GAUSS-MARKOV MODELS

JONATHAN HUANG

1. Introduction

Kalman filters and its many cousins (EKF, UKF, etc...) have seen a lot of action over the last few decades in many application domains, like robotics, computer graphics and speech recognition, to name but a few. A solid understanding of the underlying probabilistic model (the Gauss-Markov model) is fundamental as a simplified case for understanding more complicated dynamical probabilistic models like SLAM (Simultaneous Localization and Mapping).

The story of the Kalman filter begins with the humble Gaussian distribution, which can be thought of as the *maximum entropy* distribution with fixed mean and (co)variance. While Gaussian distributions are often too limited in expressiveness, they are powerful because they can be reasoned about in closed form, which typically leads to efficient probabilistic inference algorithms.

In these notes, I'll start with a discussion of basic properties of the Gaussian, then show how we can use them to derive efficient inference algorithms for Gauss-Markov models, like the Kalman filter.

2. Basic Properties of Gaussian Distributions

There are typically two ways to parameterize the density function of a Gaussian distribution. The first (more familiar) way is known as the *Moment Parameterization*:

$$P(x) \propto \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right),$$

where μ is the *mean* of the distribution and Σ, the *covariance matrix*. In the moment form, zeros in the covariance matrix imply marginal independencies (i.e. $\Sigma_{ij} = 0$ implies that $x_i \perp x_j$).

The second parametric form is known as the *Natural Parameterization*:

$$P(x) \propto \exp\left(J^T x - \frac{1}{2}x^T P x\right).$$

The two parameterizations are related by the following equations:

$$P = \Sigma^{-1}, \tag{1}$$

$$J = \Sigma^{-1}\mu. \tag{2}$$

J and P are known as the *information vector* and *matrix* respectively. In contrast to the moment form, zeros in the information matrix imply conditional independencies. For example, if $P_{ij} = 0$, then we know that x_i and x_j are independent *conditioned* on all other x_k. Alternatively put, if we draw an Markov random field for P, then we do not need an edge between x_i and x_j if $P_{ij} = 0$.

For the rest of these notes, x_1, x_2 will be vectors in real vector spaces, and distributed as:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim \mathcal{N}\left(\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12}^T & \Sigma_{22} \end{bmatrix}\right),$$

in Moment parameters and equivalently as

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim \tilde{\mathcal{N}}\left(J = \begin{bmatrix} J_1 \\ J_2 \end{bmatrix}, P = \begin{bmatrix} P_{11} & P_{12} \\ P_{12}^T & P_{22} \end{bmatrix}\right),$$

in Natural parameters.

Despite the conceptual ease of going back and forth between the moment and natural forms, each has its own advantages and disadvantages, and it is often preferable to use one over the other in many real problems. In particular, typical operations that one might perform on a Gaussian involve linear transformations and conditioning operations. While both can be done in both parameterizations, the former takes a simpler form in the moment parameterization and the latter takes a simpler form in the natural parameterization.

Linear Transformations. If we have a normally distributed vector $x \sim \mathcal{N}(\mu, \Sigma)$, we will want to know the distribution of x after passing it through a linear transformation corresponding to the matrix A. Not only will linear transformations be important for the prediction step of filtering, but marginalization can be seen as a linear transformation. It is not difficult to see that Ax must also be normally distributed, and computing moments of Ax can be done using linearity of expectation:

$$\mathbb{E}[Ax] = A\mathbb{E}[x] = A\mu \tag{3}$$

$$\mathbb{E}[(Ax)(Ax)^T] = \mathbb{E}[Axx^TA^T] = A\mathbb{E}[xx^T]A^T = A\Sigma A^T \tag{4}$$

Therefore, $Ax \sim \mathcal{N}(A\mu, A\Sigma A^T)$.

As a corollary, we can derive an easy rule for marginalizing a Gaussian in moment parameters. For example, we have:

$$\begin{aligned} x_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} &\sim \mathcal{N}\left(\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}\mu, \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}\Sigma\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}^T\right), \\ &\sim \mathcal{N}(\mu_1, \Sigma_1), \end{aligned} \tag{5}$$

which... should not be a terribly earth-shattering result.

