机器学习经典算法推导

黎雷蕾

2018 年 8 月 8 日

目录

Chapter 1

代数基础

1.1 损失函数

1.1.1 0-1 损失 (Zero-one Loss)

最简单的损失函数，如果预测值 $\hat{y}_i$ 与目标值 y_i 不相等，那么为 1，否则为 0.

$$\ell(y_i, \hat{y}_i) = \begin{cases} 1, & y_i \neq \hat{y}_i; \\ 0, & y_i = \hat{y}_i. \end{cases} \tag{1.1}$$

1.1.2 感知损失 (Perceptron Loss)

用来改进 0-1 损失中判定较为严格的问题:

$$\ell(y_i, \hat{y}_i) = \begin{cases} 1, & |y_i - \hat{y}_i| > t \\ 0, & |y_i - \hat{y}_i| \leq t. \end{cases} \tag{1.2}$$

1.1.3 Hinge Loss

Hinge Loss 可以用来解决 SVM 只能够的间隔最大化问题。

$$\begin{aligned} \ell(y_i, \hat{y}_i) = \max\{0, 1 - y_i \cdot \hat{y}_i\}&, \\ y_i \in \{-1, +1\}, \quad \hat{y}_i \in [-1, +1]&. \end{aligned} \tag{1.3}$$

1.1.4 绝对值误差

常用回归中:

$$\ell(y_i, \hat{y}_i) = |y_i - \hat{y}_i| \tag{1.4}$$

1.1.5 均方误差

常用于回归中:

$$\ell(y_i, \hat{y}_i) = \frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2, \tag{1.5}$$

1.1.6 交叉熵

神经网络、逻辑回归中常用的损失函数，二分类问题可写作:

$$\begin{aligned} \ell(y_i, \hat{y}_i) &= y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i) \\ y_i &\in \{0, 1\} \end{aligned} \tag{1.6}$$

多分类问题可写作:

$$\ell(y_i, \hat{y}_i) = -\sum_{i=0}^{n} y_i \log(\hat{y}_i) \tag{1.7}$$

1.1.7 指数误差 (Exponential)

常用于 boosting 算法:

$$\ell(y_i, \hat{y}_i) = \exp(-y_i \cdot \hat{y}_i) \tag{1.8}$$

1.2 数值优化算法

1.2.1 牛顿法 (Newton's method)

牛顿法是是一种在实数域和复数域上近似求解方程的方法。方法使用函数 $f(x)$ 的泰勒级数的前面几项来寻找方程 $f(x) = 0$ 的根。也被称为切线法。

将 $f(x)=0$ 在 x_0 处展开成泰勒级数：

$$f(x_0) \to \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x-x_0)^n \tag{1.9}$$

我们只取其线性部分，作为非线性方程的近似方程：

$$f(x_0)+(x-x_0)f'(x_0)=0 \tag{1.10}$$

设 $f'(x_0)\neq 0$，则其解为：

$$x=x_1=x_0-\frac{f(x_0)}{f'(x_0)} \tag{1.11}$$

这个公式说明 $f(x_1)$ 的值将会比 $f(x_0)$ 更加接近 $f(x)=0$，我们就可以用迭代法逼近：

$$x_{n+1}=x_n-\frac{f(x_n)}{f'(x_n)} \tag{1.12}$$

最好是用二阶导数:

$$x_{n+1}=x_n-\frac{f'(x_n)}{f''(x_n)} \tag{1.13}$$

通过迭代，上式必然会在 $f(x)=0$ 处收敛。

1.2.2 拟牛顿法 (Quasi-Newton Methods)

拟牛顿法 (Quasi-Newton Methods) 是求解非线性优化问题最有效的方法之一。

海森矩阵 (Hessian Matrix)

