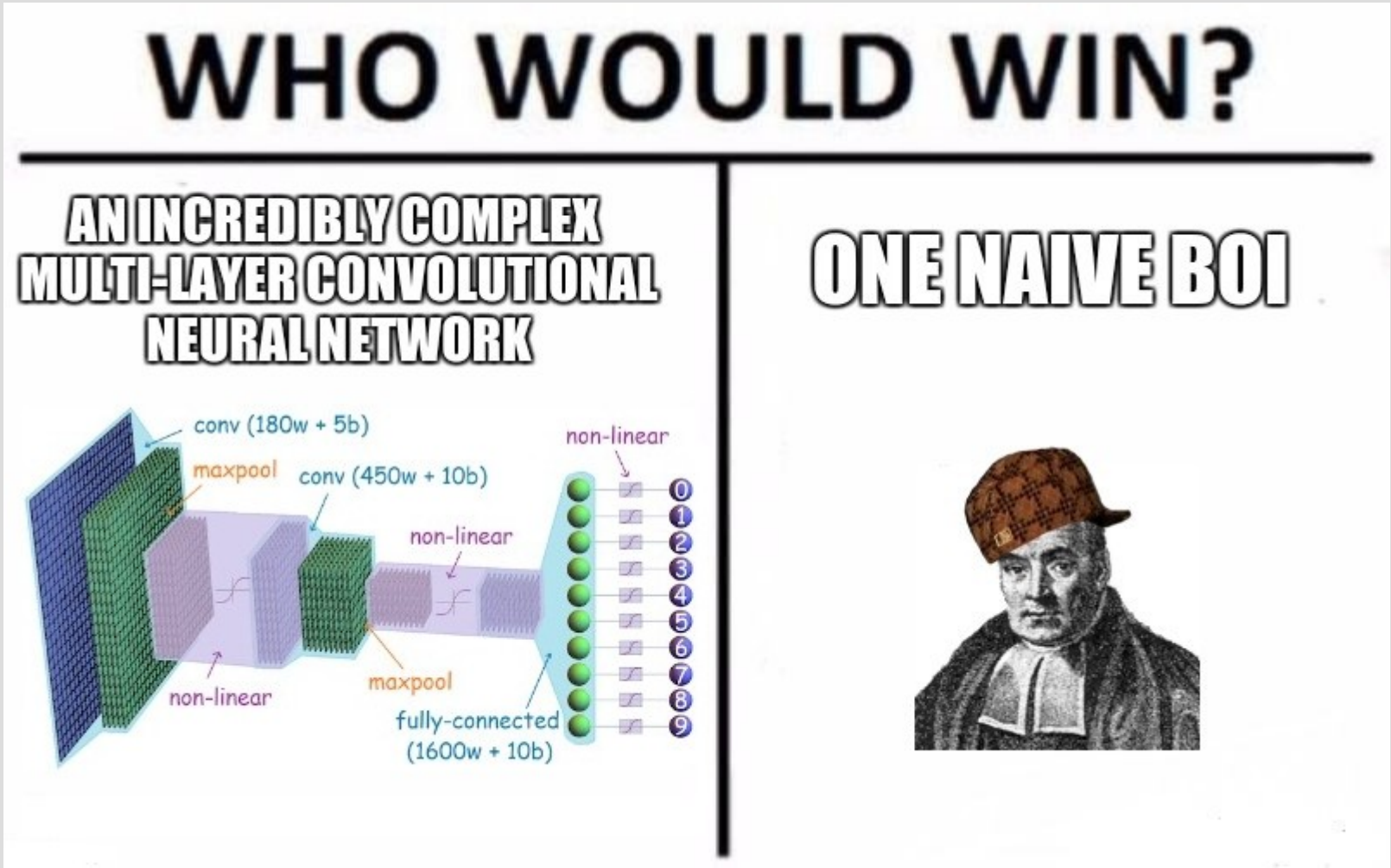
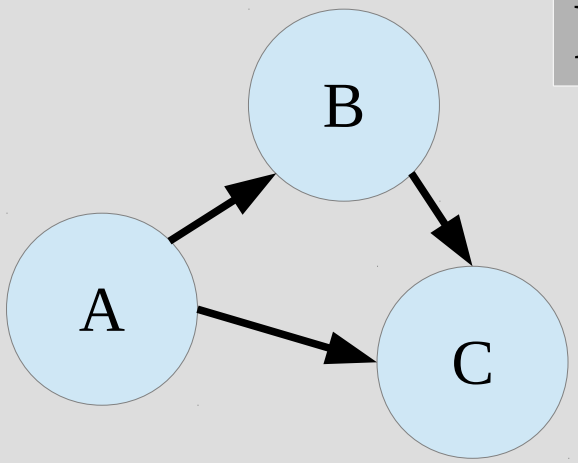
Naive Bayes & EM Algorithm (Ch. 20)



