Localization

Bayes Filter

Terminology:

- Filters produce the estimated state of the system.
- The system is what we are trying to model or filter.
- The state is its current configuration or value.
- Prior or Prediction: state after performing the prediction
- Posterior: state after performing the measurement => prior and posterior are relative to the measurement step

One cycle of prediction and updating with a measurement is called the state or system evolution, which is short for time evolution. Another term is system propagation. It refers to how the state of the system changes over time.

SciPy provides a convolution routine convolve() in the ndimage.filters module. We need to shift the pdf by offset before convolution; np.roll() does that. The move and predict algorithm can be implemented with one line: *convolve(np.roll(pdf, offset), kernel, mode='wrap')*

The problem of losing information during a prediction may make it seem as if our system would quickly devolve into having no knowledge. However, each prediction is followed by an update where we incorporate the measurement into the estimate. The update improves our knowledge. The output of the update step is fed into the next prediction. The prediction degrades our certainty. That is passed into another update, where certainty is again increased.

Mean, Variance and Standard Deviation

Mean

The expected value of a random variable is the average value it would have if we took an infinite number of samples of it and then averaged those samples together. We can formalize this by letting x_i be the i^{th} value of X, and p_i be the probability of its occurrence. This gives us

$$\mathbb{E}[X] = \sum_{i=1}^{n} p_i x_i$$

A trivial bit of algebra shows that if the probabilities are all equal, the expected value is the same as the mean:

$$\mathbb{E}[X] = \sum_{i=1}^{n} p_i x_i = \frac{1}{n}\sum_{i=1}^{n} x_i = \mu_x$$

If x is continuous we substitute the sum for an integral, like so

$$\mathbb{E}[X] = \int_a^b x f(x)\, dx$$

where $f(x)$ is the probability distribution function of x.

Variance and Standard Deviation

The equation for computing the variance is

$$VAR(X) = \mathbb{E}[(X - \mu)^2]$$

The variance is the *expected value* for how much the sample space X varies from the mean μ : ($X - \mu$).

The formula for the expected value is $\mathbb{E}[X] = \sum\limits_{i=1}^{n} p_i x_i$ so we can substitute that into the equation above to get

$$VAR(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$$

$$\sigma = \sqrt{VAR(X)} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$$

It is typical to use σ for the *standard deviation* and σ^2 for the *variance*.

Gaussians

Objective:

> We desire a unimodal, continuous way to represent probabilities that models how the real world works, and that is computationally efficient to calculate.

it is typical to shorten the name and talk about a *Gaussian* or *normal* — these are both typical shortcut names for the *Gaussian distribution* Let's explore how Gaussians work. A Gaussian is a *continuous probability distribution* that is completely described with two parameters, the mean (μ) and the variance (σ^2). It is defined as:

$$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\big[-\frac{(x-\mu)^2}{2\sigma^2}\big]$$

$\exp[x]$ is notation for e^x.

Don't be dissuaded by the equation if you haven't seen it before; you will not need to memorize or manipulate it. The computation of this function is stored in `stats.py` with the function `gaussian(x, mean, var, normed=True)`.

Shorn of the constants, you can see it is a simple exponential:

$$f(x) \propto e^{-x^2}$$

which has the familiar bell curve shape

The notation for a normal distribution for a random variable X is $X \sim\ \mathcal{N}(\mu,\sigma^2)$ where $\sim$ means *distributed according to*

An equivalent formation for a Gaussian is $\mathcal{N}(\mu,1/\tau)$ where μ is the *mean* and τ the *precision*. $1/\tau = \sigma^2$; it is the reciprocal of the variance. While we do not use this formulation in this book, it underscores that the variance is a measure of how precise our data is. A small variance yields large precision — our measurement is very precise. Conversely, a large variance yields low precision — our belief is spread out across a large area. You should become comfortable with thinking about Gaussians in these equivalent forms. In Bayesian terms Gaussians reflect our *belief* about a measurement, they express the *precision* of the measurement, and they express how much *variance* there is in the measurements. These are all different ways of stating the same fact.

The discrete Bayes filter works by multiplying and adding arbitrary probability distributions. The Kalman filter uses Gaussians instead of arbitrary distributions, but the rest of the algorithm remains the same. This means we will need to multiply and add Gaussians.