海森矩阵是一个多元函数的二阶偏导数构成的方阵，描述了函数的局部曲率。假设二元函数 $f(x_1,x_2)$ 在 $\mathbf{X}^{(0)}(x_1^{(0)},x_2^{(0)})$ 点处的泰勒展开式为：

$$\begin{aligned}
&f(x_1,x_2)= \\
&f(x_1^{(0)},x_2^{(0)})+\left.\frac{\partial f}{\partial x_1}\right|_{\mathbf{X}^{(0)}}\Delta x_1+\left.\frac{\partial f}{\partial x_2}\right|_{\mathbf{X}^{(0)}}\Delta x_2+ \\
&\frac{1}{2}\left[\left.\frac{\partial^2 f}{\partial x_1^2}\right|_{\mathbf{X}^{(0)}}\Delta x_1^2+2\left.\frac{\partial^2 f}{\partial x_1\partial x_2}\right|_{\mathbf{X}^{(0)}}\Delta x_1\Delta x_2+\left.\frac{\partial^2 f}{\partial x_2^2}\right|_{\mathbf{X}^{(0)}}\Delta x_2^2\right]+\cdots
\end{aligned} \tag{1.14}$$

其中，$\Delta x_1 = x_1 - x_1^{(0)}, \Delta x_2 = x_2 - x_2^{(0)}$.

将上式写成矩阵形式：

$$
\begin{aligned}
& f(\mathbf{X}) = \\
& f(\mathbf{X}^{(0)}) + \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}\right)_{\mathbf{X}^{(0)}} \begin{pmatrix} \Delta x_1 \\ \Delta x_2 \end{pmatrix} + \\
& \frac{1}{2}(\Delta x_1, \Delta x_2) \left.\begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{pmatrix}\right|_{\mathbf{X}^{(0)}} \begin{pmatrix} \Delta x_1 \\ \Delta x_2 \end{pmatrix} + \cdots
\end{aligned} \tag{1.15}
$$

即:

$$
f(\mathbf{X}) = f(\mathbf{X}^{(0)}) + \nabla f(\mathbf{X}^{(0)})^T \Delta \mathbf{X} + \frac{1}{2} \Delta \mathbf{X}^T H(\mathbf{X}^{(0)}) \Delta \mathbf{X} + \cdots \tag{1.16}
$$

其中:

$$
H(\mathbf{X}^{(0)}) = \left.\begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{pmatrix}\right|_{\mathbf{X}^{(0)}}, \ \Delta \mathbf{X} = \begin{pmatrix} \Delta x_1 \\ \Delta x_2 \end{pmatrix} \tag{1.17}
$$

$H(\mathbf{X}^{(0)})$ 是 $f(x_1, x_2)$ 在 $\mathbf{X}^{(0)}$ 处的海森矩阵，由 $f(x_1, x_2)$ 在 $\mathbf{X}^{(0)}$ 处的二阶偏导数组成。

将海森矩阵扩展到 n 元函数，对应的梯度 $\nabla f(\mathbf{X}^{(0)})$ 和海森矩阵可以写作 $H(\mathbf{X}^{(0)})$:

$$
\nabla f(\mathbf{X}^{(0)}) = \left.\left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_n}\right]\right|_{\mathbf{X}^{(0)}}^T \tag{1.18}
$$

$$
H(\mathbf{X}^{(0)}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}_{\mathbf{X}^{(0)}} \tag{1.19}
$$

拟牛顿法思想

在牛顿法的迭代中，需要计算海森矩阵的逆矩阵 H^{-1}，这个计算十分复杂，所以考虑到用一个 n 阶矩阵 $G_k = G(x^{(k)})$ 来近似代替 $H_k^{-1} = H^{-1}(x^{(k)})$，这就是拟牛顿法的基本思想了。