In the Natural paramterization, the same marginalization can also be performed using:

$$x_1 \sim \tilde{\mathcal{N}}(J_2 - P_{12}^T P_{11}^{-1} J_1, P_{22} - P_{12}^T P_{11}^{-1} P_{12}) \tag{6}$$

$$x_2 \sim \tilde{\mathcal{N}}(J_1 - P_{12} P_{22}^{-1} J_2, P_{11} - P_{12} P_{22}^{-1} P_{12}^T) \tag{7}$$

Conditioning. Now we turn to conditioning, which we first formulate for Natural parameters since it is simpler. If we start off with a joint distribution, $x \sim \tilde{\mathcal{N}}(J, P)$, we would like to know the distribution of $x_1|x_2$, which can be derived by simply writing out the distribution of x, and 'absorbing' all the terms which involve x_2

into the normalization constant.

$$\begin{aligned} P(x_1|x_2) &\propto \exp\left(\begin{bmatrix} J_1 \\ J_2 \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \frac{1}{2} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} P_{11} & P_{12} \\ P_{12}^T & P_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right) \\ &\propto \exp\left(J_1^T x_1 - \frac{1}{2}\left(x_1^T P_{11} x_1 + 2 x_2^T P_{12}^T x_1 + (\text{terms with } x_2) \right) \right) \\ &\propto \exp\left((J_1 - P_{12} x_2)^T x_1 - \frac{1}{2} x_1^T P_{11} x_1 \right), \end{aligned}$$

which shows that:

$$x_1|x_2 \sim \tilde{\mathcal{N}}(J_1 - P_{12}x_2, P_{11}), \tag{8}$$

with respect to the Natural parameterization. This equation is useful for Gaussian process regression.

In Moment parameters, the above conditioning update looks like:

$$x_1|x_2 \sim \mathcal{N}(\mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2), \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}), \tag{9}$$

and is usually derived using various blockwise matrix inversion identities.

Another important update which appears in the derivation of the Information filter is if we just have a prior on x_1, given by $\mathcal{N}(J_1, P_1)$, and would like to condition (on observed data, z) by multiplying by a likelihood function proportional to $\mathcal{N}(J_2, P_2)$, in which case, the posterior on x_1 is simply given by:

$$x_1|z \sim \tilde{\mathcal{N}}(J_1 + J_2, P_1 + P_2) \tag{10}$$

3. Gauss-Markov Models

We will consider a joint distribution over over a sequence of latent (continuous) state variables x_1, x_2,..., x_T and a sequence of corresponding observations z_1,z_2,...,z_T which factors as:

$$p(x_1, \ldots, x_T, z_1, \ldots, z_T) = p(x_1)p(z_1|x_1)\prod_{t=2}^{T} p(x_t|x_{t-1})p(z_t|x_t)$$

The state variables, x_t, and the obvervations z_t do not have to live in spaces with the same dimension. We will assume that $x_t \in \mathbb{R}^n$ and that $z_t \in \mathbb{R}^m$. The index t should be thought of as a discrete timeslice index. As in Hidden Markov models, conditional independencies (see Figure 1) dictate that past and future states are decorrelated given the current state, x_t at time t (this is sometimes called the *Markov assumption*). For example, if we know what x_2 is, then no information about x_1 can possibly help us to reason about what x_3 should be. We will also assume that the conditional distributions $P(x_t|x_{t-1})$ and $P(z_t|x_t)$ are Gaussian:

$$x_t|x_{t-1} \sim \mathcal{N}(A \cdot x_{t-1}, \Gamma) \tag{11}$$

$$z_t|x_t \sim \mathcal{N}(C \cdot x_t, \Sigma). \tag{12}$$

The matrices (A, Γ) and (C, Σ) are typically referred to as the parameters of the *motion model* and *observation (or measurement) model*, respectively. We will refer to this family of models as *Gauss-Markov models*. Though the parameters can in general be different for each timestep, I will assume that the parameters are static for simplicity.

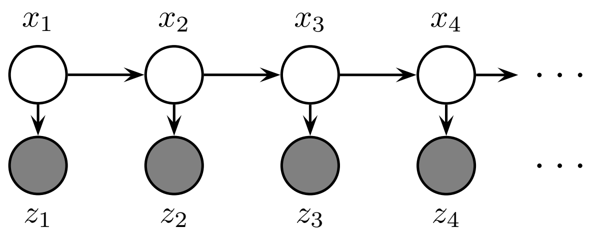

FIGURE 1. The Independence Diagram of a Gauss-Markov model

For example, if $A = I_{n\times n}$ (sometimes called the *Brownian Motion model*), then the particle does a random walk through state space. Without making measurements, the entropy of x increases monotonically at each time step until $P(x_t)$ converges to uniform.

