事例紹介

ロボット技術のセンシングデバイスへの展開

Sensing Devices applied from an Art of Robotics

千田 陽介^{*1} 伊東 利雄^{*2} ^{*1}久留米工業大学 ^{*2}(株) 富士通研究所
 Yosuke Senta^{*1} and Toshio Ito^{*2} ^{*1}Kurume Institute of Technology ^{*2}Fujitsu Laboratories LTD.

1. はじめに

富士通は2016年1月、図1のようなIoT用コアモジュールおよびこのモジュールを用いたユビキタスウェア製品群を発売した。このモジュール（以下IoTモジュール）は12×16×1.7[mm], 0.55[g]の中に、加速度、角速度、地磁気、気圧等のセンサと、歩数や活動量、各種動作の計測・検出、PDR（Pedestrian Dead Reckoning: 歩行者デッドレコニング）等のアルゴリズムが入ったマイコンを搭載したものである[1]。

このような電池駆動小型デバイスを長時間動かすためには、消費電力を小さくしなければならない。すなわちあまり性能の低い非力なマイコンを用いる必要がある。そのようなマイコンでも動作する負荷の小さいアルゴリズムを作るにあたって、小型ヒューマノイドロボットHOAP[2]のシステム記述として開発した双線形時間遅れニューラルネットワークの考え方が非常に役立った。本稿では双線形時間遅れニューラルネットワークについて改めて解説し、その後IoTコアモジュールへの適用例を述べていく。

2. 双線形型時間遅れニューラルネットワーク

双線形時間遅れニューラルネットワーク（RNN: Recurrent Neural Networks）は2002年頃に永嶋によって提唱されたロボットシステム記述法である。ここではRNNの考え方を簡単に紹介するが、詳細は文献[3][4]などを参照されたい。図2は歩行の安定性を確保するためHOAPロボットの中に組み込んだフィードバック状態方程式：

$$\frac{dx}{dt} = \begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix} x + \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} u, y = (c_0 \ c_1) x \quad (1)$$

図1 IoT コアモジュールと製品群

図2 フィードバック回路[5]

図3 RNNの要素

のRNN回路である[5]。RNNは図3に示すa, b, c三つのタイプの要素によって構成されている。なお過去の発表では、これらを一つの要素（ニューロン）としていたが本稿では便宜上別の要素として扱う。回路上の要素番号を*i*とすると、aタイプは

$$x_i = \sum_j a_{[ij]} x_j + \sum_{k \leq l} b_{[i]kl} x_k x_l \quad (2)$$

という代数演算、bタイプは

$$\varepsilon_i \frac{dx_i}{dt} + x_i = a_{[ij]} x_j \quad (3)$$

という一次遅れ（初期値 I_i ）といった動作を行う。一方c

原稿受付 2016年10月27日

キーワード: IoT Device, Pedometer, Mobile/Smart Phone, Motion Sensing, Recurrent Neural Networks

^{*1}〒830-0052 久留米市上津町 2228-66

^{*2}〒211-8588 川崎市中原区上小田中 4-4-1

^{*1}Kurume-shi, Fukuoka

^{*2}Nakahara-ku, Kawasaki-shi, Kanagawa

タイプの働きは

$$x_i = \begin{cases} 1 & (a_{[ij]x_j} > 0) \\ 0 & (a_{[ij]x_j} \leq 0) \end{cases} \quad (4)$$

とうスイッチング動作である．ここで， $\varepsilon_i (\neq 0)$ および $a_{[ij]}$, $b_{[ijkl]}$, I_i は適当な定数である．また回路表記にあたって $a_{[ij]}$ および $b_{[ijkl]}$ が 0 の場合は結線自体を，1 の場合は数値の記述を省略することとする．

a, b 二つのタイプを組み合わせた RNN 回路は，線形結合と双一次結合を用いた連立微分方程式を視覚的に表現したものと言い換えることができよう．この形の連立微分方程式は非常に表現力が豊かなため，センシング，状態認識，運動生成などロボットシステムに必要な処理そのもの，もしくは近似表現を記述できる．ここでいう近似とは，テーラ展開と似たイメージのものである．テーラ展開では式の次数を上げれば上げるほど広範囲にわたって高い精度が得られるが，その分式が複雑になる．そこで領域をいくつかに分け，各領域ごとに次数の低い近似式を作り，解析接続によって解を切り替える方法が取られる．連立微分方程式 (RNN 回路) も同様で，c のスイッチ要素はその解析接続を実現するためのものである．

