

Bayesian Methods

Peter Komar

December 25, 2018

Contents

1	Foundations	2
1.1	Definitions, identities	2
1.2	Bayesian inference	3
1.3	Model comparison	4
1.4	Prediction	5
2	Exact inference and Maximum Likelihood Estimate	6
2.1	Maximum likelihood estimate	6
2.2	Exact inference examples	8
3	Priors, Regularization, AIC, BIC, LRT	12
3.1	Improper and proper priors	12
3.2	Regularization	12
3.3	Linear regression	12
3.4	Model comparison with asymptotic metrics	13
3.5	Example: Linear regression	14
4	Graphical models	18
4.1	Elements	18
4.2	Real-life examples	19
4.3	Plate notation	19
4.4	Hierarchical models	20
4.5	Example: Beta-Binomial	23
5	Hidden variables, EM, Mixture models	25
5.1	Definitions	25
5.2	Expectation Maximization	25
5.3	Mixture models	26
5.4	Gaussian Mixture Model	27

1 Foundations

1.1 Definitions, identities

Notation

- Upper-case letters (A, B, C, X, Y): random variables
- Lower-case letters (a, b, c, x, y): real numbers
- $P(X = x) =: P(x)$
- $P(X = x \text{ and } Y = y) =: P(x, y)$
- $P(A = a, \text{ given } B = b) =: P(a | b)$
- $\int_{-\infty}^{+\infty} [\dots] da =: \sum_{a \in \mathbb{R}} [\dots] =: \sum_a [\dots]$

Conditional probability

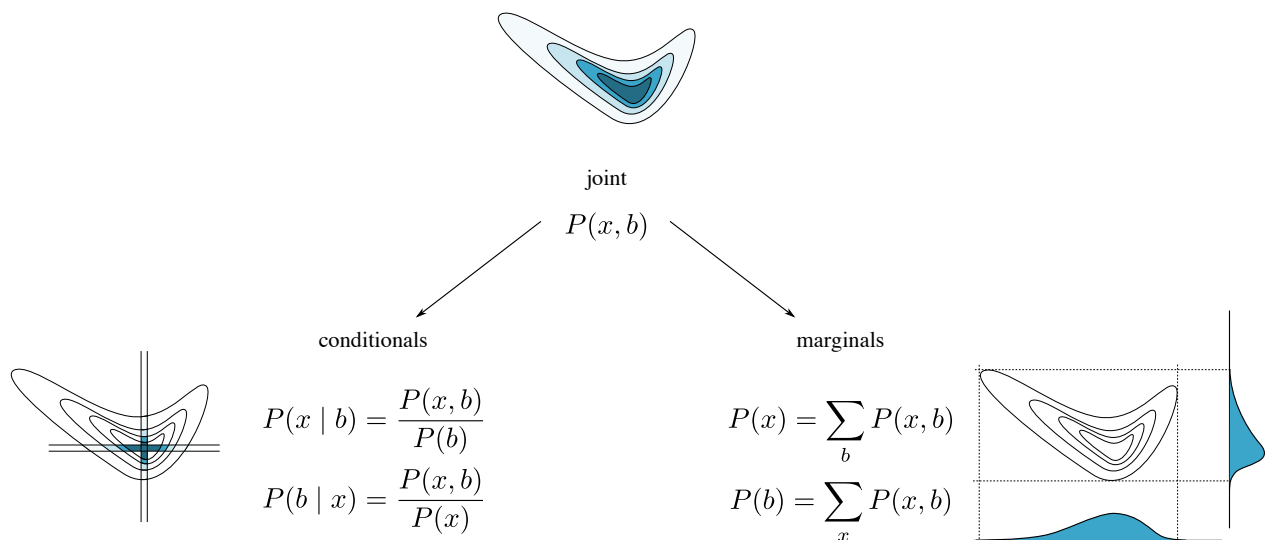
- $P(a | b) = \frac{P(a, b)}{P(b)}$
- $P(a, b) = P(a | b) P(b)$
- $P(a, b | c) = P(a | b, c) P(b | c)$
- $\sum_a P(a | b) = 1$, but $\sum_b P(a | b) \neq 1$, in general
- $\sum_b P(a | b) P(b) = \sum_b P(a, b) = P(a)$

Marginal

- $P(a) = \sum_b P(a | b) P(b)$

Bayes theorem

- $P(b | x) = \frac{1}{P(x)} P(x | b) P(b)$



1.2 Bayesian inference

Prior, likelihood, posterior

- Data: $D = \{x_1, x_2, \dots, x_n\}$, independent measurements.
- Model: M with θ : parameter(s) to estimate
- Prior: $P(\theta)$
- Likelihood: $P(D | \theta) = P(x_1 | \theta) P(x_2 | \theta) \dots P(x_n | \theta) = \prod_{i=1}^n P(x_i | \theta)$
- Unnormalized posterior: $P^*(\theta | D) = P(D | \theta) P(\theta)$
- Normalization: $Z = \sum_{\theta} P^*(\theta | D)$
- Posterior: $P(\theta | D) = \frac{1}{Z} P^*(\theta | D)$

Example

“Three light bulbs of the same make lasted 1, 2 and 5 months of continuous use. Let us estimate the lifetime of this kind of light bulb.”

- $D = \{t_1, t_2, t_3\} = \{1, 2, 5\}$
- M : Light bulbs have average lifetime of T months.
- $P(T) = \frac{1}{1000}$, uniform on $[0, 1000]$.
- $P(t | T) = \frac{1}{T} \exp\left(-\frac{t}{T}\right)$
- $P(D | T) = \prod_i P(t_i | T) = \prod_{i=1}^3 \frac{1}{T} \exp\left(-\frac{t_i}{T}\right) = \frac{1}{T^3} \exp\left(-\frac{1+2+5}{T}\right)$
- $P^*(T | D) = \frac{1}{T^3} \exp\left(-\frac{8}{T}\right)$
- Z and $P(T | D)$ can be determined numerically:

```
1 import numpy as np
2
3 T_arr = np.linspace(0.1, 1000, 10_000)
4 Pstar_arr = 1.0/T_arr**3 * np.exp(-8/T_arr)
5 Z = np.sum(Pstar_arr)
6 P_arr = Pstar_arr / Z
```

yielding $Z = 0.1562$

- $\mathbb{E}(T | D) = \sum_T T P(T | D)$
- $\text{std}(T | D) = \sqrt{\sum_T (T - \mathbb{E}(T))^2 P(T | D)}$

```
1 T_ev = np.sum(T_arr * P_arr)
2 T_std = np.sqrt(np.sum((T_arr - T_ev)**2 * P_arr))
```

yielding $\mathbb{E}(T | D) = 7.937$, $\text{std}(T | D) = 14.48$.

1.3 Model comparison

New definition: Evidence

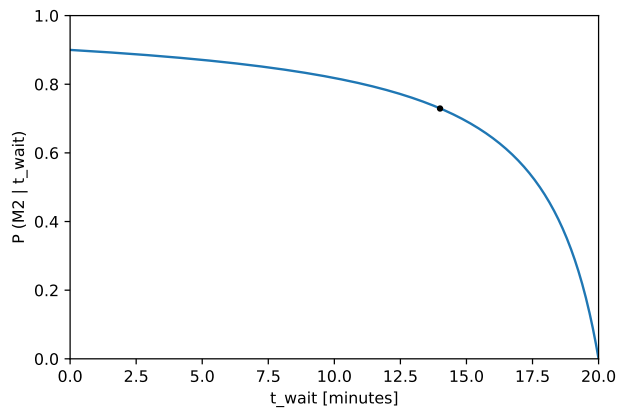
- Data: D
- Model 1: M_1 with parameter θ_1 and prior $P(\theta_1 | M_1)$, and likelihood $P(D | \theta_1, M_1)$
- Model 2: M_2 with parameter θ_2 and prior $P(\theta_2 | M_2)$, and likelihood $P(D | \theta_2, M_2)$
- Prior on models: $P(M_1) = 0.5$, $P(M_2) = 0.5$.
- Evidence for each model: $P(D | M_i) = \sum_{\theta} P(D | \theta_i, M_i) P(\theta_i | M_i)$
- Unnormalized posterior: $P^*(M_1 | D) = P(D | M_1) P(M_1)$, and $P^*(M_2 | D) = P(D | M_2) P(M_2)$
- Normalization: $Z = P^*(M_1 | D) + P^*(M_2 | D)$.

Example

“Waiting for my baggage at the airport carousel, there are two possibilities: 1) It could miss the plane, and will never come, or 2) It was on the plane and it had a $1/20$ chance of arriving within any of the 1-minute intervals between 0 and 20 minutes. Now, given what is the posterior probability of model 2 given that 14 minutes has passed and the bag has not arrived?”

- $D = \{\text{Bag has not arrived after } t_{\text{wait}} = 14 \text{ minutes}\}$
- M_1 : It never arrives, $P(D | M_1) = 1$
- M_2 : It shows up some time between 0 and 20 minutes, $P(t_{\text{bag}} | M_2) = 1/20$ for $t_{\text{bag}} \in [0, 20]$, and the likelihood is $P(D | t_{\text{bag}}, M_2) = 1$, if $t_{\text{bag}} > t_{\text{wait}}$, and 0 otherwise.
- $P(M_1) = 0.1$, $P(M_2) = 0.9$
- $P(D | M_1) = 1$
- $P(D | M_2) = \sum_{t_{\text{bag}}} P(D | t_{\text{bag}}, M_2) P(t_{\text{bag}} | M_2) = \sum_{t_{\text{bag}}} [t_{\text{bag}} > 14] \times \frac{1}{20} = \frac{20-14}{20}$
- $P^*(M_1 | D) = 1 \times 0.1$, $P^*(M_2 | D) = \frac{20-14}{20} \times 0.9$
- $Z = 0.1 + \frac{3}{10} \times 0.9 = 0.37$
- $P(M_2 | D) = P^*(M_2 | D)/Z = 0.7297$.

