

## ECE 595: Homework 1

Yi Qiao, Class ID  
(Spring 2019)

### Exercise 2

(a) For a gaussian distribution:

$$\begin{aligned} E[x] &= \int_{-\infty}^{\infty} x \frac{1}{\sqrt{2\pi}\sigma^2} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx \\ &= \int_{-\infty}^{\infty} (x + \mu) \frac{1}{\sqrt{2\pi}\sigma^2} e^{-\frac{x^2}{2\sigma^2}} dx \\ &= \int_{-\infty}^{\infty} x \frac{1}{\sqrt{2\pi}\sigma^2} e^{-\frac{x^2}{2\sigma^2}} dx + \int_{-\infty}^{\infty} \mu \frac{1}{\sqrt{2\pi}\sigma^2} e^{-\frac{x^2}{2\sigma^2}} dx \\ &= 0 + \mu \frac{1}{\sqrt{2\pi}\sigma^2} \times \sigma\sqrt{2\pi} \\ &= \mu \end{aligned} \tag{1}$$

$$\begin{aligned} Var[x] &= \int_{-\infty}^{\infty} (x - \mu)^2 \frac{1}{\sqrt{2\pi}\sigma^2} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx \\ &= \frac{1}{\sqrt{2\pi}\sigma^2} \int_{-\infty}^{\infty} x^2 e^{-\frac{x^2}{2\sigma^2}} dx \\ &\quad \text{let } y = \frac{x}{\sigma}, \text{ then } dy = \frac{1}{\sigma} dx \\ &= \frac{\sigma^3}{\sqrt{2\pi}\sigma^2} \int_{-\infty}^{\infty} y^2 e^{-\frac{y^2}{2}} dy \\ &= \sigma^2 \end{aligned} \tag{2}$$

(b) Data generated and plotted as follows.

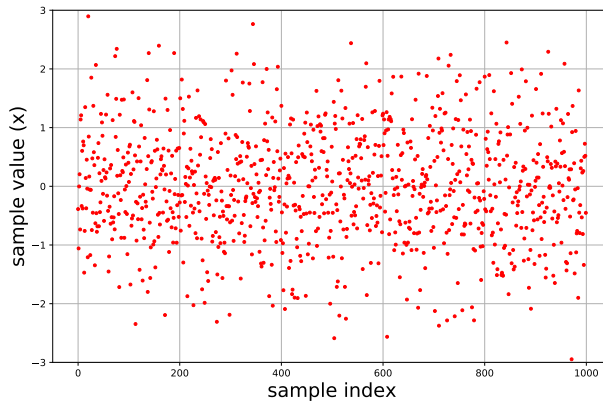
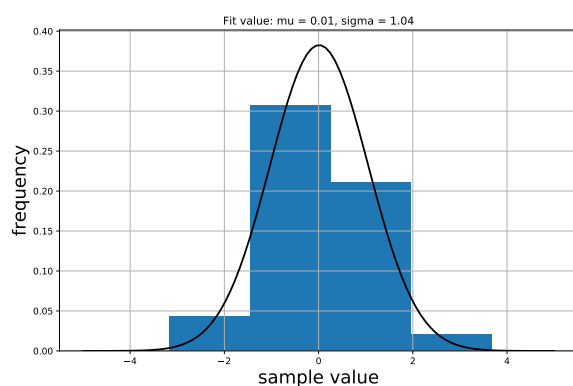


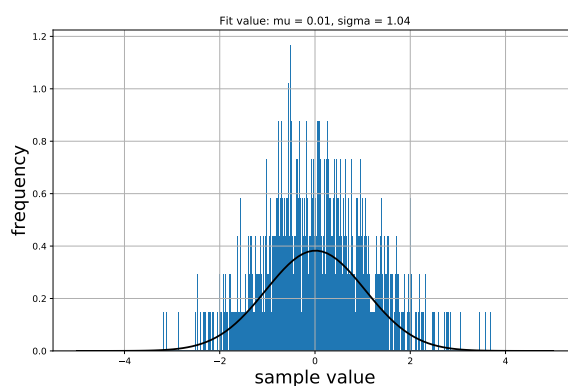
Figure 1: Gaussian random data.

