Sequential data

Much of this section follows Bishop chapter 13 (posted)

See also Murphy chapters 17 and 18

Sequential data

Sequential data is everywhere.

Examples:

- spoken language (word production)
- written language (sentence level statistics)
- weather
- human movement
- stock market data

Sequential data

Graphical models for such data?

The complexity of the representation seems to increase with time.

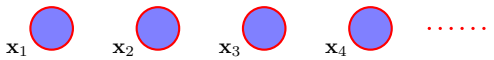
Observations over time tend to depend on the past.

We can simply life by assuming that the distant past does not matter.

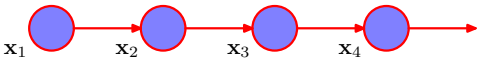
> If we assume that history does not matter other than the immediate previous entity, we have a first order Markov model.
>
> If what happens now depends on two previous entities, we have a second order Markov model.

Markov chains

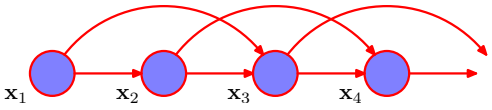
Zeroth order



First order



Second order



Temporal statistical clustering

In sequence data, cluster membership can have temporal (or sequential) structure.

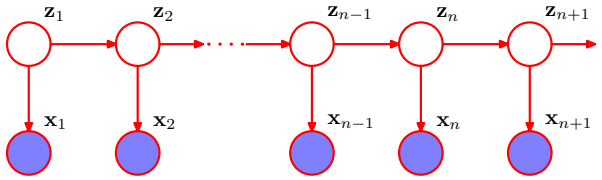
The data comes from the current cluster (as usual), but what is the next cluster?

Example, rain and sleet come from "stormy" and sunshine from "fair weather".

But now, our hidden cluster variables are what depends on the past. The previous "state" represents all history.

Temporal statistical clustering

Hidden Markov Model (HMM).



The particular state encodes the important part of history.

Temporal statistical clustering

$p(x_n|z_n)$

Once you know your cluster, things are easy.

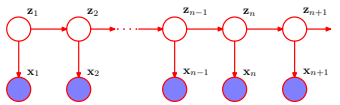
But the cluster is now changing over time.

Markovian assumptions

As before, if the current state depends on only the previous state, we have a first order Markov model.

The basic HMM is like a mixture model, with the mix of mixture components being used for the current observations depends on the last mixture component.

$$z_{n+1} \perp z_{n-1} \,|\, z_n$$



Markovian assumptions

Represent each component as a "state".

Then, for first order Markov models, this leads to the concept of "transition" probabilities.

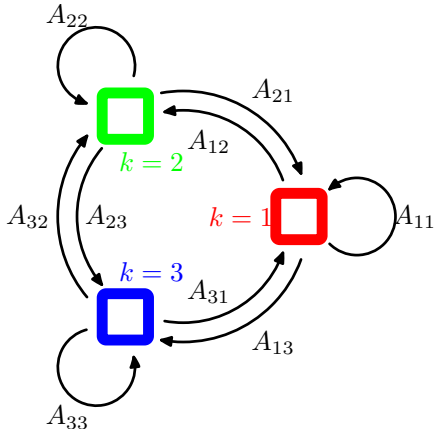
$$A_{jk} \equiv p\left(z_{nk}=1 \middle| z_{n-,j}=1\right)$$

$$0 \le A_{jk} \le 1 \quad \text{and} \quad \sum_k A_{jk} = 1$$

The random variable, z, is a vector over K possible states (e.g., two for stormy vs fair-weather), for each time point.

Transition matrix representation

(Not a graphical model)



Starting state

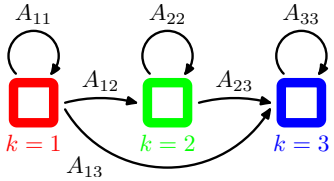
Our HMM will be a generative model, so we need to know how to start.

$$\pi_k \equiv p\left(z_{1k}=1\right)$$

$$\text{with } 0 \le \pi_k \le 1 \text{ and } \sum_k \pi_k = 1$$

Left to right HMM

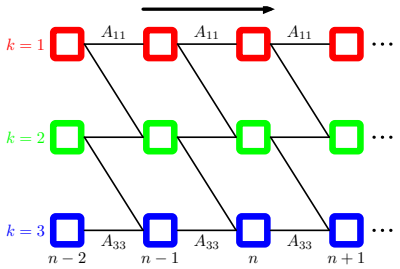
Constrain state number to increase



(State transition diagram)

Left to right HMM

Even more constrained, left to right HMM with single state jumps.



(Lattice diagram)

HMM parameter summary

$\theta = \{\pi, A, \phi\}$

π is probability over initial states

A is transition matrix

ϕ are the data emmission probabilities (e.g., means of Gaussians)

Data distribution from an HMM

$p(X|\theta)$ is a marginalization over Z.

$p(X, Z|\theta) = ?$

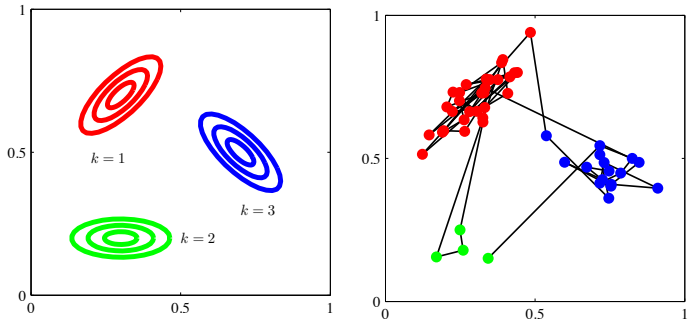
Data distribution from an HMM

An HMM is specified by: $\theta = \{\pi, A, \phi\}$

$$p(X, Z|\theta) = p(z_1|\pi)\left[\prod_{n=2}^{N} p(z_n|z_{n-1}, A)\right]\prod_{m=1}^{N} p(x_m|z_m, \phi)$$

(complete data, i.e., we can generate from this).

Data distribution from an HMM



Transition probability to another state is 5%

Classic HMM computational problems

Given data, what is the HMM (**learning**).

Given an HMM, what is the **probability distribution of states for** each state variable (z_n in our notation).

Given an HMM, what is the most likely **state sequence** for some data?

Learning the HMM (sketch)

If we know the state distributions, we can compute the parameters.

If we know the parameters, we can compute the state distributions (provided we know how to solve the second problem).

