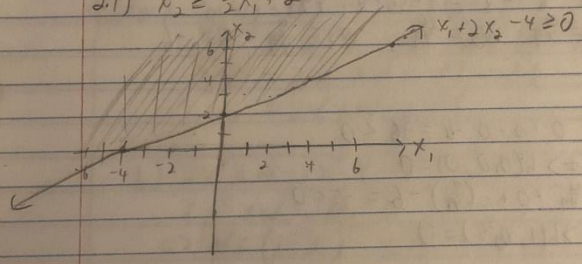
① a) True

b) False

②

2.1) $x_2 \geq \frac{1}{2}x_1 + 2$



2.2)

i) $x_2 = -x_1 + 1$

$\Rightarrow x_1 + x_2 - 1 \geq 0$

$\Rightarrow w_1 = 1 \quad w_2 = 1 \quad b = -1$

ii) A: $3 + 2 - 1 = 4 > 0$

$\Rightarrow h(3,2) = 1$

B: $-1 + 0 - 1 = -2 < 0$

$\Rightarrow h(-1,0) = 0$

2.3)

i) $\vec{AB} = \langle -4, -2, -2 \rangle$

$\vec{AC} = \langle 1, -1, 1 \rangle$

$$\begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ -4 & -2 & -2 \\ 2 & -1 & -5 \end{vmatrix} = \langle 8, -24, 12 \rangle$$

$A = (3,2,4)$

$8(x-3) - 24(y-2) + 12(z-4) \geq 0$

$2(x-3) - 6(y-2) + 3(z-4) \geq 0$

$2x - 6 - 6y + 12 + 3z - 12 \geq 0$

$2x - 6y + 3z - 6 \geq 0$

$2^2 + 6^2 + 3^2 = 4 + 36 + 9 = 49$

$\Rightarrow w_1 = \frac{4}{49} \quad w_2 = \frac{36}{49} \quad w_3 = \frac{9}{49} \quad b = -6$

