Parameter Estimation

Prof. Seungchul Lee
Industrial AI Lab.

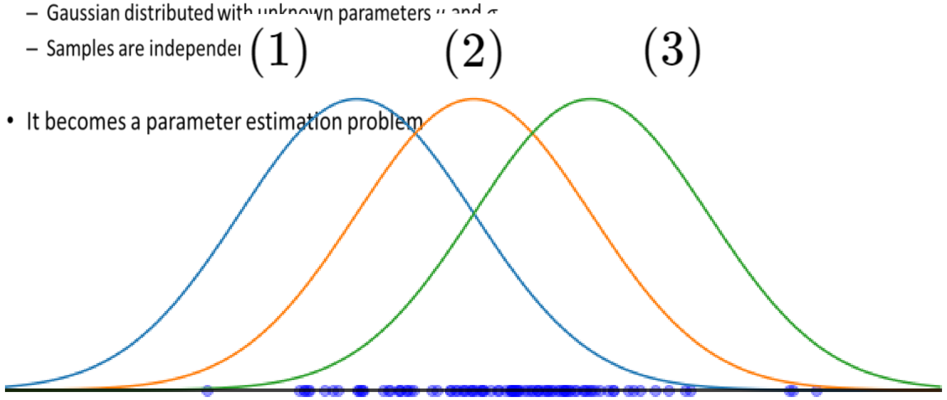
Probability Density Estimation

- Reconstructing the probability density function from a set of given data samples $y_1, y_2, \cdots, y_m$
  - Want to recover the underlying probability density function generating our dataset

Probability Density Estimation

• Reconstructing the probability density function from a set of given data samples $y_1, y_2, \cdots, y_m$
– Want to recover the underlying probability density function generating our dataset

- Reconstructing the probability density function from a set of given data samples $y_1, y_2, \cdots, y_m$
  - Want to recover the underlying probability density function generating our dataset



• Assumption
– Gaussian distributed with unknown parameters
– Samples are independent
• It becomes a parameter estimation problem

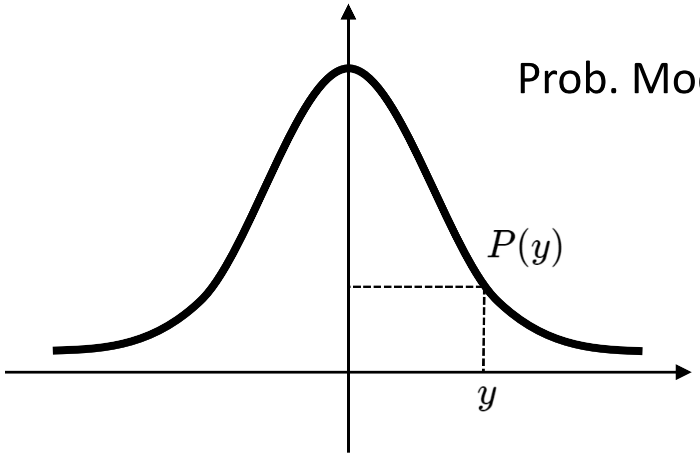
- Assumption
  - Gaussian distributed with unknown parameters μ and σ
  - Samples are independent

- It becomes a parameter estimation problem

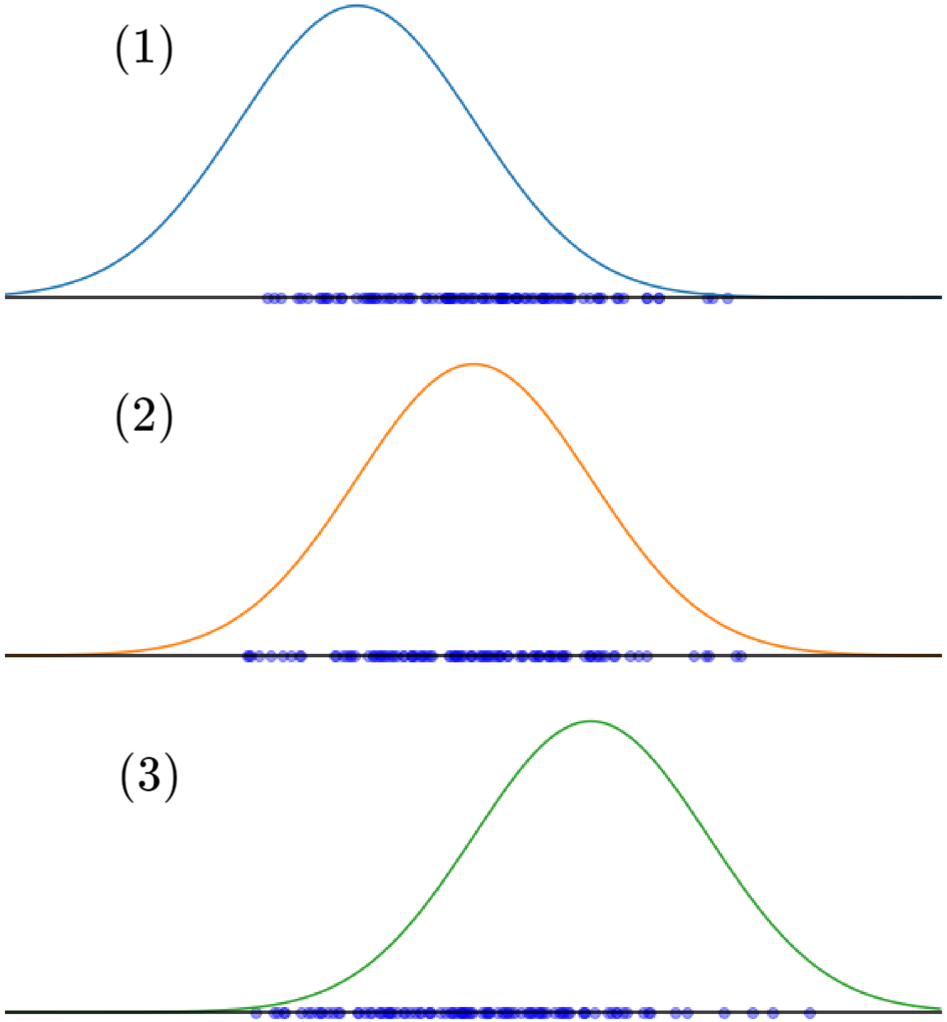
Drawn from a Gaussian Distribution

- Think about how to compute the probability
  - Gaussian distributed with unknown parameters μ and σ
  - Samples are independent

$$P\left(y_1, y_2, \cdots, y_m \,;\, \mu, \sigma^2\right) = \prod_{i=1}^{m} P\left(y_i \,;\, \mu, \sigma^2\right)$$

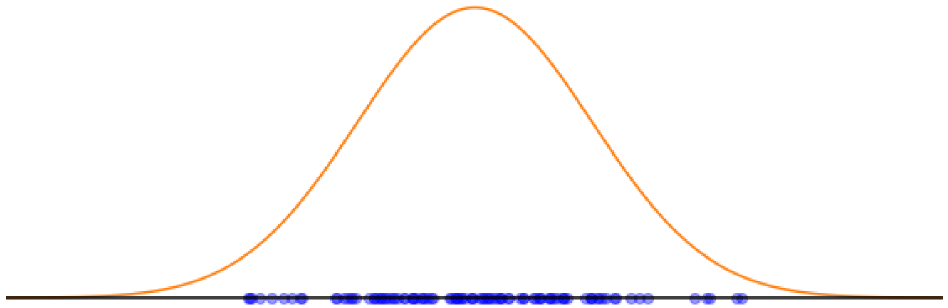


Prob. Model ↔ Data



Drawn from a Gaussian Distribution

- You will often see the following derivation



$$P\left(y = y_i \; ; \; \mu, \sigma^2\right) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(y_i - \mu)^2\right) \; : \text{generative model}$$

$$\begin{aligned} \mathcal{L} = P\left(y_1, y_2, \cdots, y_m \; ; \; \mu, \sigma^2\right) &= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(y_i - \mu)^2\right) \\ &= \frac{1}{(2\pi)^{\frac{m}{2}} \sigma^m} \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{m} (y_i - \mu)^2\right) \end{aligned}$$

$$\ell = \log \mathcal{L} = -\frac{m}{2} \log 2\pi - m \log \sigma - \frac{1}{2\sigma^2} \sum_{i=1}^{m} (y_i - \mu)^2$$

