# Decision making: Sequential Probability Ratio Test

Infer latent state from noisy measurements

neuromatch academy

## Part 1:

① 有一个 sequence，其中的数据采样为独立、同分布(iid)

② 这个 sequence 是从什么样的 Probability distribution 中采样的？有两个备选 $P_1$ $P_2$

③ 那我们就把数据采样代入两个分布，即想知道："如果数据采样真的来自分布 $P_1$ (or $P_2$) 时采到这个 sequence(目为出现的情况) 的概率 $P_1(data)$ (or $P_2(data)$) 是多少？"

我们认为已经出现的事件(采样到这个 sequence)是大概率事件，所以 P(data) 应较大.

若 $P_1(data) > P_2(data)$,

→ 分布 $P_1$ 更容易采样出 data

→ 倾向于认为 data 来自 $P_1$

④ SPRT 用于判断一个数据采样为独立、同分布(iid)的 sequence 属于哪个分布

⑤ 也可以认为 SPRT 利用数据采样为独立、同分布(iid)的 sequence，来对两个分布的"真实值"估计.

# Sequential Probability Ratio Test (SPRT)

- **Likelihood ratio test for sequential and i.i.d data**
- Binary model parameter estimation

$$S_N = \log \frac{P_1(data)}{P_0(data)} \overset{iid}{=} \log \frac{\prod_i^N p_1(data_i)}{\prod_i^N p_0(data_i)} = \sum_i^N \log \frac{p_1(data_i)}{p_0(data_i)} = \sum_i^N \log \Lambda(data_i)$$

Log likelihood ratio

- As new data comes in ...

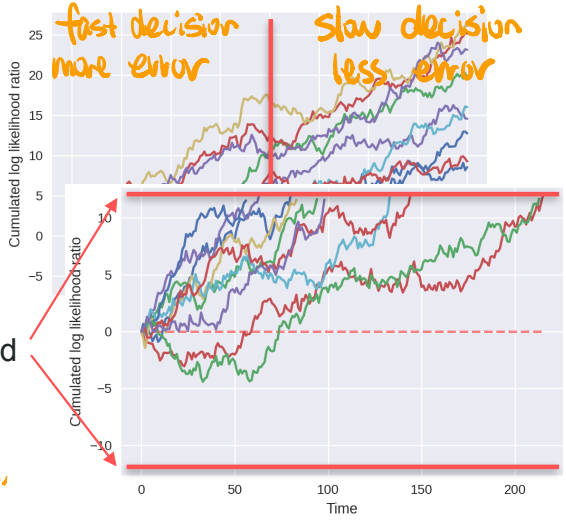
  Cumulated evidence: $S_N = S_{N-1} + \log \Lambda(data_N)$

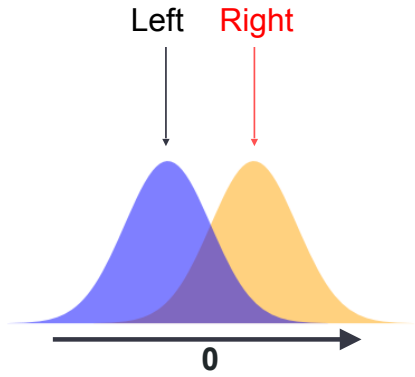
  new evidence

- **Stopping rule**: enough data to make decision?


# Stopping rule & Speed/Accuracy tradeoff

- Speed/Accuracy tradeoff
- Fixed-time stopping rule
- Thresholding stopping rule



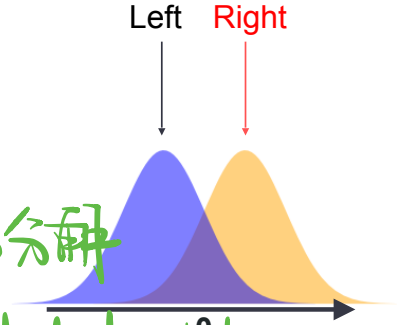




"for example, if we are sensitive about time, like we are choosing which direction to run when some animals are attacking us, it's good to make a decision within a certain amount of time. This is called the fixed-time stopping rule."

# Part 2:

Imagine you are performing a random dot motion task

- Velocity is randomly generated from 1 of 2 distributions
- Goal: Make decision fast and accurately




Q: accumulated evidence 的成分分析

what kind of process the accumulated evidence is undergoing?

# SPRT as a Drift-Diffusion Model (DDM)

$$p_{left}(v) = \mathcal{N}(\mu_{left}, \sigma^2) \qquad p_{right}(v) = \mathcal{N}(\mu_{right}, \sigma^2)$$

If $p_{right}$ is the data generating distribution

$v_n = \mu_{right} + \sigma \cdot \epsilon_n$ where $\epsilon_n \sim \mathcal{N}(0,1)$ is Gaussian noise

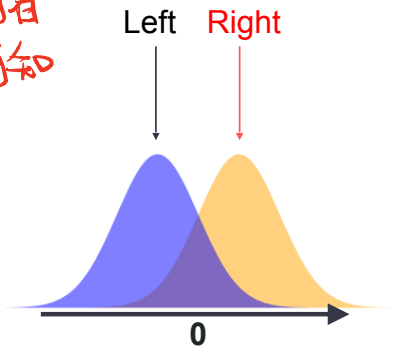
a scaled gaussian noise with zero mean

多写者不可知

**new evidence** $\log \Lambda_i = 0.5\frac{(\mu_{right} - \mu_{left})^2}{\sigma^2} + \frac{\mu_{right} - \mu_{left}}{\sigma^2}\epsilon$

Drifting term

Diffusion term




推导：

$$\log \Lambda_i = \log \frac{P_r(v_i)}{P_l(v_i)} = \log P_r(v_i) - \log P_l(v_i) = \log\left[\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(v_i-\mu_r)^2}{2\sigma^2}}\right] - \log\left[\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(v_i-\mu_l)^2}{2\sigma^2}}\right]$$

$$= \frac{1}{\sqrt{2\pi}\sigma}\left[-\frac{(v_i-\mu_r)^2}{2\sigma^2} + \frac{(v_i-\mu_l)^2}{2\sigma^2}\right] \quad \text{已知 } v_i = \mu_r + \underbrace{\sigma\varepsilon_i}_{\varepsilon} \text{ 则} = \frac{1}{\sqrt{2\pi}\sigma}\left(0.5\frac{(\mu_r-\mu_l)^2}{\sigma^2} + \frac{(\mu_r-\mu_l)\sigma\varepsilon_i}{\sigma^2}\right)$$

$$S_n = S_{n-1} + D + C \cdot \epsilon_n$$

$$\mathbb{E}[S_n] = nD$$

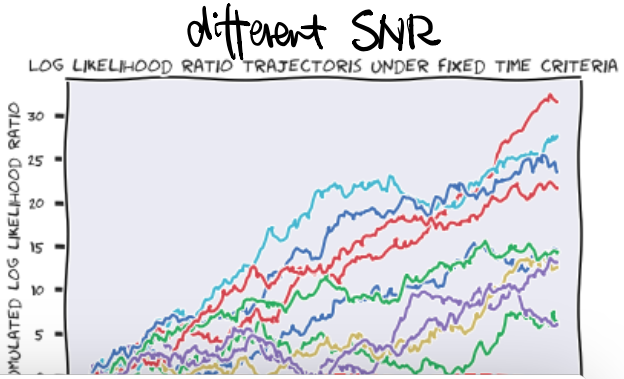
$$Cov[S_n] = nC \quad ?$$

$$SNR_n = \frac{\mathbb{E}[S_n]}{\sqrt{Cov[S_n]}} \propto \sqrt{n}$$


Shades: ± 3 std

# Exercise 1: Simulate DDM with fixed-time stopping rule

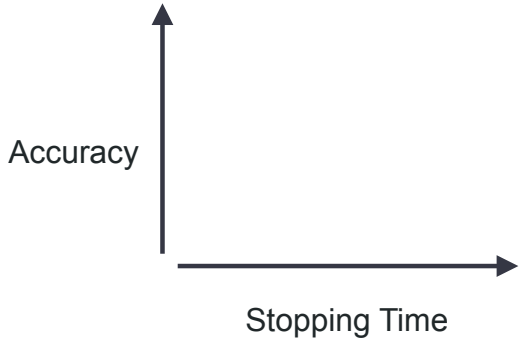
- let $\mu_{right} = 1, \mu_{left} = -1, \sigma = 3$
- Complete the code to simulate DDM with fixed length
- Run simulations and plot traces of cumulated evidence
- Use sliders to play with noise levels and stopping times




```
simulate_and_plot_SPRT_fixedtime(sigma, stop_time, num_sample)
```