HOAP 以降，RNN の考え方は携帯電話等の組み込み機器に取り入れられるようになった．このような機器は言うまでもなくデジタルコンピュータで実装され，かつその計算能力は省電力のため高くない．そのため RNN をデジタルコンピュータ向けに最適化しなくてはならない．式 (2)～(4) のうちデジタル化に向け一番扱いが厄介なのは微分項のある式 (3) であろう．この微分項をサンプリング間隔 ΔT を用いた差分法で表すと

$$\varepsilon \frac{x_i^+ - x_i^-}{\Delta T} + x_i^- = x_j^- \quad (5)$$

と変形できる．ここで上付きの $+$, $-$ はサンプリング前後を表す．さらにこれを変形すると

$$x_i^+ = (1 - k)x_i^- + kx_j^- \quad (6)$$

という漸化式になる ($k = \Delta T/\varepsilon$)．この式は単純な一次の差分法を用いて求めたため，十分な精度が得られないように思えるかもしれない．しかし精度を必要とする箇所ではより高次の差分法を用いて実装すればよいだけであり，本

図 4 遅れニューロンのデジタル表記

質的な問題とはならない．

図 4(a) は式 (6) をブロック線図で表したものである．図のように遅れ要素 Z^{-1} と定数倍と和の演算の組み合わせで実現できている．これより従来これ以上分割できない一つの要素であった一次遅れは，デジタル化された RNN (D-RNN) では複数の要素が組み合わせられたものとも考えることもできる．そこで D-RNN では一次遅れ要素の代わりに遅れ要素 (四角で表現，内部の数値は初期値) を基本要素として導入することとする．これにより一次遅れ要素 (図 3(b)) は，図 4(b) のような回路で表現できる．

ここまでの論点をまとめると以下ようになる．

- (1) RNN はロボットシステム記述を目的に開発された
- (2) RNN はシステムが要求する精度を満たすのに必要な最低限の近似を容易に記述できる
- (3) RNN を実装するために必要な要素は，遅れ，和，積，定数倍および三項演算子による演算のみであり，いずれもデジタルコンピュータが得意とする処理である
- (4) (2), (3) より RNN は少ない計算機資源で実装できる

3. IoT デバイスにおける RNN 実装例

3.1 歩数計

この章では RNN の考えを用いた歩数計の実装例を紹介する．歩数計のアルゴリズムは携帯電話内の人用歩数計 [6] として開発され，以後スマートフォン (Smart Phone: SP)，犬用歩数計 [7]，そして IoT モジュール [1] へ展開された．いずれの機器でも，持ち方や所有者の体格などに依存しない高いロバスト性を持ったアルゴリズムを非力なマイコンで演算しなくてはならないことが課題である．

歩数計は，歩行によって生じる鉛直方向の加速度の波を数えることで実現できる．ここで端末の装着姿勢が不明なため三軸加速度センサの値から鉛直方向を推定しその成分を取り出さなければならない．その方法を D-RNN 回路で表したものが図 5 である．図中 (a) は加速度センサの x 成分に対し，過去 N 個の移動平均を計算するものでローパスフィルタの働きをする．(c) も (a) と同様の処理をそれぞれ y 成分， z 成分に対して行っていることから， N を適切

図 5 鉛直方向成分の抽出回路

に取ると重力ベクトル $\mathbf{G}$ を求めることができる. 一方 (b) は x 成分に対し過去 M 個の移動平均値を減算しておりハイパスフィルタの働きをする. (d) で y, z 成分に対しても同様の処理をすることから, M を適切に取ると歩行によって生じた振動 $\mathbf{S}$ が得られる. 求めるべき鉛直方向成分は $V_{acc} = |\mathbf{S}| \cos \theta$ と表現でき (θ は $\mathbf{G}$ と $\mathbf{S}$ の成す角), $|\mathbf{G}|$ を定数と考えれば

$$\mathbf{G} \cdot \mathbf{S} = |\mathbf{G}| |\mathbf{S}| \cos \theta \propto V_{acc} \quad (7)$$

と内積の演算を行うだけで V_{acc} に比例した値を求めることができる. 図中 (e) でこの内積演算を行っている.

