Kalman Filter
(Ch. 15)
HMMs and Matrices

We can represent this Bayes net with matrices:

$P(x_{t+1}|x_t) = T = \begin{bmatrix} 0.6 & 0.4 \\ 0.9 & 0.1 \end{bmatrix}$

The evidence matrices are more complicated:

$P(e_t|x_t) = E_t = \begin{bmatrix} 0.3 & 0 \\ 0 & 0.8 \end{bmatrix}$

$P(\neg e_t|x_t) = E_t = \begin{bmatrix} 0.7 & 0 \\ 0 & 0.2 \end{bmatrix}$

both just called $E_t$, which depends on whether $e_t$ or $\neg e_t$
HMMs and Matrices

This allows us to represent our filtering eq:

\[ f(t) = \alpha P(e_t|x_t) \sum_{x_{t-1}} \left( P(x_t|x_{t-1}) f(t-1) \right) \]

... with matrices:

\[ F_{t+1} = \alpha E_{t+1} T^T F_t \]

... why?

1. Gets rid of sum (matrix mult. does this)
2. More easily to “reverse” messages

\[ F_t = \alpha' (T^T)^{-1} E_{t+1}^{-1} F_{t+1} \]
HMMs and Matrices

This actually gives rise to a smoothing alg. with constant memory (we did with linear):

Smooth (constant mem):
-1. Compute filtering from 1 to t
-2. Loop: i=t to 1
   -2.1. Smooth $X_i$ (have f(i) and backwards(i))
   -2.2. Compute backwards(i-1) in normal way
   -2.3. Compute f(i-1) using previous slide
HMMs and Matrices

Smoothing actually has issues with “online” algorithms, where you need results mid-alg.

The stock market is an example as you have historical info and need choose trades today.

But tomorrow we will have the info for today as well... need alg to not compute “from scratch”
HMMs and Matrices

With smoothing, the “forwards” message is fine, since we start it at \( f(0) \) and go to \( f(t) \)

We can then compute the “next day” easily as \( f(t+1) \) is based off \( f(t) \) in our equations

This is not the case for the “backwards” message, as this starts \( b(t) \) to get \( b(t-1) \)

\[
b(k) = \sum_{x_{k+1}} P(e_{k+1}|x_{k+1}) b(k+1) P(x_{k+1}|x_k)
\]

As matrix: \[
B_k = T E_{k+1} B_{k+1}
\]
HMMs and Matrices

The naive way would be to restart the “backwards” message from scratch.

I will switch to the book’s notation of $B_{1:t}$ as the backward message that uses $e_1$ to $e_t$ (slightly different as $B_{k:t}$ uses $e_{k+1}$ to $e_t$).

Thus we would want some way to compute $B_{j:t+1}$ from $B_{k:t}$ without doing it from scratch.
HMMs and Matrices

So we have:

\[ B_{1:1} = TE_1 \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad B_{1:2} = TE_1 TE_2 \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad B_{2:2} = TE_2 \begin{bmatrix} 1 \\ 1 \end{bmatrix} \]

In general: \[ B_{j:k} = \left( \prod_{i=j}^{k} TE_i \right) \begin{bmatrix} 1 \\ 1 \end{bmatrix} \]

This \([1,1]^T\) matrix is in the way, so let’s store:

\[ \hat{B}_{j:k} = \left( \prod_{i=j}^{k} TE_i \right) \]

... then: \[ \hat{B}_{2:2} = E_1^{-1} T^{-1} \hat{B}_{1:1} TE_2 \]

... or generally if \( j > k \): \[ \hat{B}_{j:t+1} = \left( \prod_{i=j-1}^{k} E_i^{-1} T^{-1} \right) \hat{B}_{k:t} TE_{t+1} \]
HMMs in Practice

One common place this filtering is used is in position tracking (radar)

The book gives a nice example that is more complex than we have done:

A robot is dropped in a maze (it has a map), but it does not know where...
... additionally, the sensors on the robot do not work well... where is the robot?
HMMs in Practice

(a) Posterior distribution over robot location after $E_1 = N$ $SW$

(b) Posterior distribution over robot location after $E_1 = N$, $SW$, $E_2 = N$ $S$
HMMs in Practice

20% error per direction
\((1 - 0.8^4) = 59\% \) at least one error

Average expected distance (Manhattan) from real

Perfect sensors
Kalman Filters

How does all of this relate to Kalman filters?

This is just “filtering” (in HMM/Bayes net), except with continuous variables

This heavily use the Gaussian distribution:

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

thank you alpha!
Kalman Filters

Why the preferential treatment for Gaussians?

