AI mid-term

Agent's Performance Measure:

Rationality:

1. criterion of success
2. prior knowledge
3. the actions the agent can perform
4. Agents percepts sequence to date

Simple Reflex agents

This agent function only succeeds when the environment is fully observable

Model Based Reflex agents

Model-based reflex agent
這種agent對世界環境有其猜測，認為環境符合某個model（也包括agent參與後如何改變世界的考量），maintain internal states，所以他可以在partial observable的條件下運作。

model通常是依賴percerpt history。

Goal Based agent

這種agent把未來考慮進去，所謂未來就是action是否朝向maximize goal的方向前進？

utility based agent

goal可能可以通過不同的途徑抵達，所以我們需要找到一個能讓agent最滿足的途徑，透過utility function來評估。

```
A rational utility-based agent chooses the action that maximizes the expected utili
```

CH3 solving Problem by searching

Uninformed Search

BFS (Breadth First Search)

廣度優先搜尋
即廣義的Level-Order Traversal
會用到Recursive Algorithm

```
計算量太大：假設每個node會產生b個子node，且solution在depth(level)d，則會產生b+b^2+b^3+..

It is extremely inefficient and is not ideal for large data structures. Once the al
```

Uinformed cost search

一定會找到optimal solution，且運算也較BFS快。

Depth first search (may not find the best solution)

graph traversal algorithm

```
優點: 比BFS較快
缺點：找的到goal，但不是optimal solution(可能Path cost很大)
```

Informed Search (Heuristics)

Greedy best first search

利用evaluation function f(n) for each node，已經知道某些資訊(estimate of cost from n to the goal)，每個node expand的原則就是從max or min f(n)的node expand下去

```
greedy best 不一定會找到optimal解，因為它就是只看目前狀態與goal之間的cost做決定，沒有考慮
且可能會遇到Incomplete的問題(找不到最佳解)
```

1. 可能會stuck in loops
2. time 有減少很多
3. space：keeps all nodes in memory
4. optimal：No

A* Search

精隨在於 選擇一個node並繼續expand的條件由原本僅考慮現在state至goal之間的cost，改為考慮cost to reach the node g(n) & the cost from the node to the goal h(n)，所以f(n) = g(n) + h(n)

等同於uniform-cost-search，但f(n)改為以上式子

```
有 admissible heuristic，則能找到最佳解。
解釋：the one that never overestimates the cost to reach the goal.
如: 城市間直線距離

有 Consistency : h(n) <= c(n, a, n') + h(n')
```

CH4 Beyond Classical Searching

Local Search Algo

```
概念：Path does not matter. Objective function does
```

1. very little memory usage(不用紀錄所有走過的path) - 省記憶體，速度快
2. 僅從current node 往neighbors下去尋找(條件就是max objective func)
3. find reasonal solutions in continuoius spaces

```
heuristic：the path cost to the goal is known，and we are making decisions based on
```

Hill Climbing Search

```
Greedy local search : 只找那最棒的鄰居，常常效果不錯
```

1. 問題: 卡local min & max
2. 很快就找到local max or min，等在短時間內就可以找到local已經OK

1. Stochastic hill -> 隨便選neighbor(比較不會卡Local)
2. first choice -> 找到第一個比current state好的就選(省時間)
3. random restart hill climbing -> 不同的initial state下去找

模擬退火法

隨便找鄰居，如果比較好直接取代，如果沒有的話會以固定機率取代。
如何決定機率? --> 溫度

ch13 量化不確定性

```
agent 面對的問題中，很多時候是不能100%確定狀態的，或者當前狀態不能直接imply結果，其中的不確

Decision Theory = probability theroy + utility theory
```

每個agent做出action的條件還是最佳化utility func，只不過utility以機率的方式描述(因為環境不確定性等)
當進行決策時，agent需要針對當前的條件，推斷事件發生的條件機率。

```
P(X|y) = P(X, y) // P(y), P(X|y) = a(P(X,y))
就算不知道P(y)，也可以算出P(X|y)，只要我們有data!
```

Bayes’ Rule

```
P(x|y) = P(y|x) * P(x) // P(y)
Sequence labeling -> P(labels | words) = P(words | labels) * P(labels) // P(words)
```

問題: 給定一組字下，想知道全部可能的Labels下，發生機率最高的那組label違和。此問題可以利用 p(x|y) ~ p(y|x)*p(x)

即 P(labels | words) ~ P(words | labels) * P(labels)

如何從多個evidence，推測單一原因?

