第十四章　在线学习

14.1　简介

14.1 简 介

- 随着科技的快速发展, 我们迎来了大数据时代. 通过对海量数据的处理与分析, 可以发现巨大的社会价值, 服务于生产生活. 日益增长的海量数据往往呈现“流”特征, 亦称“数据流”, 此类大数据对存储承载力与计算机的计算性能等方面提出了更高的要求, 传统的基于离线优化的机器学习算法面对极大的挑战. 因此, 在线学习的出现解决了这一问题. 在线学习算法可被理解为: 在重复决策的过程中, 算法基于之前的经验以及当前的数据做出预测, 以实现实时决策并不断地对模型进行改进, 来提高预测的精度. 在线学习算法已经被广泛应用于数据流的分析中.

- 在线学习是基于机器学习的方法, 对海量数据流进行训练, 基于之前的经验 (即保存下来的估计), 来不断更新最佳的预测, 而非传统地以批量处理的方式运行. 在数据流框架下, 传统的批量学习方法具有时间和空间成本高、效率低、扩展性差的特点, 在线学习方法仅仅基于当前数据和历史估计结果进行更新, 算法的效率和可扩展性大大提高.

14.1 简 介

- 接下来介绍一个在线学习的应用实例, 比如在云计算中使用并行处理时,需要实现一种分配资源和调度任务执行顺序的机制. 由于资源和任务在过程中不断更新, 后台研究人员无法一次性得到所有的信息进行离线训练. 因此可以利用在线学习算法根据实际任务执行的更新信息动态调整资源分配, 提高云的利用率.
- 在线学习适合大数据时代的分布式数据流的处理流程, 目前, 在线学习已经成为机器学习与人工智能领域的重要研究课题, 其研究方向可能集中在对维度更敏感的高效在线图优化算法、理论方面上下限分析及最优解的寻找等方面.
- 下面以均值模型、线性模型、岭回归以及高维模型讲解在线学习算法, 主要介绍 3 种方法：累积统计量、在线梯度下降以及正则化的在线梯度下降.

14.2　累积统计量在线学习

14.2.1 均值模型

- 平均值反映数据的位置, 描述数据中心. 对于一组 p 维的数据 $\boldsymbol{X}_1, \cdots, \boldsymbol{X}_n$, 我们可构建均值模型为

$$\boldsymbol{X}_i = \boldsymbol{\mu} + \varepsilon_i, i = 1, 2, \cdots, n.$$

  - 模型通过误差 ε_i 来描述以 $\boldsymbol{\mu}$ 为中心的样本值. 对于参数 $\boldsymbol{\mu}$, 用样本均值进行参数估计, 即

$$\hat{\boldsymbol{\mu}} = \bar{\boldsymbol{X}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{X}_i.$$

  - 然而, 对于实时获取的海量数据流来讲, 我们会按照时间观测到数据 $\{\boldsymbol{X}_{ti} : i = 1, 2 \cdots, n_t, t = 1, 2, \cdots\}$, $\boldsymbol{X}_{ti}$ 代表第 t 批次数据流第 i 个样本, n_t 是第 t批次数据的样本量. 数据的更新意味着需要对模型不断地进行重复计算. 受限于数据存储承载力与计算机的计算性能, 传统的离线算法不适用于数据流的均值估计.

- 因此, 我们可以构建在线学习均值模型, 即在保存的前面算法结果的基础上仅对新数据进行处理, 从而不断更新总体期望的估计结果. 截止到第 t批次, 均值估计过程如下:

$$\hat{\boldsymbol{\mu}}_t = \bar{\boldsymbol{X}}_t = \frac{1}{N_t}\sum_{j=1}^{t}\sum_{i=1}^{n_j} \boldsymbol{X}_{ji} = \frac{1}{N_t}\left\{\sum_{j=1}^{t-1}\sum_{i=1}^{n_j} \boldsymbol{X}_{ji} + \sum_{i=1}^{n_t} \boldsymbol{X}_{ji}\right\},$$

14.2.1 均值模型

- 其中 $N_t = n_1 + \cdots + n_t$. 通过在线学习均值模型的构建, 在已知 $\sum_{j=1}^{t-1}\sum_{i=1}^{n_j} \boldsymbol{X}_{ji}$ (已保存, 仅为 p 维向量) 的基础上, 只需计算 $\sum_{i=1}^{n_t} \boldsymbol{X}_{ti}$ 以及更新 N_t 的值, 便可得到新数据下参数 $\boldsymbol{\mu}$ 的估计. 这样一来, 可从多输入源分布式地输入数据,算法的效率和可扩展性都大大提高.

