# 梯度下降法与FW算法在模糊系统应用中的比较

王佳文

2020 年 11 月 6 日

**摘　要**

本文描述了梯度下降法，FW算法和Away step FW算法在训练单值一型模糊逻辑系统上的应用。同时分析了梯度下降法和FW算法在此处的收敛速度，两种算法的收敛速度均为$\mathcal{O}(\frac{1}{\sqrt{t}})$,但数值实验的结果显示FW算法所需迭代步更多，所用时间更长。相同训练精度下，梯度下降法所用的时间最少，Away step FW算法能显著减小FW算法的迭代步数。在设置较高的训练精度时，FW算法和Away step FW算法所需的迭代步显著增加，而梯度下降法所需迭代步不会出现大幅变化。

## §1　采用单值模糊器, 乘积推理机, 中心平均解模糊器, 高斯隶属函数的Mamdani FLS公式的一些性质简介

选择模糊规则中隶属函数均为高斯型隶属函数的单值一型模糊逻辑系统 [1]，此时模糊系统的输出公式为：

$$f=\frac{\sum\limits_{i=1}^{M}y_i\prod\limits_{j=1}^{n}\exp(-\frac{(x_j-\mu_{ij})^2}{2\sigma_{ij}^2})}{\sum\limits_{i=1}^{M}\prod\limits_{j=1}^{n}\exp(-\frac{(x_j-\mu_{ij})^2}{2\sigma_{ij}^2})}=\frac{b}{a} \tag{1}$$

其中$M$表示模糊规则的数目，$n$表示规则前件的个数，$y_i$表示第$i$条规则输出的后件，$\mu_{ij}$ 与$\sigma_{ij}$表示第$i$条规则中第$j$个前件中隶属函数的参数，$\boldsymbol{x}=(x_1,x_2,\cdots,x_n)$ 表示模糊系统的一个输入。计算$f$对与$y_i$的一阶导数为：

$$f'_{y_i}=\frac{\prod\limits_{j=1}^{n}\exp(-\frac{(x_j-\mu_{ij})^2}{2\sigma_{ij}^2})}{a} \tag{2}$$

则$f$对与$y_i$的二阶导数为：

$$f''_{y_i}=0 \tag{3}$$

所以$f$对与$y_i$的二阶导数有界。计算$f$对与$\mu_{ij}$的一阶导数为：

$$f'_{\mu_{ij}} = \frac{y_i \prod_{j=1}^{n} \exp(-\frac{(x_j-\mu_{ij})^2}{2\sigma_{ij}^2}) \frac{x_j-\mu_{ij}}{2\sigma_{ij}^2} a - b \prod_{j=1}^{n} \exp(-\frac{(x_j-\mu_{ij})^2}{2\sigma_{ij}^2}) \frac{x_j-\mu_{ij}}{2\sigma_{ij}^2}}{a^2} \tag{4}$$

$$= (y_i - \frac{b}{a}) \frac{\prod_{j=1}^{n} \exp(-\frac{(x_j-\mu_{ij})^2}{2\sigma_{ij}^2}) \frac{x_j-\mu_{ij}}{2\sigma_{ij}^2}}{a} \tag{5}$$

$$= P_1 P_2 \tag{6}$$

记$P_1 = (y_i - \frac{b}{a})$,$P_2 = \frac{\prod_{j=1}^{n} \exp(-\frac{(x_j-\mu_{ij})^2}{2\sigma_{ij}^2}) \frac{x_j-\mu_{ij}}{2\sigma_{ij}^2}}{a}$. 计算$f$对与$\mu_{ij}$的二阶导数为：

$$f''_{\mu_{ij}} = (P_1 P_2)' = P'_1 P_2 + P_1 P'_2 \tag{7}$$

其中：

$$P'_1 = -f'_{\mu_{ij}} \tag{8}$$

$$P'_2 = \frac{\Big[\prod_{j=1}^{n} \exp(-\frac{(x_j-\mu_{ij})^2}{2\sigma_{ij}^2})(\frac{x_j-\mu_{ij}}{2\sigma_{ij}^2})^2 + \prod_{j=1}^{n} \exp(-\frac{(x_j-\mu_{ij})^2}{2\sigma_{ij}^2}) \frac{-1}{2\sigma_{ij}^2}\Big] a - \Big[\prod_{j=1}^{n} \exp(-\frac{(x_j-\mu_{ij})^2}{2\sigma_{ij}^2}) \frac{x_j-\mu_{ij}}{2\sigma_{ij}^2}\Big]^2}{a^2} \tag{9}$$

计算$f$对与$\sigma_{ij}$的一阶导数为：

$$f'_{\sigma_{ij}} = \frac{y_i \prod_{j=1}^{n} \exp(-\frac{(x_j-\mu_{ij})^2}{2\sigma_{ij}^2}) \frac{(x_j-\mu_{ij})^2}{\sigma_{ij}^3} a - b \prod_{j=1}^{n} \exp(-\frac{(x_j-\mu_{ij})^2}{2\sigma_{ij}^2}) \frac{(x_j-\mu_{ij})^2}{\sigma_{ij}^3}}{a^2} \tag{10}$$

$$= (y_i - \frac{b}{a}) \frac{\prod_{j=1}^{n} \exp(-\frac{(x_j-\mu_{ij})^2}{2\sigma_{ij}^2}) \frac{(x_j-\mu_{ij})^2}{\sigma_{ij}^3}}{a} \tag{11}$$