Another common motion model takes velocities into account. In this setting, we set $x_t = [s_t v_t]^T$, where s_t and v_t represent displacement and velocity at time t, respectively. A simple model which accounts for velocity uses:

$$A = \begin{bmatrix} I_{n\times n} & (\Delta_t)I_{n\times n} \\ 0 & I_{n\times n} \end{bmatrix},$$

where Δ_t is the elapsed time between timeslices t and $t+1$.

4. Recursive State Estimation

We now focus on discussing (abstractly) the recursive update rules for *filtering*, which is the problem of computing a state distribution at each timestep t, conditioned on all past observations. In other words, we would like to compute $P(x_t|z_1,\ldots,z_t)$.

There are two steps to the recursion, a *Prediction/Rollup* step and a *Conditioning* step. Given the posterior distribution computed at the previous timestep, $P(x_{t_1}|z_t,\ldots,z_{t-1})$, the Prediction/Rollup step computes a *predictive distribution* at time t (conditioned on the same measurements) by multiplying by the motion model, then marginalizing out the previous timestep:

$$\begin{aligned} p(x_t|z_1,\ldots,z_{t-1}) &= \int p(x_t, x_{t-1}=x|z_1,\ldots,z_{t-1})dx, \\ &= \int p(x_t|x_{t-1}=x, z_1,\ldots,z_{t-1})p(x_{t-1}=x|z_1,\ldots,z_{t-1})dx, \qquad (13) \\ &= \int p(x_t|x_{t-1}=x)p(x_{t-1}=x|z_1,\ldots,z_{t-1})dx, \qquad (14) \end{aligned}$$

where Equation 13 follows from the Chain rule, and Equation 14 follows from the Markov assumption.

The Conditioning step incorporates a new observation z_t using Bayes rule:

$$\begin{aligned} p(x_t|z_1,\ldots,z_t) &\propto p(z_t|x_t, z_1,\ldots,z_{t-1})p(x_t|z_1,\ldots,z_{t-1}), \\ &\propto p(z_t|x_t)p(x_t|z_1,\ldots,z_{t-1}), \qquad (15) \end{aligned}$$

where, again, Equation 15 follows from the Markov assumption.

When the state space is discrete (HMMs), the complexity of prediction/rollup step is squared in the number of states, while the complexity of conditioning is linear. For infinite state spaces, we have several options. One way is to grid up the space into discrete parts and to just run the discrete HMM forward updates. Typically, this results in a discrete state space whose size is exponential in n, the dimension of the true continuous space. Alternatively, one can use something like Gauss Markov model, which, as we will show, can be done in cubic time in the dimension at the cost of being potentially oversimplified.

We can now derive algorithms for performing the recursive updates for a Gauss-Markov model using the Gaussian identities from the previous section. The two steps, Prediction/Rollup, and Conditioning, as will be shown, are dual to each other in the sense that Prediction/Rollup is 'easy' with respect to Moment parameters and hard with respect to the Natural Parameters, while the Conditioning step is easy with respect to the Moment parameters and hard with respect to the Natural parameters. I will use the following notation:

$$\begin{aligned} x_t|z_1,\ldots,z_{t-1} &\sim \mathcal{N}(m_{t|t-1}, V_{t|t-1}), \\ &\sim \tilde{\mathcal{N}}(j_{t|t-1}, P_{t|t-1}), \\ x_t|z_1,\ldots,z_t &\sim \mathcal{N}(m_{t|t}, V_{t|t}), \\ &\sim \tilde{\mathcal{N}}(j_{t|t}, P_{t|t}). \end{aligned}$$

4.1. **Lazy Gauss-Markov Filter.** The *Lazy Gauss-Markov Filter* always performs operations in the 'easiest' parameterization — so prediction/rollup is performed with respect to Moment parameters and conditioning is done with respect to Natural parameters. At each timestep, we will incur some cost of converting between the two parameterizations - and in particular, it will be the cost of inverting the matrix $V_{t|t}$, which is $O(n^3)$.

