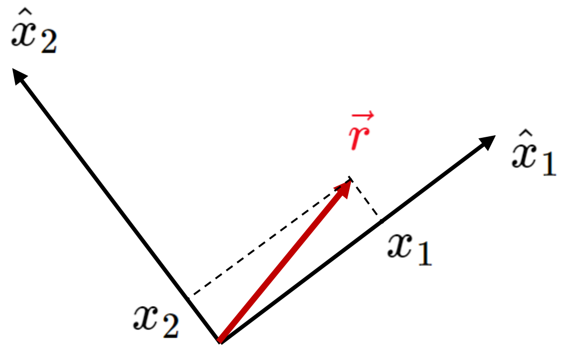
Ellipse and Gaussian Distribution

Prof. Seungchul Lee
Industrial AI Lab.

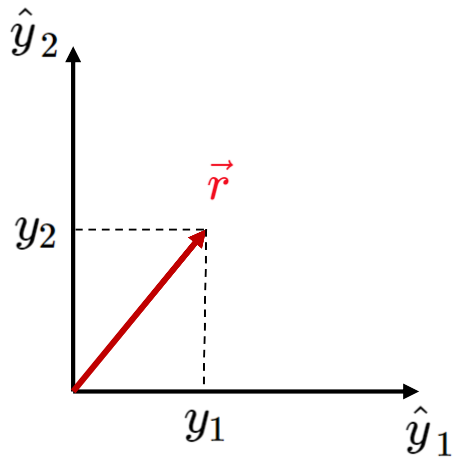
Coordinates

Coordinates with Basis

basis $\{\hat{x}_1\ \hat{x}_2\}$



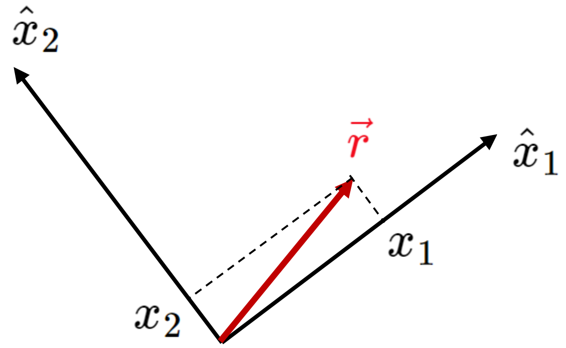
basis $\{\hat{y}_1\ \hat{y}_2\}$

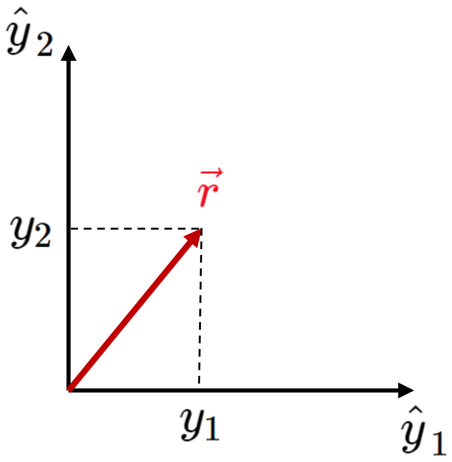


$$\vec{r}_X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} : \text{coordinate of } \vec{r} \text{ in basis } \{\hat{x}_1\ \hat{x}_2\}$$

$$\vec{r}_Y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} : \text{coordinate of } \vec{r} \text{ in basis } \{\hat{y}_1\ \hat{y}_2\}$$

Coordinate Transformation





$\vec{r}_X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$: coordinate of $\vec{r}$ in basis $\{\hat{x}_1 \ \hat{x}_2\}$

$\vec{r}_Y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$: coordinate of $\vec{r}$ in basis $\{\hat{y}_1 \ \hat{y}_2\}$

$$\vec{r} = x_1\hat{x}_1 + x_2\hat{x}_2 = y_1\hat{y}_1 + y_2\hat{y}_2$$

$$\begin{bmatrix} \hat{x}_1 & \hat{x}_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \hat{y}_1 & \hat{y}_2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$U \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \quad (U = \begin{bmatrix} \hat{x}_1 & \hat{x}_2 \end{bmatrix}, I = \begin{bmatrix} \hat{y}_1 & \hat{y}_2 \end{bmatrix})$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \xrightarrow{U} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

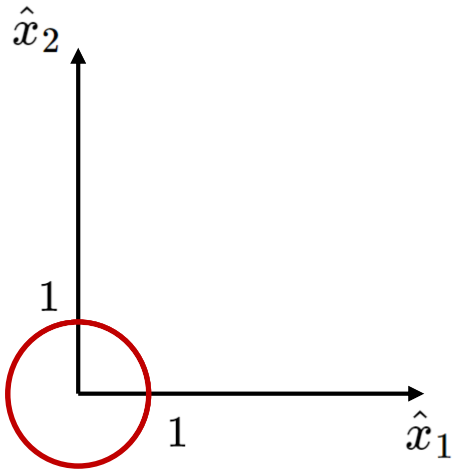
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = U^{-1} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = U^T \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \xrightarrow{U^T} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Equation of an Ellipse

Equation of an Ellipse

- Unit circle

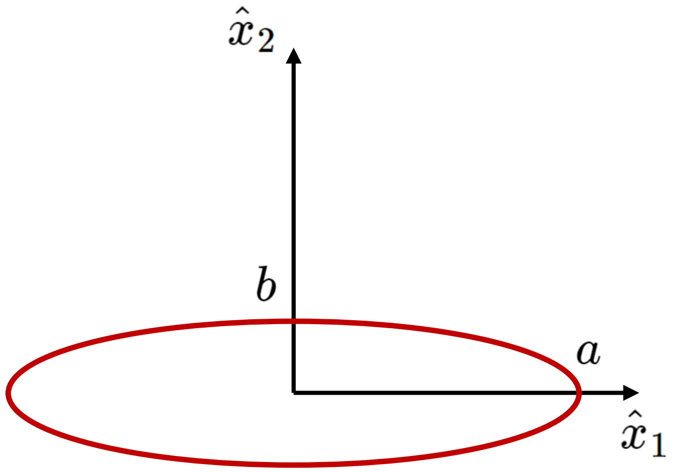


$$x_1^2 + x_2^2 = 1 \implies$$

$$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 1$$

Equation of an Ellipse

- Independent ellipse



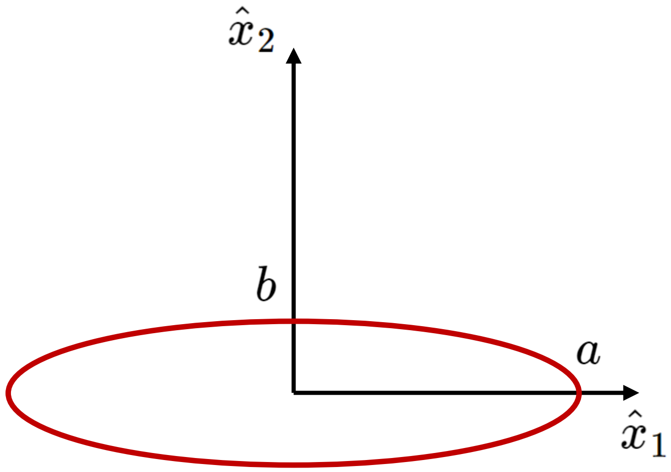
$$\frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} = 1 \implies \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} \frac{1}{a^2} & 0 \\ 0 & \frac{1}{b^2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 1$$

$$\implies \begin{bmatrix} x_1 & x_2 \end{bmatrix} \Sigma_x^{-1} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 1$$

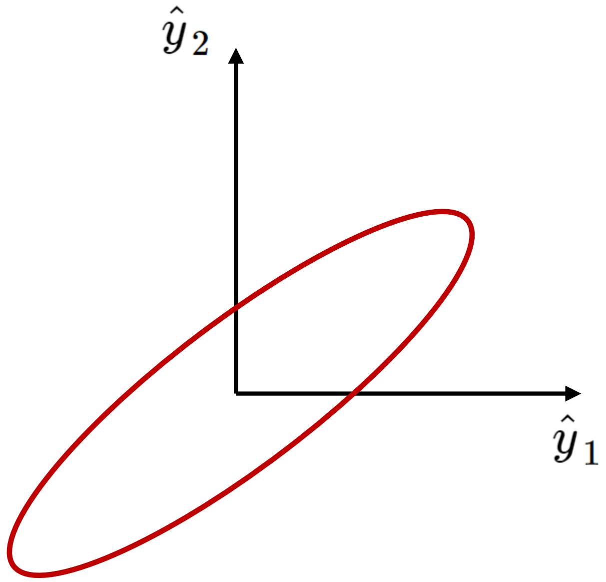
$$\text{where } \Sigma_x^{-1} = \begin{bmatrix} \frac{1}{a^2} & 0 \\ 0 & \frac{1}{b^2} \end{bmatrix}, \ \Sigma_x = \begin{bmatrix} a^2 & 0 \\ 0 & b^2 \end{bmatrix}$$

Equation of an Ellipse

- Dependent ellipse (Rotated ellipse)

