Bayesian Knowledge Tracing at Scale with `hmmsclbl` toolkit

Michael Yudelson, Ph.D., Carnegie Learning, Inc.

Student Modeling Applications, Recent Developments & Toolkits (SMART) at EDM 2015

`hmmsclbl` – HMM Scalable

- Toolkit for building Bayesian Knowledge Tracing (BKT) models of Big Datasets
  – And BKT is a Hidden Markov Model (HMM)
- How big is a Big Dataset?
  – 2010 Carnegie Learning Cognitive tutor data
  92.5 million records, 74 000 students
- How well did `hmmsclbl` do?
  – Mac Mini late 2012, 16Gb of RAM
  – EM model finished in under 5 minutes (parallel mode)
- Tested on educational data

Agenda

- **Bayesian Knowledge Tracing**
- Getting Started with `hmmsclbl`
- The Tinkering Points
- Advanced Topics

Bayesian Knowledge Tracing
Definition

Bayesian Knowledge Tracing Model $\subseteq$ Hidden Markov Model $\subseteq$ Bayes Net $\subseteq$ Graphical Model

When modeling skill acquisition

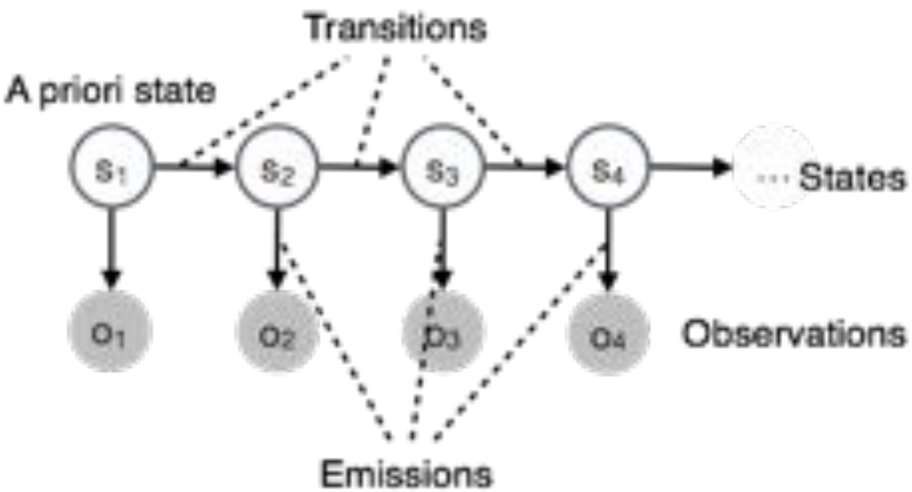
- One binary latent (unobserved) variable is represents mastery (skill is known|unknown)
- One observed binary variable represents student performance (skill is applied correctly|incorrectly)

Data

- Time-series data: correctness of skill application

Bayesian Knowledge Tracing
Representation

Unrolled view of BKT



Priors – Π

	Known	Unknown
	pL_0	1- pL_0

Transitions – A

	to Known	to Unknown
from Known	1-pF	pF
from Unknown	**pT**	1-pT

Emissions – B

	Correct	Incorrect
Known	1-pS	**pS**
Unknown	**pG**	1-pG

Parameters

- Values ∈[0,1]
- Rows sum to 1
- Forgetting (pF) = 0
- 4 parameters
  - Rows sum to 1, every last value can be omitted
  - Forgetting is 0, no 5th parameter

Bayesian Knowledge Tracing
Updating masteries, predicting

Parameters $\Pi = \begin{bmatrix} pL_0 \\ \neg pL_0 \end{bmatrix}$ $A = \begin{bmatrix} \neg pF & pF \\ pT & \neg pT \end{bmatrix}$ $B = \begin{bmatrix} \neg pS & pS \\ pG & \neg pG \end{bmatrix}$

Outputs $L = \begin{bmatrix} pL \\ \neg pL \end{bmatrix}$ $C = \begin{bmatrix} pC \\ \neg pC \end{bmatrix}$

