



MACHINE INTELLIGENCE

Ensemble Models and Bayesian Learning

K.S.Srinivas
Department of Computer Science
and Engineering

Machine Intelligence

Unit III

Emsemble Models and Bayesian Learning

Srinivas K S
Department of Computer Science

Ensemble Learning

An ensemble method is a

- technique that combines the predictions from multiple machine learning algorithms together
- to make more accurate predictions than any individual model.

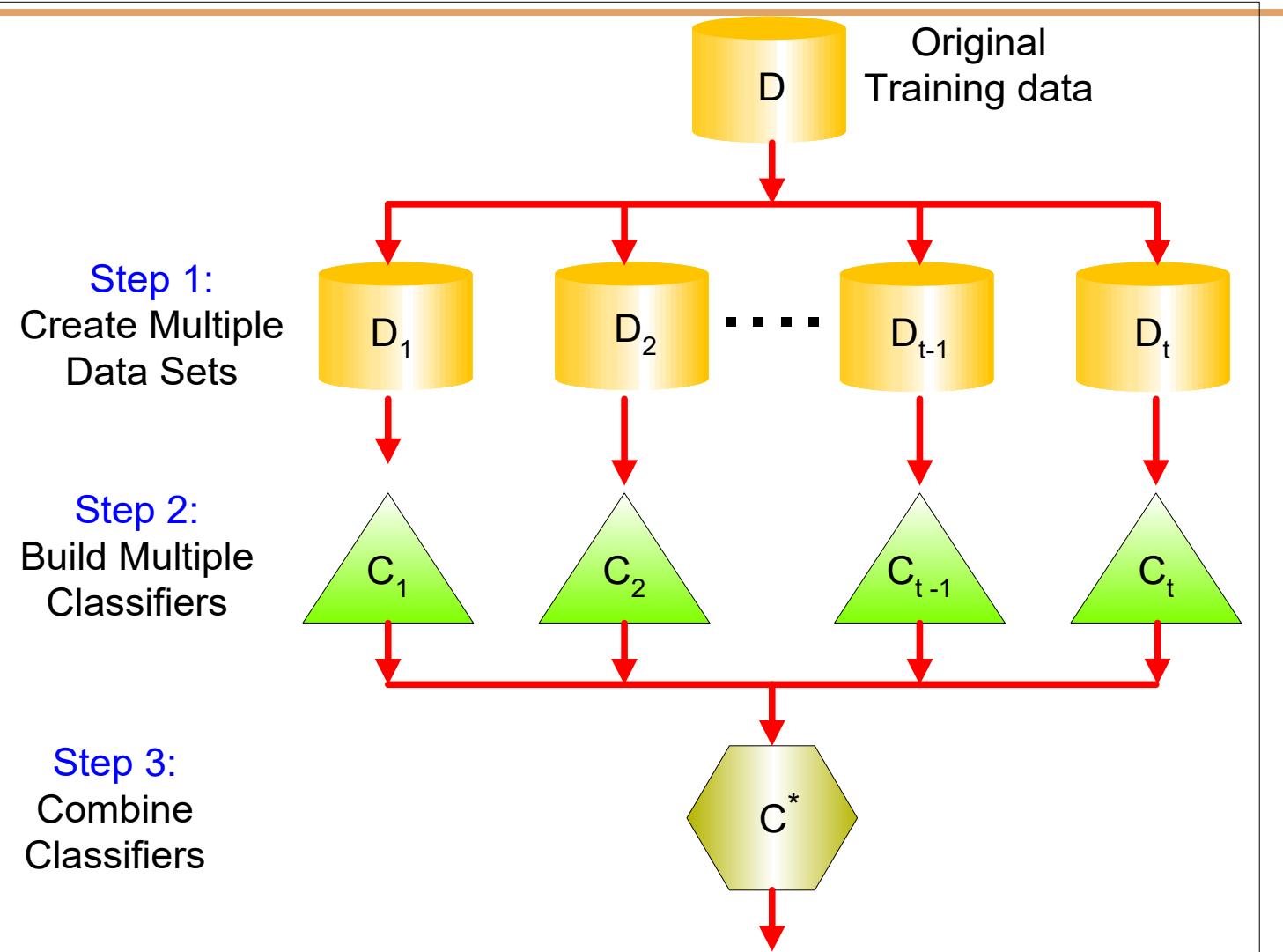
The learners that we use are usually **weak learners**

- They are among the most powerful techniques in machine learning, often outperforming other methods.
- This comes at the cost of increased algorithmic and model complexity

Ensemble Learning

- **The key idea 1** we have learners where the output is slightly better than chance i.e the accuracy is a little better than 50% but not significantly higher
- Multiple learners can be modelled using
 - Different Algorithms
 - Different Hyperparameters of the same algorithm
 - Different subsets of the training data
 - Different features of the training data
- **The key idea 2** They construct multiple, diverse predictive models from adapted versions of the training data (most often reweighted or resampled);
- They combine the predictions of these models in some way, often by simple averaging or voting (possibly weighted).

General Approach

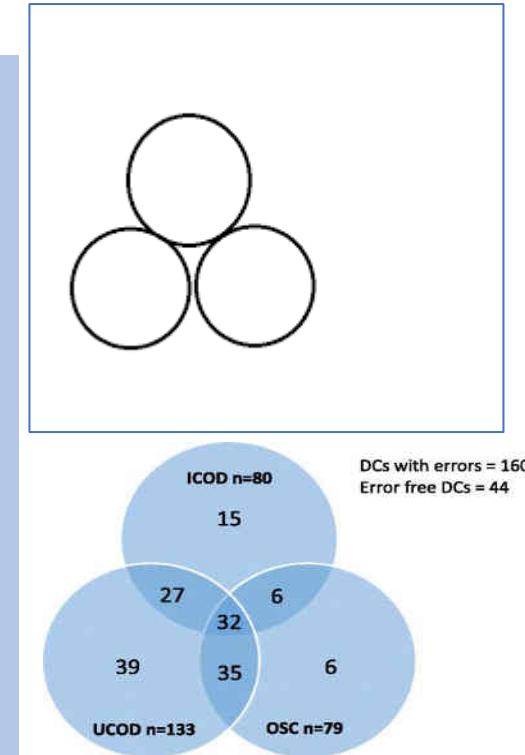


General Approach

- We have seen that decision trees earlier have a tendency to overfit i.e they have high variance
- We could offcourse prune trees but is often difficult.
- Ensemble learning ensures that the combined out of several weak learners produce a final model that has low variance
- Given a set of n independent observations Z_1, \dots, Z_n , each with variance σ^2 , the variance of the mean \bar{Z} of the observations is given by σ^2/n .
- In other words, **averaging a set of observations reduces variance.**

Intuition behind ensemble learning

- Lets take the example of the 3 learners on the right.
- Let the box be the instance space and the errors produced by each of the learners marked as circles
- Lets take an arbitrary point marked in pink and make a prediction
- Now the Red makes an error on the sample but the Green and Blue guys get it right
- The average variance is lower.



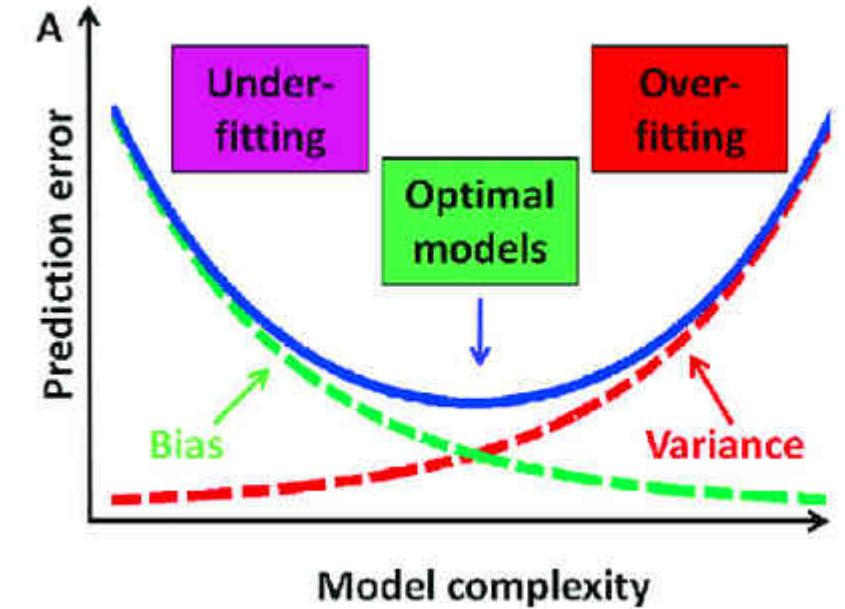
Key Concept: The errors made must be independent

1. But that's not always the case. Look at this picture .
2. The intersection areas could actually result in the voting being wrong for the pink points

Bias Variance - Recap

- A model with high bias is too simple and has low number of predictors
- Due to which it is unable to capture the underlying pattern of data.
- It pays very little attention to the training data and oversimplifies the model. This leads to high error on training and test data.
- Any model which has very large number of predictors will end up being a very complex model
 - which will deliver very accurate predictions for the training data that it has seen already but this complexity
 - makes the generalization of this model to unseen data very difficult i.e a high variance model.

Low Bias and Low Variance



Source: <https://towardsdatascience.com/holy-grail-for-bias-variance-tradeoff-overfitting-underfitting-7fad64ab5d76>

Bias and Variance

- We have seen in neural networks that if we have very large number of epochs we may overfit
- Low Bias – High flexibility (DT/ANN)
- High Variance – we give different subsets of training data we get different models
- More flexible representations(Low Bias) have High variance
- More powerful representations have high variance
- We want to have a low bias and low variance model
-

Basics models perform not so well by themselves either because they have a high bias (low degree of freedom models, for example) or because they have too much variance to be robust (high degree of freedom models, for example)

The ensemble itself will produce a model with low bias and low variance as illustrated earlier

In fact even if the individual learners have high bias the new combined learner will have a low bias

The hypothesis of the new learner may not even be in the HS of the individual learners

Many weak learners increase our confidence

- Ensemble prevent overfitting
- We don't need to worry about stopping criteria
- Lets assume we have n learners in a binary classification problem
- If all of them have an accuracy of 0.7 and predict the same class for a given instance, what would your confident be on the prediction

$$C = [1 - \{1 - A\}^n]$$

- In Reality not all learners would predict the same nor would they have the same accuracy
- Lets assume that n1 of those learners predict class1 and n2 class2
- Lets also assume that with no loss of generality that n1 > n2 , meaning that if we took a voting the prediction would always be class 1
- The probability of the class being really class1 would be

$$n_{C_R P^R (1-P)^{n-R}} = n_{C_{n_2^*}(A)} n \cdot (1 - A)^{n \perp}$$

Combining ensemble learners

- Learners can be unweighted
- Learners can be weighted wt d accuracy ,
1/variance of the learner
- For making a prediction using weights

Challenges in ensemble learners

- The critical point of ensemble learners is that they need to be independent
- **averaging a set of observations reduces variance.**
- Which can be achieved by either using different subsets or different learners .
- Shall see this in the next session – Bagging and Boosting

Types of Ensemble Methods

- Manipulate data distribution
 - Example: bagging, boosting
- Manipulate input features
 - Example: random forests
- Manipulate class labels
 - Example: error-correcting output coding

Machine Intelligence

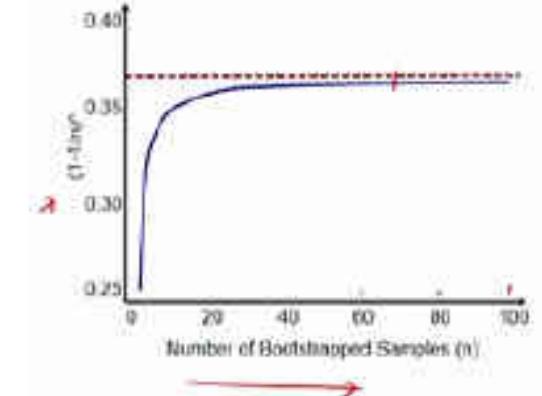
Unit III

Bagging

Srinivas K S
Department of Computer Science

Bagging

- One way of having different learners to have independent errors is by splitting the data into subsets and passing them to different learners
- But since training instances could be small we may end up with overfitting and high variance
- Instead we could randomly sample from the data set creating new data sets of the same size as the original or a very large fraction of the data set with replacement
- This method is called bootstrap aggregation or bagging
- It has been shown when we create a data set as described above we would have close to 67% of data from the original data set about 33% of original data is not selected for a sufficiently large sample size

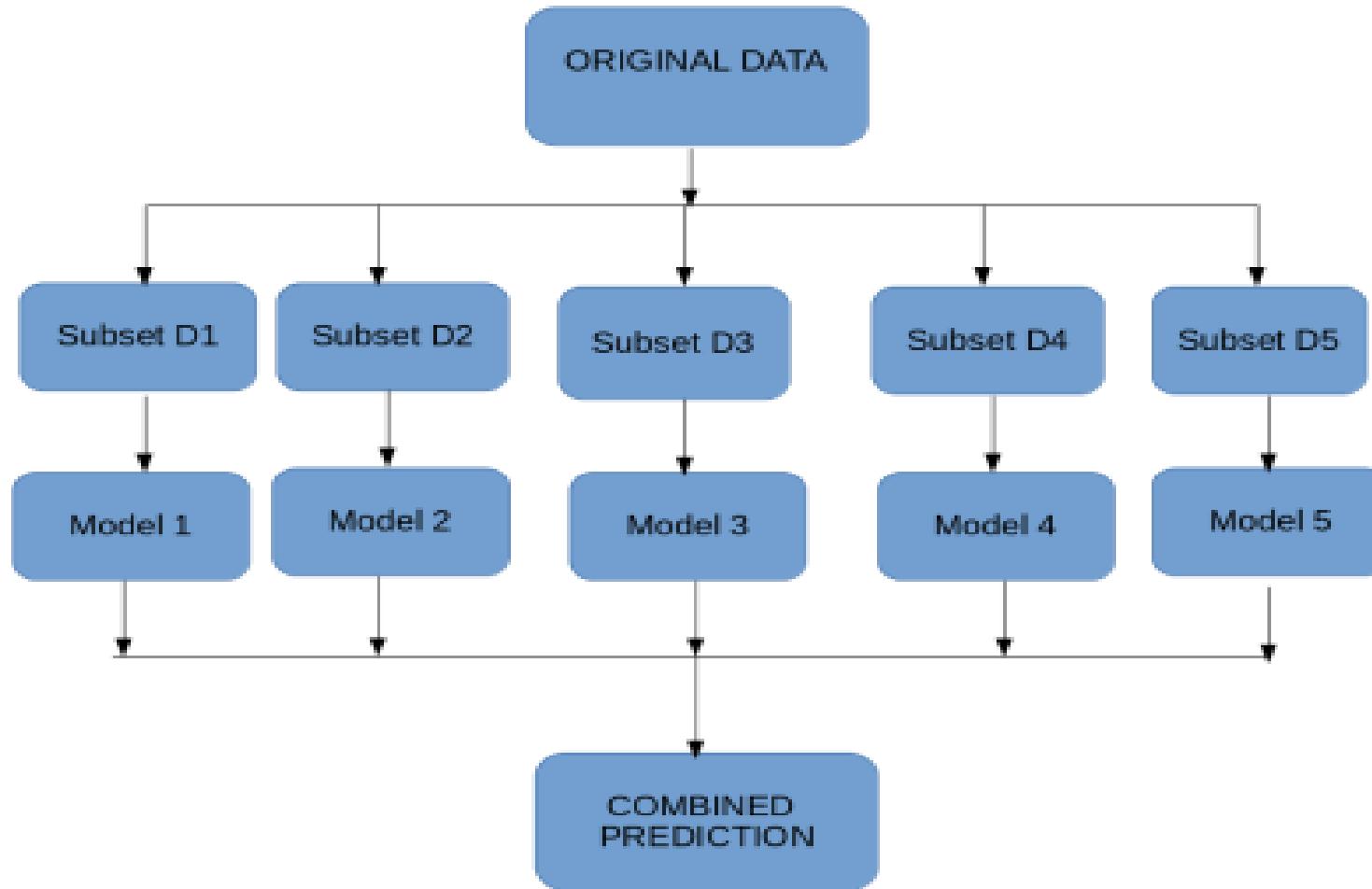


- $P(\text{Data not being selected}) = \left(1 - \frac{1}{m}\right)^n$.

Bagging

- Multiple subsets are created from the original dataset, selecting observations with replacement.
- A base model (weak model) is created on each of these subsets.
- The models run in parallel and are independent of each other.
- The final predictions are determined by combining the predictions from all the models. (Voting or averaging)

Bagging



Bagging Error Calculation

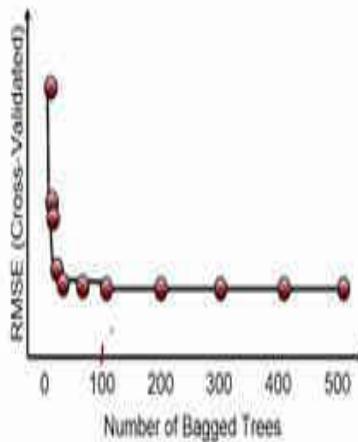
- We can do K-fold cross validation for error calculation
- Typically 1/3 of samples are left out in every subset.
- These are the out of bag examples
- All you have to do is measure the error on the unused sample in those learners that did not use them
- The average of the error from the learners gives us the error for that sample
- you can accumulate their error overall those data points that are out of bag and calculate an average
- The error calculation is close to the leave out one approach

How many learners and Advantages

- Most research has shown that about 100 learners are good enough

You can formulate these output probabilities

- Bagging performance improvements increase with more trees



- Easy to interpret and implement
- Heterogeneous input data allowed; no preprocessing required

Specific to bagging:

- Less variability than decision trees
- Can grow trees in parallel

Machine Intelligence

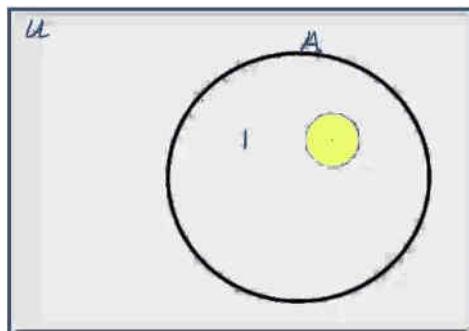
Unit III

Boosting

Srinivas K S
Department of Computer Science

Boosting - Preamble

- Let different learners have different weights (earlier stated by accuracy or by variance)
- Let each learner progressively learn from the previous learner



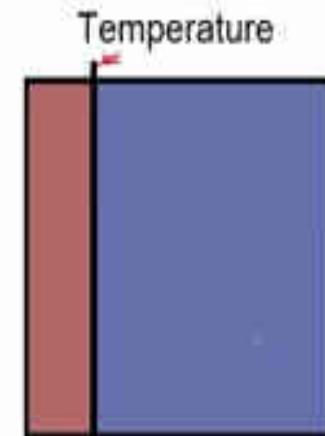
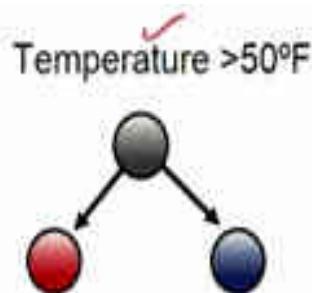
1. Let U be the instance space and A be the training data
2. Let a Hypothesis h_1 misclassify the instances in yellow
3. By learning progressively I mean the 2nd learner h_2 is told make other errors but ensure that the h_1 misclassified instances are learnt correctly
4. By giving higher weights to those sample
5. Ensure the weights of the samples that were got right reduced
6. Make sure the sum of all weights add up to 1

Boosting vs Bagging

- In the case of Bagging, any element has the same probability to appear in a new data set.
- However, for Boosting the observations are weighted and therefore some of them will take part in the new sets more often.
- One of the most popular Boosting techniques is the “Adaboost”

Boosting Overview

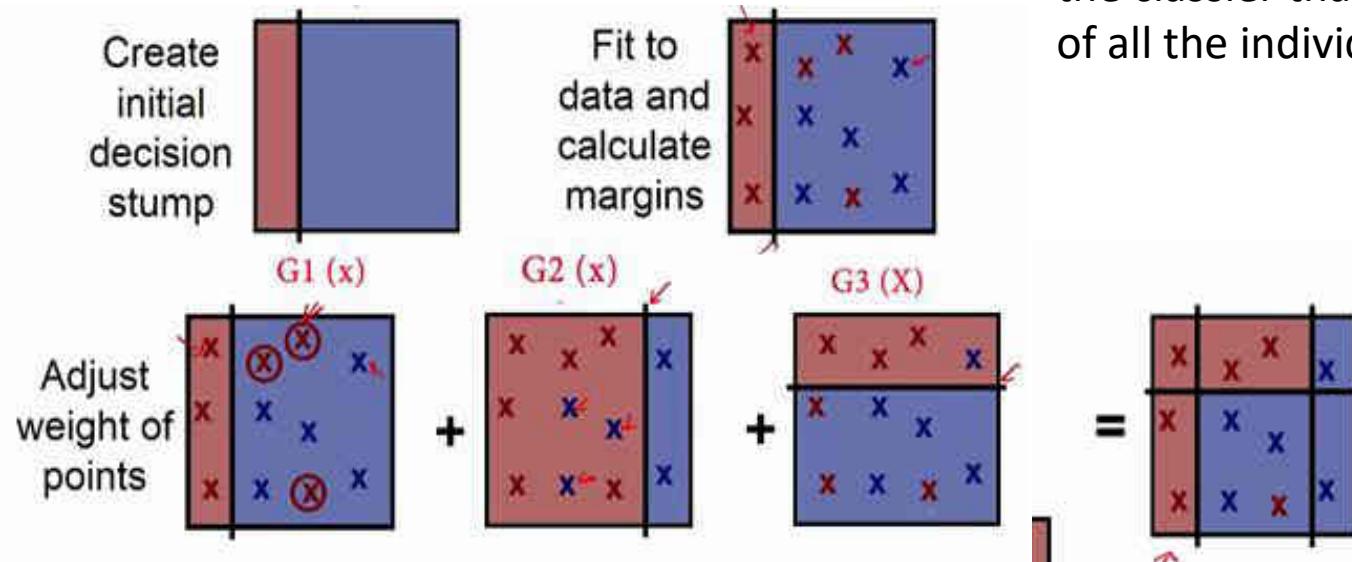
- Most example on boosting use a weak learner called a decision stump
- This has one node and based on the value does a binary split
- Boosting approach instead *learns slowly and incrementally*



Boosting Overview

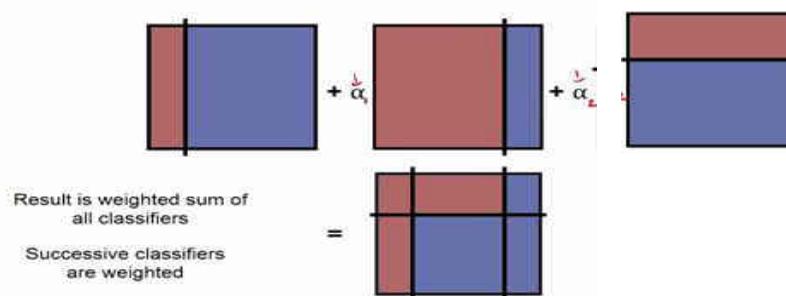
- Most example on boosting use a weak learner called a decision stump

- These red crosses on the right are basically mis-classifications
- Adjust the weights of those points.
- Further classify after assigning weights.
- Ensure the weights of the samples that were got right reduced
- the classifier that you use in the end is basically the summation of all the individual classifiers



Boosting Overview

- There is chance to overfit
- have a weighting factor here α . Have α_1 , α_2 and α_3 for each of the learners to get your final decision boundary
- These weights have some mechanism that we will see shortly which is dependent on number of errors made
- Some learners are better than others so give them higher weights





THANK YOU

Srinivas K S

Department of Computer Science & Engineering

srinivasks@pes.edu



MACHINE INTELLIGENCE

AdaBoost

K.S.Srinivas
Department of Computer Science
and Engineering

Machine Intelligence

AdaBoost

Srinivas K S
Department of Computer Science

Adaboost – The Algorithm

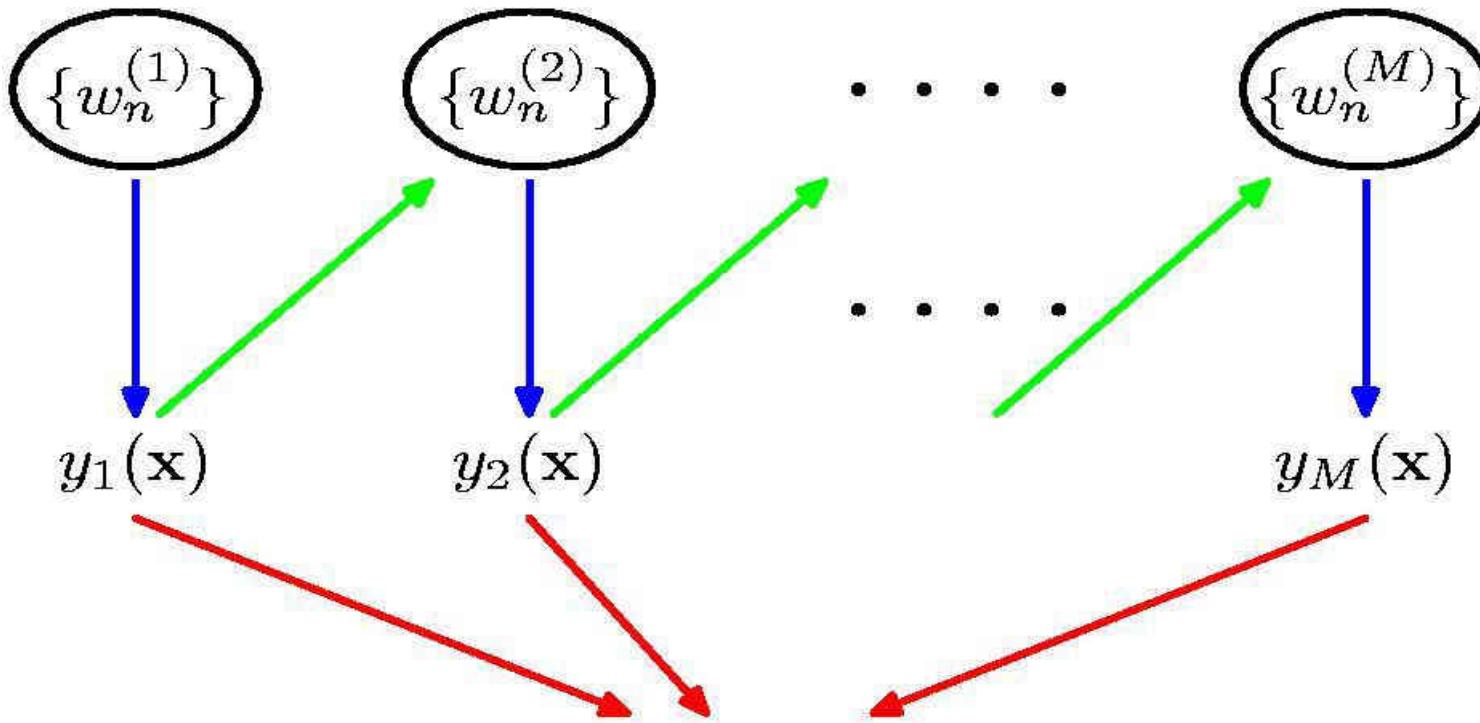
Algorithm 11.3: Boosting(D, T, \mathcal{A}) – train an ensemble of binary classifiers from reweighted training sets.

Input : data set D ; ensemble size T ; learning algorithm \mathcal{A} .

Output : weighted ensemble of models.

```
1  $w_{1t} \leftarrow 1/|D|$  for all  $x_t \in D$ ;                                // start with uniform weights
2 for  $t = 1$  to  $T$  do
3   run  $\mathcal{A}$  on  $D$  with weights  $w_{tt}$  to produce a model  $M_t$ ;
4   calculate weighted error  $\epsilon_t$ ;
5   if  $\epsilon_t \geq 1/2$  then
6     | set  $T \leftarrow t - 1$  and break
7   end
8    $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$ ;                                // confidence for this model
9    $w_{(t+1)t} \leftarrow \frac{w_{tt}}{2\epsilon_t}$  for misclassified instances  $x_f \in D$ ;      // increase weight
10   $w_{(t+1)f} \leftarrow \frac{w_{tf}}{2(1-\epsilon_t)}$  for correctly classified instances  $x_f \in D$ ; // decrease weight
11 end
12 return  $M(x) = \sum_{t=1}^T \alpha_t M_t(x)$ 
```

Schematic illustration of Boosting



$$Y_M(\mathbf{x}) = \text{sign} \left(\sum_m^M \alpha_m y_m(\mathbf{x}) \right)$$

Adaboost – Broken down in simple terms

- There is only 1 Data set being used
- Initialize each of the instances to the same normalized weight i.e $w = 1/N$, N= number of instances
- Choose the learner with the highest accuracy as your start learner say $h_1(x)$
- Run the algorithm and collect the error rate = % of misclassified examples
- Ensure that $h_2(x)$ does not classify the misclassified points of $h_1(x)$
- This perforce means that $h_2(x)$ runs after $h_1(x)$
- Continue and perform $h_3(x)$ making sure that it does get $h_2(x)$ errors right
- Finally take a vote of all the hypothesis (offcourse weighted) and state the hypothesis
- There are 2 weights – instance weights and hypothesis weights

Adaboost

- So lets walk thru the algorithm with all its gory details

- Get your instance set that you will use for training

x1	x2	Decision
2	3	true
2.1	2	true
4.5	6	true
4	3.5	false
3.5	1	false
5	7	true
5	3	false
6	5.5	true
8	6	false
8	2	false

Adaboost

- 1st weight – instance weight
- When we start the algorithm have the same weight
- So assign weights as $1/N = 1/10$ (ten sample)
- Since we are using a binary classifier convert true +1 and false to -1

x1	x2	actual	weight	weighted_actual
2	3	1	0.1	0.1
2	2	1	0.1	0.1
4	6	1	0.1	0.1
4	3	-1	0.1	-0.1
4	1	-1	0.1	-0.1
5	7	1	0.1	0.1
5	3	-1	0.1	-0.1
6	5	1	0.1	0.1
8	6	-1	0.1	-0.1
8	2	-1	0.1	-0.1

Adaboost

- Create many decision stumps such as if $x_1 < 2.1$ it to be +1 and $X_1 \geq 2.1$ as -1
- You can create many such stumps and choose the one with the lowest error rate
- Assume that the decision stump above is the best one (Not true) but for this example lets just assume
- Lets make a prediction and calculate the error rate

x1	x2	actual	weight	weighted_actual	prediction	loss	weight * loss
2	3	1	0.1	0.1	1	0	0
2	2	1	0.1	0.1	1	0	0
4	6	1	0.1	0.1	-1	1	0.1
4	3	-1	0.1	-0.1	-1	0	0
4	1	-1	0.1	-0.1	-1	0	0
5	7	1	0.1	0.1	-1	1	0.1
5	3	-1	0.1	-0.1	-1	0	0
6	5	1	0.1	0.1	-1	1	0.1
8	6	-1	0.1	-0.1	-1	0	0
8	2	-1	0.1	-0.1	-1	0	0

Adaboost

- Evaluate the error for the m^{th} classifier. Here $w_n^{(m)}$ is the weight of the n^{th} data instance in the m^{th} iteration. The Identity function:

$$I(a,b) = 1 \text{ if } a \neq b \text{ and } = 0 \text{ otherwise.}$$

$$\varepsilon_m = \frac{\sum_{n=1}^N w_n^{(m)} I(y_m(x_n) \neq t_n)}{\sum_{n=1}^N w_n^{(m)}}$$

- Then evaluate the value of the classifier using:
- $\alpha_m = \frac{1}{2} * \ln\{ (1 - \varepsilon_m)/\varepsilon_m \}$

Adaboost

- Sum of weight times loss column stores the total error
- It is 0.3 in this case.
- Compute the Stump weight . I will not prove it but the alpha is got by taking a partial derivative of the error with respect to error .

$$\alpha_s = \ln\left(\frac{1-E^s}{E^s}\right)^{\frac{1}{2}} = \frac{1}{2} \ln\left(\frac{1-E^s}{E^s}\right)$$

$$\text{alpha} = \ln[(1-\epsilon)/\epsilon] / 2 = \ln[(1 - 0.3)/0.3] / 2$$

$$\text{alpha} = 0.42$$

- Remember our final $H(x)$ is a weighted sum of individual hypothesis
- So alpha 1 is 0.42 for our first decision stump

$$H(\vec{x}) = \text{sign}(\alpha_1 h_1(\vec{x}) + \alpha_2 h_2(\vec{x}) + \dots + \alpha_s h_s(\vec{x}))$$
$$H(\vec{x}) = \text{sign}(\sum_i^s \alpha_i h_i(\vec{x}))$$

where:

$$H(\vec{x}) \in \{-1, +1\} \quad h_i(\vec{x}) \in \{-1, +1\}$$

Adaboost

- The other main funda in adaboost is that the next learner must learn from its previous learner
- The way to be doing this is by upping the weights of the instances that it got wrong and reducing the weights of the instances that it got right
- This is done thru the expression

$$w_i^{s+1} = \frac{w_i^s}{N^s} \cdot e^{-\alpha_s} \quad \text{For correctly classified samples (we } \text{reduce their weight)}$$

$$w_i^{s+1} = \frac{w_i^s}{N^s} \cdot e^{+\alpha_s} \quad \text{For incorrectly classified samples (we } \text{increase their weight)}$$

Which is take each instance and multiply it with e raised to $-\alpha$ if correctly classified and e raised to $+\alpha$ if incorrectly classified

The Term N is a normalizer is because we said that the sum of all weights must add upto 1

Lets derive N now

Adaboost



$$w_i^{s+1} = \frac{w_i^s}{N^s} \cdot e^{-\alpha_s} \quad \text{For correctly classified samples (we } \text{reduce} \text{ their weight)}$$

$$w_i^{s+1} = \frac{w_i^s}{N^s} \cdot e^{+\alpha_s} \quad \text{For incorrectly classified samples (we } \text{increase} \text{ their weight)}$$

$$N^s = \sqrt{\frac{1-E^s}{E^s}} \sum_{\text{wrong}} w_i^s + \sqrt{\frac{E^s}{1-E^s}} \sum_{\text{right}} w_i^s$$

Plug in alphas and redefine the exponential terms in terms of errors E:

$$w_i^{s+1} = \frac{w_i^s}{N^s} \sqrt{\frac{E^s}{1-E^s}} \quad \text{For correctly classified examples.}$$

$$w_i^{s+1} = \frac{w_i^s}{N^s} \sqrt{\frac{1-E^s}{E^s}} \quad \text{For incorrectly classified examples.}$$

Adaboost

$$N^s = \sqrt{\frac{1 - E^s}{E^s}} \sum_{\text{wrong}} w_i^s + \sqrt{\frac{E^s}{1 - E^s}} \sum_{\text{right}} w_i^s$$

Sum of all weights must be 1

Note that the sum of the weights over the misclassified samples is the error rate, E^s , and therefore the sum of the rest of the weights must be $1 - E^s$. Accordingly, N^s simplifies:

$$N^s = 2\sqrt{E^s(1 - E^s)}$$

Plugging this value into the formula for calculating the new weights yields:

$$w_i^{s+1} = \begin{cases} \frac{w_i^s}{2\sqrt{E^s(1 - E^s)}} \sqrt{\frac{E^s}{1 - E^s}} = \frac{1}{2} \frac{w_i^s}{(1 - E^s)} & \text{for correctly classified samples} \\ \frac{w_i^s}{2\sqrt{E^s(1 - E^s)}} \sqrt{\frac{1 - E^s}{E^s}} = \frac{1}{2} \frac{w_i^s}{E^s} & \text{for misclassified samples} \end{cases}$$

Adaboost

- We'll use alpha to update weights in the next round.
- $w_{i+1} = w_i * \text{math.exp}(-\alpha * \text{actual} * \text{prediction})$ where i refers to instance number.
- Also, sum of weights must be equal to 1. That's why, we have to normalize weight values. Dividing each weight value to sum of weights column enables normalization.

x1	x2	actual	weight	prediction	w_(i+1)	norm(w_(i+1))
2	3	1	0.1	1	0.065	0.071
2	2	1	0.1	1	0.065	0.071
4	6	-1	0.1	-1	0.153	0.167
4	3	-1	0.1	-1	0.065	0.071
4	1	-1	0.1	-1	0.065	0.071
5	7	1	0.1	-1	0.153	0.167
5	3	-1	0.1	-1	0.065	0.071
6	5	1	0.1	-1	0.153	0.167
8	6	-1	0.1	-1	0.065	0.071
8	2	-1	0.1	-1	0.065	0.071

Adaboost

- In the next round I choose $x_1 < 3.5$ as -1 and $x_1 \geq 3.5$ as +1
- Of course I use the new weights this time.

x1	x2	actual	weight	prediction	loss	weight * loss
2	3	1	0.071	-1	1	0.071
2	2	1	0.071	-1	1	0.071
4	6	1	0.167	1	0	0.000
4	3	-1	0.071	-1	0	0.000
4	1	-1	0.071	-1	0	0.000
5	7	1	0.167	1	0	0.000
5	3	-1	0.071	-1	0	0.000
6	5	1	0.167	1	0	0.000
8	6	-1	0.071	1	1	0.071
8	2	-1	0.071	-1	0	0.000

Source: <https://sefiks.com/2018/11/02/a-step-by-step-adaboost-example/>

Adaboost

- You can calculate epsilon, alpha and new weights using the same procedure
- epsilon = 0.21, alpha = 0.65
- And find weights for the next round

x1	x2	actual	weight	prediction	w_(i+1)	norm(w_(i+1))
2	3	1	0.071	-1	0.137	0.167
2	2	1	0.071	-1	0.137	0.167
4	6	1	0.167	1	0.087	0.106
4	3	-1	0.071	-1	0.037	0.045
4	1	-1	0.071	-1	0.037	0.045
5	7	1	0.167	1	0.087	0.106
5	3	-1	0.071	-1	0.037	0.045
6	5	1	0.167	1	0.087	0.106
8	6	-1	0.071	1	0.137	0.167
8	2	-1	0.071	-1	0.037	0.045

Adaboost

- At each round I update my final hypothesis

$$H(\vec{x}) = \text{sign}(\alpha_1 h_1(\vec{x}) + \alpha_2 h_2(\vec{x}) + \dots + \alpha_s h_s(\vec{x}))$$
$$H(\vec{x}) = \text{sign}(\sum_i^s \alpha_i h_i(\vec{x}))$$

- I have given a table here and the calculations for 4 rounds

round 1 alpha	round 2 alpha	round 3 alpha	round 4 alpha
0.42	0.65	0.38	1.1
round 1 prediction	round 2 prediction	round 3 prediction	round 4 prediction
1	-1	1	1
1	-1	1	1
-1	1	-1	1
-1	-1	-1	1
-1	1	-1	1
-1	-1	-1	1
-1	1	-1	1
-1	1	-1	-1
-1	-1	-1	-1

Adaboost

- At each round I update my final hypothesis

$$H(\vec{x}) = \text{sign}(\alpha_1 h_1(\vec{x}) + \alpha_2 h_2(\vec{x}) + \dots + \alpha_s h_s(\vec{x}))$$
$$H(\vec{x}) = \text{sign}(\sum_i^s \alpha_i h_i(\vec{x}))$$

- I have given a table here and the calculations for 4 rounds

round 1 alpha	round 2 alpha	round 3 alpha	round 4 alpha
0.42	0.65	0.38	1.1
round 1 prediction	round 2 prediction	round 3 prediction	round 4 prediction
1	-1	1	1
1	-1	1	1
-1	1	-1	1
-1	-1	-1	1
-1	1	-1	1
-1	-1	-1	1
-1	1	-1	1
-1	1	-1	-1
-1	-1	-1	-1

Adaboost

round 1 alpha	round 2 alpha	round 3 alpha	round 4 alpha
0.42	0.65	0.38	1.1
round 1 prediction	round 2 prediction	round 3 prediction	round 4 prediction
1	-1	1	1
1	-1	1	1
-1	1	-1	1
-1	-1	-1	1
-1	-1	-1	1
-1	1	-1	1
-1	-1	-1	1
-1	1	-1	1
-1	1	-1	-1
-1	-1	-1	-1

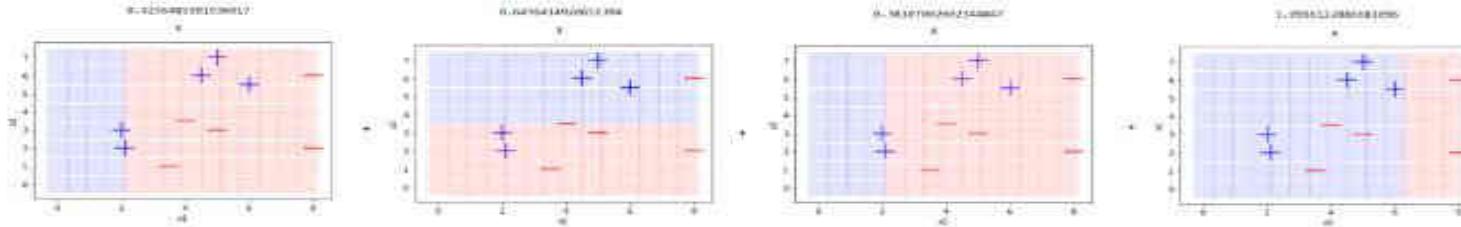
For example, prediction of the 1st instance will be

$$0.42 \times 1 + 0.65 \times (-1) + 0.38 \times 1 + 1.1 \times 1 = 1.25$$

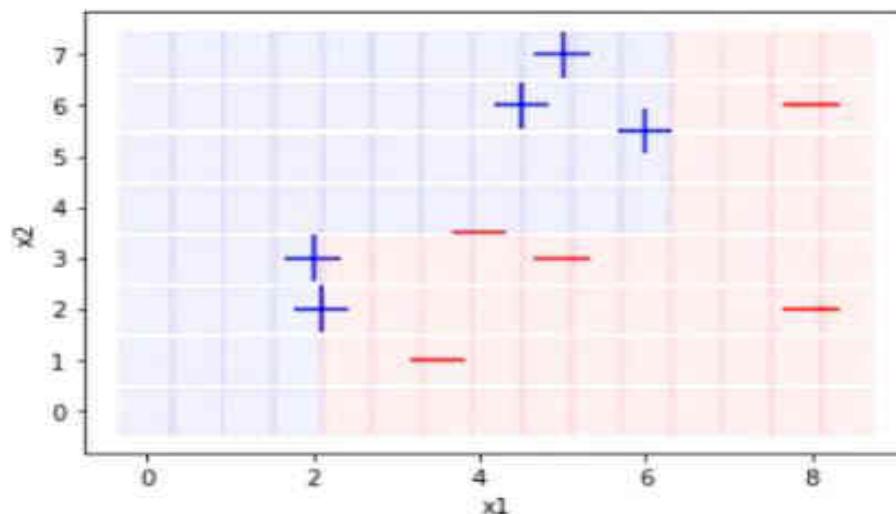
And we will apply sign function

$\text{Sign}(0.25) = +1$ aka true which is correctly classified.

