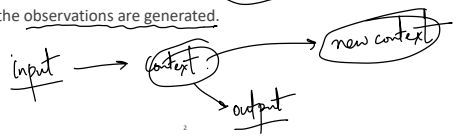
06 | Jan | 2023.

Hidden Markov Models

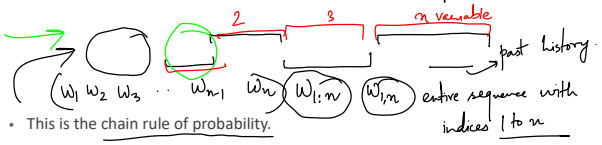
Markov Model

- Let's say there is a process which generates a sequence:
  - Sequence of written words
  - Sequence of spoken words (speech)
  - DNA, RNA protein's amino acid sequence, etc.

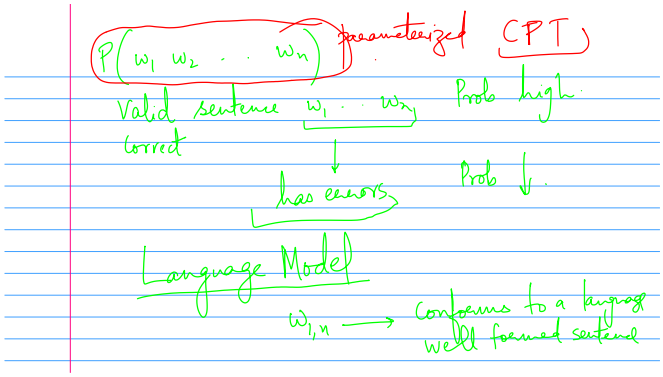
- The observed sequence is governed by some context which itself keeps updating as the observations are generated.



- The probability of a sequence can be written as

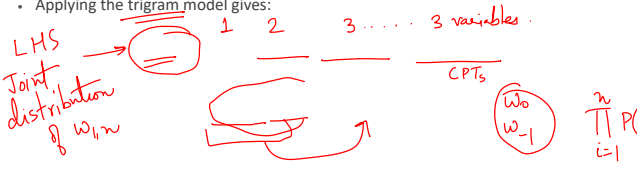


- This is the chain rule of probability.
- Modeling the sequence of words requires learning each of the component terms.
- We can do something smarter by modelling the context in a better way.

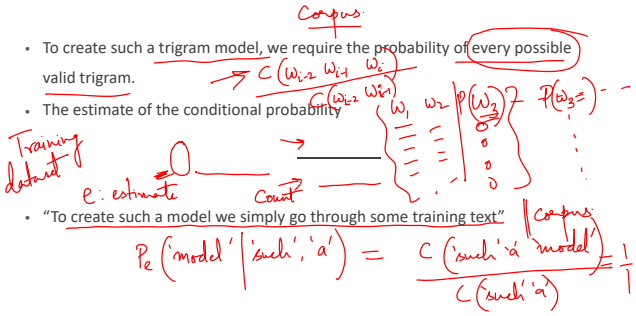


- The context sequence can be
  - Observable
  - Hidden
- An example of observable context model is the n-gram model of text.
- n-gram model: Only the previous n-1 words have any effect on the probabilities of the next word.

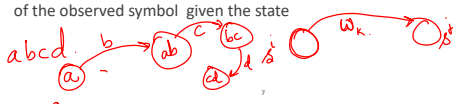
- For a trigram model:
- Applying the trigram model gives:



- To create such a trigram model, we require the probability of every possible valid trigram.
- The estimate of the conditional probability
- "To create such a model we simply go through some training text"

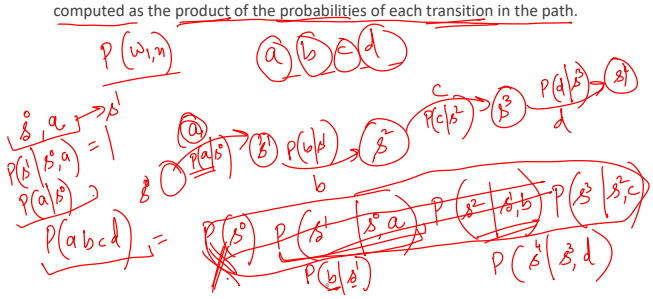


- After all the hard work, we now have a model which can generate the probability of any string.
- This model is called a Markov chain.
- It can be represented as a graph of nodes and arcs
- Nodes are the states
- Arcs represent transitions from one state to another.
  - An arc is labelled with the observation (emitted symbol) and the probability of the observed symbol given the state



- Why is the state obvious here?

- Given a Markov chain, the probability of an observed sequence can be computed as the product of the probabilities of each transition in the path.

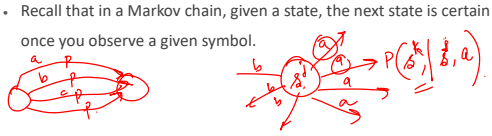


What if the states are hidden?

- Now consider the scenario when the states are non-obvious.
- That means, we are uncertain about the state sequence that generated the given observation sequence.
- That means,
  - there can be multiple state sequences that can lead to a given observation sequence.
  - a given state sequence can generate multiple observation sequences
  - for a given state, we can make transition to multiple possible states, even while generating the same output.

Hidden Markov Models

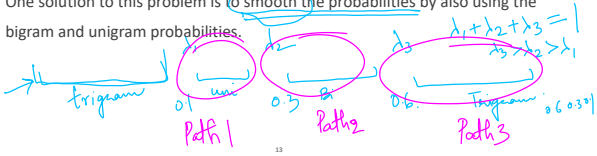
- So now, transition to multiple states is possible from a given state, even while emitting the same observation (symbol).
- HMMs are a generalisation of Markov Chains in which a given state may have several transitions out of it, all with the same label (symbol).
  - Recall that in a Markov chain, given a state, the next state is certain (fixed) once you observe a given symbol.



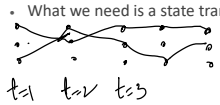
- Since more than one successor state may have the same output, in general there may be several paths through an HMM that produce the same output.
- In such cases, the probability of the output is the sum of the probabilities of all possible paths.
- The Markov chain cannot be used now.
  - What we need is a state transition diagram to represent an HMM.

Moving from a Markov Chain to an HMM

- Consider that we are learning a trigram model.
- If the training corpus is sparse, it may happen that there is a trigram in the next text that never appeared in the training corpus.
  - In that case, a 0 probability will be assigned to the trigram, i.e. probability of the third word, given the first two words.
- One solution to this problem is to smooth the probabilities by also using the bigram and unigram probabilities.