The first question we want to answer is how to get a joint distribution over (x_t, x_{t-1}). By assumption, we have $x_t = Ax_{t-1} + \epsilon$, where ϵ is an independent draw from $\mathcal{N}(0, \Gamma)$. We now write:

$$\begin{bmatrix} x_t \\ x_{t-1} \end{bmatrix} = \begin{bmatrix} A & I \\ I & 0 \end{bmatrix} \cdot \begin{bmatrix} x_{t-1} \\ \epsilon \end{bmatrix}.$$

We know that:

$$\begin{bmatrix} x_{t-1} \\ \epsilon \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m_{t-1|t-1} \\ 0 \end{bmatrix}, \begin{bmatrix} V_{t-1|t-1} & 0 \\ 0 & \Gamma \end{bmatrix} \right),$$

and therefore, by Equation 3 and 4, we have that:

$$\begin{aligned} \begin{bmatrix} x_t \\ x_{t-1} \end{bmatrix} &\sim \mathcal{N}\left(\begin{bmatrix} A & I \\ I & 0 \end{bmatrix} \begin{bmatrix} m_{t-1|t-1} \\ 0 \end{bmatrix}, \right. \\ &\qquad \left. \begin{bmatrix} A & I \\ I & 0 \end{bmatrix} \cdot \begin{bmatrix} V_{t-1|t-1} & 0 \\ 0 & \Gamma \end{bmatrix} \cdot \begin{bmatrix} A^T & I \\ I & 0 \end{bmatrix} \right) \\ &\sim \mathcal{N}\left(\begin{bmatrix} Am_{t-1|t-1} \\ m_{t-1|t-1} \end{bmatrix}, \begin{bmatrix} AV_{t-1|t-1}A^T + \Gamma & AV \\ (AV)^T & V \end{bmatrix} \right). \end{aligned} \tag{16}$$

Algorithm 1: Pseudocode for the Lazy Gauss-Markov Prediction/Rollup step

```
PredictionRollup
input : $j_{t-1|t-1}, P_{t-1|t-1}$
output: $m_{t|t-1}, V_{t|t-1}$
// Convert to Moment Parameters (if necessary)
1 $m_{t-1|t-1} \leftarrow P_{t-t|t-1} \cdot j_{t-1|t-1}$ ;
2 $V_{t-1|t-1} \leftarrow P^{-1}_{t-t|t-1}$ ;
// Motion Model update (repeat over several timesteps if
    necessary)
3 $m_{t|t-1} \leftarrow A \cdot m_{t-1|t-1}$ ;
4 $V_{t|t-1} \leftarrow A \cdot V_{t-1|t-1} \cdot A^T + \Gamma$ ;
```

Applying the marginalization rule with respect to Moment parameters (Equation 5) gives the prediction/rollup step in the Moment parameterization:

$$m_{t|t-1} \leftarrow Am_{t-1|t-1}, \tag{17}$$

$$V_{t|t-1} \leftarrow AV_{t-1|t-1}A^T + \Gamma. \tag{18}$$

Now suppose that we receive an observation z_t and would like to find $m_{t|t}$, $V_{t|t}$ by conditioning on our newest measurement. We begin by converting from the Moment parameterization to the Natural parameterization:

$$j_{t|t-1} \leftarrow V^{-1}_{t|t-1} m_{t|t-1}, \tag{19}$$

$$P_{t|t-1} \leftarrow V^{-1}_{t|t-1}. \tag{20}$$

We would like to use Equation 10, but we need to first find the Natural parameters of the likelihood function (remember, we now care about terms which include x_t, not z_t):

$$\begin{aligned} P(z_t|x_t) &\propto \exp\left(-\frac{1}{2}(z_t - Cx_t)^T \Sigma^{-1} (z_t - Cx_t)\right) \\ &\propto \exp\left((C^T\Sigma^{-1}z_t)^T x_t - \frac{1}{2} x_t^T C^T \Sigma^{-1} C x_t + (\text{constants})\right), \end{aligned}$$

which shows that the information vector and matrix of the observation model are $C^T\Sigma^{-1}z_t$ and $C^T\Sigma^{-1}C$, respectively. Note that the information matrix can be precomputed if the observation model is static. Our conditioning update, following Equation 10, is then:

$$j_{t|t} \leftarrow j_{t|t-1} + C^T\Sigma^{-1}z_t, \tag{21}$$

$$P_{t|t} \leftarrow P_{t|t-1} + C^T\Sigma^{-1}C. \tag{22}$$

4.2. **Kalman Filter.**

We can also derive an inference algorithm which works entirely with Moment parameters, which gives the well-known *Kalman Filter*. It is important to understand the fact that the Kalman Filter itself is an algorithm, and not the underlying probabilistic model.