Recall the General EM algorithm

1. Choose initial values for $\theta^{(s=1)}$
   (can also do assignments, but then jump to M step).
2. E step: Evalute $p\left(Z \middle| X, \theta^{(s)}\right)$
3. M step: Evalute $\theta^{(s+1)} = \arg\max_{\theta} \left\{ \mathrm{Q}\left(\theta^{(s+1)}, \theta^{(s)}\right) \right\}$

   where $\mathrm{Q}\left(\theta^{(s+1)}, \theta^{(s)}\right) = \sum_{Z} p\left(Z \middle| X, \theta^{(s)}\right) \log\left(p\left(X, Z \middle| \theta^{(s+1)}\right)\right)$
4. Check for convergence; If not done, goto 2.

★ At each step, our objective function is increases unless it is at a local maximum. It is important to check this is

EM for HMM (sketch)

In the simple clustering case (e.g., GMM), the E step was simple. For HMM it is a bit more involved.

The M step works a lot like the GMM. Consider it first.

EM for HMM (sketch)

$$p(X,Z|\theta) = p(z_1|\pi)\left[\prod_{n=2}^{N} p(z_n|z_{n-1}, A)\right]\prod_{n=1}^{N} p(x_n|z_n, \phi)$$

$$= \prod_{k=1}^{K} \pi^{z_{1k}} \left[\prod_{n=2}^{N}\prod_{j=1}^{K}\prod_{k=1}^{K}\left(p(z_n|z_{n-1}, A)\right)^{z_{n-1,j}\cdot z_{n,k}}\right]\prod_{n=1}^{N}\prod_{k=1}^{K}\left(p(x_n|z_n, \phi)\right)^{z_{nk}}$$

Remember our "indicator variable" notation. Z is a particular assignment of the missing values (i.e., which cluster the HMM was in at each time. For each time point, i, one of the values of z_n is one, and the others are zero. So, it "selects" the factor for the particular state at that time.

EM for HMM (sketch)

$$p(X,Z|\theta) = p(z_1|\pi)\left[\prod_{n=2}^{N} p(z_n|z_{n-1}, A)\right]\prod_{n=1}^{N} p(x_n|z_n, \phi)$$

$$= \prod_{k=1}^{K} \pi^{z_{1k}} \left[\prod_{n=2}^{N}\prod_{j=1}^{K}\prod_{k=1}^{K}\left(p(z_n|z_{n-1}, A)\right)^{z_{n-1,j}\cdot z_{n,k}}\right]\prod_{n=1}^{N}\prod_{k=1}^{K}\left(p(x_n|z_n, \phi)\right)^{z_{nk}}$$

$$\log\left(p(X,Z|\theta)\right) =$$

$$\sum_{k=1}^{K} z_{1k}\log(\pi) + \sum_{n=2}^{N}\sum_{j=1}^{K}\sum_{k=1}^{K} z_{n-1,j}z_{n,k}\log\left(p(z_n|z_{n-1}, A)\right) + \sum_{n=1}^{N}\sum_{k}^{K} z_{nk}\log\left(p(x_m|z_m, \phi)\right)$$

M step for HMM

We assume the E step computed distributions for

The degree each state explains each data point (analogous to GMM responsibilities).

$$\gamma(z_n) = p\left(z_n|X,\theta^{(s)}\right)$$

The degree that the combination of a state, and a previous one explain two data points.

$$\xi(z_{n-1},z_n) = p\left(z_{n-1},z_n|X,\theta^{(s)}\right)$$

"xi"

EM for HMM (sketch)

$$\log\left(p\left(X,Z|\theta\right)\right)=$$

$$\sum_{k=1}^{K} z_{1k}\log(\pi)+\sum_{n=2}^{N}\sum_{j=1}^{K}\sum_{k=1}^{K} z_{n-1,j}z_{n,k}\log\left(p\left(z_n|z_{n-1},A\right)\right)+\sum_{n=1}^{N}\sum_{k}^{K} z_{nk}\log\left(p\left(x_m|z_m,\phi\right)\right)$$

We define

$$\gamma\left(z_n\right)=p\left(z_n|X,\theta^{(s)}\right)$$

$$\xi\left(z_{n-1},z_n\right)=p\left(z_{n-1},z_n|X,\theta^{(s)}\right)$$

"xi"

EM for HMM (sketch)

$$\log\left(p\left(X,Z|\theta\right)\right)=$$

$$\sum_{k=1}^{K} z_{1k}\log(\pi)+\sum_{n=2}^{N}\sum_{j=1}^{K}\sum_{k=1}^{K} z_{n-1,j}z_{n,k}\log\left(p\left(z_n|z_{n-1},A\right)\right)+\sum_{n=1}^{N}\sum_{k}^{K} z_{nk}\log\left(p\left(x_m|z_m,\phi\right)\right)$$

By analogy with the GMM

$$Q\left(\theta^{(s+1)},\theta^{(s)}\right)=\sum_{z} p\left(Z|\theta^{(s)}\right)\log\left(X,Z|\theta^{(s+1)}\right)$$

$$=\sum_{k=1}^{K}\gamma\left(z_{1k}\right)\log(\pi)+\sum_{n=2}^{N}\sum_{j=1}^{K}\sum_{k=1}^{K}\xi\left(z_{n-1,j}z_{n,k}\right)\log\left(p\left(z_n|z_{n-1},A\right)\right)$$

$$+\sum_{n=1}^{N}\sum_{k}^{K}\gamma\left(z_{nk}\right)\log\left(p\left(x_m|z_m,\phi\right)\right)$$

EM for HMM (sketch)

Doing the maximization using Lagrange multipliers gives us

$$\pi_k=\frac{\gamma\left(z_{1k}\right)}{\sum_{k'}\gamma\left(z_{1k'}\right)}$$

Much like the GMM. Taking the partial derivative for π_k kills second and third terms.

$$A_{jk}=\frac{\sum_{n=2}\zeta\left(z_{n-1,j},z_{nk}\right)}{\sum_{k'}\sum_{n=2}\zeta\left(z_{n-1,j},z_{nk'}\right)}$$

EM for HMM (sketch)

The maximization of $p\left(x_n|\phi\right)$ is exactly the same as the mixture model.

For example, if we have Gaussian emmisions, then

$$\mu_k=\frac{\sum_{n} x_n\,\gamma\left(z_{nk}\right)}{\sum_{n}\gamma\left(z_{nk}\right)}$$

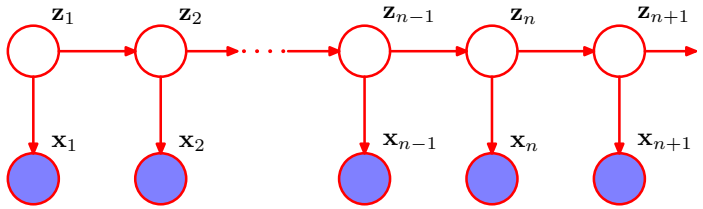
E step for EM for HMM

Computing the E step is a bit more involved.

