基于全连接神经网络的 MNIST 图片分类

汪加西，21210980069

- 算法实现

本文使用具有单隐藏层的 MLP 对 MNIST 数字图片作分类，网络包含：维数为 $28 \times 28 = 784$ 的输入层 (与图片大小相适应)；以 ReLU 为激活函数、维数为 h_1 的隐藏层 (引入 dropout，以概率 p 保留单个神经元)；10 维的输出层，其结果经 Softmax 归一化即为 10 个类别概率。使用交叉熵定义损失：

$$L = -\frac{1}{N} \cdot \sum_{i=1}^{N} \sum_{j=1}^{10} log(p_{i,j}) \cdot y_{i,j} + \frac{\lambda}{2N} \cdot (\|w_1\|_2^2 + \|w_2\|_2^2)$$

其中 $p_{i,j} \in \{0,1\}$ 表示样本 i 的标签是否为 j，λ 为 L2 正则化系数，$w_1 \in R^{784 \times h}$、$w_2 \in R^{h \times 10}$ 为全连接层的权重矩阵。由于训练集所含样本较多，使用随机梯度下降 (SGD) 更新参数，即：将全部 N 个训练样本随机分为 k 个大小为 B 且不重叠的 batch，每轮迭代仅从 B 个样本中选择一个计算梯度。记所选为 $x^\star = (x_1 \dots x_{784})$，其类别为 $j \in \{0,1 \dots 9\}$，则前向传播的过程为

$$z_1 = x^\star \cdot w_1 + b_1$$

$$a_1 = \mathrm{ReLU}(z_1) = (\max(z_{1,1}, 0), max(z_{1,2}, 0) \dots \ max(z_{1,h}, 0) \ )$$

$$a_1 = \mathrm{Drop}(a_1) = (a_{1,1} \cdot I(u_1 \le p), a_{1,2} \cdot I(u_2 \le p) \dots a_{1,n} \cdot I(u_n \le p))$$

$$z_2 = a_1 \cdot w_2 + b_2$$

$$y = \mathrm{softmax}(z_2) = (e^{z_{2,1}} \dots e^{z_{2,10}}) / \sum_{j=1}^{10} e^{z_{2,j}}$$

其中 $u_1 \dots u_n$ 为来自 (0,1) 均匀分布的随机数，$I(\cdot) \in \{0,1\}$ 为示性函数。梯度的计算公式为

$$\delta_2 = (y_1, y_2 \dots y_j - 1 \dots y_{10})$$

$$dw_2 = a_1 \cdot \delta_2 + \frac{\lambda}{B} \cdot w_2$$

$$db_2 = \delta_2$$

$$\delta_1 = (\delta_2 \cdot w_2) \odot (I(a_{1,1} \ge 0), I(a_{1,2} \ge 0) \dots I(a_{1,n} \ge 0))$$

$$dw_1 = x^T \cdot \delta_1$$

$$db_1 = \delta_1$$

其中 $\odot$ 表示矩阵间点乘。记学习率为 ν，更新参数为

$$(w_1', w_2', b_1', b_2') = (w_1, w_2, b_1, b_2) - \nu \cdot (dw_1, dw_2, db_1, db_2)$$

初始参数由 $(-0.1, 0.1)$ 上的均匀分布产生，并添加 1×10^{-4} 的偏置。为使损失能收敛到全局极小值点，使用 Adagrad (Duchi et.al., 2011) 自适应地调整学习率：设初始学习率为 ν_0，对于 w_1, w_2, b_1, b_2 中的任一参数 θ

1. 记 θ 的累积梯度为 C_θ，初始化 $C_\theta^{(0)}=0$；

2. 对第轮 r 轮迭代，令 $C_\theta^{(r+1)}=C_\theta^{(r)}+d_\theta^{(r)}\times d_\theta^{(r)}$，其中 $d_\theta^{(r)}=\partial L^{(r)}/\partial\theta^{(r)}$ 是 θ 当前梯度；

3. 更新 θ 为 $\theta^{(r+1)}=\theta^{(r)}-\nu_0 d_\theta^{(r)}\cdot\left(\delta+\sqrt{C_\theta^{(r+1)}}\right)^{-1}$，其中 δ 是一很小的常数，可设为 10^{-7}；

