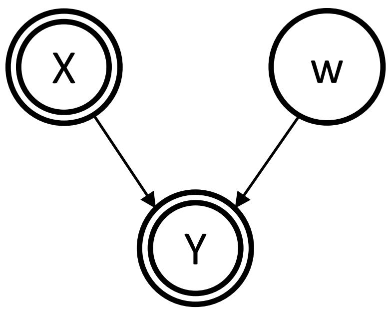
Parameter Estimation

Industrial AI Lab.

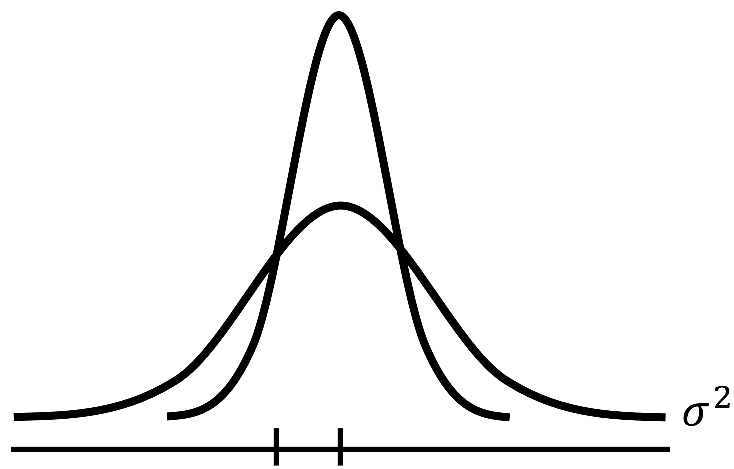
Generative Model

$$P\left(y \mid X, \omega, \sigma^2\right) = \mathcal{N}\left(\omega^T X, \sigma^2\right)$$



$$y = \omega^T x + \varepsilon$$

$$\varepsilon \sim \mathcal{N}(0, \sigma^2)$$



Maximum Likelihood Estimation (MLE)

- Estimate parameters $\theta(\omega, \sigma^2)$ such that maximize the likelihood given a generative model
  - Given observed data

$$D = \{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\}$$

  - Generative model structure (assumption)

$$\begin{aligned} y_i &= \hat{y}_i + \varepsilon \\ &= \omega^T x_i + \varepsilon, \quad \varepsilon \sim \mathcal{N}\left(0, \sigma^2\right) \end{aligned}$$

Maximum Likelihood Estimation (MLE)

- Find parameters ω and σ that maximize the likelihood over the observed data
- Likelihood:

$$\begin{aligned}
\mathcal{L}(\omega, \sigma) &= P\left(y_1, y_2, \cdots, y_m \mid x_1, x_2, \cdots, x_m;\ \underbrace{\omega, \sigma}_{\theta}\right) \\
&= \prod_{i=1}^{m} P\left(y_i \mid x_i;\ \omega, \sigma\right) \\
&= \frac{1}{\left(2\pi\sigma^2\right)^{\frac{m}{2}}} \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{m} (y_i - \omega^T x_i)^2\right)
\end{aligned}$$