假设 $g_k = g(x^{(k)}) = \nabla f(x^k)$ 是 $f(x)$ 的梯度向量在点 $x^{(k)}$ 的值。那么牛顿法的更新公式就可以写作：

$$x^{(k+1)} = x^{(k)} - H_k^{-1} g_k \tag{1.20}$$

拟牛顿法的公式可以写作：

$$x^{(k+1)} = x^{(k)} - G_k g_k \tag{1.21}$$

Davidon Fletcher Powell, DFP 算法

DFP 选择 G_{k+1} 的方法是假设每一步迭代中矩阵 G_{k+1} 是由 G_k 加上两个附加项构成的：

$$G_{k+1} = G_k + P_k + Q_k \tag{1.22}$$

设 $y_k = g_{k+1} - g_k$，$\delta_k = x^{(k+1)} - x^{(k)}$，那么可以求出 P_k, Q_k：

$$\begin{aligned} P_k &= \frac{\delta_k \delta_k^T}{\delta_k^T y_k}, \\ Q_k &= -\frac{G_k y_k y_k^T G_k^T}{y_k^T G_k y_k} \end{aligned} \tag{1.23}$$

那么 DFP 算法中 G_{k+1} 的迭代公式可以写作：

$$G_{k+1} = G_k + \frac{\delta_k \delta_k^T}{\delta_k^T y_k} - \frac{G_k y_k y_k^T G_k^T}{y_k^T G_k y_k} \tag{1.24}$$

那么优化迭代公式可以写作：

$$\begin{aligned} x^{(k+2)} &= x^{(k+1)} - G_{k+1} g_{k+1} \\ &= x^{(k+1)} - \left(G_k + \frac{\delta_k \delta_k^T}{\delta_k^T y_k} - \frac{G_k y_k y_k^T G_k^T}{y_k^T G_k y_k} \right) g_{k+1} \end{aligned} \tag{1.25}$$

Broyden Fletcher Goldfarb Shanno, BFGS 算法

与 DFP 算法不同的是，BFGS 采用一个容易求解逆矩阵的 B_k 去逼近海森矩阵本身 H，而不是海森矩阵逆矩阵 H^{-1}。

那么 BFGS 的迭代公式可以写为:

$$\begin{aligned} x^{(k+2)} &= x^{(k+1)} - B_{k+1}^{-1} g_{k+1} \\ &= x^{(k+1)} - \left(B_k + \frac{y_k y_k^T}{y_k^T \delta_k} - \frac{B_k \delta_k \delta_k^T B_k^T}{\delta_k^T B_k \delta_k} \right)^{-1} g_{k+1} \end{aligned} \tag{1.26}$$

1.2.3 梯度下降 (Gradient descent)

梯度

所谓梯度，就是指函数变化最快的地方。对于一个函数 $f(x,y)$，分别对于 x,y 求偏导，获得的梯度向量为 $(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y})^T$，简称为 $grad\ f(x,y)$ 或者 $\nabla f(x,y)$。梯度方向指的是沿着梯度向量 $\nabla f(x,y)$ 的方向。

梯度下降和梯度上升

两者的实质是一样的，梯度下降取相反数就是梯度上升了。

梯度下降的代数方法表示

假设一个线性回归函数的表示公式为:

$$h_\theta(x) = \sum_{i=0}^{n} \theta_i x_i \tag{1.27}$$

损失函数取均方误差:

$$J(\theta) = \frac{1}{n} \sum_{i=0}^{n} (h_\theta(x) - y_i)^2 \tag{1.28}$$

求出 $J(\theta)$ 的梯度:

$$\frac{\partial J(\theta)}{\partial \theta_i} = \frac{2}{n} \left(\sum_{i=0}^{n} \frac{\partial h_\theta(x)}{\partial \theta_i} - y_i \right) x_i^{(j)} \tag{1.29}$$

那么更新的梯度表达式可以写作:

$$\theta_i' = \theta_i - \alpha \frac{\partial J(\theta)}{\partial \theta_i} \tag{1.30}$$

其中 $\alpha \in [0,1]$ 称为学习步长，取值过大会造成震荡，太小会造成收敛过慢。

梯度下降的矩阵方法表示

假设一个线性回归函数的矩阵表示公式为：

$$h_\theta(\mathbf{x}) = \mathbf{X}\theta \tag{1.31}$$

那么损失函数可以表示为:

$$J(\theta) = \frac{1}{2}(\mathbf{X}\theta - \mathbf{Y})^T(\mathbf{X}\theta - \mathbf{Y}) \tag{1.32}$$

