LUKE – learning underlying knowledge of experts –
ユーザー マニュアル

T. Makino / M. Shiro

平成 26 年 3 月 31 日

1 はじめに

LUKE は環境モデルパラメータの徒弟学習 [1] を効率的に実現するためのソフトウェア [4] である。環境モデルパラメータの徒弟学習は、逆強化学習をモデルパラメータの推定へと一般化したものである。ユーザはエキスパートが行ったと想定される行動の列と観測された状態の列、およびいくつかの信じられる事前分布を与える必要がある。LUKE を使うことで、そこから報酬関数と遷移行列と観測行列を推定することができる。

通常の強化学習 [3] では、報酬 (reward) を外部から設定し、自らの行動 (action) とそれによる報酬を情報源として、環境の状態を推定してゆく。具体的には、報酬の期待和を最大にするようにモデルを更新してゆく。しかし、多くの場合、報酬関数は明らかではない。このため、逆に観測の列と行動の列から報酬関数を推定するという問題の枠組みが考えられる。これが逆強化学習である [2]。徒弟学習はそれをさらに一般化した問題で、観測の列と行動の列が与えられたとき、そこから「遷移行列」「観測行列」「報酬関数」のすべてを推定可能にするものである。このためには、観測された行動の列を「環境モデルをよく分かっているエキスパート」が行ったものと仮定する。

本マニュアルでは、前提となる環境モデルパラメータの徒弟学習について述べたあと、インストール法、利用法について説明する。

2 環境モデルパラメータの徒弟学習について

徒弟学習そのものに関して詳しくは、文献 [4, 1] を参照されたい。ここでは、必要最低限の定義を述べる。

部分観測環境 (POMDP) $\mathcal{P}$ は、$\langle \mathcal{S}, \mathcal{A}, \mathcal{Z}, T, O, \mathbf{b}^{(0)}, R, \gamma \rangle$ の組で表現される。

- $\mathcal{S}$ は S 個の状態の集合。
- $\mathcal{A}$ は A 個の行動の集合。
- $\mathcal{Z}$ は Z 個の観測の集合。
- $T = \langle T^{a_1}, \ldots, T^{a_A} \rangle$ はサイズ $S \times S$ の行列 A 個からなる状態遷移行列であり、 $T^a = (t^a_{s's})$ の各要素 $t^a_{s's} = P(s' \mid s, a)$ は状態 s で行動 a をとった結果状態 s' になる確率を表す。

- $O = \langle O^{a_1}, \ldots, O^{a_A} \rangle$ はサイズ $Z \times S$ の行列 A 個からなる観測行列であり、 $O^a = (o^a_{zs})$ の各要素 $o^a_{zs} = P(z|s,a)$ は、行動 a をとった結果状態 s に至ったときに観測 z が得られる確率を表す。
- $R = \langle R^{a_1}, \ldots, R^{a_A} \rangle$ はサイズ $S \times S$ の行列 A 個からなる報酬行列であり、 $R^a = (r^a_{s's})$ の各要素 $r^a_{s's}$ は状態 s で行動 a をとった結果状態 s' になった場合の報酬値を表す。
- $\mathbf{b}^{(0)}$ は長さ S のベクトルであり、初期状態の確率分布を表す。
- $\gamma \in [0,1)$ は割引率。

POMDP においては、学習エージェントの目的は、割引後の報酬和の期待値 $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^{\mathrm{T}} r_{a_t s_t}]$ を最大化するような方策 π を発見することである。方策は、信念 $\mathbf{b}$ が与えられたときの行動の確率分布である。ここで、信念とは、環境における現在の状態の確率分布を表すベクトルである。

いま、いくつかの未知パラメータ θ によってパラメタライズされた POMDP $\mathcal{P}_\theta = \langle \mathcal{S}, \mathcal{A}, \mathcal{Z}, T_\theta, O_\theta, \mathbf{b}^{(0)}_\theta, R_\theta, \gamma \rangle$ を考える。学習するエージェントは事前分布 $p(\theta)$ のみを知っているが、エキスパートは環境の真のパラメータ θ^* を知っており、その環境 $\mathcal{P}_{\theta^*}$ の上で行動した場合の最適価値 $Q^*(\mathbf{b}, a)$ を知っているものと考える。