図 6 は鉛直加速度から歩数を求めるための回路である. 図中 (a) は 1 サンプル前と現在の鉛直加速度の符号を比較することでゼロクロスするタイミングを検出している. 一方 (b) は加速度の積分を行う部分である. (a) によってゼロクロスが検出されると値が零に戻るため半周期ごとの積分値を求めている. この値は (c) で絶対値にした後, (d) にて歩行によって生じる最低値 S より大きいかわ調ベノイズによる誤カウントを防止している. ゼロクロスは一步につき 2 回発生するため, (e) で回数を加算し 1/2 倍したものを歩数としている.

これらは説明のために多少簡略化しており, 実際は歩行振動がより顕著に見えるようフィルタを多段にするなどの工夫を行っている. またプログラムコードは図 5 や 6 の回路をそのまま形にしたものでなく, よりデジタルコンピュータ向きにしている. 例えば図 7 は図 5 (a) で示した移動平均 (ローパスフィルタ) を C 言語でコード化したものである. 図右の `lpf()` がフィルタ本体のコードで, サンプリング間隔 ΔT ごとに呼び出されるものである. フィルタを多

図 6 鉛直方向成分による歩数計測

```
int *cur, mem[MEMSIZE];
#define sum (*(cur))
#define now (*(cur+1))
#define buf (cur+2)

// 毎サンプリングループ前に
// 呼ぶべき関数
void preproc() {
    cur = mem;
}

int lpf(int x, int n) {
    int ret;

    sum -= buf[now];
    sum += buf[now]=x;
    ret = sum/n;
    if (++now >= n)
        now = 0;
    cur += n+2;
    return ret;
}
```

※システム起動時のメモリ(mem)初期化処理, およびエラー処理省略

図 7 C 言語による移動平均

段にするため等この `lpf()` は ΔT 中に何度も呼び出してよいが, 呼び出す順番や第二引数 (フィルタ係数 n) の値は固定でなければならない. これらを変更することは回路の結線を変えることを意味する. RNN では結線を動的に変えることは想定しないため, この制約は問題とならない. 一方 `lpf()` 内部は図 5 (a) の形と異なりリングバッファによって効率化されている. これは等価な処理を RNN 回路で表現できさえすれば, 実装手段自体には制約を持たせてないためである. なお, 誌面の都合上詳細は述べないが RNN にも忌避すべき結線ルールという制約は存在する [6].

この歩数計アルゴリズムは IoT モジュールのほかに SP や犬用歩数計などに組み込まれている. ただし記述言語はそれぞれのハードやシステム構成下で最適と思われる物を用いている. 例えば IoT モジュールでは C 言語を用いた. 加速度センサデータから歩数を算出するアルゴリズムは 150 行程度であり消費メモリは 340 [byte] 強である. 一方の犬用歩数計では 8 bit CPU のアセンブラを使用し, その量はおおよそ 1,200 行程度, 使用メモリは 170 [byte] 強である. このようなコンパクトなアルゴリズムにかかわらず計測精度は 9 割以上である [7].

3.2 動作検出

本稿で挙げた各機器は, 歩数を測るだけでなく様々な動作も検出できるようになっている. 例えば, IoT デバイスを装着した者が転倒や転落など特定の動きをしたか, SP を鞆から取り出すなど特徴ある動きがあったか, 犬が胴震いしたかなどである. これらの動作検知も RNN で表記できる.

これらの動作検知は, その動きがよく現れる加速度成分を取り出し, 典型的な形状 (テンプレート) との相関を取ればよい. 人間の身長は $160 \pm$ 数十 [cm] 程度しか分布しておらず力学的性質は大差ない. そのため特定の動きに対する加速度は人によらずほとんど同じ動きをする. 図 8 はしゃがみ込み動作時の鉛直方向の加速度の様子を描いたものである. 動作開始後, 体が下方向に動き出すことによって鉛直加速度は負となる. 床に近づくにつれ減速動作のため正となり, 動作が終わると零に戻る. 加速度の極大値, 極小値, および動作完了時からの時間をそれぞれ a_1, a_2, t_1, t_2 とすると, これに動作時間 t_3 を加えた五つの値がテンプレートパラメータとなる. 時刻 t における鉛直加速度成

図 8 しゃがみ込み動作時の加速度の動き

図9 シャガみ込み動作検出回路

図10 回転検出回路

分の値を $v(t)$ とすると、相関値は

$$C(t) = v(t)^2 + (v(t - t_1) - a_1)^2 + (v(t - t_2) - a_2)^2 + v(t - t_3)^2 \quad (8)$$

