ECE209AS (Winter 2021)

Lecture 2: Making Inferences from Sensor Data (The Pre Deep Learning Era)

Mani Srivastava
mbs@ucla.edu
Networked & Embedded Systems Lab
ECE & CS Departments
UCLA

Sensor Measurement ≠ State

- Challenge: sensors themselves do not provide environment state
  - E.g. sensors do not say “There is Mr. Smith sitting on a chair and wearing a black suit”

- Rather, sensors may tell
  - Light level
  - Color
  - Whether it is touching something in an area
  - Sound level
  - Distance to nearest object
  - Etc.

- Sensors measure physical quantities
  - Need to be processed to be useful

- Same physical quantity may be measurable by different sensors
  - Can help improve accuracy in the presence of error and noise

Physical Property	→	Sensing Technology
Contact	→	bump, switch
Distance	→	ultrasound, radar, IR
Light Level	→	photocells, cameras
Sound Level	→	microphones
Strain	→	strain guages
Rotation	→	encoders, potentiometers
Acceleration	→	accelerometers, gyroscopes
Magnetism	→	compasses
Smell	→	chemical sensors
Temperature	→	thermal, IR
Inclination	→	inclinometers, gyroscopes
Pressure	→	pressure guages
Altitude	→	altimeters

Some sensors and the information they measure

Example: Measuring distance to an object

- Ultrasound
  - time of flight
- Infra-red
  - return signal intensity
- Two cameras
  - stereo
- Single camera
  - using perspective + assumption about environment structure
- Laser + fixed camera
  - triangulate distance
- Laser-based structured lighting (overlay grid pattern) + fixed camera
  - distance from distortion in pattern
- Others?

Example: Detecting people

- Use a camera?
  - Camera/vision is a very powerful modality
    - Intensity, color, texture, shape etc.
  - But very costly in processing

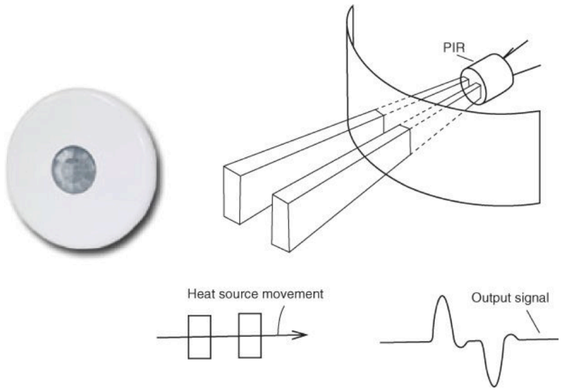
- Other ways: using sensors simpler than vision
  - Temperature: search for temperature ranges corresponding to human body temperature
  - Movement: if everything else is static, movement means people
  - Color: look for colors corresponding to human skin or clothes/uniforms
  - Distance: if an otherwise open distance range becomes blocked, there is likely a human being

- Often simpler sensors are enough
  - E.g. burglar alarm
    - can’t distinguish humans from other animals, but non-human burglars are rare
  - Plus, can help improve accuracy of vision

Activity Detection: Ambient Sensors in the Environment

- Passive Infrared (PIR)
- Magnetic Door/Window
- Temperature, Light, Humidity
- Vibration
- Pressure
- RFID
- Camera
- Microphone
- Electric meter
- Water meter

- Coverage: near-field vs far-field

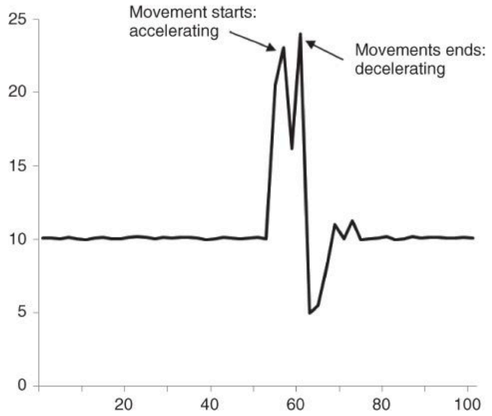


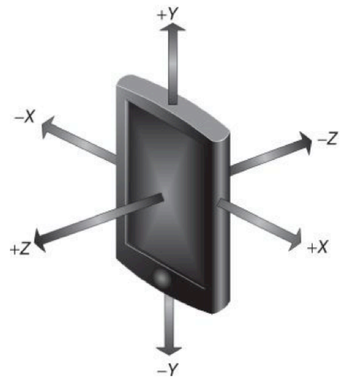
Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

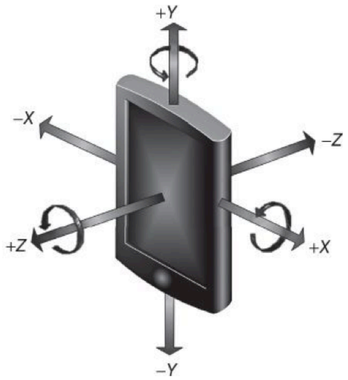
Activity Detection: Wearable Sensors on (or in) Body

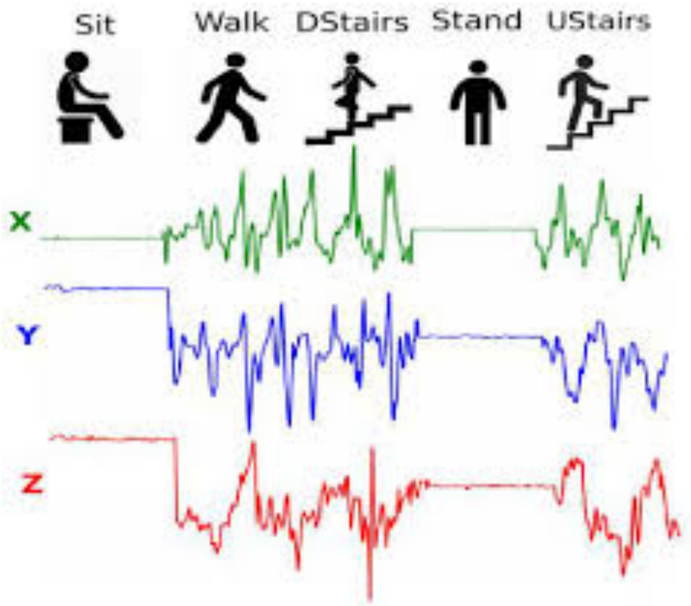
- Inertial
  - Accelerometer
  - Gyroscope
  - Magnetometer

- Physiological
  - Heart Rate
  - Heart Rate Variability
  - Breathing Rate
  - Galvanic Skin Response
  - Blood Pressure
  - EMG, EEG, etc.
  - …



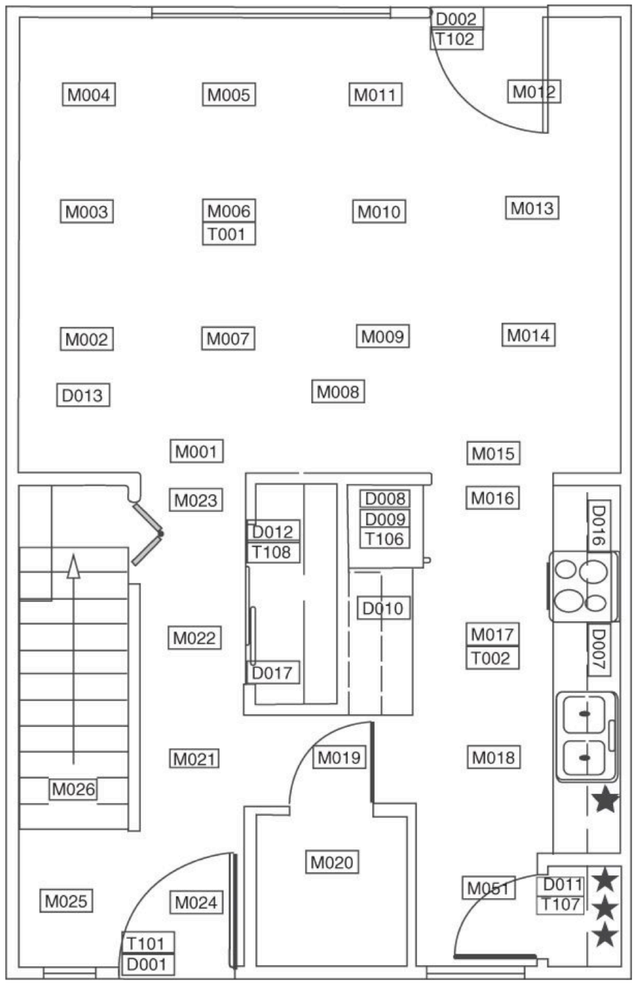


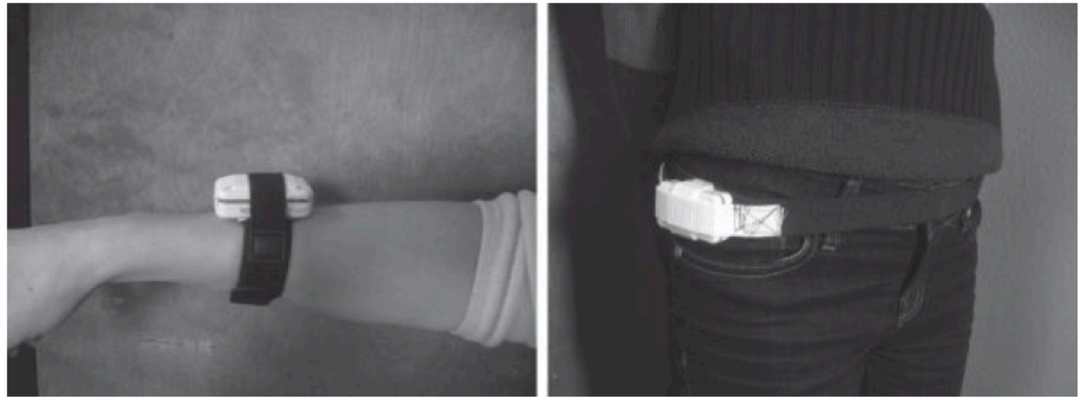


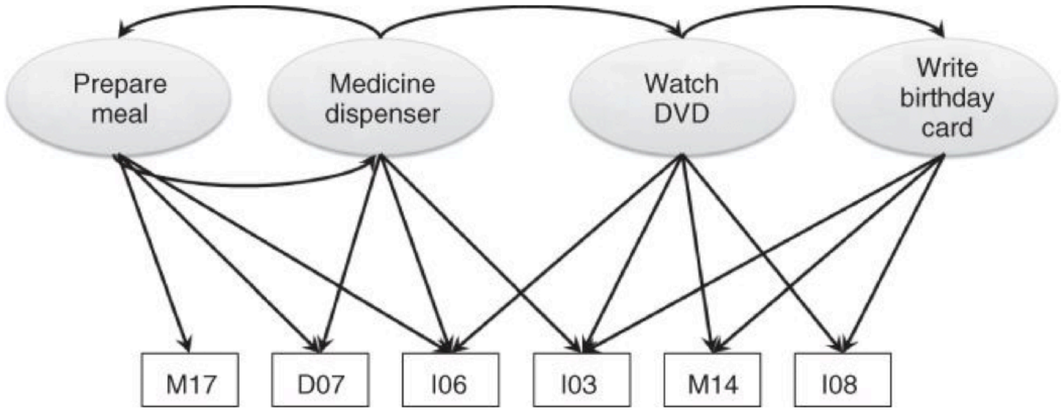


Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

Example Deployment: Ambient + Wearable Sensors



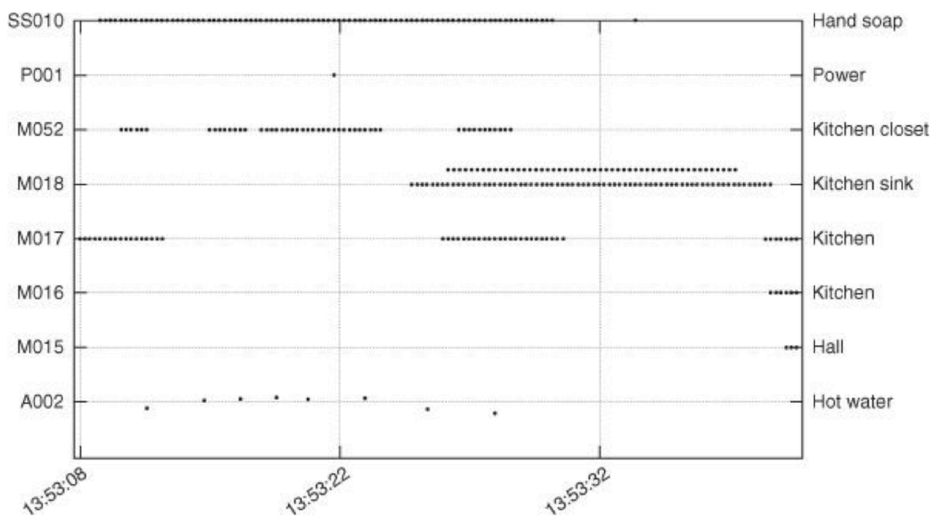


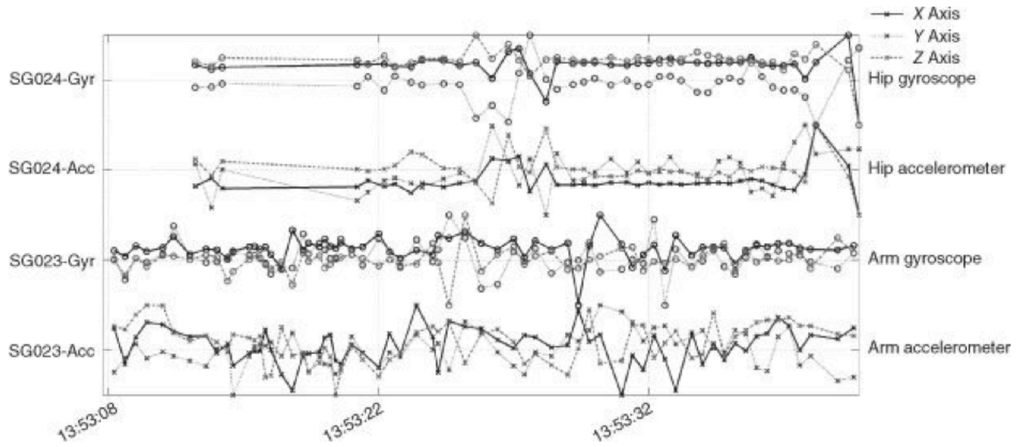


Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

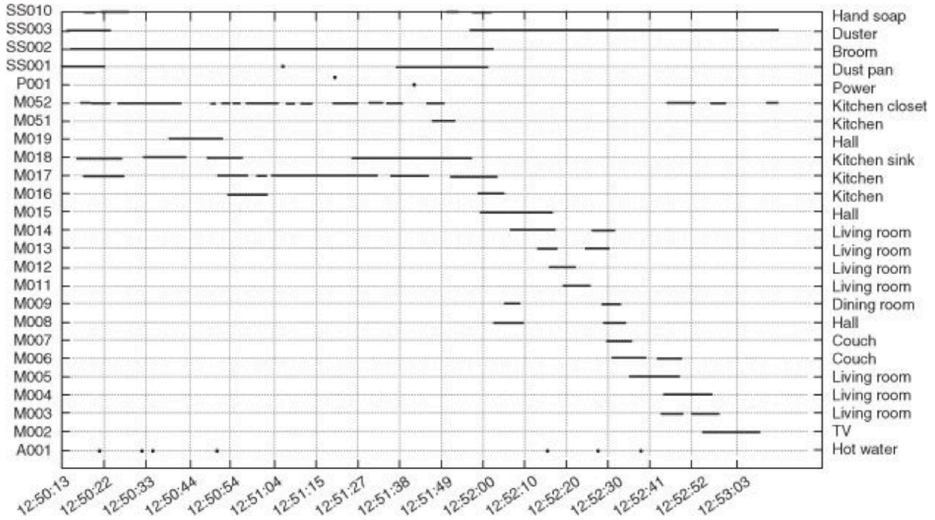
Sensor Measurements for Human Activities

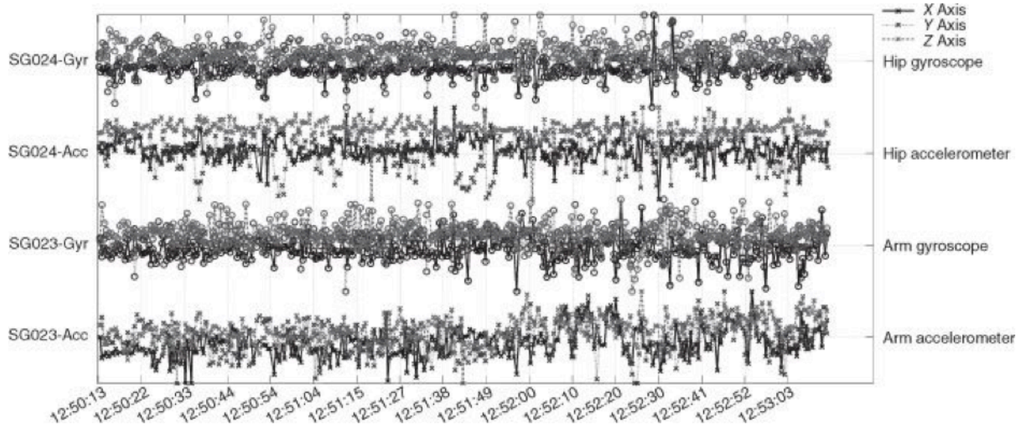
Handwashing





Sweeping





Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

Common Sensor Analytics

- Inferring latent states
- Forecasting
- Anomaly Detection
- Missing Value Imputation
- Clustering

Formalizing the Problem

- We need to track and predict the state of a *dynamic* environment in the presence of *uncertainty*
  - E.g. managing health of a diabetic individual, accounting for time spent in different activities, control HVAC system in a building, etc.
  - Unlike static situations, such as analyzing an X-ray image or diagnosing a faulty engine
- Notation
  - $S_t =$ set of unobservable state variables at time t
    - e.g. *sitting*, *walking*, *running, biking*, etc.
  - $O_t =$ set of observable evidence variables at time t
    - e.g. measurements from *accelerometer*, *gyroscope*, *PPG*, *camera* etc.
    - may be multimodal and from spatially distributed sensors
- Simplification of time
  - time t is discretized into fixed step-size Δ that is identical for S_t and O_t , and steps aligned for S_t and O_t
  - allows treating time t as an index so that time goes as $t = 0,1,2,...$
  - notation: $X_{a:b} = X_a, X_{a+1}, \ldots, X_{b-1}, X_b$
  - later in the course we will relax this regular synchronous sampling

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Physics vs Data Drive Approaches

- Physics of the environment: equations governing the evolution of the states
- Physics of the sensor: equations governing mapping of states to observation

- If these are known, we can solve an inverse problem to get state from obersvations

- But:
  - Physics may not be known at all, or may be complex to compute, or highly uncertain
  - Environment and sensor characteristics may be dynamic

- Alternative: data drive “machine learning” approaches

Wearable Sensor-based Activity Recognition

- User performing activities belonging to a predefined set $A = \{A_i\}_{i=1}^{m}$ where m is the number of activity types
- There is a sequence of sensor reading that captures the activity information $\mathbf{o_{1:n}} = [\mathbf{o}_1, \mathbf{o}_2, \cdots \mathbf{o}_t, \cdots \mathbf{o}_n]$ where $\mathbf{o}_t$ is the sensor reading (in general a vector) at time t
- We need to build a model $\mathscr{F}$ to predict the activity sequence based on sensor reading $\mathbf{s}$

$$\hat{A}_{1:n} = \mathscr{F}(\mathbf{o_{1:n}}), \quad \forall j \; \hat{A}_j \in A$$

while the true activity sequence (ground truth) is denoted as

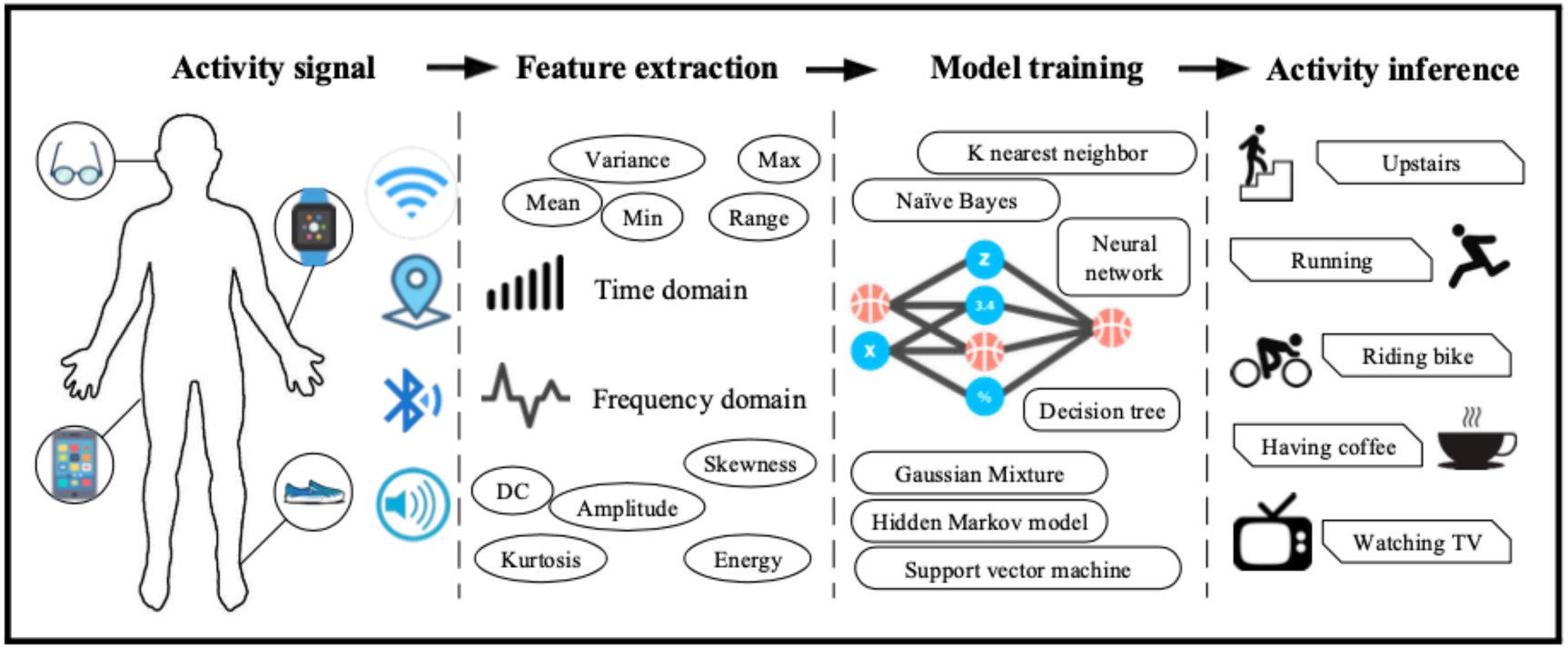
$$A^*_{1:n}, \quad \hat{A}^*_j \in A$$

- Goal is to learn $\mathscr{F}$ by minimizing the discrepancy between predicted activity sequence $\hat{A}_{1:n}$ and the ground truth activity sequence $A^*_{1:n}$
- n governs the algorithmic latency in inferring the activity state

https://arxiv.org/pdf/1707.03502.pdf

Inferring Activity State Using Conventional Patter Recognition

- Goal is to learn $\mathscr{F}$ by minimizing the discrepancy between predicted and the ground truth activity sequence
- Typically, during training a positive loss function $\mathscr{L}(\mathscr{F}(\mathbf{o_{1:n}}), A^*_{1:n})$ is constructed to reflect this discrepancy
- $\mathscr{F}$ does not take $\mathbf{o_{1:n}}$ as an input, but rather assumes there is a projection function Φ that projects each sensor reading $\mathbf{o}_i$ to a d-dimensional feature vector $\Phi(\mathbf{o}_i) \in \mathbb{R}^d$, so that loss function is $\mathscr{L}(\mathscr{F}([\Phi(\mathbf{o}_i)]^n_{i=1}), A^*_{1:n})$



https://arxiv.org/pdf/1707.03502.pdf

Common Simplifications

- $\mathbf{o_{1:n}}$ grouped into windows of some time duration
  - Activity type assumed constant within each window
  - Activity in a window assumed independent of activities in other window
  - Simplifies $\mathcal{F}$ and makes it easier to deal with irregular and missing observations
  - But also creates problems
    - Activity changes misaligned with segment boundaries
    - Activity state may change multiple times within the segment if k is large
    - Ignores temporal correlations between activities in different segments
- The assumption that all sensors contributing to $\mathbf{o}_t$ have the same sample rate is not correct
  - In fact, sampling rate of a sensor may not even be fixed
    - e.g. in Android the sensor sampling rate is an advice to the OS but it is not guaranteed that timestamps will be equally spaces
  - Preprocessing done to address this before extracting features
    - it also takes care of missing data, calibration etc.