Recall that in the mixture case it was easy because we only needed to consider the relative likelihood that each cluster independently explain the observations.

However, here the sequence also must play a role.

Graphical model for the E step



Note that our task is to compute marginal probabilities

Computing marginals in an HMM

Various names, flavors, notations, ...

Forward-Backward algorithm

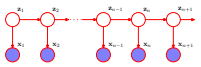
Alpha-beta algorithm

Sum-product for HMM

(Bishop also says "Baum Welch" but that is a synonym for the EM algorithm as whole).

Alpha-beta algorithm

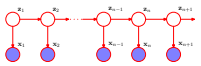
$$\gamma(z_n) = p(z_n|X)$$
$$= \frac{p(X|z_n)p(z_n)}{p(X)}$$
$$= \frac{p(x_1,....,x_n|z_n)\,p(x_{n+1},....,x_N|z_n)\,p(z_n)}{p(X)}$$
$$= \frac{p(x_1,....,x_n,z_n)\,p(x_{n+1},....,x_N|z_n)}{p(X)}$$
$$= \frac{\alpha(z_n)\beta(z_n)}{p(X)}$$



Where we define

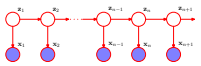
$$\alpha(z_n) = p(x_1,....,x_n,z_n)$$
$$\beta(z_n) = p(x_{n+1},....,x_N|z_n)$$

Expressing alpha recursively



$$
\begin{aligned}
\alpha(z_n) &= p(x_1,....,x_n,z_n) \\
&= p(x_1,....,x_n|z_n)\,p(z_n) && \text{(definition of "|")} \\
&= p(x_n|z_n)\,p(x_1,....,x_{n-1}|z_n)\,p(z_n) && \text{(conditional independence)} \\
&= p(x_n|z_n)\,p(x_1,....,x_{n-1},z_n) && \text{(definition of "|")} \\
&= p(x_n|z_n)\sum_{z_{n-1}} p(x_1,....,x_{n-1},z_{n-1},z_n) && \text{(marginal)} \\
&= p(x_n|z_n)\sum_{z_{n-1}} p(x_1,....,x_{n-1},z_n|z_{n-1})p(z_{n-1}) && \text{(definition of "|")} \\
&= p(x_n|z_n)\sum_{z_{n-1}} p(x_1,....,x_{n-1}|z_{n-1})p(z_n|z_{n-1})\,p(z_{n-1}) && \text{(conditional independence)} \\
&= p(x_n|z_n)\sum_{z_{n-1}} p(x_1,....,x_{n-1},z_{n-1})p(z_n|z_{n-1}) && \text{(definition of "|")} \\
&= p(x_n|z_n)\sum_{z_{n-1}} \alpha(z_{n-1})p(z_n|z_{n-1}) && \text{(definition of } \alpha(z_n))
\end{aligned}
$$

Expressing alpha recursively



$$\alpha(z_n) = p(x_n|z_n)\sum_{z_{n-1}} \alpha(z_{n-1})p(z_n|z_{n-1})$$

This is a recursive evaluation of alpha. So we can compute all of them easily if we know the first one, $\alpha(z_1)$.

$$
\begin{aligned}
\alpha(z_1) &= p(x_1,z_1) && \text{(we defined } \alpha(z_n) = p(x_1,....,x_n,z_n)\text{)} \\
&= p(z_1)\,p(x_1|z_1) && \text{(this is a K dimensional vector for fixed } x_1\text{)}
\end{aligned}
$$

$$\alpha(z_1)_k = \pi_k\, p(x_1|\phi_k)$$

Alpha-beta algorithm

Similarly, we can derive a recurrence relation for beta

$$\beta(z_n) = \sum_{z_{n+1}} \beta(z_{n+1})p(x_{n+1}|z_{n+1})\,p(z_{n+1}|z_n)$$

Alpha-beta algorithm

The details for $\beta(z_n) = \sum_{z_{n+1}} \beta(z_{n+1})p(x_{n+1}|z_{n+1})\,p(z_{n+1}|z_n)$

$$
\begin{aligned}
\beta(\mathbf{z}_n) &= p(\mathbf{x}_{n+1},\ldots,\mathbf{x}_N|\mathbf{z}_n) \\
&= \sum_{\mathbf{z}_{n+1}} p(\mathbf{x}_{n+1},\ldots,\mathbf{x}_N,\mathbf{z}_{n+1}|\mathbf{z}_n) \\
&= \sum_{\mathbf{z}_{n+1}} p(\mathbf{x}_{n+1},\ldots,\mathbf{x}_N|\mathbf{z}_n,\mathbf{z}_{n+1})p(\mathbf{z}_{n+1}|\mathbf{z}_n) \\
&= \sum_{\mathbf{z}_{n+1}} p(\mathbf{x}_{n+1},\ldots,\mathbf{x}_N|\mathbf{z}_{n+1})p(\mathbf{z}_{n+1}|\mathbf{z}_n) \\
&= \sum_{\mathbf{z}_{n+1}} p(\mathbf{x}_{n+2},\ldots,\mathbf{x}_N|\mathbf{z}_{n+1})p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1})p(\mathbf{z}_{n+1}|\mathbf{z}_n).
\end{aligned}
$$

Alpha-beta algorithm

Our recurrence relation for beta

$$\beta(z_n) = \sum_{z_{n+1}} \beta(z_{n+1}) p(x_{n+1}|z_{n+1}) p(z_{n+1}|z_n)$$

We can compute the betas if we know the last one.

$$\begin{aligned} p(z_N|X) &= \frac{\alpha(z_N)\beta(z_N)}{p(X)} \\ &= \frac{p(X,z_N)\beta(z_N)}{p(X)} \qquad \text{(we defined } \alpha(z_n) = p(x_1,\ldots,x_n,z_n)\text{)} \\ &= p(z_N|X)\beta(z_N) \end{aligned}$$

So $\beta(z_N) = 1$

Alpha-beta algorithm

Given the alphas and betas, we can compute all the quantities we need for the E step.

$$\gamma(z_n) = \frac{\alpha(z_n)\beta(z_n)}{p(X)} \qquad \text{(our definition)}$$

We know that $\sum_{z_n} \gamma(z_n) = 1$

so $\sum_{z_n} \frac{\alpha(z_n)\beta(z_n)}{p(X)} = 1$

and $p(X) = \sum_{z_n} \alpha(z_n)\beta(z_n)$

We do not need $p(X)$ for EM, but it is the likelihood which we want to monitor ($p(X) = p(X|\theta^{(s)})$).