Helpers $Le = \begin{bmatrix} pL|e \\ \neg pL|e \end{bmatrix}$ $Be(corr) = \begin{bmatrix} \neg pS \\ pG \end{bmatrix}$ $Be\ (incorr) = \begin{bmatrix} pS \\ \neg pG \end{bmatrix}$

	Matrix Version	**Simplified Computation**
1. Initialize	$L=\Pi$	$pL=pL_0$
2. Correct		
a. Predict	$C=L^T \times B$	$pC=pL\cdot(1-pS)+(1-pL)\cdot pG$
b. Pick Be	$Be=B[,1]$	
c. Compute Le	$Le=(L.\times Be)./(L^T \times Be)$	$p(L\|e=correct)=pL\cdot(1-pS)/[pL\cdot(1-pS)+(1-pL)\cdot pG]$
d. Update L	$L=Le \times A$	$pL=p(L\|e)\cdot(1-pF)+(1-p(L\|e))\cdot pT$
3. Incorrect		
b. Pick Be	$Be=B[,2]$	
c. Compute Le		$p(L\|e=incorrect)=pL\cdot pS/[pL\cdot pS+(1-pL)\cdot(1-pG)]$
4. Unknown		
d. Update L	$L=A^T \times L$	$pL=pL\cdot(1-pF)+(1-pL)\cdot pT$
5. Unknown/guess		
b. Pick Be	$Be=B[,argmax(C)]$	
continue as 2 or 3, depending on argmax(C)		

Legend
- $\times$ – vector/matrix multiply
- $.\times$ – multiply element by element
- $./$ – divide element by element
- Xe – X given evidence
- L^T – transpose
- $X[,i]$ – i^{th} column of X

Bayesian Knowledge Tracing. Special Note on Predicting

- Updating L (pL) when the observation is known
  - In the tutor, the observation (correct/incorrect) is available for the update
  - When comparing models making the observation available puts BKT at an unfair advantage
    - Especially, when comparing to AFM and PFA
  - Rule 4 from previous slide is too harsh
    - Matrix B (emissions) is ignored
    - Accuracy takes a considerable hit
  - Rule 5 is a compromise
    - Accuracy is better than under Rule 4 (but worse than under Rules 2&3)

Agenda

- Bayesian Knowledge Tracing
- **Getting Started with `hmmsclbl`**
- The Tinkering Points
- Advanced Topics

Getting started with `hmmsclbl`
Obtaining and compiling code

Get the source code from IEDMS GitHub repository

```
$> git clone https://github.com/IEDMS/standard-bkt
```

OR download and unzip

Compile

```
$> cd standard-bkt
$> make all
```

Operating system notes

- Linux – `gcc` with `Open MP` should be there by default
- Mac OS X – refer to hpc.sourceforge.net
- Windows – install `cygwin` with `awk, make, wc, time, unzip` commands

If you are on the Virtual Box Ubuntu 14.04 Machine we distributed

- User/password – usmartedm15/usmartedm15
- Files are in `~/Documents/USMARTEDM15/hmmsclbl`

Getting started with `hmmsclbl` Getting the data

KDD Cup 2010 on hosted by LearnLab's DataShop [pslcdatashop.web.cmu.edu/KDDCup] Largest freely available dataset of learner data donated by Carnegie Learning, Inc.

- Create at account
- Download Bridge to Algebra challenge set bridge_to_algebra_2008_2009.zip (461Mb)
- ~20 000 000 rows, ~6 000 students

Data is placed in subfolder `d`, models will be in subfolder `m`, prediction files in subfolder `p`, code is in the root that contains these subfolders

Getting started with `hmmsclbl` Prepping the data

Instead of unzipping a 5.3Gb training file, run:

```
$> time unzip -q -c d/bridge_to_algebra_2008_2009.zip
bridge_to_algebra_2008_2009_train.txt | awk "-F\t"
'BEGIN{OFS="\t"} {if(NR==1)next; skill=$20; gsub("~~",
"~", skill); skill=(skill=="")?".":($3"__"skill); print 2-
$14,$2,$3"__"$4,skill;}' > d/b89_train.txt
```

Reduces rich data to tab-delimited SSSS: success, student, step, skill(s). Multiple skills are ~-delimited. No skill label is denoted by . (dot).

