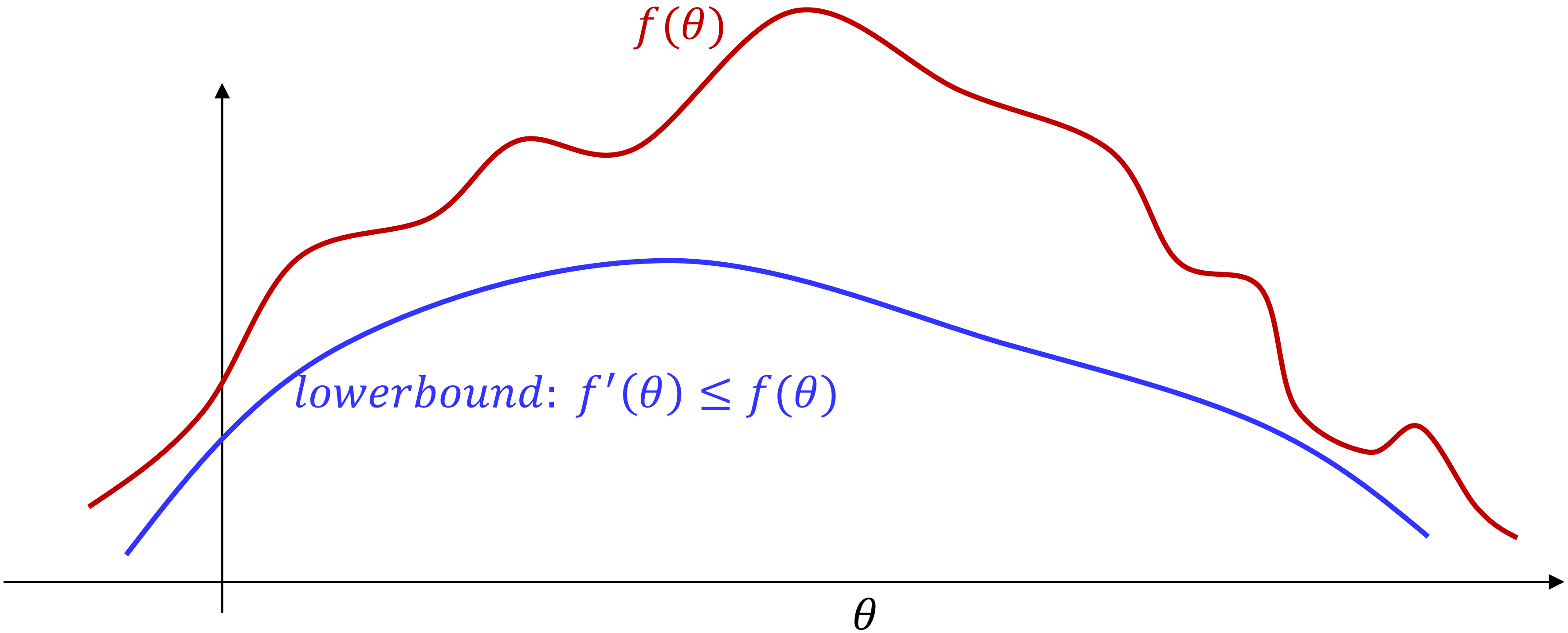
- By the concavity of the log function

$$\log P(o;\theta) \geq \sum_h Q(h) \log \frac{P(h,o;\theta)}{Q(h)}$$

  – Explicitly showing that the probability is computed by a model with parameter θ
    - We must maximize $P(o;\theta)$ w.r.t θ

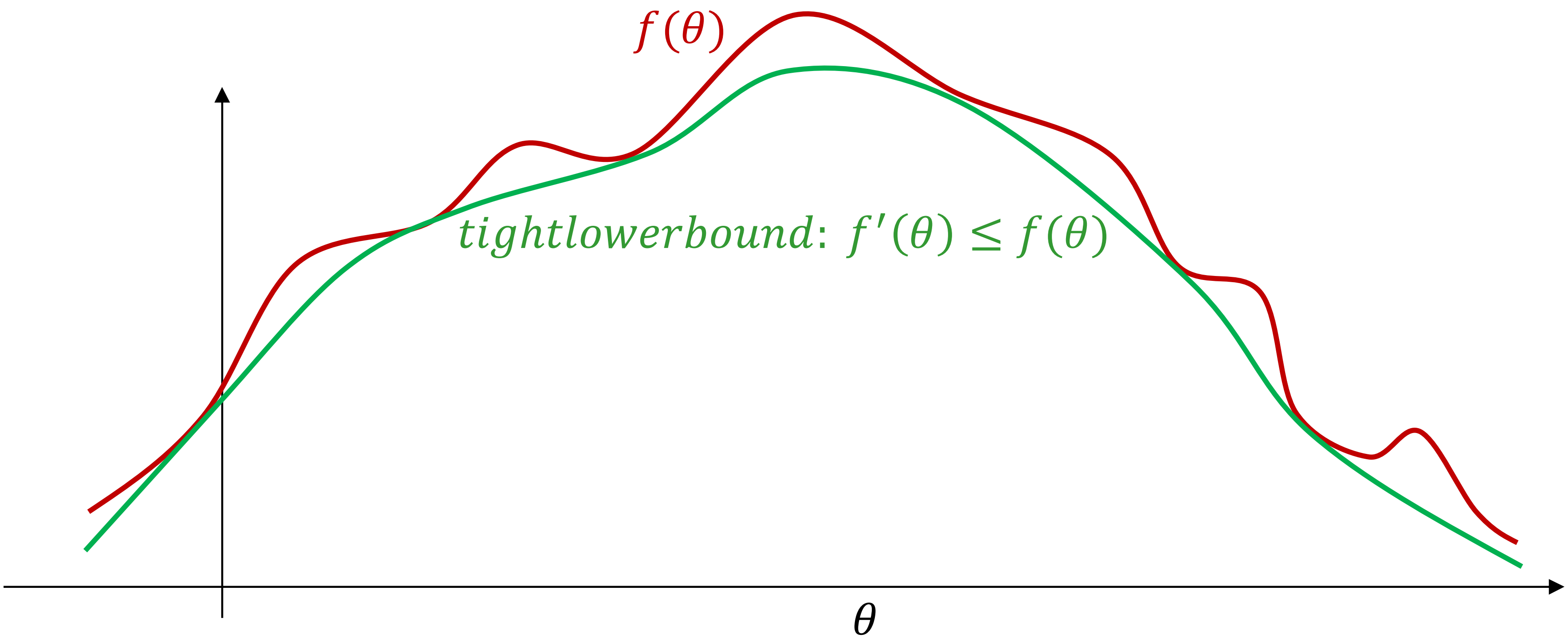
- This is the *variational lower bound* or ELBO on $\log P(o;\theta)$