- Perhaps the simplest (but widely used) parameter estimation method

Drawn from a Gaussian Distribution

- You will often see the following derivation

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

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

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

Drawn from a Gaussian Distribution

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

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

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

- 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

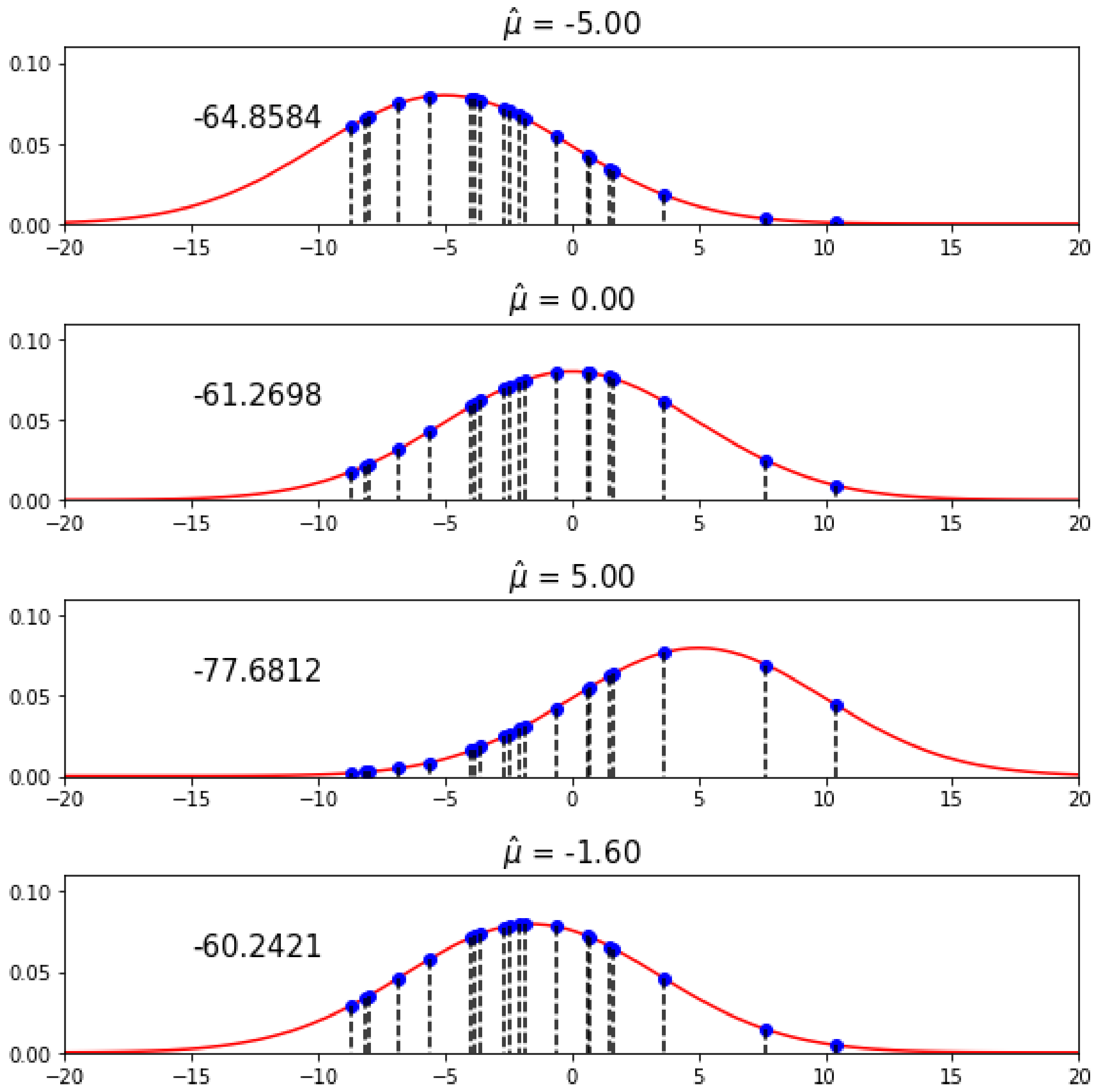
Numerical Example

- Compute the likelihood function, then
  - Maximize the likelihood function
  - Adjust the mean and variance of the Gaussian to maximize its product

Numerical Example

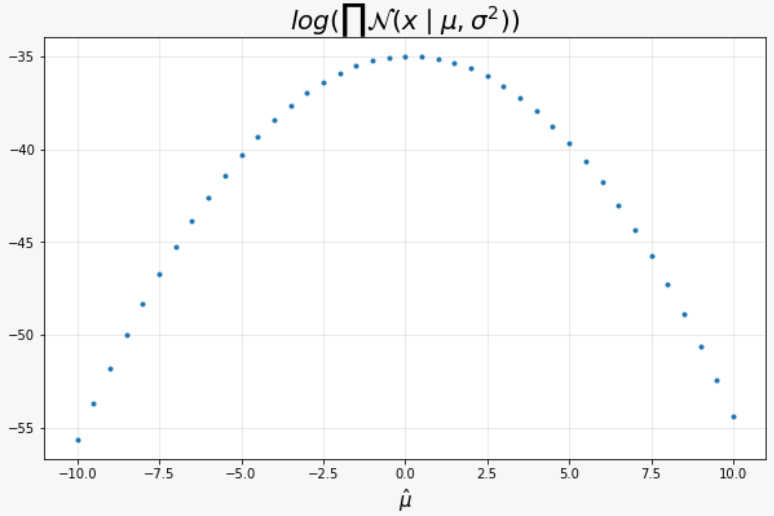
```python
# MLE of Gaussian distribution
# mu

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()
```

Numerical Example for Gaussian



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()
```

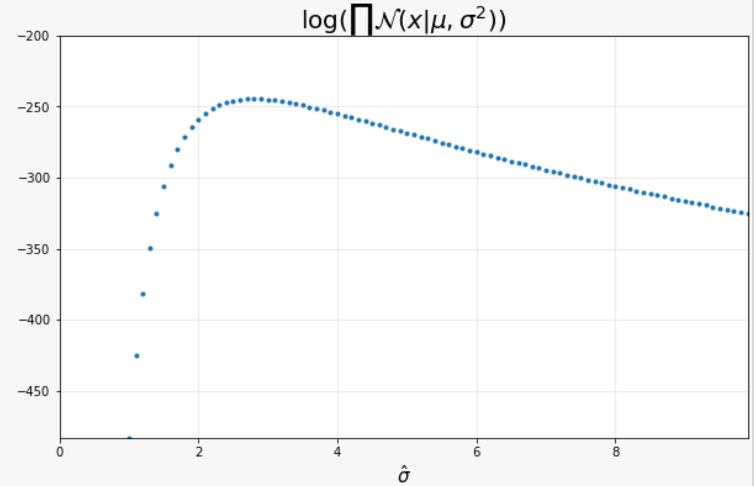


$$\boxed{\mu_{ML} = \frac{1}{m}\sum_{i=1}^{m} x_i}$$

```
0.160329485196
```

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()
```



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

```
2.79684136967
```

Probabilistic Machine Learning

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

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

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

Maximum Likelihood Estimation (MLE)

Linear Regression: A Probabilistic View

- Linear regression model with (Gaussian) normal errors

$$y = \omega^T x + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2)$$
$$y - \omega^T x = \varepsilon \sim \mathcal{N}\left(0, \sigma^2\right)$$

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

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

Linear Regression: A Probabilistic View

$$\ell = -\frac{m}{2}\log 2\pi - m\log\sigma - \frac{1}{2\sigma^2}\sum_{i=1}^{m}\left(y_i - \omega^T x_i\right)^2$$

$$\frac{d\ell}{d\omega} = -2X^T Y + 2X^T X\omega = 0 \implies \omega_{ML} = \left(X^T X\right)^{-1} X^T Y \quad \text{(look familiar ?)}$$