A key benefit is that when you do our normal operations (add and multiply), if you start with a Gaussian as input, you get Gaussian out.

In fact, if you input a linear Gaussian input, you get a Gaussian out: (linear=matrix mult)

More on this later, let’s start simple
Kalman Filters

As an example, let’s say you are playing Frisbee at night:

1. Can’t see exactly where friend is

2. Friend will move slightly to catch Frisbee
Kalman Filters

Unfortunately... the math is a bit ugly (as Gaussians are a bit complex)

Here we assume:

The probability of the next position given the current position is:

\[
P(x_{t+1} | x_t) = \alpha e^{-\frac{1}{2} \frac{(x_{t+1} - x_t)^2}{\sigma_x^2}} = N(x_t, \sigma_x^2)(x_{t+1})
\]

The probability of the error given the current position is:

\[
P(e_t | x_t) = \alpha e^{-\frac{1}{2} \frac{(e_t - x_t)^2}{\sigma_e^2}} = N(x_t, \sigma_e^2)(e_t)
\]

\(\sigma_x^2\) is how much the friend moves.

The variance is "can’t see well".

How do we compute the filtering “forward” messages (in our efficient non-recursive way)?
Kalman Filters

Unfortunately... the math is a bit ugly (as Gaussians are a bit complex)

Here we assume:

\[ P(x_{t+1} | x_t) = \alpha e^{-\frac{1}{2} \frac{(x_{t+1} - x_t)^2}{\sigma_x^2}} = N(x_t, \sigma_x^2)(x_{t+1}) \]

\[ P(e_t | x_t) = \alpha e^{-\frac{1}{2} \frac{(e_t - x_t)^2}{\sigma_e^2}} = N(x_t, \sigma_e^2)(e_t) \]

erm... let’s change variable names

How do we compute the filtering “forward” messages (in our efficient non-recursive way)?
Kalman Filters

Unfortunately... the math is a bit ugly (as Gaussians are a bit complex)

Here we assume:

\[ P(x_{t+1} | x_t) = \alpha \exp \left( \frac{-1}{2} \frac{(x_{t+1} - x_t)^2}{\sigma^2_x} \right) = N(x_t, \sigma^2_x)(x_{t+1}) \]

\[ P(z_t | x_t) = \alpha \exp \left( \frac{-1}{2} \frac{(z_t - x_t)^2}{\sigma^2_z} \right) = N(x_t, \sigma^2_z)(z_t) \]

\( \sigma^2_x \) is how much friend moves

\( \sigma^2_z \) variance is “can’t see well”

How do we compute the filtering “forward” messages (in our efficient non-recursive way)?
Kalman Filters

The same? Sorta... but we have to integrate

\[ F_1 = P(z_1 | x_1) \sum_{x_0} P(x_1 | x_0) P(x_0) \]

\[ F_1 = P(z_1 | x_1) \int_{-\infty}^{\infty} P(x_1 | x_0) P(x_0) dx_0 \]
Kalman Filters

\[ F_1 = P(z_1|x_1) \int_{-\infty}^{\infty} P(x_1|x_0)P(x_0)dx_0 \]

\[ = N(x_1, \sigma_z^2)(z_1) \int_{-\infty}^{\infty} N(x_0, \sigma_x^2)(x_1) \cdot N(0, 1)(x_0)dx_0 \]

\[ = N(x_1, \sigma_z^2)(z_1) \int_{-\infty}^{\infty} \alpha e^{-\frac{1}{2} \frac{(x_1-x_0)^2}{\sigma_x^2}} \cdot \alpha' e^{-\frac{1}{2} x_0^2} dx_0 \]

\[ = \alpha N(x_1, \sigma_z^2)(z_1) \int_{-\infty}^{\infty} e^{-\frac{1}{2} \frac{(x_1-x_0)^2+\sigma_x^2 x_0^2}{\sigma_x^2}} dx_0 \]

\[ = \alpha N(x_1, \sigma_z^2)(z_1) \int_{-\infty}^{\infty} e^{-\frac{1}{2} \left( \frac{(1+\sigma_x^2)}{\sigma_x^2} x_0^2 + \frac{(-2x_1)}{\sigma_x^2} x_0 + \frac{x_1^2}{\sigma_x^2} \right)} dx_0 \]
Kalman Filters