Review: Bayesian Networks

Bayes nets:
(1) Directed
(2) Acyclic
(3) Have tables: P(x | parents(x))



P(a)	0.2

P(b|a)	0.4
P(b|¬a)	0.01

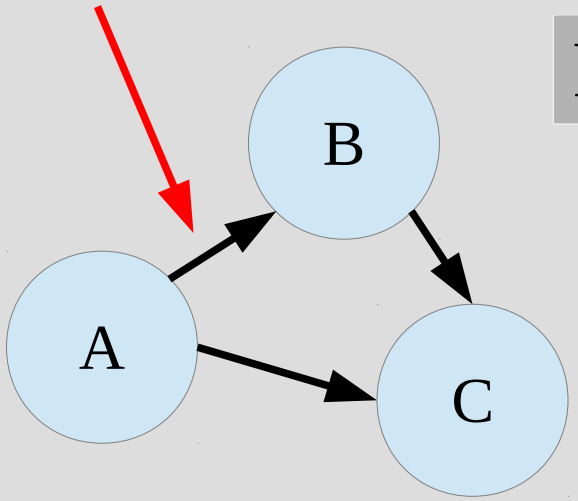
P(c|a,b)	1
P(c|a,¬b)	0.7
P(c|¬a,b)	0.3
P(c|¬a,¬b)	0

Review: Bayesian Networks

Bayes nets:
(1) Directed
(2) Acyclic
~~(3) Have tables: P(x | parents(x))~~

Now we need to learn these tables

Still need this structure & examples for learning



P(a)	???

P(b|a)	???
P(b|¬a)	???

P(c|a,b)	???
P(c|a,¬b)	???
P(c|¬a,b)	???
P(c|¬a,¬b)	???

Choose Hypothesis

We have assumed that learning data has been i.i.d. (independent and identically distributed), and we will continue to do so...

Independent(examples don't effect each other):

$$P(E_j | E_{j-1}, E_{j-2}, \ldots) = P(E_j)$$

j^{th} example

Identically distributed (no example bias):

$$P(E_j) = P(E_{j-1}) = P(E_{j-2}) = \ldots$$

Choose Hypothesis

Suppose you are registering for classes next semester and classes come in two types:

Hard classes = 25% easy HW, 75% difficult
Joke classes = 80% easy HW, 20% difficult

These are the two possible hypotheses that we are allowing for this problem

Choose Hypothesis

Hard classes = 25% easy HW, 75% difficult
Joke classes = 80% easy HW, 20% difficult

Let's say your first 3 HW are: easy, easy, diff
Assume 40% of classes in department easy

Answer me these questions three:
1. After the 3 HW, what probability hard class?
2. What is probability of 4^{th} HW being easy?
3. What if department distribution unknown?

Choose Hypothesis

1. After the 3 HW, what probability hard class?
Hard: 0.25^2*0.75*0.6 = 0.028125
Easy: 0.8^2*0.2*0.4 = 0.0512
Normalize: 64.54% easy class

2. What is probability of 4^{th} HW being easy?
0.6454*0.8 + 0.3546*0.25 = 60.5%

3. What if department distribution unknown?
0.25^2*0.75 vs 0.8^2*0.2

Choose Hypothesis

Let's put what you just did into math/Greek

For part 2, we want to find:

$$P(future|data) = \sum_i P(future|data, h_i) \cdot P(h_i|data) = \sum_i P(future|h_i) \cdot P(h_i|data)$$

prob of easy/difficult HW only depends on class type/hypothesis not previous HW

Where we can choose hypothesis using Bayes:

$$P(h_i|data) = \alpha \cdot P(data|h_i) \cdot P(h_i)$$

Using i.i.d. assumption:

$$P(data|h_i) = \prod_i P(data_i|h_i)$$

Choose Hypothesis

If you don't know the original distribution of classes in the department, you can just assume uniform distribution (i.e. 50/50) (called maximum-likelihood hypothesis)