$$= P_3 P_4 \tag{12}$$

记$P_1 = (y_i - \frac{b}{a})$,$P_2 = \frac{\prod_{j=1}^{n} \exp(-\frac{(x_j-\mu_{ij})^2}{2\sigma_{ij}^2}) \frac{(x_j-\mu_{ij})^2}{\sigma_{ij}^3}}{a}$. 计算$f$对与$\sigma_{ij}$ 的二阶导数为：

$$f''_{\sigma_{ij}} = (P_3 P_4)' = P'_3 P_4 + P_3 P'_4 \tag{13}$$

其中：

$$P'_3 = -f'_{\sigma_{ij}} \tag{14}$$

$$P'_4 = \frac{\Big[\prod_{j=1}^{n} \exp(-\frac{(x_j-\mu_{ij})^2}{2\sigma_{ij}^2}) \frac{(x_j-\mu_{ij})^4}{\sigma_{ij}^6} + \prod_{j=1}^{n} \exp(-\frac{(x_j-\mu_{ij})^2}{2\sigma_{ij}^2}) \frac{-3(x_j-\mu_{ij})^2}{\sigma_{ij}^4}\Big] a - \Big[\prod_{j=1}^{n} \exp(-\frac{(x_j-\mu_{ij})^2}{2\sigma_{ij}^2}) \frac{(x_j-\mu_{ij})^2}{\sigma_{ij}^3}\Big]^2}{a^2} \tag{15}$$

由模糊系统有规则得到的点火强度位于0 1之间，故分母$a > 0$,只要模糊规则中的前件参数$\sigma_{ij}$均为不趋于0 的正数，故可以使得$\frac{1}{\sigma_{ij}}$ 均有界，不难看出$f$ 对与$\sigma_{ij}$的二阶导数以及$f$ 对与$\mu_{ij}$的二阶导数公式中的各项均有界，所以$f$ 对与$\sigma_{ij}$的二阶导数以及$f$对与$\mu_{ij}$ 的二阶导数有界。亦即：$f'' = (f''_{y_1}, \cdots, f''_{y_n}, f''_{\mu_{11}}, \cdots, f''_{\mu_{Mn}}, f''_{\sigma_{11}}, \cdots, f''_{\sigma_{Mn}})$ 有界。

为了证明单值一型模糊系统输出公式满足梯度利普西茨条件，首先引入几个性质。设$Q$是$R^n$的子集，用$C_L^{k,p}(Q)$表示具有后面两个性质的函数类:(1)若$f \in C_L^{k,p}(Q)$,则$f$ 在$Q$ 上是$k$ 阶连续可微的;(2)$f$
的$p$阶导数在$Q$上是利普西茨连续的，利普西茨常数为$L$,即满足$||f^{(p)}(x)-f^{(p)}(x)|| \leq L||x-y||, (\forall x, y \in Q)$.

对于我们来说，最常用的函数类是$C_L^{1,1}(R^n)$,这类函数的梯度是利普西茨连续。由前面的定义，若$f \in C_L^{1,1}(R^n)$,则有:

$$||f^{(1)}(x) - f^{(1)}(x)|| \leq L||x - y||, (\forall x, y \in R^n) \tag{16}$$

下面介绍一个有助于证明的重要引理 [4]。

**引理1.** 设$f \in C_L^{2,1}(R^n)$,$C_L^{2,1}(R^n) \subset C_L^{1,1}(R^n)$当且仅当:

$$||f''(x)|| \leq L, \forall x \in R^n \tag{17}$$

证明. 事实上，对于$\forall x, y \in R^n$,有：

$$f'(y) = f'(x) + \int_0^1 f''(x + \tau(y - x))(y - x)d\tau \tag{18}$$

$$= f'(x) + \Big( \int_0^1 f''(x + \tau(y - x))d\tau \Big)(y - x) \tag{19}$$

因此，若条件$||f''(x)|| \leq L, \forall x \in R^n$满足，则有：

$$||f'(y) - f'(x)|| = \Big|\Big| \Big( \int_0^1 f''(x + \tau(y - x))d\tau \Big)(y - x) \Big|\Big| \tag{20}$$

$$\leq \Big|\Big| \int_0^1 f''(x + \tau(y - x))d\tau \Big|\Big| \cdot ||y - x|| \tag{21}$$

$$= \int_0^1 ||f''(x + \tau(y - x))||d\tau \cdot ||y - x|| \tag{22}$$

$$= L||y - x||. \tag{23}$$

反之，若$f \in C_L^{2,1}(R^n)$,则对于任意$s \in R^n$且$\alpha > 0$,我们有：

$$\Big|\Big| \Big( \int_0^\alpha f''(x + \tau s)d\tau \Big) \cdot s \Big|\Big| = ||f'(x + \alpha s) - f'(x)|| \leq \alpha L||s|| \tag{24}$$

在上式两边同时除以$\alpha$,再令$\alpha \to 0$ ,可以得到:$||f''(x)|| \leq L, \forall x \in R^n$. □

由上述引理1以及本文中所选择的单值一型模糊逻辑系统输出公式的二阶导数有界，故证明了单值一型模糊系统输出公式是满足梯度利普西茨条件的。

为方便下文的叙述，现在再介绍 [4]中的一个引理，它说明了$C_L^{1,1}(R^n)$这类函数的一个重要的性质。

**引理2.** 设$f \in C_L^{1,1}(R^n)$,对$\forall x, y \in R^n$,有：

$$|f(y) - f(x) - \langle f'(x), y, x\rangle| \leq \frac{L}{2}|y - x|^2 \tag{25}$$