We can also plot $P(M_2 | t_{\text{wait}})$ for all waiting times between 0 and 20 minutes.



1.4 Prediction

New definition: Predictive distribution

- Data: $D = \{x_1, x_2, \dots, x_n\}$
- Model: M with parameter θ , prior $P(\theta)$ and likelihood $P(x | \theta)$
- Posterior: $P(\theta | D) = P^*(\theta | D)/Z = \dots$ (see previous sections)
- Predictive distribution: $P(X_{n+1} = x | D) = \sum_{\theta} P(x | \theta) P(\theta | D)$
- Customized prediction: $P(f(\theta) | D) = \sum_{\theta} f(\theta) P(\theta | D)$

Example

“Two player, A and B are playing a game of luck, where at the beginning of the game a ball is rolled on a pool table to divide the table in two un-equal halves: A’s side and B’s side. In each subsequent round, a ball is rolled. A point is given to the player on whose side the ball stops. A and B are playing this game until one of them reaches 6 points. The current score is 5 to 3 in favor of A. What is the chance that A will win this game?”

- $D = \{n_A = 5, n_B = 3\}$
- M , first ball: $P(b) = 1$ in $[0, 1]$
- $P(\text{A scores} | b) = b$
- $P(D | b) = \text{Binomial}(5 | 5 + 3, b)$
- $P^*(b_0 | D) = \text{Binomial}(5 | 8, b) \times 1$
- $Z = \sum_b \text{Binomial}(5 | 8, b)$ can be calculated numerically

```
1 import numpy as np
2 from scipy.stats import binom
3
4 b_arr = np.linspace(0, 1, 1000)
5 Pstar_arr = binom.pmf(5, 8, b_arr)
6 Z = np.sum(Pstar_arr)
```

- $P(\text{A wins} | b, D) = 1 - P(\text{B wins} | b, D) = 1 - (1 - b)^3 = f(b)$
- $P(\text{A wins} | D) = \sum_b f(b) P^*(b | D) / Z$

```
1 P_arr = Pstar_arr / Z
2 P_Awins = np.sum((1 - (1 - b_arr)**3) * P_arr)
```

yielding $P(\text{A wins} | D) = 0.909$

2 Exact inference and Maximum Likelihood Estimate

2.1 Maximum likelihood estimate

MLE-method

- Data: $D = \{x_1, x_2, \dots, x_N\}$
- Parameter: θ
- Likelihood: $P(x_i | \theta)$
- Total log likelihood: $L(\theta) = \log P(D | \theta) = \sum_{i=1}^N \log P(x_i | \theta)$
- Maximum likelihood estimate $\theta_{\text{MLE}} = \operatorname{argmax}_{\theta} \log P(D | \theta)$,
Numerically: with gradient descent or EM methods,
Analytically: equating first derivatives to 0, and solving the system of equations.

Example 1: Normal model

- Data: $D = \{x_i\}_{i=1}^N$
- Parameters: $\mu \in \mathbb{R}, \sigma^2 > 0$
- Likelihood: $P(x_i | \mu, \sigma^2) = \text{Normal}(x_i | \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x_i - \mu)^2}{2\sigma^2}\right]$
- Total log likelihood:

$$L(\mu, \sigma^2) = \sum_{i=1}^N \log \text{Normal}(x_i | \mu, \sigma^2) = -\frac{N}{2} \log(\sigma^2) - \sum_{i=1}^N \frac{(x_i - \mu)^2}{2\sigma^2} + \text{const.}$$

- Analytical solution:

$$\begin{aligned} 0 &= \left[\frac{\partial L}{\partial \mu} \right]_{\text{MLE}} = \left[\sum_{i=1}^N \frac{\mu - x_i}{\sigma^2} \right]_{\text{MLE}} \Rightarrow \mu_{\text{MLE}} = \frac{1}{N} \sum_{i=1}^N x_i. \\ 0 &= \left[\frac{\partial L}{\partial (\sigma^2)} \right]_{\text{MLE}} = \left[-\frac{N}{2\sigma^2} + \sum_{i=1}^N \frac{(x_i - \mu)^2}{2(\sigma^2)^2} \right]_{\text{MLE}} \Rightarrow (\sigma^2)_{\text{MLE}} = \frac{1}{N} \sum_{i=1}^N (x_i - \mu_{\text{MLE}})^2 \end{aligned}$$

Example 2: Cauchy distribution

- Data: $D = \{-10, 1, 2, 5, 20\}$
- Parameters: $m \in \mathbb{R}, \quad s > 0.$
- Likelihood: $P(x_i | m, s) = \text{Cauchy}(x_i | m, s) = \frac{1}{s\pi} \frac{1}{1 + [(x_i - m)/s]^2}$
- Total log likelihood:

$$L(m, s) = \sum_{i=1}^N \log \text{Cauchy}(x_i | m, s) = -N \log(s) - \sum_{i=1}^N \log \left(1 + \left[\frac{x_i - m}{s} \right]^2 \right)$$

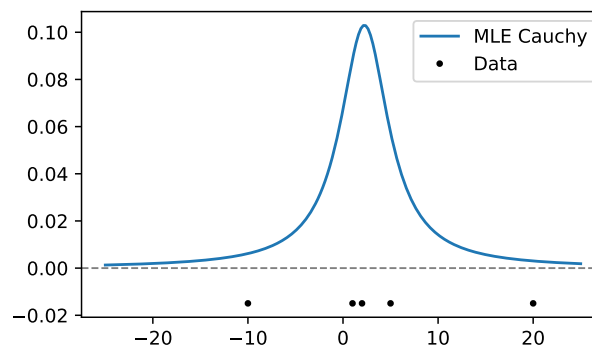
- Numerical maximization (starting from $m_0 = 0, s_0 = 10$):

```

1 import numpy as np
2 from scipy.optimize import minimize
3
4 def cauchy_total_log_likelihood(X, m, s):
5     X = np.array(X)
6
7     L = 0
8     L += -len(X)/2 * np.log(s**2)
9     L += -np.sum( np.log(1 + (X - m)**2 / s**2 ) )
10
11     return L
12
13 X = [-10, 1, 2, 5, 20]
14 def func_to_minimize(theta):
15     m = theta[0]
16     s = theta[1]
17     return - cauchy_total_log_likelihood(X, m, s)
18
19 m0 = 0
20 s0 = 10
21 result = minimize(func_to_minimize, [m0, s0])
22 m_MLE, s_MLE = result.x

```

yielding $m_{\text{MLE}} = 2.251$, $s_{\text{MLE}} = 3.090$, the resulting MLE fit is shown below.



2.2 Exact inference examples

Binomial model

- Data: $D = \{(k_1, n_1), (k_2, n_2), \dots, (k_N, n_N)\}$, where k_i (successes), n_i (attempts) $\in \mathbb{N}$ and $k_i \leq n_i$
- Parameter: p (probability of success) $\in [0, 1]$, flat prior: $P(p) = 1$, on $[0, 1]$
- Likelihood: $P(k_i | n_i, p) = \text{Binomial}(k_i | n_i, p) = \binom{n_i}{k_i} p^{k_i} (1-p)^{n_i-k_i}$
- Posterior:

$$\begin{aligned} P(p | D) &= \frac{1}{Z} \prod_{i=1}^N [p^{k_i} (1-p)^{n_i-k_i}] = \frac{1}{Z} p^{k_{\text{tot}}} (1-p)^{n_{\text{tot}}-k_{\text{tot}}} \\ &= \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1} (1-p)^{\beta-1} = \text{Beta}(p | \alpha = k_{\text{tot}} + 1, \beta = n_{\text{tot}} - k_{\text{tot}} + 1), \end{aligned}$$

where $k_{\text{tot}} = \sum_i k_i$ and $n_{\text{tot}} = \sum_i n_i$. Mean, mode and standard deviation are

$$\begin{aligned} \mathbb{E}(p) &= \frac{\alpha}{\alpha + \beta} = \frac{k_{\text{tot}} + 1}{n_{\text{tot}} + 2}, \quad \text{mode}(p) = \frac{\alpha - 1}{\alpha + \beta - 2} = \frac{k_{\text{tot}}}{n_{\text{tot}}}, \\ \text{std}(p) &= \frac{\sqrt{\alpha\beta}}{(\alpha + \beta)\sqrt{\alpha + \beta + 1}} = \frac{\sqrt{(k_{\text{tot}} + 1)(n_{\text{tot}} - k_{\text{tot}} + 1)}}{(n_{\text{tot}} + 2)\sqrt{n_{\text{tot}} + 3}} \end{aligned}$$

