(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 114611708 A
(43)申请公布日 2022.06.10

(21)申请号 202210218703.6

(22)申请日 2022.03.04

(71)申请人 中国人民解放军国防科技大学
地址 410003 湖南省长沙市开福区德雅路109号

(72)发明人 罗来龙 郭得科 胡煜晗 赵亚威

(74)专利代理机构 北京风雅颂专利代理有限公司 11403
专利代理师 李博瀚

(51)Int.Cl.
G06N 20/00(2019.01)
G06F 17/18(2006.01)
G06F 17/16(2006.01)

权利要求书1页 说明书11页 附图1页

(54)发明名称

一种平滑集上的无投影分布式在线学习方法

(57)摘要

本发明提出了一种平滑集上的无投影分布式在线学习方法，该方法试图找到一个近似梯度，以确保下一个决策在$\mathcal{K}$中，在每次迭代中，每个玩家都会执行以下步骤:查询所有邻居的对偶变量，并更新对偶变量;计算下一个动作，并判断该动作指向的新决策是否在可行集$\mathcal{K}$中;若新决策点在可行集$\mathcal{K}$中，则更新新决策;若决策点不在可行集$\mathcal{K}$中，则通过快速近似投影算法找到能落在可行集$\mathcal{K}$中的新决策。本方法利用近似投影的方法提高了计算效率，同时控制了噪声，并保证了算法对遗憾是无害的，且能够获得最优遗憾$\mathcal{O}\left(nGD\sqrt{(\mu D+1)T}\right)$，其中G用来度量损失函数的范数，D用来度量决策空间的大小，μ用来度量近似投影引起的噪声的大小。

CN 114611708 A

1.一种平滑集上的无投影分布式在线学习方法,其特征在于,所述方法包括:

对于无向网络$\mathcal{G}=(\mathcal{V},\mathcal{E})$上的分布式在线凸优化,$\mathcal{V}=[n]$是顶点集,$\mathcal{E}\subset\mathcal{V}\times\mathcal{V}$是边集,每个节点都是一个玩家,并且只能与相邻节点$N_i=\{j\in\mathcal{V}|(i,j)\in\mathcal{E}\}$进行通信,玩家$i\in\mathcal{V}$持有一个决策变量$x_i(t)$和一个对偶变量$z_i(t)$;

获取对手给出的凸损失函数$f_{t,i},\forall i\in\mathcal{V}$,并根据凸损失函数$f_{t,i}$计算次梯度$\mathbf{g}_i(t)\in\partial f_{t,i}(\mathbf{x}_i(t))$;

收集其邻居的对偶变量$\{z_j(t),j\in N_i\}$,并结合次梯度$g_i(t)$来计算一个新的对偶变量$z_i(t+1)$;

根据对偶变量$z_i(t+1)$计算下一个动作v_i,并判断基于动作v_i的决策点$x_i(t)+v_i$是否在可行集$\mathcal{K}$中;

若决策点$\mathbf{x}_i(t)+\mathbf{v}_i\in\mathcal{K}$,则将$x_i(t)+v_i$更新为新决策;若决策点$\mathbf{x}_i(t)+\mathbf{v}_i\notin\mathcal{K}$,则通过快速近似投影算法找到能落在可行集$\mathcal{K}$中的新决策。

2.根据权利要求1所述的分布式在线学习方法,其特征在于,

$$\mathbf{z}_i(t+1)==\sum_{j\in N_i}\mathbf{P}_{ij}\mathbf{z}_j(t)+\mathbf{g}_i(t)$$

$$\sum_{j=1}^{n}P_{ij}=\sum_{i=1}^{n}P_{ij}\quad for\quad\forall i,j\in\mathcal{V}$$

其中,P_{ij}为P中的元素,P是一个用来模拟每对玩家之间通信情况的双随机矩阵。

3.根据权利要求1所述的分布式在线学习方法,其特征在于,

$v_i=-\alpha_i(t)\cdot(z_i(t+1)-z_i(t))$

其中,$\alpha_i(t)$是来自非递增序列的一个正步长。

4.根据权利要求1所述的分布式在线学习方法,其特征在于,所述通过快速近似投影算法找到能落在可行集$\mathcal{K}$中的新决策,包括:

找到线段$(x_i(t),x_i(t)+v_i]$与$\mathcal{K}$边界$\partial\mathcal{K}$的交点$\tilde{\mathbf{x}}_i$;

计算交点$\tilde{\mathbf{x}}_i$的单位法向量n_i,作为新行动的投影方向;

将决策点$x_i(t)+v_i$向单位法向量n_i的反方向投影到可行集$\mathcal{K}$上得到新决策。

5.根据权利要求1-4所述的分布式在线学习方法,其特征在于,对于任何玩家$i\in\mathcal{V}$,对偶变量$z_i(1)$的初始值设置为0。

一种平滑集上的无投影分布式在线学习方法

技术领域

[0001] 本发明属于机器学习技术领域，具体涉及一种平滑集上的无投影分布式在线学习方法。

背景技术

[0002] 在线学习是一种重要的学习范式，旨在利用现有信息和历史行为预测后续行为。在每次迭代中，在线学习的玩家在获得上次行动对手的反馈(损失值)后，决定下一步采取一个特定的行动来赢得游戏。该学习范式可广泛应用于在线广告投放、在线音乐推荐、在线网站排名等产生时序数据的场景。在线梯度下降(OGD)作为一种通用的决策方法，在在线学习中得到了广泛的应用。然而在实际场景中，玩家可以选择的行动通常是受到限制的。这意味着如果通过OGD得出的决策超出了可行集，那么该决策是不可行的。因此，需要一个投影步骤来确保OGD决定的每个操作都包含在可行集中。

[0003] 然而，投影步骤通常会导致较高的计算成本。原因是在可行集上的投影需要解决一个二次规划问题，并且在使用OGD时通常会有成百上千次的迭代。因此，在约束集上执行OGD的计算开销很大。这种高昂的计算成本严重影响了在线学习方法的收敛速度。因此，简化投影的计算是很有必要的，尤其是对于大规模分布式在线学习来说。