- 在线更新样本均值算法流程可以表示如下:

- 当然, 我们可以选择终止条件, 在某一时刻终止上述在线算法, 得到收敛值, 并将其作为最终的均值估计值.

算法 1：均值模型累积统计量在线学习算法

输入： 初始化数据 $\boldsymbol{X}_{11}, \cdots, \boldsymbol{X}_{1n_1}$

输出： 实时的样本均值 $\bar{\boldsymbol{X}}_t$

for $t = 2$ to ∞ do

计算新的样本总数 $N_t = N_{t-1} + n_t$

计算并输出新的样本总和 $\sum_{j=1}^{t}\sum_{i=1}^{n_j} \boldsymbol{X}_{ji} = N_{t-1}\overline{\boldsymbol{X}}_{t-1} + \sum_{i=1}^{n_t} \boldsymbol{X}_{ti}$

计算新的样本均值 $\bar{\boldsymbol{X}}_t = \frac{1}{N_t}\sum_{j=1}^{t}\sum_{i=1}^{n_j} \boldsymbol{X}_{ji}$

14.2.2 线性模型

- 线性模型是一种描述因变量和自变量之间线性关系的模型, 假设因变量Y和p个自变量$X_1,\cdots,X_p$之间存在简单的线性关系, 可构建如下形式:

$$Y = h_\beta(\boldsymbol{X}) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \varepsilon.$$

- 对于实时获取的海量回归数据流来讲, 我们假设观测到数据$\left\{\left(\boldsymbol{X}_{ti}^{\mathrm{T}}, Y_{ii}\right)^{\mathrm{T}} : i=1,2,\cdots,n_t, t=1,2,\cdots\right\}$, 令

$$\boldsymbol{Y}_t = X_t\boldsymbol{\beta} + \boldsymbol{\varepsilon}_t,$$

$$\boldsymbol{Y}_t = \begin{pmatrix} Y_{t1} \\ Y_{t2} \\ \vdots \\ Y_{tn_t} \end{pmatrix}, \boldsymbol{X}_t = \begin{pmatrix} 1 & X_{t1,1} & \cdots & X_{t1,p} \\ 1 & X_{t2,1} & \cdots & X_{t2,p} \\ \vdots & \vdots & & \vdots \\ 1 & X_{tn_t,1} & \cdots & X_{tn_t,p} \end{pmatrix}, \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}, \boldsymbol{\varepsilon}_t = \begin{pmatrix} \varepsilon_{t1} \\ \varepsilon_{t2} \\ \vdots \\ \varepsilon_{tn_t} \end{pmatrix}.$$

14.2.2 线性模型

- 截止到第 t 批次数据的到来, 回归模型的参数估计过程如下:

$$\hat{\boldsymbol{\beta}}_t = \left(\sum_{j=1}^{t} \mathbf{X}_j^{\mathrm{T}} \mathbf{X}_j \right)^{-1} \left(\sum_{j=1}^{t} \mathbf{X}_j^{\mathrm{T}} \mathbf{Y}_j \right) = \left(\sum_{j=1}^{t-1} \mathbf{X}_j^{\mathrm{T}} \mathbf{X}_j + \mathbf{X}_t^{\mathrm{T}} \mathbf{X}_t \right)^{-1} \left(\sum_{j=1}^{t-1} \mathbf{X}_j^{\mathrm{T}} \mathbf{Y}_j + \mathbf{X}_t^{\mathrm{T}} \mathbf{Y}_t \right).$$

  - 通过在线学习线性模型的构建, 在已知 $\sum_{j=1}^{t-1} \mathbf{X}_j^{\mathrm{T}} \mathbf{X}_j$ (已保存, 仅为 $(p+1)\times(p+1)$矩阵) 和 $\sum_{j=1}^{t-1} \mathbf{X}_j^{\mathrm{T}} \mathbf{Y}_j$ (已保存, 仅为 p + 1 维向量) 的基础上, 只需计算 $\mathbf{X}_t^{\mathrm{T}} \mathbf{X}_t$ 以及 $\mathbf{X}_t^{\mathrm{T}} \mathbf{Y}_t$ 的值, 便可得到新数据下参数 $\boldsymbol{\beta}$ 的估计. 这样一来, 可从多输入源分布式地输入数据, 算法的效率和可扩展性都大大提高.

