DS 4023 25F
Assignment 1

October 6, 2025

Bohan YANG
2330016056
Section 1001

---

1. Let $\mathbf{x}, \mathbf{c} \in \mathbb{R}^n$ and $\mathbf{A} \in \mathbb{R}^{n \times n}$. For the following parts, before taking any derivatives, identify what the derivative looks like (is it a scalar, vector, or matrix?) and how we calculate each term in the derivative. Then carefully solve for an arbitrary entry of the derivative, then stack/arrange all of them to get the final result.

   Note that the convention we will use going forward is that vector derivatives of a scalar (with respect to a column vector) are expressed as a **row vector**, i.e.

$$\frac{\partial f}{\partial \mathbf{x}} = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_n} \right),$$

   since a row acting on a column gives a scalar. You may have seen alternative conventions before, but the important thing is that you need to understand the types of objects and how they map to the shapes of the multidimensional arrays we use to represent those types.

   (a) Show that
$$\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^\top \mathbf{c}) = \mathbf{c}^\top$$

   (b) Show that
$$\frac{\partial}{\partial \mathbf{x}}\|\mathbf{x}\|_2^2 = 2\mathbf{x}^\top$$

   (c) Show that
$$\frac{\partial}{\partial \mathbf{x}}(\mathbf{A}\mathbf{x}) = \mathbf{A}$$

   (d) Show that
$$\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^\top \mathbf{A}\mathbf{x}) = \mathbf{x}^\top(\mathbf{A} + \mathbf{A}^\top)$$

   (e) Under what condition is the previous derivative equal to $2\mathbf{x}^\top \mathbf{A}$? (15pt)

Ans:

(a) Show $\dfrac{\partial}{\partial x}(x^\top c) = c^\top$.

Type/shape. $x^\top c$ is a scalar; its derivative w.r.t. $x \in \mathbb{R}^n$ is a $1 \times n$ row vector.

Arbitrary entry. Write $x^\top c = \sum_{i=1}^n x_i c_i$. Then for $j \in \{1, \dots, n\}$,

$$\frac{\partial}{\partial x_j}(x^\top c) = \sum_{i=1}^n c_i \frac{\partial x_i}{\partial x_j} = \sum_{i=1}^n c_i \, \delta_{ij} = c_j.$$

Stacking. Collecting entries $j = 1, \dots, n$ into a row,

$$\frac{\partial}{\partial x}(x^\top c) = \begin{bmatrix} c_1, \dots, c_n \end{bmatrix} = c^\top.$$

(b) Show $\dfrac{\partial}{\partial x}\|x\|_2^2 = 2x^\top$.

Type/shape. $\|x\|_2^2 = x^\top x$ is a scalar; derivative is a $1 \times n$ row vector.

Arbitrary entry. $x^\top x = \sum_{i=1}^n x_i^2$. Then

$$\frac{\partial}{\partial x_j}(x^\top x) = 2x_j.$$

Stacking. Hence

$$\frac{\partial}{\partial x}\|x\|_2^2 = \begin{bmatrix} 2x_1, \dots, 2x_n \end{bmatrix} = 2x^\top.$$

(c) Show $\dfrac{\partial}{\partial x}(Ax) = A$.

Here we may allow $A \in \mathbb{R}^{n \times n}$ and $x \in \mathbb{R}^n$, so $Ax \in \mathbb{R}^n$.

Type/shape. The derivative of a vector-valued function Ax w.r.t. $x \in \mathbb{R}^n$ is its Jacobian, an $n \times n$ matrix.

Arbitrary entry. The i-th component of Ax is $(Ax)_i = \sum_{k=1}^n A_{ik} x_k$. Then for $j \in \{1, \dots, n\}$,

$$\frac{\partial (Ax)_i}{\partial x_j} = \sum_{k=1}^n A_{ik} \frac{\partial x_k}{\partial x_j} = \sum_{k=1}^n A_{ik} \, \delta_{kj} = A_{ij}.$$

Stacking. The Jacobian has entries $J_{ij} = A_{ij}$, so

$$\frac{\partial}{\partial x}(Ax) = A.$$

(d) Show $\dfrac{\partial}{\partial x}(x^\top Ax) = x^\top(A + A^\top)$.

Type/shape. $x^\top Ax$ is a scalar; derivative is a $1 \times n$ row vector.