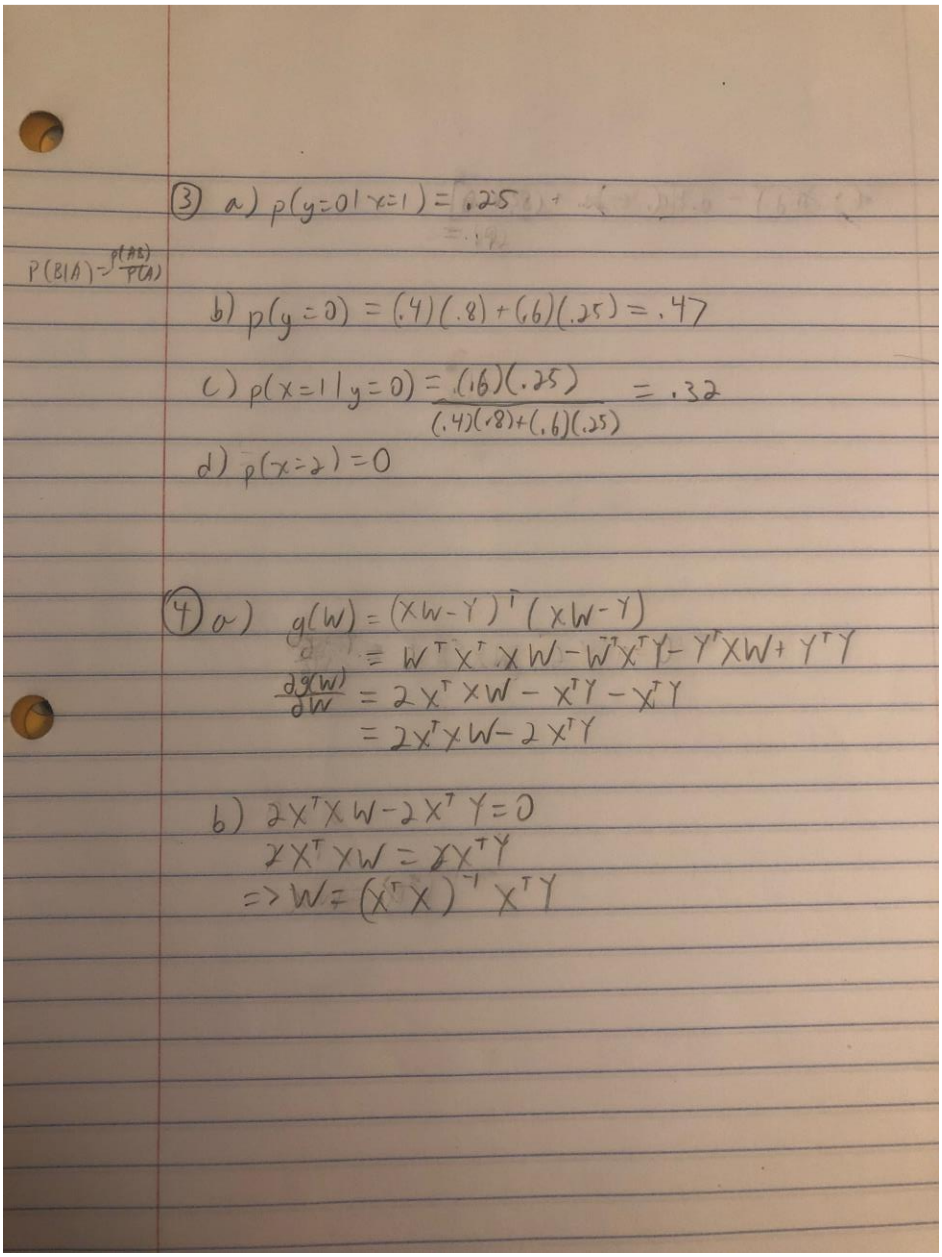
ii) p: $0 + 0 + 0 - 6 = -6 < 0$

$\Rightarrow h(0,0,0) = 0$

q: $\frac{4}{49} + 0 + 5\left(\frac{9}{49}\right) - 6 = -5 < 0$

$\Rightarrow h(1,0,5) = 0$

3&4)



P(B|A) = P(A∩B) / P(A)

(3) a) p(y=0 | x=1) = .25

b) p(y=0) = (.4)(.8) + (.6)(.25) = .47

c) p(x=1 | y=0) = (.6)(.25) / ((.4)(.8)+(.6)(.25)) = .32

d) p(x=2) = 0

(4) a) g(W) = (XW − Y)^T (XW − Y)

= W^T X^T XW − W^T X^T Y − Y^T XW + Y^T Y

dg(W)/dW = 2X^T XW − X^T Y − X^T Y

= 2X^T XW − 2X^T Y

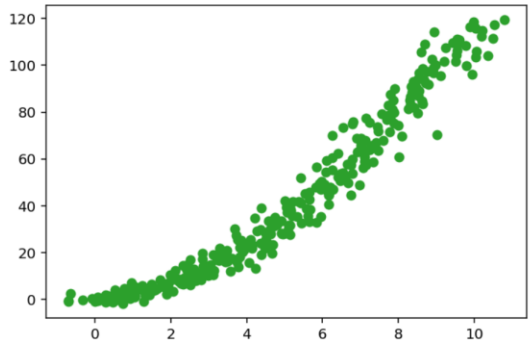
b) 2X^T XW − 2X^T Y = 0

2X^T XW = 2X^T Y

⇒ W = (X^T X)^{-1} X^T Y

2.1: 2D Scatterplot

In [24]:
```
# TODO 1: Plot the a scatter graph of data.
plt.scatter(X, Y)
plt.show()
```



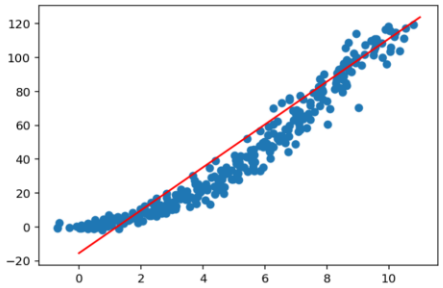
2.2: Compute the Least Sequare Line Using the Closed Form