Poisson model

- Data: $D = \{k_1, k_2, \dots, k_N\}$, where k_i (number of events) $\in \mathbb{N}$
- Parameters: λ (expected number of events) > 0 , flat prior: $P(\lambda) = \text{const.}$
- Likelihood: $P(k_i | \lambda) = \text{Poisson}(k_i | \lambda) = e^{-\lambda} \frac{\lambda^{k_i}}{k_i!}$
- Posterior:

$$P(\lambda | D) = \frac{1}{Z} \prod_{i=1}^N [e^{-\lambda} \lambda^{k_i}] = \frac{1}{Z} e^{-N\lambda} \lambda^{k_{\text{tot}}} = \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\beta\lambda} = \text{Gamma}(\lambda | \alpha = k_{\text{tot}} + 1, \beta = N),$$

where $k_{\text{tot}} = \sum_i k_i$. Mean, mode and standard deviation are

$$\mathbb{E}(\lambda) = \frac{\alpha}{\beta} = \frac{k_{\text{tot}} + 1}{N}, \quad \text{mode}(\lambda) = \frac{\alpha - 1}{\beta} = \frac{k_{\text{tot}}}{N}, \quad \text{std}(\lambda) = \frac{\sqrt{\alpha}}{\beta} = \frac{\sqrt{k_{\text{tot}} + 1}}{N}$$

Multinomial model

- Data: $D = \{(k_{1,1}, k_{1,2}, \dots, k_{1,M}), (k_{2,1}, k_{2,2}, \dots, k_{2,M}), \dots, (k_{N,1}, k_{N,2}, \dots, k_{N,M})\}$, where $k_{i,j}$ (counts of outcome $j \in \mathbb{N}$, and $\sum_j k_{i,j} = 1, \forall i$).
- Parameters: $p = (p_1, p_2, \dots, p_M)$, where p_j (probability of outcome j) > 0 and $\sum_j p_j = 1$; flat prior: $P(p) = \text{const.}$
- Likelihood: $P(\{k_{i,j}\}_{j=1}^M | p) = \text{Multinomial}(\{k_{i,j}\}_{j=1}^M | p) = k_{i,\text{tot}}! \prod_j \frac{p_j^{k_{i,j}}}{k_{i,j}!}$
- Posterior:

$$P(p | D) = \frac{1}{Z} \prod_{i=1}^N \prod_{j=1}^M (p_j)^{k_{i,j}} = \frac{1}{Z} \prod_{j=1}^M (p_j)^{k_{\text{tot},j}} = \Gamma(\alpha_{\text{tot}}) \prod_{j=1}^M \frac{(p_j)^{\alpha_j - 1}}{\Gamma(\alpha_j)} = \text{Dirichlet}(p | \alpha_j = k_{\text{tot},j} + 1),$$

where $k_{i,\text{tot}} = \sum_j k_{i,j}$, $k_{\text{tot},j} = \sum_i k_{i,j}$, and $\alpha_{\text{tot}} = \sum_j \alpha_j = k_{\text{tot},\text{tot}} + M$. Mean, mode and marginal standard deviation are

$$\begin{aligned} \mathbb{E}(p_j) &= \frac{\alpha_j}{\alpha_{\text{tot}}} = \frac{k_{\text{tot},j} + 1}{k_{\text{tot},\text{tot}} + M}, \quad \text{mode}(p) : p_j = \frac{\alpha_j - 1}{\alpha_{\text{tot}} - M} = \frac{k_{\text{tot},j}}{k_{\text{tot},\text{tot}}} \\ \text{std}(p_j) &= \frac{\sqrt{\alpha_j(\alpha_{\text{tot}} - \alpha_j)}}{\alpha_{\text{tot}} \sqrt{\alpha_{\text{tot}} + 1}} = \frac{\sqrt{(k_{\text{tot},j} + 1)(k_{\text{tot},\text{tot}} - k_{\text{tot},j} + M - 1)}}{(k_{\text{tot},\text{tot}} + M) \sqrt{k_{\text{tot},\text{tot}} + M + 1}} \end{aligned}$$

Exponential model

- Data: $D = \{t_1, t_2, \dots, t_N\}$, where t_i (waiting times) > 0
- Parameter: γ (rate) > 0 , flat prior: $P(\gamma) = \text{const.}$
- Likelihood: $P(t_i | \gamma) = \text{Exponential}(t_i | \gamma) = \gamma e^{-\gamma t_i}$
- Posterior:

$$P(\gamma | D) = \frac{1}{Z} \prod_{i=1}^N [\gamma e^{-\gamma t_i}] = \frac{1}{Z} \gamma^N e^{-\gamma t_{\text{tot}}} = \frac{\beta^\alpha}{\Gamma(\alpha)} \gamma^{\alpha-1} e^{-\beta \gamma} = \text{Gamma}(\gamma | \alpha = N + 1, \beta = t_{\text{tot}}),$$

where $t_{\text{tot}} = \sum_i t_i$. Mean, mode, standard deviation are

$$\mathbb{E}(\gamma) = \frac{\alpha}{\beta} = \frac{N + 1}{t_{\text{tot}}}, \quad \text{mode}(\gamma) = \frac{\alpha - 1}{\beta} = \frac{N}{t_{\text{tot}}}, \quad \text{std}(\gamma) = \frac{\sqrt{\alpha}}{\beta} = \frac{\sqrt{N + 1}}{t_{\text{tot}}}$$

Normal

- Data: $D = \{x_1, x_2, \dots, x_N\}$, where x (value) $\in \mathbb{R}$
- Parameters: μ (expected value) $\in \mathbb{R}$, σ^2 (variance) > 0 , uninformative prior: $P(\mu, \sigma^2) = \text{const.}$
- Likelihood: $P(x_i | \mu, \sigma^2) = \text{Normal}(x_i | \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x_i - \mu)^2}{2\sigma^2}\right]$
- Posterior:

$$\begin{aligned}
P(\mu, \sigma^2 | D) &= \frac{1}{Z} \prod_{i=1}^N \left\{ \frac{1}{\sqrt{\sigma^2}} \exp\left[-\frac{(x_i - \mu)^2}{2\sigma^2}\right] \right\} = \frac{1}{Z} \left(\frac{1}{\sigma^2}\right)^{N/2} \exp\left[-\frac{Ns^2 + N(\mu - m)^2}{2\sigma^2}\right] \\
&= \frac{\sqrt{\lambda}}{\sqrt{2\pi\sigma^2}} \frac{\beta^\alpha}{\Gamma(\alpha)} \left(\frac{1}{\sigma^2}\right)^{\alpha+1} \exp\left[-\frac{2\beta + \lambda(\mu - \mu_c)^2}{2\sigma^2}\right] \\
&= \text{Normal-Inverse-Gamma}\left(\mu, \sigma^2 \mid \alpha = \frac{N-3}{2}, \beta = \frac{Ns^2}{2}, \mu_c = m, \lambda = N\right),
\end{aligned}$$

where $m = \frac{1}{N} \sum_i x_i$ is the empirical mean, $s^2 = \frac{1}{N} \sum_i (x_i - m)^2$ is the empirical variance. The mode is identical to the MLE result

$$\text{mode}(\mu, \sigma^2) = (m, s^2).$$

The marginal, and mean, mode and standard deviation of μ is

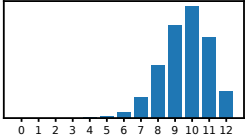
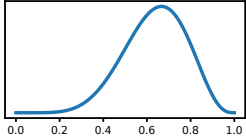
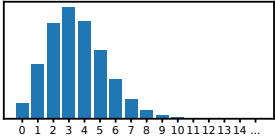
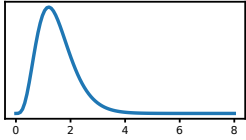
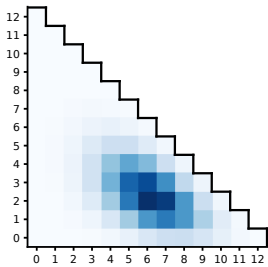
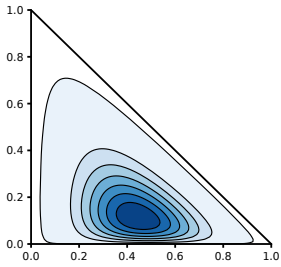
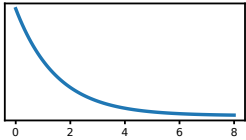
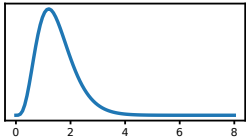
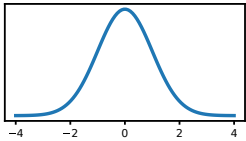
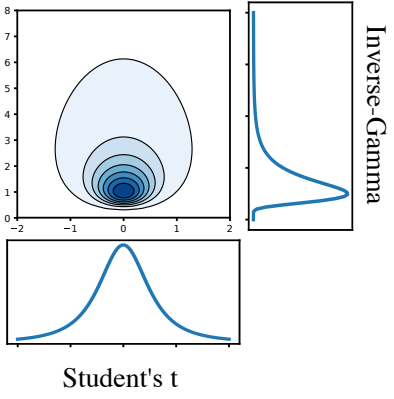
$$\begin{aligned}
P(\mu | D) &= \sum_{\sigma^2} P(\mu, \sigma^2 | D) = \frac{\Gamma\left(\frac{N-2}{2}\right)}{\Gamma\left(\frac{N-3}{2}\right)} \frac{1}{\sqrt{\pi s^2}} \left[1 + \frac{(\mu - m)^2}{s^2}\right]^{-(N-2)/2} \\
&= \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} \frac{1}{\sqrt{\pi\nu}} \left[1 + \frac{1}{\nu} \left(\frac{\mu - \text{loc}}{\text{scale}}\right)^2\right]^{-(\nu+1)/2} \frac{1}{\text{scale}} \\
&= \text{t-distr}\left(\mu \mid \text{loc} = m, \text{scale} = \frac{s}{\sqrt{N-3}}, \nu = N-3\right),
\end{aligned}$$

$$\mathbb{E}(\mu) = m, \quad \text{mode}(\mu) = m, \quad \text{std}(\mu) = \frac{s}{\sqrt{N-3}} \sqrt{\frac{\nu}{\nu-2}} = \frac{s}{\sqrt{N-5}},$$

where ν is the “degrees of freedom” of the Student’s t-distribution. The marginal, and mean, mode and standard deviation of σ^2 is

$$\begin{aligned}
P(\sigma^2 | D) &= \sum_{\mu} P(\mu, \sigma^2 | D) = \frac{\beta^\alpha}{\Gamma(\alpha)} \left(\frac{1}{\sigma^2}\right)^{\alpha+1} \exp\left(-\frac{\beta}{\sigma^2}\right) \\
&= \text{Inverse-Gamma}\left(\sigma^2 \mid \alpha = \frac{N-3}{2}, \beta = \frac{Ns^2}{2}\right)
\end{aligned}$$

$$\mathbb{E}(\sigma^2) = \frac{\beta}{\alpha-1} = s^2 \frac{N}{N-5}, \quad \text{mode}(\sigma^2) = \frac{\beta}{\alpha+1} = s^2 \frac{N}{N-1}, \quad \text{std}(\sigma^2) = \frac{\beta}{(\alpha-1)\sqrt{\alpha-2}} = s^2 \frac{\sqrt{2}N}{(N-5)\sqrt{N-7}}$$

