《大数据机器学习》第 9 次作业

姓名：刘培源　　学号：2023214278

题目 1： 写出条件随机场模型学习的梯度下降法。

答： 在条件随机场（CRF）中，对数极大似然函数可表示为：

$$L(w)=\sum_{j=1}^{N}\sum_{k=1}^{K}w_kf_k(y_j,x_j)-\sum_{j=1}^{N}\log Z_w(x_j)$$

其中，w 是模型参数，f_k 是特征函数，y_j 和 x_j 分别是对应的标签和观测序列，$Z_w(x_j)$ 是配分函数。

在应用梯度下降算法优化该函数时，首先定义目标函数 $f(w)=-L(w)$，然后计算目标函数 $f(w)$ 关于参数 w 的梯度：

$$g(w)=\nabla f(w^{(k)})=\left(\frac{\partial f(w)}{\partial w_1},\frac{\partial f(w)}{\partial w_2},\cdots,\frac{\partial f(w)}{\partial w_k}\right)$$

具体地，梯度的每一分量可表示为：

$$\begin{aligned}\frac{\partial f(w)}{\partial w_i}&=-\sum_{j=1}^{N}w_if_i(y_j,x_j)+\sum_{j=1}^{N}\frac{1}{Z_w(x_j)}\cdot\frac{\partial Z_w(x_j)}{\partial w_i}\\&=-\sum_{j=1}^{N}w_if_i(y_j,x_j)+\sum_{j=1}^{N}\frac{1}{Z_w(x_j)}\sum_{y}(\exp\sum_{k=1}^{K}w_kf_k(y,x_j))w_if_i(y,x_j)\end{aligned}$$

梯度下降算法的具体步骤如下：

1. 初始化参数 $w^{(0)}\in\mathbf{R}^n$，设 $k=0$。
2. 计算当前迭代的目标函数值 $f(w^{(k)})$。
3. 计算当前梯度 $g_k=g(w^{(k)})$。若 $\|g_k\|<\varepsilon$（即梯度足够小），则停止迭代，令最优解 $w^*=w^{(k)}$；否则，设 $p_k=-g(w^{(k)})$，并求解 λ_k 以满足：

$$f(w^{(k)}+\lambda_kp_k)=\min_{\lambda\geqslant 0}f(w^{(k)}+\lambda p_k)$$

4. 更新参数：$w^{(k+1)}=w^{(k)}+\lambda_kp_k$，计算新的目标函数值 $f(w^{(k+1)})$。
5. 若满足 $\|f(w^{(k+1)})-f(w^{(k)})\|<\epsilon$ 或 $\|w^{(k+1)}-w^{(k)}\|<\epsilon$，则停止迭代，将最优解设为 $w^*=w^{(k+1)}$；否则，令 $k=k+1$，回到步骤 3。

题目 2： 参考图1的状态转移图，假设隐状态序列 $M_1(x), M_2(x), M_3(x), M_4(x)$ 分别是

$$M_1(x) = \begin{bmatrix} 0 & 0 \\ 0.5 & 0.5 \end{bmatrix}, \quad M_2(x) = \begin{bmatrix} 0.3 & 0.7 \\ 0.7 & 0.3 \end{bmatrix}, \quad M_3(x) = \begin{bmatrix} 0.5 & 0.5 \\ 0.6 & 0.4 \end{bmatrix}, \quad M_4(x) = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$$

求以 $\text{start} = 2$ 为起点，以 $\text{stop} = 2$ 为终点的所有经过状态序列 y 的概率及概率最大的状态序列。

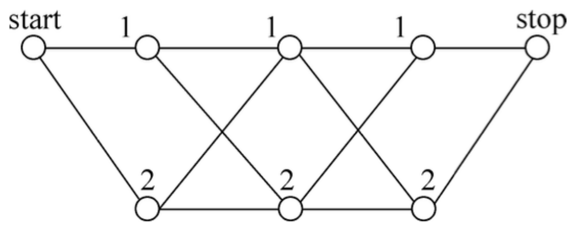

图 1: 状态路径

答： 根据题意实现代码如下：

```python
import numpy as np

# 状态转移矩阵
M = [
    np.array([[0, 0], [0.5, 0.5]]),
    np.array([[0.3, 0.7], [0.7, 0.3]]),
    np.array([[0.5, 0.5], [0.6, 0.4]]),
    np.array([[0, 1], [0, 1]])
]

# 递归计算所有路径及其概率
def compute_paths(current_path, matrix_index):
    if matrix_index == 4:  # 终止条件
        return [(current_path, np.prod([M[i][current_path[i]-1, current_path[i+1]-1]
                                        for i in range(len(current_path) - 1)]))]
    paths = []
    for next_state in [1, 2]:  # 下一个状态
        paths.extend(compute_paths(current_path + [next_state], matrix_index + 1))

    return paths

# 计算所有路径
all_paths = compute_paths([2], 0)

# 打印所有路径及其概率
print("以start=2为起点stop=2为终点的所有路径的状态序列y的概率为:")
for path in all_paths:
    if path[1] != 0:
        print("路径为:", "->".join(map(str, path[0])), f"概率为:{path[1]:.3f}")

# 打印最大概率的路径
max_path, max_prob = sorted(all_paths, key=lambda x: x[1], reverse=True)[0]
print("概率最大的状态序列为:", "->".join(map(str, max_path)), "概率为:", max_prob)
```

代码输出如下：

```
路径为： 2->1->1->1->2 概率为:0.075
路径为： 2->1->1->2->2 概率为:0.075
路径为： 2->1->2->1->2 概率为:0.210
路径为： 2->1->2->2->2 概率为:0.140
路径为： 2->2->1->1->2 概率为:0.175
路径为： 2->2->1->2->2 概率为:0.175
路径为： 2->2->2->1->2 概率为:0.090
路径为： 2->2->2->2->2 概率为:0.060
概率最大的状态序列为： 2->1->2->1->2 概率为： 0.21
```

答案即为代码输出。