這時候就會需要假設變數間條件獨立：

```
P(label|word_a, word_b) = a(P(word_a, word_b|label)) * P(label)，其中P(word_a, word_
```

補充: 假設word_a 與 Label獨立，則P(label | word_a, word_b)可簡化為 P(label | word_b)

故在假設變數間獨立的情況下 (Naive Bayes model)

$$\mathbf{P}(Cause, Effect_1, \ldots, Effect_n) = \mathbf{P}(Cause) \prod_i \mathbf{P}(Effect_i \mid Cause)$$

CH14 Probability Reasoning (機率推論)

Bayesian Network

價值: 用graph的方式表達變數間條件獨立的關係

Representing Knowledge

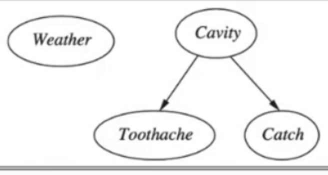
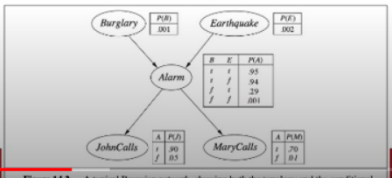


Figure 14.1 A simple Bayesian network in which *Weather* is independent of the other three variables and *Toothache* and *Catch* are conditionally independent, given *Cavity*.

用法:

1. 表達joint proba - 用於建構與理解conditional proba 計算

$$\begin{aligned} P(j,m,a,\neg b,\neg e) &= P(j\,|\,a)P(m\,|\,a)P(a\,|\,\neg b \wedge \neg e)P(\neg b)P(\neg e) \\ &= 0.90 \times 0.70 \times 0.001 \times 0.999 \times 0.998 = 0.000628 \end{aligned}$$



2. 解決chain rule大量計算的問題

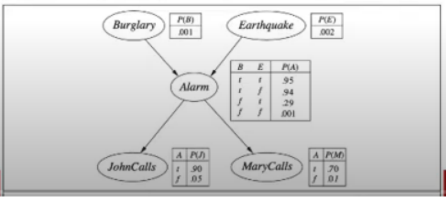
$$\begin{aligned} P(x_1,\ldots,x_n) &= P(x_n\,|\,x_{n-1},\ldots,x_1)P(x_{n-1}\,|\,x_{n-2},\ldots,x_1)\,\cdots\,P(x_2\,|\,x_1)P(x_1) \\ &= \prod_{i=1}^{n} P(x_i\,|\,x_{i-1},\ldots,x_1)\,. \end{aligned}$$

- This identity is called the **chain rule.**

3. 每個變數與其非child node，在其parent node條件下，皆獨立。
   Network 建構的方法有很多種 取決於方法間機率取得準確性與容易度決定

Markov blanket

- Another important independence property: a node is conditionally independent of all other nodes in the network, given its parents, children, and children's parents—that is, given its **Markov blanket**.
- For example, *Burglary* is independent of *JohnCalls* and *MaryCalls*, given *Alarm* and *Earthquake*. This property is illustrated in Figure 14.4(b).