Adaboost

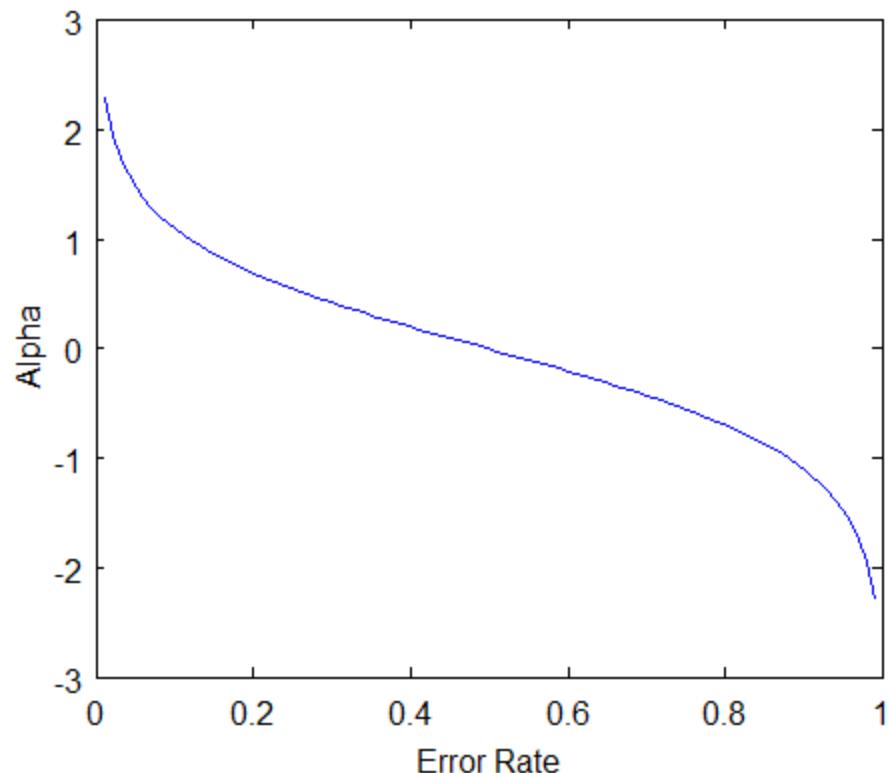


If we apply this calculation for all instances, all instances are classified correctly. In this way, you can find decision for a new instance not appearing in the train set.



Alpha vs Error

A plot of the “value” of a classifier vs the Error rate will be as follows.



Alpha vs Error

- We can see that when the error rate is 0, alpha is close to infinity (very high value for the classifier).
- If error rate = 0.5, value = 0. This makes sense because the classifier with error rate = 0.5 is as good as a “coin tosser”.
- If error rate = 1, then everything is gotten wrong by the classifier and hence value is –infinity.

Adaboost toy example

Please try a calculation of this as a home assignment

X	0	1	2	3	4	5	6	7	8	9
Y	+	+	+	-	-	-	+	+	+	-



THANK YOU

Srinivas K S

Department of Computer Science & Engineering

srinivasks@pes.edu



MACHINE INTELLIGENCE BAYESIAN LEARNING

K.S.Srinivas

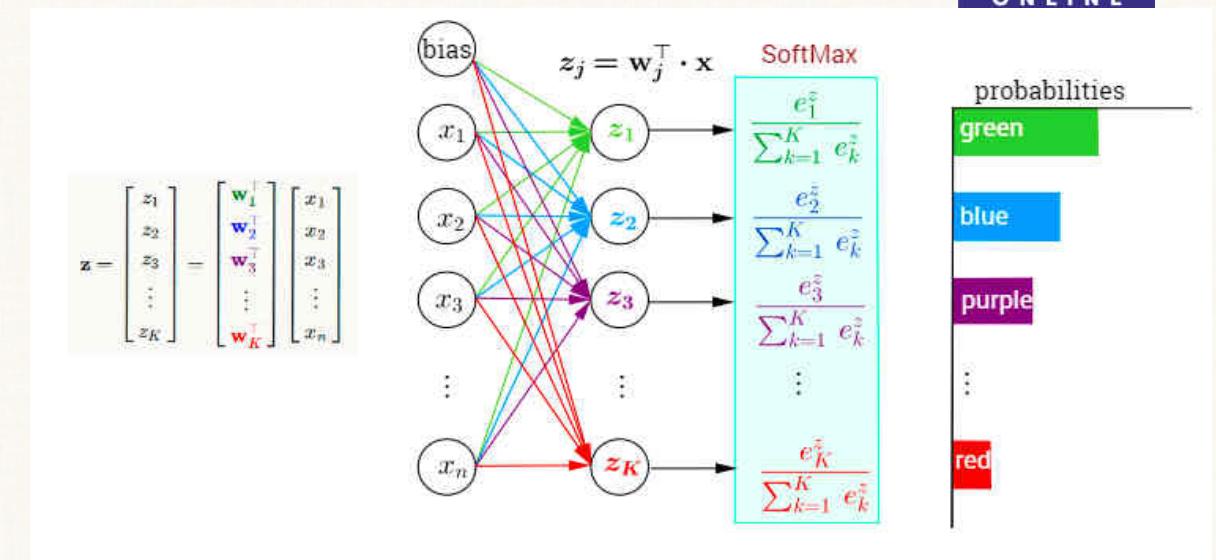
Department of Computer Science and Engineering

Probabilistic Learning



PES
UNIVERSITY
ONLINE

- Much of our dear friend the neural network outputs values are in terms of probability
- A sigmoid function is not a probability density function (PDF), as it integrates to infinity.
- However, it corresponds to the cumulative probability function of the [logistic distribution](#).
- Given that Sigmoid values lie in the interval [0,1], you can still interpret them as a confidence index.
- **SoftMax function outputs a vector that represents the probability distributions of a list of potential outcomes**



Probabilistic Learning

- Prof Winston Patrick often describes Artificial Intelligence as computational statistics
- You will see that thru the class that our intuitive error function for neural networks is indeed probabilistic
- We will see in this class how probability is used for modelling concepts
- Bayesian Probability is the notion of probability about partial beliefs
- Bayesian Estimation calculates the validity of a proposition

Probability deals with predicting the likelihood of future events

Probabilistic Learning

- Calculating the validity of a proposition is based on
 - Prior Estimates
 - New Evidence
- Based on this is the posterior estimation done
- This forms that heart of Bayes Theorem but lets step back



- Psychological research has shown that people can learn concepts from positive examples alone (

We Learn a concept from +example alone

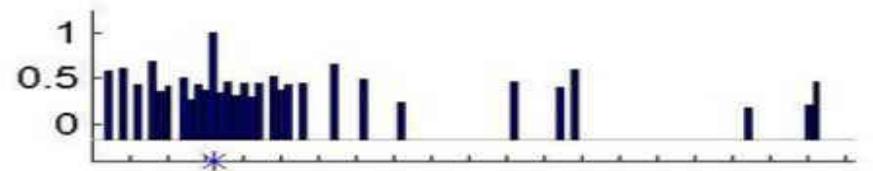
Probabilistic Learning

- We can think of learning the **meaning of a word** as equivalent to **concept learning**, which in turn is equivalent to binary classification.
- To see this, define $f(x) = 1$ if x is an example of the concept C , and $f(x) = 0$ otherwise
- Then the goal is to learn the indicator function f , which just defines which elements are in the set C .
- I am thinking of some arithmetical concept, such as:
 - Prime numbers
 - Numbers between 1 and 10
 - Even numbers
- I give you a series of randomly chosen positive examples from the chosen class

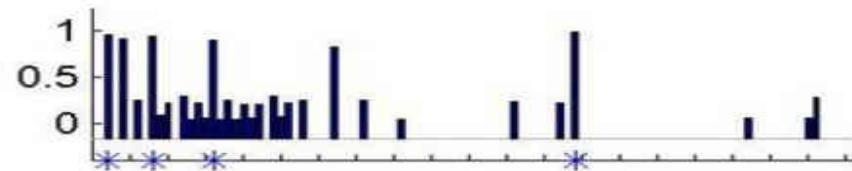
ask you whether some new test case \tilde{x} belongs to C ,
i.e., I ask you to classify \tilde{x} .

Probabilistic Learning

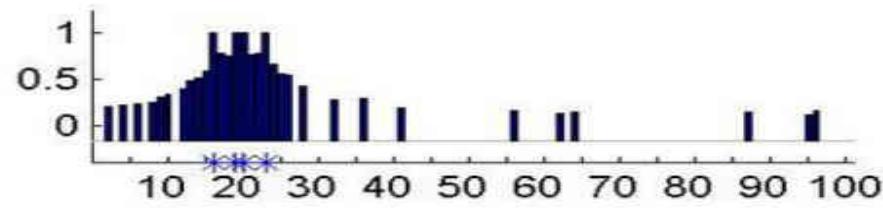
- Suppose the data set contains all integers from 1 to 100
- We are asked to learn which are the other numbers similar to this concept



$$D = \{16\}$$



$$D = \{16, 8, 2\}$$

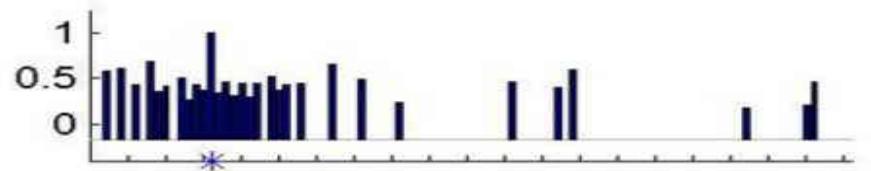


$$D = \{16, 23, 19, 20\}$$

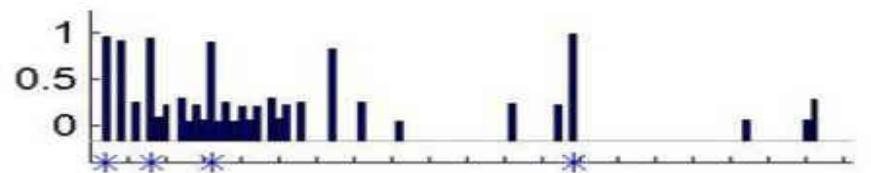
ask you whether some new test case \tilde{x} belongs to C ,
i.e., I ask you to classify \tilde{x} .

Probabilistic Learning

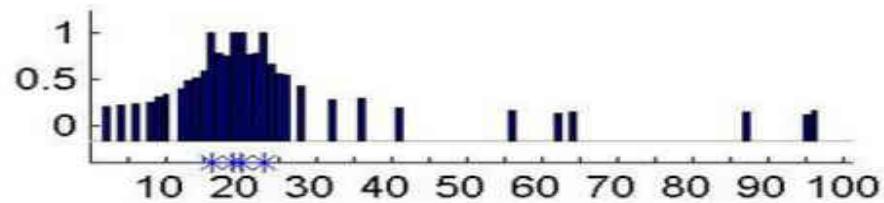
- Thus some numbers are more likely than others
- We can represent this as a probability distribution, $p(\tilde{x}/D)$, which is the probability that $\tilde{x} \in C$ given the data D for any $\tilde{x} \in \{1, \dots, 100\}$.
- This is called the **posterior predictive distribution**.



$$D = \{16\}$$



$$D = \{16, 8, 2, 64\}$$



$$D = \{16, 23, 19, 20\}$$

Probabilistic Learning

- Now suppose I tell you that 8, 2 and 64 are *also* positive examples.
- Now you may guess that the hidden concept is “powers of two”. This is an example of **induction**
- How can we explain this behavior and emulate it in a machine?
- The classic approach to induction is to suppose we have a **hypothesis space** of concepts, H , such as: odd numbers, even numbers, all numbers between 1 and 100, powers of two.
- The subset of H that is consistent with the data D is called the **version space**.
- As we see more examples, the version space shrinks and we become increasingly certain about the concept

However, the version space is not the whole story.

After seeing $D = \{16\}$, there are many consistent rules; how do you combine them to predict if $\tilde{x} \in C$?

Also, after seeing $D = \{16, 8, 2, 64\}$, why did you choose the rule “powers of two” and not, say, “all even numbers”, or “powers of two except for 32”, both of which are equally consistent with the evidence?

However, the version space is not the whole story.

After seeing $D = \{16\}$, there are many consistent rules; how do you combine them to predict if $\tilde{x} \in C$?

Also, after seeing $D = \{16, 8, 2, 64\}$, why did you choose the rule “powers of two” and not, say, “all even numbers”, or “powers of two except for 32”, both of which are equally consistent with the evidence?

Why would you say $H(\text{power of 2})$ instead of $H(\text{even number})$?

- Lets extend the set to all integers from 1...100?
- How many powers of 2 do we have - 6 and even 50
- Then $p(D/h_{\text{two}}) = 1/6$, since there are only 6 powers of two less than 100, but $p(D/h_{\text{even}}) = 1/50$, since there are 50 even numbers.
- So the likelihood that $h = h_{\text{two}}$ is higher than if $h = h_{\text{even}}$

Likelihood:

- Assume examples are sampled uniformly at random from all numbers that are consistent with the hypothesis
- Size principle: Favors smallest consistent hypotheses

$$p(\mathcal{D}|h) = \left[\frac{1}{\text{size}(h)} \right]^n = \left[\frac{1}{|h|} \right]^n$$

- Prior is the mechanism by which background knowledge can be brought to bear on a problem.
- Suppose you were told 1400, 1200, 1600, 1800 are the arithmetic output of some other numbers, would you believe 1100 to be belonging to that same concept or would you say 1183 belongs to the same concept
- Based on **prior experience**, some hypotheses are **more probable** (natural) than others

Posterior

The posterior is simply the likelihood times the prior, normalized. In this context we have

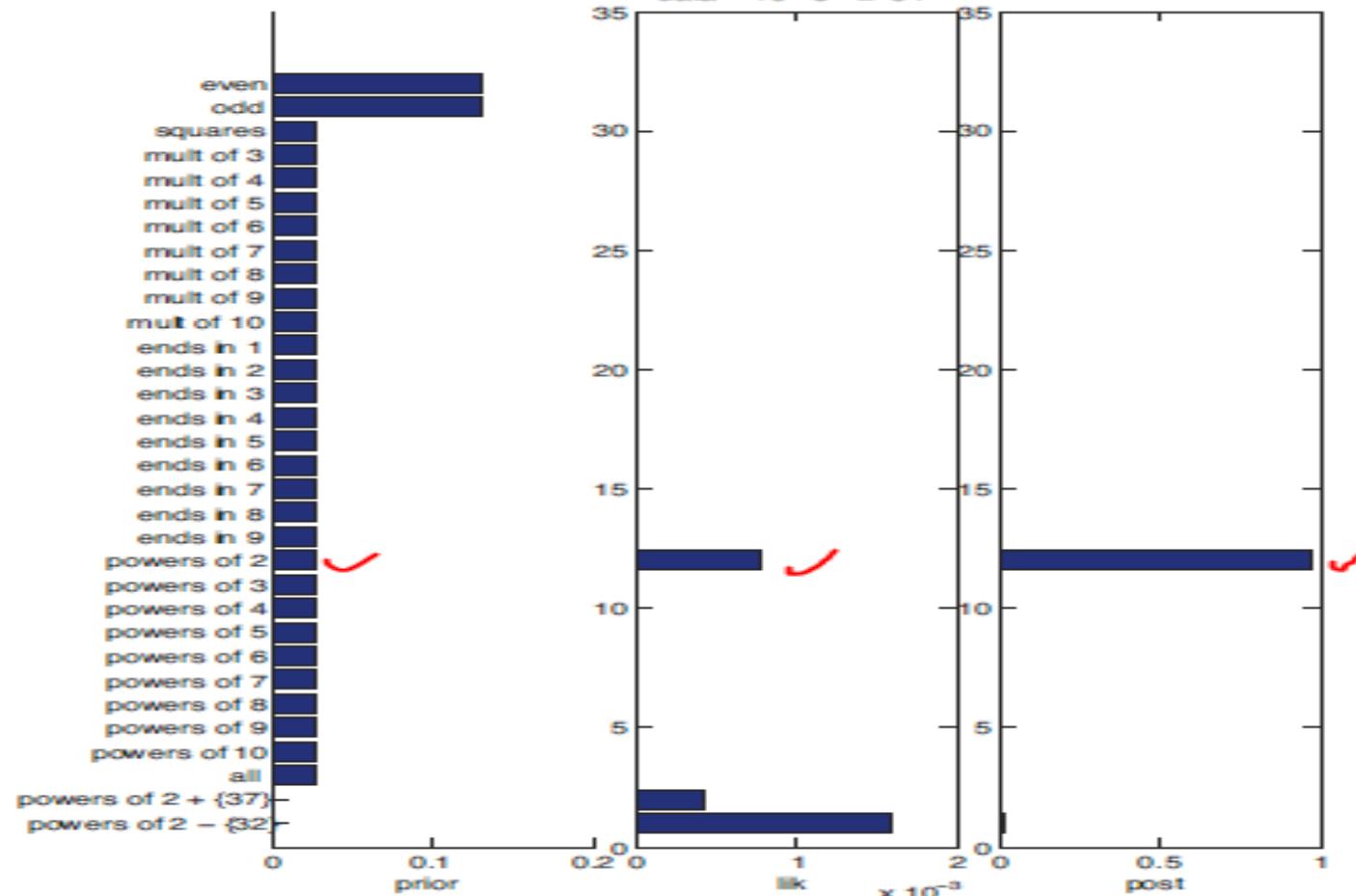
$$p(h|D) = \frac{p(D|h)p(h)}{\sum_{h' \in \mathcal{H}} p(D, h')} = \frac{p(h)\mathbb{I}(D \in h)/|h|^N}{\sum_{h' \in \mathcal{H}} p(h')\mathbb{I}(D \in h')/|h'|^N}$$

where $\mathbb{I}(D \in h)$ is 1 iff (iff and only if) all the data are in the extension of the hypothesis h

That's Bayes theorem for you

Posterior

- We see that the posterior is a combination of prior and likelihood. In the case of most of the concepts, the prior is uniform, so the posterior is proportional to the likelihood.



Posterior

The posterior is simply the likelihood times the prior, normalized. In this context we have

$$p(h|\mathcal{D}) = \frac{p(\mathcal{D}|h)p(h)}{\sum_{h' \in \mathcal{H}} p(\mathcal{D}, h')} = \frac{p(h)\mathbb{I}(\mathcal{D} \in h)/|h|^N}{\sum_{h' \in \mathcal{H}} p(h')\mathbb{I}(\mathcal{D} \in h')/|h'|^N}$$

In general, when we have enough data, the posterior $p(h|D)$ becomes peaked on a single concept, namely the MAP estimate, i.e.,

$$p(h|D) \rightarrow \delta^{\hat{h}MAP}(h)$$

where $\hat{h}MAP = \operatorname{argmax}_h p(h|D)$ is the posterior mode, and

where δ is the **Dirac measure**

defined by

$$\delta_x(A) =$$

- 1 if $x \in A$
- 0 if $x \notin A$

- As the amount of data becomes large, weak conditions on the hypothesis space and measurement process imply that

$$p(h|\mathcal{D}) \rightarrow \delta_{\hat{h}^{MAP}}(h) \quad \hat{h}^{MAP} = \operatorname{argmax}_h p(h|\mathcal{D})$$

$$\delta_x(A) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases}$$

- This is a *maximum a posteriori (MAP)* estimate:

$$\hat{h}^{MAP} = \operatorname{argmax}_h p(\mathcal{D}|h)p(h) = \operatorname{argmax}_h [\log p(\mathcal{D}|h) + \log p(h)]$$

- With a large amount of data, and/or an (almost) uniform prior, we approach the *maximum likelihood (ML)* estimate:

$$h^{mle} \triangleq \operatorname{argmax}_h p(\mathcal{D}|h) = \operatorname{argmax}_h \log p(\mathcal{D}|h)$$



MAP (“maximum a posteriori”) Learning

Bayes rule:
$$P(h | D) = \frac{P(D | h)P(h)}{P(D)}$$

Goal of learning: Find maximum a posteriori hypothesis h_{MAP} :

$$h_{\text{MAP}} = \operatorname{argmax}_{h \in H} P(h | D)$$

$$= \operatorname{argmax}_{h \in H} \frac{P(D | h)P(h)}{P(D)}$$

$$= \operatorname{argmax}_{h \in H} P(D | h)P(h)$$

This is the optimal hypothesis in the sense that no other hypothesis is more likely.

* note that $P(D)$ can be dropped, because it is a constant independent of h


$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Recap to Bayes theorem



Bayes Theorem provides a principled way for calculating a **conditional probability**.

It provides to build a **probabilistic model** to describe the relationship between **data (D)** and a **hypothesis (h)**.

Bayes Theorem for Modeling Hypotheses

In the context of classifiers, hypothesis \mathbf{h} and training data \mathbf{D} are related as

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- $P(h)$ —prior probability of hypothesis h
- $P(D)$ —prior probability of training data D
- $P(h|D)$ – posterior probability of h given D
- $P(D|h)$ – Likelihood: probability of D given h

Best hypothesis \approx most probable hypothesis



The goal of Bayesian Learning:

To locate a hypothesis that best explains the observed data.

MAP Learning: Applications

Fitting models like **linear regression** for predicting a numerical value, and **logistic regression** for binary classification can be framed and solved under the MAP probabilistic framework.

This provides an alternative to the more common **maximum likelihood estimation (MLE)** framework.

ML Hypothesis

In some cases, if every hypothesis in H is equally probable a priori i.e., $P(h_i) = P(h_j)$ for all h_i and h_j in H

$$h_{ML} = \operatorname{argmax}_{h \in H} P(D|h)$$

h_{ML} is called the “maximum likelihood hypothesis”.

P(D|h) – the likelihood of the data D given h

Summary : Classes of Hypotheses

1. MAP Hypothesis

$$h_{MAP} = \operatorname{argmax}_{h \in H} P(D|h)P(h)$$



2. ML Hypothesis

$$h_{ML} = \operatorname{argmax}_{h \in H} P(D|h)$$

When $h_{MAP} = h_{ML}$?

If $P(h_i) = P(h_j), \forall i, j$ then. $h_{MAP} = h_{ML}$

Example: Does patient have cancer or not?

A patient takes a test for cancer. The test has two outcomes: **positive** and **negative**. It is known that if the patient has cancer, the test is positive **98%** of the time. If the patient does not have cancer, the test is negative **97%** of the time. It is also known that **0.008** of the population has cancer.

The patient's test is positive.

Which is more likely: Should we diagnose a patient whose lab result is positive as having cancer?

Problem Summary

- **Hypothesis space:** **Outcome**

h_1 = Patient has cancer

$D = \{+, -\}$

h_2 = Patient does not have cancer

- **Prior Probability:** **0.008** of the population has cancer.

Thus

$$P(\text{cancer}) = P(h_1) = 0.008$$

$$P(\neg \text{cancer}) = P(h_2) = 0.992$$

- **Conditional probability:**

$$P(+) \mid h_1 = 0.98, P(-) \mid h_1 = 0.02$$

$$P(+) \mid h_2 = 0.03, P(-) \mid h_2 = 0.97$$

- **Posterior knowledge:**

Blood test is + for this patient.

What is the probability that the patient indeed has cancer?

$$h_{MAP} \equiv \operatorname{argmax}_{h \in H} P(h|D)$$

$$\equiv \operatorname{argmax}_{h \in H} \{ P(h_1|+) , \{ P(h_2|+)\}$$

$$\equiv \operatorname{argmax}_{h \in H} \{ P(+/|h_1)P(h_1), P(+/|h_2)P(h_2)\}$$

$$\equiv \operatorname{argmax}_{h \in H} \{ 0.98 * 0.08 , \quad 0.03 * 0.992\}$$

$$\equiv \operatorname{argmax}_{h \in H} \{ 0.0078, 0.0298\}$$

$$h_{MAP} \equiv h_2 (\neg \text{cancer})$$

The most probable hypothesis is h2(The patient does not have cancer)

Normalization of probabilities

The exact **posterior probabilities** can be determined by normalizing the above properties to 1

$$P(h_1|+) = \frac{0.0078}{0.0078 + 0.0298} = 0.21$$

$$P(h_2|+) = \frac{0.0298}{0.0078 + 0.0298} = 0.79$$

⇒ the result of Bayesian inference depends strongly on the prior probabilities, which must be available in order to apply the method directly



THANK YOU

Srinivas K.S

Department of Computer Science

srinivasks@pes.edu

Bayes Theorem and Concept Learning

What is the relationship between Bayes theorem and the problem of concept learning?

It can be used for designing a straightforward learning algorithm called

Brute-Force MAP LEARNING algorithm

Brute-Force MAP Learning Algorithm

The MAP hypothesis output is used to design a simple learning algorithm called “Brute force learning algorithm”

Brute-Force MAP Learning Algorithm



For each hypothesis $h \in H$, calculate the posterior probability

$$\underline{P(D|h)P(h)} \dots \dots \dots \quad (1)$$

P(D)

Output hypothesis h_{MAP} with the highest posterior probability

$$h_{MAP} = \operatorname{argmax}_{h \in H} P(h|D) \dots \dots \dots \quad (2)$$

Brute-Force MAP Learning Algorithm

Training:

Choose the hypothesis with the highest posterior probability

$$h_{MAP} = \operatorname{argmax}_{h \in H} P(h|D) = \operatorname{argmax}_{h \in H} P(D|h)P(h)$$

Testing:

Given x , compute $h_{MAP}(x)$

Drawback:

Requires to compute all probabilities $P(D|h)$ and $P(h)$.

This brute-force MAP learning algorithm may be computationally infeasible, as it requires applying the Bayes theorem to all $h \in H$.

However, this is still useful as a standard against which other concept learning approaches may be judged.

Relation to Concept learning

Consider our usual concept learning task

instance space X, hypothesis space H, training examples D

consider the Find S learning algorithm (outputs most specific hypothesis from the version space $V_{S_{H;D}}$)

What would Bayes rule produce as the MAP hypothesis ?

Does “Find S” output a MAP hypothesis ?

Relation to Concept learning

Assume fixed set of instances $\langle x_1, x_2, \dots, x_m \rangle$

Assume D is the set of classifications $D = \langle c(x_1), \dots, c(x_m) \rangle$

Choose $P(D|h)$:

Relation to Concept learning

Assume fixed set of instances $\langle x_1, x_2, \dots, x_m \rangle$

Assume D is the set of classifications $D = \langle c(x_1), \dots, c(x_m) \rangle$

Choose $P(D|h)$:

$P(D|h) = 1$ if h is consistent with D

$P(D|h) = 0$ otherwise

Choose $P(h)$ to be uniform distribution:

$$P(h) = \frac{1}{|H|} \text{ for all } h \text{ in } H$$

Brute-Force MAP Learning

Therefore, the brute force algorithm can now proceed in two ways.

$$P(h|D) = \frac{0 \cdot P(h)}{P(D)} = 0$$

If h is **consistent** with training data D

$$\begin{aligned} P(h|D) &= \frac{1 \cdot \frac{1}{|H|}}{P(D)} \\ &= \frac{\frac{1}{|H|}}{\frac{|VS_{H,D}|}{|H|}} = \frac{1}{|VS_{H,D}|} \end{aligned}$$

Thus, every consistent hypothesis has posterior probability $1/|VS_{H,D}|$ and is a MAP hypothesis.

Brute-Force MAP Learning

As data is added, certainty of hypotheses increases.

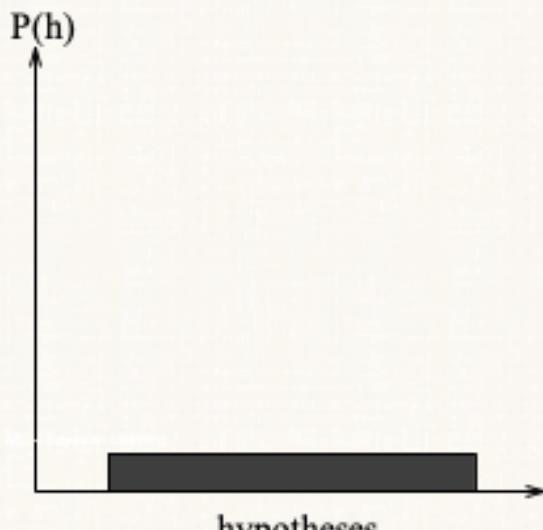


Figure (a)

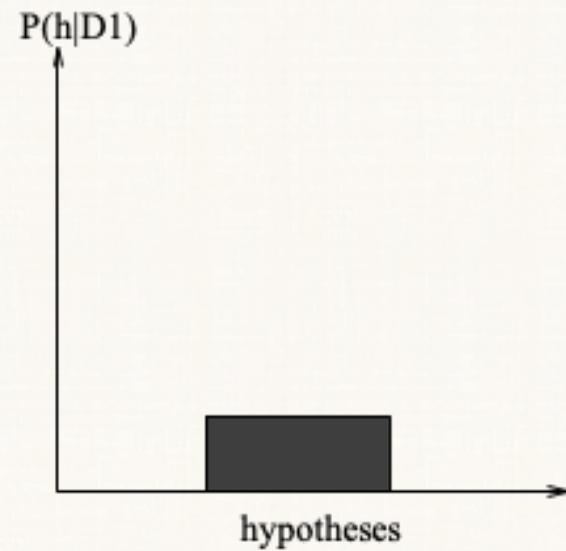


Figure (b)

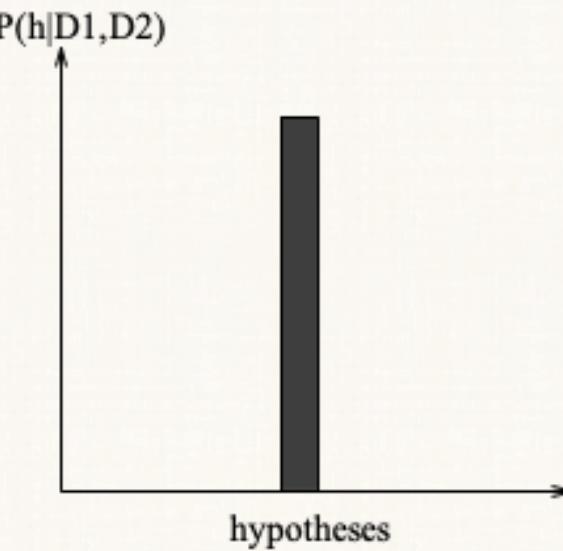


Figure (c)

evolution of probabilities

- (a) all hypotheses have the same probability
- (b) + (c) as training data accumulates, the posterior probability of inconsistent hypotheses becomes zero while the total probability summing to 1 is shared equally among the remaining consistent hypotheses

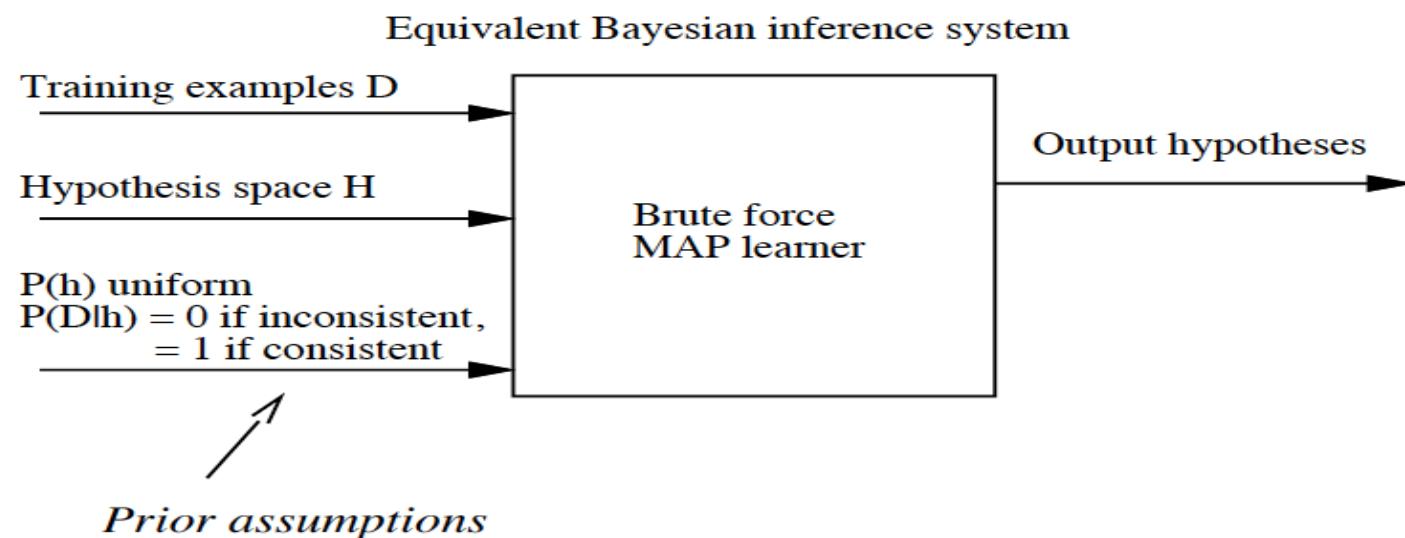
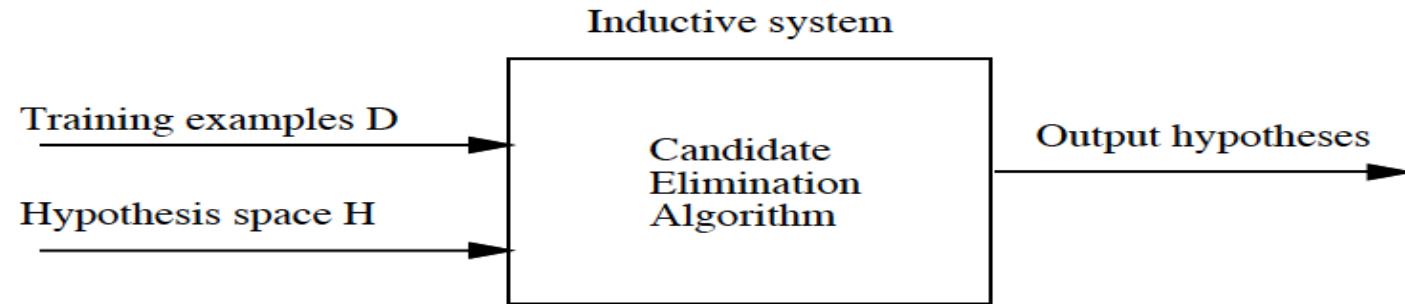
Consistent Learners

Every hypothesis consistent with D is a MAP hypothesis, if

- uniform probability over H
- target function $c \in H$
- deterministic, noise-free data

FindS will output a MAP hypothesis, even though it does not explicitly use probabilities in learning.

