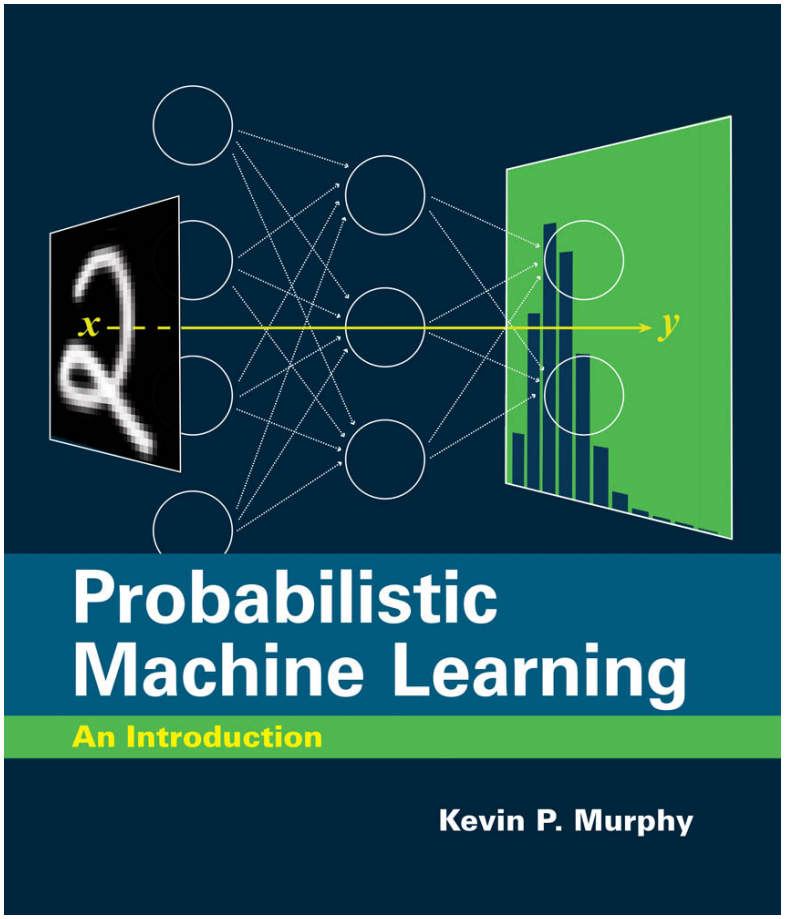
Chapter 4: Parameter learning

Lode Nachtergaele

Parameter Learning TOC

Probabilistic Machine learning



4.1 Maximum Likehood Parameter Learning

$$\hat{\theta} = \arg\max_{\theta} P(D \mid \theta)$$

where:

- θ represents the parameters of a distribution
- $P(D \mid \theta)$ is the likelihood that the probability model assigns to data D when model parameters are set to θ
- $\hat{\theta}$ is the estimate of the parameter hence the hat

Two challenges to calculate arg max

1. Choosing appropriate model by which we define $P(D \mid \theta)$
   - Assume samples of data D are *independently and identically distributed*
   - $P(D \mid \theta) = \prod_i P(o_i \mid \theta)$
2. Performing maximisation
   - For common probability models: analytically, others difficult. But log function is monotonically increasing
   - $\hat{\theta} = \arg\max_{\theta} \sum_i \log P(o_i \mid \theta)$

4.1.1 Maximum likelihood estimates for categorical distribution

$$P(D \mid \theta) = \theta^n (1-\theta)^{m-n}$$

Log likelihood:

$$\begin{aligned} l(\theta) &= \log(\theta^n (1-\theta)^{m-n}) \\ &= n \log \theta + (m-n) \log(1-\theta) \end{aligned}$$

Maximum Log likelihood

$$l(\theta) = n \log \theta + (m - n) \log(1 - \theta)$$

Set first derivative of l to zero:

$$\frac{\partial}{\partial \theta} l(\theta) = \frac{n}{\theta} - \frac{m - n}{1 - \theta}$$

Solving for $\hat{\theta}$ by setting the derivative to zero:

$$\frac{n}{\hat{\theta}} - \frac{m - n}{1 - \hat{\theta}} = 0 \implies \hat{\theta} = \frac{n}{m}$$