[0004] 在过去的几十年里，许多研究者致力于寻找一种有效的无投影方法来降低计算成本。在文献(E.Hazan and S.Kale,“Projection-free online learning,”arXiv preprint arXiv:1206.4657,2012.)中首次提出了用于解决这一二次规划问题的经典方法。该方法的在线版本，即Online Frank-Wolfe(OFW)，由Hazan和Kale提出。在OFW的基础上，更多的变体方法被提出用来解决投影问题。但是这些方法的思想都是采用线性优化方式对全投影过程进行替换从而简化计算，实现近似投影。这种方法无论是从理论上还是实验上都被证明能有效地降低计算成本。因此，这些方法也被直接地应用到了分布式在线学习中。但是这些方法仅仅能获得$\mathcal{O}\left(nT^{3/4}\right)$的遗憾边界，与最优遗憾$\mathcal{O}\left(n\sqrt{T}\right)$之间仍有很大的差距。因此，在分布式环境下投影问题仍有待进一步解决。

[0005] 目前，针对分布式网络下的在线学习已经开展了大量的研究，然而关于无投影分布式在线学习的研究却很少。第一个无投影的分布式在线学习方法(DOL)，分布式在线条件梯度(Distributed Online Conditional Gradient,D-OCG)通过对每个本地参与方进行OFW,D-OCG可以实现一个$\mathcal{O}\left(nT^{3/4}\right)$的遗憾边界。一种改进的D-OCG，即分布式分块在线条件梯度(D-BOCG)，将通信复杂度从$\mathcal{O}(T)$降低到$\mathcal{O}\left(\sqrt{T}\right)$，并保持$\mathcal{O}\left(nT^{3/4}\right)$的遗憾。在文献(Learning(ICML),pp.9818-9828,2020)中对D-BOCG进行了进一步的改进，使其能够在强凸函数下取得更好的遗憾边界$\mathcal{O}\left(nT^{2/3}\log T\right)$。但是，这些方法所获得的遗憾在分布式设置中仍远未达到最优。

发明内容

[0006] 本发明为了解决上述问题，提出了一种平滑集上的无投影分布式在线学习方法，可以很好地平衡精度和效率，所述方法包括：

[0007] 对于无向网络$\mathcal{G}=(\mathcal{V},\mathcal{E})$上的分布式在线凸优化，$\mathcal{V}=[n]$是顶点集，$\mathcal{E}\subset\mathcal{V}\times\mathcal{V}$是边集，每个节点都是一个玩家，并且只能与相邻节点$N_i=\{j\in\mathcal{V}\,|\,(i,j)\in\mathcal{E}\}$进行通信，玩家$i\in\mathcal{V}$持有一个决策变量$x_i(t)$和一个对偶变量$z_i(t)$；

[0008] 获取对手给出的凸损失函数$f_{t,i},\forall i\in\mathcal{V}$，并根据凸损失函数$f_{t,i}$计算次梯度$\mathbf{g}_i(t)\in\partial f_{t,i}\left(\mathbf{x}_i(t)\right)$；

[0009] 收集其邻居的对偶变量$\{z_j(t),j\in N_i\}$，并结合次梯度$g_i(t)$来计算一个新的对偶变量$z_i(t+1)$；

[0010] 根据对偶变量$z_i(t+1)$计算下一个动作v_i，并判断基于动作v_i的决策点$x_i(t)+v_i$是否在可行集$\mathcal{K}$中；

[0011] 若决策点$\mathbf{x}_i(t)+\mathbf{v}_i\in\mathcal{K}$，则将$x_i(t)+v_i$更新为新决策；若决策点$\mathbf{x}_i(t)+\mathbf{v}_i\notin\mathcal{K}$，则通过快速近似投影算法找到能落在可行集$\mathcal{K}$中的新决策。

[0012] 进一步的，对偶变量$z_i(t+1)$的计算如下：

[0013] $$\mathbf{z}_i(t+1)==\sum_{j\in N_i}\mathbf{P}_{ij}\mathbf{z}_j(t)+\mathbf{g}_i(t)$$

[0014] $$\sum_{j=1}^{n}P_{ij}=\sum_{i=1}^{n}P_{ij}\quad for\quad\forall i,j\in\mathcal{V}$$

[0015] 其中，P_{ij}为P中的元素，P是一个用来模拟每对玩家之间通信情况的双随机矩阵。

[0016] 进一步的，所述根据对偶变量$z_i(t+1)$计算下一个动作v_i，具体为：

[0017] $v_i=-\alpha_i(t)\cdot(z_i(t+1)-z_i(t))$

[0018] 其中，$\alpha_i(t)$是来自非递增序列的一个正步长。

[0019] 进一步的，所述通过快速近似投影算法找到能落在可行集$\mathcal{K}$中的新决策，包括：

[0020] 找到线段$(x_i(t),x_i(t)+v_i]$与$\mathcal{K}$边界$\partial\mathcal{K}$的交点$\tilde{\mathbf{x}}_i$；

[0021] 计算交点$\tilde{\mathbf{x}}_i$的单位法向量n_i，作为新行动的投影方向；

[0022] 将决策点$x_i(t)+v_i$向单位法向量n_i的反方向投影到可行集$\mathcal{K}$上得到新决策。

[0023] 进一步的，对于任何玩家$i\in\mathcal{V}$，对偶变量$z_i(1)$的初始值设置为0。

[0024] 本发明所提出的一种平滑集上的无投影分布式在线学习方法(Distributed Fast Approximate Projection，D-FAP)，在每次迭代中，它都根据几何方法进行近似投影。在模型更新过程中，利用可行集的几何结构进行近似投影，可以很好地平衡精度和效率。直观上，当可行集具有良好条件结构时，即可行集的曲率很小时，近似投影产生的噪声也会很小，因此近似投影会非常准确。然而，如果可行集是尖的，即曲率较大时，可以进行小步长更新，以保持近似投影的准确性。换句话说，本方法可以根据曲率自适应地调整步长，在近似投影中保持轻微的噪声，且每一个近似投影都能在常数时间内完成，因此其计算成本明显低于其他算法。另外，本方法能使得近似投影的每一步都具有$\mathcal{O}(1)$复杂度，同时还能达到最

优遗憾边界$\mathcal{O}\left(n\sqrt{T}\right)$，并通过大量的实验证明了D-FAP对不同的数据集具有足够的有效性和鲁棒性。

附图说明

[0025] 为了更清楚地说明本发明实施例或现有技术中的技术方案，下面将对实施例或现有技术描述中所需要使用的附图作简单地介绍，显而易见地，下面描述中的附图仅仅是本发明的一些实施例，对于本领域普通技术人员来讲，在不付出创造性劳动的前提下，还可以根据这些附图示出的结构获得其他的附图。

[0026] 图1为一种平滑集上的无投影分布式在线学习方法流程图。

[0027] 图2为分布式快速近似投影示意图。

具体实施方式

[0028] 为了使本申请的目的、技术方案及优点更加清楚明白，以下结合附图及实施例，对本申请进行进一步详细说明。应当理解，此处描述的具体实施例仅仅用于解释本申请，并不用于限定本申请。