Model	Posterior
Binomial 	Beta 
Poisson 	Gamma 
Multinomial 	Dirichlet 
Exponential 	Gamma 
Normal 	Normal-Inverse-Gamma  Student's t

3 Priors, Regularization, AIC, BIC, LRT

3.1 Improper and proper priors

Improper priors are not normalizable, i.e. $\sum_{\theta} P(\theta) = \infty$

- Flat prior: $P(\theta) = \text{const.}$ over any infinite domain.
- Uninformative priors (from transformation invariance or max-entropy principles)
 - Location parameter: $P(m) = \text{const.}$, on $m \in (-\infty, +\infty)$,
 - Scale parameter: $P(s) = \frac{\text{const.}}{s}$, on $s \in (0, \infty)$,
 - Probability parameter: $P(p) = \frac{\text{const.}}{p(1-p)}$, on $p \in (0, 1)$.

Proper priors are normalized, i.e. $\sum_{\theta} P(\theta) = 1$.

3.2 Regularization

Regularized model training

- Data: D ,
- Model: M with parameters θ , likelihood $P(D | \theta)$
- Cost function $= -\log P(D | \theta) = -L(\theta)$, the log likelihood
- Penalty: $\text{penalty}(\theta)$, which is high for implausible θ values.
- Regularized optimum: $\theta_{\text{reg.opt.}} = \arg \min_{\theta} (-L(\theta) + \text{penalty}(\theta))$

3.3 Linear regression

- Data: $D = \{ (\{x_{i,k}\}_{k=1}^K, y_i) \}_{i=1}^N$, where x_i (feature vector) $\in \mathbb{R}^K$, y_i (predicted variable) $\in \mathbb{R}$.
- Parameters: $\{b_k\}_{k=1}^K$, where b_k (coefficient or weight) $\in \mathbb{R}$.
- Model:

$$\begin{aligned}
 y_i &= \sum_{k=1}^K x_{i,k} b_k + \varepsilon_i, \quad \text{with } P(\varepsilon_i) = \text{Normal}(\varepsilon_i | \mu = 0, \sigma^2 = \sigma^2) \\
 y &= Xb + \varepsilon \\
 &\text{or equivalently} \\
 P(y | X, b) &= \prod_{i=1}^N \text{Normal}(y_i | \mu = (Xb)_i, \sigma^2 = \sigma^2)
 \end{aligned}$$

- Log likelihood: $L(b) = \log P(y | X, b) = -\frac{N}{2} \log(\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^N \left[y_i - \sum_k x_{i,k} b_k \right]^2$
- Maximum likelihood estimate: $b_{\text{MLE}} = \arg \max_b L(b) = (X^T X)^{-1} X^T y$, $(\sigma^2)_{\text{MLE}} = \frac{1}{N} \|y - Xb\|^2$
- Regularization:
 - “L1 regularization”: $\text{penalty}(b) = \alpha_1 \sum_k |b_k|$
 \Leftrightarrow Laplace prior: $P(b_k) = \text{const.} \times e^{-\alpha_1 |b_k|} = \text{Laplace}(b_k | \text{loc} = 0, \text{scale} = 1/\alpha_1)$
 - “L2 regularization”: $\text{penalty}(b) = \frac{\alpha_2}{2} \sum_k (b_k)^2$
 \Leftrightarrow Normal prior: $P(b_k) = \text{const.} \times e^{-\alpha_2 (b_k)^2/2} = \text{Normal}(b_k | \mu = 0, \sigma^2 = 1/\alpha_2)$
 - “Elastic net regularization”: $\text{penalty}(b) = \alpha_1 \sum_k |b_k| + \frac{\alpha_2}{2} \sum_k (b_k)^2$

The “hyperparameters” α_1 and α_2 can be optimized using “Leave-one-out” or “M-fold” cross-validation.

3.4 Model comparison with asymptotic metrics

Maximum likelihood results from two models

- Data $D = \{x_i\}_{i=1}^N$
- Null model: M_0 with parameters θ_0 , and $L_0(\theta_0) = P(D | \theta_0, M_0)$, $\theta_{0,\text{MLE}} = \text{argmax}_{\theta_0} L_0(\theta_0)$
- Alternate model: M_1 with parameters θ_1 , and $L_1(\theta_1) = P(D | \theta_1, M_1)$, $\theta_{1,\text{MLE}} = \text{argmax}_{\theta_1} L_1(\theta_1)$

Akaike Information Criterion (AIC)

- $\text{AIC}(M_i) = -2 \left[L_i(\theta_{i,\text{MLE}}) - \dim(\theta_i) \right]$ for both $i = 0, 1$ models.
- If $\text{AIC}(M_1) < \text{AIC}(M_0)$, then M_1 is more plausible.

Bayesian Information Criterion (BIC)

- $\text{BIC}(M_i) = -2 \left[L_i(\theta_{i,\text{MLE}}) - \frac{\ln(N)}{2} \dim(\theta_i) \right]$ for both $i = 0, 1$ models.
- If $\text{BIC}(M_1) < \text{BIC}(M_0)$, then M_1 is more plausible.

Likelihood Ratio Test (LRT)

- $\log \text{LR} = \log \frac{P(D | M_1, \theta_{1,\text{MLE}})}{P(D | M_0, \theta_{0,\text{MLE}})} = L_1(\theta_{1,\text{MLE}}) - L_0(\theta_{0,\text{MLE}})$
- $\text{LRT pvalue} = 1 - \text{cdf } \chi^2 \left(2 \log \text{LR} \mid \text{dof} = \dim(\theta_1) - \dim(\theta_0) \right)$, where $\text{cdf } \chi^2(\dots \mid \text{dof} = d)$ is the cumulative distribution function of the χ^2 distribution with degrees of freedom d .

Model evidence

- $P(D | M_i) = \sum_{\theta_i} P(D | \theta_i, M_i) \approx \exp \left(-\frac{1}{2} \text{IC} \right)$,
- where IC can be either AIC or BIC (or WAIC, WBIC)
- Under uniform prior (i.e. $P(M_0) = P(M_1)$), the posterior probability of the alternate model being correct is

$$P(M_1 | D) \approx \frac{\exp \left(-\frac{1}{2} \text{IC}_1 \right)}{\exp \left(-\frac{1}{2} \text{IC}_0 \right) + \exp \left(-\frac{1}{2} \text{IC}_1 \right)}$$

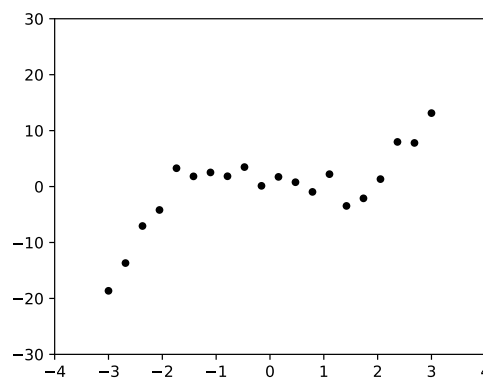
3.5 Example: Linear regression

- Data: $D = \{(x_i, y_i)\}_{i=1}^N$ (generated from $y = 1 - 3x - x^2/2 + x^3 + \varepsilon$ with $\text{std}(\varepsilon) = 2$).

```

1 import numpy as np
2 from numpy.polynomial.polynomial import polyval
3 from scipy.stats import norm
4
5 c_true = [1, -3, -0.5, 1]
6 sigma_true = 2
7 x_data = np.linspace(-3, 3, 20)
8 y_data = [polyval(x, c_true) + norm.rvs(loc=0, scale=sigma_true)
9           for x in x_data]

```



- Models: M_K : $K = 0, 1, 2, \dots$ -degree polynomial, parameters: $c = (c_0, c_1, \dots, c_K)$.
- “Linear features”: $X_i := (1, x_i, (x_i)^2, (x_i)^3, \dots, (x_i)^K)$.

```

1 def generate_polynomial_features(x_data, degree):
2     K = degree
3     N = len(x_data)
4     X = np.zeros([N, K+1])
5     for i, x in enumerate(x_data):
6         for k in range(0, K+1, 1):
7             X[i, k] = x**k
8     return X