4.1.1 Maximum likelihood for variable X with k values

Maximum likelihood estimate $P(x^i \mid n_{1:k})$ is given by

$$\hat{\theta}_i = \frac{n_i}{\sum_{j=1}^{k} n_j}$$

Maximum likelihood gender after 3 muffins

Guppi matched you with 1 non-binary, 1 female and 2 males.
Most likely gender next muffin session:

$$\hat{\theta}_{nb} = \frac{1}{1+1+2} = 1/4$$

$$\hat{\theta}_{f} = \frac{1}{1+1+2} = 1/4$$

$$\hat{\theta}_{m} = \frac{2}{1+1+2} = 2/4$$

4.1.2 MLE for Gaussian distributions

Gaussian probability density at x:

$$\mathcal{N}(x \mid \mu, \sigma^2) = \frac{1}{\sigma}\phi\left(\frac{x-\mu}{\sigma}\right)$$

with ϕ is the standard normal density function:

$$\phi(x) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right)$$

Log likelihood Gaussian

$$l(\mu, \sigma^2) \propto -m \log \sigma - \frac{\sum_{i=1}^{m} (o_i - \mu)^2}{2\sigma^2}$$

$$\frac{\partial}{\partial \mu} l(\mu, \sigma^2) = \frac{\sum_i (o_i - \hat{\mu})}{\hat{\sigma}^2} = 0$$

$$= -\frac{m}{\hat{\sigma}} + \frac{\sum_i (o_i - \hat{\mu})^2}{\hat{\sigma}^3}$$

4.1.2 MLE for Gaussian distributions

$$\hat{\mu} = \frac{\sum_i o_i}{m} \quad \hat{\sigma} = \frac{\sum_i (o_i - \hat{\mu})^2}{m}$$

4.1.3 MLE for Bayesian networks

$$\hat{\theta}_{ijk} = \frac{m_{ijk}}{\sum k^j m_{ijk}}$$

MLE fail with small data

You had 2 muffin session with a man. What is the MLE for next muffin session ?

$$\hat{\theta} = \frac{2}{2} = 1$$

Help! Bayes to the rescue.

Bayes

$$\underbrace{P(\theta \mid D)}_{\text{posterior}} = \frac{\overbrace{p(D \mid \theta)}^{\text{likelihood}}\overbrace{p(\theta)}^{\text{prior}}}{\underbrace{p(D)}_{\text{marginal likelihood}}}$$

Lagrange: Rule of succession

Probability of succes after having s successes in n trials:

- Each X_i has the same chance p of being 1.
- That chance is independent.
- The prior distribution over p is uniform: $f(p) = 1$ for $0 \leq p \leq 1$

$$P(X_{n+1} = 1 \mid S_{n=k}) = \frac{k+1}{n+2}.$$

Probability for a man next muffin = 2+1/2+2 = 3/4 = 75%
https://jonathanweisberg.org/post/inductive-logic-2/

Law of unconsious statistician

LOTUS: to calculate the expected value of a function $g(X)$ of a random variable X

- when one knows the probability distribution of X
- but one does not know the distribution of $g(X)$.

If prob dist X is discrete and one knows its PMF $P(X = x)$ (but not $P(X = g(x))$), then the expected value of $g(X)$ is:

$$\mathrm{E}[g(X)] = \sum_{x} g(x)P(X = x),$$

where the sum is over all possible values x of X.

Law of unconsious statistician

If it is a continuous distribution and one knows its PDF $P(X = x)$ (but not $P(X = g(x))$), then the expected value of $g(X)$ is

$$\mathrm{E}[g(X)] = \int_{-\infty}^{\infty} g(x)P(X = x)\,\mathrm{d}x$$

- Law of the unconscious statistician
- Harvard Statistics 101
- Lecture 14: Location, Scale, and LOTUS | Statistics 110

4.2 Bayesian Parameter Learning

Instead of point estimate we obtain distribution:

$$\hat{\theta} = \mathrm{E}_{\theta \sim p(\cdot|D)}[\theta] = \int \theta p(\theta \mid D) d\theta$$

Maximum a posteriori parameter:

$$\hat{\theta} = \arg\max_{\theta} P(\theta \mid D)$$

Do not confuse with MLE: $\hat{\theta} = \arg\max_{\theta} P(D \mid \theta)$

4.2.1. Bayes learning for binary dist

$$
\begin{aligned}
p(\theta \mid o_{1:m}) &\propto p(\theta, o_{1:m}) \\
&= p(\theta) \prod_{i=1}^{m} P(o_i \mid \theta) \\
&= \prod_{i=1}^{m} P(o_i \mid \theta) \\
&= \prod_{i=1}^{m} \theta^{o_i} (1-\theta)^{1-o_i} \\
&= \theta^{n} (1-\theta)^{m-n}
\end{aligned}
$$

