43rd IEEE Conference on Decision and Control (submitted)

A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems

Emanuel Todorov and Weiwei Li

【强化学习算法 13】iLQG

张楚珩
清华大学 交叉信息院博士在读

i打头但并不是苹果家的产品哈，其全称是 Iterative Linear Quadratic Gaussian。

原文传送门：

Todorov, Emanuel, and Weiwei Li. "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems." American Control Conference, 2005. Proceedings of the 2005. IEEE, 2005.

特色：

确切的说这里要讲的不是一个强化学习算法，而是一个最优控制问题。其区别在于强化学习中不直接知道系统的dynamics，而最优控制的问题可以知道系统的dynamics。但是由于控制理论是强化理论的重要基础，相比于更为玄学的强化学习，它的理论分析更细致，了解一些控制论对于理解强化学习很有帮助。

这篇工作把一个非线性最优控制问题，在每次迭代中都在局部归化为控制理论里面研究很成熟的Linear Quadratic Gaussian（LQG）问题，然后迭代地去求解更好的控制序列，直到收敛。这篇工作里面的结果以现在的标准来看并不惊艳，但是展示的很仔细，对于强化学习过程中理解什么是价值函数、轨迹等都很有帮助。

分类：

不是强化学习算法、continuous state space、continuous action space、不仅仅是model-based而且需要知道model dynamics

背景：

LQG的最优控制是控制理论里面一个非常经典的问题，考虑一个线性时变系统，其动力学特性

$$x(k+1) = F(k)x(k) + G(k)u(k) \quad k \in [N-1]$$

控制目标是找到一个控制序列 $u(k)$ 最小化如下二次型性能指标函数

$$J = x^T(N)Q_0x(N) + \sum_{k=k_0}^{N-1}[x^T(k)Q_1(k)x(k) + u^T(k)Q_2(k)u(k)]$$

其中要求 Q_0, Q_1 非负定， Q_2 正定。

该问题的解决方式就是使用动态规划求解，依次写出 $J_N, J_{N-1}, \cdots$ ，把 $J(N-1)$ 用 $J(N)$ 表示，然后对于

控制量求导，然后依次得到最优的控制序列 $u(N-1), u(N-2), \cdots$ 。得到最优控制（Riccati方程）

$$\begin{cases} u(k) = -L(k)x(k) \\ L(k) = [Q_2(k) + G^T(k)S(k+1)G(k)]^{-1}[G^T(k)S(k+1)F(k)] \\ S(k) = [F(k) - G(k)L(k)]^T S(k+1)[F(k) - G(k)L(k)] + Q_1(k) + L^T(k)Q_2(k)L(k) \\ S(N) = Q_0 \end{cases}$$

相应的最优性能指标函数为 $J_{\min} = x^T(k_0)S(k_0)x(k_0)$ 。

若考虑一个定常系统，即 F, G, Q_1, Q_2 与时间无关，在求解该方程的时候可以发现 $S(k)$ 在远离 $k = N$ 的地方几乎为一个稳定的常数，并且当 N 很大的时候，不论 $S(N) = Q_0$ 取值如何该常数值都不变，因此有定常的控制规律

$$\begin{cases} u(k) = -Lx(k) \\ L = [Q_2 + G^T S G]^{-1}[G^T S F] \\ S = [F - GL]^T S[F - GL] + Q_1 + L^T Q_2 L \end{cases}$$

过程：

iLQG的**主要思路**就是先任意找一个控制序列 $\{\bar{u}(k)\}$ ，然后按照这个控制做一次rollout，得到轨迹 $\{\bar{x}(k)\}$ 。在这个轨迹附近，将系统的动力学特性线性化、将损失函数二次化，并考虑如何在 $\{\bar{u}(k)\}$ 和 $\{\bar{x}(k)\}$ 附近找到扰动 $\delta u = u - \bar{u}$ 和 $\delta x = x - \bar{x}$ 使得新的控制比之前的更好。迭代地做更新直到收敛即得到最优控制序列。