Alpha-beta algorithm

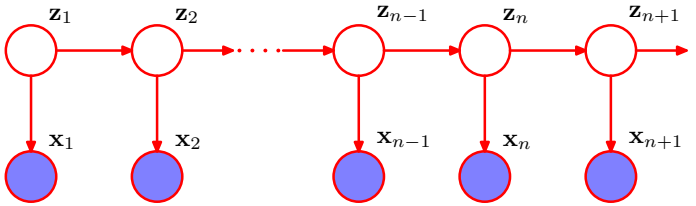
Given the alphas and betas, we can compute all the quantities we need for the E step.

$$\begin{aligned} \xi(\mathbf{z}_{n-1}, \mathbf{z}_n) &= p(\mathbf{z}_{n-1}, \mathbf{z}_n|\mathbf{X}) \\ &= \frac{p(\mathbf{X}|\mathbf{z}_{n-1}, \mathbf{z}_n) p(\mathbf{z}_{n-1}, \mathbf{z}_n)}{p(\mathbf{X})} \\ &= \frac{p(\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}|\mathbf{z}_{n-1}) p(\mathbf{x}_n|\mathbf{z}_n) p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N|\mathbf{z}_n) p(\mathbf{z}_n|\mathbf{z}_{n-1}) p(\mathbf{z}_{n-1})}{p(\mathbf{X})} \\ &= \frac{\alpha(\mathbf{z}_{n-1}) p(\mathbf{x}_n|\mathbf{z}_n) p(\mathbf{z}_n|\mathbf{z}_{n-1}) \beta(\mathbf{z}_n)}{p(\mathbf{X})} \end{aligned} \qquad (13.43)$$

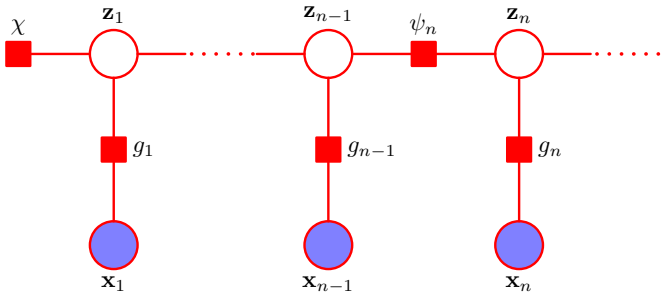
(in Bishop)

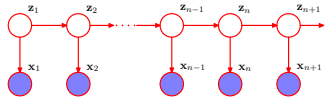
Computing marginals, version two

We can apply sum-product to our E step graph.



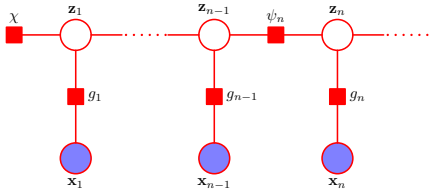
Factor graph





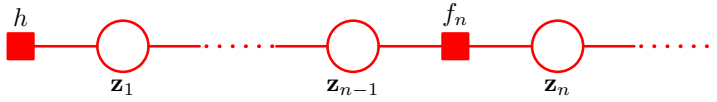
(Directed graph for reference)

Simplified factor graph

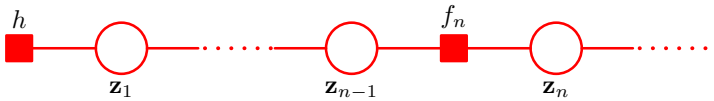


(Canonical factor graph from previous slide)

Since we condition on all the x's, we can simplify the graph by treating the emissions as constants, and putting them into the factors for the z's to get a simple chain.



Review of sum-product concepts



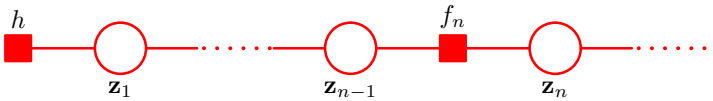
The marginal for each node is a product of the incoming messages.

This is analogous to setting up the marginal as a product of alpha and beta factors in the previous treatment.

Since we have a chain, this is just two messages, one coming from the left, the other from the right.

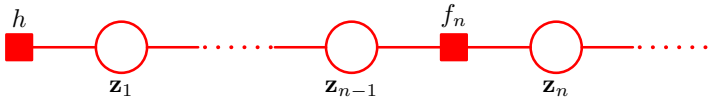
To compute all marginals, we pass the left and right messages from one end to the other.

Sum-product for HMM



$$h = p(z_1)\underbrace{p(x_1|z_1)}_{\text{extra for the nodes we pruned}}$$

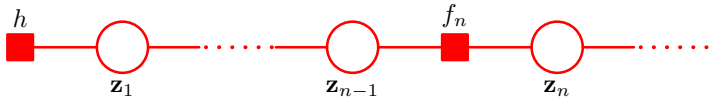
Sum-product for HMM



$$h = p(z_1)\,p(x_1|z_1)$$

$$f_n = p(z_n|z_{n-1})\underbrace{p(x_n|z_n)}_{\text{extra for pruned nodes}}$$

Sum-product for HMM

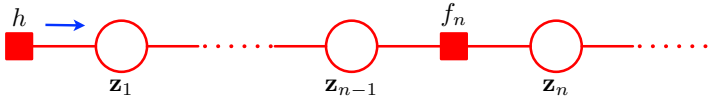


The nodes all have at most two links (it is a chain) so they just pass the incoming message to the outgoing link.

$$\text{i.e.,}\quad \mu_{f_n \to z_n}(z_n) = \mu_{f_n \to f_{n+1}}(z_n)$$

The nodes also (metaphorically) is where we think of the messages being stored if we are computing multiple marginals (which we are in this case).