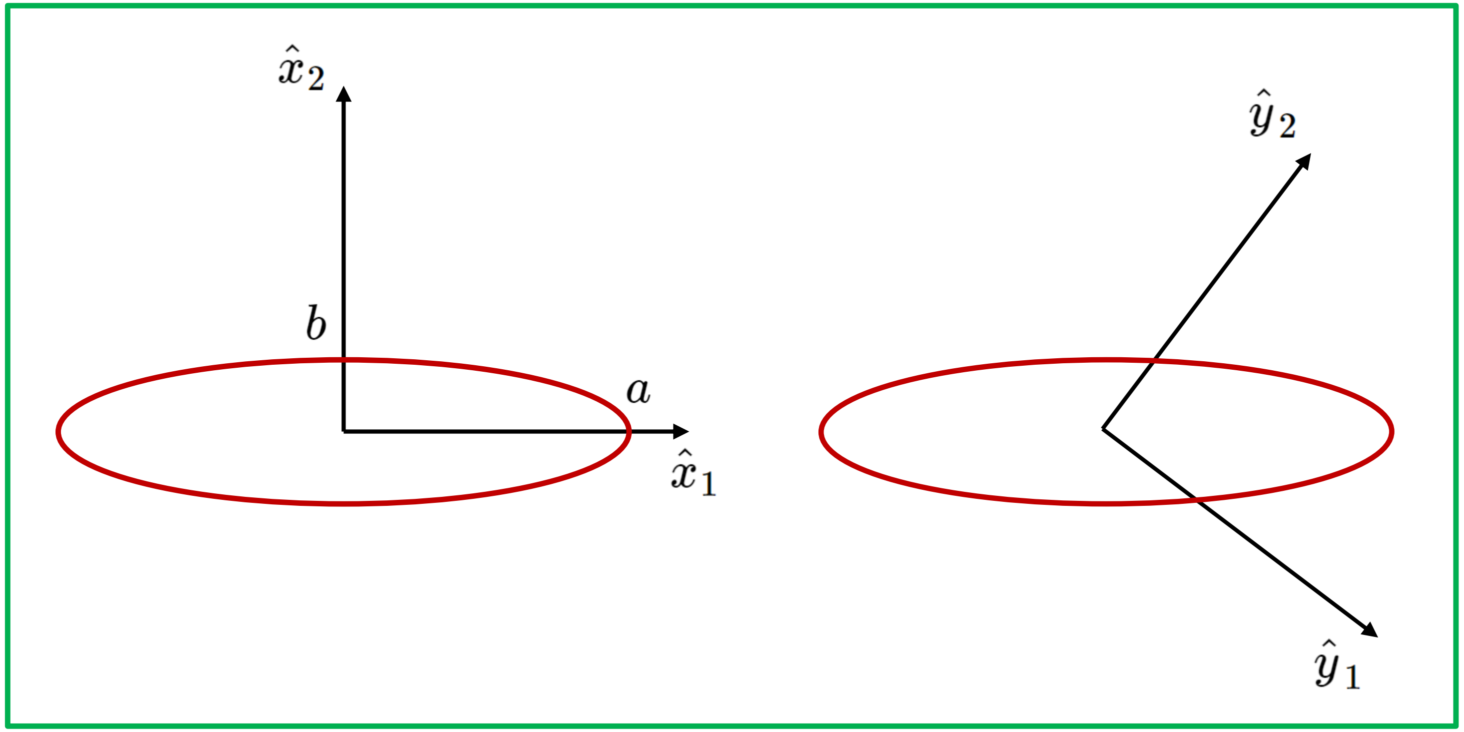


To find the equation of dependent ellipse

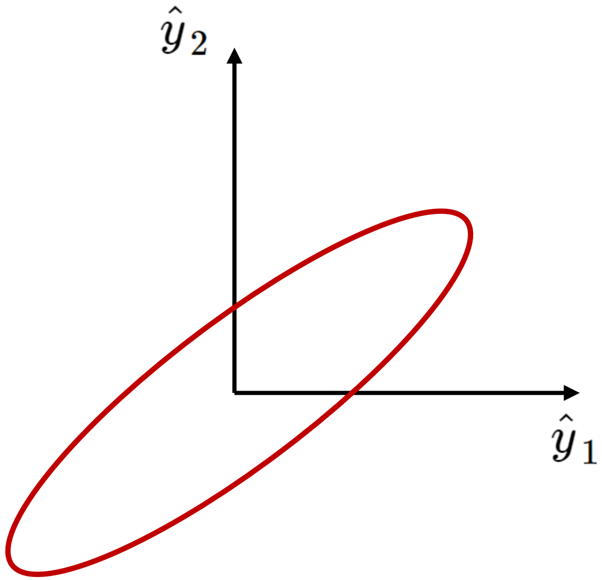


Equation of an Ellipse

- Dependent ellipse (Rotated ellipse)



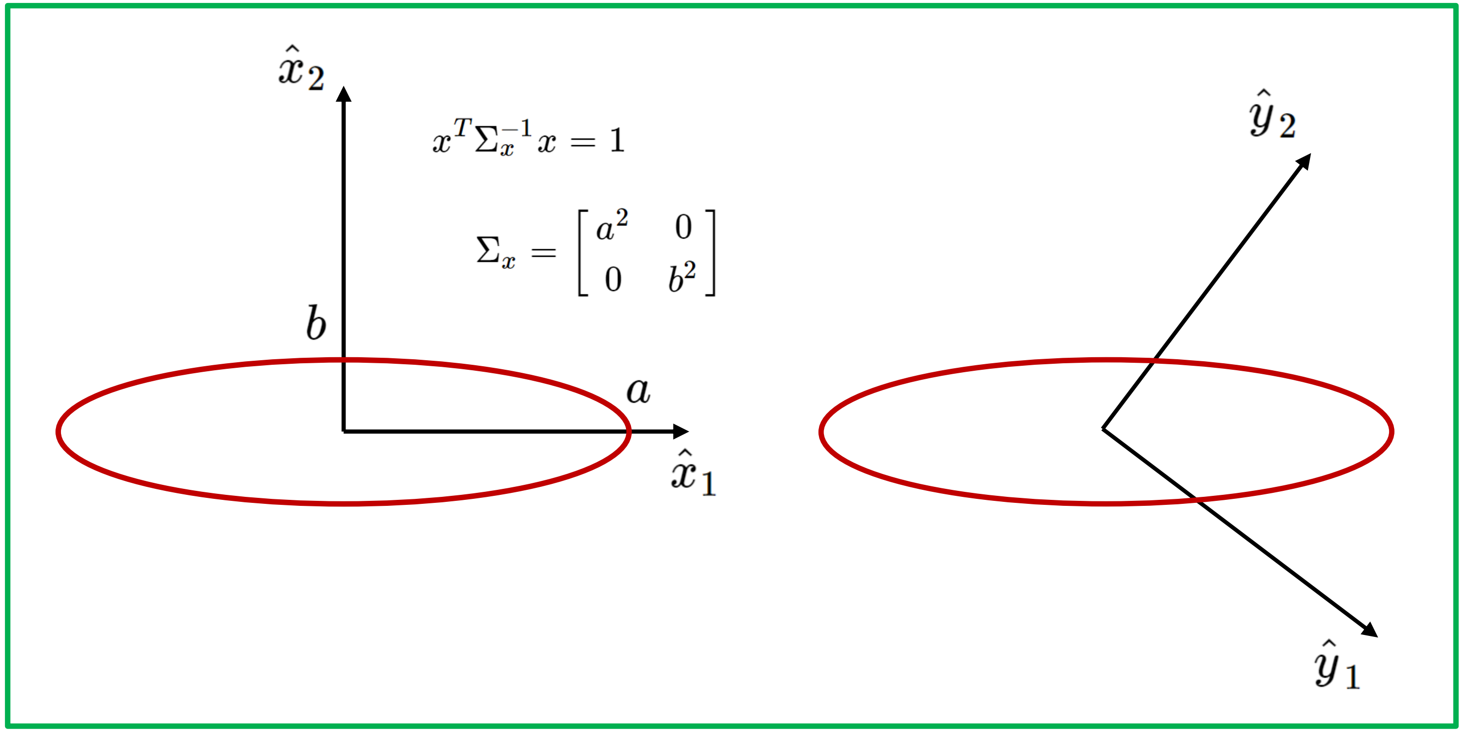
To find the equation of dependent ellipse



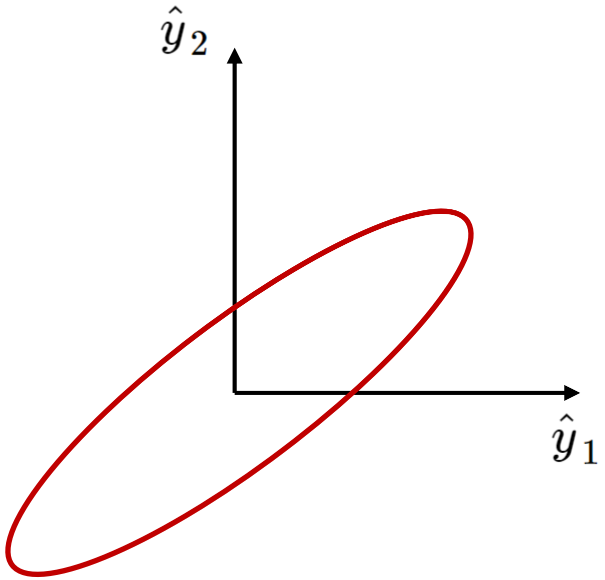
- Coordinate changes

Equation of an Ellipse

- Dependent ellipse (Rotated ellipse)



To find the equation of dependent ellipse



- Coordinate changes

$$U \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \quad (U = \begin{bmatrix} \hat{x}_1 & \hat{x}_2 \end{bmatrix}, I = \begin{bmatrix} \hat{y}_1 & \hat{y}_2 \end{bmatrix})$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = U^{-1} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = U^T \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$x = U^T y$$
$$Ux = y$$

POSTECH

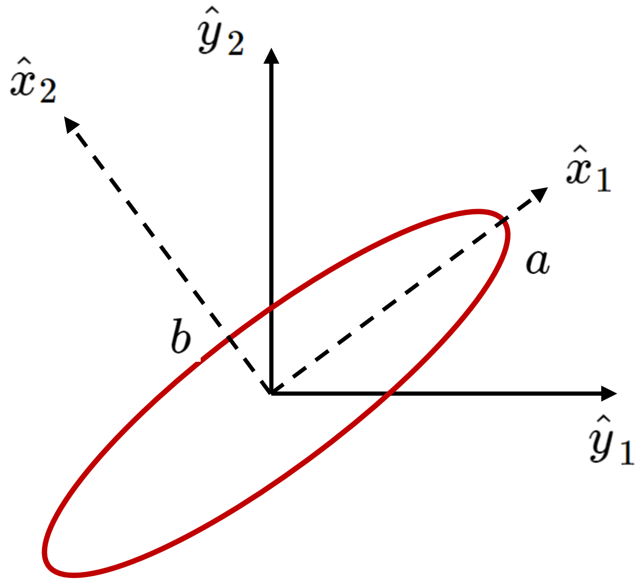
Equation of an Ellipse

- Dependent ellipse (Rotated ellipse)