Bayesian interpretation of inductive bias : use Bayes theorem, define restrictions on $P(h)$ and $P(D | h)$



References

1. “Machine Learning”, Tom Mitchell, McGraw Hill Education (India), 2013.

Notations used

h (hypothesis): A single hypothesis, e.g. an instance or specific candidate model that maps inputs to outputs and can be evaluated and used to make predictions.

H (hypothesis set): A space of possible hypotheses for mapping inputs to outputs that can be searched, often constrained by the choice of the framing of the problem, the choice of model and the choice of model configuration.

D :



THANK YOU

Srinivas K.S

Department of Computer Science

srinivasks@pes.edu



MACHINE INTELLIGENCE

Maximum Likelihood and Bayes Optimal Classifier

K.S.Srinivas

Department of Computer Science and Engineering



MACHINE INTELLIGENCE

Maximum Likelihood and Bayes Optimal Classifier

K.S.Srinivas

Department of Computer Science and Engineering

MACHINE INTELLIGENCE

Maximum Likelihood and Bayes Optimal Classifier

K.S.Srinivas

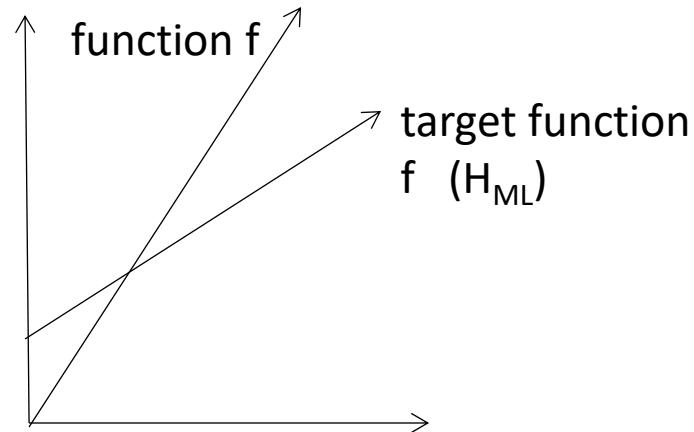
Department of Computer Science and Engineering

Maximum Likelihood and Least Squared Error Hypothesis

- Let us consider a continuous valued target function.
- Probably one of the problem faced by many learning approaches such as Neural network, Linear Regression, Polynomial Curve fitting
- A straight forward Bayesian analysis will show that under certain assumption
- any learning algorithm that minimizes output hypothesis prediction and the training data will output maximum likelihood hypothesis (H_{ML})
- we have already witnessed this???
- In neural network method the attempt to minimize the sum of the squared error over the training data

Maximum Likelihood and Least Squared Error Hypothesis

- Learner L : $X \rightarrow R$,where R represents a set of real number.
- L to learn unknown target function $f: X \rightarrow R$ drawn from H
- number of training examples be m
- We will assume our training data t be noisy
- So,each instance has training examples $\langle x_i, d_i \rangle$,where d_i is noisy training value such that $d_i = f(x_i) + e_i$,where e_i is a random variable noise that is drawn from a Gaussian Distribution with mean 0.



Maximum Likelihood and Least Squared Error Hypothesis

- Let us assume all hypothesis start with equal probability.

- Now,

$$\begin{aligned} h_{ML} &= \operatorname{argmax}_{h \in H} p(D|h) \quad \text{where } p \text{ is probability distribution} \\ &= \operatorname{argmax}_{h \in H} \prod_{i=1}^m p(d_i|h) \end{aligned}$$

- Given noise e_i obeys Normal Distribution, then d_i obeys Gaussian Distribution
- According to Gaussian Distribution

$$= \operatorname{argmax}_{h \in H} \prod_{i=1}^m \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(d_i - \mu)^2}$$

- where $\mu = h(x_i)$

$$= \operatorname{argmax}_{h \in H} \prod_{i=1}^m \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(\frac{d_i - h(x_i)}{\sigma})^2}$$

Maximum Likelihood and Least Squared Error Hypothesis

- we now apply a transformation i.e common in max likelihood calculations
- Rather than maximizing this as complicated expression we shall choose to maximize its (loss complicated) logarithm.
- The first term in this expression is a constant ,independent of \ln and can therefore be discarded yielding.

$$= \operatorname{argmax}_{h \in H} \prod_{i=1}^m \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{d_i - h(x_i)}{\sigma}\right)^2}$$

$$\operatorname{argmax}_{h \in H} \sum_{i=-1}^n \ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} (d_i - h(\kappa_i))^2$$

$$\operatorname{argmax}_{h \in H} \sum_{i=-1}^n -\frac{1}{2\sigma^2} (d_i - h(\kappa_i))^2$$

Maximum Likelihood and Least Squared Error Hypothesis

- maximizing those negative quantity is same as minimizing the positive quantity ,yielding us

$$= \operatorname{argmin}_{h \in H} \prod_{i=1}^m \frac{1}{2\sigma^2} (d_i - h(x_i))^2$$

- finally we can discard the constants ,independent og h ,giving us

$$= \operatorname{argmax}_{h \in H} \prod_{i=1}^m (d_i - \mu)^2$$

- We can also easily prove that the Maximum Likelihood hypothesis while predicting probabilities will be the same as minimizing the cross entropy loss.

$$= \operatorname{argmax}_{h \in H} \prod_{i=1}^m \frac{-1}{2\sigma^2} (d_i - h(x_i))^2$$

MACHINE INTELLIGENCE

Maximum Likelihood and Bayes Optimal Classifier

K.S.Srinivas

Department of Computer Science and Engineering

Bayes Optimal Classifier

- We have so far covered – “What is the most probable hypothesis given the training data?”.
 - The most probable classification is -ve.
 - In this case it is different from the classification generated by MAP hypothesis
- But we can now attempt to answer the question, “What is the most probable *classification* of a new instance given the training data?”
- We can answer this by using MAP hypothesis to new instance, but we can do better
- Consider a hypothesis space consisting of 3 hypothesis h_1, h_2, h_3 .
- Suppose positive probability of these hypothesis given the training data are 0.4, 0.3 and 0.3 respectively.
- Suppose a new instance x is encountered, which is classified as +ve by h_1 and -ve by h_2 and h_3 .
- Taking all hypothesis into account, the probability that x is positive is 0.4 and probability that x being negative is 0.6

Bayes Optimal Classifier

- In general the most probable classification of new instance is obtained by combined the prediction of all hypothesis ,weighted by their posterior probabilities.
- If the possible classification of the new instance can take one of any value V_j from set V, then the probability $P(V_j | D)$ that the correct classification for the new instance is V_j is

$$P(V_j | D) = \sum_{V_j \in V} P(V_j | h_j) * P(h_j | D)$$

- where $P(h_i | D)$ is the weight associated with hypothesis h_i
- The optimal classification of the new instance is the value V_j ,for which $P(V_j | D)$ is maximum i.e

$$\operatorname{argmax}_{V_j \in V} = \sum_{h_i \in H} P(V_j | h_j) * P(h_i | D)$$

Bayes Optimal Classifier

- To illustrate in terms of the above example ,the set of possible values of new instance is V

V	+ve	-ve
---	-----	-----

and h_1, h_2 and h_3 are three hypothesis

$P(h_1 D)$	0.4	$P(-ve h_1)$	0	$P(+ve h_1)$	1
$P(h_2 D)$	0.3	$P(-ve h_2)$	1	$P(+ve h_2)$	0
$P(h_3 D)$	0.3	$P(-ve h_3)$	1	$P(+ve h_3)$	0

$$\operatorname{argmax}_{v_j \in V} = \sum_{h_i \in H} P(V_j | h_j) * P(h_i | D)$$

THIS EQUATION IS CALLED Bayes Optimal Classifier

or
Bayes Optimal Learner

Therefore,

$$\sum_{h_i \in H} P(+ve|h_i).P(h_i|D) = 1 \times 0.4 + 0 \times 0.3 + 0 \times 0.3 = 0.4$$

$$\sum_{h_i \in H} P(-ve|h_i).P(h_i|D) = 0 \times 0.4 + 1 \times 0.3 + 1 \times 0.3 = 0.6$$

$$\operatorname{argmax}_{v_j \in V} \sum_{h_i \in H} P(V_j | h_j) * P(h_i | D) = -ve$$

Bayes Optimal Classifier

- This method maximizes the probability that new instance is classified correctly,given the available data,hypothesis space and prior probabilities over the hypothesis.

$$\operatorname{argmax}_{v_j \in V} = \sum_{h_i \in H} P(V_j | h_j) * P(h_i | D)$$

Gibbs Algorithm

- Bayes optimal Classifier obtains the best performance that can be achieved from the training data,it is quite costly to apply.
- The expense is due to the fact that it computes the posterior probability for every hypothesis in H and combines the prediction of each hypothesis to classify new instance
- An alternative less optimal method is the “GIBBS ALGORITHM” defined as follows
 1. Choose a hypothesis h from H at random according to posterior probability distribution of over H .
 2. use ' h ' to predict the classification of the next instance x

Note: surprisingly, it can be shown that under certain conditions the expected misclassification error for GIBBS ALGORITHM is **at most** twice the expected error of Bayes Optimal Classifier



THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701



MACHINE INTELLIGENCE

Naïve Bayes and Applications

K.S.Srinivas

Department of Computer Science and Engineering

MACHINE INTELLIGENCE

Naïve Bayes and Applications

K.S.Srinivas

Department of Computer Science and Engineering

Naive Bayes Classifier

- Highly practical Bayesian Learning method
- Comparable performance with neural network and decision tree learning.
- Naive Bayes classifier applies to learning task where each instance x is classified by the conjunction of attribute value and where the target function $f(x)$ can take any value from finite set V .
- A set of training examples of the target function is provided, and a new instance is presented in tuple of $(a_1, a_2, a_3, a_4, \dots, a_n)$
- The learner/classifier is asked to predict the target value or the classification of the new instance.
- The Bayesian approach to classify the new instance is to assign the most probable value.
- v_{MAP} given attribute values (a_1, \dots, a_n) that describe the instance is

$$v_{MAP} = \operatorname{arg\!max}_{v_j \in V} P(v_j | a_1, a_2, \dots, a_n)$$

Naive Bayes Classifier

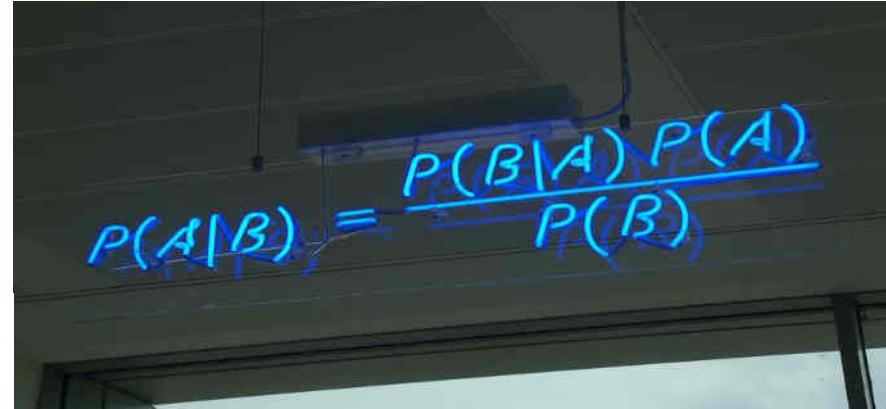
$$v_{MAP} = \operatorname{argmax}_{v_j \in V} P(v_j | a_1, a_2, \dots, a_n)$$

- Lets recall Bayes theorem
- using this

$$v_{MAP} = \operatorname{argmax}_{v_j \in V} \frac{P(a_1, a_2, \dots, a_n | v_j) P(v_j)}{P(a_1, a_2, \dots, a_n)}$$

$$= \operatorname{argmax}_{v_j \in V} P(a_1, a_2, \dots, a_n | v_j) P(v_j)$$

- It is easy to count $P(V_j)$:
- Count of the no of frequency with target value V_j . Occurring in the training set.
- Looks easy to you until ,we think what if the training data set is very very large



$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Naive Bayes Classifier

$$v_{MAP} = \operatorname{argmax}_{v_j \in V} P(v_j | a_1, a_2, \dots, a_n)$$

- Naive Bayes Classifier is based on simplifying assumption that the attribute values are conditionally independent given the target value
- i.e the assumption is that given the target value of the instance, the probability of observing the conjunction $a_1, a_2, a_3, \dots, a_n$ is just the product of the probabilities for the individual attributes

$$P(a_1, a_2, \dots, a_n | v_j) = \prod_i P(a_i | v_j)$$

- using this in our V_{MAP} equation

$$v_{NBB} = \operatorname{argmax}_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$$

Naive Bayes Classifier

$$v_{NB} = \operatorname{argmax}_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$$

- Lets note few things
 - Number of distinct $P(a_i | V_j)$ terms that must be estimated from the training set:'D' is equal to the number of distinct attribute value times the number of distinct target values
 - Based on the frequencies over the training set
 - whenever NB assumption of conditional independence is satisfied, this NB classification=MAP classification

Example- Play Tennis

- Recall the data set that we used to build a Decision Tree
- with 14 samples of target concept Play Tennis with values YES or NO
- Each day is described by attributes, (OUTLOOK, TEMPERATURE, HUMIDITY, WIND)
- use NB classifier and the training data from the table to classify the following novel instance.
 $x=(\text{Outlook}=\text{Sunny}, \text{Temp}=\text{Cool}, \text{Hum}=\text{High}, \text{Wind}=\text{strong})$

$$v_{NB} = \operatorname{argmax}_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$$

Training set- Play Tennis

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Example- Play Tennis

$$v_{NB} = \operatorname{argmax}_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$$

= $\operatorname{argmax}[$
 $P(V_j).P(\text{outlook}=\text{sunny}|v_j).P(\text{temperature}=\text{cool}|v_j).$
 $P(\text{humidity}=\text{high}|v_j).P(\text{wind}=\text{strong}|v_j).]$

V_j can be YES or NO

probabilities of the different target values can easily be estimated based on their frequency over the 14 training examples

$$P(\text{PlayTennis}=\text{yes}) = 9/14 = 0.64$$

$$P(\text{PlayTennis}=\text{no}) = 5/14 = 0.36$$

Training set- Play Tennis

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Example- Play Tennis

$$v_{NB} = \operatorname{argmax}_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$$

$$P(\text{PlayTennis} = \text{yes}) = 9/14 = 0.64$$

$$P(\text{PlayTennis} = \text{no}) = 5/14 = 0.36$$

similarly we estimate the conditional probabilities

$$P(\text{wind=strong} | \text{play tennis=YES}) = 3/9 = 0.33$$

$$P(\text{wind=strong} | \text{play tennis=NO}) = 3/5 = 0.60$$

$$P(\text{outlook=sunny} | \text{play tennis=YES}) = 2/9 = 0.2222$$

$$P(\text{outlook=sunny} | \text{play tennis=NO}) = 3/5 = 0.60$$

$$P(\text{Temp=cool} | \text{play tennis=YES}) = 3/9 = 0.333$$

$$P(\text{Temp=cool} | \text{play tennis=NO}) = 1/5 = 0.2$$

$$P(\text{humidity=high} | \text{play tennis=YES}) = 3/9 = 0.333$$

$$P(\text{humidity=high} | \text{play tennis=NO}) = 4/5 = 0.80$$

Training set- Play Tennis

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Example- Play Tennis



$$P(\text{PlayTennis} = \text{yes}) = 9/14 = 0.64 \quad P(\text{wind=strong} | \text{play tennis=YES}) = 3/9 = 0.33$$

$$P(\text{PlayTennis} = \text{no}) = 5/14 = 0.36 \quad P(\text{wind=strong} | \text{play tennis=NO}) = 3/5 = 0.60$$

$$P(\text{Temp=cool} | \text{play tennis=YES}) = 3/9 = 0.333$$

$$P(\text{outlook=sunny} | \text{play tennis=YES}) = 2/9 = 0.2222$$

$$P(\text{Temp=cool} | \text{play tennis=NO}) = 1/5 = 0.2$$

$$P(\text{outlook=sunny} | \text{play tennis=NO}) = 3/5 = 0.60$$

$$P(\text{humidity=high} | \text{play tennis=YES}) = 3/9 = 0.3333$$

$$P(\text{humidity=high} | \text{play tennis=NO}) = 4/5 = 0.80$$

$$P(\text{yes}).P(\text{cool|yes})P(\text{sunny|yes})P(\text{high|yes})P(\text{strong|yes}) = 0.64 \times 0.333 \times 0.2222 \times 0.333 \times 0.333 = 0.0211$$

$$P(\text{no}).P(\text{cool|no})P(\text{sunny|no})P(\text{high|no})P(\text{strong|no}) = 0.36 \times 0.2 \times 0.6 \times 0.8 \times 0.6 = 0.02$$

Therefore $\text{PlayTennis}(x) = \text{NO}$

Special Case

what if none of the training instances with target value v_j have attribute value a_i ? Then

$$\hat{P}(a_i|v_j) = 0, \text{ and...}$$

$$\hat{P}(v_j) \prod_i \hat{P}(a_i|v_j) = 0$$

Typical solution is Bayesian estimate for $\hat{P}(a_i|v_j)$

$$\hat{P}(a_i|v_j) \leftarrow \frac{n_c + mp}{n + m}$$

where

- n is number of training examples for which $v = v_j$,
- n_c number of examples for which $v = v_j$ and $a = a_i$
- p is prior estimate for $\hat{P}(a_i|v_j)$
- m is weight given to prior (i.e. number of “virtual” examples)

Example2-Text Classification

- consider the following data set
- the task is to classify the sentence “A very close game”as sports or not sports
- In this data set we do not have numbers but we have only text
- We need to convert all this text into numbers that we can use for calculation. HOW?????
- One solution is to use frequency of words
- Ignore word order and sentence construction
- Treat every document as a set of words it contains.
- Now the feature used in this case is the counts of words i.e(words frequency)
- Its a simplistic approach, but works surprisingly well

sentence	class
A great game	sports
The election is over	Not sports
very clean match	sports
a clean but forgettable game	sports
it was a close election	not sports

MACHINE INTELLIGENCE

Example2-Text Classification

- Now, we need to transform the probability we want to calculate into something that can be calculated using word frequencies.
- Bayes Theorem for example:

$$P(\text{sports} \mid \text{game}) = \frac{P(\text{game} \mid \text{sports}) \cdot P(\text{sports})}{P(\text{game})}$$

- since in our classifier, we are just trying to find out which category has bigger probability we can discard the divisor
- This is same for both the categories
- we can compare

$$P(\text{A very close game/sports}) \times P(\text{sports})$$

with

$$P(\text{A very close game/not sports}) \times P(\text{not sports})$$

sentence	class
A great game	sports
The election is over	Not sports
very clean match	sports
a clean but forgettable game	sports
it was a close election	not sports

MACHINE INTELLIGENCE

Example2-Text Classification

- The probabilities can be calculated:
 - count how many times the sentence 'A very close game' appears in sports category
 - Divide by the total
 - obtain $P(\text{a very close game} | \text{sports})$
- PROBLEM:** we do not have the 'sentence' in the training set
=>probability is zero
- unless every sentence appears in the training set, what we want to classify, the model wont classify

sentence	class
A great game	sports
The election is over	Not sports
very clean match	sports
a clean but forgettable game	sports
it was a close election	not sports

Example2-Text Classification

so

- we assume that every word in a sentence is independent of the other ones
- no longer we will look for entire sentences, but for only words(individual)
- i,e for a sentence “This was a funny party” is same as “funny is party was this” is same as “party funny this was a”

- we can write this as:

$$P(a \text{ very close game}) = P(a) \times P(\text{very}) \times P(\text{close}) \times P(\text{game})$$

- This enables to make the model work well,
- Now lets apply,

$$P(a \text{ very close game}/\text{sports}) =$$

$$P(a/\text{sports}) \times P(\text{very}/\text{sports}) \times P(\text{close}/\text{sports}) \times P(\text{game}/\text{sports}) \times P(\text{sports})$$

- Now as all these individual words actually show up several times in our training set, we can do our calculations

sentence	class
A great game	sports
The election is over	Not sports
very clean match	sports
a clean but forgettable game	sports
it was a close election	not sports

Example2-Text Classification

CALCULATING PROBABILITIES

- the final step is just to calculate every probability and see which one turns to be larger
- First: calculate a priori probability for each category,i.e for the sentence given in the training set

$$P(\text{sports})=3/5=0.6$$

$$P(\text{not sports})=2/5=0.4$$

- calculate $P(\text{game}/\text{sports})$: counting number of times the word game appears in the sports sample,divided by the total no of words in sports
i.e it appears twice for 11 words.

$$P(\text{game}/\text{sports})=2/11=0.18181$$

- A problem again**
- the word close does not appear in any sports ,and would lead us 0 when multiplied with other probability

sentence	class
A great game	sports
The election is over	Not sports
very clean match	sports
a clean but forgettable game	sports
it was a close election	not sports

MACHINE INTELLIGENCE

Example2-Text Classification

To resolve this we do something called Laplace smoothing

- i.e we add 1 to every count so its never zero
- to again balance this,we add the no of possible words to divisor,
- in our case the possible words are:
 $\{a,great,game,the\ election,is,over,.....election\}=14$
- Applying smoothing we get

WORD	P(word/sports)	P(word/Not sports)
a	$(2+1)/(14+11)=3/25$	$(1+1)/(9+14)=2/23$
very	$(1+1)/(14+11)=2/25$	$(0+1)/(9+14)=1/23$
close	$(0+1)/(14+11)=1/25$	$(1+1)/(9+14)=2/23$
game	$(2+1)/(14+11)=3/25$	$(0+1)/(9+14)=1/23$

sentence	class
A great game	sports
The election is over	Not sports
very clean match	sports
a clean but forgettable game	sports
it was a close election	not sports

MACHINE INTELLIGENCE

Example2-Text Classification

WORD	P(word/sports)	P(word/Not sports)
a	$(2+1)/(14+11)=3/25$	$(1+1)/(9+14)=2/23$
very	$(1+1)/(14+11)=2/25$	$(0+1)/(9+14)=1/23$
close	$(0+1)/(14+11)=1/25$	$(1+1)/(9+14)=2/23$
game	$(2+1)/(14+11)=3/25$	$(0+1)/(9+14)=1/23$

sentence	class
A great game	sports
The election is over	Not sports
very clean match	sports
a clean but forgettable game	sports
it was a close election	not sports

Now we multiply all the probabilities to see which is bigger

$$P(a/\text{sports}) \times P(\text{very}/\text{sports}) \times P(\text{close}/\text{sports}) \times P(\text{game}/\text{sports}) = 0.000027648$$

$$P(a/\text{not sports}) \times P(\text{very}/\text{not sports}) \times P(\text{close}/\text{not sports}) \times P(\text{game}/\text{not sports}) = 5.717532 \times 10^{-6}$$

by this we successfully classify it as “sports category”

- **Removing stop words:**
example: a,able,the
a very close game =>>>very close game
- **Words like election/elected are grouped together and counted as one word**
- **Using n-grams:**
instead of counting individual words we can count sequence of words
example:'clean match','close election'
- **TFIDF**
term frequency-inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.



THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701



MACHINE INTELLIGENCE EXPECTATION MAXIMIZATION

K.S.Srinivas

Department of Computer Science and Engineering



MACHINE INTELLIGENCE EXPECTATION MAXIMIZATION

K.S.Srinivas

Department of Computer Science
and Engineering

Machine Intelligence

Unit III

Expectation Maximization

Srinivas K.S
Department of Computer Science

Expectation Maximization

- Maximum likelihood estimation is an approach to **density estimation** for a dataset by **searching across probability distributions** and their **parameters**.
- It is a general and effective approach that underlies many machine learning algorithms, although it **requires that the training dataset is complete**, e.g. all relevant **interacting random variables are present**.
- Maximum likelihood becomes **intractable** if there are variables that interact with those in the dataset but were **hidden or not observed**, so-called latent variables.
- **The expectation-maximization algorithm** is an approach for performing **maximum likelihood estimation in the presence of latent variables**

It does this by f

estimating the values for the latent variables, (E)
then optimizing the model(M),

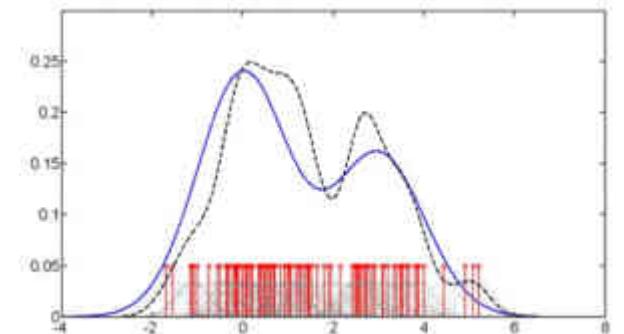
then repeating these two steps until convergence.

Unsupervised Learning and EM

- A central application of unsupervised learning is in the field of density estimation.

unsupervised learning intends to infer an a priori probability distribution $p_X(x)$.

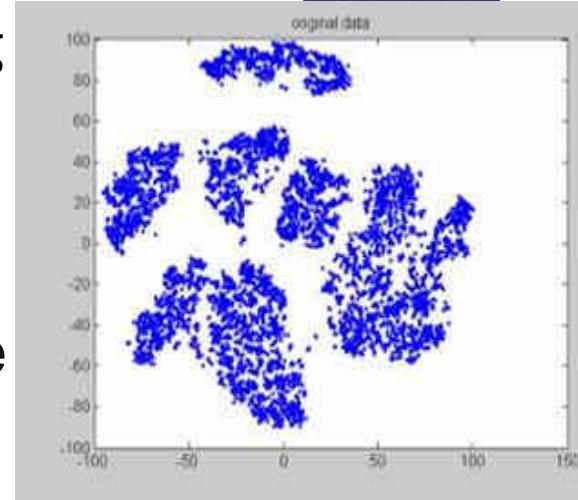
- We will cover unsupervised learning in Unit 4 some 8 hours from now – but lets understand one of the simplest unsupervised learning algorithm to set the context for expectation maximization
- K-Means clustering



- K-means clustering is a simple and elegant approach for partitioning a data set into K **distinct, non-overlapping clusters**.
- To perform K -means clustering, we must first specify the desired number of clusters K
- K -means algorithm will assign each observation to exactly one of the K clusters.

Algorithm 10.1 K -Means Clustering

1. Randomly assign a number, from 1 to K , to each of the observations. These serve as initial cluster assignments for the observations.
2. Iterate until the cluster assignments stop changing:
 - (a) For each of the K clusters, compute the cluster *centroid*. The k th cluster centroid is the vector of the p feature means for the observations in the k th cluster.
 - (b) Assign each observation to the cluster whose centroid is closest (where *closest* is defined using Euclidean distance).



K-Means Clustering

Let C_1, \dots, C_K denote sets containing the indices of the observations in each cluster

They must satisfy 2 properties

1. $C_1 \cup C_2 \cup \dots \cup C_K = \{1, \dots, n\}$. In other words, each observation belongs to at least one of the K clusters.
2. $C_k \cap C_{k'} = \emptyset$ for all $k = k'$. In other words, the clusters are nonoverlapping: no observation belongs to more than one cluster.

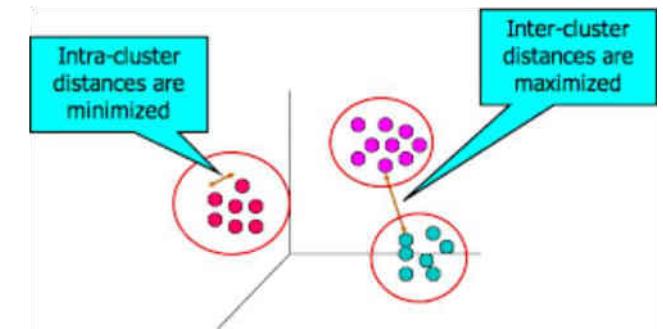
K-Means Clustering

- The idea behind K -means clustering is that a *good* clustering is one for which the *within-cluster variation* is as small as possible.
- The within-cluster variation for cluster C_k is a measure $W(C_k)$ of the amount by which the observations within a cluster differ from each other.

$$\underset{C_1, \dots, C_K}{\text{minimize}} \left\{ \sum_{k=1}^K W(C_k) \right\}.$$

- In words, this formula says that we want to partition the observations into K clusters such that the total within-cluster variation, summed over all K clusters, is as small as possible
- The intra-cluster distance is measured using the Euclidian distance between pair wise instances in the cluster

$$W(C_k) = \frac{1}{|C_k|} \sum_{i, i' \in C_k} \sum_{j=1}^p (x_{ij} - x_{i'j})^2$$



Expectation Maximization of K-Means

- The E-step is assigning the data points to the closest cluster.
- The M-step is computing the centroid of each cluster.
- Lets prove that convergence is guaranteed
- **E-Step**
- where $w_{ik}=1$ for data point x_i if it belongs to cluster k ; otherwise, $w_{ik}=0$

$$J = \sum_{i=1}^m \sum_{k=1}^K w_{ik} \|x^i - \mu_k\|^2 \quad (1)$$

$$\begin{aligned} \frac{\partial J}{\partial w_{ik}} &= \sum_{i=1}^m \sum_{k=1}^K \|x^i - \mu_k\|^2 \\ \Rightarrow w_{ik} &= \begin{cases} 1 & \text{if } k = \operatorname{argmin}_j \|x^i - \mu_j\|^2 \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

- **M-Step**

$$\begin{aligned} \frac{\partial J}{\partial \mu_k} &= 2 \sum_{i=1}^m w_{ik} (x^i - \mu_k) = 0 \\ \Rightarrow \mu_k &= \frac{\sum_{i=1}^m w_{ik} x^i}{\sum_{i=1}^m w_{ik}} \end{aligned} \quad (3)$$

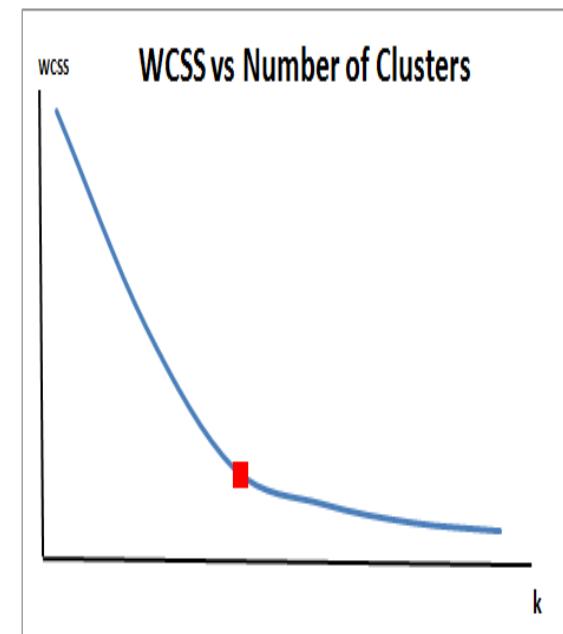
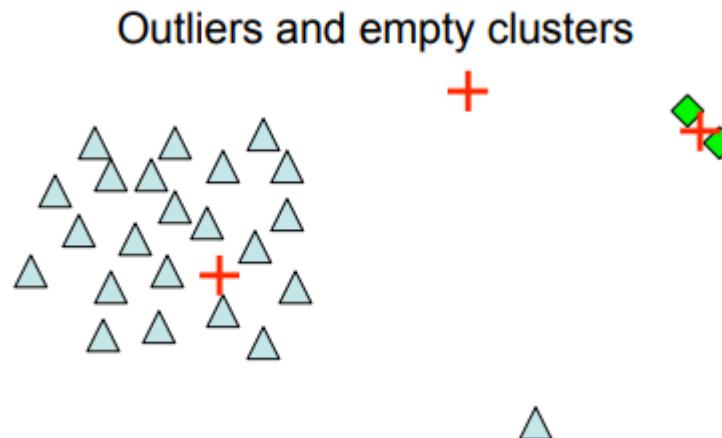
Closing Notes on K-Means

A normal distribution such that the mean $\mu = 0$ and standard deviation $\sigma = 1$ for your data

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} (-\infty < x < \infty).$$

K-Means algorithm converges to a local minimum:

- Can try multiple random restarts



Expectation Maximization

In the expectation, or E-step, **the missing data are estimated** given the

- observed data and
- current estimate of the model parameters

In the maximization or the M-step, **the likelihood function is maximized**

- under the assumption that the **missing data are known**

The estimate of the missing data from the E-step are used in lieu of the actual missing data.

Expectation Maximization

Assume that we have two coins, C1 and C2

Assume the bias of C1 is θ_1 (i.e., probability of getting heads with C1)

Assume the bias of C2 is θ_2 (i.e., probability of getting heads with C2)

We want to find θ_1, θ_2 by performing a number of trials (i.e., coin tosses)

Expectation Maximization



So far we have kind of looked at EM

Lets look at a real EM using a binomial model

A coin experiment

Suppose your friend has posed a challenge:

Estimate the bias of two coins in her possession.

They might be fair coins, be more heavily weighted towards heads; you don't know.

Here's the clue she's provided: a piece of paper with 5 records of an experiment where she's:

Expectation Maximization

Chosen one of the two coins at random.

Flipped that same coin 10 times.

How can you provide a reasonable estimate of each coin bias? Let's refer to these coins as coin A and coin B and their bias as θ_A and θ_B .



H T T T H H T H T H

H H H H T H H H H H

H T H H H H H T H H

H T H T T T H H T T

T H H H T H H H T H

Expectation Maximization

$$\theta_1 = \frac{\text{number of heads using } C1}{\text{total number of flips using } C1}$$

$$\theta_2 = \frac{\text{number of heads using } C2}{\text{total number of flips using } C2}$$

Expectation Maximization

Thus : (If A represents θ_1 and B represents θ_2)

$$\theta_1 = 24/30 = 0.8$$

$$\theta_2 = 9/20 = 0.45$$

Expectation Maximization

Assume a more challenging problem. We do not know the identities of the coins used for each set of tosses (we treat them as hidden variables).

H	T	T	T	H	H	T	H	T	H	
H	H	H	H	T	H	H	H	H	H	
H	T	H	H	H	H	H	T	H	H	
H	T	H	T	T	T	H	H	T	T	
T	H	H	H	T	H	H	H	H	T	H

Expectation Maximization

This can be modelled as a binomial distribution .

Each trial belongs to either Coin A or Coin B

- We only know that each coin has an equal chance of being chosen each time.
- In this scenario, the coin is not observed, and could be considered a hidden or latent variable
- This is the setup

coin	flips	# coin A heads	# coin B heads
?	HTTTHHTH	?	?
?	HHHHHTHHHH	?	?
?	HTHHHHHTHH	?	?
?	HTHTTTHHTT	?	?
?	THHHTHHHTH	?	?

Expectation Maximization

Right now we're stuck, because we'd like to count up the number of heads for each coin, but we don't know which coin is being flipped in each trial.

It turns out that we can make progress by starting with a guess for the coin biases

Which will allow us to estimate which coin was chosen in each trial and come up with an **estimate for the expected number of heads and tails for each coin across the trials (E-step)**

We then use these counts to **recompute a better guess for each coin bias (M-step)**

By repeating these two steps, we continue to get a better estimate of the two coin biases and converge at a solution that turns out to be a local maximum to the problem.

coin	flips	# coin A heads	# coin B heads
?	HTTTHHTHTH	?	?
?	HHHHHTHHHHHH	?	?
?	HTHHHHHTHH	?	?
?	HTHTTTTHHTT	?	?
?	THHHTHHTH	?	?

Expectation Maximization

Estimating likelihood each coin was chosen

Estimate the probability that each coin is the true coin given the flips we see in the trial

Which will allow us to estimate which coin was chosen in each trial .

Use that to proportionally assign heads and tails counts to each coin.

Let's make this concrete with one of the examples we just mentioned:

Lets initially **guess** that

our current biases for coin A and B are 0.4 and 0.7

- we observe the following flips: HHHHHHHHHTT
- what is the probability that these flips came from coin A and coin B? Let's call this series of flips event E, the event we chose A be Z_A and B Z_B .
- Both coin are equally likely to be chosen to $P(Z_A) = P(Z_B) = 0.5$
- Now we need to estimate the $P(E|Z_A)$ and $P(E|Z_B)$ for example $P(\text{HHHHHHHHTT}|Z_A)$
- Recollect since there are only 2 choices we could use

$$P(X) = \frac{n!}{(n-X)! X!} \cdot (p)^X \cdot (q)^{n-X}$$

Expectation Maximization

$$P(E|Z_A) = P(\text{HHHHHHHHHTT} | \text{A chosen}) = \frac{10!}{8!2!} 0.4^8 0.6^2$$

$$P(X) = \frac{n!}{(n-X)! X!} \cdot (p)^X \cdot (q)^{n-X}$$

$$P(E|Z_B) = P(\text{HHHHHHHHHTT} | \text{B chosen}) = \frac{10!}{8!2!} 0.7^8 0.3^2$$

It looks like the first trial came from Coin B

But what we wish to find is $P(Z_A | E)$ and $P(Z_B | E)$

Expectation Maximization

$$P(Z_A|E) = \frac{P(E|Z_A)P(Z_A)}{P(E|Z_A)P(Z_A) + P(E|Z_B)P(Z_B)}$$

$$\begin{aligned} P(Z_A|E) &= \frac{P(E|Z_A)}{P(E|Z_A) + P(E|Z_B))} \\ &= \frac{\frac{10!}{8!2!} 0.4^8 0.6^2}{\frac{10!}{8!2!} 0.4^8 0.6^2 + \frac{10!}{8!2!} 0.7^8 0.3^2} \\ &= \frac{0.4^8 0.6^2}{0.4^8 0.6^2 + 0.7^8 0.3^2} = 0.0435 \end{aligned}$$

$$P(Z_B|E) = \frac{0.7^8 0.3^2}{0.4^8 0.6^2 + 0.7^8 0.3^2} = 0.0956$$

$$P(X) = \frac{n!}{(n-X)! X!} \cdot (p)^X \cdot (q)^{n-X}$$

$$P(Z_A) = P(Z_B) = 0.5$$

We can eliminate the values from the equation

Thanks to Baye's theorem and the law of total probability, we can partition all of the events in Z (which coin we choose) over ZA and ZB as we have to choose one or the other.

Expectation Maximization

Here's one "E-step" filled out assuming our current parameters are $\theta_A^0 = 0.6$ and $\theta_B^0 = 0.5$.