```

- Likelihood: $y = \sum_{k=0}^K X_{i,k} c_k + \varepsilon$, with $P(\varepsilon) = \text{Normal}(\varepsilon \mid 0, \sigma^2)$

```

1 def log_likelihood(X, y, c, sigma2):
2     N = len(y_data)
3     log_like = 0
4     log_like += - N/2.0 * np.log(sigma2)
5     log_like += - 1.0/(2 * sigma2) * vector_norm(y - X.dot(c))**2
6     return log_like

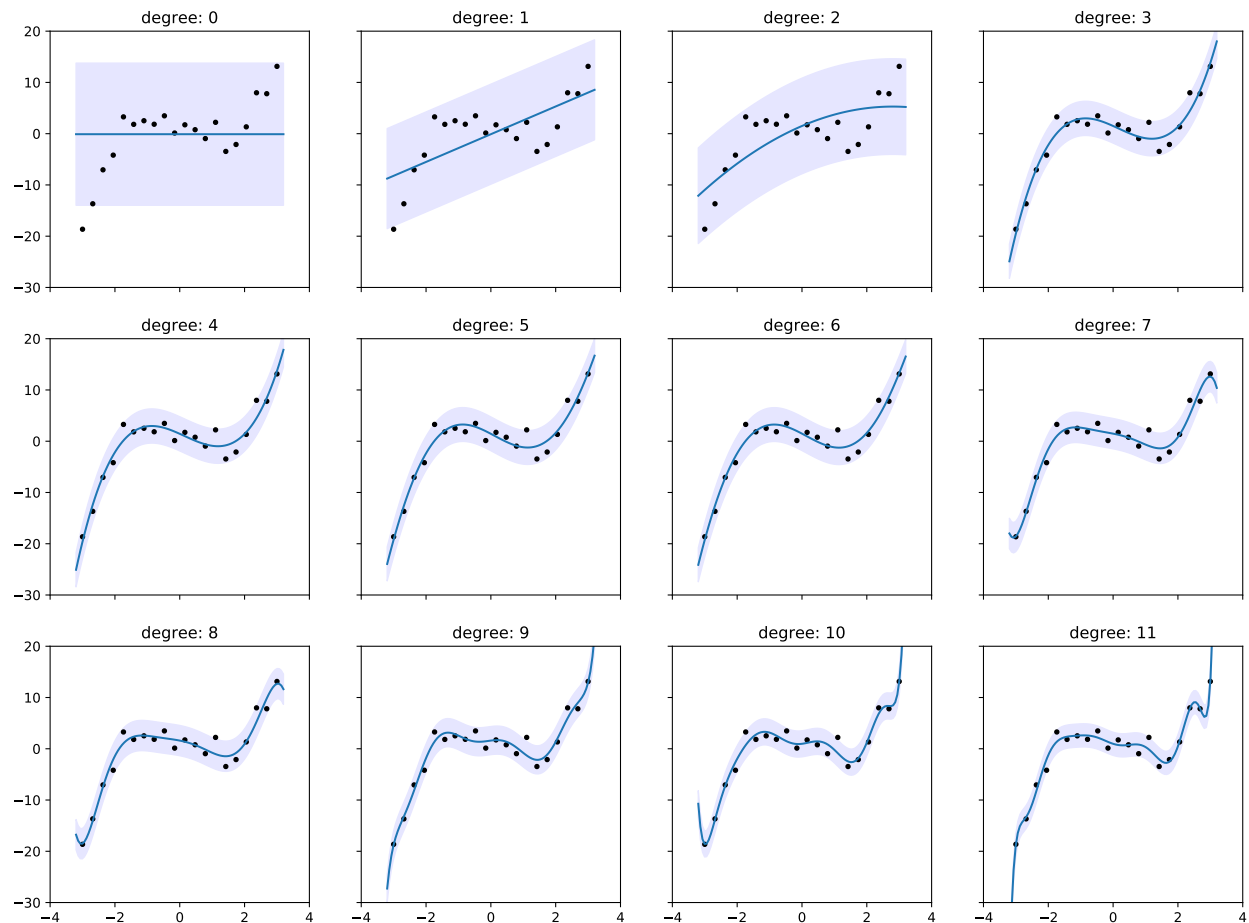
```

- MLE solution: $c_{\text{MLE}} = (X^\top X)^{-1} X^\top y$, $(\sigma^2)_{\text{MLE}} = \frac{1}{N} \|y - X c_{\text{MLE}}\|^2$

```

1 def fit_MLE_ploynomial(X, y):
2     c_MLE = inv(X.T.dot(X)).dot(X.T).dot(y_data)
3     sigma2_MLE = 1.0/len(y) * vector_norm(y - X.dot(c_MLE))**2
4     return c_MLE, sigma2_MLE

```



- $AIC_k = -2[L_k - (k + 2)]$

```

1 def AIC(X, y, c, sigma2):
2     dim = len(c) + 1
3     loglike = log_likelihood(X, y, c, sigma2)
4     return -2 * (loglike - dim)

```

- $BIC_k = -2[L_k - \frac{\log N}{2}(k + 2)]$

```

1 def BIC(X, y, c, sigma2):
2     N = len(y)
3     dim = len(c) + 1
4     loglike = log_likelihood(X, y, c, sigma2)
5     return -2 * (loglike - np.log(N)/2.0 * dim)

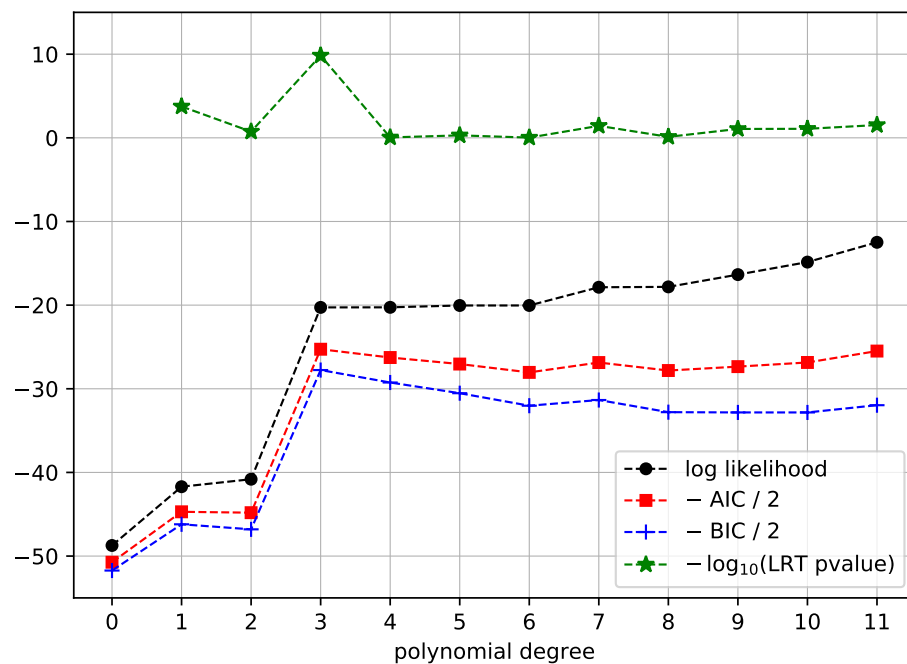
```

- $LRT\ pvalue_k = 1 - cdf\ \chi^2(2(L_k - L_{k-1}) \mid dof = 1)$
(calculated against the model with one less degree)

```

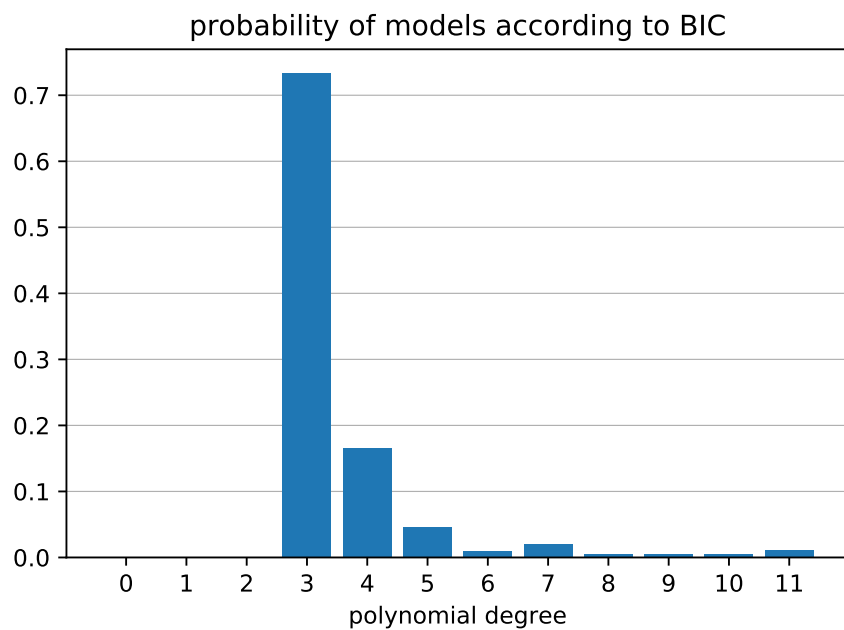
1 from scipy.stats import chi2
2
3 pvalues = [np.nan]
4 for deg in degrees[1:]:
5     L1 = loglikes[deg]
6     L0 = loglikes[deg-1]
7     logLR = L1 - L0
8     dof = 1
9     pvalue = chi2.sf(2*logLR, dof)
10    pvalues.append(pvalue)

```



- BIC weights, $P(D | M_k) \approx e^{-\text{BIC}_k/2} / \sum_{k'=0}^K e^{-\text{BIC}_{k'}/2}$

```
1 def BIC_weights(BICs):  
2     BICs = np.array(BICs)  
3     w = BICs - np.min(BICs) # for numerical stability  
4     w = np.exp(-0.5*(w))  
5     w /= np.sum(w)  
6     return w
```



4 Graphical models

4.1 Elements

- $P(x, y) = P(y | x)P(x)$ is represented by

$$x \longrightarrow y$$

- $P(x, y, z) = P(y | z, x)P(z | x)P(x)$.