# Exercise 2: Speed/Accuracy Tradeoff

- Simulate DDM with different stopping times
- Calculate average accuracy for each stopping time
- Visualize how accuracy changes with stopping time
- How SNR affect the curve




accumulated evidence 的域值的大小与
做出判断的可信程度的关系?

# Thresholding stopping rule

置信程度：我们允许的出现错误判断的最大程度 $\alpha$，即 choosing right (left) when the truth is the opposite.

- Have a desired maximum error rate $\alpha$

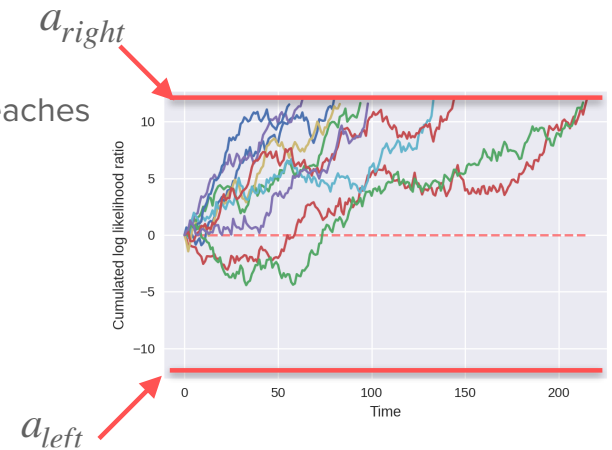
↓

- Keep making measurements until cumulated evidence reaches

$$a_{left} = \log \frac{\alpha}{1-\alpha} < 0 \quad \text{or} \quad a_{right} = \log \frac{1-\alpha}{\alpha} > 0$$

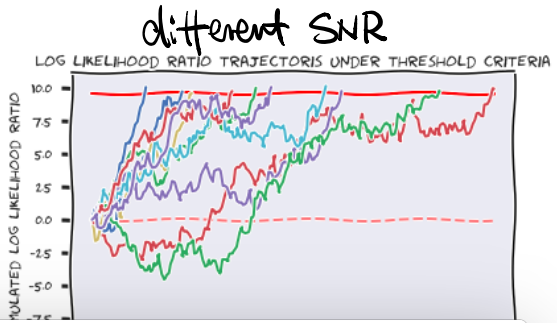
↓

当 cumulated evidence 达到 $a_{right}$ 时，我们停止 sequence 取从 Right 的决策，而这个决策出错概率最大为 $\alpha$！



# Exercise 3: Simulate DDM with thresholding stopping rule

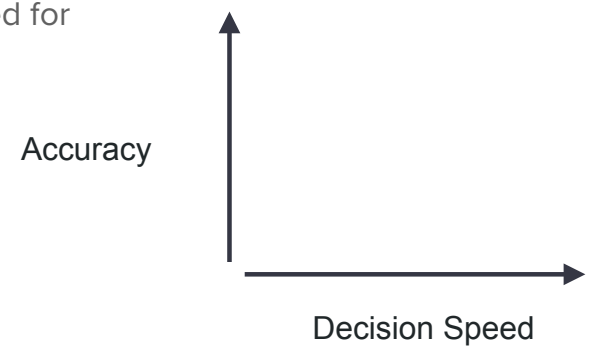
- Implement the thresholding stopping rule give an error rate $\alpha$
- Simulate DDM with thresholding stopping rule
- Visualize cumulated evidence as a function of time
- Use slider to play with different noise levels and error rates




```
simulate_and_plot_SPRT_fixedthreshold(sigma, num_sample, alpha)
```

# Exercise 4: Speed/Accuracy Tradeoff

- Simulate DDM with thresholding stopping rule under different error rates
- Calculate average accuracy and average decision speed for each error rate
- Visualize how accuracy changes with decision speed
- How SNR affects the curve

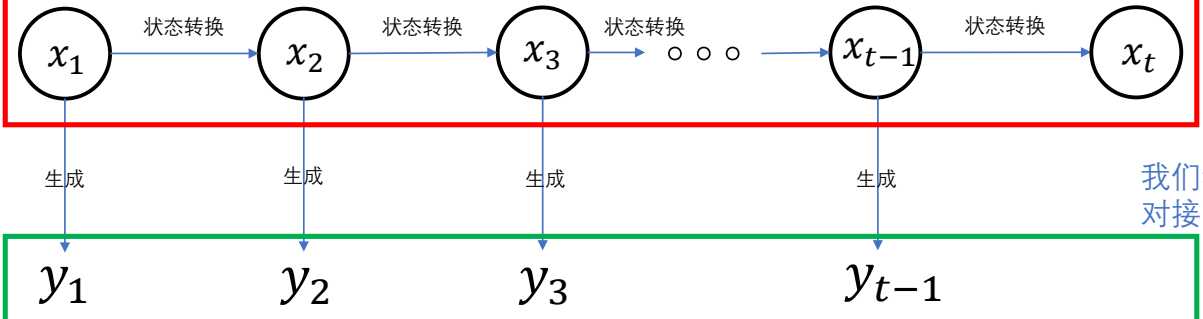


# 隐马尔可夫模型

# 场景描述 (与HMM无关的实际现象)

系统隐状态随时间转换，隐状态是不可知的。

In neuroscience, we often think a system like a single neuron or population of neurons is transitioning between a set of discrete **hidden** states. For example, we believe a system is switching between the active state and inactive state, or between a wake state and a sleep state.




And we also believe the neural activities, like electric signals or calcium imaging signals, are generated from these discrete dynamics.

Then we try to infer the discrete states (discrete dynamics) from the noisy observations.