Drawn from a Gaussian Distribution

- To maximize, $\frac{\partial \ell}{\partial \mu} = 0, \frac{\partial \ell}{\partial \sigma} = 0$

$$\frac{\partial \ell}{\partial \mu} = \frac{1}{\sigma^2} \sum_{i=1}^{m} (y_i - \mu) = 0 \quad \Longrightarrow \quad \mu_{MLE} = \frac{1}{m} \sum_{i=1}^{m} y_i \quad : \text{sample mean}$$

$$\frac{\partial \ell}{\partial \sigma} = -\frac{m}{\sigma} + \frac{1}{\sigma^3} \sum_{i=1}^{m} (y_i - \mu)^2 = 0 \quad \Longrightarrow \quad \sigma^2_{MLE} = \frac{1}{m} \sum_{i=1}^{m} (y_i - \mu)^2 \quad : \text{sample variance}$$

- Big lesson
  - We often compute a mean and variance to represent data statistics
  - We kind of assume that a data set is Gaussian distributed
  - Good news: sample mean is Gaussian distributed by the central limit theorem

Maximum Likelihood Estimation (MLE)

Maximum Likelihood Estimation

- Estimate the parameters of a probability distribution by maximizing a likelihood function
- Under the assumed statistical model the observed data is most probable

- Likelihood function
  - The probability density at a particular outcome D when the true value of the parameter is θ

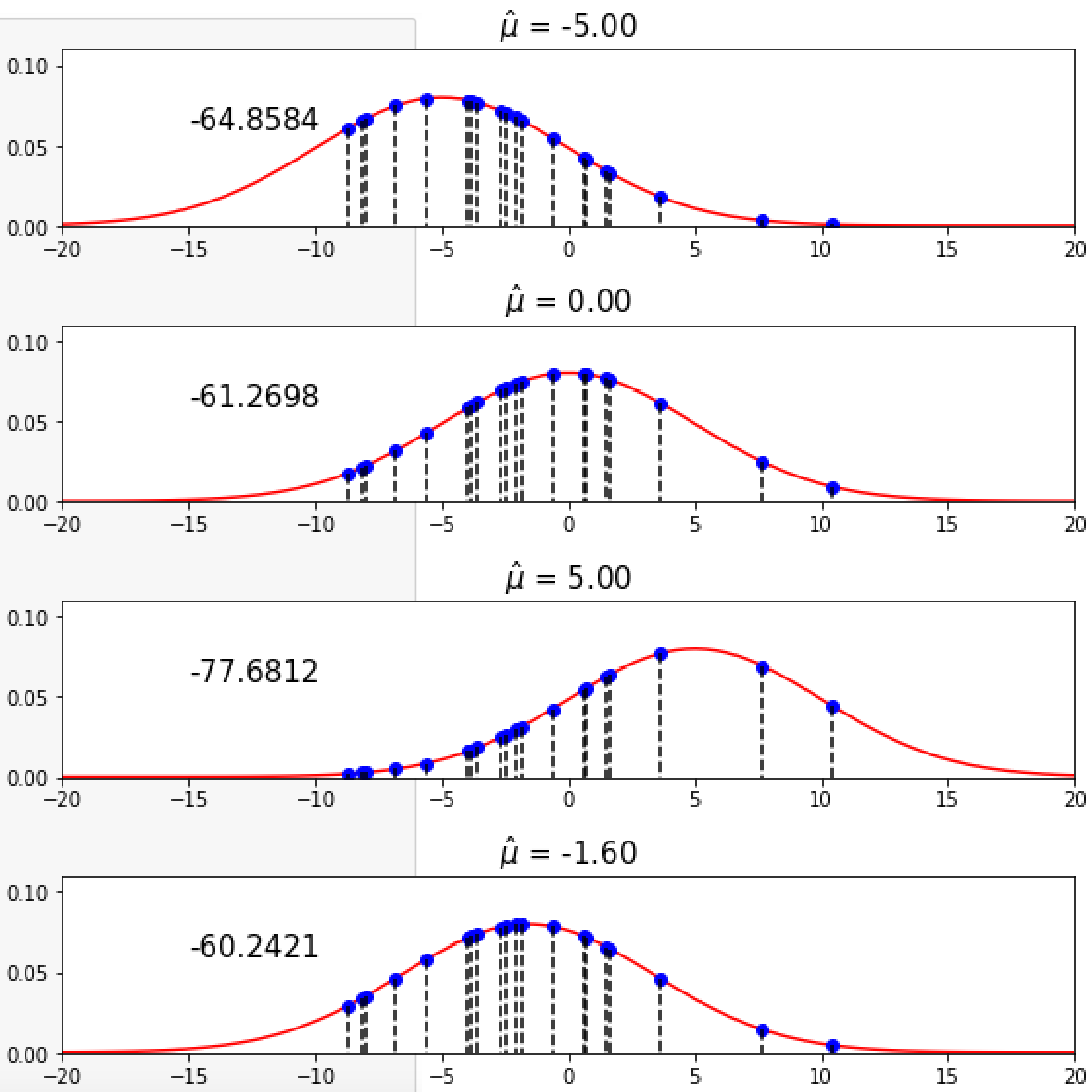
$$\mathcal{L} = P(D \mid \theta) = P(D\,;\,\theta)$$

$$\theta_{\mathrm{MLE}} = \underset{\theta}{\operatorname{argmax}} \; P(D\,;\,\theta)$$

- A model exists first and it generates samples

Numerical Example for Gaussian

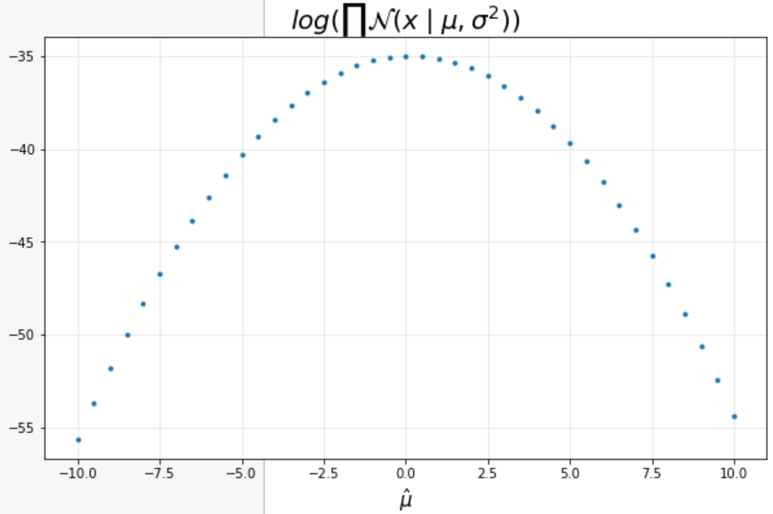
```python
# MLE of Gaussian distribution
# mu

m = 20
mu = 0
sigma = 5

x = np.random.normal(mu,sigma,[m,1])
xp = np.linspace(-20, 20, 100)
y0 = np.zeros([m, 1])

muhat = [-5, 0, 5, np.mean(x)]

plt.figure(figsize=(8, 8))

for i in range(4):
    yp = norm.pdf(xp, muhat[i], sigma)
    y = norm.pdf(x, muhat[i], sigma)
    logL = np.sum(np.log(y))

    plt.subplot(4, 1, i+1)
    plt.plot(xp, yp, 'r')
    plt.plot(x, y, 'bo')
    plt.plot(np.hstack([x, x]).T, np.hstack([y, y0]).T, 'k--')

    plt.title(r'$\hat\mu$ = {0:.2f}'.format(muhat[i]), fontsize=15)
    plt.text(-15,0.06,np.round(logL,4),fontsize=15)
    plt.axis([-20, 20, 0, 0.11])

plt.tight_layout()
plt.show()
```