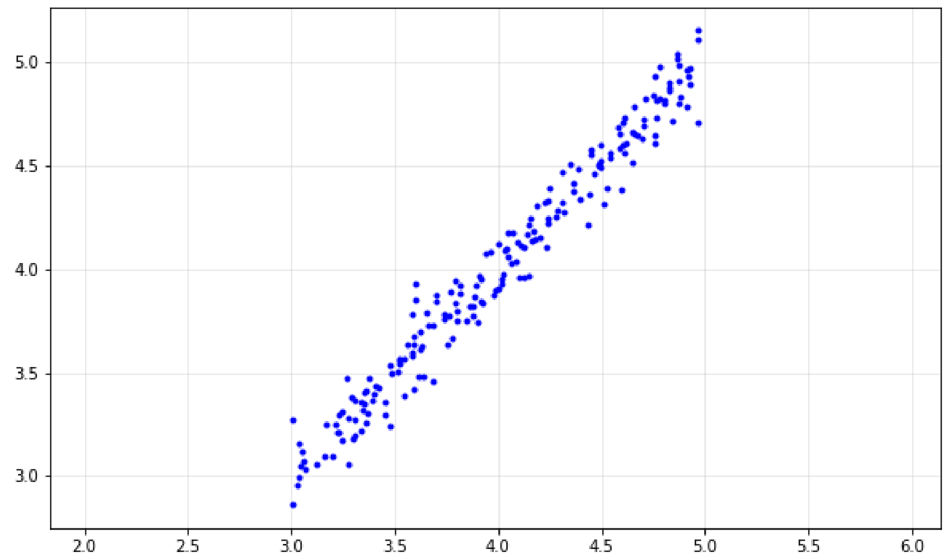
$$\frac{d\ell}{d\sigma} = -\frac{m}{\sigma} + \frac{1}{\sigma^3}\sum_{i=1}^{m}\left(y_i - \omega^T x_i\right)^2 = 0 \implies \sigma_{ML}^2 = \frac{1}{m}\sum_{i=1}^{m}\left(y_i - \omega^T x_i\right)^2$$

- BIG Lesson
  - Same as the least squared optimization

$$\begin{aligned}
\text{loss function} &= \sum_{i=1}^{m}\left(y_i - \omega^T x_i\right)^2 \\
&= \|Y - X\omega\|_2^2 \\
&= (Y - X\omega)^T (Y - X\omega) \\
&= Y^T Y - \omega^T X^T Y - Y^T X\omega + \omega^T X^T X\omega
\end{aligned}$$

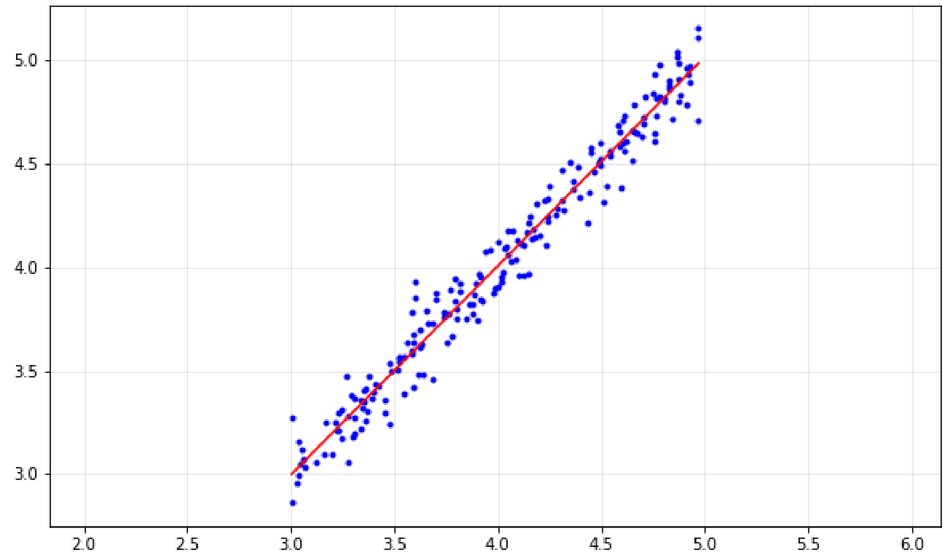
Linear Regression: A Probabilistic View

```
m = 200

a = 1
x = 3 + 2*np.random.uniform(0,1,[m,1])
noise = 0.1*np.random.randn(m,1)

y = a*x + noise;
y = np.asmatrix(y)

plt.figure(figsize=(10, 6))
plt.plot(x, y, 'b.')
plt.axis('equal')
plt.grid(alpha=0.3)
plt.show()
```



