数据隐私实验报告

Project_DP 部分

PB16001680，李瀚民

[补全代码]：

说明：为了便于测试，我在源文件 Project_DP 中加入了一些代码，以显示当前的参数值，如果需要可以直接查看源代码。

（1）LR_GD：

```
def LR_GD(X, Y, eps, delta, T, C = 1., eta = 0.1):  # Solve the Linear regression with (eps, delta)-
differentially private SGD
    N, d = X.shape                                  # Get the dimension of X, here N = 100, d = 2, X:100x2, Y:100x1
    w = np.zeros((d,1))                             # w 是 2x1 的 00 矩阵
    # 计算参数
    eps_u, delta_u = comp_reverse(eps, delta, T)    # Compute the privacy parameter of each update, (eps_u, delta_u),
given (eps, delta, T)
    sigmasq = sigmasq_func(eps, delta)          # Compute the variance when sensitivity = 1
    L = 0.01 * N
    print("σ square:  " + str(sigmasq))

    for i in range(T):
        tmp = np.dot(X,w)-Y                         # tmp = Xw - Y
        gradient = 2*np.dot(X.T, tmp)               # Compute the gradient g = 2 * X^T * (Xw - Y), g now 2 x 1

        # to do: Clip gradient
        for grad_item in gradient:
            grad_item[0] = grad_item[0] / (max (1, math.sqrt((grad_item[0]) ** 2) / C))
        # print(gradient)
        # to do: Add noise                          # L = 2
        sum = 0
        index = 0
        for grad_item in gradient:
            sum += gradient[index][0]
            index += 1
        gradient_temp = []
        sigma_sq_dis = sigmasq * (C ** 2)
        norm_dis = np.random.normal(0.0, sigma_sq_dis, d)
        # print(norm_dis)
        for i in range(d):
            gradient_temp.append((1 / L) * (sum + norm_dis[i]))
        gradient_temp_toarr = [[item] for item in gradient_temp]
        gradient_temp_arr = np.array(gradient_temp_toarr)
        # to do: Gradient decent
        w = w - eta * gradient_temp_arr
        # print(w)
    return w
```

（2）LR_FM：

```
def LR_FM(X, Y, eps, delta):
    N, d = X.shape                                  # Get the dimension of X, here d = 2
```

```python
    sens = 2.*(1+d)**2
    sigmasq = sigmasq_func(eps, delta, sens)        # Variance in Functional Mechanism
    noise_1 = np.random.randn(d,d)
    noise_1 *= np.sqrt(sigmasq)/2.
    noise_1 = np.triu(noise_1)
    noise_1 = noise_1 + noise_1.T                   # Compute the noise matrix for X^T*X
    noise_2 = np.random.randn(d,1)
    noise_2 *= np.sqrt(sigmasq)                     # Compute the noise matrix for X^T*Y
    Phi = np.dot(X.T, X)                            # Phi = X^T * X (Phi hat 2 x 2)
    Phi_hat = Phi + noise_1
    # print(type(Phi_hat))
    # print(Phi_hat)
    Identity_matrix = np.array([[1.0, 0.0], [0.0, 1.0]])
    Phi_hat += Identity_matrix
    XY = np.dot(X.T, Y)
    XY_hat = XY + noise_2
    tmp = np.linalg.inv(Phi_hat)
    w = np.dot(tmp, XY_hat)                         # w = (X^T*X)^(-1) * X^TY
    return w
```

[结果分析]:

(1) LR_GD:

首先，根据文献 Deep Learning with Differential Privacy 里给的值来确定参数，可以进行如下参数测试:

```
(data_privacy) C:\Users\lihanming>python C:\Users\lihanming\Desktop\数据隐私实验\DP_SGD\project_DP.py
 ε : 1.26
 δ : 1e-05
  T: 10000
 σ  square:  14.7846674430391
[[39151.41389014]]

(data_privacy) C:\Users\lihanming>python C:\Users\lihanming\Desktop\数据隐私实验\DP_SGD\project_DP.py
 ε : 1.26
 δ : 1e-05
  T: 10000
 σ  square:  14.7846674430391
[[2169.52904135]]
```