エキスパートの行動選択は、最適価値の Soft-max policy に基づくものと仮定する。すなわち、

$$\pi^*(a \mid \mathbf{b}) = \frac{max_{n:a_n=a} \exp(\beta Q^*(\mathbf{b}, a))}{\sum_{a' \in A} \exp(\beta Q^*(\mathbf{b}, a'))} \tag{1}$$

ここで、β は行動選択の最適性を制御する逆温度パラメータである。

すると、LUKE が扱う環境モデルの徒弟学習は、エキスパートによる L ステップの演示 $D = (a^1, z^1, \ldots, a^L, z^L)$ が与えられたときに、パラメータの事後分布 $p(\theta \mid D)$ を最大化するパラメータ $\hat{\theta}$ を求める (そして、$\mathcal{P}_{\hat{\theta}}$ の最適方策を求める) 問題と定式化できる。

LUKE は、次の 3 つの動作モードを備えている。

- 演示生成モード (GEN): パラメータが既知の POMDP $\mathcal{P}$ を入力し、その POMDP の最適解 $\pi_{\mathcal{P}}$ を計算し、演示 D を出力するモード。本来は必要ないが、実際の演示を利用する前の段階で、モデルやパラメータの設定を調べたりするために利用する。
- 環境学習モード (IOHMM): パラメタライズされた POMDP $\mathcal{P}_\theta$, 事前分布 $p(\theta)$, 適当な行動選択による演示 D の 3 要素を入力し、行動選択に関して仮定を置かずに $\hat{\theta}$ を計算するモード。徒弟学習を利用しない、従来の環境学習手法に対応するモードである。
- 徒弟学習モード (APPL): パラメタライズされた POMDP $\mathcal{P}_\theta$, 事前分布 $p(\theta)$, エキスパートによる演示 D の 3 要素を入力し、エキスパートが環境の最適解で行動していることを仮定して $\hat{\theta}$ を計算するモード。

3 インストール

3.1 要求システムとライブラリ

Ubuntu 13.10 以降なら、CPLEX 以外はまとめて

```
apt-get install python g++-4.8 libboost1.53-dev libtbb-dev libnlopt-dev graphviz
```

でインストールできる。

- Linux 3.2 以上 (ディストリビューション、32bit/64bit は問わない)
  Cygwin は intel tbb がコンパイルできないため、現状では未対応である。Windows 環境下の Visual C++ でも動くかもしれない (最近まで動作していたが、Visual C++の内部エラーを引き起こすためコンパイルできなくなった。避ける方法が見つかれば動作させられるかもしれない)。

- GCC version 4.8 以降 (必要なバージョンに注意)
  もし手元で gcc をビルドするのであれば、必要なソフトウェア (libgmp-dev libmpc-dev libmpfr-dev liblibisl-dev libcloog-isl-dev) をインストールしたあと、`./configure --prefix=/usr/local --enable-languages=c,c++,lto` などどとして config ファイルを作り、`make -j12`、`sudo make install` などとする。`j` オプションはコンパイルする計算機の Core の数にする。

- Boost version 1.53 以降 (必要なバージョンに注意)
  Boost が最新でない場合は、
  ```
  ./bootstrap.sh --prefix=/usr/local
  ./b2 variant=debug install -j12
  ./b2 variant=release install -j12
  ```
  等でコンパイルが必要。`j` オプションはコンパイルする計算機の Core の数にする。Boost は template ライブラリであるが、LUKE は単にヘッダをコピーするだけでは動作しない。

- Python
  waf を動作させるために必要。大抵のシステムには標準的にインストールされているが、なければインストールする。

- Intel tbb
  `http://threadingbuildingblocks.org/download` から取得してインストールする。

- nlopt version 2.3 以上
  `http://ab-initio.mit.edu/wiki/index.php/NLopt` から取得してインストールする。インストール方法は、`./configure` のあと `make` と `make install` を順に実行する。

- CPLEX: 現バージョンの LUKE は線形計画問題を解くために IBM 製品である CPLEX を利用する。教育機関のスタッフであれば、IBM Academic Initiative に登録することで無料で利用可能である。線形計画問題を解くためだけに使っているので、ほかのフリーのソフトウェアに置き換えることは難しくないはずである。インストールされていなければ、使わずに動作しようとする (が、現在のところうまくいかない)。