The (variational) lower bound



- The lower bound is always at or below the original function

The (variational) lower bound



- The lower bound is always at or below the original function
- If it is a tight lower bound, the max of the lower bound can be expected to be near the max of the function
  - To make the lower bound tight, we need to choose $Q(h)$ properly

Choosing a good $Q(h)$

- Let $Q(h) = P(h|o;\theta')$

$$\log P(o;\theta) \geq \sum_h P(h|o;\theta') \log \frac{P(h,o;\theta)}{P(h|o;\theta')}$$

- Let

$$J(\theta,\theta') = \sum_h P(h|o;\theta') \log \frac{P(h,o;\theta)}{P(h|o;\theta')}$$

- We get

$$\log P(o;\theta) \geq J(\theta,\theta')$$

- And

$$\log P(o;\theta) = J(\theta,\theta)$$

Choosing a good $Q(h)$

- Let $Q(h) = P(h|o;\theta')$

$$\log P(o;\theta) \geq \sum_h P(h|o;\theta') \log \frac{P(h,o;\theta)}{P(h|o;\theta')}$$

- Let

$$J(\theta,\theta') = \sum_h P(h|o;\theta') \log \frac{P(h,o;\theta)}{P(h|o;\theta')}$$

- We get

$$\log P(o;\theta) \geq J(\theta,\theta')$$

- And

$$\log P(o;\theta) = J(\theta,\theta)$$

$$P(o;\theta) = J(\theta,\theta)$$

$$J(\theta,\theta) = \sum_h P(h|o;\theta)\log\frac{P(h,o;\theta)}{P(h|o;\theta)}$$

$$= \sum_h P(h|o;\theta)\log P(o;\theta)$$

$$\log P(o;\theta)\sum_h P(h|o;\theta) \quad = \log P(o;\theta)$$

Expectation Maximization

- We have

$$J(\theta, \theta') = \sum_h P(h|o;\theta') \log \frac{P(h,o;\theta)}{P(h|o;\theta')}$$

- where

$$\log P(o;\theta) \geq J(\theta, \theta')$$

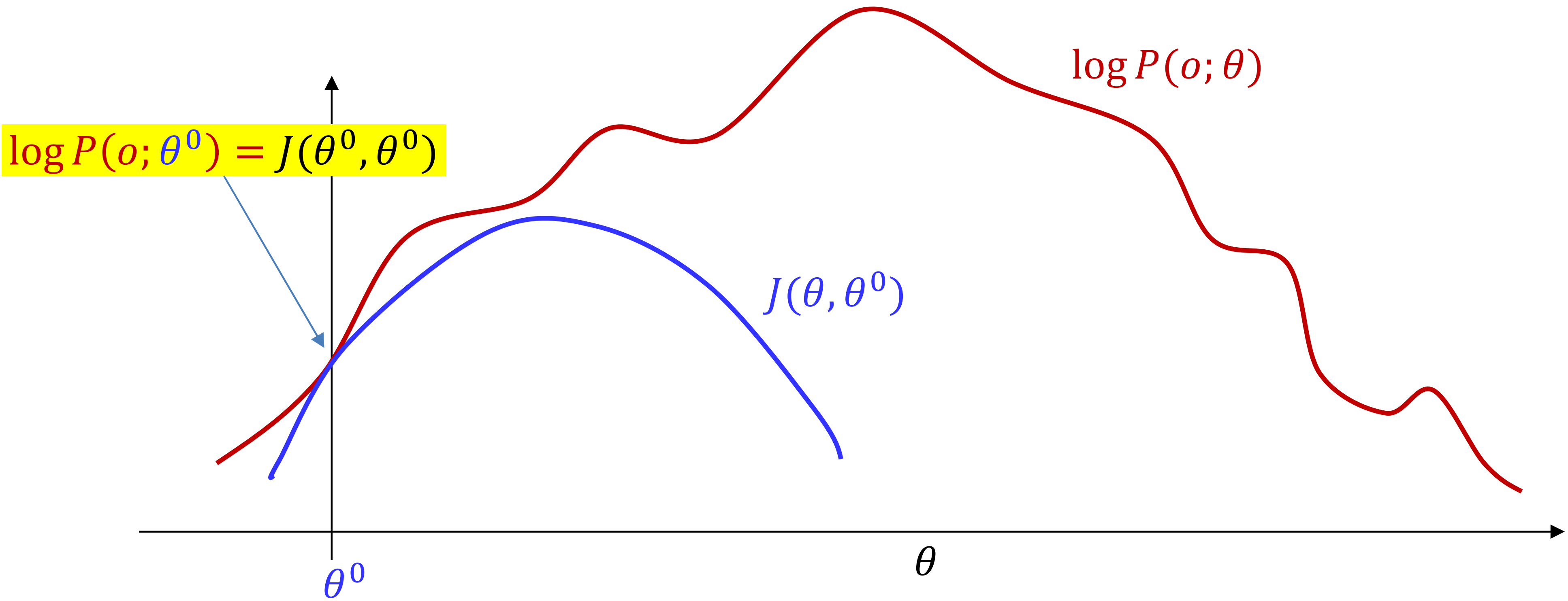
- And

$$\log P(o;\theta) = J(\theta, \theta)$$

- This gives us the following iterative algorithm that guarantees non-decreasing $P(o;\theta)$ with iterations:

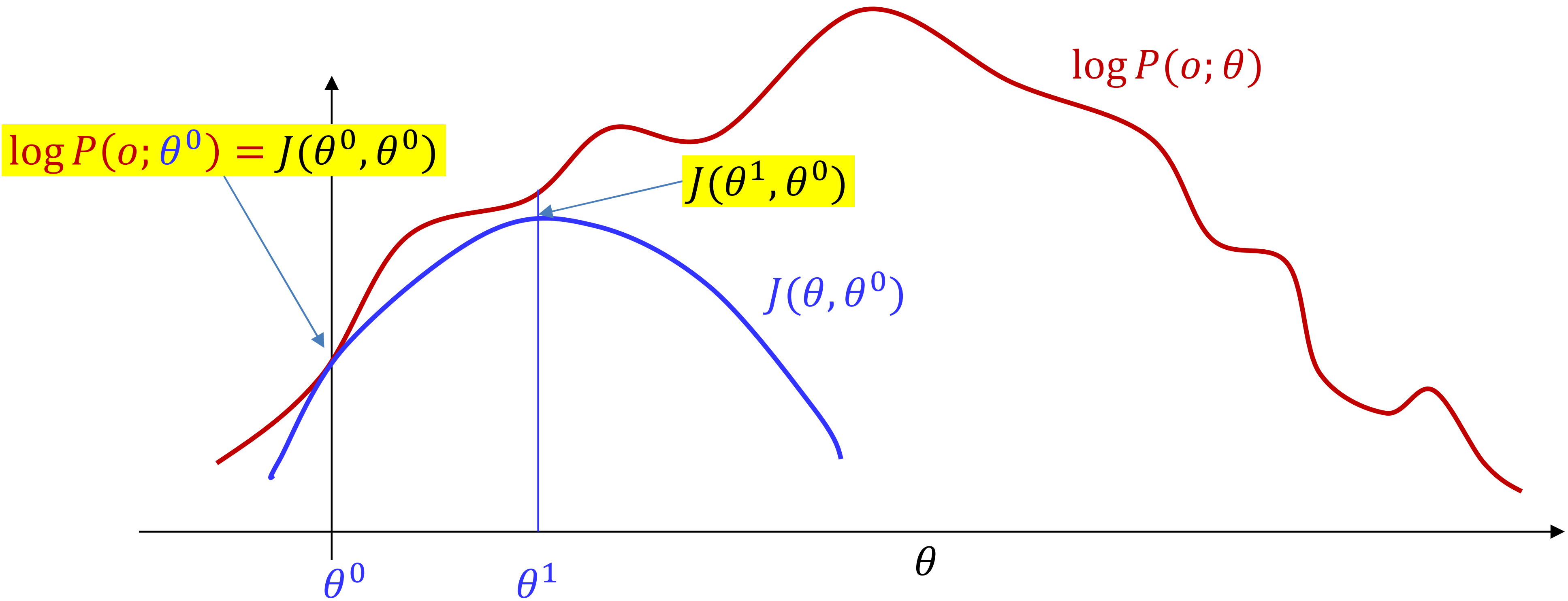
$$\theta^{k+1} \leftarrow \underset{\theta}{\operatorname{argmax}} J\left(\theta, \theta^k\right)$$

$$\theta^{k+1} \leftarrow \underset{\theta}{\operatorname{argmax}} J(\theta, \theta')$$



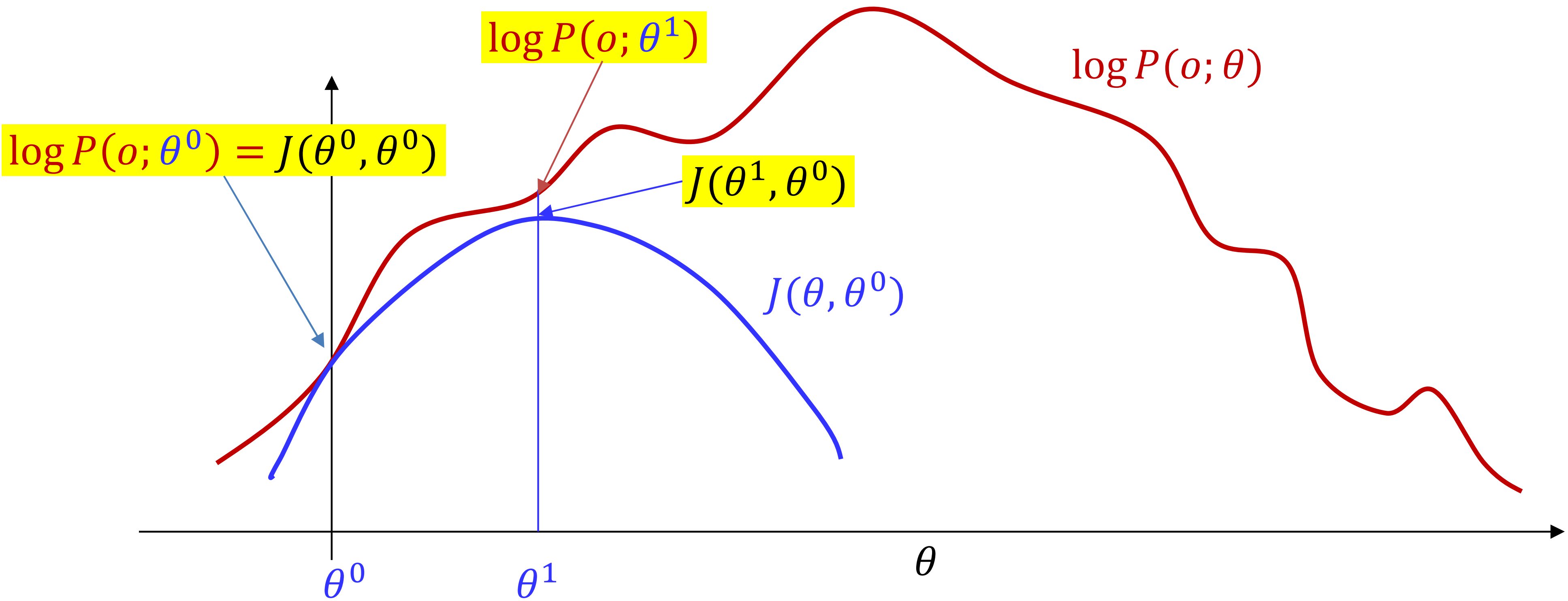
- Initialize θ^0
- Construct $J(\theta, \theta^0)$
  - It touches $\log P(o;\theta)$ at θ^0 because $\log P(o;\theta^0) = J(\theta^0, \theta^0)$

$$\theta^{k+1} \leftarrow \underset{\theta}{\operatorname{argmax}} J(\theta, \theta')$$