To find the equation of dependent ellipse

$$x^T \Sigma_x^{-1} x = 1 \quad \text{and} \quad \Sigma_x = \begin{bmatrix} a^2 & 0 \\ 0 & b^2 \end{bmatrix}$$

$$x = U^T y$$
$$Ux = y$$



$$y^T \Sigma_y^{-1} y = 1 \quad \text{and} \quad \Sigma_y = ?$$

$$\Longrightarrow \quad x^T \Sigma_x^{-1} x = y^T U \Sigma_x^{-1} U^T y = 1 \quad (\Sigma_y^{-1} : \text{similar matrix to } \Sigma_x^{-1})$$

$$\therefore \quad \Sigma_y^{-1} = U \Sigma_x^{-1} U^T \quad \text{or}$$

$$\Sigma_y = U \Sigma_x U^T$$

Equation of an Ellipse

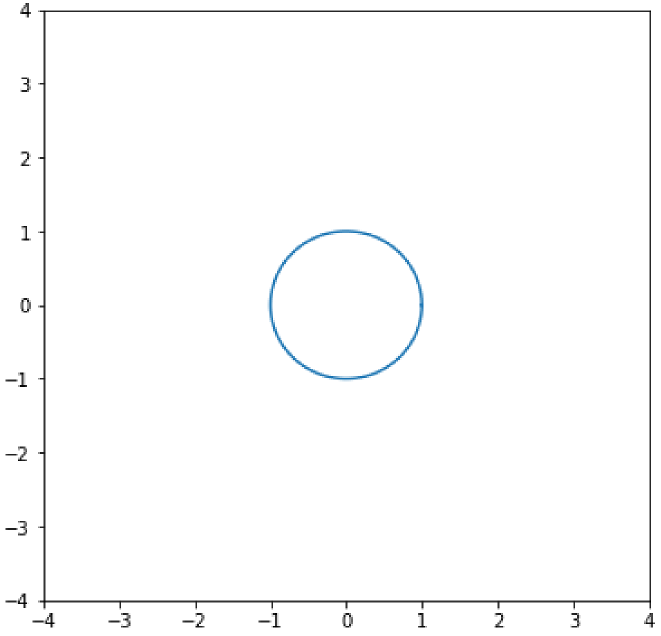
$$U = [\hat{x}_1 \; \hat{x}_2] = \begin{bmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix}$$

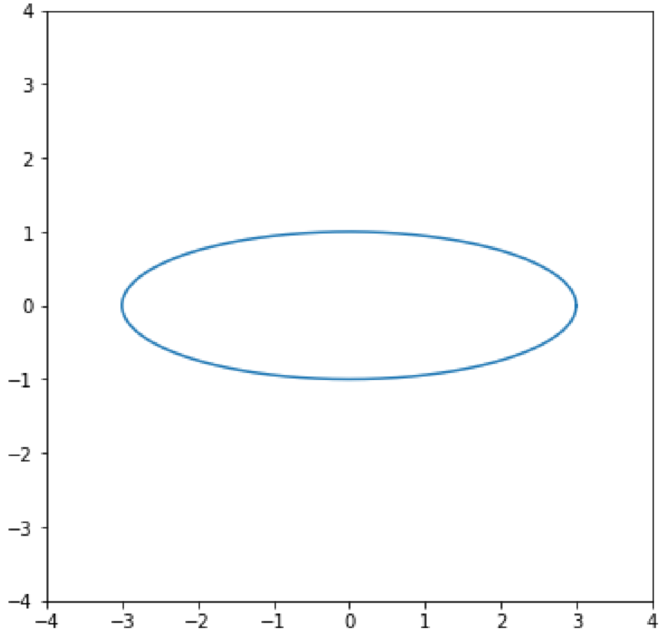
```
theta = np.arange(0,2*np.pi,0.01)
x1 = np.cos(theta)
x2 = np.sin(theta)
```

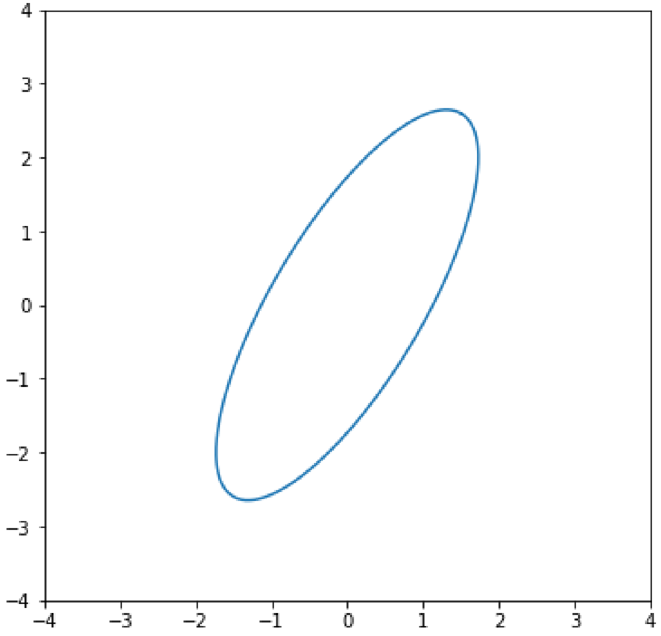
```
x1 = 3*np.cos(theta);
x2 = np.sin(theta);
```

```
U = np.matrix([[1/2, -np.sqrt(3)/2],
               [np.sqrt(3)/2, 1/2]])

x = np.matrix([x1, x2])

y = U*x
print("U = \n", U)
```

```
[[ 0.5        -0.8660254]
 [ 0.8660254  0.5       ]]
```







Equation of an Ellipse

```
Sx = np.matrix([[9, 0],
               [0, 1]])

Sy = U*Sx*U.T

print ("Sx = \n", Sx, "\n")
print ("Sy = \n", Sy)
```

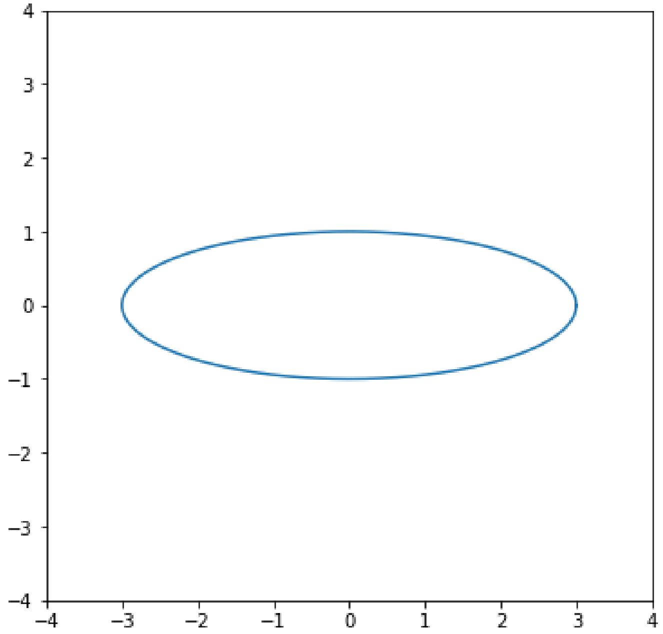
```
Sx =
 [[9 0]
 [0 1]]

Sy =
 [[3.         3.46410162]
 [3.46410162 7.        ]]
```

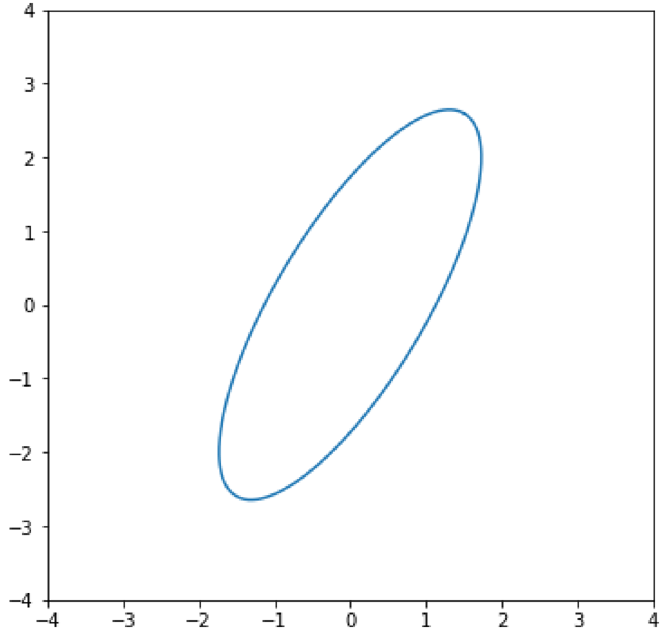
$$x^T \Sigma_x^{-1} x = 1 \quad \text{and} \quad \Sigma_x = \begin{bmatrix} a^2 & 0 \\ 0 & b^2 \end{bmatrix}$$

$$\Sigma_y^{-1} = U \Sigma_x^{-1} U^T \quad \text{or}$$

$$U = [\hat{x}_1 \ \hat{x}_2] = \begin{bmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix}$$

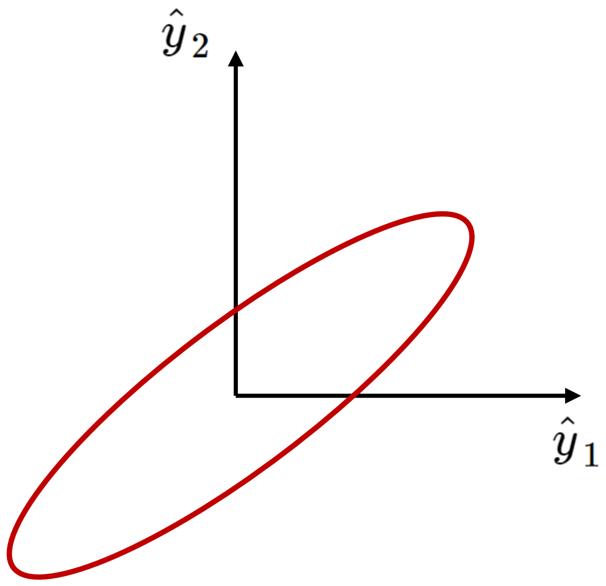
$$\Sigma_y = U \Sigma_x U^T$$





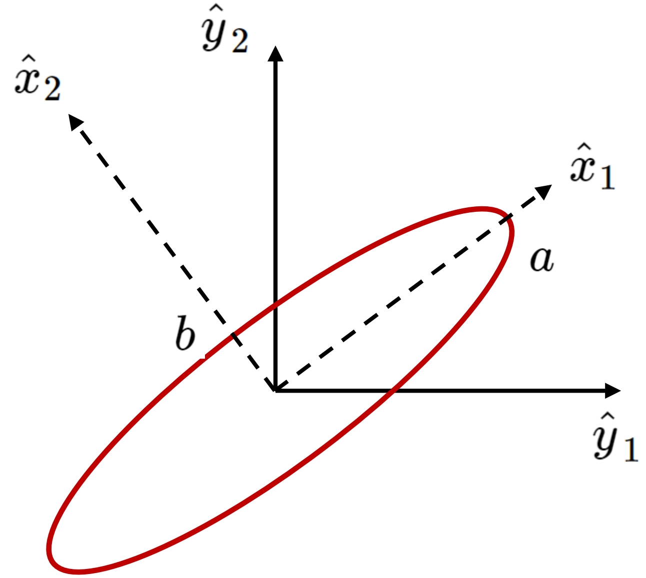
Question (Reverse Problem)