4. 对 $r=0,1...R$，重复第 2、3 步。

相比学习率固定的优化算法，Adagrad 对每个参数的收敛速度不同：累积梯度越小的参数学习率越大，但整体上学习率是随迭代次数增加而下降的。以上算法请见 model.py。

- 模型训练

为确定一组较优的超参数 (隐藏层维数 h_1、初始学习率 ν_0 及正则化系数 λ)，将初始的六万个训练样本按 4:1 随机划分为训练和测试集。令 $h_1\in\{32,64,128\}$、$\nu_0\in\{7\times10^{-3},8\times10^{-3},1\times10^{-2}\}$、$\lambda\in\{1\times10^{-4},\ 5\times10^{-4},1\times10^{-3}\}$，对任一参数组合：在训练集上训练 200 个 epoch，记录每个 epoch 后模型在验证集上的分类准确率，记其所达到的最大值为 $s^{(h_1,\nu_0,\lambda)}$，则选择 $\{h_1^\star,\nu_0^\star,\lambda^\star\}=argmax\ s^{(h_1,\nu_0,\lambda)}$。批量大小和保留率统一设置为 $B=20,p=0.95$。图 1 为其中三个模型在训练集、验证集上的损失和准确率 (为更好显示图形，截取 loss 在 0.6 以下、准确率在 0.85 以上的部分)。可见模型在训练和验证集上的损失 / 准确率都随 epoch 增加而降低 / 提升，但在验证集上收敛更快 (此时若继续迭代则可能导致过拟合)。表 1 列出分类准确率排名前十的模型。

表 1：验证集上最高分类准确率及达到该准确率的迭代次数

h_1	ν_0	λ	$accuracy$	$epoch$
128	0.01	0.0005	0.9672	193
128	0.01	0.0001	0.9655	178
128	0.01	0.001	0.9645	161
128	0.008	0.001	0.9633	194
64	0.008	0.001	0.9626	194
128	0.008	0.0001	0.9624	173
128	0.008	0.0005	0.9616	195
64	0.007	0.001	0.9611	196
128	0.007	0.0001	0.9606	166
64	0.01	0.0005	0.9606	183