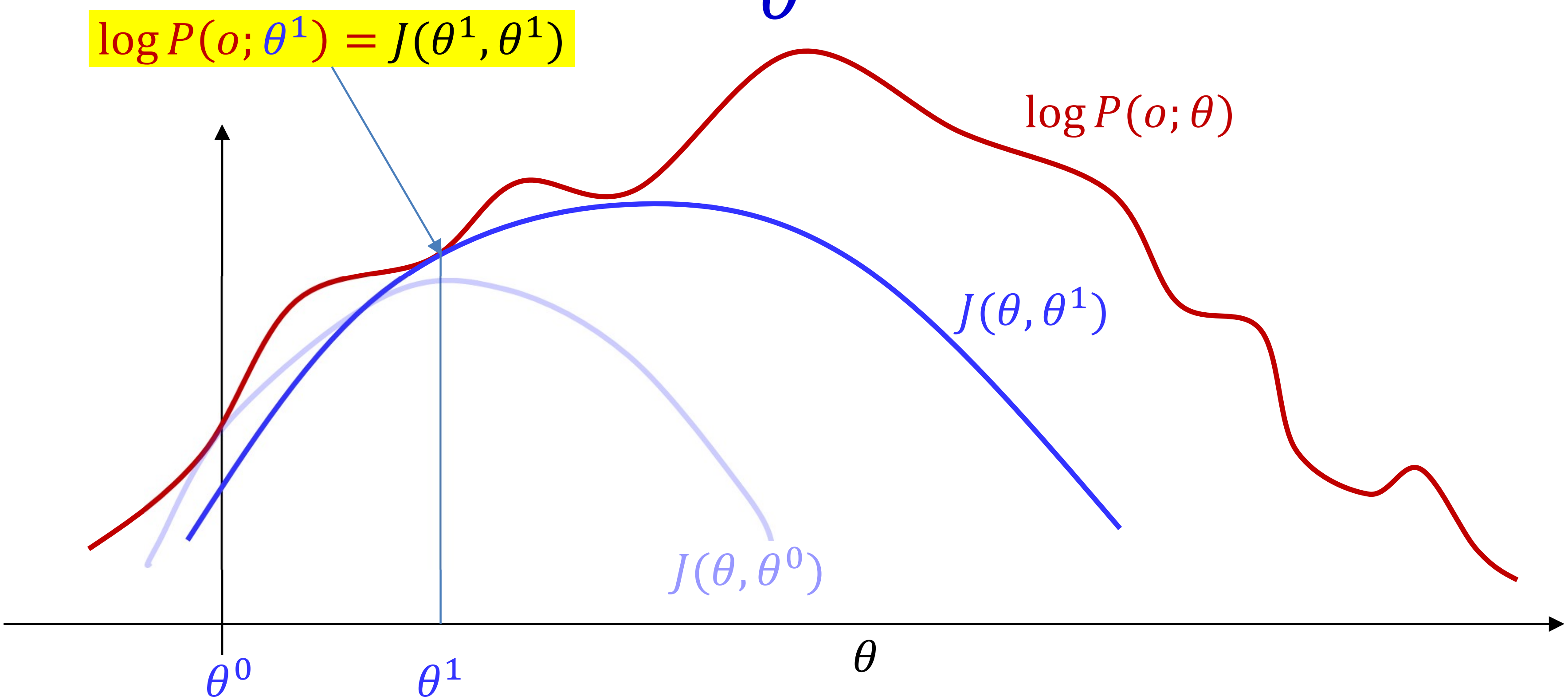
- Find $\theta^1 = \underset{\theta}{\operatorname{argmax}} J(\theta, \theta^0)$
  - $J(\theta^1, \theta^0) \geq J(\theta^0, \theta^0)$ (since you're maximizing $J(\theta, \theta^0)$ w.r.t θ)

$$\theta^{k+1} \leftarrow \underset{\theta}{\operatorname{argmax}} J(\theta, \theta')$$



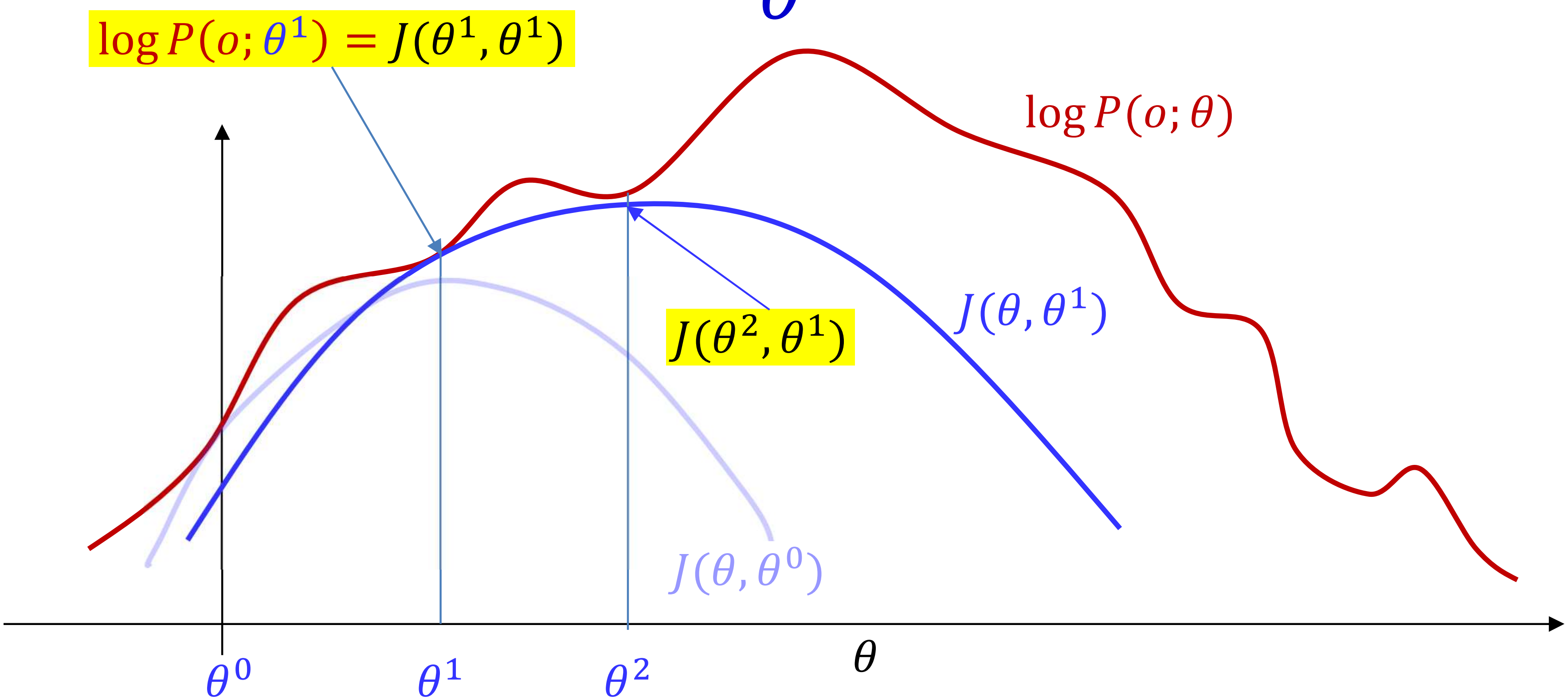
- Find $\theta^1 = \underset{\theta}{\operatorname{argmax}} J(\theta, \theta^0)$
  - $J(\theta^1, \theta^0) \geq J(\theta^0, \theta^0)$ (since you're maximizing $J(\theta, \theta^0)$ w.r.t θ)
- $\log P(o; \theta^1) \geq J(\theta^1, \theta^0)$
  - since $J(\theta, \theta^0)$ is a lower bound on $\log P(o; \theta)$
- So the iteration increases $\log P(o; \theta)$

$$\theta^{k+1} \leftarrow \underset{\theta}{\operatorname{argmax}} J(\theta, \theta')$$



