

# 深度学习

## Lab4-multilayer perceptron

兰韵诗

**本次Lab有作业，请在3月28日结束前提交！**

# Lab2参考答案

## • 小批量随机梯度下降

```
def train(self, x, y, k = 8):  
    '''  
    x and y are the data for training a linear regression  
    k is the batch size  
    please simply update the value of self.w and not include any other parameters  
    '''  
  
    # =====  
    # todo '''使用小批量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)  
  
        iter_num = int(np.ceil(len(x)*1./k))  
        for n in range(iter_num):  
            delta_w = []  
            for j in ids[n*k: (n+1)*k]:  
                xii = x[j]  
                yii = y_train[j]  
                delta_w += [np.dot(xii, yii - np.dot(xii, self.w))] |  
            self.w += self.lr*(np.mean(delta_w, 0))  
  
    # =====
```

---

### 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  $\alpha$

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



# Lab2常见错误

```
def train(self, x, y, k = 8):  
    '''  
    x and y are the data for training a linear regression  
    k is the batch size  
    please simply update the value of self.w and not include any other parameters  
    '''  
  
    # =====  
    # todo '''使用小批量随机梯度下降法优化对self.w进行更新'''  
    theta0 = np.expand_dims(np.ones_like(x), axis=1)  
    theta1 = np.expand_dims(x, axis=1)  
    theta = np.concatenate([theta1, theta0], axis=1)  
    for i in range(self.epoch):  
        print("循环次数: ", i, "参数 w_train = ", self.w)  
        grad = theta.T.dot(theta.dot(self.w)-y)*2.0/len(theta)  
        self.w=self.w-self.lr*grad  
    # =====
```

- 扣分点：没有用到小批量

## Lab2常见错误

```
def train(self, x, y, k=8):  
  
    n = x.shape[0]  
    for _ in range(self.epoch):  
        for i in range(0, n, k):  
  
            x_batch = x[i:i+k]  
            y_batch = y[i:i+k]  
  
            beta0 = np.expand_dims(np.ones_like(x_batch), axis=1)  
            beta1 = np.expand_dims(x_batch, axis=1)  
            x_batch = np.concatenate([beta1, beta0], axis=1)  
  
            y_pred = np.dot(x_batch, self.w)  
            error = y_pred - y_batch  
  
            gradient = np.dot(x_batch.T, error) / k  
            self.w -= self.lr * gradient  
  
    return self.w
```

- 扣分点：没有随机

## Lab2常见错误

- 扣分点：调用adam

```
num_batch = x.shape[0] // k
    for epoch in range(self.epoch):

        for batch in range(num_batch):
            params = {"w": self.w}
            optimizer = Adam(params, lr=self.lr)
            grad = {}

            batch_x = x[batch * k: (batch + 1) * k]
            batch_y = y[batch * k: (batch + 1) * k]

            predictions = self.predict(batch_x)
            errors = predictions - batch_y
            dw = 2 * np.dot(batch_x.T, errors) / k
            grad['w'] = dw
            self.w = optimizer.update(params, grad)['w']

#         self.w -= dw * self.lr
```

## Lab2常见错误

```
w = self.w
for i in range(self.epoch):
    np.random.shuffle(data)
    x_batch = data[0: k, :2]
    y_batch = data[0:k, 2:3]
    grads = 0
    for i in range(k):
        x_ = np.array(x_batch[i])
        y_ = np.array(y_batch[i])
        x_.resize(len(x_), 1)
        y_.resize(len(y_), 1)
        c = np.matmul(-1 * x_, y_ - np.matmul(x_.T, w))
        grads = grads + c[0]

    grads = grads / k
    w = w - grads * self.lr
```

- 扣分点：只选取第一个批量做梯度下降

# 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)
```