Resulting text file is 2.9Gb.

Note: skills are prefixed with unit and section name. Identical skill labels could mean different things in different sections.

Getting started with `hmmsclbl` Running the tool

Let's start from EM solver

```
$> ./trainhmm -s 1.1 -d ~ -m 1 -p 1 d/b89_train.txt m/
b89_modelEM.txt p/b89_predictEM.txt
```

Here, `-s 1.1` is an EM solver, `-d ~` is setting skill delimiter, `-m 1` asks for fit metrics, and `-p 1` requires predictions to be written into a file

It takes a little under 2.5 minutes, fitting alone 1 minute

```
trained model LL=6903034.9861273 (4490671.7596719),
AIC=13820821.972255, BIC=13930074.307203, RMSE=0.318739
(0.352797), Acc=0.870607 (0.830841)
```

```
timing: overall 139.576931 sec, read 42.205482 sec, fit 60.997095
sec, predict 36.144902 sec
```

Here, LL – sum of negative log-likelihoods, AIC and BIT are Akaike and Bayesian Information Criterions, RMSE - root mean squared error, Acc – accuracy.

In braces are the RMSE and Accuracy for the rows without skill labels. Otherwise the metric is given for all rows, even if skill label was omitted.

Getting started with `hmmsclbl` Switching solvers

Now let's try a Stochastic Gradient Descent solver

```
$> ./trainhmm -s 1.2 -d ~ -m 1 -p 1 d/b89_train.txt m/b89_modelGD.txt p/
b89_predictGD.txt
```

It takes a little longer

```
trained model LL=7230246.8188073 (4817883.5923708), AIC=14475245.637615,
BIC=14584497.972563, RMSE=0.328491 (0.368380), Acc=0.863745 (0.818622)
timing: overall 169.495414 sec, read 44.967394 sec, fit 100.809863 sec,
predict 23.490269 sec
```

The overall accuracy is a little worse 0.863745 (vs. 0.870607), just as the overall RMSE – 0.328491 (vs. 0.318739).

- How much is ***a little*** and what does it mean
  - Individualized BKT (Yudelson et al., AIED 2013)
    - Increment in the number of rows we are predicting correctly (~50K)
    - **When** the improvement happens: earlier attempts – more valuable
  - Small Improvement for the Model Accuracy Big Difference for the Students (Yudelson & Ritter, AIED 2015)
    - Small difference in accuracy could mean tangible changes in how students work
    - Larger pLearn → faster mastery → fewer problems and less time

Getting started with `hmmsclbl` Process reports

First output: there are 2 observations, 6043 students, 1844 skills, and 61848 unique steps

```
input read, nO=2, nG=6043, nK=1844, nI=61848
```

As fitting progresses, results for every skill are given

```
skill     0, seq  4027, dat    32217, iter#  1 p(O|param)=   18360.7093550 >>   18343.6728364, conv=1
skill     1, seq  3889, dat    62244, iter#  7 p(O|param)=   21660.7485382 >>   16798.8022745, conv=1
skill     2, seq  4386, dat    98701, iter#  6 p(O|param)=   33991.2924686 >>   27522.3817164, conv=1
skill     3, seq  3888, dat    62923, iter#  6 p(O|param)=   23075.1928335 >>   19305.2954508, conv=1
...
```

To fit skill #1

- We use 3889 student sequences of data comprised of 62244 data points
- It takes 7 iterations to finally converge
- The sum of negative log-likelihoods decreases from 21660.7485382 to 16798.8022745
- Average likelihood of a student-skill sequence goes up 0.0038114422 → 0.01330562239 (3.5 times)

Getting started with `hmmsclbl` Comparing to baseline

Accuracies/RMSE of the Majority Class prediction (here, always predicting "correct")

```
$> awk -F"\t" '{N++; C+=$1==1; SSE+= (1-$1)*(1-
$1); }END{print C/N"\t"sqrt(SSE/N);}' d/
b89_train.txt
```

Accuracy=0.861663 and **RMSE=0.371936**. Majority class accuracy is slightly worse than BKT models we've seen. The majority class RMSE is tangibly worse.