我们希望利用这些观测值，来对接下来的隐状态进行推断。

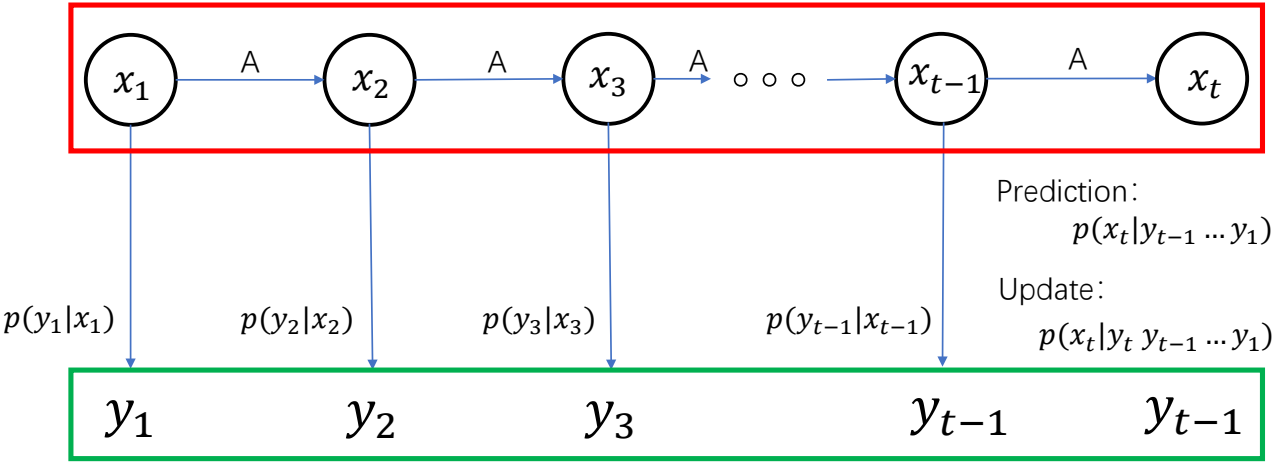
在每个时刻，我们只能得到基于隐状态的观测值。

综上，我们希望通过不同时刻的观测值来去推断（研究）系统的隐状态。

# HMM模型 对上述场景的 数学建模

Markov chain: **you cannot directly observe，** the current latent state $x_t$ only depends on the previous $x_{t-1}$, 即，W2D2的马尔可复性。

The probability to jump from one latent state to another at next time is specified by the element of a transition matrix A. The matrix element $a_{ij}$ represents the probability to transition from state i to state j.



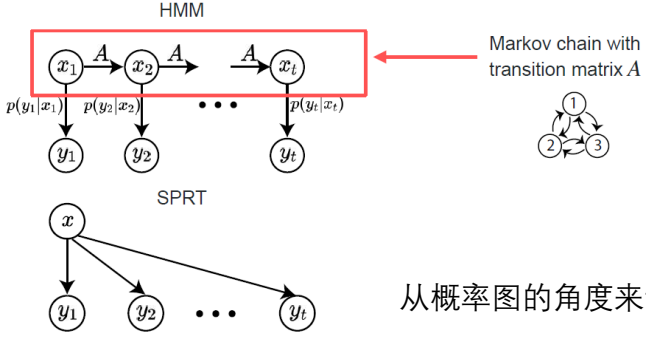

Observation model: produces **noisy measurements of the latent states**

Basically, the forward inference of Hidden Markov Model and Kalman filter in tutorial 3 is **a repeated procedure of making prediction from the data at hand now, and then correcting our prediction with new evidence** (prediction & update) . Imagine when you are predicting without evidence, you always increase your variance but you can decrease it again by combining new data. **This is the key idea of inferencing Hidden Markov Model.**

# HMM模型 与 SPRT 的联系和区别

联系：两者都是对隐状态的推断。
- SPRT可以看成是利用i.i.d观测数据，对真实分布的时不变参数的推断。
- HMM可以看成是利用某一时刻及之前的观测数据，对该时刻的隐变量的推断。

区别：
- Different from the sequential probability ratio test, **now the latent state is no longer fixed over time.** Instead **we have a full dynamic**.
- So, in Hidden Markov Model we don't have the chance to increase our confidence by making more measurements. Instead what we do now is to predict the current latent state from previous state and then combine that prediction with the current measurement or new data point to make a complete inference.




从概率图的角度来讲，SPRT像是朴素贝叶斯？

# 用 HMM模型 描述一个neuroscience 场景

## Exercise 1: Simulation a binary HMM with Gaussian observable

**1. 隐变量：**

- Binary state variable $x_t \in \{0,1\}$

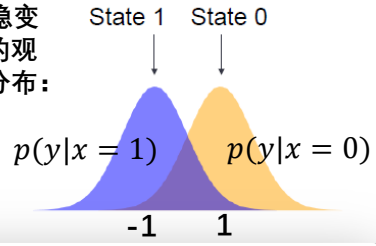
**2. 隐变量状态转换矩阵A：**

- Switch to the other state with prob $p_{switch}$

  Stay in the same state with prob $1 - p_{switch}$

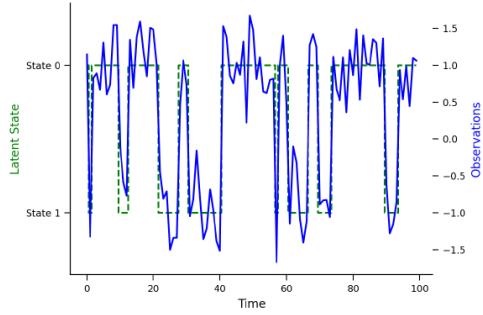
$$A = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{pmatrix} 1-p_{switch} & p_{switch} \\ p_{switch} & 1-p_{switch} \end{pmatrix} \end{array}$$

**3. 不同隐变量对应的观测值的分布：**



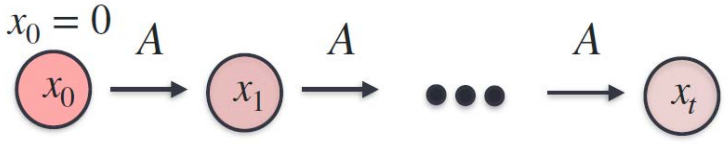

**上述三点设置，就给出了HMM在进行推断时要用到的三种概率分布。**

**4. Latent states和对应的observation**




```
n_components = 2
model = hmm.GaussianHMM(n_components=n_components, covariance_type="full")
# Markov chain
model.startprob_ = np.asarray(startprob)
model.transmat_ = ... # Transition matrix
# Observation model
model.means_ = np.array([[1.0],[-1.0]])
model.covars_ = ...  # Observation noise covariances
```

# HMM模型中，
# 如果不用observations，是否可以预测接下来的隐状态？




$$p(x_t = i) = \sum_j \left[ p(x_{t-1} = j) A_{ji} \right]$$

$$A = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{pmatrix} 1 - p_{switch} & p_{switch} \\ p_{switch} & 1 - p_{switch} \end{pmatrix} \end{array}$$

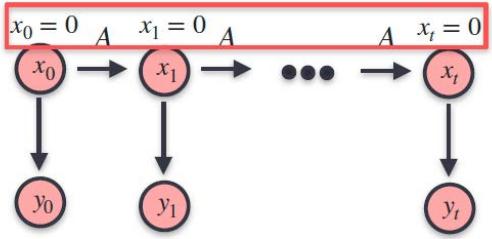
Here assuming we are in state zero in the beginning and we have 1% chance to switch at each time step. As you can see in the first few steps, we are still pretty sure we will stay in state zero, but as time goes, we will be more and more uncertain which state we are in. Actually, the probability of being in state zero $p(x_t = 0)$ decays exponentially from 1 to 1/2 over time, 因为 未来状态的数量呈指数增长。如果我们往后看， 初始状态的贡献将呈指数下降。 Mathematically this can be calculated. The probability can be calculated using the probability propagation relation of the Markov chain.

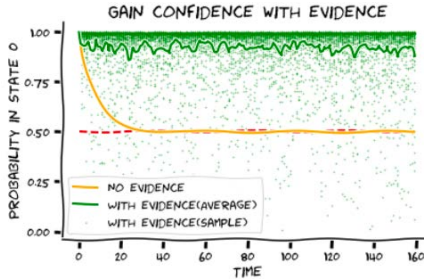
- If the $p_{switch}$ is small, you should see an exponential decay from one towards one half like the one in the in the last slide.
- What if $p_{switch}$ is large? In this case we are almost certain that we will switch to the other state next time. So, if you plot the probability of states 0 over time, you will see a damping oscillation of that.