When Mean is Unknown

```python
# mean is unknown in this example
# variance is known in this example

m = 10
mu = 0
sigma = 5

x = np.random.normal(mu,sigma,[m,1])

mus = np.arange(-10, 10.5, 0.5)
LOGL = []

for i in range(np.size(mus)):
    y = norm.pdf(x, mus[i], sigma)
    logL = np.sum(np.log(y))
    LOGL.append(logL)

muhat = np.mean(x)
print(muhat)

plt.figure(figsize=(10, 6))
plt.plot(mus, LOGL, '.')
plt.title('$log (\prod \mathcal{N}(x \mid \mu , \sigma^2))$', fontsize=20)
plt.xlabel(r'$\hat \mu$', fontsize=15)
plt.grid(alpha=0.3)
plt.show()
```

```
0.160329485196
```

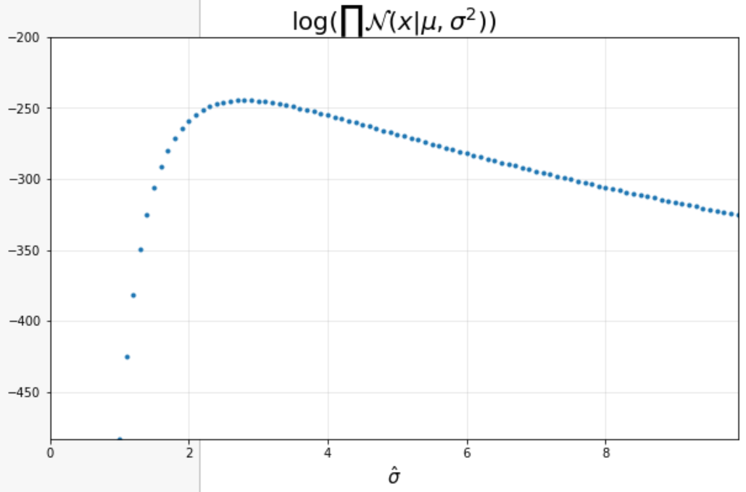


$$\mu_{MLE} = \frac{1}{m} \sum_{i=1}^{m} x_i$$

When Variance is Unknown

```python
# mean is known in this example
# variance is unknown in this example

m = 100
mu = 0
sigma = 3

x = np.random.normal(mu,sigma,[m,1])  # samples

sigmas = np.arange(1, 10, 0.1)
LOGL = []

for i in range(sigmas.shape[0]):
    y = norm.pdf(x, mu, sigmas[i])      # likelihood
    logL = np.sum(np.log(y))
    LOGL.append(logL)

sigmahat = np.sqrt(np.var(x))
print(sigmahat)

plt.figure(figsize=(10,6))
plt.title(r'$\log (\prod \mathcal{N} (x|\mu,\sigma^2))$',fontsize=20)
plt.plot(sigmas, LOGL, '.')
plt.xlabel(r'$\hat \sigma$', fontsize=15)
plt.axis([0, np.max(sigmas), np.min(LOGL), -200])
plt.grid(alpha=0.3)
plt.show()
```

```
2.79684136967
```

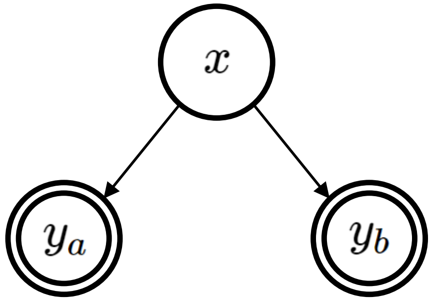


$$\sigma^2_{MLE} = \frac{1}{m}\sum_{i=1}^{m}(y_i - \mu)^2$$

Data Fusion

Data Fusion with Uncertainties

- Two sensors



$$y_a = x + \varepsilon_a,\ \varepsilon_a \sim \mathcal{N}\left(0, \sigma_a^2\right)$$
$$y_b = x + \varepsilon_b,\ \varepsilon_b \sim \mathcal{N}\left(0, \sigma_b^2\right)$$

- In a matrix form

$$y = \begin{bmatrix} y_a \\ y_b \end{bmatrix} = Cx + \varepsilon = \begin{bmatrix} 1 \\ 1 \end{bmatrix} x + \begin{bmatrix} \varepsilon_a \\ \varepsilon_b \end{bmatrix} \qquad \varepsilon \sim \mathcal{N}\left(0, R\right), \quad R = \begin{bmatrix} \sigma_a^2 & 0 \\ 0 & \sigma_b^2 \end{bmatrix}$$

$$\begin{aligned} P\left(y \mid x\right) &\sim \mathcal{N}\left(Cx, R\right) \\ &= \frac{1}{\sqrt{(2\pi)^2 |R|}} \exp\left(-\frac{1}{2}(y - Cx)^T R^{-1} (y - Cx)\right) \end{aligned}$$

Learning Theory (Prof. Reza Shadmehr, Johns Hopkins University)
http://courses.shadmehrlab.org/learningtheory.html

Data Fusion with Uncertainties

- Find $\hat{x}$

$$P(y \mid x) \sim \mathcal{N}(Cx, R)$$
$$= \frac{1}{\sqrt{(2\pi)^2 |R|}} \exp\left(-\frac{1}{2} (y - Cx)^T R^{-1} (y - Cx) \right)$$

$$\ell = -\log 2\pi - \frac{1}{2} \log |R| - \frac{1}{2} \underbrace{(y - Cx)^T R^{-1} (y - Cx)}$$

$$(y - Cx)^T R^{-1} (y - Cx) = y^T R^{-1} y - y^T R^{-1} Cx - x^T C^T R^{-1} y + x^T C^T R^{-1} Cx$$

$$\implies \frac{d\ell}{dx} = 0 = -2C^T R^{-1} y + 2C^T R^{-1} Cx$$
$$\therefore \quad \hat{x} = \left(C^T R^{-1} C \right)^{-1} C^T R^{-1} y$$

Data Fusion with Uncertainties

- $(C^T R^{-1} C)^{-1} C^T R^{-1}$

$$\left(C^T R^{-1} C\right) = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sigma_a^2} & 0 \\ 0 & \frac{1}{\sigma_b^2} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{\sigma_a^2} + \frac{1}{\sigma_b^2}$$

$$C^T R^{-1} = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sigma_a^2} & 0 \\ 0 & \frac{1}{\sigma_b^2} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sigma_a^2} & \frac{1}{\sigma_b^2} \end{bmatrix}$$

$$\hat{x} = \left(C^T R^{-1} C\right)^{-1} C^T R^{-1} y = \left(\frac{1}{\sigma_a^2} + \frac{1}{\sigma_b^2}\right)^{-1} \begin{bmatrix} \frac{1}{\sigma_a^2} & \frac{1}{\sigma_b^2} \end{bmatrix} \begin{bmatrix} y_a \\ y_b \end{bmatrix}$$

$$= \frac{\frac{1}{\sigma_a^2} y_a + \frac{1}{\sigma_b^2} y_b}{\frac{1}{\sigma_a^2} + \frac{1}{\sigma_b^2}}$$

Data Fusion with Uncertainties

$$\therefore \quad \hat{x} = \left(C^T R^{-1} C\right)^{-1} C^T R^{-1} y$$

$$\begin{aligned}
\operatorname{var}(\hat{x}) &= \left(\left(C^T R^{-1} C\right)^{-1} C^T R^{-1}\right) \cdot \operatorname{var}(y) \cdot \left(\left(C^T R^{-1} C\right)^{-1} C^T R^{-1}\right)\right)^T \\
&= \left(\left(C^T R^{-1} C\right)^{-1} C^T R^{-1}\right) \cdot R \cdot \left(\left(C^T R^{-1} C\right)^{-1} C^T R^{-1}\right)\right)^T \\
&= \left(C^T R^{-1} C\right)^{-1} C^T \cdot \left(R^{-1}\right)^T C \left(\left(C^T R^{-1} C\right)^{-1}\right)^T \\
&= \underbrace{\left(C^T R^{-1} C\right)^{-1}} \underbrace{C^T R^{-1} C} \left(\left(C^T R^{-1} C\right)^{-1}\right)^T = \left(C^T R^{-1} C\right)^{-1} \\
&= \frac{1}{\frac{1}{\sigma_a^2} + \frac{1}{\sigma_b^2}} \leq \quad \sigma_a^2, \ \sigma_b^2
\end{aligned}$$

$$\left(C^T R^{-1} C\right) = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sigma_a^2} & 0 \\ 0 & \frac{1}{\sigma_b^2} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{\sigma_a^2} + \frac{1}{\sigma_b^2}$$

