1.

(1) L1-norm: 9 L2-norm: √35

(2) L1-norm: 20 L2-norm: √272

2.

(1) cos(θ) = 0

(2) cos(θ) = -⅓

3.

(a) Non-convex

(b) Non-convex

(c) Convex

(d) Convex

4.

(1) Data matrix:

8	8	16	4
4	1	1	16
6	4	4	2
4	2	4	1
8	4	8	2

Label matrix:

1
-1
1
-1
1

(2)

code:

```
import numpy as np
data_matrix = np.array([[8,8,16,4], [4,1,1,16], [6, 4, 4, 2], [4, 2, 4, 1], [8, 4, 8, 2]])
label_matrix = np.array([1, -1, 1, -1, 1])
W = np.array([1.2, 2, 0.5, 0.7])
np.transpose(W);
predict = np.dot(data_matrix, W)-20
for i in range(0, len(predict)):
    if predict[i] > 0:
        predict[i] = 1
```

```
    else:
        predict[i] = -1
print(predict)
```

Answer:
[1 -1 -1 -1 1]

(3)

Recall: true positive/(true positive + false negative) = 2/3
Precision: true positive/ guess positive = 2/2 = 1
F-score: 4/5
Accuracy: 4/5

5.

(1)

least square estimation

$$w^* = \arg\min_w \sum_i (x_i^T \cdot W - y_i)^2$$

Let: $X = (x_i \cdots x_n)^T \qquad Y = (y_1, \cdots, y_n)^T$

$$W^* = \arg\min_W g(W) = (X \cdot W - Y)^T (X \cdot W - Y)$$

$$g(W) = W^T X^T X W - W^T X^T Y - Y^T X W + Y^T Y$$

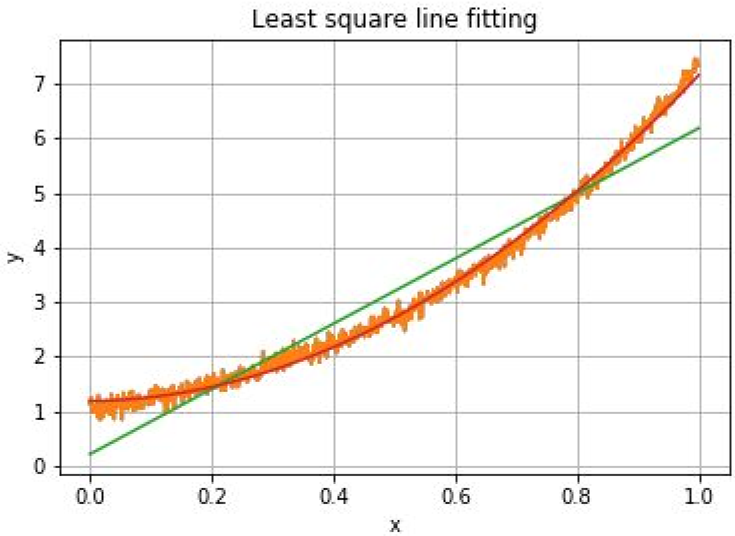
$$\frac{dg(x)}{dW} = 2X^T X W - 2X^T Y = 0$$

$$W^* = (X^T X)^{-1} X^T Y$$

(2)code:

```
import matplotlib.pyplot as plt
import numpy as np
data = np.loadtxt('data.txt',dtype='float')
x = data[:,0].reshape(len(data),1)
y = data[:,1].reshape(len(data),1)
plt.plot(x,y)
plt.grid()
X = np.hstack((np.ones((len(x),1)),np.power(x,1)))
X = np.hstack((X,np.power(x,2)))
X_t = X.transpose((1,0))
sol = np.dot(np.linalg.inv(np.dot(X_t,X)),np.dot(X_t,y))
plt.plot(x,y)
plt.plot(x,sol[0]+sol[1]*x+sol[2]*x*x)
plt.title('Least square line fitting')
plt.xlabel('x')
```

```
plt.ylabel('y')
plt.savefig('./Q6.png')
```