- **the sum of two independent Gaussians is another Gaussian** The sum of two Gaussians is given by

$$\begin{aligned} \mu &= \mu_1 + \mu_2 \\ \sigma^2 &= \sigma_1^2 + \sigma_2^2 \end{aligned}$$

- **the product of 2 Gaussians is not Gaussian, but proportional to a Gaussian**. There we can say that the result of multipying two Gaussian distributions is a Gaussian function (recall function in this context means that the property that the values sum to one is not guaranteed).

This is a key reason why Kalman filters are computationally feasible. Said another way, Kalman filters use Gaussians *because* they are computationally nice.

The product of two independent Gaussians is given by:

$$\begin{aligned} \mu &= \frac{\sigma_1^2 \mu_2 + \sigma_2^2 \mu_1}{\sigma_1^2 + \sigma_2^2} \\ \sigma^2 &= \frac{\sigma_1^2 \sigma_2^2}{\sigma_1^2 + \sigma_2^2} \end{aligned}$$

Multivariate Gaussians

Covariance is short for *correlated variances*

The equation for the covariance between X and Y is

$$COV(X,Y) = \sigma_{xy} = \mathbb{E}\big[(X-\mu_x)(Y-\mu_y)\big]$$

Where $\mathbb{E}[X]$ is the *expected value* of X, defined as

$$\mathbb{E}[X] = \begin{cases} \sum_{i=1}^n p_i x_i & \backslash\text{mbox}discrete \\ \int_{-\infty}^{\infty} f(x)\, x & \backslash\text{mbox}continuous \end{cases}$$

We assume each data point is equally likely, so the probability of each is $\frac{1}{N}$, giving

$$\mathbb{E}[X] = \frac{1}{N}\sum_{i=1}^n x_i$$

for the discrete case we will be considering.

Compare the covariance equation to the equation for the variance. As you can see they are very similar:

$$\begin{aligned} VAR(X) &= \sigma_x^2 = \mathbb{E}[(X - \mu)^2] \\ COV(X,Y) &= \sigma_{xy} = \mathbb{E}\big[(X-\mu_x)(Y-\mu_y)\big] \end{aligned}$$

We use a *covariance matrix* to denote covariances of a multivariate normal distribution, and it looks like this:

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_n^2 \end{bmatrix}$$

Recall the equation for the normal distribution:

$$f(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\Big[-\frac{1}{2}(x-\mu)^2/\sigma^2\Big]$$

Here is the multivariate normal distribution in n dimensions.

$$f(\mathbf{x},\, \mu,\, \Sigma) = \frac{1}{\sqrt{(2\pi)^n|\Sigma|}}\, \exp\Big[-\frac{1}{2}(\mathbf{x}-\mu)^\mathsf{T}\Sigma^{-1}(\mathbf{x}-\mu)\Big]$$

The correlation between two variables can be given a numerical value with *Pearson's Correlation Coefficient*. It is defined as

$$\rho_{xy} = \frac{COV(X,Y)}{\sigma_x \sigma_y}$$

This value can range in value from -1 to 1. If the covariance is 0 than $\rho = 0$. A value greater than 0 indicates that the relationship is a positive correlation, and a negative value indicates that there is a negative correlation. Values near -1 or 1 indicate a very strong correlation, and values near 0 indicate a very weak correlation.

Correlation and covariance are very closely related. Covariance has units associated with it, and correlation is a unitless ratio.

We can use `scipy.stats.pearsonr` function to compute the Pearson coefficient. It returns a tuple of the Pearson coefficient and of the 2 tailed p-value.