But wait! There’s hope!
We can use a little fact that:

\[ ax^2 + bx + c = a\left(x - \frac{-b}{2a}\right)^2 + \left(c - \frac{b^2}{4a}\right) \]

does not contain \( x \)

\[
F_1 = \hat{a}N(x_1, \sigma^2_x)(z_1) \int_{-\infty}^{\infty} e^{-\frac{1}{2}} \left( \frac{1+\sigma^2_x}{\sigma^2_x} x^2_0 + \frac{-2x_1}{\sigma^2_x} x_0 + \frac{x^2_0}{\sigma^2_x} \right) e^{-\frac{1}{2} \left(a\left(x_0 - \frac{-b}{2a}\right)^2 + \left(c - \frac{b^2}{4a}\right) \right)} dx_0
\]

\[
= \hat{a}N(x_1, \sigma^2_x)(z_1) \int_{-\infty}^{\infty} e^{-\frac{1}{2} \left(a\left(x_0 - \frac{-b}{2a}\right)^2 + \left(c - \frac{b^2}{4a}\right) \right)} dx_0
\]

\[
= \hat{a}N(x_1, \sigma^2_x)(z_1) e^{-\frac{1}{2} \left(c - \frac{b^2}{4a}\right)} \int_{-\infty}^{\infty} e^{-\frac{1}{2} \left(a\left(x_0 - \frac{-b}{2a}\right)^2 \right)} dx_0
\]
Kalman Filters

But wait! There’s hope!

We can use a little fact that:

\[ ax^2 + bx + c = a(x - \frac{-b}{2a})^2 + \left( c - \frac{b^2}{4a} \right) \]

does not contain \( x \)

\[
F_1 = \hat{\alpha} N(x_1, \sigma_z^2)(z_1) \int_{-\infty}^{\infty} e^{-\frac{1}{2} \left( \frac{(1+\sigma_x^2)}{\sigma_x^2} x_0^2 + \frac{(-2x_1)}{\sigma_x^2} x_0 + \frac{x_0^2}{\sigma_x^2} \right)} dx_0 \\
= \hat{\alpha} N(x_1, \sigma_z^2)(z_1) \int_{-\infty}^{\infty} e^{-\frac{1}{2} (a(x_0-\frac{-b}{2a})^2 + (c-\frac{b^2}{4a}))} dx_0 \\
= \hat{\alpha} N(x_1, \sigma_z^2)(z_1) \int_{-\infty}^{\infty} e^{-\frac{1}{2} (a(x_0-\frac{-b}{2a})^2)} e^{-\frac{1}{2} (c-\frac{b^2}{4a})} dx_0 \\
= \hat{\alpha} N(x_1, \sigma_z^2)(z_1)e^{-\frac{1}{2} (c-\frac{b^2}{4a})} \int_{-\infty}^{\infty} e^{-\frac{1}{2} (a(x_0-\frac{-b}{2a})^2)} dx_0 \\

This is just:

\[ N\left(\frac{-b}{2a}, \frac{1}{a}\right) \]
Kalman Filters

\[ a = \frac{1 + \sigma_x^2}{\sigma_x^2}, \quad b = \frac{-2x_1}{\sigma_x^2}, \quad c = \frac{x_1^2}{\sigma_x^2} \]

\[ F_1 = \hat{\alpha} N(x_1, \sigma_z^2)(z_1)e^{-\frac{1}{2}(c-b^2/4a)} \int_{-\infty}^{\infty} e^{-\frac{1}{2}(a(x_0-b/2a)^2) \int_{-\infty}^{\infty} N\left(\frac{-b}{2a}, \frac{1}{a}\right)(x_0)dx_0} \]

\[ = \hat{\alpha} N(x_1, \sigma_z^2)(z_1)e^{-\frac{1}{2}(c-b^2/4a)} \cdot 1 \]

\[ = \hat{\alpha} N(x_1, \sigma_z^2)(z_1)e^{-\frac{1}{2}\left(\frac{x_1^2}{\sigma_x^2} - \frac{(-2x_1)^2}{\sigma_x^2}\right)} \cdot 1 \]

\[ = \hat{\alpha} N(x_1, \sigma_z^2)(z_1)e^{-\frac{1}{2}\left(\frac{x_1^2}{\sigma_x^2} - \frac{4x_1^2/\sigma_x^2}{4(1+\sigma_x^2)/\sigma_x^2}\right)} \cdot 1 \]

\[ = \hat{\alpha} N(x_1, \sigma_z^2)(z_1)e^{-\frac{1}{2}\left(\frac{x_1^2(1+\sigma_x^2)}{\sigma_x^2(1+\sigma_x^2)} - \frac{x_1^2}{(1+\sigma_x^2)\sigma_x^2}\right)} \cdot 1 \]

\[ = \hat{\alpha} N(x_1, \sigma_z^2)(z_1)e^{-\frac{1}{2}\left(\frac{x_1^2+\sigma_x^2x_1^2-x_1^2}{(1+\sigma_x^2)\sigma_x^2}\right)} \cdot 1 \]

