# Boosting Frank-Wolfe by Chasing Gradients

高世杰、晏浩洋

SIST University

May 24, 2021

# 凸问题

对带有凸约束的凸问题，一阶算法中有两种较为常用的算法，本质均为选择近似负梯度方向作为迭代方向。一是梯度投影法，计算梯度并投影回可行域。二是 Frank-wolfe 算法，通过选择可行域内与负梯度内积最大方向进行迭代。

Frank-wolfe 算法优点在于不需要投影，计算量小，只需起始点在可行域内，此后的迭代点均在可行域内。问题形式：

Let $\mathcal{H}$ be a Euclidean space,consider

$$\min f(x)$$
$$\text{s.t. } x \in \mathcal{C}$$

where

$f: \mathcal{H} \to \mathbb{R}$ is a smooth convex function

$\mathcal{C} \subset \mathcal{H}$ is a compact convex set, $\mathcal{C} = \text{conv}(\mathcal{V})$

# Frank-Wolfe 算法及其存在的问题

**Algorithm 1:** Frank-Wolfe(FW)

1: **Input:** Start point $x_0 \in \mathcal{C}$ , $\gamma_t \in [0, 1]$.
2: **for** $i = 1$ **to** ... **do**
3: $\quad v_t \leftarrow \arg\max_{v \in \mathcal{V}} \langle \nabla f(x_t) , v \rangle$ (1)
4: $\quad x_{t+1} \leftarrow x_t + \gamma_t (v_t - x_t)$ (2)
5: **end for**

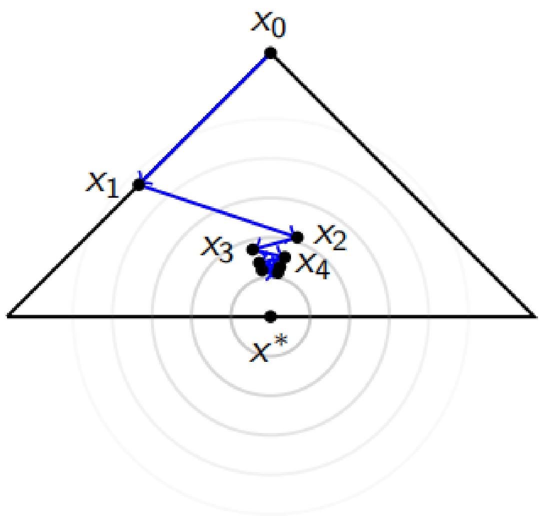
考虑问题:

$\min \frac{1}{2}\|x\|_2^2$

$x \in \text{conv}\left(\begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}\right)$

start point:$x_0 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

可以看到 Frank-Wolfe 算法容易出现 zig-zagging 的情况，其问题就在于无法保证迭代方向与负梯度方向有较好的拟合



# 改进想法及案例分析

改进想法：找拟合 $-\nabla f(x_t)$ 更好的方向进行迭代。考虑右例

(1) $v_0 \leftarrow \arg\max_{v\in\mathcal{V}} \langle \nabla f(x_t)\,,v\rangle$

(2) $\lambda_0 u_0 = \frac{\langle -\nabla f(x_t), v_0 - x_t\rangle}{\|v_0 - x_t\|^2}\,(v_0 - x_t)$

(3) $r_1 = -\nabla f(x_t) - \lambda_0 u_0$

(4) $v_1 \leftarrow \arg\max_{v\in\mathcal{V}} \langle r_1, v\rangle$

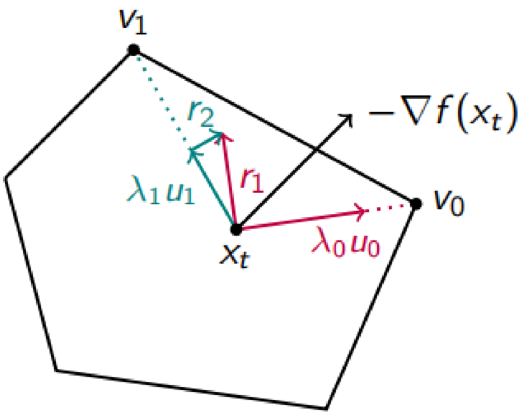
(5) $\lambda_1 u_1 = \frac{\langle r_1, v_1 - x_t\rangle}{\|v_1 - x_t\|^2}\,(v_1 - x_t)$