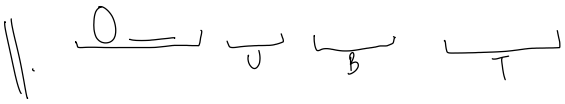


- Since more than one successor state may have the same output, in general there may be several paths through an HMM that produce the same output.
- In such cases, the probability of the output is the sum of the probabilities of all possible paths.
- The Markov chain cannot be used now.
  - What we need is a state transition diagram to represent an HMM.

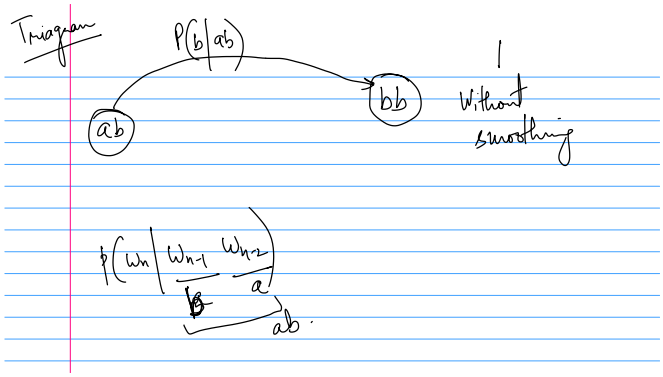


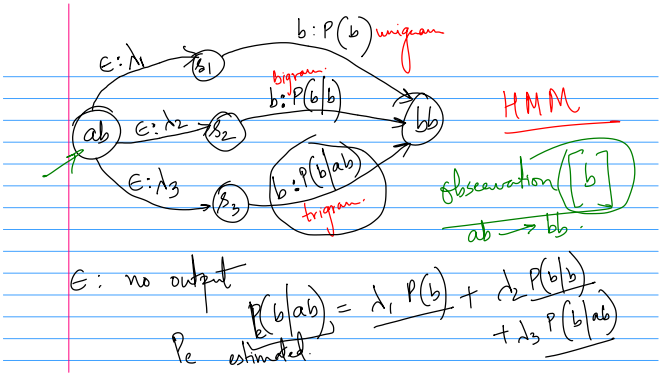
Moving from a Markov Chain to an HMM

- Consider that we are learning a trigram model.
- If the training corpus is sparse, it may happen that there is a trigram in the next text that never appeared in the training corpus.
  - In that case, a 0 probability will be assigned to the trigram, i.e. probability of the third word, given the first two words.
- One solution to this problem is to smooth the probabilities by also using the bigram and unigram probabilities.



- If we have missing trigram then the bigram and unigram probabilities take over.
- If we have missing trigram and bigram, then we fall back on the unigram probability.
- The probability of a transition has become an addition of multiple components (arcs) between the two states.
  - An HMM will allow multiple paths between two states.



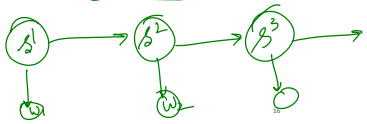


HMM state transition diagram for a trigram model

- Consider that we have seen the symbol ab.
- The next symbol in the sequence is a.
- So the next state is ba
- The trigram $P(a \mid ab)$ itself can be represented as an HMM

Smoothed trigram: $P(a \mid ab) = \lambda_1 P_e(a) + \lambda_2 P_e(a \mid b) + \lambda_3 P_e(a \mid ab)$ — from the corpus

- An HMM is a 4 tuple $\langle s^1, S, W, E \rangle$ — s^1 : starting state
- S: the set of states — $S = \{s^1, s^2, s^3 \cdots\}$
- $s^1 \in S$ is the initial state of the model — $S_1 = s^1$
- W : the set of output symbols generated/emitted/accepted
- E : the set of edges or transitions. between states — $S_{10} = s^2$



- For each of the sets S, W, E we assume a canonical ordering of the elements

$S = \langle s^1, s^2, \ldots, s^{\sigma} \rangle$

$W = \langle w^1, w^2, \ldots, w^{\omega} \rangle$

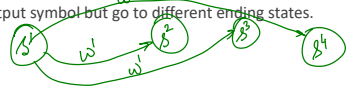
$E = \langle e^1, e^2, \ldots, e^{\epsilon} \rangle$

$p \equiv P(s^j \mid s^i, w^k)$

- The starting state of the HMM is the first element in the ordering of states.
- A transition is a 4-tuple $\langle s^i, s^j, w^k, p \rangle$ where s^i is the state from which the transition starts, s^j is the state where the transition ends, w^k is the output symbol generated by the model, p is the probability of taking the transition

$S_t \; w_t \longrightarrow S_{t+1} \; w_{t+1} \longrightarrow S_{t+2}$

- p is the probability of the transition — $s^i \xrightarrow{w^k} s^j$
- No two transitions can have the same starting and ending states as well as the same output value.
- We can leave out the zero probability paths from the graphical representation of the HMM.
- Note that a state s can be the starting state for several transitions that have the same output symbol but go to different ending states.



- It is not possible to know what state the machine has gone into simply by looking at the output. $(w_1 \cdots w_n)$ output
- Thus, the state sequence followed by an HMM is not deductible from the input. It is hidden.

$S_t \; w_t \longrightarrow S_{t+1} \; w_{t+1}$ emitted.

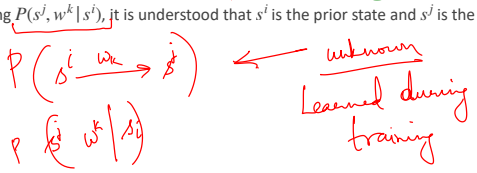
- There are n+1 states for n outputs

$S_t \; i_t \; w_t \longrightarrow S_{t+1} \; i_{t+1} \; w_{t+1}$

- The probability p associated with a transition $s^i \xrightarrow{w_k} s^j$

$$P(s^i \xrightarrow{w_k} s^j) = P(S_{t+1} = s^j, W_t = w^k \mid S_t = s^i)$$
$$= P(s^j, w^k \mid s^i)$$

- When writing $P(s^j, w^k \mid s^i)$, it is understood that s^i is the prior state and s^j is the next state.

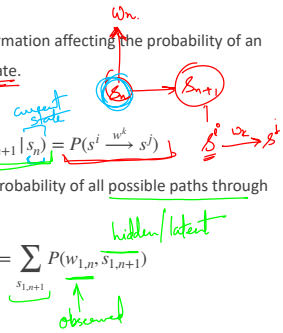


- Markov models assume that the only information affecting the probability of an output, or of the next state, is the prior state.
- As per this Markov Assumption

$$P(w_n, s_{n+1} \mid w_{1,n-1}, s_{1,n}) = P(w_n, s_{n+1} \mid s_n) = P(s^i \xrightarrow{w^k} s^j)$$

- The probability of a sequence $w_{1,n}$ is the probability of all possible paths through the HMM that can produce this sequence.