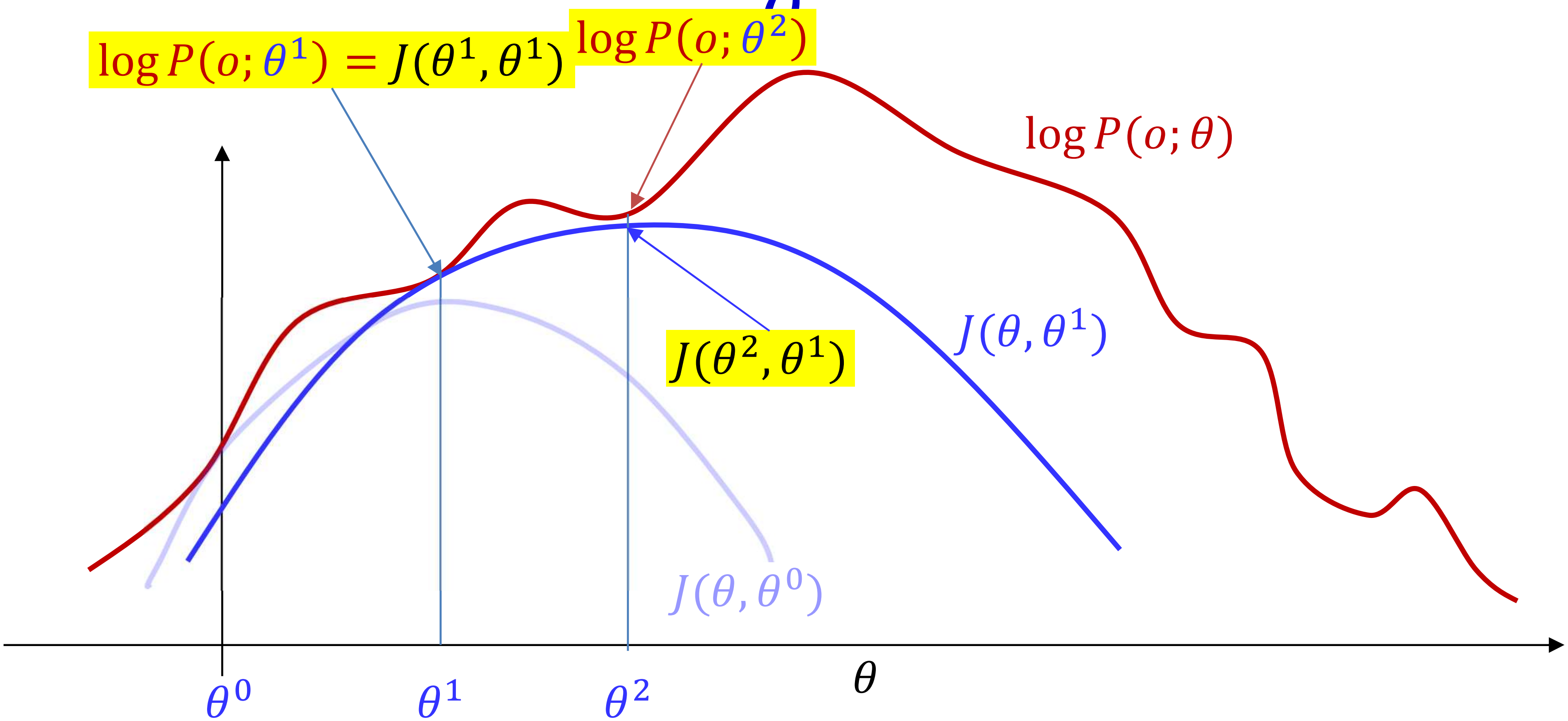
- Construct $J(\theta, \theta^1)$
  - It touches $\log P(o; \theta)$ at θ^1 because $\log P(o; \theta^1) = J(\theta^1, \theta^1)$

$$\theta^{k+1} \leftarrow \underset{\theta}{\operatorname{argmax}} J(\theta, \theta')$$