\[ = \hat{\alpha} N(x_1, \sigma_z^2)(z_1)e^{-\frac{1}{2}\left(\frac{x_1^2}{1+\sigma_x^2}\right)} \cdot 1 \]
Kalman Filters

\[ F_1 = \hat{\alpha} N(x_1, \sigma_z^2)(z_1)e^{-\frac{1}{2}\left(\frac{x_1^2}{1+\sigma_x^2}\right)} \]

\[ = \hat{\alpha}e^{-\frac{1}{2}\left(\frac{(z_1-x_1)^2}{\sigma_z^2}\right)}e^{-\frac{1}{2}\left(\frac{x_1^2}{1+\sigma_x^2}\right)} \]

\[ = \hat{\alpha}e^{-\frac{1}{2}\left(\frac{(z_1-x_1)^2}{\sigma_z^2} + \frac{x_1^2}{1+\sigma_x^2}\right)} \]

\[ = \hat{\alpha}e^{-\frac{1}{2}\left(\frac{(z_1-x_1)^2}{\sigma_z^2(1+\sigma_x^2)} + \frac{x_1^2}{\sigma_z^2(1+\sigma_x^2)}\right)} \]

\[ = \hat{\alpha}e^{-\frac{1}{2}\left(\frac{1+\sigma_x^2+\sigma_z^2}{\sigma_z^2(1+\sigma_x^2)}\right)x_1^2 + (-2z_1/\sigma_z^2)x_1 + z_1^2/\sigma_z^2} \]

\[ = \hat{\alpha}e^{-\frac{1}{2}a(x_1 - \frac{-b}{2a})^2 + (c - \frac{b^2}{4a})} \]

\[ = \hat{\alpha}e^{-\frac{1}{2}\left(c - \frac{b^2}{4a}\right)}e^{-\frac{1}{2}a(x_1 - \frac{-b}{2a})^2} \]

\[ = \hat{\alpha}' e^{-\frac{1}{2}a(x_1 - \frac{-b}{2a})^2} \]

Gross after plugging in a, b, c (see book)
Kalman: Frisbee in the Dark

\[ a = \frac{1 + \sigma_x^2 + \sigma_z^2}{\sigma_z^2(1 + \sigma_x^2)} , \quad b = -2z_1/\sigma_z^2 , \quad c = z_1^2/\sigma_z^2 \]

Initially your friend is \(N(0,1)\)
Initially your friend is $N(0,1)$

Throw not perfect, so friend has to move $N(0,1.5)$ (i.e. move from black to red)

$$a = \frac{1 + \sigma^2_x + \sigma^2_z}{\sigma_z^2(1 + \sigma^2_x)}, \quad b = -2z_1/\sigma_z^2, \quad c = z_1^2/\sigma_z^2$$
Kalman: Frisbee in the Dark

\[ a = \frac{1 + \sigma_x^2 + \sigma_z^2}{\sigma_z^2(1 + \sigma_x^2)} \], \quad \sigma^2 = 1

\[ b = -\frac{2z_1}{\sigma_z^2} \], \quad \sigma^2 = 0.2

\[ c = \frac{z_1}{\sigma_z^2} \]

But you can’t actually see your friend too clearly in the dark

You thought you saw them at 0.75 (\sigma^2=0.2)
Kalman: Frisbee in the Dark

\[ a = \frac{1+\sigma_x^2+\sigma_z^2}{\sigma_z^2(1+\sigma_x^2)}, \quad b = -2z_1/\sigma_z^2, \quad c = z_1^2/\sigma_z^2 \]

Where is your friend actually?

\[ F_1 = \hat{\alpha}' e^{-\frac{1}{2} a (x_1 - \frac{b}{2a})^2} \]
Kalman: Frisbee in the Dark

Where is your friend actually?

\[ a = \frac{1 + 1.5 + 0.2}{0.2(1 + 1.5)} = 5.4, \quad b = -2(0.75)/0.2 = -7.5, \quad c = (0.75)^2/0.2 = 2.8125 \]

\[ F_1 = \hat{\alpha} e^{\frac{1}{2} a(x_1 - \frac{-b}{2a})^2} = N\left(\frac{-b}{2a}, \frac{1}{a}\right) = N(0.694, 0.185) \]

Probably 0.05 “left” of where you “saw” them
Kalman Filters

So the filtered “forward” message for throw 1 is: \( N(0.694, 0.185) \)

To find the filtered “forward” message for throw 2, use \( N(0.694, 0.185) \) instead of \( N(0, 1) \) (this does change the equations as you need to involve a \( \mu \) for the old \( N(0, 1) \))