Note! BKT, traditionally, uses first attempt at a problem step. Further work until student gets right is omitted.

- MC Accuracy ≈86% doesn't mean data is severely skewed or the CL Cognitive Tutor tutor is simple or the students are incredibly smart
  - 20 012 498 first attempts, 17 244 034 are correct = 86.1663% correct
  - 26 798 736 all attempts, 17 244 034 are correct = 64.3464% correct
- Contextual slip and guess BKT (Baker et al., 2010) uses all data.

Getting started with `hmmsclbl` Tweaking tolerance criterions

Default stopping criterion is the change in parameter values of 0.01. Here, we lower tolerance `-e 0.001` and increase the maximum number of iterations `-i 400` (from the default 200)

```
$> ./trainhmm -s 1.2 -e 0.001 -i 400 -d ~ -m 1 -p 1 d/b89_train.txt m/
b89_modelGDe3i4.txt p/b89_predictGDe3i4.txt
```

It takes a lot longer to build and the change in fit is small (compare to 0.863745 accuracy).

```
trained model LL=7045687.2615030 (4633324.0351113), AIC=14106126.523006,
BIC=14215378.857954, RMSE=0.322916 (0.359497), Acc=0.868310 (0.826750)
timing: overall 2115.538318 sec, read 44.908378 sec, fit 2041.510535 sec, predict
28.882947 sec
```

In addition, 4 skills actually failed to converge after 400 iterations.

The stopping criterion can be defined in terms of mean negative log-likelihood change per data point by using the following parameter signature `-e 0.001,l`

```
$> ./trainhmm -s 1.2 -e 0.001,l -d ~ -m 1 -p 1 d/b89_train.txt m/b89_modelGDe2l.txt p/
b89_predictGDe2l.txt
```

Fitting results achieved show small improvement in accuracy (compare to 0.863745.

```
trained model LL=7184702.3227653 (4772339.0963109), AIC=14384156.645531,
BIC=14493408.980479, RMSE=0.327298 (0.366485), Acc=0.864662 (0.820255)
timing: overall 185.898219 sec, read 2.105894 sec, fit 159.845276 sec, predict 23.690081
sec
```

Getting started with `hmmsclbl` Let forgetting happen

Default initial parameter values are: pL_0=0.5, 1-pF=1, pT=0.4, 1-pS=0.8, pG=0.2. To set defaults, one can specify a comma separated list via `-0 0.5,`**`1.0`**`,0.4,0.8,0.2`

Let's set a starting version of forgetting to 0.001

```
$> ./trainhmm -s 1.2 -d ~ -m 1 -p 1
-0 0.5,0.999,0.4,0.8,0.2 d/b89_train.txt
m/b89_modelGDpf0.01.txt
p/b89_predictGDpf0.01.txt
```

Known	Unknown
pL_0	1- pL_0

	to Known	to Unknown
from Known	1-pF	pF
from Unknown	pT	1-pT

	Correct	Incorrect
Known	1-pS	pS
Unknown	pG	1-pG

Small decrement in accuracy (rf. to 0.863745)

```
trained model LL=7747431.9193728 (5335068.6927902),
AIC=15509615.838746, BIC=15618868.173694, RMSE=0.340602
(0.387471), Acc=0.861669 (0.814925)
timing: overall 101.492081 sec, read 44.915405 sec, fit
30.866999 sec, predict 25.471139 sec
```

Getting started with `hmmsclbl` Speeding up reading the data file

Previous run – almost 1/3 of time is reading the input file

```
timing: overall 117.920293 sec, read 45.535030 sec, fit
35.287492 sec, predict 36.842098 sec
```

There is a helper utility to convert text file to binary file

```
$> ./inputconvert -s t -t b -d ~ d/b89_train.txt d/
b89_train.bin
```

