深度学习
Lab4-multilayer perceptron

兰韵诗

本次Lab有作业，请在3月17日结束前提交！

Lab2参考答案

- 随机梯度下降

```python
def train(self, x, y):
    '''
    x and y are the data for traning a linear regression
    please simply update the value of self.w and not include any other parameters
    '''

    # ==========
    # todo '''使用随机梯度下降法优化对self.w进行更新'''

    beta0 = np.expand_dims(np.ones_like(x), axis=1)
    beta1 = np.expand_dims(x, axis=1)
    x = np.concatenate([beta1, beta0], axis=1)

    for i in range(self.epoch):
        ids = np.arange(len(x))
        random.shuffle(ids)

        for j in ids:
            delta_w = []
            xii = x[j]
            yii = y_train[j]
            delta_w = np.dot(xii, yii - np.dot(xii, self.w))
            self.w += self.lr*delta_w
    # ==========
```

算法 2.1: 随机梯度下降法

输入: 训练集 $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$，验证集 $\mathcal{V}$，学习率 α

随机初始化 θ;
repeat
  对训练集 $\mathcal{D}$ 中的样本随机重排序;
  for $n = 1 \cdots N$ **do**
    从训练集 $\mathcal{D}$ 中选取样本 $(\mathbf{x}^{(n)}, y^{(n)})$;
    // 更新参数
    $\theta \leftarrow \theta - \alpha \frac{\partial \mathcal{L}(\theta; x^{(n)}, y^{(n)})}{\partial \theta}$;
  end
until 模型 $f(\mathbf{x}; \theta)$ 在验证集 $\mathcal{V}$ 上的错误率不再下降;
输出: θ

Why?

Lab2常见错误

SGD一次epoch仅随机取了一个样本更新梯度，而非遍历所有样本

```python
def train(self, x, y):
    '''
    x and y are the data for traning a linear regression
    please simply update the value of self.w and not include any other parameters
    '''

    # ==========
    # todo '''使用随机梯度下降法优化对self.w进行更新'''
    for _ in range(self.epoch):
        idx = random.randint(0, len(x)-1)
        xi = x[idx]
        yi = y[idx]
        x_vec = np.array([xi, 1.0])
        y_pred = np.dot(x_vec, self.w)

        error = y_pred - yi
        grad = error * x_vec

        self.w -= self.lr * grad

    # ==========
```

Lab2常见错误

使用全量梯度下降（BGD）而非SGD

python 复制

```python
def train(self, x, y):
    '''
    x and y are the data for traning a linear regression
    please simply update the value of self.w and not include any other parameters
    '''

    # ==========
    # todo '''使用随机梯度下降法优化对self.w进行更新'''
    for _ in range(self.epoch):
        y_pred = self.predict(x)
        error = y_pred - y
        beta0 = np.expand_dims(np.ones_like(x), axis=1)
        beta1 = np.expand_dims(x, axis=1)
        X = np.concatenate([beta1, beta0], axis=1)
        gradient = np.dot(X.T, error) / len(x)
        self.w = self.w - self.lr * gradient
    return self.w
    # ==========
```

未按照题目要求使用随机梯度下降法（SGD），而使用了全量梯度下降或小批量梯度下降法

Lab2常见错误

没有打乱顺序，每次随机抽取一个样本进行梯度更新，一个epoch抽取次数为样本数

```python
def train(self, x, y):
    '''
    x and y are the data for traning a linear regression
    please simply update the value of self.w and not include any other parameters
    '''

    # ==========
    # todo '''使用随机梯度下降法优化对self.w进行更新'''
    for _ in range(self.epoch):
        for i in range(len(x)):
            # 随机选择一个样本
            idx = random.randint(0, len(x) - 1)
            xi = x[idx]
            yi = y[idx]
            # 计算预测值
            y_pred = self.predict(np.array([xi]))
            # 计算梯度
            gradient = (y_pred - yi) * np.array([xi, 1])
            # 更新参数
            self.w -= self.lr * gradient

    # ==========
```

Lab2常见错误

没有打乱顺序，按顺序遍历所有样本，单样本更新梯度

```python
def train(self, x, y):
    '''
    x and y are the data for traning a linear regression
    please simply update the value of self.w and not include any other parameters
    '''

    # ==========
    # todo '''使用随机梯度下降法优化对self.w进行更新'''
    beta0 = np.expand_dims(np.ones_like(x), axis=1)
    beta1 = np.expand_dims(x, axis=1)
    X = np.concatenate([beta1, beta0], axis=1)

    for _ in range(self.epoch):
        for i in range(len(X)):
            prediction = np.dot(X[i], self.w)
            error = prediction - y[i]
            self.w -= self.lr * error * X[i]
    # ==========
```

Lab2常见错误

未添加偏置项，导致偏置项的梯度更新错误

```python
def train(self, x, y):
    '''
    x and y are the data for traning a linear regression
    please simply update the value of self.w and not include any other parameters
    '''

    # ==========
    # todo '''使用随机梯度下降法优化对self.w进行更新'''

    m = len(x)
    for epoch in range(self.epoch):
        # 遍历每一个训练样本：预测输出，计算损失函数的梯度、损失函数对参数的梯度，更新权重
        for i in range(m):
            xi = np.expand_dims(x[i], axis=0)
            yi = y[i]
            prediction = self.predict(xi)
            error = prediction - yi

            grad = xi.T * error
            # 使用学习率更新参数，squeeze将维度调整为和self.w一致
            self.w -= self.lr * grad.squeeze()

    # ==========
```

Lab4

1.理解图像识别的代码实现流程
2.补全MLP模型
3.比较numpy版本和pytorch版本的MLP模型实现

Multilayer Perceptron

- 读懂exercise_mlp.py文件中的代码，熟悉图像识别任务的代码流程，可通过display_mnist看到数据库的图像
- 理解pytorch版本的MLP代码实现

Multilayer Perceptron

- 用numpy实现基于**MLP模型**的图像分类任务
  - 完成ReLU激活和softmax函数的**梯度后传**
  - 利用设定好的对象属性**补全MLP前向传，后向传播**和**参数更新**
  - 不能修改给定的对象属性，不能调用其他工具包， 只能在 “to do” 下面书写代码
  - 提交之后，测试集上的准确率应该降到一个正确的范围内可多次提交。即使对自己的代码没有自信也一定要提交，我们会酌情给过程分
- **TO DO**： 完成《Multilayer Perceptron》项目。补全feedforward_np_version.py文件使exercise_mlp.py文件中的train_with_Model_NP()可以顺利执行。

Evaluation脚本

```
def compute_acc(pred_file):
    with gzip.open(r'.\data\t10k-labels.gz', 'rb') as f:
        gold = np.frombuffer(f.read(), np.uint8, offset=8)

    with open(pred_file) as f:
        pred = f.readlines()
    pred = [int(sent.strip()) for sent in pred]
    correct_case = [i for i, _ in enumerate(gold) if gold[i] == pred[i]]

    acc = len(correct_case)*1./len(gold)
    print('The predicted accuracy is %s' %acc)

if __name__ == '__main__':
    pred_file = 'data/predict.txt'
    compute_acc(pred_file)
```