  In other words: $P(w_{1,n}) = \sum_{s_{1,n+1}} P(w_{1,n}, s_{1,n+1})$



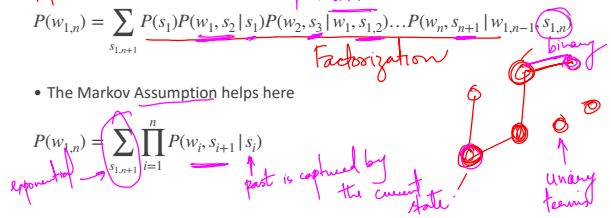
The Markov Assumption helps

$$P(w_{1,n}) = \sum_{s_{1,n+1}} P(w_{1,n}, s_{1,n+1})$$

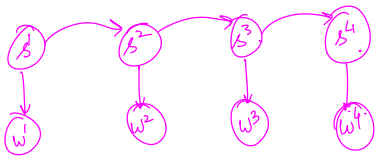
$$P(w_{1,n}) = \sum_{s_{1,n+1}} P(s_1)P(w_1, s_2 \mid s_1)P(w_2, s_3 \mid w_1, s_{1,2}) \ldots P(w_n, s_{n+1} \mid w_{1,n-1}, s_{1,n})$$

- The Markov Assumption helps here

$$P(w_{1,n}) = \sum_{s_{1,n+1}} \prod_{i=1}^{n} P(w_i, s_{i+1} \mid s_i)$$



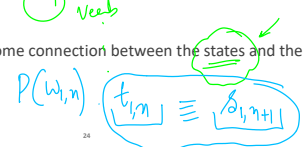
Exercise: Graphically visualize the simplification offered by Markov Assumption



An application of HMM: Part of Speech Tagging

- English words can be in more than one PoS class.
- PoS tags can be assigned using HMM.
- But to use HMM we need to phrase the problem of assigning PoS tags to the words as one of assigning probabilities to the input text.
- We assume that the HMM generates outputs which are the words of the corpus.

- We assume that there is some connection between the states and the transitions of the tags.



- Earlier we had for Language Model (LM)

$$P(w_{1,n}) = \sum_{s_{1,n+1}} P(w_{1,n}, s_{1,n+1})$$

- For PoS tagging, we assume correspondence between states and the tags
- So,

$$P(w_{1,n}) = \sum_{t_{1,n+1}} P(w_{1,n}, t_{1,n+1})$$

- Here, $t_{1,n+1}$ is a sequence of $n+1$ parts of speech or tags.

Boltzmann Machines

Given an HMM (learned parameters)

1) Generative Model.

$P(w_1, w_2, \ldots w_n)$ — GMM

Markov Assumption

$\stackrel{def}{=} P(w_{1,n})$

How to learn the parameters of an HMM

Estimate

2) ~~Generate~~ the most likely seq of states which has generated the observed sequence.

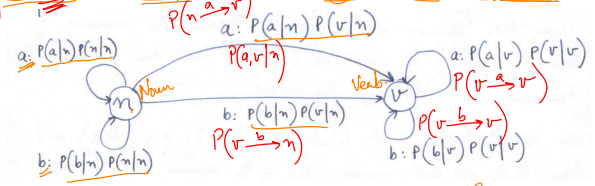
- The last tag t_{n+1} is the tag achieved after emitting w_n.
- So t_{n+1} is a tag which corresponds to the non-existent word w_{n+1}
- We define the problem of PoS tagging as finding

$$\arg\max_{t_{1,n}} P(t_{1,n} | w_{1,n}) = \arg\max_{t_{1,n}} \frac{P(w_{1,n}, t_{1,n})}{P(w_{1,n})} = \arg\max_{t_{1,n}} P(w_{1,n}, t_{1,n})$$

Most likely tag sequence.

State sequence

- There can be multiple ways of defining states in terms of PoS tags.
- Option 1: The HMM state corresponds to the PoS of the word produced next.



Transitions from v to n are not shown.

Vocabulary {a, b}

- The joint probability of all the words is

$$P(w_n) = \sum_{t_{1,n+1}} P(w_{1,n}, t_{1,n+1})$$

P(obs seq)

- We use Markov assumptions to simplify the joint distribution over the words
  - The probability of a word appearing at a particular position given that its PoS occurs at that position is independent of everything else

$$P(w_n | w_{1,n-1}, t_{1,n}) = P(w_n | t_n)$$

past history — Assumption

  - The probability of a PoS coming next is dependent only on the previous PoS

$$P(t_n | w_{1,n-1} t_{1,n-1}) = P(t_n | t_{n-1})$$

past history — prev state

$$P(w_{1,n}) = \sum_{t_{1,n+1}} P(w_{1,n}, t_{1,n+1})$$

Big term — Markov Assumptions

$$= \sum_{t_{1,n+1}} \prod_{i=1}^{n} P(w_i | t_i) P(t_{i+1} | t_i)$$

3

- Several state paths can lead to the observed word sequence.
- We can also find the most likely sequence of states that can generate a given word sequence as.

$$\arg\max_{t_{1,n+1}} \prod_{i=1}^{n} P(w_i | t_i) P(t_{i+1} | t_i)$$

tags (as per design option 1)

Design

Option 2

$t_i t_{i-1} \equiv$ a state

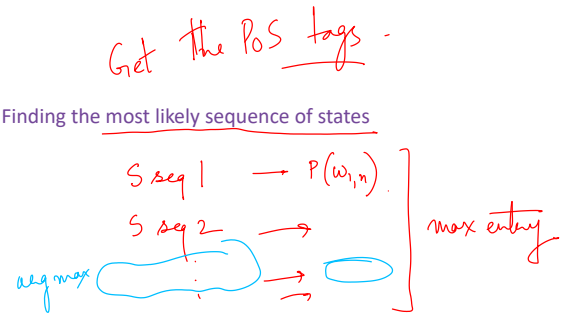
A state can be formulated as representing two parts of speech, that of previous and the current word

$$P(w_{1,n}) = \sum_{t_{1,n+1}} \prod_{i=1}^{n} P(w_i | t_i) P(t_{i+1} | t_i, t_{i-1})$$

State

$S_i \equiv (t_i, t_{i-1})$.