Arbitrary entry. Expand $x^\top Ax = \sum_{i=1}^n \sum_{k=1}^n x_i A_{ik} x_k$. For a fixed j,

$$\frac{\partial}{\partial x_j}(x^\top Ax) = \sum_{i,k} \frac{\partial}{\partial x_j}\big(x_i A_{ik} x_k\big) = \sum_{i,k}\Big(\delta_{ij} A_{ik} x_k + x_i A_{ik}\,\delta_{kj}\Big).$$

Evaluate the sums:

$$\sum_{i,k} \delta_{ij} A_{ik} x_k = \sum_k A_{jk} x_k = (Ax)_j, \qquad \sum_{i,k} x_i A_{ik} \delta_{kj} = \sum_i x_i A_{ij} = (A^\top x)_j.$$

Hence

$$\frac{\partial}{\partial x_j}(x^\top Ax) = (Ax)_j + (A^\top x)_j = \big((A + A^\top)x\big)_j.$$

Stacking. Using the row-gradient convention,

$$\frac{\partial}{\partial x}(x^\top Ax) = \big((A + A^\top)x\big)^\top = x^\top(A + A^\top).$$

(e) When is the previous derivative equal to $2x^\top A$?

We have $\dfrac{\partial}{\partial x}(x^\top Ax) = x^\top(A + A^\top)$. This equals $2x^\top A$ for all x if and only if $A + A^\top = 2A$, i.e.,

$$A = A^\top \quad (\,A \text{ symmetric})\,.$$

2. Assume the probability of a certain disease is **0.01**. The probability of test positive given thata person is infected with the disease is **0.95** and the probability of test positive given theperson is not infected with the disease is **0.05**.
   (a) Calculate the probability of test positive.
   (b) Use Bayes' Rule to calculate the probability of being infected with the disease given that the test is positive. (10pt)

Ans:

Let:

$$P(D) = 0.01, \quad P(\neg D) = 0.99,$$

$$P(+|D) = 0.95, \quad P(+|\neg D) = 0.05,$$

where:

- D = event that a person *has* the disease,
- $\neg D$ = event that a person *does not have* the disease,
- $(+)$ = event that the test result is *positive*.

(a) Probability of testing positive.

We apply the **law of total probability**:

$$P(+) = P(+|D)P(D) + P(+|\neg D)P(\neg D).$$

Substitute the given values:

$$P(+) = (0.95)(0.01) + (0.05)(0.99).$$

Compute step-by-step:

$$P(+) = 0.0095 + 0.0495 = 0.059.$$

$$\boxed{P(+) = 0.059.}$$

(b) Probability of being infected given that the test is positive.

We apply **Bayes' Rule**:

$$P(D|+) = \frac{P(+|D)P(D)}{P(+)}.$$

Substitute known quantities:

$$P(D|+) = \frac{(0.95)(0.01)}{0.059}.$$

Calculate:

$$P(D|+) = \frac{0.0095}{0.059} \approx 0.161.$$

$$\boxed{P(D|+) \approx 0.161.}$$

3. Gradient descent is the primary algorithm to search optimal parameters for our models. Typically, we want to solve optimization problems stated as

$$\min_{\theta \in \mathbb{R}^d} \mathcal{L}(f_\theta, \mathcal{D})$$

where $\mathcal{L}$ are differentiable functions.

In this example, we look at a simple supervised learning problem where, given a dataset

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N,$$

we want to find the optimal parameters θ that minimize some loss. We consider different models for learning the mapping from input to output, and examine the behavior of gradient descent for each model.

(a) The simplest parametric model entails learning a single-parameter constant function, where we set

$$\hat{y}_i = \theta.$$

We wish to find

$$\theta^\star = \arg\min_{\theta \in \mathbb{R}} \mathcal{L}(f_\theta, \mathcal{D}) = \arg\min_{\theta \in \mathbb{R}} \frac{1}{N} \sum_{i=1}^N (y_i - \theta)^2.$$

(i) What is the gradient of $\mathcal{L}$ with respect to θ?

(ii) What is the optimal value of θ?

(iii) Write the gradient descent update.

(iv) Stochastic Gradient Descent (SGD) is an alternative optimization algorithm, where instead of using all N samples, we use a single sample per optimization step to update the model. What is the contribution of each data-point to the full gradient update?

(b) Instead of constant functions, we now consider a single-parameter linear model

$$\hat{y}_i(x_i) = \theta x_i$$

where we search for θ such that

$$\theta^\star = \arg\min_{\theta \in \mathbb{R}} \frac{1}{N} \sum_{i=1}^N (y_i - \theta x_i)^2.$$

(i) What is the gradient of $\mathcal{L}$ with respect to θ?

