John Jacobsen

820405580

HMM Implementation Homework

Forward Algorithm Hand Calculations

Given Model

givenA = {{0.2, 0.8}, {0.6, 0.4}};

givenB = {{0.7, 0.3}, {0.4, 0.6}};

givenPi = {1.0, 0.0};

givenObs = {"H", "H", "T"};

Initialization

1:

a1(1) = π1b1(o1)

= 1.0 \* 0.7

= 0.7

a1(2) = π2b2(o1)

= 0.0 \* 0.4

= 0.0

Induction

2:

a2(1) = [(a1(1) \* a1,1) + (a1(2) \* a2,1)] \* b1(H)

= [(0.7 \* 0.2) + (0.0 \* 0.6)] \* 0.7

= 0.098

a2(2) = [(a1(1) \* a1,2) + (a1(2) \* a2,2)] \* b2(H)

= [(0.7 \* 0.8) + (0.0 \* 0.4)] \* 0.4

= 0.224

3:

a3(1) = [(a2(1) \* a1,1) + (a2(2) \* a2,1)] \* b1(T)

= [(0.098 \* 0.2) + (0.224 \* 0.6)] \* 0.3

= 0.0462

a3(2) = [(a2(1) \* a1,2) + (a2(2) \* a2,2)] \* b2(T)

= [(0.098 \* 0.8) + (0.224 \* 0.4)] \* 0.6

= 0.1008

Termination

P(O|Lambda) = a3(1) + a3(2)

= 0.0462 + 0.1008

= 0.147

Chosen Model

chosenThreeA = {{0.7, 0.3}, {0.9, 0.1}};

chosenThreeB = {{0.5, 0.5}, {1.0, 0.0}};

chosenThreePi = {0.6, 0.4};

chosenThreeObs = {"T", "T", "H"};

Initialization

1:

a1(1) = π1b1(o1)

= 0.6 \* 0.5

= 0.3

a1(2) = π2b2(o1)

= 0.4 \* 1.0

= 0.0

Induction

2:

a2(1) = [(a1(1) \* a1,1) + (a1(2) \* a2,1)] \* b1(T)

= [(0.3 \* 0.7) + (0.0 \* 0.9)] \* 0.5

= 0.105

a2(2) = [(a1(1) \* a1,2) + (a1(2) \* a2,2)] \* b2(T)

= [(0.3 \* 0.3) + (0.0 \* 0.1)] \* 0.0

= 0.0

3:

a3(1) = [(a2(1) \* a1,1) + (a2(2) \* a2,1)] \* b1(H)

= [(0.105 \* 0.7) + (0.0 \* 0.9)] \* 0.5

= 0.03675

a3(2) = [(a2(1) \* a1,2) + (a2(2) \* a2,2)] \* b2(H)

= [(0.105 \* 0.3) + (0.0 \* 0.1)] \* 1.0

= 0.0315

Termination

P(O|Lambda) = a3(1) + a3(2)

= 0.03675+ 0.0315

= 0.06825

For my implementation of the forward algorithm I made a forward class in the algorithms package. This class served as the main point and was passed a model from the model class. After receiving the model, it first took the size of the model, as the program was designed to be able to handle any number of states or observations. It then called the initialization method to get the first set of results. Using these results, it called its induction method so long as the number of iterations completed was less than the number of observations. During this point it always used the results of the previous induction as the arguments for the next, as it should. Lastly, it passed the final induction results to termination to calculate the final probability, which was then returned.

Viterbi Hand Calculations

Given Model

givenA = {{0.2, 0.8}, {0.6, 0.4}};

givenB = {{0.7, 0.3}, {0.4, 0.6}};

givenPi = {1.0, 0.0};

givenObs = {"H", "H", "T"};

Initialization

1:

Delta 1(1) = π1b1(o1)

= 1.0 \* 0.7

= 0.7

Delta 1(2) = π2b2(o1)

= 0.0 \* 0.4

= 0.0

Induction

2:

Delta 2(1) = max[(Delta1(1) \* a1,1) + (Delta1(2) \* a2,1)] \* b1(H)

= max[(0.7 \* 0.2), (0.0 \* 0.6)] \* 0.7

= 0.098

Psi2(1) = 1

Delta 2(2) = max[(Delta1(1) \* a1,2), (Delta 1(2) \* a2,2)] \* b2(H)

= max[(0.7 \* 0.8), (0.0 \* 0.4)] \* 0.4

= 0.224

Psi2(2) = 1

3:

Delta 3(1) = max[(Delta2(1) \* a1,1), (Delta2(2) \* a2,1)] \* b1(T)

= max[(0.098 \* 0.2), (0.224 \* 0.6)] \* 0.3

= 0.04032

Psi3(1) = 2

Delta3(2) = max[(Delta2(1) \* a1,2) + (Delta2(2) \* a2,2)] \* b2(T)

= max[(0.098 \* 0.8) + (0.224 \* 0.4)] \* 0.6

= 0.05376

Psi3(2) = 2

Termination

Viterbi Score = max[Delta3(1), Delta3(2)]

= max[0.04032, 0.05376]

= 0.05376

q\* = 2

Backtracking

q1 = 1

q2 = 2

q3 = 2

q1\*, q2\*, q3\* = 1, 2, 2 represents the best path through the HMM

Chosen Model

chosenThreeA = {{0.7, 0.3}, {0.9, 0.1}};

chosenThreeB = {{0.5, 0.5}, {1.0, 0.0}};

chosenThreePi = {0.6, 0.4};

chosenThreeObs = {"T", "T", "H"};

Initialization

1:

Delta1(1) = π1b1(o1)

= 0.6 \* 0.5

= 0.3

Delta 1(2) = π2b2(o1)

= 0.4 \* 1.0

= 0.0

Induction

2:

Delta2(1) = max[(Delta 1(1) \* a1,1), (Delta 1(2) \* a2,1)] \* b1(T)

= max[(0.3 \* 0.7), (0.0 \* 0.9)] \* 0.5

= 0.105

Psi2(1) = 1

Delta2(2) = max[(a1(1) \* a1,2), (Delta 1(2) \* a2,2)] \* b2(T)

= max[(0.3 \* 0.3), (0.0 \* 0.1)] \* 0.0

= 0.0

Psi2(1) = 1

3:

Delta3(1) = max[(Delta 2(1) \* a1,1) + (Delta 2(2) \* a2,1)] \* b1(H)

= max[(0.105 \* 0.7), (0.0 \* 0.9)] \* 0.5

= 0.03675

Psi3(1) = 1

Delta3(2) = max[(Delta 2(1) \* a1,2), (Delta 2(2) \* a2,2)] \* b2(H)

= max[(0.105 \* 0.3), (0.0 \* 0.1)] \* 1.0

= 0.0315

Psi3(2) = 1

Termination

Viterbi Score = max[Delta3(1), Delta3(2)]

= max[0.03675, 0.0315]

= 0.03675

q\* = 1

Backtracking

q1 = 1

q2 = 1

q3 = 1

q1\*, q2\*, q3\* = 1, 1, 1 represents the best path through the HMM

My Viterbi implantation was very similar to my forward algorithm implementation. As before I used a general Viterbi method which then called submethods for initialization, induction, and termination. Throughout this process I utilized Collections frequently to make finding the maximum values easier. I didn’t actually make a backtracking method. If I were in Python I would have had the q values returned in addition to the calculated probabilities, but stupid Java won’t let you do that, so instead I had a static qVals arrayList that could be called from any method to update it.