給定某node的parents, childs, 及 childs parents, 該node與其他node有cond independent的關係。

markov assumption -> 某一變數僅其相鄰的變數有關係，其他在相鄰變數給定情況下，無關。

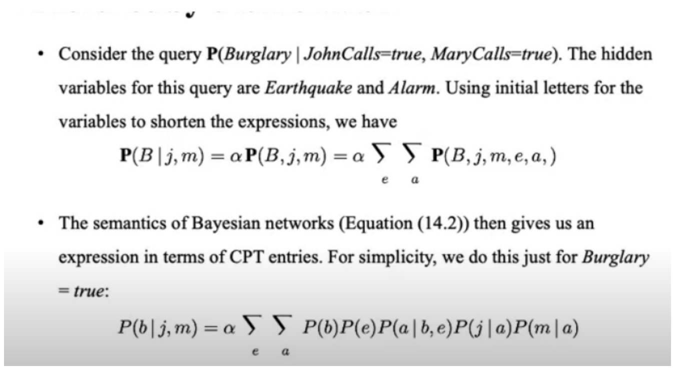
若變數為連續，如何使用bayesian nets?

```
一個變數發生的條件機率，可以利用gaussian distribution來Model。以參數化母數u, sigma^2
```

如果node & parent node皆為連續，可用高斯分布Model他們的機率關係

如果node是binary & parent node為連續，可將parent變數分布取積分變成累積pdf，則取一個threshold就可以決定current node的0,1(Probit distribution)，也可以用logit distribution。

exact inference



- Consider the query **P**(*Burglary* | *JohnCalls=true*, *MaryCalls=true*). The hidden variables for this query are *Earthquake* and *Alarm*. Using initial letters for the variables to shorten the expressions, we have

$$\mathbf{P}(B \mid j, m) = \alpha \mathbf{P}(B, j, m) = \alpha \sum_e \sum_a \mathbf{P}(B, j, m, e, a,)$$

- The semantics of Bayesian networks (Equation (14.2)) then gives us an expression in terms of CPT entries. For simplicity, we do this just for *Burglary = true*:

$$P(b \mid j, m) = \alpha \sum_e \sum_a P(b)P(e)P(a \mid b, e)P(j \mid a)P(m \mid a)$$

The variable elimination algorithm

```
解決機率重複計算的問題
```

Approximate inference

```
若遇到multiply connected network 用近似的方式推估某query variable的條件機率。
```

當條件機率計算非常複雜時，可以用randomized sampling algo，如Monte Carlo algo的方式得到query variable的條件機率

Monte carlo 的方式

Direct sampling
根據貝氏網路順序抽樣
為何要有貝氏網路? -> 方便計算joint probability!!!
但是當variable很多，joint probability很難算
但如果sample很多次(n足夠大)，則抽樣Joint prob的機率會逼近真實P(x1,x2,...,xn)的聯合機率(estimate is consistent)。

$$\lim_{N\to\infty}\frac{N_{PS}(x_1,\ldots,x_n)}{N}=S_{PS}(x_1,\ldots,x_n)=P(x_1,\ldots,x_n)$$

Rejectio sampling ??

缺點，當evidence出現的機率很低的時候，sampling會需要花很多時間才能完成

Likelihood weighting

not every event is equal，每個事件組合在抽樣出現的機率會經過weight修正

Markov Chain Monte Carlo algo

不是一直重新抽樣，而是從之前抽樣的結果稍微改變而產生新的sample。

重點概念就是，從前一次的抽樣結果，稍微改一下內容，產生新的抽樣結果

Gibbs sampling in Bayesian networks

核心概念: 一次變一個variable，從上次結果變成下一個sample，是與direct sample不同的地方。 可節省inference推論所需時間

針對X,Y,Z, E這四個變數，假設query為P(x|y,z)，則抽樣一次後將y,z的觀察值固定，僅改變non evidence的變數抽樣結果(改變X,E)，分別討論:

1. X 抽樣: 給定X的markov blanket(child, parents, childs parents)的值下，去抽樣X變數。
2.

```
MCMC的重點在於，sampling的依據是根據目前的狀態下，再去做sampling，得到query prob，而非重
```