(ii) What is the optimal value of θ?

(iii) Write the gradient descent update.

(iv) Do all points get the same vote in the update? Why or why not?

(20pt)

Ans:

(a) Constant model $\hat{y}_i = \theta$

Loss:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \theta)^2.$$

1. **Gradient $\partial\mathcal{L}/\partial\theta$.**

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{1}{N}\sum_{i=1}^{N} 2(\theta - y_i) = \frac{2}{N}\Big(N\theta - \sum_{i=1}^{N} y_i\Big) = 2\Big(\theta - \bar{y}\Big),$$

where $\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$.

2. **Optimal $\theta^\star$.** Set the gradient to zero:

$$2(\theta^\star - \bar{y}) = 0 \implies \boxed{\theta^\star = \bar{y}}.$$

3. **Gradient descent (GD) update.** For learning rate $\eta > 0$,

$$\boxed{\theta_{t+1} = \theta_t - \eta\,\frac{\partial \mathcal{L}}{\partial \theta}(\theta_t) = \theta_t - 2\eta\big(\theta_t - \bar{y}\big)}.$$

Equivalently, $\theta_{t+1} = (1 - 2\eta)\theta_t + 2\eta\,\bar{y}$ (a contraction toward the sample mean when $0 < \eta < 1$).

4. **Per-sample contribution to the full gradient & SGD.** The full gradient decomposes as

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{1}{N}\sum_{i=1}^{N} \underbrace{2(\theta - y_i)}_{\text{per-sample term}}.$$

Thus the contribution of data point i to the *full* gradient is

$$\boxed{\frac{2}{N}(\theta - y_i)}.$$

In SGD with a single sampled index i_t, the stochastic gradient is $2(\theta - y_{i_t})$, yielding the SGD step $\theta_{t+1} = \theta_t - 2\eta\,(\theta_t - y_{i_t})$.

(b) One-parameter linear model $\hat{y}_i = \theta x_i$

Loss:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \theta x_i)^2.$$

1. **Gradient $\partial\mathcal{L}/\partial\theta$.** By the chain rule,

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{1}{N}\sum_{i=1}^{N} 2(\theta x_i - y_i)\, x_i = \frac{2}{N}\sum_{i=1}^{N} x_i(\theta x_i - y_i).$$

2. **Optimal $\theta^\star$.** Set the gradient to zero:

$$\sum_{i=1}^{N} x_i(\theta^\star x_i - y_i) = 0 \implies \theta^\star \sum_{i=1}^{N} x_i^2 = \sum_{i=1}^{N} x_i y_i \implies \boxed{\theta^\star = \frac{\sum_{i=1}^{N} x_i y_i}{\sum_{i=1}^{N} x_i^2}}.$$

(Assumes not all $x_i = 0$, so $\sum x_i^2 > 0$.)

3. **Gradient descent (GD) update.**

$$\boxed{\theta_{t+1} = \theta_t - \eta\,\frac{\partial \mathcal{L}}{\partial \theta}(\theta_t) = \theta_t - \eta\,\frac{2}{N}\sum_{i=1}^{N} x_i\big(\theta_t x_i - y_i\big)}.$$

In SGD with a single index i_t: $\theta_{t+1} = \theta_t - 2\eta\, x_{i_t}(\theta_t x_{i_t} - y_{i_t})$.

4. **Do all points get the same vote? Why/why not?**
No. The per-point contribution to the *full* gradient is

$$\boxed{\frac{2}{N}\, x_i(\theta x_i - y_i)}.$$

Its magnitude scales with $|x_i|$ (and with the residual $|\theta x_i - y_i|$). Points with larger $|x_i|$ exert larger leverage on the update; points with $x_i = 0$ contribute nothing. The sign is determined by the product $x_i(\theta x_i - y_i)$, pushing θ to reduce the residual on that sample.

4. Consider the Ridge Regression estimator

$$\arg\min_{\mathbf{w}} \; \|X\mathbf{w} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$$

(Hint: here X is the feature matrix, equivalent to $X^\top$ in our slides.)

We know this is solved by

$$\hat{\mathbf{w}} = (X^\top X + \lambda I)^{-1} X^\top \mathbf{y}.$$

One interpretation of Ridge Regression is to find the **Maximum A Posteriori (MAP)** estimate on $\mathbf{w}$, the parameters, assuming that the prior of $\mathbf{w}$ is

$$\mathbf{w} \sim \mathcal{N}(0, I)$$

and that the random variable $\mathbf{Y}$ is generated using

$$\mathbf{Y} = X\mathbf{w} + \sqrt{\lambda}\,\mathbf{N},$$

where each entry of the vector $\mathbf{N}$ is zero-mean, unit-variance normal.