[0029] 本发明提供的一种平滑集上的无投影分布式在线学习方法，通过寻找一个基于可行集几何结构的近似投影点来代替完整的投影步骤。由于近似投影步骤可以在常数时间内完成，因此其计算成本比其他方法低得多。

[0030] 在本申请中使用了以下符号：

[0031] $\mathcal{A}$表示了所有可能的在线算法族；

[0032] $\mathcal{F}$表示所有可用的损失函数族，其中对于任何损失函数$f_{t,i}\in\mathcal{F}:\mathcal{K}\to\mathbb{R}$满足以下的假设1。$\mathcal{F}^T$表示函数乘积空间$\underbrace{\mathcal{F}\times\mathcal{F}\times\cdots\times\mathcal{F}}_{T\text{ times}}$；

[0033] 粗体的小写字母，例如x，表示向量。普通的字母，例如α，表示标量；

[0034] $||\cdot||_p$表示l_p范数。$||\cdot||$默认表示l_2范数；

[0035] [n]:＝{1,...,n}和[T]:＝{1,...,T}；

[0036] [x,y]表示向量x向量y之间的线段，其中$\mathbf{x},\mathbf{y}\in\mathbb{R}^{\mathbf{d}}$；

[0037] 集合$\mathcal{K}$的直径定义为$D:=\sup_{\mathbf{x},\mathbf{y}\in\mathcal{K}}\|\mathbf{x}-\mathbf{y}\|$；

[0038] 集合$\mathcal{K}$的边界和内部分别用$\partial\mathcal{K}$和$int(\mathcal{K})$进行表示；

[0039] $\{\alpha_i(t)\}_{t=1}^{T}$表示包含T个向量的一个序列，即{$\alpha_i(1),\alpha_i(2),...,\alpha_i(T)$}；

[0040] $\lesssim$表示“小于或等于一个常数因子”，而$\gtrsim$表示“大于或等于一个常数因子”；

[0041] ∂表示次梯度运算符。

[0042] 所述分布式在线学习方法具体为：

[0043] 在无向网络$\mathcal{G}=(\mathcal{V},\mathcal{E})$上定义分布式在线凸优化(OCO)，其中$\mathcal{V}=[n]$是顶点集，$\mathcal{E}\subset\mathcal{V}\times\mathcal{V}$是边集，OCO可以被看作是玩家和对手之间的游戏。而在分布式OCO的设置中，每个节点都是

一个玩家，并且只能与相邻节点$N_i=\{j\in\mathcal{V}|(i,j)\in\mathcal{E}\}$进行通信。在每一次迭代t∈[T]中，每个玩家$i\in\mathcal{V}$会做出一个决策$\mathbf{x_i(t)}\in\mathcal{K}$，然后从对手那里得到一个凸损失函数$f_{t,i}(\mathbf{x}):\mathcal{K}\to\mathbb{R}$。因此，全局损失函数$F_t(x)$可以定义为所有局部损失函数的总和。

[0044] $$F_t(\mathbf{x}):=\sum_{j=1}^{n}f_{t,j}(\mathbf{x})$$

[0045] 此外，可行集$\mathcal{K}$是在假设1下的光滑集，

[0046] $$\mathcal{K}:=\left\{\mathbf{x}\in\mathbb{R}^d:h(\mathbf{x})\le 0\right\}$$

[0047] 其中，$h(\cdot):\mathbb{R}^d\to\mathbb{R}$。

[0048] 所述假设1为：

[0049] 可行集$\mathcal{K}:=\left\{\mathbf{x}\in\mathbb{R}^{\mathbf{d}}:h(\mathbf{x})\le\mathbf{0}\right\}$是一个$\beta_h$-smooth的集合，因为h(•)是一个$\beta_h$-smooth的凸函数，即$h(\mathbf{y})\le h(\mathbf{x})+\nabla h(\mathbf{x})^{\top}(\mathbf{y}-\mathbf{x})+\frac{\beta_h}{2}\|\mathbf{x}-\mathbf{y}\|^2,\forall\mathbf{x},\mathbf{y}\in\mathbb{R}^d$。

[0050] 对于$\forall t\in[T]$且$\forall i\in[n]$，假设$f_{t,i}$是凸的，并且有G-Lipschitz梯度。

[0051] 函数$f:=\mathcal{K}\to\mathbb{R}$在$\mathcal{K}$集合中是β-smooth的，如果对$\forall\mathbf{x},\mathbf{y}\in\mathcal{K}$有，

[0052] $$f(\mathbf{y})\le f(\mathbf{x})+\nabla f(\mathbf{x})^{\top}(\mathbf{y}-\mathbf{x})+\frac{\beta}{2}\|\mathbf{x}-\mathbf{y}\|^2$$

[0053] 设P是一个双随机矩阵。则有$\rho:=\left\|\mathbf{P}-\frac{\mathbf{1}\mathbf{1}^{\top}}{n}\right\|$，并且我们假设ρ＜1。

[0054] 基于上述的假设和定义，分布式OCO中每个玩家的目标都是最小化遗憾，该遗憾通过最优决策$\mathbf{x}^*\in\operatorname{argmin}_{\mathbf{x}\in\mathcal{K}}\sum_{t=1}^{T}F_t(\mathbf{x})$的全局损失进行衡量，具体定义为

[0055] $$R_T\left(\mathbf{x}_i,\mathbf{x}^*\right):=\sum_{t=1}^{T}\left(F_t(\mathbf{x}_i(t))-F_t(\mathbf{x}^*)\right)$$

[0056] 其中$\mathbf{x_i(t)}\in\mathcal{K}$。

[0057] 在分布式在线对偶平均(D-ODA)中，每个玩家$i\in\mathcal{V}$将持有一个决策变量$x_i(t)$和一个对偶变量$z_i(t)$来计算每一轮的新变量。对于任何玩家$i\in\mathcal{V}$，对偶变量$z_i(1)$的初始值总是为设置为0。在每一轮t＝[T]中，每个玩家$i\in\mathcal{V}$首先利用t-1轮计算得到的决策变量$x_i(t)$和由对手在本轮中给出的损失函数$f_{t,i}$计算其新的次梯度$g_i(t)$。在此之后，每个玩家$i\in\mathcal{V}$会收集其邻居的对偶变量$\{z_j(t),j\in N_i\}$来计算一个新的对偶变量$z_i(t+1)$，它被定义为

[0058] $$\mathbf{z}_i(t+1):=\sum_{j\in N_i}\mathbf{P}_{ij}\mathbf{z}_j(t)+\mathbf{g}_i(t)$$