In the Kalman Filter setting, the Prediction/Rollup step (commonly just called the Prediction step) is the same as it was in the lazy version. The only work we need to do is to rewrite the conditioning step (commonly called the *Correction*) with

Algorithm 2: Pseudocode for the Lazy Gauss-Markov Conditioning step

```
CONDITION
input : $m_{t|t-1}, V_{t|t-1}$
output: $j_{t|t}, P_{t|t}$
// Convert to Natural Parameters (if necessary)
1 $j_{t|t-1} \leftarrow V_{t|t-1}^{-1} \cdot m_{t|t-1}$ ;
2 $P_{t|t-1} \leftarrow V_{t|t-1}^{-1}$ ;
// Observation Model update (repeat if many observations)
3 $j_{t|t} \leftarrow j_{t|t-1} + C^T \Sigma^{-1} z_t$ ;
4 $P_{t|t} \leftarrow P_{t|t-1} + C^T \Sigma^{-1} C$ ;
```

respect to the Moment parameterization, and it will be necessary to use a matrix identity known as the *Matrix Inversion Lemma* (or *Sherman-Morrison-Woodbury formula*).

Lemma 4.1 (Matrix Inversion Lemma)**.**

$$(V^{-1} + C^T \Sigma^{-1} C)^{-1} = V - VC^T(\Sigma + CVC^T)^{-1} CV \tag{23}$$

Proof. It really isn't worth your time, but if you care, just multiply both sides by the matrix $(V^{-1} + C^T\Sigma^{-1}C)$, then sit back and watch things cancel. □

There is a related identity which we will also use:

$$(V^{-1} + C^T \Sigma^{-1} C)^{-1} C^T \Sigma^{-1} = VC^T(CVC^T + \Sigma)^{-1} \tag{24}$$

Applying the above matrix identities to the updates in the Lazy form yields the following:

$$\begin{aligned} V_{t|t} &= P_{t|t}^{-1} \\ &= \left(P_{t|t-1} + C^T \Sigma^{-1} C\right)^{-1} \\ &= \left(V_{t|t-1}^{-1} + C^T \Sigma^{-1} C\right)^{-1}, \\ &= V_{t|t-1} - V_{t|t-1} C^T \left(\Sigma + C V_{t|t-1} C^T\right)^{-1} C V_{t|t-1}, \\ &= (I - K_t C) V_{t|t-1}, \qquad (25) \end{aligned}$$

where we define $K_t = V_{t|t-1} C^T (\Sigma + CV_{t|t-1}C^T)^{-1}$ (sometimes called the *Kalman Gain*). Similar matrix inversion acrobatics yields an update for $m_{t|t}$, and so our Kalman filter conditioning update is given by:

$$m_{t|t} \leftarrow m_{t|t-1} + K_t \cdot (z_t - C \cdot m_{t|t-1}), \tag{26}$$

$$V_{t|t} \leftarrow (I - K_t \cdot C) \cdot V_{t|t-1}. \tag{27}$$

The term $(z_t - C \cdot m_{t|t-1})$ is typically called the *innovation* and intuitively measures how far a new observation deviates from the expected.

5. Conclusion

There are two important things which I have not addressed at all. One is the *smoothing* problem, which takes observations at all times $[1, T]$ into account to form a posterior marginal at $t \in [1, T]$. There is a similar recursion (sometimes called

Algorithm 3: Pseudocode for the Kalman Filter Prediction step

```
PredictionRollup
input  : $m_{t-1|t-1}, V_{t-1|t-1}$
output: $m_{t|t-1}, V_{t|t-1}$
1 $m_{t|t-1} \leftarrow A \cdot m_{t-1|t-1}$ ;
2 $V_{t|t-1} \leftarrow A \cdot V_{t-1|t-1} \cdot A^T + \Gamma$ ;
```

Algorithm 4: Pseudocode for the Kalman Filter Correction step

```
Condition
input  : $m_{t|t-1}, V_{t|t-1}$
output: $m_{t|t}, V_{t|t}$
1 $K_t \leftarrow V_{t|t-1} \cdot C^T(\Sigma + C \cdot V_{t|t-1} \cdot C^T)^{-1} \cdot C \cdot V_{t|t-1}$ ;
2 $m_{t|t} \leftarrow m_{t|t-1} + K_t \cdot (z_t - C \cdot m_{t|t-1})$ ;
3 $V_{t|t} \leftarrow (I - K_t \cdot C) \cdot V_{t|t-1}$ ;
```

RTS smoothing) which solves this problem efficiently. Another important problem is how we should set the model parameters — A, Γ, C, and Σ — and there exist some approaches for automatically learning these from data.

At the end of the day, the vanilla Kalman filter such as the one derived above is fairly limited since things are rarely linear Gaussian. It serves primarily as an important training example for the hundreds of variants which have been invented to fix its shortcomings, and is an important case where we can do things exactly, which is nice.