求出 $J(\theta)$ 的梯度：

$$\begin{aligned}
&\frac{\partial}{\partial \mathbf{X}}(\mathbf{X}\mathbf{X}^T) = 2\mathbf{X} \\
&\frac{\partial}{\partial \theta}(\mathbf{X}\theta) = \mathbf{X}^T \\
&\frac{\partial}{\partial \theta}J(\theta) = \mathbf{X}^T(\mathbf{X}\theta - \mathbf{Y})
\end{aligned} \tag{1.33}$$

那么更新的梯度表达式可以写作：

$$\theta_i' = \theta_i - \alpha\frac{\partial}{\partial \theta_i}J(\theta) \tag{1.34}$$

梯度下降参数调优

1. 步长 α：太大会造成震荡从而错过最优解，太小会造成迭代速度太慢。
2. 参数初始值选择，梯度下降求出的是局部最优解，所以参数初始值不同求出的解也会不同，最好是正态分布随机生成。
3. 归一化，可以加快迭代速度。

与其它优化算法比较

梯度下降法和最小二乘法相比，梯度下降法需要选择步长，而最小二乘法不需要。梯度下降法是迭代求解，最小二乘法是计算解析解。如果样本量不算很大，且存在解析解，最小二乘法比起梯度下降法要有优势，计算速度很快。但是如果样本量很大，用最小二乘法由于需要求一个超级大的逆矩

阵，这时就很难或者很慢才能求解解析解了，使用迭代的梯度下降法比较有优势。

梯度下降法和牛顿法/拟牛顿法相比，两者都是迭代求解，不过梯度下降法是梯度求解，而牛顿法/拟牛顿法是用二阶的海森矩阵的逆矩阵或伪逆矩阵求解。相对而言，使用牛顿法/拟牛顿法收敛更快。但是每次迭代的时间比梯度下降法长。

1.2.4 Momentum

Momentum 借鉴了物理学中动量的思想，通过积累之前的动量 m_{t-1} 来加速当前的梯度。设 μ 是动量因子，通常设为 0.9 或其近似值：

$$
\begin{aligned}
m_t &= \mu \cdot m_{t-1} + \alpha \nabla J(\theta) \\
\theta_t' &= \theta_t - m_t
\end{aligned}
\tag{1.35}
$$

特点：

- 接近局部最优解时，μ 的存在可以使更新幅度变大，从而跳出局部最优；
- 当前后两次梯度方向一致时，可以加速收敛，当前后两次梯度方向不一致时，可以减少震荡。

1.2.5 Nesterov

Nesterov 是基于 Momentum 的算法，可以在梯度更新时对当前梯度进行一个矫正，避免前进太快，同时提高灵敏度：

$$
\begin{aligned}
m_t &= \mu \cdot m_{t-1} + \alpha \nabla J(\theta - \mu \cdot m_{t-1}) \\
\theta_t' &= \theta_t - m_t
\end{aligned}
\tag{1.36}
$$

1.2.6 Adagrad

Adagrad 对学习速率进行了一个约束。设梯度 $g_t = \nabla_\theta J(\theta)$，那么:

$$\begin{aligned} n_t &= n_{t-1} + (g_t)^2 \\ \theta_t &= \theta_{t-1} - \frac{\eta}{\sqrt{n_t + \epsilon}} \cdot g_t = \theta_{t-1} - \frac{\eta}{\sqrt{\sum_{i=1}^{t} g_r^2 + \epsilon}} \cdot g_t \end{aligned} \tag{1.37}$$

其中 η 是一个全局学习率，ϵ 是一个常数来保证分母不为 0。

- 优点:
  - 前期 n_t 较小的时候，梯度的系数较大，能够放大梯度;
  - 后期 n_t 较大的时候，梯度的系数较小，能够缩小梯度;
- 缺点:
  - 中后期梯度系数会逐渐趋于 0，可以使得训练提前结束，这可能会导致无法取得最优值。