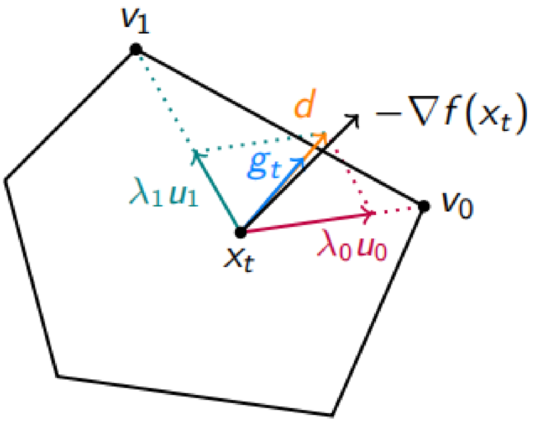
(6) $r_2 = r_1 - \lambda_1 u_1$

(7) $d = \lambda_0 u_0 + \lambda_1 u_1$

(8) $g_t = d/\,(\lambda_0 + \lambda_1)$

在同一迭代点 $x_t$ 进行多轮拟合近似 $-\nabla f(x_t)$。右例中第一轮对 $-\nabla f(x_t)$ 选择内积最大方向进行投影（同 FW）。若拟合残差 $r_1$ 过大，则继续对残差进行投影拟合。通过多轮拟合缩小残差最终求得近似 $-\nabla f(x_t)$ 方向作为迭代方向。



# 算法思想及收敛性分析

## 算法思想:

Frank-Wolfe 算法存在无法保证迭代方向与 $-\nabla f(x_t)$ 方向较好拟合的问题。一种改进的算法应满足更好拟合 $-\nabla f(x_t)$ 方向，且不可过多增加额外计算量，并且也应满足 Frank-Wolfe 算法的性质即迭代点 $x_t \in \mathcal{C}, \forall t$。

Boosting Frank-Wolfe 算法使用了与 Frank-Wolfe 相同的信息 ($\mathcal{V}$(顶点集合),$\nabla f(x_t)$)。通过不断对残差 $r$ 向 $v_s - x_t$ for some $v_s \in \mathcal{V}$ 方向投影，既优化了拟合方向，也因使用的均为顶点的信息，在对迭代方向 $d_t = \Lambda_t g_t$ 进行一定比例缩放后，保证了 $x_t + g_t \in \mathcal{C}$，也即有 $x_{t+1} = x_{t+1} + \gamma_t g_t \in \mathcal{C}$。详细证明可见附录或 report。

## 收敛性分析:

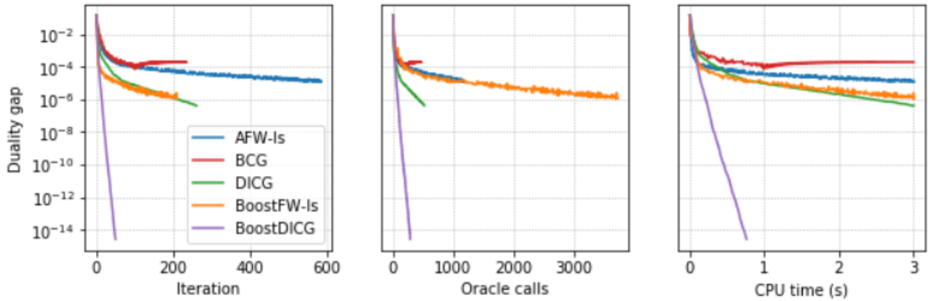
函数满足条件: f is convex，L-smooth 和 $\mu$-gradient(Frank-Wolfe 为 convex，L-smooth)