In [25]:
```
# TODO 2: Compute the least square line over the given data
# Assume Y = w0 + w1 * X = (w0, w1).(1, X) = W.X1
# You might find the following functions useful: np.matrix, np.hstack, np.ones, np.reshape, dot
# Example: make a numpy matrix. https://docs.scipy.org/doc/numpy/reference/generated/numpy.matrix.html
# Example: stack arrays horizontally. https://docs.scipy.org/doc/numpy/reference/generated/numpy.hstack.html
# Example: create a new array filled with ones https://docs.scipy.org/doc/numpy/reference/generated/numpy.ones.html
# Example: reshape array without changing data https://docs.scipy.org/doc/numpy/reference/generated/numpy.reshape.html
# Example: A*B. Dot product of two arrays https://docs.scipy.org/doc/numpy/reference/generated/numpy.dot.html

XNew = X.T
a = np.ones(XNew.shape[0])
XNew = np.vstack((a, XNew))
XNew = XNew.T

W = (np.linalg.inv(XNew.T.dot(XNew)).dot(XNew.T)).dot(Y)
w0 = W[0]
w1 = W[1]
print('Y = {:.2f} + {:.2f}*X'.format(w0, w1))
```

```
Y = -15.47 + 11.61*X
```

2.3: 2D Scatterplot & the Estimated Least Square Line

In [28]:
```
# TODO 3.  Plot the the estimated least square line on top of the scatter plot in (2).
# The scatterplot and the line should be in the same figure.
plt.scatter(X, Y)
x = np.linspace(0,12,12)
y = w0 + w1 * x
plt.plot(y, color='r')
plt.show()
```



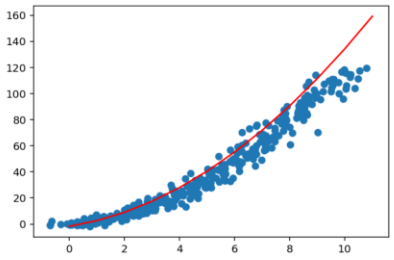
2.4: Compute the Least Square Parabola Using the Closed Form

In [61]:
```
# TODO 4. Compute the least square parabola over the given data
# Assume Y = w0 + w1 * X + w2 * X^2 = (w0, w1, w2).(1, X, X^2) = W.X2

X2 = np.vstack((XNew.T, X ** 2)).T
W = (np.linalg.inv(X2.T.dot(X2)).dot(X2.T)).dot(Y)
w0 = W[0]
w1 = W[1]
w2 = W[2]
print('Y = {:.2f} + {:.2f}*X + {:.2f}*X^2'.format(w0, w1, w2))
```

```
Y = -1.71 + 3.02*X + 0.87*X^2
```

2.5: 2D Scatterplot & the Estimated Parabola

In [62]:
```
# TODO 5.  Plot the the estimated parabola on top of the scatter plot in (2).
# The scatterplotand the parabola should be in the same figure
plt.scatter(X, Y)
x = np.linspace(0,12,12)
y = w0 + w1 * x + w2 * (x ** 2)
plt.plot(y, color='r')
plt.show()
```



5)

In [44]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score

# Pre-defined W is given
reg = LinearRegression().fit(X_train, Y_train)
print(reg.coef_)
print(reg.intercept_)

def regression (x,y):
    w = reg.coef_
    b = reg.intercept_
    reg_diff = 0
    # TODO: ******** To be filled ********
    # Hint: use a for loop loop through x, and make prediction.
    # Then, you can compare your prediction with y to calculate error rate

    ypred = np.zeros(y.shape)
    for i in range(len(y)):
        ypred[i] = w.T.dot(x[i]) + b

    loss = 0
    for i in range(len(ypred)):
        loss += (ypred[i] - y[i]) ** 2
    loss /= len(x)
    reg_diff = loss ** .5
    return reg_diff

def classification (x,y):
    w = reg.coef_
    b = reg.intercept_
    clss_diff = 0
    # TODO: ******** To be filled ********
    # Hint: use a for loop loop through x, and make prediction.
    # Then, you can compare your prediction with y to calculate error rate

    ypred = np.zeros(y.shape)
    h = np.zeros(y.shape)
    for i in range(len(x)):
        ypred[i] = w.T.dot(x[i]) + b
        if(ypred[i] >= .5):
            h[i] = 1
        if(h[i] != y[i]):
            clss_diff += 1