总结：只用确定的初始状态和状态转换矩阵A，是无法准确的推测在t时刻的状态的。因此我们要用observations。

# HMM模型中，
# incorporate evidence to improve the inference of latent states

$$\underbrace{p(x_t \mid y_{1:t})}_{\text{Posterior}(x_t \mid y_{1:t})} \propto \underbrace{p(x_{t-1} \mid y_{1:t-1})}_{\text{Posterior}(x_{t-1} \mid y_{1:t-1})} \underbrace{p(x_t \mid x_{t-1})}_{\text{Prediction}(x_t \mid y_{t-1})} \underbrace{p(y_t \mid x_t)}_{\text{Likelihood}(y_t \mid x_t)}$$




我们假设x的状态始终保持在0，并在每个时间点生成不同的观测值。

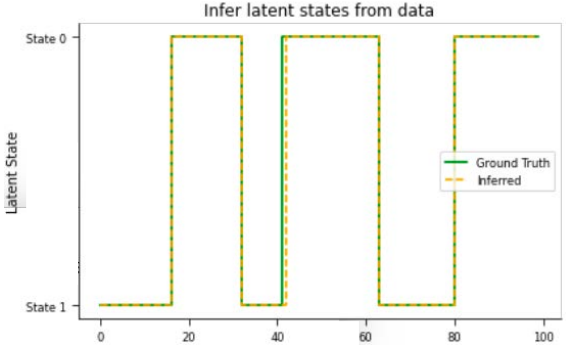
- 基于上述inference，我们利用observations来对$x_t$进行推断；
- 只采用$x_0$和A来对$x_t$进行推断（我们知道效果不好）；

我们想要看一下incorporate evidence后，对隐变量的状态推断是不是更好了

After finishing this, you should be able to get a plot like this. As you can see the posterior probability of staying at 0 will be close to 1 with low transition rate and small noise. Please try different values of transition rates and noise levels, observe how that affects the curve. What if you have a high transition rate but low noise?

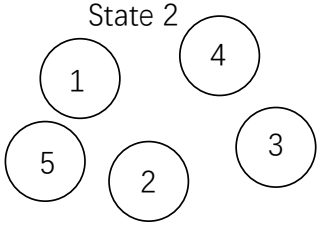
Now let's unfreeze the state sequence (不再限制x一定要为1了) and see for a randomly generated state sequence, is forward inference able to recover it from noisy data?

In this exercise, you will generate a random state sequence in corresponding measurements, then use the forward inference function to get the posterior probabilities. You need to finish the code to extract the inferred states by choosing the one with higher posterior probability at each time. We've already provided some code to visualize the true state sequence together with the inferred ones like this. You can see here the inferences making error at time around time 41. Please try different switching probabilities and noise levels and see how inference accuracy changes with them.

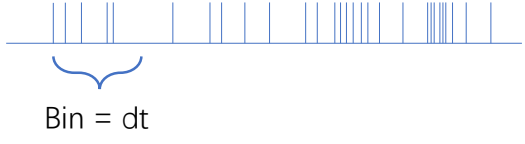


# EM算法讲解前建立的场景

**a network of Poisson spiking neurons**




Firing rate for cell 3 at state 2 = $\lambda_{state=2}^{cell=3}$




- $K = 3$ latent states, $C = 5$ cells.
- All the neurons share the same latent state at a given time.
- Each neuron has a distinctive firing rate at each state.
  Neuron $c$ has firing rate $\lambda_i^c$ in state i

The corresponding spike counts within time bins will follow a Poisson distribution

Discretize: time bins of length $dt$

- Spike counts of cell $c$ in state $i$: $y_t^c \sim \text{Poisson}(\lambda_i^c dt)$
- Spike count distribution of cell=3 at state=2: $y_t^{cell=3} \sim \text{Poisson}(\lambda_2^3 dt)$

---

The transition probability of this HMM is characterized by switching rates to other states (unit: Hz)

$A_{rate} =$

| | 1 | 2 | 3 |
|---|---|---|---|
| 1 | Hz | | |
| 2 | | | |
| 3 | | | |

For the transition matrix A(ij) is given by the product of transition rate times dt.

# EM算法

EM算法利用HMM模型生成的数据来对HMM模型的参数进行估计。
通过NMA的视频：

https://www.youtube.com/watch?v=umU4wUWIKvg&list=PLkBQOLLbi18NaQb7HyKp-O3Ag5JJ9MmXi&index=11

很难理解EM具体的流程。

EM是一种只针对于HMM的参数估计方法。我暂时不做详细的研究，等到以后真的需要用了，建议看邹博的视频。在回过头来看上面这个，或许会有更好的认识。

# 卡尔曼滤波器

知识点

# 卡尔曼滤波器 — 最佳线性滤波器

- 实现简单
- 纯时域滤波器，不需要进行频域变换
- 工程应用较多

## 1.状态转移


对于一个系统，有

- **状态量：**在本例中，用t时刻位置$p_t$、速度$v_t$表示系统的状态$x_t$；
- **控制量：**在本例中，是驾驶员对系统的加速度$u_t$。如果驾驶员不踩油门or刹车，则$u_t$=0，汽车将会做匀速直线运动；

已知
t-1时刻的状态$x_{t-1}$，
则t时刻的状态$x_t$为：

$x_t \Leftarrow x_{t-1}$

$$p_t = p_{t-1} + v_{t-1} \times \Delta t + u_t \times \frac{\Delta t^2}{2} \quad [1]$$

$$v_t = v_{t-1} + u_t \times \Delta t \quad [2]$$

$$\begin{bmatrix} p_t \\ v_t \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_{t-1} \\ v_{t-1} \end{bmatrix} + \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} u_t$$

[1]是如何求出的？设汽车做 匀加速 直线运动，则[2]易知。而且在t ~ t+Δt时间段的平均速度为$\frac{v_t+v_{t-1}}{2} = v_{t-1} + \frac{u_t \times \Delta t}{2}$，则在该时间段增加的距离为$(v_{t-1} + \frac{u_t \times \Delta t}{2}) \times \Delta t$，由此得$p_t$。

- 实际上，这两个公式的目的，是为了推导出 系统状态变换 的一般形式。
- 输入变量$p_{t-1}$、$v_{t-1}$的线性组合构成了输出变量$p_t$、$v_t$。这就是为什么卡尔曼滤波器是线性滤波器，因为它只能描述状态之间的线性关系。

**设：状态转移矩阵：如何从上一时刻的状态来推测当前时刻的状态。**

$$F_t = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$$

**控制矩阵：控制量$u_t$如何作用于当前状态**

$$B_t = \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix}$$

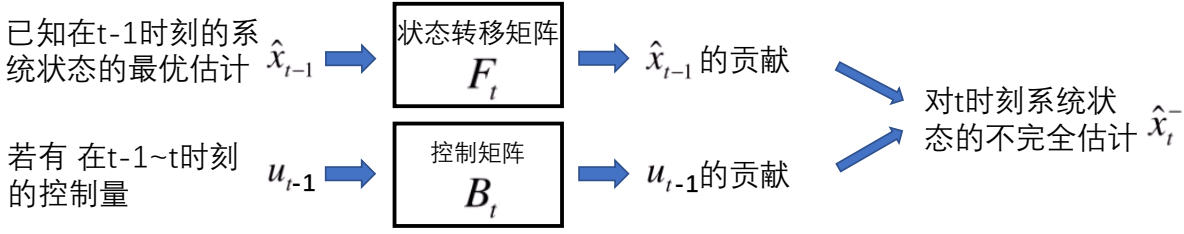
**公式一：状态预测公式**

$$\hat{x}_t^- = F_t \hat{x}_{t-1} + B_t u_{t-1}$$

“^”表示估计量，我们永远无法得知真实值，我们只能根据观测，来尽可能估计状态的真实值。
“-”表示状态值是由上一时刻的状态值推测而来的。接下来我们会根据观测量$z_t$来修正这个状态值。因此，“-”表示“不完全”。修正了之后的值才是最佳估计值，没有“-”。

