学　　号：21110850017

復旦大學

神经网络和深度学习

Neural Network and Deep Learning

院　　系：类脑智能科学与技术研究院

专　　业：应用数学

姓　　名：翟荣泉

指 导 教 师：赵冬华　教授

完 成 日 期：2021 年 11 月 4 日

目　录

插图

第 1 章　第一题

1.1　使用波士顿房价数据集，以线性回归为例 (或者自行设定模型) 实现以及对比梯度下降方式：随机梯度下降、牛顿法、Adagrad。

三种梯度下降的原理及公式：

1. 随机梯度下降

随机梯度下降在计算下降最快的方向时时随机选一个数据进行计算，而不是扫描全部训练数据集，这样就加快了迭代速度。随机梯度下降并不是沿着 $J(\theta)$ 下降最快的方向收敛，而是震荡的方式趋向极小点。随机梯度下降表达式如下：

Loop {

for i=1 to m,{

$\square_j := \theta_j + \alpha\left(y^{(i)} - h_\theta\left(x^{(i)}\right)\right) x_j^{(i)}$　(for every j)

}

2.Adagrad

AdaGrad 算法会使用一个小批量随机梯度 $\boldsymbol{g}_t$ 按元素平方的累加变量 $\boldsymbol{s}_t$。在时间步 0，AdaGrad 将 $\boldsymbol{s}_0$ 中每个元素初始化为 0。在时间步 t，首先将小批量随机梯度 $\boldsymbol{g}_t$ 按元素平方后累加到变量 $\boldsymbol{s}_t$：

$$\boldsymbol{s}_t \leftarrow \boldsymbol{s}_{t-1} + \boldsymbol{g}_t \odot \boldsymbol{g}_t,$$

其中 $\odot$ 是按元素相乘。接着，我们将目标函数自变量中每个元素的学习率通过按元素运算重新调整一下：

$$\boldsymbol{x}_t \leftarrow \boldsymbol{x}_{t-1} - \frac{\eta}{\sqrt{\boldsymbol{s}_t + \epsilon}} \odot \boldsymbol{g}_t,$$

其中 η 是学习率，ϵ 是为了维持数值稳定性而添加的常数，如 10^{-6}。这里开方、除法和乘法的运算都是按元素运算的。这些按元素运算使得目标函数自变量中每个元素都分别拥有自己的学习率。

3. 牛顿法

对于非线性优化问题, 牛顿法提供了一种求解的办法。假设任务是优化一个目标函数 f, 求函数 f 的极大极小问题, 可以转化为求解函数的导数 $f\,' = 0$ 的问题, 这样求可以把优化问题看成方程求解问题 $\left(f' = 0\right)$。剩下的问题就和第一部分提到的牛顿法求解很相似了。这次为了求解 $f' = 0$ 的根，把 $f(x)$ 的泰勒展开，展开到 2 阶形式：

$$f(x + \Delta x) = f(x) + f'(x)\Delta x + \frac{1}{2} f''(x)\Delta x^2$$

这个式子是成立的，当且仅当 Δx 无线趋近于 0。此时上式等价与:

$$f'(x) + f''(x)\Delta x = 0$$

求解:

$$\Delta x = -\frac{f'\left(x_n\right)}{f''\left(x_n\right)}$$

得出迭代公式:

$$x_{n+1} = x_n - \frac{f'\left(x_n\right)}{f''\left(x_n\right)}, n = 0, 1, \ldots$$

三种方法列表比较:

为了从各个方面比较这三种优化算法的特点，本题采用波士顿房价数据集进行实例分析，本次比较拟合了两个模型，拟合了一个只有一个隐藏层，神经元为 100 个，输入维度为 13（数据中的特征数），输出维度为 1（预测目标），损失函数为 MSE 的多层感知机模型，采用 pytorch 自带的 SGD 优化器及 Adagrad 优化器，在迭代次数相同（均为 1000 次）分别比较两种优化器的性能。代码详见 MLP 回归预测并比较 SGD 和 Adagrad.py

$$optimizer = torch.optim.SGD(net.parameters(), lr = 0.01)$$

$$optimizer = torch.optim.Adagrad(net.parameters(), lr = 0.01)$$

二者的初始学习率均为 0.01。数据集一共 506 个样本，其中 404 个用来训练，102 个用于测试。在相同迭代次数下两种优化器的结果为:

$testloss$:0.02234669029712677(Adagrad)

$testloss$:0.5019428133964539(SGD)

可见 Adagrad 的结果更好，带有动量的优化算法表现更好。

对于牛顿法，由于 pytorch 中没有相应的优化器，且 pytroch 计算图框架不能计算二阶导，所以要手动算出需要更新参数的二阶导，一阶导还可以利用 pytorch 中的计算图自行计算。此次采用的模型为普通的线性回归模型，此例为了展示为何牛顿法理论上收敛迅速却没有被广泛采用。代码详见使用线性回归模型并用牛顿法做优化.py