Features

- Two types of observations from sensors
  - Sensors that generate time series of periodic numeric measurements of state
    - e.g. periodic sampling of temperature
  - Sensors that generate event reports about change in state
    - e.g. send event when there is motion

- Features computed over windows of observations basically get rid of the temporal aspects of the problem

- Quality of the learned model depends on expressiveness of features

Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

Features

Sample of window events from the sweeping activity

Time	ID	Message
12:50:13.102682	Dustpan	MOVED
12:50:13.149904	HIP	(1019,1921,1520,1850,1800,1898)
12:50:13.367222	ARM	(1584,2402,2318,2040,1838,1736)
12:50:14.128257	HIP	(973,1951,1545,1839,1918,1900)
12:50:14.217971	Duster	MOVED
12:50:14.357487	ARM	(948,1851,1837,1886,1820,2028)
12:50:15.129119	HIP	(1055,1745,1792,1840,1814,1872)
12:50:15.873883	ARM	(926,1867,2119,1751,1853,1863)
12:50:16.134036	HIP	(1047,2137,1487,1819,1594,1992)
12:50:16.235281	Broom	MOVED
12:50:16.36758	ARM	(1055,1677,2182,1705,1613,1862)
12:50:17.001771	M018	ON
12:50:17.11736	HIP	(997,1970,1625,1752,1438,1907)
12:50:17.345406	ARM	(1189,1935,2232,1840,1682,1813)
12:50:18.119112	HIP	(1072,2052,1559,1880,1796,1949)
12:50:18.341054	ARM	(1071,1889,1978,1780,1721,1982)
12:50:18.452039	M017	ON
12:50:18.790212	HandSoap	MOVED
12:50:19.103787	HIP	(1159,2026,1598,1863,1744,1951)
12:50:19.349545	ARM	(1043,1330,1688,1676,1825,1872)
12:50:20.101233	HIP	(986,1966,1573,1838,1727,1921)
12:50:20.334268	ARM	(1007,1674,1700,1702,1868,1916)
12:50:20.741536	HandSoap	STILL
12:50:21.097739	HIP	(981,2074,1527,1853,1724,1930)
12:50:21.342635	ARM	(1208,2346,1888,2217,1234,1483)
12:50:21.895284	Burner	2.8227
12:50:22.090522	HIP	(1149,2003,1701,1857,1615,1949)
12:50:22.339887	ARM	(1134,2594,2088,1894,1845,1872)
12:50:22.412985	HandSoap	MOVED
12:50:22.8739	Dustpan	STILL

Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

Features: Broad Categories

- Characteristics of sensor event sequence
  - time of occurrence
    - time of day, day of week etc.
  - sequence duration
    - 12:50:22.8739-12:50:13.102682

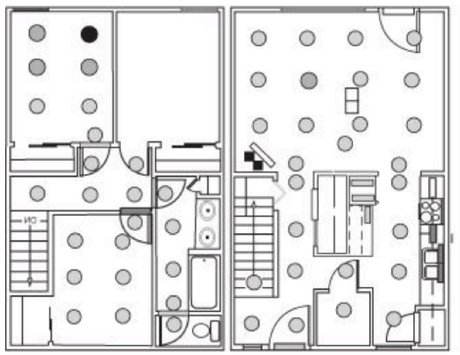
Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

Features: Broad Categories

- Characteristics of sensor event sequence
  - time of occurrence
    - time of day, day of week etc.
  - sequence duration
    - 12:50:22.8739-12:50:13.102682

- Characteristics of discrete sensor values
  - bag of sensors
    - set of events with associated frequencies
  - elapsed time since last event from sensor

Discrete-Based Features			
Sensor Counts		Sensor Elapsed Times	
Dustpan	1	Dustpan	97,712
Duster	1	Duster	86,559
Broom	1	Broom	66,386
Hand soap	2	Hand soap	40,837
M017 (Kitchen)	2	M017 (Kitchen)	44,219
M018 (Sink)	1	M018 (Sink)	58,721
Burner	1	Burner	9,786
All other sensors	0	All other sensors	?



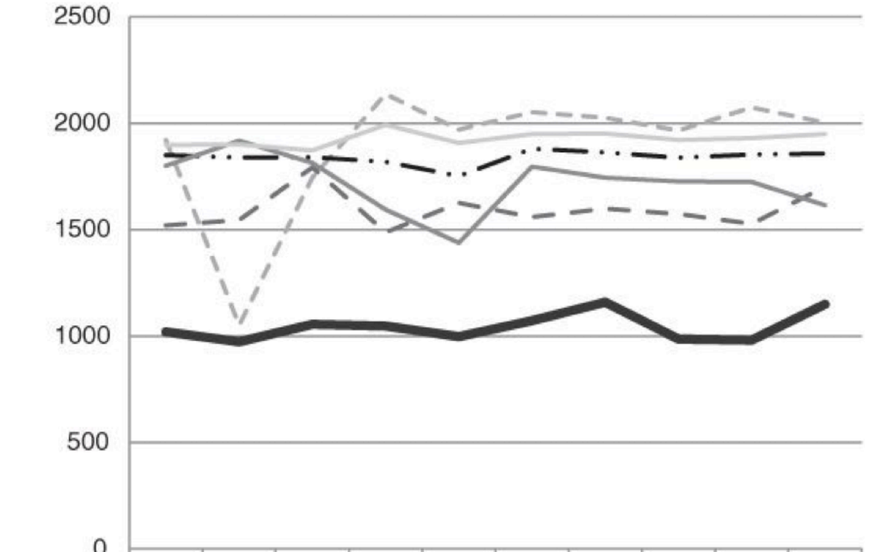
Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

Features: Broad Categories

- Characteristics of sensor event sequence
  - time of occurrence
    - time of day, day of week etc.
  - sequence duration
    - 12:50:22.8739-12:50:13.102682

- Characteristics of discrete sensor values
  - bag of sensors
    - set of events with associated frequencies
  - elapsed time since last event from sensor

- Statistical features over a time window



	1	2	3	4	5	6	7	8	9	10
Ax	1019	973	1055	1047	997	1072	1159	986	981	1149
Ay	1921	1051	1745	2137	1970	2052	2026	1966	2074	2003
Az	1520	1545	1792	1487	1625	1559	1598	1573	1527	1701
Rx	1850	1839	1840	1819	1752	1880	1863	1838	1853	1857
Ry	1800	1918	1814	1594	1438	1796	1744	1727	1724	1615
Rz	1898	1900	1872	1992	1907	1949	1951	1921	1930	1949

Plot of acceleration (Ax,Ay,Az) and rotational velocity (RxRy,Rz) values from the HIP IMU from sweeping activity data sample

Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

Features: Broad Categories

- Characteristics of sensor event sequence
  - time of occurrence
    - time of day, day of week etc.
  - sequence duration
    - 12:50:22.8739-12:50:13.102682

- Characteristics of discrete sensor values
  - bag of sensors
    - set of events with associated frequencies
  - elapsed time since last event from sensor

- Statistical features over a time window

- Activity context features
  - previous activity
  - previous dominant sensor
  - weighted features from previous window

Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

Common Statistical Features: *Temporal*

- Max, Min
- Sum, Mean, Median
- Mean Absolute Deviation, Median Absolute Deviation
- Standard Deviation
- Coefficient of Variation
- Zero Crossings
- Percentiles, Inter-quartile Range
- Square Sum of Percentile Observations
- Histogram
- Skewness (degree of asymmetry of distribution)
- Kurtosis (peakiness of distribution around mean)
- Correlation (across dimensions)
- Autocorrelation
- Signal Energy, Log Energy, Signal Power
- Signal Magnitude Area
- Peak-to-Peak Amplitude
- Time between Peaks

Hip A*x* Continuous-Based Feature Values for Sweeping Activity					
Max	1,159.00	CV	0.09	SqSumPt(80)	8,272,0
Min	973.00	ZC	5	BD(1)	0.5
Sum	10,438.00	PT(20)	981.01	BD(2)	0.3
Mean	1,043.00	PT(50)	1019.01	BD(3)	0.2
Median	1,033.00	PT(80)	1072.01	Skewness	0.83
MeanAbsDev	19.12	IQ	91.00	Kurtosis	−0.48
MedAbsDev	25.50	SqSumPt(20)	1,909,090	Corr(A*x*,A*y*)	0.37
StDev	66.98	SqSumPt(50)	4,913,656	AC_1	−0.20

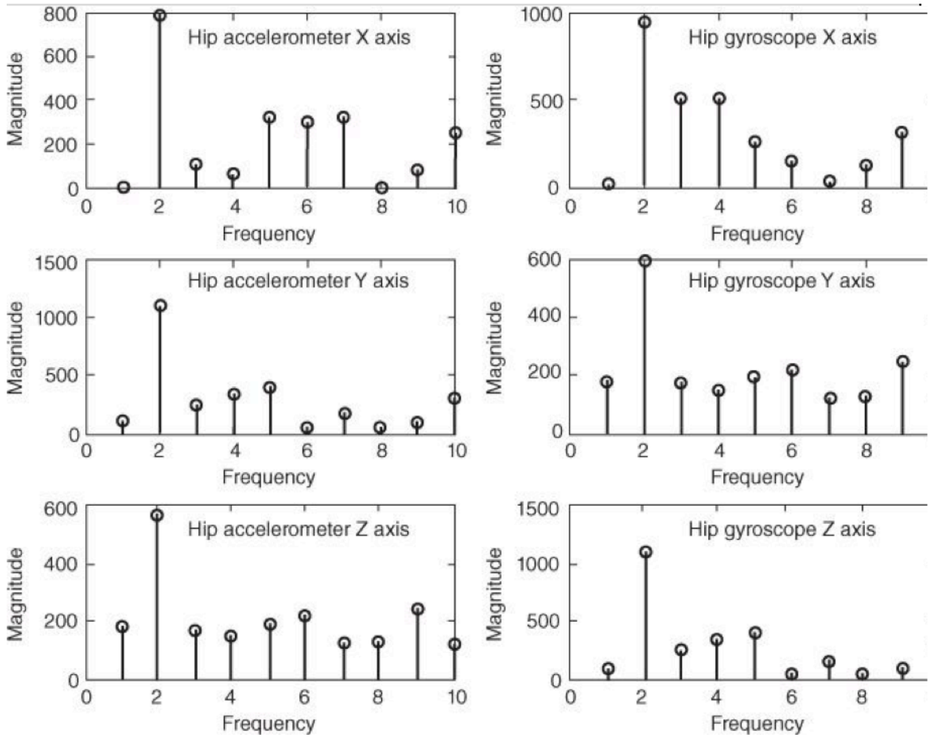
A*x* Signal-Based Features for Sweeping Activity			
Signal energy	10,935,556.0	P2PA	186.0
Log signal energy	200.5	TBPeaks	3.5
Power	1,093,555.6	NumPeaks	3

Common Statistical Features: *Spectral*

- Spectral Centroid
- Spectral Energy
- Spectral Entropy

$$S_C = \frac{\sum_{i=1}^{N} \mathcal{F}(n) \times n}{\sum_{i=1}^{N} \mathcal{F}(n)} \qquad S_E = \sum_{i=1}^{N} \mathcal{F}(n)^2$$

$$S_{EN} = -\sum_{i=1}^{N} \hat{\mathcal{F}}(n) \times \log(\hat{\mathcal{F}}(n)) \qquad \hat{\mathcal{F}}(n) = \frac{\mathcal{F}(n)}{\sum_{i=1}^{N} \mathcal{F}(n)}$$

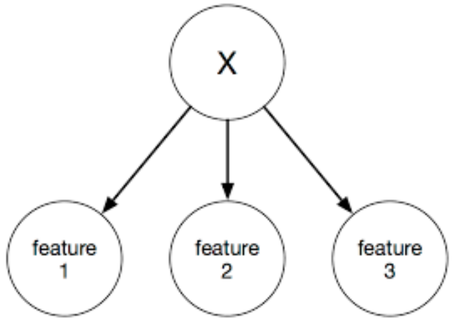


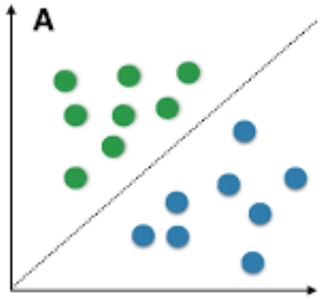
Spectral Features from Hip A*x* Data for the Sweeping Activity Window			
of 10 Events			
Spectral centroid	4.9550	Spectral energy	1007864
Spectral entropy	1.8238	Normalized spectral energy	0.2002

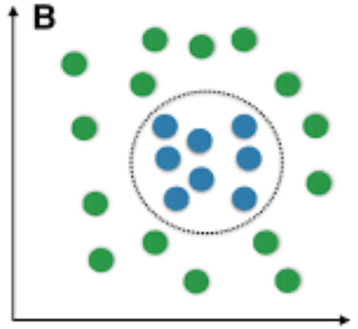
Common forms of $\mathcal{F}$

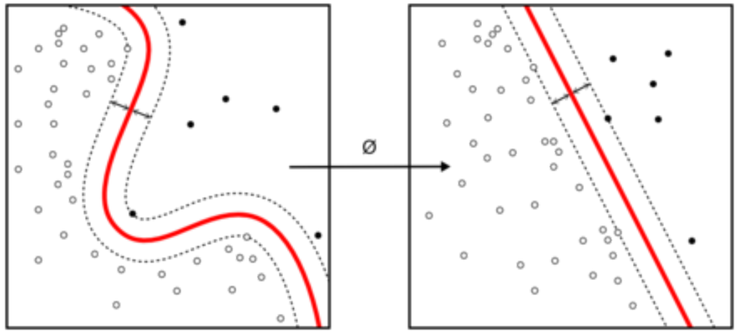
- Naive Bayes
- Linear
- Linear with Non-linear Kernels
- Support Vector Machines
- Decision Tree
- Random Forests
- Bayesian Network
- HMM
- CRF

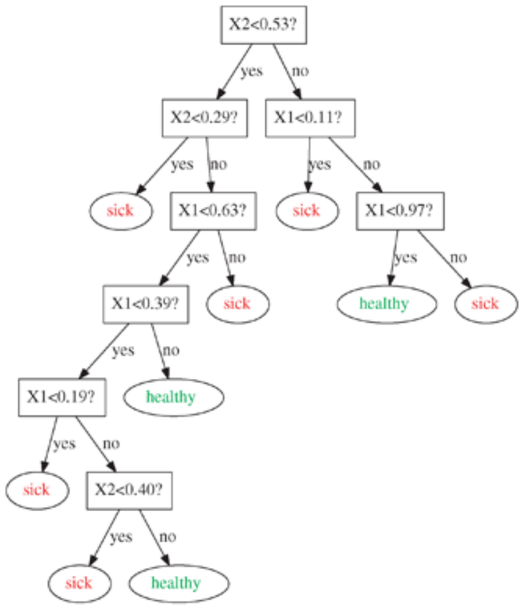
  …
- Being replaced by neural networks in many cases

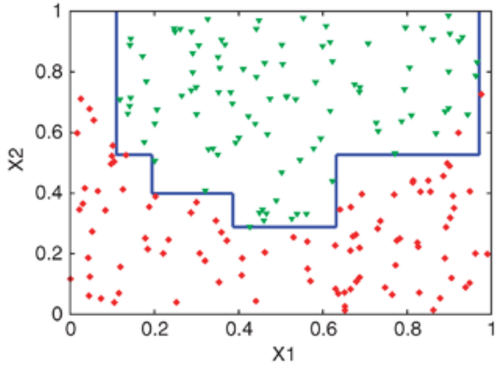


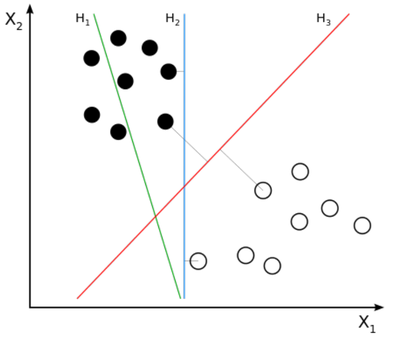












Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

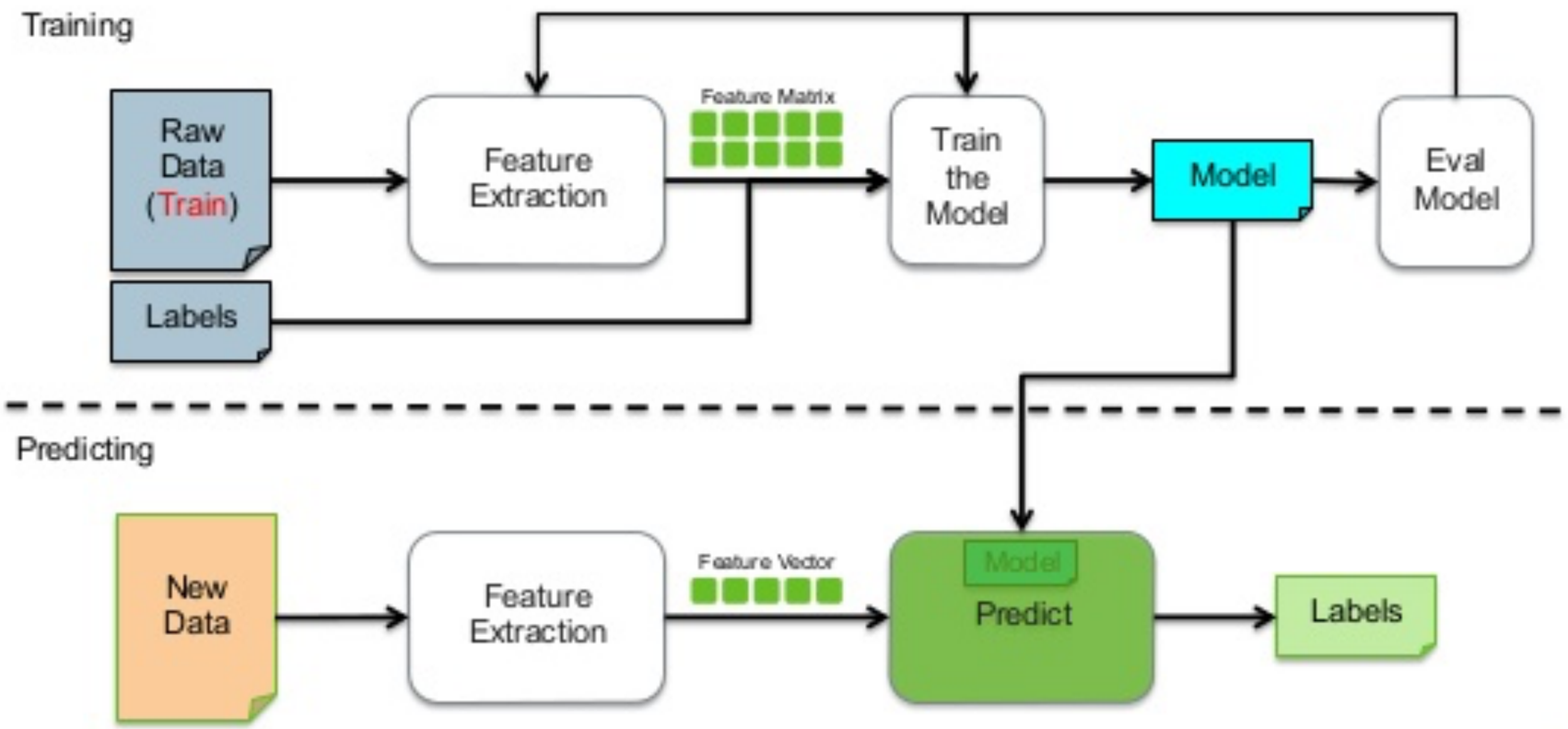
What if there are multiple sensors?

- Complementary vs overlapping information
- Three approaches
  - Data fusion
    - average sensor measurements
    - weighted by trust/quality/variance
  - Feature fusion
    - concatenate feature vectors
    - dimensionality reduction methods in case of overlap
  - Classifier fusion
    - simple voting
    - weighted majority voting
    - highest quality (lowest uncertainty)
    - learn a second level classifier
    - Bayesian
      - summing K independent classifiers

$$P(C|S_1, \ldots, S_K) = \frac{P(S_1, \ldots, S_K|C)P(C)}{P(S_1, \ldots, S_K)}$$

$$C^* = \text{argmax}_c \left\{ P(C = c) \prod_{k=1}^{K} P(S_k = s_k|C = c) \right\}$$

Supervised Learning of parameters of model $\mathcal{F}$



https://en.proft.me/2015/12/24/types-machine-learning-algorithms/

A Simple but Very Useful Classifier: Naive Bayes

- Notation
  - X: feature vector of dimension D over a window
  - Y: activity label

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

- Objective: given X, find Y that maximizes the likelihood P(Y|X)
- Learning $P\left(X^1, X^2, \ldots, X^D|Y\right)$ from training data would need lots of data
- Assumes that data attributes are conditionally independent given class label

$$P\left(X^1, X^2, \ldots, X^D|Y\right) = \prod_{d=1}^{D} P(X^d|Y)$$

- Then:

$$P\left(Y = k|x^1, x^2, \ldots, x^D\right) = \frac{\prod_{d=1}^{D} P\left(x^d|Y = k\right)P\left(Y = k\right)}{\sum_j \prod_{d=1}^{D} P\left(x^d|Y = j\right)P\left(Y = j\right)}$$

$$Y = \text{argmax}_k \frac{\prod_{d=1}^{D} P(x^d|Y = k)P(Y = k)}{\sum_j \prod_{d=1}^{D} P(x^d|Y = j)P(Y = j)} = \text{argmax}_k \prod_{d=1}^{D} P(x^d|Y = k)P(Y = k)$$

Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

Naive Bayes Classifier (NBC) *contd.*

- Learning model parameters from training data
  - using maximum likelihood estimators

$$P(X^i = v_j | Y = k) = \frac{\#\text{data samples with } X^i = v_j \text{ and } Y = k}{\#\text{data samples with } Y = k}$$

  - problem: if no data for a given value of X then likelihood probability 0
    solution: smoothing (M is the # of distinct values X can take)

$$P(X^i = v_j | Y = k) = \frac{(\#\text{data samples with } X^i = v_j \text{ and } Y = k) + l}{(\#\text{data samples with } Y = k) + lM}$$

$$P(Y = k) = \frac{(\#\text{data samples with label } k) + l}{(\#\text{data samples}) + lK}$$

  - What if features are continuous instead of categorical?

Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

Applying NBC

X	x_1	x_2	x_3	x_4	x_5	x_6	x_7
Day of the week (X^1)	1	1	1	2	2	2	2
Time of the day in hours (X^2)	2	7	20	6	10	14	21
Bathroom sensor count (X^3)	13	10	5	11	3	12	3
Medicine cabinet sensor count (X^4)	0	2	6	1	4	1	9
Energy in the Hip accelerometer Z axis (X^5)	455	500	200	506	207	521	293
Activity label (Y)	2	1	3	1	4	1	3

- Probability of attribute X^1 (day of week, values in 1-7) taking value 1 given that activity is Personal Hygiene (Y=1): $P(X^1 = 1|Y = 1) = \frac{1}{3}$
- Prior probability that activity is Personal Hygiene (Y=1): $P(Y = 1) = \frac{3}{7}$
- Similarly P(X1=1|Y=3) and P(Y=3) are 1/3 and 2/7

Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

NBC with Continuous Features

- Typical approach is to model $P(X^i|Y=j)$ as a Gaussian Distribution $N(\mu_j^i, \sigma_j^{2^i})$
- Estimate the parameters of the Gaussian Distribution as:

$$\mu_j^i = E[X^i|Y=j]$$

$$\sigma_j^{2^i} = E[(X^i - \mu_j^i)^2|Y=j]$$

- This estimate can be done from training data using maximum likelihood estimation process:

$$\mu_j^i \approx \frac{\sum_{m:y_m=j} x_m^i}{\#\text{data samples with activity label } j}$$

$$\sigma_j^{2^i} \approx \frac{\sum_{m:y_m=j} (x_m^i - \mu_j^i)^2}{\#\text{data samples with activity label } j}$$

Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

Applying NBC *contd.*

X	x_1	x_2	x_3	x_4	x_5	x_6	x_7
Day of the week (X^1)	1	1	1	2	2	2	2
Time of the day in hours (X^2)	2	7	20	6	10	14	21
Bathroom sensor count (X^3)	13	10	5	11	3	12	3
Medicine cabinet sensor count (X^4)	0	2	6	1	4	1	9
Energy in the Hip accelerometer Z axis (X^5)	455	500	200	506	207	521	293
Activity label (Y)	2	1	3	1	4	1	3

- Consider X^5 (Energy in hip accelerometer Z axis)

$$\mu_1^5 = \frac{500+506+521}{3} = 509 \qquad \sigma_1^{2^5} = \frac{(500-509)^2+(506-509)^2+(521-509)^2}{3}.$$

$$\mu_3^5 = 246.5 \qquad \sigma_3^{2^5} = 2162.5$$

Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

Applying NBC *contd.*

X	x_1	x_2	x_3	x_4	x_5	x_6	x_7
Day of the week (X^1)	1	1	1	2	2	2	2
Time of the day in hours (X^2)	2	7	20	6	10	14	21
Bathroom sensor count (X^3)	13	10	5	11	3	12	3
Medicine cabinet sensor count (X^4)	0	2	6	1	4	1	9
Energy in the Hip accelerometer Z axis (X^5)	455	500	200	506	207	521	293
Activity label (Y)	2	1	3	1	4	1	3

- Given a test sample:

  $X^1=1$
  $X^5=250$

$$P(Y = 1|x^1 = 1,\ x^5 = 250) = P(x^1 = 1|Y = 1) \times P(x^5 = 250|Y = 1) \times P(Y = 1)$$

$$= \frac{1}{3} \times (8.04 \times 10^{-189}) \times \frac{3}{7} = 1.419 \times 10^{-189}$$

$$P(Y = 3|x^1 = 1, x^5 = 250) = P(x^1 = 1|Y = 3) \times P(x^5 = 250|Y = 3) \times P(Y = 3)$$

$$= \frac{1}{3} \times 0.086 \times \frac{2}{7} = 8.1905 \times 10^{-4}$$

Since $P(Y = 3|x^1 = 1, x^5 = 250) > P(Y = 1|x^1 = 1,\ x^5 = 250)$, the NBC will label this data point as Take Medicine $(Y = 3)$.

Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

NBC Summary

- Probabilistic approach to classification that is popular for several reasons
  - Simple
  - Explicit mechanism for calculating explicit probabilities for different hypothesis
  - Interpretable

- Initial probabilities
  - Domain knowledge
  - Learnt from training data

- Often gives good performance even when independence is violated in real world

- Assuming continuous attribute is Gaussian has nice analytical properties and yields a simple methods for parameter estimation, but is often not true

Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

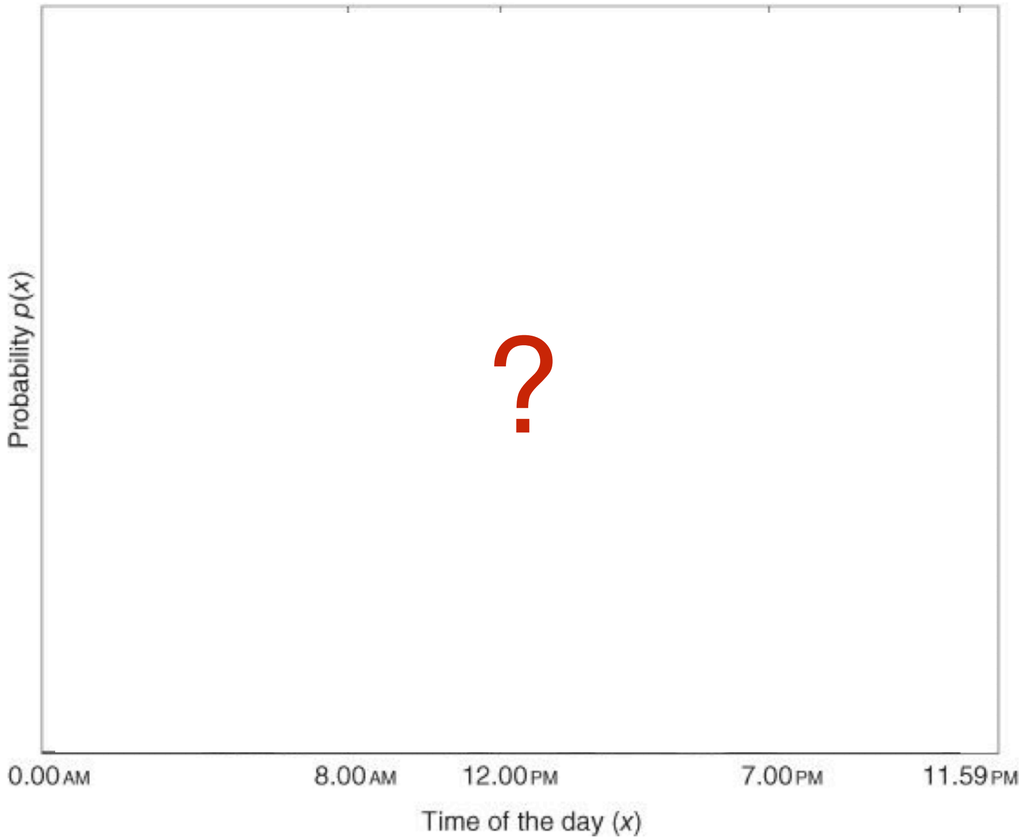
Extending NBC to Continuous Attributes

- Assuming attribute is Gaussian distribution
  - Nice analytical properties
  - Simple method for parameter estimation

- But, is it sufficient?

  E.g. Activity Start Time attribute
  for Eating Activity.

  What do you expect the distribution to look like?



Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

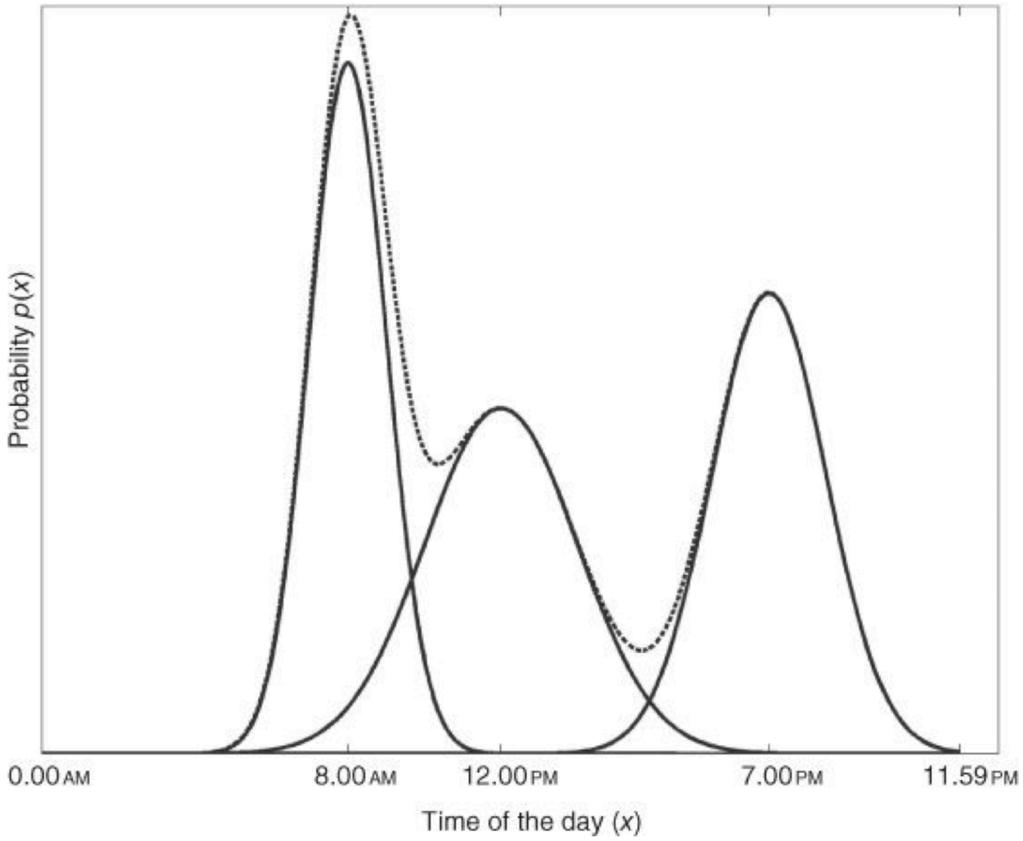
Extending NBC to Continuous Attributes

- Assuming attribute is Gaussian distribution
  - Nice analytical properties
  - Simple method for parameter estimation

- But, is it sufficient?

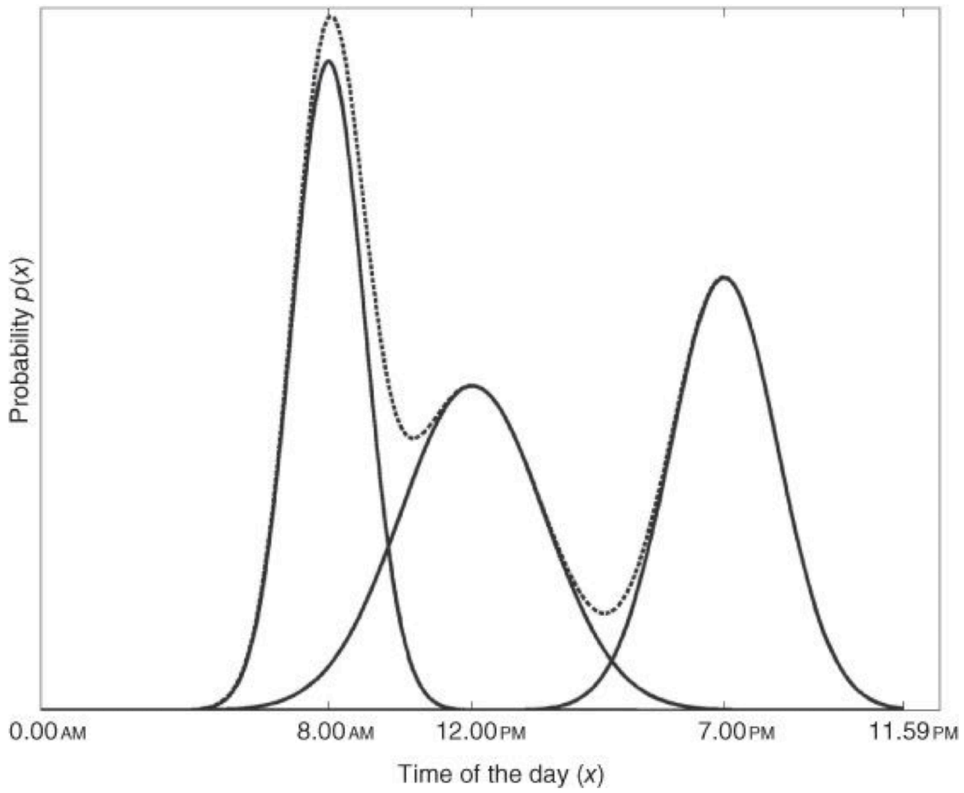
  E.g. Activity Start Time attribute
  for Eating Activity.

  What do you expect the distribution to look like?



Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

Gaussian Mixture Models



- Distribution P(Xi|Y=k) is modeled as a combination of M Gaussian probability distribution functions

$$P(X^i|Y=k) = \sum_{m=1}^{M} \pi_m N(\mu_{km}^i, \sigma_{km}^{2^i})$$

$$\pi_m \geq 0 \text{ and } \sum_{m=1}^{M} \pi_m = 1$$

- Generalize to cover all attributes by considering M multivariate Gaussians where μ_{km} & σ_{km}^2 are D-dimensional vectors
- Parameters and mixing coefficients estimated using an iterative Expectation-Maximization algorithm

Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

Expectation Maximization Algorithm

Algorithm **GMM_EM**(x)

// $x = x_1, x_2, .., x_{Nk}$ is the subset L_k of the training data that is marked with activity k
// M is the number of Gaussian mixture components required to describe L_k

for m = 1 .. M

Initialize the multivariate Gaussian mean μ_{km} and variance σ^2_{km}

done

repeat

for i = 1 .. N_k // Expectation step

$p_{im} = N(x_i \mid \mu_{km}, \sigma^2_{km})$ // p_{im} represents probability that data point x_i has been
// sampled from the mth mixture component $N(\mu_{km}, \sigma^2_{km})$

$z_i = \underset{m}{\operatorname{argmax}}\, p_{im}$ // z_i is a latent variable representing the mixture
// component to which data point x_i is assigned

done

for m = 1 .. M // Maximization step

A_m = number of data points assigned to mth component

$$\mu_{km} = \frac{1}{A_m} \sum \{x_i : z_i = m\}$$ // Update the mean

$$\sigma^2_{km} = \frac{1}{A_m} \sum \{(x_i - \mu_{km})^2 : z_i = m\}$$ // Update the variance

$$\pi_{km} = \frac{A_m}{N_k}$$ // Update the mixing coefficient

done

until convergence

Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

Expectation Maximization Algorithm

Initial Estimate

Algorithm **GMM_EM**(x)

// $x = x_1, x_2, .., x_{Nk}$ is the subset L_k of the training data that is marked with activity k
// M is the number of Gaussian mixture components required to describe L_k

for m = 1 .. M

Initialize the multivariate Gaussian mean μ_{km} and variance σ^2_{km}

done

repeat

for i = 1 .. N_k // Expectation step

$p_{im} = N(x_i \mid \mu_{km}, \sigma^2_{km})$ // p_{im} represents probability that data point x_i has been
// sampled from the m[th] mixture component $N(\mu_{km}, \sigma^2_{km})$

$z_i = \underset{m}{\text{argmax}}\, p_{im}$ // z_i is a latent variable representing the mixture
// component to which data point x_i is assigned

done

for m = 1 .. M // Maximization step

A_m = number of data points assigned to m[th] component

$\mu_{km} = \frac{1}{A_m} \sum \{x_i : z_i = m\}$ // Update the mean

$\sigma^2_{km} = \frac{1}{A_m} \sum \{(x_i - \mu_{km})^2 : z_i = m\}$ // Update the variance

$\pi_{km} = \frac{A_m}{N_k}$ // Update the mixing coefficient

done

until convergence

Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

Expectation Maximization Algorithm

Initial Estimate

M Step

parameters are updated using the modified data assignments

Algorithm **GMM_EM**(x)

// $x = x_1, x_2, .., x_{Nk}$ is the subset L_k of the training data that is marked with activity k
// M is the number of Gaussian mixture components required to describe L_k

for m = 1 .. M

Initialize the multivariate Gaussian mean μ_{km} and variance σ^2_{km}

done

repeat

for i = 1 .. N_k // Expectation step

$p_{im} = N(x_i \mid \mu_{km}, \sigma^2_{km})$ // p_{im} represents probability that data point x_i has been
// sampled from the m[th] mixture component $N(\mu_{km}, \sigma^2_{km})$

$z_i = \underset{m}{\text{argmax}}\, p_{im}$ // z_i is a latent variable representing the mixture
// component to which data point x_i is assigned

done

for m = 1 .. M // Maximization step

A_m = number of data points assigned to m[th] component

$$\mu_{km} = \frac{1}{A_m} \sum \{x_i : z_i = m\}$$ // Update the mean

$$\sigma^2_{km} = \frac{1}{A_m} \sum \{(x_i - \mu_{km})^2 : z_i = m\}$$ // Update the variance

$$\pi_{km} = \frac{A_m}{N_k}$$ // Update the mixing coefficient

done

until convergence

Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

Expectation Maximization Algorithm

Initial Estimate

M Step

parameters are updated using the modified data assignments

E Step

each datapoint is assigned to the nearest mixture component

Algorithm **GMM_EM**(x)

// $x = x_1, x_2, .., x_{Nk}$ is the subset L_k of the training data that is marked with activity k
// M is the number of Gaussian mixture components required to describe L_k

for m = 1 .. M
  Initialize the multivariate Gaussian mean μ_{km} and variance σ^2_{km}
done

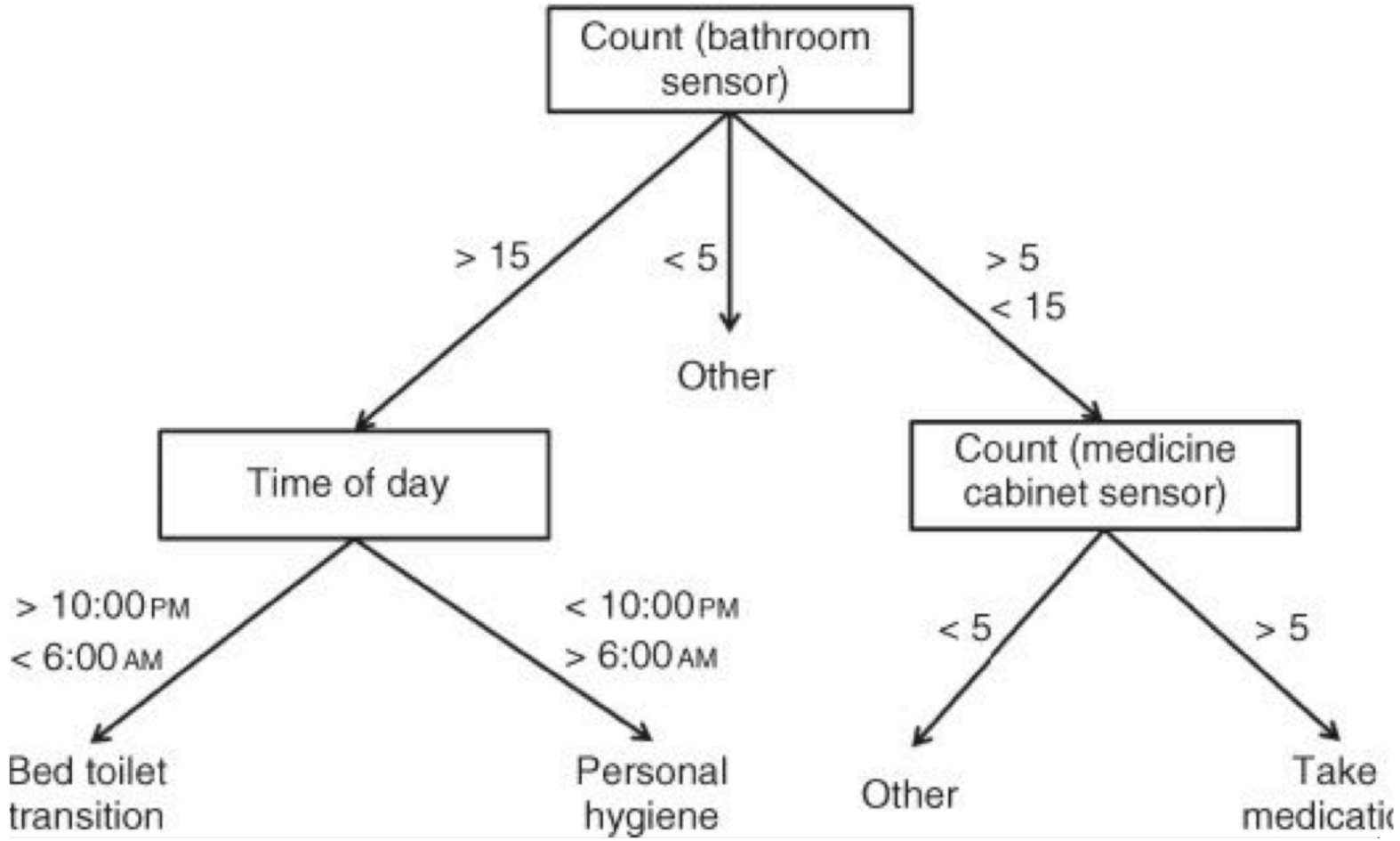
repeat
  for i = 1 .. N_k // Expectation step
    $p_{im} = N(x_i \mid \mu_{km}, \sigma^2_{km})$ // p_{im} represents probability that data point x_i has been
    // sampled from the m[th] mixture component $N(\mu_{km}, \sigma^2_{km})$
    $z_i = \underset{m}{\operatorname{argmax}}\, p_{im}$ // z_i is a latent variable representing the mixture
    // component to which data point x_i is assigned
  done

  for m = 1 .. M // Maximization step
    A_m = number of data points assigned to m[th] component
    $\mu_{km} = \frac{1}{A_m} \sum \{x_i : z_i = m\}$ // Update the mean
    $\sigma^2_{km} = \frac{1}{A_m} \sum \{(x_i - \mu_{km})^2 : z_i = m\}$ // Update the variance
    $\pi_{km} = \frac{A_m}{N_k}$ // Update the mixing coefficient
  done

until convergence

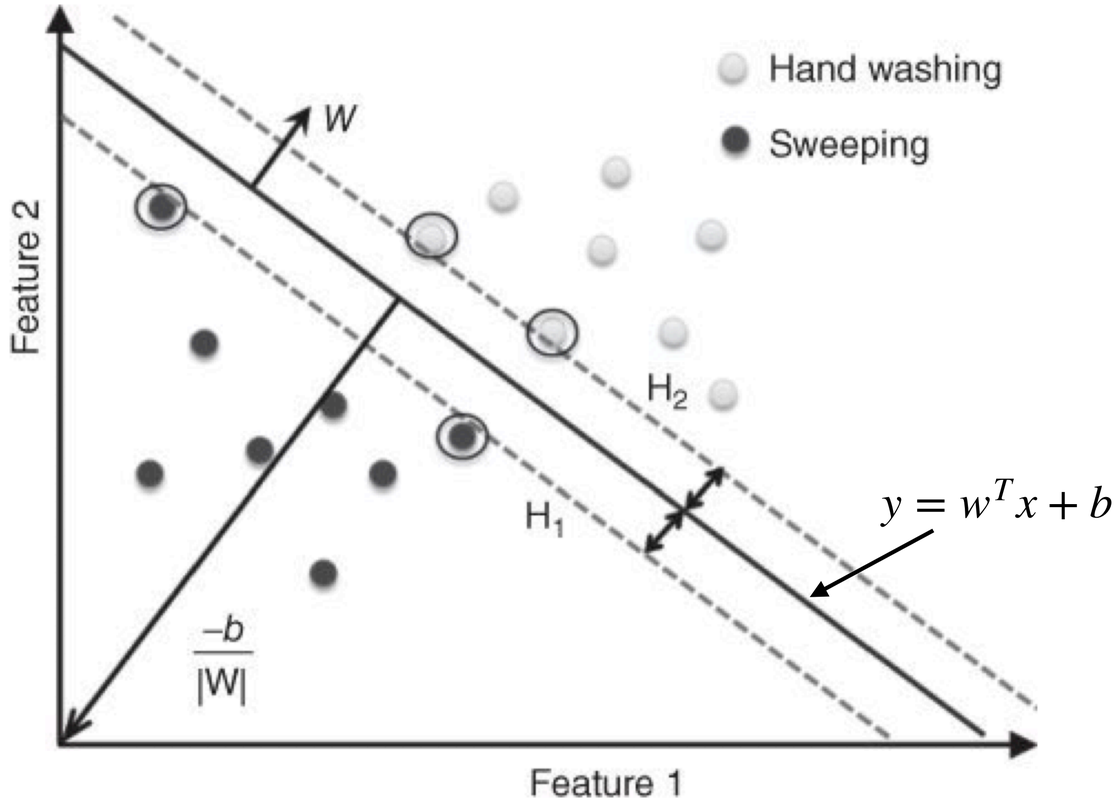
Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

Some Other Classifiers: Decision Tree



Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

Some Other Classifiers: Support Vector Machine

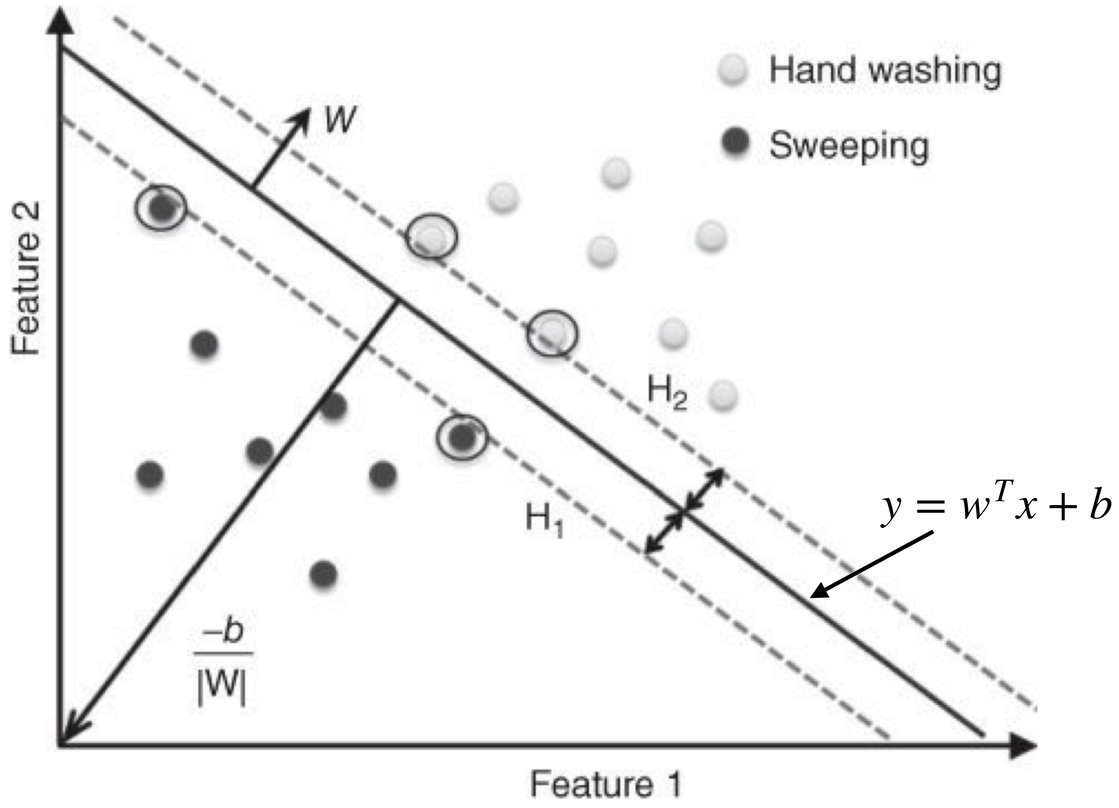


Separable Case: Hard Margin

$$min\ \frac{1}{2}||w||^2 \text{ such that } y_i(w^T x + b) \geq 0 \quad \forall i$$

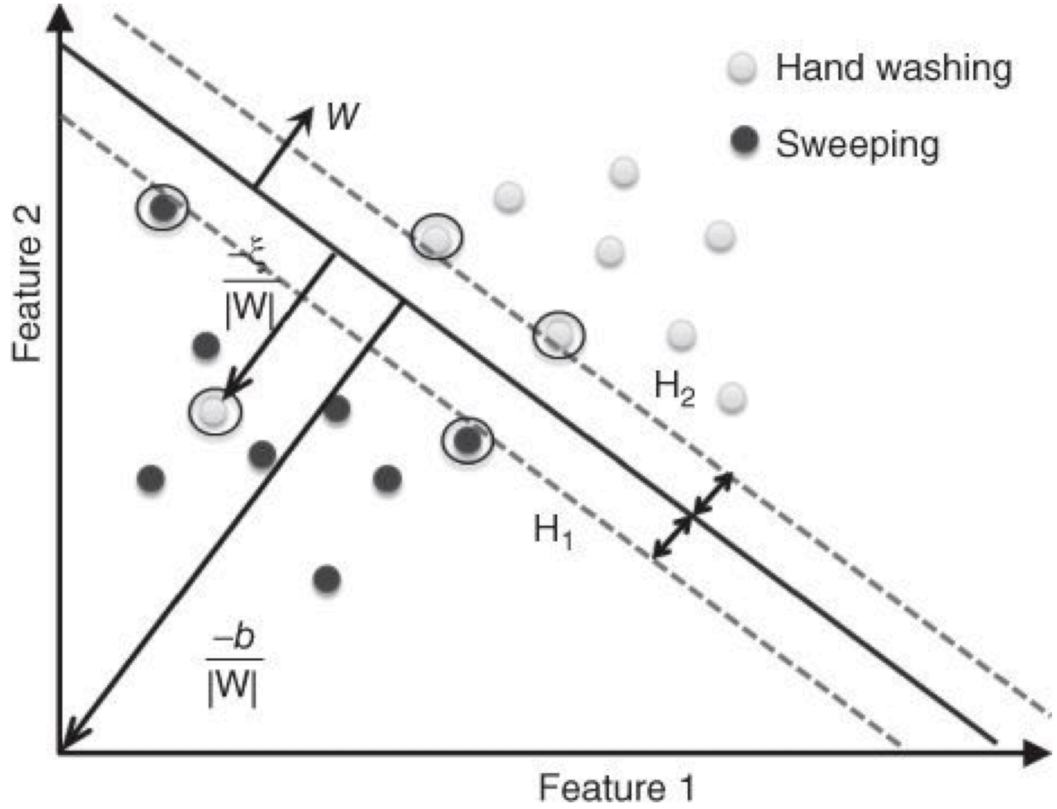
Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