1.2.7 Adadelta

针对 Adagrad 会提前结束的问题，Adadelta 只累计固定大小的项，并且也不直接存储这些项，仅仅计算对应的近似平均值。

$$\begin{aligned} g_t &= \nabla_\theta J(\theta) \\ n_t &= v \cdot n_{t-1} + (1 - v) \cdot g_t^2 \\ \theta_t &= \theta_{t-1} - \frac{\eta}{\sqrt{n_t + \epsilon}} \cdot g_t \\ E[g^2]_t &= \rho \cdot E[g^2]_{t-1} + (1 - \rho) \cdot g_t^2 \end{aligned} \tag{1.38}$$

综上，系数项可以写为:

$$-\frac{\sum_{i=1}^{t-1} \Delta\theta_i}{\sqrt{E[g^2]_t + \epsilon}} \tag{1.39}$$

其中 E 代表期望，v, ρ 是常数。

特点:

- 训练初期，加速效果很好;
- 训练后期，反复在局部最优解附近震荡，但是不会停止。

1.2.8 RMSprop

RMSprop 是 Adadelta 的一个特例，即 $\rho = 0.5$。

1.2.9 Adam

Adam 是一个非常好用的优化器，基本可以当成很多算法的初始优化器了。Adam 本质是带有动量项的 Adadelta。

$$\begin{aligned} m_t &= \mu \cdot m_{t-1} + (1 - \mu_t) \cdot g_t \\ n_t &= v \cdot n_{t_1} + (1 - v) \cdot g_t^2 \\ \hat{m}_t &= \frac{m_t}{1 - \mu^t} \\ \hat{n}_t &= \frac{n_t}{1 - v^t} \\ \Delta\theta &= -\frac{\hat{m}_t}{\sqrt{\hat{n}_t} + \epsilon} \cdot \eta \end{aligned} \tag{1.40}$$

其中的参数初始设置：$\mu = 0.9, v = 0.999.\epsilon = 10^{-8}$。该算法特点:

- Adma 参数比较平稳。
- 善于处理非平稳目标。
- 学习速率是自动计算出来的，不需要自己设置。
- 适用于大多数非凸优化问题。

1.2.10 Adamax

Adamax 是 Adam 的一种变化，使得 Adma 的学习率边界范围更简单。

$$\begin{aligned} n_t &= \max(v \cdot n_{t-1}, |g_t|) \\ \Delta\theta_t &= -\frac{\hat{m}_t}{n_t + \epsilon} \cdot \eta \end{aligned} \tag{1.41}$$

1.2.11 Nadam

Nadam 类似于带有 Nexterov 动量项的 Adam 算法.

$$
\begin{aligned}
&\hat{g}_t = \frac{g_t}{1-\prod_{i=1}^{t}\mu_i} \\
&m_t = \mu \cdot m_{t-1} + (1-\mu_t)\cdot g_t \\
&\hat{m}_t = \frac{m_t}{1-\prod_{i=1}^{t+1}\mu_i} \\
&n_t = v \cdot n_{t-1} + (1-v)\cdot g_t^2 \\
&\hat{n}_t = \frac{n_t}{1-v^t}\hat{m}_t = (1-\mu_t)\cdot \hat{g}_t + \mu_{t+1}\cdot \hat{m}_t \\
&\Delta\theta_t = -\frac{\hat{m}_t}{\sqrt{\hat{n}_t}+\epsilon}\cdot \eta
\end{aligned} \tag{1.42}
$$

一般而言，在使用带动量的 RMSprop 或 Adam 问题，使用 Nadam 可以取得更好的效果。

1.3 拉格朗日乘子法

拉格朗日乘子法就是求函数 $f(x_1, x_2, \cdots)$ 在约束条件 $g(x_1, x_2, \cdots) = 0$ 下的极值的方法。步骤如下:

1. 设需要求极值的目标函数为 $f(x_1, x_2, \cdots)$, 限制条件为 $g(x_1, x_2, \cdots) = 0$;