Since you will normalize it away, you can just drop this multiplication if doing this way

Can also use $P(h_i)$ for "model complexity": more complex = smaller prob (to overfit less)

Choose Hypothesis

Another option is called maximum a posteriori which uses most likely hypothesis as truth

So instead of computing 4th HW easy as:
0.6454*0.8 + 0.3546*0.25 = 60.5%

... Since “Joke class” is more likely hypothesis it would simply estimate 4th HW easy as:
1.0*0.8 + 0*0.25 = 0.8

Choose Hypothesis

While all of this might seem like “general probability” (like we did at start of semester)

You actually were picking between two (simple) Bayesian networks:

P(HW?)	0.75

(“hard class” model)

P(HW?)	0.2

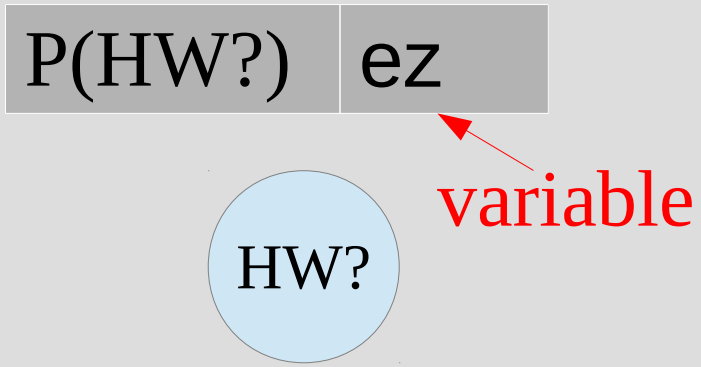
(“joke class” model)

Generating Hypothesis

However, we don't need to restrict ourselves to just two possibilities

We could allow the probability to be anything and find what works best for data

In the case where we had:
easy, easy, difficult



... what should "ez" be? (i.e. P(HW?=easy))

Generating Hypothesis

To formally find this, we just go back to our equations:

$$P(data|h_i) = \prod_i P(data_i|h_i)$$

So using our “ez” variable with 3 HW:

$$P(data|h_{ez}) = ez^2 \cdot (1 - ez)$$

could think of as h(ez), a function too

We want “ez” to best fit, so it is just an optimization problem:

$$\frac{\partial}{\partial ez}\left(ez^2 \cdot (1 - ez)\right) = 0$$

Generating Hypothesis

Let's transform this into a more easily solved problem (with same optimum)

Originally want: $\max_{ez} P(data|h_{ez})$

... same solution as: $\max_{ez} \log(P(data|h_{ez}))$
(book calls this the log likelihood or $L(data|h_{ez})$)

A logarithm might seem like a weird choice... until you do the derivative!

Generating Hypothesis

Before: $\frac{\partial}{\partial ez}\left(ez^2 \cdot (1 - ez)\right) = 0$

Now: $\frac{\partial}{\partial ez}\left(2 \cdot \log(ez) + \log(1 - ez)\right) = 0$

Much easier as don't need to use product rule, (still use chain though): $\frac{2}{ez} + (-1) \cdot \frac{1}{1-ez} = 0$

Algebra (multiply by ez*(1-ez)):

$2 \cdot (1 - ez) - ez = 0$

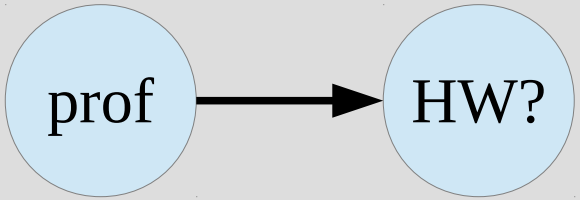
... or ez = 2/3, which makes sense as 2 of our 3 examples/data had easy homework

Generating Hypothesis

Let’s aim a bit higher, instead of:

P(HW?)	ez

... make our Bayesian network more advanced:



P(prof)	x

P(HW? | prof=nice)	y
P(HW? | prof=¬nice)	z

What changes from previous time (other than more variables to solve for)?