- 在线更新回归参数算法流程可以表示如下:

算法 2: 线性模型累积统计量在线学习算法

输入: 初始化数据 $\mathbf{X}_1, \boldsymbol{Y}_1$

输出: 实时的回归参数估计 $\widehat{\boldsymbol{\beta}}_t$

for $t = 2$ to ∞ do

更新 $p \times p$ 矩阵 $\sum_{j=1}^{t} \mathbf{X}_j^{\mathrm{T}} \mathbf{X}_j = \sum_{j=1}^{t-1} \mathbf{X}_j^{\mathrm{T}} \mathbf{X}_j + \mathbf{X}_t^{\mathrm{T}} \mathbf{X}_t$

更新 p 维向量 $\sum_{j=1}^{t} \mathbf{X}_j^{\mathrm{T}} \boldsymbol{Y}_j = \sum_{j=1}^{t-1} \mathbf{X}_j^{\mathrm{T}} \boldsymbol{Y}_j + \mathbf{X}_t^{\mathrm{T}} \boldsymbol{Y}_t$

估计回归参数 $\widehat{\boldsymbol{\beta}}_t = \left(\sum_{j=1}^{t} \mathbf{X}_j^{\mathrm{T}} \mathbf{X}_j \right)^{-1} \left(\sum_{j=1}^{t} \mathbf{X}_j^{\mathrm{T}} \boldsymbol{Y}_j \right)$

14.3　在线梯度下降

14.3 在线梯度下降

- 本章节介绍在线梯度下降算法 (online gradient descent, OGD), 并研究算法收敛性以及在线性模型和岭回归模型中的应用.

14.3.1 OGD 算法一般形式

- 假设第 t 次批量数据到来, 在传统离线优化算法中, 我们采用一般形式的损失函数去估计感兴趣参数 $\boldsymbol{\beta}$:

$$L_t(\boldsymbol{\beta}) = \sum_{j=1}^{t} \ell_j(\boldsymbol{\beta}),$$

  - 其中 $\ell_j(\boldsymbol{\beta}) = \ell(h_{\boldsymbol{\beta}}(\mathbf{X}_j), \boldsymbol{Y}_j)$ 代表第 j 批量数据下的损失函数, 例如二次损失或负对数似然等, 且满足利普希茨连续性. 然而, 若采用上述形式损失函数更新参数, 则需要使用数据流的所有样本进行估计, 模型训练速度会随着时间的增加变得越来越慢.

- 接下来, 我们采用在线梯度下降算法对其进行无约束优化, 估计模型参数. 梯度下降法是一种最优化算法, 从几何意义上来讲, 梯度是函数变化增加最快的方向, 沿着梯度向量相反的方向, 梯度减少更快, 更加容易找到函数的最小值. 根据损失函数的选取、更新参数时使用样本数量的不同, 常见的梯度下降法分为批量梯度下降法 BGD、随机梯度下降法 SGD、小批量梯度下降法 MBGD 等.

14.3.1 OGD 算法一般形式

- 对于日益增加的样本量, 若使用传统离线的优化方法, 虽然模型拟合精度得到提高, 但是计算复杂度不断提高, 训练速度会越来越来慢, 并且历史数据也被重复使用. 因此, Zinkevich[156] 提出了在线梯度下降的在线学习算法.在线梯度下降算法思想是在第 $t+1$ 阶段, 仅使用第 t 阶段的数据对应的损失函数 $l_t(\boldsymbol{\beta})$ 进行梯度下降. 计算出直接下降结果后, 若可行域 $\mathcal{K}$ 受限, 还需将梯度下降后的结果投影回可行空间.