3.2 あれば良いライブラリ

- graphviz
  推定結果を可視化するためのツールである。LUKE の結果は`.dot` ファイルで与えられる。Debian 系列のディストリビューションでは `apt-get install graphviz` でインストールできる。そうでなければ `http://www.graphviz.org/content/fsm` から取得し、コンパイルする。使い方は `dot -Tpng nodes.dot > results.png` などとする。

3.3 インストール方法

- パッケージをダウンロードし、展開する [1]。
- コンパイル
  1. pomdp 下の trunk フォルダへ入る: `cd pomdp/trunk`
  2. C++コンパイラと必要なライブラリがデフォルトの場所にない場合は、環境変数を設定する。
     `export CXX=/usr/local/bin/g++` (bash 系)
     `export LD_LIBRARY_PATH=/usr/local/lib:/usr/local/lib64` (bash 系)
  3. 各ライブラリの場所を引数に与えて config ファイルを作成する:
     `./waf configure`
  4. ビルドする:
     `./waf release`
- サンプルの実行:
  `build/release/src/bayesmain --model=tiger.model --expertmodel=experttiger.model --task=APPL_EM` などとしてみる。`--task=APPL_EM` は徒弟学習を行うことを意味する。graphviz がインストールされているなら、サンプル実行後に `dot -Tpng nodes.dot > nodes.png` とすると学習結果が見られる。

多くのトラブルは config ファイルの生成において発生する。特にライブラリの位置がデフォルトと異なるときは明示的に与えなくてはならない。生成される `build/config.log` を見ることで、原因が判明する場合もある。

- `--boost-libs`=ブーストのライブラリ位置
- `--boost-includes`=ブーストのヘッダファイル位置
- `--cplex`=CPLEX の位置
- `--tbb`=TBB の位置

次は筆者の環境での config ファイル作成例である。
`./waf configure --cplex=/usr/local/lib/cplex --tbb=/usr/local/lib/tbb`

4 設定と実行

4.1 実行とコマンドライン引数

無事にコンパイルが成功したら、`build/release/src/bayesmain` という実行ファイルができあがっている。ユーザは次節のフォーマットで model ファイルを書き、`bayesmain` のコマンドライン引数に与えて実行する。引数は次の通りで`--`引数名`=`パラメータの形式で与える。引数を与える順番は自由に変えて構わない。

[1] 将来的には tarball で配布されるが、現状では subversion での配布である。`svn checkout https://dav.snowelm.com:4443/pomdp/`から取得する。

4.1.1 基本となる引数

引数--task によって動作モードを指定する。

- LOAD_ONLY: 指定されたファイルの読み込みだけを行う。文法チェックなどで利用する。
- GEN_DEMO: エキスパートのモデルを与えて、それに従った時系列データを生成する。--expertmodel、--demolength でエキスパートのモデル（十分に学習された結果のモデル）と、出力時系列の長さを指定する。既定値ではこれが選択される。
- APPL_EM: 徒弟学習を行う。
- IOHMM_EM: Input-Output 隠れマルコフモデルを解く。上記の環境学習モードに対応する。

4.1.2 その他の引数

task によって必要になる追加引数。簡単な一覧と既定値は引数に--help を与えることで表示される。

- --model: ひな形の model ファイルを指定する。
- --demodata: エキスパートの行った時系列データファイルを指定する。
- --algo: 最適化に利用するアルゴリズムを指定する [2]。負の値は勾配データを利用する場合。詳細は、http://ab-initio.mit.edu/wiki/index.php/NLopt_Algorithms を参照する。

  0 Nelder-Mead Simplex 法
  1 BOBYQA 法
  2 COBYLA 法
  3 Principal AXIS 法
  4 Subplex ベースの方法
  5 Controlled Random Search 法
  6 Improved Stochastic Ranking Evolution Strategy 法
  −1 Moving Asymptotes 法
  −3 SLSQP 法
  −4 Low-storage BFGS 法
  −5 Preconditioned truncated Newton 法
  −6 Shifted limited-memory variable-metric, Rank1
  −7 Shifted limited-memory variable-metric, Rank2