(3)code:

```
import matplotlib.pyplot as plt
import numpy as np
data = np.loadtxt('data.txt',dtype='float')
x = data[:,0].reshape(len(data),1)
y = data[:,1].reshape(len(data),1)
plt.plot(x,y)
plt.grid()
X = np.hstack((np.ones((len(x),1)),np.power(x,1)))
X2 = np.hstack((X,np.power(x,2)))
X_t = X.transpose((1,0))
X2_t = X2.transpose((1,0))
sol = np.dot(np.linalg.inv(np.dot(X_t,X)),np.dot(X_t,y))
sol2 = np.dot(np.linalg.inv(np.dot(X2_t,X2)),np.dot(X2_t,y))
plt.plot(x,y)
plt.plot(x,sol[0]+sol[1]*x)
plt.plot(x,sol2[0]+sol2[1]*x+sol2[2]*x*x)
plt.title('Least square line fitting')
plt.xlabel('x')
plt.ylabel('y')
plt.savefig('./Q5.png')

line_L2 = np.dot((np.dot(X,sol)-y).transpose(),np.dot(X,sol)-y) ** 0.5
polynomial_L2 = np.dot((np.dot(X2,sol2)-y).transpose(),np.dot(X2,sol2)-y) ** 0.5
print line_L2
```

print polynomial_L2

Answer:
L2 distance for line: 14.35311332
L2 distance for polynomial: 3.96995097

6.
(1)

gradient of L_1 loss.

$$W^* = \text{argmin}_W \sum_{i=1}^{n} |x_i^T \cdot W - y_i|$$

$$\frac{\partial |f(w)|}{\partial w} = \begin{cases} \frac{\partial |f(w)|}{\partial w} & \text{if } f(w) \geqslant 0 \\ -\frac{\partial |f(w)|}{\partial w} & \text{if } f(w) < 0 \end{cases}$$

$$= \text{sign}(f(w)) \cdot \frac{\partial f(w)}{\partial w}$$

$$L(w) = \sum_{i=1}^{n} |x_i^T w - y_i|$$

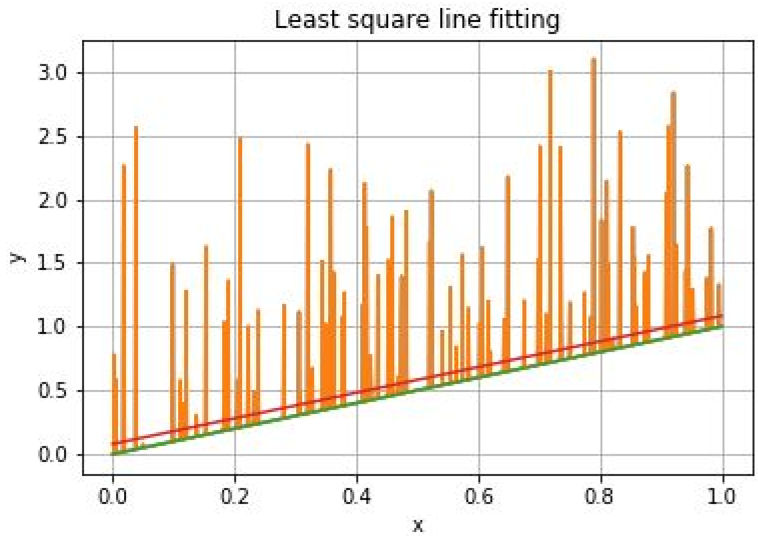
$$\frac{\partial L(w)}{\partial (w)} = \sum_{i=1}^{n} \text{sign}(x_i^T W - y_i) \cdot x_i$$

$$W_{t+1} = W_t - \lambda_t \frac{\partial L(w)}{\partial (w)}$$

(2)

```
import matplotlib.pyplot as plt
import numpy as np
data = np.loadtxt('data_2.txt',dtype='float')
x = data[:,0].reshape(len(data),1)
y = data[:,1].reshape(len(data),1)
plt.plot(x,y)
plt.grid()
X = np.hstack((np.ones((len(x),1)),np.power(x,1)))
X_t = X.transpose((1,0))
sol = np.dot(np.linalg.inv(np.dot(X_t,X)),np.dot(X_t,y))
W = np.array([[0], [0]])
for i in range(0, 2000):
    tmp = (np.dot(X, W)-y)
    gradient = 0;
    for j in range(0, len(tmp)):
        if tmp[j][0] >= 0:
            gradient += X[j];
        else:
            gradient -= X[j];
    W_t = W.transpose((1,0)) - 0.00001*gradient
    W = W_t.transpose((1,0));
plt.plot(x,y)
plt.plot(x, W[0]+W[1]*x)
plt.plot(x,sol[0]+sol[1]*x)
plt.title('Least square line fitting')
plt.xlabel('x')
plt.ylabel('y')
plt.savefig('./Q6.png')
```

(3)


Least square line fitting
3.0
2.5
2.0
1.5
1.0
0.5
0.0
y
0.0
0.2
0.4
0.6
0.8
1.0
x