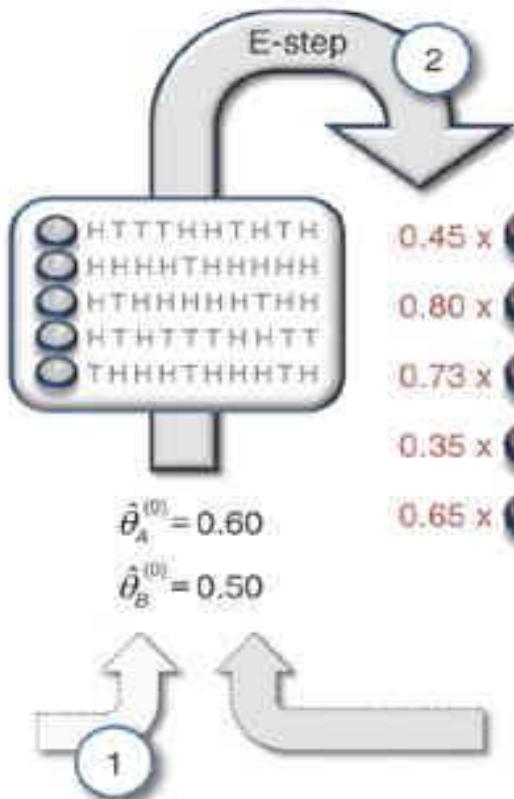
flips	probability it was coin A	probability it was coin B
HTTTHHTH	0.45	0.55
HHHHHTHHHH	0.8	0.2
HTHHHHHTHH	0.73	0.27
HTHTTTHHTT	0.35	0.65
THHHTHHHTH	0.65	0.35

$$P(Z_A|E) = \frac{\theta_A^h(1 - \theta_A)^t}{\theta_A^h(1 - \theta_A)^t + \theta_B^h(1 - \theta_B)^t}$$

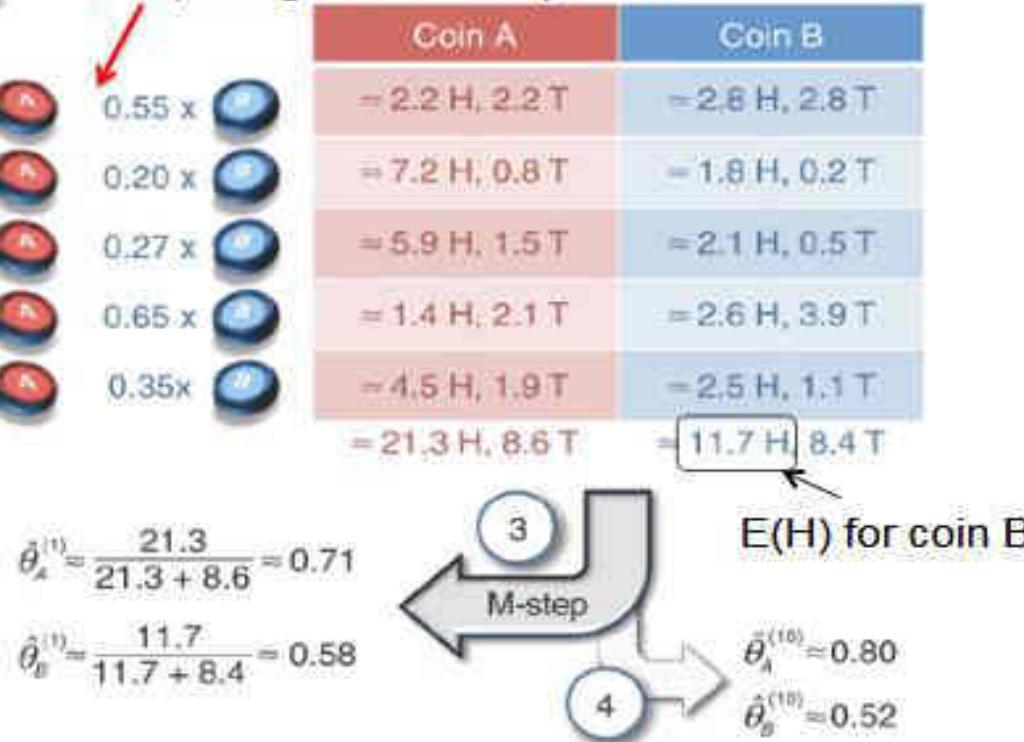
$$P(Z_B|E) = \frac{\theta_B^h(1 - \theta_B)^t}{\theta_A^h(1 - \theta_A)^t + \theta_B^h(1 - \theta_B)^t}$$

When the identities of the coins are unknown

b Expectation maximization



Instead of picking up the single best guess, the EM algorithm computes probabilities for each possible completion of the missing data, using the current parameters



		E(H) for coin B	
Coin A	Coin B		
= 2.2 H, 2.2 T	= 2.6 H, 2.8 T		
= 7.2 H, 0.8 T	= 1.8 H, 0.2 T		
= 5.9 H, 1.5 T	= 2.1 H, 0.5 T		
= 1.4 H, 2.1 T	= 2.6 H, 3.9 T		
= 4.5 H, 1.9 T	= 2.5 H, 1.1 T		
= 21.3 H, 8.6 T	= 11.7 H, 8.4 T		
		$\hat{\theta}_A^{(10)} = 0.80$	$\hat{\theta}_B^{(10)} = 0.52$

Expectation Maximization

flips	probability it was coin A	probability it was coin B	# heads attributed to A	# heads attributed to B
HTTTHHTH	0.45	0.55	2.2	2.8
HHHHTHHHHH	0.8	0.2	7.2	1.8
HTHHHHHTHH	0.73	0.27	5.9	2.1
HTHTTTHHTT	0.35	0.65	1.4	2.6
THHHTHHHTH	0.65	0.35	4.5	2.5

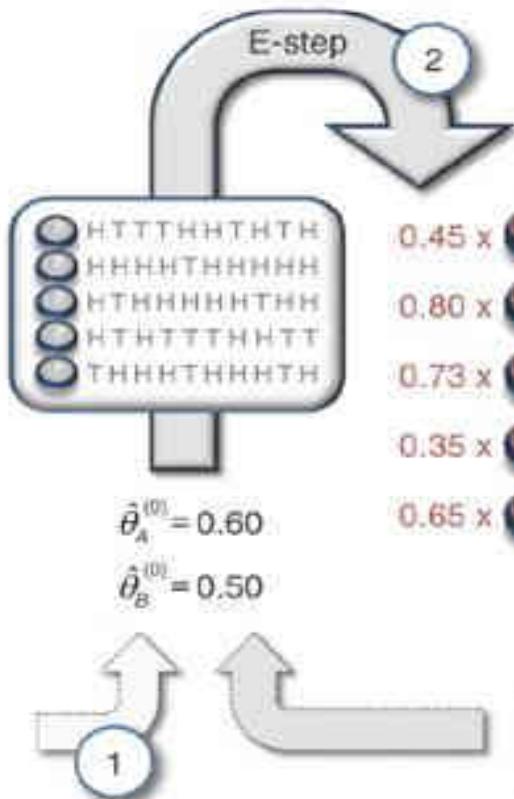
The M-Step

$$\theta_a^1 = \frac{2.2 + 7.2 + 5.9 + 1.4 + 4.5}{10 * (.45 + .8 + .73 + .35 + .65)} = 0.71$$

$$\theta_b^1 = \frac{2.8 + 1.8 + 2.1 + 2.6 + 2.5}{10 * (.55 + .2 + .27 + .65 + .35)} = 0.58$$

When the identities of the coins are unknown

b Expectation maximization

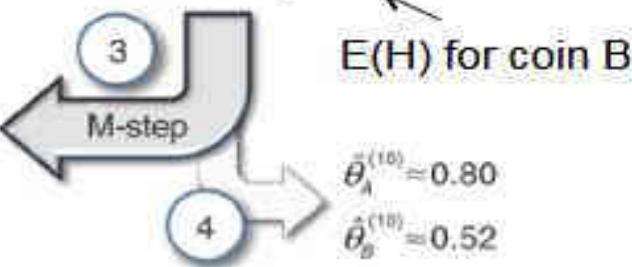


Instead of picking up the single best guess, the EM algorithm computes probabilities for each possible completion of the missing data, using the current parameters

Coin A	Coin B
= 2.2 H, 2.2 T	= 2.6 H, 2.8 T
= 7.2 H, 0.8 T	= 1.8 H, 0.2 T
= 5.9 H, 1.5 T	= 2.1 H, 0.5 T
= 1.4 H, 2.1 T	= 2.6 H, 3.9 T
= 4.5 H, 1.9 T	= 2.5 H, 1.1 T
= 21.3 H, 8.6 T	= 11.7 H, 8.4 T

$$\hat{\theta}_A^{(1)} = \frac{21.3}{21.3 + 8.6} = 0.71$$

$$\hat{\theta}_B^{(1)} = \frac{11.7}{11.7 + 8.4} = 0.58$$





MACHINE INTELLIGENCE EXPECTATION MAXIMIZATION - GMM

K.S.Srinivas
Department of Computer Science
and Engineering

Machine Intelligence

Unit III

Gaussian Mixture Models

Srinivas K.S
Department of Computer Science

Gaussian Distributions

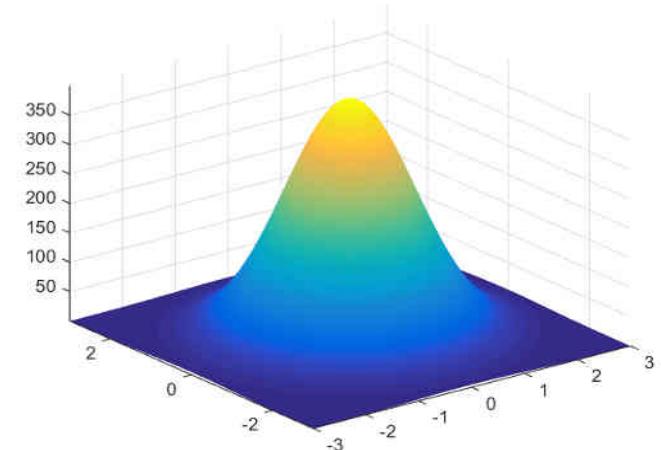


Univariate Gaussian Distribution

$$\mathcal{N}(x | \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Multivariate Gaussian Distribution

A multivariate normal distribution is a vector in multiple normally distributed variables, such that any linear combination of the variables is also normally distributed.



$$\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi|\Sigma|)^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})\right\}$$


The diagram shows two blue arrows originating from the words "mean" and "covariance" located below the equation. The first arrow points to the term $\boldsymbol{\mu}$ in the equation. The second arrow points to the term Σ in the equation.

Gaussian Mixture Models

$$\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi|\boldsymbol{\Sigma}|)^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right\}$$

mean **covariance**

We need to estimate these parameters of a distribution
One method – Maximum Likelihood (ML) Estimation.

Gaussian Mixture Models

- Consider log of Gaussian Distribution

$$\ln p(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{1}{2} \ln(2\pi) - \frac{1}{2} \ln|\boldsymbol{\Sigma}| - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})$$

- Take the derivative and equate it to zero

$$\frac{\partial \ln p(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\mu}} = 0$$

↓

$$\boldsymbol{\mu}_{ML} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n$$

$$\frac{\partial \ln p(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\Sigma}} = 0$$

↓

$$\boldsymbol{\Sigma}_{ML} = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \boldsymbol{\mu}_{ML})(\mathbf{x}_n - \boldsymbol{\mu}_{ML})^T$$

$$\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi|\boldsymbol{\Sigma}|)^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\}$$

mean covariance

Where, N is the number of samples or data points

Gaussian Mixture Models

- Linear super-position of Gaussians

$$p(x) = \sum_{k=1}^K \pi_k \mathcal{N}(x | \mu_k, \Sigma_k)$$

Number of Gaussians Mixing coefficient: weightage for each Gaussian dist.

- Normalization and positivity require $0 \leq \pi_k \leq 1, \sum_{k=1}^K \pi_k = 1$

$$\mathcal{N}(x | \mu, \Sigma) = \frac{1}{(2\pi|\Sigma|)^{1/2}} \exp\left\{-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right\}$$

mean covariance

- Consider log likelihood

$$\ln p(X | \mu, \Sigma, \pi) = \sum_{n=1}^N \ln p(x_n) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k) \right\}$$

ML does not work here as there is no closed form solution

Parameters can be calculated using Expectation Maximization (EM) technique

Gaussian Mixture Models – The Algorithm

- Initialize parameters.
 w_j, μ_j, σ^2_j for each Gaussian j in our model.
- E step: calculate posterior dist. of latent variables
probability that these Gaussians generated the data
- M step: update parameters.
update w_j, μ_j, σ^2_j for each Gaussian j
- Repeat E and M steps until convergence.
go until parameters don't change much
- It converges to some local optimum.

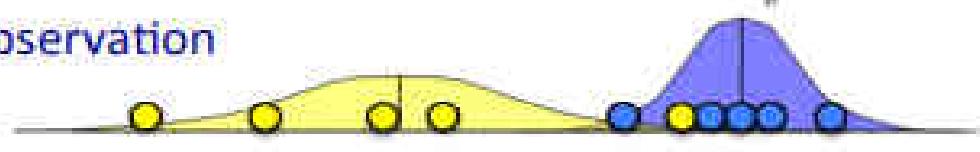
Lets understand this using a 1-d example

EM-GMM Example

Mixture models in 1-d

- Observations $x_1 \dots x_n$
 - K=2 Gaussians with unknown μ, σ^2
 - estimation trivial if we know the source of each observation

$$\mu_b = \frac{x_1 + x_2 + \dots + x_{n_b}}{n_b}$$
$$\sigma_b^2 = \frac{(x_1 - \mu_b)^2 + \dots + (x_{n_b} - \mu_b)^2}{n_b}$$



- What if we don't know the source?
- If we knew parameters of the Gaussians (μ, σ^2)
 - can guess whether point is more likely to be a or b

$$P(b|x_i) = \frac{P(x_i|b)P(b)}{P(x_i|b)P(b) + P(x_i|a)P(a)}$$

$$P(x_i|b) = \frac{1}{\sqrt{2\pi\sigma_b^2}} \exp\left(-\frac{(x_i - \mu_b)^2}{2\sigma_b^2}\right)$$



Expectation Maximization (EM)

- Chicken and egg problem
 - need (μ_a, σ_a^2) and (μ_b, σ_b^2) to guess source of points
 - need to know source to estimate (μ_a, σ_a^2) and (μ_b, σ_b^2)
- EM algorithm
 - start with two randomly placed Gaussians $(\mu_a, \sigma_a^2), (\mu_b, \sigma_b^2)$

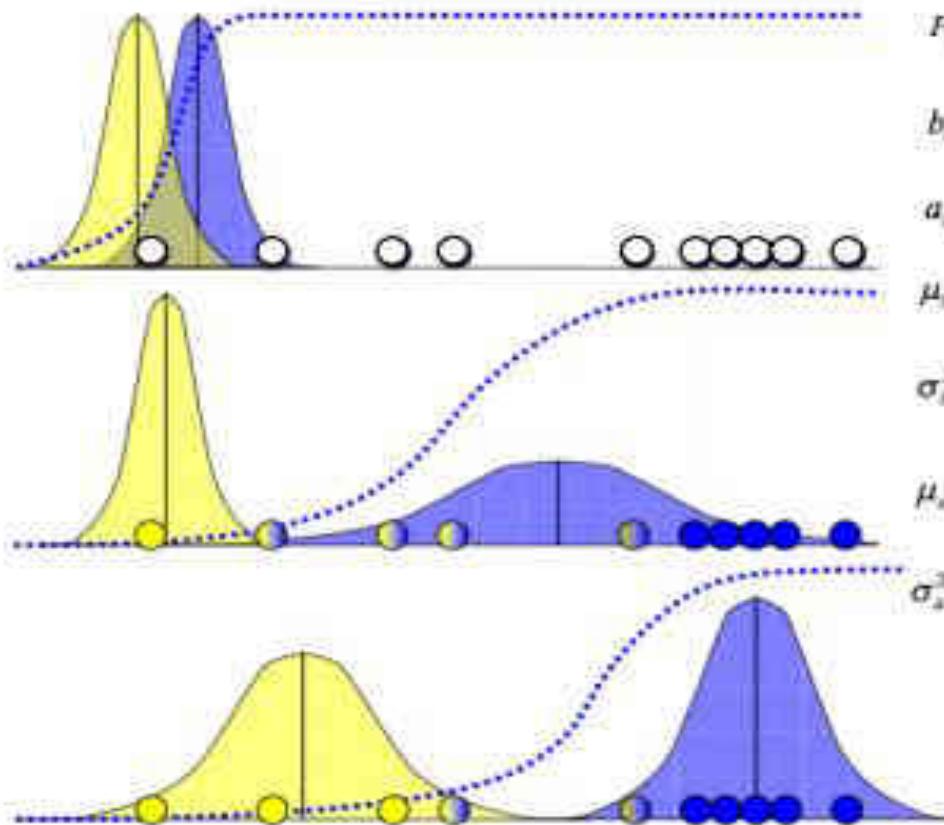
E-step: – for each point: $P(b|x_i) =$ does it look like it came from b?

M-step: – adjust (μ_a, σ_a^2) and (μ_b, σ_b^2) to fit points assigned to them

- iterate until convergence

EM-GMM Example

EM: 1-d example



$$P(x_i | b) = \frac{1}{\sqrt{2\pi\sigma_b^2}} \exp\left(-\frac{(x_i - \mu_b)^2}{2\sigma_b^2}\right)$$

$$b_i = P(b | x_i) = \frac{P(x_i | b)P(b)}{P(x_i | b)P(b) + P(x_i | a)P(a)}$$

$$a_i = P(a | x_i) = 1 - b_i$$

$$\mu_b = \frac{b_1 x_1 + b_2 x_2 + \dots + b_n x_n}{b_1 + b_2 + \dots + b_n}$$

$$\sigma_b^2 = \frac{b_1 (x_1 - \mu_b)^2 + \dots + b_n (x_n - \mu_b)^2}{b_1 + b_2 + \dots + b_n}$$

$$\mu_a = \frac{a_1 x_1 + a_2 x_2 + \dots + a_n x_n}{a_1 + a_2 + \dots + a_n}$$

$$\sigma_a^2 = \frac{a_1 (x_1 - \mu_a)^2 + \dots + a_n (x_n - \mu_a)^2}{a_1 + a_2 + \dots + a_n}$$

could also estimate priors:

$$P(b) = (b_1 + b_2 + \dots + b_n) / n$$

$$P(a) = 1 - P(b)$$

D>1 Example

Gaussian mixture models: d>1

- Data with d attributes, from k sources
- Each source c is a Gaussian
- Iteratively estimate parameters:

— prior: what % of instances came from source c ?

$$P(c) = \frac{1}{n} \sum_{i=1}^n P(c | \vec{x}_i)$$

— mean: expected value of attribute j from source c

$$\mu_{c,j} = \sum_{i=1}^n \left(\frac{P(c|\vec{x}_i)}{nP(c)} \right) x_{i,j}$$

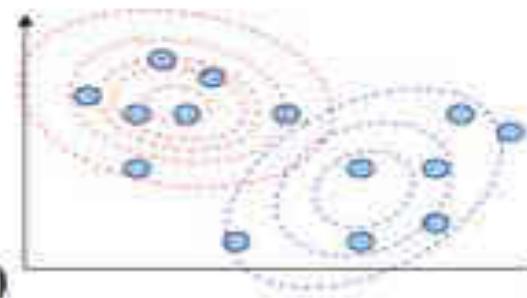
— covariance: how correlated are attributes j and k in source c ?

$$(\Sigma_c)_{j,k} = \sum_{i=1}^n \left(\frac{P(c|\vec{x}_i)}{nP(c)} \right) (x_{i,j} - \mu_{c,j})(x_{i,k} - \mu_{c,k})$$

— based on: our guess of the source for each instance

$$P(c | \vec{x}_i) = \frac{P(\vec{x}_i | c)P(c)}{\sum_{c'=1}^k P(\vec{x}_i | c')P(c')}$$

$$P(\vec{x}_i | c) = \frac{1}{\sqrt{2\pi|\Sigma_c|}} \exp\left(-\frac{1}{2} (\vec{x}_i - \vec{\mu}_c)^T \underbrace{\Sigma_c^{-1}}_{\sum_{j=1, k=1}^d (x_{i,j} - \mu_{c,j})(x_{i,k} - \mu_{c,k})} (\vec{x}_i - \vec{\mu}_c)\right)$$



Applications

Estimating parameters of a Gaussian Mixture model.

Baum Welch Algorithm in Hidden Markov Models.

Clustering.



THANK YOU

Srinivas K S

Department of Computer Science & Engineering

srinivasks@pes.edu



MACHINE INTELLIGENCE

Hidden Markov Model

K.S.Srinivas

Department of Computer Science and Engineering

MACHINE INTELLIGENCE

Discrete Markov Processes

K.S.Srinivas

Department of Computer Science and Engineering

We have so far commonly dealt with random samples.

A random sample can be thought of as a set of objects that are chosen randomly. Or, more formally, it's "a sequence of independent, identically distributed (IID) random variables"

Identically Distributed means that there are no overall trends—the distribution doesn't fluctuate and all items in the sample are taken from the same probability distribution.

Independent means that the sample items are all independent events. In other words, they aren't connected to each other in any way.

Random variables X and Y on the same probability space are said to be independent if the events $X = a$ and $Y = b$ are independent for all values a,b. Equivalently, the joint distribution of independent r.v.'s decomposes as

$$P(X = a, Y = b) = P(X = a)P(Y = b) \quad \forall a, b.$$

Examples: Put m balls with numbers written on them in an urn. Draw n balls from the urn with replacement, and let X_i be the number on the i th ball. Then X_1, X_2, \dots, X_n will be i.i.d.

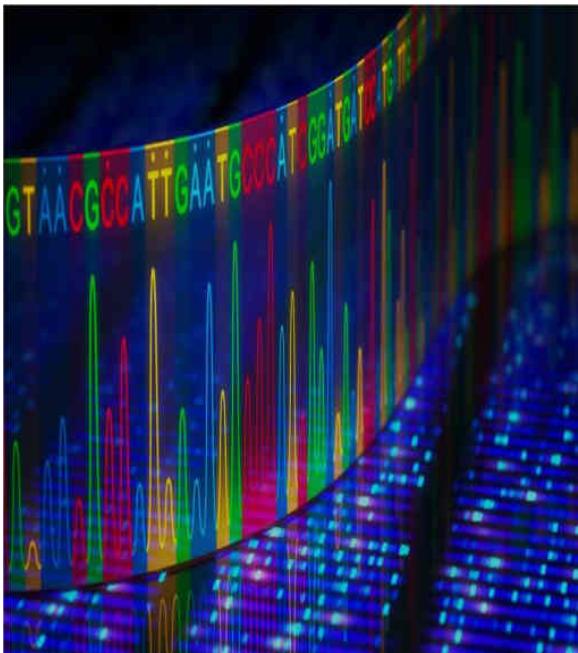
But if we draw the balls without replacement, X_1, X_2, \dots, X_n will not be i.i.d. - they will all have the same distribution, but will not be independent.

If we draw the balls with replacement, but let X_i be i times the number on the i th ball, then X_1, X_2, \dots, X_n will not be i.i.d. - they will be independent, but they will have different distributions.

MACHINE INTELLIGENCE

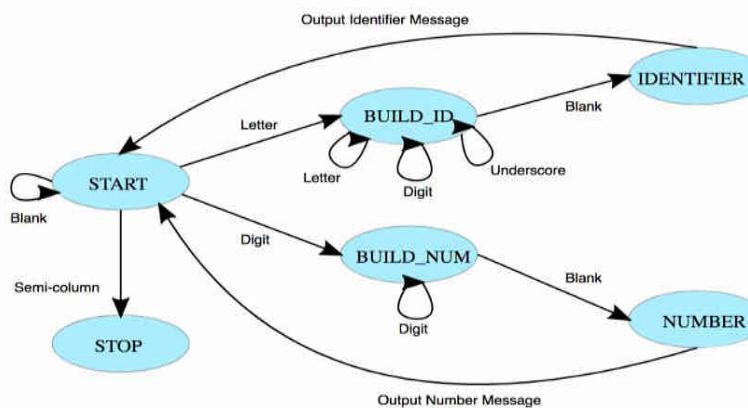
Discrete Markov Process

- The IID assumption that we make during Naïve Bayes is a very strong assumption
- This does not always hold good in many cases

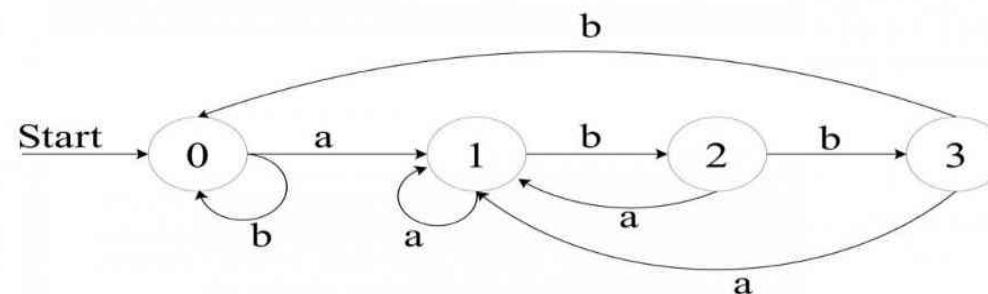
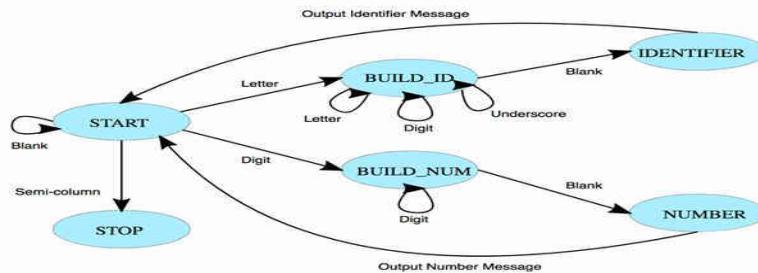


- In each of these cases that we just saw the occurrence of a value of a feature effects that value of another feature
- As an example if it had rainy over the last 48 hours what are the chances of today being rainy as opposed to be sunny.
- In each of these cases we need to consider the probabilities effect of one feature on another.
- Modelling of such cases is done on the lines of Finite State Machines

90% accurate					80% accurate		50% accurate		
Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed
76°	74°	70°	70°	71°	76°	75°			



- We know that state machines usually have a start end state, an alphabet that takes you to a new state and we have an end state.
- The alphabets can be thought of the change in conditions that cause a state transition.
- We modify the conditions slightly in modelling Markov Process there is no THE start state
- Transitions can happen between all states (more on this later)



Random variables

- The **possible states of the outcomes** are also known as the **domain** of the random variable, and the outcome is **based on the probability distribution defined over the domain** of the random variable.
- In rolling a six sided die, the domain of the random variable outcome, O , is given by $\text{domain}(O) = \{1, 2, 3, 4, 5, 6\}$, and the **probability distribution is given by a uniform distribution $P(o) = 1/6 \forall o \in \text{domain}(O)$** .
- The domain of the random variable has **discrete variables**; such random variables are known as **discrete random variables**.
- Consider the random variable representing the stock price of Google tomorrow. The **domain of this random variable will be all positive real numbers** with most of the probability mass distributed around $\pm 5\%$ of today's price. Such random variables are known as **continuous random variables**.

Random processes

Random variables are able to mathematically represent the outcomes of a single random phenomenon.

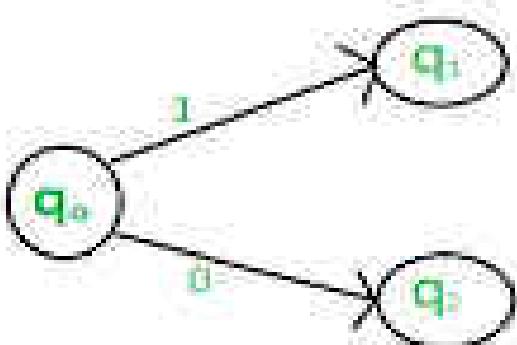
What if we want to **represent these random events over some period of time** or the length of an experiment?

- let's say we want to represent the stock prices for a whole day at intervals of every one hour
- we want to represent the height of a ball at intervals of every one second after being dropped from some height in a vacuum.

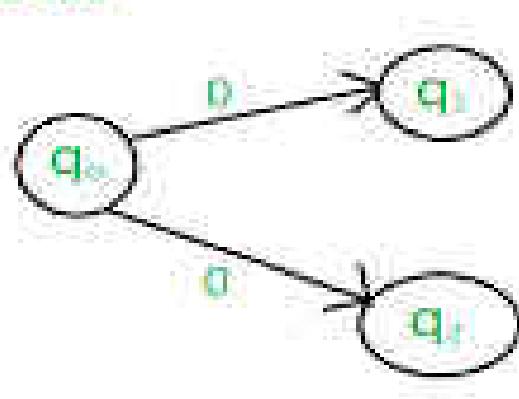
For such situations, **we would need a set of random variables**, each of which will represent the outcome at the given instance of time. **These sets of random variables** that represent random variables over a period of time are also known as **random processes**. It is worth noting that the **domains** of all these random variables are **the same**.

- Such **random processes**, in which we can **deterministically find the state of each random variable** given the initial conditions (in this case, dropping the ball, zero initial velocity) and the **parameters of the system** (in this case, the value of gravity), are known as deterministic random processes (commonly called **deterministic processes**).
- Random processes, in which **we can't determine the state of a process**, even if we are **given** the initial conditions and all **the parameters of the system**, are known as **stochastic random processes** (commonly called stochastic processes).

GeekforGeeks



Deterministic Algorithm



Non-Deterministic Algorithm

Markov processes

A stochastic process is called a Markov process if the state of the **random variable at the next instance** of time depends only on the **outcome of the random variable at the current time**.

$$P(R_{n+1} | R_1, R_2, \dots, R_n) = P(R_{n+1} | R_n)$$

Markov property

This property of a system, such that the future states of the system depend only on the current state of the system, is also known as the **Markov property**.

Systems satisfying the Markov property are also known as **memoryless systems**

A Discrete Markov process (Markov chain)

- The start state is not defined
- Is a **stochastic process** over a **discrete state space** satisfying the Markov property.
- The **probability of moving from the current state to the next state depends only on the present state** and not on any of the previous states.
- It is said to be **irreducible** if we can reach any state of the given Markov chain from any other state.
- state j is said to be accessible from state i if an integer $n_{ij} \geq 0$ exists such that the following condition is met:

$$Pr(X_{n_{ij}} = j | X_0 = i) = p_{ij}^{n_{ij}} > 0$$

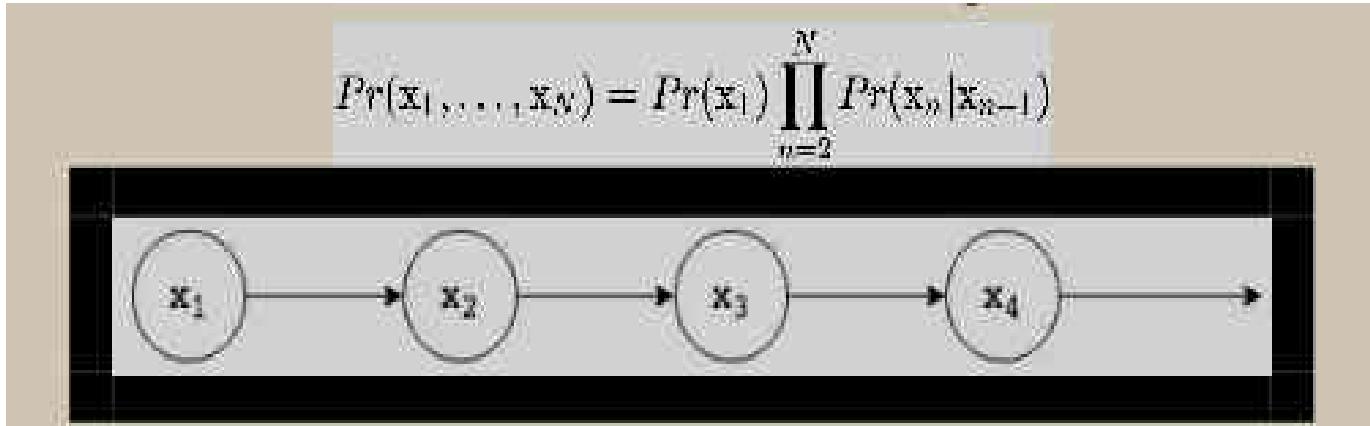
- This is the probability of a system being in state R_{n+1} given that the machine has been in R_1 at $t=1$, R_2 at $t=2$ and R_n at $t=n$ can be represented as follows

$$P(R_{n+1} | R_1, R_2, \dots, R_n)$$

- Such a process is called as a 'n' order Markov Process where the machine being in a state at $n+1$ is conditioned by all the previous states leading up to n .
- Generally Markov property is applies in which the above expressed is reduced to

$$P(R_{n+1} | R_1, R_2, \dots, R_n) = P(R_{n+1} | R_n)$$

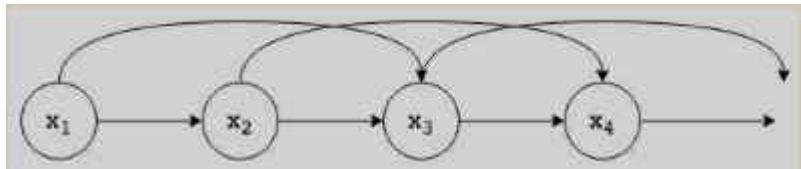
- A graphical representation of a first order Markov Chain with conditional dependence on only the previous state can be represented as



- This is how probabilistic graphical models are generally represented,
 - Nodes represent random variables
 - Edges represent a conditional probability distribution between these two variables.
 - This graphical representation gives us insight **into the causal relationships** between random variables.

Discrete Markov Processes

- As an example this is a 2nd order Markov Process

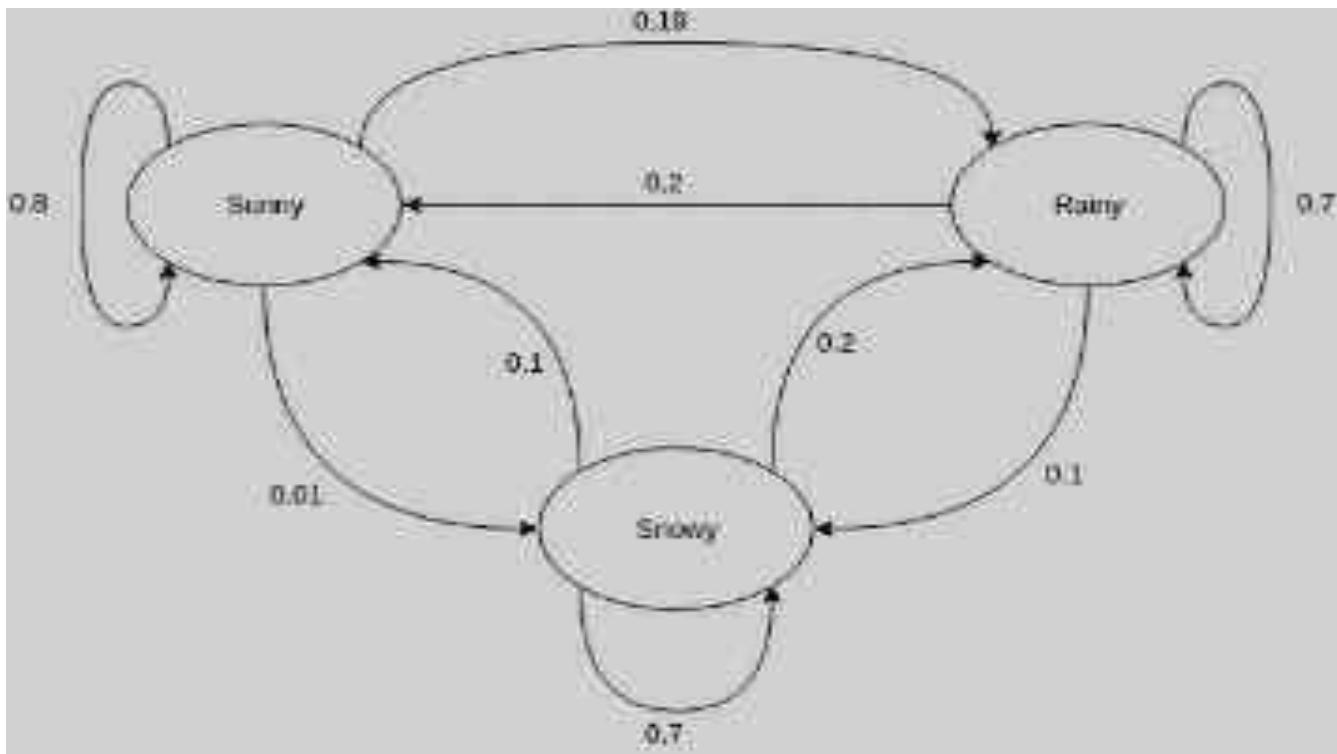


- Where the conditional probability is represented as follows

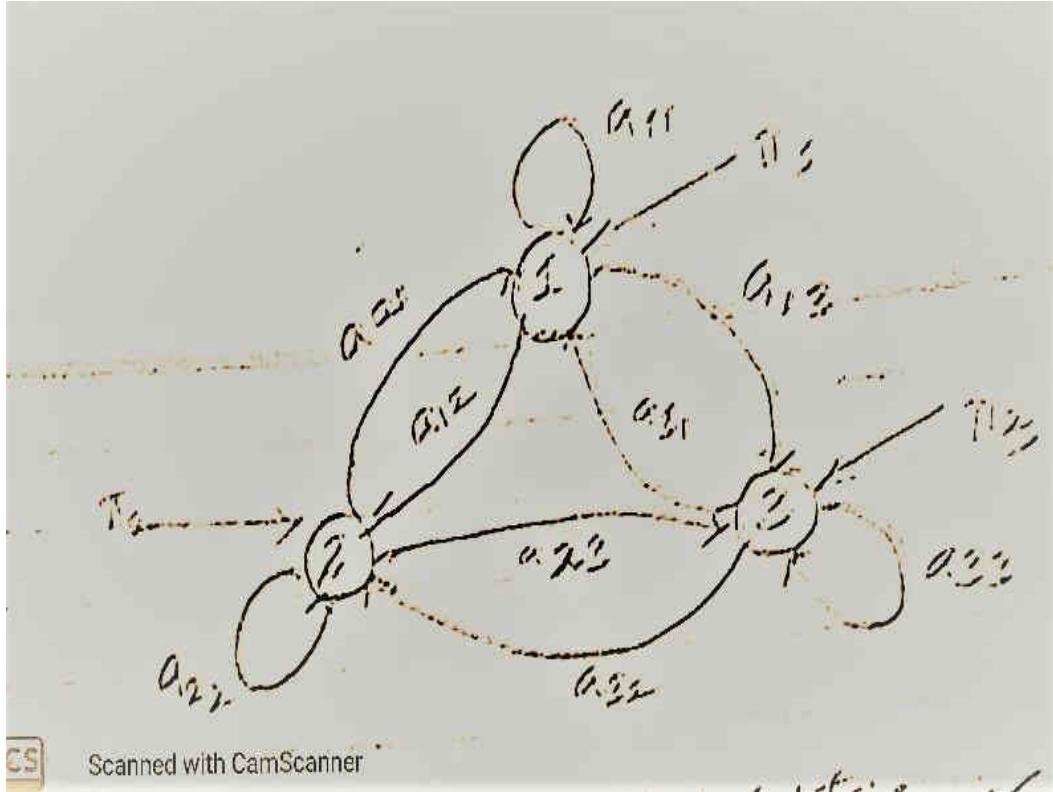
$$Pr(x_1, \dots, x_n) = Pr(x_1) Pr(x_2|x_1) \prod_{n=3}^N Pr(x_n|x_{n-1}, x_{n-2})$$

Discrete Markov Processes

- Let us represent this with an example of the weather being sunny rainy or cloudy only thru a first order Markov chain



- Lets understand the model a little better and introduce some more terms that we need for the model that we will apply.