Worst-case rate: $O(\frac{1}{\varepsilon})$ 次线性收敛 (同 Frank-Wolfe)

Practical rate: $O\left(\ln \frac{1}{\varepsilon}\right)$ 线性收敛

总结：实际收敛速率的证明所增加的条件为在完整过程中满足步长 $\gamma_s < 1$ 且使用 Boosting 拟合 $-\nabla f(x_t)$ 超过一次的迭代点 $x_t$ 有一定的数量 (也即假设 Boosting procedure 被频繁使用)，这也是符合实际实验情况的。

# 计算实验，算法意义及可能改进分析




**算法意义：**

Boosting Frank-Wolfe 算法能够在不增加过多额外计算量的同时更好的拟合负梯度方向，从而提升了 FW 算法的效率，并且对于同样使用寻找迭代方向一步的类 FW 算法也能添加 Boosting procedure 提升其效率，从上述数值实验可以观察到这一点。

**存在问题：**

在一个迭代点 $x_t$ 处使用的 Boosting 的次数可能非常多，其原因在于拟合方向与负梯度方向夹角非常大，也即每次投影残差来拟合的改进量非常小。因此可以考虑算法增加每次改进的角度。由于 Boosting 每轮均需算 $v_k \leftarrow \arg\max_{v \in \mathcal{V}} \langle r_k, v\rangle$，且已计算所有 $\langle r_k, v\rangle$(拥有大量已知信息)，因此若仅增加额外几步计算来减少拟合轮数能大幅降低计算量。

**可能改进：**

构造新顶点 $\lambda v_t + (1-\lambda) v_t \in \mathcal{C}$(凸集) 或更多顶点的线性组合，使得在一轮中就能更好拟合负梯度方向 $d = v' - x_t$，既只是用了已有信息，又因使用同 Boosting 中的顶点信息，可易证也满足 $x_{t+1} \in \mathcal{C}$ 的重要性质。

# 附录:Boosting Frank-Wolfe Algorithm

**Algorithm 2** Boosted Frank-Wolfe (BoostFW)

**Input:** Input point $y \in \mathcal{C}$, maximum number of rounds $K \in \mathbb{N}\setminus\{0\}$, alignment improvement tolerance $\delta \in ]0,1[$, step-size strategy $\gamma_t \in [0,1]$.

**Output:** Point $x_T \in \mathcal{C}$.

$x_0 \leftarrow \arg\min_{v\in\mathcal{V}} \langle \nabla f(y), v\rangle$

**for** $t = 0$ **to** $T-1$ **do**

  $d_0 \leftarrow 0$

  $\Lambda_t \leftarrow 0$

  flag $\leftarrow$ **false**

  **for** $k = 0$ **to** $K-1$ **do**

    $r_k \leftarrow -\nabla f(x_t) - d_k$ ▷ $k$-th residual

    $v_k \leftarrow \arg\max_{v\in\mathcal{V}} \langle r_k, v\rangle$ ▷ FW oracle

    $u_k \leftarrow \arg\max_{u\in\{v_k - x_t, -d_k/\|d_k\|\}} \langle r_k, u\rangle$

    $\lambda_k \leftarrow \dfrac{\langle r_k, u_k\rangle}{\|u_k\|^2}$

    $d'_k \leftarrow d_k + \lambda_k u_k$

    **if** align$(-\nabla f(x_t), d'_k)$ − align$(-\nabla f(x_t), d_k) \geqslant \delta$ **then**

      $d_{k+1} \leftarrow d'_k$

      $\Lambda_t \leftarrow \begin{cases} \Lambda_t + \lambda_k & \textbf{if } u_k = v_k - x_t \\ \Lambda_t(1 - \lambda_k/\|d_k\|) & \textbf{if } u_k = -d_k/\|d_k\| \end{cases}$