(c)

(i)..(iv) plots shown below



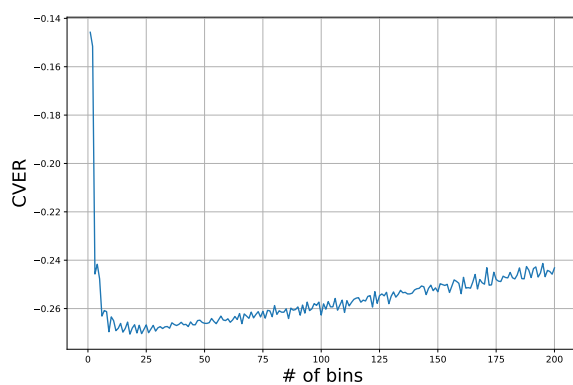
(a) 4 bins



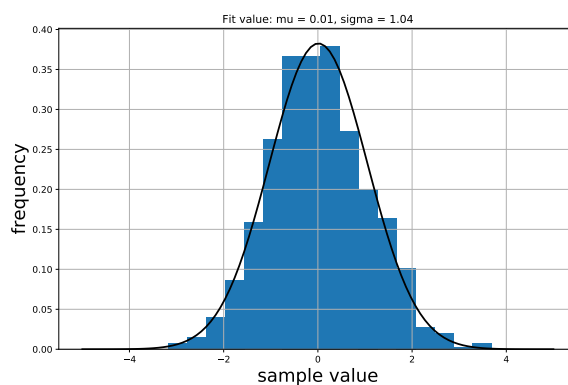
(b) 1000 bins

(v) TODO: fill this

(d) compare to part (c), the histogram fits a lot better with the PDF plots shown below



(c) Cross validation estimator of risk vs. # of bins



(d) Histogram and PDF overlaid with optimized number of bins

### Exercise 3

(a)

$$f_{\mathbf{X}}(\mathbf{x}) = \frac{1}{\sqrt{2\pi^2|\boldsymbol{\Sigma}|}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right\}$$

(i) plug in

$$\mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \boldsymbol{\mu} = \begin{bmatrix} 2 \\ 6 \end{bmatrix}, \text{ and } \boldsymbol{\Sigma} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

we get

$$\begin{aligned} f \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \left( \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right) &= \frac{1}{\sqrt{2\pi^2 \begin{vmatrix} 2 & 1 \\ 1 & 2 \end{vmatrix}}} \exp \left\{ -\frac{1}{2} \begin{bmatrix} x_1 - 2 \\ x_2 - 6 \end{bmatrix}^T \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}^{-1} \begin{bmatrix} x_1 - 2 \\ x_2 - 6 \end{bmatrix} \right\} \\ &= \frac{1}{\pi\sqrt{6}} \exp \left\{ -\frac{1}{3} \left( (x_1 - 2)^2 - (x_1 - 2)(x_2 - 6) + (x_2 - 6)^2 \right) \right\} \end{aligned} \quad (3)$$

(ii) plot shown below

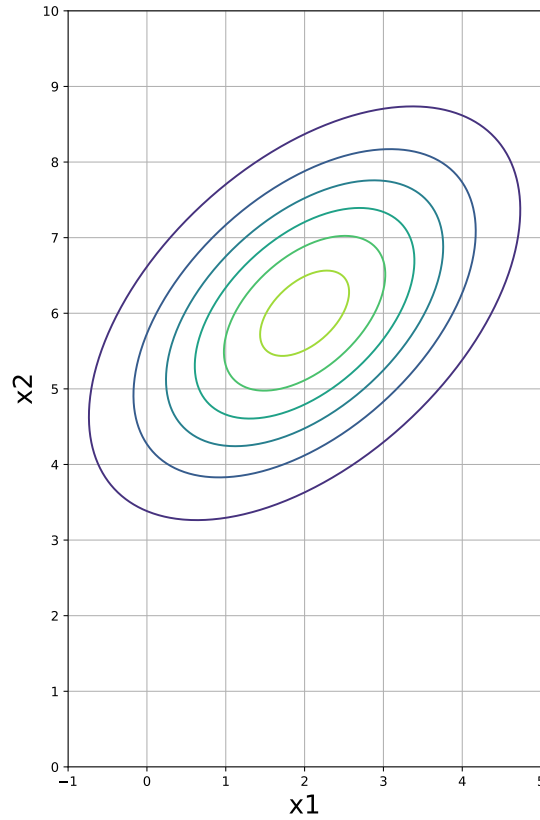


Figure 2: Gaussian random data

(b)