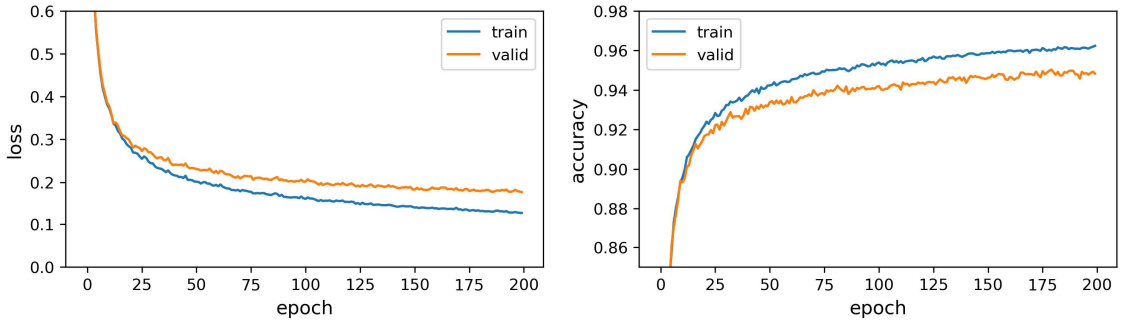


图 1(a)：$h_1 = 32, \nu_0 = 7 \times 10^{-3}, \lambda = 1 \times 10^{-4}$

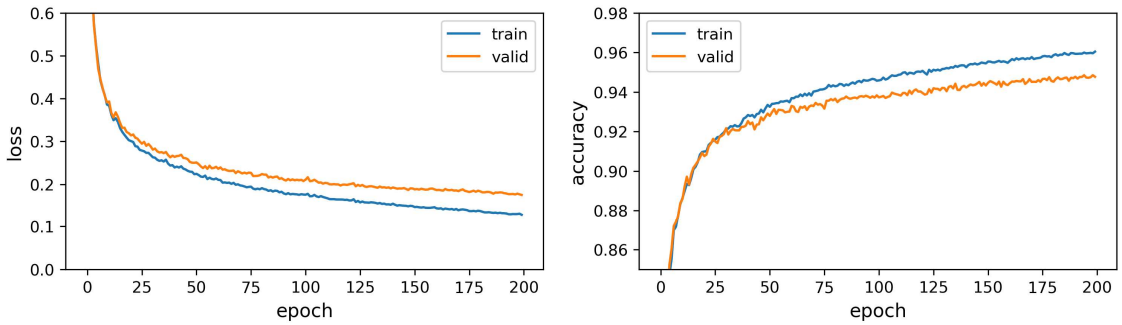


图 1(b)：$h_1 = 32, \nu_0 = 7 \times 10^{-3}, \lambda = 5 \times 10^{-4}$

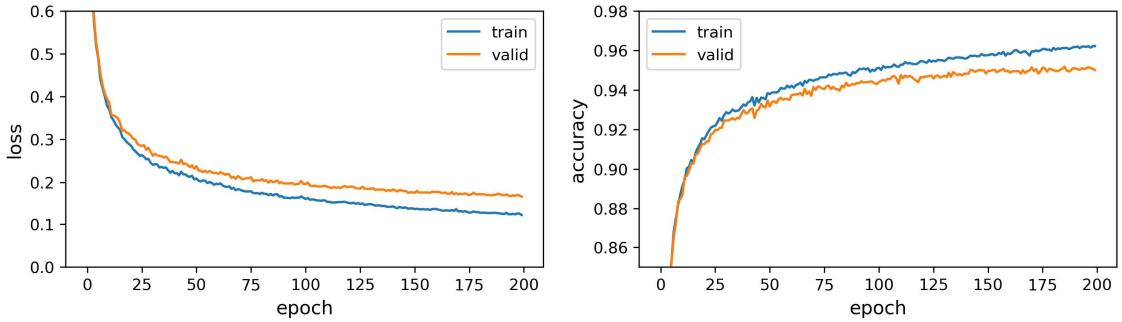


图 1(c)：$h_1 = 32, \nu_0 = 7 \times 10^{-3}, \lambda = 1 \times 10^{-3}$

从而超参数确定为 $h_1 = 128, \nu_0 = 1 \times 10^{-2}, \lambda = 1 \times 10^{-4}$。使用 pickle 将训练后模型保存到本地，见 model.pickle。以上代码见 train.py，训练过程记录于 train_record.txt。

- 模型测试

表 2 为所训练 MLP 及其他分类模型在测试集上的准确率。为作参照，额外使用 Pytorch 训练了一结构稍复杂的神经网络（见 model_Pytorch.pkl），其包含两个维度分别为 256、512 的隐

藏层，以 ReLU 为激活函数。由表 2，神经网络相比逻辑回归、RF 取得更高准确率，且在实验条件下，维数越高的网络一般性能更好 (见表 1)。图 2 为部分 MLP 错误分类的图片，表 3 为 MLP 分类的混淆矩阵，其中行、列分别表示预测和真实标签。