    **else**

      flag $\leftarrow$ **true**

      **break** ▷ exit $k$-loop

    **end if**

  **end for**

  $K_t \leftarrow k$ **if** flag $=$ **true else** $K$

  $g_t \leftarrow d_{K_t}/\Lambda_t$ ▷ normalization

  $x_{t+1} \leftarrow x_t + \gamma_t g_t$

**end for**

# 附录: 定理及证明

Lemma: $x^* = \arg\min_{v\in\mathcal{V}} \langle r, v\rangle = \arg\min_{z\in\mathcal{C}} \langle r, v\rangle, \forall r \in \mathcal{R}^n$.Argmax has the same result.

$$
\begin{aligned}
\forall z \in C, C &= \mathrm{conv}(\mathcal{V}) \\
z &= \sum_{v_i\in\mathcal{V}} a_{v_i} v_i, \sum_{v_i\in\mathcal{V}} a_{v_i} = 1 \\
\langle r, x^*\rangle - \langle r, z\rangle &= \langle r, x^*\rangle - \langle r, \sum_{v_i\in\mathcal{V}} a_{v_i} v_i\rangle \\
&\leq \langle r, x^*\rangle - \langle r, \sum_{v_i\in\mathcal{V}} a_{v_i} v_i\rangle \\
&= 0 \\
x^* &= \arg\min_{v\in\mathcal{V}} \langle r, v\rangle = \arg\min_{z\in\mathcal{C}} \langle r, v\rangle, \forall r \in \mathcal{R}^n
\end{aligned}
$$

# 附录: 定理及证明

Theorem1 $\forall\, k \in N, \langle r_k, u_k \rangle \geq 0, \lambda_k > 0$

$$\begin{aligned} &By\ algorithm,\ \ v_k = \arg\max_{v \in \mathcal{V}} \langle r_k, v \rangle = \arg\max_{z \in \mathcal{C}} \langle r_k, z \rangle \\ &\langle r_k, u_k \rangle = \langle r_k, v_k \rangle - \langle r_k, x_t \rangle \\ &\geq 0\ (By\ Lemma) \\ &\lambda_k = \frac{\langle r_k, u_k \rangle}{\left\| u_k \right\|^2} > 0 \end{aligned}$$

*Since if* $\lambda_k = 0$ *then* $r_k = 0$ *or* $\langle r_k, u_k \rangle = 0$, *the algorithm is already convergence by algorithm.*

# 附录: 定理及证明

Theorem2 *If* $x_t \in \mathcal{C}, x_t + g_t \in \mathcal{C}$

*Define* $K_t$ *as times of update of d, by algorithm, we have* $K_t \geq 1$

$$d = \sum_{k=0}^{k_t-1} \lambda_k \left(v_k - x_t\right)$$

$$g_t = \frac{1}{\Lambda_t} \sum_{k=0}^{k_t-1} \lambda_k \left(v_k - x_t\right)$$

$$= \frac{1}{\Lambda_t} \sum_{k=0}^{k_t-1} \lambda_k v_k - x_t$$

$$\frac{1}{\Lambda_t} \sum_{k=0}^{k_t-1} \lambda_k v_k \in \mathcal{C} \ \mathit{for} \ v_k \in \mathcal{C} \ \mathit{and} \ \frac{1}{\Lambda_t} \sum_{k=0}^{k_t-1} \lambda_k = 1$$

$$x_t + g_t \in \mathcal{C}$$

# 附录: 定理及证明

Theorem3 *If* $x_t \in \mathcal{C}, x_{t+1} \in \mathcal{C}$

$$\begin{aligned}
Let\ y_t &= \frac{1}{\Lambda_t}\sum_{k=0}^{k_t-1}\lambda_k v_k \\
x_{t+1} &= x_t + \gamma_t g_t \\
&= x_t t + \gamma_t\left(y_t - x_t\right) \\
&= \left(1-\gamma_t\right)x_t + \gamma_t y_t \in \mathcal{C}
\end{aligned}$$

# 结尾

谢谢大家