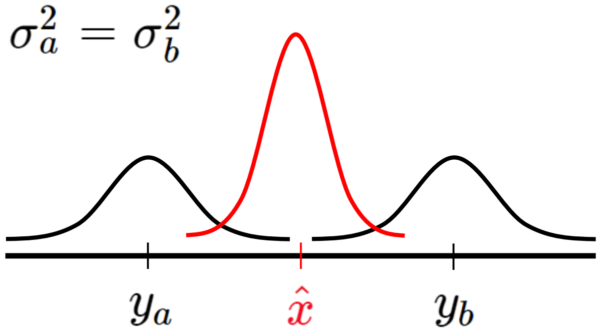
Data Fusion with Less Uncertainties

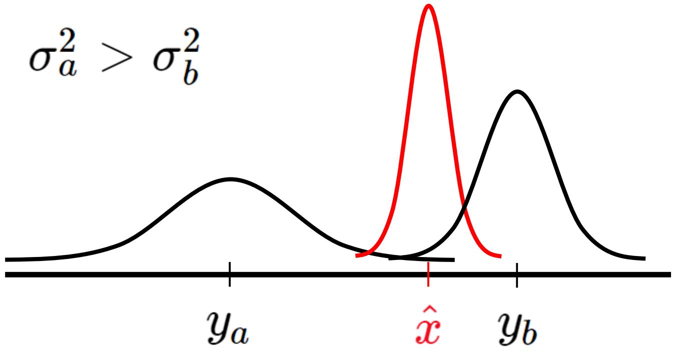
- Summary

$$\hat{x} = \frac{\frac{1}{\sigma_a^2} y_a + \frac{1}{\sigma_b^2} y_b}{\frac{1}{\sigma_a^2} + \frac{1}{\sigma_b^2}}$$

$$\operatorname{var}(\hat{x}) = \frac{1}{\frac{1}{\sigma_a^2} + \frac{1}{\sigma_b^2}} \leq \quad \sigma_a^2, \ \sigma_b^2$$

- Big lesson:
  - Two sensors are better than one sensor $\Longrightarrow$ less uncertainties
  - Accuracy or uncertainty information is also important in sensors





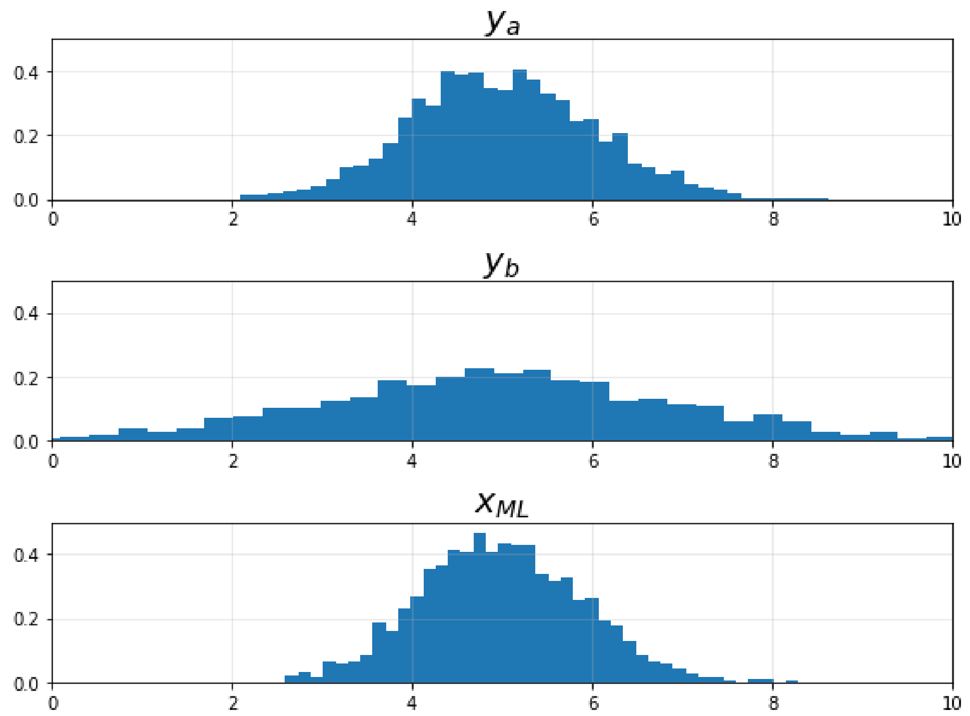
1D Examples

- Example of two rulers

- How brain works on human measurements from both *haptic* and *visual* channels

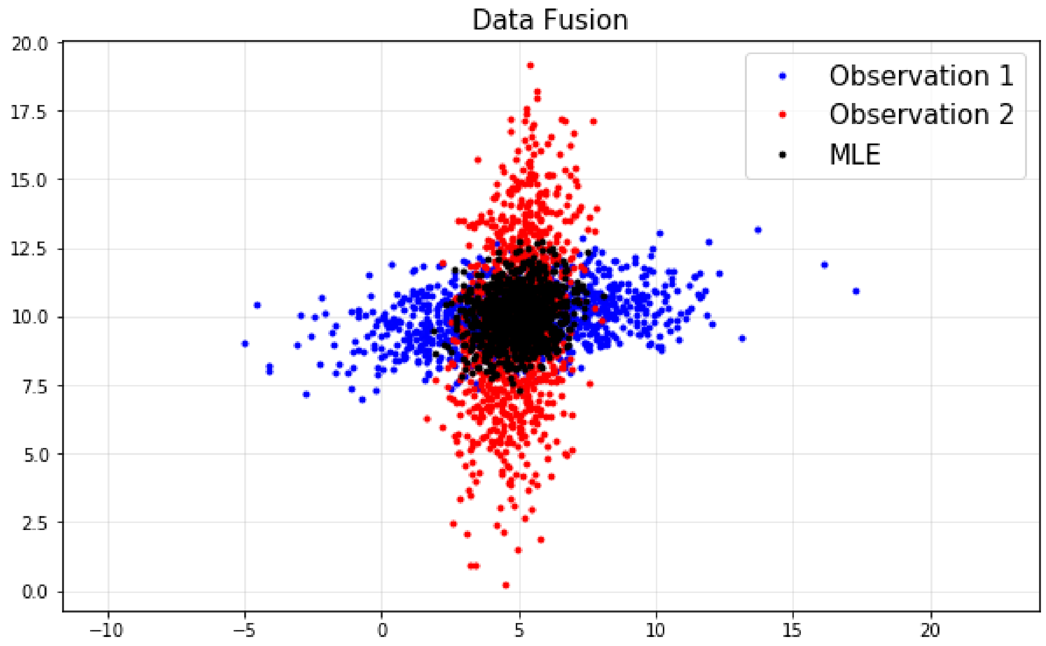
Data Fusion with 1D Example

```
x = 5        # true state (length in this example)
a = 1        # sigma of a
b = 2        # sigma of b

YA = []
YB = []
XML = []

for i in range(2000):
    ya = x + np.random.normal(0,a)
    yb = x + np.random.normal(0,b)
    xml = (1/a**2*ya + 1/b**2*yb)/(1/a**2+1/b**2)
    YA.append(ya)
    YB.append(yb)
    XML.append(xml)
```



Data Fusion with 2D Example

```python
x = np.array([5, 10]).reshape(-1, 1) # true position

mu = np.array([0, 0])
Ra = np.matrix([[9, 1],
               [1, 1]])
Rb = np.matrix([[1, 1],
                [1, 9]])

YA = []
YB = []
XML = []

for i in range(1000):
    ya = x + np.random.multivariate_normal(mu, Ra).reshape(-1, 1)
    yb = x + np.random.multivariate_normal(mu, Rb).reshape(-1, 1)
    xml = (Ra.I+Rb.I).I*(Ra.I*ya+Rb.I*yb)
    YA.append(ya.T)
    YB.append(yb.T)
    XML.append(xml.T)
```