- 在明确回归函数 $h_{\boldsymbol{\beta}}(\boldsymbol{X})$ 以及损失函数$l_t(\boldsymbol{\beta})$ 的具体形式后, OGD 算法流程的一般框架如下所示:

算法 3–1：OGD 算法

输入： 初始化参数 $\boldsymbol{\beta}_0$, 终止距离 δ, 学习速率 α, 可行域 $\mathcal{K}$

输出： 参数 $\boldsymbol{\beta}$ 的收敛值

for $t = 0$ to ∞ do

计算梯度下降点 $\boldsymbol{\theta}_{t+1} = \boldsymbol{\beta}_t - \alpha l_t'(\boldsymbol{\beta}_t)$

计算投影点 $\boldsymbol{\beta}_{t+1} = \underset{\boldsymbol{\beta}\in\mathcal{K}}{\operatorname{argmin}} \|\boldsymbol{\theta}_{t+1} - \boldsymbol{\beta}\|^2$

计算当前损失 $l_t(\boldsymbol{\beta}_{t+1})$ 及其梯度 $l_t'(\boldsymbol{\beta}_{t+1})$

如果前后步距离 $\|\boldsymbol{\beta}_{t+1} - \boldsymbol{\beta}_t\| < \delta$

end for

14.3.2 GD 算法收敛性分析

- 定义遗憾函数 (regret function)

$$R_T = \sum_{t=1}^{T}\left[\ell_t(\boldsymbol{\beta}_t) - \ell_t(\boldsymbol{\beta}^*)\right], \tag{14.3.1}$$

  - 其中 $\boldsymbol{\beta}^* = \arg\min_{\beta}\sum_{t=1}^{T}\ell_t(\boldsymbol{\beta})$,$\boldsymbol{\beta}_t$ 表示第 t 轮迭代值. 随着时间 T 的增加, 遗憾函数的值接近于一个常量, 则算法的收敛值 $\boldsymbol{\beta}_T$ 与最优的 $\boldsymbol{\beta}^*$ 是一致的, 即可证明此在线学习方法是有效的.

- 接下来, 我们分析 OGD 算法的收敛性. 根据投影点的对应关系, 可以得到

$$\left\|\boldsymbol{\beta}_{t+1} - \boldsymbol{\beta}^*\right\|^2 \le \left\|\boldsymbol{\theta}_{t+1} - \boldsymbol{\beta}^*\right\|^2 = \left\|\boldsymbol{\beta}_t - \boldsymbol{\beta}^*\right\|^2 - 2\alpha\left\langle \ell_t'(\boldsymbol{\beta}_t), (\boldsymbol{\beta}_t - \boldsymbol{\beta}^*)\right\rangle + \alpha^2\left\|\ell_t'(\boldsymbol{\beta}_t)\right\|^2,$$

  - 由此式变形后, 可以得到

$$\left\langle \ell_t'(\boldsymbol{\beta}_t), (\boldsymbol{\beta}_t - \boldsymbol{\beta}^*)\right\rangle \le \frac{1}{2\alpha}\left(\left\|\boldsymbol{\beta}_t - \boldsymbol{\beta}^*\right\|^2 - \left\|\boldsymbol{\beta}_{t+1} - \boldsymbol{\beta}^*\right\|^2\right) + \frac{\alpha}{2}\left\|\ell_t'(\boldsymbol{\beta}_t)\right\|^2.$$

14.3.2 GD 算法收敛性分析

- 另一方面, 根据损失函数的凸性可以得到

$$R_T = \sum_{t=1}^{T}\left[\ell_t\left(\boldsymbol{\beta}_t\right)-\ell_t\left(\boldsymbol{\beta}^*\right)\right] \leq \sum_{t=1}^{T}\left\langle \ell_t'\left(\boldsymbol{\beta}_t\right),\left(\boldsymbol{\beta}_t-\boldsymbol{\beta}^*\right)\right\rangle,$$

- 合并上述两式可得

$$R_T \leq \frac{1}{2\alpha}\left\|\boldsymbol{\beta}_1-\boldsymbol{\beta}^*\right\|^2+\frac{\alpha}{2}\sum_{t=1}^{T}\left\|\ell_t'\left(\boldsymbol{\beta}_t\right)\right\|^2.$$

- 再由损失函数的利普希茨连续性 (假设常数为 L) 可知, $\forall t,\left\|\ell_t'\left(\boldsymbol{\beta}_t\right)\right\| \leq L$ 因此得到 R_T 上界的最终形式如下:

$$R_T \leq \frac{1}{2\alpha}\left\|\boldsymbol{\beta}_1-\boldsymbol{\beta}^*\right\|^2+\frac{\alpha L^2 T}{2},$$