- Given Σ_y^{-1} (or Σ_y), how to find a (major axis) and b (minor axis) or
- How to find the Σ_x or
- How to find the proper matrix U



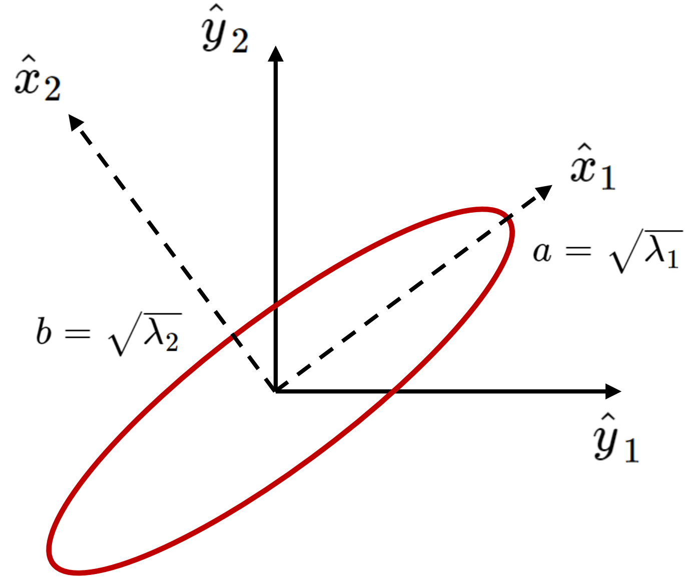
Question (Reverse Problem)

- Given Σ_y^{-1} (or Σ_y),
  - How to find a (major axis) and b (minor axis) or
  - How to find the Σ_x or
  - How to find the proper matrix U



Question (Reverse Problem)

- Given Σ_y^{-1} (or Σ_y),
  - How to find a (major axis) and b (minor axis) or
  - How to find the Σ_x or
  - How to find the proper matrix U

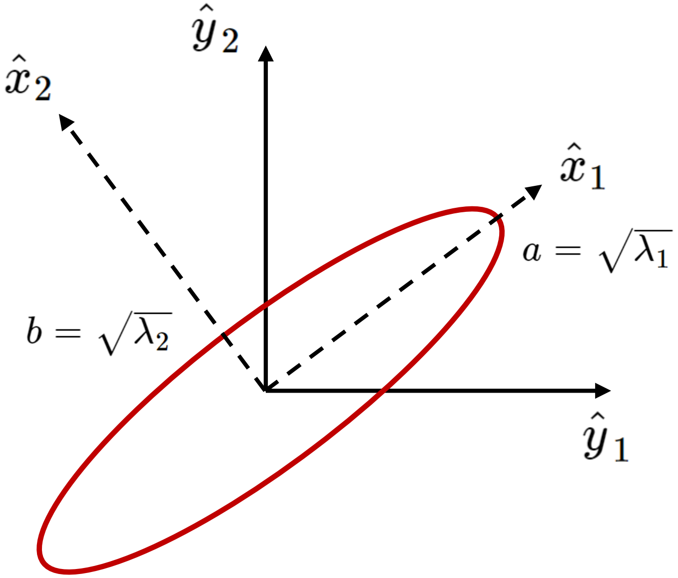


- Eigenvectors of Σ

$$A = S\Lambda S^T \qquad \text{where } S = [v_1 \; v_2] \text{ eigenvector of } A, \text{ and } \Lambda = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

$$\text{here, } \Sigma_y = U\Sigma_x U^T = U\Lambda U^T \qquad \text{where } U = [\, \hat{x}_1 \quad \hat{x}_2 \,] \text{ eigenvector of } \Sigma_y, \text{ and } \Lambda = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} = \begin{bmatrix} a^2 & 0 \\ 0 & b^2 \end{bmatrix}$$

Question (Reverse Problem)



$$\text{eigen-analysis} \begin{cases} \Sigma_y \hat{x}_1 = \lambda_1 \hat{x}_1 \\ \Sigma_y \hat{x}_2 = \lambda_2 \hat{x}_2 \end{cases} \Longrightarrow \Sigma_y \underbrace{\begin{bmatrix} \hat{x}_1 & \hat{x}_2 \end{bmatrix}}_{U} = \underbrace{\begin{bmatrix} \hat{x}_1 & \hat{x}_2 \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}}_{\Lambda}$$

$$\Sigma_y U = U\Lambda$$
$$\Sigma_y = U\Lambda U^T = U\Sigma_x U^T$$

$$x = U^T y$$

$$a = \sqrt{\lambda_1}$$
$$b = \sqrt{\lambda_2}$$

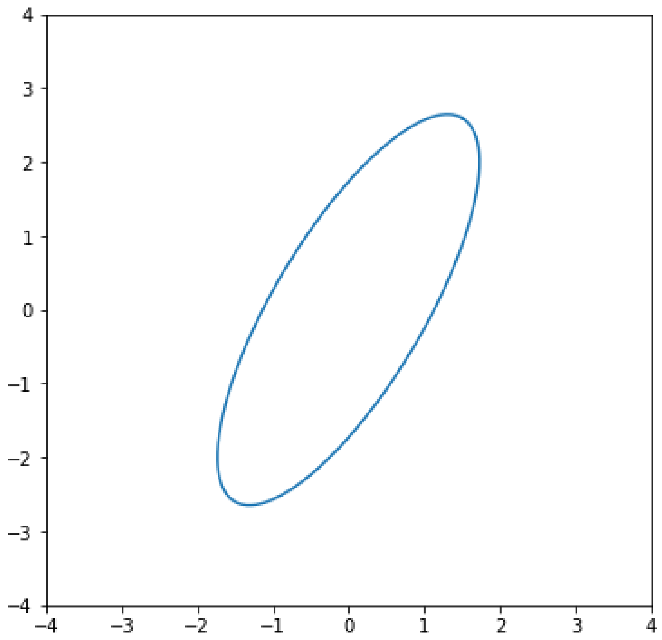
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = U^T \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$\text{major axis} = \hat{x}_1$$
$$\text{minor axis} = \hat{x}_2$$

Question (Reverse Problem)

```
D, U = np.linalg.eig(Sy)

idx = np.argsort(-D)
D = D[idx]
U = U[:,idx]

print ("D = \n", np.diag(D))
print ("U = \n", U)
```

```
D =
 [[9. 0.]
 [0. 1.]]
U =
 [[-0.5        -0.8660254]
 [-0.8660254  0.5       ]]
```

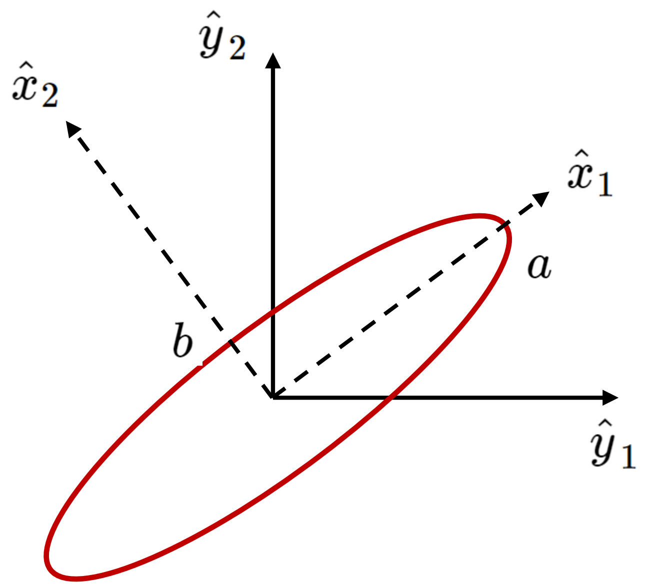


$$U = [\hat{x}_1 \ \hat{x}_2] = \begin{bmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix}$$

Summary

$$x = U^T y$$

$$U = \begin{bmatrix} \hat{x}_1 & \hat{x}_2 \end{bmatrix}$$



- Independent ellipse in $\{\hat{x}_1, \hat{x}_2\}$
- Dependent ellipse in $\{\hat{y}_1, \hat{y}_2\}$
- Decouple
  - Diagonalize
  - Eigen-analysis