[0059] 其中，

[0060] $$\sum_{j=1}^{n}P_{ij}=\sum_{i=1}^{n}P_{ij}\quad for\quad\forall i,j\in\mathcal{V}$$

[0061] P是一个用来模拟每对玩家之间通信情况的双随机矩阵。以玩家i为例，它将本地

信息$z_i(t)$直接发送给邻居，并接受来自邻居的信息。同时，$z_i(t)$可以通过玩家j与其邻居玩家j+1之间的通信进一步传播给与玩家i没有直接联系的玩家，如j+1。也就是说，只要两个玩家(节点)之间存在一条路径，例如玩家i和玩家j+1之间，那么这两个玩家的信息就可以被彼此获取到。此外，当网络中不存在孤立玩家时，每个玩家的信息在网络中都将是可获得的。换而言之，如果一个玩家是孤立的，那么其他玩家就无法获得他的信息。双随机矩阵P描述了不同玩家之间信息交互的难易程度。作为一个非负矩阵，P的元素只有当$(i,j)\in\varepsilon$或$(i,j_i),(j_i,j_i+1),\ldots,(j_i+n,j)\in\varepsilon$且$i\neq j$时，满足$P_{ij}>0$。矩阵的元素值越大，说明这对玩家越邻近，也意味着这对玩家的信息交互越容易，如图1网络中的$P_{i,j}>P_{i,j+1}>0$。基于新的对偶变量$z_i(t+1)$，新的决策变量$x_i(t+1)$也可以按照如下形式进行计算：

[0062] $$\mathbf{x}_i(t+1)=\Pi_{\mathcal{K}}^{\psi}(\mathbf{z}_i(t+1),\alpha_i(t))$$

[0063] 其中，

[0064] $$\Pi_{\mathcal{K}}^{\psi}(\mathbf{z},\alpha):=\underset{\mathbf{x}\in\mathcal{K}}{\operatorname{argmin}}\left\{\langle\mathbf{z},\mathbf{x}\rangle+\frac{1}{\alpha}\psi(\mathbf{x})\right\}$$

[0065] 在投影中，$\alpha_i(t)$是来自非递增序列的一个正步长，而$\psi(\mathbf{x}):\mathcal{K}\to\mathbb{R}$是一个近端函数。

[0066] 在每次迭代中，每个玩家都会执行如图1所示的步骤，具体为：

[0067] 第一步，查询玩家i所有邻居的对偶变量，并更新玩家i的对偶变量。

[0068] 获取对手给出的凸损失函数$f_{t,i},\forall i\in\mathcal{V}$，并根据凸损失函数$f_{t,i}$计算次梯度$\mathbf{g}_i(t)\in\partial f_{t,i}\left(\mathbf{x}_i(t)\right)$；收集其邻居的对偶变量$\{z_j(t),j\in N_i\}$，并结合次梯度$g_i(t)$来计算一个新的对偶变量$z_i(t+1)$。

[0069] 第二步，计算下一个动作，判断该动作指向的新决策是否在可行集$\mathcal{K}$中。

[0070] 根据对偶变量$z_i(t+1)$计算下一个动作v_i，并判断基于动作v_i的决策点$x_i(t)+v_i$是否在可行集$\mathcal{K}$中，若决策点$\mathbf{x}_i(t)+\mathbf{v}_i\in\mathcal{K}$，则将$x_i(t)+v_i$更新为新决策。

[0071] 第三步，当新的决策不在可行集$\mathcal{K}$中时，执行快速近似投影(FAstProj)步骤。

[0072] 如图2所示，可以找到第三步的可视化描述。当基于动作v_i的新决策$x_i(t)+v_i$不在可行集$\mathcal{K}$之内时，首先找到线段$(x_i(t),x_i(t)+v_i]$与$\mathcal{K}$边界$\partial\mathcal{K}$的交点$\tilde{\mathbf{x}}_i$，并计算$\tilde{\mathbf{x}}_i$的单位法向量n_i。然后，可以通过将决策点$x_i(t)+v_i$向单位法向量n_i的反方向投影到可行集$\mathcal{K}$上来实现近似投影。该方法的本质是求一个近似梯度，使最终决策能够落在$\mathcal{K}$中。这个近似过程会在近似梯度中引入噪声。因此，为了确保这个近似梯度是有效的，噪声的约束是必要的，如定理3所示。此外，D-FAP算法的更详细的描述如表1所示。

[0073] 对于一个闭凸集$\mathcal{K}$和$\mathbf{x}\in\partial\mathcal{K}$，$n_x$是$\mathcal{K}$在x处的法向量，如果

[0074] $$\langle\mathbf{n}_{\mathbf{x}},\mathbf{y}-\mathbf{x}\rangle\leq 0;\quad\forall\mathbf{y}\in\mathcal{K}$$

[0075] 对于β_h-smooth凸紧集$\mathcal{K}$和$\mathbf{x}\in\partial\mathcal{K}$，$\mathcal{K}$在x处的局部曲率(LC)可以定义为，

[0076] $$\mu_{\mathbf{x}}:=\frac{\beta_h}{\|\nabla h(\mathbf{x})\|}$$

[0077] $\mathcal{K}$在x处的全局曲率(GC)可以定义为，

[0078] $\mu \coloneqq \max_{\mathbf{x}\in\partial\mathcal{K}} \mu_{\mathbf{x}}$

[0079] FAstProj过程的目标是找到一个属于$\mathcal{K}$的近似投影点。当$\mathbf{x}\in\mathcal{K},\mathbf{v}\in\mathbb{R}^d$已知时，可以通过FAstProj找到$\widetilde{\Pi}_{\mathcal{K}}(\mathbf{x}+\mathbf{v})\in\mathcal{K}$。具体为：

[0080] 如果$(\mathbf{x}+\mathbf{v})\in\mathcal{K}$，毫无疑问我们可以直接获得$\widetilde{\Pi}_{\mathcal{K}}(\mathbf{x}+\mathbf{v})=(\mathbf{x}+\mathbf{v})$。但是，如果$(\mathbf{x}+\mathbf{v})\notin\mathcal{K}$，那么就会执行FAstProj过程。在这种情况下，首先找到线段(x,x+v]和$\mathcal{K}$的边界$\partial\mathcal{K}$的交点$\tilde{\mathbf{x}}$。随后定义向量$\tilde{\mathbf{v}}\coloneqq\mathbf{x}+\mathbf{v}-\tilde{\mathbf{x}}$，并计算单位向量$\|\mathbf{n}_{\tilde{\mathbf{x}}}\|=\frac{\nabla h(\tilde{\mathbf{x}})}{\|\nabla h(\tilde{\mathbf{x}})\|}$，如此可以按下式计算$\hat{\mathbf{v}}$投影到$\|\mathbf{n}_{\tilde{\mathbf{x}}}\|$上的向量$\tilde{\mathbf{v}}$，