- 当且仅当 $\frac{1}{2\alpha}\left\|\boldsymbol{\beta}_1-\boldsymbol{\beta}^*\right\|^2=\frac{\alpha L^2 T}{2}$, 即 $\alpha=\frac{\left\|\boldsymbol{\beta}_1-\boldsymbol{\beta}^*\right\|^2}{L\sqrt{T}}$ 时, 式右边取最小值, 即 $R_T \leq \left\|\boldsymbol{\beta}_1-\boldsymbol{\beta}^*\right\| L\sqrt{T}$.因此, 我们推出 OGD 算法的性能最差情况是 $\mathcal{O}\sqrt{T}$.

14.3.3 线性模型的 OGD 算法

- 注意到上述 OGD 算法是对于一般模型而言的, 实现了在线梯度下降的算法功能, 具有一定的包容性与可扩展性, 相比于传统离线优化算法大大减少了计算的复杂度, 因为损失函数 $l_t(\boldsymbol{\beta})$ 仅使用第 t 阶段的数据. 借助OGD的算法思想, 假设第 t 阶段的二次损失函数为

$$l_t(\boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{Y}_t - \mathbf{X}_t\boldsymbol{\beta}\|_2^2,$$

  - 对应的梯度是

$$l_t'(\boldsymbol{\beta}) = -\mathbf{X}_t^{\mathrm{T}}(\boldsymbol{Y}_t - \mathbf{X}_t\boldsymbol{\beta}).$$

- 在基于二次损失构造线性模型在线学习算法时, 我们不妨在第 t 阶段, 选取第 $t-1$ 阶段求得的 $\boldsymbol{\beta}_t-1$ 计算梯度值, 这样不仅能通过新数据不断迭代更新参数估计, 还减少了第 t 阶段模型拟合带来的计算复杂度. 因此第 t 阶段的梯度的计算公式如下:

$$l_t'(\boldsymbol{\beta}_{t-1}) = -\boldsymbol{X}_t^{\mathrm{T}}(\boldsymbol{Y}_t - \boldsymbol{X}_t\boldsymbol{\beta}_{t-1}).$$

14.3.3 线性模型的 OGD 算法

- 另外, 考虑到线性模型的可行域为 Rp+1, 梯度下降后无需将中间迭代结果投影回可行域. 因此基于二次损失的线性模型在线学习算法流程可以表示如下:

算法 3–2：OGD 算法—基于二次损失

输入： 初始化参数 $\boldsymbol{\beta}_0$, 终止距离 δ, 学习速率 α, 参数 λ

输出： 参数 $\boldsymbol{\beta}$ 的收敛值

for $t = 1$ to ∞ do

计算第 t 阶段梯度 $l_t'(\boldsymbol{\beta}_{t-1}) = -\mathbf{X}_t^{\mathrm{T}}(\boldsymbol{Y}_t - \mathbf{X}_t\boldsymbol{\beta}_{t-1})$

更新学习速率 $\alpha = \lambda t^{-\frac{1}{2}}$

更新参数 $\boldsymbol{\beta}_t = \boldsymbol{\beta}_{t-1} - \alpha l_t'(\boldsymbol{\beta}_{t-1})$

如果前后步距离 $\|\alpha l_t'(\boldsymbol{\beta}_{t-1})\| < \delta$

end for

14.3.4 岭回归模型的 OGD 算法

- 岭回归方法是一种解决回归数据共线性问题的监督学习方法. 其使用的目标函数是在二次损失的基础上, 增加 L_2 正则项, 详细介绍参考回归模型章节. 下面介绍基于二次损失的岭回归在线学习算法. 假设观测到第 t 次批量数据的到来, 在传统岭回归模型离线优化算法中, 采用如下二次损失函数去估计感兴趣参数 $\boldsymbol{\beta}$:

$$l_t(\boldsymbol{\beta}) = \frac{1}{2n_t}\left\|\boldsymbol{Y}_t - \mathbf{X}_t\,\boldsymbol{\beta}\right\|_2^2 + \frac{\lambda}{2}\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{\beta},$$

  - 对应的梯度是

$$l_t'(\boldsymbol{\beta}) = -\frac{1}{n_t}\mathbf{X}_t^{\mathrm{T}}\left(\boldsymbol{Y}_t - \mathbf{X}_t\,\boldsymbol{\beta}\right) + \lambda\boldsymbol{\beta}.$$