4.2.1 Normalization constant

$$\int_0^1 \theta^n (1-\theta)^{m-n} d\theta = \frac{\Gamma(n+1)\Gamma(m-n+1)}{\Gamma(m+2)}$$

where Γ is the gamma function. Hence MAP bin dist:

$$\begin{aligned} p(\theta \mid o_{1:m}) &= \frac{\Gamma(m+2)}{\Gamma(n+1)\Gamma(m-n+1)} \theta^n (1-\theta)^{m-n} \\ &= Beta(\theta \mid n+1, m-n+1) \end{aligned}$$

4.2.2 Bayesian learning for categorical distributions

Suppose X is a discrete random variable that can take integer values from 1 to n. We define the parameters of the distribution to be $\theta_{1:n}$ where $P(x^i) = \theta_i$. Dirichlet distribution can represent both prior and posterior distribution and is parameterised by $\alpha_{1:n}$. The densitity of the Dirichlet distribution:

$$Dir(\theta_{1:n} \mid \alpha_{1:n}) = \frac{\Gamma(\alpha_0)}{\prod_{i=1}^{n} \Gamma(\alpha_i)} \prod_{i=1}^{n} \theta_i^{\alpha_i - 1}$$

4.2.2 Dirichlet distribution

Common to use uniform prior: $\alpha_{1:n} = 1$. If the prior over $\theta_{1:n}$ is given by $Dir(\alpha_{1:n})$ and there are m_i observations of $X = 1$ then the posterior is given by:

$$p(\theta_{1:n} \mid \alpha_{1:n}, m_{1:n}) = Dir(\theta_{1:n} \mid \alpha_1 + m_1, \cdots, \alpha_n + m_n)$$

The distribution $Dir(\alpha_{1:n})$ has a mean vector whose ith component is

$$\frac{\alpha_i}{\sum_{j=1}^{n} \alpha_j}$$

4.2.3 Bayesian learning for bayesian networks

We can apply Bayesian parameter learning to discrete Bayesian networks. The prior over the Bayesian network parameters θ can be factorized as follows:

$$P(\theta \mid G) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} p(\theta_{ij})$$

where $\theta_{ij} = (\theta_{ij1}, \cdots, \theta_{ijr_i})$.

4.2.3 Bayesian learning for bayesian networks

$$P(\theta \mid G) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} p(\theta_{ij})$$

where $\theta_{ij} = (\theta_{ij1}, \cdots, \theta_{ijr_i})$.

```
function prior(vars, G)
  n = length(vars)
  r = [vars[i].r for i in 1:n]
  q = [prod([r[j] for j in inneighbors(G,i)]) for i in 1:n]
  return [ones(q[i], r[i]) for i in 1:n]
end
```