Some Other Classifiers: Support Vector Machine



Separable Case: Hard Margin

$$min\ \frac{1}{2}\|w\|^2 \text{ such that } y_i(w^T x + b) \geq 0 \quad \forall i$$

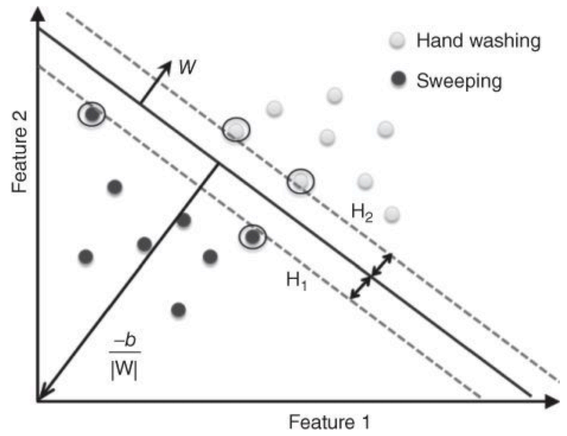


Non-separable Case Soft Margin

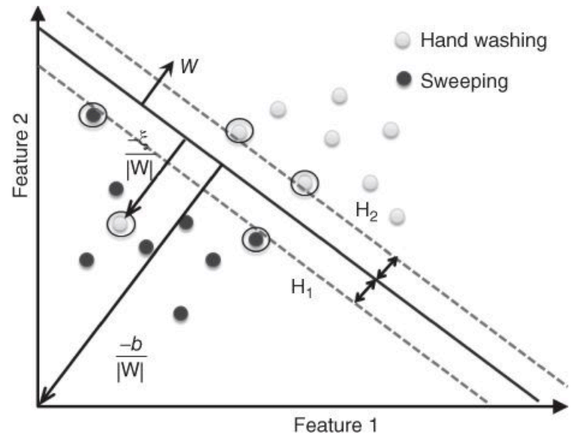
$$min\ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\zeta_i \text{ such that } y_i(w^T x + b) - 1 + \zeta_i \geq 0 \text{ and } \zeta_i \geq 0 \quad \forall i$$

Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

Some Other Classifiers: Non-Linear Support Vector Machine



Separable Case

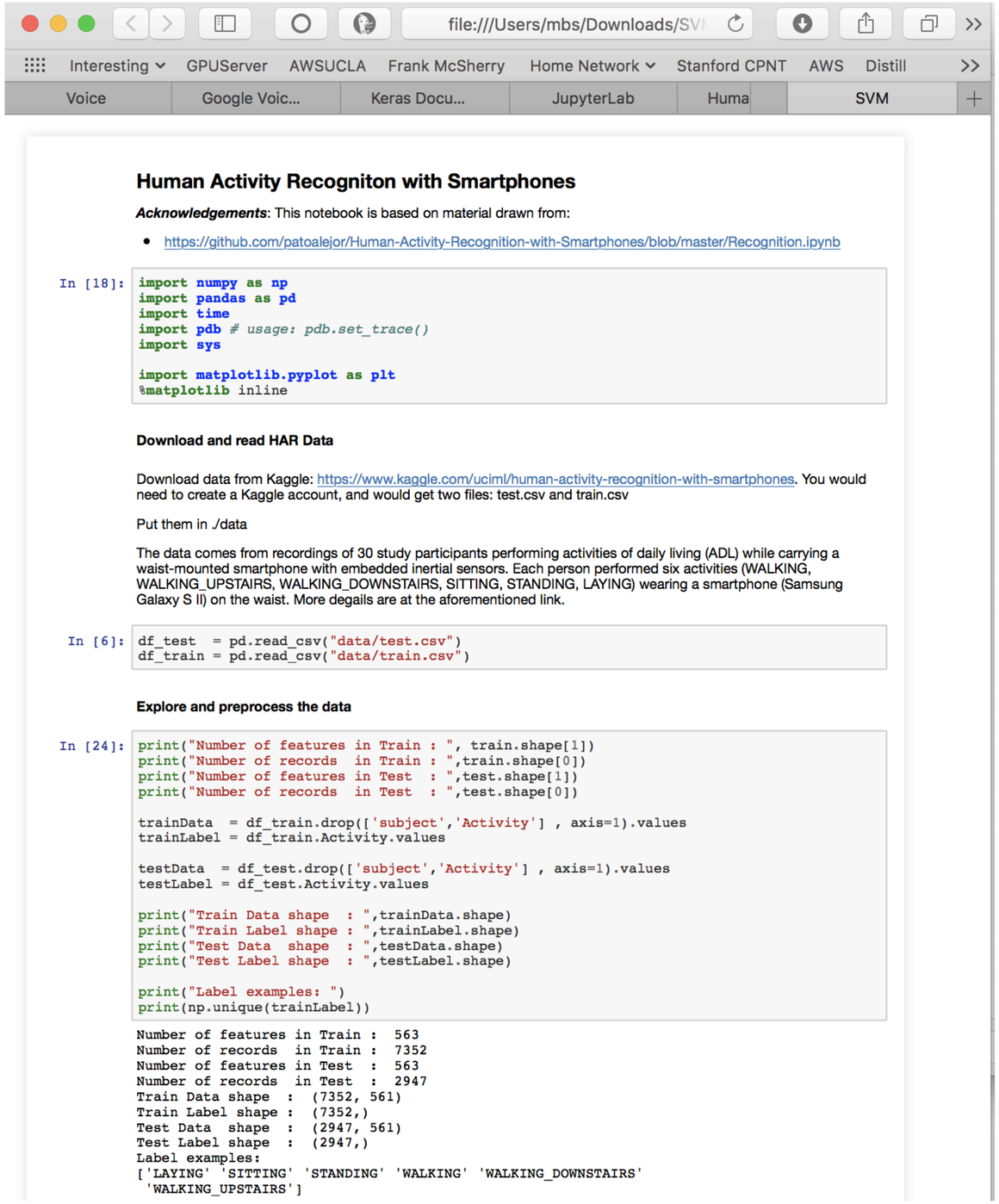


Non-separable Case

Linear kernel	$x_i^T x_j$
Polynomial kernel	$(x_i^T x_j + c)^d$
Radial basis function kernel	$\exp\left(-\frac{\|\|x_i - x_j\|\|_2^2}{2\sigma^2}\right)$
Hyperbolic tangent kernel	$\tan h(\kappa x_i^T x_j + c)$ for $\kappa > 0$ and $c < 0$

Cook & Krishnan. Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. 2015.

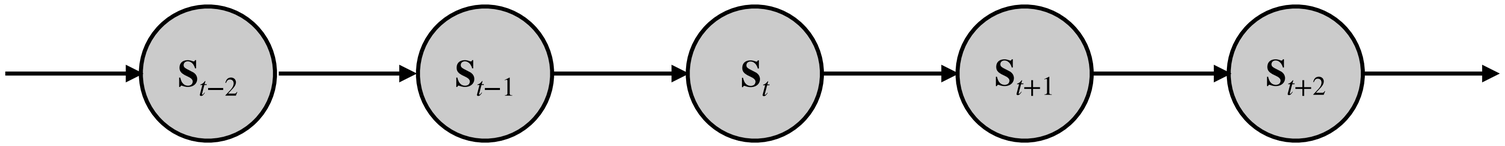
SVM in practice…



Temporal Probabilistic Models

State Transition Model

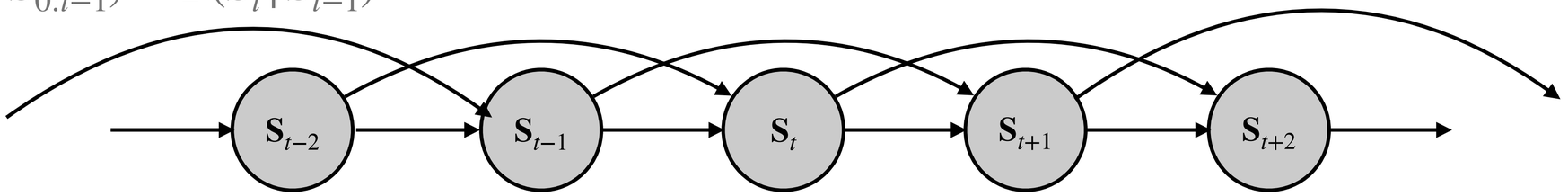
- Specifies how the state of the environment evolves, i.e. $\mathbf{P}(\mathbf{S}_t \mid \mathbf{S}_{0:t-1})$
  - Initial state is considered known $\mathbf{S}_0 = \mathbf{s}_0$
- Problem #1: the set $\mathbf{S}_{0:t-1}$ is unbounded as t grows
  - Solve by **Markov assumption**: current state $\mathbf{S}_t$ depends only on a finite fixed number of previous states
  - Many flavors
    - simplest is first-order Markov process in which current state depends only on the previous state, i.e. $\mathbf{P}(\mathbf{S}_t \mid \mathbf{S}_{0:t-1}) = \mathbf{P}(\mathbf{S}_t \mid \mathbf{S}_{t-1})$



Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

State Transition Model

- Specifies how the state of the environment evolves, i.e. $\mathbf{P}(\mathbf{S}_t \mid \mathbf{S}_{0:t-1})$
  - Initial state is considered known $\mathbf{S}_0 = \mathbf{s}_0$
- Problem #1: the set $\mathbf{S}_{0:t-1}$ is unbounded as t grows
  - Solve by **Markov assumption**: current state $\mathbf{S}_t$ depends only on a finite fixed number of previous states
  - Many flavors
    - simplest is first-order Markov process in which current state depends only on the previous state, i.e. $\mathbf{P}(\mathbf{S}_t \mid \mathbf{S}_{0:t-1}) = \mathbf{P}(\mathbf{S}_t \mid \mathbf{S}_{t-1})$

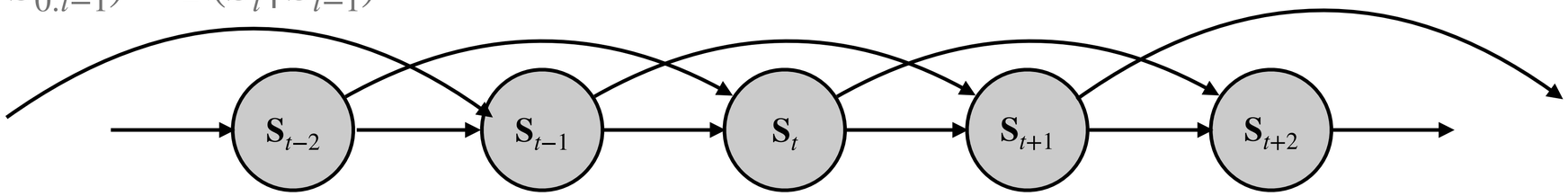


    - more complex, such as second-order Markov process $\mathbf{P}(\mathbf{S}_t \mid \mathbf{S}_{0:t-1}) = \mathbf{P}(\mathbf{S}_t \mid \mathbf{S}_{t-1}, \mathbf{S}_{t-2})$
      - question: can you reduce second-order Markov process to a first-order one?
  - First-order Markov process is commonly used but often insufficient for real world
    - Solution: increase order of Markov process, or augment the state

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

State Transition Model

- Specifies how the state of the environment evolves, i.e. $\mathbf{P}(\mathbf{S}_t \mid \mathbf{S}_{0:t-1})$
  - Initial state is considered known $\mathbf{S}_0 = \mathbf{s}_0$
- Problem #1: the set $\mathbf{S}_{0:t-1}$ is unbounded as t grows
  - Solve by **Markov assumption**: current state $\mathbf{S}_t$ depends only on a finite fixed number of previous states
  - Many flavors
    - simplest is first-order Markov process in which current state depends only on the previous state, i.e. $\mathbf{P}(\mathbf{S}_t \mid \mathbf{S}_{0:t-1}) = \mathbf{P}(\mathbf{S}_t \mid \mathbf{S}_{t-1})$



    - more complex, such as second-order Markov process $\mathbf{P}(\mathbf{S}_t \mid \mathbf{S}_{0:t-1}) = \mathbf{P}(\mathbf{S}_t \mid \mathbf{S}_{t-1}, \mathbf{S}_{t-2})$
      - question: can you reduce second-order Markov process to a first-order one?
  - First-order Markov process is commonly used but often insufficient for real world
    - Solution: increase order of Markov process, or augment the state
- Problem #2: we have infinitely many t, each with its own distribution for state transition
  - Solve by Stationarity Assumption, i.e. $\mathbf{P}(\mathbf{S}_t \mid \mathbf{S}_{0:t-1}) = \mathbf{P}(\mathbf{S}_{t'} \mid \mathbf{S}_{0:t'-1}) \forall t, t'$

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Sensor Model (aka Observation Model)

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

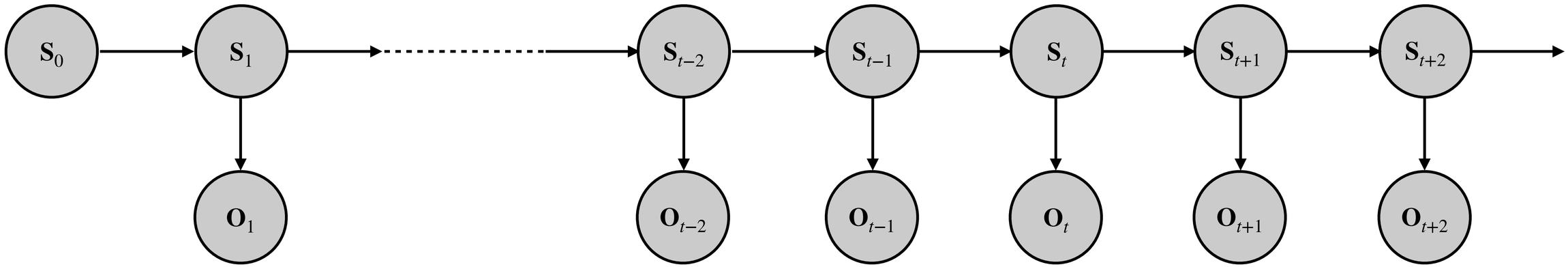
Sensor Model (aka Observation Model)

- Specifies how the sensor observations depend on the current state and previous variables, i.e. $\mathbf{P}(\mathbf{O}_t \mid \mathbf{S}_{0:t}, \mathbf{O}_{1:t-1})$
- A good choice of state should suffice to generate the current sensor value, which leads to **sensor Markov assumption**, i.e. $\mathbf{P}(\mathbf{O}_t \mid \mathbf{S}_{0:t}, \mathbf{O}_{1:t-1}) = \mathbf{P}(\mathbf{O}_t \mid \mathbf{S}_t)$

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Sensor Model (aka Observation Model)

- Specifies how the sensor observations depend on the current state and previous variables, i.e. $\mathbf{P}(\mathbf{O}_t \mid \mathbf{S}_{0:t}, \mathbf{O}_{1:t-1})$
- A good choice of state should suffice to generate the current sensor value, which leads to **sensor Markov assumption**, i.e. $\mathbf{P}(\mathbf{O}_t \mid \mathbf{S}_{0:t}, \mathbf{O}_{1:t-1}) = \mathbf{P}(\mathbf{O}_t \mid \mathbf{S}_t)$
- Combining with the simplified first order Markov transition model, we get:

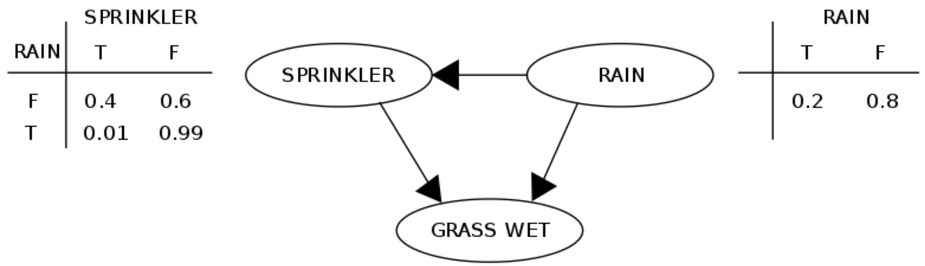


- Complete joint distribution: $\mathbf{P}(\mathbf{S}_{0:t}, \mathbf{O}_{1:t}) = \mathbf{P}(\mathbf{S}_0) \prod_{i=1}^{t} \mathbf{P}(\mathbf{S}_i \mid \mathbf{S}_{i-1}) \mathbf{P}(\mathbf{O}_i \mid \mathbf{S}_i)$ **Dynamic Bayesian Network**

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Digression: Bayesian Network

- A probabilistic graphical model that represents a set of variables and their conditional dependencies via a directed acyclic graph (DAG)
  - Nodes represent variables: observable quantities, latent variables, unknown parameters or hypotheses
  - Edges represent conditional dependencies
    - nodes that are not connected (no path connects one node to another) represent conditionally independent variables
- Each node is associated with a probability function that takes, as input, a particular set of values for the node's parent variables, and gives the probability distribution of the variable represented by the node.
- Efficient algorithms can perform inference and learning in Bayesian networks
  - Using a Bayesian network can save considerable amounts of memory over exhaustive probability tables,



RAIN	SPRINKLER T	SPRINKLER F
F	0.4	0.6
T	0.01	0.99

RAIN T	RAIN F
0.2	0.8

SPRINKLER	RAIN	GRASS WET T	GRASS WET F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

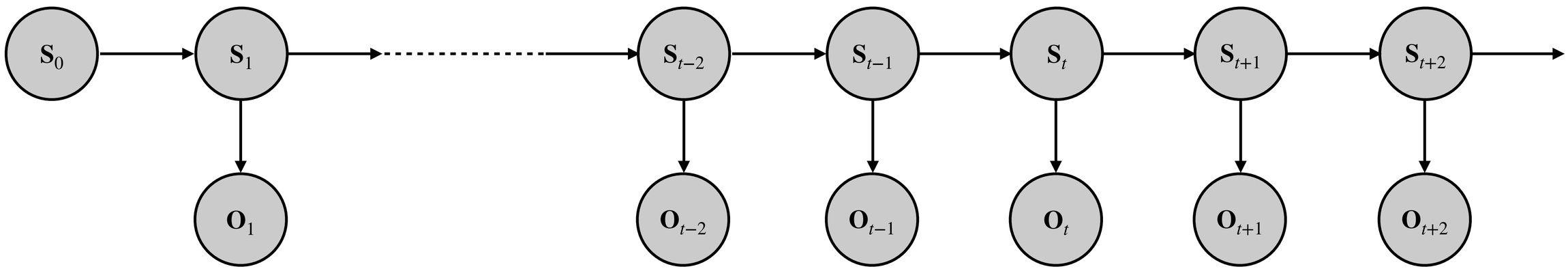
$$P(G,S,R) = P(G \mid S,R)P(S \mid R)P(R)$$

E.g. The model can answer questions about the presence of a cause given the presence of an effect (so-called inverse probability) like "What is the probability that it is raining, given the grass is wet?"

$$P(R = T \mid G = T) = \frac{P(G = T, R = T)}{P(G = T)} = \frac{\sum_{S \in \{T,F\}} P(G = T, S, R = T)}{\sum_{S,R \in \{T,F\}} P(G = T, S, R)}$$

https://en.wikipedia.org/wiki/Bayesian_network

Inference Tasks: Latent States from Sensor Observations



- *Filtering*: $\mathbf{P}(\mathbf{S}_t \mid \mathbf{O}_{1:t} = \mathbf{o}_{1:t})$ — **keep track of current state for rational decision-making**
- *Forecasting*: $\mathbf{P}(\mathbf{S}_{t'} \mid \mathbf{O}_{1:t} = \mathbf{o}_{1:t})$ where $t' > t$ — **evaluate possible courses of action based on their expected outcomes**
- *Smoothing*: $\mathbf{P}(\mathbf{S}_{t'} \mid \mathbf{O}_{1:t} = \mathbf{o}_{1:t})$ where $t' < t$ — **better estimate of the state than was available at the time, useful for learning**
- *Most likely explanation*: $\text{argmax}_{\mathbf{s}_{0:t}} \mathbf{P}(\mathbf{S}_{1:t} \mid \mathbf{O}_{1:t} = \mathbf{o}_{1:t})$ — **tasks such as speech recognition**

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Learning Task

- The transition and sensor models, if not yet known, can be learnt from observations
- Learning can be done as a byproduct of inference
  - Inference provides an estimate of what transitions actually occurred and of what states generated the sensor readings
  - These estimates can be used to update the models
  - The updated model provides new estimates, and the process iterates to convergence
  - Overall process is an instance of the Expectation Maximization (EM) algorithms

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Filtering

- A useful filtering algorithm needs to maintain a current state estimate and update it
  - rather than going back over the entire history of percepts for each update. as then the cost of update will increase with time
- In other words, we need a recursive estimation algorithm of the form $\mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{o}_{1:t+1}) = f(\mathbf{o}_{t+1}, \mathbf{P}(\mathbf{S}_t \mid \mathbf{o}_{1:t}))$ for some function f
- Computation can be viewed as being composed of two parts: *project* current state forward in time, and then *update* it in light of the new sensor observation o_{t+1}

$$\begin{aligned}
\mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{o}_{1:t+1}) &= \mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{o}_{1:t}, \mathbf{o}_{t+1}) && \text{(dividing up the observations)} \\
&= \alpha \mathbf{P}(\mathbf{o}_{t+1} \mid \mathbf{S}_{t+1}, \mathbf{o}_{1:t}) \mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{o}_{1:t}) && \text{(using Bayes' rule, } \alpha \text{ is normalizing constant)} \\
&= \alpha \mathbf{P}(\mathbf{o}_{t+1} \mid \mathbf{S}_{t+1}) \mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{o}_{1:t}) && \text{(by the sensor Markov assumption)}
\end{aligned}$$

update prediction

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Filtering

- A useful filtering algorithm needs to maintain a current state estimate and update it
  - rather than going back over the entire history of percepts for each update. as then the cost of update will increase with time
- In other words, we need a recursive estimation algorithm of the form $\mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{o}_{1:t+1}) = f(\mathbf{o}_{t+1}, \mathbf{P}(\mathbf{S}_t \mid \mathbf{o}_{1:t}))$ for some function f
- Computation can be viewed as being composed of two parts: *project* current state forward in time, and then *update* it in light of the new sensor observation $\mathbf{o}_{t+1}$

$$
\begin{aligned}
\mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{o}_{1:t+1}) &= \mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{o}_{1:t}, \mathbf{o}_{t+1}) && \text{(dividing up the observations)} \\
&= \alpha \mathbf{P}(\mathbf{o}_{t+1} \mid \mathbf{S}_{t+1}, \mathbf{o}_{1:t}) \mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{o}_{1:t}) && \text{(using Bayes' rule, } \alpha \text{ is normalizing constant)} \\
&= \alpha \mathbf{P}(\mathbf{o}_{t+1} \mid \mathbf{S}_{t+1}) \mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{o}_{1:t}) && \text{(by the sensor Markov assumption)} \\
&= \alpha \mathbf{P}(\mathbf{o}_{t+1} \mid \mathbf{S}_{t+1}) \sum_{\mathbf{s}_t} \mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{s}_t, \mathbf{o}_{1:t}) \mathbf{P}(\mathbf{s}_t \mid \mathbf{o}_{1:t}) \\
&= \alpha \underbrace{\mathbf{P}(\mathbf{o}_{t+1} \mid \mathbf{S}_{t+1})}_{\text{sensor model}} \sum_{\mathbf{s}_t} \underbrace{\mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{s}_t)}_{\text{transition model}} \underbrace{\mathbf{P}(\mathbf{s}_t \mid \mathbf{o}_{1:t})}_{\text{recursion}} && \text{(by Markov assumption)}
\end{aligned}
$$

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

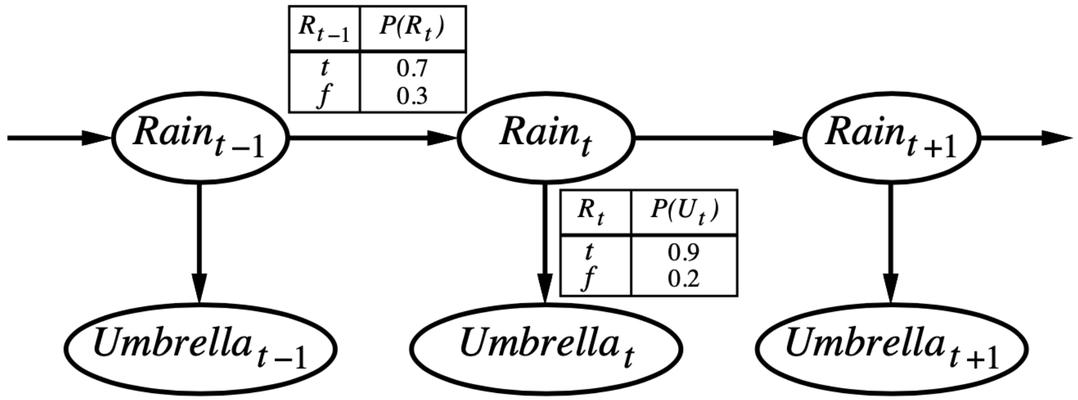
Filtering

- A useful filtering algorithm needs to maintain a current state estimate and update it
  - rather than going back over the entire history of percepts for each update. as then the cost of update will increase with time
- In other words, we need a recursive estimation algorithm of the form $\mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{o}_{1:t+1}) = f(\mathbf{o}_{t+1}, \mathbf{P}(\mathbf{S}_t \mid \mathbf{o}_{1:t}))$ for some function f
- Computation can be viewed as being composed of two parts: *project* current state forward in time, and then *update* it in light of the new sensor observation $\mathbf{o}_{t+1}$

$$\mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{o}_{1:t+1}) = \alpha \mathbf{P}(\mathbf{o}_{t+1} \mid \mathbf{S}_{t+1}) \sum_{\mathbf{s}_t} \mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{s}_t) \mathbf{P}(\mathbf{s}_t \mid \mathbf{o}_{1:t})$$