2. 引入 n 个拉格朗日参数 $\lambda_i, i = 1, 2, \cdots, n$;

3. 建立；拉格朗日函数

$$
L(x_1, x_2, \cdots, x_n, \lambda_1, \lambda_2, \cdots, \lambda_n) = f(x_1, x_2, \cdots, x_n) + \sum_{i=1}^{n} \lambda_i g_i(x_1, x_2, \cdots, x_n) \tag{1.43}
$$

4. 对每一个参数求偏导，并令其为 0，从而得到极值点:

$$\begin{aligned}&\frac{\partial L}{\partial x_i}=0\\&\frac{\partial L}{\partial \lambda_i}=0\\&i\in\{1,2,\cdots,n\}\end{aligned}\tag{1.44}$$

1.4 最小二乘法

最小二乘估计法，又称最小平方法，是一种数学优化技术。它通过最小化误差的平方和寻找数据的最佳函数匹配。利用最小二乘估计法可以简便地求得未知的数据，并使得这些求得的数据与实际数据之间误差的平方和为最小。常用于线性回归模型。

假设要回归的线性方程为 $y=\beta_0+\beta_1 x$，它的损失函数也是十分简单，就是假设 y_i 是真实值，$\hat{y}_i=\beta_0+\beta_1 x$ 是估计值。那么损失函数可以写为:

$$L=\sum_i^n(y_i-\hat{y}_i)^2=\sum_i^n(y_i-\beta_0-\beta_1 x)^2\tag{1.45}$$

要令 L 取最小值，分别对所有系数 β_0,β_1 进行求导:

$$\begin{aligned}\frac{\partial L}{\partial \beta_0}&=2\sum_i^n(y_i-\beta_0-\beta_1 x_i)(-1)=0\\\frac{\partial L}{\partial \beta_1}&=2\sum_i^n(y_i-\beta_0-\beta_1 x_i)(-x_i)=0\end{aligned}\tag{1.46}$$

经过整理我们就可以求出 β_0,β_1:

$$\begin{aligned}\beta_0&=\frac{\sum_i^n x_i^2\sum_i^n y_i-\sum_i^n x_i\sum_i^n x_i y_i}{n\sum_i^n x_i^2-(\sum_i^n x_i)^2}\\\beta_1&=\frac{n\sum_i^n x_i y_i-\sum_i^n x_i\sum_i^n y_i}{n\sum_i^n x_i^2-(\sum_i^n x_i)^2}\end{aligned}\tag{1.47}$$

有了 β_0,β_1 两个参数，就可以回归线性方程了。

Chapter 2

BP 算法

假设样本输入为 x_i，对应的标签为 y_i。

假设第 h 个隐藏层输入为：$\alpha_h = \sum_i v_{ih} x_i$，输出为：$b_h$。

假设第 j 个输出层输入为：$\beta_j = \sum_j w_{hj} b_h$，输出为：$\hat{y}_j = f(\beta_j - \theta_j)$，其中 $f(x)$ 为激活函数，这里取 sigmoid，θ 为阈值。

综上，我们可以建立从输入到输出的的联系:

$$\begin{cases} \alpha_h = \sum_i v_{ih} x_i \\ b_h = f_1(\alpha_h - \gamma_h) \\ \beta_j = \sum_j w_{hj} b_h \\ \hat{y}_j = f_2(\beta_j - \theta_j) \end{cases} \tag{2.1}$$

对于某个样本 k，计算均方误差，为了方便，增加一个系数 $\frac{1}{2}$:

$$E_k = \frac{1}{2} \sum_j (\hat{y}_j^k - y_j^k)^2 \tag{2.2}$$

我们进行随机梯度下降，满足:

$$g(x + \Delta x) < g(x) \tag{2.3}$$

根据泰勒展开式展开到一阶:

$$\begin{aligned} f(x + \Delta x) &\approx f(x) + \Delta x f'(x) + \frac{1}{2!} \Delta x f^{(2)}(x) + \cdots + \frac{1}{n!} \Delta x f^{(n)}(x) \\ &\approx f(x) + \Delta x f'(x) \end{aligned} \tag{2.4}$$