Algorithm 4.1

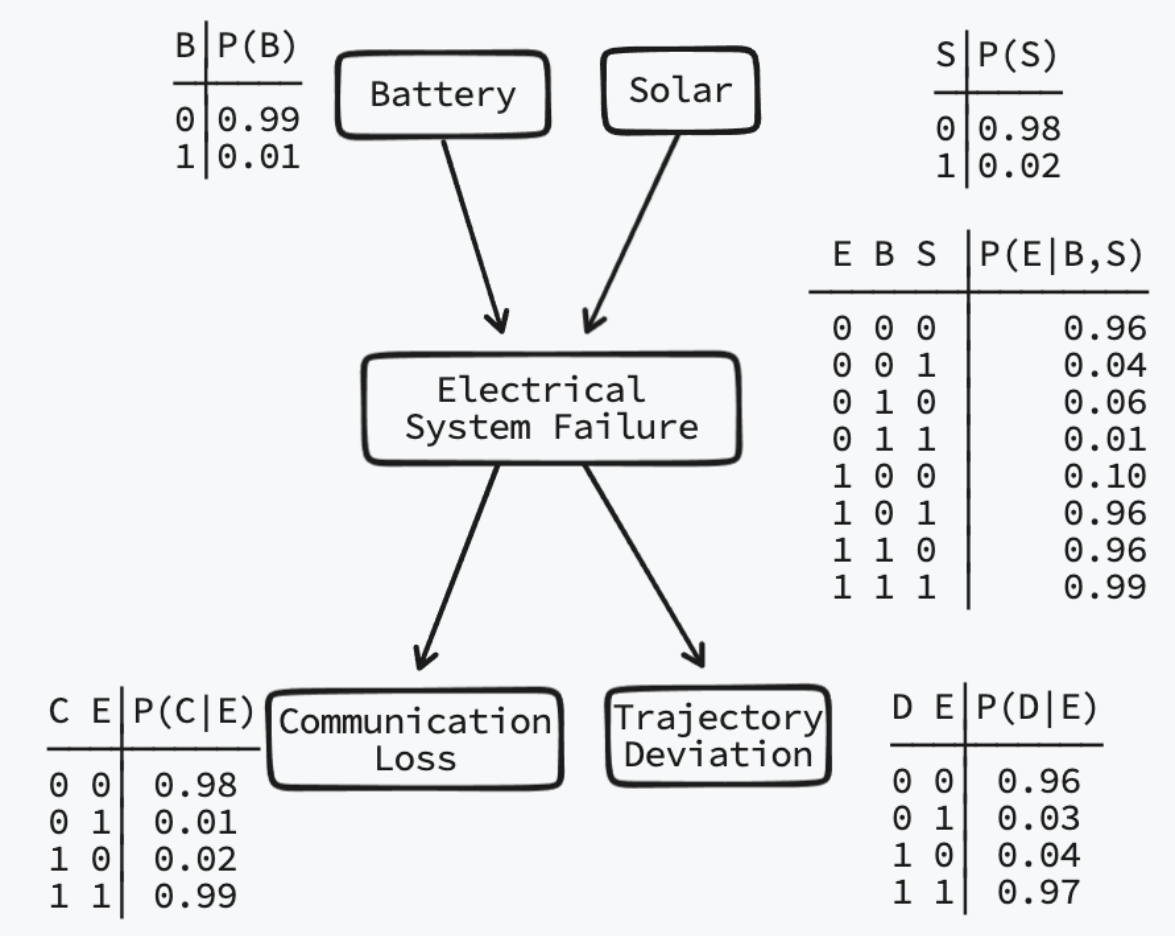
```
function sub2ind(siz, x)
  k = vcat(1, cumprod(siz[1:end-1]))
  return dot(k, x .- 1) + 1
end

function statistics(vars, G, D::Matrix{Int})
  n = size(D, 1)
  r = [vars[i].r for i in 1:n]
  q = [prod([r[j] for j in inneighbors(G,i)]) for i in 1:n]
  M = [zeros(q[i], r[i]) for i in 1:n]
  for o in eachcol(D)
    for i in 1:n
      k = o[i]
      parents = inneighbors(G,i)
      j=1
      if !isempty(parents)
        j = sub2ind(r[parents], o[parents])
      end
```


4.2.3 Bayesian learning for bayesian networks

After observing data in the form of m_{ijk} counts the posterior is then:

$$P(\theta_{ij} \mid \alpha_{ij}, m_{ij}) = Dir(\theta_{ij} \mid \alpha_{ij1} + m_{ij1}, \cdots, \alpha_{ijr_i} + m_{ijr_i})$$

