CAP 6619 Deep Learning

2024 Summer

Question 1 [1 pt] Show artificial neuron (perceptron) structure and explain function of each component, including inputs, weights, summing function, activation function, and output [1 pt].

Perceptron is a single layer neural network that is used for binary classification.

a. **Input:** It is the features or attributes of the input data.

b. **Weights:** This is the value associated to the input which determines the significance of each input to correctly predict the output of the perceptron. It gets updated using backpropagation algorithm.

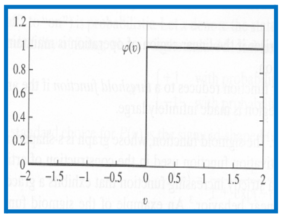
c. **Summing Function:** It calculates the weighted sum of the input features and the weight associated with it.

d. **Activation Function:** The output of the summing function is passed through another function, namely, Activation function, to bound the input.

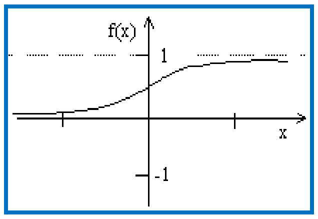
e. **Output:** It is the output value of the activation function.

f. **Bias:** Bias is a term added to the summing function irrespective of the input. This is is done to make adjustments independent of the input.

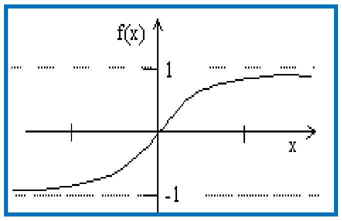
Question 2 [1pt] Figure 1 shows four activation function in the neural network, please show the mathematical formulation of each activation function [0.5], and explain the characteristics of each activation function [0.5 pt].



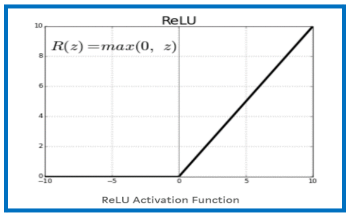
(a)



(b)



(c)



(d)

Figure 1 Activation Functions.

(a) **Hard Limiter:** It is the simplest activation function, where the output is 1 if the input is greater than 0 or else its 0.

f(x) = 0 if $x<0$

f(x) = 1 if $x>0$

(b) **Sigmoid Function:** It is a continuous non linear transformation which transforms an output bounded between 0 and 1.

$f(x) = 1/ (1 + e^{-x})$

(c) **tanh or hyperbolic tangent:** It is quite similar to sigmoid function but the range is between -1 and 1.

$f(x) = (e^{x} - e^{-x}) / (e^{x} + e^{-x})$

(d) **ReLU:** This is the most common activation function in neural networks, the range is from 0 to infinity. The output is 0 for inputs less than 0.

$f(x) = \max(0, x)$

Question 3 [1 pt]. Drive gradient of Sigmoid activation function Figure 1(b) and the gradient of the ReLU activation function Figure1(d) [0.5pt]. Explain advantage vs. disadvantage of each of the activation function, respectively [0.5 pt].

Sigmoid Function:

Advantages:

1. Continuous and differentiable
2. Bounded

Disadvantages:

1. Vanishing Gradient Problem
2. Bounded

Gradient of Sigmoid Function

Derivative of ~~sigma~~ sigmoid function

$$\frac{d}{dx} \frac{1}{1+e^{-x}}$$

where, $\sigma(x) = \frac{1}{1+e^{-x}}$

$$\Rightarrow \frac{d}{dx}(1+e^{-x})^{-1}$$

$$\Rightarrow -(1+e^{-x})^{-2} \cdot \frac{d}{dx}(1+e^{-x})$$

$$\Rightarrow -(1+e^{-x})^{-2}(-e^{-x})$$

$$\Rightarrow e^{-x} \frac{1}{(1+e^{-x})^2}$$

$$\Rightarrow \frac{e^{-x}}{1+e^{-x}} \cdot \frac{1}{1+e^{-x}}$$

$$\Rightarrow \frac{1}{1+e^{-x}}\left(\frac{1-1+e^{-x}}{1+e^{-x}}\right)$$

$$\Rightarrow \sigma(x)(1-\sigma(x)$$

ReLU Function:

Advantages:

1. Consistent Gradient
2. Computationally efficient
3. Sparsity

Disadvantages:

1. Unbounded

Derivative of ReLu

$$\text{relu}(x) = \max(0, x)$$

$$= \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$

$$\frac{d}{dx} \begin{cases} \frac{d}{dx} 0, & x \leq 0 \\ \frac{d}{dx} x, & x \geq 0 \end{cases}$$

$$\Rightarrow \frac{d}{dx} \begin{cases} 0, & x < 0 \\ \underline{1}, & x \geq 0 \end{cases}$$

Question 4 [2 pts] Figure 2 shows a set of samples (dots) which are labeled as red and green (red dots belong to class C2, and green dots belong to class C1).

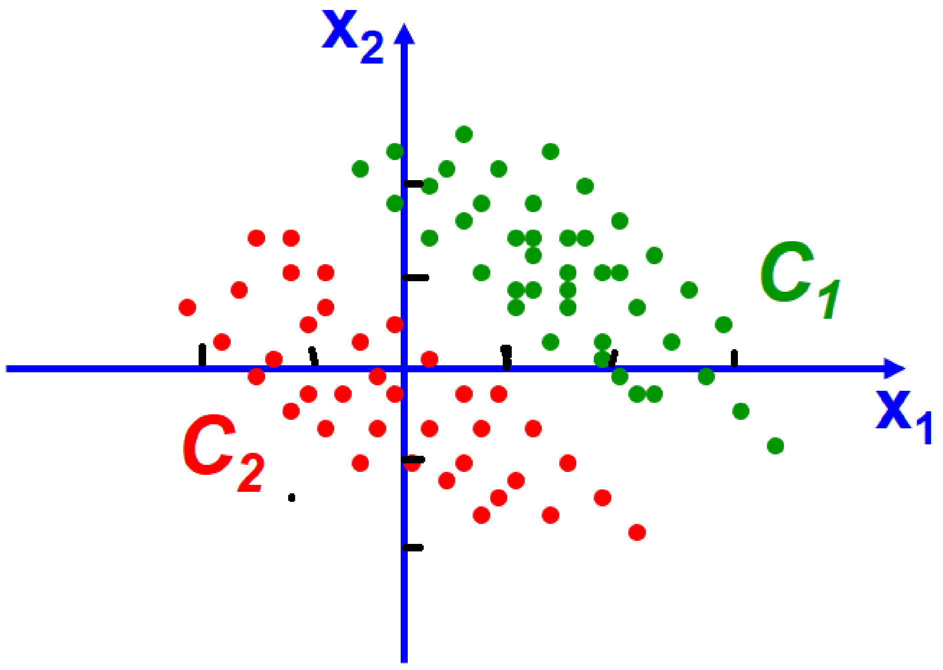


Figure 2. Examples of a linearly separable classification task with two feature dimensions

- What are the roles of the weight values of the neuron [0.5 pt]

  The weights define the significance of the input features. An important feature will have a higher weight than one with low importance.