Finding the most likely sequence of states

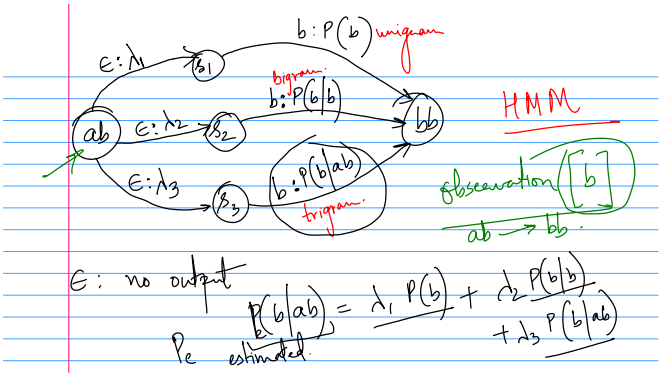


Finding the most likely sequence of states: The Viterbi Algorithm

- We need a way to find the most likely path through an HMM given a particular input $w_{1,n}$
- The most likely path corresponds to the most likely tags for the sentence.
- For a string of length n, the Viterbi algorithm finds $\alpha(n+1)$, which is defined as

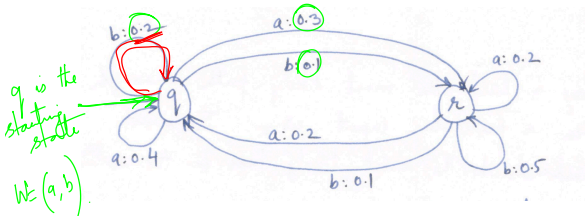
$$\alpha(t) = \arg\max_{s_{1,t}} P(s_{1,t} \mid w_{1,t-1})$$

$$= \arg\max_{s_{1,t}} P(s_{1,t}, w_{1,t-1})$$



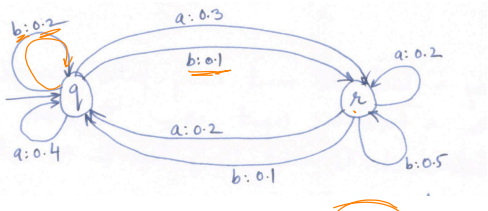
The Viterbi Algorithm

- The Viterbi algorithm computes the most likely state sequence starting with the empty output sequence, working up to the answer one output at a time.
- At each stage one need only compute the most likely sequence (and its probability) that ends in state s^i for all states.



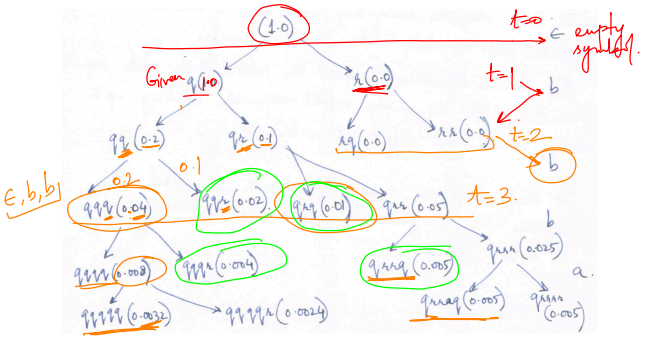
- The HMM has only 2 states. HMM emits the sequence 'bbba'

Symbols Emitted	-	b	Bb	Bbb	bbba
Sequence	q	qq	qqq	qqqq	qrrrq
Probability	1.0	0.2	0.04	0.008	0.005



- The HMM has only 2 states. HMM emits the sequence 'bbba'

Symbols Emitted	-	b	Bb	Bbb	bbba
Sequence	r	qr	qrr	qrrr	qrrrr
Probability	1.0	0.1	0.05	0.025	0.005



The Viterbi Algorithm

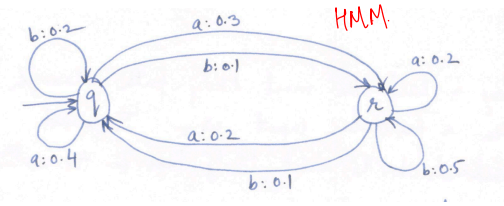
- The Viterbi algorithm computes the most likely state sequence starting with the empty output sequence, working up to the answer one output at a time.
- At each stage one need only compute the most likely sequence (and its probability) that ends in state s^i for all states.

$$\text{Most likely path} = \underset{S_{1,n+1}}{\arg\max}\ P(w_{1,n}, S_{1,n+1})$$

(Best overall path) which has generated the observation

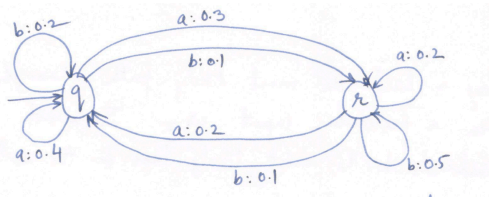
① Best path to each state at each step

② Overall best path



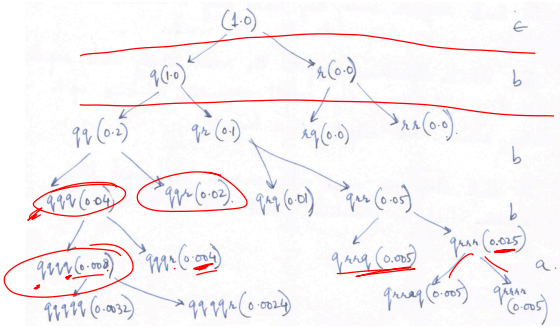
- The HMM has only 2 states. HMM emits the sequence 'bbba'

Symbols Emitted	-	b	Bb	Bbb	bbba
Sequence1	q	qq	qqq	qqqq	qrrrq
Probability	1.0	0.2	0.04	0.008	0.005

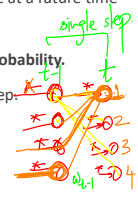


- The HMM has only 2 states. HMM emits the sequence 'bbba'

Symbols Emitted	-	b	Bb	Bbb	bbba
Sequence2	r	qr	qrr	qrrr	qrrrr
Probability	1.0	0.1	0.05	0.025	0.005



- For each time tick it is necessary to store the best path and its probability for each possible state.
- Storing only the overall best path is not sufficient.
- This allows us to discover paths that become more probable at a future time step.
- **So we need to store the best path to each state and the probability.**
- We never need to store more than $\sigma = |S|$ paths at any step.

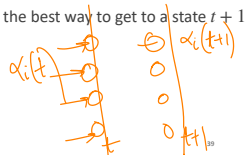


Step by Step constructing the **best forward path** from time t

- Let $\alpha_i(t)$ be the state sequence that has maximum probability of generating $w_{1,t-1}$ and ends up in state s^i at time t

$$\alpha_i(t) = \arg\max_{s,t} P(w_{1,t-1}, s_{1,t-1}, S_t = s^i)$$

- Given the best way to get to all states at time t, $\alpha_k(t)$ for $k = 1$ to σ, we can compute the best way to get to a state $t+1$ as a simple extension.