- One can view the filtered estimate $\mathbf{P}(\mathbf{S}_t \mid \mathbf{o}_{1:t})$ as a message $\mathbf{f}_{1:t}$ that is propagated along the sequence, modified by each transition and updated by each new sensor observation via $\mathbf{f}_{1:t+1} = \text{FORWARD}(\mathbf{f}_{1:t}, \mathbf{o}_{t+1})$ where FORWARD implements the above equation and $\mathbf{f}_{1:0} = \mathbf{P}(\mathbf{S}_0)$

Example of Filtering



- Goal: compute $P(R_2 \mid u_{1:2})$
- Day 0: no observation - only prior belief $P(R_0) = \langle 0.5,0.5 \rangle$.
- Day 1: umbrella appears, so $U_1 = true$

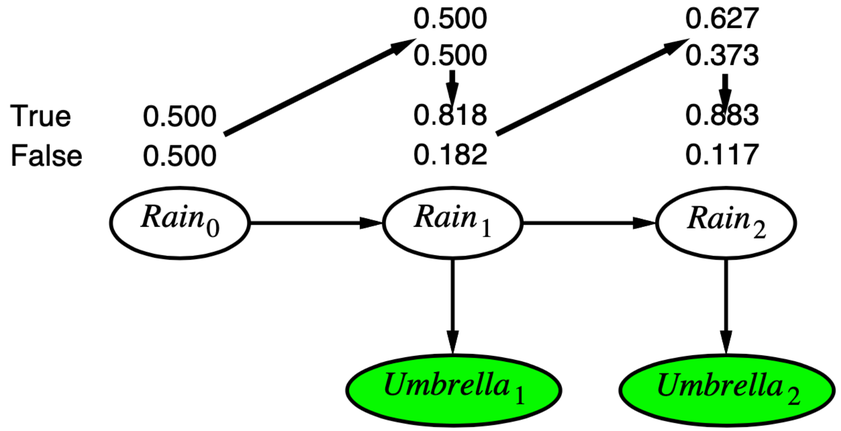
$$P(R_1) = \sum_{r_0} P(R_1 \mid r_0)P(r_0) = \langle 0.7,0.3 \rangle \times 0.5 + \langle 0.3,0.7 \rangle \times 0.5 = \langle 0.5,0.5 \rangle$$

$$P(R_1 \mid u_1) = \alpha P(u_1 \mid R_1)P(R_1) = \alpha \langle 0.9,0.2 \rangle \langle 0.5,0.5 \rangle = \alpha \langle 0.45,0.1 \rangle \approx \langle 0.818,0.182 \rangle$$

- Day 2: umbrella appears, so $U_2 = true$

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

$$P(R_2 \mid u_1, u_2) = \alpha P(u_2 \mid R_2)P(R_2 \mid u_1) = \alpha \langle 0.9,0.2 \rangle \langle 0.627,0.373 \rangle = \alpha \langle 0.565,0.075 \rangle \approx \langle 0.883,0.117 \rangle$$



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

Forecasting

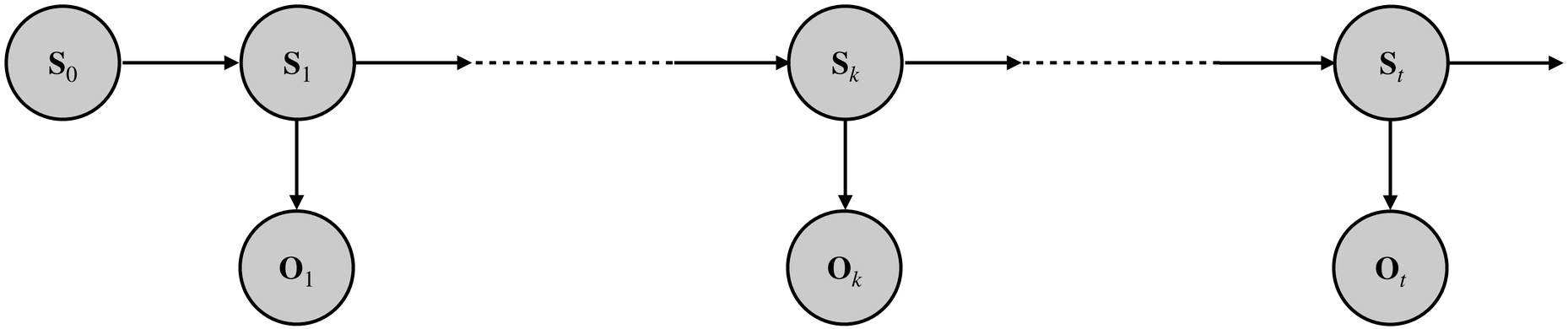
- The task of forecasting can be seen simply as filtering without the addition of new sensor observation
- The filtering process already incorporates a one-step prediction
- It is easy to derive the following recursive computation for predicting the state at $t + k + 1$ from a prediction for $t + k$

$$\mathbf{P}(\mathbf{S}_{t+k+1} \mid \mathbf{o}_{1:t}) = \sum_{\mathbf{s}_{t+k}} \underbrace{\mathbf{P}(\mathbf{S}_{t+k+1} \mid \mathbf{s}_{t+k})}_{\text{transition model}} \underbrace{\mathbf{P}(\mathbf{s}_{t+k} \mid \mathbf{o}_{1:t})}_{\text{recursion}}$$

- Note that no sensor model is involved in forecasting
- if we try to predict further and further into the future, the predicted distribution for the states will converge to the stationary distribution of the Markov process defined by the transition model
  - **mixing time:** roughly, the time taken to reach the fixed point
    - in practical terms, this dooms to failure any attempt to predict the actual state for a number of steps that is more than a small fraction of the mixing time, unless the stationary distribution itself is strongly peaked in a small area of the state space
    - the more uncertainty there is in the transition model, the shorter will be the mixing time and the more the future is obscured

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Smoothing



- Process of computing the distribution over past states given evidence up to the present
  i.e. $\mathbf{P}(\mathbf{S}_k \mid \mathbf{o}_{1:t})$ for $0 \leq k < t$
- Another recursive message-passing approach by splitting the computation into two parts: the observations up to k and the observations from $k+1$ to t

$$
\begin{aligned}
\mathbf{P}(\mathbf{S}_k \mid \mathbf{o}_{1:t}) &= \mathbf{P}(\mathbf{S}_k \mid \mathbf{o}_{1:k}, \mathbf{o}_{k+1,t}) \\
&= \alpha \mathbf{P}(\mathbf{S}_k \mid \mathbf{o}_{1:k}) \mathbf{P}(\mathbf{o}_{k+1,t} \mid \mathbf{S}_k, \mathbf{o}_{1:k}) && \text{(using Bayes' rule, given } \mathbf{o}_{1:k}) \\
&= \alpha \mathbf{P}(\mathbf{S}_k \mid \mathbf{o}_{1:k}) \mathbf{P}(\mathbf{o}_{k+1,t} \mid \mathbf{S}_k) && \text{(using conditional independence)} \\
&= \alpha \mathbf{f}_{1:k} \times \underbrace{\mathbf{b}_{k+1:1}}_{\text{backward message}} && (\times \text{ represents pointwise multiplication of vectors})
\end{aligned}
$$

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Computing Backward Message $b_{k+1:t}$

- By a recursive process that runs backward from t

$$\mathbf{P}(\mathbf{o}_{k+1:t}|\mathbf{S}_k) = \sum_{\mathbf{s}_{k+1}} \mathbf{P}(\mathbf{o}_{k+1:t}|\mathbf{S}_k, \mathbf{s}_{k+1})\mathbf{P}(\mathbf{s}_{k+1}|\mathbf{S}_k) \qquad \text{(conditioning on } \mathbf{S}_{k+1}\text{)}$$

$$= \sum_{\mathbf{s}_{k+1}} \mathbf{P}(\mathbf{o}_{k+1:t}|\mathbf{s}_{k+1})\mathbf{P}(\mathbf{s}_{k+1}|\mathbf{S}_k) \qquad \text{(by conditional independence)}$$

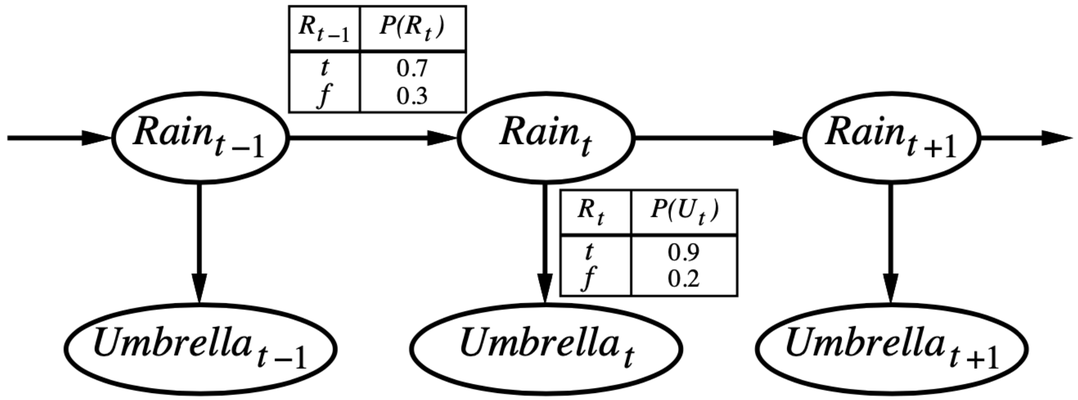
$$= \sum_{\mathbf{s}_{k+1}} \mathbf{P}(\mathbf{o}_{k+1}, \mathbf{o}_{k+2:t}|\mathbf{s}_{k+1})\mathbf{P}(\mathbf{s}_{k+1}|\mathbf{S}_k)$$

$$= \sum_{\mathbf{s}_{k+1}} \underbrace{\mathbf{P}(\mathbf{o}_{k+1}|\mathbf{s}_{k+1})}_{\text{sensor model}}\underbrace{\mathbf{P}(\mathbf{o}_{k+2:t}|\mathbf{s}_{k+1})}_{\text{recursion}}\underbrace{\mathbf{P}(\mathbf{s}_{k+1}|\mathbf{S}_k)}_{\text{transition model}} \qquad \text{(by conditional independence of } \mathbf{o}_{k+1} \text{ \& } \mathbf{o}_{k+2:t} \text{ given } \mathbf{s}_{k+1}\text{)}$$

- In message form, we have $\mathbf{b}_{k+1:t} = \text{BACKWARD}(\mathbf{b}_{k+2:t}, \mathbf{o}_{k+1})$
- Initialization with $\mathbf{b}_{t+1:t} = \mathbf{P}(\mathbf{o}_{t+1:t}|\mathbf{S}_t) = \mathbf{P}(|\mathbf{S}_t) = \mathbf{1}$ where $\mathbf{1}$ is a vector of 1s because $\mathbf{o}_{t+1:t}$ is an empty sequence and so the probability of observing it is 1.

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Example of Smoothing



- Goal: computing the smoothed estimate for the probability of rain at time $k = 1$ given the umbrella observations on days 1 and 2

$$P(R_1 \mid u_1, u_2) = \alpha P(R_1 \mid u_1) P(u_2 \mid R_1)$$

we know $P(R_1 \mid u_1) = \langle 0.818, 0.182 \rangle$

while $P(u_2 \mid R_1)$ can be computed by applying the backward recursion

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

so that $P(R_1 \mid u_1, u_2) = \alpha \langle 0.818, 0.182 \rangle \times \langle 0.69, 0.41 \rangle \approx \langle 0.883, 0.117 \rangle$

- Note that the smoothed estimate for rain on day 1 is higher than the filtered estimate (0.818) because the umbrella on day 2 makes it more likely to have rained on day 2; in turn, because rain tends to persist, that makes it more likely to have rained on day 1.

Forward-Backward Algorithm for Smoothing

- Both the forward and backward recursions take a constant amount of time per step; hence, the time complexity of smoothing at time step k with respect to observation $\mathbf{o}_{1:t}$ is $O(t)$
- If we want to smooth the whole sequence, one obvious method is simply to run the whole smoothing process once for each time step to be smoothed.
  - This results in a time complexity of $O(t^2)$
- A better approach uses a simple application of dynamic programming to reduce the complexity to $O(t)$
  - The key to the linear-time algorithm is to *record the results* of forward filtering over the whole sequence
  - Then we run the backward recursion from t down to 1, computing the smoothed estimate at each step k from the computed backward message $\mathbf{b}_{k+1:t}$ and stored forward message $\mathbf{f}_{1:k}$

function FORWARD-BACKWARD(**ev**, *prior*) **returns** a vector of probability distributions
  inputs: **ev**, a vector of evidence values for steps $1, \ldots, t$
    prior, the prior distribution on the initial state, **P**(S_0)
  local variables: **fv**, a vector of forward messages for steps $0, \ldots, t$
    b, a representation of the backward message, initially all 1s
    sv, a vector of smoothed estimates for steps $1, \ldots, t$

  fv[0] $\leftarrow$ *prior*
  for $i = 1$ **to** t **do**
    fv[i] $\leftarrow$ FORWARD(**fv**[$i-1$], **ev**[i])
  for $i = t$ **down to** 1 **do**
    sv[i] $\leftarrow$ NORMALIZE(**fv**[i] $\times$ **b**)
    b $\leftarrow$ BACKWARD(**b**, **ev**[i])
  return sv

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Drawbacks of Forward-Backward Algorithm in Practice

- Space complexity can be too high when the state space is large and the sequences are long
  - It uses $O(|\mathbf{f}|t)$ space where $|\mathbf{f}|$ is the size of the representation of the forward message
  - Can be reduced to $O(|\mathbf{f}|\log t)$ with a concomitant increase in the time complexity by a factor of log t

- It needs to be modified to work in an online setting where smoothed estimates must be computed for earlier time slices as new observations are continuously added to the end of the sequence
  - The most common requirement is for fixed-lag smoothing, which requires computing the smoothed estimate $\mathbf{P}(\mathbf{S}_{t-d}|\mathbf{o}_{1:t})$ for fixed d,
    i.e. smoothing is done for time step that is d steps behind the current time
  - Inefficient to run Forward-Backward over the d-step window for each sensor observation
  - Instead, fixed-lag smoothing can, in some cases, be done in constant time per update independent of d

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Finding the Most Likely Sequence

- Problem: given sensor observation sequence $\mathbf{o}_{1:t}$ what state sequence is most likely to explain this?
- Brute force approach
  - enumerate all state sequences and computer their likelihood: expensive!
- Incorrect approach
  - use smoothing to find the posterior distribution for the state at each time step; then construct the sequence, using at each step the state that is most likely according to the posterior
  - most likely sequence ≠ sequence of most likely states
    - posterior distributions computed by smoothing are distributions over *single* time steps, whereas to find the most likely *sequence* we must consider *joint* probabilities over all the time steps.
- Efficient linear time (and space) algorithm: the Viterbi algorithm

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Viterbi Algorithm

- View each state sequence as a path through a graph whose nodes are the possible states at each time step
- The likelihood of any path is the product of the transition probabilities along the path and the probabilities of the given observations at each state
- Because of the Markov property, there is a recursive relationship between most likely paths to each state $\mathbf{s}_{t+1}$ and most likely paths to each state $\mathbf{s}_t$
  - Most likely path to each $\mathbf{s}_{t+1}$ = most likely path to some $\mathbf{s}_t$ plus one more step

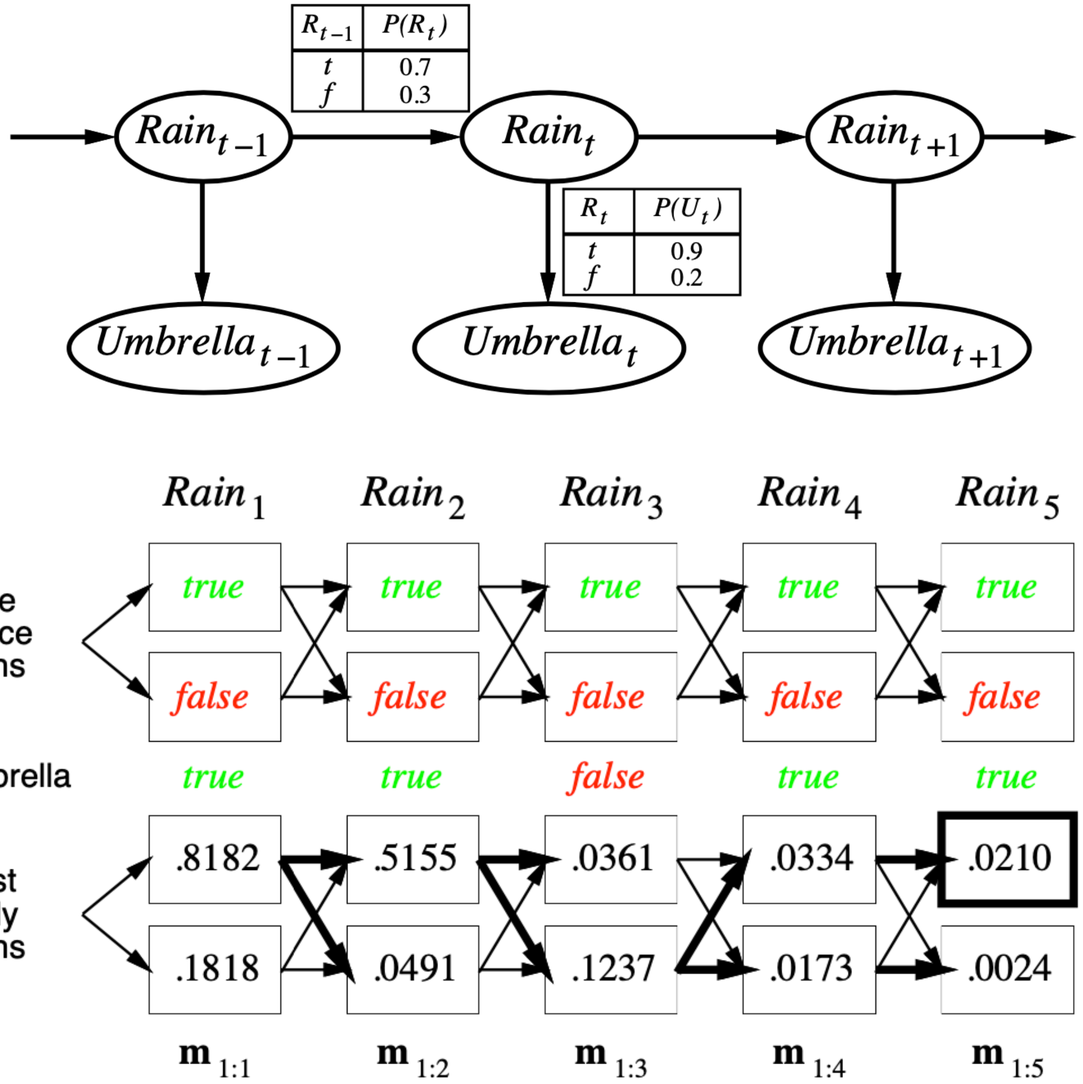
$$\max_{\mathbf{s}_{1:t}} \mathbf{P}(\mathbf{s}_{1:t}, \mathbf{S}_{t+1} \mid \mathbf{o}_{1:t+1}) = \mathbf{P}(\mathbf{o}_{t+1} \mid \mathbf{S}_{t+1}) \max_{\mathbf{s}_t}(\mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{s}_t) \max_{\mathbf{s}_{1:t-1}} \mathbf{P}(\mathbf{s}_{1:t-1}, \mathbf{s}_t \mid \mathbf{o}_{1:t}))$$

or, $\mathbf{m}_{1:t+1} = \mathbf{P}(\mathbf{o}_{t+1} \mid \mathbf{S}_{t+1}) \max_{\mathbf{s}_t}(\mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{s}_t)\mathbf{m}_{1:t})$ where $\mathbf{m}_{1:t} = \max_{\mathbf{s}_{1:t-1}} \mathbf{P}(\mathbf{s}_{1:t-1}, \mathbf{S}_t \mid \mathbf{o}_{1:t}))$, and $\mathbf{m}_{1:0} = \mathbf{P}(\mathbf{S}_0)$

  - Similar to filtering algorithm: it starts at time 0 and and then runs forward along the sequence, computing the vector message $\mathbf{m}$
  - Eventually $\mathbf{m}_{1:t}$ will contain the probability for the most likely sequence reaching each of the final states
    - One can thus easily select the final state of the most likely sequence overall
    - In order to identify the actual sequence, as opposed to just computing its probability, the algorithm will also need to record, for each state, the best state that leads to it

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Example of Most Likely Sequence

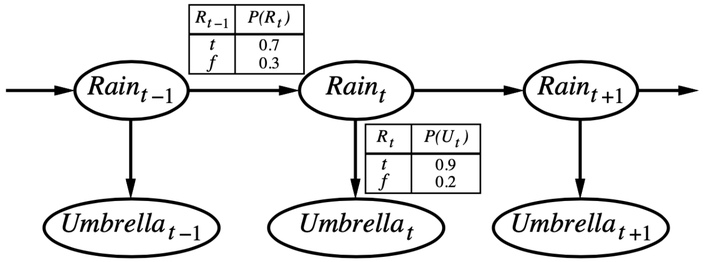


Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Hidden Markov Models

- A temporal probabilistic model in which the process state is described by a *single*, *discrete* RV
  - The possible values of the variable are the possible states of the world
- The restricted structure allows for a simple and elegant matrix implementation of all the basic algorithms
  - Let the state variable be an integer $S_t \in \{1,2,\cdots\ S\}$ where S is the number of possible states
  - Transition model $\mathbf{P}(S_t \mid S_{t-1})$ becomes an $S \times S$ matrix $\mathbf{T}$ where $\mathbf{T}_{ij} = P(S_t = j \mid S_{t-1} = i)$
  - Rain/Umbrella example

$$\mathbf{T} = \mathbf{P}(S_t = S_{t-1}) = \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix}$$



- What happens if one has a model with two or more state variables?
  - One can still fit it into the HMM framework by combining the variables into a single “megavariable” whose values are all possible tuples of values of the individual state variables.

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach”

Hidden Markov Models (contd.)