## 公式一：状态预测公式

$$\hat{x}_t^- = F_t\hat{x}_{t-1} + B_t u_{t\text{-}1}$$




$F_t$ $B_t$ 是如何得到的？个人理解，它们是由特定的理论推导出的。在本例中

- 若是匀速直线运动，且Δt=1，则Ft=[1,1;0,1]
- 若是匀加速直线运动，则还有Bt=[1/2;1]

这两个矩阵都是由实际的物理理论推导出来的。

但是，所有状态，无论是

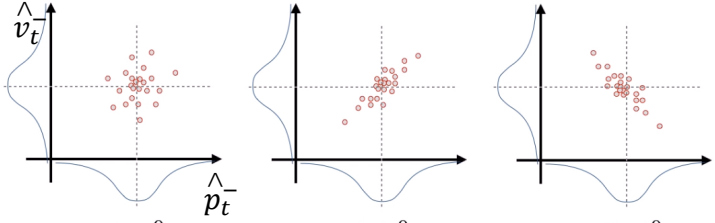
- t-1 (t) 时刻的系统状态的最优估计，还是
- t时刻系统状态的不完全估计

都是具有不确定性的。因为状态中的每个量都是随机变量。

我们如何来衡量状态的不确定性（噪声）呢？

**2.（基础）**对于一个服从高斯分布的变量，其不确定性噪声可以用方差来衡量；
对于两个服从高斯分布的**独立**变量，其不确定性噪声可以用**各自**的方差来衡量；
但是，如果两个服从高斯分布的变量之间**有关联**，仅仅用各自的方差是无法完整衡量不确定性的。
这时，就要用到两个变量之间的协方差 https://www.zhihu.com/question/20852004，
并构成协方差矩阵：**衡量由两个变量各自产生的、两个变量联合产生的不确定性。**

## 协方差矩阵




$\sigma_{12}=0$ 两个变量之间无关　　$\sigma_{12}>0$ 同向变化　　$\sigma_{12}<0$ 反向变化

可以通俗的理解为：两个变量在变化过程中是同方向变化？还是反方向变化？同向或反向程度如何？

你变大，同时我也变大，说明两个变量是同向变化的，这时协方差就是正的。

你变大，同时我变小，说明两个变量是反向变化的，这时协方差就是负的。

从数值来看，协方差的数值越大，两个变量同向程度也就越大。反之亦然。

咱们从公式出发来理解一下：

$$Cov(X,Y)=E[(X-\mu_x)(Y-\mu_y)]$$

因此，在卡尔曼滤波器中，状态（本例中包含两个变量）的 不确定性or波动or噪声 要用协方差矩阵来完整的衡量。

$$=\begin{bmatrix}\sigma_{11} & \sigma_{12}\\ \sigma_{12} & \sigma_{22}\end{bmatrix}$$

# 2. 噪声协方差矩阵的传递

设：$\hat{x}_t^- = \begin{pmatrix} \hat{p}_t^- \\ \hat{v}_t^- \end{pmatrix}$ 的协方差矩阵$P_t^-$衡量其状态的不确定性，

$\hat{x}_{t-1} = \begin{pmatrix} \hat{p}_{t-1} \\ \hat{v}_{t-1} \end{pmatrix}$ 的协方差矩阵$P_{t-1}$衡量其状态的不确定性，

状态预测公式 $\hat{x}_t^- = F_t\hat{x}_{t-1} + B_t u_t$

问：如何计算出$\hat{x}_t^-$的不确定性？

如何让状态的不确定性在不同时刻传递？

$$P_t^- \xleftarrow{?} P_{t-1}$$

**公式二：不确定性在各个时刻之间的传递关系**

$$P_t^- = FP_{t-1}F^T + Q$$

注：预测模型不是100%准确，会有噪声，所以要引入

协方差矩阵Q，即状态转移协方差矩阵，来表示预测模型本身带来的噪声。

以上两步在说什么？

当我们已知在t-1时刻的最优估计 $\hat{x}_{t-1}$ 以及该状态估计的不确定性度量$P_{t-1}$

1. 我们可以利用理论，仅仅利用手头有的t-1时刻的状态估计，来对t时刻的状态 $\hat{x}_t^-$ 进行不完美的预测 (公式1);

但是，这个“大致”估计并不是最优的，因为它没有考虑到我们在t时刻的观测值$z_t$所提供的信息！

那问题来了，既然在t时刻有观测值$z_t$，为什么不直接用这个观测值来表示t时刻的最优状态 $\hat{x}_t$？

因为，1) 观测值并不一定包含状态的所有元素，例如，接下来你会看到，本例中的观测值仅仅是系统在t时刻的位置测量，并没有包含速度信息。更深层次来讲，各状态量与观测值之间并不一定是对应关系，即，状态值也可能是某种抽象的表征。2) 观测值可能具有较大的噪声，单独使用可能效果也不理想。

因此，就引出了

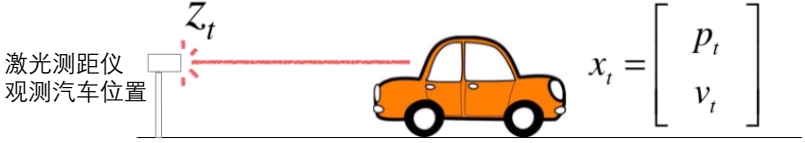
利用 t-1 时刻的最优估计状态 来先预测t时刻的状态 (prediction),

再利用 t 时刻的观测值对预测值进行修正 (update)

的过程。这个过程和HMM的inference过程相似。

2. 同时，也要对该预测的不确定性程度进行衡量(公式2)。公式2的第一项，是一个由数学原理推导出来的结果，是一个由公式1和协方差性质确定的形式。也就是说，$\hat{x}_t^-$的不确定性程度是被确定的。但是，由于我们的预测本身也具有不确定性，因此，我们在预测出的不确定性中加入噪声Q，表示由预测模型本身带来的预测不确定性。在本例的实现中，Q=[0.0001, 0; 0, 0.00001], 表示加载到不完美预测的协方差矩阵中的、由系统提供的噪声。这是系统本身固有的。我们在实现中人为设定。

# 3. 观察矩阵




汽车的真实$x_t$与观测状态$z_t$之间的转换矩阵 $H = \begin{bmatrix} 1 & 0 \end{bmatrix}$

**H构建了状态与观测之间的联系**

理想情况：$z_t = Hx_t$

实际情况：$z_t = Hx_t + v$

观测值并不可靠，具有观测噪声v

**观测值的协方差矩阵R**，表示各个观测量之间的不确定性。

在本例中，由于观测值是一维的，所以R是一个值，表示观测值的方差。

接下来，在举一个观测值是多维的例子。

$$z_t = Hx_t + v$$

$$\underset{3\times1}{\begin{bmatrix}\text{观测值}\end{bmatrix}} = \underset{3\times5}{\begin{bmatrix}\text{观察矩阵 H}\end{bmatrix}} \cdot \underset{5\times1}{\begin{bmatrix}\text{真实状态值}\end{bmatrix}} + \underset{3\times1}{\begin{bmatrix}\text{噪声值}\end{bmatrix}}$$

- 各个状态值可能是内部隐变量的状态
- 观测值的协方差矩阵R是对三个变量的协方差衡量
- 观察矩阵是连接状态和观测的桥梁

第一、二步，基于t-1时刻的信息$\hat{x}_{t-1}$ $P_{t-1}$ ，基于纯理论，得出了预测 $\hat{x}_t^-$ $P_t^-$

第三步，得出了 t时刻观测量$z_t$ 和 对应的不确定性程度R

如何基于观测值来对状态的预测进行更新?