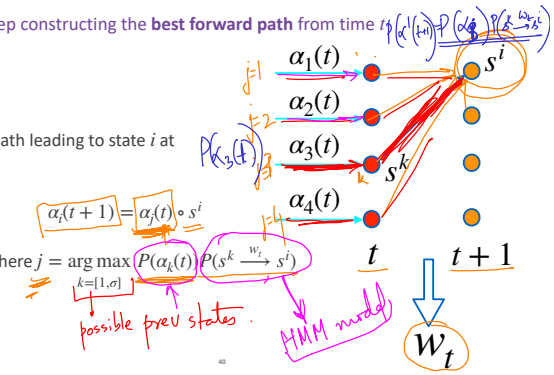


Step by Step constructing the **best forward path** from time t

- The best path leading to state i at time $t+1$

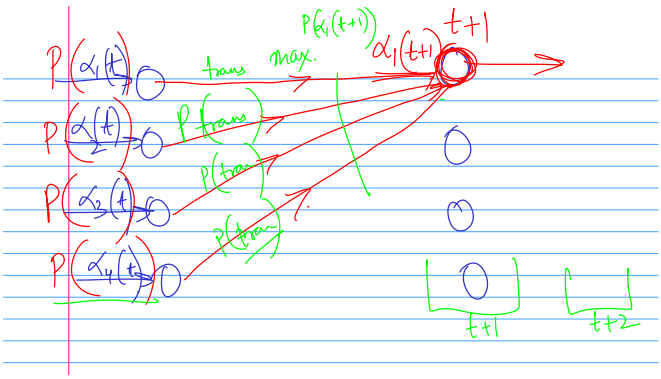
$$\alpha_i(t+1) = \alpha_j(t) \circ s^i$$

where $j = \arg\max_{k=[1,\sigma]} P(\alpha_k(t))\, P(s^k \xrightarrow{w_t} s^i)$



possible prev states

HMM model



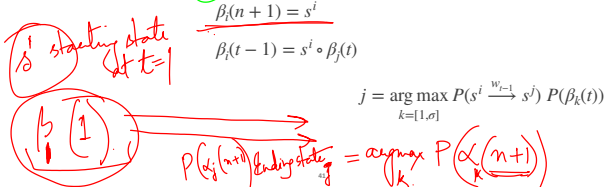
Step by Step constructing the **best backward path** from time t

- It is also possible to use the same algorithm to work backward from the end of the output state sequence to the beginning.
- Let $\beta_i(t)$ be a state sequence with the maximum probability of generating $w_{t,n}$ while starting in state s^i

$$\beta_i(n+1) = s^i$$

$$\beta_i(t-1) = s^i \circ \beta_j(t)$$

$$j = \arg\max_{k=[1,\sigma]} P(s^i \xrightarrow{w_{t-1}} s^j)\, P(\beta_k(t))$$

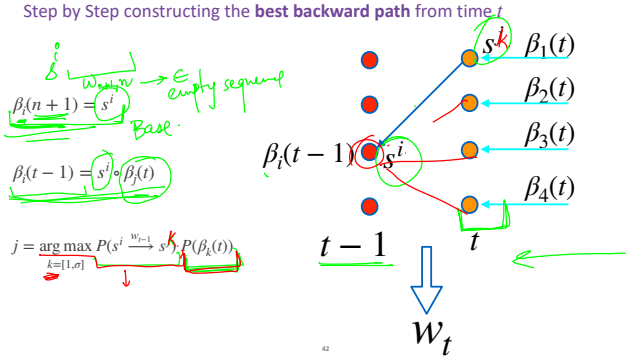


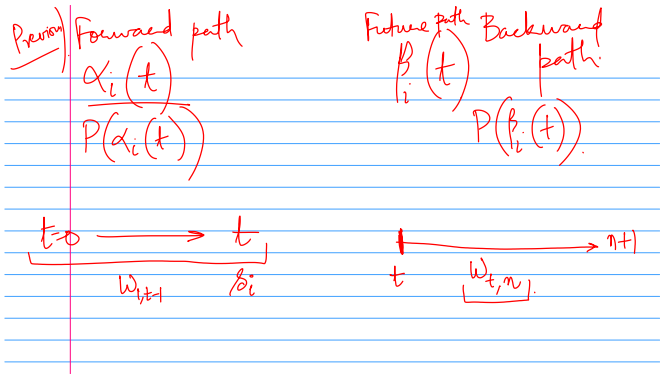
Step by Step constructing the **best backward path** from time t

$$\beta_i(n+1) = s^i$$

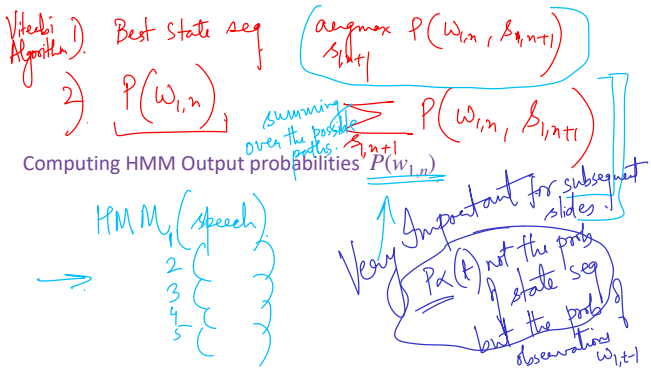
$$\beta_i(t-1) = s^i \circ \beta_j(t)$$

$$j = \arg\max_{k=[1,\sigma]} P(s^i \xrightarrow{w_{t-1}} s^k)\, P(\beta_k(t))$$





- We have seen how the Viterbi algorithm is able to construct the best path (state sequence) using forward recursion or backward recursion.



Computing HMM output probabilities using **forward recursion**

- There are two algorithms for computing the probability of any particular HMM output. The algorithms are mirror images of each other.
- Let $P\alpha_i(t)$ be the probability of producing $w_{1,t-1}$ while ending up in state s^i.

  i.e. $P\alpha_i(t) = P(w_{1,t-1}, S_t = s^i)$ for $t > 1$
- For $t = 1$ we can think of $w_{1,0}$ as the empty string

  Given that we start in state s^1

$$P\alpha_i(1) = \begin{cases} 1.0 & \text{if } i = 1 \\ 0 & \text{otherwise} \end{cases}$$

- The quantity $P\alpha_i(t)$ is called the forward probability.
  - That is, the probability of generating the word sequence up to time t and ending in state i
- We can compute $P\alpha_j(t)$ $\forall t$ in linear time by working forward through the entire output sequence, word by word.

$$P\alpha_j(t+1) = P(w_{1,t}, S_{t+1} = s^j)$$

$$\begin{aligned} P\alpha_j(t+1) &= P(w_{1,t}, S_{t+1} = s^j) \\ &= \sum_{i=1}^{\sigma} P(w_{1,t}, S_t = i, S_{t+1} = s^j) \\ &= \sum_{i=1}^{\sigma} P(w_t, S_{t+1} = s^j \mid w_{1,t-1}, S_t = s^i)\, P(w_{1,t-1}, S_t = s^i) \\ &= \sum_{i=1}^{\sigma} P(w_t, S_{t+1} = s^j \mid S_t = s^i)\, P(w_{1,t-1}, S_t = s^i) \end{aligned}$$

$$P\alpha_j(t+1) = \sum_{i=1}^{\sigma} P(s^i \xrightarrow{w_t} s^j)\, P\alpha_i(t)$$