Sum-product for HMM



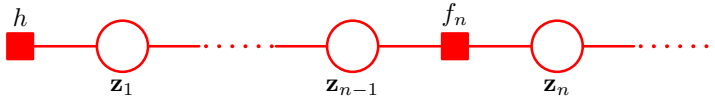
Factor node actions on left to right messages

$$\mu_{f_n \to f_{n+1}}(z_n) = \sum_{z_{n-1}} f_n(z_{n-1}, z_n)\,\mu_{f_{n-1} \to f_n}(z_{n-1})$$

The first message is

$$h = p(z_1)\,p(x_1|z_1) = p(x_1, z_1) = \alpha(z_1)$$

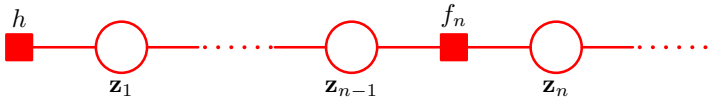
Sum-product for HMM



If we identify $\mu_{f_n \to f_{n+1}}(z_n) = \alpha(z_n)$

$$\begin{aligned}
\alpha(z_n) &= \sum_{z_{n-1}} f_n(z_{n-1}, z_n)\,\alpha(z_{n-1}) \\
&= \sum_{z_{n-1}} p(z_n|z_{n-1})\,p(x_n|z_n)\,\alpha(z_{n-1}) \\
&= p(x_n|z_n) \sum_{z_{n-1}} p(z_n|z_{n-1})\,\alpha(z_{n-1})
\end{aligned}$$

Sum-product for HMM



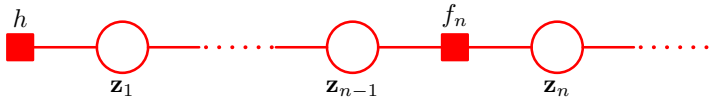
Factor node actions on right to left messages

$$\mu_{f_{n+1}\to f_n}(z_n)=\sum_{z_{n+1}} f_{n+1}(z_n,z_{n+1})\mu_{f_{n+2}\to f_{n+1}}(z_{n+1})$$

Identify $\beta(z_n)\equiv\mu_{f_{n+1}\to f_n}(z_n)$ to get

$$\beta(z_n)=\sum_{z_{n+1}} f_{n+1}(z_n,z_{n+1})\beta(z_{n+1})$$

Sum-product for HMM



Identify $\beta(z_n)\equiv\mu_{f_{n+1}\to f_n}(z_n)$ to get

$$\beta(z_n)=\sum_{z_{n+1}} f_{n+1}(z_n,z_{n+1})\beta(z_{n+1})$$

Recalling that $f_{n+1}=p(z_{n+1}|z_n)p(x_{n+1}|z_{n+1})$

$$\beta(z_n)=\sum_{z_{n+1}} p(z_{n+1}|z_n)p(x_{n+1}|z_{n+1})\beta(z_{n+1})$$

Sum-product for HMM

We have re-derived the alpha-beta version of forward-backward

Forward

$$\alpha(z_1)=p(z_1)p(x_1|z_1)$$
$$\alpha(z_n)=p(x_n|z_n)\sum_{z_{n-1}} p(z_n|z_{n-1})\alpha(z_{n-1})$$

Backward

$$\beta(z_N)=1$$
$$\beta(z_n)=\sum_{z_{n+1}} p(z_{n+1}|z_n)p(x_{n+1}|z_{n+1})\beta(z_{n+1})$$

Sum-product for E step in the HMM learning problem (review)

Given all $\alpha(z_n)$ and $\beta(z_n)$

$$\gamma(z_n)=\frac{\alpha(z_n)\beta(z_n)}{p(X)}$$

$$p(X)=\sum_{z_n}\alpha(z_n)\beta(z_n)$$

$$\xi(z_{n-1},z_n)=\frac{\alpha(z_{n-1})p(x_n|z_n)p(z_n|z_{n-1})\beta(z_n)}{p(X)}$$

Rescaled alpha beta (Bishop, 13.2.4)

The alpha-beta algorithm has similar precision problems to the ones for EM where we discussed the fix of scaling log quantities by the max, before exponentiation for normalizing.

One way to handle this is to reformulate the alpha-beta algorithm in terms of:

$$\hat{\alpha}(z_n) = p(z_n | x_1, ..., x_n) = \frac{\alpha(z_n)}{p(x_1, ..., x_n)}$$

$$\hat{\beta}(z_n) = \frac{p(x_{n+1}, ..., x_N | z_n)}{p(x_{n+1}, ..., x_N | x_1, ..., x_n)}$$

Rescaled alpha beta (Bishop, 13.2.4)

$$\hat{\alpha}(z_n) = p(z_n | x_1, ..., x_n) = \frac{\alpha(z_n)}{p(x_1, ..., x_n)}$$

Let $c_n = p(x_n | x_1, ..., x_{n-1})$

and note that $p(x_1, ..., x_n) = \prod_{m=1}^{n} c_m$. Then

$$c_n \hat{\alpha}(z_n) = p(x_n | z_n) \sum_{z_{n-1}} \hat{\alpha}(z_{n-1}) p(z_n | z_{n-1})$$

and we get c_n as the normalizer of the RHS.

(See Bishop for the betas).

Classic HMM computational problems

Given data, what is the HMM (**learning**). ✓

Given an HMM, what is the **distribution over the state** variables. Also, **how likely** are the observations, given the model. ✓

Given an HMM, what is the most likely **state sequence** for some data?

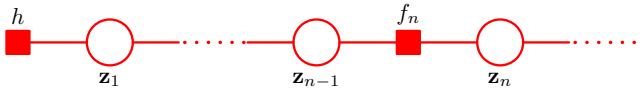
Viterbi algorithm (special case of max-sum)

Recall max-sum

Forward direction is like sum-product, except
- We take the max instead of sum
- We use sum of logs instead of product
- We remember incoming variable values that give max (*)

Backwards direction is simply backtracking on (*).

Recall simplified factor graph



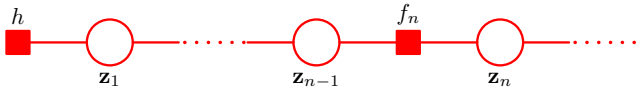
$$h = p(z_1)\,p(x_1|z_1) \qquad f_n = p(z_n|z_{n-1})\,p(x_n|z_n)$$

Left to right messages

$$\omega(z_n) = \log(x_n|z_n) + \max_{z_{n-1}}\left\{\log\left(p(z_n|z_{n-1}) + \omega(z_{n-1})\right)\right\}$$

$$\omega(z_1) = \log\left(p(z_1)\right) + \log\left(p(x_1|z_1)\right)$$

Intuitive understanding



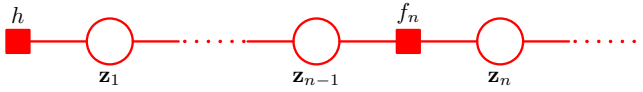
$$\omega(z_n) = \log(x_n|z_n) + \max_{z_{n-1}}\left\{\log\left(p(z_n|z_{n-1}) + \omega(z_{n-1})\right)\right\}$$

Consider all possible paths to each of the *k* states for time *n*.