  - 考虑到模型的训练速度以及计算的复杂度, 不妨借助上一节线性模型在线学习的思想, 第 t 阶段对应的梯度函数, 选取第 $t-1$ 阶段求得的 $\boldsymbol{\beta}_t{-}1$ 进行下一步的迭代, 这样不仅能通过新数据不断迭代更新参数估计, 还不用基于当前数据拟合模型, 即此时的梯度是

$$l_t'(\boldsymbol{\beta}_{t-1}) = -\frac{1}{n_t}\mathbf{X}_t^{\mathrm{T}}\left(\boldsymbol{Y}_t - \mathbf{X}_t\,\boldsymbol{\beta}_{t-1}\right) + \lambda\boldsymbol{\beta}_{t-1}.$$

14.3.3 线性模型的 OGD 算法

- 以上岭回归模型的在线学习算法流程可以表示如下：

算法 4：岭回归在线学习算法—基于二次损失

输入： 初始化参数 $\boldsymbol{\beta}_0$, 终止距离 δ, 学习速率 α, 调谐参数 λ

输出： 参数 $\boldsymbol{\beta}$ 的收敛值

for $t = 1$ to ∞ do

计算第 t 阶段梯度 $l_t'(\boldsymbol{\beta}_{t-1}) = -\frac{1}{n_t}\boldsymbol{X}_t^{\mathrm{T}}(\boldsymbol{Y}_t - \boldsymbol{X}_t\boldsymbol{\beta}_{t-1}) + \lambda\boldsymbol{\beta}_{t-1}$

更新参数 $\boldsymbol{\beta}_t = \boldsymbol{\beta}_{t-1} - \alpha l_t'(\boldsymbol{\beta}_{t-1})$

如果前后步距离 $\|\alpha l_t'(\boldsymbol{\beta}_{t-1})\| < \delta$

end for

14.4　基于正则化的在线梯度下降

14.4.1 FTL 算法

- FTL(follow the leader) 算法在损失函数强凸的情形下能有效解决在线优化问题, 其算法的思想是通过最小化累积损失来更新参数, 公式如下:

$$\boldsymbol{\beta}_{t+1} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \sum_{j=1}^{n} l_j(\boldsymbol{\beta}).$$

  - 由归纳法得 FTL 算法的遗憾函数上限为

$$R_T \leq \sum_{t=1}^{T} \left(\ell_t(\boldsymbol{\beta}_t) - \ell_t(\boldsymbol{\beta}_{t+1}) \right).$$

- 可见, 在损失函数强凸的情形下, $\boldsymbol{\beta}_t$ 会收敛到 $\boldsymbol{\beta}^*$. 但是, 在在线线性优化(online linear optimization) 问题中不一定成立. 接下来, 我们考虑一维的感兴趣参数 β, 损失函数关于 β 是线性的, 假设为 $l_t(\beta) = G_t\beta$, $\beta \in [-1, 1]$. 此时 G_t 是损失函数的梯度, 其取值为

$$G_t = \begin{cases} -0.5, & t = 1, \\ 1, & t\text{为偶数}, \\ -1, & t > 1\text{且}t\text{为奇数}. \end{cases}$$

14.4.1 FTL 算法

- 由参数更新公式得

$$\beta_{t+1} = \underset{\beta}{\operatorname{argmin}} \sum_{j=1}^{t} G_j \beta.$$

- 使用 FTL 算法会出现参数震荡现象:

t	G_t	$\beta_{t+1} = \underset{\beta}{\operatorname{argmin}} \sum_{j=1}^{t} G_j \beta$
1	-0.5	$\underset{\beta}{\operatorname{argmin}}\{-0.5\beta\}=1$
2	1	$\underset{\beta}{\operatorname{argmin}}\{-0.5\beta + 1\beta\} = -1$
3	-1	$\underset{\beta}{\operatorname{argmin}}\{-0.5\beta + 1\beta - 1\beta\}=1$
4	1	$\underset{\beta}{\operatorname{argmin}}\{-0.5\beta + 1\beta - 1\beta + 1\beta\} = -1$
...	...	...