Show that

$$\hat{\mathbf{w}} = (X^\top X + \lambda I)^{-1} X^\top \mathbf{y}$$

is indeed the MAP estimate for $\mathbf{w}$ given an observation on $\mathbf{Y} = \mathbf{y}$.

(20pt)

Ans:

Posterior up to a constant. By Bayes' rule,

$$p(\mathbf{w} \mid \mathbf{y}) \;\propto\; p(\mathbf{y} \mid \mathbf{w})\, p(\mathbf{w}).$$

Using the Gaussian forms,

$$p(\mathbf{y} \mid \mathbf{w}) \propto \exp\left(-\frac{1}{2\lambda} \|\mathbf{y} - X\mathbf{w}\|_2^2 \right), \qquad p(\mathbf{w}) \propto \exp\left(-\frac{1}{2} \|\mathbf{w}\|_2^2 \right).$$

Hence

$$p(\mathbf{w} \mid \mathbf{y}) \propto \exp\left(-\frac{1}{2\lambda} \|\mathbf{y} - X\mathbf{w}\|_2^2 - \frac{1}{2} \|\mathbf{w}\|_2^2 \right).$$

MAP as a minimizer. Maximizing the posterior is equivalent to minimizing the negative log-posterior:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \left\{ \frac{1}{2\lambda} \|\mathbf{y} - X\mathbf{w}\|_2^2 + \frac{1}{2} \|\mathbf{w}\|_2^2 \right\}.$$

Multiplying the objective by the positive constant 2λ leaves the minimizer unchanged:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \left\{ \|\mathbf{y} - X\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2 \right\},$$

which is exactly the ridge regression objective.

Normal equations and closed form. Differentiate and set to zero:

$$\nabla_{\mathbf{w}}\big(\|\mathbf{y} - X\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2\big) = -2X^\top(\mathbf{y} - X\mathbf{w}) + 2\lambda\mathbf{w} = \mathbf{0}.$$

Rearrange:

$$(X^\top X + \lambda I)\,\mathbf{w} = X^\top\mathbf{y}.$$

Since $X^\top X + \lambda I$ is positive definite for $\lambda > 0$, it is invertible, giving the unique solution

$$\boxed{\hat{\mathbf{w}} = (X^\top X + \lambda I)^{-1}X^\top\mathbf{y}}.$$

2330016056

October 6, 2025

1 Lab 1 Essential Libraries and Tools

Task 1: In the code cell below, complete the function `insertSecond` which takes two arguments, array `a` and element `b`, and returns an array which `b` is inserted before the second element in `a`.