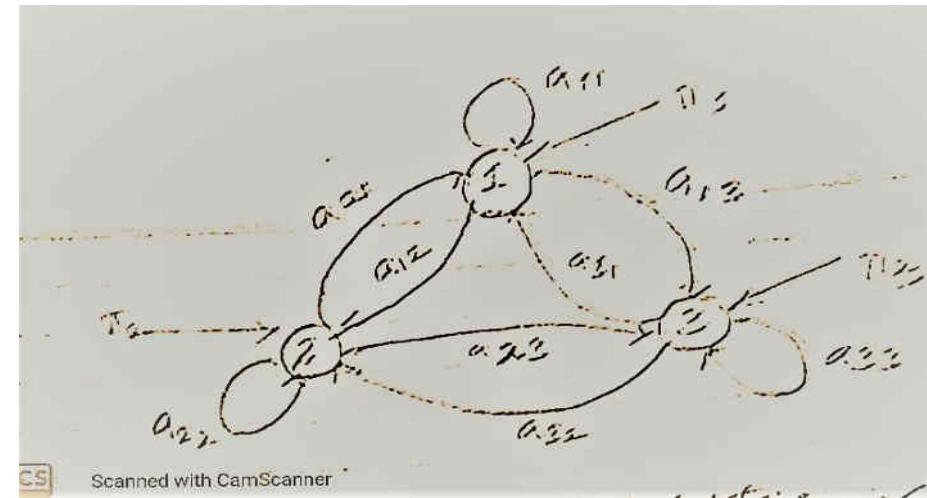
- The π that you see here are the starting probabilities since we can start from any state to any other state
- a_{12} states the transition probability of moving from state 1 to state 2
- Σ of all a's from a state must add upto to one.

Discrete Markov Process

- The π that you see here are the starting probabilities since we can start from any state to any other state
- All of the transition probabilities can be represented as a matrix called as the transition matrix
- $A = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$

a_{11} represents the probability of moving from a_1 to a_1 .

We can now fully define our model
 $\Lambda (\Pi, A)$ for a discrete Markov process



Scanned with CamScanner

$$\sum_{j=1}^N a_{ij} = 1 \quad \forall i \quad \sum_{i=1}^N \pi_i = 0$$

Example Problem (1)

- Consider a first order Markov process with 3 states = { Sunny, Cloudy, Rainy }
- $\pi = (0.3, 0.3, 0.4)$

	S	C	R
S	0.6	0.2	0.2
C	0.2	0.5	0.3
R	0.1	0.4	0.5

What is the probability of seeing a sequence of Cloudy, Sunny, Rainy, Cloudy, Rainy over the next five days?

$$\sum_{j=1}^N a_{ij} = 1 \quad \forall i \quad \sum_{i=1}^N \pi_i = 0$$

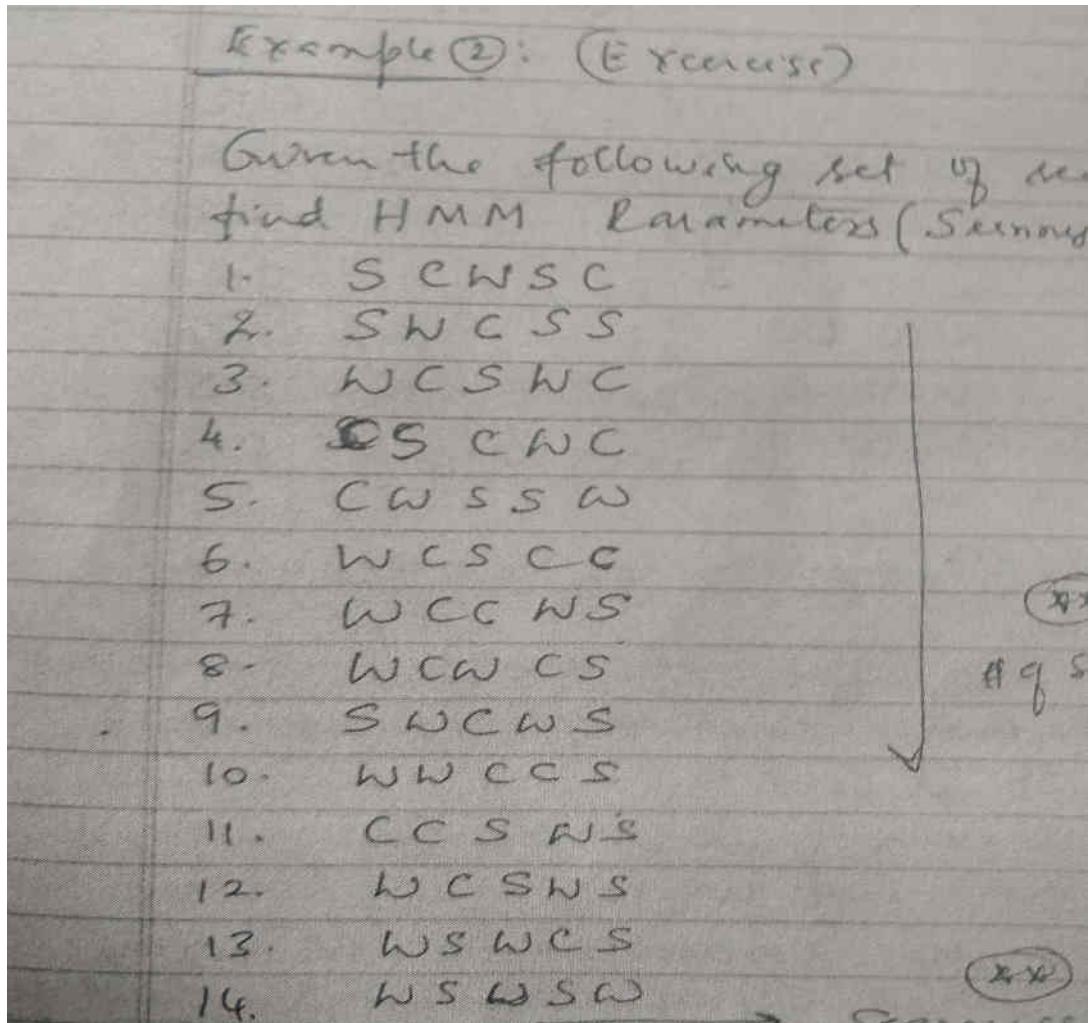
- Starting state is cloudy $\pi_C = 0.3$
- The trellis is CSRCR
- So we have a starting $P_c = 0.3$ and the multiply the $P(x)$ each of the next state given the current state

$$i.e P(CSRCR | \pi, A) = P(\pi_c) \cdot P(S|C) \cdot P(R|S) \cdot P(C|R) \cdot P(R|C)$$

$$\begin{aligned} & 0.3 \times 0.2 \times 0.2 \times 0.4 \times 0.3 \\ & 0.00144 \end{aligned}$$

Example Problem (2)

- Consider the trellis over a 14 day period



Compute the parameters of the Discrete Markov Model

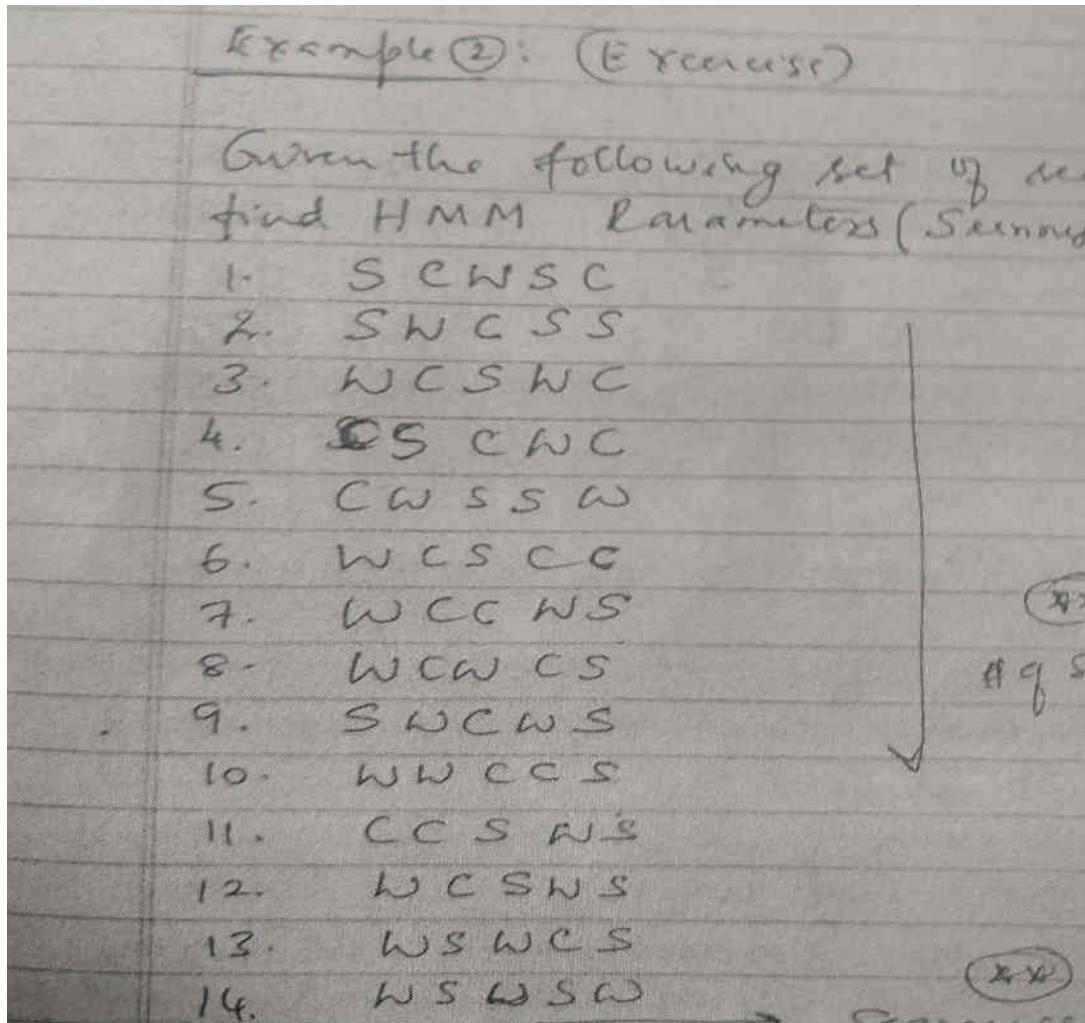
i.e compute

Π And the transition matrix

Solution:

Example Problem (2)

- Consider the trellis over a 14 day period



Hidden State

- Consider the following problem. We have 2 friends Karan and Vijay one in Bangalore and the other in Shimoga
- They speak to each other every day
- The only thing Karan states to Vijay on any day is whether is happy or Angry.
- His anger or happiness is defined by the weather it being sunny or rainy
- Given that he says he is HHSHS can you guess the weather in Shimoga



We have another probability called the emission probability
i.E given a state – weather what is probability of being happy or sad
This is represented by another matrix called the emission matrix. See you in the next class



THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701



MACHINE INTELLIGENCE

Hidden Markov Model

K.S.Srinivas

Department of Computer Science and Engineering

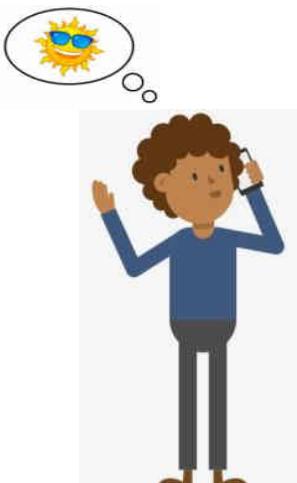
MACHINE INTELLIGENCE

Discrete Markov Processes

K.S.Srinivas

Department of Computer Science and Engineering

- Let us Recall the problem we framed
- vijay says karan only his mood which is dependent on the weather



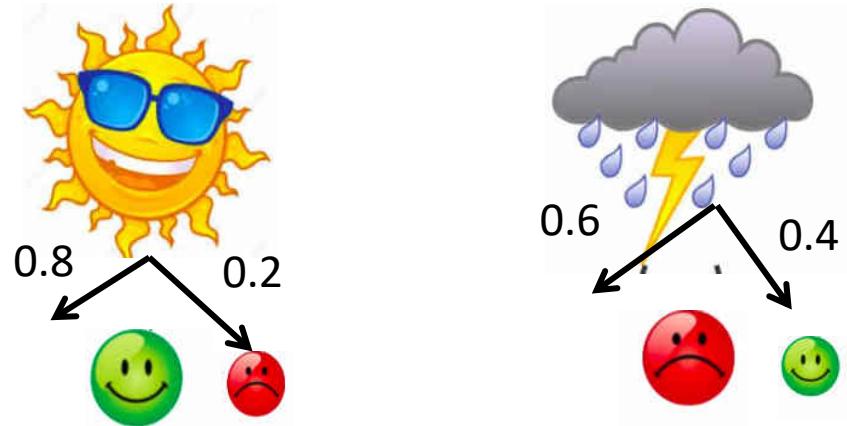
KARAN



VIJAY

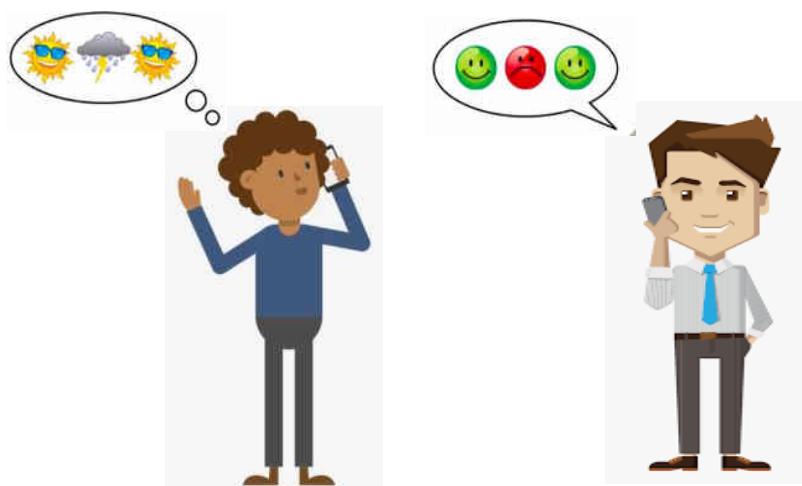


- weather can be sunny or rainy
- if its sunny vijay is happy and if its rainy vijay is grumpy
- so vijay tells karan his mood and he guesses the weather based on his mood

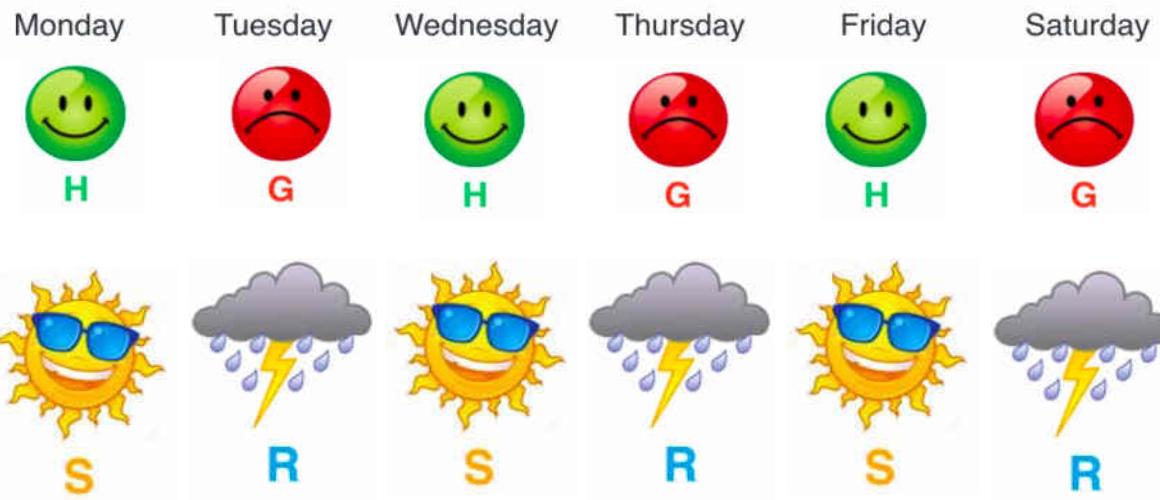


- now let say vijay is mostly happy when its sunny and mostly grumpy when its rainy
- that is there are some exceptions, and say we have probabilities for this

- so conversation between karan and vijay looks something like this

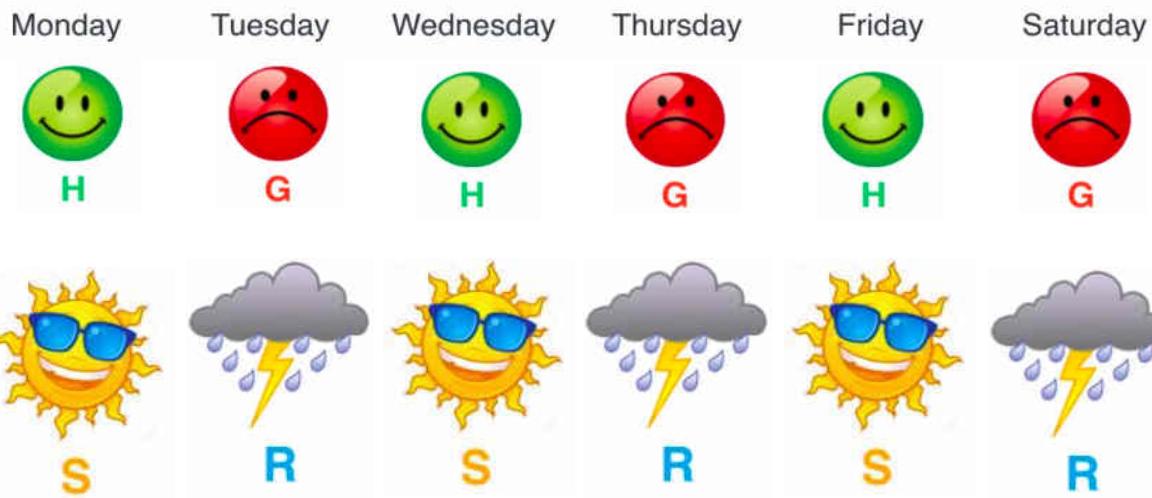


Weather

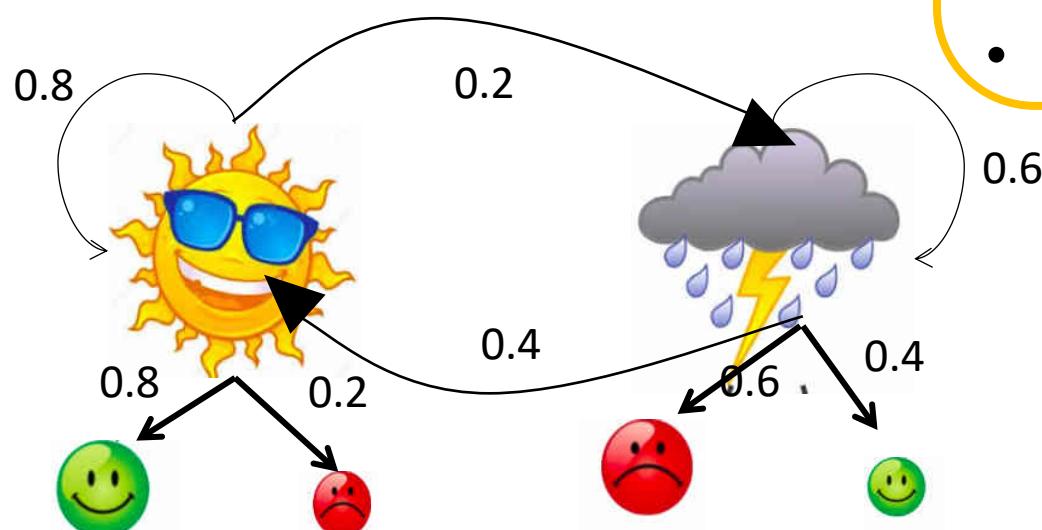


- so conversation between karan and vijay looks something like this
- consider a week conversation between karan and vijay

Weather

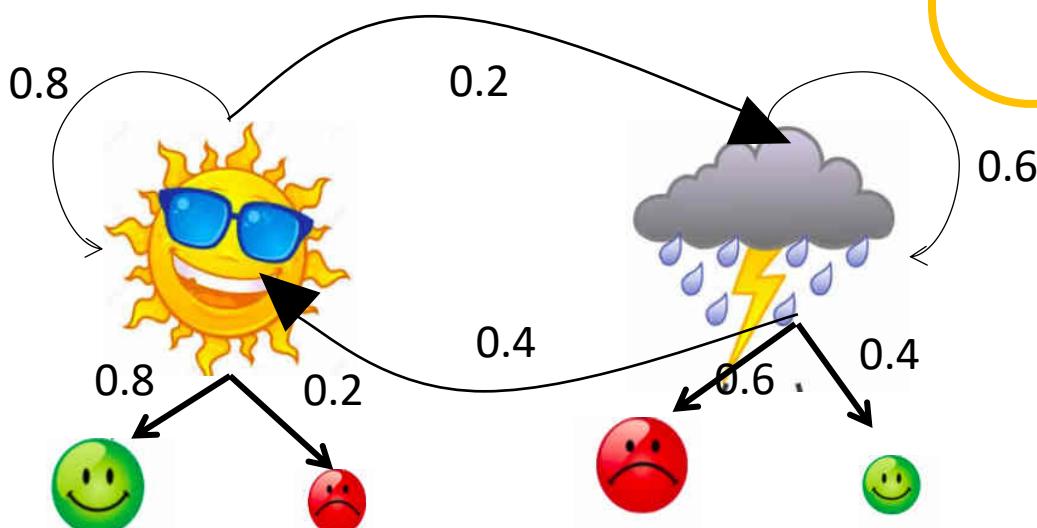


- but does this unlikely ,its more likely that if a day say is sunny ,next day is going to be sunny and same with rainy



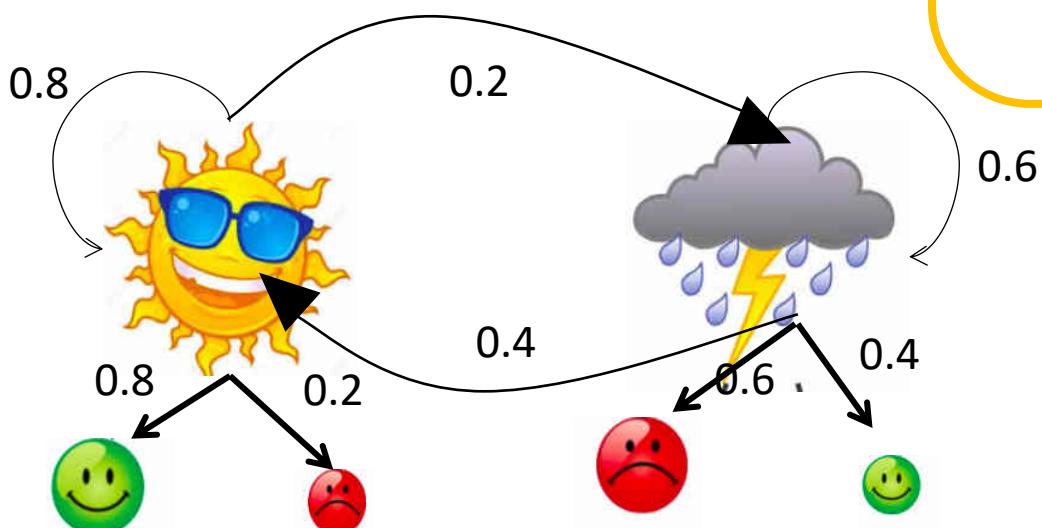
- lets get back to our model
- lets say if today is sunny the probability that next day is sunny is 0.8,
- then probability of next day being rainy is 0.2
- similarly

Hidden Markov Model



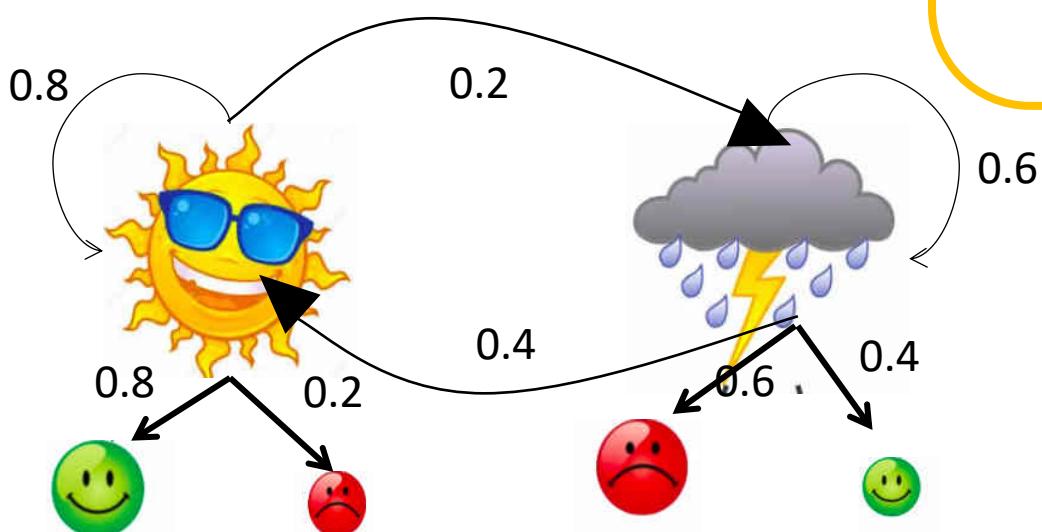
- what we have now is our hidden markov model
- this HMM has two states
- some observations we can see
- and weather is the hidden states

Hidden Markov Model



- this is called transition probability

Hidden Markov Model



- this are called emission probability

- How did we find these probabilities?
 - What's the probability that a random day is sunny or rainy
 - If Vijay is happy today, what's the probability that it's sunny or rainy?
 - If for three days Vijay is Happy, Grumpy, Happy, what was the weather
- We will try to answer four questions now

MACHINE INTELLIGENCE

How do we find probabilities?

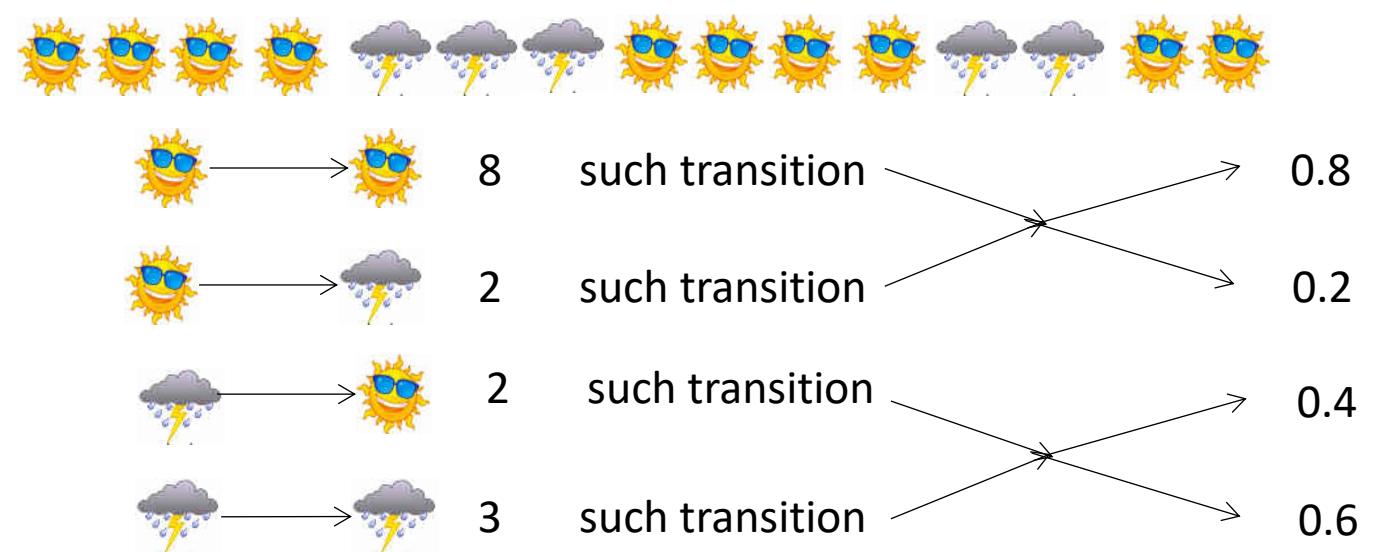


- lets say we some data that we looked at and that we've been able to study the past weathers
 - so we have the following data



How do we find probabilities?

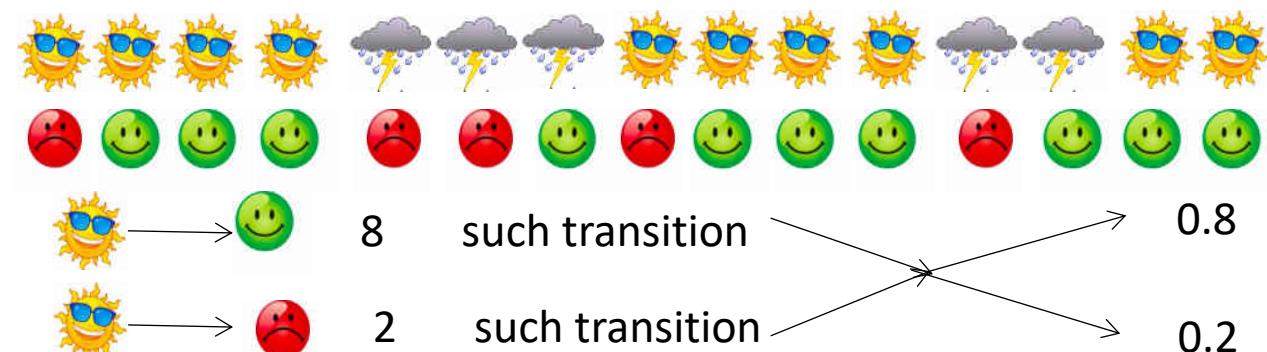
- lets say we have some data that we looked at and that we've been able to study the past weather
- so we have the following data
- and we are going to infer probabilities from this



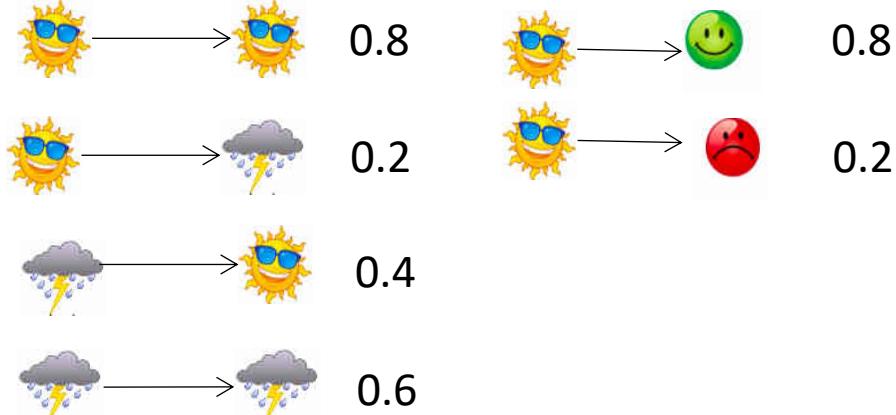
How do we find probabilities?

	→		0.8
	→		0.2
	→		0.4
	→		0.6

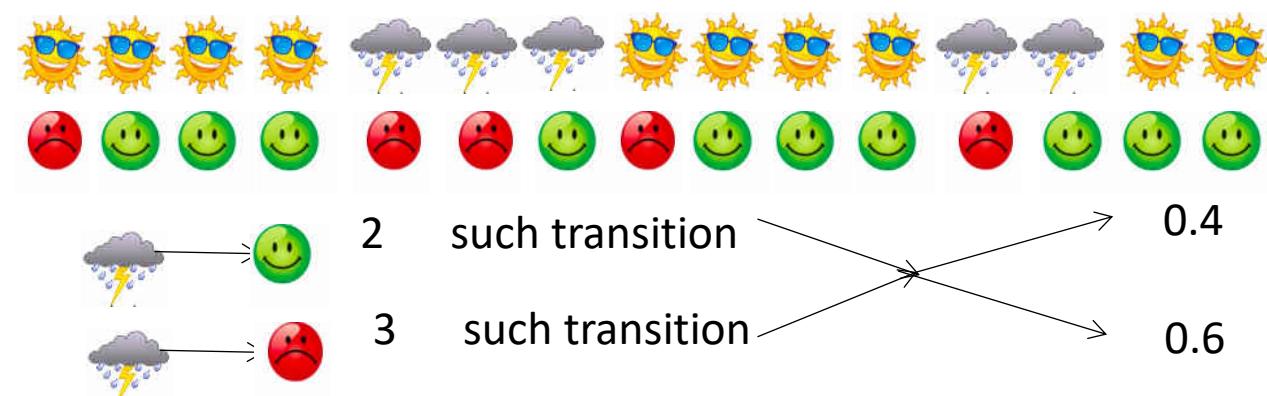
- now consider the vijay's mood on these days
- from these we can calculate emission probability



How do we find probabilities?