下面来具体地说各个步骤。

首先可以将系统的动力学特性线性化并且将损失函数二次化，有

$$\begin{aligned} \delta \mathbf{x}_{k+1} &= A_k \delta \mathbf{x}_k + B_k \delta \mathbf{u}_k + \mathcal{C}_k \left(\delta \mathbf{u}_k\right) \boldsymbol{\xi}_k \\ \mathcal{C}_k \left(\delta \mathbf{u}_k\right) &\triangleq \left[\mathbf{c}_{1,k} + C_{1,k} \delta \mathbf{u}_k \ \cdots \ \mathbf{c}_{p,k} + C_{p,k} \delta \mathbf{u}_k\right] \\ \operatorname{cost}_k &= q_k + \delta \mathbf{x}_k^{\mathsf{T}} \mathbf{q}_k + \frac{1}{2} \delta \mathbf{x}_k^{\mathsf{T}} Q_k \delta \mathbf{x}_k \\ &\quad + \delta \mathbf{u}_k^{\mathsf{T}} \mathbf{r}_k + \frac{1}{2} \delta \mathbf{u}_k^{\mathsf{T}} R_k \delta \mathbf{u}_k \end{aligned}$$

其中 A_k 和 B_k 可以由已知的动力学规律在轨迹 $\{\bar{x}(k)\}$ 附近求导得到，各种Q和R都可以对于损失函数在现有轨迹附近求导得到， C_k 是由上述列向量拼成的，反映的是控制引起的噪声。即，认为各种A、B、C、Q、R已知。

假设有state value function写成如下形式

$$v_k \left(\delta \mathbf{x}\right) = s_k + \delta \mathbf{x}^{\mathsf{T}} \mathbf{s}_k + \tfrac{1}{2} \delta \mathbf{x}^{\mathsf{T}} S_k \delta \mathbf{x}$$

应用Bellman方程

$$v_k \left(\delta \mathbf{x}\right) = \text{immediate cost} + \mathrm{E}\left[v_{k+1} \left(\text{next state}\right)\right]$$

可以求得到

$$
\begin{aligned}
v_k(\delta\mathbf{x}) &= q_k + s_{k+1} + \tfrac{1}{2}\textstyle\sum_i \mathbf{c}_i^{\mathsf{T}} S_{k+1}\mathbf{c}_i \\
&\quad + \delta\mathbf{x}^{\mathsf{T}}\left(\mathbf{q}_k + A_k^{\mathsf{T}}\mathbf{s}_{k+1}\right) \\
&\quad + \tfrac{1}{2}\delta\mathbf{x}^{\mathsf{T}}\left(Q_k + A_k^{\mathsf{T}} S_{k+1} A_k\right)\delta\mathbf{x} \\
&\quad + \boldsymbol{\pi}^{\mathsf{T}}\left(\mathbf{g} + G\delta\mathbf{x}\right) + \tfrac{1}{2}\boldsymbol{\pi}^{\mathsf{T}} H\boldsymbol{\pi}
\end{aligned}
$$

假设具有形如 $\delta u = \pi_k(\delta x) = l_k + L_k \delta x$ 的闭环控制形式，可以得到最优控制（具体讨论见附注）