(i)

To prove:  $\boldsymbol{\mu}_Y = b$

$$\begin{aligned}\boldsymbol{\mu}_Y &= \mathbb{E}[Y] = \mathbb{E}[AX + b] = \mathbb{E}[AX] + b \\ &= A\mathbb{E}[X] + b \\ &\text{since } X \in \mathcal{N}(0, I), \mathbb{E}[X] = 0 \\ &= A0 + b \\ &= b\end{aligned}\tag{4}$$

To prove:  $\Sigma_Y = AA^T$

$$\begin{aligned}\Sigma_Y &= \mathbb{E}[(Y - \boldsymbol{\mu}_Y)(Y - \boldsymbol{\mu}_Y)^T] \\ &= \mathbb{E}[AXX^T A^T] \\ &= A\mathbb{E}[XX^T]A^T \\ &= A(I + 0)A^T \\ &= AA^T\end{aligned}\tag{5}$$

(ii) To prove:  $\Sigma_Y$  is symmetric and positive semi-definite

$$\begin{aligned}\Sigma_{Yij} &= \Sigma_{x=1}^n \mathbf{A}_{ix} \mathbf{A}_{xj}^T \\ &= \Sigma_{x=1}^n \mathbf{A}_{jx} \mathbf{A}_{xi}^T \\ &= \Sigma_{Yji} \\ &\text{thus symmetric} \\ \mathbf{x}^T \Sigma_Y \mathbf{x} &= \mathbf{x}^T AA^T \mathbf{x} \\ \text{let } \mathbf{y} &= \mathbf{x}^T \mathbf{A} \\ &= \mathbf{y} \mathbf{y}^T \\ &= \|\mathbf{y}\|^2 \geq 0 \\ &\text{thus positive semi-definite}\end{aligned}\tag{6}$$

(iii)

$$\text{Null}(\mathbf{A}) = \mathbf{0}$$

(iv) By inspection,

$$b = \boldsymbol{\mu}_Y = \begin{bmatrix} 2 \\ 6 \end{bmatrix}$$

By Cholesky decomposition,

$$\begin{aligned}\Sigma_Y &= AA^T \\ \mathbf{A} &= \begin{bmatrix} \sqrt{2} & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{6}}{2} \end{bmatrix}\end{aligned}$$

(c)

(i) Data points drawn from 2D standard gaussian distribution are shown below.

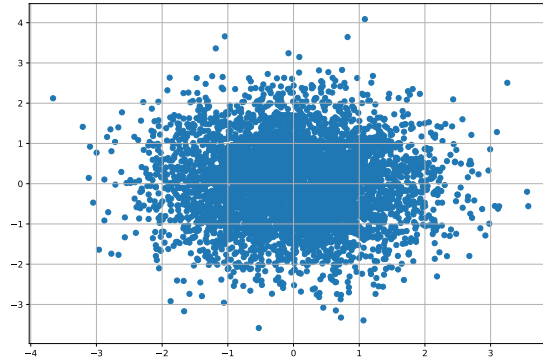
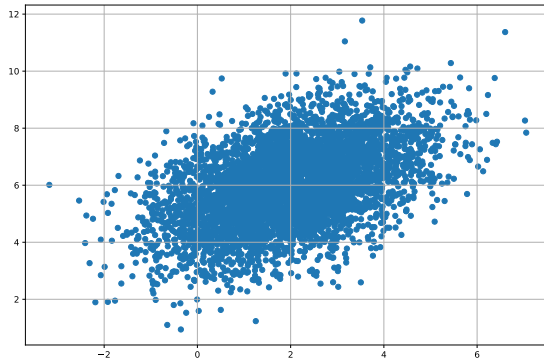
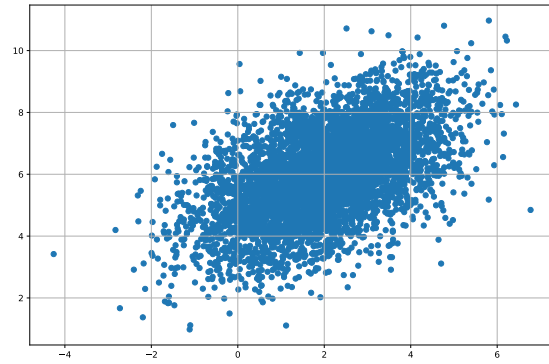