CH15 根據時間的機率推論。

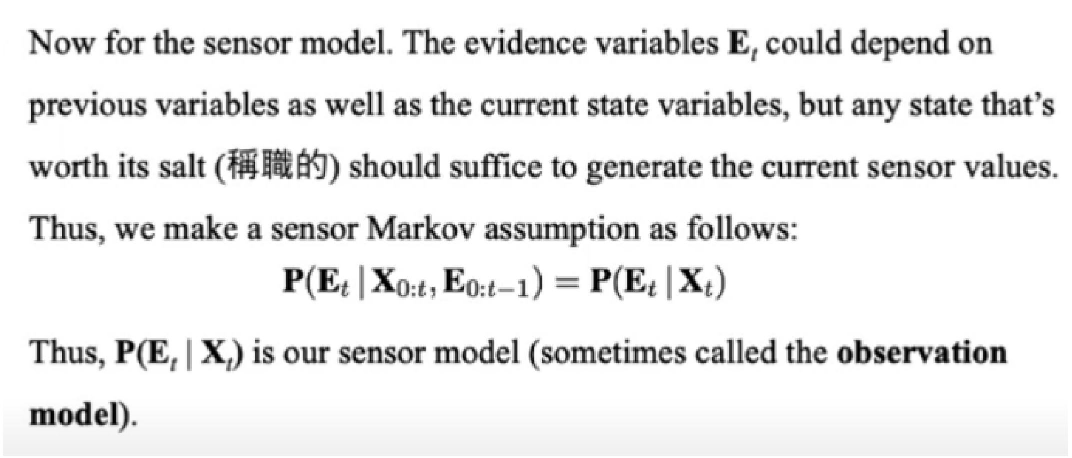
當機率隨時間會產生改變(變數隨時間變化)，就會牽涉時間上的機率推論。

定義問題:

Transition model ： 給定之前的state，下一個state的機率。
問題來了，P(Xt|X0...Xt-1)中，條件的部分需要假設，才方便計算。
這個假設就會利用markov assumption -> 假設時間t的state只跟t-1的state有關。

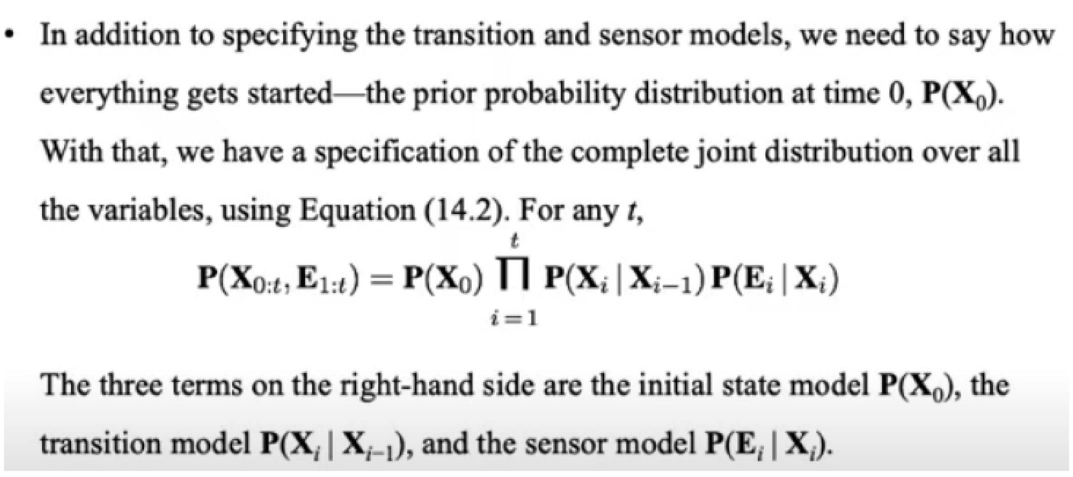
first order Markov process, transition model -> P(Xt|Xt-1)
並假設time series是平穩 (transition model 和絕對t無關)。

transition & sensor model 假設:


Now for the sensor model. The evidence variables $\mathbf{E}_t$ could depend on previous variables as well as the current state variables, but any state that's worth its salt (稱職的) should suffice to generate the current sensor values. Thus, we make a sensor Markov assumption as follows:

$$\mathbf{P}(\mathbf{E}_t \mid \mathbf{X}_{0:t}, \mathbf{E}_{0:t-1}) = \mathbf{P}(\mathbf{E}_t \mid \mathbf{X}_t)$$

Thus, $\mathbf{P}(\mathbf{E}_t \mid \mathbf{X}_t)$ is our sensor model (sometimes called the **observation model**).