当损失函数为 MSE 时，线性回归模型中的 ω 和 b 的二阶导数通过推导可知为 x_i^2 和-2，一阶导可通过 pytorch 自动求导算出，所以更新公式为:

```
with torch.no_grad():
    w -= w.grad /(traindata[i]**2)
    b -= b.grad /2
    w.grad.zero_()
    b.grad.zero_()
```

图 1-1

fig: 牛顿法更新

在参数随机初始化时，loss 为：108055.2734 迭代后的 loss 为：21.0477 loss 下降了，但在实际操作中发现，牛顿法并不容易收敛，极其容易数值溢出，原因在于要计算两个导数相

除，容易造成梯度爆炸，且不同损失函数的 Hessian 矩阵求逆不一定存在，故我认为这是牛顿法没有被 pytorch 框架采用的原因。

三种方法总结如下：

	优点	缺点
SGD	计算代价小，收敛快	有可能会陷入局部最小值； 不会收敛，最终会一直在最小值附近波动，并不会达到最小值并停留在此； 下降速度慢； 选择合适的 learning rate 比较困难； 在所有方向上统一的缩放梯度，不适用于稀疏数据
Adagrad	不同的变量提供不同的学习率。减少摆动，在稀疏数据场景下表现会非常好；允许使用一个更大的学习率 α，从而加快算法的学习速度；这个在后期，会通过分母来整体减小学习率。	因为是不断累积单调递增的，会使得学习率单调递减至 0，可能会使得训练过程提前结束，即使后续还有数据也无法学到需要的知识。
Newton	收敛快，利用了二阶信息	梯度相除容易出问题，DL 中很少用

图 1-2

fig: 比较

第 2 章　案例计算：分别使用误差率，信息熵，基尼指数来计算计算下图不纯度。哪个不纯度最高？哪个不纯度最低？

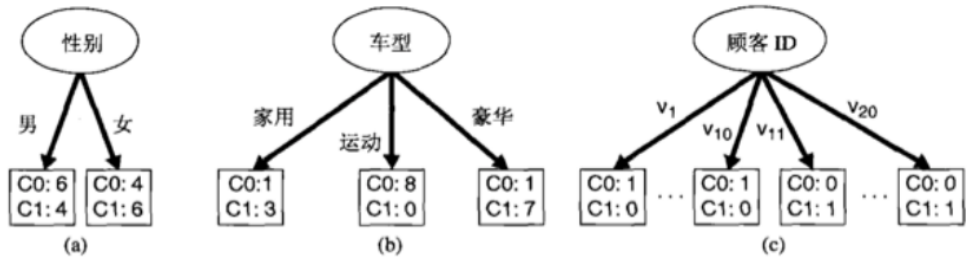

图 2-1

fig: 三种属性构造决策树分类

(a)

误差率: $1-\frac{6}{10}=\frac{4}{10}$

信息熵：$(1)=-\frac{4}{10}\log_2\frac{4}{10}(2)=-\frac{6}{10}\log_2\frac{6}{10}$

Entropy $=(1)\times\frac{1}{2}+(2)\times\frac{1}{2}=0.9710$

基尼指数: $(1)=1-\left(\left(\frac{6}{10}\right)^2+\left(\frac{4}{10}\right)^2\right)$

$(2)=1-\left(\left(\frac{4}{10}\right)^2+\left(\frac{6}{10}\right)^2\right)$

Gini $=\frac{1}{2}\times(1)+\frac{1}{2}\times(2)$

$=1-\left(\left(\frac{4}{10}\right)^2+\left(\frac{6}{10}\right)^2\right)=0.48$

误差率: $1-\frac{7}{8}=\frac{1}{8}$

信息熵:

(1) $=-\frac{1}{4}\log_2\frac{1}{4}-\frac{3}{4}\log_2\frac{3}{4}$

(2) $=-1\log_2 1-0\log_2 0$

(3) $=-\frac{1}{8}\log_2\frac{1}{8}-\frac{7}{8}\log\frac{7}{8}$

(b) Entropy $=\frac{1}{5}\times(1)+\frac{2}{5}\times(2)+\frac{2}{5}\times(3)=0.3797$

基尼指数: $(1)=1-\left(\left(\frac{1}{4}\right)^2+\left(\frac{3}{4}\right)^2\right)$

$(2)=1-\left((1)^2+(0)^2\right)$

(3) $=1-\left(\left(\frac{1}{8}\right)^2+\left(\frac{7}{8}\right)^2\right)$

Gini $=\frac{1}{5}\times(1)+\frac{2}{5}\times(2)+\frac{2}{5}\times(3)=0.1625$

(c) 误差率：1-1=0

信息熵：$\sum_{i=1}^{20}\frac{1}{20}\times 0=0$

Gini：$\sum_{i=1}^{20}\frac{1}{20}\left(1-1^2\right)=0$

综上，(a) 不纯度最高，(c) 的不纯度最低。以上三个指标都是越大纯度越低。