Forward Recursion

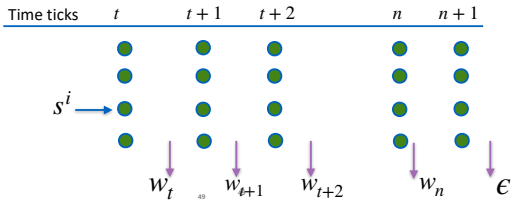
If we compute all the $P\alpha_i(n+1)'s$ then it is easy to compute $P(w_{1,n})$

$$P(w_{1,n}) = \sum_{i=1}^{\sigma} P(w_{1,n}, S_{n+1} = s^i)$$

$$= \sum_{i=1}^{\sigma} P\alpha_i(n+1)$$

Computing $P(w_{1,n})$ using a Mirror Algorithm (Backward recursion)

- To calculate the probability of a sequence, we work backward from the end.
- Backward probability $\underline{P\beta}_i(t)$ is the probability of seeing the sequence $w_{t,n}$ if the state of the HMM at time t is s^i



$$\underline{P\beta}_i(t) = P(w_{t,n} \mid S_t = s^i)$$

$$\underline{P\beta}_1(1) = P(w_{1,n} \mid S_1 = s^1)$$
$$= P(w_{1,n})$$

$$\underline{P\beta}_i(n+1) = P(\epsilon \mid S_{n+1} = s^i)$$
$$= 1$$

- We can calculate $\underline{P\beta}_i(t)$ starting from the end of the output sequence and working our way backward, at each step calculating $\beta_i(t)$ for all states s^i.
- Computation of $\underline{P\beta}_i(t-1)$ can be formulated as backward recursion.

$$\underline{P\beta}_i(t-1) = P(w_{t-1,n} \mid S_{t-1} = s^i)$$
$$= \sum_{j=1}^{\sigma} P(w_{t-1,n}, S_t = s^j \mid S_{t-1} = s^i)$$

$$\underline{P\beta}_i(t-1) = \sum_{j=1}^{\sigma} P(w_{t-1}, S_t = s^j \mid S_{t-1} = s^i)\, P(w_{t,n} \mid w_{t-1}, S_t = s^j, S_{t-1} = s^i)$$

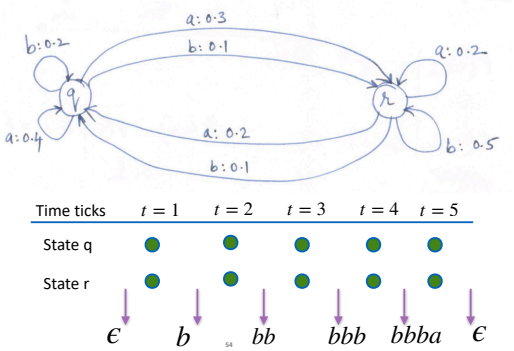
$$= \sum_{j=1}^{\sigma} P(w_{t-1}, S_t = s^j \mid S_{t-1} = s^i)\, P(w_{t,n} \mid w_{t-1}, S_t = s^j, S_{t-1} = s^i)$$

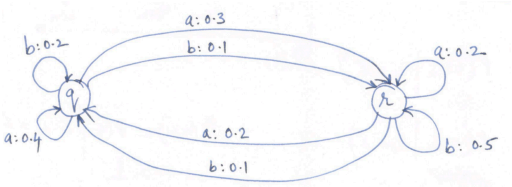
$$= \sum_{j=1}^{\sigma} P(w_{t-1}, S_t = s^j \mid S_{t-1} = s^i)\, P(w_{t,n} \mid S_t = s^j)$$

$$= \sum_{j=1}^{\sigma} P(s^i \xrightarrow{w_{t-1}} s^j)\, \underline{P\beta}_j(t)$$

- $\underline{P\beta}_i(n+1) = 1 \qquad \forall i$

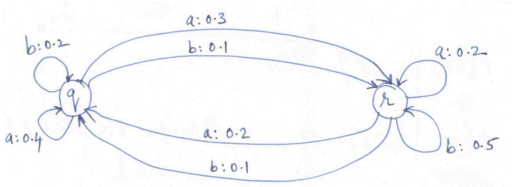
$$\underline{P\beta}_i(t-1) = \sum_{j=1}^{\sigma} P(s^i \xrightarrow{w_{t-1}} s^j)\, \underline{P\beta}_j(t)$$





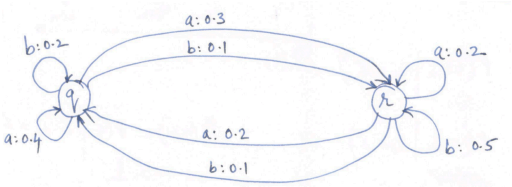
Time ticks	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$
$P\alpha_q(t)$	1.0	0.2	0.05	0.017	0.0148
$P\alpha_r(t)$	0.0	0.1	0.07	0.04	0.0131
$P(w_{1,t})$	1.0	0.3	0.12	0.057	0.0279

$$= \sum_{i=1}^{n} P\alpha_i(t+1)$$



Time ticks	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$
State q					
State r					

$bbba$ bba ba a ϵ



Time ticks	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$
$P\beta_q(t)$	0.0279	0.063	0.18	0.7	1
$P\beta_r(t)$		0.153	0.27	0.4	1

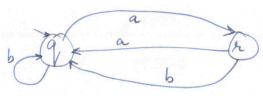
HMM Training: Learning the parameters

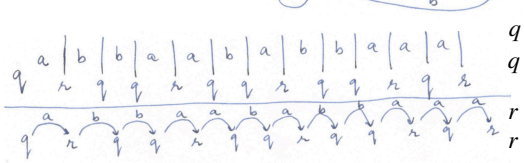
HMM Training

- The training algorithm is called as the forward-backward algorithm or the **Baum-Welch algorithm.**
- The algorithm adjusts the probabilities to make the training sequence as likely as possible.
- First, let's consider how to train a Markov chain model

Estimating Probabilities for a Markov Chain

- Given the representative text created by some Markov Chain (with unknown arc probabilities), we want to estimate those probabilities.
- Training text: **abbababbaaa**





$q \xrightarrow{a} r = 5$
$q \xrightarrow{b} q = 3$
$r \xrightarrow{a} q = 2$
$r \xrightarrow{b} q = 2$

Estimating Probabilities for a Markov Chain

$$P_e(q \xrightarrow{a} r) = \frac{5}{8} \qquad P_e(q \xrightarrow{b} q) = \frac{3}{8}$$

- The estimate of the transition probability $P_e(s^i \xrightarrow{w^k} s^j)$ can be written as

$$P_e(s^i \xrightarrow{w^k} s^j) = \frac{C(s^i \xrightarrow{w^k} s^j)}{\sum_{l,m} C(s^i \xrightarrow{w^m} s^l)}$$

How to estimate the transition counts for an HMM?