Linear Regression: A Probabilistic View

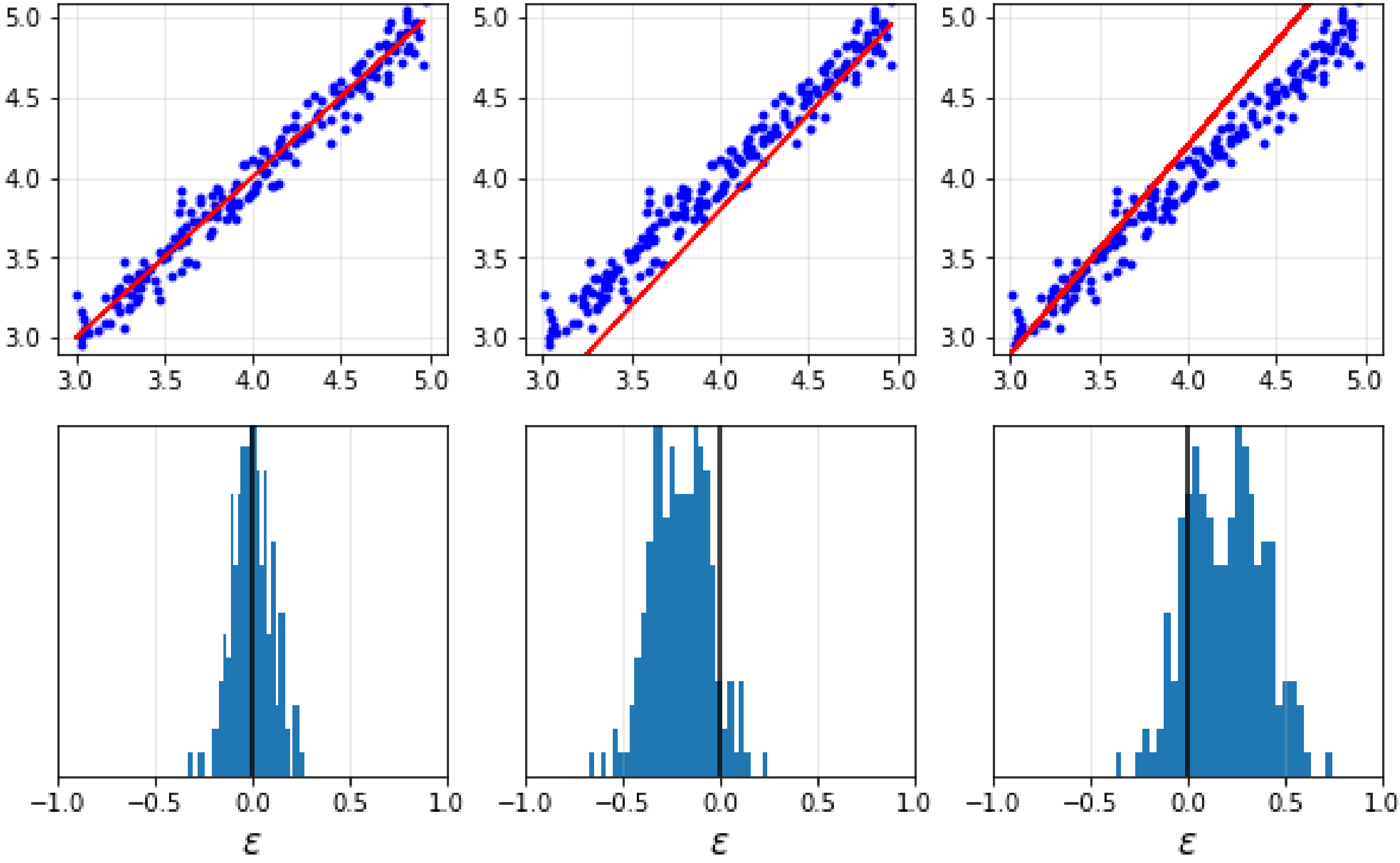
```
# compute theta(1) and theta(2) which are coefficients of y = theta(1)*x + theta(2)
A = np.hstack([np.ones([m, 1]), x])
A = np.asmatrix(A)

theta = (A.T*A).I*A.T*y

# to plot the fitted line
xp = np.linspace(np.min(x), np.max(x))
yp = theta[1,0]*xp + theta[0,0]

plt.figure(figsize=(10, 6))
plt.plot(x, y, 'b.')
plt.plot(xp, yp, 'r')
plt.axis('equal')
plt.grid(alpha=0.3)
plt.show()
```



Linear Regression: A Probabilistic View

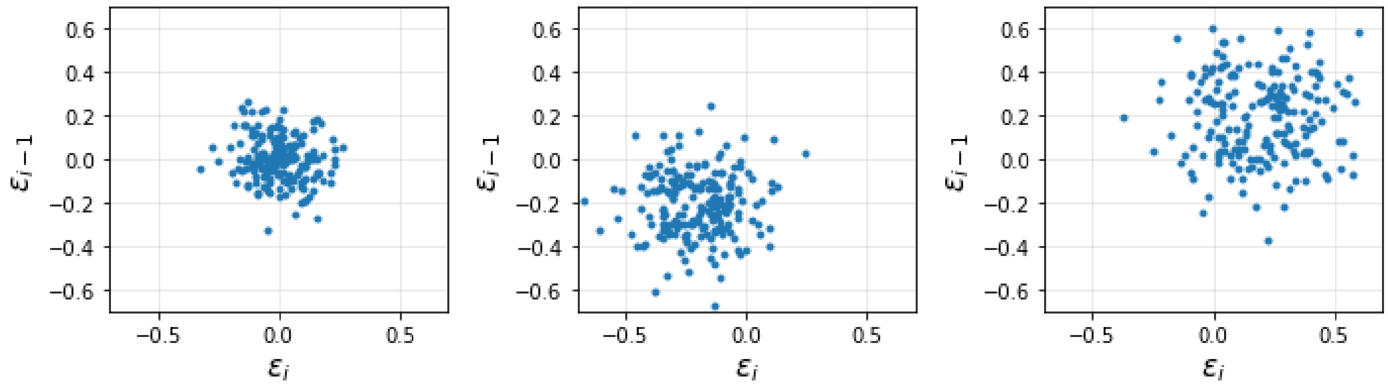
```
yhat0 = theta[1,0]*x + theta[0,0]
err0 = yhat0 - y

yhat1 = 1.2*x - 1
err1 = yhat1 - y

yhat2 = 1.3*x - 1
err2 = yhat2 - y

plt.figure(figsize=(10, 6))
plt.subplot(2,3,1), plt.plot(x,y,'b.',x,yhat0,'r'),
plt.axis([2.9, 5.1, 2.9, 5.1]), plt.grid(alpha=0.3)
plt.subplot(2,3,2), plt.plot(x,y,'b.',x,yhat1,'r'),
plt.axis([2.9, 5.1, 2.9, 5.1]), plt.grid(alpha=0.3)
plt.subplot(2,3,3), plt.plot(x,y,'b.',x,yhat2,'r'),
plt.axis([2.9, 5.1, 2.9, 5.1]), plt.grid(alpha=0.3)
plt.subplot(2,3,4), plt.hist(err0,31), plt.axvline(0, color='k'),
plt.xlabel(r'$\epsilon$', fontsize=15),
plt.yticks([]), plt.axis([-1, 1, 0, 15]), plt.grid(alpha=0.3)
plt.subplot(2,3,5), plt.hist(err1,31), plt.axvline(0, color='k'),
plt.xlabel(r'$\epsilon$', fontsize=15),
plt.yticks([]), plt.axis([-1, 1, 0, 15]), plt.grid(alpha=0.3)
plt.subplot(2,3,6), plt.hist(err2,31), plt.axvline(0, color='k'),
plt.xlabel(r'$\epsilon$', fontsize=15),
plt.yticks([]), plt.axis([-1, 1, 0, 15]), plt.grid(alpha=0.3)
plt.show()
```