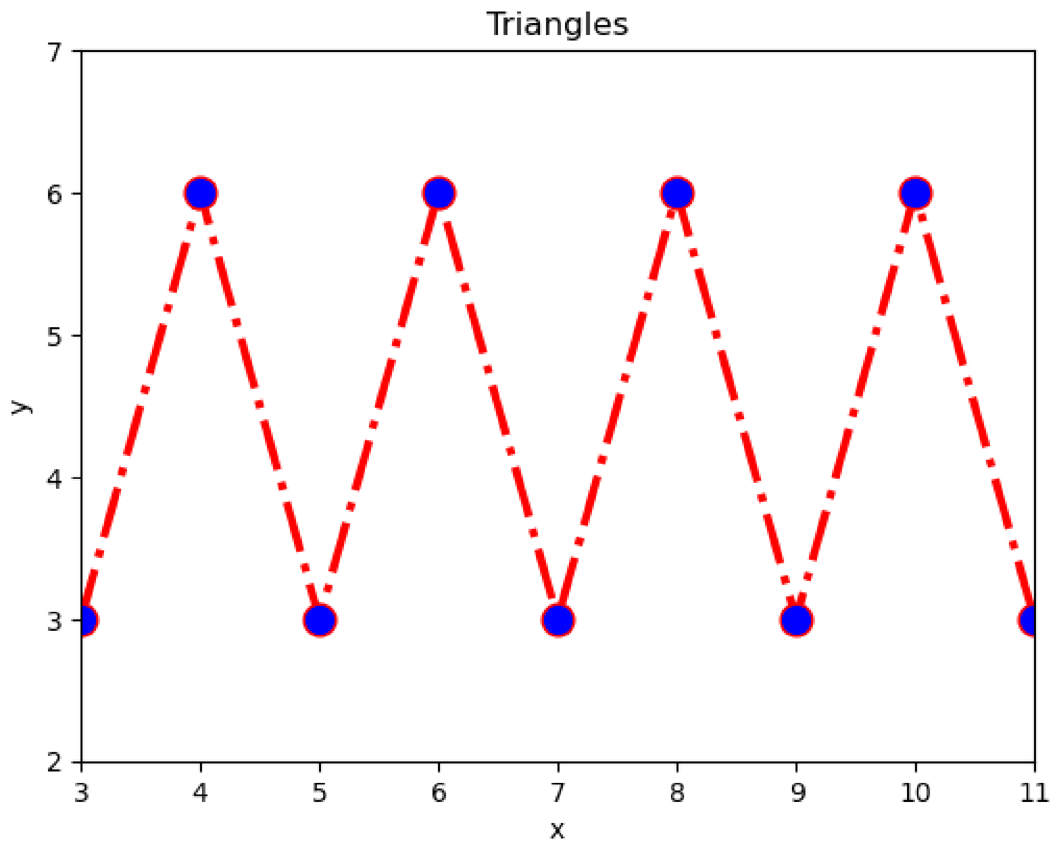
[9]:
```python
# You should return your result.
import numpy as np


def insertSecond(a, b):
    res = a.tolist()
    res.insert(1, b)
    return np.array(res)


# Test cases
assert np.array_equal(insertSecond(
    np.array([-5, -10, -12, -6]), 5), np.array([-5, 5, -10, -12, -6]))
assert np.array_equal(insertSecond(
    np.array([1, 2, 3]), 7), np.array([1, 7, 2, 3]))
assert np.array_equal(insertSecond(
    np.array([-5, -10, -12, -6]), 8), np.array([-5, 8, -10, -12, -6]))
assert np.array_equal(insertSecond(
    np.array([1, 2, 3]), 12), np.array([1, 12, 2, 3]))
```

Task 2: In the code cell below, complete the function `mergeArrays` takes two array arguments `a` and `b`. The two arrays are merged, in which process the duplicate elements are removed. Finally the merged array are sorted.

[8]:
```python
import numpy as np


def mergeArrays(a, b):
    res = np.concatenate((a, b))
    res = np.unique(res)
    return res
```

```
# Test cases
assert np.array_equal(mergeArrays(
    np.array([1, 1, 4, 8, 1]), np.array([2, 3])), np.array([1, 2, 3, 4, 8]))
assert np.array_equal(mergeArrays(np.array(
    [-5, -10, -10, -6]), np.array([-5, 8, -10, -12, -6])), np.array([-12, -10,␣
 ↪-6, -5, 8]))
assert np.array_equal(mergeArrays(
    np.array([1, 1, 6, 8, 1]), np.array([2, 3])), np.array([1, 2, 3, 6, 8]))
```