[0081] $\hat{\mathbf{v}}=\tilde{\mathbf{v}}-(\|\mathbf{n}_{\tilde{\mathbf{x}}}\|\cdot\tilde{\mathbf{v}})\|\mathbf{n}_{\tilde{\mathbf{x}}}\|$

[0082] 最后，使用二分法找到能使射线$\tilde{\mathbf{x}}+\hat{\mathbf{v}}-\alpha\|\mathbf{n}_{\tilde{\mathbf{x}}}\|$:α≥0与$\mathcal{K}$的$\partial\mathcal{K}$相交的最小步长α。因此，这个交点是最终的近似投影点，即$\widetilde{\Pi}_{\mathcal{K}}(\mathbf{x}+\mathbf{v})=\tilde{\mathbf{x}}+\hat{\mathbf{v}}-\alpha\|\mathbf{n}_{\tilde{\mathbf{x}}}\|$。

[0083] 图2中，在可行集$\mathcal{K}$的边界上找到一个临近新行动点$x_i(t)+v_i$的点$\tilde{\mathbf{x}}_i$，并计算该点的法向量$\mathbf{n}_{\tilde{\mathbf{x}}_i}$作为新行动的投影方向。毫无疑问，这种近似投影的过程会产生噪声。然而，如果可行集不是尖的，则噪声会很小。众所周知，当范围足够小时，可以把一个曲面看成一个平面，把一条曲线看成一条直线。因此，可以联合可行集的全局曲率μ和行动(模型)的更新步长$\alpha_i(t)$来控制噪声，即当曲率较大时，进行小步长更新。

[0084] 然而，在分布式场景下，这些噪声存在扩散的风险。如果在每个玩家的本地进行近似投影，那么每个玩家做出的决策都会包含额外的噪声。除了影响局部决策外，由于玩家之间的通信，局部噪声也会流入网络并影响其他玩家的决策。因此，有必要更多的关注控制噪声在分布式网络中的影响。在DOL中，由于对偶变量是对每个玩家的历史梯度信息的累积，因此这种由近似投影产生的噪声会在对偶变量中进行累积。在进行下一轮决策的过程中，存在着对历史信息的好处和噪音累积的影响之间的权衡。为了尽量减少累积噪声的影响，本方法不直接使用对偶变量进行下一轮决策，而是利用新一轮和上一轮的对偶变量的差值来寻找下一个决策，具体如表1中第8行所示。

[0085] 表1 D-FAP算法

算法 1 D-FAP: Distributed Fast Approximate Projection

[0086] 1: **输入：** 可行集$\mathcal{K}$，步长$\{\alpha_i(t)\}_{t=1}^{T},\forall i\in\mathcal{V}$

2: **初始化：** $\mathbf{x}_i(1)\in\mathcal{K},\mathbf{z}_i(1)=\mathbf{0},\forall i\in\mathcal{V}$

3: **对** $t \in [T]$ **执行**

4: 对手给出 $f_{t,i}, \forall i \in \mathcal{V}$

5: 计算次梯度 $\mathbf{g}_i(t) \in \partial f_{t,i}\left(\mathbf{x}_i(t)\right), \forall i \in \mathcal{V}$

6: **对**每个玩家 $i \in \mathcal{V}$ **执行**

7: $\mathbf{z}_i(t+1) == \sum_{i \in N_i} \mathbf{P}_{ij}\mathbf{z}_j(t) + \mathbf{g}_i(t)$

8: $\mathbf{v}_i = -\alpha_i(t) \cdot (\mathbf{z}_i(t+1) - \mathbf{z}_i(t))$

9: **如果** $\mathbf{x}_i(t) + \mathbf{v}_i \in \mathcal{K}$ **那么**

[0087] 10: $\mathbf{x}_i(t+1) = \tilde{\Pi}_{\mathcal{K}}\left(\mathbf{x}_i(t) + \mathbf{v}_i\right) = \mathbf{x}_i(t) + \mathbf{v}_i$

11: **否则**

12: 找到 $\tilde{\mathbf{x}}_i \in (\mathbf{x}_i(t), \mathbf{x}_i(t) + \mathbf{v}_i] \cap \partial\mathcal{K}$，并使 $\tilde{\mathbf{v}}_i \leftarrow \mathbf{x}_i(t) + \mathbf{v}_i - \tilde{\mathbf{x}}_i$

13: 计算 $\mathbf{n}_i = \nabla h(\tilde{\mathbf{x}}_i) \Big/ \left\|\nabla h(\tilde{\mathbf{x}}_i)\right\|, \nabla h(\tilde{\mathbf{x}}_i) \in \partial h(\tilde{\mathbf{x}}_i)$

14: 使 $\hat{\mathbf{v}}_i \leftarrow \tilde{\mathbf{v}}_i - \left\langle \mathbf{n}_i, \tilde{\mathbf{v}}_i \right\rangle \cdot \mathbf{n}_i$

15: 计算投影步长 η_i：

16: $\eta_i \in \operatorname{argmin}\left\{\eta_i > 0 : \tilde{\mathbf{x}}_i + \hat{\mathbf{v}}_i - \eta_i \mathbf{n}_i \in \mathcal{K}\right\}$

17: 执行本地更新 $\mathbf{x}_i(t+1) = \tilde{\Pi}_{\mathcal{K}}(\mathbf{x}_i(t) + \mathbf{v}_i) = \tilde{\mathbf{x}}_i + \hat{\mathbf{v}}_i - \eta_i \mathbf{n}_i$

[0088] 可行集$\mathcal{K}$是光滑凸的，任何损失函数$f_{t,i}$是凸函数，并且具有G-Lipschitz梯度，这意味着任意次梯度$\mathbf{g}_i(t) \in \partial f_{t,i}$满足$||g_i(t)|| \leq G$。在假设1中关于ρ的假设是分布式设置中的一个基本假设。双随机矩阵的最大特征值是1。1-ρ总是用来衡量网络的连通性，ρ越小，意味着网络的连通性越好。

[0089] 下面分析D-FAP的遗憾。

[0090] 定理1.在假设1下，算法1中取步长值为$\alpha_i(t) > 0$，能得到，

$$\sup_{\{f_{t,i}\}_{t=1}^{T} \in \mathcal{F}^T} R_T(\mathbf{x}_i, \mathbf{x}^*)$$