4.2.3 Bayesian learning for bayesian networks



B	P(B)
0	0.99
1	0.01

S	P(S)
0	0.98
1	0.02

E	B	S	P(E|B,S)
0	0	0	0.96
0	0	1	0.04
0	1	0	0.06
0	1	1	0.01
1	0	0	0.10
1	0	1	0.96
1	1	0	0.96
1	1	1	0.99

C	E	P(C|E)
0	0	0.98
0	1	0.01
1	0	0.02
1	1	0.99

D	E	P(D|E)
0	0	0.96
0	1	0.03
1	0	0.04
1	1	0.97

4.2.3 Bayesian networks in Julia

```
# Example 2.5
B = Variable(:b, 2); S = Variable(:s, 2)
E = Variable(:e, 2)
D = Variable(:d, 2); C = Variable(:c, 2)
vars = [B, S, E, D, C]
factors = [
  Factor([B], FactorTable((b=1,) => 0.99, (b=2,) => 0.01)),
  Factor([S], FactorTable((s=1,) => 0.98, (s=2,) => 0.02)),
  Factor([E,B,S], FactorTable(
    (e=1,b=1,s=1) => 0.90, (e=1,b=1,s=2) => 0.04,
    (e=1,b=2,s=1) => 0.05, (e=1,b=2,s=2) => 0.01,
    (e=2,b=1,s=1) => 0.10, (e=2,b=1,s=2) => 0.96,
    (e=2,b=2,s=1) => 0.95, (e=2,b=2,s=2) => 0.99)),
  Factor([D, E], FactorTable(
    (d=1,e=1) => 0.96, (d=1,e=2) => 0.03,
    (d=2,e=1) => 0.04, (d=2,e=2) => 0.97)),
  Factor([C, E], FactorTable(
    (c=1,e=1) => 0.98, (c=1,e=2) => 0.01, (c=2,e=1) => 0.02, (c=2,e=2) => 0.99))
]
graph = SimpleDiGraph(5)
add_edge!(graph, 1, 3); add_edge!(graph, 2, 3)
add_edge!(graph, 3, 4); add_edge!(graph, 3, 5)
bn = BayesianNetwork(vars, factors, graph)
```


4.2.3 Bayesian learning for bayesian networks

```
# Example 2.5
B = Variable(:b, 2); S = Variable(:s, 2)
E = Variable(:e, 2)
D = Variable(:d, 2); C = Variable(:c, 2)
vars = [B, S, E, D, C]
factors = [
  Factor([B], FactorTable((b=1,) => 0.99, (b=2,) => 0.01)),
  Factor([S], FactorTable((s=1,) => 0.98, (s=2,) => 0.02)),
  Factor([E,B,S], FactorTable(
    (e=1,b=1,s=1) => 0.90, (e=1,b=1,s=2) => 0.04,
    (e=1,b=2,s=1) => 0.05, (e=1,b=2,s=2) => 0.01,
    (e=2,b=1,s=1) => 0.10, (e=2,b=1,s=2) => 0.96,
    (e=2,b=2,s=1) => 0.95, (e=2,b=2,s=2) => 0.99)),
  Factor([D, E], FactorTable(
    (d=1,e=1) => 0.96, (d=1,e=2) => 0.03,
    (d=2,e=1) => 0.04, (d=2,e=2) => 0.97)),
  Factor([C, E], FactorTable(
    (c=1,e=1) => 0.98, (c=1,e=2) => 0.01, (c=2,e=1) => 0.02, (c=2,e=2) => 0
```


4.2.3 Example 4.1

observations

$$D = \begin{matrix} 1 & 2 & 2 & 1 & A \\ 1 & 2 & 2 & 1 & B \\ 2 & 2 & 2 & 2 & C \end{matrix}$$

A=1	A=2
2	2

B=1	B=2	
0	0	A=1 & C=1
0	0	A=2 & C=1
2	0	A=1 & C=2
0	2	A=2 & C=2

C=1	C=2
0	4

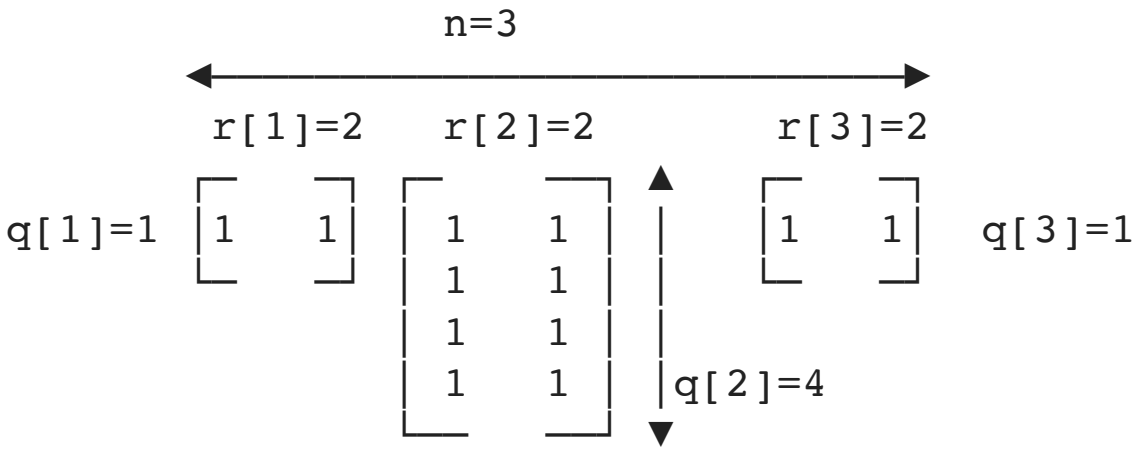
4.2.3 Bayesian networks in Julia

```
G = SimpleDiGraph(3)
add_edge!(G, 1, 2)
add_edge!(G, 3, 2)
vars = [Variable(:A,2), Variable(:B,2), Variable(:C,2)]
D = [1 2 2 1; 1 2 2 1; 2 2 2 2]
M = statistics(vars, G, D)
```

```
3-element Vector{Matrix{Float64}}:
 [2.0 2.0]
 [0.0 0.0; 0.0 0.0; 2.0 0.0; 0.0 2.0]
 [0.0 4.0]
```