- Assume a neuron with weight values [w_0, w_1, w_2] is used to learn decision surface to separate the two group instances (w_0 is the weight value for bias), Draw decision surfaces corresponding to [0, 1, 1], [0, 1, -1], and [-1, 1, 1], respectively (mark each line on the plot) [0.5 pt]

For $[0, 1, 1]$,

$0 + x_1 + x_2 = 0$

$\Rightarrow x_1 + x_2 = 0$

So, if $x_2 = 0$, $x_1 = 0$
$x_1 = 0$, $x_2 = 0$

Again, for $[0, 1, -1]$,

$0 + x_1 - x_2 = 0$

$\Rightarrow x_1 - x_2 = 0$

So, if $x_2 = 0$, $x_1 = 0$

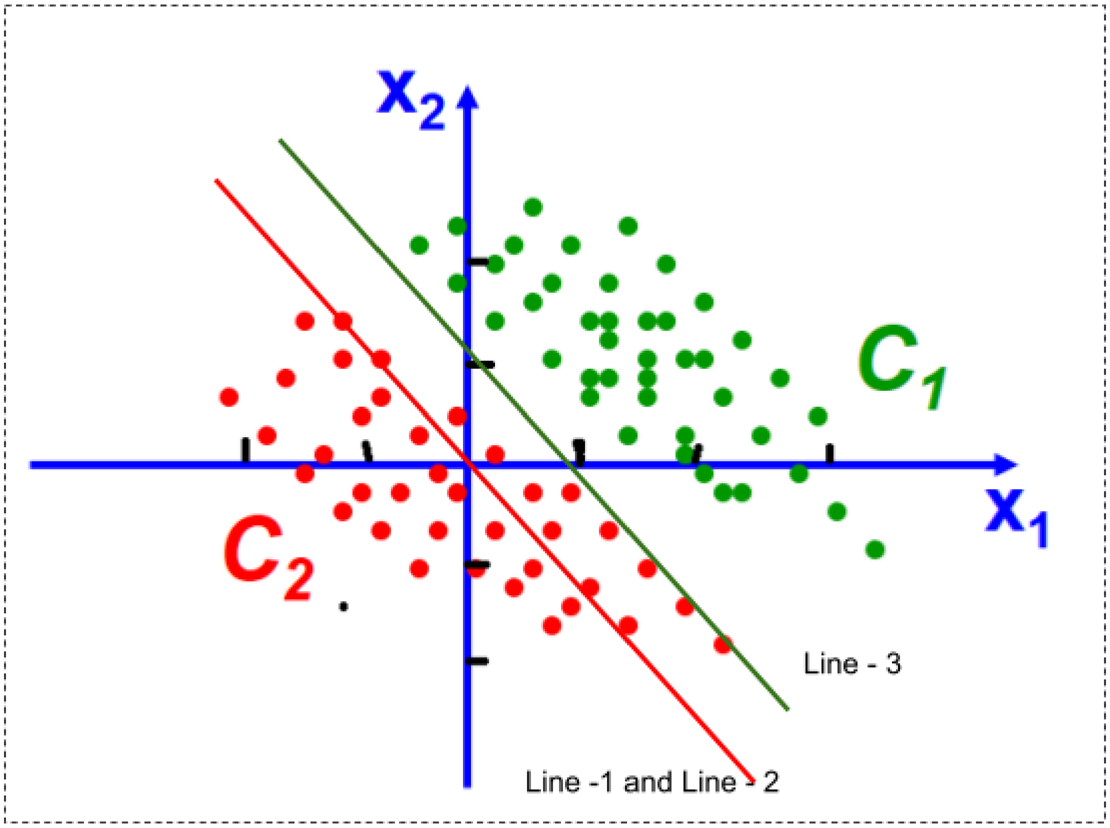
$x_1 = 0$, $x_2 = 0$

For $[-1, 1, 1]$

$-1 + x_1 + x_2 = 0$

$\Rightarrow x_1 + x_2 = 1$

So, if $x_1 = 0$, $x_2 = 1$
$x_2 = 0$, $x_1 = 1$



- Explain how does each weight values [w_0, w_1, w_2] control the decision surface, respectively [0.5 pt]

  The weight and bias values are the only variable in the summing function. So, they can be changed to get the best decision surface.

- Among the three decision surfaces, which line is the best decision surface to separate instances, why? [0.5 pt]

  The line 3 is the best since it classifies the most points accurately.

Question 5 [1 pt] Figure 3 shows a single layer neural network with three weight values (including bias). Given a training instance x(n), assume desired label of the instance is d(n),

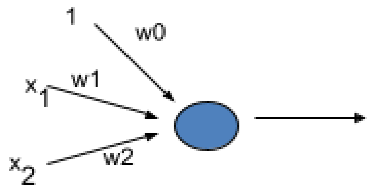


Figure 3: Single layer neural network

1. Define squared error of the instance with respect to the network [0.5 pt].

   Here, o(n) = w0(n) + x1(n) * w1(n) + x2(n) * w2(n)
   e = ∑ (o(n) - d(n))

   E = $(e)^2$ / 2
   = (∑ ((w0 + x1 * w1 + x2 * w2) - d(n)))^2 / 2

2. Use gradient descent learning to derive weight updating rules for w_0, w_1, and w_2, respectively [0.5 pt]

   w0(k + 1) = w0(k) + learning rate * (∑ ((w0 + x1 * w1 + x2 * w2) - d(n))) * 1

   w0(k + 1) = w1(k) + learning rate * (∑ ((w0 + x1 * w1 + x2 * w2) - d(n))) * x1(n)

   w2(k + 1) = w2(k) + learning rate * (∑ ((w0 + x1 * w1 + x2 * w2) - d(n))) * x2(n)

Question 6 [1 pt]: The following figure shows a quadratic function $y=2x^2-4x+1$. Assume we are at the point x=-1, and are searching for the next movement to find the minimum value of the quadratic function using gradient descent (the learning rate is 0.1).

- What is the gradient at point x=-1? (Show your solutions) [0.5 pt]

  $y = 2x^2-4x+1$

  dy/dx = 4x - 4

At x = -1,

dy/dx = -8

- Following gradient descent principle, find the next movement towards the global minimum [0.5 pt]

  dy/dx = 0

  4x - 4 = 0

  x = 1

  For x = 0.5, dy/dx = -2

  For x = 1.5, dy/dx = 2

  The slope is moving from negative to positive.

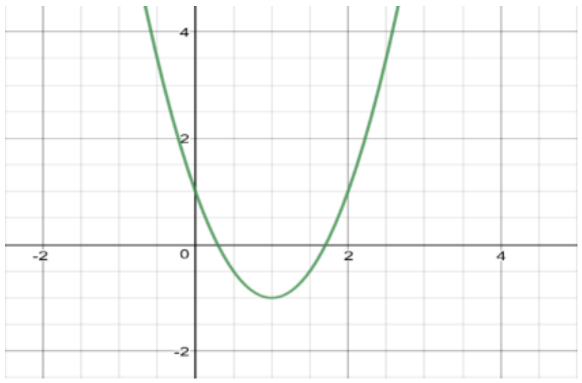


Figure 4. A quadratic function

Question 7 [1.5 pts] Assuming we have two sets of instances, which belong to two classes, with each class containing three instances. C1={(1, 0), (1, 1), (0, -1)}; C2={(0, 1), (-1, 0), (-1, -1)}. Assuming the class labels for C1 and C2 are 1 and 0, respectively, the learning η=0.1, and the initial weights are w_0=1, w_1=1, and w_2=1. Please use gradient learning rule to learn a linear decision surface to separate the two classes. List the results in the first two rounds by using tables in the following form (Report the mean squared errors of all instances with respect to the initial weight values, and the mean squared errors E(W) AFTER the weight updating for each round).