证明. 对$\forall x, y \in R^n$有:

$$f(y) = f(x) + \int_0^1 \langle f'(x + \tau(y - x)), y - x\rangle d\tau \tag{26}$$

$$= f(x) + \langle f'(x), y - x\rangle + \int_0^1 \langle f'(x + \tau(y - x)) - f'(x), y - x\rangle d\tau \tag{27}$$

故：

$$|f(y) - f(x) - \langle f'(x), y - x\rangle| = |\int_0^1 \langle f'(x + \tau(y - x)) - f'(x), y - x\rangle| d\tau \tag{28}$$

$$\leq \int_0^1 |\langle f'(x + \tau(y - x)) - f'(x), y - x\rangle| d\tau \tag{29}$$

$$\leq \int_0^1 \|f'(x + \tau(y - x)) - f'(x)\| \cdot \|y - x\| d\tau \tag{30}$$

$$\leq \int_0^1 \tau L\|y - x\|^2 d\tau = \frac{L}{2}\|y - x\|^2 \tag{31}$$

□

## §2 优化算法

本节主要介绍梯度下降法和FW算法的实现步骤与收敛性，并介绍了FW算法的变形Away step FW。

### §2.1 梯度下降法

虽然梯度下降在深度学习领域中很少被直接使用，但是理解梯度的意义以及利用梯度反方向传播可以降低目标函数值是后续学习的基础. 以简单的一维梯度下降为例，解释梯度下降算法的作用。假设连续可导的函数$f : \mathbb{R} \to \mathbb{R}$的输入和输出都是标量。给定绝对值足够小的数$\epsilon$，由泰勒公式得到以下的近似：

$$f(x+\epsilon) \approx f(x) + \epsilon f'(x)$$

这里$f'(x)$是函数$f$在$x$处的梯度。接下来，找到一个常数$\eta > 0$，使得$|\eta f'(x)|$足够小，那么可以将$\epsilon$ 替换为$-\eta f'(x)$并得到：

$$f(x - \eta f'(x)) \approx f(x) - \eta f'(x)^2$$

如果导数$f'(x) \neq 0$，那么$\eta f'(x)^2 > 0$，所以:

$$f(x - \eta f'(x)) \leq f(x)$$

如果通过: $x \leftarrow x - \eta f'(x)$ 来迭代$x$，函数$f(x)$的值可能会降低。因此在梯度下降中，我们先选取一个初始值$x$ 和常数$\eta > 0$，然后不断通过上式来迭代$x$，直到达到停止条件。梯度下降算法中的正数$\eta$通常叫作学习率,也可以称为步长，这是一个超参数。

梯度下降法有很多的变形，主要是通过修改步长的选择策略得到的，令$\eta = h_k(k$ 为迭代的次数)，几个常用的步长策略如下所示：

- **方法1**

$$h_k = h > 0$$
$$h_k = \frac{h}{\sqrt{k+1}}$$

- **方法2**

$$h_k = \underset{h \geq 0}{\operatorname{argmin}} f(x_k - h f'(x_k))$$

- **方法3(Goldstein-Armijo rule)**

找到$x_{k+1} = x_k - h f'(x_k)$,$0 < \alpha < \beta < 1$是固定的参数，使得下面式子成立：

$$\alpha \langle f'(x_k), x_k - x_{k+1} \rangle \leq f(x_k) - f(x_{k+1}) \tag{32}$$

$$\beta \langle f'(x_k), x_k - x_{k+1} \rangle \geq f(x_k) - f(x_{k+1}) \tag{33}$$

接下来估计梯度下降方法的收敛速度，考虑如下问题：

$$\min_{x \in R^n} f(x) \tag{34}$$

其中$f \in C_L^{1,1}(R^n)$,且$f(x)$在$R^n$上有界，令$y = x - h f'(x)$,由引理2 可得：

$$f(y) \leq f(x) + \langle f'(x), y - x \rangle + \frac{L}{2}\|y - x\|^2 \tag{35}$$

$$= f(x) - h\|f'(x)\|^2 + \frac{h^2}{2} L \|f'(x)\|^2 \tag{36}$$

$$= f(x) - h(1 - \frac{h}{2}L)\|f'(x)\|^2 \tag{37}$$

为了找到最大的下降量，需要求解如下的一维问题：

$$\Delta(h) = -h(1 - \frac{h}{2}L) \to \min_h . \tag{38}$$

易知$\Delta(h)$在$h^* = \frac{1}{L}$取得最小值。从而目标函数至少下降：

$$f(y) \le f(x) - \frac{1}{2L}\|f'(x)\|^2. \tag{39}$$

令$x_{k+1} = x_k - h_k f'(x_k)$,对于常数步长$h_k = h$有：

$$f(x_k) - f(x_{k+1}) \ge h(1 - \frac{1}{2}Lh)\|f'(x_k)\|^2. \tag{40}$$

若选择$h_k = \frac{2\alpha}{L}(\alpha \in (0,1))$,则有：

$$f(x_k) - f(x_{k+1}) \ge \frac{2}{L}\alpha(1-\alpha)\|f'(x_k)\|^2. \tag{41}$$

对于第二种步长选择策略有：

$$f(x_k) - f(x_{k+1}) \ge \frac{1}{2L}\|f'(x_k)\|^2. \tag{42}$$

对于第三种步长选择策略(Goldstein-Armijo rule)有：

$$f(x_k) - f(x_{k+1}) \le \beta\langle f'(x_k), x_k - x_{k+1}\rangle = \beta h_k \|f'(x_k)\|^2. \tag{43}$$

$$f(x_k) - f(x_{k+1}) \ge h_k(1 - \frac{h_k}{2}L)\|f'(x_k)\|^2 \tag{44}$$

因此$h_k \ge \frac{2}{L}(1-\beta)$,由第三种步长选择策略所要满足的条件，有：

$$f(x_k) - f(x_{k+1}) \ge \alpha\langle f'(x_k), x_k - x_{k+1}\rangle = \alpha h_k \|f'(x_k)\|^2 \tag{45}$$