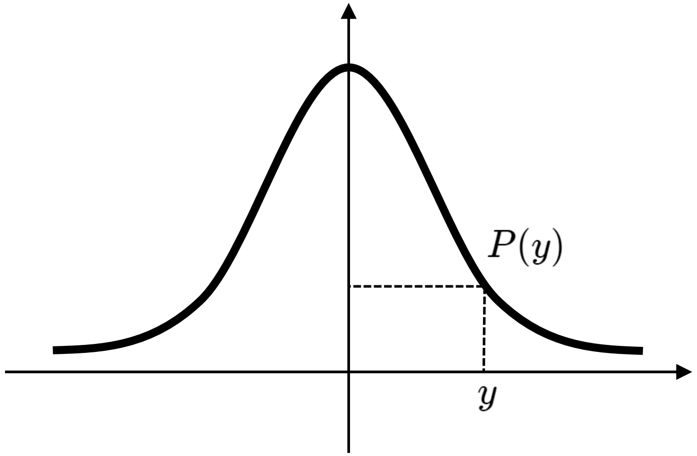
Gaussian Distribution

Standard Univariate Normal Distribution

- It is a continuous pdf, but
  - Parameterized by only two terms, $\mu = 0$ and $\sigma = 1$
  - This is a big advantage of using Gaussian

$$P_Y(Y = y) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}y^2\right)$$

$$\frac{1}{2}y^2 = \text{const} \implies \text{prob. contour}$$



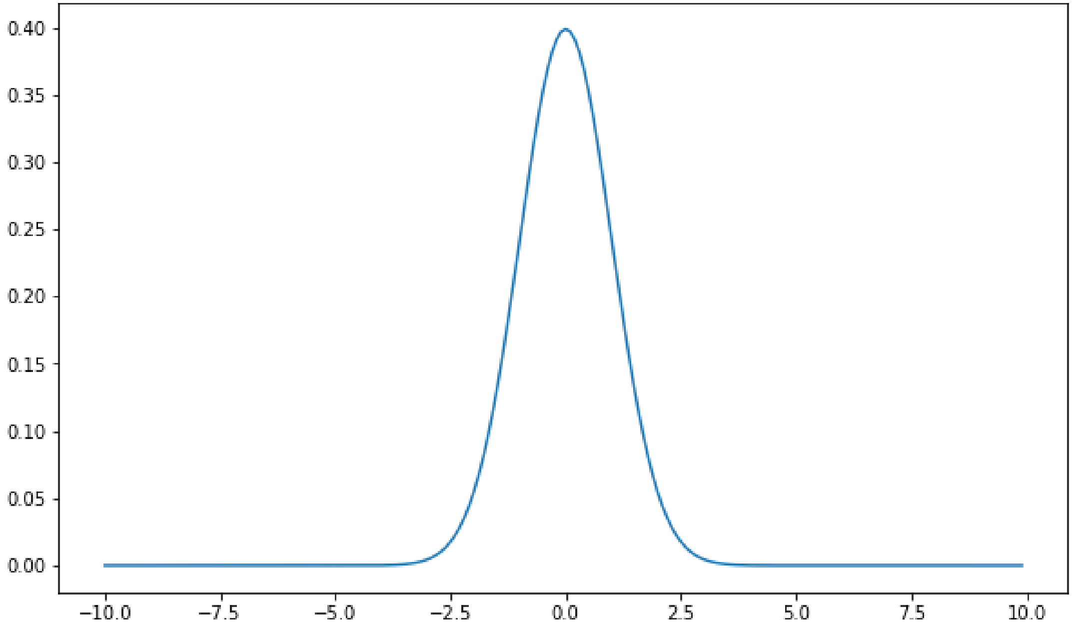
POSTECH

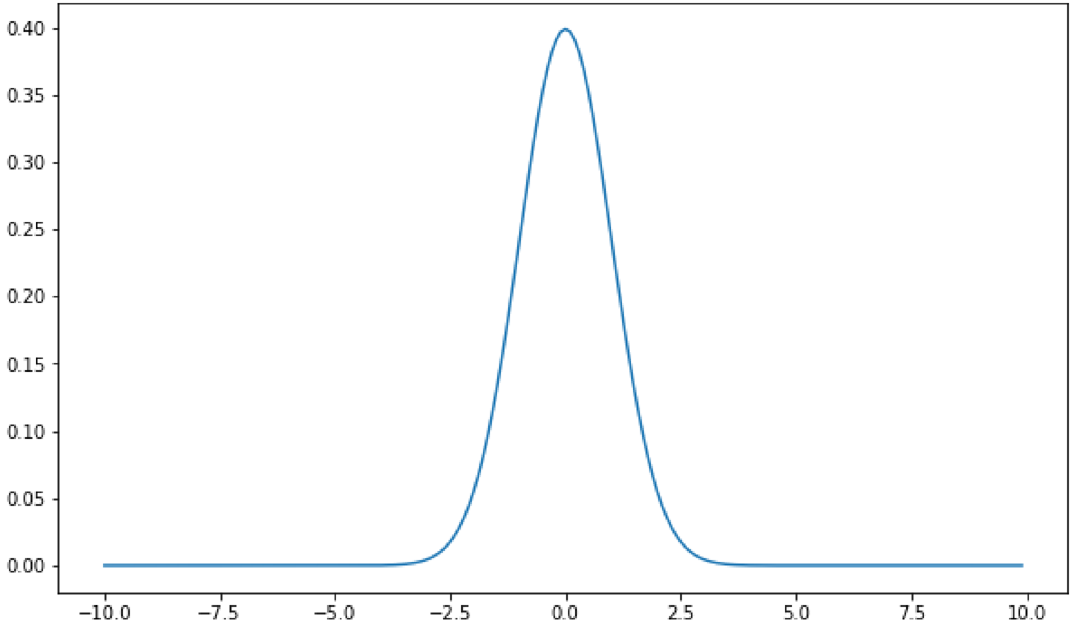
Standard Univariate Normal Distribution

```
y = np.arange(-10,10,0.1)

ProbG = 1/np.sqrt(2*np.pi)*np.exp(-1/2*y**2)
```

$$P_Y\left(Y=y\right)=\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}y^2\right)$$

```
from scipy.stats import norm

ProbG2 = norm.pdf(y)
```

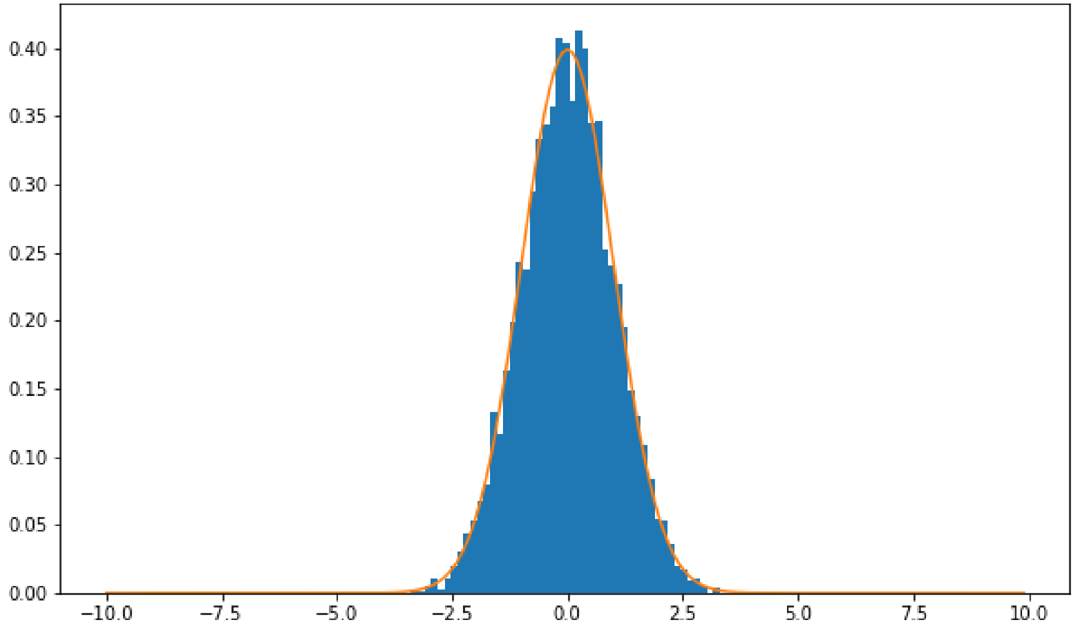




Standard Univariate Normal Distribution

- How to generate data from Gaussian distribution

```
x = np.random.randn(5000,1)

plt.figure(figsize=(10,6))
plt.hist(x, bins=51, normed=True)
plt.plot(y, ProbG2, label='G2')
plt.show()
```

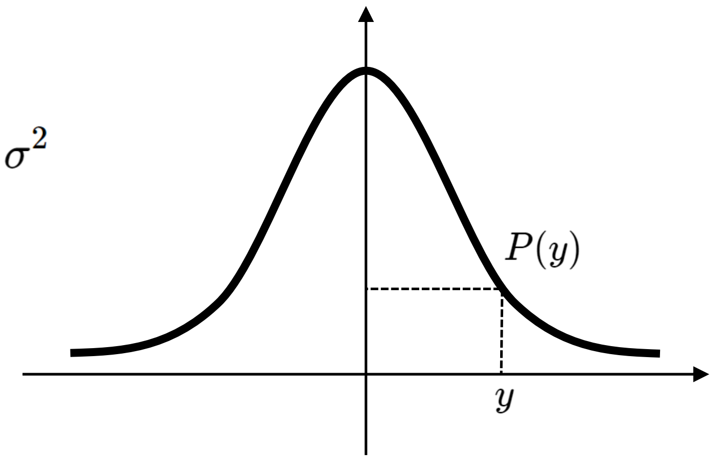


Univariate Normal Distribution

- Gaussian or normal distribution, 1D (mean μ, variance σ^2)
- It is a continuous pdf, but parameterized by only two terms, μ and σ

$$\mathcal{N}(x;\, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2} \frac{(x-\mu)^2}{\sigma^2} \right)$$

$$E[x] = \mu$$
$$\mathrm{var}(x) = E[(x-\mu)^2] = \sigma^2$$



$$x \sim \mathcal{N}\left(\mu, \sigma^2\right)$$
$$\Longrightarrow \quad P_Y(y) = P_X(x), \quad y = \frac{x-\mu}{\sigma}, \quad x = \sigma y + \mu$$

$$P_X(X = x) \sim \exp\left(-\frac{1}{2} \left(\frac{x-\mu}{\sigma} \right)^2 \right)$$
$$= \exp\left(-\frac{1}{2} \frac{(x-\mu)^2}{\sigma^2} \right)$$

