线性模型实践

助教：赖伟

5222023330042@smail.nju.edu

作业介绍

1. 基于梯度下降和闭式解求解线性回归任务。
2. 对实验结果进行可视化

导入相关包

在本样例中，仅使用可视化工具matplotlib及numpy

In [11]:
```
import numpy as np
import matplotlib.pyplot as plt
```

数据生成

随机生成一批数据，这里我们假设拟合y = 3x + 4， 同时添加部分噪声模拟真实情况。

In [2]:
```
def generate_linear_data(n_samples=100):
    np.random.seed(42)
    X = 2 * np.random.rand(n_samples, 1)
    y = 4 + 3 * X + np.random.randn(n_samples, 1) * 0.5  # 添加部分噪声
    return X, y
```

梯度下降求解法

通过梯度下降的方式进行拟合

1. 随机生成w和b，由$y = w * x + b$得到预测值
2. 计算预测误差
3. 根据预测误差计算梯度，并进行参数更新

In [3]:
```python
def linear_regression_gd(X, y, learning_rate=0.1, n_iterations=1000):
    m = len(y)
    w = np.random.randn(1)
    b = np.random.randn(1)

    for i in range(n_iterations):
        y_pred = w * X + b
        error = y_pred - y

        # 计算梯度
        dw = (2/m) * np.sum(error * X)
        db = (2/m) * np.sum(error)

        # 更新参数
        w -= learning_rate * dw
        b -= learning_rate * db

        if i % 100 == 0:
            loss = np.mean(error**2)
            print(f"Iteration {i}: Loss = {loss:.4f}")

    return w, b
```

闭式解求解法

通过令损失函数导数为0，可以求出闭式解为 $w = (X^T X)^{-1} X^T Y$

注意: $y = w * x + b = w' * [x, 1]$，其中$w' = [w, b]$

In [4]:
```python
def linear_regression_closed_form(X, y):
    X_b = np.c_[np.ones((len(X), 1)), X]  # 添加偏置项
```

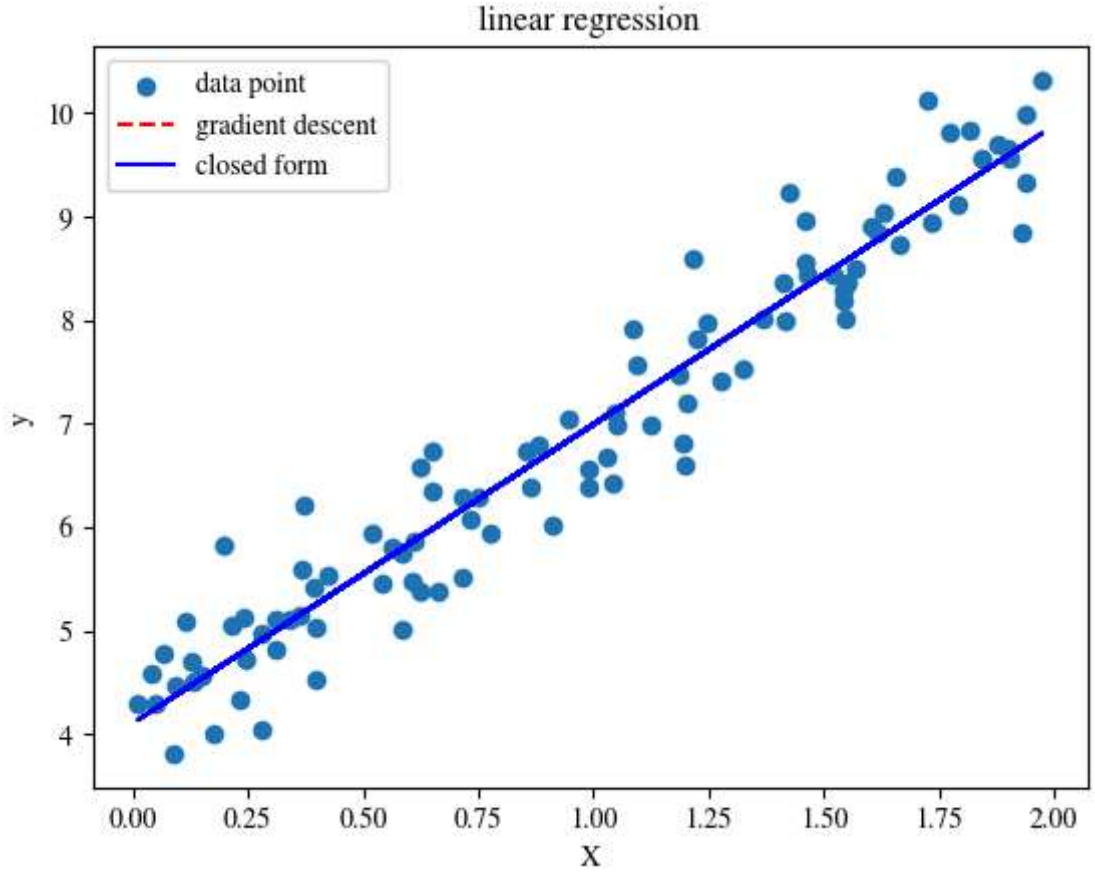
```
    theta_best = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)  # 计算闭式解
    return theta_best[1], theta_best[0]  # 返回 w 和 b
```

分别基于两种方法进行计算并进行可视化

In [13]:
```
def linear_regression_task():
    X, y = generate_linear_data()

    # 使用梯度下降求解
    w_gd, b_gd = linear_regression_gd(X, y)
    print("梯度下降 - 学习到的参数: w =", w_gd[0], "b =", b_gd[0])

    # 使用闭式解求解
    w_cf, b_cf = linear_regression_closed_form(X, y)
    print("闭式解 - 学习到的参数: w =", w_cf, "b =", b_cf)

    # 画图
    plt.scatter(X, y, label='data point')
    plt.plot(X, w_gd * X + b_gd, color='red', linestyle='dashed', label='gradient descent')
    plt.plot(X, w_cf * X + b_cf, color='blue', label='closed form')
    plt.xlabel("X")
    plt.ylabel("y")
    plt.legend()
    plt.title("linear regression")
    plt.show()
```

In [14]:
```
print("运行线性回归任务...")
linear_regression_task()
```

```
运行线性回归任务...
Iteration 0: Loss = 31.7662
Iteration 100: Loss = 0.2016
Iteration 200: Loss = 0.2016
Iteration 300: Loss = 0.2016
Iteration 400: Loss = 0.2016
Iteration 500: Loss = 0.2016
Iteration 600: Loss = 0.2016
Iteration 700: Loss = 0.2016
Iteration 800: Loss = 0.2016
Iteration 900: Loss = 0.2016
梯度下降 - 学习到的参数: w = 2.88505669321925 b = 4.107548078773364
闭式解 - 学习到的参数: w = [2.88505669] b = [4.10754808]
```



思考

如果需要需要添加L1或L2正则项，是否还能按照上述两种做法来做？

1. 是否还能使用闭式解，为什么？
2. 添加正则项以后，梯度的计算应该如何更新？