Mean squared errors E(W) corresponding to the initial weights:

First Round

Input	Weight	Desired	Output	v	Δw
(1,1,0)	1, 0, 0	1	1	0, 0, 0	0, 0, 0
(1,-1,0)	1, 0, 0	0	1	-0.1, 0.1, 0	-0.1, 0.1, 0
(1,0,-1)	1, 0, 0	1	1	0, 0, 0	-0.1, 0.1, 0
(1,0,1)	1, 0, 0	0	1	-0.1, 0, -0.1	-0.2, 0.1, -0.1
(1,1,1)	1, 0, 0	1	1	0, 0 ,0	-0.2, 0.1, -0.1
(1,-1,-1)	1, 0, 0	0	1	-0.1, 0.1, 0.1	-0.3, 0.2, 0

New weight after first round: 0.7, 0.2, 0

Mean squared errors E(W) after the first-round weight updating: 3/2

Second Round

Input	Weight	Desired	Output	v	Δw
(1,1,0)	0.7, 0.2, 0	1		0.01, 0.01, 0	0.01, 0.01, 0
(1,-1,0)	0.7, 0.2, 0	0	1	-0.05, 0.05, 0	-0.04, 0.06, 0
(1,0,-1)	0.7, 0.2, 0	1	1	0.03, 0, -0.03	-0.01, 0.06, -0.03
(1,0,1)	0.7, 0.2, 0	0	1	-0.07, 0, -0.07	-0.08, 0.06, -0.10
(1,1,1)	0.7, 0.2, 0	1	1	0.01, 0.01, 0.01	-0.07, 0.07, -0.09
(1,-1,-1)	0.7, 0.2, 0	0	1	-0.05, 0.05, 0.05	-0.12, 0.12, -0.04

New weight after second round: 0.58, 0.32, -0.04

Mean squared errors E(W) after the second-round weight updating: 0.55

Question 8 [1.5 pts] Assuming we have two sets of instances, which belong to two classes, with each class containing three instances. C1={(1, 0), (1, 1), (0, -1)}; C2={(0, 1), (-1, 0), (-1, -1)}. Assuming the class label for C1 and C2 are 1 and 0, respectively, the learning rate η=0.1, and the initial weights are w_0=1, w_1=1, and w_2=1. Use Delta rule (AdaLine) to learn a linear decision surface to separate the two classes. List the results in the first round by using tables in the following form (Please report the mean squared errors of all instances with respect to the initial weight values, and also report the mean squared errors E(W) AFTER the weight updating of the last instance).

Input	Weight	Desired	Output	v	Δw
(1,1,0)	1, 1, 1	1	2	-0.1, -0.1, 0	1, 1, 1
(1,-1,0)	1, 1, 1	0	0	0, 0, 0	1, 1, 1
(1,0,-1)	1, 1, 1	1	0	0.1, 0, -0.1	1.10, 1, 0.90
(1,0,1)	1.10, 1, 0.90	0	2	-0.2, 0, -0.2	0.90, 1, 0.70
(1,1,1)	0.90, 1, 0.70	1	2.60	-0.16, -0.16, -0.16	0.74, 0.84, 0.54
(1,-1,-1)	0.74, 0.84, 0.54	0	-0.64	0.064, -0.064, -0.064	0.804, 0.776, 0.476

Mean squared errors E(W) after the weight updating of the last instance: 4.4848

In [1]:
```
import pandas as  pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import math
import tensorflow as tf
import random
```

```
2024-05-27 23:16:05.198665: I tensorflow/core/platform/cpu_feature_guard.cc:
193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Lib
rary (oneDNN) to use the following CPU instructions in performance-critical
operations:  AVX2 AVX_VNNI FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
2024-05-27 23:16:05.289732: I tensorflow/core/util/port.cc:104] oneDNN custo
m operations are on. You may see slightly different numerical results due to
floating-point round-off errors from different computation orders. To turn t
hem off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2024-05-27 23:16:05.665160: W tensorflow/compiler/xla/stream_executor/platfo
rm/default/dso_loader.cc:64] Could not load dynamic library 'libnvinfer.so.7
'; dlerror: libnvinfer.so.7: cannot open shared object file: No such file or
directory; LD_LIBRARY_PATH: /usr/local/cuda-12.3/lib64:/usr/local/cuda-12.3/
lib64:
2024-05-27 23:16:05.665781: W tensorflow/compiler/xla/stream_executor/platfo
rm/default/dso_loader.cc:64] Could not load dynamic library 'libnvinfer_plug
in.so.7'; dlerror: libnvinfer_plugin.so.7: cannot open shared object file: N
o such file or directory; LD_LIBRARY_PATH: /usr/local/cuda-12.3/lib64:/usr/l
ocal/cuda-12.3/lib64:
2024-05-27 23:16:05.665784: W tensorflow/compiler/tf2tensorrt/utils/py_util
s.cc:38] TF-TRT Warning: Cannot dlopen some TensorRT libraries. If you would
like to use Nvidia GPU with TensorRT, please make sure the missing libraries
mentioned above are installed properly.
```

In [2]:
```
df = pd.read_csv("housing.header.txt")
```

1. Read housing.header.txt as a dataframe. Report number of instances and features, and report all samples with respect to the Crim index on a plot (the x-axis shows the index of the sample, and the y-axis shows the crim index of the sample). [0.1 pt]

In [3]:
```
df.head()
```

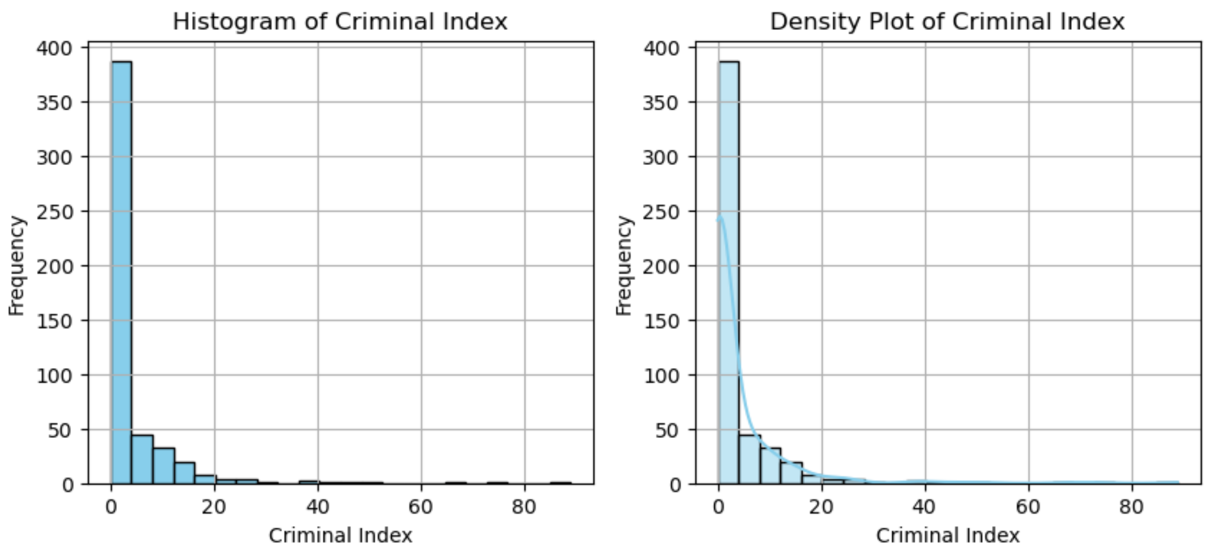
Out[3]:

	Crim	Zn	Indus	Chas	Nox	Rm	Age	Dis	Rad	Tax	Ptratio	B	Lstat
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33