将上面式子相结合可得：

$$f(x_k) - f(x_{k+1}) \ge \frac{2}{L}\alpha(1-\beta)\|f'(x_k)\|^2 \tag{46}$$

从而对于所有情况都有：

$$f(x_k) - f(x_{k+1}) \ge \frac{w}{L}\|f'(x_k)\|^2 \tag{47}$$

其中$w$是正的常数。将上式从$k=0$加到$N$可得：

$$\frac{w}{L}\sum_{k=0}^{N}\|f'(x_k)\|^2 \le f(x_0) - f(x_{N+1}) \le f(x_0) - f^* \tag{48}$$

其中$f^*$是$\min\limits_{x\in R^n} f(x)$的最优值。且有:$\|f'(x_k)\| \to 0 (k \to \infty)$.令$g_N^* = \min\limits_{0\le k\le N} g_k$,$g_k = \|f'(x_k)\|$. 从而有：

$$g_N^* \le \frac{1}{\sqrt{N+1}}\big[\frac{1}{w}L(f(x_0) - f^*)\big]^{1/2} \tag{49}$$

上式表明了数列$g_N^*$趋于0的收敛速度。上述一维情形下得到的结果可以容易的扩展到高维的情形。

## §2.2 Frank-Wolfe算法及其变形

### §2.2.1 Frank-Wolfe算法

Frank-Wolfe算法是求解非线性约束的方法之一，近年来由于对内存要求较小和无投影迭代的特点而得到了广泛的应用。它可以解决如下的问题：

$$\underset{\boldsymbol{x}\in\mathcal{D}}{\text{minimize}}\, f(\boldsymbol{x}) \tag{50}$$

其中$f$可微并且满足梯度Lipschitz，定义域$\mathcal{D}$是凸紧集。

FW算法是一个容易实现的算法，它给出了一个初始点$\boldsymbol{x}_0$，并由此构造了一个迭代序列$\boldsymbol{x}_1, \boldsymbol{x}_2, ...$，这个序列可以收敛到优化问题的解。FW 算法的过程如下：

**算法 1** FW Algorithm

1: **Input**:initial guess $\boldsymbol{x}_0$,tolerance $\delta > 0$
2: **for** $t = 0, 1, ...$ **do**
3: $\quad \boldsymbol{s}_t \in \operatorname{argmax}_{\mathbf{s}\in\mathcal{D}} \langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{s}\rangle$
4: $\quad \boldsymbol{d}_t = \boldsymbol{s}_t - \boldsymbol{x}_t$
5: $\quad g_t = \langle -\nabla f(\boldsymbol{x}_t), \boldsymbol{d}_t\rangle$
6: $\quad$ **if** $g_t < \delta$ **then**
7: $\quad\quad$ Return $\boldsymbol{x}_t$
8: $\quad$ **end if**
9: $\quad$ **Variant 1**:设置步长 $\gamma_t = \min\left\{\frac{\mathrm{g_t}}{\mathrm{L}\|\boldsymbol{d}_\mathrm{t}\|^2}, 1\right\}$
$\quad$ **Variant 2**:线搜索设置步长 $\gamma_t = \operatorname{argmin}_{\gamma\in[0,1]} f(\boldsymbol{x}_t + \gamma\boldsymbol{d}_t)$
10: $\quad \boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \gamma_t\boldsymbol{d}_t$
11: **end for**
12: **Return** $\boldsymbol{x}_{t+1}$

与其他约束优化算法(如投影梯度下降法)相反，FW算法不需要投影，因此有时称之为无投影算法，但它需要求解**算法 1** 第3 步中的问题：

$$\boldsymbol{s}_t \in \underset{\boldsymbol{s}\in\mathcal{D}}{\operatorname{argmin}} \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{s}\rangle \tag{51}$$

算法的其余部分主要是寻找合适的步长，以便$f(\boldsymbol{x})$朝减小的方向移动。在FW 算法所允许的许多不同的步长规则中，我们将其分为变形1 和变形2。第1 个变形很容易计算，只依赖于Lipschitz常数L 的知识。第2 个变形需要在每次迭代时解决一个一维问题。在某些情况下如当目标函数$f$为最小二乘问题且约束集$\mathcal{D}$ 是$\ell_1$ 范数球时，步长选择这个问题有一个显式的最优解，在这种情况下，应首选变型2这种方法。但是，在一般情况下如果没有显式的最优解，在这种情况下，应首选变形1。

可以看到Frank-Wolfe算法是一种通过求解一系列线性规划问题来解决潜在非线性问题的算法，这种方法的有效性与快速解决线性子问题的能力紧密相连。事实证明，对于规模较大的问题的线性子问题既有闭式解，也有有效算法。与投影相比，使用线性最小化还有其他重要的结果。根据线性规划的性质，这种线性极小化问题的最优解总是约束集的一个顶点。还有一些其它设置步长大小的策略。例如，步长也可以设置为$\gamma_t = \frac{2}{t+2}$,它不依赖于优化产生的任何其它变量。因此，它在实践中不能与其它选择步长的策略竞争，尽管它确实达到了相同的理论收敛速度。对于$\gamma_t$还有一种选择，是由Demyanov和Rubinov 提出来的，与变形1比较相似：

$$\gamma_t = \min\Big\{\frac{\mathrm{g_t}}{\mathrm{Ldiam}(\mathcal{D})^2}, 1\Big\} \tag{52}$$