14.4.1 FTL 算法

- 经过计算, β 最终的 “收敛值” 在 1 和 -1 之间震荡, 并不收敛, FTL 算法在这个在线线性优化问题上失效. 因此, FTL 算法在损失函数强凸的情形下虽然有效, 但在一般凸函数情形下不满足遗憾函数次线性, 导致参数不收敛,可见算法不稳定.

14.4.2 FTRL 算法

- 为了解决上述问题, FTRL(follow the regularized leader) 算法在 FTL 算法的基础上增加正则化项 $P_\alpha(\boldsymbol{\beta})$ 来使算法更稳定. FTRL 算法的参数更新公式如下:

$$\boldsymbol{\beta}_{t+1} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \sum_{j=1}^{n} l_j(\boldsymbol{\beta}) + P_\alpha(\boldsymbol{\beta}) \right\}.$$

  - 容易证明 FTRL 算法的遗憾函数上限为

$$R_T \leq P_\alpha(\boldsymbol{\beta}^*) + \sum_{t=1}^{T} \left[\ell_t(\boldsymbol{\beta}_t) - \ell_t(\boldsymbol{\beta}_{t+1}) \right].$$

- 针对上述的在线线性优化问题, 不妨令 $P_\alpha(\boldsymbol{\beta}) = \frac{1}{2\alpha}\|\boldsymbol{\beta}\|_2^2$,求导得出参数更新公式为: $\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \alpha \boldsymbol{G}_t$, 注意此时我们考虑的是多维的感兴趣参数 $\boldsymbol{\beta}, \boldsymbol{G}_t$ 是梯度向量.

- FTRL 算法的遗憾函数上限为

$$R_T \leq \frac{1}{2\alpha}\|\boldsymbol{\beta}^*\|_2^2 + \alpha \sum_{t=1}^{T} \|\boldsymbol{G}_t\|_2^2,$$

14.4.2 FTRL 算法

- 若参数 $\beta \in \left\{\beta : \|\beta\|_2 \leq B\right\}$, 损失函数 l 满足 L-Lipschitz 条件, 即 $\frac{1}{T}\sum_{t=1}^{T}\|\boldsymbol{G}_t\|_2^2 \leq L^2$,则

$$R_T \leq \frac{1}{2\alpha}B^2 + \alpha T L^2,$$

- 当且仅当 $\frac{1}{2\alpha}B^2 = \alpha T L^2$, 即 $\alpha = \frac{B}{L\sqrt{2T}}$ 时, 上式右边取得最小值. 因此代入α, 则遗憾函数满足

$$R_T \leq BL\sqrt{2T}.$$

- 可以发现

$$\lim_{T\to\infty}\frac{\mathrm{d}R_T}{\mathrm{d}T} = \lim_{T\to\infty}\frac{BL}{\sqrt{2T}} = 0,$$

- 因此, $\boldsymbol{\beta}_t$ 收敛, 则 FTRL 为有效算法, 可见其相比于 FTL 算法更加稳定.

14.4.3 FTRL-Proximal 算法

- FTRL-Proximal 算法是一种用于点击率预估的在线机器学习系统的核心算法, 其可以看作 RDA(regularized dual averaging algorithm) 算法与FOBOS(forward backward splitting) 算法的结合体, 在 FTL 算法的基础上通过增加 L_1、L_2 正则项的方式来兼顾算法的稀疏性与精确度.

- 为了进一步提高算法的稀疏性, FTRL-Proximal 算法在 FTRL 算法的基础上添加 L1 正则项. FTRL-Proximal 算法使用如下损失函数更新参数:

$$\boldsymbol{\beta}_{t+1} = \operatorname{argmin}\left(\sum_{j=1}^{t} \boldsymbol{G}_j^{\mathrm{T}} \boldsymbol{\beta} + \frac{1}{2} \sum_{j=1}^{t} \sigma_j \| \boldsymbol{\beta} - \boldsymbol{\beta}_j \|_2^2 + \lambda \| \boldsymbol{\beta} \|_1 \right),$$

  - 其中 $\sigma_j = \frac{1}{\alpha_j} - \frac{1}{\alpha_{j-1}}$, α_t 是 t 时刻的学习率, λ 是 L_1 正则化调谐参数, β 为感兴趣参数. 值得注意的是, $\sum_{j=1}^{t} \boldsymbol{G}_j^{\mathrm{T}} \boldsymbol{\beta}$ 保证向正确的方向更新, 而使用历史累积梯度保证不会过早地将重要特征的参数约束为 0. 要求新产生的参数不要偏离历史参数太远, 即参数更新不要太激进, $\lambda \| \boldsymbol{\beta} \|_1$ 保证解的稀疏性.