2. Show both histogram of the Crim index and the density of the Crim index on a 1x2 frame (one row two columns). [0.1 pt]

```
In [4]: plt.figure(figsize = [10,4])

        plt.subplot(1,2,1)
        plt.hist(df["Crim"], bins= int(math.sqrt(len(df))), color='skyblue', edgecol
        plt.grid()
        plt.xlabel('Criminal Index')
        plt.ylabel('Frequency')
        plt.title('Histogram of Criminal Index')


        plt.subplot(1,2,2)
        sns.histplot(df["Crim"], kde=True, bins= int(math.sqrt(len(df))), color = 's
        plt.grid()
        plt.xlabel('Criminal Index')
        plt.ylabel('Frequency')
        plt.title('Density Plot of Criminal Index')
```

```
Out[4]: Text(0.5, 1.0, 'Density Plot of Criminal Index')
```

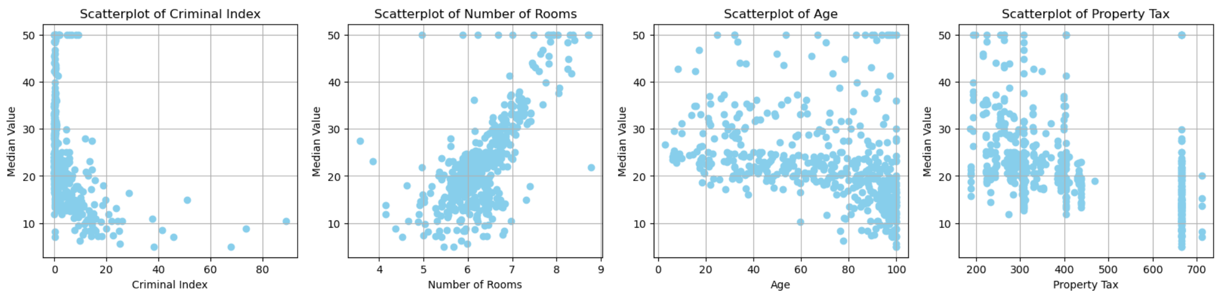


3. Show following four scatter plots in one frame (1x4), crim-medv, Rm-medv, Age-medv,Tax-medv, and explain how are they (Crim, Rm, Age, Tax) correlated to the medium house value (Medv) [0.1 pt]

```
In [5]: plt.figure(figsize = [20,4])

        plt.subplot(1,4,1)
        plt.scatter(x = df["Crim"], y = df["Medv"] , color='skyblue')
        plt.grid()
        plt.xlabel('Criminal Index')
        plt.ylabel('Median Value')
        plt.title('Scatterplot of Criminal Index')


        plt.subplot(1,4,2)
        plt.scatter(x = df["Rm"], y = df["Medv"] , color='skyblue')
        plt.grid()
        plt.xlabel('Number of Rooms')
        plt.ylabel('Median Value')
        plt.title('Scatterplot of Number of Rooms')

        plt.subplot(1,4,3)
        plt.scatter(x = df["Age"], y = df["Medv"] , color='skyblue')
        plt.grid()
        plt.xlabel('Age')
        plt.ylabel('Median Value')
        plt.title('Scatterplot of Age')


        plt.subplot(1,4,4)
        plt.scatter(x = df["Tax"], y = df["Medv"] , color='skyblue')
        plt.grid()
        plt.xlabel('Property Tax')
        plt.ylabel('Median Value')
        plt.title('Scatterplot of Property Tax')
```

Out[5]:

```
Text(0.5, 1.0, 'Scatterplot of Property Tax')
```



> The lower the crime rate in a property, the higher the property value is.

> The median value increases with the number of rooms.

> The Median value of houses and Age is pretty consistent for Age < 80 but it drops once Age > 80.

> There is no clear relationship between property tax and median value

4. Create a subset which only includes properties with Crim less than 1 (inclusive), and Rm greater than 6 (inclusive). [0.1 pt]

In [6]:

```
df_subset = df[(df["Crim"] < 1) & (df["Rm"] > 6)]
```

In [7]:

```
df_subset.head(10)
```

Out[7]:

	Crim	Zn	Indus	Chas	Nox	Rm	Age	Dis	Rad	Tax	Ptratio	B	Lstat
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33
5	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222	18.7	394.12	5.21
6	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.60	12.43
7	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.90	19.15
9	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.10
10	0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311	15.2	392.52	20.45

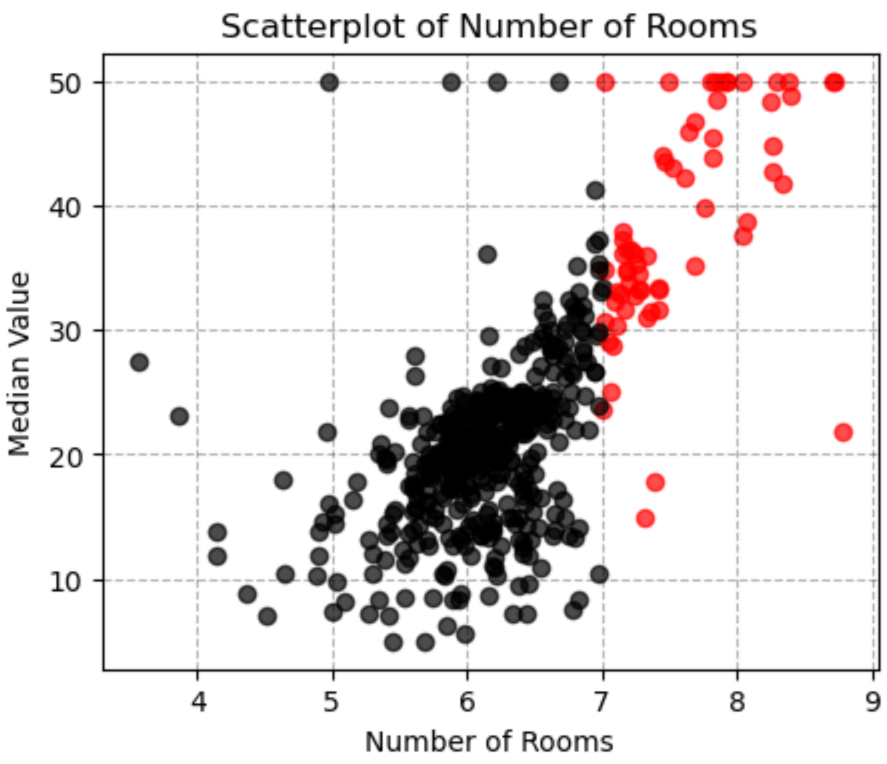
In [8]:
```
print("Maximum value of Criminal Index: ", max(df_subset["Crim"]))
print("Minimum value of Number of Rooms: ", min(df_subset["Rm"]))
```

```
Maximum value of Criminal Index:  0.95577
Minimum value of Number of Rooms:  6.004
```

5. Show a scatter plot between Rm and Medv (x-axis shows Rm and y-axis denote Medv), please color all properties with 7 or larger Rm values as “red”, and rest properties as “black”. [0.2 pt]

In [9]:
```
col =  []

for i in range(len(df)):

    if df["Rm"][i] >= 7:
        col.append("red")

    if df["Rm"][i] <7:
        col.append("black")
```