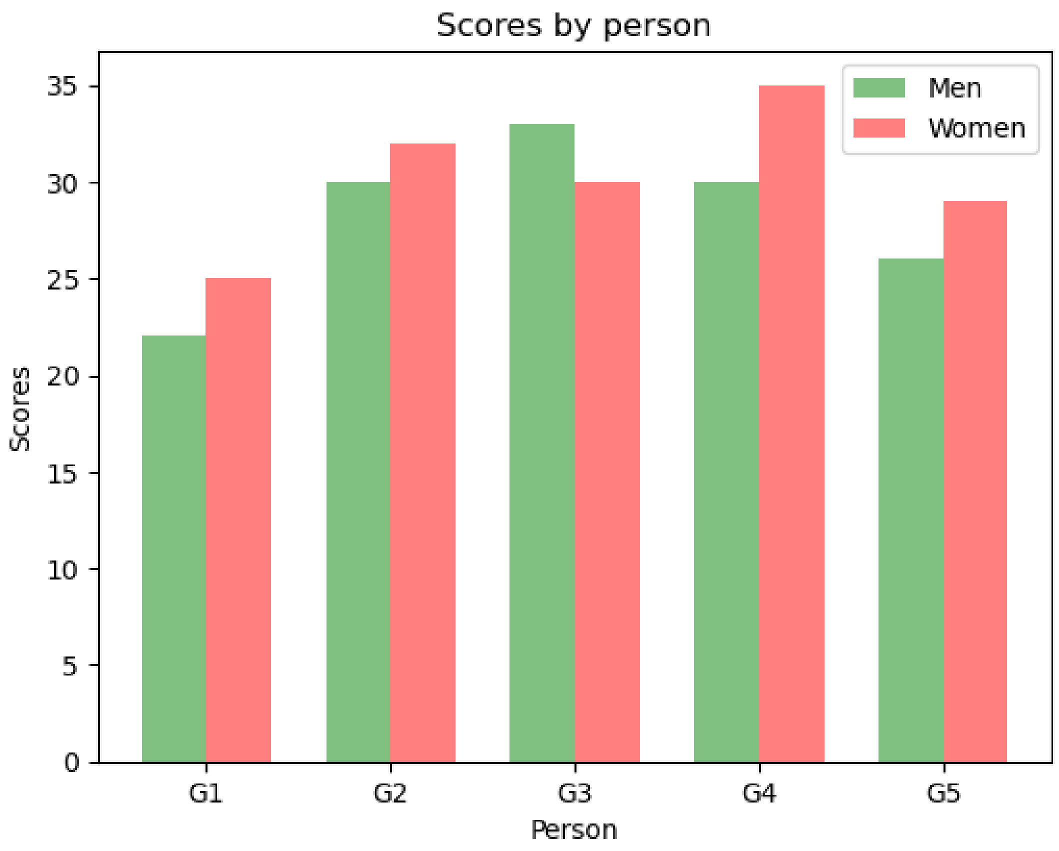
[1]:
```
import matplotlib.pyplot as plt

# x axis values
x = [3, 4, 5, 6, 7, 8, 9, 10, 11]

# y axis values
y = [3, 6, 3, 6, 3, 6, 3, 6, 3]

# plotting the points
plt.plot(
    x, y,
    color='red',           # line color
    linestyle='dashdot',   # line style
    linewidth=3,           # line thickness
    marker='o',            # circular markers
    markerfacecolor='blue',   # marker fill color
    markersize=12          # marker size
)

# Set the y-limits of the current axes
plt.ylim(2, 7)

# Set the x-limits of the current axes
plt.xlim(3, 11)

# naming the x axis
plt.xlabel('x')

# naming the y axis
plt.ylabel('y')

# giving a title to the graph
plt.title('Triangles')

# display the plot
plt.show()
```



Task 3: Complete the code below to create a image which looks exactly the same as the follow one:

[3]:
```
import numpy as np
import matplotlib.pyplot as plt

n_groups = 5
men_means = (22, 30, 33, 30, 26)
women_means = (25, 32, 30, 35, 29)
alpha = 0.5

fig, ax = plt.subplots()

index = np.arange(n_groups)
bar_width = 0.35

rects1 = plt.bar(index, men_means, bar_width,
                 alpha=alpha, color='g', label='Men')
rects2 = plt.bar(index + bar_width, women_means, bar_width,
                 alpha=alpha, color='r', label='Women')
```

```
plt.xlabel('Person')
plt.ylabel('Scores')
plt.title('Scores by person')
plt.xticks(index + bar_width / 2, ('G1', 'G2', 'G3', 'G4', 'G5'))
plt.legend()

plt.show()
```



Task 4: In the code cell below, complete the function `setDataFrameZeros` which takes a dataFrame `df` as an arguement, and returns a new dataFrame. The label and index in the new dataFrame is the same as those in `df` . If an element is 0 in `df` , set its entire row and column to 0 in the new dataFrame. All other elements are the same as those in `df` .

[7]:
```
import pandas as pd


def setDataFrameZeros(df):
    res = df.copy()
    res.loc[df.index[df.eq(0).any(axis=1)], :] = 0
    res.loc[:, df.columns[df.eq(0).any(axis=0)]] = 0
    return res


# Test cases
df1 = pd.DataFrame({'c1': [1, 4, 7], 'c2': [2, 0, 8], 'c3': [3, 6, 9]})
df2 = pd.DataFrame({'c1': [1, 0, 7], 'c2': [0, 0, 0], 'c3': [3, 0, 9]})
assert (df2.equals(setDataFrameZeros(df1)))

df1 = pd.DataFrame({'c1': [0, 3, 1], 'c2': [1, 4, 3],
                    'c3': [2, 5, 1], 'c4': [0, 2, 5]})
df2 = pd.DataFrame({'c1': [0, 0, 0], 'c2': [0, 4, 3],
                    'c3': [0, 5, 1], 'c4': [0, 0, 0]})
assert (df2.equals(setDataFrameZeros(df1)))
```

[ ]:

```

```