# 4. 状态更新

利用观测值的状态修正

$$\hat{x}_t = \hat{x}_t^- + K_t(z_t - H\hat{x}_t^-)$$

公式三

对t时刻观测值　对t时刻观测值的预测

相对于t时刻不完全预测值观测值提供的额外信息。这个额外信息用来修正状态预测。

卡尔曼系数

$$K_t = P_t^- H^T \left(HP_t^- H^T + R\right)^{-1}$$

公式四

定性分析:

1. 衡量 预测状态协方差矩阵$P_t^-$ 与 观察量的协方差矩阵R的大小
   - 若R较大，说明观测值的波动较大，相对不可信。因此，提供的额外信息的权重应较小;
   - 反之，若R较小，则说明观测值较稳定，相对可信。对额外信息的权重要较大。
2. 把额外信息的表现形式从“观察域”转换到“状态域”

# 5. 噪声协方差矩阵的更新

$$P_t = (I - K_t H) P_t^-$$

公式五

- 更新t时刻最佳估计状态的噪声分布(协方差矩阵)，留给下一轮使用。
- 在这一步中，状态的不确定性是减小的。而在下一轮迭代中，由于传递噪声的引入，不确定性又会增大。
- 卡尔曼滤波器就是在这种不确定性的变化中寻求一种平衡的。

# 综上：

预测

$$\hat{x}_t^- = F\hat{x}_{t-1} + Bu_{t-1}$$

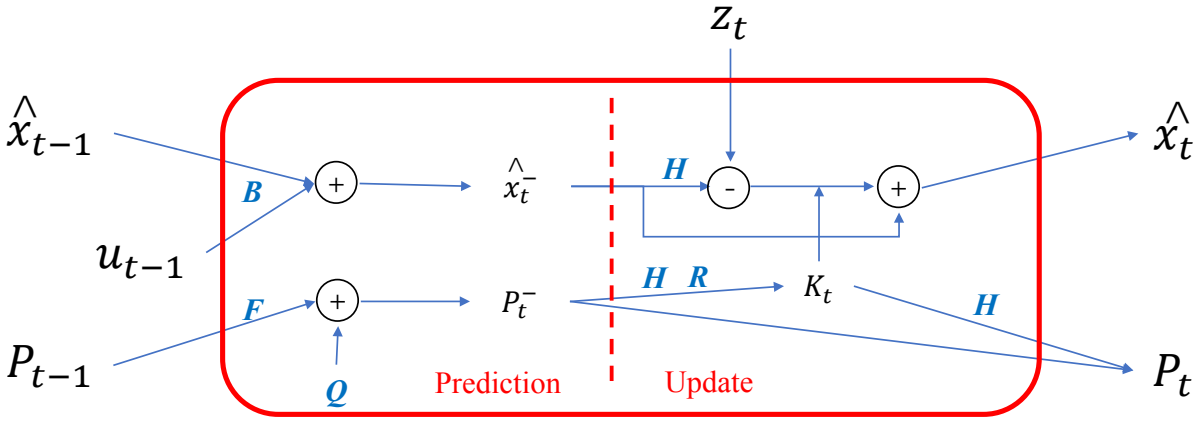
$$P_t^- = FP_{t-1}F^T + Q$$

更新

$$K_t = P_t^- H^T \left(HP_t^- H^T + R\right)^{-1}$$

$$\hat{x}_t = \hat{x}_t^- + K_t(z_t - H\hat{x}_t^-)$$

$$P_t = (I - K_t H)P_t^-$$




卡尔曼(线性)滤波器

可见，这五个矩阵的设置至关重要

***F*，*B*：** 由系统相关的理论、场景决定

***Q*：** 引入系统预测的不确定性

***H*：** 观测值与状态的固有关联

***R*：** 观测值的噪声衡量

假设汽车做匀速直线运动，每秒一米

```
Z=(1:100); %观测值
noise=randn(1,100); %方差为1的高斯噪声
Z=Z+noise;

X=[0; 0]; %状态
P=[1 0; 0 1]; %状态协方差矩阵
F=[1 1; 0 1]; %状态转移矩阵
Q=[0.0001, 0; 0 0.0001]; %状态转移协方差矩阵
H=[1 0]; %观测矩阵
R=1; %观测噪声方差

figure;
hold on;

for i=1:100

  X_ = F*X;
  P_ = F*P*F'+Q;
  K = P_*H'/(H*P_*H'+R);
  X = X_+K*(Z(i)-H*X_);
  P = (eye(2)-K*H)*P_;

  plot(X(1), X(2)); %画点，横轴表示位置，纵轴表示速度
end
```

每一秒对汽车距离的理想观测值，没有加观测噪声

观测噪声，较大

真实观测值

初始状态

初始状态自身决定自身，相互无关联

这是由物理学原理决定的。由于Δt=1，所以第一行第二列为1

不是非常重要，经过几次迭代，状态就会趋近于真实状态了

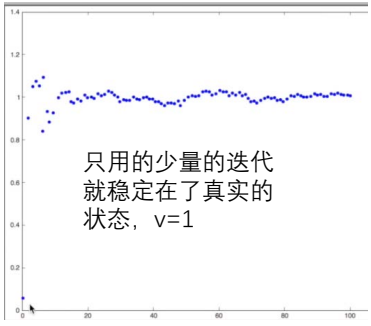
对状态转移矩阵非常有信心，所以将Q设成非常小

状态量与观测值之间的对应关系

对于一维的观测，观测值的协方差矩阵就是这个观测值的方差，即，第二行的设定，为1

基于t-1时刻信息的不完全预测

基于t时刻观测值的状态更新



只用的少量的迭代就稳定在了真实的状态，v=1

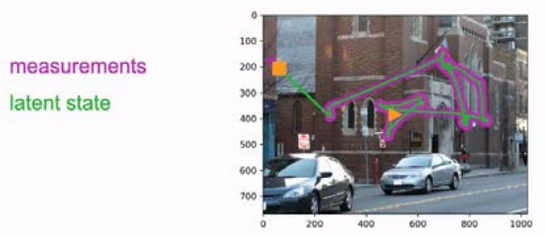
# 卡尔曼滤波器

在NMA中的讲解

**latent space models** 利用观察数据(the population activity of neurons) 作为输入 来对 隐过程(attention) 进行推断的模型, **HMM** and **linear dynamical system (e.g., Kalman filter)**。