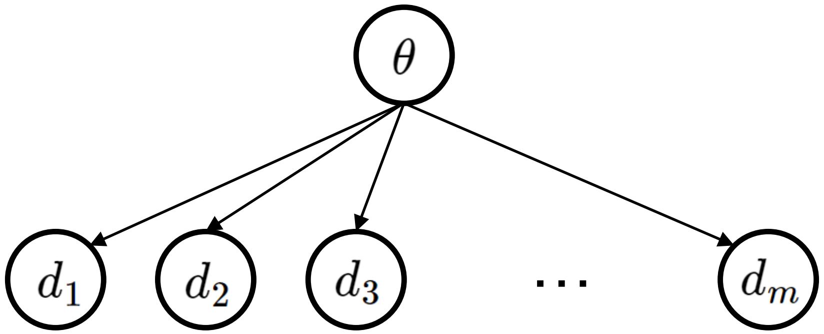
POSTECH

Maximum a Posteriori (MAP) Estimation

Think Differently

- Given $d_1, d_2, \cdots, d_m$,
  - We estimated a probability density function $P(D)$
  - The most reasonable guess for the next sample: $d_{m+1} = E[D]$

- What if there is a hidden variable θ, and we know that samples are generated from it?
  - Here, θ is a random variable



Maximum-a-Posteriori Estimation (MAP)

- Choose θ that maximizes the posterior probability of θ (*i.e.* probability in the light of the observed data)
  - Here, θ is a random variable

$$\theta_{MAP} = \underset{\theta}{\operatorname{argmax}} \; P(\theta \mid D)$$

- Posterior probability of θ is given by the Bayes Rule
  - $P(\theta|D)$: Posterior probability
  - $P(\theta)$: Prior probability of θ (without having seen any data)
  - $P(D|\theta)$: Likelihood
  - $P(D)$: Probability of the data (independent of θ)

$$P(\theta \mid D) = \frac{P(D \mid \theta)P(\theta)}{P(D)}$$

$$P(D) = \int P(\theta)P(D \mid \theta)d\theta$$

- The Bayes rule lets us update our belief about θ in the light of observed data

Maximum-a-Posteriori Estimation (MAP)

- While doing MAP, we usually maximize the log of the posterior probability

$$
\begin{aligned}
\theta_{MAP} = \underset{\theta}{\operatorname{argmax}} \; P(\theta \mid D) &= \underset{\theta}{\operatorname{argmax}} \; \frac{P(D \mid \theta)P(\theta)}{P(D)} \\
&= \underset{\theta}{\operatorname{argmax}} \; P(D \mid \theta)P(\theta) \\
&= \underset{\theta}{\operatorname{argmax}} \; \log P(D \mid \theta)P(\theta) \\
&= \underset{\theta}{\operatorname{argmax}} \; \{\log P(D \mid \theta) + \log P(\theta)\}
\end{aligned}
$$

- For multiple observations $D = \{d_1, d_2, \cdots, d_m\}$
  - Assume independent

$$
\theta_{MAP} = \underset{\theta}{\operatorname{argmax}} \; \left\{ \sum_{i=1}^{m} \log P(d_i \mid \theta) + \log P(\theta) \right\}
$$

- Same as MLE except the extra log-prior-distribution term

Maximum-a-Posteriori Estimation (MAP)

- MAP allows incorporating our prior knowledge about θ in its estimation

$$\theta_{MAP} = \underset{\theta}{\operatorname{argmax}} \; P(\theta \mid D)$$

$$\theta_{MLE} = \underset{\theta}{\operatorname{argmax}} \; P(D \mid \theta)$$

$$\theta_{MAP} = \underset{\theta}{\operatorname{argmax}} \; \left\{ \sum_{i=1}^{m} \log P\left(d_i \mid \theta\right) + \log P(\theta) \right\}$$

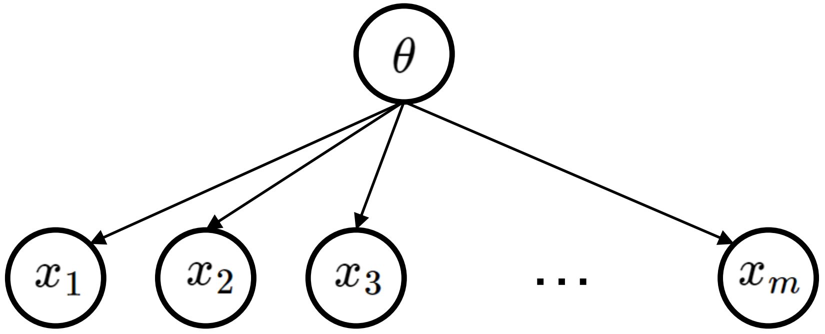
$$\theta_{MLE} = \underset{\theta}{\operatorname{argmax}} \; \left\{ \sum_{i=1}^{m} \log P\left(d_i \mid \theta\right) \right\}$$

MAP for a Univariate Gaussian

- Observations $D = \{x_1, x_2, \cdots, x_m\}$: conditionally independent given θ

$$x_i \sim \mathcal{N}(\theta, \sigma^2)$$

  – Suppose that θ is a random variable with $\theta \sim \mathcal{N}(\mu, 1^2)$, but a prior knowledge
    - Unknown θ but known μ, σ^2



  – Joint Probability

$$P(x_1, x_2, \cdots, x_m \mid \theta) = \prod_{i=1}^{m} P(x_i \mid \theta)$$

MAP for a Univariate Gaussian

- MAP: choose θ_{MAP}

$$\begin{aligned}\theta_{MAP} &= \underset{\theta}{\operatorname{argmax}} \;\; P(\theta \mid D) = \frac{P(D \mid \theta)P(\theta)}{P(D)} \\ &= \underset{\theta}{\operatorname{argmax}} \;\; P(D \mid \theta)P(\theta) \\ &= \underset{\theta}{\operatorname{argmax}} \;\; \{\log P\left(D \mid \theta\right) + \log P(\theta)\}\end{aligned}$$

$$\frac{\partial}{\partial\theta}(\log P\left(D \mid \theta\right)) = \;\cdots\; = \frac{1}{\sigma^2}\left(\sum_{i=1}^{m} x_i - m\theta\right) \qquad \text{(we did in MLE)}$$

$$\begin{aligned}\frac{\partial}{\partial\theta}(\log P\left(\theta\right)) &= \frac{\partial}{\partial\theta}\left(\log\left(\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}(\theta-\mu)^2}\right)\right) \\ &\vdots \\ &= \frac{\partial}{\partial\theta}\left(-\frac{1}{2}\log 2\pi - \frac{1}{2}(\theta-\mu)^2\right) \\ &= \mu - \theta\end{aligned}$$

MAP for a Univariate Gaussian

- MAP: choose θ_{MAP}

$$\Longrightarrow \frac{\partial}{\partial\theta}(\log P(D \mid \theta)) + \frac{\partial}{\partial\theta}(\log P(\theta))$$

$$= \frac{1}{\sigma^2}\left(\sum_{i=1}^{m} x_i - m\theta^*\right) + \mu - \theta^* = 0$$

$$= \frac{1}{\sigma^2}\sum_{i=1}^{m} x_i + \mu - \left(\frac{m}{\sigma^2} + 1\right)\theta^* = 0$$

$$\theta^* = \frac{\frac{1}{\sigma^2}\sum_{i=1}^{m} x_i + \mu}{\frac{m}{\sigma^2} + 1} = \frac{\frac{m}{\sigma^2}\cdot\frac{1}{m}\sum_{i=1}^{m} x_i + 1\cdot\mu}{\frac{m}{\sigma^2} + 1}$$

$$\therefore\ \theta_{MAP} = \frac{\frac{m}{\sigma^2}}{\frac{m}{\sigma^2} + 1}\bar{x} + \frac{1}{\frac{m}{\sigma^2} + 1}\mu \quad \text{: look familiar ?}$$

$$\boxed{\hat{x} = \frac{\frac{1}{\sigma_a^2}y_a + \frac{1}{\sigma_b^2}y_b}{\frac{1}{\sigma_a^2} + \frac{1}{\sigma_b^2}}}$$