One Dimensional Kalman Filters

The equations for the univariate Kalman filter are:

Predict

\hlineEquation	Implementation	Kalman Form
\hline$\bar{x} = x + f_x$	$\bar{\mu} = \mu + \mu_{f_x}$	$\bar{x} = x + dx$
	$\bar{\sigma}^2 = \sigma^2 + \sigma_{f_x}^2$	$\bar{P} = P + Q$
\hline		

Update

\hlineEquation	Implementation	Kalman Form
\hline$x = \|\mathcal{L}\bar{x}\|$	$y = z - \bar{\mu}$	$y = z - \bar{x}$
	$K = \frac{\bar{\sigma}^2}{\bar{\sigma}^2 + \sigma_z^2}$	$K = \frac{\bar{P}}{\bar{P}+R}$
	$\mu = \bar{\mu} + Ky$	$x = \bar{x} + Ky$
	$\sigma^2 = \frac{\bar{\sigma}^2 \sigma_z^2}{\bar{\sigma}^2 + \sigma_z^2}$	$P = (1-K)\bar{P}$
\hline		

Multivariate Kalman Filters

Reminder: correlation between variables drastically improves the posterior (e.g. tracking position and velocity, even if only position is measured - via a radar for example)

Kalman Filter Algorithm

Initialization

```
1. Initialize the state of the filter
2. Initialize our belief in the state
```

Predict

```
1. Use process model to predict state at the next time step
2. Adjust belief to account for the uncertainty in prediction
```

Update

```
1. Get a measurement and associated belief about its accuracy
2. Compute residual between estimated state and measurement
3. Compute scaling factor based on whether the measurement
or prediction is more accurate
4. set state between the prediction and measurement based
on scaling factor
5. update belief in the state based on how certain we are
in the measurement
```

Predict

\hlineUnivariate	Univariate (Kalman form)	Multivariate
\hline$\bar\mu = \mu + \mu_{f_x}$ $\bar\sigma^2 = \sigma_x^2 + \sigma_{f_x}^2$ \hline	$\bar x = x + dx$ $\bar P = P + Q$	$\bar{\mathbf x} = \mathbf{Fx} + \mathbf{Bu}$ $\bar{\mathbf P} = \mathbf{FPF}^\mathsf{T} + \mathbf{Q}$

Without worrying about the specifics of the linear algebra, we can see that:

$\mathbf x,\ \mathbf P$ are the state mean and covariance. They correspond to x and σ^2.

$\mathbf F$ is the *state transition function*. When multiplied by $\mathbf x$ it computes the prior.

$\mathbf Q$ is the process covariance. It corresponds to $\sigma^2_{f_x}$.

$\mathbf B$ and $\mathbf u$ are new to us. They let us model control inputs to the system.

Update

\hlineUnivariate	Univariate (Kalman form)	Multivariate
\hline $\mu = \frac{\bar\sigma^2\,\mu_z + \sigma_z^2\,\bar\mu}{\bar\sigma^2 + \sigma_z^2}$ $\sigma^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 + \sigma_2^2}$ \hline	$y = z - \bar x$ $K = \frac{\bar P}{\bar P + R}$ $x = \bar x + Ky$ $P = (1 - K)\bar P$	$\mathbf y = \mathbf z - \mathbf{H}\bar{\mathbf x}$ $\mathbf K = \bar{\mathbf P}\mathbf H^\mathsf{T}(\mathbf H\bar{\mathbf P}\mathbf H^\mathsf{T} + \mathbf R)^{-1}$ $\mathbf x = \bar{\mathbf x} + \mathbf{Ky}$ $\mathbf P = (\mathbf I - \mathbf{KH})\bar{\mathbf P}$

$\mathbf H$ is the measurement function. We haven't seen this yet in this book and I'll explain it later. If you mentally remove $\mathbf H$ from the equations, you should be able to see these equations are similar as well.

$\mathbf{z}$, $\mathbf{R}$ are the measurement mean and noise covariance. They correspond to z and σ_z^2 in the univariate filter (I've substituted μ with x for the univariate equations to make the notation as similar as possible).

$\mathbf{y}$ and $\mathbf{K}$ are the residual and Kalman gain.

The details will be different than the univariate filter because these are vectors and matrices, but the concepts are exactly the same:

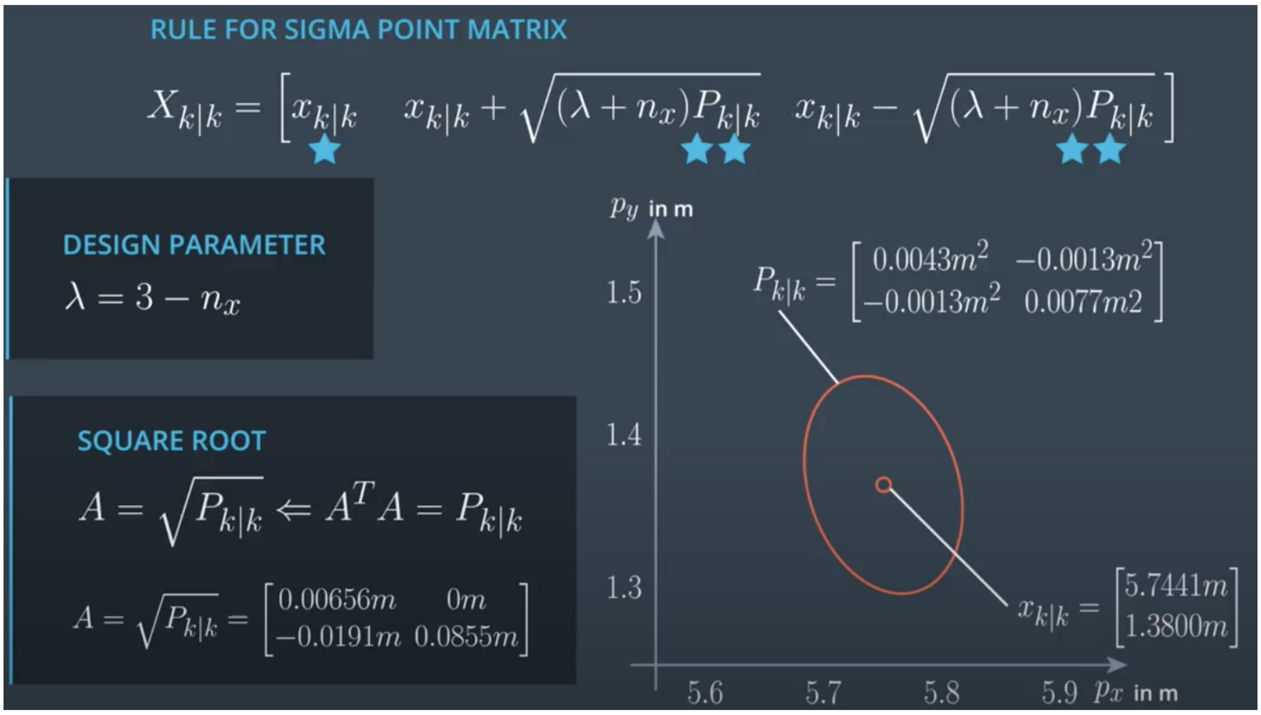
- Use a Gaussian to represent our estimate of the state and error
- Use a Gaussian to represent the measurement and its error
- Use a Gaussian to represent the process model
- Use the process model to predict the next state (the prior)
- Form an estimate part way between the measurement and the prior

Your job as a designer will be to design the state $(\mathbf{x}, \mathbf{P})$, the process $(\mathbf{F}, \mathbf{Q})$, the measurement $(\mathbf{z}, \mathbf{R})$, and the measurement function $\mathbf{H}$. If the system has control inputs, such as a robot, you will also design $\mathbf{B}$ and $\mathbf{u}$.

Unscented Kalman Filter

Generate sigma points

The number of sigma points depends on the state dimension: 2n + 1 (mean + 2 points by dimension)



RULE FOR SIGMA POINT MATRIX

$$X_{k|k} = \left[x_{k|k} \quad x_{k|k} + \sqrt{(\lambda + n_x) P_{k|k}} \quad x_{k|k} - \sqrt{(\lambda + n_x) P_{k|k}} \right]$$

DESIGN PARAMETER

$$\lambda = 3 - n_x$$

SQUARE ROOT

$$A = \sqrt{P_{k|k}} \Leftarrow A^T A = P_{k|k}$$

$$A = \sqrt{P_{k|k}} = \begin{bmatrix} 0.00656m & 0m \\ -0.0191m & 0.0855m \end{bmatrix}$$

$$P_{k|k} = \begin{bmatrix} 0.0043m^2 & -0.0013m^2 \\ -0.0013m^2 & 0.0077m2 \end{bmatrix}$$

$$x_{k|k} = \begin{bmatrix} 5.7441m \\ 1.3800m \end{bmatrix}$$

Predict sigma points

Predict Mean and Covariance