Figure 3: 2D standard Gaussian random data

(ii) After applying the affine transformation, the data plot shown below.



(a) After applying affine transformation

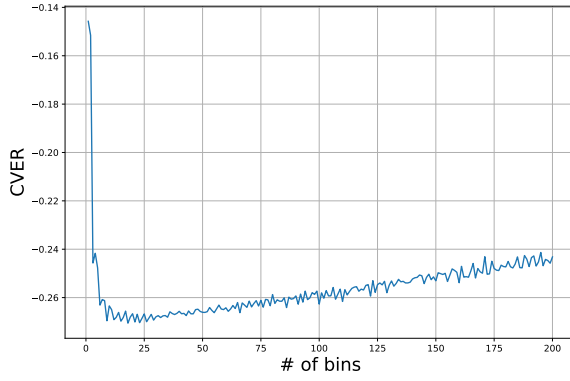


(b) After applying transform get from eigen

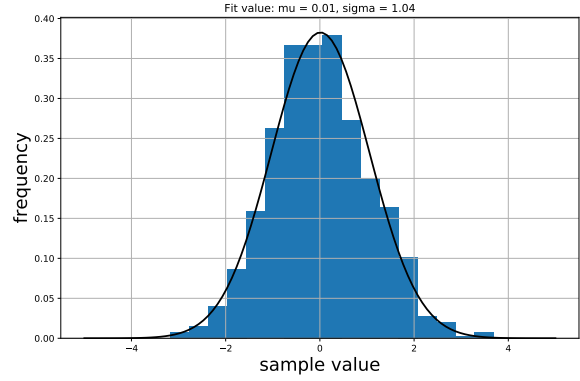
(iii) From my favorite python and numpy, for the data above:

$$\boldsymbol{\mu}_Y = \begin{bmatrix} 2.003 \\ 6.004 \end{bmatrix}$$

$$\boldsymbol{\Sigma}_Y = \begin{bmatrix} 2.017 & 1.050 \\ 1.050 & 2.074 \end{bmatrix}$$



(c) Cross validation estimator of risk vs. # of bins



(d) Histogram and PDF overlaid with optimized number of bins

## Exercise 4

(a)

**Proof**

$$\begin{aligned}
 |\mathbf{x}^T \mathbf{A} \mathbf{y}| &= \left| \sum_i \sum_j a_{ij} x_i y_j \right| \\
 &\leq \sum_i \sum_j |a_{ij}| |x_i| |y_j| \\
 &= \sum_i \sum_j (|a_{ij}|^{\frac{1}{2}})^2 |x_i| |y_j| \\
 &\text{by Cauchy Schwarz} \\
 &\leq \sqrt{\sum_i \sum_j |a_{ij}| |x_i|^2} \sqrt{\sum_i \sum_j |a_{ij}| |y_j|^2} \\
 &= \sqrt{\sum_i |x_i|^2 \sum_j |a_{ij}|} \sqrt{\sum_j |y_j|^2 \sum_i |a_{ij}|} \\
 &\leq \sqrt{\sum_i |x_i|^2 C} \sqrt{\sum_j |y_j|^2 R} \\
 &= \sqrt{RC} \|x\|_2 \|y\|_2
 \end{aligned} \tag{7}$$

(b)

(i) For a invertible  $n \times n$  matrix  $\mathbf{A}$

$$\begin{aligned}
 \mathbf{A} &= \mathbf{P} \mathbf{D} \mathbf{P} \\
 \mathbf{A}^{-1} &= (\mathbf{P} \mathbf{D} \mathbf{P}^{-1})^{-1} \\
 &= \mathbf{P}^{-1} \mathbf{D}^{-1} \mathbf{P}
 \end{aligned} \tag{8}$$

Since  $\mathbf{A}$  is positive definite,  $\lambda_i > 0$  for  $1 \leq i \leq n$ ,  $\mathbf{D}$  is invertible.