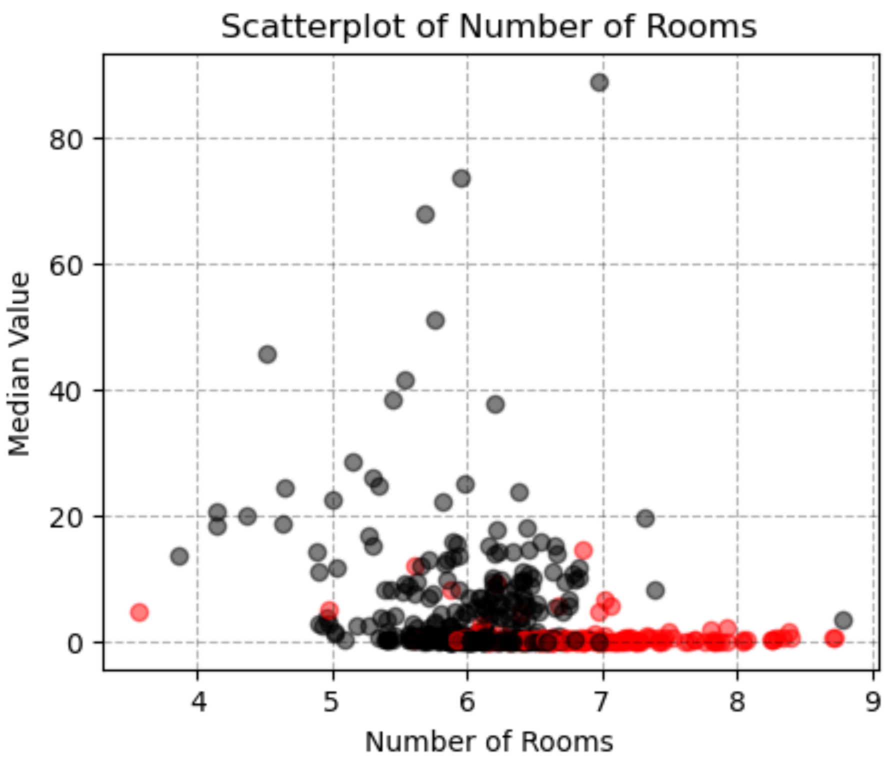
In [10]:
```
plt.figure(figsize = [5,4])

for i in range(len(df)):

    plt.scatter(x = df["Rm"][i], y = df["Medv"][i] , color= col[i], alpha =
    plt.xlabel('Number of Rooms')
    plt.ylabel('Median Value')
    plt.title('Scatterplot of Number of Rooms')

plt.grid( color='black', linestyle='--', alpha = 0.3)
```



6. Create a scatter plot between Rm and Crim and show all 506 properties on the plot. Color property whose Medv value greater or equal to 24 as red, and the rest as blue. [0.2 pt]

```
In [11]: col =  []

         for i in range(len(df)):

             if df["Medv"][i] >= 24:
                 col.append("red")

             else:
                 col.append("black")
```

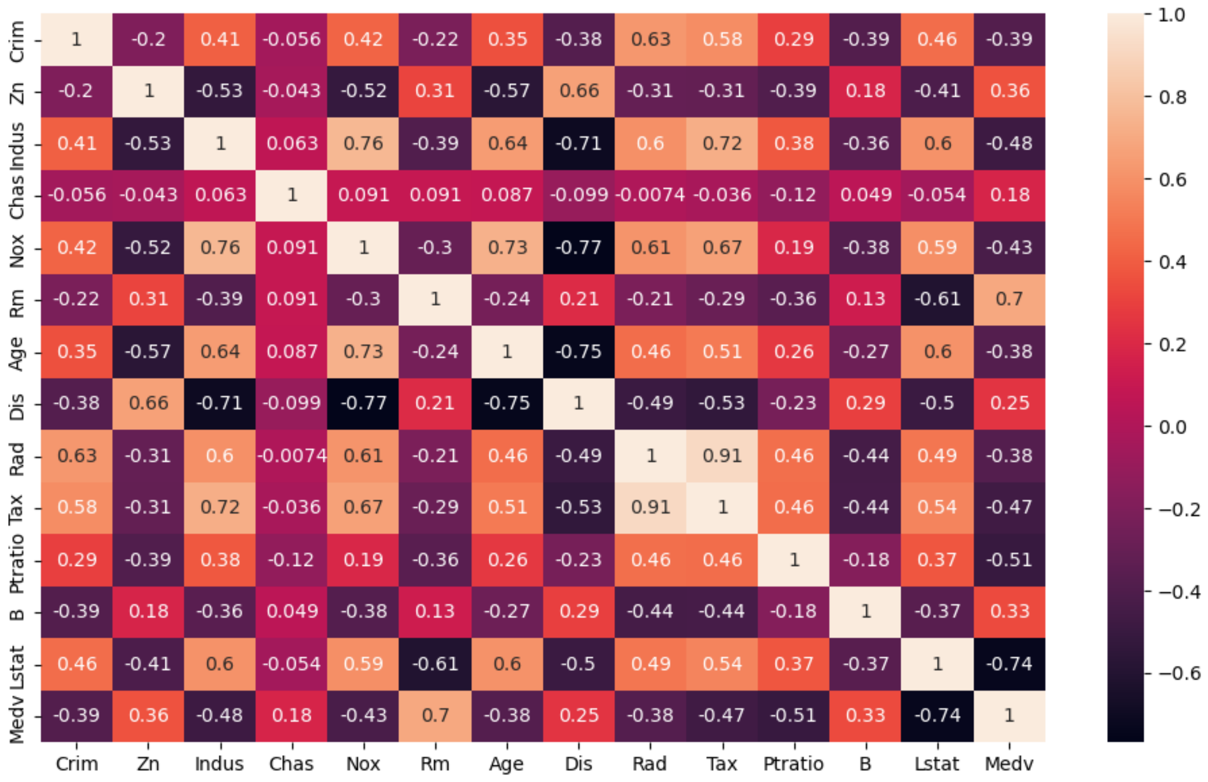
```
In [12]: plt.figure(figsize = [5,4])

         for i in range(len(df)):

             plt.scatter(x = df["Rm"][i], y = df["Crim"][i] , color= col[i], alpha =
             plt.xlabel('Number of Rooms')
             plt.ylabel('Median Value')
             plt.title('Scatterplot of Number of Rooms')

         plt.grid( color='black', linestyle='--', alpha = 0.3)
```



7. Report the pairwise correlation between every two variables (either as a matrix or as a level plot) [0.2 pt]

In [13]:
```
plt.figure(figsize = [12,7])

sns.heatmap(df.corr(), annot = True)
```