The message encodes the probabilities for the maximum probability path for each of the *K* states.

EG, if you are in state *k*, this records is the probability of being there by via the maximal probably sequence.

Intuitive understanding



The message is the vector of probabilities for the maximum probability path for each of the *K* states.

$$\omega(z_n) = \log(x_n|z_n) + \max_{z_{n-1}}\left\{\log\left(p(z_n|z_{n-1}) + \omega(z_{n-1})\right)\right\}$$

For each state *k*

Consider getting there from each previous state k'

The message is the vector of probabilities for the maximum probability path for each of the K states.

$$\omega(z_n) = \log(x_n|z_n) + \max_{z_{n-1}}\left\{\log\left(p(z_n|z_{n-1}) + \omega(z_{n-1})\right)\right\}$$

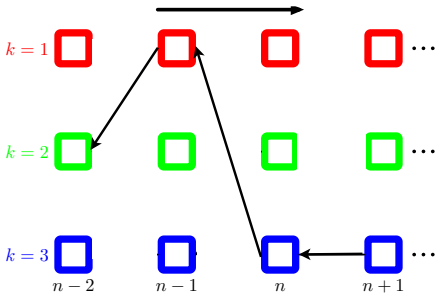
For each state k

Consider getting there from each previous state k'

We can see that this is the new maximum

For Viterbi, we need to remember the previous state, k', for each k.

Intuitive understanding



The max path is shown (but we only know it when we get to the end).

To find the path, we need to chase the back pointers.

Final comments on learning

In many applications, the states have specified meaning, and are available in training data, so EM is not needed.

(Most authors still call this an HMM because states are hidden when the model is used).

We described training the HMM based on a single data sequence, but often multiple sequences that come from the same HMM are used (modifying the algorithm is very straightforward).

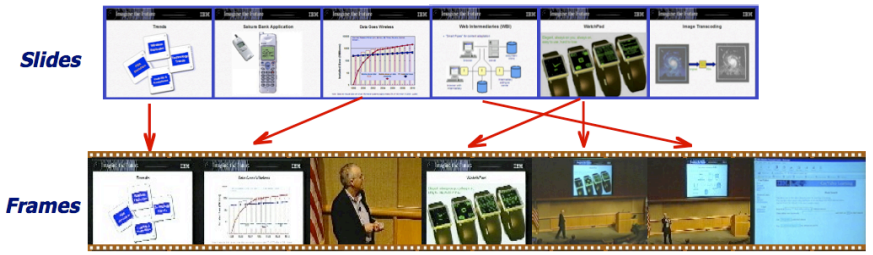
Two HMM examples (specified states)

Domain is SLIC (Semantically Linked Instructional Content).

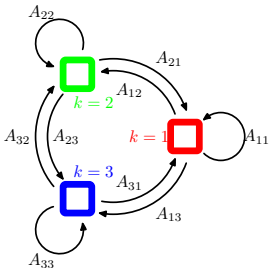
1) Temporal information for matching video frames to slides.

2) Aligning noisy speech transcripts with slides.

Matching slides to video frames



Matching slides to video frames



Our state sequence corresponds to what slide is being shown.

$$p(X,Z|\theta) = p(z_1|\pi)\left[\prod_{n=2}^{N} p(z_n|z_{n-1}, A)\right]\prod_{m=1}^{N} p(x_m|z_m, \phi)$$

Matching slides to video frames

$$p(X,Z|\theta) = p(z_1|\pi)\left[\prod_{n=2}^{N} p(z_n|z_{n-1}, A)\right]\prod_{m=1}^{N} p(x_m|z_m, \phi)$$

From image matching

Matching slides to video frames

$$p(X,Z|\theta) = p(z_1|\pi)\left[\prod_{n=2}^{N} p(z_n|z_{n-1}, A)\right]\prod_{m=1}^{N} p(x_m|z_m, \phi)$$

$p(z_n|z_{n-1}, A) = f(z_n - z_{n-1})$ encodes slide jump statistics.

We assume that only the jump matters. IE, going from slide 6 to 8 has the same chance of going from 10 to 12.

$p(z_1|\pi)\left[\prod_{n=2}^{N} p(z_n|z_{n-1}, A)\right]$ says how likely a sequence is, without looking at the images.

Aligning speech to slides

Why bother?

Mistakes in speech transcripts can be corrected.
Speech transcripts are noisy and to poorly on jargon
But jargon words often appear on slides.

We can highlight or auto-laser-point what the speaker is pointing to

We can improve close-captioning.

Aligning speech to slides

A reasonable model for some speakers is that they say some approximation of their bullet points, with some extra stuff before and after.

Automated speech recognizers try to produce results that are plausible on a phoneme level.

If a slide word is used, its phoneme sequence will likely be approximated in the phonemes in the speech transcript.

We can calibrate the phoneme "confusion matrix."

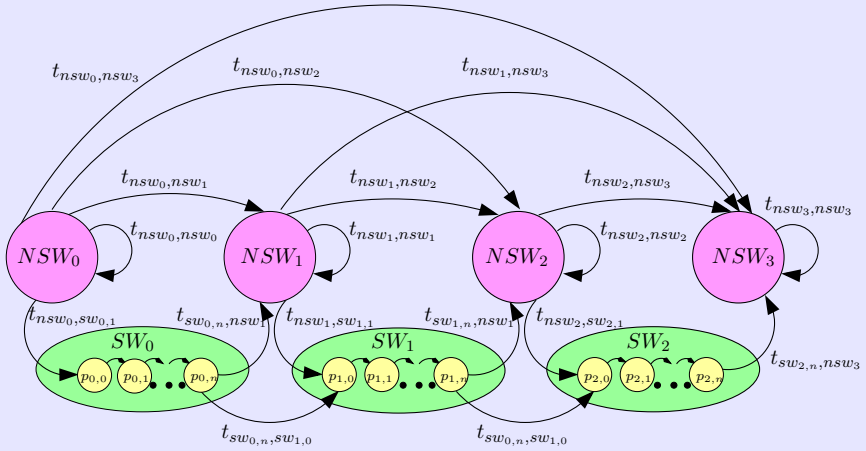
Aligning speech to slides

We assume that going backwards does not happen.

We have an HMM state for each slide word

We also have an HMM state for emitting phonemes between slide words.