- Although the state must be a single, discrete variable, there is no similar restriction on observations.
  - This is because observations variations are known, so that there is no need to keep track their distributions
  - If a variable is not observed, it can simply be dropped from the model for that time step.
  - There can be many observation variables, both discrete and continuous
- However, one can also put sensor model in matrix form
  - Because observation $\mathbf{o}_t$ at every time step t is known, only need to specify for each state how likely is $\mathbf{o}_t$ to occur, i.e. $P(\mathbf{o}_t \mid S_t = i) \quad \forall i \in \{1,2,\cdots \ S\}$
  - Specified as $S \times S$ diagonal matrix $\mathbf{O}_t$ where the i-th diagonal entry is $\mathbf{P}(\mathbf{o}_t \mid S_t = i)$
- Umbrella world:
  - If $Umbrella_t = true$ then

$$\mathbf{O}_t = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.2 \end{pmatrix}$$

  - and if $Umbrella_t = false$ then

$$\mathbf{O}_t = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.8 \end{pmatrix}$$

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Forward-Backward Algorithm in Matrix-Vector Form

- Column vectors to represent the forward & backward messages
  - forward equation becomes:

$$\mathbf{f}_{1:t+1} = \alpha \mathbf{O}_{t+1} \mathbf{T}^{\top} \mathbf{f}_{1:t}$$

  - backward equation becomes:

$$\mathbf{b}_{k+1:t} = \mathbf{T} \mathbf{O}_{k+1} \mathbf{b}_{k+2:t}$$

- Complexity
  - Time: $O(S^2 t)$
    - each step requires multiplying an S element vector with an $S \times S$ matrix
  - Space: $O(St)$
    - the forward pass stores stores t vectors of size S

- Matrix-Vector formulation allows for improvements too, e.g.,
  - constant space algorithm for smoothing, independent of t
  - online smoothing with fixed lag

function FORWARD-BACKWARD(**ev**, *prior*) **returns** a vector of probability distributions
  inputs: **ev**, a vector of evidence values for steps $1, \ldots, t$
    prior, the prior distribution on the initial state, $\mathbf{P}(S_0)$
  local variables: **fv**, a vector of forward messages for steps $0, \ldots, t$
    b, a representation of the backward message, initially all 1s
    sv, a vector of smoothed estimates for steps $1, \ldots, t$

  fv$[0] \leftarrow$ *prior*
  for $i = 1$ **to** t **do**
    fv$[i] \leftarrow$ FORWARD(**fv**$[i-1]$, **ev**$[i]$)
  for $i = t$ **down to** 1 **do**
    sv$[i] \leftarrow$ NORMALIZE(**fv**$[i] \times$ **b**)
    b $\leftarrow$ BACKWARD(**b**, **ev**$[i]$)
  return sv

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Constant Space Algorithm for Smoothing (independent of t)

- Idea: Smoothing for any particular time index k requires the simultaneous presence of both the forward and backward messages $\mathbf{f}_{1:k}$ and $\mathbf{b}_{k+1:t}$

$$P(S_k \mid \mathbf{o}_{1:t}) = \alpha \mathbf{f}_{1:k} \times \mathbf{b}_{k+1:1}$$

- Forward-Backward algorithm achieves this by storing the $\mathbf{f}$s computed on the forward pass so that they are available during the backward pass.
- Instead, one can do a single pass that propagates both $\mathbf{f}$ and $\mathbf{b}$ in the same direction
  - First do standard forward pass to compute $\mathbf{f}_{t:t}$
    - forgetting all the intermediate results
  - Then run backward pass for both $\mathbf{f}$ and $\mathbf{b}$ together using a backward propagation of $\mathbf{f}$ as

$$\mathbf{f}_{1:t} = \alpha'(\mathbf{T}^{\top})^{-1}\mathbf{O}_{t+1}^{-1}\mathbf{f}_{1:t+1}$$

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Online Smoothing with Fixed Lag d

- Goal: smooth at time $t-d$ when the current time is t, i.e. compute $\alpha \mathbf{f}_{1:t-d} \times \mathbf{b}_{t-d+1:1}$
- Then, when a new observation arrives, we need to compute $\alpha \mathbf{f}_{1:t-d+1} \times \mathbf{b}_{t-d+2:1}$
- How to do this incrementally?
  - First, we can compute $\mathbf{f}_{1:t-d+1}$ from $\mathbf{f}_{1:t-d}$ using the standard filtering process

$$P(S_{t+1} \mid \mathbf{o}_{1:t+1}) = \alpha P(\mathbf{o}_{t+1} \mid S_{t+1}) \sum_{s_t} P(S_{t+1} \mid s_t) P(s_t \mid \mathbf{o}_{1:t})$$

  - But, there is no analogous simple relationship between new $b_{t-d+2:1}$ and old $b_{t-d+1:1}$ ☹️
    - Trick: repeatedly use $\mathbf{b}_{k+1:t} = \mathbf{T}\mathbf{O}_{k+1}\mathbf{b}_{k+2:t}$ to get

$$\mathbf{b}_{t-d+1:1} = \left(\prod_{i=t-d+1}^{t} \mathbf{T}\mathbf{O}_i\right)\mathbf{b}_{t+1:t} = \mathbf{B}_{t-d+1:t}\mathbf{1} \quad \text{and} \quad \mathbf{b}_{t-d+2:1} = \left(\prod_{i=t-d+2}^{t+1} \mathbf{T}\mathbf{O}_i\right)\mathbf{b}_{t+2:t+1} = \mathbf{B}_{t-d+2:t+1}\mathbf{1}$$

which gives the incremental update $\mathbf{B}_{t-d+2:t+1} = \mathbf{O}_{t-d+1}^{-1}\mathbf{T}^{-1}\mathbf{B}_{t-d+1:t}\mathbf{T}\mathbf{O}_{t+1}$

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

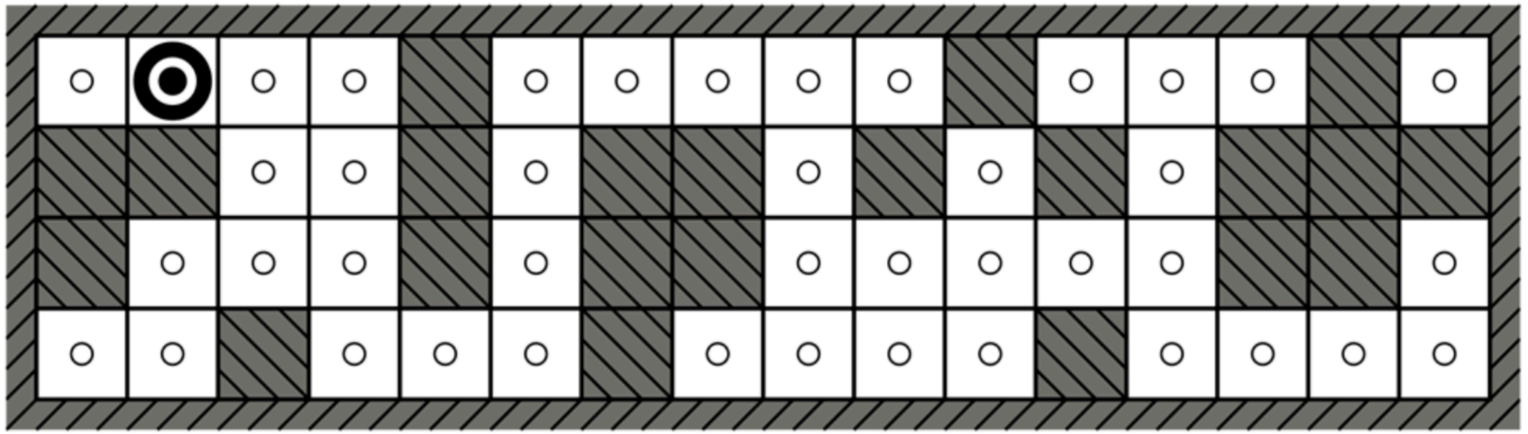
Online Algorithm for Smoothing with Fixed Lag d

function FIXED-LAG-SMOOTHING(e_t, hmm, d) **returns** a distribution over $\mathbf{X}_{t-d}$
  inputs: e_t, the current evidence for time step t
    hmm, a hidden Markov model with $S \times S$ transition matrix $\mathbf{T}$
    d, the length of the lag for smoothing
  persistent: t, the current time, initially 1
    $\mathbf{f}$, the forward message $\mathbf{P}(X_t \mid e_{1:t})$, initially hmm.PRIOR
    $\mathbf{B}$, the d-step backward transformation matrix, initially the identity matrix
    $e_{t-d:t}$, double-ended list of evidence from $t-d$ to t, initially empty
  local variables: $\mathbf{O}_{t-d}, \mathbf{O}_t$, diagonal matrices containing the sensor model information

  add e_t to the end of $e_{t-d:t}$
  $\mathbf{O}_t \leftarrow$ diagonal matrix containing $\mathbf{P}(e_t \mid X_t)$
  if $t > d$ **then**
    $\mathbf{f} \leftarrow$ FORWARD$(\mathbf{f}, e_{t-d})$
    remove e_{t-d-1} from the beginning of $e_{t-d:t}$
    $\mathbf{O}_{t-d} \leftarrow$ diagonal matrix containing $\mathbf{P}(e_{t-d} \mid X_{t-d})$
    $\mathbf{B} \leftarrow \mathbf{O}_{t-d}^{-1}\mathbf{T}^{-1}\mathbf{B}\mathbf{T}\mathbf{O}_t$
  else $\mathbf{B} \leftarrow \mathbf{B}\mathbf{T}\mathbf{O}_t$
  $t \leftarrow t+1$
  if $t > d+1$ **then return** NORMALIZE$(\mathbf{f} \times \mathbf{B1})$ **else return** null

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

HMM Application: Vacuum Robot Localization



- Discrete grid where $s_t \in 1,2,3,\cdots 42$ represents the current location of the robot
- Robot can move randomly to any of the adjacent empty squares from its current location
  i.e., $P(S_{t+1} = j \mid S_t = i) = \mathbf{T}_{ij} = 1/(\#\text{ of empty adjacent locations})$ if $j \in$ NEIGHBORS(i) else 0
- Noisy binary sensors with error rate ϵ report whether the four adjacent NESW locations are empty or not,
  i.e. O_t is a 4-bit sequence taking values such as $o_t = 1011$
  so that $P(O_t = o_t \mid S_t = i) = (\mathbf{o}_t)_{ii} = (1-\epsilon)^{4-d_{it}}\epsilon^{d_{it}}$
  where $d_{it} =$ is the discrepancy between the true values for square i and the actual reading o_t
- The robot can use filtering to estimate current location, and smoothing to get past location

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Limitations of HMM

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Limitations of HMM

- Consider vacuum robot that has the policy of going straight for as long as it can
  - Then the state will need to including both *location* and *heading,* and so could take $42 \times 8 = 168$ values, so that $\mathbf{T}$ will be of size $168^2 = 28{,}224$
    - still manageable though large

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Limitations of HMM

- Consider vacuum robot that has the policy of going straight for as long as it can
  - Then the state will need to including both *location* and *heading,* and so could take $42 \times 8 = 168$ values, so that $\mathbf{T}$ will be of size $168^2 = 28{,}224$
    - still manageable though large
- What if we add the possibility of dirt in each of the 42 squares?
  - Now the number of states is multiplied by 2^{42} and $\mathbf{T}$ has more than 10^{29} entries 😕

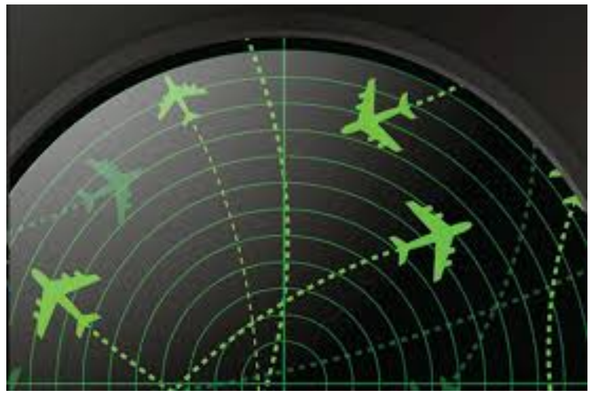
Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Limitations of HMM

- Consider vacuum robot that has the policy of going straight for as long as it can
  - Then the state will need to including both *location* and *heading,* and so could take $42 \times 8 = 168$ values, so that $\mathbf{T}$ will be of size $168^2 = 28{,}224$
    - still manageable though large
- What if we add the possibility of dirt in each of the 42 squares?
  - Now the number of states is multiplied by 2^{42} and $\mathbf{T}$ has more than 10^{29} entries 😕
- In general if state is composed of n discrete variables that can take at most d values each, then the state transition matrix will have size $O(d^{2n})$ and the per-update computation time will be also $O(d^{2n})$
  - While HMMs used a lot (e.g. speech recognition), they are fundamentally limited in their ability to represent complex processes
  - Problem: represent states as integers and don't exploit any internal structure

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Kalman Filter

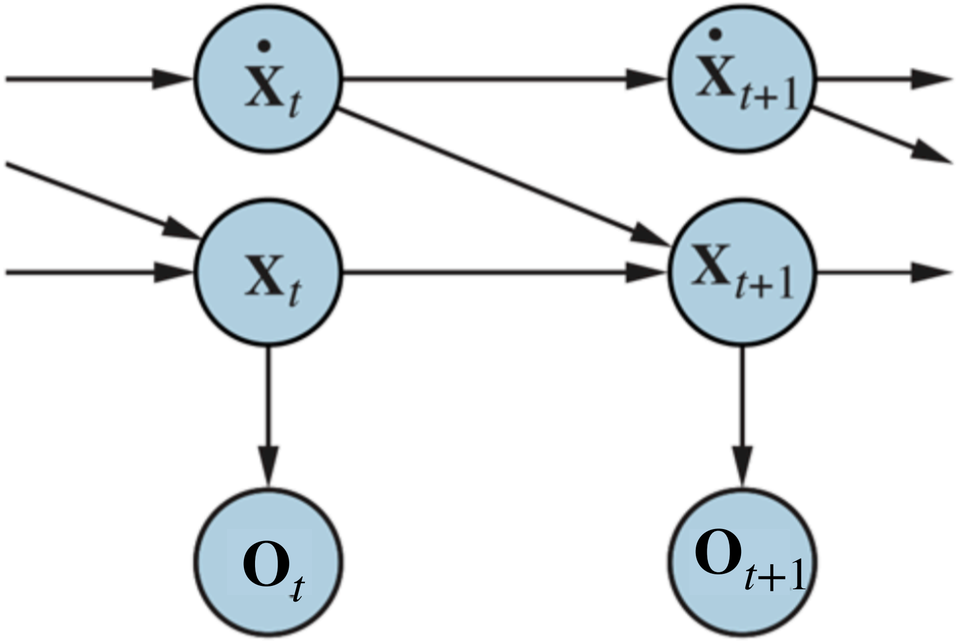


- In many applications we need to reconstruct trajectory of some variable of interest (position, speed, blood sugar level, etc.) from irregular and noisy observations
- If variables are discrete, one can use HMMs
- Kalman Filter targets the case when the variables of interest are continuous
  - E.g. flight of incoming enemy aircraft or a pedestrian on the road may be specified by six continuous variables: three for position (X_t, Y_t, Z_t) and three for velocity $(\dot{X}_t, \dot{Y}_t, \dot{Z}_t)$
- Uses **linear-Gaussian** distributions for conditional densities to represent the transition and sensor models
  - next state $\mathbf{S}_{t+1}$ is a linear function of current state $\mathbf{S}_t$ plus some Gaussian noise
  - sensor observation $\mathbf{O}_t$ is a linear function of current state $\mathbf{S}_t$ plus some Gaussian noise

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Consider Tracking a Vehicle from Sporadic Observations

- $\mathbf{S}_t = (X_t, Y_t, Z_t, \dot{X}_t, \dot{Y}_t, \dot{Z}_t)$
- Consider just X_t coordinate and let Δ be the time interval between observations
- Then, assuming constant velocity during the interval and no noise $X_{t+\Delta} = X_t + \dot{X}_t\Delta$
- Adding Gaussian noise (to account for wind variation, etc.), we obtain a linear–Gaussian transition model: $P(X_{t+\Delta} = x_{t+\Delta} \mid X_t = x_t, \dot{X}_t = \dot{x}_t) = N(x_{t+\Delta}; x_t + \dot{x}_t\Delta, \sigma^2)$
- Note: this is a specific case of linear-Gaussian, and when generalizing to d dimensions then we need to use multivariate Gaussian with a d-element mean and a $d \times d$ covariance matrix Σ



Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Updating Gaussian Distributions

- Special property of linear-Gaussian: *closed* under Bayesian updating
  - i.e. given an observation, the posterior distribution is still in the linear–Gaussian family

- Recall filtering: $\mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{o}_{1:t+1}) = \alpha \mathbf{P}(\mathbf{o}_{t+1} \mid \mathbf{S}_{t+1}) \sum_{\mathbf{s}_t} \mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{s}_t) \mathbf{P}(\mathbf{s}_t \mid \mathbf{o}_{1:t})$

  sensor model | transition model | recursion

- If current distribution $\mathbf{P}(\mathbf{S}_t \mid \mathbf{o}_{1:t})$ is Gaussian and the transition model $\mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{s}_t)$ is linear-Gaussian then the the one-step predicted distribution given by the following is also Gaussian.

$$\mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{o}_{1:t}) = \int_{\mathbf{s}_t} \mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{s}_t) \mathbf{P}(\mathbf{s}_t \mid \mathbf{o}_{1:t}) d\mathbf{s}_t$$

- If the prediction $\mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{o}_{1:t})$ is Gaussian and the sensor model $\mathbf{P}(\mathbf{o}_{t+1} \mid \mathbf{S}_{t+1})$ is linear–Gaussian, then, after conditioning on the new observation $\mathbf{o}_{t+1}$, the updated distribution below is also Gaussian.

$$\mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{o}_{1:t+1}) = \alpha \mathbf{P}(\mathbf{o}_{t+1} \mid \mathbf{S}_{t+1}) \mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{o}_{1:t})$$

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Kalman Filter in Summary

- FORWARD operator for Kalman filtering takes a Gaussian forward message $\mathbf{f}_{1:t}$ specified by a mean μ_t and covariance Σ_t, and produces a new multivariate Gaussian forward message $\mathbf{f}_{1:t+1}$ specified by a mean μ_{t+1} and covariance Σ_{t+1}.
  - This translates into computing a new mean and covariance from the previous mean and covariance
- So if we start with a Gaussian prior $\mathbf{f}_{1:0} = \mathbf{P}(\mathbf{S}_0) = N(\mu_0, \Sigma_0)$, filtering with a linear–Gaussian model produces a Gaussian state distribution for all time
  - Not only nice and elegant, but also important: except for a few special cases such as this, filtering with continuous or hybrid (discrete and continuous) networks generates state distributions whose representation grows without bound over time
- Simple univariate case: state variable X_t with nosy observation Z_t

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

- General multivariate case

$$\begin{aligned} P(\mathbf{x}_{t+1}|\mathbf{x}_t) &= N(\mathbf{x}_{t+1}; \mathbf{F}\mathbf{x}_t, \Sigma_x) \\ P(\mathbf{z}_t|\mathbf{x}_t) &= N(\mathbf{z}_t; \mathbf{H}\mathbf{x}_t, \Sigma_z), \end{aligned}$$

$$\begin{aligned} \mu_{t+1} &= \mathbf{F}\mu_t + \mathbf{K}_{t+1}(\mathbf{z}_{t+1} - \mathbf{H}\mathbf{F}\mu_t) \\ \Sigma_{t+1} &= (\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H})(\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x) \end{aligned}$$

$$\text{where } \mathbf{K}_{t+1} = (\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)\mathbf{H}^\top(\mathbf{H}(\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)\mathbf{H}^\top + \Sigma_z)^{-1}$$ **Kalman gain matrix**

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Making Intuitive Sense of Kalman Filter General Case

$$
\begin{aligned}
P(\mathbf{x}_{t+1}|\mathbf{x}_t) &= N(\mathbf{x}_{t+1}; \mathbf{F}\mathbf{x}_t, \Sigma_x) \\
P(\mathbf{z}_t|\mathbf{x}_t) &= N(\mathbf{z}_t; \mathbf{H}\mathbf{x}_t, \Sigma_z), \\
\mu_{t+1} &= \mathbf{F}\mu_t + \mathbf{K}_{t+1}(\mathbf{z}_{t+1} - \mathbf{H}\mathbf{F}\mu_t) \\
\Sigma_{t+1} &= (\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H})(\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)
\end{aligned}
$$

$$
\text{where } \mathbf{K}_{t+1} = (\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)\mathbf{H}^\top(\mathbf{H}(\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)\mathbf{H}^\top + \Sigma_z)^{-1}
$$

Kalman gain matrix

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Making Intuitive Sense of Kalman Filter General Case

Predicted state at $t+1$

$$
\begin{aligned}
P(\mathbf{x}_{t+1}|\mathbf{x}_t) &= N(\mathbf{x}_{t+1}; \mathbf{F}\mathbf{x}_t, \Sigma_x) \\
P(\mathbf{z}_t|\mathbf{x}_t) &= N(\mathbf{z}_t; \mathbf{H}\mathbf{x}_t, \Sigma_z), \\
\mu_{t+1} &= \mathbf{F}\mu_t + \mathbf{K}_{t+1}(\mathbf{z}_{t+1} - \mathbf{H}\mathbf{F}\mu_t) \\
\Sigma_{t+1} &= (\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H})(\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)
\end{aligned}
$$

$$
\text{where } \mathbf{K}_{t+1} = (\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)\mathbf{H}^\top(\mathbf{H}(\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)\mathbf{H}^\top + \Sigma_z)^{-1}
$$

Kalman gain matrix

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Making Intuitive Sense of Kalman Filter General Case

Predicted state at $t+1$

Predicted observation at $t+1$

$$
\begin{aligned}
P(\mathbf{x}_{t+1}|\mathbf{x}_t) &= N(\mathbf{x}_{t+1}; \mathbf{F}\mathbf{x}_t, \Sigma_x) \\
P(\mathbf{z}_t|\mathbf{x}_t) &= N(\mathbf{z}_t; \mathbf{H}\mathbf{x}_t, \Sigma_z), \\
\mu_{t+1} &= \mathbf{F}\mu_t + \mathbf{K}_{t+1}(\mathbf{z}_{t+1} - \mathbf{H}\mathbf{F}\mu_t) \\
\Sigma_{t+1} &= (\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H})(\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)
\end{aligned}
$$

$$
\text{where } \mathbf{K}_{t+1} = (\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)\mathbf{H}^\top(\mathbf{H}(\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)\mathbf{H}^\top + \Sigma_z)^{-1}
$$

Kalman gain matrix

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Making Intuitive Sense of Kalman Filter General Case

Predicted state at $t+1$

Predicted observation at $t+1$

Error in the predicted observation

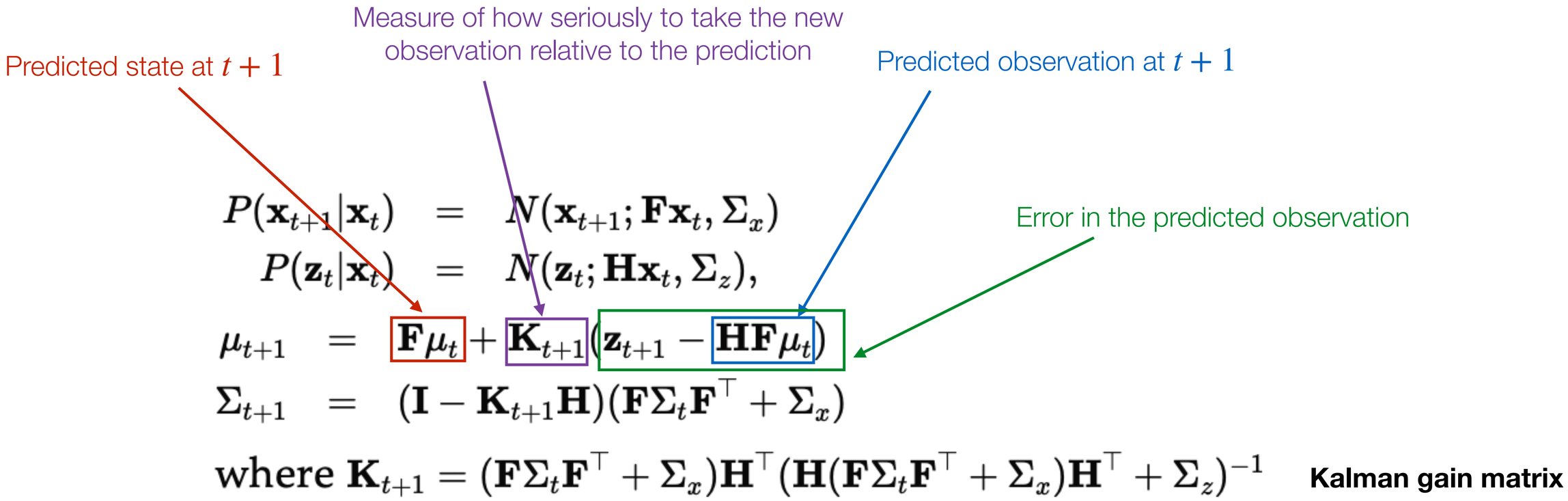
$$
\begin{aligned}
P(\mathbf{x}_{t+1}|\mathbf{x}_t) &= N(\mathbf{x}_{t+1}; \mathbf{F}\mathbf{x}_t, \Sigma_x) \\
P(\mathbf{z}_t|\mathbf{x}_t) &= N(\mathbf{z}_t; \mathbf{H}\mathbf{x}_t, \Sigma_z), \\
\mu_{t+1} &= \mathbf{F}\mu_t + \mathbf{K}_{t+1}(\mathbf{z}_{t+1} - \mathbf{H}\mathbf{F}\mu_t) \\
\Sigma_{t+1} &= (\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H})(\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)
\end{aligned}
$$

$$
\text{where } \mathbf{K}_{t+1} = (\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)\mathbf{H}^\top(\mathbf{H}(\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)\mathbf{H}^\top + \Sigma_z)^{-1}
$$

Kalman gain matrix

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Making Intuitive Sense of Kalman Filter General Case



Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Real-time Computation of Kalman Filter

$$
\begin{aligned}
P(\mathbf{x}_{t+1}|\mathbf{x}_t) &= N(\mathbf{x}_{t+1}; \mathbf{F}\mathbf{x}_t, \Sigma_x) \\
P(\mathbf{z}_t|\mathbf{x}_t) &= N(\mathbf{z}_t; \mathbf{H}\mathbf{x}_t, \Sigma_z), \\
\mu_{t+1} &= \mathbf{F}\mu_t + \mathbf{K}_{t+1}(\mathbf{z}_{t+1} - \mathbf{H}\mathbf{F}\mu_t) \\
\Sigma_{t+1} &= (\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H})(\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)
\end{aligned}
$$

$$
\text{where } \mathbf{K}_{t+1} = (\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)\mathbf{H}^\top(\mathbf{H}(\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)\mathbf{H}^\top + \Sigma_z)^{-1}
$$

Kalman gain matrix

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Real-time Computation of Kalman Filter

Independent of observations $\mathbf{z}_t$

$$
\begin{aligned}
P(\mathbf{x}_{t+1}|\mathbf{x}_t) &= N(\mathbf{x}_{t+1}; \mathbf{F}\mathbf{x}_t, \Sigma_x) \\
P(\mathbf{z}_t|\mathbf{x}_t) &= N(\mathbf{z}_t; \mathbf{H}\mathbf{x}_t, \Sigma_z), \\
\mu_{t+1} &= \mathbf{F}\mu_t + \mathbf{K}_{t+1}(\mathbf{z}_{t+1} - \mathbf{H}\mathbf{F}\mu_t) \\
\Sigma_{t+1} &= (\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H})(\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x) \\
\text{where } \mathbf{K}_{t+1} &= (\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)\mathbf{H}^\top(\mathbf{H}(\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)\mathbf{H}^\top + \Sigma_z)^{-1}
\end{aligned}
$$

Kalman gain matrix

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Real-time Computation of Kalman Filter