# 1. HMM 与 Kalman 的区别：

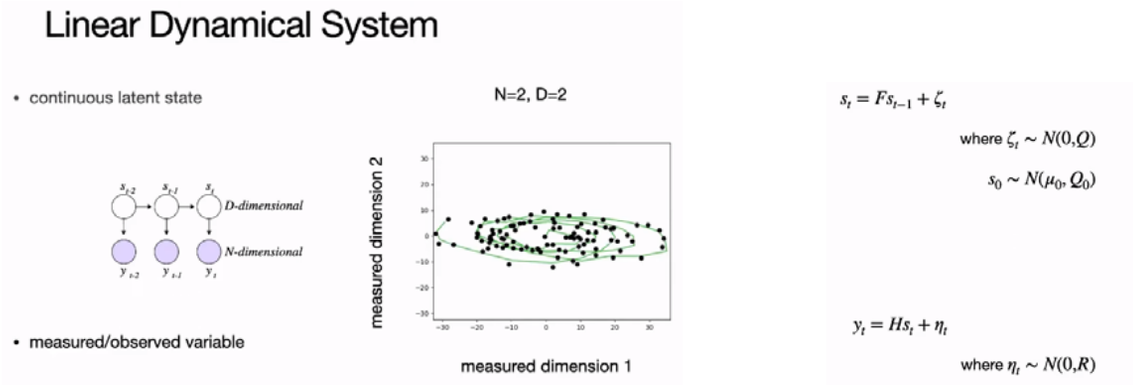
So far, you've learned about **detecting discrete hidden or latent states** from data measurements, which can be well modeled, by, for instance, the **hidden Markov model**. In this tutorial, we will think about what happens if the hidden underlying process cannot be described by a discrete number of states. So, take for instance the example of eye tracking data. We have noisy measurements of an eye tracker, here in purple, that tries to capture the trajectory of a subject's eye as they view an image. So, our aim is to estimate the true trajectory of the eye, which is here in green. **The eye, of course, does not jump between a couple of discrete states, but moves in a continuous space here (continuous latent states)**, the picture space. As we see here. So, instead of a hidden Markov model, we can use what is called **a linear dynamical system**.




In the framework of the linear dynamical systems we try to estimate the hidden **continuous process** that underlies a dataset. However, we could also think of this as a problem that the brain needs to solve: when it gets noisy measurements from its sensory receptors (input data). it needs to extract hidden information about the outside world which causes the activation of the sensory receptors.

# 2. Linear Dynamical System 对隐过程的数学假设：

So now let's assume we have some data measurements Y with two dimensions. This data is collected over time and our aim is to find a latent process that underlies our measured data. typically, the latent process would have a lower dimension D, but for visualization we will plot it in the same dimensions that same two dimensions




**重点：对隐过程的数学假设**

we assumed **a very specific structure for this later process**. specifically, we assume that

- the latent structure can be expressed by **continuous variable *s*** that propagates through time t with some temporal smoothness which is expressed by **the transition matrix F** and **some Gaussian noise** which has some covariance Q

  **这实际就是卡尔曼滤波器的公式 1&2**。首先，隐状态 s 之间的连续的线性转换被转换矩阵 F 定义，状态转换之间必有噪声存在。

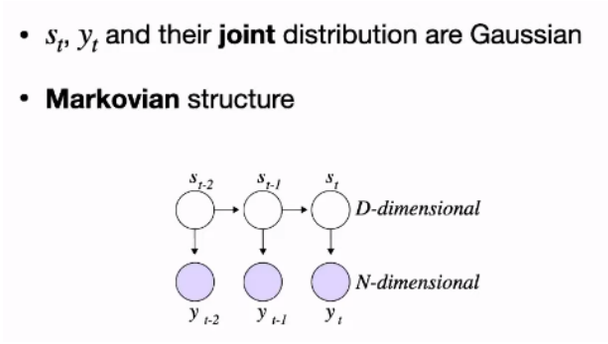
  - ✧ 若状态 s 是一维的，则线性状态转换用一个常量 F 就可以描述。噪声变量$\zeta_t$ 同样也是一维的，Q 表示噪声的方差；

- ✧ 若状态 s 是多维的，则线性状态转换就要矩阵 F 表示。噪声$\zeta_t$ 也要有多个噪声变量来表示，噪声的联合分布同样是高斯分布，而表示这个联合分布的波动就要用到噪声变量的协方差矩阵 Q。
  - 可见，**Q 用来衡量$s_t$的波动程度**！这一点十分关键!

- this latent variable $s$ drives the observed or measured data Y which has its own gaussian noise term with its own covariance R

  **这是卡尔曼滤波器中的观察矩阵公式，将状态域与观察域用观察矩阵 H 联系起来。**噪声项要与观察域具有相同的维度，R 表示各维度噪声之间的协方差矩阵。可见，**R 用来衡量观测值$y_t$之间的波动程度**。

-




where the latent at time point t is conditionally independent of all the previous time points given its direct predecessor

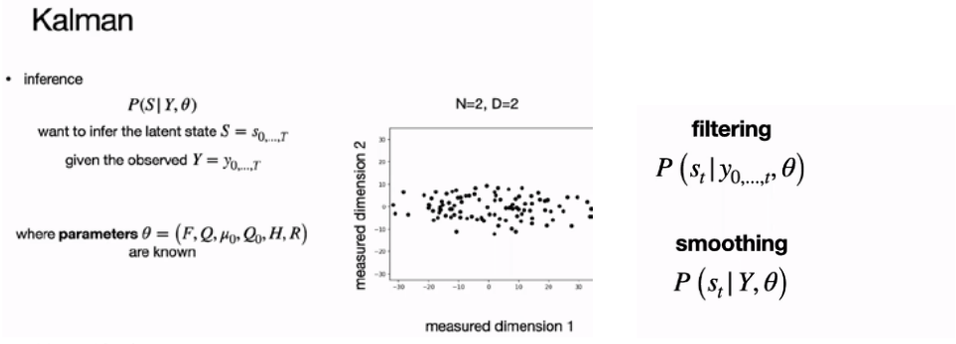
## 3. Kalman 滤波器的 inference

Next, we're going to think about how to infer the latent state distribution from the data.
我们假设隐变量服从高斯分布，而我们要推断出高斯分布的均值和方差(变量为一维) or 均值向量和协方差矩阵(变量为多维)。This is where we use the Kalman filter. We need to infer the latent state trajectory directly from the observed data.

**卡尔曼滤波器在 neuroscience 中的应用**:
For instance, take a recording from a population of neurons. **We do not know what they are coding for， but want to infer it from their activity.** This process is called inference, and has a nice solution formalized by the Kalman filter for the linear dynamical system.

**总体来讲，卡尔曼滤波器进行隐状态推断的两个步骤：**
The Kalman inference can be done in two steps.
- Filtering: we estimate the latent online, so as data comes in. We're working forward in time basically, every time we get a new data point, we're estimating the corresponding parameter(s) of the latent distribution.
- Smoothing: using the results from the filtering step, we estimate the latent given the whole data sequence. this is also called the backward pass since we are working from the last sample back to the first.

注：在做 inference 时，参数θ时已知的




**卡尔曼滤波器的具体流程：**
**FILTERING STEP:**

已知在 $t-1$ 时刻对隐变量(Z)分布的最优预测 $\hat{S}_{t-1}$，我们来预测 $t$ 时刻的"不完美"分布 $\hat{S}_t^-$。注意，在实际计算中的 latent variable 是分布的向量和方差(or 向量向量，协方差阵)

由于还没有考虑 $t$ 时刻观测值 $y_t$，此时的预测是不完美的，用 $-$ 表示。

$$\hat{S}_{t-1} \sim N(\hat{\mu}_{t-1}, \hat{\Sigma}_{t-1}) \xrightarrow{\text{Prediction}} \hat{S}_t^- \sim N(\hat{\mu}_t^-, \hat{\Sigma}_t^-)$$

$$\hat{S}_t \sim N(\hat{\mu}_t, \hat{\Sigma}_t)$$

$$\hat{\mu}_t^- = F \cdot \hat{\mu}_{t-1}$$

$$\hat{\Sigma}_t^- = F \cdot \hat{\Sigma}_{t-1} \cdot F^T + Q$$

$Q$ 流在预测中，对 $\hat{S}_t^-$ 新增的不确定性程度。

$$\mu_t = F \cdot \mu_{t-1} + \gamma$$
$$\gamma \sim N(0, Q)$$
$$\text{cov}(\mu_t) = \text{cov}(F \cdot \mu_{t-1} + \gamma)$$
$$= \text{cov}(F \cdot \mu_{t-1}) + \text{cov}(\gamma) + 2\text{cov}(F \cdot \mu_{t-1}, \gamma)$$
$$= F \cdot \mu_{t-1} \cdot F^T + Q + 0$$