可以看出来；其实最后算出的 L_2 Loss 的值会比较大，效果不是特别的理想，接下来尝试如下一组值:

```
(data_privacy) C:\Users\lihanming>python C:\Users\lihanming\Desktop\数据隐私实验\DP_SGD\project_DP.py
 ε : 2
 δ : 1e-05
  T: 10000
 σ  square:  5.868034508142219
[[2454.27463565]]

(data_privacy) C:\Users\lihanming>python C:\Users\lihanming\Desktop\数据隐私实验\DP_SGD\project_DP.py
 ε : 4
 δ : 1e-05
  T: 10000
 σ  square:  1.4670086270355547
[[727.85192441]]

(data_privacy) C:\Users\lihanming>python C:\Users\lihanming\Desktop\数据隐私实验\DP_SGD\project_DP.py
 ε : 8
 δ : 1e-05
  T: 10000
 σ  square:  0.36675215675888867
[[475.4835085]]
```

可以看出来，随着ε的变大，效果似乎越变越好，同样地，我们可以尝试一下将δ的值改变一下：然后从中发现规律

```
(data_privacy) C:\Users\lihanming>python C:\Users\lihanming\Desktop\数据隐私实验\DP_SGD\project_DP.py
ε : 8
δ : 1e-06
 T: 10000
σ  square:  0.43870794091495263
[[130.74469683]]

(data_privacy) C:\Users\lihanming>python C:\Users\lihanming\Desktop\数据隐私实验\DP_SGD\project_DP.py
ε : 8
δ : 1e-07
 T: 10000
σ  square:  0.5106637250710165
[[1959.67731391]]

(data_privacy) C:\Users\lihanming>python C:\Users\lihanming\Desktop\数据隐私实验\DP_SGD\project_DP.py
ε : 8
δ : 1e-09
 T: 10000
σ  square:  0.6545752933831444
[[2037.02605954]]
```

由此可以看出，似乎当δ变小时，似乎表现下降了，但是由于其中有很多地方引入了随机性，其实并不能一定的说明这个 loss 就是变大的趋势。

综合上述的测试，再结合进一步的参数调整，可以发现

```
(data_privacy) C:\Users\lihanming>python C:\Users\lihanming\Desktop\数据隐私实验\DP_SGD\project_DP.py
ε : 4
δ : 0.0001
 T: 5000
σ  square:  1.179185490411299
[[98.23786642]]
```

在参数选取时，取上述参数的时候，可以让 L2 loss 较小，并且整体性能较好。

（2） LR_FM：

```
(data_privacy) C:\Users\lihanming>python C:\Users\lihanming\Desktop\数据隐私实验\DP_SGD\project_D
ε : 1.25
δ : 1e-05
 T: 10000
[[19.3192292]]
```

取之前的文献中的数据，效果其实不错，

再测试几组数据：

```
(data_privacy) C:\Users\lihanming>python C:\Users\lihanming\Desktop\数据隐私实验\DP_SGD\project_DP.py
ε : 1.25
δ : 0.0001
 T: 10000
[[75.82877907]]

(data_privacy) C:\Users\lihanming>python C:\Users\lihanming\Desktop\数据隐私实验\DP_SGD\project_DP.py
ε : 4
δ : 0.0001
 T: 10000
[[82.73833973]]

(data_privacy) C:\Users\lihanming>python C:\Users\lihanming\Desktop\数据隐私实验\DP_SGD\project_DP.py
ε : 4
δ : 0.0001
 T: 5000
[[108.37822524]]

(data_privacy) C:\Users\lihanming>python C:\Users\lihanming\Desktop\数据隐私实验\DP_SGD\project_DP.py
ε : 4
δ : 0.0001
 T: 100000
[[8532.26358481]]

(data_privacy) C:\Users\lihanming>python C:\Users\lihanming\Desktop\数据隐私实验\DP_SGD\project_DP.py
ε : 8
δ : 1e-07
 T: 10000
[[15.9373995]]
```

除了有一组的 L2-loss 比较大以外，其余的 L2 loss 都比较小，并且由于随机性的原因，实际上是可以忽略

的，因此取第一组数据，其性能就已经较好。