  標準では 1 の BOBYQA 法が選択される。
- --skiptest: 本実行の前のモデルテストをスキップするかどうか。1 でスキップする。
- --verbosity: 出力の冗長性を指定する。model のデバグなどをしたいときは 2 を指定する。

[2] これらは内部で Nlopt ライブラリに渡される。

- --test_length: 徒弟学習完了後に、学習結果を評価するためのテスト時系列の長さ（1000000）
- --beta: エキスパートのデータから内部でモデルを作る際の正規化指数関数の逆温度パラメータ。
- --stepsize: 最適化の際のパラメータ変更の刻み幅[3]。
- --xtimeout: パラメータソルバが全体の最適化を諦めるまでの時間。単位は秒。既定値は 86400=24 時間。
- --xtol: パラメータ最適化の際の許容範囲。この値以下であれば最適化されたと見なす。
- --ftol: 関数値の相対許容誤差。
- --ptimeout: 内部的に繰り返される POMDP ソルバが 1 回の最適化を諦めるまでの時間。単位は秒。既定値は 600=10 分。短すぎると、POMDP ソルバの解が不安定になり、最適解ではない結果に収束する可能性がある。
- --ptol: POMDP ソルバにおける価値の許容誤差。
- --delta: 確率値が同じであると見なす幅。既定値は 10^{-5}
- --seed: 乱数の種
- --transfer_solution: 1 を指定すると、高速化のために、POMDP ソルバーが前回の解を再利用する。
- --single: 単精度実数で問題を解く。多少使用メモリが減り、高速化されるかもしれない。
- --ndemos: エキスパートデータをモデル形式で与える場合に、内部で生成する時系列を何個出力するか。
- --demo_length: エキスパートデータをモデル形式で与える場合に、内部で時系列データに展開するときの時系列長。既定値では 100。
- --expertmode: エキスパートのモデル（十分に学習されたあとのモデル） demo ファイルを指定しない
- --outfile: 出力ファイル。GEN_DEMO モードでは、デモファイルが出力される。指定しなければ標準出力で結果をだす。（NULL）

4.2 demo ファイルの記述

エキスパートの行った時系列データのファイルであり、各行は次の構造を持つ。空白行を挟んでよい。「%」以下行末まではコメントとみなされる。

行動，観測

行動、観測は、model ファイルで記述された表記法で記述される。

[3] これは内部で NLopt ライブラリに渡される。

4.3 model ファイルの記述

model ファイルの文法は MATLAB に近いが、行列の添え字の開始が 0 になっている点は注意する。文の終わりは「;」である。「%」以下行末まではコメントである。model ファイルの記述は複雑なので、ここでは部分観測マルコフモデル (POMDP) において有名な Tiger 問題の徒弟学習版を例に model ファイルの書き方を説明する。

オリジナルの Tiger 問題は、目の前に二つの扉がありその向こうにそれぞれ部屋がある状況を考える。片方の部屋にトラがいて、もう片方の部屋にはお金がある。エージェントは、どちらにトラがいるか分からない状況下で、右の扉を開ける (open_right) か左の扉を開けるか (open_left)、音を聴く (listen) かの 3 つの行動をとれる。扉を開けてトラがいれば食べられてしまうので大きな負の報酬が設定される。お金があれば正の報酬、音を聴くと若干の負の報酬が設定される。そうして期待報酬を最大にするようにエージェントが行動を最適化し、トラのいない扉を開けることを学習してゆく。

徒弟学習では、こうした問題設定のモデル（それは状態とそれらの遷移）と、環境の真の設定 (トラがいる確率、音が間違って聞こえる確率、トラのいるドアを開けた場合の損失など) を十分に分かっているエキスパートの行動とその結果の観測の時系列 (演示) が与えられる。以下はその実際の model ファイルである。

```
%行動の集合としては左の戸を開ける，右の戸を開ける，音を聴くの 3 つが与えられる
[open_left, open_right, listen] = ActionSet;

%状態の集合としてはトラが左にいる場合と右にいる場合があり得る
[tiger_left tiger_right] = StateSet;

%観測の集合としては左から吠える声が聞こえるか右から吠える声が聞こえるかの二種
[grawl_left, grawl_right] = ObservationSet;

%パラメータ badreward を定義し、正規分布に従うものとする。
badreward = Normal( -50, 50 );

%パラメータ InitProb、ListenLeft、ListenRight を定義し、ベータ分布に従うものとする。
InitProb = Beta( 3, 3 );
ListenLeft = Beta( 5, 3 );
ListenRight = Beta( 5, 3 );

%Init は予約語。初期状態配列を列ベクトルで与える。
Init = [InitProb ; 1-InitProb];

%Trans は予約語。状態間の遷移行列を与える。eye(2) は 2 × 2 の単位行列。
Trans(:,:,open_left) = [Init Init ];
Trans(:,:,open_right) = [Init Init ];
Trans(:,:,listen)    = eye(2);
```