Projection

Correction

$$\hat{\mu}_t = \hat{\mu}_t^- + K_t(y_t - H\hat{\mu}_t^-)$$
$$\hat{\Sigma}_t = \hat{\Sigma}_t^- - KH\hat{\Sigma}_t^-$$
$$K_t = \hat{\Sigma}_t^- H^T (H\hat{\Sigma}_t^- H^T + R)^{-1}$$

Latent Space

Observed Space

$$\hat{z}_t^- \sim N(H \cdot \hat{\mu}_t^-, H \cdot \hat{\Sigma}_t^- \cdot H^T + R)$$

基于当前预测 $\hat{S}_t^-$，我们对 $t$ 时刻观测值的分布的估计。

$H$：latent space → observed space

$R$：$\hat{y}_t$ 的不确定性。

$$\hat{y}_t = H \cdot \hat{\mu}_t^-$$

$\hat{y}_t$ 的不确定性同样由协方差阵来衡量：

$$H \cdot \hat{\Sigma}_t^- \cdot H^T + R$$

$R$ 表示在空间进行投影时，对预测的观测值 $\hat{y}_t$ 增加的不确定性程度。

$K_t$ 用来权衡是之前预测的更重要，还是由 $t$ 时刻观测提供的新的信息更重要。

$K_t$ 与 $R$ 呈负相关。

$R$↑，$y_t$ 越不确定，可信度↓，$(y_t - H\hat{\mu}_t^-)$ 权重 $K_t$↓，small correction.

$R$↓，$y_t$ 越确定，可信度↑，$(y_t - H\hat{\mu}_t^-)$ 权重 $K_t$↑，big correction.

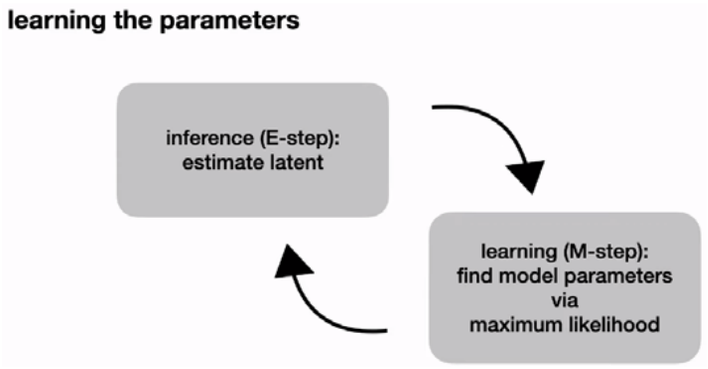
**SMOOTHING STEP:**

we go backwards from the last time point to the first. we're smoothing over the latent estimates that we got from the filtering. So, we're not using the data anymore directly, but we're relying on the filter estimates.

# 4. Kalman 滤波器的 parameter learning

For this we would use the expectation maximization algorithm. The expectation maximization algorithm is iterative and alternates between estimating the latent, in the E-step, and learning the parameters of the model in the M-step using maximum likelihood. Within each iteration we converge to a better solution here.

- The E-step is essentially what you have already know above.
- The M-step which we will not implement here, but you can find more details about it again in the appendix of the notebook.




# 5. Kalman 滤波器应用

**Application:** how we can use Kalman filtering to smooth potentially noisy eye-tracking gaze data? what in an image draws our attention (visual saliency)?

Subjects were asked to stare at a screen, fixate at the center of a screen and an image like this appeared and they were given a few seconds to just sort of freely gaze around the image as whatever drew their attention. And then their gaze was tracked over time using an eye tracking device and it looks like this.

And we have a series of samples that were taken at regular intervals as the subject gazed around the image. so, what we want to look at is how can we take this data and find a smooth estimate to remove potential sources of noise. in an eye tracking experiments noise can come from just changes an ambient light, the subjects head moved too far away from the initial calibration point, they blinked, things like that.

## Pykalman:

Setting up a pykalman filter object

```
kf = pykalman.KalmanFilter(
  n_dim_state=n_dim_state,
  n_dim_obs=n_dim_obs,
  em_vars=['transition_matrices', 'transition_covariance',
           'observation_matrices', 'observation_covariance']
)
```

specify

- the hidden state (n_dim_state), which will be 2, namely, x and y coordinates of where the eye gaze is.
- The dimensionality of our observations, which in this case is also 2 (n_dim_obs), the actual observed x and y coordinates of the pixels.

Then we tell it what parameters we want it to fit using the expectation maximization algorithm. So in this case:

- the transition matrices F,
- the transition covariance Q,
- the observation matrices H, and
- the observation covariance R.

## Estimating parameters with EM

```
data = et_subjects[subject_id][image_id]

# Provide the initial states
kf.initial_state_mean = data[0]
kf.initial_state_covariance = 0.1*np.eye(n_dim_state)

# Estimate the parameters from data using the EM algorithm
kf.em(data)
```

we take our data and we need to give

- an initial state mean value, which we're going to take as the center of the image because we know subjects we're all looking at the center of the image when they started. And that's the first data point in our data stream.
- And then we're going to give a small amount of noise as our initial covariance estimate.

Finally, we just say 'em' on the data. the pykalman library will then run the expectation-maximization algorithm on our data set to try to fit those parameters that we specified.

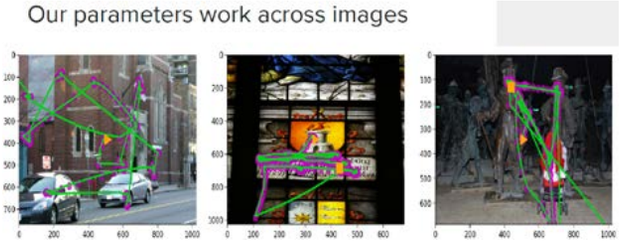
$$F = \begin{bmatrix} 1.004 & -0.01 \\ 0.005 & 0.989 \end{bmatrix} \quad Q = \begin{bmatrix} 278.016 & 219.292 \\ 219.292 & 389.774 \end{bmatrix}$$

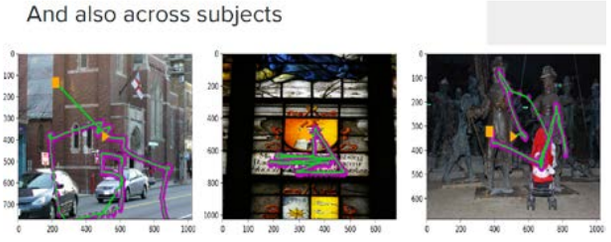
$$H = \begin{bmatrix} 0.999 & 0.003 \\ -0.004 & 1.01 \end{bmatrix} \quad R = \begin{bmatrix} 26.026 & 19.596 \\ 19.596 & 26.745 \end{bmatrix}$$

- For the transition matrices F and the observation matrices H, the model fit is basically the identity matrix. we can interpret that is that these are largely independent.
- The noise covariances though are pretty large for this subject's data. Because these values are in pixels. So that's why they might look large especially in this case.

So now that we have trained parameters, we can go back and apply our filtering or Kalman smoothing to the data to see if we can get a better true estimate of that continuous gaze sweep.

既然已经训练出了参数，我们在同一个 subject 的数据上测试一下：




同样，我们再用这组参数在不同的参与者的数据上测试一下：