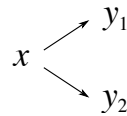
If $P(y | z, x) = P(y | z)$, then it is represented by a **chain**

$$x \longrightarrow z \longrightarrow y$$

Note: $y \perp\!\!\!\perp x | z$, but $y \not\perp\!\!\!\perp x | \emptyset$.

- $P(x, y_1, y_2) = P(y_1, y_2 | x)P(x)$.

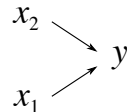
If $P(y_1, y_2 | x) = P(y_1 | x)P(y_2 | x)$, then it is represented by a **fork**



Note: $y_1 \perp\!\!\!\perp y_2 | x$, but $y_1 \not\perp\!\!\!\perp y_2 | \emptyset$.

- $P(x_1, x_2, y) = P(y | x_1, x_2)P(x_1, x_2)$.

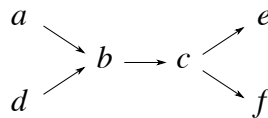
If $P(x_1, x_2) = P(x_1)P(x_2)$, then it is represented by a **collider**



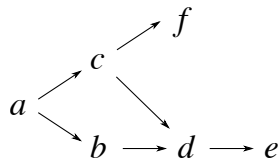
Note: $x_1 \perp\!\!\!\perp x_2 | \emptyset$, but $x_1 \not\perp\!\!\!\perp x_2 | y$ (!).

Examples

- $P(a, b, c, d, e) = P(a)P(d)P(b | a, d)P(c | b)P(e | c)P(f | c)$ is represented by



- $P(a, b, c, d, e, f) = P(a)P(c | a)P(b | a)P(f | c)P(d | b, c)P(e | d)$ is represented by

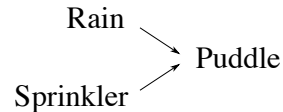


4.2 Real-life examples

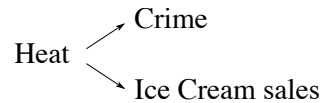
- Fire causes Smoke, Smoke causes Alarm to set off, but given Smoke, there's no correlation between Fire and Alarm, i.e. $\text{Fire} \perp\!\!\!\perp \text{Alarm} \mid \text{Smoke}$. This is represented by a chain

$$\text{Fire} \longrightarrow \text{Smoke} \longrightarrow \text{Alarm}$$

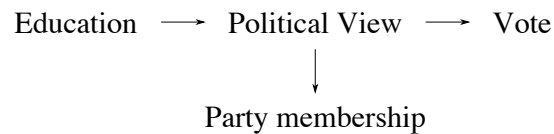
- Both rain and the Sprinkler can cause the formation of a Puddle, they are however independent (until we observe the Puddle), i.e. $\text{Rain} \perp\!\!\!\perp \text{Sprinkler} \mid \emptyset$. This is represented by a collider



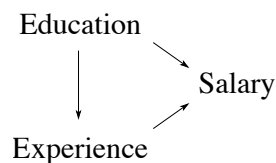
- Heat causes both Ice Cream sales and Crime to increase, but once we know there was a heatwave, they become independent, i.e. $\text{Crime} \perp\!\!\!\perp \text{Ice Cream} \mid \text{Heat}$. This is represented by a fork



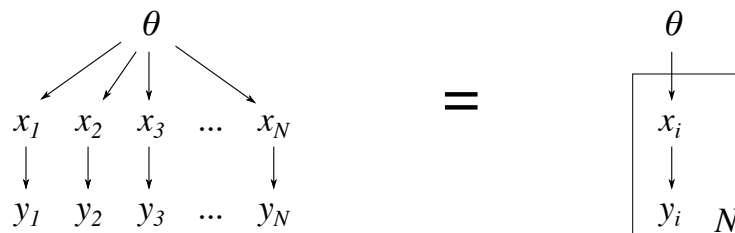
- Education affects Political View, which affects both Party membership and Voting behavior. This can be represented as



- Education and Experience both affect Salary, but Education also affects Experience. This can be represented as



4.3 Plate notation

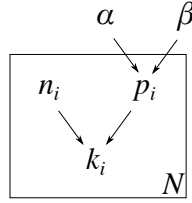


4.4 Hierarchical models

Beta-Binomial model

- Data: $D = \{(k_i, n_i)\}_{i=1}^N$, where k_i (successes) $\in \{0, 1, \dots, n_i\}$, and n_i (attempts) $\in \mathbb{N}$
- Model:
 - level 1: the parameters, $p = \{p_i\}_{i=1}^N$, where $p_i \in [0, 1]$, define $P(k_i | n_i, p_i) = \text{Binomial}(k_i | n_i, p_i)$
 - level 2: the parameters, α, β (both > 0), define $P(p_i | \alpha, \beta) = \text{Beta}(p_i | \alpha, \beta)$.

This hierarchy can be represented as



- Joint likelihood:

$$P(D, p | \alpha, \beta) = \prod_{i=1}^N \left[\text{Binom}(k_i | n_i, p_i) \text{Beta}(p_i | \alpha, \beta) \right]$$

- Marginal likelihood:

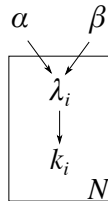
$$P(D | \alpha, \beta) = \prod_{i=1}^N \left[\int dp_i \text{Binom}(k_i | n_i, p_i) \text{Beta}(p_i | \alpha, \beta) \right] = \prod_{i=1}^N \left[\text{Beta-Binom}(k_i | n_i, \alpha, \beta) \right]$$

$$\text{where } \text{Beta-Binom}(k | n, \alpha, \beta) = \frac{\Gamma(n+1)\Gamma(\alpha+\beta)}{\Gamma(n+\alpha+\beta)} \frac{\Gamma(k+\alpha)}{\Gamma(k+1)\Gamma(\alpha)} \frac{\Gamma(n-k+\beta)}{\Gamma(n-k+1)\Gamma(\beta)}$$

Gamma-Poisson model (aka. Negative Binomial model)

- Data: $D = \{k_i\}_{i=1}^N$, where k_i (events) $\in \mathbb{N}$.
- Model:
 - level 1: the parameters $\lambda = \{\lambda_i\}_{i=1}^N$, where $\lambda_i > 0$, define $P(k_i | \lambda) = \text{Poisson}(k_i | \lambda_i)$
 - level 2: the parameters α, β (both > 0), define $P(\lambda_i | \alpha, \beta) = \text{Gamma}(\lambda_i | \alpha, \beta)$.

This hierarchy can be represented as



- Joint likelihood:

$$P(D, \lambda | \alpha, \beta) = \prod_{i=1}^N \left[\text{Poisson}(k_i | \lambda_i) \text{Gamma}(\lambda_i | \alpha, \beta) \right]$$

- Marginal likelihood:

$$P(D \mid \alpha, \beta) = \prod_{i=1}^N \left[\int d\lambda_i \text{Poisson}(k_i \mid \lambda_i) \text{Gamma}(\lambda_i \mid \alpha, \beta) \right] = \prod_{i=1}^N \left[\text{Gamma-Poisson}(k_i \mid \alpha, \beta) \right]$$

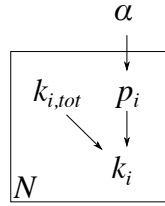
where $\text{Gamma-Poisson}(k \mid \alpha, \beta) = \frac{\Gamma(k + \alpha)}{\Gamma(k + 1)\Gamma(\alpha)} \left(\frac{1}{\beta + 1} \right)^k \left(\frac{\beta}{\beta + 1} \right)^\alpha =$

$$= \text{NegativeBinom}(k \mid r, p) = \binom{k + r - 1}{k} p^k (1 - p)^r, \quad \text{with } r = \alpha, p = \frac{1}{\beta + 1}$$

Dirichlet-Multinomial model

- Data: $D = \{k_i \in \mathbb{N}^M\}_{i=1}^N = \{(k_{i,1}, k_{i,2}, \dots, k_{i,M})\}_{i=1}^N$, where $k_{i,j}$ (number of outcome j) $\in \mathbb{N}$
- Model:
 - level 1: the parameters $p = \{p_i \in \mathbb{R}^M\}_{i=1}^N = \{(p_{i,1}, p_{i,2}, \dots, p_{i,M})\}_{i=1}^N$, where $p_{i,j}$ (probability of outcome j in sample i) > 0 , and $\sum_{j=1}^M p_{i,j} = 1, \forall i$, define $P(k_i \mid p_i) = \text{Multinomial}(k_i \mid k_{i,\text{tot}}, p_i)$, where $k_{i,\text{tot}} = \sum_{j=1}^M k_{i,j}$
 - level 2: the parameters $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_M)$, where each $\alpha_j > 0$, define $P(p_i \mid \alpha) = \text{Dirichlet}(p_i \mid \alpha)$

This hierarchy can be represented as



- Joint likelihood:

$$P(D \mid p \mid \alpha) = \prod_{i=1}^N \left[\text{Multinomial}(k_i \mid k_{i,\text{tot}}, p_i) \text{Dirichlet}(p_i \mid \alpha) \right]$$

