CS4341 Introduction to Artificial Intelligence

# HW 5 A term 2013

Team: Bohao Li (bli) and Tushar Narayan (tnarayan)

1. The outputs of the **make\_knowledge\_model** and the **fit\_parameters** functions were as follows. The output of the **generate\_data** function is in Appendix 1.

>> bnet = make\_knowledge\_model

bnet =

equiv\_class: [1 2 2 2 2 3 3 3 3 3]

dnodes: [1 2 3 4 5 6 7 8 9 10]

observed: [6 7 8 9 10]

names: {}

hidden: [1 2 3 4 5]

hidden\_bitv: [1 1 1 1 1 0 0 0 0 0]

dag: [10x10 double]

node\_sizes: [2 2 2 2 2 2 2 2 2 2]

cnodes: [1x0 double]

parents: {1x10 cell}

members\_of\_equiv\_class: {[1] [2 3 4 5] [6 7 8 9 10]}

CPD: {1x3 cell}

rep\_of\_eclass: [1 2 6]

order: [1 6 2 7 3 8 4 9 5 10]

>> fit\_parameters(bnet, sampdata)

EM iteration 1, ll = -1049.3904

EM iteration 2, ll = -300.3440

EM iteration 3, ll = -300.3440

intial params: prior: 0.631, learn: 0.355, forget: 0.000, guess: 0.997, slip: 0.224

learned params: prior: 0.540, learn: 0.164, forget: 0.000, guess: 0.172, slip: 0.080

true params: prior: 0.500, learn: 0.140, forget: 0.000, guess: 0.200, slip: 0.080

Mean Absolute Error of parameter learning: 0.0185

Thus, the learning parameters reported were:

* prior = 0.540
* learn = 0.164
* forget = 0.000
* guess = 0.172
* slip = 0.080

The mean absolute error reported was 0.0185.

Note that we edited line 5 of **generate\_data.m** from *RandStream.setDefaultStream(stream);* to *RandStream.setGlobalStream(stream);*.

1. The modified version of the **generate\_data.m** file is in Appendix 2.

Once the file was modified, the steps of problem 1 were repeated.

The learning parameters reported were:

* prior = 0.574
* learn = 0.172
* forget = 0.000
* guess = 0.117
* slip = 0.084

The mean absolute error reported was 0.0385.

1. The figure below is the result from running the experiment.

The experiment is run using the dataset containing only the values of the observables.

### 

### The x axis represents the values of ‘guess’ while the the y axis represents the values of ‘slip’.

From this graph we can observe two major convergence points, the first one is centered at about (0.12, 0.09) while the second one is centered at about (0.9, 0.8).

A general pattern can be observed that all points that are below or on the line x + y = 1.2 converge to (0.12, 0.09) while all points that are above the line x + y = 1.2 converge to (0.9, 0.8)

With a larger amount of data the average error is expected to reduce and the learned points will converge even tighter.

### Appendix 1: output of generate\_data

>> sampdata = generate\_data(bnet, num\_samples)

sampdata =

Columns 1 through 8

[2] [2] [2] [2] [2] [2] [2] [2]

[1] [2] [2] [2] [2] [2] [2] [2]

[2] [2] [2] [2] [2] [1] [2] [2]

[2] [2] [2] [2] [2] [1] [1] [2]

[1] [1] [1] [1] [1] [1] [1] [1]

[1] [1] [1] [2] [2] [1] [1] [1]

[2] [2] [2] [2] [2] [2] [2] [2]

[2] [2] [2] [2] [2] [2] [2] [2]

[1] [1] [2] [2] [2] [2] [1] [2]

[1] [1] [2] [2] [2] [1] [1] [2]

[1] [1] [1] [1] [1] [1] [1] [1]

[1] [1] [1] [1] [1] [1] [2] [2]

[1] [1] [1] [1] [1] [2] [1] [2]

[1] [1] [1] [2] [2] [2] [1] [1]

[2] [2] [2] [2] [2] [2] [2] [1]

[1] [2] [2] [2] [2] [1] [2] [2]

[2] [2] [2] [2] [2] [2] [2] [2]

[1] [1] [1] [1] [1] [1] [1] [1]

[2] [2] [2] [2] [2] [2] [2] [2]

[2] [2] [2] [2] [2] [2] [2] [2]

[2] [2] [2] [2] [2] [2] [2] [2]

[1] [1] [1] [1] [1] [1] [1] [1]

[1] [1] [2] [2] [2] [2] [1] [2]

[2] [2] [2] [2] [2] [2] [2] [2]

[1] [1] [2] [2] [2] [1] [1] [2]

[2] [2] [2] [2] [2] [2] [2] [2]

[1] [2] [2] [2] [2] [2] [2] [2]

[1] [1] [1] [2] [2] [1] [1] [2]

[1] [1] [2] [2] [2] [1] [1] [2]

[1] [2] [2] [2] [2] [1] [1] [2]