Where, `-s t` means the source is text, `-t b` – the target is binary.

```
overall time running is 42.524978 seconds
```

If we rerun previous task, where `-b 1` means binary input and specify a new `.bin` file

```
$> ./trainhmm -s 1.2 -d ~ -m 1 -p 1
-0 0.5,0.999,0.4,0.8,0.2 -b 1 d/b89_train.bin
m/b89_modelGDpf0.01.txt
p/b89_predictGDpf0.01.txt
```

The reading time is much better

```
timing: overall 57.193008 sec, read 1.146349 sec, fit
31.356436 sec, predict 24.474850 sec
```

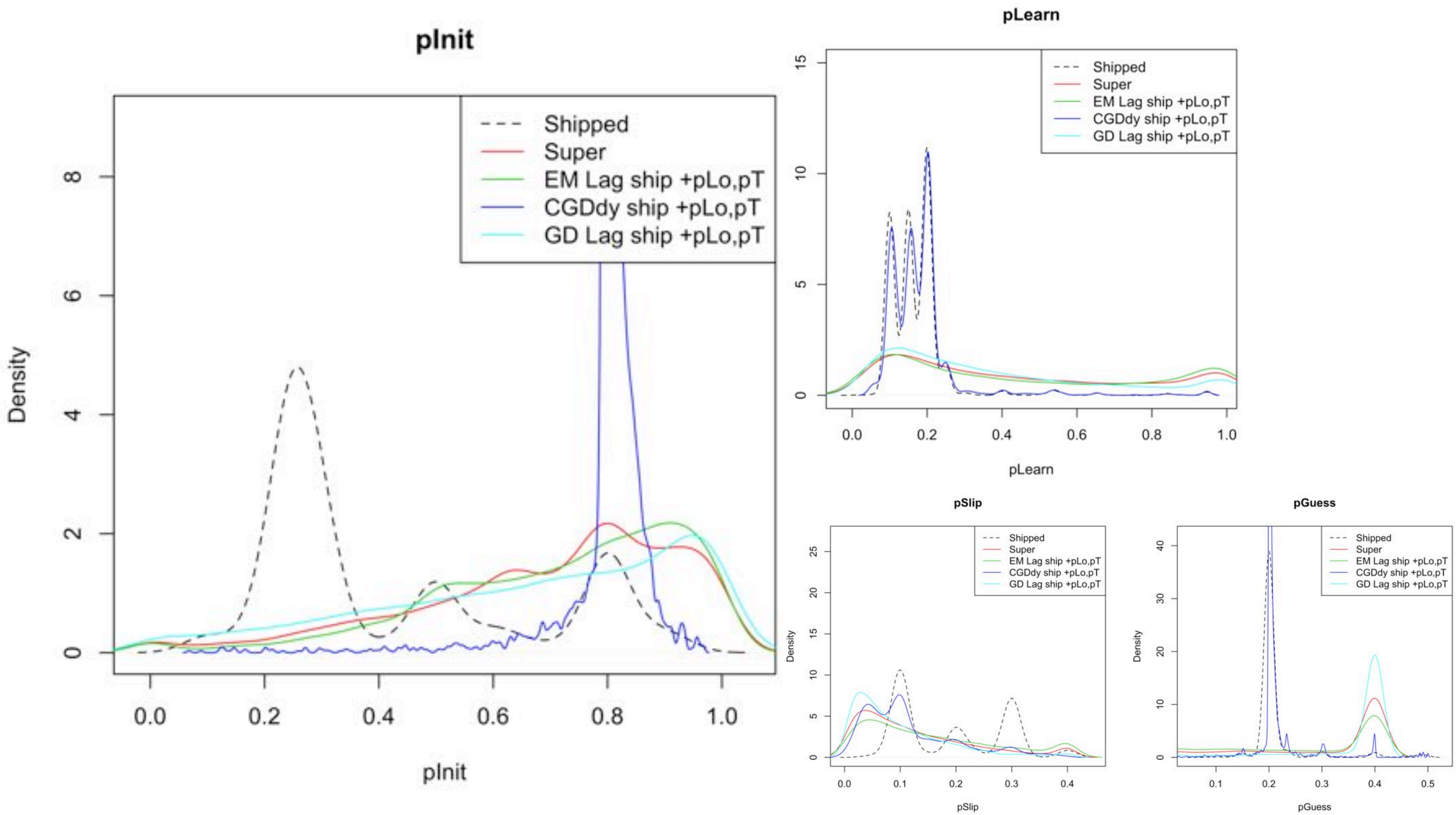
Why didn't we see drastic improvements?