Generating Hypothesis

Only real difference is:

$$P(data_i|h_i) = P(prof = nice, HW? = ez|h_i)$$
$$= P(prof = nice|h_i) \cdot P(HW? = ez|prof = nice, h_i)$$

"given h_i" just means we know the variables x, y, and z

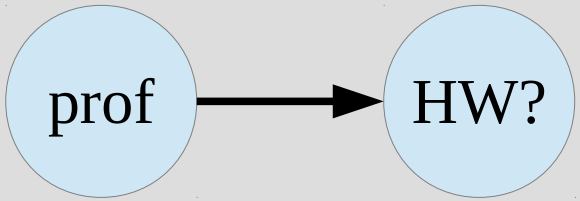
Assume you took three classes:

Class 1: prof = nice, HW = {easy, easy}

Class 2: prof = nice, HW = {easy, hard}

Class 3: prof = jerk, HW = {hard, easy, hard}

... what are x, y and z?



Generating Hypothesis

This can be extended to any size/complexity Bayesian network pretty easily

Assuming we are doing “supervised learning” (i.e. have all info in examples), you can break each example into: $\prod_{node} P(node|Parents(node))$

Then, just multiply all examples together and optimize parameters (after taking log)

Naive Bayesian Classifier

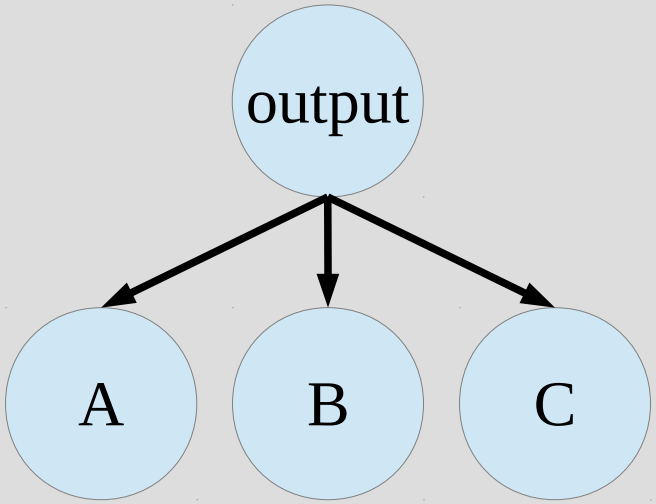
The problem with this is, how do you build a network?

You could hand-craft it or learn/search a good network on top of the current learning task

Another approach is to just assume conditional independence into the network (called a naive Bayesian classifier)

Naive Bayesian Classifier

So if you had 3 “inputs”: A, B and C
... then you would build the following bay-net:



This might seem a bit “backwards” as the output “determines” the input

Naive Bayesian Classifier

However, think back to our movie example where output=genre, A=violence, B=funny...

Does the movie genre determine if it's funny? Or does a movie being funny determine genre?

Both seem plausible and it is much more efficient to store as Bayesian network in second way (linear) than first (exponential)

Naive Bayesian Classifier

After we use examples to learn table probabilities, we can simply classify new data:

$$P(new|inputs) = \alpha \cdot P(new) \prod_i P(input_i|new)$$

"new" is root, so this is normal bay-net formula: P(n | parent(n))

The assumption that inputs are conditionally independent is a bit strong, but still works decent in practice

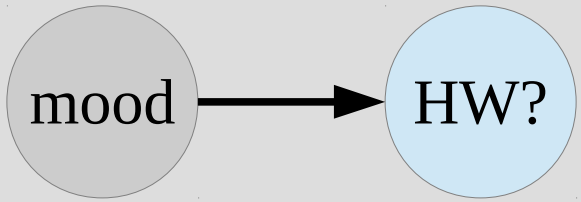
Naive Bayesian classification is also robust to noisy or missing data

EM Algorithm

We will do the same thing as before to make stuff more complicated: add hidden variables (general Bayesian networks for now)

Assume instead of having known information (the prof), the homework difficulty is based on something you can't see: my mood

Just seeing the difficulty of HW, can you estimate my mood?