Linear Regression: A Probabilistic View



Linear Regression: A Probabilistic View

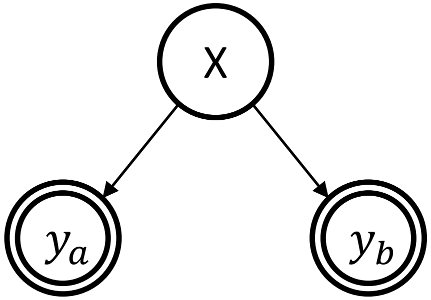
```
a0x = err0[1:]
a0y = err0[0:-1]

a1x = err1[1:]
a1y = err1[0:-1]

a2x = err2[1:]
a2y = err2[0:-1]

plt.figure(figsize=(10, 3))
plt.subplot(1, 3, 1), plt.plot(a0x, a0y, '.'),
plt.axis([-0.7, 0.7, -0.7, 0.7]), plt.grid(alpha=0.3)
plt.xlabel(r'$\epsilon_i$', fontsize=15), plt.ylabel(r'$\epsilon_{i-1}$', fontsize=15)
plt.subplot(1, 3, 2), plt.plot(a1x, a1y, '.'),
plt.axis([-0.7, 0.7, -0.7, 0.7]), plt.grid(alpha=0.3)
plt.xlabel(r'$\epsilon_i$', fontsize=15), plt.ylabel(r'$\epsilon_{i-1}$', fontsize=15)
plt.subplot(1, 3, 3), plt.plot(a2x, a2y, '.'),
plt.axis([-0.7, 0.7, -0.7, 0.7]), plt.grid(alpha=0.3)
plt.xlabel(r'$\epsilon_i$', fontsize=15), plt.ylabel(r'$\epsilon_{i-1}$', fontsize=15)
plt.tight_layout()
plt.show()
```



Maximum a Posterior (MAP)

Data Fusion with Uncertainties

- [Learning Theory (Reza Shadmehr, Johns Hopkins University)]
  - youtube [link]



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

- In a matrix form

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

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

Data Fusion with Uncertainties

- Find $\hat{x}_{ML}$

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

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

$$\implies \frac{d\ell}{dx} = 0 = -2C^T R^{-1} y + 2C^T R^{-1} C x$$

$$\therefore \; x_{ML} = \left(C^T R^{-1} C\right)^{-1} C^T R^{-1} y$$

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

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

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

Data Fusion with Uncertainties

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

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

$$\operatorname{var}\left(\hat{x}_{ML}\right) = \left(\left(C^T R^{-1} C\right)^{-1} C^T R^{-1}\right) \cdot \operatorname{var}(y) \cdot \left(\left(C^T R^{-1} C\right)^{-1} C^T R^{-1}\right)^T$$

$$= \left(\left(C^T R^{-1} C\right)^{-1} C^T R^{-1}\right) \cdot R \cdot \left(\left(C^T R^{-1} C\right)^{-1} C^T R^{-1}\right)^T$$

$$= \left(C^T R^{-1} C\right)^{-1} C^T \cdot \left(R^{-1}\right)^T C \left(\left(C^T R^{-1} C\right)^{-1}\right)^T$$

$$= \underbrace{\left(C^T R^{-1} C\right)^{-1}}_{} \underbrace{C^T R^{-1} C}_{} \left(\left(C^T R^{-1} C\right)^{-1}\right)^T = \left(C^T R^{-1} C\right)^{-1}$$

$$= \frac{1}{\frac{1}{\sigma_a^2} + \frac{1}{\sigma_b^2}} \leq \sigma_a^2,\ \sigma_b^2$$

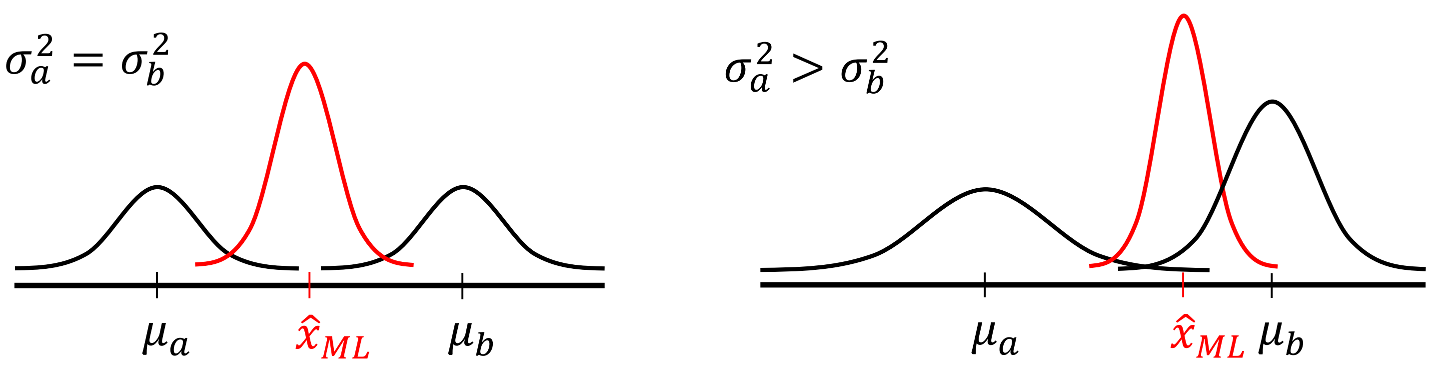
Data Fusion with Less Uncertainties

- Summary

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

$$\mathrm{var}\left(\hat{x}_{ML}\right) = \frac{1}{\frac{1}{\sigma_a^2} + \frac{1}{\sigma_b^2}} \leq \quad \sigma_a^2, \ \sigma_b^2$$