The book gives you the full messy equations:

\[
\mu_{t+1} = \frac{(\sigma_t^2 + \sigma_x^2)z_{t+1} + \sigma_z^2 \mu_t}{\sigma_t^2 + \sigma_x^2 + \sigma_z^2}
\]

\[
\sigma_{t+1}^2 = \frac{(\sigma_t^2 + \sigma_x^2)\sigma_z^2}{\sigma_t^2 + \sigma_x^2 + \sigma_z^2}
\]
So the filtered “forward” message for throw 1 is: $N(0.694, 0.185)$

To find the filtered “forward” message for throw 2, use $N(0.694, 0.185)$ instead of $N(0, 1)$ (this does change the equations as you need to involve a $\mu$ for the old $N(0, 1)$)

The book gives you the full messy equations:

$$\mu_{t+1} = \frac{(\sigma_t^2 + \sigma_x^2)z_{t+1} + \sigma_z^2 \mu_t}{\sigma_t^2 + \sigma_x^2 + \sigma_z^2}$$

$$\sigma_{t+1}^2 = \frac{(\sigma_t^2 + \sigma_x^2)\sigma_z^2}{\sigma_t^2 + \sigma_x^2 + \sigma_z^2}$$
Kalman Filters

So the filtered “forward” message for throw 1 is: \( N(0.694, 0.185) \)

To find the filtered “forward” message for throw 2, use \( N(0.694, 0.185) \) instead of \( N(0, 1) \) (this does change the equations as you need to involve a \( \mu \) for the old \( N(0, 1) \))

The book gives you the full messy equations:

\[
\mu_{t+1} = \frac{\left( \sigma_z^2 + \sigma_x^2 \right) z_{t+1} + \sigma_z^2 \mu_t}{\sigma_t^2 + \sigma_x^2 + \sigma_z^2}
\]

\[
\sigma_{t+1}^2 = \frac{(\sigma_t^2 + \sigma_x^2) \sigma_z^2}{\sigma_t^2 + \sigma_x^2 + \sigma_z^2}
\]
Kalman Filters

The full Kalman filter is done with multiple numbers (matrices)

Here a Gaussian is: \(N(\mu, \Sigma) = a e^{-\frac{1}{2}((x-\mu)^T \Sigma^{-1} (x-\mu))}\)

Bayes net is: (F and H are “linear” matrix)

\[
P(x_{t+1}|x_t) = N(Fx_t, \Sigma_x)(x_{t+1})
\]

\[
P(z_t|x_t) = N(Hx_t, \Sigma_z)(z_t)
\]

Then filter update is:

\[
\Sigma_{t+1} = (I - K_{t+1}H)(F\Sigma_t F^T + \Sigma_x)
\]

\[
\mu_{t+1} = F\mu_t + K_{t+1}(z_{t+1} - HF\mu_t)
\]

\[
K_{t+1} = (F\Sigma_t F^T + \Sigma_x)H^T(H(F\Sigma_t F^T + \Sigma_x)H^T + \Sigma_z)^{-1}
\]

\(\text{covariance matrix}\)

\(\text{identity matrix}\)

\(\text{yikes...}\)
Kalman Filters

Often we use $\begin{bmatrix} x \\ v_x \end{bmatrix}$ for a 1-dimensional problem with both position and velocity.

To update $x_{t+1}$, we would want: $x_{t+1} = x_t + v_x$

In matrix form:

$$P(x_{t+1} | x_t) = N(F x_t, \Sigma_x)(x_{t+1})$$

$$F = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

so:

$$Fx_t = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ v_x \end{bmatrix} = \begin{bmatrix} x + v_x \\ v_x \end{bmatrix}$$

So our “mean” at $t+1$ is $\begin{bmatrix} x \end{bmatrix}$.
Kalman Filters

Here’s a Pokemon example (not technical)

https://www.youtube.com/watch?v=bm3cwEP2nUo
Kalman Filters

Downsides?

In order to get “simple” equations, we are limited to the linear Gaussian assumption.

However, there are some cases when this assumption does not work very well at all.
Kalman Filters

Consider the example of balancing a pencil on your finger

How far to the left/right will the pencil fall?

Below is not a good representation:
Instead it should probably look more like:

... where you are deciding between two options, but you are not sure which one

The Kalman filter can handle this as well (just keep 2 sets of equations and use more likely)
Kalman Filters

Unfortunately if you repeat this “pencil balance” on the new spot... you would need 4 sets of equations

3rd attempt: 8 equations
4th attempt: 16 equations

... this exponential amount of work/memory cannot be done for a large HMM