- We are using same set of starting parameters
  - Harder and easier skills start in the same place
  - Parameter space has multiple local optima
  - \+ Seeding pL_0 and pT using AFM – *pre-classify* skills (e.g., via modified `liblinear` in the IEDMS GitHub)
- We are not taking context into consideration
  - Learning and forgetting could be different
  - Same day, next week, after Thanksgiving break
  - \+ Multiplexing Transition (A) and Emission (B) matrices
- Presence of noise in the data

  Tomorrow, Saturday, June 27, Room 1: Salón de Actos. 10:30-11:50

  M. Falakmasir, M. Yudelson, S. Ritter and K. Koedinger
  Spectral Bayesian Knowledge Tracing. (short)

How different model parameters are?



Agenda

- Bayesian Knowledge Tracing
- Getting Started with `hmmsclbl`
- **The Tinkering Points**
- Advanced Topics

The Tinkering Points
Set Starting Parameters and Limits

- Starting parameters `-0`
  - List all but the last in every row
  - Default is `-0 0.5,1,0.4,0.8,0.2`

	Known	Unknown
	pL_0	1- pL_0

	to Known	to Unknown
from Known	**1-pF**	pF
from Unknown	**pT**	1-pT

	Correct	Incorrect
Known	**1-pS**	pS
Unknown	**pG**	1-pG

- Set parameter limits
  - List all values in every row
  - Default upper `-u 1,1,1,0,1,1,1,0.3,0.3,1`
    - Slip and Guess are capped at 0.3
    - Forget is capped at 0
  - Default lower `-l 0,0,1,0,0,0,0,0,0,0`
    - 1-Forget is no less than 1

	Known	Unknown
	pL_0	**1- pL_0**

	to Known	to Unknown
from Known	**1-pF**	**pF**
from Unknown	**pT**	**1-pT**

	Correct	Incorrect
Known	**1-pS**	**pS**
Unknown	**pG**	**1-pG**

The Tinkering Points
Controlling the Process. Solvers `-s`

- Baum-Welch (EM) `-s 1.1` – classical algorithm
- Stochastic Gradient Descent `-s 1.2`
  - Scales gradients so that the maximum step of 1 does not take any of them over 1 or below 0 (finds n to divide all parameters by 10^n)
  - Halves the step until either Wolfe Conditions are met (a step is far enough but not too far) or 20 attempts. If only first WC is met, us that (Nocedal & Wright, 1999).
  - Re-align parameters so that rows sum to 1 (project onto simplex)
  - Attempts to make large leaps in parameter space (Wolfe Conditions counter-balance that)
- Conjugate Gradient Descent `-s 1.3.[1-4]`
  - Make first step as in Stochastic Gradient Descent
  - Computes step direction on subsequent next step using formulas
    - Fletcher-Reeves `-s 1.3.1`
    - Polak-Ribière `-s 1.3.2`
    - Hestenes-Stiefel `-s 1.3.3`
    - Dai-Yuan `-s 1.3.4`
  - Makes large leaps in parameter space
- Gradient Descent using Lagrangian multipliers `-s 1.4`
  - Uses gradients in a manner that does not need realignment of parameters (Levinson et al., 1983)
  - Smooth[er] parameter changes, just like EM
- Temporarily excluded Barzilai-Borwein and exponentiated Gradient updates