14.4.3 FTRL-Proximal 算法

- 在参数更新过程中, 我们将特征参数的各个维度拆解成独立的标量最小化问题, 从而简化算法, 具体计算过程如下:
- 令 $\boldsymbol{Z}_t=\sum_{j=1}^{t}\boldsymbol{G}_j-\sum_{j=1}^{t}\sigma_j\boldsymbol{\beta}_j, \sigma_t=\frac{1}{\alpha_t}-\frac{1}{\alpha_{t-1}}$, 不妨构造函数 $F(\boldsymbol{\beta})$ 为

$$F(\boldsymbol{\beta})=\boldsymbol{Z}_t^{\mathrm{T}}\boldsymbol{\beta}+\frac{1}{2\alpha_t}\|\boldsymbol{\beta}\|_2^2+\lambda\|\boldsymbol{\beta}\|_1+(\text{常数}),$$

  - 即 $\boldsymbol{\beta}_{t+1}=\arg\min_{\boldsymbol{\beta}}F(\boldsymbol{\beta})$ 下面求解使得 $F(\boldsymbol{\beta})$ 最小的 $\boldsymbol{\beta}$. 由于 $\|\boldsymbol{\beta}\|_1$ 在零点处不可导, 我们考虑使用次梯度方法.

$$\partial_{\boldsymbol{\beta}}F(\boldsymbol{\beta})=\boldsymbol{Z}_t+\frac{\boldsymbol{\beta}}{\alpha_t}+\lambda\partial_{\boldsymbol{\beta}}\|\boldsymbol{\beta}\|_1,$$

  - 令其次梯度为 0, 并根据条件限制得到各维度参数更新公式如下:

$$\beta_{t+1,k}=\begin{cases}(\lambda-Z_{t,k})\alpha_t, & Z_{t,k}>\lambda,\\ 0, & |Z_{t,k}|\bullet\lambda,\\ (-\lambda-Z_{t,k})\alpha_t, & Z_{t,k}<-\lambda.\end{cases}$$

14.4.3 FTRL-Proximal 算法

- 另外, FTRL-Proximal 算法考虑了数据在不同维度上的特征分布的不均匀性, 建议每个维度采用的学习率 α_t 是不一样的并改为如下形式的 α_t,k:

$$\alpha_{t,k} = \frac{\omega}{\gamma + \sqrt{\sum_{j=1}^{t} G_{j,k}^2}},$$

  - 其中, α_t,k 为 t 时刻 k 维度的学习率, ω, γ 为超参数, $G_{t,k}$ 为 t 时刻 k 维度的损失函数的梯度.

- FTRL-Proximal 算法流程如下:

算法 5: FTRL-Proximal 算法

输入: 超参数 ω, γ, 正则化参数 λ, 总时间 T, 以及初始化参数 $\boldsymbol{Z}_0$, $\boldsymbol{\alpha}_0$, $\boldsymbol{\beta}_1$

输出: $\boldsymbol{\beta}_T$

for $t = 1$ to T do

  计算损失梯度向量 $\boldsymbol{G}_t = \boldsymbol{G}_t(\boldsymbol{\beta}_t)$

  for $k = 1$ to p do

    计算每一维度的学习速率 $\alpha_{t,k} = \dfrac{\omega}{\gamma + \sqrt{\sum_{j=1}^{t} G_{j,k}^2}}$

  end for

  计算对角矩阵 $\boldsymbol{\sigma}_t = \mathrm{diag}\left\{\dfrac{1}{\boldsymbol{\alpha}_t} - \dfrac{1}{\boldsymbol{\alpha}_{t-1}}\right\}$

  计算向量 $\boldsymbol{Z}_t = \boldsymbol{Z}_{t-1} + \boldsymbol{G}_t - \sigma_t \boldsymbol{\beta}_t$

  使用式 (14.4.3) 更新参数下一步迭代值 $\boldsymbol{\beta}_{t+1}$

end for

14.5　在线学习实践

实践代码