表 2：测试集上分类准确率

	MLP	MLP-Pytorch	逻辑回归	随机森林
accuracy	0.967	0.967	0.923	0.867

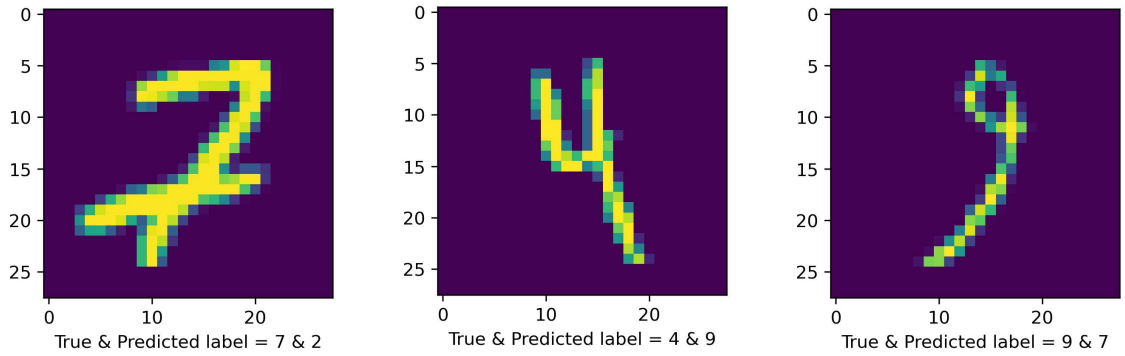


图 2：MLP 错误分类的图像

表 3：MLP 分类的混淆矩阵

	0	1	2	3	4	5	6	7	8	9
0	0.989	0.000	0.002	0.001	0.000	0.002	0.004	0.001	0.000	0.001
1	0.000	0.986	0.003	0.002	0.000	0.001	0.002	0.000	0.007	0.000
2	0.007	0.001	0.964	0.007	0.001	0.001	0.002	0.010	0.008	0.000
3	0.000	0.001	0.004	0.967	0.000	0.015	0.000	0.006	0.004	0.003
4	0.004	0.001	0.003	0.000	0.958	0.001	0.005	0.001	0.002	0.024
5	0.004	0.000	0.000	0.010	0.001	0.964	0.011	0.000	0.007	0.002
6	0.008	0.002	0.004	0.001	0.009	0.010	0.958	0.000	0.006	0.000
7	0.000	0.005	0.011	0.006	0.002	0.001	0.000	0.963	0.003	0.010
8	0.008	0.001	0.004	0.006	0.003	0.007	0.001	0.004	0.960	0.005
9	0.007	0.003	0.001	0.008	0.010	0.004	0.000	0.008	0.004	0.955

◆ 参数可视化

使用矩阵热力图可视化神经网络的权重参数 w_1, w_2 和偏置 b_1, b_2，如图 3 所示，其中 $w_1 \in R^{784\times128}, w_2 \in R^{128\times10}, b_1 \in R^{128}, b_2 \in R^{10}$，图3(a)、3(b)分别表示训练前随机初始化的网络和训练后网络，可见训练前后网络参数变化较大。类似地，使用热力图可视化隐藏层的输入 z_1 和输出 a_1 (由 1×128 转换为 16×8)，如图 4。