其中diam表示相对于欧几里德范数的直径,然而，由于根据直径的定义，我们总是有:

$$\|\boldsymbol{x}_t - \boldsymbol{s}_t\|^2 \leq \mathrm{diam}(\mathcal{D})^2 \tag{53}$$

这个变形提供的步长总是小于变形1的步长，并且给出了更差的收敛性。对该步长的进一步改进包括用局部估计替换Lipschitz常数$L$，从而允许更大的步长。

### §2.2.2 FW算法应用在非凸目标函数上的收敛性

**定义1.** 将Frank – Wolfe间隙记成$g_t$,定义为：

$$g_t = \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{s}_t \rangle . \tag{54}$$

引入下面的引理，该引理将两个连续迭代点的目标函数值联系起来，当目标函数非凸时，它在证明收敛过程中起到了重要作用。

**引理3.** 令$\boldsymbol{x}_0, \boldsymbol{x}_1, ...$是由**算法*1***产生的迭代序列，对于任意的$\xi \in [0,1]$，有下面的不等式：

$$f(\boldsymbol{x}_{t+1}) \leq f(\boldsymbol{x}_t) - \xi g_t + \frac{1}{2}\xi^2 L \mathrm{diam}(\mathcal{D})^2 \tag{55}$$

证明. 关于$f$的Lipschitz梯度假设的一个结果是，在$y \in \mathcal{D}$ 的每一点上有上界，即：

$$f(\boldsymbol{y}) \leq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{x}\|^2 \tag{56}$$

我们可以应用这个不等式，对定义域中的任何$\boldsymbol{x}, \boldsymbol{y}$ 都有效。对于$\gamma \in [0,1]$ 且$\boldsymbol{x} = \boldsymbol{x}_t, \boldsymbol{y} = (1-\gamma)\boldsymbol{x}_t + \gamma \boldsymbol{s}_t$的情况，$\boldsymbol{y}$依然在定义域中，我们有：

$$f((1-\gamma)\boldsymbol{x}_t + \gamma \boldsymbol{s}_t) \leq f(\boldsymbol{x}_t) + \gamma \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{s}_t - \boldsymbol{x}_t \rangle + \frac{L\gamma^2}{2}\|\boldsymbol{s}_t - \boldsymbol{x}_t\|^2 \tag{57}$$

当$\gamma \in [0,1]$时，需要最小化上面公式的右边部分。这是关于$\gamma$ 的二次函数，将$\gamma_t^*$记为：

$$\gamma_t^* = \min\Big\{\frac{\mathrm{g_t}}{\mathrm{L}\|\boldsymbol{x}_\mathrm{t} - \boldsymbol{s}_\mathrm{t}\|^2}, 1\Big\}. \tag{58}$$

当$\gamma = \gamma_t^*$时，有：

$$
\begin{aligned}
f((1-\gamma_t^*)\boldsymbol{x}_t+\gamma_t^*\boldsymbol{s}_t) &\le f(\boldsymbol{x}_t)-\gamma_t^* g_t+\frac{L(\gamma_t^*)^2}{2}\left\|\boldsymbol{s}_t-\boldsymbol{x}_t\right\|^2 \\
&= f(\boldsymbol{x}_t)+\min_{\xi\in[0,1]}\left\{-\xi \mathrm{g_t}+\frac{\mathrm{L}\xi^2}{2}\left\|\boldsymbol{s}_\mathrm{t}-\boldsymbol{x}_\mathrm{t}\right\|^2\right\} \\
&\le f(\boldsymbol{x}_t)-\xi g_t+\frac{L\xi^2}{2}\left\|\boldsymbol{s}_t-\boldsymbol{x}_t\right\|^2 \quad (\forall \xi\in[0,1]) \\
&\le f(\boldsymbol{x}_t)-\xi g_t+\frac{L\xi^2}{2}\mathrm{diam}(\mathcal{D})^2. \qquad (59)
\end{aligned}
$$

上述不平等式的右边已经包含引理中的结果，对于算法的变形1，我们有 $f(\boldsymbol{x}_{t+1})=f((1-\gamma_t^*)\boldsymbol{x}_t+\gamma_t^*\boldsymbol{s}_t)$,这种情况下 $\gamma_t$ 和 $\gamma_t^*$ 相等。对于算法的变形2, 我们有 $f(\boldsymbol{x}_{t+1})\le f((1-\gamma_t^*)\boldsymbol{x}_t+\gamma_t^*\boldsymbol{s}_t)$ ,,根据线搜索的定义，$f(\boldsymbol{x}_{t+1})$ 是最小化的目标函数值，因此两种情况都有：

$$
f(\boldsymbol{x}_{t+1})\le f((1-\gamma_t^*)\boldsymbol{x}_t+\gamma_t^*\boldsymbol{s}_t). \qquad (60)
$$

将该式与前一个不等式联系起来，便得到要证明的不等式。 □

下面是收敛速度结果，对于$f$可微且梯度Lipschitz的目标函数是有效的。

**定理1.** 如果 $f$ 是可微的且满足梯度Lipschitz，那么最小FW间隙的收敛速度以 $\mathcal{O}(1/\sqrt{t})$ 为上界：

$$
\min_{0\le i\le t} g_i \le \frac{\max\{2\mathrm{h}_0, \mathrm{Ldiam}(\mathcal{D})^2\}}{\sqrt{t+1}}, \qquad (61)
$$

其中 $h_0=f(\boldsymbol{x}_0)-\min_{\boldsymbol{x}\in\mathcal{D}} f(\boldsymbol{x})$ .