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



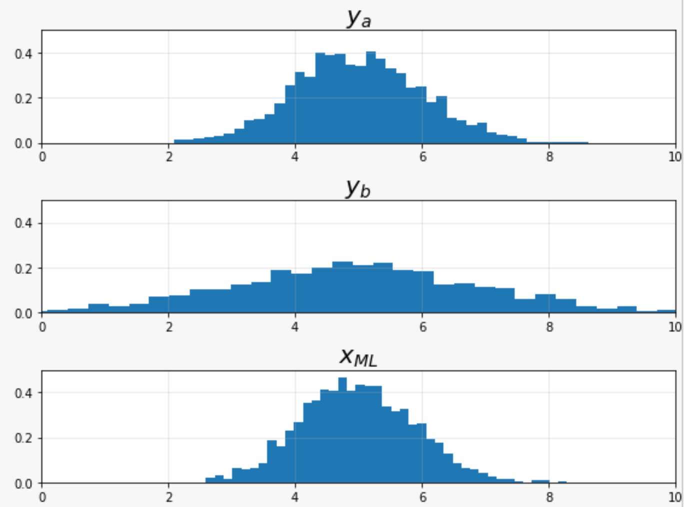
1D Examples

- Example of Two Rulers

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

Data Fusion with 1D Example

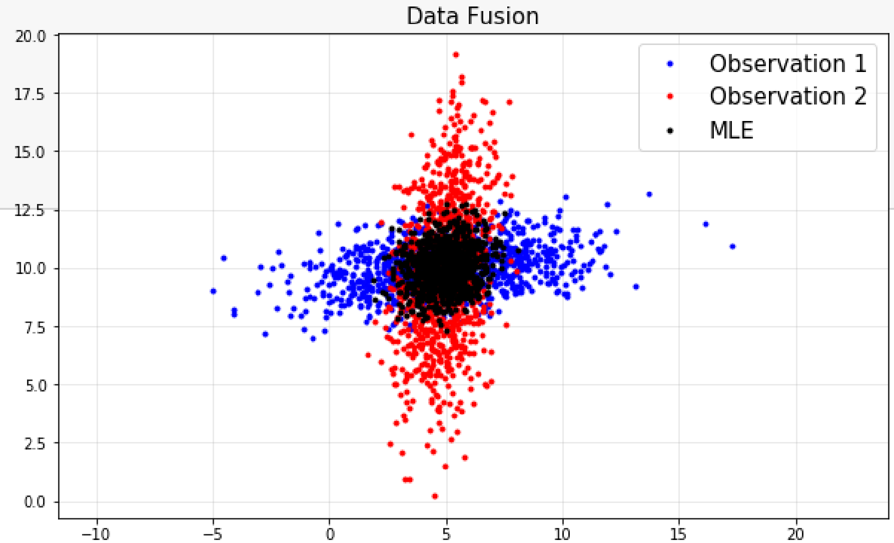
```
x = 5        # true state (length in this example)
a = 1        # sigma of a
b = 2       # sigma of b

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)
```




```
plt.figure(figsize=(8, 6))
plt.subplot(3, 1, 1), plt.hist(YA, 41, normed=True), plt.axis([0, 10, 0, 0.5]),
plt.title(r'$y_a$', fontsize=20), plt.grid(alpha=0.3)
plt.subplot(3, 1, 2), plt.hist(YB, 41, normed=True), plt.axis([0, 10, 0, 0.5]),
plt.title(r'$y_b$', fontsize=20), plt.grid(alpha=0.3)
plt.subplot(3, 1, 3), plt.hist(XML, 41, normed=True), plt.axis([0, 10, 0, 0.5]),
plt.title(r'$x_{ML}$', fontsize=20), plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```

Data Fusion with 2D Example

```
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)

YA = np.vstack(YA)
YB = np.vstack(YB)
XML = np.vstack(XML)
```



Maximum-a-Posterior Estimation (MAP)

- Choose θ that maximizes the posterior probability of θ (*i.e.* probability in the light of the observed data)
- Posterior probability of θ is given by the Bayes Rule

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

  - $P(\theta)$: Prior probability of θ (without having seen any data)
  - $P(D|\theta)$: Likelihood
  - $P(D)$: Probability of the data (independent of θ)

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

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

Maximum-a-Posterior 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 $D = \{d_1, d_2, \cdots, d_m\}$

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

- same as MLE except the extra log-prior-distribution term
- MAP allows incorporating our prior knowledge about θ in its estimation

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

MAP for mean of a univariate Gaussian

- 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)
  - Observations $D = \{d_1, d_2, \cdots, d_m\}$: conditionally independent given θ

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

  - Joint Probability

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

MAP for mean of 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 mean of a univariate Gaussian

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

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

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

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

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

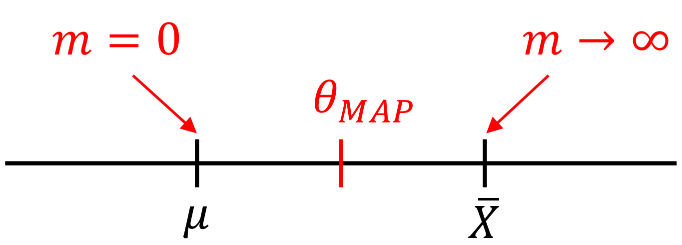
MAP for mean of a univariate Gaussian

- ML interpretation:

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

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

- BIG Lesson: a prior acts as a data



- Note: prior knowledge
  - Education
  - Get older
  - School ranking

MAP for mean of a univariate Gaussian

Example) Experiment in class

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

MAP Python code

- 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 Python code

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

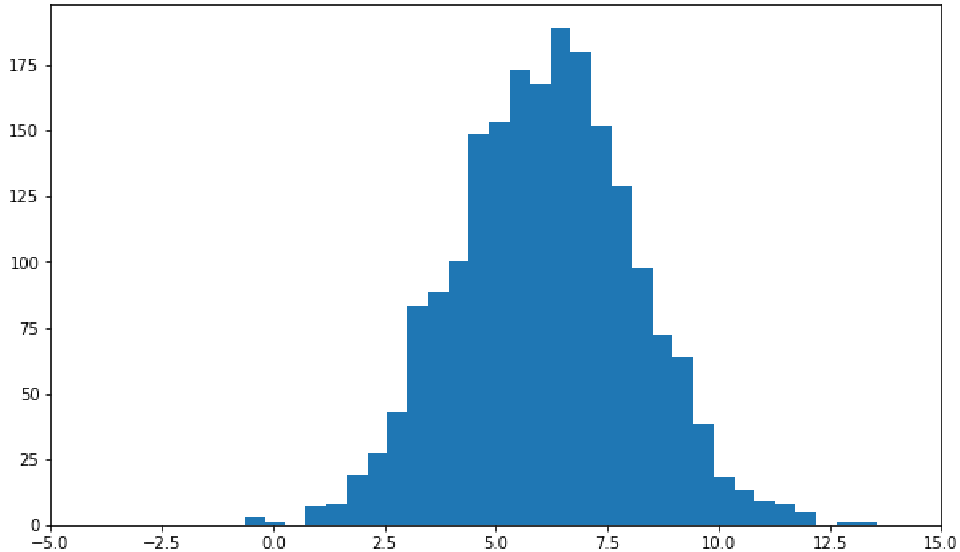
```
# MAP