SEQUENTIAL ALIGNMENT MODEL



Aligned speech for correction

Speaker says : maliciousness

ASR produces: my dishes nests *m ay d ih sh ah z n eh s t*

with slide word phoneme sequence

Slide word : maliciousness *m ah l ih sh ah s n ah s*

If the same mistake is made later, where the word is not on the slide, we can propagate the correction.

State space models

- Making some use of Murphy, chapter 18, as well as Bishop 13
- State space models are Like HMMs except that the states are continuous variables
- Example
  - A kicked soccer ball traveling under the influence of gravity (ignore air friction for now---this soccer match is on the moon).
  - One representation of state is position, velocity, and acceleration (all these are continuous variables)

State space models

- Notation
  - At a time, t, the state vector is $\mathbf{z}_t$
  - At each (discrete) time point, we make measurements $\mathbf{y}_t$
  - The system could also be influenced by a time varying control signal, which we will ignore in this course.

$$\mathbf{z}_t = g(\mathbf{z}_{t-1}, \varepsilon_t)$$
$$\mathbf{y}_t = g(\mathbf{z}_t, \delta_t)$$

where ε_t is the "system noise" and

δ_t is the "observation noise"

Linear dynamical systems (LDS)

- Special case of state-space models where the transition function g() is linear, and all random processes are Gaussian
  - Also known as linear-Gaussian SSM (LG-SSM)

$\mathbf{z}_t = A_t \mathbf{z}_{t-1} + \varepsilon_t$ where $\varepsilon_t \sim \mathcal{N}(0, Q_t)$ and A is a transition matrix

$\mathbf{y}_t = C_t \mathbf{z}_t + \delta_t$ where $\delta_t \sim \mathcal{N}(0, R_t)$

where ε_t is the "system noise" and δ_t is the "observation noise"

Often we assume that the parameters do not change over time.

This is known as a stationary model. Here,

$A_t = A \quad C_t = C \quad Q_t = Q \quad R_t = R$

Linear dynamical systems (LDS)

- For example, let $\mathbf{z}$ be the position in 2D of a hockey puck on the ice, moving with constant velocity.
- What is A?

Linear dynamical systems (LDS)

- For example, let **z** be the position in 2D of a hockey puck on the ice, moving with constant velocity.
- What is A?

Ignoring noise, we have

$$\mathbf{X}_t = \mathbf{X}_{t-1} + \mathbf{V}_{t-1} \bullet \Delta t$$

$$\mathbf{V}_t = \mathbf{V}_{t-1} + \mathbf{V}_{t-1} \bullet \Delta t$$

Linear dynamical systems (LDS)

- For example, let **z** be the position in 2D of a hockey puck on the ice, moving with constant velocity.
- What is A?

Ignoring noise, we have

$$\mathbf{X}_t = \mathbf{X}_{t-1} + \mathbf{V}_{t-1} \bullet \Delta t$$

$$\mathbf{V}_t = \mathbf{V}_{t-1} + \mathbf{V}_{t-1} \bullet \Delta t$$

$$A = \begin{pmatrix} 1 & & \Delta t & \\ & 1 & & \Delta t \\ & & 1 & \\ & & & 1 \end{pmatrix}$$

Linear dynamical systems (LDS)

- For example, let **z** be the position in 2D of a hockey puck on the ice, moving with constant velocity.
- If the ice is rough, then **z** might be buffeted about. Then our system noise component becomes relevant

$$\mathbf{X}_t = \mathbf{X}_{t-1} + \mathbf{V}_{t-1} \bullet \Delta t \; + \; \varepsilon_t$$

$$\mathbf{V}_t = \mathbf{V}_{t-1} + \mathbf{V}_{t-1} \bullet \Delta t \; + \; \varepsilon_t$$

- (sometimes called random acceleration model)

Linear dynamical systems (LDS)

- Finally, the observations are a linear function off the state variable, with added Gaussian noise. But perhaps we only measure position. Then

$$\mathbf{y}_t = C\,\mathbf{z}_t + \delta_t$$

$$\text{Where} \quad C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

LDS computational problems

Given data, what is the LDS (**learning**).

Given an LDS, what is the **distribution over the state** variables. Also, **how likely** are the observations, given the model.

Unlike a general HMM, in the Gaussian posterior for LDS, means the most likely **state sequence** for the data is simply the most likely states computed from the previous.

LDS computational problems

Learning the LDS can be accomplished by EM in analogy with HMM, as well as other means.

Traditionally, given an LDS, the **distribution over the state** variables is computed using the Kalman filter and the Kalman smoother (like alpha and beta respectively).

Conventionally, the Kalman filter is analogous to computing the rescaled alphas

$$\hat{\alpha}(z_n) = p(z_n | x_1, ..., x_n) = \frac{\alpha(z_n)}{p(x_1, ..., x_n)}$$

and the Kalman smoother is analogous to computing the products

$$\gamma(z_n) = \hat{\alpha}(z_n)\hat{\beta}(z_n)$$

Recall that $\hat{\beta}(z_n) = \frac{p(x_{n+1}, ..., x_N | z_n)}{p(x_{n+1}, ..., x_N | x_1, ..., x_n)}$

The complete log likelihood

In analogy with HMM, we have

$$p(X, Z | \theta) = p(\mathbf{z}_1)\left[\prod_{n=2}^{N} p(\mathbf{z}_n | \mathbf{z}_{n-1})\right]\prod_{m=1}^{N} p(\mathbf{x}_m | \mathbf{z}_m)$$

$$= \mathcal{N}(\mathbf{z}_1 | \boldsymbol{\mu}_0, \mathbf{V}_0)\left[\prod_{n=2}^{N} \mathcal{N}(\mathbf{z}_n | \mathbf{A}\mathbf{z}_{n-1}, \Gamma)\right]\prod_{m=1}^{N} \mathcal{N}(\mathbf{x}_m | \mathbf{C}\mathbf{z}_m, \Sigma)$$

Manipulating Gaussians

Recall that for multivariate Gaussians, if we partition the variables into two sets, **x** and **y**, then:

$$p(\mathbf{x}|\mathbf{y}) \sim \mathcal{N}(\ )$$

$$p(\mathbf{x}) = \int_y p(\mathbf{x},\mathbf{y}) \sim \mathcal{N}(\ )$$

Manipulating Gaussians

In addition, if

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \Lambda^{-1})$$
$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\mathbf{x}+\mathbf{b}, \mathbf{L}^{-1})$$

then

$$p(\mathbf{x},\mathbf{y}) = p(\mathbf{x})p(\mathbf{y}|\mathbf{x}) \sim \mathcal{N}(\ )$$

Manipulating Gaussians

More specifically (from Bishop p. 91), if

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \Lambda^{-1})$$
$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\mathbf{x}+\mathbf{b}, \mathbf{L}^{-1})$$

then

$$p\left(\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix}\right) = \mathcal{N}\left(\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix} \middle| \begin{matrix}\boldsymbol{\mu}\\ \mathbf{A}\boldsymbol{\mu}+\mathbf{b}\end{matrix}, \mathbf{R}^{-1}\right)$$

where

$$\mathbf{R}^{-1} = \begin{pmatrix}\mathbf{A}+\mathbf{A}^T\mathbf{L}\mathbf{A} & -\mathbf{A}^T\mathbf{L}\\ -\mathbf{L}\mathbf{A} & \mathbf{L}\end{pmatrix}$$

Kalman filtering/smoothing

Using these rules, you can easily convince yourself that the joint probability of all the LDS variables is one big Gaussian.