    clss_diff /= len(x)
    return clss_diff



print('Training regression and classification errors are:')
print(regression(X_train, Y_train))
print(classification(X_train, Y_train))
print('Testing regression and classification errors are:')
print(regression(X_test, Y_test))
print(classification(X_test, Y_test))

```

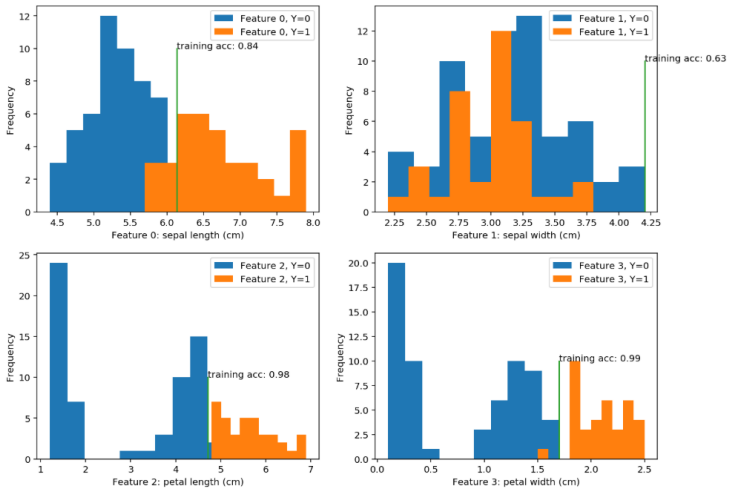
```
[ 0.12975624  0.12249935 -0.11714156  0.67102651]
-1.1698768088050127
Training regression and classification errors are:
0.27976412743241214
0.06
Testing regression and classification errors are:
0.33100713441395574
0.14
```

6)

In [41]:

```python
# Calculate the accuracy of prediction given feature, target and threshold.
def calc_acc(Xj, Y, thres):
    """
    Calculate the accuracy given feature, target and threshold.
        Xj:    j-th feature. This array only contains 1 feature for all data points,
               so the shape should be (count of data points,)
        Y:     Target array. Shape: (count of data points,)
        thres: Threshold.
    Return the accuracy of prediction.
    """
    # Step 1. Count the number of correct predictions and incorrect predictions.
    #         Here, for simplicity, we assume:
    #               If feature <= threshold, we predict it as Y = 0.
    #               If feature  > threshold, we predict it as Y = 1.
    n_correct = 0
    n_incorrect = 0

    f = np.zeros(Y.shape)
    index = 0
    for i,j in zip(Xj,Y):
         # TODO: ******** To be filled ********
         # Check if result is above threshold
         # and then check if prediction is correct or incorrect
        if(i > thres):
            f[index] = 1
        if(f[index] == j):
            n_correct = n_correct + 1
        else:
            n_incorrect = n_incorrect + 1
        index = index + 1



    # Step 2. Calculate the accuracy.
    acc = 1.0 * n_correct / (n_correct + n_incorrect)

    return acc
```

```python
    # Plot the histograms and the best decision stump in current feature.
    plt.subplot(2, 2, j+1)
    plt.hist(Xj_when_Y0_train, label='Feature {}, Y=0'.format(j))
    plt.hist(Xj_when_Y1_train, label='Feature {}, Y=1'.format(j))
    plt.plot([current_thres, current_thres], [0, 10])
    plt.text(current_thres, 10, 'training acc: {}'.format(current_max_acc))
    plt.xlabel('Feature {}: {}'.format(j, iris.feature_names[j]))
    plt.ylabel('Frequency')
    plt.legend()
plt.show()
```



In [43]:

```python
# Use the best feature and best threshold on test set.

Xj_test = X_test[:, all_feature] # Array of best feature.
test_acc = calc_acc(Xj_test, Y_test, all_thres)
print('Best feature: {}'.format(all_feature))
print('Best threshold: {:.2f}'.format(all_thres))
print('Training accuracy of best feature: {:.2f}'.format(all_max_acc))
print('Test accuracy of best feature: {:.2f}'.format(test_acc))
```

```
Best feature: 3
Best threshold: 1.70
Training accuracy of best feature: 0.99
Test accuracy of best feature: 0.90
```

In [ ]:

```python

```