$$
\mathbf{l}_k = -H^{-1}\mathbf{g},\ L_k = -H^{-1}G
$$

并且可以求得到state value function

$$
\begin{aligned}
v_k(\delta\mathbf{x}) &= q_k + s_{k+1} + \tfrac{1}{2}\textstyle\sum_i \mathbf{c}_i^{\mathsf{T}} S_{k+1}\mathbf{c}_i \\
&\quad + \delta\mathbf{x}^{\mathsf{T}}\left(\mathbf{q}_k + A_k^{\mathsf{T}}\mathbf{s}_{k+1}\right) \\
&\quad + \tfrac{1}{2}\delta\mathbf{x}^{\mathsf{T}}\left(Q_k + A_k^{\mathsf{T}} S_{k+1} A_k\right)\delta\mathbf{x} \\
&\quad + \mathbf{l}_k^{\mathsf{T}}\mathbf{g} + \tfrac{1}{2}\mathbf{l}_k^{\mathsf{T}} H\mathbf{l}_k + \delta\mathbf{x}^{\mathsf{T}}\left(G^{\mathsf{T}}\mathbf{l}_k + L_k^{\mathsf{T}}\mathbf{g} + L_k^{\mathsf{T}} H\mathbf{l}_k\right) \\
&\quad + \tfrac{1}{2}\delta\mathbf{x}^{\mathsf{T}}\left(L_k^{\mathsf{T}} H L_k + L_k^{\mathsf{T}} G + G^{\mathsf{T}} L_k\right)\delta\mathbf{x}
\end{aligned}
$$

进行比对可以得到

$$
\begin{aligned}
S_k &= Q_k + A_k^{\mathsf{T}} S_{k+1} A_k + L_k^{\mathsf{T}} H L_k + L_k^{\mathsf{T}} G + G^{\mathsf{T}} L_k \\
\mathbf{s}_k &= \mathbf{q}_k + A_k^{\mathsf{T}}\mathbf{s}_{k+1} + L_k^{\mathsf{T}} H\mathbf{l}_k + L_k^{\mathsf{T}}\mathbf{g} + G^{\mathsf{T}}\mathbf{l}_k \\
s_k &= q_k + s_{k+1} + \tfrac{1}{2}\textstyle\sum_i \mathbf{c}_{i,k}^{\mathsf{T}} S_{k+1}\mathbf{c}_{i,k} + \tfrac{1}{2}\mathbf{l}_k^{\mathsf{T}} H\mathbf{l}_k + \mathbf{l}_k^{\mathsf{T}}\mathbf{g}
\end{aligned}
$$

其中

$$
\begin{aligned}
\mathbf{g} &\triangleq \mathbf{r}_k + B_k^{\mathsf{T}}\mathbf{s}_{k+1} + \textstyle\sum_i C_{i,k}^{\mathsf{T}} S_{k+1}\mathbf{c}_{i,k} \\
G &\triangleq B_k^{\mathsf{T}} S_{k+1} A_k \\
H &\triangleq R_k + B_k^{\mathsf{T}} S_{k+1} B_k + \textstyle\sum_i C_{i,k}^{\mathsf{T}} S_{k+1} C_{i,k}
\end{aligned}
$$

相应算法的每一次迭代都在轨迹 $\{\bar{x}(k), \bar{u}(k)\}$ 附近先找到控制序列 $u = \bar{u} + \delta u$ ，然后根据动力学规律找到新的轨迹 $x = \bar{x} + \delta x$ ，反复迭代直到收敛。

算法：

找到一条初始的轨迹 $\{\bar{x}(k), \bar{u}(k)\}$ ，然后反复进行如下迭代：

1. 做动态规划找到更好的控制序列：按照 $k = K, K-1, \cdots, 1$ 的顺序迭代，计算新的控制规律参数 $\delta u_k = I_k + L_k \delta x_k$ 和 $v_k(\delta x) = s_k + s_k^T \delta x + \frac{1}{2} \delta x^T S_k \delta x$ ；
2. 更新新的轨迹：按照 $k = 1, 2, \cdots, K$ 的顺序迭代，根据上步计算到的控制规律和 $\delta x_{k+1} = A_k \delta x_k + B_k \delta u_k$ 计算新的轨迹；