证明. 根据引理3,对于任意的$\xi\in[0,1]$有:

$$
\begin{aligned}
f(\boldsymbol{x}_{t+1}) &\le f(\boldsymbol{x}_t)-\xi g_t+\frac{L\xi^2}{2}\mathrm{diam}(\mathcal{D})^2 \\
&\le f(\boldsymbol{x}_t)-\xi g_t+\frac{C\xi^2}{2} \qquad (62)
\end{aligned}
$$

其中$C=L\mathrm{diam}(\mathcal{D})^2$ .考虑使右边公式最小化的$\xi$ 的值，可得$\xi^*=\min\{\mathrm{g_t/C},1\}$现在根据$\xi^*$的值做出选择，如果$g_t\le C$ , 则$\xi^*=g_t/C$，利用上一个不等式得到：

$$
f(\boldsymbol{x}_{t+1})\le f(\boldsymbol{x}_t)-\frac{g_t^2}{2C} \qquad (63)
$$

如果$g_t>C$ ,则$\xi^*=1$ ,有下面的不等式:

$$
\begin{aligned}
f(\boldsymbol{x}_{t+1}) &\le f(\boldsymbol{x}_t)-g_t+\frac{C}{2} \\
&\le f(\boldsymbol{x}_t)-\frac{g_t}{2} \qquad (64)
\end{aligned}
$$

结合两式得：

$$
f(\boldsymbol{x}_{t+1})\le f(\boldsymbol{x}_t)-\frac{g_t}{2}\min\Big\{\frac{g_t}{C},1\Big\} \qquad (65)
$$

其中$g_t^* = \min\limits_{0\le i\le t} g_i$,接下来对$g_t^*$进行选择，如果$g_t^* \le C$,则$\min\{g_t^*/C, 1\} = g_t^*/C$解前面关于$g_t^*$的不等式得：

$$g_t^* \le \sqrt{\frac{2Ch_0}{t+1}} \le \frac{2h_0 + C}{2\sqrt{t+1}} \le \frac{\max\{2\mathrm{h}_0, \mathrm{C}\}}{\sqrt{t+1}} \tag{66}$$

在第二个不等式中利用Young不等式$ab \le \frac{a^2}{2} + \frac{b^2}{2}$, $a = \sqrt{2h_0}$, $b = \sqrt{C}$.

如果$g_t^* > C$，则$\min\{g_t^*/C, 1\} = 1$,且有：

$$g_t^* \le \frac{2Ch_0}{t+1} \le \frac{2h_0}{\sqrt{t+1}} \le \frac{\max\{2\mathrm{h}_0, \mathrm{C}\}}{\sqrt{t+1}} \tag{67}$$

因此，在这两种情况下可得：

$$g_t^* \le \frac{\max\{2\mathrm{h}_0, \mathrm{C}\}}{\sqrt{t+1}} \tag{68}$$

□

### §2.2.3 Away step FW算法简介

FW算法从当前迭代点$\boldsymbol{x}^{(t)}$处，寻找下一个迭代点需要解一个线性规划步，这个解是可行集的顶点，再在这个顶点和迭代点$\boldsymbol{x}^{(t)}$之间作线搜索,这样会出现Zig-Zagging现象。下面介绍的Away step FW算法 [5]有效减少了Zig-Zagging 现象的发生。

**算法 2** Away step Frank Wolfe Algorithm :

1: 取初始可行点$\boldsymbol{x}^{(t)} \in \mathcal{A}$,令$\mathcal{S}^{(0)} = \boldsymbol{x}^{(0)}$,设置误差$\epsilon$. （若$\boldsymbol{v} = \boldsymbol{x}^{(0)}$,则$\alpha_{\boldsymbol{v}}^{(0)} = 1$,否则,$\alpha_{\boldsymbol{v}}^{(0)} = 0$)
2: **for** $t = 0, 1, ...T$ **do**
3: 令$\boldsymbol{s}_t = \underset{\boldsymbol{r}\in\mathcal{A}}{\operatorname{argmin}}\langle \boldsymbol{r}, \nabla f(\boldsymbol{x}^{(t)})\rangle, \boldsymbol{d}_t^{\mathrm{FW}} = \boldsymbol{s}_t - \boldsymbol{x}^{(t)}$.(FW方向)
4: 令$\boldsymbol{v}_t = \underset{\boldsymbol{v}\in\mathcal{S}^{(t)}}{\operatorname{argmax}}\langle \boldsymbol{v}, \nabla f(\boldsymbol{x}^{(t)})\rangle, \boldsymbol{d}_t^{\mathrm{A}} = \boldsymbol{x}^{(t)} - \boldsymbol{v}_t$.(Away step方向)
5: **if** $g_t^{\mathrm{FW}} = \langle -\nabla f(\boldsymbol{x}^{(t)}), \boldsymbol{v}\rangle \le \epsilon$ **then**
6: **return** $\boldsymbol{x}^{(t)}$
7: **end if**
8: **if** $\langle -\nabla f(\boldsymbol{x}^{(t)}), \boldsymbol{d}_t^{\mathrm{FW}}\rangle \ge \langle -\nabla f(\boldsymbol{x}^{(t)}), \boldsymbol{d}_t^{\mathrm{A}}\rangle$ **then**
9: **return** $\boldsymbol{d}_t = \boldsymbol{d}_t^{\mathrm{FW}}, \gamma_{\max} = 1$,(选择FW方向).
10: **else**
11: **return** $\boldsymbol{d}_t = \boldsymbol{d}_t^{\mathrm{A}}, \gamma_{\max} = \alpha_{\boldsymbol{v}_t}/(1 - \alpha_{\boldsymbol{v}_t})$,(选择Away step方向,极大化可行步长).
12: **end if**
13: 线搜索:$\gamma_t \in \underset{\gamma\in[0,\gamma_{\max}]}{\operatorname{argmin}} f(\boldsymbol{x}^{(t)} + \gamma\boldsymbol{d}_t)$.
14: 更新$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} + \gamma\boldsymbol{d}_t$.
15: 更新$\mathcal{S}^{(t+1)} = \mathbf{v} \in \mathcal{A}\, s.t.\, \alpha_{\boldsymbol{v}}^{(t+1)} > 0$.
16: **end for**