- In addition to specifying the transition and sensor models, we need to say how everything gets started—the prior probability distribution at time 0, $\mathbf{P}(\mathbf{X}_0)$. With that, we have a specification of the complete joint distribution over all the variables, using Equation (14.2). For any t,

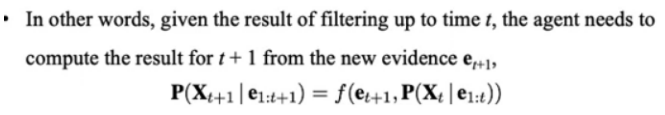
$$\mathbf{P}(\mathbf{X}_{0:t}, \mathbf{E}_{1:t}) = \mathbf{P}(\mathbf{X}_0) \prod_{i=1}^{t} \mathbf{P}(\mathbf{X}_i \mid \mathbf{X}_{i-1}) \mathbf{P}(\mathbf{E}_i \mid \mathbf{X}_i)$$

The three terms on the right-hand side are the initial state model $\mathbf{P}(\mathbf{X}_0)$, the transition model $\mathbf{P}(\mathbf{X}_i \mid \mathbf{X}_{i-1})$, and the sensor model $\mathbf{P}(\mathbf{E}_i \mid \mathbf{X}_i)$.

Filtering

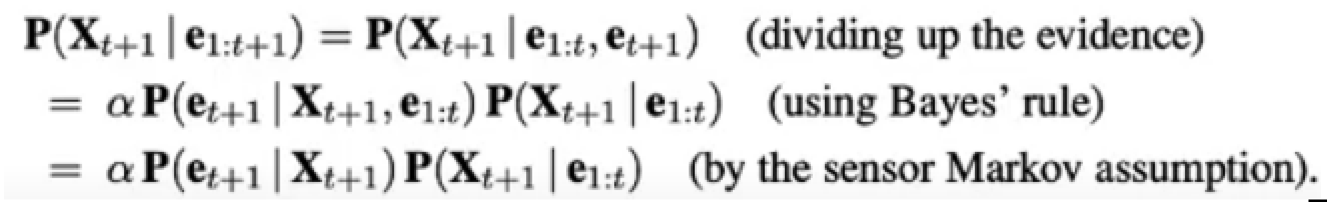
(state esimation) filtering的定義: 給定t0 至t的觀察(evidence)，預估時間t的狀態。P(Xt | e1:t) -> 一直從頭疊代

prediction: P(Xt+k | e1:t)