那么公式 2.3可以转化为：

$$
\begin{aligned}
g(x+\Delta x) < g(x) &\Rightarrow g(x) + \Delta x g'(x) < g(x) \\
&\Rightarrow \Delta x g'(x) < 0
\end{aligned} \tag{2.5}
$$

我们只需让 $\Delta x g'(x)$ 趋近于一个接近零的极小负数即可，引入学习速率 η，由梯度下降的公式及偏导数的定义：

$$
v \leftarrow v + \Delta v \tag{2.6}
$$

$$
\Delta x = -\eta \frac{\partial L}{\partial x} \tag{2.7}
$$

其中，L 就是我们定义的损失函数，x 就是需要优化的参数了。

2.1 Δw 更新公式推导

对于某个样本 k，由公式 2.7我们可以推出 Δw 的优化公式:

$$
\Delta w_{hj} = -\eta \frac{\partial E_k}{\partial w_{hj}} \tag{2.8}
$$

由公式 2.1我们可以知道:w 先影响 β 再影响 $\hat{y}$ 最后影响 E，这就构成了一个误差逆向传播链，那么由链式法则可以知道：

$$
\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}} \tag{2.9}
$$

因为 $\beta_j = \sum_j w_{hj} b_h$，那么：

$$
\begin{aligned}
\frac{\partial \beta_j}{\partial w_{hj}} &= \frac{\partial \left(\sum_j w_{hj} b_h \right)}{\partial w_{hj}} \\
&= \frac{\partial \left(w_{h1} b_h + w_{h2} b_h + \cdots + w_{hj} b_h \right)}{\partial w_{hj}} \\
&= b_h
\end{aligned} \tag{2.10}
$$

接下来推导 $\frac{\partial E_k}{\partial \hat{y}_j^k}$:

$$
\begin{aligned}
\frac{\partial E_k}{\partial \hat{y}_j^k} &= \frac{\left(\frac{1}{2}\sum_j(\hat{y}_j^k - y_j^k)^2\right)}{\partial \hat{y}_j^k} \\
&= \frac{\partial\left(\frac{1}{2}\left((\hat{y}_1^k - y_1^k)^2 + (\hat{y}_2^k - y_2^k)^2 + \cdots + (\hat{y}_j^k - y_j^k)^2\right)\right)}{\partial \hat{y}_j^k} \\
&= \frac{\partial\left(\frac{1}{2}(\hat{y}_j^k - y_j^k)^2\right)}{\partial \hat{y}_j^k} \\
&= \hat{y}_j^k - y_j^k
\end{aligned} \tag{2.11}
$$

往下继续推导 $\frac{\partial \hat{y}_j^k}{\partial \beta_j}$，这里需要借用一个 sigmoid 函数的特殊性质:

$$
f'(x) = f(x)(1 - f(x)) \tag{2.12}
$$

借用 sigmoid 函数的性质，我们进行如下推导:

$$
\begin{aligned}
\frac{\partial \hat{y}_j^k}{\partial \beta_j} &= \frac{\partial f(\beta_j^k - \theta_j)}{\partial \beta_j} \\
&= f'(\beta_j^k - \theta_j) \\
&= f(\beta_j^k - \theta_j)(1 - f(\beta_j^k - \theta_j)) \\
&= \hat{y}_j^k(1 - \hat{y}_j^k)
\end{aligned} \tag{2.13}
$$

我们简化偏导数的表达公式:

$$
\begin{aligned}
g_j &= \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \\
&= (\hat{y}_j^k - y_j^k)\hat{y}_j^k(1 - \hat{y}_j^k)
\end{aligned} \tag{2.14}
$$

至此，综合公式 2.9、2.11、2.13、2.10、2.14我们可以求出 Δw_{hj} 的更新公式了。

$$
\begin{aligned}
\Delta w_{hj} &= -\eta g_j b_h \\
&= -\eta(\hat{y}_j^k - y_j^k)\hat{y}_j^k(1 - \hat{y}_j^k)b_h
\end{aligned} \tag{2.15}
$$