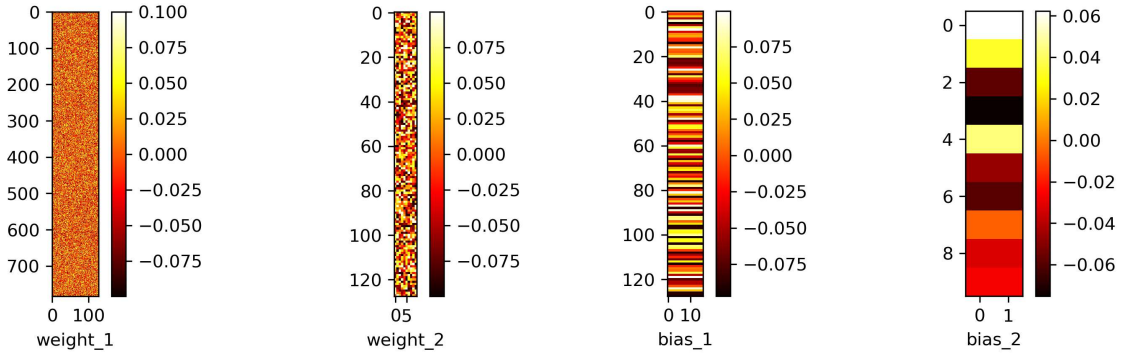


图 3(a)：初始网络

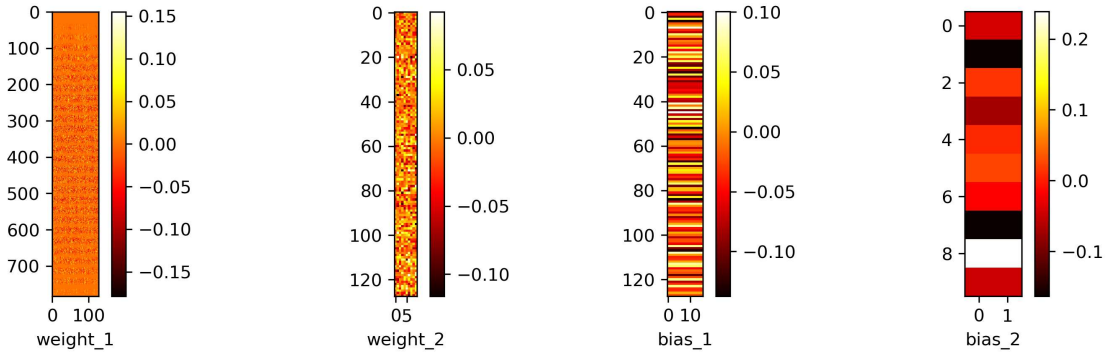
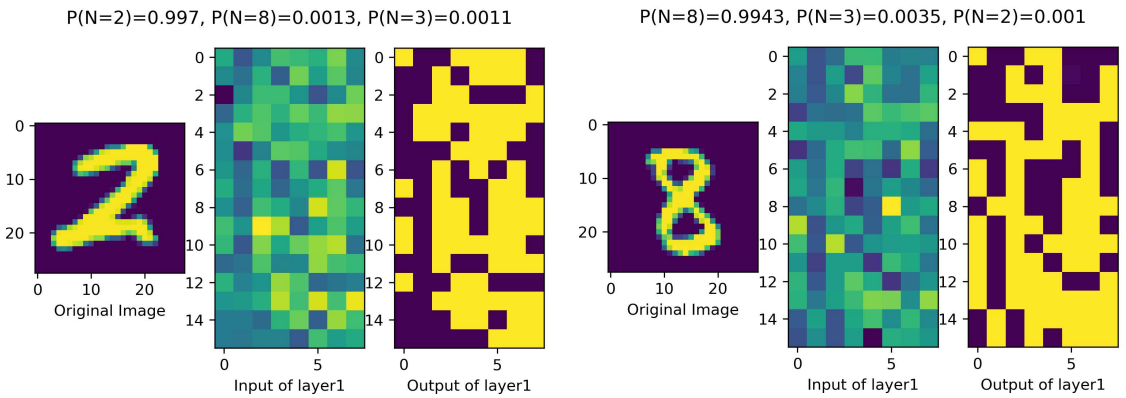


图 3(b)：训练后网络



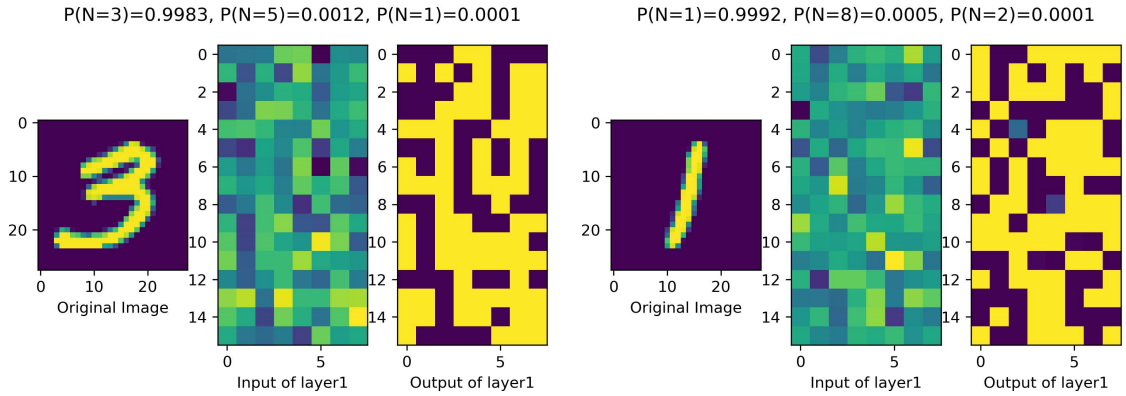


图 4：原始图片、隐藏层输入和输出

- 链接

Github repo：https://github.com/pitter-patterz/pj1

Model：https://pan.baidu.com/s/1fLbs9rh4dFOmK6e921Gslg?pwd=sjwl (提取码：sjwl)