另外还有一些有意思的点和比较繁琐的讨论放在附录中供参考。

---

iLQG 与强化学习里面的 Policy Gradient 方法有什么区别和联系？

首先，这里的设定是已知环境的 dynamics，但是一般强化学习的设定是不知道环境的 dynamics。在知道环境 dynamics 的情况下，如果有了控制的扰动 δu 之后，就能直接算出状态空间的扰动 δx 了，因此这里的算法每一次迭代都会直接通过计算状态空间的扰动得到新的状态空间轨迹。而在强化学习的设定下，一般会用新的控制序列重新做 rollout 得到新的状态空间轨迹。

其次，两者对于控制的扰动 δu 的计算方式不同。state value function $v_x(\delta x, \delta u)$ 中关于控制扰动的项提出来可以写作 $a(\delta u) = \delta u^T (g + G \delta x) + \frac{1}{2} \delta u^T H \delta u$ 。做如下近似1）**认为控制的扰动是开环的**，即和 δx 无关；2）**认为扰动是微小的**，即可以忽略二阶项 $\frac{1}{2} \delta u^T H \delta u$ ，并且对于一个小的 ϵ ，有 $\delta u_k = I_k = -\epsilon g_k$ ；3）观察到 $g_k = r_k + B_k^T s_{k+1} = \frac{\partial J}{\partial u_k} + \frac{\partial f}{\partial u_k} (\frac{\partial J}{\partial x_{k+1}} + (I + \frac{\partial f}{\partial x_k})^T s_{k+2} + G^T I_{k+1})$ ，如果**假定每一步 k 上的扰动都不影响前后的状态空间轨迹**，那么上式后一项就没有了。通过上述三个假设，可以自然得到 policy gradient 的公式 $\delta u_k = -\epsilon \frac{\partial J}{\partial u_k}$ 。

通过上面的计算我们可以发现，梯度下降方法可以看做是做了以上近似之后 iLQG 的特殊情况。

上述公式里面的 H 是否能够保证正定？如果不正定应该怎么办？

一个物理上可行的系统，求出来的 H 肯定是正定的，原因如下。注意到 $a(\delta u) = \delta u^T (g + G \delta x) + \frac{1}{2} \delta u^T H \delta u$ ，如果 H 有负特征值，那么存在控制扰动使得目标函数无限小。但是实际的数值计算中肯定存在负数或者接近零的特征值，那么我们考虑找一个正定的 $\mathcal{H}$ 作为它的替代。

1. 一个方案是使用 Levenberg-Marquardt trick，即 $\mathcal{H} = H + (\epsilon - \lambda_{\min}) I$ 。可以理解为把原矩阵的特征值都加上一个数值使得其为正定。由于 $update \propto \mathcal{H}^{-1}(\cdot)$ ，每个特征值都变大相对于更新来说就更保守。
2. 另一个方案是先做特征值分解 $H = V D V^T$ ，将 D 矩阵中特征值小于零的元素都设置为 ϵ ，然后形成新的替代矩阵。注意到这样做并不会增加更多的运算量，因为本身我们就要对矩阵求逆，这样逆可以直接算出来 $\mathcal{H}^{-1} = V D^{-1} V^T$ 。

如果对于action space 有 constraint，即有约束 $u(t) \in U$ ，应该怎么办？

首先每次迭代中原本的控制肯定是满足约束的，考虑 $\delta u = I_k + L_k \delta x$ 。首先不管 δx 是多少，开环控制部分我们希望没有超过约束边界；如果 I_k 部分穿过了约束边界，那么可以把它收缩成 ϵI_k ， $0 \le \epsilon \le 1$ 可以通过 backtrack 得到。而第二部分如果穿过了约束边界，我们可以直接把这一项设置为零。当

约束是对于控制的每一位独立约束的时候，可以不这么保守，即只把超过的那一位设置为零。

ps. 如果遇到公式显示有问题，再次刷新，有一定几率该公式能正常显示，同时也有一定几率使得其他公式显示不出来，请多刷几次= =

编辑于 2018-10-01

算法 机器学习 强化学习 (Reinforcement Learning)

文章被以下专栏收录

强化学习前沿
读呀读paper

进入专栏