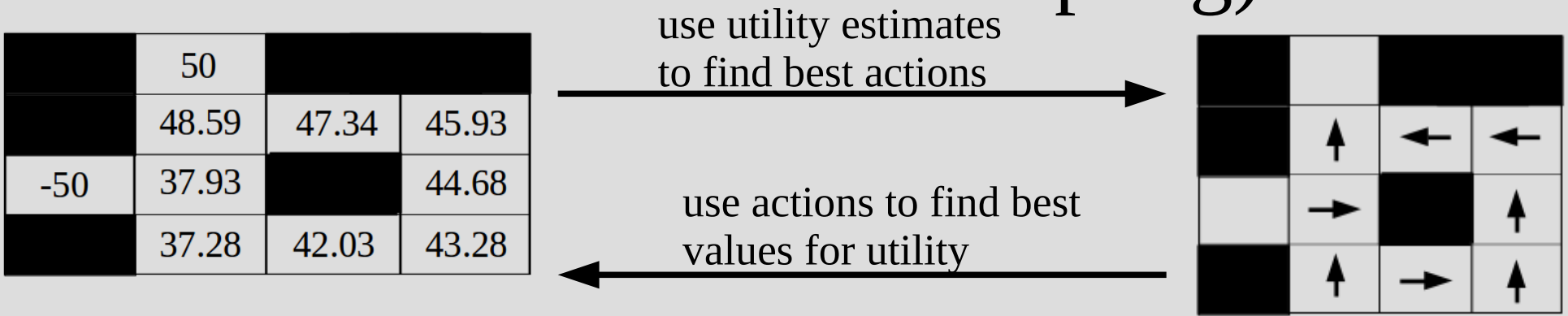


EM Algorithm

Actually, yes (to some extent)

How does this work if you can't see "mood"?
Our favorite friend: guess and iterate

We did a similar thing in policy iteration (and to a lesser extent in Gibbs sampling)



EM Algorithm

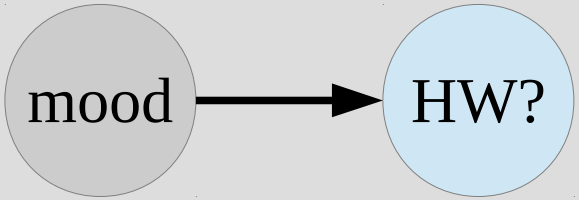
This algorithm is called the expectation-maximization algorithm and does:

1. Start with initial guess (reasonable)
2. Estimate unknowns with current parameters
3. Update parameters to best-fit unknowns
4. Repeat steps 2. and 3. until convergence

Step 2. is often called E-step (pretend know parameters and find expected outcome)
and step 3. is “M-step” (maximize param’s)

Step 1. EM Algorithm

For an example, let's go back to the original data but convert for hidden:



$P(mood) = 0.5$

$P(HW=easy \mid mood) = 0.8$

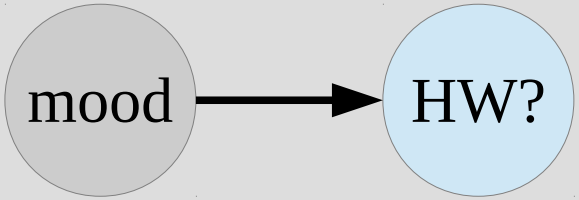
$P(HW=easy \mid \neg mood) = 0.25$

... saw 3 HW: easy, easy, hard

Step 1. is done (initialize parameter guess as the above probabilities)

Step 2. EM Algorithm

Step 2: estimate unknown



In other words we need to find P(mood|data)

In the case where all variables were visible, this would just have been:
[number of positive mood] / total

However, since we can’t see which ones, we have to estimate using parameters

Step 2. EM Algorithm

If “N” is our total, then we let “$\hat{N}$” be our estimate count, where: (Bayes rule)

$$\hat{N}(mood) = \sum_i P(mood|HW_i) = \sum_i \frac{P(HW_i|mood)\cdot P(mood)}{\sum_j P(HW_i|mood{=}j)\cdot P(mood{=}j)}$$