The filtering/smoothing messages (alpha/beta) enable fast computation of the marginals.

We will do the alphas briefly

Kalman filtering

In analogy with the rescaled version of the alpha/beta algorithm:

$$c_n \hat{\alpha}(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n) \int \hat{\alpha}(\mathbf{z}_{n-1}) p(\mathbf{z}_n|\mathbf{z}_{n-1}) d\mathbf{z}_{n-1}$$

Denote $\hat{\alpha}(\mathbf{z}_n) = \mathcal{N}(\mathbf{z}_n|\boldsymbol{\mu}_n, \mathbf{V}_n)$ to get

$$\begin{aligned} c_n \mathcal{N}(\mathbf{z}_n|\boldsymbol{\mu}_n, \mathbf{V}_n) &= \mathcal{N}(\mathbf{x}_n|\mathbf{C}\mathbf{z}_n, \Sigma) \int \mathcal{N}(\mathbf{z}_n|\mathbf{A}\mathbf{z}_{n-1}) \mathcal{N}(\mathbf{z}_{n-1}|\boldsymbol{\mu}_{n-1}, \mathbf{V}_{n-1}) d\mathbf{z}_{n-1} \\ &= \mathcal{N}(\mathbf{x}_n|\mathbf{C}\mathbf{z}_n, \Sigma) \mathcal{N}(\mathbf{z}_n|\mathbf{A}\boldsymbol{\mu}_{n-1}, \mathbf{P}_{n-1}) \end{aligned}$$

where $\mathbf{P}_{n-1} = \mathbf{A}\mathbf{V}_{n-1}\mathbf{A}^T + \Gamma$, using rules about manipulating Gaussians.

Kalman filtering

Applying Gaussian manipulations and Matrix inversion formulas (see Bishop page 696):

$$\begin{aligned} \boldsymbol{\mu}_n &= \mathbf{A}\boldsymbol{\mu}_{n-1} + \mathbf{K}_n(\mathbf{x}_n - \mathbf{C}\mathbf{A}\boldsymbol{\mu}_{n-1}) \\ \mathbf{V}_n &= (\mathbf{I} - \mathbf{K}_n\mathbf{C})\mathbf{P}_{n-1} \\ c_n &= \mathcal{N}(x_n|\mathbf{C}\mathbf{A}\boldsymbol{\mu}_{n-1}, \mathbf{C}\mathbf{P}_{n-1}\mathbf{C}^T + \Sigma) \end{aligned}$$

where $\mathbf{K}_n = \mathbf{P}_{n-1}\mathbf{C}^T(\mathbf{C}\mathbf{P}_{n-1}\mathbf{C}^T + \Sigma)^{-1}$

(This is the Kalman gain matrix)

Kalman filtering

For completeness:

$$\begin{aligned} \boldsymbol{\mu}_1 &= \mathbf{A}\boldsymbol{\mu}_0 + \mathbf{K}_1(\mathbf{x}_1 - \mathbf{C}\boldsymbol{\mu}_0) \\ \mathbf{V}_1 &= (\mathbf{I} - \mathbf{K}_1\mathbf{C})\mathbf{V}_0 \\ c_1 &= \mathcal{N}(x_1|\mathbf{C}\boldsymbol{\mu}_0, \mathbf{C}\mathbf{V}_0\mathbf{C}^T + \Sigma) \end{aligned}$$

where $\mathbf{K}_1 = \mathbf{V}_0\mathbf{C}^T(\mathbf{C}\mathbf{V}_0\mathbf{C}^T + \Sigma)^{-1}$

Kalman filtering

Despite the tedious details, the result is somewhat intuitive. Consider the update of the mean:

$$\boldsymbol{\mu}_n = \underbrace{\mathbf{A}\boldsymbol{\mu}_{n-1}}_{\substack{\text{Believing the model,} \\ \text{ignoring the observation}}} + \mathbf{K}_n\left(\mathbf{x}_n - \underbrace{\mathbf{C}\mathbf{A}\boldsymbol{\mu}_{n-1}}_{\substack{\text{Where we think we} \\ \text{should see } \mathbf{x}_n}}\right)$$

The new mean is the propagated previous one, with a correction for the new evidence.

The Kalman gain matrix is a factor of the relation between state variables and observations (C), and the variances.

Kalman filtering

Despite the tedious details, the result is somewhat intuitive. Consider the update of the mean:

$$\boldsymbol{\mu}_n = \underbrace{\mathbf{A}\boldsymbol{\mu}_{n-1}}_{\substack{\text{Believing the model,} \\ \text{ignoring the observation}}} + \mathbf{K}_n \left(\mathbf{x}_n - \underbrace{\mathbf{C}\mathbf{A}\boldsymbol{\mu}_{n-1}}_{\substack{\text{Where we think we} \\ \text{should see } \mathbf{x}_n}} \right)$$

The new mean is the propagated previous one, with a correction for the new evidence.

The Kalman gain matrix is a factor of the relation between state variables and observations (C), and the variances.

Kalman filtering/smoothing

Kalman filtering by itself makes sense if you are tracking an object on-line and in real time.

$$\boldsymbol{\mu}_n = \underbrace{\mathbf{A}\boldsymbol{\mu}_{n-1}}_{\substack{\text{Believing the model,} \\ \text{ignoring the observation}}} + \mathbf{K}_n \left(\mathbf{x}_n - \underbrace{\mathbf{C}\mathbf{A}\boldsymbol{\mu}_{n-1}}_{\substack{\text{Where we think we} \\ \text{should see } \mathbf{x}_n}} \right)$$

However, the future observations can improve the estimates made by only considering the past.

The second pass computes the posterior as function of both.

This is the “smoother” which is analogous to the beta pass.

For details see the rest of Bishop 13.3.1.