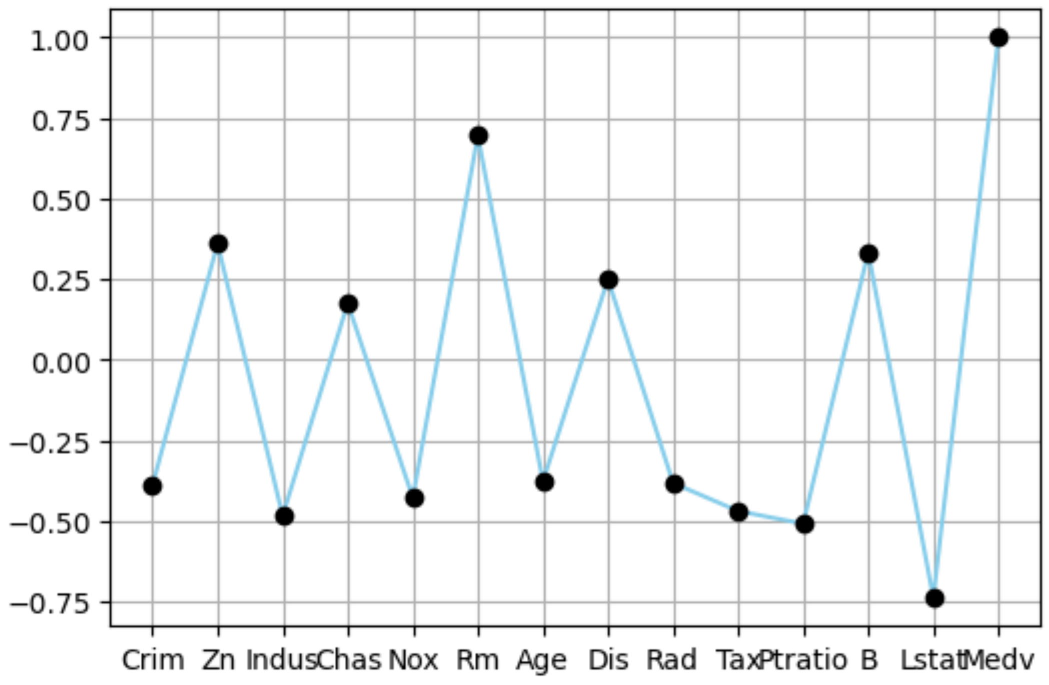
Out[13]:
```
<Axes: >
```



	Crim	Zn	Indus	Chas	Nox	Rm	Age	Dis	Rad	Tax	Ptratio	B	Lstat	Medv
Crim	1	-0.2	0.41	-0.056	0.42	-0.22	0.35	-0.38	0.63	0.58	0.29	-0.39	0.46	-0.39
Zn	-0.2	1	-0.53	-0.043	-0.52	0.31	-0.57	0.66	-0.31	-0.31	-0.39	0.18	-0.41	0.36
Indus	0.41	-0.53	1	0.063	0.76	-0.39	0.64	-0.71	0.6	0.72	0.38	-0.36	0.6	-0.48
Chas	-0.056	-0.043	0.063	1	0.091	0.091	0.087	-0.099	-0.0074	-0.036	-0.12	0.049	-0.054	0.18
Nox	0.42	-0.52	0.76	0.091	1	-0.3	0.73	-0.77	0.61	0.67	0.19	-0.38	0.59	-0.43
Rm	-0.22	0.31	-0.39	0.091	-0.3	1	-0.24	0.21	-0.21	-0.29	-0.36	0.13	-0.61	0.7
Age	0.35	-0.57	0.64	0.087	0.73	-0.24	1	-0.75	0.46	0.51	0.26	-0.27	0.6	-0.38
Dis	-0.38	0.66	-0.71	-0.099	-0.77	0.21	-0.75	1	-0.49	-0.53	-0.23	0.29	-0.5	0.25
Rad	0.63	-0.31	0.6	-0.0074	0.61	-0.21	0.46	-0.49	1	0.91	0.46	-0.44	0.49	-0.38
Tax	0.58	-0.31	0.72	-0.036	0.67	-0.29	0.51	-0.53	0.91	1	0.46	-0.44	0.54	-0.47
Ptratio	0.29	-0.39	0.38	-0.12	0.19	-0.36	0.26	-0.23	0.46	0.46	1	-0.18	0.37	-0.51
B	-0.39	0.18	-0.36	0.049	-0.38	0.13	-0.27	0.29	-0.44	-0.44	-0.18	1	-0.37	0.33
Lstat	0.46	-0.41	0.6	-0.054	0.59	-0.61	0.6	-0.5	0.49	0.54	0.37	-0.37	1	-0.74
Medv	-0.39	0.36	-0.48	0.18	-0.43	0.7	-0.38	0.25	-0.38	-0.47	-0.51	0.33	-0.74	1

8. Please explain which variable is mostly positively correlated to Medv (medium house value), and which variable is mostly negatively correlated to Medv. [0.2 pt]

In [14]:
```python
plt.figure(figsize = [6,4])

plt.plot(pd.DataFrame(df.corr()["Medv"]), marker = 'o', color = "skyblue", m
plt.grid()
```

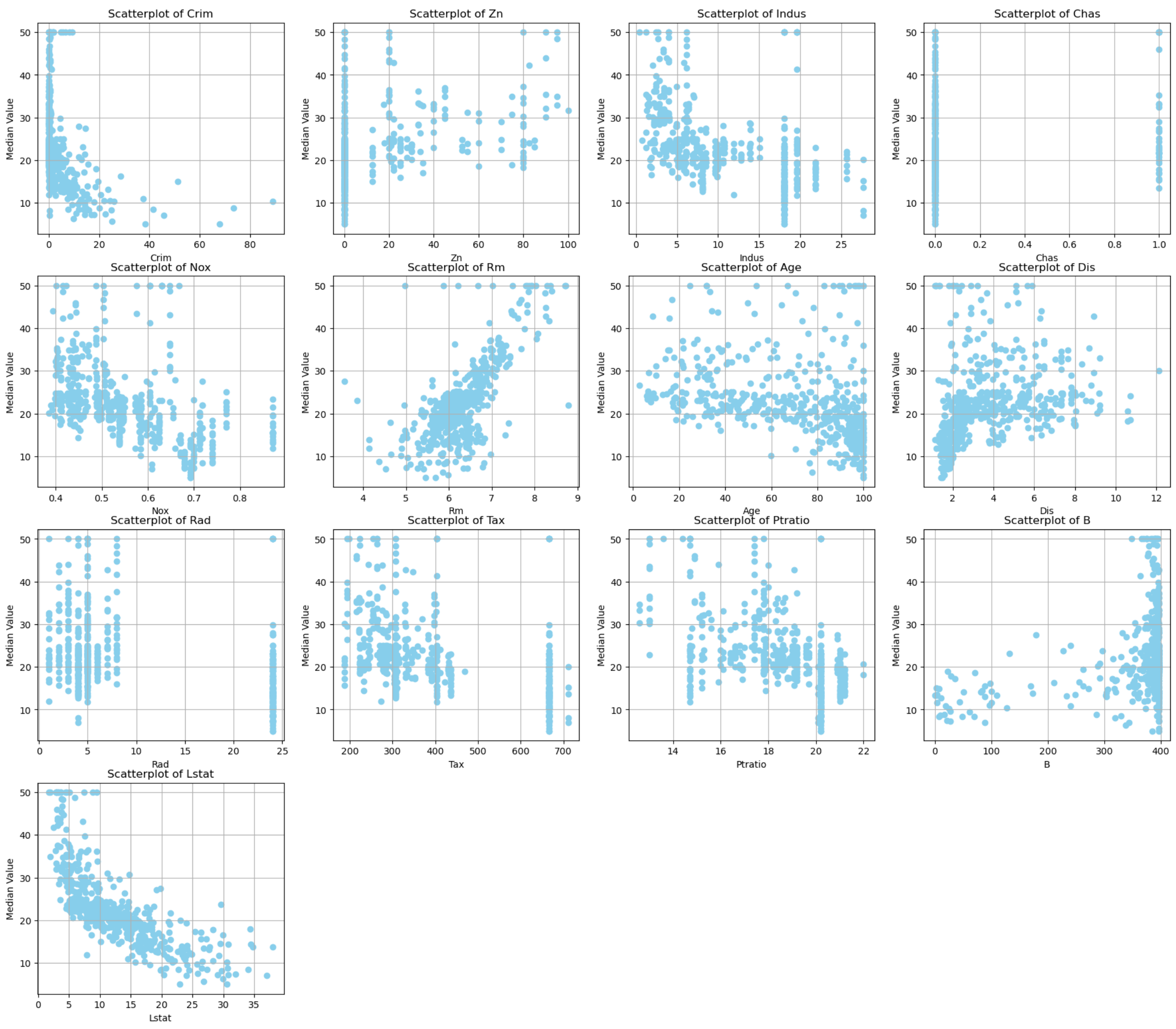


> The most positively correlated value is the number of rooms, i.e. with the increase in number of rooms the median value of the property increases.

