Homework 2

张思源　　*21110850018*

November 4, 2021

1 Ex1

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.
NOTE:1. 需要附有梯度下降表达式.2. 列表对比三种方法的区别.

Sol 1.1 *首先计算梯度, 在此之前计算 MSE 函数表达式为:*

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(w^T x_i + b - y_i)^2$$

其中 n 为样本数, 在此基础上, 计算其梯度为:

$$\nabla_w MSE = \frac{2}{n}\sum_{i=1}^{n}(w^T x_i + b - y_i)x_i$$

$$\nabla_b MSE = \frac{2}{n}\sum_{i=1}^{n}(w^T x_i + b - y_i)$$

更进一步的, 其 Hessian 矩阵为:

$$H(w) = \frac{2}{n}X^T X$$

$$H(b) = \frac{2}{n}I$$

因此, 三种优化方法的迭代公式为 (为了简便表示, 记参数为 $\theta = (w, b)$):

- *SGD 的迭代表达式为:*

  $$\theta_{t+1} \leftarrow \theta_t - \eta \nabla MSE_i,$$

  其中 η 为学习率,MSE_i 为随机样本的 MSE 函数.

- *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:

  $$\theta_{t+1} \leftarrow \theta_t - \frac{\eta}{\sqrt{s_t + \epsilon}} \odot \boldsymbol{g}_t$$

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.

- *Newton* 法的迭代表达式为:

$$\theta_{t+1} \leftarrow \theta_t - H^{-1}\nabla MSE.$$

这里 H 为 *Hessian* 矩阵.

而数值实验的结果如下图所示:

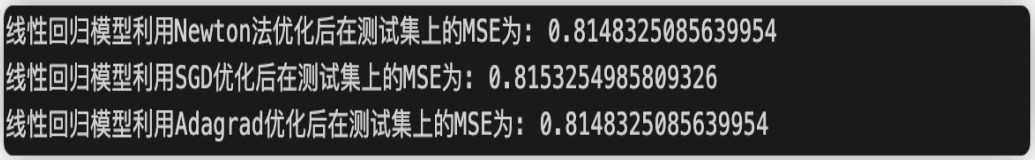

Figure 1: Results of 3 different methods of optimization

接下来, 比较三种不同的优化方法:

算法	优点	缺点
SGD	计算速度快, 计算成本低	*1.* 选择合适的 *learning rate* 比较困难, 对所有的参数更新使用同样的 *learning rate.2.* 容易收敛到局部最优, 并且在某些情况下可能被困在鞍点.
Adagrad	前期 g_t 较小的时候,*regularizer* 较大, 能够放大梯度; 后期 g_t 较大的时候,*regularizer* 较小, 能够约束梯度. 适合处理稀疏梯度.	*1.*η 设置过大的话, 会使 *regularizer* 过于敏感, 对梯度的调节太大 *2.* 中后期, 分母上梯度平方的累加将会越来越大, 使 $g_t \to 0$, 使得训练提前结束.
Newton	收敛速度快, 二阶方法精确度高	*1.* 容易收敛到鞍点 *2.* 计算逆矩阵容易出现数值不稳定

2 Ex2

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?

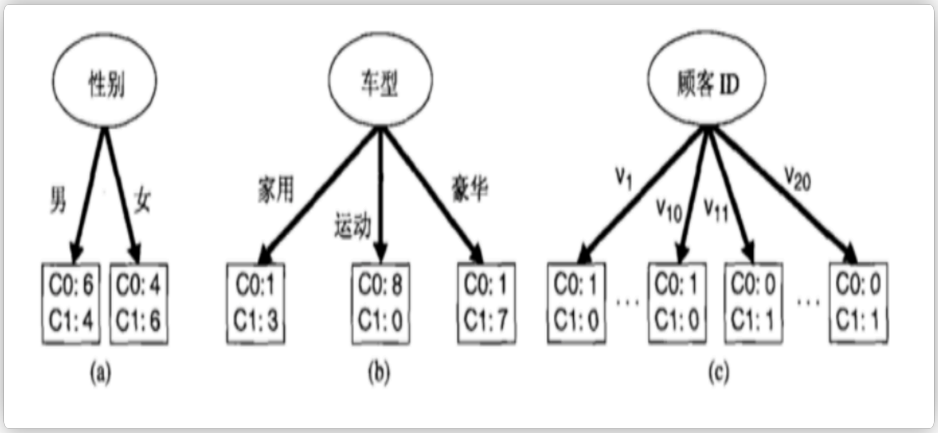

Figure 2: Ex2

Sol 2.1 在计算中，以 Ce 表示误差率，以 $Entropy$ 表示信息熵，以 $Gini$ 表示基尼系数.
则对 (a) 图，注意到两个节点的对称性，有:

$$
\begin{aligned}
Ce(a) &= 1 - \max(0.4, 0.6) = 0.4 \\
Entropy(t_1) &= -(0.4 \times \log_2 0.4 + 0.6 \times \log_2 0.6) = 0.9710 = Entropy(t_2) \\
Entropy(a) &= \frac{6+4}{20} Entropy(t_1) + \frac{4+6}{20} Entropy(t_2) = 0.9710 \\
Gini(t_1) &= 1 - (0.6^2 + 0.4^2) = 0.48 = Gini(t_2) \\
Gini(a) &= \frac{6+4}{20} Gini(t_1) + \frac{4+6}{20} Gini(t_2) = 0.48
\end{aligned}
$$

对图 (b)，有:

$$
\begin{aligned}
Ce(b) &= 1 - \max(0.75, 0, 0.875) = 0.125 \\
Entropy(t_1) &= -(0.25 \times \log_2 0.25 + 0.75 \times \log_2 0.75) = 0.8113 \\
Entropy(t_2) &= -(1 \times \log_2 1 + 0 \times \log_2 0) = 0 \\
Entropy(t_3) &= -(0.125 \times \log_2 0.125 + 0.875 \times \log_2 0.875) = 0.5436 \\
Entropy(b) &= \frac{1+3}{20} Entropy(t_1) + \frac{8+0}{20} Entropy(t_2) + \frac{1+7}{20} Entropy(t_3) = 0.3797 \\
Gini(t_1) &= 1 - (0.25^2 + 0.75^2) = 0.375 \\
Gini(t_2) &= 1 - (1^2 + 0^2) = 0 \\
Gini(t_3) &= 1 - (0.125^2 + 0.875^2) = 0.21875 \\
Gini(b) &= \frac{1+3}{20} Gini(t_1) + \frac{8+0}{20} Gini(t_2) + \frac{1+7}{20} Gini(t_3) = 0.1625
\end{aligned}
$$

对图 (c), 同样注意到对称性, 有:

$$
\begin{aligned}
Ce(c) &= 1 - \max(0, \ldots, 0, 1, \ldots, 1) = 0 \\
Entropy(t_i) &= -(1 \times \log_2 1 + 0 \times \log_2 0) = 0, i = 1, 2, \ldots, 10 \\
Entropy(t_j) &= -(0 \times \log_2 0 + 1 \times \log_2 1) = 0, j = 11, 12, \ldots, 20 \\
Entropy(c) &= \sum_{i=1}^{20} \frac{1}{20} Entropy(t_i) = 0 \\
Gini(t_i) &= 1 - (1^2 + 0^2) = 0, i = 1, 2, \ldots, 10 \\
Gini(t_j) &= 1 - (0^2 + 1^2) = 0. j = 11, 12, \ldots, 20 \\
Gini(c) &= \sum_{i=1}^{20} \frac{1}{20} Gini(t_i) = 0
\end{aligned}
$$

故有:

$$
\begin{aligned}
&Ce(a) = 0.4 > Ce(b) = 0.125 > Ce(c) = 0 \\
&Entropy(a) = 0.9710 > Entropy(b) = 0.3797 > Entropy(c) = 0 \\
&Gini(a) = 0.48 > Gini(b) = 0.1625 > Gini(c) = 0
\end{aligned}
$$

即图 (a) 的不纯度最大, 图 (c) 的不纯度最小.

References

[1] 李航. 统计学习方法 [M]. 清华大学出版社, 2012.

[2] Goodfellow, Ian, et al. Deep Learning[M]. MIT Press, 2016.