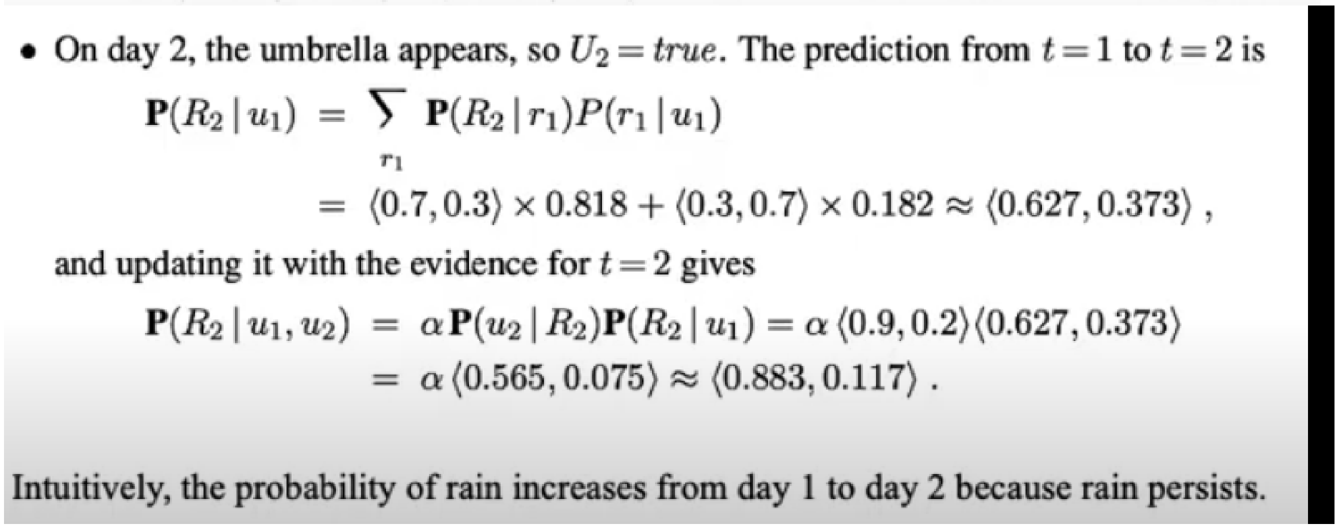


- In other words, given the result of filtering up to time t, the agent needs to compute the result for $t+1$ from the new evidence $\mathbf{e}_{t+1}$,

$$\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1}) = f(\mathbf{e}_{t+1}, \mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t}))$$



$$\begin{aligned}\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1}) &= \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t}, \mathbf{e}_{t+1}) \quad \text{(dividing up the evidence)}\\ &= \alpha\, \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}, \mathbf{e}_{1:t})\, \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t}) \quad \text{(using Bayes' rule)}\\ &= \alpha\, \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1})\, \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t}) \quad \text{(by the sensor Markov assumption).}\end{aligned}$$



- On day 2, the umbrella appears, so $U_2 = true$. The prediction from $t=1$ to $t=2$ is

$$\begin{aligned}\mathbf{P}(R_2 \mid u_1) &= \sum_{r_1} \mathbf{P}(R_2 \mid r_1) P(r_1 \mid u_1)\\ &= \langle 0.7, 0.3 \rangle \times 0.818 + \langle 0.3, 0.7 \rangle \times 0.182 \approx \langle 0.627, 0.373 \rangle\ ,\end{aligned}$$

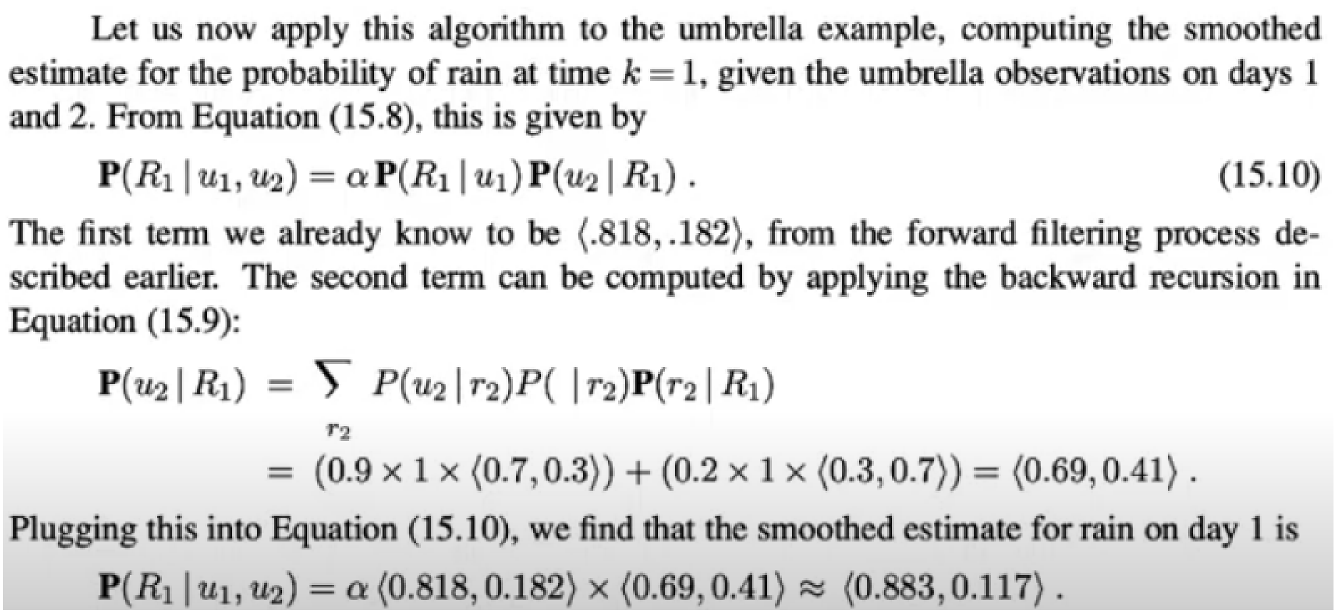
and updating it with the evidence for $t=2$ gives