[1] [1] [1] [1] [1] [1] [1] [1]

[2] [2] [2] [2] [2] [2] [2] [2]

[1] [1] [1] [1] [1] [1] [2] [1]

[1] [1] [1] [1] [1] [1] [2] [1]

[2] [2] [2] [2] [2] [2] [2] [2]

[1] [1] [1] [1] [2] [1] [1] [1]

[1] [2] [2] [2] [2] [1] [2] [2]

[2] [2] [2] [2] [2] [2] [2] [2]

[1] [1] [1] [1] [1] [1] [1] [2]

[2] [2] [2] [2] [2] [2] [2] [1]

[1] [1] [1] [1] [1] [1] [1] [1]

[2] [2] [2] [2] [2] [2] [2] [2]

[2] [2] [2] [2] [2] [2] [2] [2]

[2] [2] [2] [2] [2] [2] [2] [2]

[1] [1] [1] [1] [1] [1] [2] [1]

[2] [2] [2] [2] [2] [1] [2] [2]

[1] [1] [1] [1] [1] [1] [1] [1]

[1] [1] [1] [1] [1] [1] [1] [1]

[2] [2] [2] [2] [2] [2] [2] [2]

[1] [1] [1] [1] [1] [1] [1] [1]

[2] [2] [2] [2] [2] [2] [2] [2]

[2] [2] [2] [2] [2] [2] [2] [2]

[1] [1] [1] [2] [2] [1] [1] [1]

[2] [2] [2] [2] [2] [2] [2] [2]

[1] [1] [1] [1] [1] [1] [1] [1]

[2] [2] [2] [2] [2] [2] [2] [1]

[2] [2] [2] [2] [2] [2] [2] [2]

[2] [2] [2] [2] [2] [2] [2] [2]

[2] [2] [2] [2] [2] [2] [2] [2]

[2] [2] [2] [2] [2] [2] [2] [2]

[2] [2] [2] [2] [2] [2] [2] [2]

[2] [2] [2] [2] [2] [2] [2] [2]

[2] [2] [2] [2] [2] [2] [2] [2]

[2] [2] [2] [2] [2] [2] [2] [2]

[2] [2] [2] [2] [2] [2] [2] [1]

[1] [1] [1] [1] [1] [1] [1] [1]

[1] [1] [1] [1] [1] [1] [2] [1]

[2] [2] [2] [2] [2] [2] [2] [2]

[2] [2] [2] [2] [2] [1] [2] [2]

[1] [2] [2] [2] [2] [1] [2] [2]

[2] [2] [2] [2] [2] [2] [2] [2]

[2] [2] [2] [2] [2] [2] [2] [2]

[1] [1] [1] [1] [1] [1] [1] [1]

[2] [2] [2] [2] [2] [2] [2] [2]

[2] [2] [2] [2] [2] [1] [1] [1]

[1] [2] [2] [2] [2] [1] [2] [1]

[2] [2] [2] [2] [2] [2] [2] [2]

[2] [2] [2] [2] [2] [1] [2] [2]

[2] [2] [2] [2] [2] [2] [1] [2]

[2] [2] [2] [2] [2] [2] [2] [1]

[2] [2] [2] [2] [2] [2] [2] [2]

[2] [2] [2] [2] [2] [2] [2] [2]

[2] [2] [2] [2] [2] [2] [2] [2]

[1] [1] [1] [1] [1] [1] [1] [1]

[1] [1] [2] [2] [2] [1] [1] [2]

[1] [1] [1] [1] [1] [2] [1] [1]

[2] [2] [2] [2] [2] [2] [2] [2]

[1] [1] [1] [1] [1] [1] [1] [1]

[1] [1] [1] [1] [2] [1] [1] [1]

[2] [2] [2] [2] [2] [2] [1] [2]

[2] [2] [2] [2] [2] [2] [2] [2]

[1] [1] [2] [2] [2] [1] [1] [2]

[2] [2] [2] [2] [2] [2] [2] [2]

[1] [1] [1] [2] [2] [1] [1] [2]

[2] [2] [2] [2] [2] [2] [2] [2]

[1] [2] [2] [2] [2] [1] [2] [2]

[2] [2] [2] [2] [2] [2] [2] [2]

[1] [1] [1] [1] [1] [2] [2] [2]

[2] [2] [2] [2] [2] [2] [2] [2]

[1] [1] [1] [2] [2] [2] [2] [1]

Columns 9 through 10

[2] [2]

[2] [2]

[2] [2]

[2] [1]

[1] [1]

[2] [2]

[2] [2]

[2] [2]

[2] [2]

[2] [2]

[1] [1]

[2] [1]

[1] [2]

[2] [2]

[2] [2]

[2] [2]

[2] [1]

[1] [1]

[2] [2]

[2] [2]

[2] [2]

[1] [1]

[2] [2]

[2] [2]

[2] [2]

[2] [2]

[2] [2]

[2] [2]

[2] [2]

[2] [2]

[2] [1]