The Tinkering Points

Controlling the Process. Iterating and Stopping

- Iterating `-i 200`
- Stopping
  - Mean parameter change (per parameter) `-e 0.01`
    - Decreasing the tolerance increases fitting time, but
    - Not necessarily improves the fit drastically
  - Mean negative log-likelihood (per sequence) `-e 0.01,l`
    - On CLCT data, results in a slight improvement of the fit due to more iterations required to converge
- There is a precaution built in against parameter oscillation
  - Parameter values are stored for 2 iterations
  - If parameters at time *t* are close to parameters at time *t*-2 the fitting stops

The Tinkering Points
Controlling the Process. Regularization

- Regularization is often used to fight with over-fitting
- E.g., in Regularized Logistic Regression – penalize parameters for deviating from zero
  - Objective function for parameter x (neg. log-likelihood) – $f(x_i)$
  - Objective function with penalty – $f(x_i) + \lambda(\Sigma x_i)^{½}$
  - Gradient of a single parameter x – $g(x_i)$
  - Gradient of a single parameter with L2 penalty – $g(x_i)+½\lambda|x_i|$
    - Use center *c* other than 0 – $g(x_i)+½\lambda|x_i\text{-}c|$
- Specify penalty
  - All for being away from 0.5: `-c 1.0,0.5,0.5,0.5`
    ***λ*=1**, centroids for π, A and B are 0.5.
  - Gravitate priors and transitions to 0.5 and emissions to 0
    `-c 1.0,0.5,0.5,0.0`

```
$> ./trainhmm -s 1.2 -d ~ -m 1 -p 1 -b 1 -c 1.0,0.5,0.5,0.0 d/b89_train.bin m/
b89_modelGDreg.txt p/b89_predictGDreg.txt
trained model LL=7220628.1340159 (4808264.9075270), AIC=14456008.268032,
BIC=14565260.602980, RMSE=0.327139 (0.366232), Acc=0.864803 (0.820505)
timing: overall 215.000000 seconds, read 1.441625, fit 190.277211, predict 23.271876
```

Ran longer than no-regularized Gradient Descent. Small improvement in RMSE and Accuracy: 0.32**7139** vs. 0.32**8491** and 0.86**4803** vs. 0.86**3745** respectively (additional 21 173 correct predictions)

Agenda

- Bayesian Knowledge Tracing
- Getting Started with `hmmsclbl`
- The Tinkering Points
- **Advanced Topics**

Advanced Topics
Prediction

- Updating probability distribution of states (PDS)
  - In tutor, observation (correctness) is always known
  - When assessing the model, revealing observation is not fair (e.g., AFM and PFA always *fly blind*)
- Controlling how update of the PDS is done

  `-U` **`r`**`,`**`t`** – for known observation **reveal** them for PDS update, for unknown observation, use **transition** matrix only (Tutor)

  `-U` **`g`**`,`**`g`** – for known and unknown observations – guess most probable observation from PDO (Model Assessment)
- Probability distribution of observations (PDO) are always computed before the update and without *peeking*

Advanced Topics
Cross-Validation. Primer

- Cross-validation
  - Divide data into *n* pieces (folds)
  - Hide $1/n^{th}$ of the data, train on the rest
  - Predict the hidden data
  - Pick next fold, repeat
  - Combine *n* prediction folds, compute accuracy metric(s)
- Stratification
  - By student – $1/n^{th}$ of the students (whole student-skill sequences)
  - By-item – $1/n^{th}$ of the items (student-skill sequences are patched)
  - Un-stratified – $1/n^{th}$ of the rows (student-skill sequences are patched)
- Implementation
  - By-student stratification – simple
  - By-item/un-stratified – save $1/n^{th}$ of observations, and mark them as unknown

Advanced Topics
Cross-Validation. Practice

- 5-fold student-stratified (default) cross-validation (rf. EM Acc.=0.870607)

```
./trainhmm -s 1.1 -d ~ -p 1 -b 1 -v 5 d/b89_train.bin m/b89_modelEMv.txt p/
b89_predictEMv.txt
...
5-fold cross-validation: LL=6932561.9857819, AIC=13879875.971564,
BIC=13989128.306512, RMSE=0.319513 (0.354041), Acc=0.870297 (0.830288)
timing: overall 206.000000 seconds, read 1.271167, fit 186.711080, predict
17.153612
```