MAP for a Univariate Gaussian

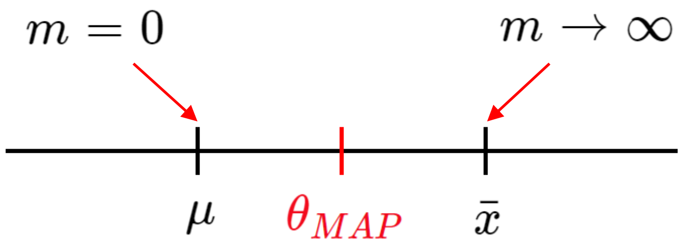
- MLE interpretation:

$$\theta_{MAP} = \frac{\frac{m}{\sigma^2}}{\frac{m}{\sigma^2}+1}\bar{x} + \frac{1}{\frac{m}{\sigma^2}+1}\mu$$

$$\begin{cases} \mu = \text{prior mean} \\ \bar{x} = \text{sample mean} \end{cases}$$

$$\begin{cases} \mu = \text{1st observation} & \sim \mathcal{N}\left(0, 1^2\right) \\ \bar{x} = \text{2nd observation} & \sim \mathcal{N}\left(0, \left(\frac{\sigma}{\sqrt{m}}\right)^2\right) \end{cases}$$

- Big lesson: a prior acts as a data



- Note: prior knowledge
  - Education
  - Get older

$$\boxed{\hat{x} = \frac{\frac{1}{\sigma_a^2}y_a + \frac{1}{\sigma_b^2}y_b}{\frac{1}{\sigma_a^2} + \frac{1}{\sigma_b^2}}}$$

MAP Example

Example) Experiment in class

- Which one do you think is heavier?
  - with eyes closed

MAP Example

Example) Experiment in class

- Which one do you think is heavier?
  - with eyes closed
  - with visual inspection
  - with haptic (touch) inspection

MAP in Python

- Suppose that θ is a random variable with $\theta \sim N(\mu, 1^2)$, but a prior knowledge
  - (unknown θ and known μ, σ^2)

$$x_i \sim \mathcal{N}(\theta, \sigma^2)$$

  - for mean of a univariate Gaussian

```python
# known
mu = 5
sigma = 2

# unknown theta
theta = np.random.normal(mu,1)
x = np.random.normal(theta, sigma)

print('theta = {:.4f}'.format(theta))
print('x = {:.4f}'.format(x))
```

```
theta = 3.8211
x = 5.7443
```

MAP in Python

```
theta = 3.8211
```

$$\theta_{MAP} = \frac{\frac{m}{\sigma^2}}{\frac{m}{\sigma^2} + 1}\bar{x} + \frac{1}{\frac{m}{\sigma^2} + 1}\mu$$

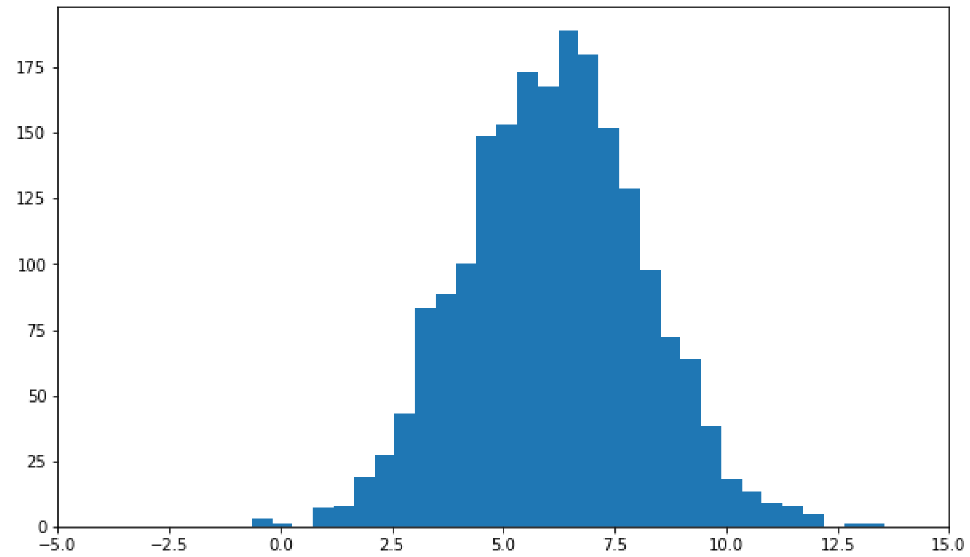
```python
# MAP

m = 4
X = np.random.normal(theta,sigma,[m,1])

xbar = np.mean(X)
theta_MAP = m/(m+sigma**2)*xbar + sigma**2/(m+sigma**2)*mu

print('mu = 5')
print('xbar = {:.4f}'.format(xbar))
print('theta_MAP = {:.4f}'.format(theta_MAP))
```

```
mu = 5
xbar = 2.2625
theta_MAP = 3.6313
```

MAP in Python

```
# theta
mu = 5
theta = np.random.normal(mu,1)

sigma = 2
m = 2000

X = np.random.normal(theta,sigma,[m,1])
X = np.asmatrix(X)

print('theta = {:.4f}'.format(theta))
plt.figure(figsize=(10,6))
plt.hist(X,31)
plt.xlim([-5,15])
plt.show()
```

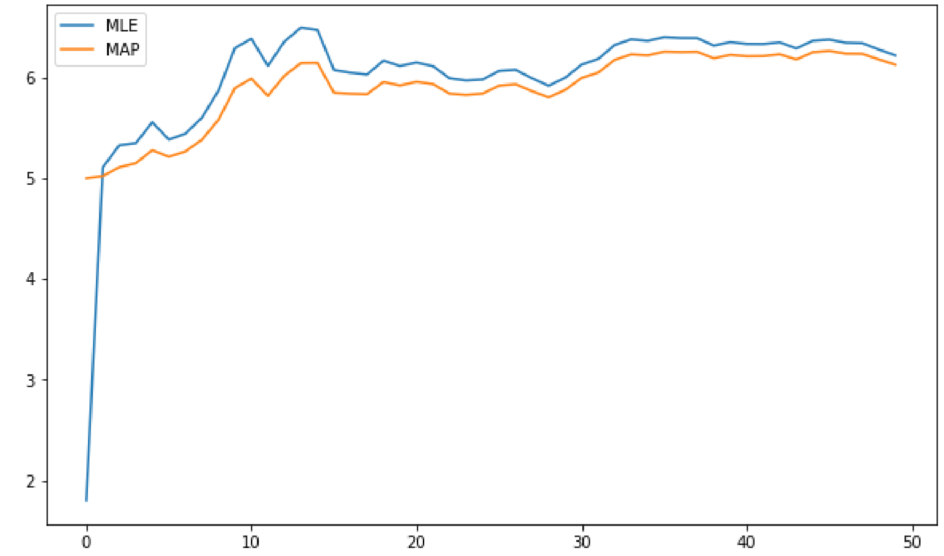
```
theta = 6.1839
```



MAP in Python

```
n = 50
XMLE = []
XMAP = []

for k in range(n):
    xmle = np.mean(X[0:k+1,0])
    xmap = k/(k+sigma**2)*xmle + sigma**2/(k + sigma**2)*mu
    XMLE.append(xmle)
    XMAP.append(xmap)

print('theta = {:.4f}'.format(theta))
plt.figure(figsize=(10,6))
plt.plot(XMLE)
plt.plot(XMAP)
plt.legend(['MLE','MAP'])
plt.show()
```

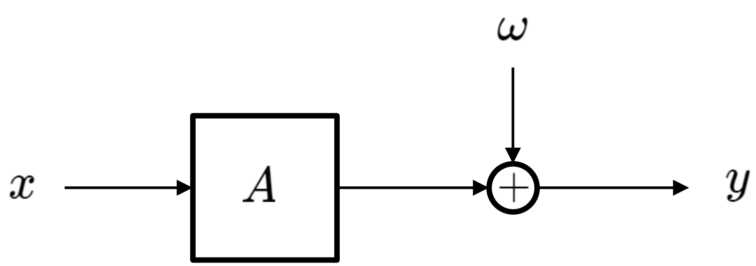
```
theta = 6.1839
```



Linear Measurement with Noise

Linear Measurement

$$y = Ax + \omega$$



- x is what we want to estimate
- y is measured
- A characterizes sensors or measurements
- ω is sensor noise

- Common assumptions
  - $x \sim \mathcal{N}(\mu_x, \Sigma_x)$
  - $\omega \sim \mathcal{N}(\mu_\omega, \Sigma_\omega)$
  - x and ω are independent

$$\begin{bmatrix} x \\ \omega \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_x \\ \mu_\omega \end{bmatrix}, \begin{bmatrix} \Sigma_x & 0 \\ 0 & \Sigma_\omega \end{bmatrix}\right)$$