%Obser は予約語。観測行列を与える。ones(2,2) は要素がすべて 1 の 2 × 2 行列。
%allot は rem_の部分に残りの値を均等割する予約関数。

```
Obser(:,:,open_left) = ones(2,2) * 0.5;
Obser(:,:,open_right) = 0.5;
Obser(:,:,listen)    = allot([ListenLeft 1-ListenRight ; PlaceHolder PlaceHolder] );
```

%Rewar は予約語。推定すべき報酬行列を与える。

```
Rewar(:, :, listen) = -1;
Rewar(:, tiger_left, open_left) = badreward;
Rewar(:, tiger_right, open_right) = badreward;
Rewar(:, tiger_right, open_left) = +10;
Rewar(:, tiger_left, open_right) = 10;
```

%Discount は割引率を与える予約語。

```
Discount = 0.75;
```

予約されている語、変数、組み込み関数について一覧にまとめる。予約語は、以下の 3 つである。

- ActionSet: 行動集合
- StateSet: 状態集合
- ObservationSet: 観測集合

これらの予約語は、複数回呼び出すことで 2 次元以上の空間を指定する。例えば、
`[s11, s12, s13] = StateSet; [s21, s22, s23] = StateSet;`
では、状態が値のペアによって表現され、各々が 3 種類の値をとることを意味する。つまり、状態は $s11+s21, s11+s22, s11+s23, s12+s21, s12+s22, s12+s23, s13+s21, s13+s22, s13+s23$ の合計 9 通りの値を持つことができる。実際、これらには $0, \ldots, 8$ の値が割り当てられる。状態、行動、観測は、ここで定義された名前で参照してもよいし、対応する整数で参照してもよい。演示データファイルでも同様である。
予約された変数名は以下の 5 つである。

- Init: 初期状態 $\mathbf{b}_0(s)$
- Trans: 状態遷移行列 $p(s'|s,a)$
- Obser: 観測行列 $p(z|s,a)$
- Rewar: 報酬行列 $R(s',s,a)$
- Discount : 割引率 γ

特に Trans、Obser、Rewar は 3 次元行列であることに注意されたい。
LUKE では MATLAB に準拠した記法により、行列のスライスを自由に作ることができる。例えば Obser(:,:,open_left) と記述することで、Obser の第三成分を open_left であるようなスラ

イス (この場合は、第一成分と第二成分全体の行列) を表す。また、予約された変数名以外を=の左辺で使うことで、自由に変数を定義し、利用してよい。
組み込まれている確率分布は以下 2 つである。

- `Normal(`μ`,`σ`)`: 正規分布。正規分布の共役事前分布であり、連続変数の初期値としてよく使われる。
- `Beta(`α`,`β`)`: ベータ分布。二項分布の共役事前分布であり、確率変数の初期値としてよく使われる。多項分布はベータ分布の組み合わせで表現することができる。

これらはいったん変数に代入しないと利用することができない。環境の不確かさを表すベイズ的な確率分布であるため、環境の確率的な振る舞いを記述することはできないことに注意。たとえば、ギャンブルの結果の報酬分布を表すことはできない (遷移先状態を分割し、各々に別の報酬値を割り当てる必要がある)。
組み込み関数は主に、簡潔に記述するための補助関数で、以下が実装されている。

