Computer Vision - Homework 1

赵沁宜 18307110468 Project link: https://github.com/lrreel/DATA130051

- 模型结构
- 训练过程
- 参数查找
- 测试
- 可视化

模型结构：类定义

通过类定义神经网络基本结构，比如NNFC类定义了基本全连接层模块。 在类的初始化 `__init__` 中，定义模块相关参数及权重，并使用 `Xavier Initialization` 初始化权重。此外，定义 `self._training` 值为 `True` 时代表模型切换成训练模式，在forward过程中会保存中间计算结果； `self._training` 值为 `False` 时则跳过这一步骤。以NNFC类为例：

In [ ]:
```
class NNFC():
    # Full Connection Layer for Neural Network
    ...

    def initialization(self):
        """Xavier Initialization"""
        scale_factor = np.sqrt(1 / self.n_in)
        self.w = np.random.randn(self.n_in, self.n_out) * scale_factor
        return

    def forward(self, x):
        y = x @ self.w
        ...
        if self._training:
            # Keep the result
            self.X = x
            self.Y = y
        return y
```

backward中则利用上述保存的计算结果反向传播。以NNFC类为例：

In [ ]:
```
class NNFC():
    # Full Connection Layer for Neural Network
    ...
    def backward(self, grad_y, lr, ...):
        """_summary_
        Args:
            grad_y (_type_): the gradient of loss with respect to layer's output y
            lr (_type_): learning rate
        """
        ...
        grad_w = self.X.T @ grad_y
        self.w -= lr * grad_w
```

模型结构：NNClassifier

NNClassifier类定义了完整的两层神经网络分类器，模型结构由输入至输出依次为： 第一层全连接层，Sigmoid激活函数，第二层全连接层。

由于输入维度为MNIST数据集图片flatten长度（784），第一层全连接层输入维度为（batch size，784） 输出为（batch size，dim_hid） dim_hid隐藏层维度为超参；由于类别数为10，第二层全连接层输入（batch size，dim_hid）输出（batch size，10）

In [ ]:
```
class NNClassifier():
    def __init__(self, n_in, n_hid, n_out=10):
        self.n_hid = n_hid
        self.FC1 = NNFC(n_in, n_hid)
        self.FC2 = NNFC(n_hid, n_out)
        # Activation function
        self.Sigmoid = Sigmoid

        self._training = False
        ...


def Sigmoid(x):
    return 1 / (1 + np.exp(-x))
```

训练过程

任务使用MNIST数据集的训练数据及测试数据，同时再切割训练数据集为新的训练数据集和验证数据集。

在训练中，数据预处理将label数据转换成One-hot Encoder形式编码的向量，利用NNClassifer.forward()前向传播计算预测值。 使用MSE作为损失函数并加上L2正则化项，定义 `NNClassifierLoss` 计算损失函数，如下：

In [ ]:
```
class NNClassifierLoss():
    ...
    def __call__(self, y, target_y):
        """Calculate MSE loss and L2 regularization term

        Args:
            y (_type_): _description_
            target_y (_type_): _description_

        Returns:
            _type_: _description_
        """
        ...
        self.batch_size = y.shape[0]

        # MSE loss with size average
        loss_mse = MSELoss(y, target_y)

        # L2 Regularization
        reg_term = 0
        weights = [self.model.FC1.w, self.model.FC2.w]
        for weight in weights:
            reg_term += np.sum(np.square(weight))

        return 0.5 * loss_mse + self.lamda * reg_term
```

训练过程中，调用 `NNClassifierLoss.backward()` 使用SGD反向传播更新参数

In [ ]:
```
class NNClassifierLoss():
    ...
    def backward(self):
        """Stochastic Gradient Descending
        """
        grad_y = (self.Y - self.target_Y) / self.batch_size
        self.model.backward(grad_y, self.lr, self.lamda)
        return
```

训练过程：学习率衰减策略

在 `DECAY_PER_EPOCH` 轮Epoch之后使学习率衰减 `DECAY_FACTOR` 倍，如下所示

In [ ]:
```
    for i in range(epoch):
        # Decay strategy
        if i % DECAY_PER_EPOCH == 0:
            lr *= DECAY_FACTOR
```

实际训练中定义 `DECAY_FACTOR = 0.25` 和 `DECAY_PER_EPOCH = 150`

训练过程：保存模型

在训练过程中，记录在验证数据集上最小损失 `val_loss_best` 和最大准确率 `acc_best` 的模型，调用 `save_checkpoint` 方法保存。如下所示：

In [ ]:
```
    for i in range(epoch):
        ...
        # 使用SGD更新参数并在验证数据集上评估
        train_loss, valid_loss, valid_acc = training(i, lr, reg_term, model)
        ...
        if valid_loss <= val_loss_best and valid_acc >= acc_best:
            ...
            acc_best = valid_acc
            val_loss_best = valid_loss
            model_best = model

    save_checkpoint(model_best, ...)
```

训练过程：参数查找

使用Grid Search方法查找超参数，在训练过程中仅考虑学习率、隐藏层维度、正则化强度三个超参数，注意实际问题中可以考虑更多超参。 超参数网格设计如下

In [ ]:
```
# Grid Search
LR_GRID = [1e-2, 1e-3, 1e-4, 1e-5] # Learning rate
HID_GRID = [196, 98, 49] # Hidden layer imension
LAMBDA_GRID = [1, 0.5, 0.25] # Regularization term
```

参数查找算法即通过遍历网格中所有参数组合，训练模型，找到在测试数据集上loss最低及accuracy最高的参数组合。代码见 `grid_search.py` 。

测试及结果可视化

导入模型并从.json中读入参数查找的结果进行测试，可视化经过参数查找后模型在训练数据集及验证数据集上的loss曲线及accuracy，在测试数据集上的loss，accuracy，以及可视化网络参数，结果见 https://github.com/lrreel/DATA130051。