$$\begin{aligned}\mathbf{P}(R_2 \mid u_1, u_2) &= \alpha\, \mathbf{P}(u_2 \mid R_2) \mathbf{P}(R_2 \mid u_1) = \alpha \langle 0.9, 0.2 \rangle \langle 0.627, 0.373 \rangle\\ &= \alpha \langle 0.565, 0.075 \rangle \approx \langle 0.883, 0.117 \rangle\ .\end{aligned}$$

Intuitively, the probability of rain increases from day 1 to day 2 because rain persists.

Smoothing: P(Xk | e1:t) for (1<k<t)

同樣利用recursive方式用backward(從t到k+1)，從後面推回來。



Let us now apply this algorithm to the umbrella example, computing the smoothed estimate for the probability of rain at time $k=1$, given the umbrella observations on days 1 and 2. From Equation (15.8), this is given by

$$\mathbf{P}(R_1 \mid u_1, u_2) = \alpha\,\mathbf{P}(R_1 \mid u_1)\,\mathbf{P}(u_2 \mid R_1)\,. \tag{15.10}$$

The first term we already know to be $\langle .818, .182\rangle$, from the forward filtering process described earlier. The second term can be computed by applying the backward recursion in Equation (15.9):

$$\begin{aligned}\mathbf{P}(u_2 \mid R_1) &= \sum_{r_2} P(u_2 \mid r_2)P(\ \mid r_2)\mathbf{P}(r_2 \mid R_1)\\ &= (0.9 \times 1 \times \langle 0.7, 0.3\rangle) + (0.2 \times 1 \times \langle 0.3, 0.7\rangle) = \langle 0.69, 0.41\rangle\,.\end{aligned}$$

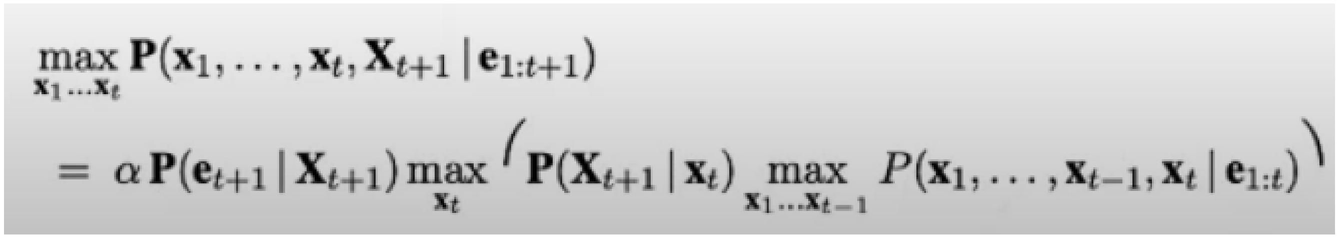
Plugging this into Equation (15.10), we find that the smoothed estimate for rain on day 1 is

$$\mathbf{P}(R_1 \mid u_1, u_2) = \alpha\,\langle 0.818, 0.182\rangle \times \langle 0.69, 0.41\rangle \approx \langle 0.883, 0.117\rangle\,.$$