[0091]
$$\leq \left(\frac{D^2}{2\alpha_i(1)} + 2\alpha_i(1)(\mu D + 1)\left(\frac{2G}{1-\rho}\right)^2 \right) n\sqrt{T} + \frac{3n}{2}\mu^2\alpha_i(1)^3\left(\frac{2}{1-\rho}\right)^4 G^4$$

[0092] 通过选择适当的学习率$\alpha_i(t)$，可得到如下的次线性遗憾，

[0093] 推论1.1.常数C表示，

[0094]
$$C = \frac{3nG\mu^2 D^3}{8(1-\rho)(\mu D + 1)^{\frac{3}{2}}}$$

[0095] 在假设1下，算法1中取步长值为$\alpha_i(t) = \frac{D(1-\rho)}{4G\sqrt{(\mu D + 1)t}}$，能得到，

[0096] $$\sup_{\{f_{t,i}\}_{t=1}^{T}\in\mathcal{F}^{T}} R_T(\mathbf{x}_i,\mathbf{x}^*)\lesssim\frac{4\sqrt{\mu D+1}}{1-\rho}nDG\sqrt{T}+C$$

[0097] 定理2.假设假设1成立。那么遗憾下界是，

[0098] $$\inf_{A\in\mathcal{A}}\sup_{\{f_{t,i}\}_{t=1}^{T}\in\mathcal{F}^{T}} R_T^A\left(\mathbf{x}_i,\mathbf{x}^*\right)\gtrsim n\sqrt{T}$$

[0099] 其中$\mathcal{A}$是是所有可能的学习算法的集合。$f_{t,i}$是损失函数，$\forall t\in[T]$有$\{f_{t,i}\}_{t=1}^{T}\in\mathcal{F}^{T}$。

[0100] 定理2中的下界在文献(M.Akbari,B.Gharesifard,and T.Linder,“Distributed online convex optimization on time-varying directed graphs,”IEEE Transactions on Control ofNetwork Systems,vol.4,no.3,pp.417-428,2015.)中已经得到了证明。由于推论1.1中的上界与下界匹配，可以说D-FAP对于DOL中的遗憾是最优的。

[0101] 为了简化讨论，把D,G和1-ρ当作常数。那么遗憾可以简化为$\mathcal{O}\left(nGD\sqrt{(\mu D+1)T}\right)$。接下来将从以下几个方面对遗憾进行详细的讨论。

[0102] (紧密性。)为了了解紧密性，考虑几个特殊场景：

[0103] -(n＝1。)在这一条件下，D-FAP就退化成为了集中式在线学习设置下的FAstProj。遗憾$\mathcal{O}\left(GD\sqrt{(\mu D+1)T}\right)$和文献(D.Mateos-Núnez and J.Cortés,“Distributed online convex optimization over jointly connected digraphs,”IEEE Transactions on Network Science and Engineering,vol.1,no.1,pp.23-37,2014.)中的遗憾是同阶的，且该遗憾被证明在近似投影设置下是最优的。这表明D-FAP的遗憾在对μ和T的依赖上是紧的。

[0104] -(μ→0。)与全投影OGD的最优遗憾界相比，近似投影会产生额外的μ。μ是决策空间的最大曲率。μ越大，可行集越尖，近似投影点携带的噪声越大。因此，如果μ可以无限接近于0，则遗憾值可以无限接近于最优水平$\mathcal{O}\left(n\sqrt{T}\right)$，它与n有紧密的依赖关系。

[0105] •(聚焦网络拓扑。)1-ρ用来衡量网络的连通性。一个连通性更好的网络会有更大的1-ρ值。从直观上看，一个连通性更好的网络可以帮助每个玩家获取更多有用的梯度信息，这样收敛速度就会更快。理论结果表明，1-ρ越大，后悔值越小，即网络的连通性越好。

[0106] •(聚焦步长$\alpha_i(t)$。)如推论1.1所示，作为一个重要参数，步长$\alpha_i(t)$和1-ρ是成正比的而与μ成反比。上面已经讨论了1-ρ是一个用来衡量网络连通性的指标。可以理解的是如果网络的连通性较好，那么我们就能采取大步长的更新。当1-ρ越大，网络的连通性越好，$\alpha_i(t)$也会越大。此外，在上文中还将μ作为可行集结构的描述进行了讨论。直观上，当可行集的结构不是很好时，即可行集是尖的，μ很大时，我们希望能以较小的步长进行更新，从而对噪声进行控制。这种与我们的结果是一致的，较大的μ会导致较小的$\alpha_i(t)$。

[0107] •(提升在光滑集上分布式设置下现有的遗憾。)如果把μ也视为常数，我们的分析将获得遗憾$\mathcal{O}\left(n\sqrt{T}\right)$。与现有文献的结果相比，D-FAP的遗憾是在光滑集上关于T最优的。

[0108] 定理1的简要证明：

[0109] 定理3.(噪声约束).对任意$i\in\mathcal{V}$和任意t∈[T],可以在$\mathbf{x}_i(t)\in\mathcal{K},\mathbf{v}_i\in\mathbb{R}^d$和$\|\mathbf{v}_i\|\leq\frac{1}{2\mu}$下通过D-FAP找到满足下列不等式的近似梯度$\tilde{\Pi}_{\mathcal{K}}(\mathbf{x}_i(t)+\mathbf{v}_i)-\mathbf{y}$,

[0110]
$$\begin{aligned}&\left\|\tilde{\Pi}_{\mathcal{K}}(\mathbf{x}_i(t)+\mathbf{v}_i)-\mathbf{y}\right\|\\&\leq\left\|(\mathbf{x}_i(t)+\mathbf{v}_i)-\mathbf{y}\right\|+\mu\left\|\mathbf{v}_i\right\|^2,\forall\mathbf{y}\in\mathcal{K}\end{aligned}$$

[0111] 引理1.对任意$i\in\mathcal{V}$和任意t∈[T],对偶变量$z_i(t)$和$z_i(t+1)$满足如下边界,

[0112]
$$\left\|\mathbf{z}_i(t+1)-\mathbf{z}_i(t)\right\|\leq\frac{2}{1-\rho}G$$

[0113] 我们对定理1的证明主要依赖于定理3和引理1。定理1中的不等式可以进一步表示为,

[0114]
$$\begin{aligned}&\left\|\tilde{\Pi}_{\mathcal{K}}(\mathbf{x}_i(t)-\alpha_i(t)(\mathbf{z}_i(t+1)-\mathbf{z}_i(t)))-\mathbf{y}\right\|\\&\leq\left\|(\mathbf{x}_i(t)-\alpha_i(t)(\mathbf{z}_i(t+1)-\mathbf{z}_i(t)))-\mathbf{y}\right\|+\mu\alpha_i(t)^2\left\|\mathbf{z}_i(t+1)-\mathbf{z}_i(t)\right\|^2\end{aligned}$$