と計算することができる。この式はテンプレートとの二乗距離を表すことから値が小さいほど相関が高いこととなる。常時この値を計算し、閾値 L より小さくなると動作を確定すればよい。これを RNN 回路で示したのが図9である。本来、鉛直加速度 $v(t)$ を得るためには図中 X で平方根を求める必要があるが、本回路では省略している。これは平方根を求めるのに計算コストが掛かり、また二乗値のまま相関をとってもほとんど性能が変わらないためである。

一方、図10は端末が短時間にある角度 ϕ 以上回転したことを検出する回路である。転倒や机に置いた SP を傾けて画面を確認するといった動作の検知に活用されている。本来、こういった回転の有無はジャイロセンサを使う方が適切である。しかしジャイロは加速度センサに比べ消費電力が1桁以上大きい問題があり常時計測の用途には不向きである。三軸加速度センサに対しローパスフィルタを通して得た重力成分 G と、遅れ要素を用いて得た d 秒前の重力成分との

内積を求めている。この値は d 秒間の重力方向の動きを θ とすると $g^2 \cos \theta$ と考えることができる ($g = |G| = \text{const}$ と仮定)。これよりこの値が閾値 $g^2 \cos \phi$ より小さいか否かで ϕ 以上回転したか判定することができる。

4. おわりに

ロボットシステムの記述法として開発された双線形時間遅れニューラルネットワーク (RNN) を IoT デバイス向けセンシングアルゴリズムへ展開した事例を紹介した。RNN は必要十分な近似演算を和積演算とメモリ処理のみで行うため計算機資源の乏しい組み込み系への実装に適している。今回紹介した各回路は、非常に単純で一見上手く動くとは思えないかもしれない。しかし適切に調整すると驚くほどの性能を出すことができる。現状 RNN は一般的なニューラルネットが持つ学習機能はなく開発者が手動でアルゴリズム開発とパラメータ調整をしている。学習の実現に向けた研究を進めており、すでにいくつかの興味深い知見が得られている [8]。将来的には開発者の介在なしに自動的に動作検知アルゴリズムが構築できることを目指している。

参考文献

- [1] 富士通ユビキタスウェア, <http://www.fmwworld.net/biz/uware/>, 2016.
- [2] 村瀬有一, 永嶋史朗, 安川裕介ほか: “小型ヒューマノイドロボットプラットフォーム: HOAP-1, -2 および NueROMA”, 日本ロボット学会誌, vol.23, no.1, pp.48–49, 2005.
- [3] 永嶋史朗: “双線形時間遅れニューラルネットワークによるロボットソフトウェアシステム”, 日本ロボット学会誌, vol.24, no.6, pp.735–746, 2006.
- [4] F. Nagashima: “Bilinear Time Delay Neural Network System for Humanoid Robot Software,” collection of articles “Humanoid Robot, Human-like Machines,” edited by M. Hackel, pp.497–518, 2007.
- [5] R. Zaier and F. Nagashima: “Motion Generation of Humanoid Robot based on Polynomials Generated by Recurrent Neural Network,” Proc. of the First Asia Int. Symp. on Mechatronics, pp.659–664, 2004.
- [6] T. Ito, Y. Senta and F. Nagashima: “Analyzing Bilinear Neural Networks with New Curve Fitting for Application to Human Motion Analysis,” Proc. of the 2012 IEEE Int. Conf. on Systems, Man and Cybernetics, pp.345–352, 2012.
- [7] 千田陽介, 伊東利雄, 永嶋史朗: “犬用歩数計の開発”, 映像情報メディア学会誌, vol.69, no.5, pp.207–212, 2015.
- [8] T. Ito, Y. Senta and F. Nagashima: “Continuous-time System Identification for Discrete Data by Curve Fitting,” Proc. of 2015 Int. Automatic Control Conf., pp.40–47, 2015.

千田陽介 (Yosuke Senta)

1994 年東北大学大学院精密工学専攻修士課程修了。1997 年 (株) 富士通研究所に入社。2005～2008 年 (財) 九州システム情報技術研究所へ出向。2016 年 6 月より久留米工業大学情報ネットワーク工学科教授。IoT 組み込み機器の研究に従事。博士 (工学)。

伊東利雄 (Toshio Ito)

2001 年東京大学大学院数理科学研究科博士課程修了。2003 年 (株) 富士通研究所に入社。2007 年カリフォルニア大学サンディエゴ校客員研究員。神経科学と人工知能の研究に従事。博士 (数理学)。