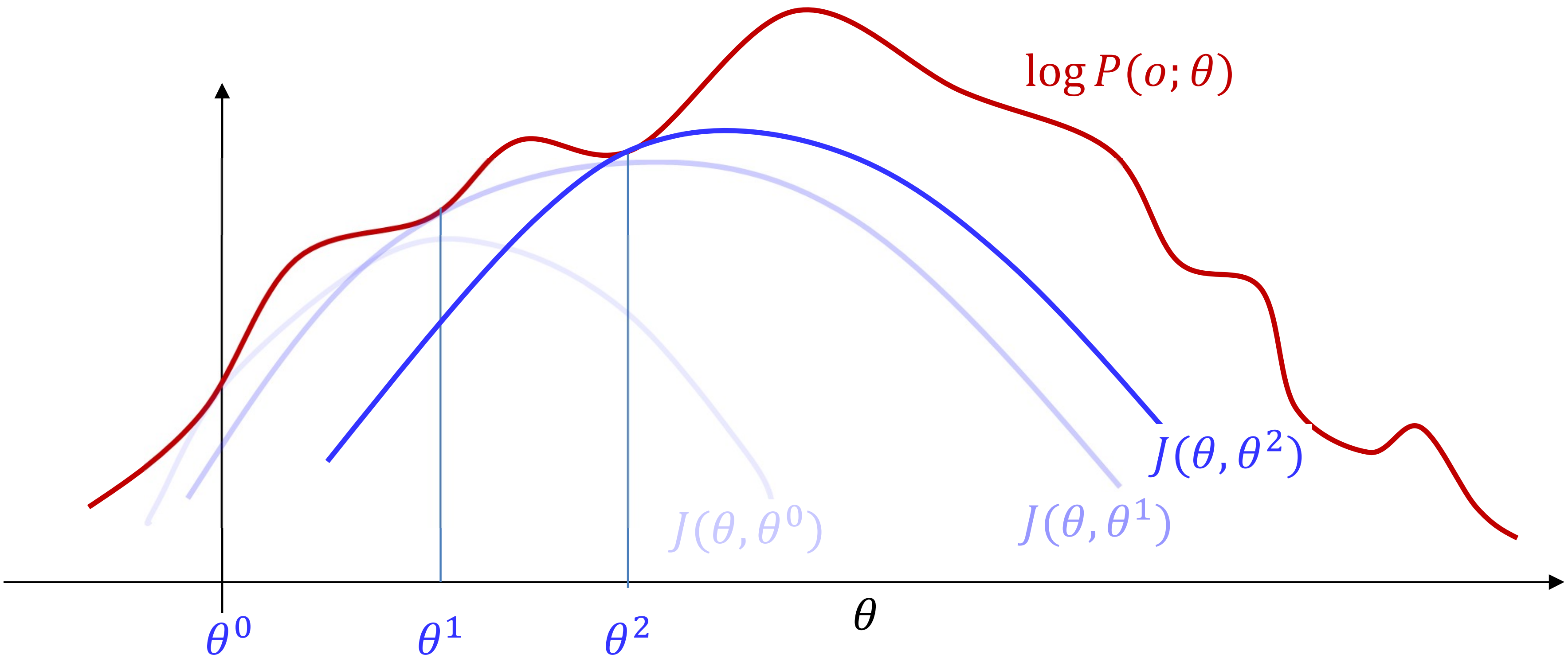
- Find $\theta^2 = \underset{\theta}{\operatorname{argmax}} J(\theta, \theta^1)$
  - $J(\theta^2, \theta^1) \geq J(\theta^1, \theta^1)$ (since you're maximizing $J(\theta, \theta^1)$ w.r.t θ)

$$\theta^{k+1} \leftarrow \underset{\theta}{\operatorname{argmax}} J(\theta, \theta')$$



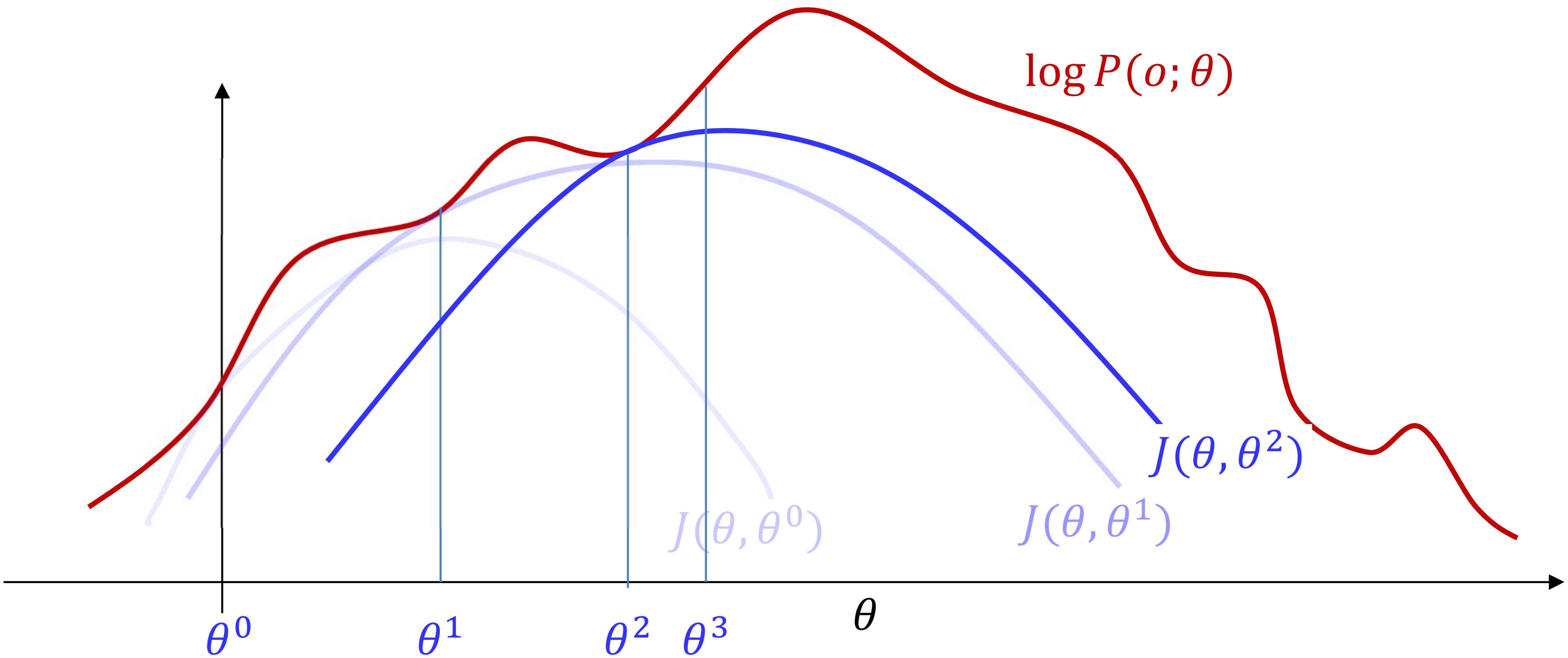
- Find $\theta^2 = \underset{\theta}{\operatorname{argmax}} J(\theta,\theta^1)$
  - $J(\theta^2,\theta^1) \geq J(\theta^1,\theta^1)$ (since you're maximizing $J(\theta,\theta^1)$ w.r.t θ)
- $\log P(o;\theta^2) \geq J(\theta^2,\theta^1)$
  - Since $J(\theta,\theta^1)$ is a lower bound on $\log P(o;\theta)$
- So the iteration increases $\log P(o;\theta)$

$$\theta^{k+1} \leftarrow \underset{\theta}{\operatorname{argmax}} J(\theta, \theta')$$