- An HMM has many possible paths which can generate the same output symbol sequence.
- However, unlike a Markov Chain, we do not know which transitions were actually used.
- Therefore, we simply pretend that all possible paths were used, and weigh (pro rate) the transitions according to the probability of the path in which the arc appears.

$$C(s^i \xrightarrow{w^k} s^j) = \sum_{s_{1,n+1}} P(s_{1,n+1} \mid w_{1,n}) \;\; \eta(s^i \xrightarrow{w^k} s^j, s_{1,n}, w_{1,n})$$

- $C(s^i \xrightarrow{w^k} s^j) = \frac{1}{P(w_{1,n})} \sum_{s_{1,n+1}} P(s_{1,n+1}, w_{1,n}) \;\; \eta(s^i \xrightarrow{w^k} s^j, s_{1,n}, w_{1,n})$
- Here $\eta(s^i \xrightarrow{w^k} s^j, s_{1,n}, w_{1,n})$ counts the number of times the transition appears in the state sequence (path) $s_{1,n+1}$ when generating the output $w_{1,n}$
- But note the chicken and the egg problem
  Computing $P(w_{1,n})$, $P(s_{1,n+1}, w_{1,n})$ and path probabilities $P(s_{1,n+1} \mid w_{1,n})$ require the transition probabilities

- The iterative process continues till the parameter estimates converge.
  OR
  we can compute $P(w_1, n)$ after every parameter update and declare convergence if the probability assigned to the training corpus does not change much.
- Corpus/training sequence 01011
  4 possible paths could produce this output
  ababaa
  abaaaa
  aaabaa
  aaaaaa

- The term $\frac{1}{P(w_{1,5})}$ need not be calculated since it gets cancelled for the numerator and the denominator when computing $P_e(s^i \xrightarrow{w_k} s^j)$
- A limitation of the training procedure is that it requires iterating through all possible paths for the input sequence.
- Number of paths grow exponentially with the length of the sequence.
- We make certain changes in our computation so that we can train in a more reasonable time.

$$C(s^i \xrightarrow{w_k} s^j) = \sum_{s_{1,n+1}} P(s_{1,n+1} \mid w_{1,n}) \; \eta(s^i \xrightarrow{w_k} s^j, s_{1,n}, w_{1,n})$$

$$= \frac{1}{P(w_{1,n})} \sum_{s_{1,n+1}} P(s_{1,n+1}, w_{1,n}) \; \eta(s^i \xrightarrow{w_k} s^j, s_{1,n}, w_{1,n})$$

$$= \frac{1}{P(w_{1,n})} \sum_{t=1}^{n} \sum_{s_{1,n+1}} P(s_{1,n+1}, w_{1,n}) \delta_t(s^i \xrightarrow{w_k} s^j, s_{1,n+1}, w_{1,n})$$

$$= \frac{1}{P(w_{1,n})} \sum_{t=1}^{n} sum_{s_{1,n+1}} P(S_t = s^i, S_{t+1} = s^j, w_t = w^k, s_{1,n+1}, w_{1,n})$$

- The function $\delta_t(s^i \xrightarrow{s_k} s^j, s_{1,n}, w_{1,n})$takes value 1 if the transition is used at time t and 0 otherwise.
- We then get rid of the δ_t function by putting the requirement that the path under consideration would go through the transition $s^i \xrightarrow{s_k} s^j$ at time tick t.
- There is no harm in putting this requirement because if the path does not go through the transition $s^i \xrightarrow{s_k} s^j$ at time t, then the probability will be zero.

- Also note that

$$\sum_{s_{1,n+1}} P(S_t = s^i, S_{t+1} = s^j, w_t = w^k, s_{1,n+1}, w_{1,n}) = P(S_t = s^i, S_{t+1} = s^j, w_t = w^k, w_{1,n})$$

$$\therefore \quad C(s^i \xrightarrow{s_k} s^j) = \frac{1}{P(w_{1,n})} \sum_{t=1}^{n} P(S_t = s^i, S_{t+1} = s^j, w_t = w^k, w_{1,n})$$

$$= \sum_{t=1}^{n} P(s^i \xrightarrow{s_k} s^j \text{ at time tick } t \mid w_{1,n})$$

$$= \frac{1}{P(w_{1,n})} \sum_{t=1}^{n} P(S_t = s^i, S_{t+1} = s^j, W_t = w^k, w_{1,n})$$

$$= \frac{1}{P(w_{1,n})} \sum_{t=1}^{n} P(w_{1,t-1}, S_t = s^i, W_t = w^k, S_{t+1} = s^j, w_{t+1,n})$$

$$= \frac{1}{P(w_{1,n})} \sum_{t=1}^{n} P(w_{1,t-1}, S_t = s^i) \; P(W_t = w^k, S_{t+1} = s^j \mid w_{1,t-1}, S_t = s^i)$$

- We have already seen how to compute the values for $\underline{P\alpha}$ and $\underline{P\beta}$

$$\underline{P\alpha}_j(t) = P(w_{1,t-1}, S_t = s^j)$$

$$\underline{P\beta}_j(t+1) = P(w_{t+1,n} \mid S_{t+1} = s^j)$$

- $P(w_{t+1,n} \mid w_{1,t}, S_t = s^i, S_{t+1} = s^j)$

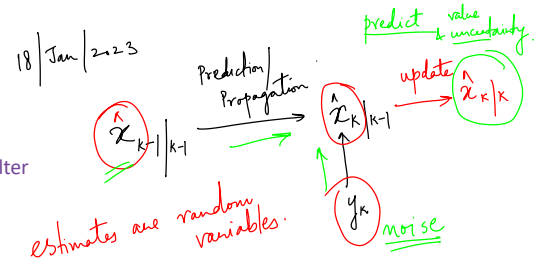
$$= \frac{1}{P(w_{1,n})} \sum_{t=1}^{n} P(w_{1,t-1}, S_t = s^i)$$

$$P(W_t = w^k, S_{t+1} = s^j \mid S_t = s^i)$$

$$P(w_{t+1,n} \mid S_{t+1} = s^j)$$

$$= \frac{1}{P(w_{1,n})} \sum_{t=1}^{n} \underline{P\alpha}_i(t) \; P(s^i \xrightarrow{s_k} s^j) \; \underline{P\beta}_j(t+1)$$

Kalman Filter



Kalman Filter

$\underline{y}$: observation

$\underline{x}$: state

$\underline{n}$: random noise (measurement noise

$\underline{n} \sim \mathcal{N}(0, \underline{R})$

$$\underline{y} = h(\underline{x}) + \underline{n}$$

The estimation problem $\hat{x} = \mathscr{L}(y)$

$\hat{x}$: estimate

$\mathscr{L}$: estimator

- If the observation function $h(\cdot)$ is linear, then
- $$\underline{y} = \underline{H}\underline{x} + \underline{n}$$

where $\underline{y}$ is a linear function of h but with some added noise.