$$x = \sigma y + \mu$$

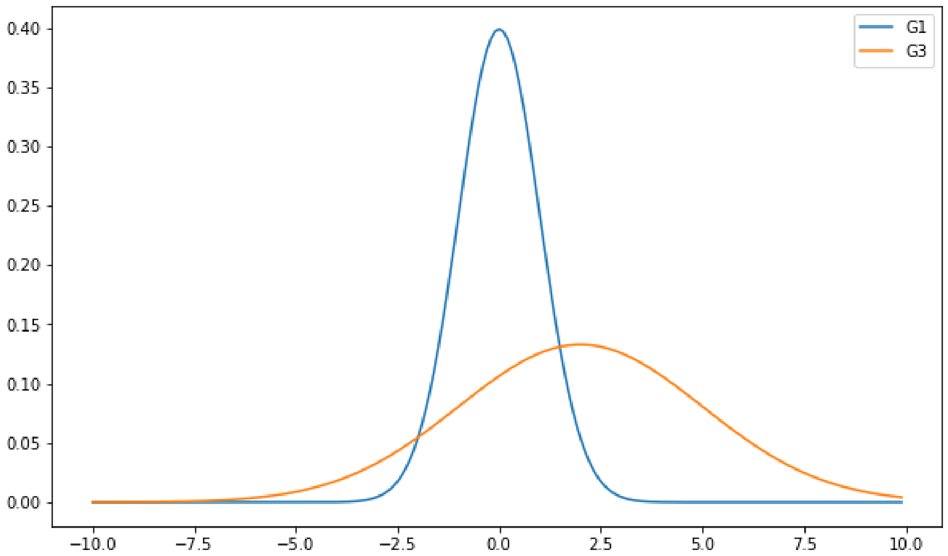
Affine transformation

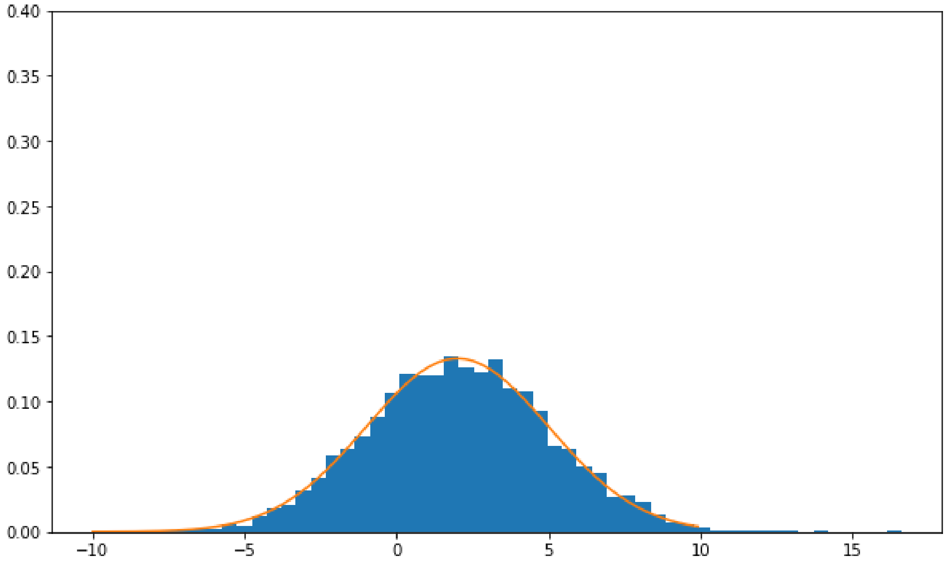
Univariate Normal Distribution

$$x = \sigma y + \mu$$

```
mu = 2
sigma = 3

x = np.arange(-10, 10, 0.1)

ProbG3 = 1/(np.sqrt(2*np.pi)*sigma) * np.exp(-1/2*(x-mu)**2/(sigma**2))

plt.figure(figsize=(10,6))
plt.plot(y,ProbG, label='G1')
plt.plot(x,ProbG3, label='G3')
plt.legend()
plt.show()
```

```
x = mu + sigma*np.random.randn(5000,1)

plt.figure(figsize=(10,6))
plt.hist(x, bins=51, normed=True)
plt.plot(y,ProbG2, label='G2')
plt.ylim([0,0.4])
plt.show()
```





Multivariate Gaussian Models

- Similar to a univariate case, but in a matrix form

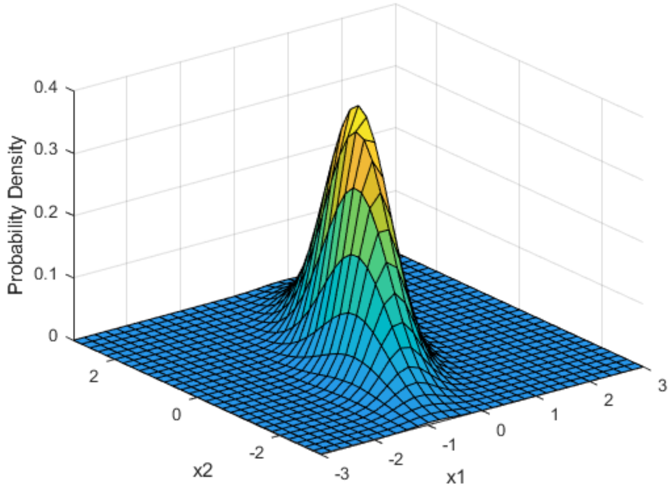
$$\mathcal{N}(x;\, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2} (x-\mu)^T \Sigma^{-1} (x-\mu) \right)$$

$$\begin{aligned} E[x] &= \mu \\ \mathrm{cov}(x) &= E[(x-\mu)(x-\mu)^T] = \Sigma \end{aligned}$$

$$\begin{aligned} \mu &= \text{length } n \text{ column vector} \\ \Sigma &= n \times n \text{ matrix (covariance matrix)} \\ |\Sigma| &= \text{matrix determinant} \end{aligned}$$

- Multivariate Gaussian models and ellipse
  - Ellipse shows constant Δ^2 value…

$$\Delta^2 = (x-\mu)^T \Sigma^{-1} (x-\mu)$$



  - The contours of equal probability is ellipse
    - Ellipsoidal probability contours
    - Bell shaped

Two Independent Variables

$$P\left(X_1 = x_1, X_2 = x_2\right) = P_{X_1}\left(x_1\right) P_{X_2}\left(x_2\right)$$

$$\sim \exp\left(-\frac{1}{2}\frac{\left(x_1 - \mu_{x_1}\right)^2}{\sigma_{x_1}^2}\right) \cdot \exp\left(-\frac{1}{2}\frac{\left(x_2 - \mu_{x_2}\right)^2}{\sigma_{x_2}^2}\right)$$

$$\sim \exp\left(-\frac{1}{2}\left(\frac{x_1^2}{\sigma_{x_1}^2} + \frac{x_2^2}{\sigma_{x_2}^2}\right)\right)$$

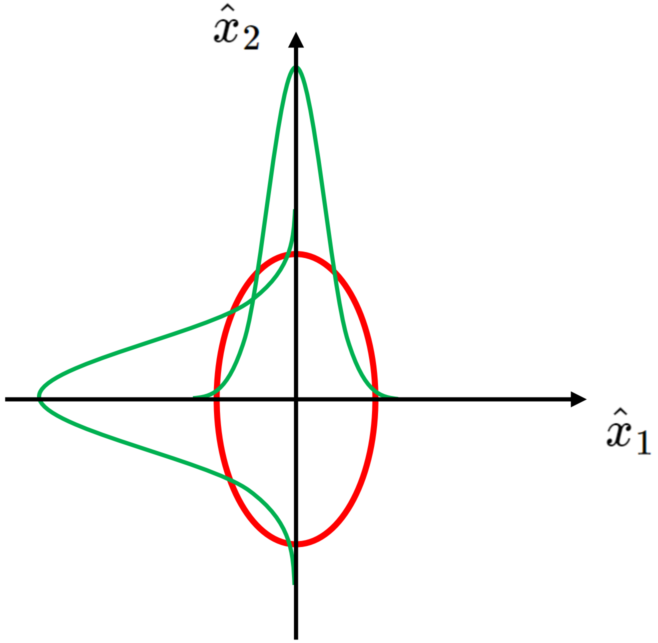
- In a matrix form
  - Diagonal covariance

$$P(x_1) \cdot P(x_2) = \frac{1}{Z_1 Z_2} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right)$$

$$\left(x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} \sigma_{x_1}^2 & 0 \\ 0 & \sigma_{x_2}^2 \end{bmatrix}\right)$$

Two Independent Variables

- Geometry of Gaussian



$$\frac{x_1^2}{\sigma_{x_1}^2} + \frac{x_2^2}{\sigma_{x_2}^2} = c \quad \text{(ellipse)}$$

$$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} \frac{1}{\sigma_{x_1}^2} & 0 \\ 0 & \frac{1}{\sigma_{x_2}^2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = c \qquad (\sigma_{x_1} < \sigma_{x_2})$$

- Summary in a matrix form

$$\mathcal{N}\left(0, \Sigma_x\right) \sim \exp\left(-\frac{1}{2} x^T \Sigma_x^{-1} x\right)$$

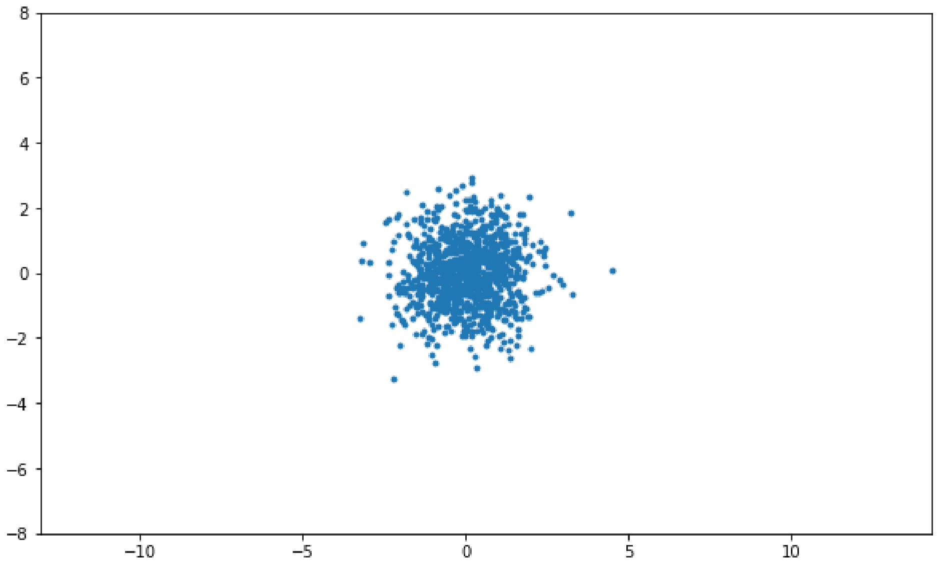
$$\mathcal{N}\left(\mu_x, \Sigma_x\right) \sim \exp\left(-\frac{1}{2} (x - \mu_x)^T \Sigma_x^{-1} \left(x - \mu_x\right)\right)$$