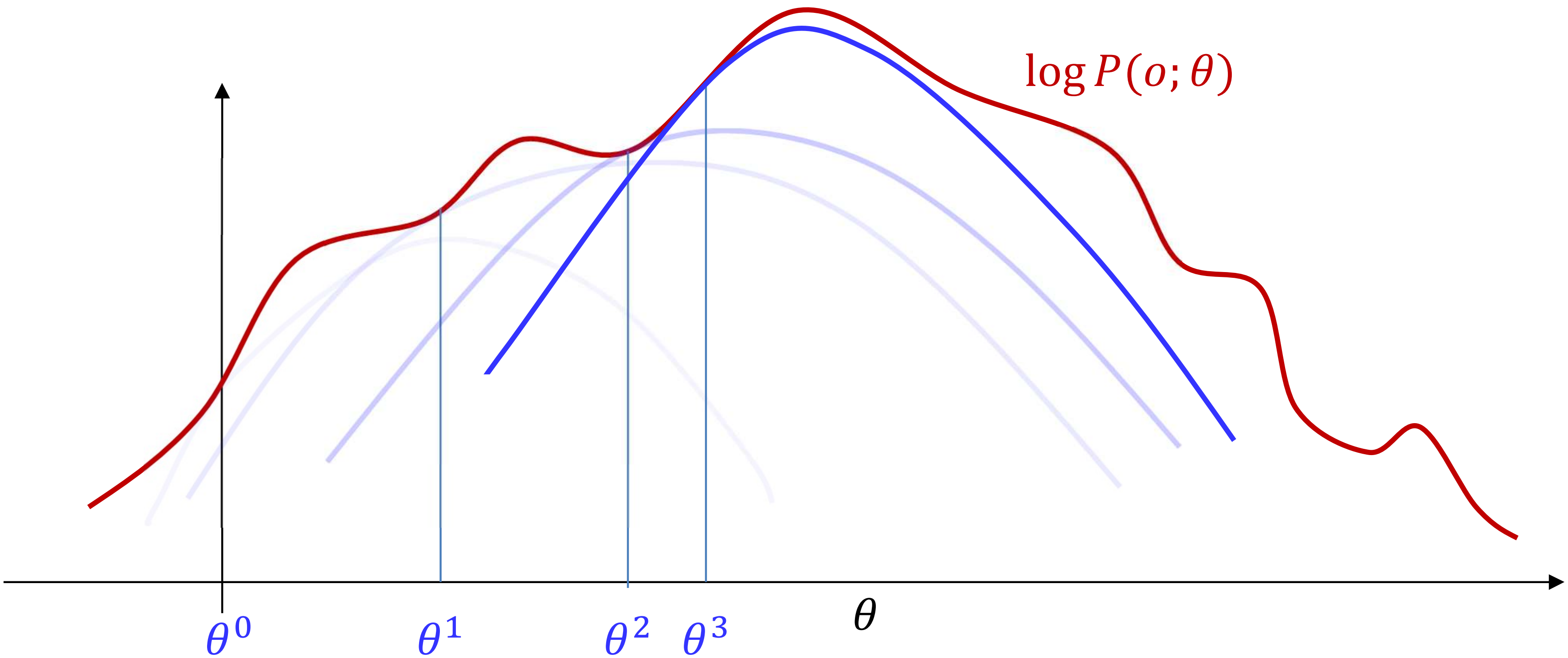
- Repeat the steps:
  - Compose $J(\theta, \theta^k)$ to "touch" $\log P(o;\theta)$ at the current estimate θ^k
  - Set $\theta^{k+1} \leftarrow \underset{\theta}{\operatorname{argmax}} J(\theta, \theta^k)$
- Each step is guaranteed to increase (or at least not decrease) $\log P(o;\theta)$
  - Stop when $\log P(o;\theta)$ stops increasing

$$\theta^{k+1} \leftarrow \underset{\theta}{\operatorname{argmax}} J(\theta, \theta')$$



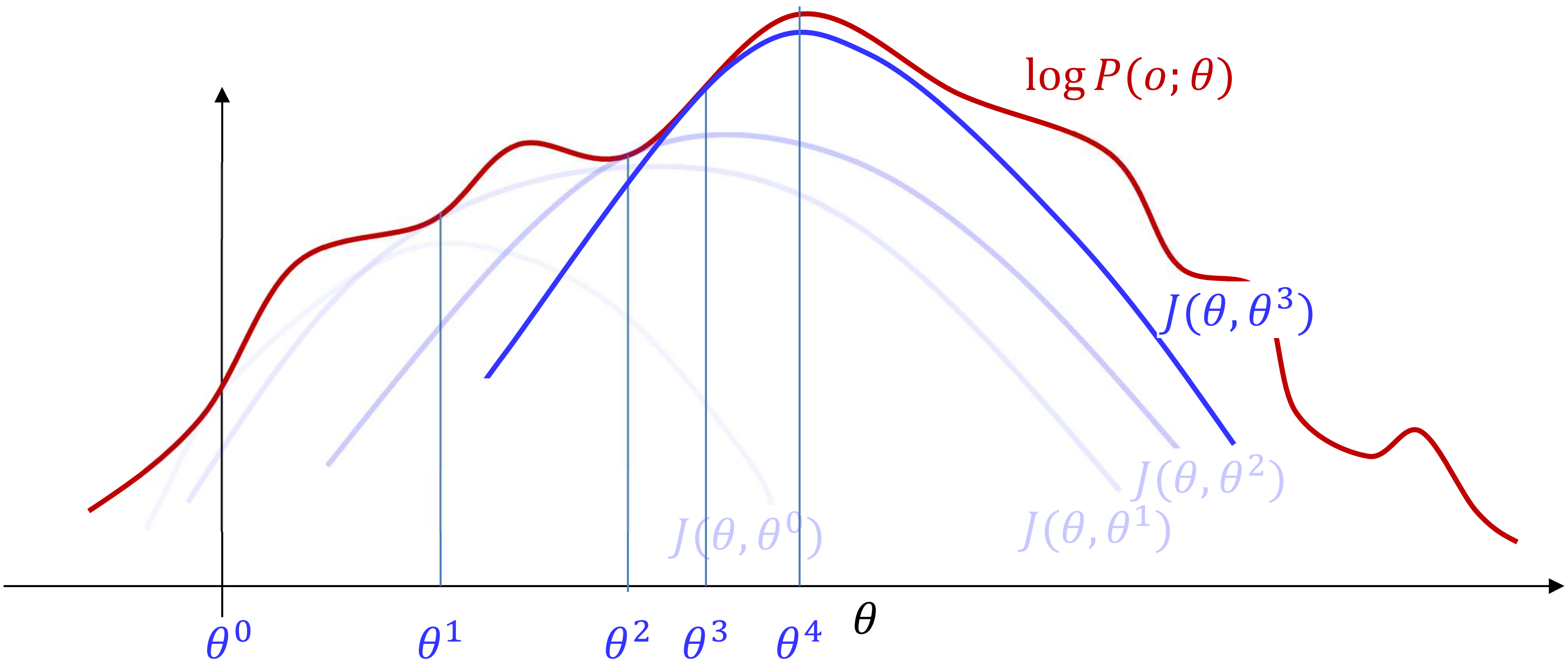
- Repeat the steps:
  - Compose $J(\theta, \theta^k)$ to "touch" $\log P(o;\theta)$ at the current estimate θ^k
  - Set $\theta^{k+1} \leftarrow \underset{\theta}{\operatorname{argmax}} J(\theta, \theta^k)$
- Each step is guaranteed to increase (or at least not decrease) $\log P(o;\theta)$
  - Stop when $\log P(o;\theta)$ stops increasing