这说明 w 的更新完全可以由输出 $\hat{y}$，label y 和上一层的输出 b_h 进行更新了。

2.2 $\Delta\theta$ 更新公式推导

同理，我们可以由公式 2.7来确定 θ_j 的更新公式:

$$
\begin{aligned}
\Delta\theta_j &= -\eta\frac{\partial E_k}{\partial \theta_j} \\
&= -\eta\frac{\partial\left(\frac{1}{2}\sum_j(\hat{y}_j^k-y_j^k)^2\right)}{\partial\theta_j} \\
&= -\eta\frac{\partial\left(\frac{1}{2}\sum_j(f(\beta_j^k-\theta_j)-y_j^k)^2\right)}{\partial\theta_j} \\
&= -\eta\left((f(\beta_j^k-\theta_j)-y_j^k)\cdot f'(\beta_j^k-\theta_j)\right) \\
&= -\eta\left((\hat{y}_j^k-y_j^k)\cdot f(\beta_j^k-\theta_j)\cdot(1-f(\beta_j^k-\theta_j))\right) \\
&= -\eta\left((\hat{y}_j^k-y_j^k)\cdot \hat{y}_j^k\cdot(1-\hat{y}_j^k)\right) \\
&= -\eta g_j
\end{aligned}
\tag{2.16}
$$

2.3 Δv 更新公式推导

同公式 2.8一致，结合链式法则，我们可以写出 Δv 的更新公式:

$$
\begin{aligned}
\Delta v_{ih} &= -\eta\cdot\frac{\partial E_k}{\partial v_{ih}} \\
&= -\eta\cdot\frac{\partial E_k}{\partial b_k}\cdot\frac{\partial b_k}{\partial \alpha_h}\cdot\frac{\partial \alpha_h}{\partial v_{ih}} \\
&= -\eta\cdot\frac{\partial E_k}{\partial v_{ih}}\cdot\frac{\partial f(\alpha_h-\gamma_h)}{\partial \alpha_h}\cdot\frac{\partial\sum_i v_{ih}x_i}{\partial v_{ih}} \\
&= -\eta\cdot\frac{\partial E_k}{\partial v_{ih}}\cdot f'(\alpha_h-\gamma_h)\cdot x_i \\
&\approx -\eta\cdot\frac{\partial E_k}{\partial \hat{y}_j^k}\frac{\partial \hat{y}_j^k}{\partial \beta_j}\frac{\partial \beta_j}{\partial b_h}\cdot f(\alpha_h-\gamma_h)(1-f(\alpha_h-\gamma_h))\cdot x_i \\
&= -\eta\cdot g_j\frac{\partial\sum_j w_{hj}b_h}{\partial b_h}\cdot b_h(1-b_h)\cdot x_i \\
&= -\eta\cdot g_j\sum_j w_{hj}\cdot b_h(1-b_h)\cdot x_i = -\eta e_h x_i
\end{aligned}
\tag{2.17}
$$

其中 $e_h = b_h(1-b_h)\sum_j w_{hj}g_j$.

2.4 $\Delta\gamma_h$ 更新公式推导

同理，我们可以结合链式法则，由公式 2.8来确定 γ_h 的更新公式:

$$\begin{aligned}
\Delta\gamma_h &= -\eta \cdot \frac{\partial E_k}{\partial \gamma_h} \\
&= -\eta \cdot \frac{\partial E_k}{\partial b_h} \frac{\partial b_h}{\partial \alpha_h} \cdot \frac{\partial \alpha_h}{\partial b_h} \frac{\partial b_h}{\partial \gamma_h} \\
&= -\eta \cdot e_h \cdot \frac{1}{\frac{\partial b_h}{\partial \alpha_h}} \cdot -1 \\
&= \eta e_h
\end{aligned} \tag{2.18}$$

Chapter 3

下一个算法