Thus  $\mathbf{A}$  is also invertible.

(ii)

$$f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = x_1^2 - x_2^2$$

the Hessian of this function is

$$\begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}$$

which is invertible but not positive definite

(iii) If a matrix is invertible while positive semi-definite, the matrix is positive definite.  
 By definition of positive semi-definite,  $\mathbf{xAx}^T \geq 0$ , and for a eigenvalue of  $\mathbf{A}$ ,  $\lambda_i$ ,  $\lambda_i \mathbf{u}_i = \mathbf{A}\mathbf{u}_i$   
 So,

$$\begin{aligned}\mathbf{u}_i^T \mathbf{A}\mathbf{u}_i &= \lambda_i \\ \lambda_i &\geq 0\end{aligned}\tag{9}$$

If the matrix is invertible, it cannot have  $\lambda_i = 0$ .  
 So,  $\lambda_i > 0$ , thus  $\mathbf{xAx}^T > 0$ .  
 The matrix  $\mathbf{A}$  is positive definite.

(c) To prove:  $(\exists \mathbf{A}^\dagger) \mathbf{AA}^\dagger \mathbf{A} = \mathbf{A}$

$$\begin{aligned}\mathbf{A} &= \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \\ \mathbf{AA}^\dagger \mathbf{A} &= \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \mathbf{A}^\dagger \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \\ &= \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \\ &\quad \text{*simplify*} \\ \mathbf{U}^T \mathbf{A}^\dagger \mathbf{U}\mathbf{\Lambda} &= \mathbf{\Lambda}\mathbf{\Lambda}^- \\ \mathbf{A}^\dagger &= \mathbf{U}\mathbf{\Lambda}^- \mathbf{U}^T\end{aligned}\tag{10}$$

Since  $\mathbf{A}$  is symmetric, it is guaranteed that can be decompose, thus  $\mathbf{A}^\dagger$  exist.



