EKF

Расширенный фильтр Калмана для обучения весов сети в общем виде представлен рекурсивной системой уравнений:

$$\begin{aligned} A_k &= [R_k + H_k^T P_k H_k]^{-1}, \\ K_k &= P_k H_k A_k, \\ \hat{w}_{k+1} &= \hat{w}_k + K_k \xi_k, \\ P_{k+1} &= P_k - K_k H_k^T P_k + Q_k. \end{aligned}$$

Однако в нашем случае эта система несколько видоизменена, и на самом деле мы работаем со след. системой:

$$\begin{aligned} P_л^- &= P_k + Q_k. \\ S_k &= [R_k + H_k^T P_k H_k]^{-1}, \\ K_k &= P_k H_k S_k, \\ \hat{w}_{k+1} &= \hat{w}_k + K_k \xi_k, \\ P_{k+1} &= P_k - (I - K_k H_k) P_k (I - K_k H_k)^T + K_k R_k K_k^T. \end{aligned}$$

Варьируемые параметры

Основные параметры, которые управляют вычислениями:

$P_k = \epsilon^{-1} I, \qquad (\epsilon = \text{от } 0.01 \text{ до } 0.001)$

$R_k = \eta^{-1} I, \qquad (\eta \text{ от } 0.001 \text{ до } 1)$

$Q_k = qI, \qquad (q \text{ от } -\infty \text{до } 0.1)$

Описание функций

Для EKF разработаны 3 метода вычислений, соотв. реализуются они 3-мя различными функциями:

1. `trainekf` – итеративная функция (по существу ограничена кол-вом эпох), принципиальное отличие от адаптивной версии состоит в том, что на каждом шаге рассчитывает погрешность и якобиана для всего набора обучающих примеров. Запускается через функцию `train`.
2. `adaptekf` –прямая функция (выполняется столько раз, сколько у нас образцов). Рассчитывает на каждом шаге якобиан и вектор ошибок на основе одного обучающего примера. Запускается через функцию `adapt_2`[1].
3. `adaptekf_ms` – примитивная реализация многопоточной версии EKF. Последовательно выполняет действия, аналогичные `adaptekf`, за исключением этапа обновления, который осуществляется через глобальную (собранную на основе расчетов всех потоков) матрицы S_k (A_k). Основная идея в обучении Ns копий нейросети на разных обучающих наборах с координированным обновлением весов. В итоге получаем не один, а Ns векторов весов матрицы. Запускается через функцию `adapt_ms`[1].

1 – Замена функций `adapt` мотивирована необходимостью передавать в вычисляющую функцию массива входных данных X для расчета Якобиана.

UKF

Нечувствительный фильтр Калмана в нашем случае представлен в классическом виде, без дополнительных ухищрений:

$$P_{w_k}^- = P_{w_{k-1}} + R_{k-1}^r,$$

$$\hat{w}_k^- = \hat{w}_{k-1},$$

$$W_{k|k-1} = \left[\hat{w}_k^- \quad \hat{w}_k^- + \gamma\sqrt{P_{wk}^-} \quad \hat{w}_k^- - \gamma\sqrt{P_{wk}^-}\right],$$

$$D_{k|k-1} = G\left(x_k, W_{k|k-1}\right),$$

$$\hat{d}_k = \sum_{i=1}^{2L} W_i^{(m)} D_{i,k|k-1},$$

$$P_{d_k d_k} = \sum_{i=1}^{2L} W_i^{(c)} (D_{i,k|k-1} - \hat{d}_k)(D_{i,k|k-1} - \hat{d}_k)^T + R_k^e,$$

$$P_{w_k d_k} = \sum_{i=1}^{2L} W_i^{(c)} (W_{i,k|k-1} - \hat{w}_k^-)(D_{i,k|k-1} - \hat{d}_k)^T + R_k^e,$$

$$K_k = P_{w_k d_k} P_{d_k d_k}^{-1},$$

$$\hat{w}_k = \hat{w}_k^- + K_k\left(d_k - \hat{d}_k\right),$$

$$P_{w_k} = P_{w_k}^- - K_k P_{d_k d_k} K_k^T,$$

$$W_0^{(m)} = \frac{\lambda}{L+\lambda},$$

$$W_0^{(c)} = \frac{\lambda}{L+\lambda} + 1 - \alpha^2 + \beta,$$

$$W_i^{(m)} = W_i^{(c)} = \frac{1}{2(L+\lambda)}, \quad i = 1, \ldots\ldots, 2L.$$

где $\gamma = \sqrt{L+\lambda}, \quad \lambda = (L+k) - L,$

$R^r, R^e -$ матрицы ковариации шума (процесса и наблюдений соотвт.)

Варьируемые параметры

$P_k = \epsilon^{-1} I,$	α (от 1 до 0.0001)	– усиливает распространение $W_{i,k\|k-1}$ вокруг x_k
$R_k = \eta^{-1} I,$	β (2)	– усиливает влияние начальных значений
$Q_k = qI,$	$k(L-3)$	– масштабирующий параметр.

Описание функций

`trainukf` – функция вычисления UKF. Выходы сети рассчитывает через `nn7.y_all`. Запускается через функцию `adapt_2`.