通过上述算法的第4步找到Away step方向，需要注意的是这里的线性规划问题是在积极集$\mathcal{S}^{(t)}$上求解的，它比直接在可行集$\mathcal{A}$上求解更容易。如同原始的FW算法，FW间隙$g_t^{\mathrm{FW}}$可以作为未知次

优性的上界，从而可以作为算法的停止条件：

$$g_t^{\mathrm{FW}} = \langle -\nabla f(\boldsymbol{x}^{(t)}), \boldsymbol{d}_t^{\mathrm{FW}} \rangle \geq \langle -\nabla f(\boldsymbol{x}^{(t)}), \boldsymbol{x}^* - \boldsymbol{x}^{(t)} \rangle \geq f(\boldsymbol{x}^{(t)}) - f(\boldsymbol{x}^*) \tag{69}$$

如果$\gamma_t = \gamma_{\max}$,我们称这一步为Drop step,因为这一步将$\boldsymbol{v}_t$从当前的积极集$\mathcal{S}^{(t)}$中移出(通过设置权重为0)。对于FW步，若$\gamma_t = 1$,则有$\mathcal{S}^{(t+1)} = \{\boldsymbol{s}_t\}$,否则$\mathcal{S}^{(t+1)} = \mathcal{S}^{(t)} \cup \{\boldsymbol{s}_t\}$.若$\boldsymbol{v} \in \mathcal{S}^{(t)} \cup \{\boldsymbol{s}_t\}$, $\alpha_{\boldsymbol{s}_t}^{(t+1)} = (1-\gamma_t)\alpha_{\boldsymbol{s}_t}^{(t)} - \gamma_t$,$\alpha_{\boldsymbol{v}_t}^{(t+1)} = (1-\gamma_t)\alpha_{\boldsymbol{v}_t}^{(t)}$ 对于Away step,若$\gamma_t = \gamma_{\max}$,则有$\mathcal{S}^{(t+1)} = \mathcal{S}^{(t)} \backslash \{\boldsymbol{v}_t\}$,否则$\mathcal{S}^{(t+1)} = \mathcal{S}^{(t)}$,对于$\boldsymbol{v} \in \mathcal{S}^{(t)} \backslash \{\boldsymbol{v}_t\}$ 有:$\alpha_{\boldsymbol{v}_t}^{(t+1)} = (1+\gamma_t)\alpha_{\boldsymbol{v}_t}^{(t)} - \gamma_t$, $\alpha_{\boldsymbol{v}_t}^{(t+1)} = (1+\gamma_t)\alpha_{\boldsymbol{v}_t}^{(t)}$.

## §3 在实际应用中的比较

### §3.1 Mackey-Glass方程

1977 年Mackey 和Glass 发表了一篇重要的论文，在论文中他们“把建模生理系统的一阶微分延迟方程的动力学分叉和疾病的发作联系起来”,Mackey-Glass 方程是一个非线性延迟微分方程，方程的形式为：

$$\frac{ds(t)}{dt} = \frac{0.2s(t-\tau)}{1+s^{10}(t-\tau)} - 0.1s(t) \tag{70}$$

使用3个前件来进行预测，即用$x(k-2), x(k-1)$ 和$x(k)$ 来预测$x(k+1)$. 每个前件采用1个高斯型模糊集，每个高斯型模糊集包含两个参数：均值和方差，一个后件参数。设置模糊规则数目为M，分别用梯度下降法，FW 算法以及Away step FW算法来设计单值一型模糊逻辑系统，需要根据原始的数据进行参数初始化。计算误差Error 时选择的公式为：

$$\mathrm{Error} = \frac{1}{2}\sum_{i=1}^{n}[f(\boldsymbol{x}_i) - y_i]^2 \tag{71}$$

其中$f(\boldsymbol{x}_i)$表示当输入样本为$\boldsymbol{x}_i$时单值一型模糊逻辑系统的输出，$y_i$ 表示样本对应的真实值。数值实验中选择的编程环境为MATLAB R2019a,每种方法的最大迭代次数设置为5000，梯度下降法选择的步长为2，这是多次实验后得到的使得训练速度较快的步长。