Then $\hat{\underline{x}} = \underline{L}\underline{y}$

where $\underline{L} = \underline{H}^{-1}$

Over constrained System

- If $\underline{H}$ has more rows than the number of columns then there is no inverse of $\underline{H}$.
- But we can find the best $\hat{\underline{x}}$ such that the residue $\|\underline{y} - \underline{H}\hat{\underline{x}}\|$ is as small as possible.
- The best estimator for a linear observation function happens to be a linear function.
- We denote the residue as $\underline{n} = \underline{y} - \underline{H}\hat{\underline{x}}$

- Different equations can have noise terms of different variance. So some equations may be more reliable.
- We formulate minimisation of the norm of $\underline{W}\underline{n}$ where $\underline{W}$ is a matrix as

$$\|\underline{W}\underline{n}\|^2 = \underline{n}^\top \underline{W}^\top \underline{W}\underline{n}$$

which is equivalent to solving the system

$$\underline{W}\underline{y} = \underline{W}\underline{H}\underline{x}$$

because $\underline{W}\underline{n} = \underline{W}(\underline{y} - \underline{H}\underline{x}) = \underline{W}\underline{y} - \underline{W}\underline{H}\underline{x}$

- That is we want to solve $\underline{W}\underline{y} = \underline{W}\underline{H}\underline{x}$ subject to minimising $\|\underline{W}\underline{y} - \underline{W}\underline{H}\underline{x}\|^2$

- Consider the standard notation for a system of equations $\underline{A}\underline{x} = \underline{b}$ to be solved for $\hat{\underline{x}}$ so as to minimise $\|\underline{b} - \underline{A}\underline{x}\|^2$
- The objective to be minimised can be re-written as $\|\underline{A}\underline{x} - \underline{b}\|^2$ or

$$(\underline{A}\underline{x} - \underline{b})^\top(\underline{A}\underline{x} - \underline{b}) \quad \text{or} \quad (\underline{x}^\top\underline{A}^\top - \underline{b}^T)(\underline{A}\underline{x} - \underline{b})$$

- Or $f \equiv (\underline{x}^\top\underline{A}^T\underline{A}\underline{x} - \underline{x}^T\underline{A}^\top\underline{b} - \underline{b}^T\underline{A}\underline{x} + \underline{b}^T\underline{b})$

$$\frac{df}{d\underline{x}} = 2\underline{A}^\top\underline{A}\underline{x} - \underline{A}^\top\underline{b} - \underline{A}^\top\underline{b} + 0$$

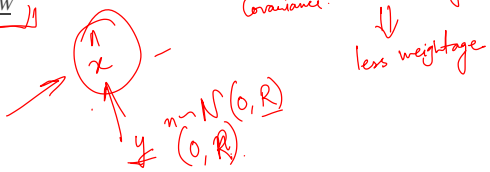
- $\frac{df}{d\underline{x}} = 2\underline{A}^\top\underline{A} * \underline{x} - 2A^\top b = 0$
- Therefore $\underline{A}^\top\underline{A}\underline{x} = \underline{A}^\top\underline{b}$ or $\hat{\underline{x}} = (\underline{A}^\top\underline{A})^{-1}\underline{A}^T\underline{b}$
- The given system of equations $\underline{W}\underline{H}\underline{x} = \underline{W}\underline{y}$

Substituting $\underline{A} \equiv \underline{W}\underline{H}$ and $\underline{b} \equiv \underline{W}\underline{y}$ in $\hat{\underline{x}} = (\underline{A}^\top\underline{A})^{-1}\underline{A}^T\underline{b}$

- $\hat{\underline{x}} = ((\underline{W}\underline{H})^\top\underline{W}\underline{H})^{-1}(\underline{W}\underline{H})^\top\underline{W}\underline{y} = (\underline{H}^\top\underline{W}^\top\underline{W}\underline{H})^{-1}\underline{H}^\top\underline{W}^\top\underline{W}\underline{y}$
$= (\underline{H}^\top\underline{R}^{-1}\underline{H})^{-1}(\underline{H}^\top\underline{R}^{-1})\underline{y}$

- R is the covariance matrix of the measurement noise.
- Entries in R which correspond to a large variance or covariance will translate to smaller entries in the W matrix.

$$R^{-1} = W^T W$$



Quality of the estimator

- The size of the covariance matrix should be small.
- Covariance matrix $P = \mathbb{E}\left[(x - \hat{x})(x - \hat{x})^T\right]$
- A smaller norm of P implies reduced uncertainty or fluctuations
- The estimate $\hat{x}$ should be unbiased

$$\mathbb{E}\left[x - \hat{x}\right] = 0$$

- For a linear estimator $\hat{x} = Ly$ corresponding to the system $y = Hx + n$

$$\mathbb{E}\left[x - \hat{x}\right] = \mathbb{E}\left[x - Ly\right] = \mathbb{E}\left[x - L(Hx + n)\right] = \mathbb{E}\left[(I - LH)x\right] - \mathbb{E}\left[Ln\right] = 0$$

- $\mathbb{E}\left[(I - LH)x\right] = 0$

  or $(I - LH)\,\mathbb{E}\left[x\right] = 0$

  $\therefore I = LH$ or $LH = I$

- $\mathbb{E}\left[\hat{x}\right] = x$

$y = Hx + n$

$\hat{x} = Ly$

$\hat{x} = H^{-1}y$

- The covariance matrix of our estimate

$$P = \mathbb{E}\left[(x - \hat{x})(x - \hat{x})^T\right]$$

$$= \mathbb{E}\left[(x - Ly)(x - Ly)^T\right]$$

- Substituting $y = Hx + n$ we have

$$P = \mathbb{E}\left[(x - LHx - Ln)\,(x - LHx - Ln)^T\right]$$

$$= \mathbb{E}\left[\left[(I - LH)x - Ln\right]\left[(I - LH)x - Ln\right]^T\right] = \mathbb{E}\left[(Ln)\,(Ln)^T\right]$$

($LH = I$; $(I - LH)x = 0$)

- $= \mathbb{E}\left[Lnn^T L\right]$

$$= L\,\mathbb{E}\left[nn^T\right]\,L$$

$$P = L\,R\,L^T$$

We define $P \equiv LRL^T$

- Here R is the covariance matrix of the measurement noise.
- Substituting for the estimator $L = (H^T R^{-1} H)^{-1} H^T R^{-1}$ we get

n: measurement noise