- Marginal likelihood:

$$P(D \mid \alpha) = \prod_{i=1}^N \left[\int dp_i \text{Multinomial}(k_i \mid k_{i,\text{tot}}, p_i) \text{Dirichlet}(p_i \mid \alpha) \right] = \prod_{i=1}^N \left[\text{Dirichlet-Multinomial}(k_i \mid k_{i,\text{tot}}, \alpha) \right]$$

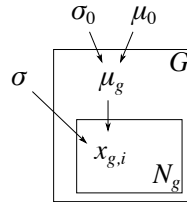
where $\text{Dirichlet-Multinomial}(k \mid k_{\text{tot}}, \{\alpha_j\}_{j=1}^M) = \frac{\Gamma(k_{\text{tot}} + 1)\Gamma(\alpha_{\text{tot}})}{\Gamma(k_{\text{tot}} + \alpha_{\text{tot}})} \prod_{j=1}^M \frac{\Gamma(k_j + \alpha_j)}{\Gamma(k_j + 1)\Gamma(\alpha_j)},$

where $\alpha_{\text{tot}} = \sum_{j=1}^M \alpha_j$

Random Effect Model

- Data: $D = \{x_g\}_{g=1}^G = \{(x_{g,1}, x_{g,2}, \dots, x_{g,N_g})\}_{g=1}^G$, where $x_{g,i} \in \mathbb{R}$ is measurement i in group g , and groups can be of different sizes N_g .
- Model:
 - level 1: The parameters $\mu = \{\mu_g\}_{g=1}^G$ and σ^2 define $P(x_{g,i} \mid \mu_g, \sigma) = \text{Normal}(x_{g,i} \mid \mu_g, \sigma^2)$
 - level 2: The parameter σ_0^2 define $P(\mu_g \mid \mu_0, \sigma_0) = \text{Normal}(\mu_g \mid \mu_0, \sigma_0^2)$

This hierarchy can be represented as



- Joint likelihood:

$$P(D, \mu \mid \mu_0, \sigma_0, \sigma) = \prod_{g=1}^G \left[\text{Normal}(\mu_g \mid \mu_0, \sigma_0^2) \times \prod_{i=1}^{N_g} \text{Normal}(x_{g,i} \mid \mu_g, \sigma^2) \right]$$

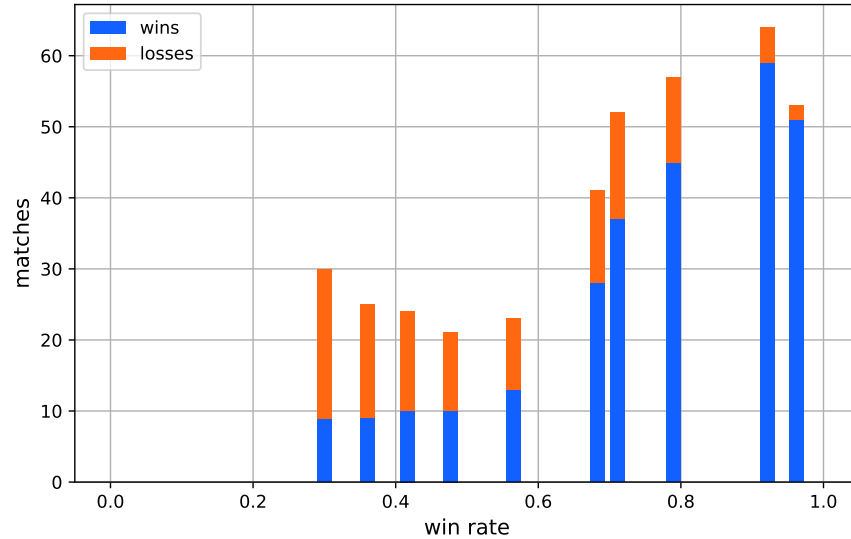
- Marginal likelihood:

$$\begin{aligned} P(D \mid \mu_0, \sigma_0, \sigma) &= \prod_{g=1}^G \int_{-\infty}^{+\infty} d\mu_g \left[\text{Normal}(\mu_g \mid \mu_0, \sigma_0^2) \times \prod_{i=1}^{N_g} \text{Normal}(x_{g,i} \mid \mu_g, \sigma^2) \right] \\ &= (2\pi\sigma_0^2)^{-\frac{G}{2}} \prod_{g=1}^G \left[\frac{1}{\sqrt{\xi + N_g}} (2\pi\sigma^2)^{-\frac{N_g-1}{2}} \exp \left(-\frac{N_g}{2\sigma^2} \left[\frac{\xi}{\xi + N_g} (\mu_g - m_g)^2 + s_g^2 \right] \right) \right] \end{aligned}$$

$$\text{where} \quad \xi = \frac{\sigma^2}{\sigma_0^2}, \quad m_g = \frac{1}{N_g} \sum_{i=1}^{N_g} x_{g,i}, \quad s_g^2 = \frac{1}{N_g} \sum_{i=1}^{N_g} x_{g,i}^2 - m_g^2$$

4.5 Example: Beta-Binomial

Life-time performance 10 different boxers are collected. Wins (k_i) and losses ($n_i - k_i$) are tallied, and the observed win rate $p_{i,\text{obs}} = k_i/n_i$ is calculated. This is shown below



We would like to determine the win rate of an “typical boxer”, i.e. the distribution of the win rate p . While the observed values p_{obs} are good estimates of the individual win rates, 5 boxers played less than 30 matches, while 5 played more than 40, which means the second group provides more information, and their observed win rates need to be taken with more certainty. The Beta-Binomial hierarchical model accounts for this inhomogeneity.

- Data: $\{n_i\} = [24, 23, 30, 21, 25, 53, 41, 52, 64, 57]$, $\{k_i\} = [10, 13, 9, 10, 9, 51, 28, 37, 59, 45]$, with $N = 10$.
- Parameters: $\alpha, \beta > 0$
- Model:

$$\log P(D \mid \alpha, \beta) = \sum_{i=1}^N \log \left(\text{Beta-Binom}(k_i \mid n_i, \alpha, \beta) \right)$$

$$\text{where } \log \left(\text{Beta-Binom}(k \mid n, \alpha, \beta) \right) = -f(n, \alpha + \beta) + f(k, \alpha) + f(n - k, \beta),$$

$$\text{where } f(k, \alpha) = \log \Gamma(k + \alpha) - \log \Gamma(k + 1) - \log \Gamma(\alpha)$$

which can be implemented as

```

1 from scipy.special import gammaln
2
3 def log_three_gamma_term(k, a):
4     return gammaln(k+a) - gammaln(k+1) - gammaln(a)
5
6 def log_beta_binom(k, n, a, b):
7     target = 0
8     target += - log_three_gamma_term(n, a+b)
9     target += log_three_gamma_term(k, a)
10    target += log_three_gamma_term(n-k, b)
11    return target

```

```

12
13 def log_likelihood(k, n, a, b):
14     target = 0
15     for ki, ni in zip(k, n):
16         target += log_beta_binom(ki, ni, a, b)
17     return target

```

- While assuming flat priors for α and β , we can numerically calculate their joint posterior and means.

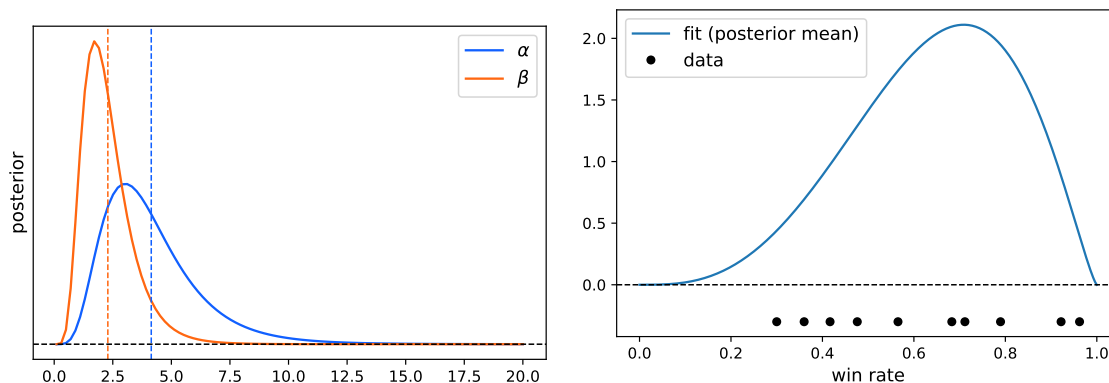
```

1 import numpy as np
2 import pandas as pd
3
4 a_arr, b_arr = np.meshgrid(np.linspace(0.1, 20, 100),
5                             np.linspace(0.1, 20, 100))
6 a_arr = a_arr.flatten()
7 b_arr = b_arr.flatten()
8
9 loglike = []
10 for a, b in zip(a_arr, b_arr):
11     loglike.append(log_likelihood(k, n, a, b))
12
13 df = pd.DataFrame({
14     'a': a_arr,
15     'b': b_arr,
16     'loglike': loglike
17 })
18
19 df['pstar'] = np.exp(df['loglike'] - df['loglike'].max())
20 Z = df['pstar'].sum()
21 df['posterior'] = df['pstar'] / Z
22
23 post_a = df.groupby(by='a')['posterior'].sum().reset_index()
24 post_b = df.groupby(by='b')['posterior'].sum().reset_index()
25
26 a_mean = (post_a['posterior'] * post_a['a']).sum()
27 b_mean = (post_b['posterior'] * post_b['b']).sum()

```

giving $\mathbb{E}(\alpha | D) = 4.142$, $\mathbb{E}(\beta | D) = 2.289$.