稱為forward, backward algo

Most likely explanation : given sequence e1:t, 計算X1:t的機率。

利用smoothing的方式，給定e1...et，可以推測第k天的state(1<k<t)，則依照這個概念可以計算P(X1:t | e1:t)



$$\begin{aligned}&\max_{\mathbf{x}_1\ldots\mathbf{x}_t} \mathbf{P}(\mathbf{x}_1, \ldots, \mathbf{x}_t, \mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1})\\ &= \alpha\,\mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}) \max_{\mathbf{x}_t}\left(\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t) \max_{\mathbf{x}_1\ldots\mathbf{x}_{t-1}} P(\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}, \mathbf{x}_t \mid \mathbf{e}_{1:t})\right)\end{aligned}$$

以上的計算法為 Viterbi algorithm

Viterbi 三大概念

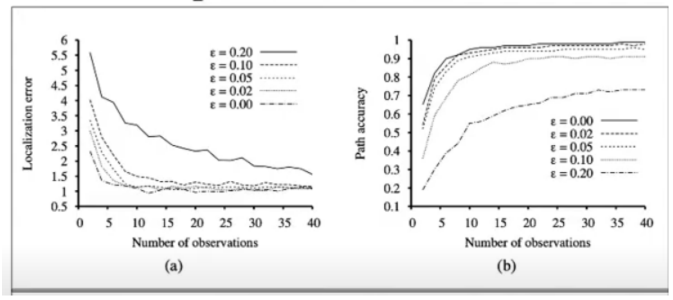
1. recursive
2. transition model
3. sensor model

Hidden Markov Models

```
1. hidden : 只能看到evidence，只能推估states。
2. markov : state 只和附近states有關
3. state is single discrete random variable
```

agent除了可用filter預估目前位置外，也能利用smoothing方法加上viterbi 去找出最佳可能路徑，估算目前距離。則應過越多序列後，儘管error rate高，準確度也能到不錯的水準

HMM example: Localization



Kalman Filters 演算法

假設states為連續變數，則HMM不適用，需使用kalman filters來判斷hidden state

假設transition model及sensor model 皆follow linear Gaussian distribution

下一個連續變數state會是目前連續變數state的高斯函數值+高斯noise

```
為何選擇高斯函數? -> closed under standard 貝氏網路
高斯函數中變數做計算完後仍然為高斯分布
```

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

```
kalman filtering 的限制: 變數的模型僅限制於線性關係，若實際狀況為極度非線性，則不適用。
```

若有多個objects

Training Tips

影響訓練accu表現的因素

1. loss function (cross entropy vs MSE)
2. mini - batch (訓練更快，更容易找到最佳解)
3. activation function (ReLU解決sigmoid遇到的gradient vanishing問題)
4. adaptive learning rate (adagrad針對梯度下降的幅度做修正)
5. Momentum (避免卡在local min or saddle pnt, or plateau)
6. model structure

影響測試accu因素，及解決

1. Early Stopping
2. Regularization (weight decay)
3. Dropout
4. simpler network structure
5. data augmentation

CNN

Why CNN?

```
沒有cnn的話 FC的參數太多。
將圖片特徵擷取，且不針對位置而分類。
sumsampling (resize)，不用整張圖，後依然可以分類。
```

Convolution layer類似dropout功能，僅針對特定pixel進行neuron計算
，減少參數使用，也減少overfitting。

RNN

RNN 在訓練時因為neuron之間將input連乘的關係 gradient經常有巨大變動，training效果不好。但是LSTM的output gate機制能解決gradient vanishing的問題。