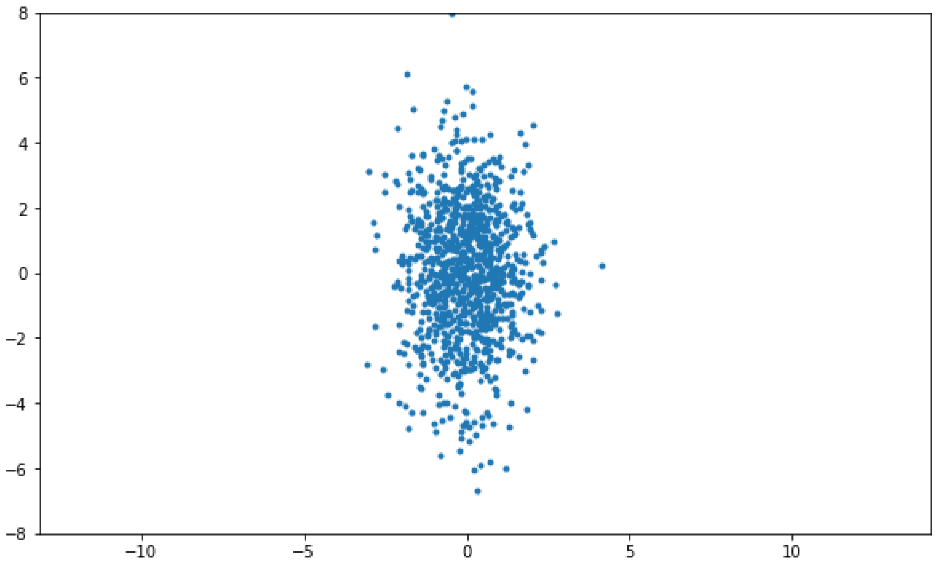
Two Independent Variables

```
mu = np.array([0, 0])
sigma = np.eye(2)

m = 1000
x = np.random.multivariate_normal(mu, sigma, m)
print(x.shape)
```

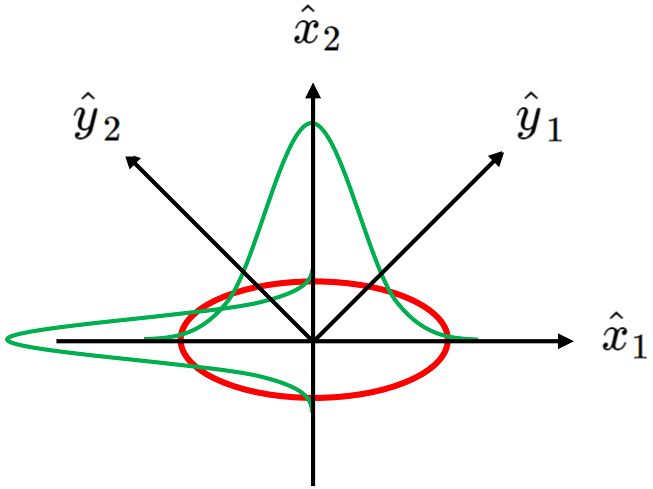
```
(1000, 2)
```

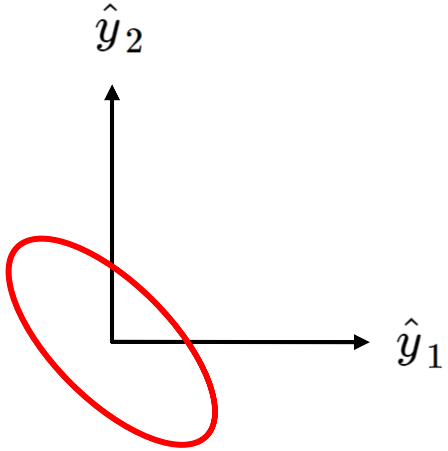
```
mu = np.array([0, 0])
sigma = np.array([[1, 0], [0, 4]])

m = 1000
x = np.random.multivariate_normal(mu, sigma, m)
```





Two Dependent Variables in $\{y_1, y_2\}$





- Compute $P_Y(y)$ from $P_X(x)$

$$P_X(x) = P_Y(y) \quad \text{where} \quad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

- Relationship between y and x

$$x = [\, \hat{x}_1 \quad \hat{x}_2 \,]^T y = U^T y$$

Two Dependent Variables in $\{y_1, y_2\}$

$$x^T \Sigma_x^{-1} x = y^T U \Sigma_x^{-1} U^T y = y^T \Sigma_y^{-1} y$$

$$\therefore \quad \Sigma_y^{-1} = U \Sigma_x^{-1} U^T$$

$$\Sigma_y = U \Sigma_x U^T$$

- Σ_x : covariance matrix of x
- Σ_y : covariance matrix of y

- If u is an eigenvector matrix of Σ_y, then Σ_x is a diagonal matrix

Two Dependent Variables in $\{y_1, y_2\}$

- Remark

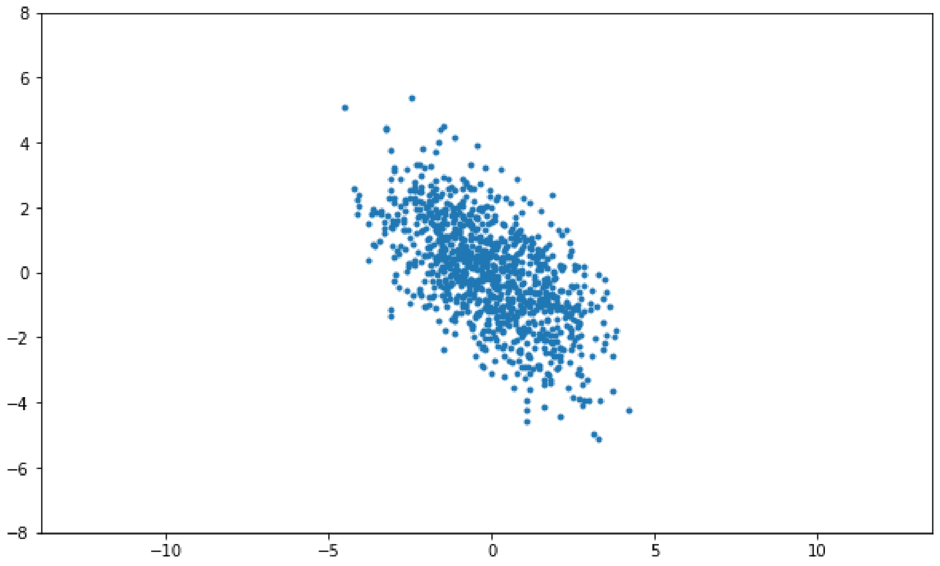
$$x \sim \mathcal{N}(\mu_x, \Sigma_x) \text{ and } y = Ax + b \text{ affine transformation}$$

$$\implies y \sim \mathcal{N}(\mu_y, \Sigma_y) = \mathcal{N}(A\mu_x + b, A\Sigma_x A^T)$$

$$\implies y \;\; \text{is also Gaussian with} \;\; \mu_y = Ax + b, \;\; \Sigma_y = A\Sigma_x A^T$$

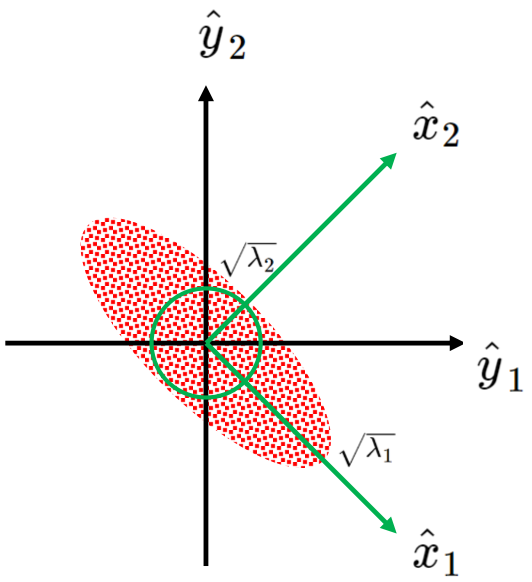
Two Dependent Variables in $\{y_1, y_2\}$

```
mu = np.array([0, 0])
sigma = 1./2.*np.array([[5, -3], [-3, 5]])

m = 1000
x = np.random.multivariate_normal(mu, sigma, m)
```



Decouple using Covariance Matrix

- Given data, how to find Σ_y and major (or minor) axis (assume $\mu_y = 0$)
- Statistics



$$\Sigma_y = \begin{bmatrix} \mathrm{var}(y_1) & \mathrm{cov}(y_1, y_2) \\ \mathrm{cov}(y_2, y_1) & \mathrm{var}(y_2) \end{bmatrix}$$

eigen-analysis

$$\Sigma_y \hat{x}_1 = \lambda_1 \hat{x}_1$$
$$\Sigma_y \hat{x}_2 = \lambda_2 \hat{x}_2$$

$$\Sigma_x^{-1} = \begin{bmatrix} \frac{1}{\sqrt{\lambda_1}^2} & 0 \\ 0 & \frac{1}{\sqrt{\lambda_2}^2} \end{bmatrix}$$

$$\Sigma_x = \begin{bmatrix} \sqrt{\lambda_1}^2 & 0 \\ 0 & \sqrt{\lambda_2}^2 \end{bmatrix}$$

$$\begin{aligned} \Sigma_y \begin{bmatrix} \hat{x}_1 & \hat{x}_2 \end{bmatrix} &= \begin{bmatrix} \hat{x}_1 & \hat{x}_2 \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \\ &= \begin{bmatrix} \hat{x}_1 & \hat{x}_2 \end{bmatrix} \Sigma_x \end{aligned}$$

$$y = Ux \implies U^T y = x$$
$$\begin{bmatrix} \hat{x}_1 & \hat{x}_2 \end{bmatrix} = U$$