So in our 2 easy, 1 difficult example:

$$2 \cdot \frac{0.8\cdot 0.5}{0.8\cdot 0.5 + 0.25\cdot 0.5} + 1 \cdot \frac{0.2\cdot 0.5}{0.2\cdot 0.5 + 0.75\cdot 0.5} = 1.7343358396$$

just Bayes rule:
P(A|B) = P(A,B)/P(B)
=P(B|A)P(A)/[P(A,B) + P(~A,B)]
... A=mood, B=HW

So our new estimate is 1/N that or:

$$P(mood) = \frac{1.7343358396}{3} = 0.57811194653$$

more easy HW estimate I’m in a good mood

Step 3. EM Algorithm

Step 3: find best parameters

Now that we have P(mood) estimate,
we use it to compute table for P(HW? | mood)

Again, we have to approximate the number of homework that came from good/bad mood:

$$\hat{N}(HW? = easy, mood) = \sum_{i \in HW? = easy} P(mood|HW_i)$$

(same as before, but don't include "hards")

Step 3. EM Algorithm

So before we used this to calculate the total number of stuff caused by a good "mood":

$$2 \cdot \underbrace{\frac{0.8 \cdot 0.578}{0.8 \cdot 0.578 + 0.25 \cdot (1 - 0.578)}}_{\text{mood from "easy" HW}} + 1 \cdot \cancel{\underbrace{\frac{0.2 \cdot 0.578}{0.2 \cdot 0.578 + 0.75 \cdot (1 - 0.578)}}_{\text{mood from "hard" HW}}} = 1.628$$

$\hat{N}(mood, easy)$

Now if we want to find a new estimate for number of easy homeworks caused by mood, ignore the hard part

Step 3. EM Algorithm

This means we estimate 1.628 of the “easy” HW happened when in a good mood

We just estimated that P(mood) = 0.5781, so with 3 examples “mood” happens 1.734 (same number as original sum)

Thus: $\frac{\hat{N}(mood, HW=easy)}{\hat{N}(mood)} = \frac{1.628}{1.734} = 0.939$

an increase from our original 0.8 P(hw=easy|mood)

like P(easy|mood) = P(easy,mood)/P(mood)

Step 4. EM Algorithm

Then we go off and do a similar equation to get a new estimate for P(HW=easy | ¬mood)

After that, we just iterate the process, so with new value recompute P(mood)

Recompute: P(HW=easy | mood) and P(HW=easy | ¬mood) using new P(mood)

More complex example at bottom:
http://pages.cs.wisc.edu/~dpage/cs760/BNall.pdf

Re-recompute: P(mood)...

EM Algorithm

You can also use the EM algorithm on HMMs, but you have to group together all transitions (since they use the same probability)

$$P(X_{t+1} = j | X_t = i) \sum_t \hat{N}(X_{t+1} = j, X_t = i) / \sum_t \hat{N}(x_t = i)$$

The EM algorithm is also not limited to just all things Bayesian, and can be generalized:

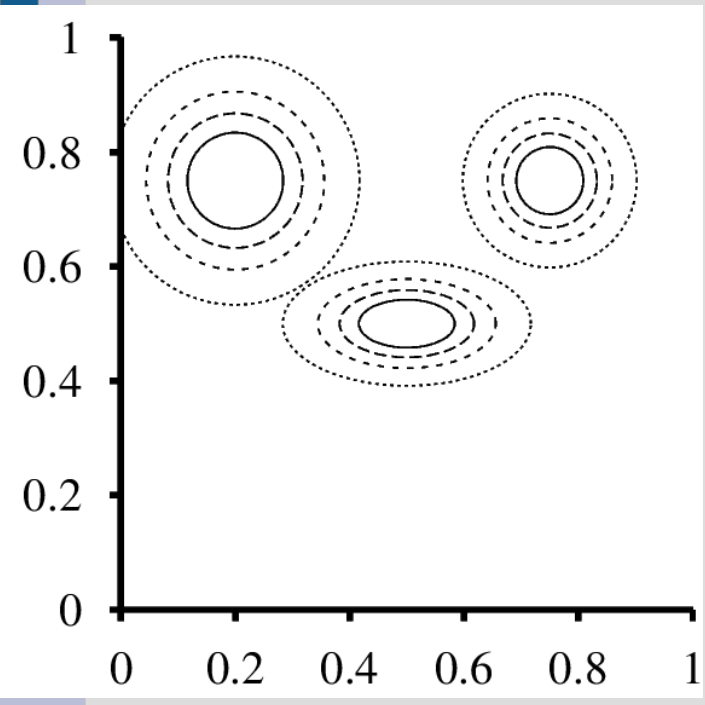
$$\theta^{(i+1)} = \arg\max_{\theta} \sum_x P(Z = z | x, \theta^{(i)}) \cdot L(x, Z = z | \theta^{(i)})$$