[0115] 用引理1,$||x_i(t)-x||$和$\frac{2\alpha_i(t)G}{1-\rho}\leq\frac{1}{2\mu}$化简右项的平方。最后,能够得到如下不等式,

[0116]
$$\begin{aligned}&\left\|\mathbf{x}_i(t+1)-\mathbf{x}\right\|^2\\&=\left\|\tilde{\Pi}_{\mathcal{K}}\left(\mathbf{x}_i(t)-\alpha_i(t)\left(\mathbf{z}_i(t+1)-\mathbf{z}_i(t)\right)\right)-\mathbf{y}\right\|\\&\leq\left\|\mathbf{x}_i(t)-\mathbf{x}\right\|^2-2\alpha_i(t)(\mathbf{z}_i(t+1)-\mathbf{z}_i(t))(\mathbf{x}_i(t)-\mathbf{x})+\mu^2\alpha_i(t)^4G^4\left(\frac{2}{1-\rho}\right)^4+2(\mu D+1)\alpha_i(t)^2G^2\left(\frac{2}{1-\rho}\right)^2\end{aligned}$$

[0117] 从上面的不等式中,可以得到如下信息,

[0118]
$$\begin{aligned}&\left\langle\mathbf{g}_i(t),\mathbf{x}_i(t)-\mathbf{x}\right\rangle\\&\leq\left\langle\mathbf{z}_i(t+1)-\mathbf{z}_i(t),\mathbf{x}_i(t)-\mathbf{x}\right\rangle\\&\leq\frac{1}{2\alpha_i(t)}(\left\|\mathbf{x}_i(t+1)-\mathbf{x}\right\|^2-\left\|\mathbf{x}_i(t)-\mathbf{x}\right\|^2)+\frac{1}{2}\mu^2\alpha_i(t)^3\left(\frac{2G}{1-\rho}\right)^4+(\mu D+1)\alpha_i(t)\left(\frac{2G}{1-\rho}\right)^2\end{aligned}$$

[0119] 因为任意$f_{t,i}$是凸的,因此满足$f_{t,i}(\mathbf{x}_i(t))-f_{t,i}(\mathbf{x})\leq\left\langle\nabla f(\mathbf{x}_i(t)),\mathbf{x}_i(t)-\mathbf{x}\right\rangle$。代入这一不等式并对上述的不等式进行全轮次和全玩家的求和,然后我们能够得到如定理1所示的最终结果。

[0120] 实证研究

[0121] 此处将D-FAP运用到在线logistic回归下的二分类实践中。为了约束可行集为光滑凸集,设$h(\mathbf{x})=\frac{\beta_h}{2}\|\mathbf{x}\|^2-H,H>0$,H是常数。因此,对于任意$i\in\mathcal{V}$和任意t∈[T],给定一个实例$\mathbf{A}\in\mathbb{R}^d$和它的标签y∈{1,-1},损失函数为

[0122]
$$f_{t,i}(\mathbf{x}_i(t))=\log(1+\exp(-y_{t,i}\mathbf{A}_{t,i}^{\top}\mathbf{x}_i(t)))$$

[0123] 其中$\beta_h=10^{-3}$是一个给定的超参数,此外在实验中将H的值设为5。在此设置下,我

们比较了D-FAP和快速近似投影方法在集中式在线梯度(FAstProj-COG)中的性能。

[0124] 学习率$\alpha_i(t)$在每一个数据集的实验中都已经调为最优。通过计算平均损失：$\frac{1}{nT}\sum_{t=1}^{T}\sum_{i=1}^{n}f_{t,i}(\mathbf{x}_i(t))$来评估学习效果，而不直接使用遗憾$R_T(\mathbf{x}_i,\mathbf{x}^*)=\sum_{t=1}^{T}\sum_{i=1}^{n}\left(f_{t,i}(\mathbf{x}_i(t))-f_{t,i}(\mathbf{x}^*)\right)$。在相同条件下，D-FAP和FAstProj-COG的最优参考点x^*是相同的。我们在12个真实数据集上进行了实证评估。数据集中实例和特征数量的详细信息可以在表2中找到。

[0125] 表2实验数据集汇总

[0126]

数据集	# 实例数	# 特征数	数据集	# 实例数	# 特征数
HIGGS	11,000,000	28	*covtype*	581,012	54
skin-nonskin	245,057	3	*SUSY*	5,000,000	18
Adult	32,561	14	*bank*	45,210	16
cod-rna	59,539	8	*mushrooms*	8124	112
ijcnn	49,990	22	*room-occupancy*	8,143	6
spam	9,324	499	*phishing*	11,055	68

[0127] 这12个数据集都是在三个开源网站上开放的：LIBSVM，UCI，MIKD。值得注意的是，这些数据集来自不同的学科，包括物理学(大型数据集HIGGS和SUSY)，生物学(skin-nonskin和cod-rna)，经济学(bank和Adult)，植物学(covtype和mushrooms)，计算机科学(NLP数据集ijcnn，spam和phishing)，等等。保证了实验结果具有客观指导意义。对于所有这些数据集，每个实例的所有特征值都按照均值为0，方差为1进行了归一化。

[0128] 为了研究网络拓扑的影响，我们对四种不同的拓扑设置进行了实证评估。

[0129] (全连接网络)在该拓扑中，所有节点之间都是互联的，这样每个节点在每一轮中都能获得所有节点的梯度信息。

[0130] (Watts-Strogatz)这是由Watts&Strogatz提出的一种随机图生成技术。利用该技术，可以通过设置节点数n、网络平均度k和连接概率p来生成随机网络。为了简化用于生成网络的参数，在所有实验中设置$k=\lfloor n\cdot p\rfloor$。节点的数量n是根据数据集的大小进行选择的。我们对连接概率p进行了两种设置，即p＝0.5，p＝1，以考察D-FAP关于网络拓扑结构的敏感性。