$$\Sigma_y = U \Sigma_x U^T$$

Decouple using Covariance Matrix

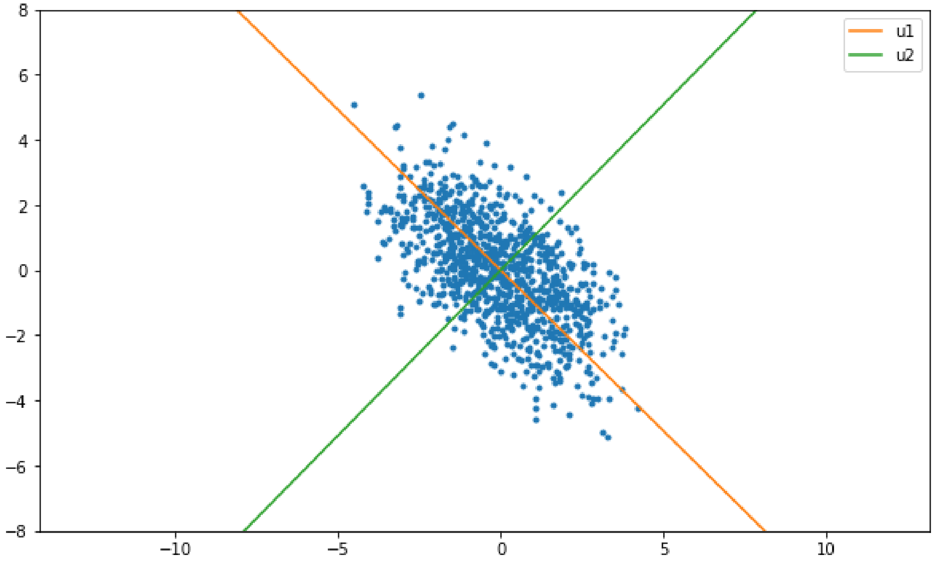
```
S = np.cov(x.T)
print ("S = \n", S)
```

```
S =
 [[ 2.59216411 -1.54924881]
 [-1.54924881  2.54567035]]
```

```
D, U = np.linalg.eig(S)

idx = np.argsort(-D)
D = D[idx]
U = U[:,idx]

print ("U = \n", U)
print ("D = \n", D)
```

```
U =
 [[ 0.7123916   0.70178217]
 [-0.70178217  0.7123916 ]]
D =
 [ 4.11834045  1.01949402]
```

```
xp = np.arange(-10, 10)

plt.figure(figsize=(10,6))
plt.plot(x[:,0],x[:,1],'.')
plt.plot(xp, U[1,0]/U[0,0]*xp, label='u1')
plt.plot(xp, U[1,1]/U[0,1]*xp, label='u2')
plt.axis('equal')
plt.ylim([-8, 8])
plt.legend()
plt.show()
```



Nice Properties of Gaussian Distribution

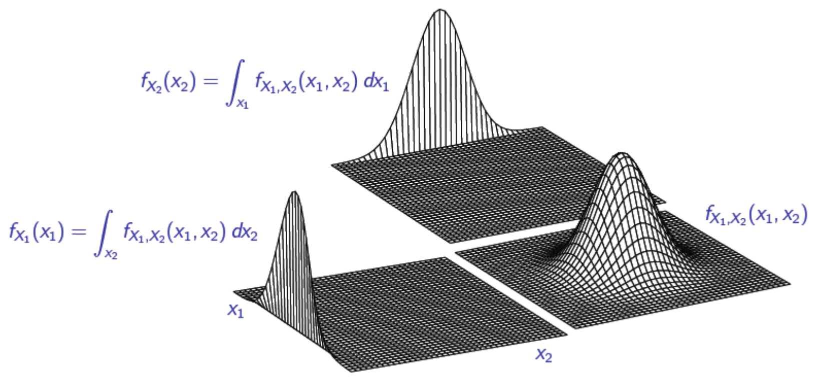
Properties of Gaussian Distribution

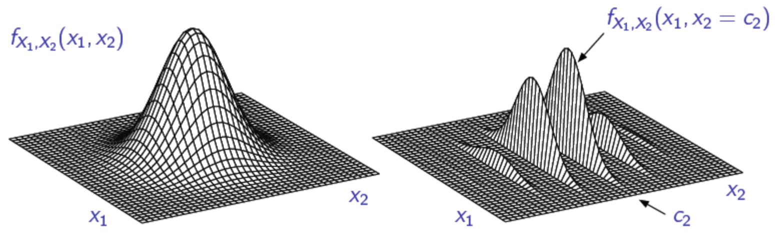
- Symmetric about the mean
- Parameterized

- Uncorrelated ⇒ independent

- Gaussian distributions are closed to
  - Linear transformation
  - Affine transformation

  - Reduced dimension of multivariate Gaussian
    - Marginalization (projection)
    - Conditioning (slice)
      - Highly related to inference





Affine Transformation of Gaussian

- Suppose $x \sim \mathcal{N}(\mu_x, \Sigma_x)$

- Consider affine transformation of x

$$y = Ax + b$$

- Then it is amazing that y is Gaussian with

$$E[y] = AE[x] + b = A\mu_x + b$$

$$\text{cov}(y) = \Sigma_y = A\text{cov}(x)A^T = A\Sigma_x A^T$$

Marginal Probability of Gaussian

- Suppose $x \sim \mathcal{N}(\mu, \Sigma)$

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}$$

- Let's look at the component x_1

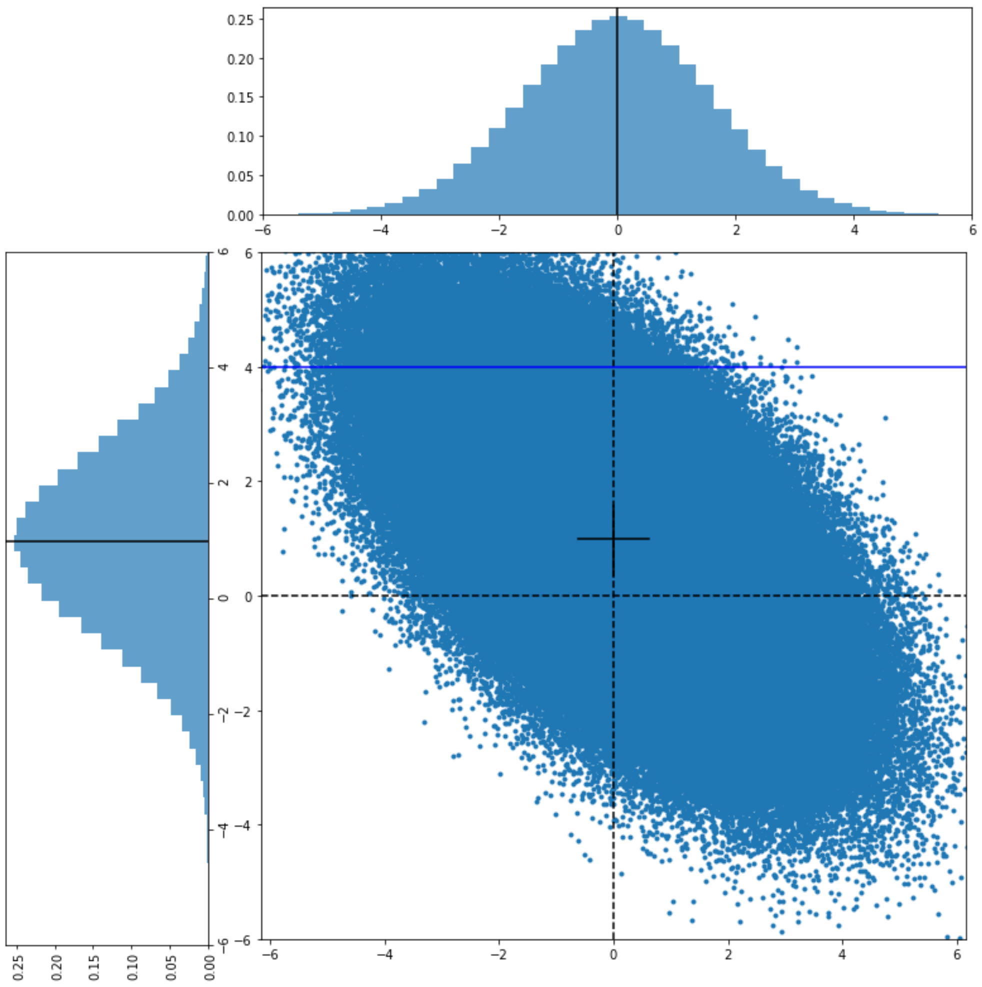
$$x_1 = \begin{bmatrix} I & 0 \end{bmatrix} x = Ax \quad \text{(affine transformation)}$$

$$E[x_1] = \begin{bmatrix} I & 0 \end{bmatrix} E[x] = \mu_1$$

$$\mathrm{cov}(x_1) = \begin{bmatrix} I & 0 \end{bmatrix} \mathrm{cov}(x) \begin{bmatrix} I \\ 0 \end{bmatrix} = \begin{bmatrix} I & 0 \end{bmatrix} \Sigma \begin{bmatrix} I \\ 0 \end{bmatrix} = \Sigma_{11}$$

- In fact, the random vector x_1 is also Gaussian.
  - (this is not obvious)

Marginalization (Projection)



Component of Gaussian Random Vector

- Suppose $x \sim \mathcal{N}(0, \Sigma)$, $c \in \mathbb{R}^n$ be a unit vector

$$y = c^T x$$

- y is the component of x in the direction c
- y is Gaussian with $E[y] = 0, \mathrm{cov}(y) = c^T \Sigma c$
- So $\mathrm{E}[y^2] = c^T \Sigma c$
- The unit vector that minimizes $c^T \Sigma c$ is the eigenvector of Σ with the smallest eigenvalue

$$E[y^2] = \lambda_{\min}$$

- Notice that we have seen this in PCA

Conditional Probability of Gaussian

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

- The conditional pdf of x given y is Gaussian

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

- The conditional mean is

$$E[x \mid y] = \mu_x + \Sigma_{xy}\Sigma_y^{-1}(y - \mu_y)$$

- The conditional covariance is

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

- Notice that conditional confidence intervals are narrower. i.e., measuring y gives information about x

Conditioning (Slice)