- now consider the vijay's mood on these days
- from these we can calculate emission probability

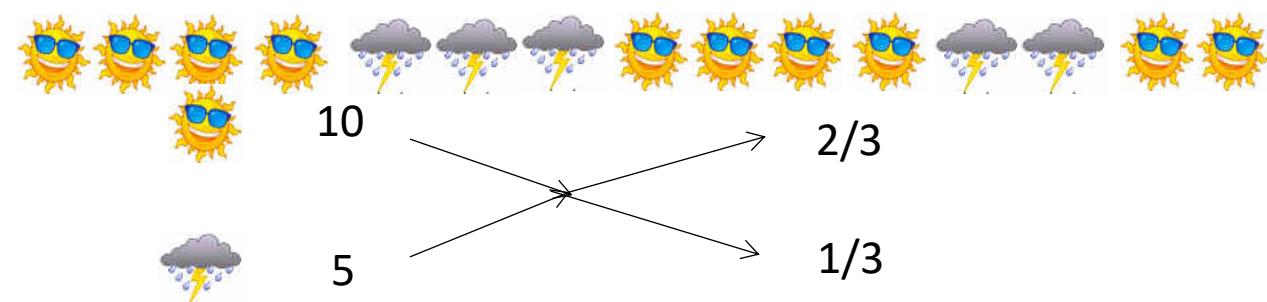


- ~~How did we find these probabilities?~~
- Whats the probability that a random day is sunny or rainy
- if Vijay is happy today,whats the probaility ,that its sunny or rainy?
- if for three days Vijay is Happy,Grumpy,Happy,what was the weather

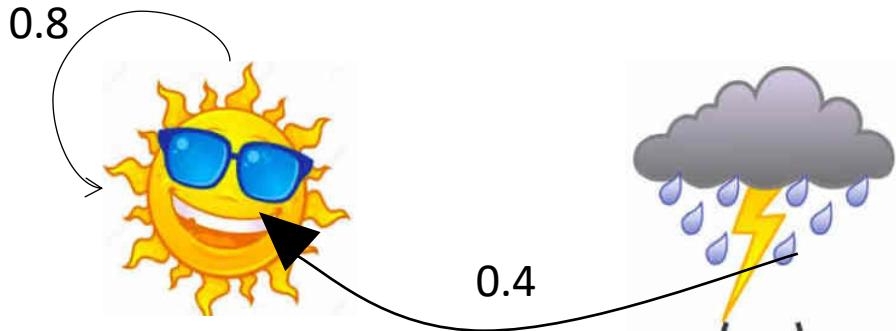


What is the probability that a random day is sunny or rainy

- Suppose vijay didnt want to talk to karan,now karan has to figure out what weather it is
- let us get the data back
- let us calculate probability of sunny or rainy
- looks cheating,lets use some other method



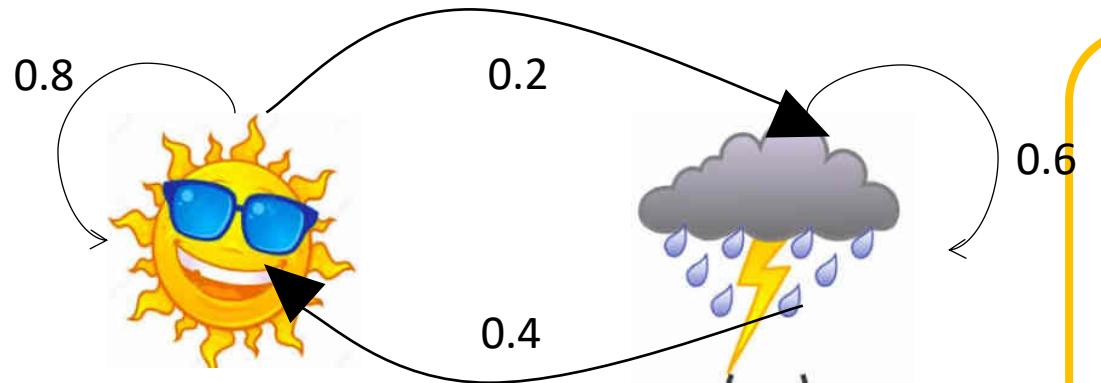
What is the probability that a random day is sunny or rainy



- if today is sunny it could be because yesterday was sunny ,or yesterday was rainy
- we can have the following equation

$$S=0.8S+0.4R$$

What is the probability that a random day is sunny or rainy



- similarly
- so now we can solve the system of these equation, but this two equation are almost same , but we know that $S+R=1$

$$S = 0.8S + 0.4R$$

$$R = 0.2S + 0.6R$$

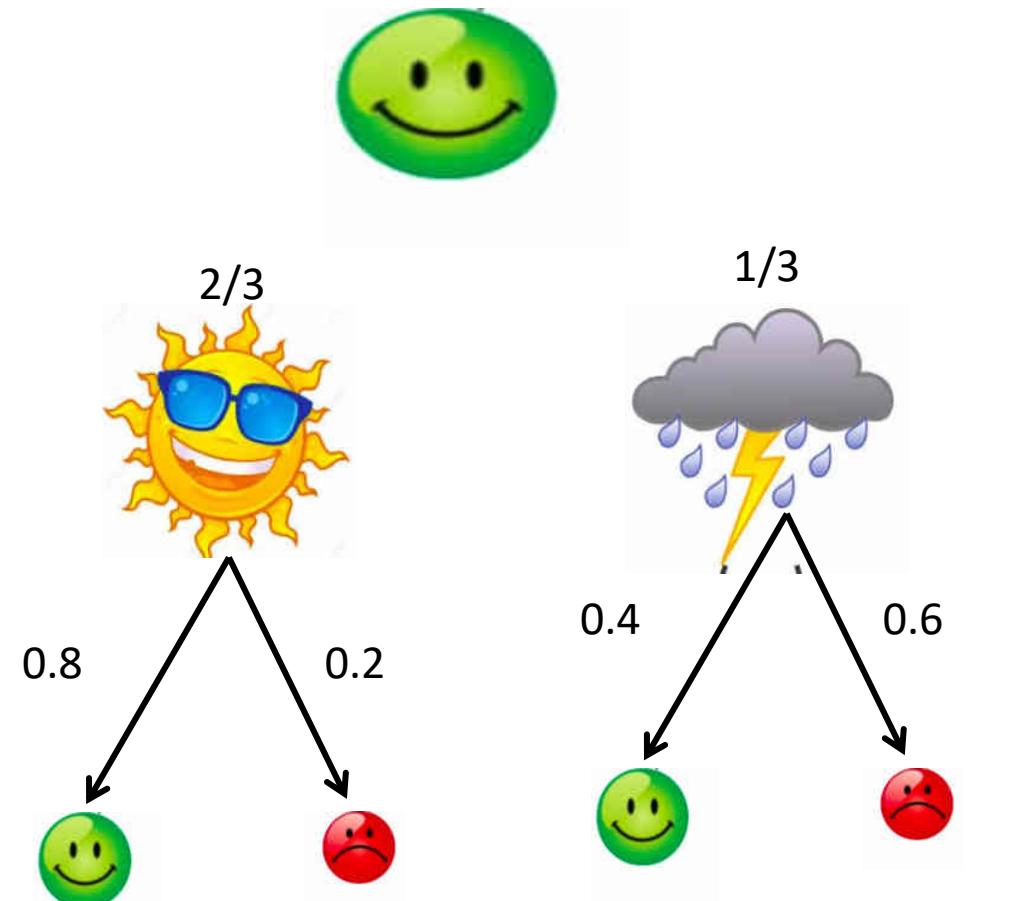
$$S+R=1$$

$$S=2/3 \quad R=1/3$$

- ~~How did we find these probabilities?~~
- ~~Whats the probability that a random day is sunny or rainy~~
- if Vijay is happy today,whats the probaility ,that its sunny or rainy?
- if for three days Vijay is Happy,Grumpy,Happy,what was the weather

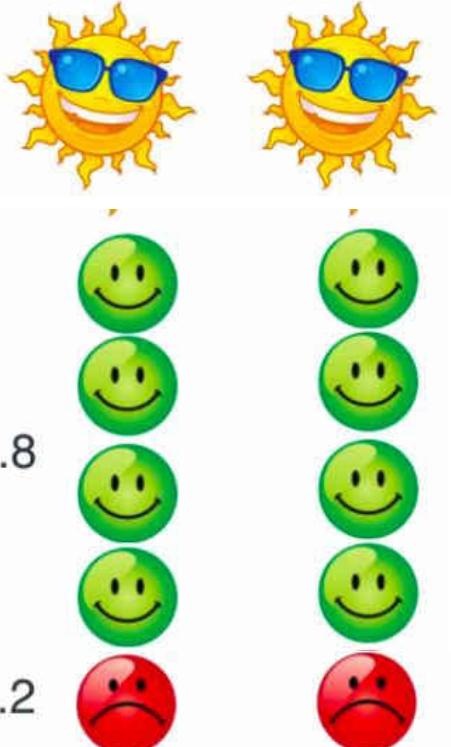


Question 3

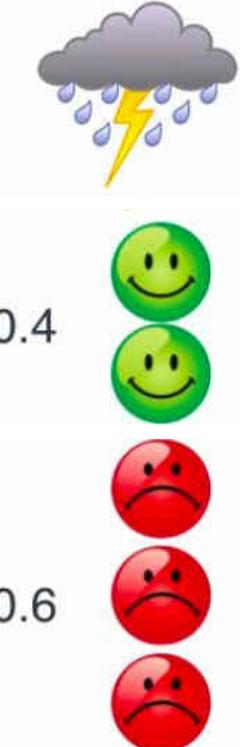


- we will use Bayes theorem
- suppose today Vijay is happy
- then there are two possibilities
- since we are talking about particular day we need not care of transition probability

2/3

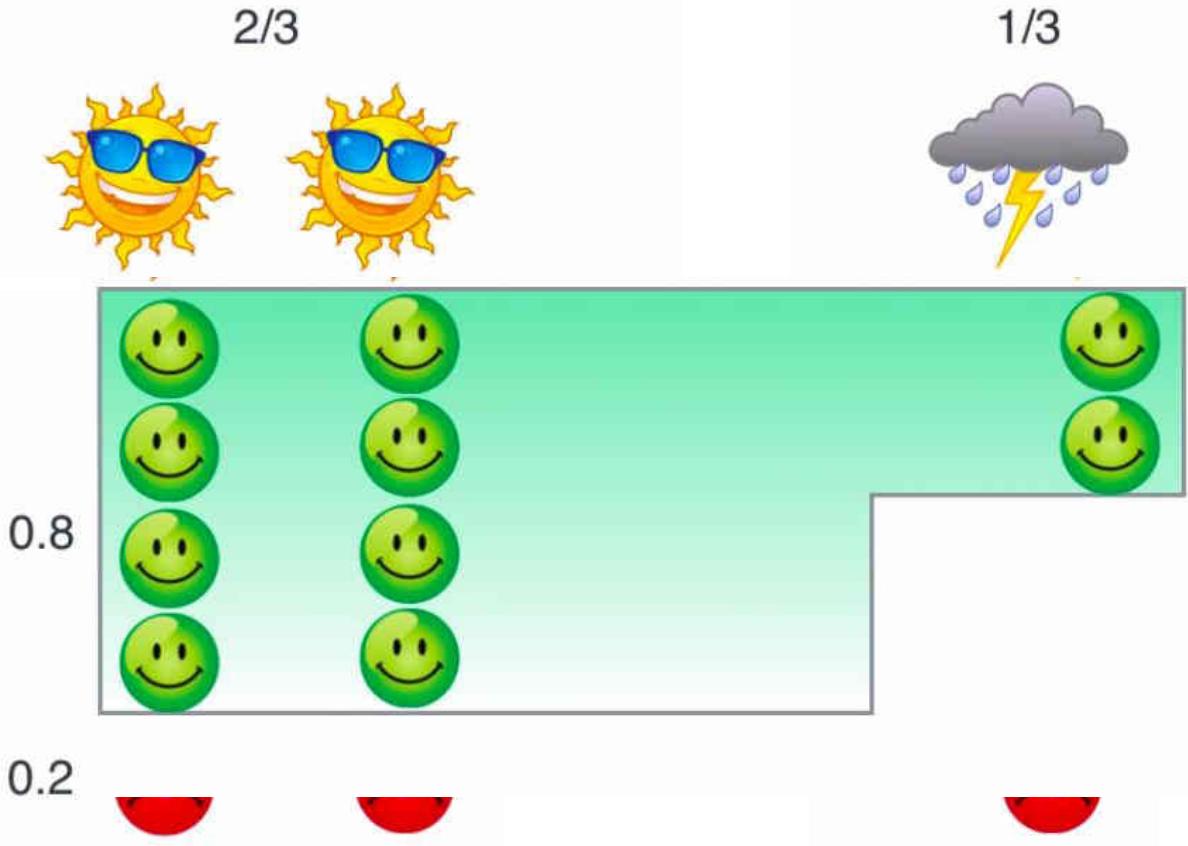


1/3



Bayes Theorem

If 



Bayes Theorem

If 

$$P(\text{☀️} \mid \text{😊}) = \frac{8}{10}$$

$$P(\text{🌧} \mid \text{😊}) = \frac{2}{10}$$

2/3



1/3



Bayes Theorem

If ☹

$$P(\text{☀️} | \text{☹}) = \frac{2}{5}$$

$$P(\text{☁️} | \text{☹}) = \frac{3}{5}$$



- ~~How did we find these probabilities?~~
- ~~Whats the probability that a random day is sunny or rainy~~
- ~~if Vijay is happy today,whats the probaility ,that its sunny or rainy?~~
- if for three days Vijay is Happy,Grumpy,Happy,what was the weather



- suppose vijay says happy grympy and happy for three consecutive days
- so karan is supposed to guess the weather now



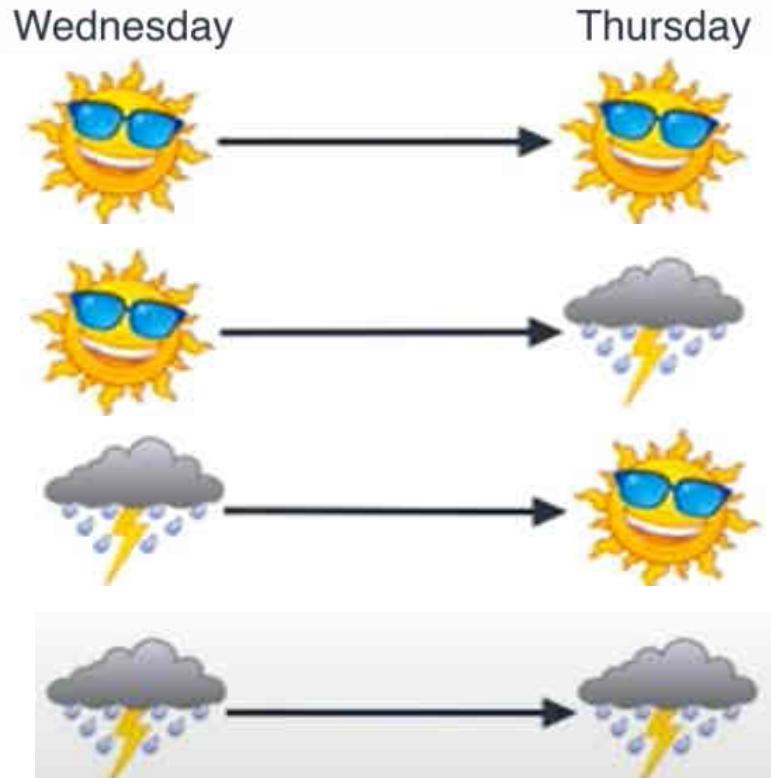
Wednesday



Thursday



- lets look at a simpler case suppose we have two days
- how many possible scenarios do we have for the weather ???
- so four cases to study



- what we will do is take each one of them and calculate probability that given these ways vijay was happy and then grumpy
- and then we are going to pick whatever gave us the highest

Wednesday



Thursday



- lets calculate the entire probability of all this happening at the same time

Wednesday

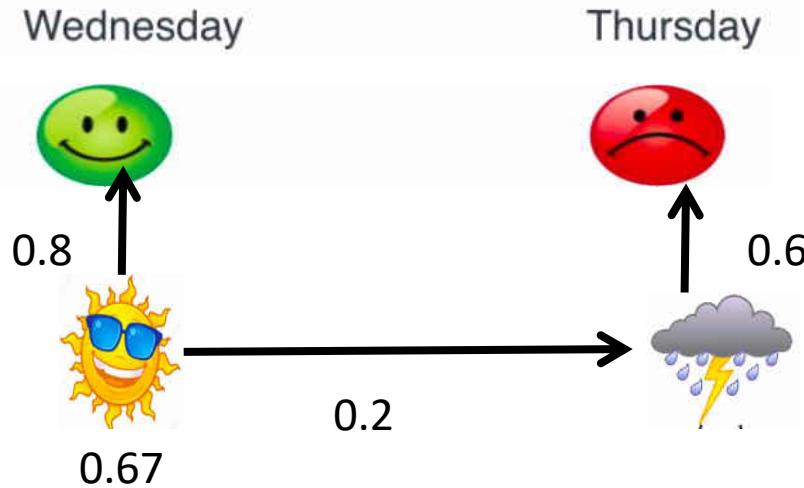


0.67

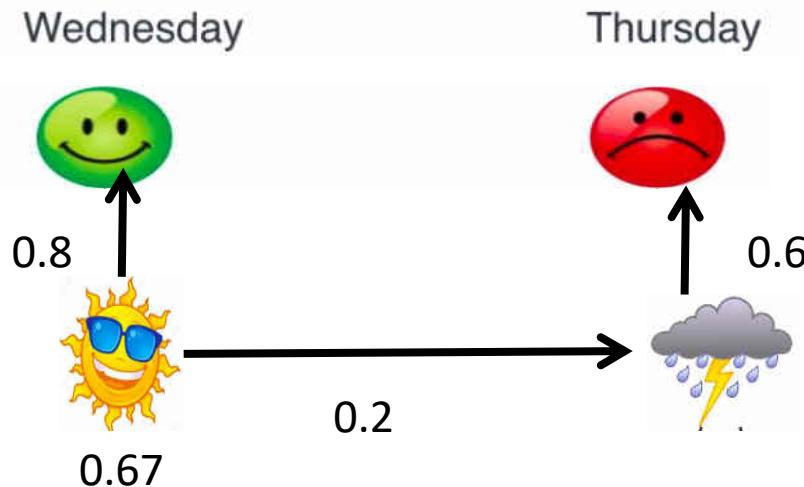
Thursday



- what is the probability wednesday was sunny
- we calculated previously that is $2/3=0.67$

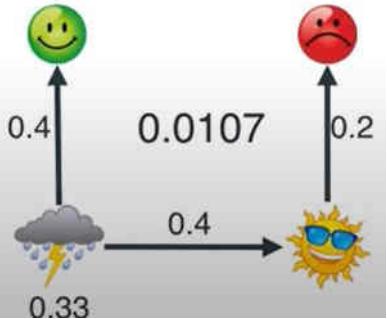
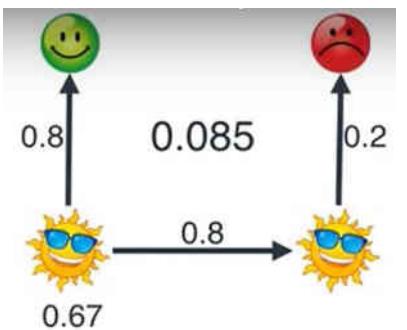


- now if wed was sunny what is the probability of vijay being happy
- now given wed was sunny what was the probability that thursday being rainy
- now thursday being rainy,what is the probabiltiy that made vijay grumpy

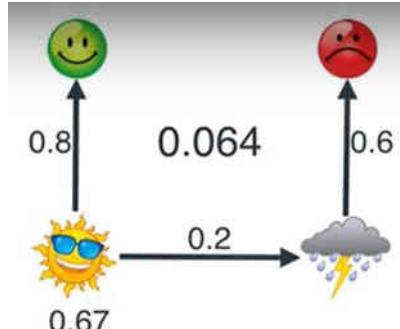
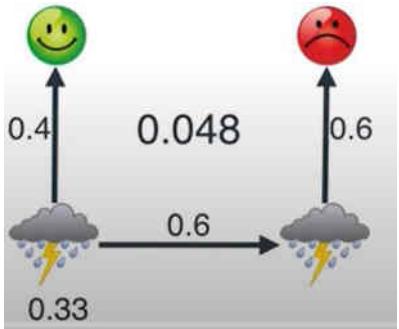


- now by law of conditional probability ,the probability of all this happening is product of all these

0.06432



- doing the same for all the four case we get this results
- we pick the one that is going to make happy grumpy likely happen



MACHINE INTELLIGENCE

HMM

- now back to our question what if vijay says Happy,grumpy,happy



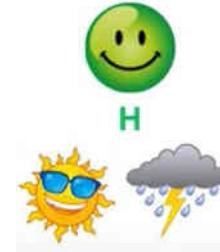
Day1



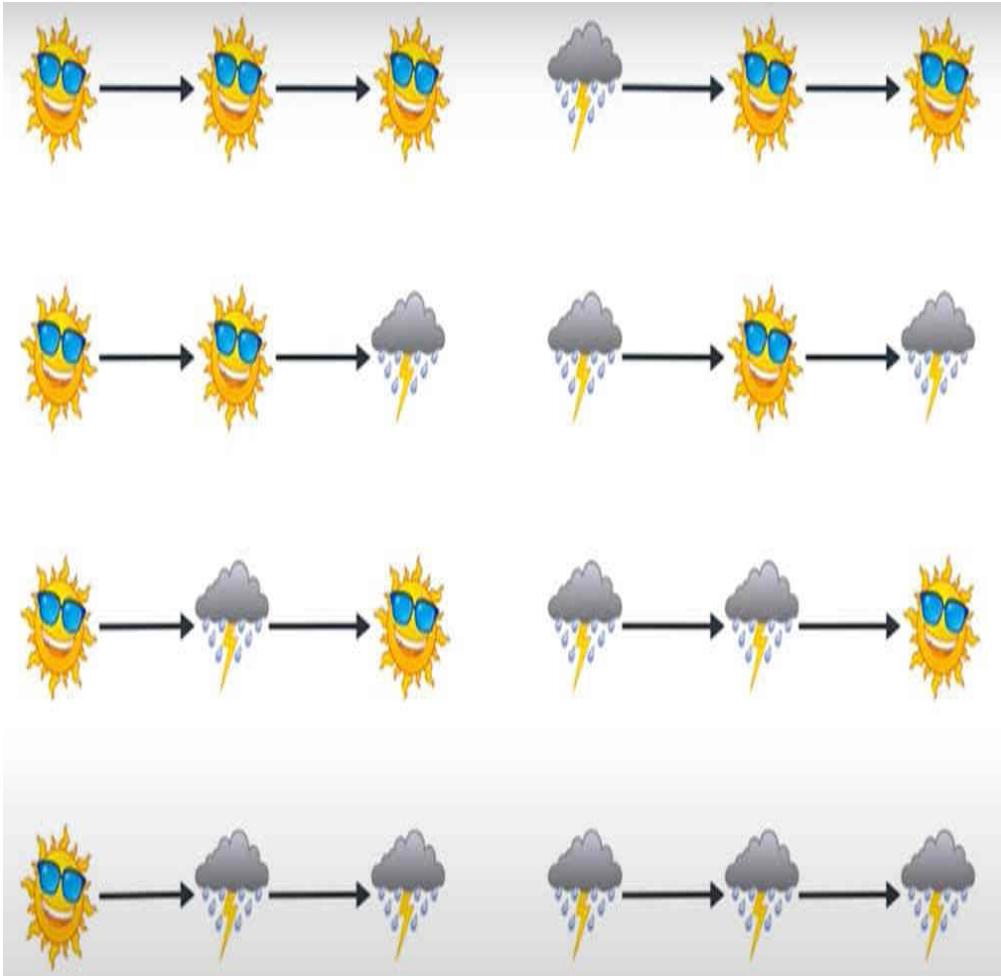
Day2



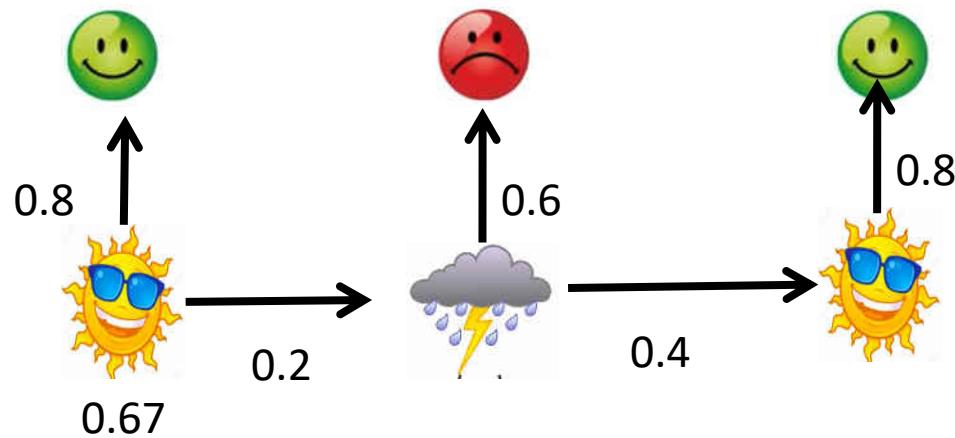
Day3



- same thing three days ,each with one mood ,and two possibilities for each day
- hence 8 possible scenario

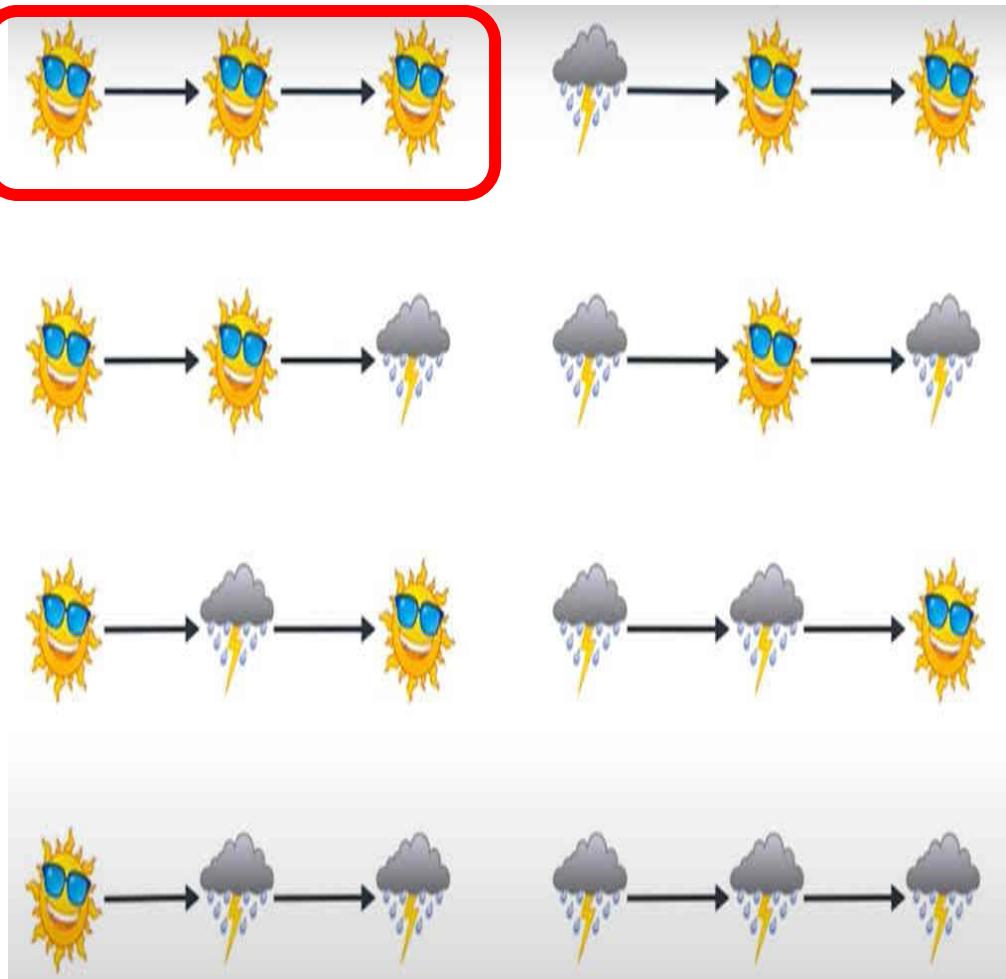


- same thing three days ,each with one mood ,and two possibilities for each day
- hence 8 possible scenario



- Let us consider this case, sunny, rainy, sunny

0.0205824



- doing the same process for all the cases, we get max probability for this



MACHINE INTELLIGENCE

Hidden Markov Model

K.S.Srinivas

Department of Computer Science and Engineering

MACHINE INTELLIGENCE

Hidden Markov Model - Estimation

K.S.Srinivas

Department of Computer Science and Engineering

Hidden Markov Model

- Until now we assumed that the instances that constitute a sample are **ID**

$$\text{Likelihood(sample)} = \prod \text{Likelihood(instance)}$$

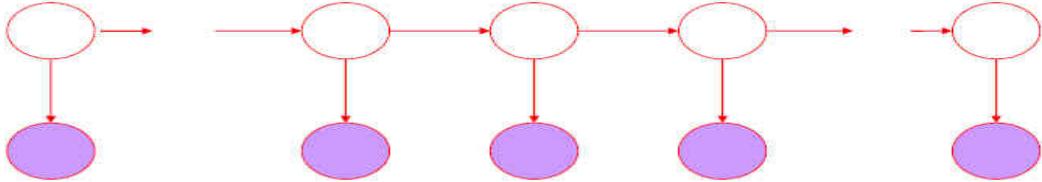
- But consider the following samples
 1. Dependence of letter in a word
 2. Dependence of base pairs in DNA sequence
 3. Dependence of successive phonemes in a speech
- With HMM, we determine the internal state by making observations

Hidden Markov Model

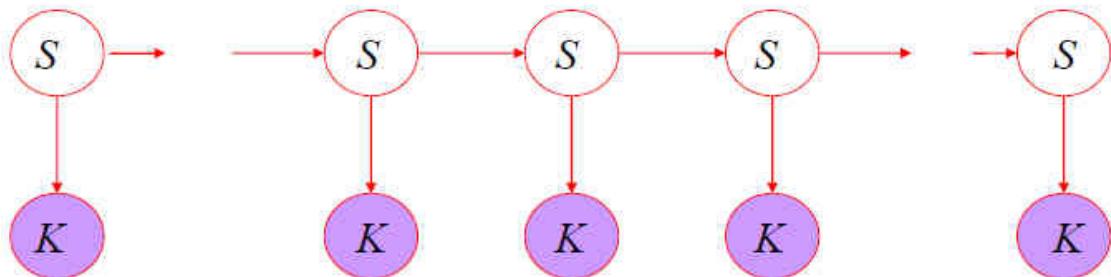
- HMM models a process with a First Order Markov process.
- It includes the initial state distribution π (the probability distribution of the initial state)
- The transition probabilities A from one state (x_t) to another.
- HMM also contains the likelihood B of the observation (y_t) given a hidden state. Matrix B is called the emission probabilities. It demonstrates the probability of our observation given a specific internal state.

Hidden states – Markov chain:
–Dependent only on the previous state
–“The past is independent of the future given the present.”

Hidden Markov Model - Formalism



- Shaded nodes are observed variables
- Dependent only on their corresponding hidden state

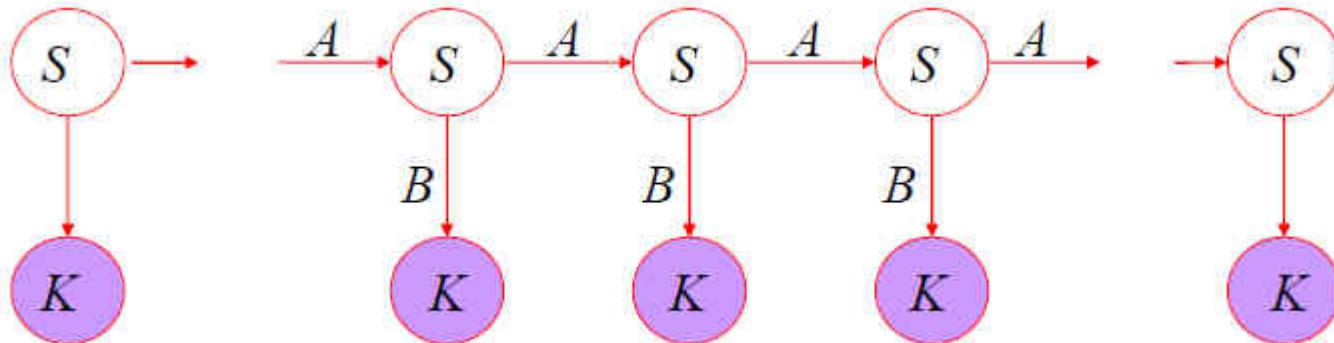


- $S : \{s_1 \dots s_N\}$ are the values for the hidden states
- $K : \{k_1 \dots k_M\}$ are the values for the observations

Hidden states – Markov chain:
– Dependent only on the previous state
– “The past is independent of the future given the present.”

N and M are defined implicitly

Hidden Markov Model - Formalism

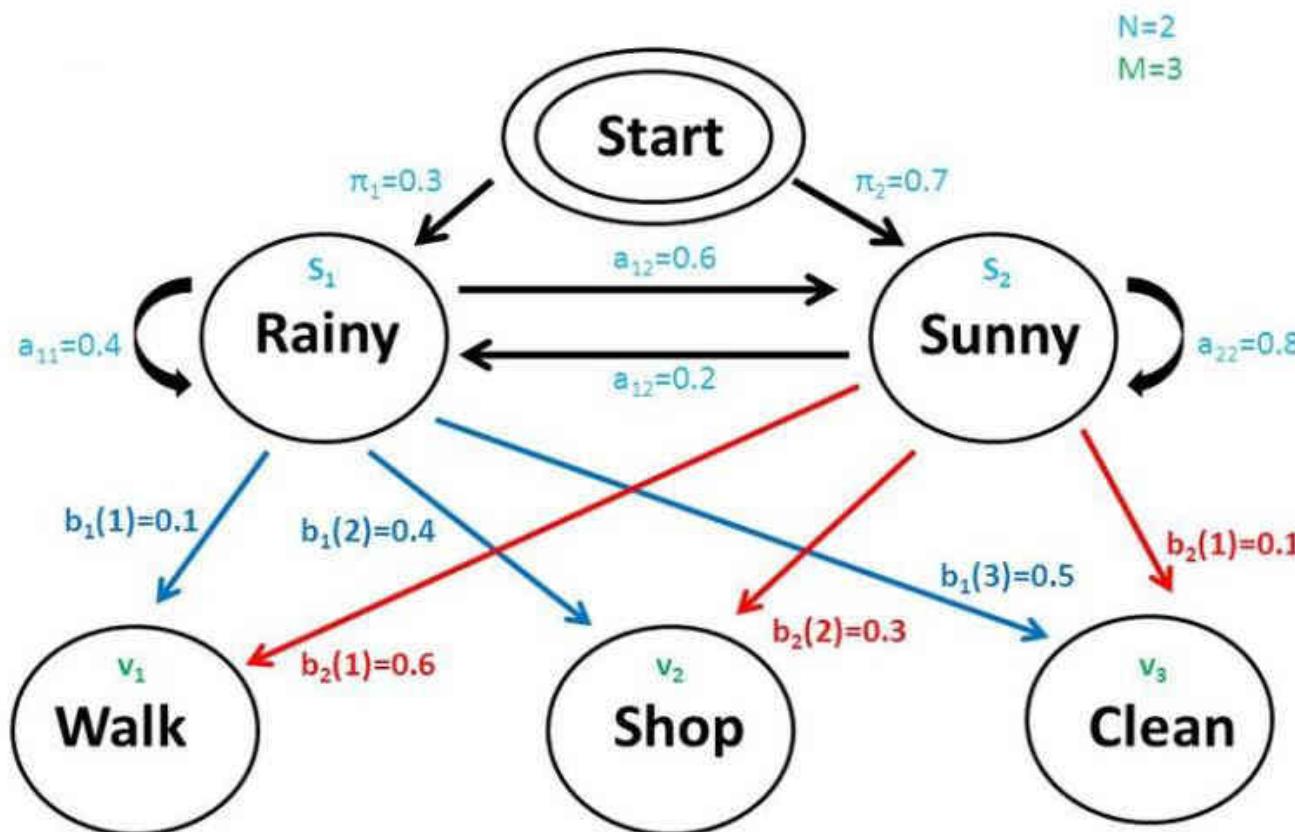


- Parameters: $\{S, K, P, A, B\}$
- Initial hidden state probabilities: $P = \{p_i\}$
- Transition probabilities. $A = \{a_{ij}\}$ are the state transition probabilities.
- Emission probabilities. $B = \{b_{ik}\}$ are the observation state probabilities (HMM can also work with continuous emission probabilities).

Hidden states – Markov chain:
–Dependent only on the previous state
–“The past is independent of the future given the present.”

N and M are defined implicitly

Hidden Markov Model



Parameters: $\{S, K, P, A, B\}$

- Initial hidden state probabilities: $P = \{\pi_1\}$
- Transition probabilities. $A = \{a_{ij}\}$ are the state transition probabilities.
- Emission probabilities. $B = \{b_{ik}\}$ are the observation state probabilities

The complexity of the problem is that the same observations may be originated from different states (happy or not).

 π $P(x_i)$ $P(x_i = \text{happy}) = 0.8$ $P(x_i = \text{sad}) = 0.2$

		x_{t+1}		$P(y_t x_t)$					
		Happy	Sad	movie	book	party	dinning		
x_t		Happy	0.99	0.01	Given being happy	0.2	0.2	0.4	0.2
		Sad	0.1	0.9	Given being sad	0.4	0.3	0.1	0.2

For example, $P(\text{Happy}_{t+1}|\text{Happy}_t) = 0.99$

 B $P(y_t|x_t)$:

	movie	book	party	dinning
Given being happy	0.2	0.2	0.4	0.2
Given being sad	0.4	0.3	0.1	0.2

Observation likelihoods or Emission probabilities B

Initial state distribution

 π Transition probability matrix A in Markov Processmodel λ .

Two major assumptions are made in HMM. The next state and the current observation solely depend on the current state only.

$$\begin{aligned} P(x_i | x_1, x_2, \dots, x_{i-1}) &= P(x_i | x_{i-1}) && \text{(Markov process)} \\ O(o_i | x_1, x_2, \dots, x_{i-1}) &= P(o_i | x_i) && \text{(Output independence)} \end{aligned}$$

$A + B = \text{HMM model } \lambda$

A

x_{t+1}

	Happy	Sad
Happy	0.99	0.01
Sad	0.1	0.9

For example, $P(\text{Happy}_{t+1} | \text{Happy}_t) = 0.99$

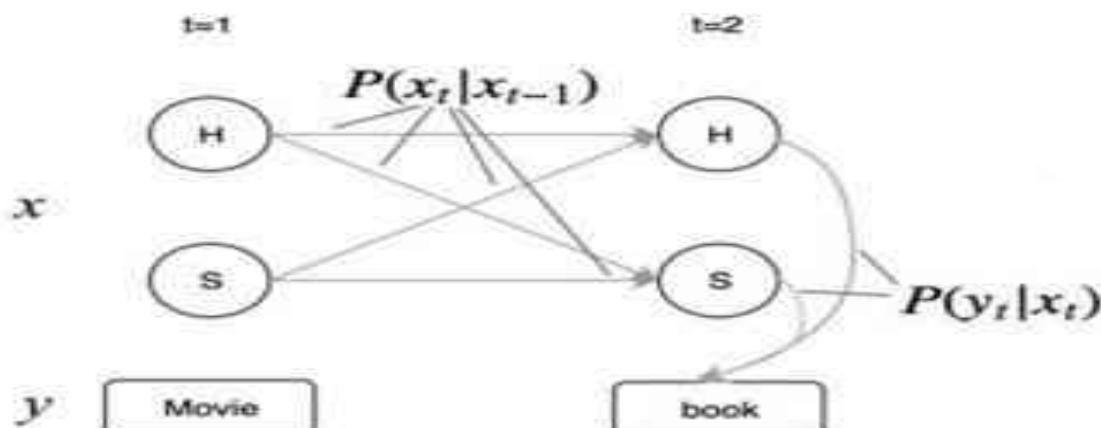
B

$P(y_t | x_t):$

	movie	book	party	dinning
Given being happy	0.2	0.2	0.4	0.2
Given being sad	0.4	0.3	0.1	0.2

Observation likelihoods or Emission probabilities B

- Given all the observable and the initial state distribution, we can compute a pretty complex equation for the probability for the internal state x_t $P(x_t | y_1, y_2, y_3, \dots, y_t)$ at time t.
- For simplicity here, we will not include π in our equation. All equations assume π is a given condition, like $P(y) \rightarrow P(y|\pi)$.



- The equation uses the transition probability and the emission probability to compute the probability of the internal state based on all observations.

$$P(x_t|y_{1:t}) = \frac{P(y_t|x_t) P(x_t|y_{1:t-1})}{\sum_{x_{t-1}} P(y_t|x_t) \sum_{x_{t-1}} P(x_t|x_{t-1}) P(x_{t-1}|y_{1:t-1})}$$

from time 1 to t

Depending on the situation, we usually ask three different types of questions regarding an HMM problem.

- **Likelihood:** How likely are the observations based on the current model or the probability of being at a state at a specific time step.
- **Decoding:** Find the internal state sequence based on the current model and observations.
- **Learning.** Learn the HMM model.

1. Probability of an Observation Sequence:

Given a model $\mu = (A, B)$ over S, K , how do we (efficiently) compute the likelihood of a particular sequence, $P(O|\mu)$?

2. Finding the “Best” State Sequence:

Given an observation sequence and a model, how do we choose a state sequence (X_1, \dots, X_{T+1}) to best explain the observation sequence?

3. HMM Parameter Estimation:

Given an observation sequence (or corpus thereof), how do we acquire a model $\mu = (A, B)$ that best explains the data?