[0131] (不连通网络)在这个设置中，所有节点之间都是断开连接的，这意味着每个节点只利用本地数据进行FAstProj-COG。

[0132] D-FAP的性能与FAstProj-COG相当。我们在每个数据集下模拟一个Watts-Strogatz随机网络。用于生成随机网络的连接概率保持在0.7，而节点的数量是基于数据集的大小进行选择的。在这些设置下，可以的到D-FAP和FAstProj-COG在所有数据集上进行在线学习任务都是有效的。此外，D-FAP比FAstProj-COG收敛速度稍慢。收敛速度慢是可以理解的，因为分布式场景下每个节点获得的梯度信息会比集中式场景少，因此也会需要更多的计算去达到收敛。

[0133] D-FAP的性能对网络规模具有鲁棒性。我们在每个数据集中设置不同的节点数量，

并使用Watts-Strogatz生成连接概率p＝0.7的网络。可以得到，在每个数据集中，不同节点规模的平均损失曲线大多是重叠的。这一结果表明，D-FAP对网络规模(或玩家数量)具有鲁棒性，即平均遗憾不会受到网络规模的影响不会受到网络大小的影响。

[0134] 网络连通性越好，D-FAP的收敛速度越快。我们采用了4种不同的拓扑设置：全连接网络，Watts-Strogatz(0.5)，Watts-Strogatz(1)和不连通网络。关于这四个设置的更多细节可以在网络拓扑中找到。结果表明全连接网络的性能最好，因为此时ρ＝0，而不连通网络的性能最差，因为ρ＝1。其他拓扑的性能介于全连接网络和不连通网络之间，因为它们的ρ满足0＜ρ＜1。

[0135] 依照本发明的实施例如上文所述，这些实施例并没有详尽叙述所有的细节，也不限制该发明仅为所述的具体实施例。根据以上描述，可作很多的修改和变化。本说明书选取并具体描述这些实施例，是为了更好地解释本发明的原理和实际应用，从而使所属技术领域技术人员能很好地利用本发明以及在本发明基础上的修改使用。本发明仅受权利要求书及其全部范围和等效物的限制。

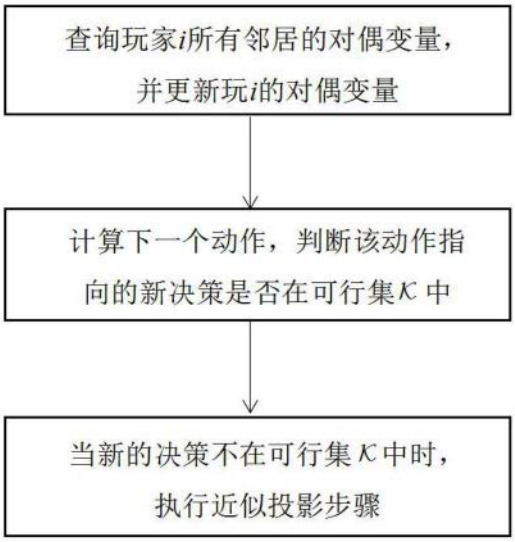
查询玩家i所有邻居的对偶变量，
并更新玩i的对偶变量
计算下一个动作，判断该动作指
向的新决策是否在可行集$\mathcal{K}$中
当新的决策不在可行集$\mathcal{K}$中时，
执行近似投影步骤

图1

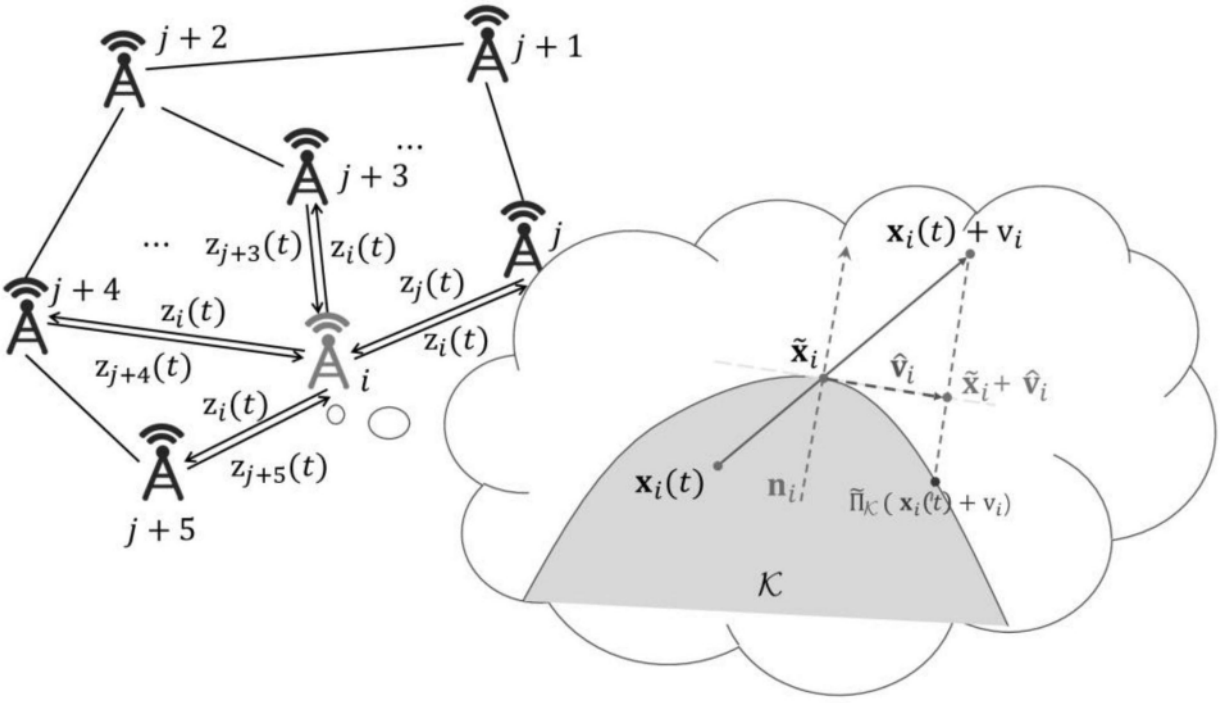
$j+2$
$j+1$
$j+3$
j
$j+4$
i
$j+5$
$z_{j+3}(t)$
$z_i(t)$
$z_j(t)$
$z_{j+4}(t)$
$z_{j+5}(t)$
$\mathbf{x}_i(t)+\mathbf{v}_i$
$\tilde{\mathbf{x}}_i$
$\hat{\mathbf{v}}_i$
$\tilde{\mathbf{x}}_i+\hat{\mathbf{v}}_i$
$\mathbf{x}_i(t)$
$\mathbf{n}_i$
$\tilde{\Pi}_{\mathcal{K}}(\mathbf{x}_i(t)+\mathbf{v}_i)$
$\mathcal{K}$

图2