Linear Measurement

$$y = Ax + \omega$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} I & 0 \\ A & I \end{bmatrix} \begin{bmatrix} x \\ \omega \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} I & 0 \\ A & I \end{bmatrix} \begin{bmatrix} \mu_x \\ \mu_\omega \end{bmatrix}, \begin{bmatrix} I & 0 \\ A & I \end{bmatrix} \begin{bmatrix} \Sigma_x & 0 \\ 0 & \Sigma_\omega \end{bmatrix} \begin{bmatrix} I & 0 \\ A & I \end{bmatrix}^T \right)$$

$$\sim \mathcal{N}\left(\begin{bmatrix} \mu_x \\ A\mu_x + \mu_\omega \end{bmatrix}, \begin{bmatrix} \Sigma_x & \Sigma_x A^T \\ A\Sigma_x & A\Sigma_x A^T + \Sigma_\omega \end{bmatrix} \right) = \mathcal{N}\left(\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} \Sigma_x & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_y \end{bmatrix} \right)$$

$$\Sigma_y = \underbrace{A\Sigma_x A^T}_{\text{signal covariance}} + \underbrace{\Sigma_\omega}_{\text{noise covariance}}$$

Linear Measurement

- Back to estimation problem: estimate x given y

$$\begin{bmatrix} x \\ y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_x \\ A\mu_x + \mu_\omega \end{bmatrix}, \begin{bmatrix} \Sigma_x & \Sigma_x A^T \\ A\Sigma_x & A\Sigma_x A^T + \Sigma_\omega \end{bmatrix}\right)$$

$$= \mathcal{N}\left(\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} \Sigma_x & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_y \end{bmatrix}\right)$$

$$x \mid y \sim \mathcal{N}\left(\mu_x + \Sigma_{xy}\Sigma_y^{-1}(y - \mu_y),\ \Sigma_x - \Sigma_{xy}\Sigma_y^{-1}\Sigma_{yx}\right)$$

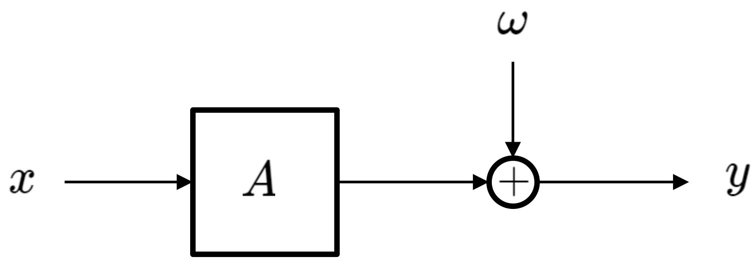
$$\boxed{\begin{aligned} \hat{x} = \phi(y) &= E[x \mid y] \\ &= \mu_x + \Sigma_{xy}\Sigma_y^{-1}(y - \mu_y) \\ \operatorname{cov}(x \mid y) &= \Sigma_x - \Sigma_{xy}\Sigma_y^{-1}\Sigma_{yx} \end{aligned}}$$

Linear Measurement

- Interpretation
  - μ_x is our best prior guess of x (before measurement)
  - $y - \mu_y$ is the discrepancy between what we actually measure (y) and the expected value of y
  - Estimator modifies prior guess by K times this discrepancy
  - Estimator blends prior information with measurement

$$y = Ax + \omega$$

$$\begin{aligned}\hat{x} = \phi(y) &= E[x \mid y] \\ &= \mu_x + \Sigma_{xy}\Sigma_y^{-1}(y - \mu_y) \\ &= \mu_x + K(y - \mu_y)\end{aligned}$$



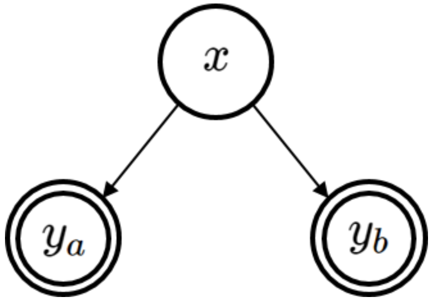
  - Covariance of estimation error is always less than prior covariance of x

$$\mathrm{cov}(x \mid y) = \Sigma_x - \Sigma_{xy}\Sigma_y^{-1}\Sigma_{yx} \leq \Sigma_x$$

Bayesian Thinking

Re-visit Two Sensors Problem

- Assumptions
  - Follows Gaussian distribution
  - Variance of two sensors



$$\mathrm{var}(y_a) = \sigma_a^2$$
$$\mathrm{var}(y_b) = \sigma_b^2$$

- Optimal position estimation $\hat{x}$

$$E[\hat{x}] = \frac{\frac{1}{\sigma_a^2} y_a + \frac{1}{\sigma_b^2} y_b}{\frac{1}{\sigma_a^2} + \frac{1}{\sigma_b^2}} = \frac{\frac{1}{\sigma_a^2}}{\frac{1}{\sigma_a^2} + \frac{1}{\sigma_a^2}} y_a + \frac{\frac{1}{\sigma_b^2}}{\frac{1}{\sigma_a^2} + \frac{1}{\sigma_b^2}} y_b$$

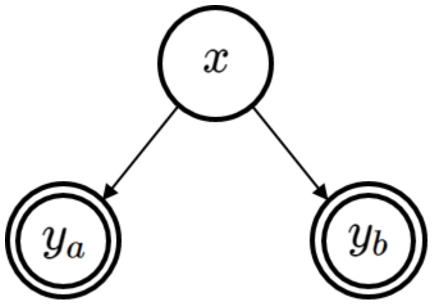
$$\mathrm{var}[\hat{x}] = \frac{1}{\frac{1}{\sigma_a^2} + \frac{1}{\sigma_b^2}} \quad < \quad \sigma_a^2,\ \sigma_b^2 \qquad \Longrightarrow \text{ more accurate}$$

POSTECH

Different Perspective

- Observe sensor A first y_a

$$\hat{x}_1 = y_a$$
$$\hat{\sigma}_1^2 = \sigma_a^2$$



- Combine sensor B observation y_b

$$E[\hat{x}] = \frac{\frac{1}{\sigma_a^2} y_a + \frac{1}{\sigma_b^2} y_b}{\frac{1}{\sigma_a^2} + \frac{1}{\sigma_b^2}} = \frac{\frac{1}{\sigma_a^2}}{\frac{1}{\sigma_a^2} + \frac{1}{\sigma_b^2}} y_a + \frac{\frac{1}{\sigma_b^2}}{\frac{1}{\sigma_a^2} + \frac{1}{\sigma_b^2}} y_b \quad \text{and set } K = \frac{\frac{1}{\sigma_b^2}}{\frac{1}{\sigma_a^2} + \frac{1}{\sigma_b^2}}$$

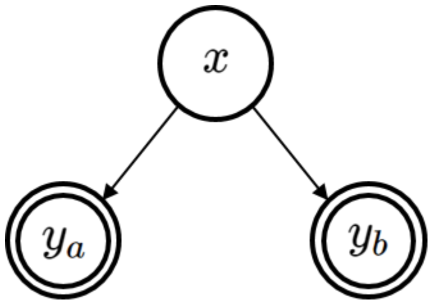
$$\begin{aligned} \hat{x}_2 = E[\hat{x}] &= (1 - K) y_a + K y_b \\ &= (1 - K) \hat{x}_1 + K y_b \\ &= \hat{x}_1 + K (y_b - \hat{x}_1) \end{aligned}$$

- Optimal estimation
  - Prediction first, then correct or update with prediction error

$$\underbrace{\hat{x}_2}_{\text{final estimation}} = \underbrace{\hat{x}_1}_{\text{prediction}} + K\overbrace{(\underbrace{y_b}_{\text{measured}} - \hat{x}_1)}^{\text{prediction error}}$$