> The most negatively correlated value is Lstat, i.e. with the decrease in population the median value of the property increases.

9. Draw scatterplots to show relationship between each attribute and Medv, respectively. [0.2 pt]

In [15]:
```
plt.figure( figsize = [22,19])
names = ['Crim', 'Zn', 'Indus', 'Chas', 'Nox', 'Rm', 'Age', 'Dis', 'Rad', 'T
       'Ptratio', 'B', 'Lstat']

for i in range(len(names)):
    plt.subplot(4,4,i+1)

    col= names[i]
    plt.scatter(x = df[col], y = df["Medv"] , color='skyblue')
    plt.grid()
    plt.xlabel(col)
    plt.ylabel('Median Value')
    plt.title('Scatterplot of ' + col)
```



10. Explain how to use scatterplots to find attributes which are positively correlated, negatively correlated, or independent of Medv, respectively. [0.2 pt]

Positive correlation occurs when the density of the points rises as the attribute's value grows, whereas negative correlation occurs when the density of the points falls as the attribute's value rises.

> Positve Correlation: Rm (Number of Rooms), Dis (weighted distances to five Boston employment centres), B (number of black people)

> Negative Correlation: Crim (Crime Index), Age (proportion of owner-occupied units built prior to 1940), Lstat (% lower status of the population)

11. Please create a new instance with following attribute value Crim=1.0, Zn=0.2,Indus=6,Chas=0.1,Nox=6.5,Rm=5,Age=100,Dis=4.1,Rad=4.5, Tax=21,Ptratio=20,B=300,Lstat=12,Medv=20.5 include this instance into the original dataframe, and report the number of instances and features of the new dataframe. [0.2 pt]

In [16]:
```
new = {'Crim' :1.0, 'Zn' :0.2, 'Indus' :6, 'Chas' :0.1, 'Nox' :6.5, 'Rm' :5,
 'Dis' :4.1, 'Rad' :4.5, 'Tax':21, 'Ptratio' :20, 'B' :300, 'Lstat' :12, 'Me

df = df._append(new, ignore_index = True)
```

In [17]:
```
len(df)
```

Out[17]:
```
507
```

In [18]:
```
df.tail(5)
```

Out[18]:

	Crim	Zn	Indus	Chas	Nox	Rm	Age	Dis	Rad	Tax	Ptratio	B	Lst
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90	9.(
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90	5.(
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45	6.4
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90	7.8
506	1.00000	0.2	6.00	0.1	6.500	5.000	100.0	4.1000	4.5	21.0	20.0	300.00	12.(

12. Create a new feature (named “Dummy”), and include the new feature into the new dataframe as the last feature. The values of the Dummy feature for each instances are randomly generated within range [0,5]. [0.2 pt]

In [19]:
```
dummy = []

for i in range(len(df)):
    val = random.uniform(0,5)
    dummy.append(val)
```

In [20]:
```
dummy = pd.DataFrame(dummy, columns = ["Dummy"])
```

In [21]:
```
frames = [df, dummy]
df = pd.concat(frames, axis = 1)
```

In [22]:
```
df.head()
```

Out[22]:

	Crim	Zn	Indus	Chas	Nox	Rm	Age	Dis	Rad	Tax	Ptratio	B	Lstat
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

12. Given two functions f1(x,y)=(x-2)2 + (y-3)2, and

f2(x,y)=(1 - (y-3))2 + 20((x+3) - (y-3)2)2, please implement gradient descent learning to search for global minimum value for each function.

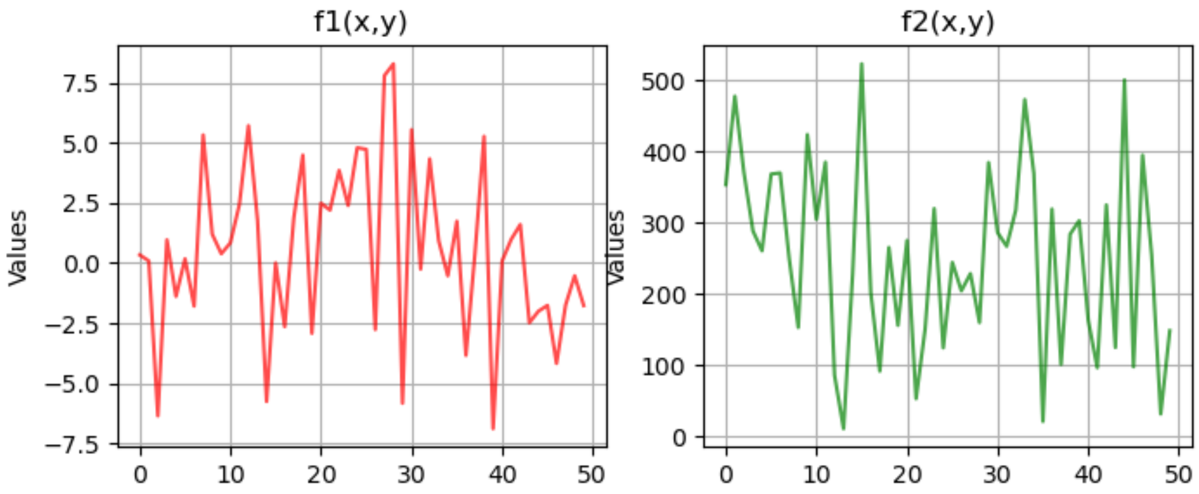
Use two Plots to show f1(x,y) and f2(x,y) values with respect to the two dimensional input x and y (You can specify the range of the x and y values) [0.5 pt]

In [23]:
```
x = []

for i in range(50):
    val = random.uniform(0,5)
    x.append(val)

y = []

for i in range(50):
    val = random.uniform(0,5)
    y.append(val)

data = {'x': x, 'y': y}
func = pd.DataFrame(data)
```

In [24]:
```
f1 = []
f2 = []

for i in range(len(func)):
    a = (func['x'][i]-2) * 2 + (func['y'][i]-3) * 2
    f1.append(a)

    b = (1 - (func['x'][i]-3)) *2 + 20 * ((func['x'][i] + 3) - (func['y'][i]
    f2.append(b)
```

In [25]:
```
plt.figure(figsize = [8,3])
plt.subplot(1,2,1)
plt.plot(f1, color = 'r', alpha = 0.7)
plt.grid()
plt.ylabel("Values")
plt.title(" f1(x,y) ")

plt.subplot(1,2,2)
plt.plot(f2, color = 'g', alpha = 0.7)
plt.grid()
plt.ylabel("Values")
plt.title(" f2(x,y) ")
```