[2] [2]

[1] [1]

[1] [1]

[2] [2]

[1] [2]

[2] [2]

[2] [2]

[1] [1]

[2] [2]

[1] [1]

[2] [2]

[2] [2]

[2] [1]

[1] [1]

[2] [1]

[1] [2]

[1] [1]

[2] [2]

[2] [1]

[2] [2]

[2] [1]

[2] [2]

[2] [2]

[1] [1]

[2] [2]

[2] [2]

[2] [2]

[2] [2]

[2] [1]

[2] [2]

[2] [2]

[2] [2]

[2] [2]

[2] [2]

[1] [1]

[1] [1]

[1] [2]

[2] [2]

[2] [2]

[2] [2]

[2] [2]

[2] [1]

[2] [2]

[2] [2]

[1] [2]

[2] [2]

[2] [2]

[2] [2]

[2] [2]

[2] [2]

[2] [2]

[2] [2]

[1] [1]

[2] [2]

[1] [1]

[2] [2]

[1] [1]

[1] [2]

[2] [2]

[2] [2]

[2] [2]

[2] [2]

[2] [2]

[2] [2]

[2] [1]

[2] [2]

[1] [1]

[2] [2]

[2] [2]

### 

### 

### Appendix 2: modified generate\_data.m

function sampdata = generate\_data(bnet,num\_samples)

% set random seed for reproducibility

stream = RandStream('mt19937ar');

RandStream.setGlobalStream(stream);

N = size(bnet.dnodes,2);

sampdata = cell(num\_samples,N);

observedvalues = make\_knowledge\_model;

observedvalues = observedvalues.observed;

lowerrange = observedvalues(1);

upperrange = observedvalues(length(observedvalues));

% sample data one row at a time

for n=1:num\_samples

samp = sample\_bnet(bnet);

% keep all the sampled data

% sampdata(n,1:N) = samp(1:N);

% keep only the observed data

sampdata(n, lowerrange:upperrange) = samp(lowerrange:upperrange);

end

**Appendix 3: modified fit\_parameters.m**

function [f\_guess, f\_slip] = fit\_parameters(bnet, sampdata, guess, slip)

% values of the ground truth parameters that generate the data

t\_prior = CPD\_to\_CPT(bnet.CPD{1});

t\_prior = t\_prior(2);

t\_trans = CPD\_to\_CPT(bnet.CPD{2});

t\_learn = t\_trans(3);

t\_forget = t\_trans(2);

t\_emit = CPD\_to\_CPT(bnet.CPD{3});

t\_guess = t\_emit(3);

t\_slip = t\_emit(2);

% intial values for EM parameter learning

i\_prior = rand;

i\_learn = rand;

i\_forget = 0;

i\_guess = guess;

i\_slip = slip;

% prior

bnet.CPD{1} = tabular\_CPD(bnet, bnet.rep\_of\_eclass(1), 'CPT', [1-i\_prior i\_prior]);

% learn/forget

bnet.CPD{2} = tabular\_CPD(bnet, bnet.rep\_of\_eclass(2), 'CPT', [1-i\_learn i\_forget i\_learn 1-i\_forget]);

% guess/slip

bnet.CPD{3} = tabular\_CPD(bnet, bnet.rep\_of\_eclass(3), 'CPT', [1-i\_guess i\_slip i\_guess 1-i\_slip]);

% initialize inference engine

engine = jtree\_inf\_engine(bnet);

% max iterations for EM parameter fitting

max\_iter = 200;

% learn parameters

[bnet, LLtrace] = learn\_params\_em(engine, sampdata',max\_iter);

% values of fit parameters

f\_prior = CPD\_to\_CPT(bnet.CPD{1});

f\_prior = f\_prior(2);

f\_trans = CPD\_to\_CPT(bnet.CPD{2});

f\_learn = f\_trans(3);

f\_forget = f\_trans(2);

f\_emit = CPD\_to\_CPT(bnet.CPD{3});

f\_guess = f\_emit(3);

f\_slip = f\_emit(2);

fprintf('intial params:\t prior: %.3f, learn: %.3f, forget: %.3f, guess: %.3f, slip: %.3f\n',...

i\_prior, i\_learn, i\_forget, i\_guess, i\_slip);

fprintf('learned params:\t prior: %.3f, learn: %.3f, forget: %.3f, guess: %.3f, slip: %.3f\n',...

f\_prior, f\_learn, f\_forget, f\_guess, f\_slip);

fprintf('true params:\t prior: %.3f, learn: %.3f, forget: %.3f, guess: %.3f, slip: %.3f\n',...

t\_prior, t\_learn, t\_forget, t\_guess, t\_slip);

MAE = mean([abs(f\_prior-t\_prior) abs(f\_learn-t\_learn) abs(f\_forget-t\_forget) abs(f\_guess-t\_guess) abs(f\_slip-t\_slip)]);

fprintf('\nMean Absolute Error of parameter learning: %.4f\n',MAE);