Julia code to calculate prior for G

```julia
function prior(vars, G)
  n = length(vars)
  r = [vars[i].r for i in 1:n]
  q = [prod([r[j] for j in inneighbors(G,i)]) for i in 1:n]
  return [ones(q[i], r[i]) for i in 1:n]
end
α = prior(vars, G)
```



4.2.3 Calculating posterior parameters for graph

$$\text{Prior:} \quad \begin{bmatrix} 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 \end{bmatrix}$$

$$\text{Counts:} \quad \begin{bmatrix} 2 & 2 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 2 & 0 \\ 0 & 2 \end{bmatrix} \quad \begin{bmatrix} 0 & 4 \end{bmatrix}$$

$$\text{Posterior:} \quad \begin{bmatrix} 3 & 3 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 3 & 1 \\ 1 & 3 \end{bmatrix} \quad \begin{bmatrix} 1 & 5 \end{bmatrix}$$

4.2.3 Normalition of the posterior

Posterior: $\begin{bmatrix} 3 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 3 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 1 & 5 \end{bmatrix}$

```
1  θ = [mapslices(x->normalize(x,1), Mi, dims=2) for Mi in M + α]
```

$\begin{bmatrix} 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \\ 0.75 & 0.25 \\ 0.25 & 0.75 \end{bmatrix} \begin{bmatrix} 1/6 & 5/6 \end{bmatrix}$

4.3 Nonparametric learning

```
# Algorithm 4.3

gaussian_kernel(b) = x->pdf(Normal(0,b), x)

function kernel_density_estimate(φ, O)
  return x -> sum([φ(x - o) for o in O])/length(O)
end
```

Jake VanderPlas: In depth: Kernel density Estimation

4.4 Learning with missing data

Suppose that we have a binary Bayesian network with $A \rightarrow B$. We start by assuming that $\hat{\theta}$ implies

$$P(a^1) = 0.5 \qquad P(b^1 \mid a^0) = 0.2 \qquad P(b^1 \mid a^1) = 0.6$$

Using these parameters, we can expand the data set with missing values (left) to a weighted data set with all possible individual completions (right):

A	B
1	1
0	1
0	?
?	0

A	B	weight
1	1	1
0	1	1
0	0	$1 - P(b^1 \mid a^0) = 0.8$
0	1	$P(b^1 \mid a^0) = 0.2$
0	0	$\alpha P(a^0)P(b^0 \mid a^0) = \alpha 0.4 = 2/3$
1	0	$\alpha P(a^1)P(b^0 \mid a^1) = \alpha 0.2 = 1/3$

The α in the calculation here is a normalization constant, which enforces that each instance is expanded to instances whose weights sum to 1. The count matrices are then

$$\begin{bmatrix} (2+2/3) & (1+1/3) \end{bmatrix} \qquad \begin{bmatrix} (0.8+2/3) & 1.2 \\ 1/3 & 1 \end{bmatrix}$$