观察**表1**，设置训练时总的误差为5(总的误差较大),当规则数目确定时，梯度下降法所需的时间是最少的，迭代步数介于FW算法与Away step FW算法所需的迭代步数之间。Away step FW算法相对于FW算法显著减少了迭代的步数，算法的几何直观可以帮助更好的理解。3个算法每次寻找下一个迭代点时，Away step FW算法所需的时间比FW算法所需的时间多，FW算法所需的时间比梯度下降法所需的时间多。不难发现，Away step FW 算法在每一步需要求解梯度以及两个线性规划问题，FW 算法在每一步需要求解梯度和一个线性规划问题，梯度下降法只需要找到梯度即可。

**表 1: 设置训练时的总误差不超过5，不同算法的比较**

| 规则数 | FW 算法 | | | Away step FW 算法 | | | 梯度下降法 | | |
|---|---|---|---|---|---|---|---|---|---|
| | 迭代步数 | 测试误差 | 训练时间(秒) | 迭代步数 | 测试误差 | 训练时间(秒) | 迭代步数 | 测试误差 | 训练时间(秒) |
| 5 | 810 | 2.9505 | 15.5325 | 571 | 3.1863 | 15.9101 | 232 | 2.8617 | 0.5681 |
| 10 | 339 | 2.9469 | 10.0384 | 54 | 3.2634 | 2.1639 | 102 | 2.7978 | 0.5398 |
| 15 | 274 | 2.7232 | 9.6166 | 88 | 3.0181 | 3.8039 | 136 | 3.2573 | 0.7657 |
| 20 | 224 | 2.8806 | 9.5654 | 99 | 3.3880 | 4.8348 | 118 | 3.3845 | 0.8555 |
| 25 | 190 | 2.8170 | 9.6859 | 61 | 2.8887 | 3.4777 | 121 | 3.2427 | 1.0106 |
| 30 | 202 | 2.9027 | 11.9584 | 47 | 2.6805 | 3.0076 | 122 | 3.3139 | 1.2305 |
| 35 | 167 | 2.8917 | 10.9796 | 45 | 3.0015 | 3.2224 | 102 | 3.2245 | 1.1646 |
| 40 | 231 | 2.7581 | 17.0882 | 55 | 2.8683 | 4.3241 | 81 | 3.2438 | 1.0419 |
| 45 | 276 | 2.6750 | 22.5134 | 47 | 3.4711 | 4.0938 | 76 | 3.3009 | 1.0817 |
| 50 | 186 | 2.7341 | 16.7744 | 107 | 3.5702 | 10.4060 | 72 | 3.2838 | 1.1343 |

**表 2: 设置训练时的总误差不超过1，不同算法的比较**

| 规则数 | FW 算法 | | | Away step FW 算法 | | | 梯度下降法 | | |
|---|---|---|---|---|---|---|---|---|---|
| | 迭代步数 | 测试误差 | 训练时间(秒) | 迭代步数 | 测试误差 | 训练时间(秒) | 迭代步数 | 测试误差 | 训练时间(秒) |
| 5 | 5000 | 1.2342 | 88.4383 | 3170 | 0.5705 | 76.6651 | 180 | 0.5631 | 0.4379 |
| 10 | 1931 | 0.5696 | 48.3018 | 447 | 0.5603 | 14.0305 | 146 | 0.6270 | 0.5725 |
| 15 | 2654 | 0.5620 | 86.4427 | 197 | 0.6465 | 7.6898 | 251 | 0.6338 | 1.3543 |
| 20 | 1988 | 0.5326 | 79.3721 | 479 | 0.5709 | 22.6719 | 146 | 0.6498 | 1.0037 |
| 25 | 1894 | 0.5385 | 91.2719 | 368 | 0.6600 | 20.1005 | 125 | 0.6426 | 1.0662 |
| 30 | 1727 | 0.5329 | 96.423 | 180 | 0.6682 | 11.2563 | 97 | 0.6475 | 0.9997 |
| 35 | 1819 | 0.5375 | 115.5236 | 131 | 0.6737 | 9.1827 | 111 | 0.6484 | 1.2471 |
| 40 | 1578 | 0.5566 | 111.2758 | 74 | 0.5879 | 5.6297 | 133 | 0.5693 | 1.7640 |
| 45 | 1603 | 0.5368 | 131.0039 | 127 | 0.6379 | 10.9496 | 106 | 0.6378 | 1.5038 |
| 50 | 1594 | 0.5376 | 141.2413 | 94 | 0.5314 | 8.6341 | 97 | 0.6742 | 1.5606 |

观察**表2**，当设置训练时总的误差为1(总的误差较小)时，对于这样小规模的问题，使用FW算法训练模糊逻辑系统所费的时间几乎是梯度下降法的几百倍。虽然Away step FW算法有效降低了迭代步数，花费了相对于FW 算法的较少时间，但是无法与梯度下降法相媲美。即使在使用梯度下降法时，我们可能需要花费一定的时间去寻找最优的步长从而加快算法的收敛速度。

## 参考文献


[1] Mamdani E H. Applications of fuzzy algorithms for simple dynamic plant [J]. Proc. IEE., 1974, 121(12): 1585 1588.

[2] 王立新著, 王迎军译, 模糊系统与模糊控制教程[M], 北京：清华大学出版社, 2003.

[3] J. M. Mendel, Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions[M], Prentice-Hall International (UK) Limited, 2001.

[4] Yurii Nesterov,Introductory lectures on convex optimization[M],Kluwer Academic Publishers,2004.

[5] Simon Lacoste-Julien,Martin Jaggi,On the Global Linear Convergence of Frank-Wolfe Optimization Variants[J].