- `allot(rem_)`: 確率の均等割。行列の各列の和が 1 になるように、プレースホルダー `PlaceHolder` (短縮形 `_` も利用可能) の値を調節する。計算は列ごとに行われる。

$$\mathrm{allot}\begin{pmatrix} 0.3 & 0.2 & 0.5 \\ 0 & \_ & 2\times\_ \\ \_ & \_ & 3\times\_ \end{pmatrix} = \begin{pmatrix} 0.3 & 0.2 & 0.5 \\ 0 & 0.4 & 0.2 \\ 0.7 & 0.4 & 0.3 \end{pmatrix}$$

- `conv2(M, N)`: 与えられた 2 つの疎行列 M, N について、その畳込みを返す。M に状態集合 S_1 に関する遷移行列、N に状態集合 S_2 に関する遷移行列を指定し、畳み込みを行うことで、状態集合 $S_1 \times S_2$ に関する遷移行列を得ることができる。プレースホルダーを含んだ行列に畳み込みを行い、一部を変更したあと `allot` 関数を呼び出すことで、特別な場合にのみ相互作用する状態空間を記述することが容易になる。
- `conv3(M, N)`: 与えられた 2 つの疎行列 M, N について、その 3 次元畳込みを返す。
- `carteset(s1, s2)`: 集合 1 と集合 2 の直積を作る。
- `expand(m, v1, v2 ...)`: 行列 m を、添え字ベクトル v1, v2, ... に指定した移動量でずらして和をとる。添え字ベクトルが 2 個指定されているなら `temp(v1,v2) = 1;` として作った行列との畳み込み `conv2(m, temp)` と等価である。部分空間 1 に対して定義した報酬関数を部分空間 2 方向に拡張する場合に利用する。

その他、MATLAB にある以下の関数が利用可能である。

- `eye(n)`: サイズ $n \times n$ の単位行列を返す
- `ones(n1, ...)`: 指定されたサイズの、すべての要素が 1 の行列を返す。
- `zeros(n1, ...)`: 指定されたサイズの、すべての要素が 0 の行列を返す。
- `length(v)`: ベクトルの長さを返す。
- `size(m)`: 行列のサイズをベクトルとして返す。
- `horzcat(m, n)`: 行列を水平方向に連結する。`[m, n]` または `[m n]` と書いてもよい。

- `vertcat(m, n)`：行列を垂直方向に連結する。`[m; n]` と書いてもよい。
- `cat3(m, n)`：行列を 3 次元目方向に連結する。
- `display(x)`：x の中身を表示する。

4.4 出力フォーマット

実行が終了すると、nodes.dot ファイルが生成されるので、これを graphviz にて処理することで結果を PNG 画像等で得られる。実行ログは`--outfile` で指定されたファイルまたは、標準出力に対して出力される。出力は設定一覧、乱数で初期化されたサンプル値、与えられたモデルが出力されたあと、`Solve` 、`Calculate FIB` 以下に、状態が示される。

```
@Iteration Time(s) #Policy #Belief #BelHash #node/#free UpperBound LowerBound     Gap
@       16  15.72       2     481   59328 59330/    0    -16.845    -17.069    0.22405
@       32  23.63       2     581   70732 70734/    0    -16.848    -17.069    0.22115
@       48  23.72       2     574   70133 70135/    0    -16.848    -17.069    0.22115
@       59  28.56       2     493   70133 70216/    0    -17.025    -17.069   0.044227
@       60  32.92       2     493   70133 70216/    0    -17.031    -17.069   0.037939
```

さらに、`Finish solve, loglike=-***`という形で状態の対数尤度が示され、これらが繰り返される。

5 問い合わせ先

牧野 貴樹 t-luke@snowelm.com

謝辞

本研究は、科学研究費補助金 (25730128)、および総合科学学術会議により制度設計された最先端研究開発支援プログラム (FIRST 合原最先端数理モデルプロジェクト) により、日本学術振興会を通じて助成を受けた。

参考文献

[1] Takaki Makino and Johane Takeuchi. Apprenticeship learning for model parameters of partially observable environments. In *Proc. of the 29th International Conference on Machine Learning (ICML)*, pages 1495–1502, July 2012.

[2] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 663–670, San Francisco, CA, USA, 2000.

[3] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[4] 牧野 貴樹, 城 真範, and 合原 一幸. 部分観測環境のパラメトリック記述に基づく高速モデルパラメータ逆強化学習プログラム. In **第** *28* **回人工知能学会全国大会論文集**, pages 2H1–1. 2014.