Out[25]:
```
Text(0.5, 1.0, ' f2(x,y) ')
```



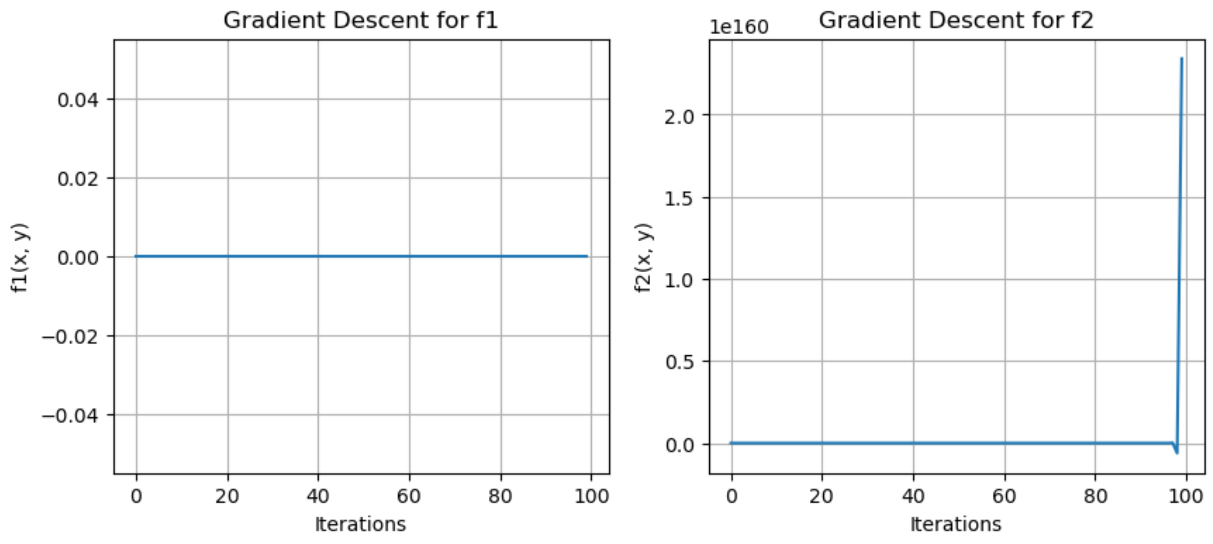
```
In [26]: random.uniform(0,1)

Out[26]: 0.9355798623504464
```

Starting from initial value: (x,y)=(0,0), use learning rate =0.5, report f1(x,y) and f2(x,y) values in T=100 iterations. (your code can report f1(x,y) and f2(x,y) values as tables, or simple print out the values). Explain whether the gradient descent learning is effective finding the solutions for f1(x,y) and f2(x,y), why or why not? [0.5 pt]

```
In [30]: def func1(x, y):
             f1 = (x-2) *2 + (y-3) * 2
             return f1

         def func2(x, y):
             f2 = (1 - (y-3)) * 2 + 20 * ((x+3) - (y-3) * 2) * 2
             return f2

         def gradient_func1(x, y):
             dx = 2 * (x - 2)
             dy = 2 * (y - 3)
             return dx, dy

         def gradient_func2(x, y):
             dx =  1
             dy = -2 * (y -3) + 2 * 20 * 2 * ( y - 3)
             return dx, dy
```

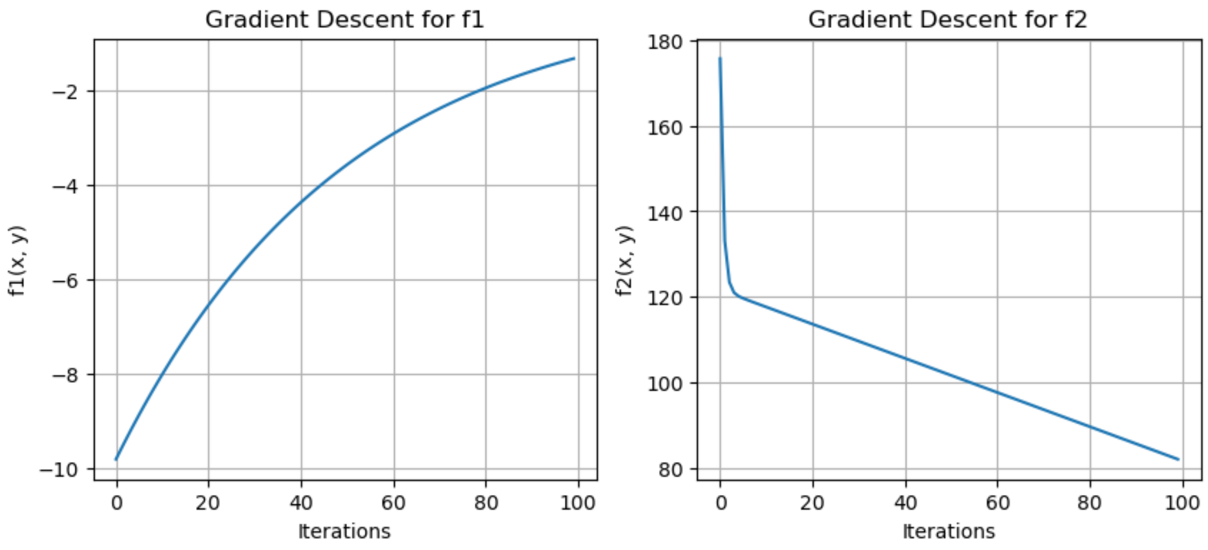
```
In [31]: def gradient_descent(func, gradient_func, initial_point, learning_rate, iter
             x, y = initial_point
             history = []

             for _ in range(iterations):
                 gradient = gradient_func(x, y)
                 x -= learning_rate * gradient[0]
                 y -= learning_rate * gradient[1]
                 history.append(func(x, y))
             return x, y, history
```

In [33]:
```
x1, y1, history_f1 = gradient_descent(func1, gradient_func1, (0, 0), 0.5, 10

x2, y2, history_f2 = gradient_descent(func2, gradient_func2, (0, 0), 0.5, 10

plt.figure(figsize=(10, 4))

plt.subplot(1, 2, 1)
plt.plot(range(100), history_f1)
plt.xlabel('Iterations')
plt.ylabel('f1(x, y)')
plt.title('Gradient Descent for f1')
plt.grid()

plt.subplot(1, 2, 2)
plt.plot(range(100), history_f2)
plt.xlabel('Iterations')
plt.ylabel('f2(x, y)')
plt.title('Gradient Descent for f2')
plt.grid()
```



Following step 2, please change your code (e.g., using different learning rates, such as =0.01) to try to search minimum for f1(x,y) and f2(x,y), respectively. Run algorithms for T=100 iterations Explain the motivation of your changes, and the final minimum values [0.5 pt]

In [34]:
```
x1, y1, history_f1 = gradient_descent(func1, gradient_func1, (0, 0), 0.01, 1

x2, y2, history_f2 = gradient_descent(func2, gradient_func2, (0, 0), 0.01, 1

plt.figure(figsize=(10, 4))

plt.subplot(1, 2, 1)
plt.plot(range(100), history_f1)
plt.xlabel('Iterations')
plt.ylabel('f1(x, y)')
plt.title('Gradient Descent for f1')
plt.grid()
plt.subplot(1, 2, 2)
plt.plot(range(100), history_f2)
plt.xlabel('Iterations')
plt.ylabel('f2(x, y)')
plt.title('Gradient Descent for f2')
plt.grid()
```



Explain why gradient descent learning can be used to help search solutions for f1(x,y) and f2(x,y), and what are the impact of the learning rate in the gradient descent learning

> In order to minimize the function, gradient descent iteratively modifies the parameters. The algorithm should converge to a global minimum at a suitable learning rate.

> A high learning rate could lead to no convergence by making the algorithm jump over the global minima. On the other hand, if the learning rate is too low, it may also cause the system to become stuck in a local minima and cause slow convergence.

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```