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 Python code

```
# 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()
```

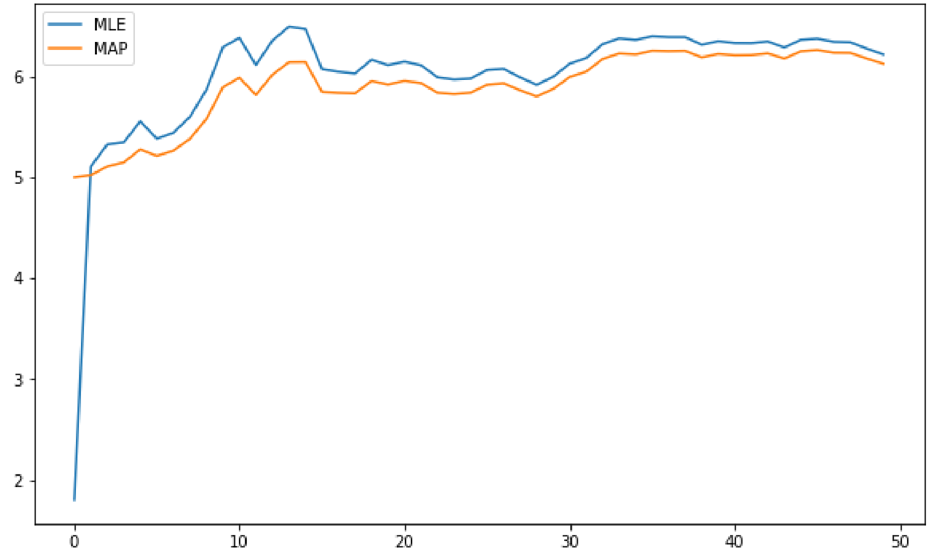
```
theta = 6.1839
```



MAP Python code

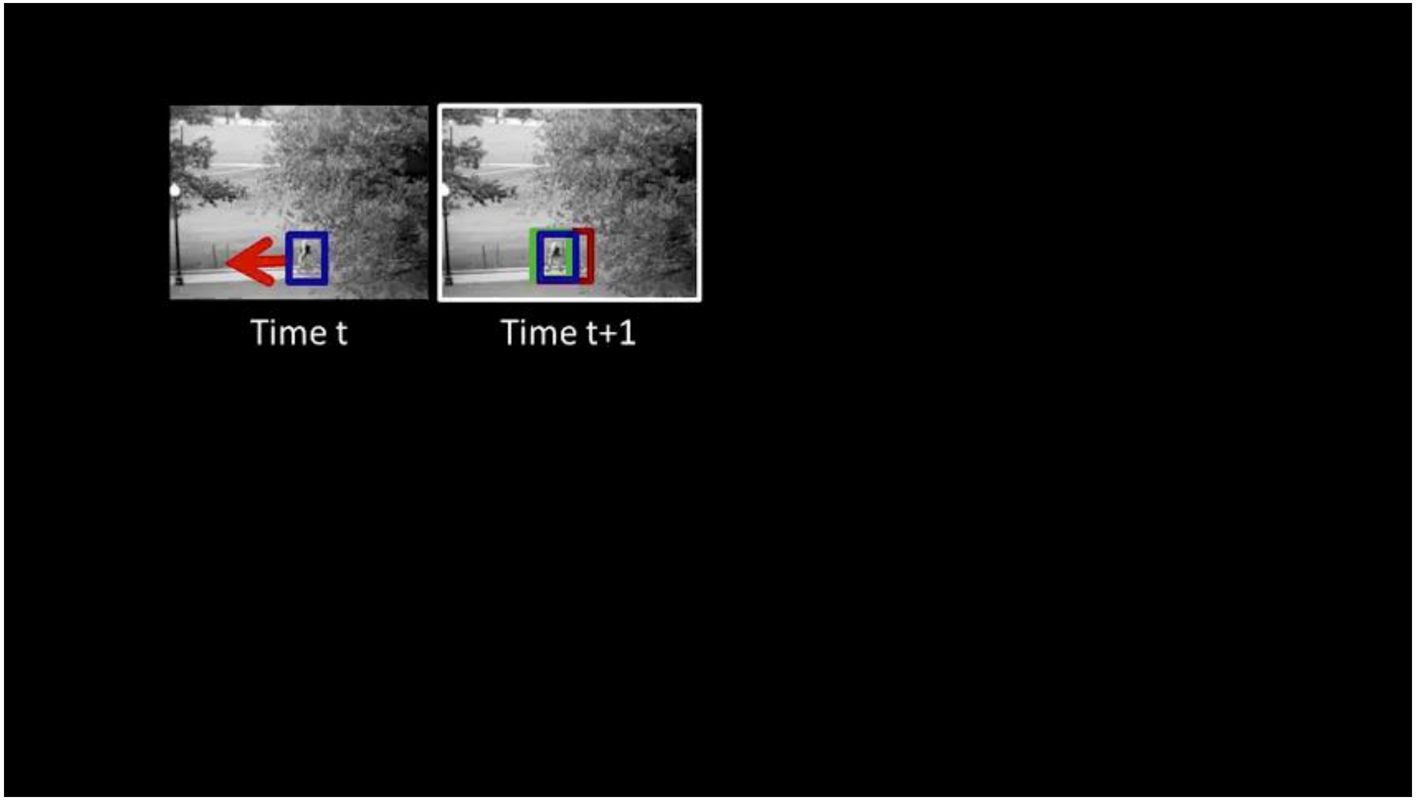
```
n = 50
XMLE = []
XMAP = []

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
```