# Code

## Exercise2

```
import matplotlib.pyplot as plt
from scipy.stats import norm
import numpy as np

def newfig():
    fig = plt.figure(figsize=(9,6), dpi=300)
    ax = fig.add_subplot(111)
    return fig, ax

def final_adjust(fn):
    plt.tight_layout()
    plt.savefig(fn, bbox='tight')

if __name__ == '__main__':

    # (b)
    fig, ax = newfig()
    mu, sigma = 0, 1
    X = np.random.normal(mu, sigma, 1000)
    ax.set_ylim([-3,3])
    ax.set_xlabel('sample index', fontsize = 20)
    ax.set_ylabel('sample value (x)', fontsize = 20)
    ax.grid(True)
    ax.plot(X, 'r.')
    final_adjust('../pix/exercise2_b.pdf')

    # (c)
    X = np.random.normal(mu, sigma, 1000)

    # 4 bins
    fig, ax = newfig()
    ax.hist(X, 4, density=True)
    mu, sigma = norm.fit(X)
    x_ticks = np.linspace(-5, 5, 100)
    pdf = norm.pdf(x_ticks, mu, sigma)
    ax.grid(True)
    ax.plot(x_ticks, pdf, 'k', linewidth=2)
    ax.set_title("Fit value: mu = %.2f, sigma = %.2f"%(mu, sigma))
    ax.set_xlabel('sample value', fontsize = 20)
    ax.set_ylabel('frequency', fontsize = 20)
    final_adjust('../pix/exercise2_c1.pdf')

    # 1000 bins
    fig, ax = newfig()
```

```

ax.hist(X, 1000, density=True)
mu, sigma = norm.fit(X)
x_ticks = np.linspace(-5, 5, 100)
pdf = norm.pdf(x_ticks, mu, sigma)
ax.grid(True)
ax.plot(x_ticks, pdf, 'k', linewidth=2)
ax.set_title("Fit value: mu = %.2f, sigma = %.2f"%(mu, sigma))
ax.set_xlabel('sample value', fontsize = 20)
ax.set_ylabel('frequency', fontsize = 20)
final_adjust('.../pix/exercise2_c2.pdf')

# (d)
n = 1000
J_h = np.zeros(200)
for m in range(1,201):
    h = (max(X) - min(X))/m
    p = np.array([(X >= min(X) + h*x) & (X < min(X) + h*(x+1))].sum()/n for x in range(m))
    J_h[m-1] = 2 / h / (n-1) - (n+1) / h / (n-1) * sum(p**2)

fig, ax = newfig()
ax.grid(True)
ax.plot([x for x in range(1,201)], J_h)
ax.set_xlabel('# of bins', fontsize = 20)
ax.set_ylabel('CVER', fontsize = 20)
final_adjust('.../pix/exercise2_d1.pdf')

m_star = J_h.argmin()

fig, ax = newfig()
ax.hist(X, m_star, density=True)
mu, sigma = norm.fit(X)
x_ticks = np.linspace(-5, 5, 100)
pdf = norm.pdf(x_ticks, mu, sigma)
ax.grid(True)
ax.plot(x_ticks, pdf, 'k', linewidth=2)
ax.set_title("Fit value: mu = %.2f, sigma = %.2f"%(mu, sigma))
ax.set_xlabel('sample value', fontsize = 20)
ax.set_ylabel('frequency', fontsize = 20)
final_adjust('.../pix/exercise2_d2.pdf')

```

### Exercise3 (a)

```
import numpy as np
import matplotlib.pyplot as plt

def newfig():
    fig = plt.figure(figsize=(6,9), dpi=300)
    ax = fig.add_subplot(111)
    return fig, ax

def final_adjust(fn):
    plt.tight_layout()
    plt.savefig(fn, bbox='tight')

if __name__ == '__main__':
    x1 = np.arange(-1,5,0.01)
    x2 = np.arange(0,10,0.01)
    Z = np.zeros((1000,600))
    for i in range(len(x1)):
        for j in range(len(x2)):
            Z[j][i] = 1/(np.pi*6**(1/2))*np.exp(-1/3*((x1[i]-2)**2
                -(x1[i]-2)*(x2[j]-6)+(x2[j]-6)**2))

    fig, ax = newfig()
    ax.contour(x1,x2,Z)
    ax.grid(True)
    ax.set_yticks([i for i in range(11)])
    ax.set_xticks([i for i in range(-1,6)])
    ax.set_xlabel('x1', fontsize = 20)
    ax.set_ylabel('x2', fontsize = 20)

    final_adjust('../pix/exercise3_a.pdf')
```

### Exercise3 (c)

```
import numpy as np
import matplotlib.pyplot as plt

def newfig():
    fig = plt.figure(figsize=(6,9), dpi=300)
    ax = fig.add_subplot(111)
    return fig, ax

def final_adjust(fn):
    plt.tight_layout()
    plt.savefig(fn, bbox='tight')

if __name__ == '__main__':
    x1 = np.arange(-1,5,0.01)
    x2 = np.arange(0,10,0.01)
    Z = np.zeros((1000,600))
    for i in range(len(x1)):
        for j in range(len(x2)):
            Z[j][i] = 1/(np.pi*6**(1/2))*np.exp(-1/3*((x1[i]-2)**2
                -(x1[i]-2)*(x2[j]-6)+(x2[j]-6)**2))

    fig, ax = newfig()
    ax.contour(x1,x2,Z)
    ax.grid(True)
    ax.set_yticks([i for i in range(11)])
    ax.set_xticks([i for i in range(-1,6)])
    ax.set_xlabel('x1', fontsize = 20)
    ax.set_ylabel('x2', fontsize = 20)

    final_adjust('../pix/exercise3_a.pdf')
```