Linear Measurement

$$\hat{x}_1 = y_a$$
$$\hat{\sigma}_1^2 = \sigma_a^2$$



$$\begin{bmatrix} x \\ y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_x \\ A\mu_x + \mu_\omega \end{bmatrix}, \begin{bmatrix} \Sigma_x & \Sigma_x A^T \\ A\Sigma_x & A\Sigma_x A^T + \Sigma_\omega \end{bmatrix}\right)$$
$$= \mathcal{N}\left(\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} \Sigma_x & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_y \end{bmatrix}\right)$$

$$\hat{x}_2 = \hat{x}_1 + K(y_b - \hat{x}_1)$$

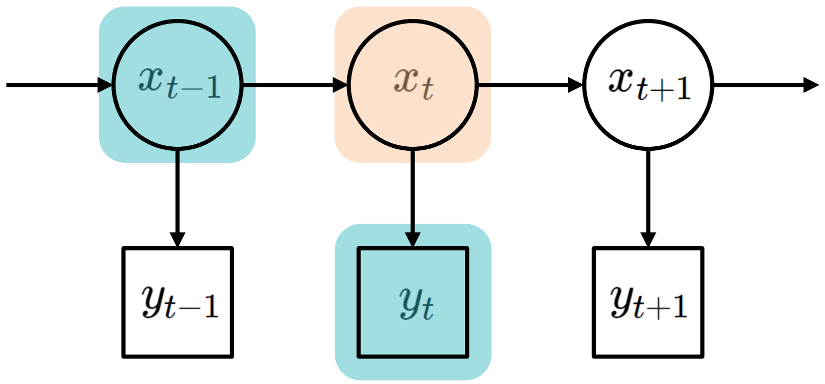
$$\hat{x}_2 = y_a + K(y_b - y_a)$$

$$\begin{aligned} \hat{x} = \phi(y) &= E[x \mid y] \\ &= \mu_x + \Sigma_{xy}\Sigma_y^{-1}(y - \mu_y) \\ &= \mu_x + \Sigma_x A^T(A\Sigma_x A^T + \Sigma_\omega)^{-1}(y - \mu_y) \\ &= \mu_x + K(y - \mu_y) \end{aligned}$$

$$\begin{aligned} K = \Sigma_x A^T(A\Sigma_x A^T + \Sigma_\omega)^{-1} &= \sigma_a^2(\sigma_2^2 + \sigma_b^2)^{-1} \\ &= \frac{\sigma_a^2}{\sigma_a^2 + \sigma_b^2} = \frac{\frac{1}{\sigma_b^2}}{\frac{1}{\sigma_a^2} + \frac{1}{\sigma_b^2}} \end{aligned}$$

$$\mathrm{cov}(x \mid y) = \Sigma_x - \Sigma_{xy}\Sigma_y^{-1}\Sigma_{yx} \leq \Sigma_x$$

Bayesian Kalman Filter



- Discrete linear dynamical system of motion

$$
\begin{aligned}
x_{t+1} &= Ax_t + Bu_t \\
y_t &= Cx_t
\end{aligned}
$$

- Kalman filter has a very nice Bayesian interpretation
  - Model x_t with a Gaussian

$$p(x_t) = \mathcal{N}(x_t, \Sigma_t)$$

  - Prediction using state dynamics model

$$p(x_t \mid x_{t-1})$$

  - Inference from noisy measurements

$$p(y_t \mid x_t)$$

- Applicable to time series data

Bayesian Kalman Filter

- Prediction
  - We do not have any more data, we just update our earlier posterior of $\hat{x}_{t-1}$ to give a new posterior of $\hat{x}_t$
  - We are just propagating the pmf under the linear dynamics

$$
\begin{aligned}
x_{t+1} &= Ax_t + Bu_t \\
y_t &= Cx_t
\end{aligned}
$$

- Correction
  - We start with $\hat{x}_t$, and use it as the prior for our next measurement of $\hat{y}_t$

$$\hat{x}_t \leftarrow \hat{x}_t + K(y_t - C\hat{x}_t)$$

- (Recursive) Repeat over time

Object Tracking in Computer Vision

- Lecture: Introduction to Computer Vision by Prof. Aaron Bobick at Georgia Tech

Object Tracking in Computer Vision

Kalman Filter

- Computationally efficient
  - Recursive

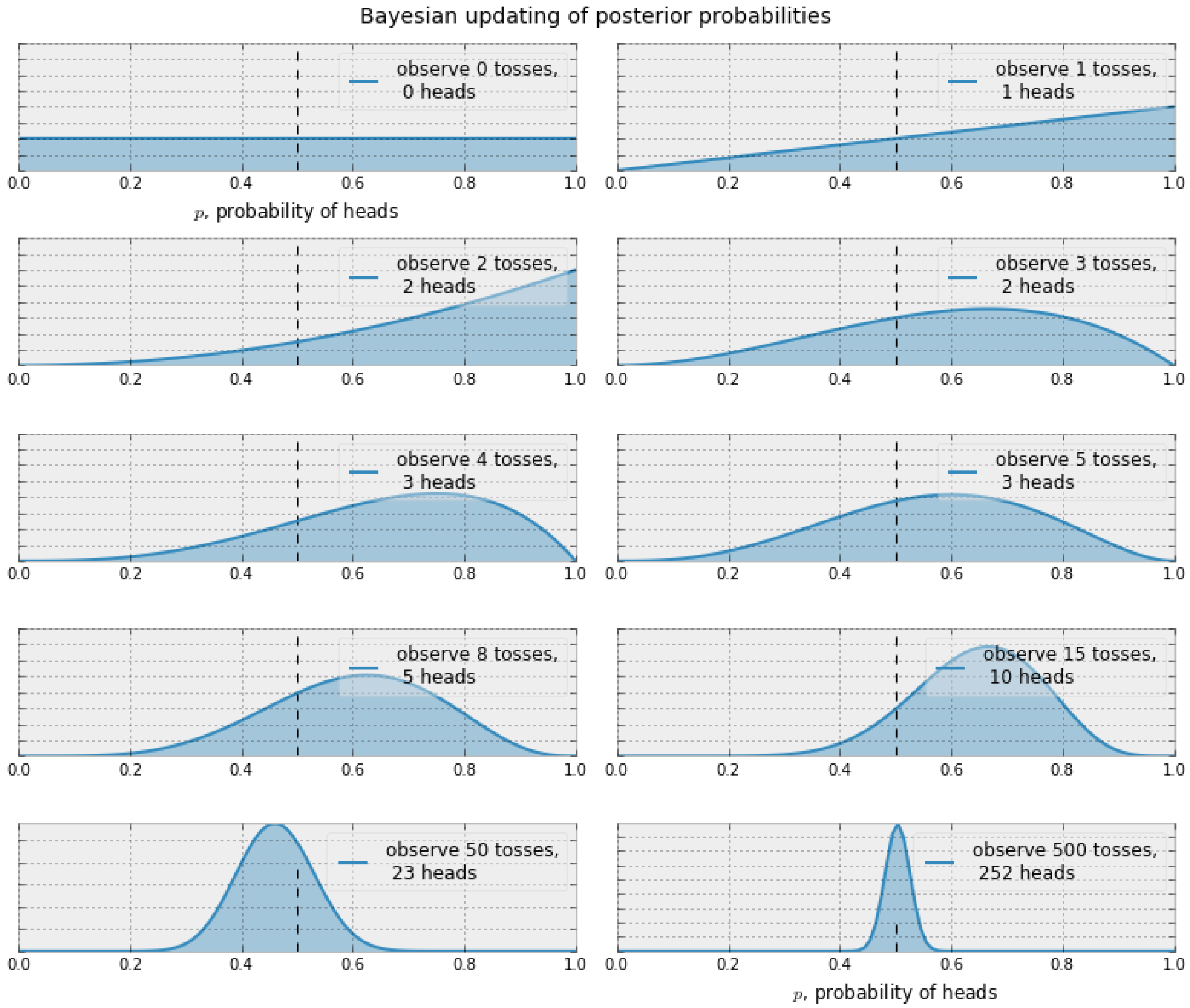
- Bayesian
  - Data fusion

Recursive Bayesian Estimation

Bayesian Inference

- Start with prior beliefs, which can be thought of as a summary of opinions.
  - might be subjective
- Given our prior, we can update our opinion, and produce a new opinion.
  - This new distribution is called the posterior
- Iterate
  - if more data is available

Coin-Flip Example



Probabilistic Machine Learning

- Maximum Likelihood Estimation (MLE)
- Maximum a Posteriori Estimation (MAP)

- Probabilistic Machine Learning
  - I personally believe this is a more fundamental way of looking at machine learning

- Probabilistic Regression
- Probabilistic Classification
- Probabilistic Clustering
- Probabilistic Dimension Reduction