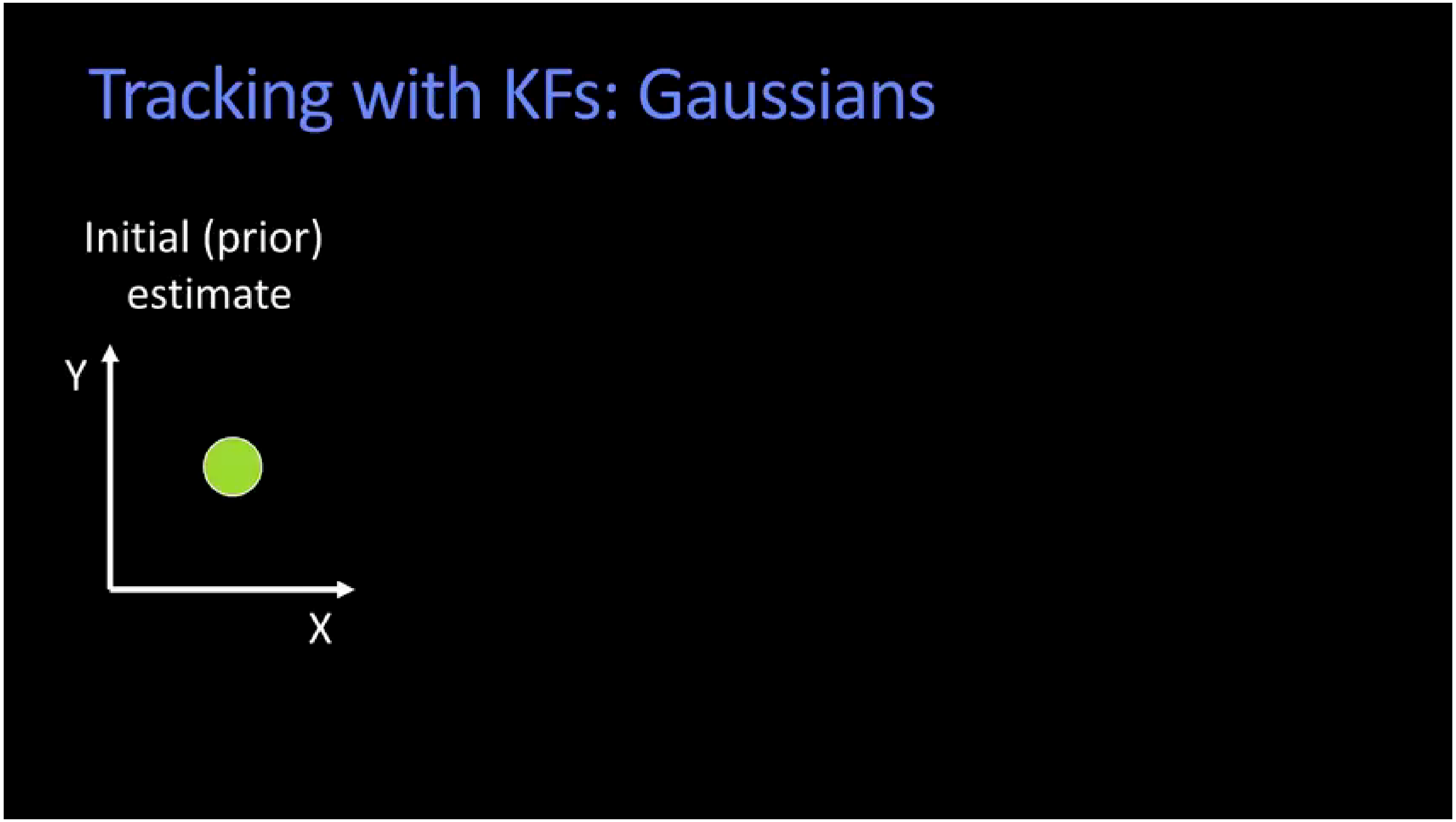


Object Tracking in Computer Vision

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



Object Tracking in Computer Vision



Kernel Density Estimation

- *non-parametric* estimate of density
- Lecture: Learning Theory (Reza Shadmehr, Johns Hopkins University)

Kernel Density Estimation

```
m = 10
mu = 0
sigma = 5

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

X = []

for i in range(m):
    X.append(norm.pdf(xp,x[i,0],sigma))

X = np.array(X).T
Xnorm = np.sum(X,1)/m

plt.figure(figsize=(10,6))
plt.plot(x,y0,'kx')
plt.plot(xp,X,'b--')
plt.plot(xp,Xnorm,'r',linewidth=5)
plt.show()
```