step 3. maximize outcomes

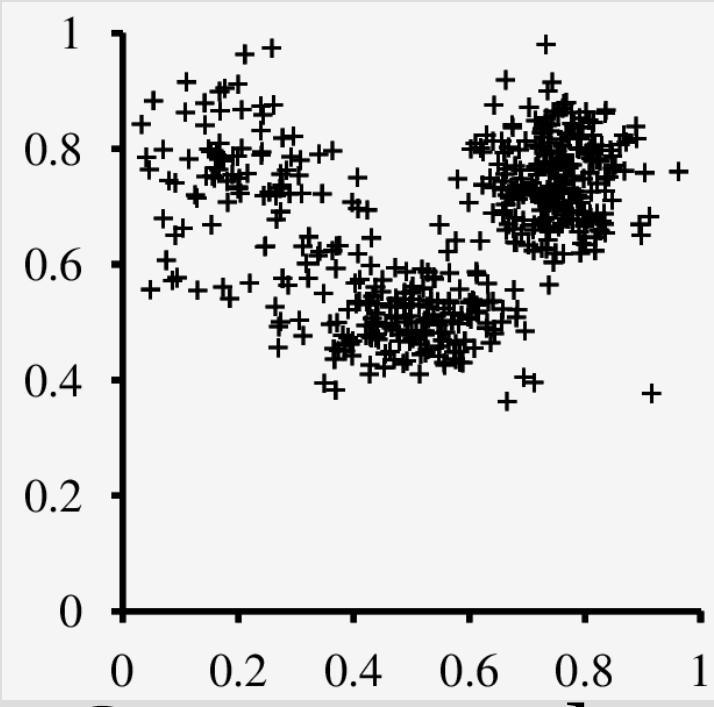
step 2. assume parameters, θ

EM Algorithm

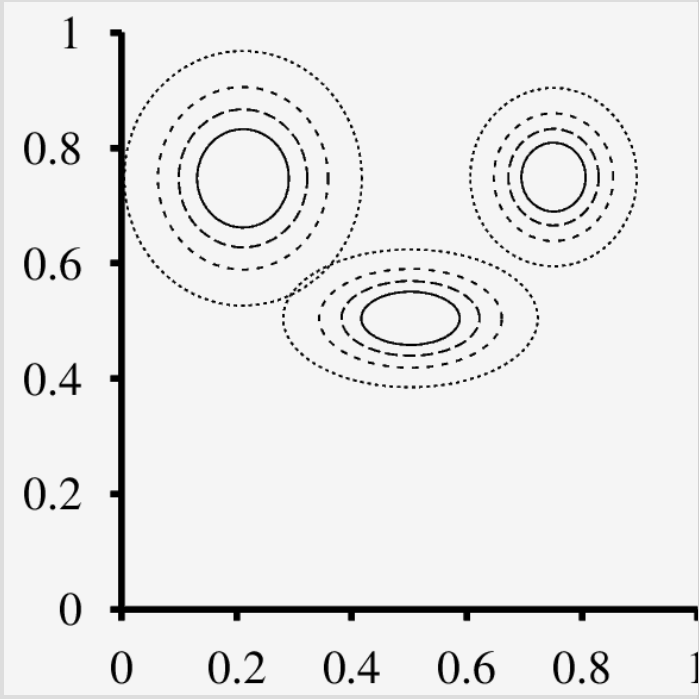
The EM algorithm is a form of gradient descent (or hill-climbing, but no α)



Real distribution



Some samples



EM algorithm reverse-eng.