4.4.1 Data imputation

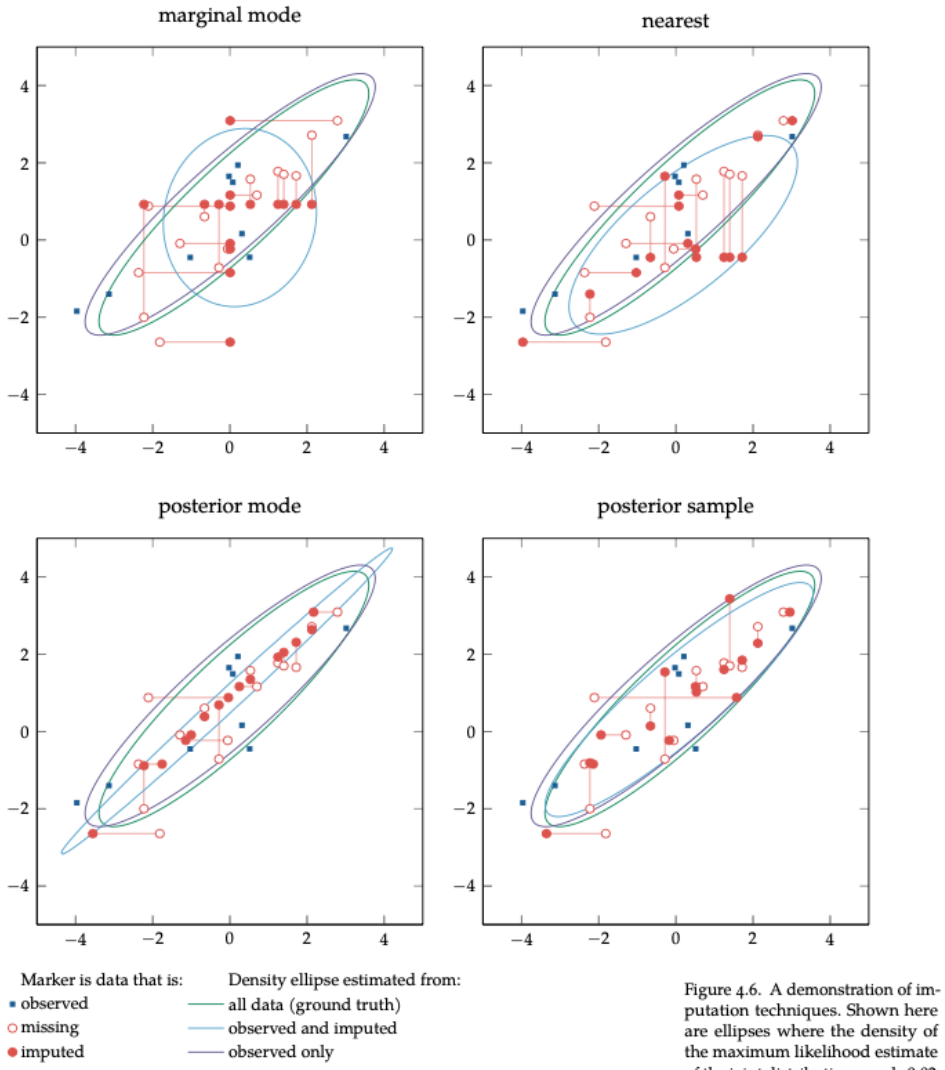


Figure 4.6. A demonstration of imputation techniques. Shown here are ellipses where the density of the maximum likelihood estimate of the joint distribution equals 0.02.

- `df.dropna()`
- `df.fillna(df.mean())` if number
- `df.fillna(df.mode())` if categorical
- Partial Missing Multivariate Observation and What to Do With Them by Junpeng Lao

4.4.2 Expectation-Maximization

impyute

```
nan_xy = matrix.nan_indices(data)
for x_i, y_i in nan_xy:
    previous = 1
    for i in range(5):
        col = data[:, y_i]
        # Expectation
        mu =  np.nanmean(col)
        std = np.nanstd(col)
        # Maximization
        value_to_impute_with = np.random.normal(loc=mu, scale=std)
        data[x_i][y_i] = value_to_impute_with
        # Break out of loop if likelihood doesn't change at least 10%
        delta = np.abs(value_to_impute-previous)/previous
        if i and delta < eps:
            break
        previous = value_to_impute_with
return data
```


EM results for Age in Titanic

EM applied on Titanic

AUC with HistGradientBoosting

Method	AUC	Std Dev
Leave Nan	0.8324065257411288	+/- 0.03
Impute Median	0.8357773122579826	+/- 0.02
E.M. Norm	0.8200723671681585	+/- 0.04
E.M. Fisk	0.8211705706849489	+/- 0.01

Conclusions

- Parameter learning via MLE vs MAP
- Binomial and Dirichlet distribution
- Kernel Density
- Expectation Maximization