Likelihood (likelihood of the observation)

Likelihood is to find the likelihood of observation Y.

$$p(Y) = \sum_X p(Y, X) = \sum_X p(Y | X) p(X)$$

calculated from emission probability calculated from transition probability

the observed events sum over all possible time sequences of internal states

This is computationally intense.

$$P(O|X, \mu) = \prod_{t=1}^T P(O_t|X_t, X_{t+1}, \mu) = b_{X_1 X_2 O_1} b_{X_2 X_3 O_2} \dots b_{X_T X_{T+1} O_T}$$

$$P(O, \mu) = \sum_X P(O|X, \mu) P(X, \mu) = \sum_{X_1 \dots X_{T+1}} \pi_{X_1} \prod_{t=1}^T a_{X_t X_{t+1}} b_{X_t X_{t+1} O_t}$$

Complexity : $(2T + 1)N^{T+1}$, too inefficient

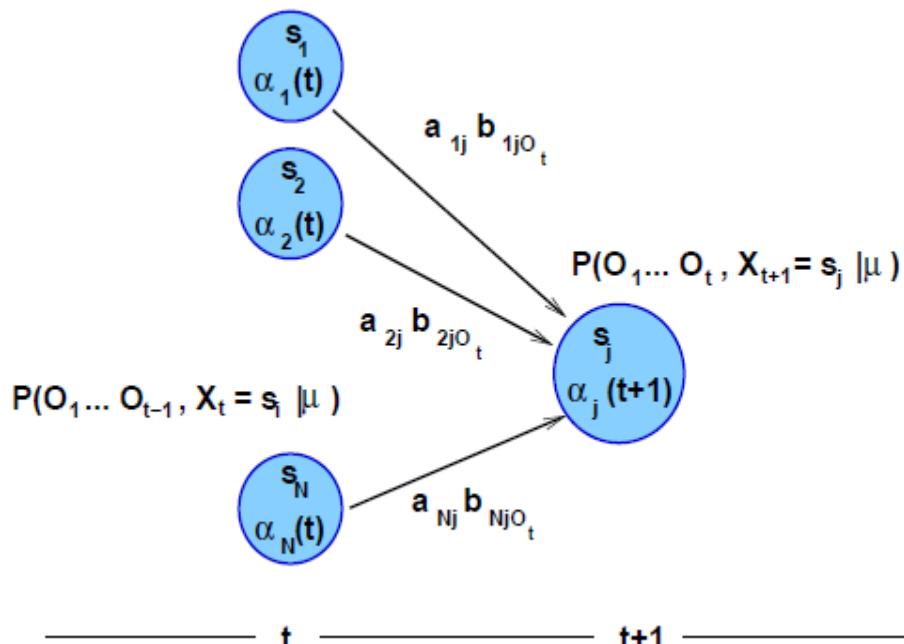
better : use dynamic prog. to store partial results

$$\alpha_i(t) = P(O_1 O_2 \dots O_{t-1}, X_t = s_i | \mu).$$

Probability of an Observation Sequence

$$\alpha_i(t) = P(O_1 O_2 \dots O_t, X_t = s_i | \mu).$$

$\alpha_i(t)$ is the probability of a machine being in state i at time t and producing observation $o_1, o_2, o_3, \dots, o_{t-1}$

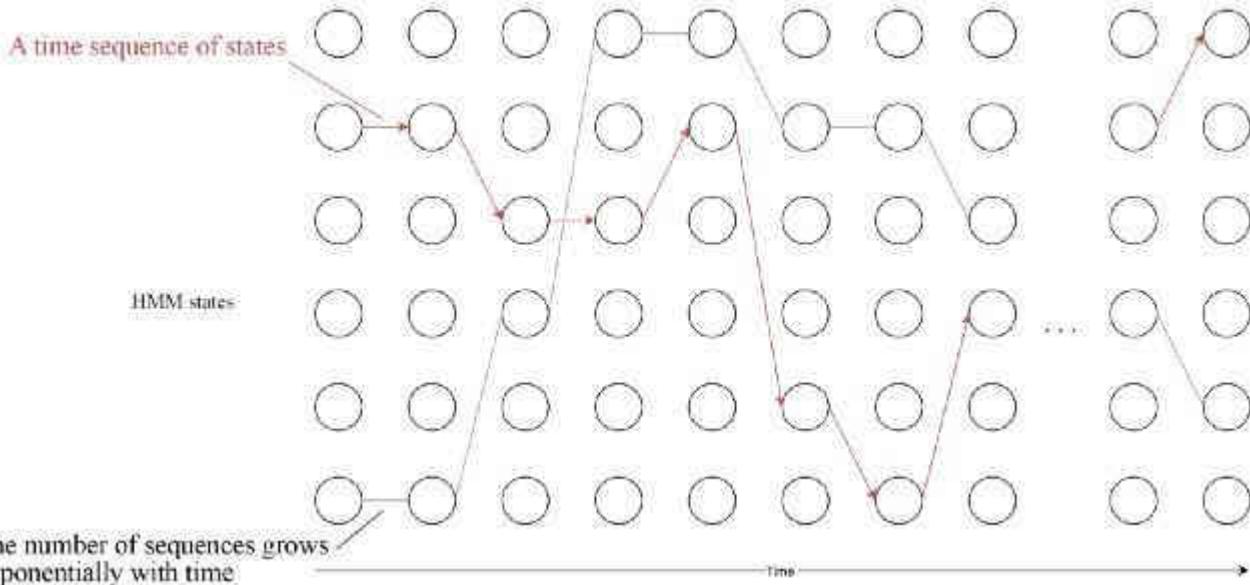


- The probability of moving from state S_i to state S_j is given by the probability in the transition matrix A more formally as a_{ij}
- Having moved to S_j the probability of emitting O_{t+1} is given by $b_j(O_{t+1})$

Likelihood (likelihood of the observation)

In HMM, we solve the problem at time t by using the result from time t-1

- A circle below represents an HMM hidden state j at time t. So even the number of state sequence increases exponentially with time, we can solve it linear if we can express the calculation recursively with



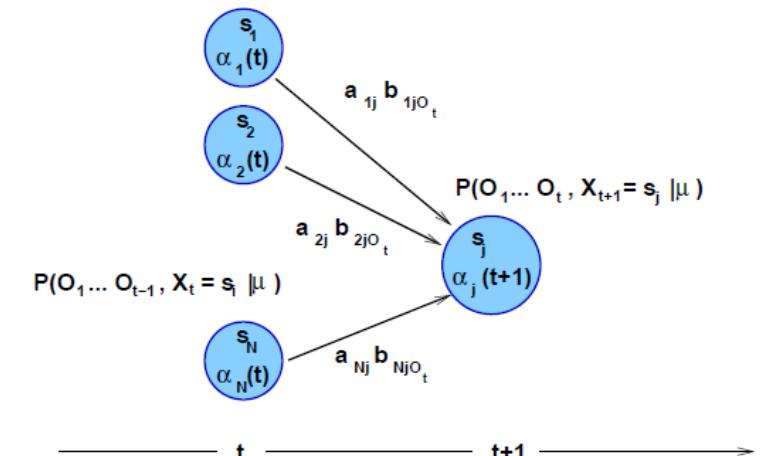
- This is the idea of **dynamic programming** that breaks the **exponential curse**

Probability of an Observation Sequence

$$\alpha_i(t) = P(O_1 O_2 \dots O_t, X_t = s_i | \mu).$$

As we can see from the diagram on the right as we explained earlier we express this recursively in terms of the earlier α 's

$$\text{Induction: } \alpha_j(t+1) = \sum_{i=1}^N \alpha_i(t) a_{ij} b_{ij O_t}, \quad 1 \leq t \leq T, \quad 1 \leq j \leq N$$



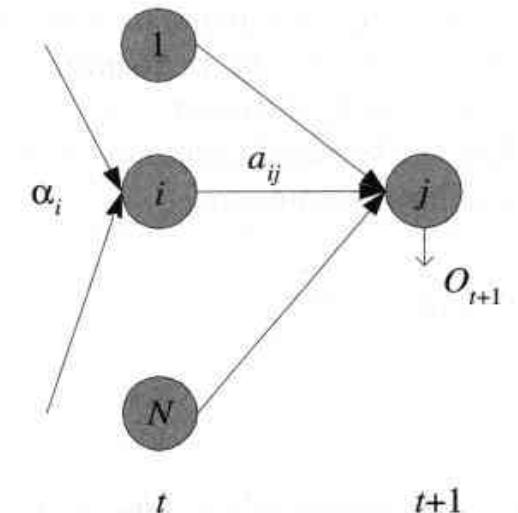
We Will prove this in our next slide and explain this with an examples

Probability of an Observation Sequence

$$\alpha_i(t) = P(O_1 O_2 \dots O_t, X_t = s_i | \mu).$$

As we can see from the diagram on the right as we explained earlier we can express this recursively in terms of the earlier α 's

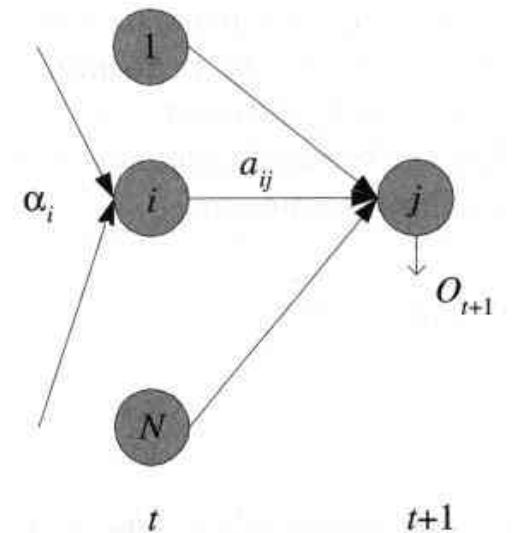
$$\text{Induction: } \alpha_j(t+1) = \sum_{i=1}^N \alpha_i(t) a_{ij} b_{ij} o_t, \quad 1 \leq t \leq T, \quad 1 \leq j \leq N$$



We Will prove this in our next slides, explain the forward algorithm and explain this with an examples

Proof of the alpha probability

$$\begin{aligned}
 \alpha_j(t+1) &= P(O_1 O_2 \dots O_{t-1} O_t, X_{t+1} = j | \mu) \\
 &= \sum_{i=1}^N P(O_1 O_2 \dots O_{t-1} O_t, X_t = i, X_{t+1} = j | \mu) \\
 &= \sum_{i=1}^N P(O_t, X_{t+1} = j | O_1 O_2 \dots O_{t-1}, X_t = i, \mu) P(O_1 O_2 \dots O_{t-1}, X_t = i | \mu) \\
 &= \sum_{i=1}^N P(O_1 O_2 \dots O_{t-1}, X_t = i | \mu) P(O_t, X_{t+1} = j | O_1 O_2 \dots O_{t-1}, X_t = i, \mu) \\
 &= \sum_{i=1}^N \alpha_i(t) P(O_t, X_{t+1} = j | X_t = i, \mu) \\
 &= \sum_{i=1}^N \alpha_i(t) P(O_t | X_t = i, X_{t+1} = j, \mu) P(X_{t+1} = j | X_t = i, \mu) = \sum_{i=1}^N \alpha_i(t) b_{ij} o_t a_{ij}
 \end{aligned}$$



The forward algorithm

Thus the likelihood of the observations can be calculated recursively for each time step below.:

1. initialize

$$\alpha_1(j) = \pi_j b_j(y_1)$$

initial state distribution

probability of observing y_1 given state j

2. For each time step

$$\alpha_t(j) = \sum_{i=1}^N \alpha_{t-1}(i) a_{ij} b_j(y_t)$$

transition probability

sum over all states

probability of observing y_t given current state = j

probability of all previous observations give last state i

3. Result

$$P(Y|\lambda) = \sum_{i=1}^N \alpha_T(i)$$

all observations

final step

sum over all possible state

Toy Example of Forward Algorithm

At time t, the probability of our observations up to time t is:

$$P(y_1, y_2, \dots, y_t | \lambda) = \sum_j P(y_1, y_2, \dots, y_t, x_t = s_j | \lambda)$$

HMM model parameters

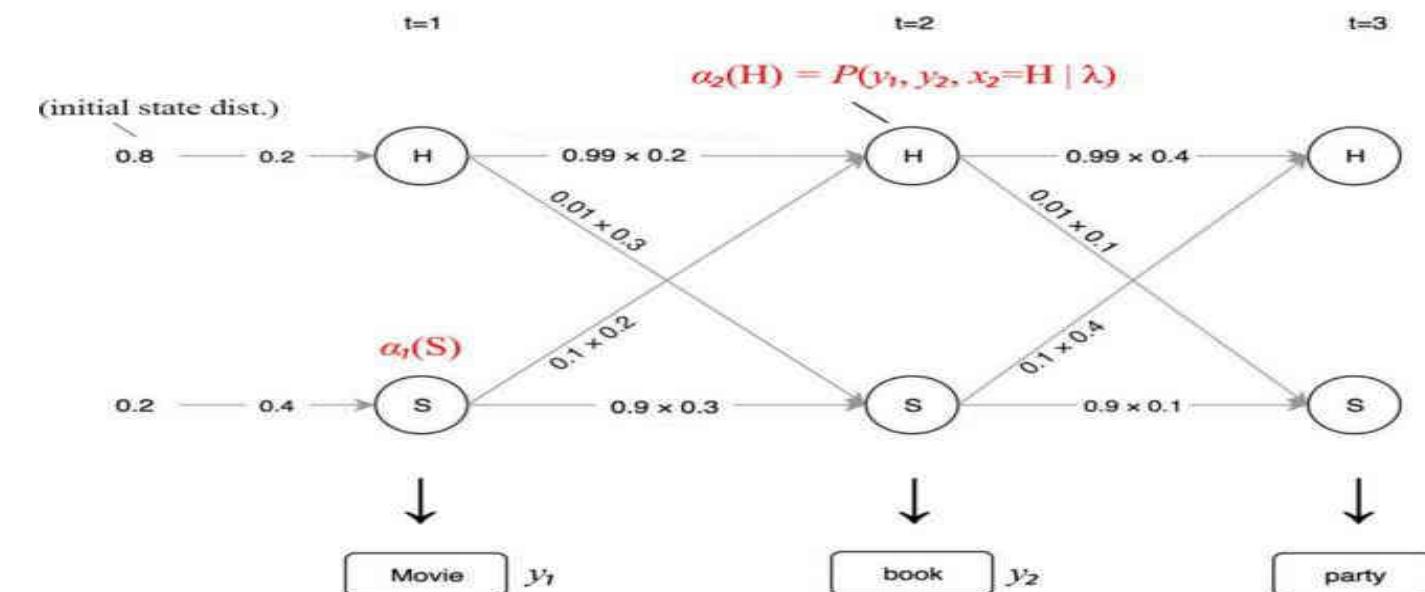
observations up to time t sum over all internal states $\alpha_t(j)$

- Let's rename the term underlined in red above as $\alpha_t(j)$ (forward probability) and we can express it recursively.

$$\alpha_t(j) = \sum_{i=1}^N \alpha_{t-1}(i) a_{ij} b_j(y_t)$$

transition probability from state i to j

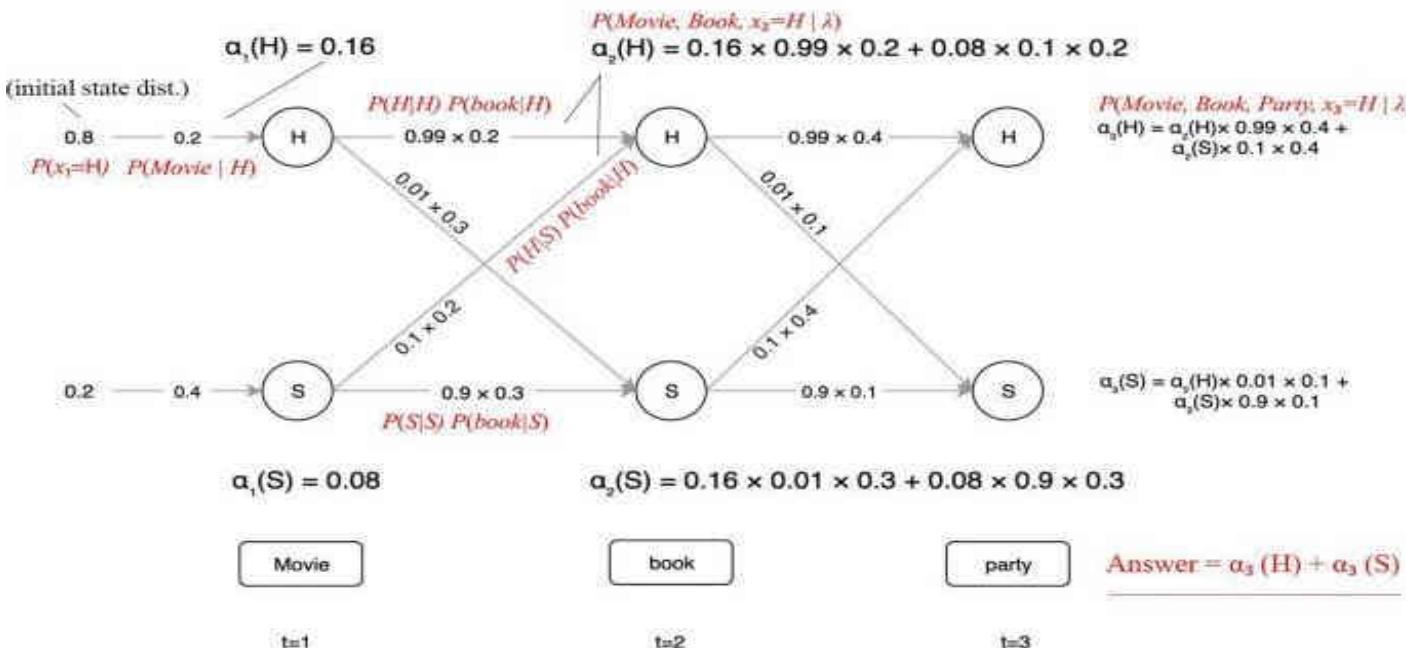
sum over all internal states emission probability



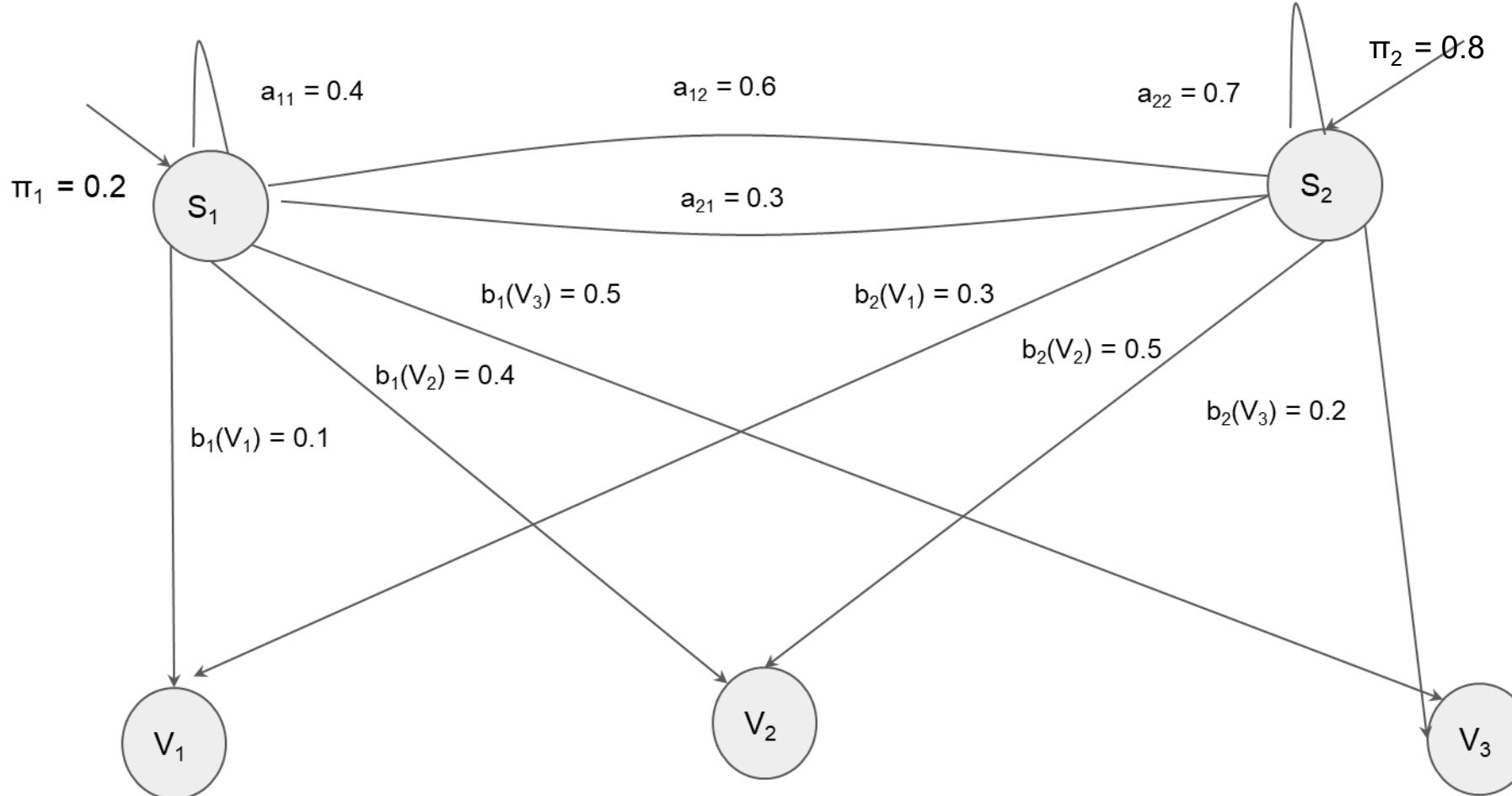
- Consider this example in which we start with the initial state distribution on the left.
- Then we propagate the value of α to the right for each timestep.
- Therefore, we break the curse of exponential complexity.

		x_{t+1}	
		Happy	Sad
x_t	Happy	0.99	0.01
	Sad	0.1	0.9

$P(y_t x_t)$:				
	movie	book	party	dinning
Given being happy	0.2	0.2	0.4	0.2
Given being sad	0.4	0.3	0.1	0.2



HMM – Canonical Example Problem



Alpha rule method to find Probability of an Observation Sequence

To Find $P(O = \{V_1, V_3, V_2\} | \lambda)$, make alpha table, sum up last column

	t = 1 (V_1 is observed)	t = 2 (V_3 is observed)	t = 3 (V_2 is observed)
S_1			
S_2			

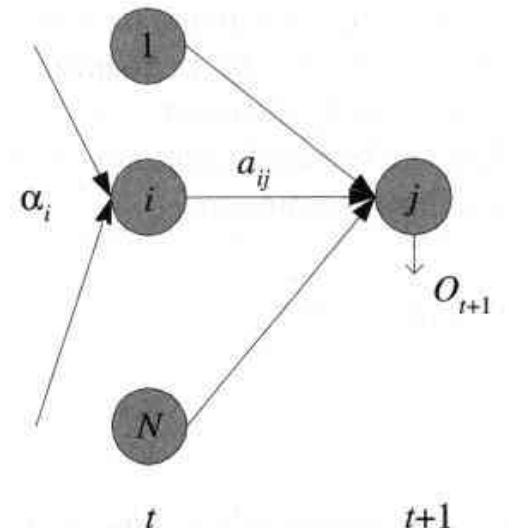
To calculate some cell, take previous time step alpha values and multiply each with transition probability of corresponding cells and add them up. $(\sum \alpha_t(i) * a_{ij})$. Multiply this sum with observation probability $b_i(O_{t+1})$ to get $(\sum \alpha_t(i) * a_{ij}) * b_i(O_{t+1}) = \alpha_{t+1}$ at this cell.

Alpha rule method to find Probability of an Observation Sequence

To Find $P(O = \{V_1, V_3, V_2\} | \lambda)$, make alpha table, sum up last column

	$t=1 (V_1 \text{ is observed})$	$t=2 (V_3 \text{ is observed})$	$t=3 (V_2 \text{ is observed})$
S_1	α_i		
S_2			

To calculate this cell, take previous time step alpha values and multiply each with transition probability of corresponding cells and add them up. But, This is first column. $\alpha_1(i) = \pi_i * b_i(O_1)$



MACHINE INTELLIGENCE

HMM

Alpha rule method to find Probability of an Observation Sequence

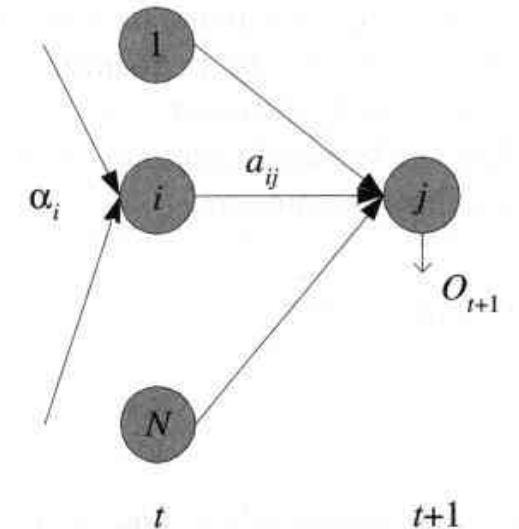
To Find $P(O = \{V_1, V_3, V_2\} | \lambda)$, make alpha table, sum up last column

	t = 1 (V_1 is observed)	t = 2 (V_3 is observed)	t = 3 (V_2 is observed)
S_1			
S_2			

$$\pi_i = \pi_1 = 0.2$$

$$b_i(O_1) = b_1(V_1) = 0.1$$

$$\alpha_1(i) = \pi_i * b_i(O_1) = 0.02$$



Alpha rule method to find Probability of an Observation Sequence

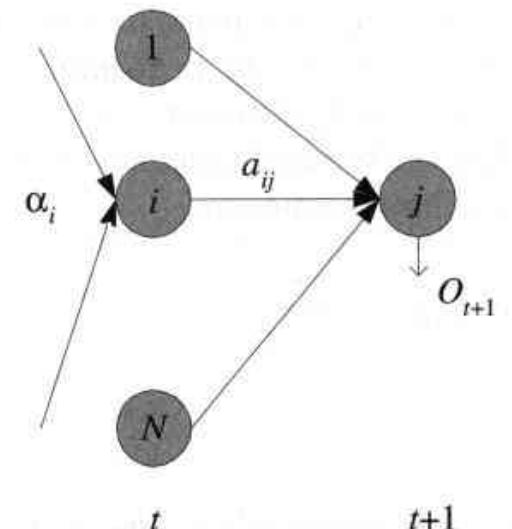
To Find $P(O = \{V_1, V_3, V_2\} | \lambda)$, make alpha table, sum up last column

	t = 1 (V_1 is observed)	t = 2 (V_3 is observed)	t = 3 (V_2 is observed)
S_1	0.02		
S_2			

$$\pi_i = \pi_2 = 0.8$$

$$b_i(O_1) = b_2(V_1) = 0.3$$

$$\alpha_1(i) = \pi_i * b_i(O_1) = 0.24$$



Alpha rule method to find Probability of an Observation Sequence

To Find $P(O = \{V_1, V_3, V_2\} | \lambda)$, make alpha table, sum up last column

	t = 1 (V_1 is observed)	t = 2 (V_3 is observed)	t = 3 (V_2 is observed)
S_1	0.02		
S_2	0.24		

To calculate this cell, take previous time step alpha values and multiply each with transition probability of corresponding cells and add them up. ($\sum \alpha_t(i) * a_{ij}$). Multiply this sum with observation probability $b_i(O_{t+1})$ to get $(\sum \alpha_t(i) * a_{ij}) * b_i(O_{t+1}) = \alpha_{t+1}$ at this cell.

Alpha rule method to find Probability of an Observation Sequence

To Find $P(O = \{V_1, V_3, V_2\} | \lambda)$, make alpha table, sum up last column

	$t=1 (V_1 \text{ is observed})$ $a_{11} = 0.4$	$t=2 (V_3 \text{ is observed})$	$t=3 (V_2 \text{ is observed})$
S_1	0.02		
S_2	0.24		

$$(\sum \alpha_t(i) * a_{ij}) = (0.02 * 0.4 + 0.24 * 0.3) = 0.08$$

$$b_t(O_{t+1}) = b_1(V_3) = 0.5$$

$$\alpha_{t+1} \text{ at this cell} = (\sum \alpha_t(i) * a_{ij}) * b_i(O_{t+1}) = 0.04$$

Alpha rule method to find Probability of an Observation Sequence

To Find $P(O = \{V_1, V_3, V_2\} | \lambda)$, make alpha table, sum up last column

	$t=1 (V_1 \text{ is observed})$ $a_{11} = 0.4$	$t=2 (V_3 \text{ is observed})$	$t=3 (V_2 \text{ is observed})$
S_1	0.02		
S_2	0.24		

$$(\sum \alpha_t(i) * a_{ij}) = (0.02 * 0.4 + 0.24 * 0.3) = 0.08$$

$$b_t(O_{t+1}) = b_1(V_3) = 0.5$$

$$\alpha_{t+1} \text{ at this cell} = (\sum \alpha_t(i) * a_{ij}) * b_i(O_{t+1}) = 0.04$$

Alpha rule method to find Probability of an Observation Sequence

To Find $P(O = \{V_1, V_3, V_2\} | \lambda)$, make alpha table, sum up last column

	$t=1 (V_1 \text{ is observed})$ $a_{11} = 0.4$	$t=2 (V_3 \text{ is observed})$	$t=3 (V_2 \text{ is observed})$
S_1	0.02		
S_2	0.24		

$$(\sum \alpha_t(i) * a_{ij}) = (0.02 * 0.4 + 0.24 * 0.3) = 0.08$$

$$b_t(O_{t+1}) = b_1(V_3) = 0.5$$

$$\alpha_{t+1} \text{ at this cell} = (\sum \alpha_t(i) * a_{ij}) * b_i(O_{t+1}) = 0.04$$

Alpha rule method to find Probability of an Observation Sequence

To Find $P(O = \{V_1, V_3, V_2\} | \lambda)$, make alpha table, sum up last column

	t = 1 (V_1 is observed)	t = 2 (V_3 is observed)	t = 3 (V_2 is observed)
S_1	0.02	0.04	$a_{11} = 0.4$
S_2	0.24	0.036	$a_{21} = 0.3$

$$(\sum \alpha_t(i) * a_{ij}) = (0.04 * 0.4 + 0.036 * 0.3) = 0.0268$$

$$b_t(O_{t+1}) = b_1(V_2) = 0.4$$

$$\alpha_{t+1} \text{ at this cell} = (\sum \alpha_t(i) * a_{ij}) * b_i(O_{t+1}) = 0.01072$$

Alpha rule method to find Probability of an Observation Sequence

Computing $\alpha_t(i)$ is $\mathcal{O}(N^2 T)$

To Find $P(O = \{V_1, V_3, V_2\} | \lambda)$, make alpha table, sum up last column

	t = 1 (V_1 is observed)	t = 2 (V_3 is observed)	t = 3 (V_2 is observed)
S_1	0.02	0.04	0.01072
S_2	0.24	0.036	

$$(\sum \alpha_t(i) * a_{ij}) = (0.04 * 0.6 + 0.036 * 0.7) = 0.0492$$

$$b_t(O_{t+1}) = b_2(V_2) = 0.5$$

$$\alpha_{t+1} \text{ at this cell} = (\sum \alpha_t(i) * a_{ij}) * b_i(O_{t+1}) = 0.0246$$



MACHINE INTELLIGENCE

Hidden Markov Model

K.S.Srinivas

Department of Computer Science and Engineering

MACHINE INTELLIGENCE

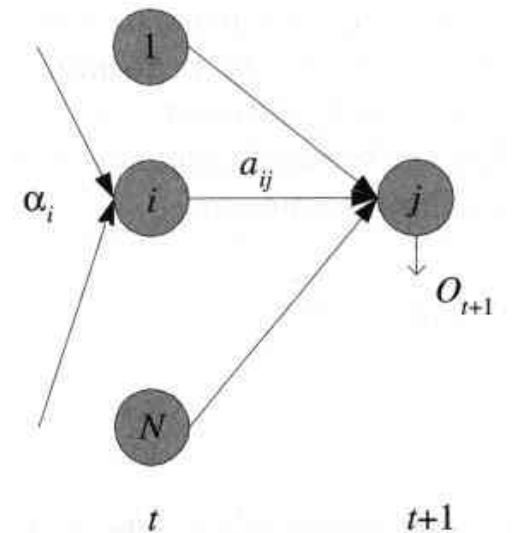
Decoding – Hidden Markov Model

K.S.Srinivas

Department of Computer Science and Engineering

Proof of the alpha probability

$$\begin{aligned}
 \alpha_j(t+1) &= P(O_1 O_2 \dots O_{t-1} O_t, X_{t+1} = j | \mu) \\
 &= \sum_{i=1}^N P(O_1 O_2 \dots O_{t-1} O_t, X_t = i, X_{t+1} = j | \mu) \\
 &= \sum_{i=1}^N P(O_t, X_{t+1} = j | O_1 O_2 \dots O_{t-1}, X_t = i, \mu) P(O_1 O_2 \dots O_{t-1}, X_t = i | \mu) \\
 &= \sum_{i=1}^N P(O_1 O_2 \dots O_{t-1}, X_t = i | \mu) P(O_t, X_{t+1} = j | O_1 O_2 \dots O_{t-1}, X_t = i, \mu) \\
 &= \sum_{i=1}^N \alpha_i(t) P(O_t, X_{t+1} = j | X_t = i, \mu) \\
 &= \sum_{i=1}^N \alpha_i(t) P(O_t | X_t = i, X_{t+1} = j, \mu) P(X_{t+1} = j | X_t = i, \mu) = \sum_{i=1}^N \alpha_i(t) b_{ij} o_t a_{ij}
 \end{aligned}$$



The forward algorithm

Thus the likelihood of the observations can be calculated recursively for each time step below.:

1. initialize

$$\alpha_1(j) = \pi_j b_j(y_1)$$

initial state distribution

probability of observing y_1 given state j

2. For each time step

$$\alpha_t(j) = \sum_{i=1}^N \alpha_{t-1}(i) a_{ij} b_j(y_t)$$

transition probability

sum over all states

probability of observing y_t given current state = j

probability of all previous observations give last state i

3. Result

$$P(Y|\lambda) = \sum_{i=1}^N \alpha_T(i)$$

all observations

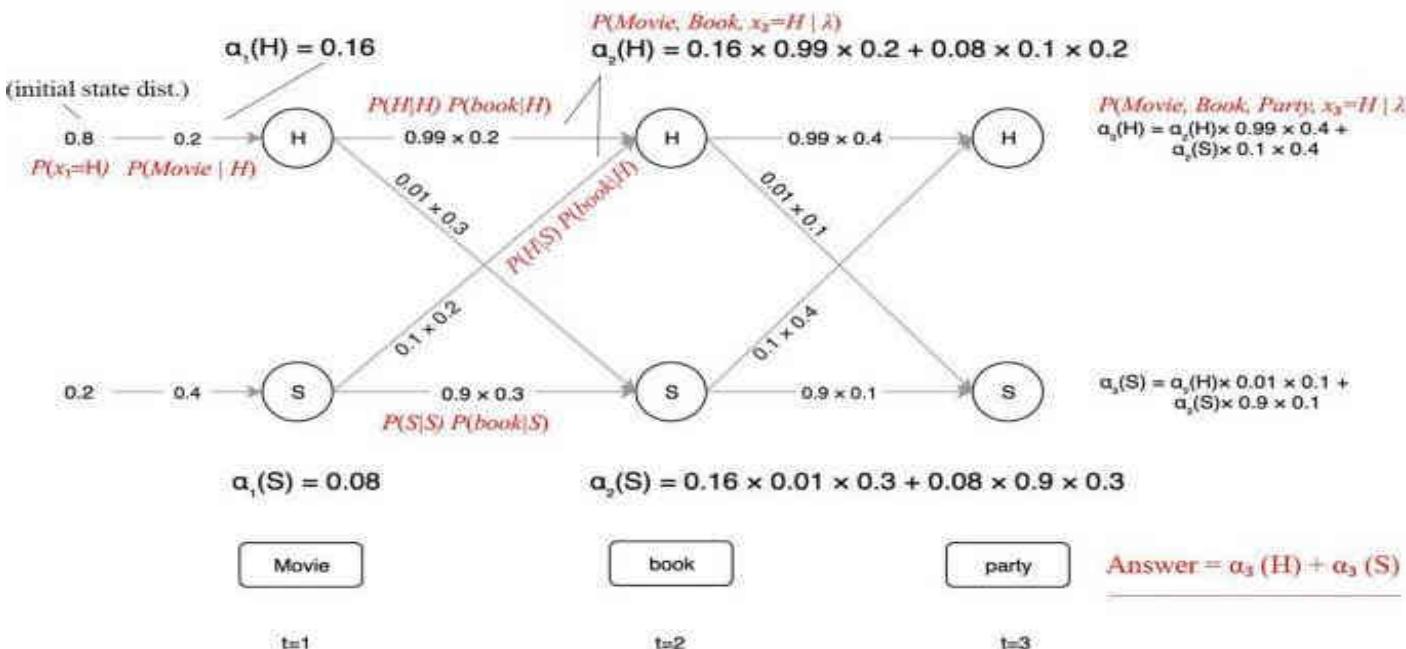
final step

sum over all possible state

- Consider this example in which we start with the initial state distribution on the left.
- Then we propagate the value of α to the right for each timestep.
- Therefore, we break the curse of exponential complexity.

