巧妙利用range 函数

ex1:

probs[range(num_train), y]

range(num_train)：这会生成一个从 0 到 N-1 的序列 [0, 1, 2, ..., N-1]。它将作为行索引。

y：是一个形状为 (N,) 的向量，y[i] 存储的是第 i 个样本的正确类别的索引。它将作为列索引

当您用两个同样长度的数组（或列表）去索引一个 NumPy 数组时，NumPy 会将它们逐一配对，取出对应 (行, 列) 位置的元素。

结果：这个索引操作会返回一个形状为 (N,) 的一维向量，其中包含了每一个样本的正确类别的预测概率。

ex2:

dscores[range(num_train), y] -= 1

- 这里不利用range取出，而是利用这个巧妙索引，把数组中对应的数进行处理

参数的利用

ex1：

np.sum(exp_scores, axis=1, keepdims=True)

参数1：axis:
沿着 axis=0 求和，意味着对每一列的元素进行相加。
沿着 axis=1 求和，意味着对每一行的元素进行相加。

参数2：keepdims
keepdims=False (默认情况)：当 NumPy 沿着某个轴求和时，它会“压缩”掉那个维度。

- 对于形状为 (N, C) 的 exp_scores，np.sum(..., axis=1) 的结果形状会是 (N,)，即一个一维向量。

keepdims=True：这个参数告诉 NumPy 不要压缩掉求和的那个维度，而是将它的大小变为 1。

- 对于形状为 (N, C) 的 exp_scores，np.sum(..., axis=1, keepdims=True) 的结果形状会是 (N, 1)，即一个列向量。
- 这个处理是为了对后续向量的broadcast作准备。

python同C的区别

python 的变量名更类似于C的指针，不会给变量分配内存，所以如果像建立一个新的变量，应该这样：dscores = probs.copy()

- 类似于创建副本这样修改dscores就不会改变probs的值了，如果直接dscores = probs，那么会改变值

.dot()函数

当两个输入都是二维矩阵时，会严格按照矩阵乘法来。

当矩阵 (2D) 和一维数组 (N,)时NumPy 会将一维数组“临时”当作向量处理，但最终结果会**“压缩”回一个一维数组**。

ex:

```
matrix = np.array([[1, 2, 3],
                   [4, 5, 6]]) # shape: (2, 3)
vector = np.array([1, 0, 1])     # shape: (3,)

result = matrix.dot(vector)
print(result)                    # 输出: [4 10]
print(result.shape)              # 输出: (2,) <-- 仍然是一维数组
```

outer（） 函数：

ex1:

np.outer(X[i], dscores)

np.outer(a, b)：这是一个外积操作。如果 X[i] 的形状是 (D,)，dscores 的形状是 (C,)，那么外积的结果就是一个 (D, C) 的矩阵。这正是我们需要的单个样本的梯度矩阵 dW_i 的形状。

np.argmax（）：

np.argmax() 函数就是用来查找数组中最大值索引的。
axis=1 这个参数至关重要。它告诉 argmax 函数沿着每一行（跨列）独立地查找最大值的索引。这正好对应了为每一个样本找出得分最高的那个类别。