- **10**-fold **item**-stratified cross-validation when predicting **incorrect observation**, and **save** the folds into a **file**

```
./trainhmm -s 1.1 -d ~ -p 1 -b 1 -P 1 -v 10,i,2,folds.txt,o d/
b89_train.bin m/b89_modelEMv2.txt p/b89_predictEMv2.txt
...
10-fold cross-validation: LL=19324495.2940466, AIC=38663742.588093,
BIC=38772994.923041, RMSE=0.321796 (0.357636), Acc=0.871027 (0.831588)
timing: overall 638.000000 seconds, read 1.312280, fit 618.701899,
predict 17.522869
```

- Saved folding can later be reused by using the following parameter

```
-v 10,i,2,folds.txt,i
```

Advanced Topics
Parallel Execution

- Parallel code uses Open MP
  - Individual skills are fit in N (for me N=8) recycled parallel threads
  - Prediction is done in parallel on partitioned data
- Checked-in code is in sequential state
  - Sequential code is easier to debug
  - Convert to parallel state by running `./seq2par.sh`
  - Convert back to sequential state `./par2seq.sh`
- To enable parallel execution use `-P 1` switch

```
 ./trainhmm -s 1.1 -d ~ -p 1 -b 1 -P 1 -m 1 d/
b89_train.bin m/b89_modelEM.txt p/b89_predictEM.txt
...
timing: overall 33.484420 sec, read 1.232642 sec, fit
14.389842 sec, predict 17.632139 sec
```

Compare to sequential execution

```
timing: overall 139.576931 sec, read 42.205482 sec, fit
60.997095 sec, predict 36.144902 sec
```

Advanced Topics

What if there are just too many skills/students?

- Skill/student internal index variables
  - 4 byte signed integer
  - 2,147,483,647 (2 billion)
  - Default skill/student map is non-sparse
- 2010 Big Dataset unfiltered
  - 56000 students * 3700 skills → 56000*3700*2*4≅1.54Gb of memory for PSD alone
- `hmmsclbl` with BOOST library’s sparse 2D map
  - Compact representation
  - Currently not in public code

What I Didn’t Tell You

Talk to Me If You Are Interested

- Individualized BKT (Yudelson et al., AIED 2013)
  - Parameters have a per-skill and per-student (individual) components
  - Fit using block coordinate descent (fit skills, fit students, repeat)
  - Improved model accuracy
  - More focused investigation is necessary w.r.t. interpreting parameter
- Multiplexing (under experimental investigation)
  - Using different parameters for different contexts of skill application
  - Different pLearn, pSlip, & pGuess based on response time, time since last skill was used (forgetting), etc.
- Spectral BKT (the noise fighter)

  Tomorrow, Saturday, June 27, Room 1: Salón de Actos. 10:30-11:50

  M. Falakmasir, M. Yudelson, S. Ritter and K. Koedinger
  Spectral Bayesian Knowledge Tracing. (short)

What I'd Like to add to `hmmsclbl`

Talk to Me If You Are Interested

- Multiple groupings of sequences
  - Student, Class, School – hierarchical effects
- Conjunctive BKT
  - Address multiple skill coding or other [item context] factors (in the same spirit as José and Yun did in FAST)
- Continuous observations
  - Math is already worked out
- Speed improvements
  - Optimized computation of forward-backward variables
  - NVIDIA Compute Unified Device Architecture (CUDA)
    - On this laptop: 384 GPU cores vs. 8 CPU cores

Thank You!

Project Page http://bit.ly/hmmsclbl

Acknowledgements

- All of you (for coming)
- Yun Huang (for helping me present)
- Geoffrey J. Gordon & Kenneth R. Koedinger (post-doctoral mentorship)
- Steve Ritter (for keeping the project going as part of Big Data Optimization of Carnegie Learning Cognitive Tutor)
- Mohamad Falakmasir & Young-Jin Lee (attention to detail and asking hard questions)