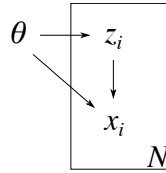
Their (marginal) posteriors and the distribution of the win rate (for the mean α and β values) are below.



5 Hidden variables, EM, Mixture models

5.1 Definitions

- Known values:
 - Observations (or data), $D = \{x_i\}_{i=1}^N$
- Unknown values:
 - Parameters: $\theta = \{\theta_k\}_{k=1}^K$
 - Hidden variables (or hidden data): $Z = \{z_i\}_{i=1}^N$ (i.e. Z is as numerous as D)



5.2 Expectation Maximization

Goal

- Given the likelihood $P(D | Z, \theta)$, and prior on hidden variables $P(Z | \theta)$,
- The joint is $P(D, Z | \theta) = P(D | Z, \theta) P(Z | \theta)$
- The marginal is $P(D | \theta) = \sum_Z P(D, Z | \theta)$
- We wish to find θ that maximizes the marginal, i.e.

$$\theta^{\text{MLE}} = \arg \max_{\theta} \left[\log P(D | \theta) \right] = \arg \max_{\theta} \left[\log \left(\sum_Z P(D, Z | \theta) \right) \right]$$

- Direct numerical optimization is usually feasible, but the EM method is often faster.

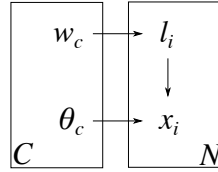
Expectation Maximization (EM) algorithm:

1. Start with realistic $\theta = \theta^{\text{old}}$
 2. E-step: Calculate $P(Z | D, \theta^{\text{old}}) = \frac{P(D, Z | \theta^{\text{old}})}{\sum_{Z'} P(D, Z' | \theta^{\text{old}})}$
 3. M-step: Find the optimal $\theta = \theta^{\text{new}}$ that maximizes $\sum_Z P(Z | D, \theta^{\text{old}}) \log P(D, Z | \theta)$
 4. Set $\theta^{\text{old}} \leftarrow \theta^{\text{new}}$, check convergence, and return to E-step if needed.
- The one-liner iteration formula is based on the joint $P(D, Z | \theta)$:

$$\theta^{\text{new}} = \arg \max_{\theta} \left[\sum_Z \frac{P(D, Z | \theta^{\text{old}})}{\sum_{Z'} P(D, Z' | \theta^{\text{old}})} \log P(D, Z | \theta) \right]$$

5.3 Mixture models

- Data: $D = \{x_i\}_{i=1}^N$
- Parameters: θ, w
 - Components: $c \in \{1, 2, \dots, C\}$
 - Parameters for each component $\theta = \{\theta_c\}_{c=1}^C$
 - Mixing proportions: $w = \{w_c \in [0, 1]\}_{c=1}^C$, such that $\sum_c w_c = 1$.
- Hidden variables: labels $L = \{l_i\}_{i=1}^N$, where $l_i \in \{1, 2, \dots, C\}$
- Model: mixture of distinct distributions
 - Generative distribution for each component: $P(x_i | l_i = c, \theta) = P(x_i | \theta_c)$
 - Probability of an observation coming from a component: $P(l_i = c) = w_c$



- Joint

$$P(D, L | \theta, w) = P(D | L, \theta) P(L | w) = \prod_{i=1}^N P(x_i | \theta_{l_i}) w_{l_i}$$

- Marginal

$$P(D | \theta, w) = \sum_L P(D, L | \theta, w) = \prod_{i=1}^N \left[\sum_{c=1}^C w_c P(x_i | \theta_c) \right]$$

EM algorithm

1. Start with initial values: $\theta^{\text{old}}, w^{\text{old}}$
2. In E-step, calculate

$$P(l_i = c | x_i, \theta^{\text{old}}) = \frac{w_c^{\text{old}} P(x_i | \theta_c^{\text{old}})}{\sum_{c'} w_{c'}^{\text{old}} P(x_i | \theta_{c'}^{\text{old}})} =: r_{i,c}$$

3. In M-step, calculate

$$\begin{aligned} w_c^{\text{new}} &= \frac{1}{N} \sum_{i=1}^N r_{i,c} \\ \theta_c^{\text{new}} &= \arg \max_{\theta_c} \left[\sum_{i=1}^N r_{i,c} \log P(x_i | \theta_c) \right] \\ &= \text{MLE of } \theta \text{ with data weights } \{r_{i,c}\}_{i=1}^N \end{aligned}$$

5.4 Gaussian Mixture Model

(aka. GMM or “soft K-means clustering”)

- Data: $\{x_i\}_{i=1}^N$, where $x_i \in \mathbb{R}^d$ (point in d dimension)
- Parameters:
 - Clusters: $k \in \{1, 2, \dots, K\}$
 - Mixture proportions: $\{w_k \in [0, 1]\}_{k=1}^K$, where $\sum_k w_k = 1$
 - Cluster means: $\mu = \{\mu_k \in \mathbb{R}^d\}_{k=1}^K$
 - Cluster covariances: $\Sigma = \{\Sigma_k \in \mathbb{R}^{d \times d}, \text{positive definite}\}_{k=1}^K$.
- Hidden variables: cluster labels, $L = \{l_i\}_{i=1}^N$, where $l_i \in \{1, 2, \dots, K\}$
- Model

$$P(l_i = k) = w_k$$

$$P(x_i | l_i = k, \mu, \Sigma) = \text{Normal}(x_i | \mu_k, \Sigma_k) = \frac{1}{\sqrt{\det(2\pi\Sigma_k)}} \exp\left(-\frac{1}{2}(x_i - \mu_k)^\top (\Sigma_k)^{-1} (x_i - \mu_k)\right)$$

- Marginal:

$$P(D | \mu, \Sigma, w) = \prod_{i=1}^N \left[\sum_{k=1}^K w_k \text{Normal}(x_i | \mu_k, \Sigma_k) \right]$$

- EM algorithm

1. Choose realistic $w^{\text{old}}, \mu^{\text{old}}, \Sigma^{\text{old}}$ initial values.
2. In E-step, calculate

$$r_{i,k} = \frac{w_k^{\text{old}} \text{Normal}(x_i | \mu_k^{\text{old}}, \Sigma_k^{\text{old}})}{\sum_{k'} w_{k'}^{\text{old}} \text{Normal}(x_i | \mu_{k'}^{\text{old}}, \Sigma_{k'}^{\text{old}})}$$

3. In M-step, calculate

$$w_k^{\text{new}} = \frac{1}{N} \sum_{i=1}^N r_{i,k}$$

$$\mu_k^{\text{new}} = \frac{1}{w_k^{\text{new}} N} \sum_{i=1}^N r_{i,k} x_i$$

$$\Sigma_k^{\text{new}} = \frac{1}{w_k^{\text{new}} N} \sum_{i=1}^N r_{i,k} (x_i - \mu_k^{\text{new}})(x_i - \mu_k^{\text{new}})^\top$$

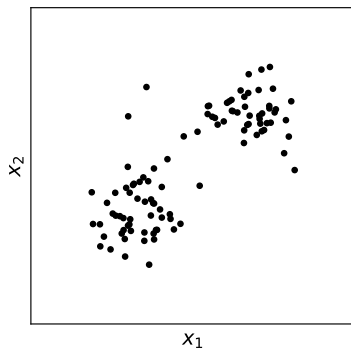
The following python class implements the EM algorithm for fitting GMM.

```

1 class GmmEm:
2     def __init__(self, x):
3         self.x = np.array(x)
4         self.N, self.d = self.x.shape
5         self.K = None
6         self.weights = None
7         self.means = None
8         self.covs = None
9
10    def initialize(self, K):
11        self.K = K
12        m0 = np.mean(x, axis=0)
13        cov0 = np.cov(x.T)
14
15        self.weights = [1.0/K] * K
16        self.means = multivariate_normal.rvs(mean=m0, cov=cov0, size=K)
17        cov_values, _ = np.linalg.eig(cov0)
18        self.covs = np.array([np.eye(self.d) * 0.1 * cov_values.max()
19                               for _ in range(K)])
20
21    def e_step(self):
22        r = []
23        for k in range(K):
24            r.append(self.weights[k] *
25                    multivariate_normal.pdf(self.x,
26                                            mean=self.means[k],
27                                            cov=self.covs[k]))
28
29        r = np.array(r).T
30        r_sum = np.einsum('ik->i', r)
31        r = np.einsum('ik,i->ik', r, 1.0/r_sum)
32        return r
33
34    def m_step(self, r):
35        weights_new = 1.0/N * np.einsum('ik->k', r)
36        means_new = 1.0/N * \
37            np.einsum('k,ik,id->kd',
38                    1.0/weights_new,
39                    r,
40                    self.x)
41        deviations = np.array([self.x - means_new[k] for k in range(self.K)])
42        covs_new = 1.0/N * \
43            np.einsum('k,ik,kid,kiD->kdD',
44                    1.0/weights_new,
45                    r,
46                    deviations,
47                    deviations)
48
49        self.weights = weights_new
50        self.means = means_new
51        self.covs = covs_new

```

Example: GMM in 2D with $K = 2$



Iterating the E- and M-steps a couple of times, we arrive to the final set of r values.

```

1 gmm = GmmEm(x)
2
3 K = 2
4 gmm.initialize(K)
5 for it in range(12):
6     r = gmm.e_step()
7     gmm.m_step(r)

```