$$\theta^{k+1} \leftarrow \underset{\theta}{\operatorname{argmax}} J(\theta, \theta')$$



- Repeat the steps:
  - Compose $J(\theta, \theta^k)$ to “touch” $\log P(o;\theta)$ at the current estimate θ^k
  - Set $\theta^{k+1} \leftarrow \underset{\theta}{\operatorname{argmax}} J(\theta, \theta^k)$
- Each step is guaranteed to increase (or at least not decrease) $\log P(o;\theta)$
  - Stop when $\log P(o;\theta)$ stops increasing

$$\theta^{k+1} \leftarrow \underset{\theta}{\operatorname{argmax}} J(\theta, \theta')$$



- Repeat the steps:
  - Compose $J(\theta, \theta^k)$ to "touch" $\log P(o;\theta)$ at the current estimate θ^k
  - Set $\theta^{k+1} \leftarrow \underset{\theta}{\operatorname{argmax}} J(\theta, \theta^k)$
- Each step is guaranteed to increase (or at least not decrease) $\log P(o;\theta)$
  - Stop when $\log P(o;\theta)$ stops increasing

Expectation Maximization

- Initialize θ^0
- $k = 0$
- Iterate (over k) until $\log P(O; \theta)$ converges:
  - Construct ELBO function

$$J(\theta, \theta^k) = \sum_{o \in O} \sum_h P(h|o; \theta^k) \log \frac{P(h, o; \theta)}{P(h|o; \theta^k)}$$

  - Maximization step

$$\theta^{k+1} \leftarrow \operatorname*{argmax}_\theta J(\theta, \theta^k)$$

- Let's simplify a bit

Expectation Maximization

- Initialize θ^0
- $k = 0$
- Iterate (over k) until $\log P(O; \theta)$ converges:
  - Construct ELBO function

$$J(\theta, \theta^k) = \sum_{o \in O} \sum_h P(h|o; \theta^k) \log P(h, o; \theta) - \sum_{o \in O} \sum_h P(h|o; \theta^k) \log P(h|o; \theta^k)$$

  - Maximization step

$$\theta^{k+1} \leftarrow \underset{\theta}{\operatorname{argmax}} J(\theta, \theta^k)$$

Expectation Maximization

- Initialize θ^0
- $k = 0$
- Iterate (over k) until $\log P(O; \theta)$ converges:
  - Construct ELBO function

Not a function of θ

$$J(\theta, \theta^k) = \sum_{o \in O} \sum_h P(h|o; \theta^k) \log P(h, o; \theta) - \sum_{o \in O} \sum_h P(h|o; \theta^k) \log P(h|o; \theta^k)$$

Can be ignored for maximization

  - Maximization step

$$\theta^{k+1} \leftarrow \underset{\theta}{\operatorname{argmax}} J(\theta, \theta^k)$$

Expectation Maximization

- Initialize θ^0
- $k = 0$
- Iterate (over k) until $\log P(O;\theta)$ converges:
  - Expectation Step:

    Compute $P(h|o;\theta^k)$ for all $o \in O$ for all h

  - Maximization step

$$\theta^{k+1} \leftarrow \underset{\theta}{\operatorname{argmax}} \sum_{o \in O} \sum_{h} P(h|o;\theta^k) \log P(h,o;\theta)$$

Expectation Maximization for Maximum Likelihood Estimation

- Objective: Estimate

$$\theta^* = \operatorname*{argmax}_\theta \sum_{o \in O} \log \sum_h P(h, o; \theta)$$

- Solution: Iteratively perform the following optimization instead

$$\theta^{k+1} \leftarrow \operatorname*{argmax}_\theta \sum_{o \in O} \sum_h P(h|o; \theta^k) \log P(h, o; \theta)$$

- This maximizes an Empirical Lower Bound (ELBO) and guarantees increasing log likelihood with iterations
  - Giving you a *local maximum log likelihood* estimate for θ^*

Expectation Maximization: In summary

- Construct an *Empirical Lower Bound* function $J\left(\theta, \theta^k\right)$

$$J\left(\theta, \theta^k\right)$$

$$= \sum_{o \in O} \sum_h P(h|o; \theta^k) \log P(h, o; \theta)$$

$$- \sum_{o \in O} \sum_h P(h|o; \theta^k) \log P(h|o; \theta^k)$$

- Iteratively maximize the ELBO fucntion

$$\theta^{k+1} \leftarrow \operatorname*{argmax}_{\theta} J\left(\theta, \theta^k\right)$$

Expectation Maximization

- Initialize θ^0
- $k = 0$
- Iterate (over k) until $\sum_{o \in O} \log P(o; \theta)$ converges:
  - Expectation Step:

    Compute $P(h|o; \theta^k)$ for all $o \in O$ for all h

  - Maximization step

$$\theta^{k+1} \leftarrow \underset{\theta}{\operatorname{argmax}} \sum_{o \in O} \sum_{h} P(h|o; \theta^k) \log P(h, o; \theta)$$

Poll 3: tinyurl.com/mlsp23-20231102-3

- EM iteratively estimates a tight “variational lower bound” to the likelihood function and maximizes it with respect to the parameters
  - True
  - False

- We could alternately compute a tight upper bound to the likelihood and minimize it
  - True
  - False

Poll 3:

- EM iteratively estimates a tight "variational lower bound" to the likelihood function and maximizes it with respect to the parameters
  - **True**
  - False

- We could alternately compute a tight upper bound to the likelihood and minimize it
  - True
  - **False**

That’s so much math, but what does it really do?

- What does EM practically do when we have missing data?
  - What is the intuition behind how it resolves the problem?

- Next class…