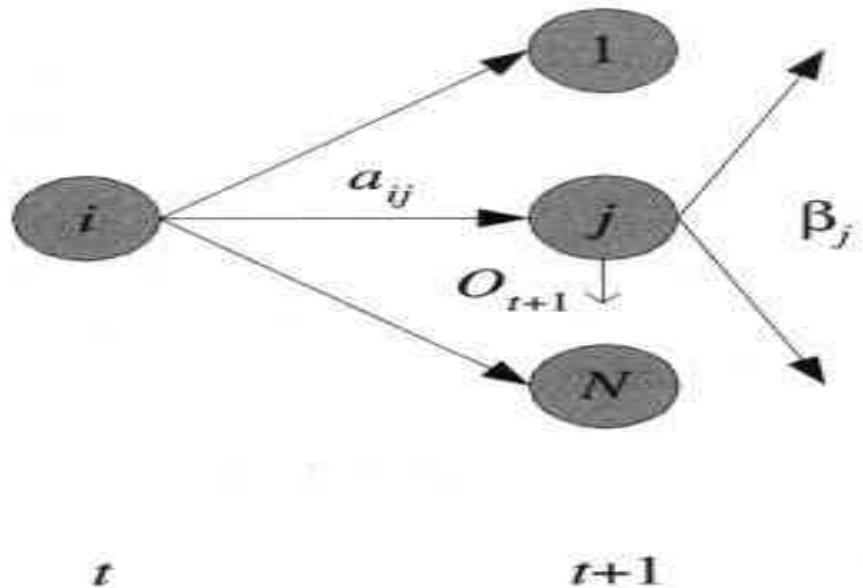
		x_{t+1}	
		Happy	Sad
x_t	Happy	0.99	0.01
	Sad	0.1	0.9

$P(y_t x_t)$:				
	movie	book	party	dinning
Given being happy	0.2	0.2	0.4	0.2
Given being sad	0.4	0.3	0.1	0.2



Backward Probability

The backward **probability** β is the probability of seeing the observations from time $t+1$ to the end, given that we are in state i at time t (and given the automaton I):



$$\text{Induction: } \beta_i(t) = \sum_{j=1}^N a_{ij} b_{ij} c_j \beta_j(t+1), \quad 1 \leq t \leq T, \quad 1 \leq i \leq N$$

Backward Probability Proof

Induction:

$$\begin{aligned}
 \beta_i(t) &= P(O_t O_{t+1} \dots O_T | X_t = i, \mu) \\
 &= \sum_{j=1}^N P(O_t O_{t+1} \dots O_T, X_{t+1} = j | X_t = i, \mu) \\
 &= \sum_{j=1}^N P(O_t O_{t+1} \dots O_T | X_t = i, X_{t+1} = j, \mu) P(X_{t+1} = j | X_t = i, \mu) \\
 &= \sum_{j=1}^N P(O_{t+1} \dots O_T | O_t, X_t = i, X_{t+1} = j, \mu) P(O_t | X_t = i, X_{t+1} = j, \mu) a_{ij} \\
 &= \sum_{j=1}^N P(O_{t+1} \dots O_T | X_{t+1} = j, \mu) b_{ij O_t} a_{ij} = \boxed{\sum_{j=1}^N \beta_j(t+1) b_{ij O_t} a_{ij}}
 \end{aligned}$$

Total: $P(O|\mu) = \sum_{i=1}^N P(O_1 O_2 \dots O_T | X_1 = i, \mu) P(X_1 = i | \mu) = \sum_{i=1}^N \beta_i(1) \pi_i$

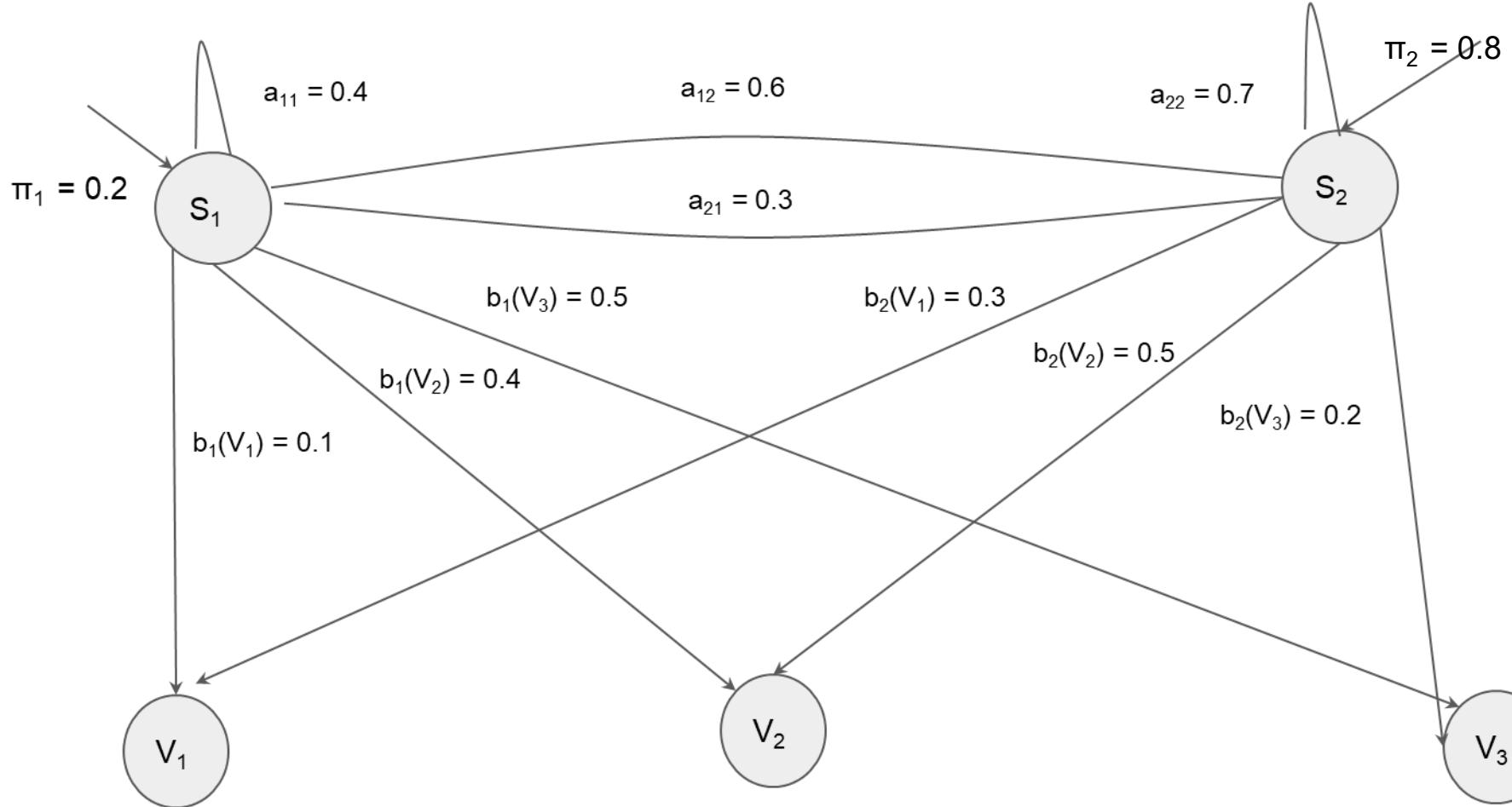
Backward Probability Algorithm

$$\beta_i(t) = P(O_t \dots O_T | X_t = i, \mu)$$

1. Initialization: $\beta_i(T+1) = 1$, for $1 \leq i \leq N$
2. Induction: $\beta_i(t) = \sum_{j=1}^N a_{ij} b_{ij} O_j \beta_j(t+1)$, $1 \leq t \leq T$, $1 \leq i \leq N$
3. Total: $P(O|\mu) = \sum_{i=1}^N \pi_i \beta_i(1)$

Complexity: $2N^2T$

HMM – Canonical Example Problem with backward probability



Beta Table

Observation sequence = $O = \{V_1, V_3, V_2\}$

	$t = 1 (V_1 \text{ is observed})$	$t = 2 (V_3 \text{ is observed})$	$t = 3 (V_2 \text{ is observed})$
S_1			
S_2			

To fill up beta table, initialize last column as 1's

Beta Table

Observation sequence = $O = \{V_1, V_3, V_2\}$

	$t=1 (V_1 \text{ is observed})$	$t=2 (V_3 \text{ is observed})$	$t=3 (V_2 \text{ is observed})$
S_1			1
S_2			1

To calculate Beta value at some cell, take beta values at next column, multiply each with corresponding transition probabilities to get $a_{ij} * \beta_{t+1}(j)$. Multiply each of these values with corresponding $b_j(O_{t+1})$ to get $a_{ij} * \beta_{t+1}(j) * b_j(O_{t+1})$. Finally add them up to get $\sum a_{ij} * \beta_{t+1}(j) * b_j(O_{t+1}) =$ Beta value at this cell.

Beta Table

Observation sequence = $O = \{V_1, V_3, V_2\}$

	$t=1 (V_1 \text{ is observed})$	$t=2 (V_3 \text{ is observed})$	$t=3 (V_2 \text{ is observed})$
S_1			$a_{11} = 0.4$ $b_1(V_2) = 0.4$
S_2			$a_{12} = 0.6$ $b_2(V_2) = 0.5$

$$b_1(V_2) = 0.4, \quad b_2(V_2) = 0.5$$

$$\text{corresponding beta at next cell } *a_{11}* b_1(V_2) = 1 * 0.4 * 0.4 = 0.16$$

$$\text{corresponding beta at next cell } *a_{12}* b_2(V_2) = 1 * 0.6 * 0.5 = 0.30$$

$$\text{Beta value at this cell} = 0.16 + 0.30 = 0.46$$

Beta Table

Observation sequence = $O = \{V_1, V_3, V_2\}$

	$t=1 (V_1 \text{ is observed})$	$t=2 (V_3 \text{ is observed})$	$t=3 (V_2 \text{ is observed})$
S_1		0.46	$a_{21} = 0.3$ 1 $b_1(V_2) = 0.4$
S_2			$a_{22} = 0.7$ 1 $b_2(V_2) = 0.5$

$$b_1(V_2) = 0.4, b_2(V_2) = 0.5$$

$$\text{corresponding beta at next cell } *a_{21}* b_1(V_2) = 1 * 0.3 * 0.4 = 0.12$$

$$\text{corresponding beta at next cell } *a_{22}* b_2(V_2) = 1 * 0.7 * 0.5 = 0.35$$

$$\text{Beta value at this cell} = 0.12 + 0.35 = 0.47$$

Beta Table

Observation sequence = $O = \{V_1, V_3, V_2\}$

	$t=1 (V_1 \text{ is observed})$	$t=2 (V_3 \text{ is observed})$	$t=3 (V_2 \text{ is observed})$
S_1	$a_{11} = 0.4$	0.46	1
S_2	$a_{12} = 0.6$	0.47	1

$b_1(V_3) = 0.5$

$b_2(V_3) = 0.2$

The diagram shows arrows originating from the beta values in the table and pointing towards the final result. One arrow points from $a_{11} = 0.4$ to the value 0.46. Another arrow points from $a_{12} = 0.6$ to the value 0.47. A third arrow points from the value 1 to the final result of 0.1484. A fourth arrow points from the value 0.46 to the final result. A fifth arrow points from the value 0.47 to the final result.

$$b_1(V_3) = 0.5, b_2(V_3) = 0.2$$

$$\text{corresponding beta at next cell } *a_{11}* b_1(V_2) = 0.46 * 0.4 * 0.5 = 0.0920$$

$$\text{corresponding beta at next cell } *a_{12}* b_2(V_2) = 0.47 * 0.6 * 0.2 = 0.0564$$

$$\text{Beta value at this cell} = 0.0920 + 0.0564 = 0.1484$$

Beta Table

Observation sequence = $O = \{V_1, V_3, V_2\}$

	$t=1 (V_1 \text{ is observed})$	$t=2 (V_3 \text{ is observed})$	$t=3 (V_2 \text{ is observed})$
S_1	0.1484 $a_{21} = 0.3$	0.46	1
S_2		$a_{22} = 0.7$	0.47

$$b_1(V_3) = 0.5, b_2(V_3) = 0.2$$

$$\text{corresponding beta at next cell } *a_{21}* b_1(V_2) = 0.46 * 0.3 * 0.5 = 0.0690$$

$$\text{corresponding beta at next cell } *a_{22}* b_2(V_2) = 0.47 * 0.7 * 0.2 = 0.0658$$

$$\text{Beta value at this cell} = 0.0690 + 0.0658 = 0.1348$$

Beta Table

Observation sequence = $O = \{V_1, V_3, V_2\}$

	$t = 1 (V_1 \text{ is observed})$	$t = 2 (V_3 \text{ is observed})$	$t = 3 (V_2 \text{ is observed})$
S_1	0.1484	0.46	1
S_2	0.1348	0.47	1

Forward and Backward Procedure

- To learn the HMM model, we need to know what states we are to explain the observations the best.
- That will be the occupation probability γ — the probability of state i at time t given all the observations.**
- Given the HMM model parameters fixed, we can apply the forward and backward algorithm to calculate α and β from the observations. γ can be calculated by simply multiplying α with β , and then renormalize it.

Computing $\gamma_i(t)$:

$$\gamma_i(t) = P(X_t = i | O, \mu) = \frac{P(X_t = i, O | \mu)}{P(O | \mu)} = \frac{\alpha_i(t) \beta_i(t)}{\sum_{j=1}^N \alpha_j(t) \beta_j(t)}$$

$$p(A, B) = p(A|B) \cdot p(B)$$

$$p(A|B) = \frac{p(A, B)}{p(B)}$$

Decoding- 2 methods -1st Method

$$P(O|\mu) = \sum_{i=1}^N P(O, X_t = i|\mu) = \sum_{i=1}^N \alpha_i(t)\beta_i(t)$$

That will be the occupation probability γ – the probability of state i at time t given all the observations.

$$\gamma_i(t) = P(X_t = i|O, \mu) = \frac{P(X_t = i, O|\mu)}{P(O|\mu)}$$

$$\begin{aligned} P(O, X_t = i|\mu) &= P(O_1 \dots O_T, X_t = i|\mu) \\ &= P(O_1 \dots O_{t-1}, X_t = i, O_t \dots O_T|\mu) \\ &= P(O_1 \dots O_{t-1}, X_t = i|\mu) P(O_t \dots O_T | O_1 \dots O_{t-1}, X_t = i, \mu) \\ &= \alpha_i(t) P(O_t \dots O_T | X_t = i, \mu) \\ &= \alpha_i(t) \beta_i(t) \end{aligned}$$

$$P(O|\mu) = \sum_{j=1}^N \alpha_j(t) \beta_j(t)$$

$$\gamma_i(t) = P(X_t = i|O, \mu) = \frac{P(X_t = i, O|\mu)}{P(O|\mu)} = \frac{\alpha_i(t) \beta_i(t)}{\sum_{j=1}^N \alpha_j(t) \beta_j(t)}$$

Finding the “best” state sequence Posterior decoding

One way to find the most likely state sequence underlying the observation sequence: choose the states individually

$$\gamma_i(t) = P(X_t = i | O, \mu)$$

$$\hat{X}_t = \underset{1 \leq i \leq N}{\operatorname{argmax}} \gamma_i(t) \text{ for } 1 \leq t \leq T + 1$$

Computing $\gamma_i(t)$:

$$\gamma_i(t) = P(X_t = i | O, \mu) = \frac{P(X_t = i, O | \mu)}{P(O | \mu)} = \frac{\alpha_i(t) \beta_i(t)}{\sum_{j=1}^N \alpha_j(t) \beta_j(t)}$$

Remark:

\hat{X} maximizes the expected number of states that will be guessed correctly. However, it may yield a quite unlikely/unnatural state sequence.

To Find Probability of an Observation Sequence using both Alpha and Beta Tables

Observation sequence = $O = \{V_1, V_3, V_2\}$

Alpha table:

	$t = 1 (V_1 \text{ is observed})$	$t = 2 (V_3 \text{ is observed})$	$t = 3 (V_2 \text{ is observed})$
S_1	0.02	0.04	0.01072
S_2	0.24	0.036	0.0246

Beta table

	$t = 1 (V_1 \text{ is observed})$	$t = 2 (V_3 \text{ is observed})$	$t = 3 (V_2 \text{ is observed})$
S_1	0.1484	0.46	1
S_2	0.1348	0.47	1

To Find Probability of an Observation Sequence using both Alpha and Beta Tables

Observation sequence = $\mathbf{O} = \{\mathbf{V}_1, \mathbf{V}_3, \mathbf{V}_2\}$

Take any column of alpha table and then take corresponding column of beta table and do dot product.

If we consider first column, to find $P(\mathbf{O} = \{\mathbf{V}_1, \mathbf{V}_3, \mathbf{V}_2\} | \lambda)$:

First column values of Alpha table = {0.02, 0.24}

First column values of Beta table = {0.1484, 0.1348}

$$\begin{aligned} P(\mathbf{O} = \{\mathbf{V}_1, \mathbf{V}_3, \mathbf{V}_2\} | \lambda) &= 0.02 * 0.1484 + 0.24 * 0.1348 \\ &= 0.03532 \end{aligned}$$

To Find Probability of an Observation Sequence using both Alpha and Beta Tables

Observation sequence = $O = \{V_1, V_3, V_2\}$

Take any column of alpha table and then take corresponding column of beta table and do dot product.

If we consider second column, to find $P(O = \{V_1, V_3, V_2\} | \lambda)$:

Second column values of Alpha table = {0.04, 0.036}

Second column values of Beta table = {0.46, 0.47}

$$P(O = \{V_1, V_3, V_2\} | \lambda) = 0.04 * 0.46 + 0.036 * 0.47 = 0.03532$$

To Find Probability of an Observation Sequence using both Alpha and Beta Tables

Observation sequence = $O = \{V_1, V_3, V_2\}$

Take any column of alpha table and then take corresponding column of beta table and do dot product.

If we consider last column, to find $P(O = \{V_1, V_3, V_2\} | \lambda)$:

Last column values of Alpha table = {0.01072, 0.0246}

Last column values of Beta table = {1,1}

$$P(O = \{V_1, V_3, V_2\} | \lambda) = 0.01072 * 1 + 0.0246 * 1 = 0.03532$$

Gamma Table

	$t = 1 (V_1 \text{ is observed})$	$t = 2 (V_3 \text{ is observed})$	$t = 3 (V_2 \text{ is observed})$
Alpha table	s_1	0.02	0.04
	s_2	0.24	0.036
	$t = 1 (V_1 \text{ is observed})$	$t = 2 (V_3 \text{ is observed})$	$t = 3 (V_2 \text{ is observed})$
Beta Table	s_1	0.1484	0.46
	s_2	0.1348	0.47

To Find Gamma value at some cell multiply corresponding alpha and beta values at that cell position and divide by probability of observation sequence

	$t = 1 (V_1 \text{ is observed})$	$t = 2 (V_3 \text{ is observed})$	$t = 3 (V_2 \text{ is observed})$
s_1			
s_2			

Gamma Table

	$t = 1 (V_1 \text{ is observed})$	$t = 2 (V_3 \text{ is observed})$	$t = 3 (V_2 \text{ is observed})$
S_1	0.02	0.04	0.01072
S_2	0.24	0.036	0.0246

gamma at this cell =
 $0.01072 * 1 / 0.03532 =$
 0.30351

	$t = 1 (V_1 \text{ is observed})$	$t = 2 (V_3 \text{ is observed})$	$t = 3 (V_2 \text{ is observed})$
S_1	0.1484	0.46	1
S_2	0.1348	0.47	1

	$t = 1 (V_1 \text{ is observed})$	$t = 2 (V_3 \text{ is observed})$	$t = 3 (V_2 \text{ is observed})$
S_1			
S_2			

Gamma Table

	$t = 1 (V_1 \text{ is observed})$	$t = 2 (V_3 \text{ is observed})$	$t = 3 (V_2 \text{ is observed})$
S_1	0.02	0.04	0.01072
S_2	0.24	0.036	0.0246
	$t = 1 (V_1 \text{ is observed})$	$t = 2 (V_3 \text{ is observed})$	$t = 3 (V_2 \text{ is observed})$
S_1	0.1484	0.46	1
S_2	0.1348	0.47	1

gamma at this cell =
 $0.0246 * 1 / 0.03532 =$
 0.69648

	$t = 1 (V_1 \text{ is observed})$	$t = 2 (V_3 \text{ is observed})$	$t = 3 (V_2 \text{ is observed})$
S_1			0.30351
S_2			

Gamma Table

The diagram illustrates the calculation of Gamma from Alpha and Beta tables. It shows two tables: Alpha table and Beta table, with their respective columns for observed values. Arrows point from the Beta table's values to the corresponding cells in the Alpha table, and a final arrow points to the calculated Gamma value.

	$t=1 (V_1 \text{ is observed})$	$t=2 (V_3 \text{ is observed})$	$t=3 (V_2 \text{ is observed})$	
Alpha table	S_1	0.02	0.04	0.01072
	S_2	0.24	0.036	0.0246

	$t=1 (V_1 \text{ is observed})$	$t=2 (V_3 \text{ is observed})$	$t=3 (V_2 \text{ is observed})$	
Beta Table	S_1	0.1484	0.46	1
	S_2	0.1348	0.47	1

	$t=1 (V_1 \text{ is observed})$	$t=2 (V_3 \text{ is observed})$	$t=3 (V_2 \text{ is observed})$
S_1			0.30351
S_2			0.69648

gamma at this cell =
 $0.04 * 0.46 / 0.03532 =$
 0.52095

Gamma Table

Alpha table

	$t=1 (V_1 \text{ is observed})$	$t=2 (V_3 \text{ is observed})$	$t=3 (V_2 \text{ is observed})$
S_1	0.02	0.04	0.01072
S_2	0.24	0.036	0.0246

Beta Table

	$t=1 (V_1 \text{ is observed})$	$t=2 (V_3 \text{ is observed})$	$t=3 (V_2 \text{ is observed})$
S_1	0.1484	0.46	1
S_2	0.1348	0.47	1

	$t=1 (V_1 \text{ is observed})$	$t=2 (V_3 \text{ is observed})$	$t=3 (V_2 \text{ is observed})$
S_1		0.52095	0.30351
S_2			0.69648

gamma at this cell =
 $0.036 * 0.47 / 0.03532 =$
 0.47904

Gamma Table

The diagram illustrates the calculation of Gamma values from Alpha and Beta tables. Arrows point from specific cells in the Alpha and Beta tables to a central formula for calculating Gamma.

Alpha table:

	t = 1 (V_1 is observed)	t = 2 (V_3 is observed)	t = 3 (V_2 is observed)
S_1	0.02	0.04	0.01072
S_2	0.24	0.036	0.0246

Beta Table:

	t = 1 (V_1 is observed)	t = 2 (V_3 is observed)	t = 3 (V_2 is observed)
S_1	0.1484	0.46	1
S_2	0.1348	0.47	1

Central formula for Gamma calculation:

$$\text{gamma at this cell} = \frac{0.02 * 0.1484}{0.03532} = 0.08403$$

Gamma Table

Alpha table

	$t=1 (V_1 \text{ is observed})$	$t=2 (V_3 \text{ is observed})$	$t=3 (V_2 \text{ is observed})$
S_1	0.02	0.04	0.01072
S_2	0.24	0.036	0.0246

Beta Table

	$t=1 (V_1 \text{ is observed})$	$t=2 (V_3 \text{ is observed})$	$t=3 (V_2 \text{ is observed})$
S_1	0.1484	0.46	1
S_2	0.1348	0.47	1

	$t=1 (V_1 \text{ is observed})$	$t=2 (V_3 \text{ is observed})$	$t=3 (V_2 \text{ is observed})$
S_1	0.08403	0.52095	0.30351
S_2		0.47904	0.69648

gamma at this cell =
 $0.24 * 0.1348 / 0.03532 =$
 0.91596

Gamma Table

	$t = 1 (V_1 \text{ is observed})$	$t = 2 (V_3 \text{ is observed})$	$t = 3 (V_2 \text{ is observed})$
Alpha table	S_1	0.02	0.04
	S_2	0.24	0.036

Sum of gamma values in one column should be 1.
(By definition)

	$t = 1 (V_1 \text{ is observed})$	$t = 2 (V_3 \text{ is observed})$	$t = 3 (V_2 \text{ is observed})$
Beta Table	S_1	0.1484	0.46
	S_2	0.1348	0.47

	$t = 1 (V_1 \text{ is observed})$	$t = 2 (V_3 \text{ is observed})$	$t = 3 (V_2 \text{ is observed})$
S_1	0.08403	0.52095	0.30351
S_2	0.91596	0.47904	0.69648

Decoding-Viterbi algorithm

- The decoding problem is finding the optimal internal states sequence given a sequence of observations.
- Again, we want to express our components recursively.
- Given the state is j at time t , $v_t(j)$ is the joint probability of the observation sequence with the best state sequence.
- If we examine closely, the resulting equation is close to the forward algorithm except the summation is replaced by the max function.

$$v_t(j) = \max_{s_0, s_1, \dots, s_{t-1}} P(s_0, s_1, \dots, s_{t-1}, x_1, x_2, \dots, x_t, S_t = s_j | \lambda)$$

$$v_t(j) = \max_{i=1}^N v_{t-1}(i) a_{ij} b_j(x_t)$$

The Viterbi algorithm

Compute the probability of the most likely path

$$\underset{X}{\operatorname{argmax}} P(X|O, \mu) = \underset{X}{\operatorname{argmax}} P(X, O|\mu)$$

through a node in the trellis

$$\delta_i(t) = \max_{X_1 \dots X_{t-1}} P(X_1 \dots X_{t-1}, O_1 \dots O_{t-1}, X_t = s_i | \mu)$$

1. **Initialization:** $\delta_j(1) = \pi_j$, for $1 \leq j \leq N$
2. **Induction:** (see the similarity with the Forward algorithm)

$$\delta_j(t+1) = \max_{1 \leq i \leq N} \delta_i(t) a_{ij} b_{ij} O_t, \quad 1 \leq t \leq T, \quad 1 \leq j \leq N$$

$$\psi_j(t+1) = \operatorname{argmax}_{1 \leq i \leq N} \delta_i(t) a_{ij} b_{ij} O_t, \quad 1 \leq t \leq T, \quad 1 \leq j \leq N$$

3. **Termination and readout of best path:**

$$P(\hat{X}, O | \mu) = \max_{1 \leq i \leq N} \delta_i(T+1)$$

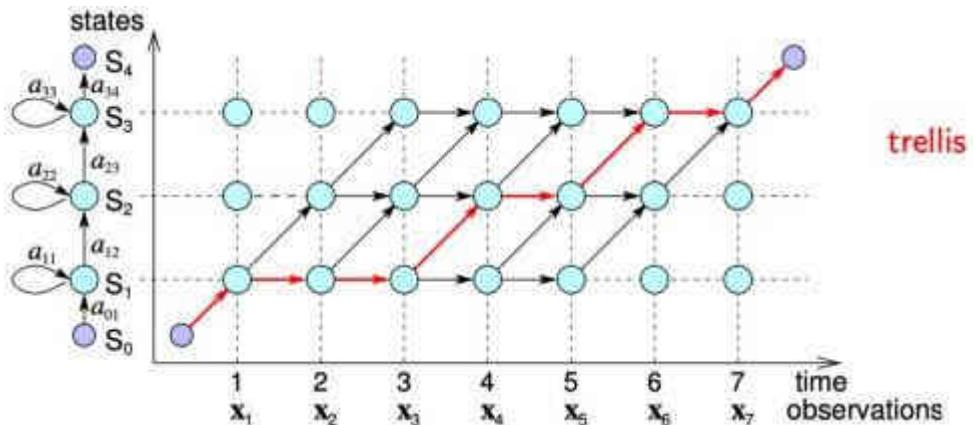
$$\hat{X}_{T+1} = \operatorname{argmax}_{1 \leq i \leq N} \delta_i(T+1), \quad \hat{X}_t = \psi_{\hat{X}_{t+1}}(t+1)$$

Decoding-Viterbi algorithm

$$v_t(j) = \max_{s_0, s_1, \dots, s_{t-1}} P(s_0, s_1, \dots, s_{t-1}, x_1, x_2, \dots, x_t, S_t = s_j | \lambda)$$

$$v_t(j) = \max_{i=1}^N v_{t-1}(i) a_{ij} b_j(x_t)$$

- So not only it can be done, the solution is similar to the forward algorithm except the summation is replaced by the maximum function.
- Here, instead of summing over all possible state sequences in the forward algorithm, the Viterbi algorithm finds the most likely path.

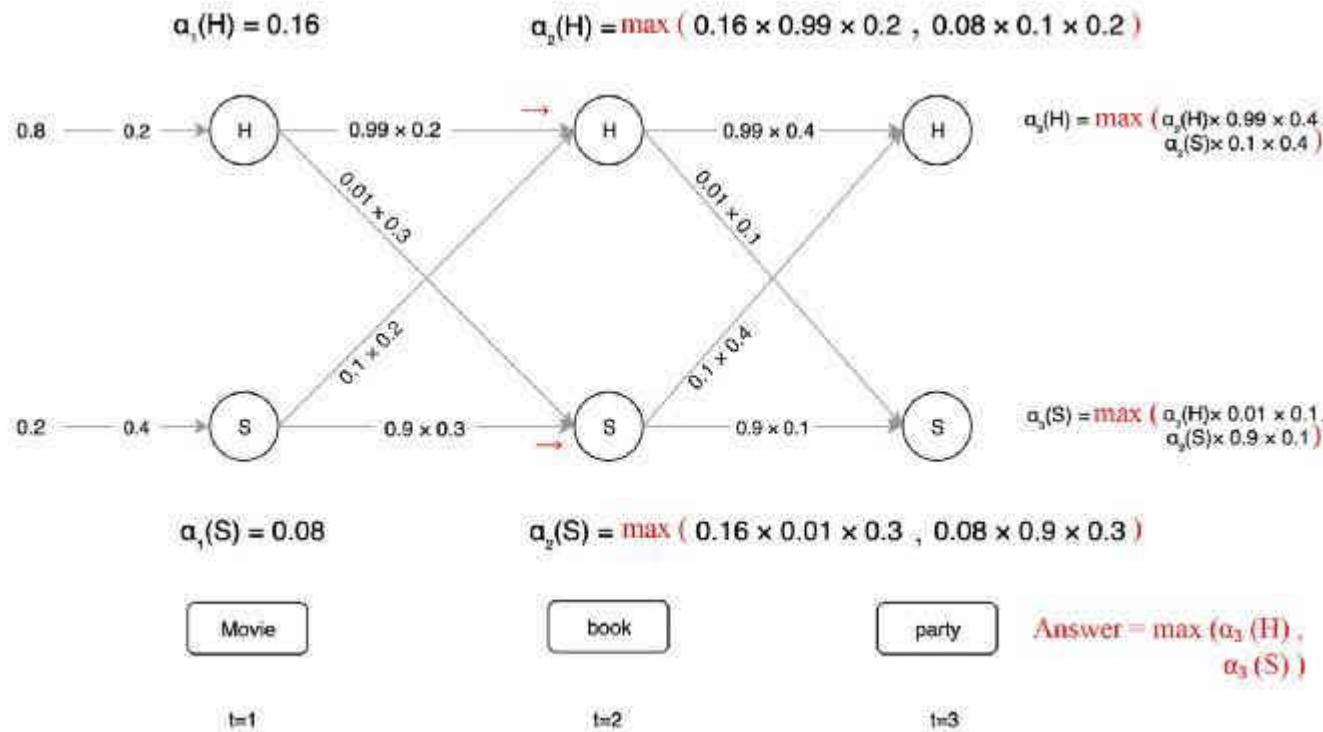


$$p(\mathbf{X}, \text{path}_\ell | \lambda) = p(\mathbf{X} | \text{path}_\ell, \lambda) P(\text{path}_\ell | \lambda)$$

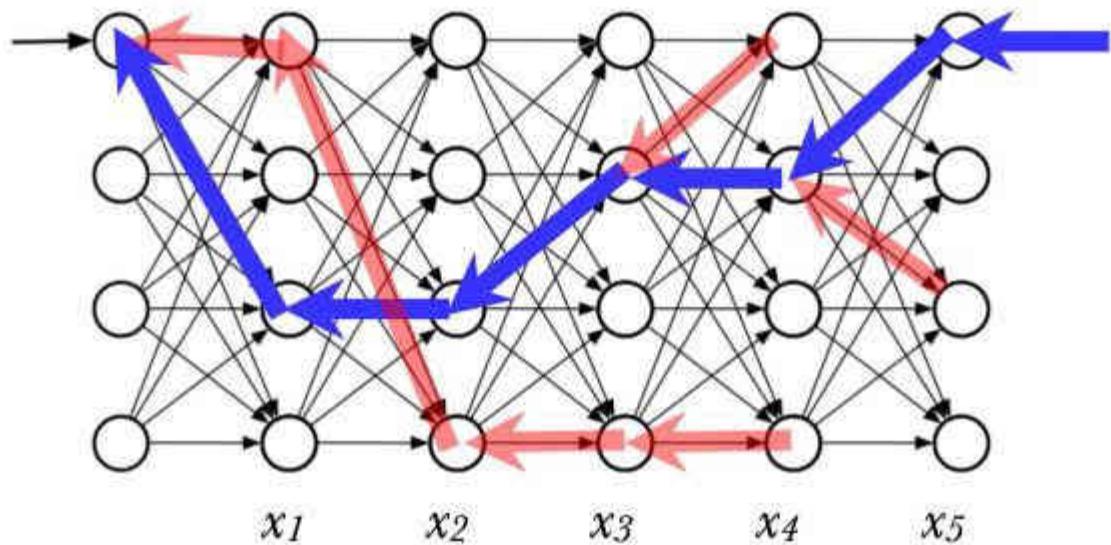
likelihood: $\sum_{\{\text{path}_\ell\}} p(\mathbf{X}, \text{path}_\ell | \lambda)$

decode: $\max_{\text{path}_\ell} p(\mathbf{X}, \text{path}_\ell | \lambda)$

- Finding the internal states that maximize the likelihood of observations is similar to the likelihood method.
- We just replace the summation with the maximum function.



In this algorithm, we also record the maximum path leading to each node at time t (the red arrow above). e.g. we are transited from a happy state H at t=1 to the happy state H at t=2 above since it is the most optimal (likely) path.





THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701



MACHINE INTELLIGENCE

Hidden Markov Model

K.S.Srinivas

Department of Computer Science and Engineering

MACHINE INTELLIGENCE

Finding the parameters – Hidden Markov Model

K.S.Srinivas

Department of Computer Science and Engineering

Baum-Welch Algorithm

- Besides likelihood and decoding, the last algorithm learns the HMM model parameters λ given the observation.
- Here, we will use the Baum–Welch algorithm to learn the transition and the emission probability.
- if we know the state occupation probability (the state distribution at time t), we can derive the emission probability and the transition probability.
- If we know these two probabilities, we can derive the state distribution at time t

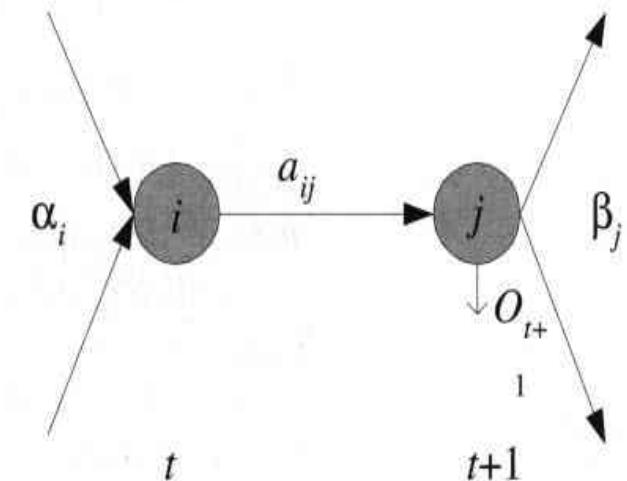
Baum-Welsh Algorithm

ξ is the probability of transiting from state i to j after time t given all the observations. It can be computed by α and β similarly

$$\xi_{ij}(t) = P(X_t = i, X_{t+1} = j | Y, \theta)$$

$$= \frac{P(X_t = i, X_{t+1} = j, Y | \theta)}{P(Y | \theta)}$$

$$= \frac{\alpha_i(t) a_{ij} \beta_j(t+1) b_j(y_{t+1})}{\sum_{i=1}^N \sum_{j=1}^N \alpha_i(t) a_{ij} \beta_j(t+1) b_j(y_{t+1})}$$



Baum-Welsh Algorithm

Intuitively, with a fixed HMM model, we refine the state occupation probability (γ) and the transition (ξ) with the given observations.

$$\gamma_i(t) = P(X_t = i | Y, \theta)$$

$$\xi_{ij}(t) = P(X_t = i, X_{t+1} = j | Y, \theta)$$

Here comes the chicken and egg part. Once the distribution of γ and $\xi(\theta_2)$ are refined, we can perform a point estimate on what will be the best transition and emission probability (θ_1 : a , b).

$$a_{ij}^* = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$$

$$b_i^*(v_k) = \frac{\sum_{t=1}^T \mathbf{1}_{y_t=v_k} \gamma_i(t)}{\sum_{t=1}^T \gamma_i(t)}$$

Baum-Welsh Algorithm

$$a_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i}$$

$$\gamma_i(t) = P(X_t = i | Y, \theta)$$

$$\xi_{ij}(t) = P(X_t = i, X_{t+1} = j | Y, \theta)$$

$$b_j(v_k) = \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j}$$

$$a_{ij}^* = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$$

$$b_i^*(v_k) = \frac{\sum_{t=1}^T \mathbf{1}_{y_t=v_k} \gamma_i(t)}{\sum_{t=1}^T \gamma_i(t)}$$

$$\Pi_i = \gamma_i(1)$$

Probability of the system being in state i at time t

Baum Welsh Algorith,

Estimation step

Forward

1. $\alpha_i(1) = \pi_i b_i(y_1)$,
2. $\alpha_i(t+1) = b_i(y_{t+1}) \sum_{j=1}^N \alpha_j(t) a_{ji}$.

Backward

1. $\beta_i(T) = 1$,
2. $\beta_i(t) = \sum_{j=1}^N \beta_j(t+1) a_{ij} b_j(y_{t+1})$.

Update

$$\begin{aligned}\gamma_i(t) &= P(X_t = i | Y, \theta) = \frac{P(X_t = i, Y | \theta)}{P(Y | \theta)} \\ &= \frac{\alpha_i(t) \beta_i(t)}{\sum_{j=1}^N \alpha_j(t) \beta_j(t)}\end{aligned}$$

$$\xi_{ij}(t) = P(X_t = i, X_{t+1} = j | Y, \theta)$$

$$\begin{aligned}&= \frac{P(X_t = i, X_{t+1} = j, Y | \theta)}{P(Y | \theta)} \\ &= \frac{\alpha_i(t) a_{ij} \beta_j(t+1) b_j(y_{t+1})}{\sum_{i=1}^N \sum_{j=1}^N \alpha_i(t) a_{ij} \beta_j(t+1) b_j(y_{t+1})}\end{aligned}$$

equals α for state i at time t \times transition prob. between i and j
 $\times \beta$ for state j at time $t+1$ \times observe y_{t+1} for state j

Maximization step

$$\pi_i^* = \gamma_i(1)$$

$$a_{ij}^* = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$$

$$P(X_t = i, X_{t+1} = j | Y, \theta)$$

$$P(X_t = i | Y, \theta)$$

$$b_i^*(v_k) = \frac{\sum_{t=1}^T 1_{y_t=v_k} \gamma_i(t)}{\sum_{t=1}^T \gamma_i(t)}$$

(sum γ over all time steps where the observation y_t is the same as v_k at time t)

where

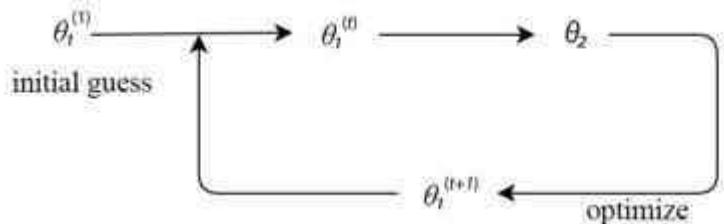
$$1_{y_t=v_k} = \begin{cases} 1 & \text{if } y_t = v_k, \\ 0 & \text{otherwise} \end{cases}$$

equals α (forward) \times β (backward) for state i at time t

MACHINE INTELLIGENCE

HMM

We fix one set of parameters to improve others and continue the iteration until the solution converges.



The Estimation Maximization algorithm is usually defined as:

$$\theta_i^* = \arg \max_{\theta_i} \sum_{x \in D} \mathbb{E}_{\theta_2} \log p(x, \theta_2; \theta_i)$$

a, b : transition and emission probabilities

γ, ξ : internal state distribution and transition

Here, the E-step establishes $p(\gamma, \xi | x, a, b)$. Then, the M-step finds a, b that roughly maximizes the objective below.

$$\theta_1^* = \arg \max_{\theta_1} \sum_{x \in D} \mathbb{E}_{\theta_2} p(x, \theta_2; \theta_1)$$

observations γ, ξ a, b

\diagup \diagdown \diagup \diagdown

M-step: find the optimal θ_1 given θ_2 and X E-step: establish the probability distribution of θ_2 given θ_1 and X



THANK YOU

K.S.Srinivas
srinivasks@pes.edu

+91 80 2672 1983 Extn 701