Independent of observations $\mathbf{z}_t$

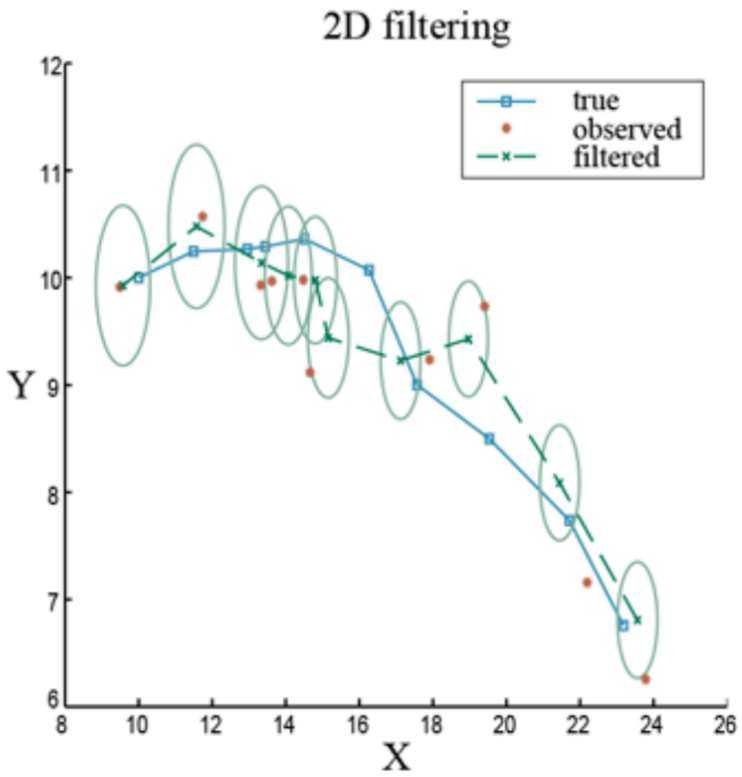
$$
\begin{aligned}
P(\mathbf{x}_{t+1}|\mathbf{x}_t) &= N(\mathbf{x}_{t+1}; \mathbf{F}\mathbf{x}_t, \Sigma_x) \\
P(\mathbf{z}_t|\mathbf{x}_t) &= N(\mathbf{z}_t; \mathbf{H}\mathbf{x}_t, \Sigma_z), \\
\mu_{t+1} &= \mathbf{F}\mu_t + \mathbf{K}_{t+1}(\mathbf{z}_{t+1} - \mathbf{H}\mathbf{F}\mu_t) \\
\Sigma_{t+1} &= (\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H})(\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x) \\
\text{where } \mathbf{K}_{t+1} &= (\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)\mathbf{H}^\top(\mathbf{H}(\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)\mathbf{H}^\top + \Sigma_z)^{-1}
\end{aligned}
$$

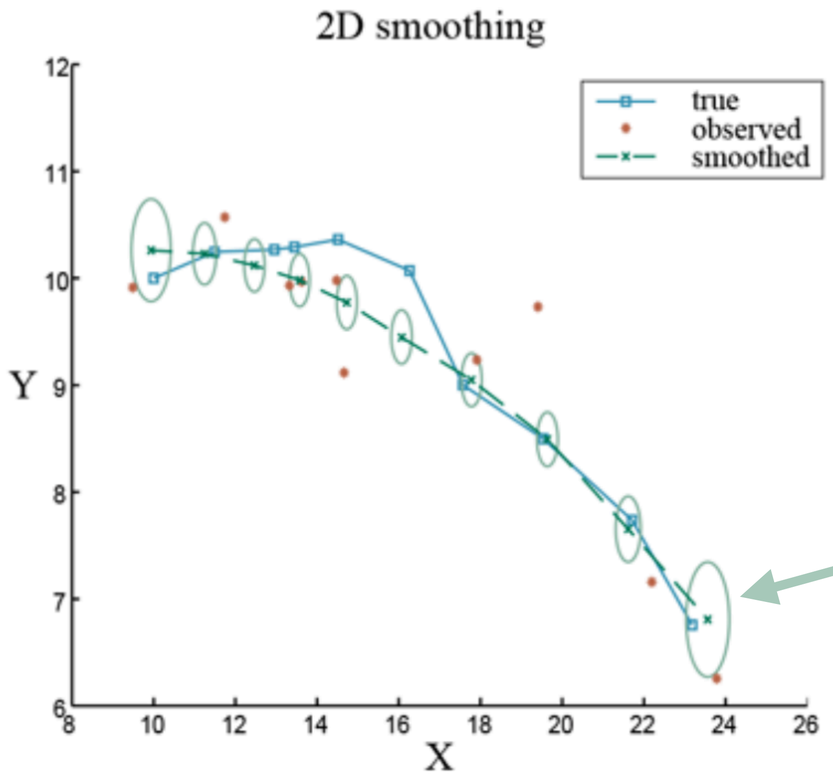
Kalman gain matrix

The sequence of values for Σ_t and $\mathbf{K}_t$ can be computed offline, and so calculations during online tracking are modest.

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Kalman Filtering and Smoothing for Object Moving in $X - Y$ Plane





ovals indicated the variance in the position estimate

- State variables are $\mathbf{X} = (X, Y, \dot{X}, \dot{Y})$ so that $\mathbf{F}, \Sigma_x, \mathbf{H}$, and Σ_z are 4×4 matrices
- Variance in the right plot (Kalman Smoothing) is much reduced except at the start and the end. *Why?*

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

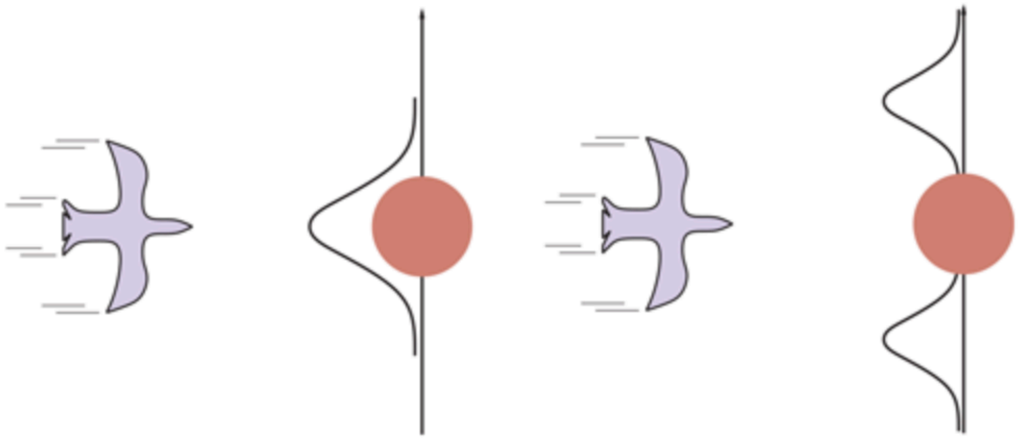
Applicability of Kalman Filtering

- Classical applications
  - radar tracking of aircrafts and missiles, acoustic tracking of submarines and ground vehicles, visual tracking of vehicles and people
- More esoteric applications
  - reconstruct particle trajectories from bubble chamber photographs, ocean currents from satellite surface measurements
- Applicable to any system with continuous state variables and noisy measurements
  - national economies, human body, chemical plants, nuclear reactors, plant ecosystems, etc.
- But results may not be valid or useful
  - strong assumption of linear-Gaussian transition and sensor models
- Extended Kalman Filter (EKF): overcome nonlinearities in the system being modeled
  - models the system as locally linear in $\mathbf{x}_t$ in the neighborhood of $\mathbf{x}_t = \mu_t$
  - works well for smooth, well-behaved systems
    - allows the tracker to maintain and update a Gaussian state distribution that is a reasonable approximation to the true posterior
- What about systems that are “unsmooth” and “poorly behaved”?

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach”

Switching Kalman Filter

- Track a bird as it flies through the jungle and appears to be heading at high speed straight for a tree trunk
- The Kalman filter, whether regular or extended, can make only a Gaussian prediction of the location of the bird, and the mean of this Gaussian will be centered on the trunk
- A reasonable model of the bird, on the other hand, would predict evasive action to one side or the other
  - Such a model is highly nonlinear, because the bird's decision varies sharply depending on its precise location relative to the trunk

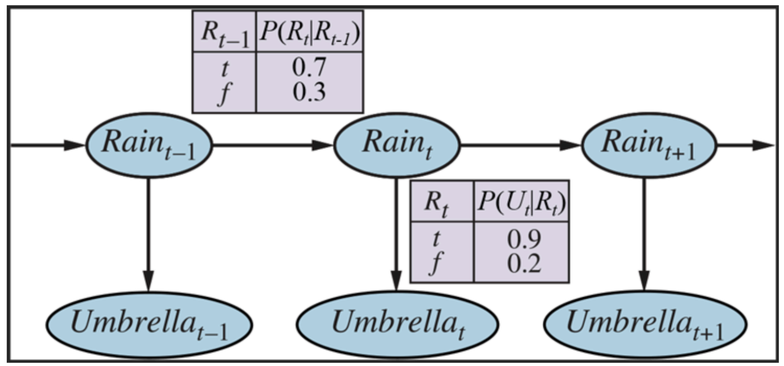


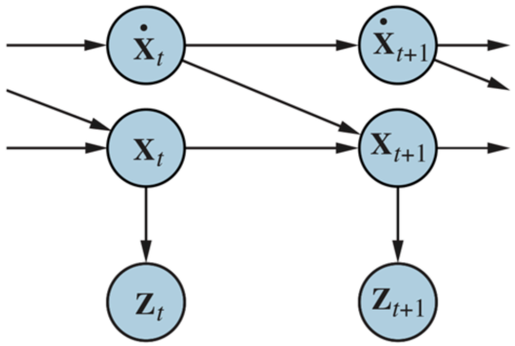
- Switching Kalman Filter: multiple Kalman filters run in parallel, each using a different model of the system
  - E.g., one for straight flight, one for sharp left turns, and one for sharp right turns
  - A weighted sum of predictions is used, where the weight depends on how well each filter fits the current data
  - A special case of the general dynamic Bayesian network model

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Dynamic Bayesian Networks (DBNs)

- Special case of Bayesian Networks: infinitely many variables grouped in time slices
  - Each slice can have any number of state variables $\mathbf{s_t}$ and observation variables $\mathbf{o_t}$





- For simplicity, we assume that
  - the variables, their links, and their conditional distributions are exactly replicated from slice to slice (*time-homogeneous*)
  - the DBN represents a first-order Markov process, i.e., each variable can have parents only in its own slice or the immediately preceding slice.

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

DBN vs. HMM

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

DBN vs. HMM

- Every hidden Markov model can be represented as a DBN with a single state variable and a single evidence variable

DBN vs. HMM

- Every hidden Markov model can be represented as a DBN with a single state variable and a single evidence variable

- Every discrete-variable DBN can be represented as an HMM
  - We can combine all the state variables in the DBN into a single state variable whose values are all possible tuples of values of the individual state variables

DBN vs. HMM

- Every hidden Markov model can be represented as a DBN with a single state variable and a single evidence variable

- Every discrete-variable DBN can be represented as an HMM
  - We can combine all the state variables in the DBN into a single state variable whose values are all possible tuples of values of the individual state variables

- If every HMM is a DBN and every DBN can be translated into an HMM, what's the difference?

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

DBN vs. HMM

- Every hidden Markov model can be represented as a DBN with a single state variable and a single evidence variable

- Every discrete-variable DBN can be represented as an HMM
  - We can combine all the state variables in the DBN into a single state variable whose values are all possible tuples of values of the individual state variables

- If every HMM is a DBN and every DBN can be translated into an HMM, what's the difference?

- The difference: *by decomposing the state of a complex system into its constituent variables, we can take advantage of sparseness in the temporal probability model.*

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

DBN vs. HMM

- Every hidden Markov model can be represented as a DBN with a single state variable and a single evidence variable

- Every discrete-variable DBN can be represented as an HMM
  - We can combine all the state variables in the DBN into a single state variable whose values are all possible tuples of values of the individual state variables

- If every HMM is a DBN and every DBN can be translated into an HMM, what's the difference?

- The difference: *by decomposing the state of a complex system into its constituent variables, we can take advantage of sparseness in the temporal probability model.*
- Consider: a temporal process with n discrete variables, each with up to d values
  - HMM model will need a transition matrix of size $O(d^{2n})$
  - DBN model has size $O(nd^k)$, assuming each node can have at most k parents
  - E.g. vacuum robot with 42 possibly dirty locations: size reduces from 5×10^{29} to a few thousands

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

DBN vs. Kalman Filter

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

DBN vs. Kalman Filter

- Every Kalman filter model can be represented in a DBN with continuous variables and linear–Gaussian conditional distributions

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

DBN vs. Kalman Filter

- Every Kalman filter model can be represented in a DBN with continuous variables and linear–Gaussian conditional distributions

- However, *not* every DBN can be represented by a Kalman filter model
  - In a Kalman filter, the current state distribution is always a single multivariate Gaussian distribution
    - e.g. a single “bump” in a particular location in a position tracking system
  - DBNs, on the other hand, can model arbitrary distributions
  - Many real-world applications require this

DBN vs. Kalman Filter

- Every Kalman filter model can be represented in a DBN with continuous variables and linear–Gaussian conditional distributions

- However, *not* every DBN can be represented by a Kalman filter model
  - In a Kalman filter, the current state distribution is always a single multivariate Gaussian distribution
    - e.g. a single “bump” in a particular location in a position tracking system
  - DBNs, on the other hand, can model arbitrary distributions
  - Many real-world applications require this

- E.g. consider a system that tracks location of my keys
  - They might be in my pocket, on the bedside table, on the kitchen counter, dangling from the front door, or locked in the car, …
  - A single Gaussian bump that included all these places would have to allocate significant probability to the keys being in mid-air above the front garden!

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach”

DBN vs. Kalman Filter

- Every Kalman filter model can be represented in a DBN with continuous variables and linear–Gaussian conditional distributions

- However, *not* every DBN can be represented by a Kalman filter model
  - In a Kalman filter, the current state distribution is always a single multivariate Gaussian distribution
    - e.g. a single “bump” in a particular location in a position tracking system
  - DBNs, on the other hand, can model arbitrary distributions
  - Many real-world applications require this

- E.g. consider a system that tracks location of my keys
  - They might be in my pocket, on the bedside table, on the kitchen counter, dangling from the front door, or locked in the car, …
  - A single Gaussian bump that included all these places would have to allocate significant probability to the keys being in mid-air above the front garden!

- Real world has purposive agents, obstacles, and pockets which introduce “nonlinearities” and require combinations of discrete and continuous variables in order to get reasonable models

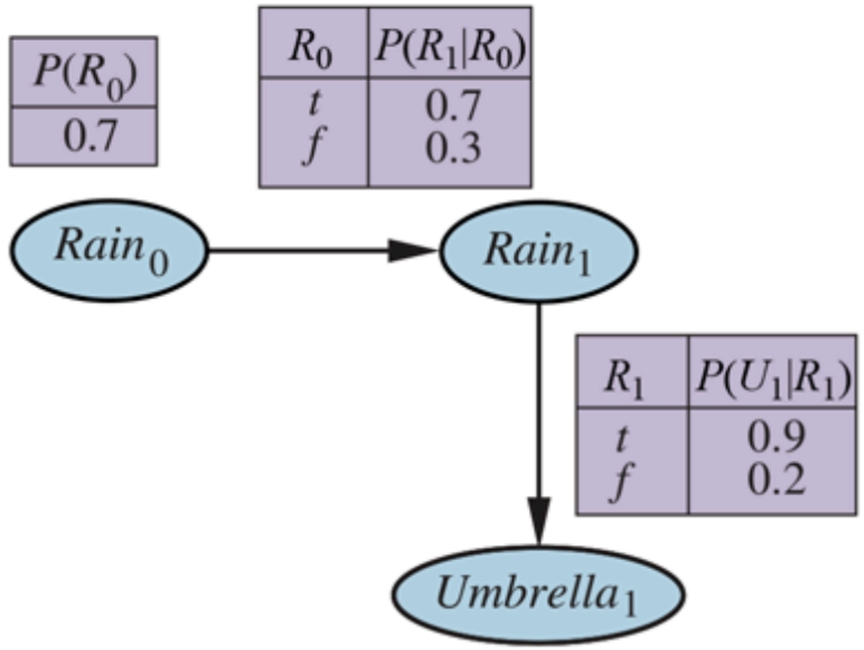
Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach”

Constructing DBNs

- To construct a DBN, one must specify three kinds of information: the prior distribution over the state variables, $\mathbf{P}(\mathbf{S}_0)$; the transition model, $\mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{S}_t)$; and the sensor model, $\mathbf{P}(\mathbf{O}_t \mid \mathbf{S}_t)$

- To specify the transition and sensor models, one must also specify the topology of the connections between successive slices and between the state and evidence variables.

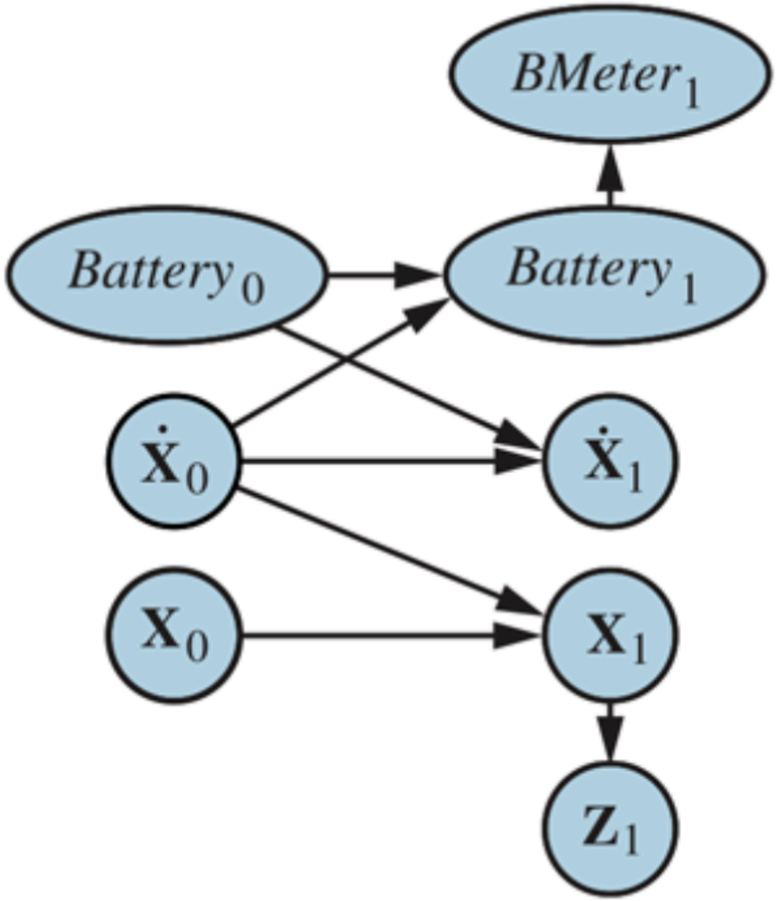
- Because the transition and sensor models are assumed to be time-homogeneous—the same for all t—one only needs to specify $\mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{S}_t)$ and $\mathbf{P}(\mathbf{O}_t \mid \mathbf{S}_t)$ for the first slice
  - the complete DBN with an unbounded number of time slices can be constructed as needed by copying the first slice



Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Example: Battery Powered Robot Moving in $X - Y$ Plane

- State Variables
  - Position: $\mathbf{X}_t = (X_t, Y_t)$
  - Velocity: $\dot{\mathbf{X}}_t = (\dot{X}_t, \dot{Y}_t)$
  - Battery Status: $Battery_t$
- Topology:
  - The position at the next time step depends on the current position and velocity
  - The velocity at the next step depends on the current velocity and the state of the battery
  - The battery state at the next step depends on the current velocity and the current state of the battery
- Sensor observations
  - Some method of measuring position—perhaps a fixed camera or onboard GPS (Global Positioning System)—yielding $\mathbf{z}_t$
  - Battery charge meter yielding $BMeter_t$



Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

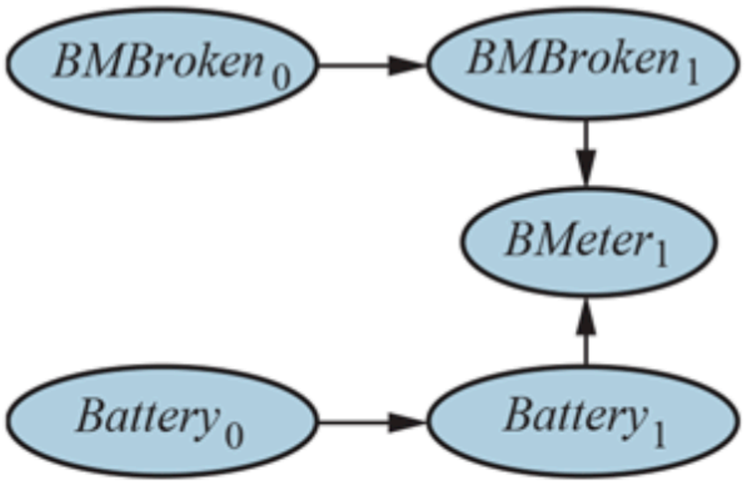
A Deeper Look at the Sensor Model $BMeter_t$

- Assume discrete values in the range 0 to 5 for both $Battery_t$ and $BMeter_t$
  - If the meter was always accurate then the conditional probability table (CPT) for $\mathbf{P}(BMeter_t \mid Battery_t)$ will have probabilities of 1.0 along the diagonal and 0 elsewhere

- Reality: noise always creeps into measurements
  - For continuous measurements, a Gaussian distribution with a small variance might be used
  - For our discrete variables, we can approximate a Gaussian
    - using a distribution in which the probability of error drops off in the appropriate way, so that the probability of a large error is very small

- However, in real world sensors also *fail*
  - When a sensor fails, it simply sends nonsense
  - **Transient failure**: the sensor occasionally decides to send nonsense
    - model by a higher probability for large errors than gaussian noise
  - **Persistent failure**: the sensor fails and stays failed
    - model by an additional state variable $BMBroken$
  - Other issues: sensor drift, sudden decalibration, and the effects of exogenous conditions (such as weather) on sensor readings

B_0	$P(B_1)$
t	1.000
f	0.001



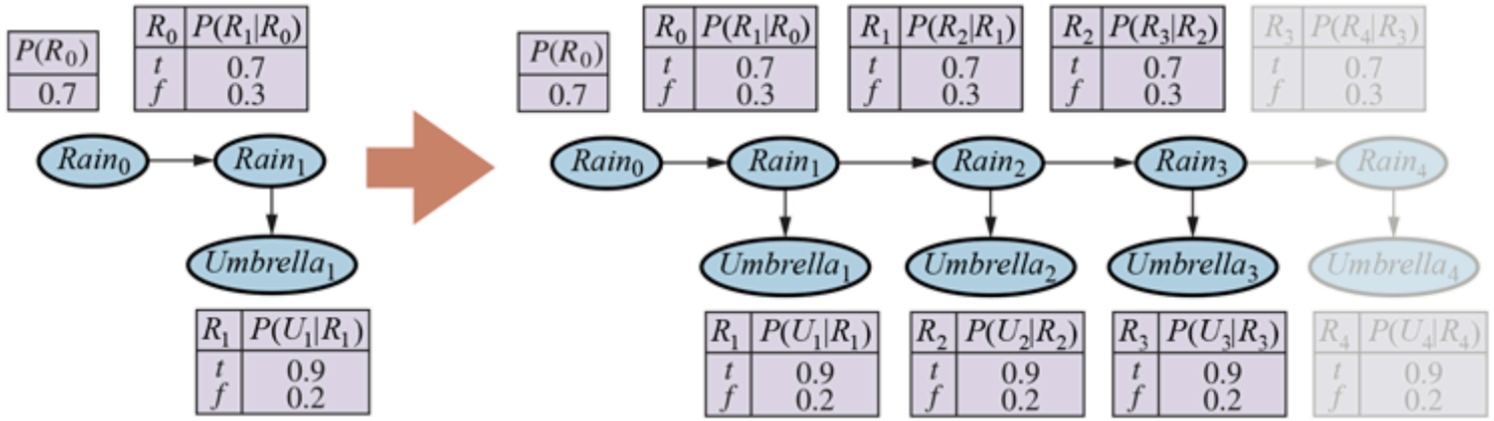
Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Exact inference in DBNs by Unrolling

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Exact inference in DBNs by Unrolling

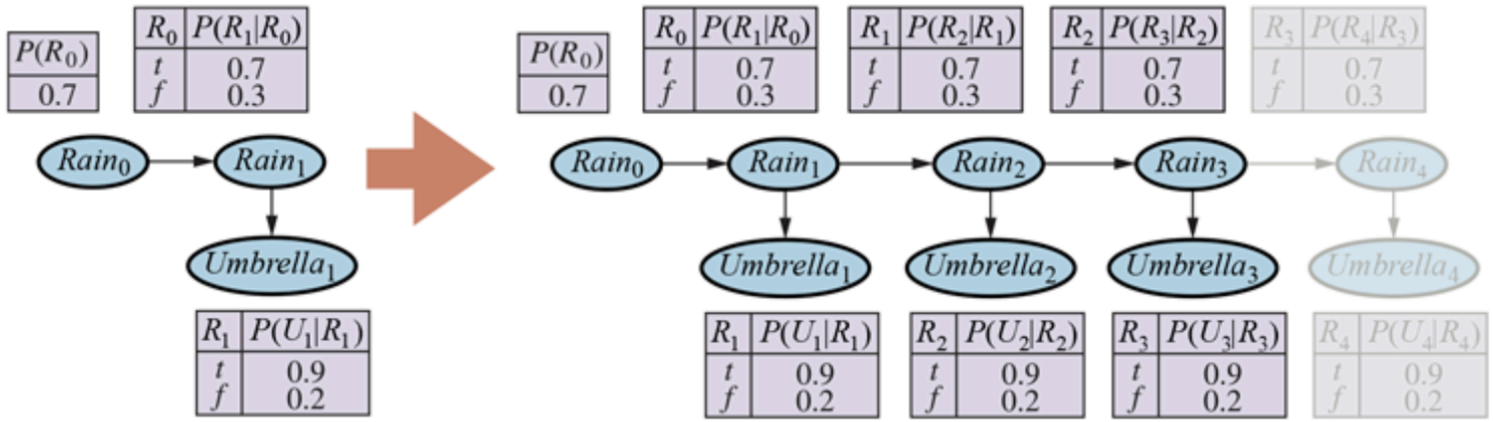
- Given a sequence of observations, one can construct the full Bayesian network representation of a DBN by replicating slices until the network is large enough to accommodate the observations



Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Exact inference in DBNs by Unrolling

- Given a sequence of observations, one can construct the full Bayesian network representation of a DBN by replicating slices until the network is large enough to accommodate the observations

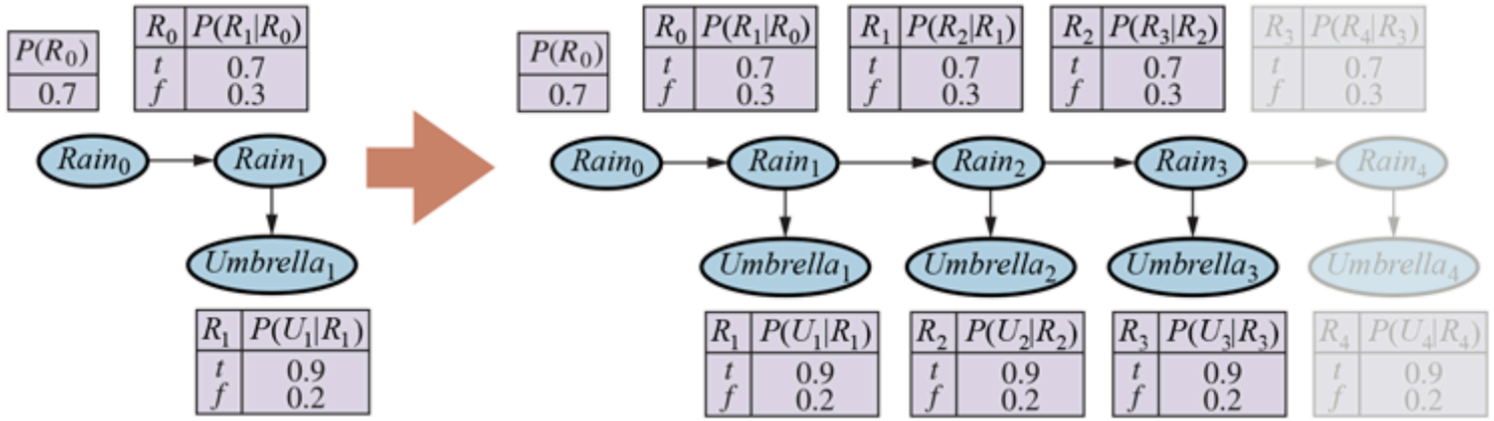


- But very inefficient 😰
  - Space requires to filter or smooth on $\mathbf{o}_{1:t}$ would require $O(t)$ space and thus grown without bound
  - Inference time per update will also increase as $O(t)$

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Exact inference in DBNs by Unrolling

- Given a sequence of observations, one can construct the full Bayesian network representation of a DBN by replicating slices until the network is large enough to accommodate the observations



- But very inefficient 😰
  - Space requires to filter or smooth on $\mathbf{o}_{1:t}$ would require $O(t)$ space and thus grown without bound
  - Inference time per update will also increase as $O(t)$
- What about recursive algorithms with “constant” time and space complexity per filtering update?
  - Unfortunately exponential: pace complexity $O(d^{n+k})$ and time complexity $O(nd^{n+k})$ with n state variables of domain size d and maximum number of parents k

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Approximate inference in DBNs

- Idea: randomized sampling algorithms, aka **Monte Carlo** algorithms
  - Work by generating random events based on the probabilities in the DBN and counting up the different answers found in those random events
  - Provide approximate answers whose accuracy depends on the number of samples generated

- Two families of algorithms used for general Bayes Networks
  - **Direct sampling**
    - generate each sample from scratch
    - e.g. Likelihood Weighting
  - **Markov chain sampling**
    - generate a sample by making a random change to the preceding sample
    - e.g. Markov Chain Monte Carlo (MCMC)

- These can be adapted for DBNs
  - Require several improvements to be practical

- E.g. Particle Filtering, an adaption of Likelihood Weighting

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Particle Filtering

- First we generate a population of N samples from the prior distribution $\mathbf{P}(\mathbf{S}_0)$
- Then the update cycle is repeated for each time step:
  - Each sample is propagated forward by sampling the next state value $\mathbf{s}_{t+1}$ given the current value $\mathbf{s}_t$ for the sample, given the transition model $\mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{S}_t)$
  - Each sample is weighted by the likelihood it assigns to the new observation, $\mathbf{P}(\mathbf{o}_{t+1} \mid \mathbf{s}_{t+1})$
  - The population is *resampled* to generate a new population of N samples
    - Each new sample is selected from the current population; the probability that a particular sample is selected is proportional to its weight. The new samples are unweighted.

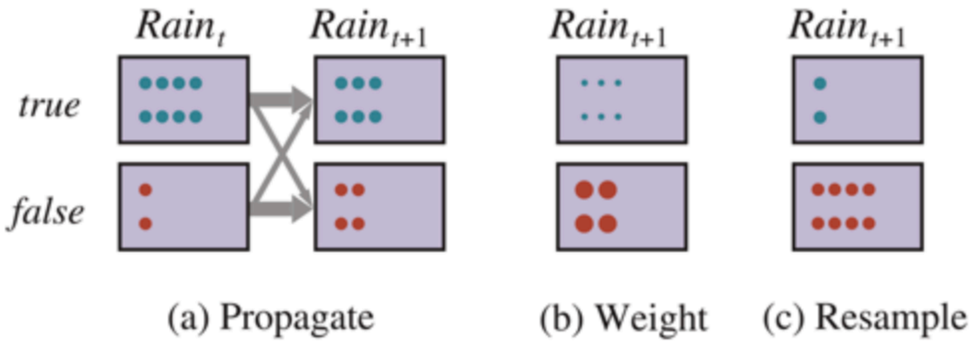
function PARTICLE-FILTERING($\mathbf{e}, N, dbn$) **returns** a set of samples for the next time step
  inputs: $\mathbf{e}$, the new incoming evidence
    N, the number of samples to be maintained
    dbn, a DBN defined by $\mathbf{P}(\mathbf{X}_0)$, $\mathbf{P}(\mathbf{X}_1 \mid \mathbf{X}_0)$, and $\mathbf{P}(\mathbf{E}_1 \mid \mathbf{X}_1)$
  persistent: S, a vector of samples of size N, initially generated from $\mathbf{P}(\mathbf{X}_0)$
  local variables: W, a vector of weights of size N

  for $i = 1$ to N **do**
    $S[i] \leftarrow$ sample from $\mathbf{P}(\mathbf{X}_1 \mid \mathbf{X}_0 = S[i])$ // *step 1*
    $W[i] \leftarrow \mathbf{P}(\mathbf{e} \mid \mathbf{X}_1 = S[i])$ // *step 2*
  $S \leftarrow$ WEIGHTED-SAMPLE-WITH-REPLACEMENT(N, S, W) // *step 3*
  return S

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Particle Filtering

- First we generate a population of N samples from the prior distribution $\mathbf{P}(\mathbf{S}_0)$
- Then the update cycle is repeated for each time step:
  - Each sample is propagated forward by sampling the next state value $\mathbf{s}_{t+1}$ given the current value $\mathbf{s}_t$ for the sample, given the transition model $\mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{S}_t)$
  - Each sample is weighted by the likelihood it assigns to the new observation, $\mathbf{P}(\mathbf{o}_{t+1} \mid \mathbf{s}_{t+1})$
  - The population is *resampled* to generate a new population of N samples
    - Each new sample is selected from the current population; the probability that a particular sample is selected is proportional to its weight. The new samples are unweighted.



The particle filtering update cycle for the umbrella DBN with $N = 10$, showing the sample populations of each state. (a) At time t, 8 samples indicate *rain* and 2 indicate $\neg$*rain*. Each is propagated forward by sampling the next state through the transition model. At time $t + 1$, 6 samples indicate *rain* and 4 indicate $\neg$*rain*. (b) $\neg$*umbrella* is observed at $t + 1$. Each sample is weighted by its likelihood for the observation, as indicated by the size of the circles. (c) A new set of 10 samples is generated by weighted random selection from the current set, resulting in 2 samples that indicate *rain* and 8 that indicate $\neg$*rain*.

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

Particle Filtering

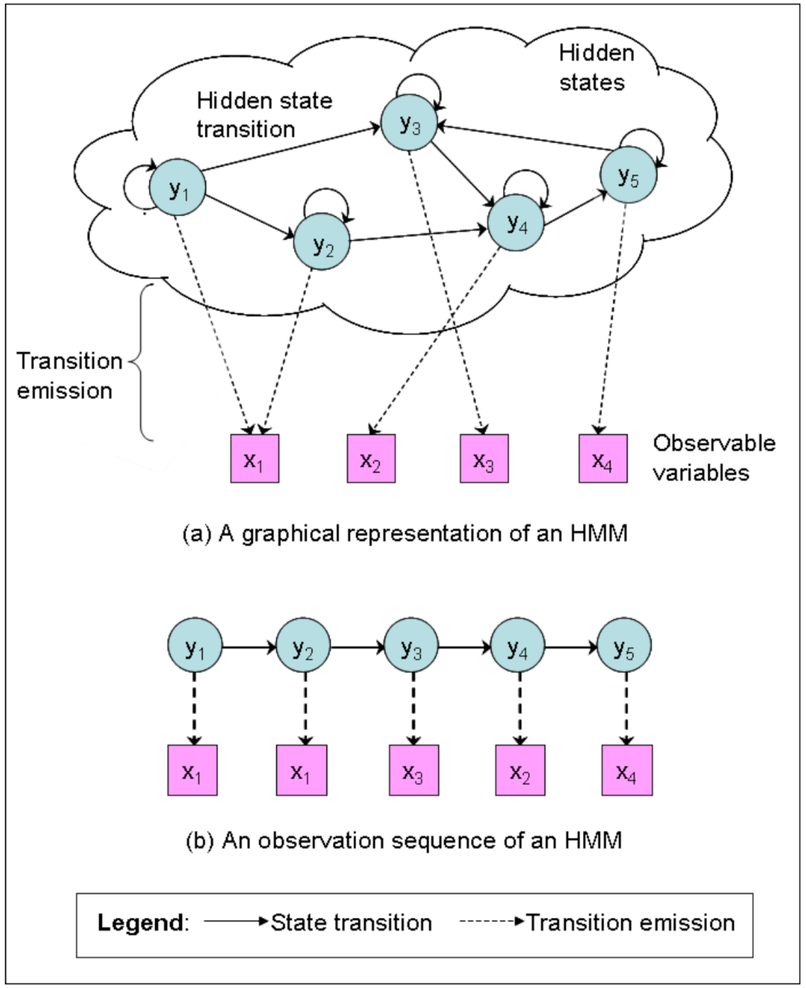
- First we generate a population of N samples from the prior distribution $\mathbf{P}(\mathbf{S}_0)$
- Then the update cycle is repeated for each time step:
  - Each sample is propagated forward by sampling the next state value $\mathbf{s}_{t+1}$ given the current value $\mathbf{s}_t$ for the sample, given the transition model $\mathbf{P}(\mathbf{S}_{t+1} \mid \mathbf{S}_t)$
  - Each sample is weighted by the likelihood it assigns to the new observation, $\mathbf{P}(\mathbf{o}_{t+1} \mid \mathbf{s}_{t+1})$
  - The population is *resampled* to generate a new population of N samples
    - Each new sample is selected from the current population; the probability that a particular sample is selected is proportional to its weight. The new samples are unweighted.
- Important property of Particle Filtering: it is *consistent*
  - i.e. gives the correct probabilities as N tends to infinity

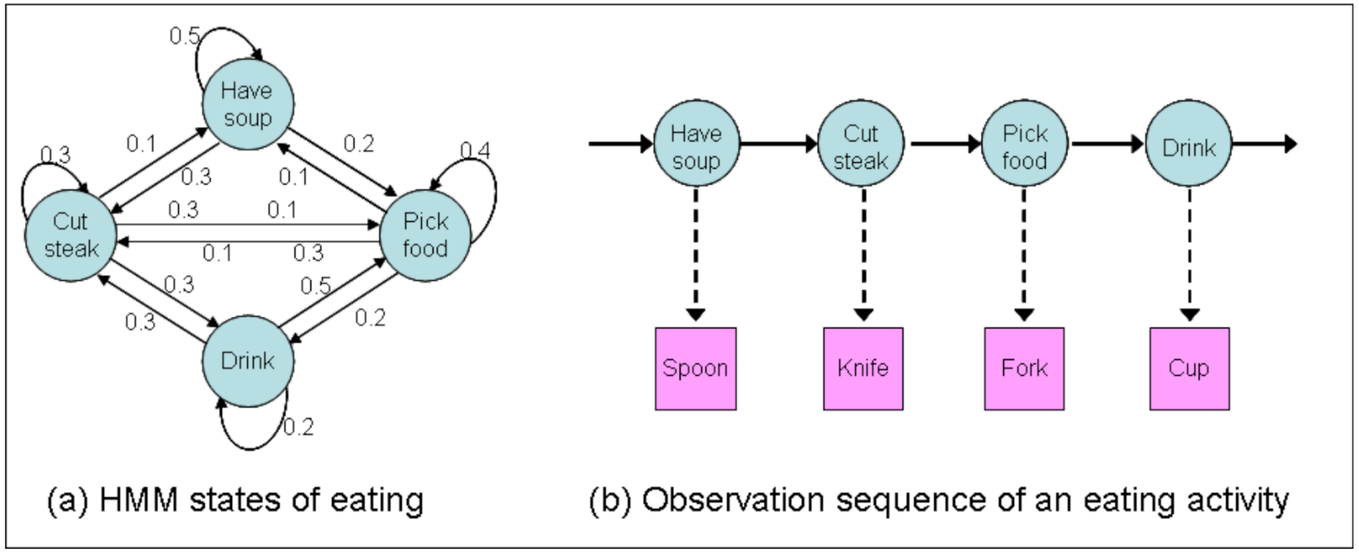
    $$\lim_{n \to \infty} N(\mathbf{s}_t \mid \mathbf{o}_{1:t})/N = P(\mathbf{s}_t \mid \mathbf{o}_{1:t}) = \mathbf{f}_t$$

    where $N(\mathbf{s}_t \mid \mathbf{o}_{1:t})$ is the # of samples occupying state $\mathbf{s}_t$ after observations $\mathbf{o}_{1:t}$ have been processed
- Also, efficient for many practical applications and is therefore widely used
  - Maintains a good approximation to the true posterior using a constant number of samples
  - Handles combinations of discrete and continuous variables
  - Handles nonlinear and non-Gaussian models for continuous variables

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

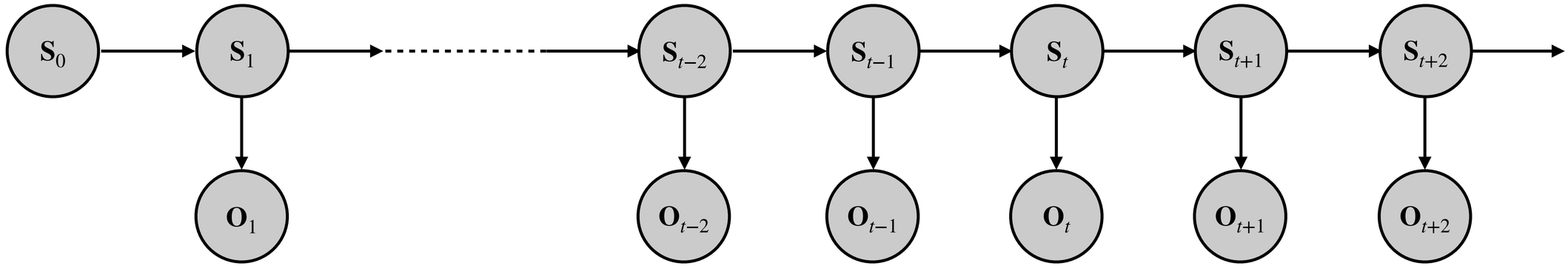
Revisit HMM





Kim, Eunju, Sumi Helal, and Diane Cook. "Human activity recognition and pattern discovery."

Learning HMM Model



- Supervised: Learn transition and sensor models from labeled data
  - Access to both observations $\mathbf{o}_{1:t}$ and ground truth $\mathbf{s}_{0:t}$
  - Ground truth via manual labeling, addition sensor etc. *Costly!*
- Unsupervised: Learning from observations alone
  - Maximum likelihood estimate of the parameters of the HMM given the set of output sequences (intractable 😨)
  - Efficient practical approach: learning as a byproduct of inference - similar to Expectation Maximization algorithm
    - Inference provides an estimate of what transitions actually occurred and of what states generated the observations
    - These estimates can be used to update the models
    - The updated model provides new estimates, and the process iterates to convergence
- Training is also possible with prior expert-provided knowledge of some aspects of the model
- individually trained HMM can be combined to construct a larger HMM model
  - e.g., of a complex activities with clear sub-activities structure

Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach"

A Common Algorithm: Baum–Welch

- It gives local optimum: gradient descent

- References:
  - https://www.youtube.com/watch?v=JRsdt05pMoI - part of an excellent series on HMM
  - https://en.wikipedia.org/wiki/Baum–Welch_algorithm
  - https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-410-principles-of-autonomy-and-decision-making-fall-2010/lecture-notes/MIT16_410F10_lec21.pdf

Algorithm 1: The Baum-Welch algorithm

Initialization: Θ_0, $\{O_{1:T}\}$

Looping:

for $l = 1, \ldots, l_{\max}$ **do**

1. Forward-Backward calculations:

$$\alpha_1(i) = \pi_i b_i(O_1),\ \beta_T(i) = 1,$$

$$\alpha_t(i) = \Big[\sum_{j=1}^{K} \alpha_{t-1}(j) a_{ji}\Big] b_j(O_t),\ \beta_t(i) = \sum_{j=1}^{K} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$$

$$\text{for } 1 \le i \le K,\ 1 \le t \le T-1$$

2. E-step:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{K} \alpha_t(j)\beta_t(j)},\ \xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}\beta_t(j))}$$

$$\text{for } 1 \le i \le K,\ 1 \le j \le K,\ 1 \le t \le T-1$$

3. M-step:

$$\pi_i = \frac{\gamma_1(i)}{\sum_{j=1}^{K} \gamma_1(j)},\ a_{ij} = \frac{\sum_{t=1}^{T} \varepsilon_t(i,j)}{\sum_{k=1}^{K}\sum_{t=1}^{T} \varepsilon_t(i,k)},\ w_{kd} = \frac{\sum_{t=1}^{T} \gamma_t(k,d)}{\sum_{t=1}^{T}\sum_{r=1}^{D} \gamma_t(k,r)}$$

$$\text{for } 1 \le i \le K,\ 1 \le j \le K,\ 1 \le k \le K, 1 \le d \le D$$

end

Result: $\{\Theta_l\}_{l=0}^{l_{\max}}$

https://tex.stackexchange.com/questions/480005/baum-welch-algorithm-description

Limitations of HMM

- Difficulty in representing multiple interacting activities (concurrent or interwoven)
- Incapable of capturing long-range or transitive dependencies of the observations due to its very strict independence assumptions (on the observations)
- Without significant training, an HMM may not be able to recognize all of the possible observation sequences that can be consistent with a particular activity.
- In practice:
  - Many activities are concurrent or interwoven
  - Many activities may have non-deterministic natures
    - some steps of the activities may be done in any order

Kim, Eunju, Sumi Helal, and Diane Cook. "Human activity recognition and pattern discovery."

A More Flexible Alternative: Conditional Random Field (CRF)

- Addresses practical requirements that HMM cannot
- Two types of models
  - *Generative*: finds a joint probability distribution $p(x, y)$ of a hidden variable y and an observed variables x
  - *Discriminative*: finds only the conditional probability $p(y \mid x)$
- Both can be used to find a hidden state transition from observation sequences
- What type is HMM?

Kim, Eunju, Sumi Helal, and Diane Cook. "Human activity recognition and pattern discovery."

A More Flexible Alternative: Conditional Random Field (CRF)

- Addresses practical requirements that HMM cannot
- Two types of models
  - *Generative*: finds a joint probability distribution $p(x, y)$ of a hidden variable y and an observed variables x
  - *Discriminative*: finds only the conditional probability $p(y \mid x)$
- Both can be used to find a hidden state transition from observation sequences
- What type is HMM? *Generative!*
- CRF: a conditional distribution with an associated graphical structure
  - Attempts to find only the conditional probability $p(y \mid x)$, i.e. *discriminative* approach
  - Allows for arbitrary, non-independent relationships among the observation sequences
  - Relaxes the independence assumptions
    - the hidden state probabilities may depend on the past and even future observations
  - Modeled as an undirected acyclic graph with two types of nodes: observed & unobserved

Kim, Eunju, Sumi Helal, and Diane Cook. "Human activity recognition and pattern discovery."

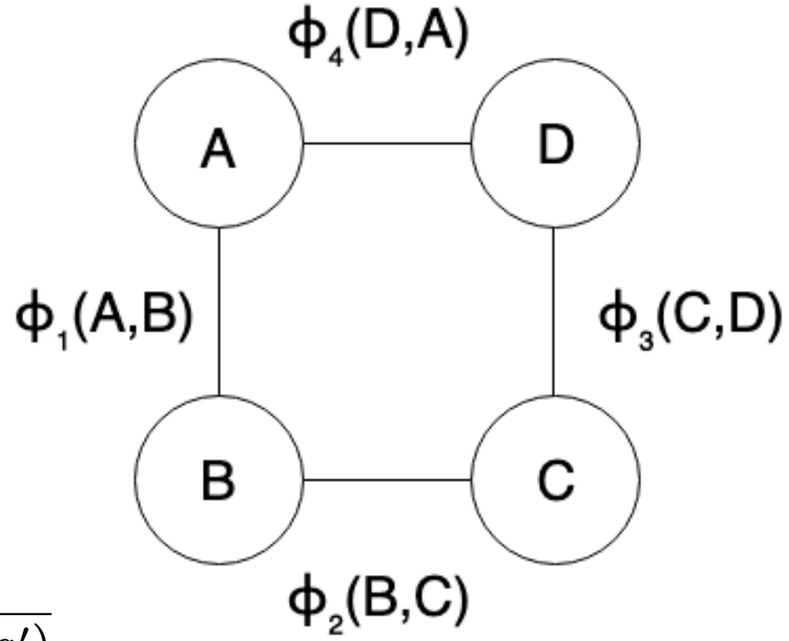
Digression: Markov Random Fields (MRFs)

https://towardsdatascience.com/conditional-random-fields-explained-e5b8256da776

Digression: Markov Random Fields (MRFs)

- Represented by a graph $G = (V, E)$
  - nodes represent random variables
  - edges collectively represent the dependencies between them
- The graph can be factorized into J different cliques or factors
  - each governed by a factor function Φ_j with its scope being a subset of random variables D_j where $\Phi_j(d_j) > 0 \forall d_j \in \mathrm{D}_j$
- The unnormalized joint probability of the variables is the product of all the factor function, so that

$$Pr(A = a, B = b, C = c, D = d) = \frac{\phi_1(a, b)\phi_2(b, c)\phi_3(c, d)\phi_4(d, a)}{\sum_{a'}\sum_{b'}\sum_{c'}\sum_{d'}\phi_1(a', b')\phi_2(b', c')\phi_3(c', d')\phi_4(d', a')}$$



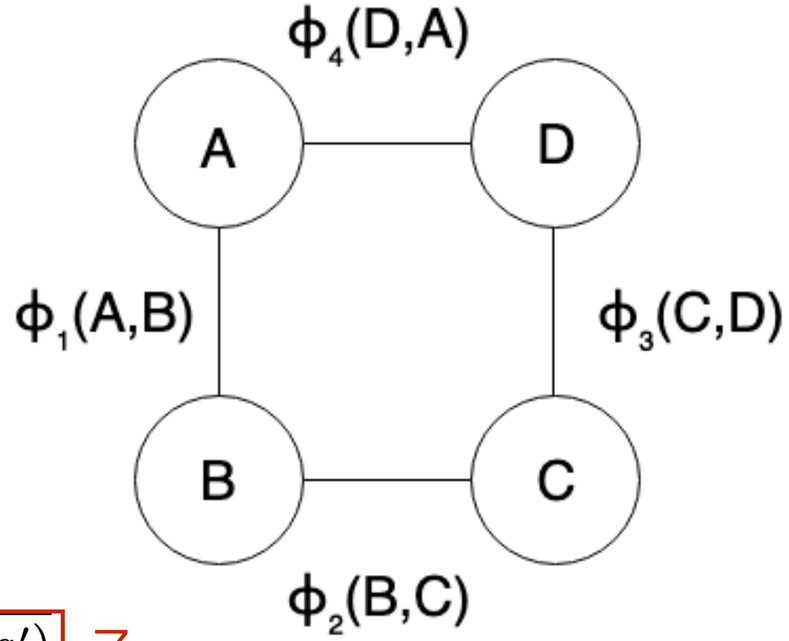
https://towardsdatascience.com/conditional-random-fields-explained-e5b8256da776

Digression: Markov Random Fields (MRFs)

- Represented by a graph $G = (V, E)$
  - nodes represent random variables
  - edges collectively represent the dependencies between them
- The graph can be factorized into J different cliques or factors
  - each governed by a factor function Φ_j with its scope being a subset of random variables D_j where $\Phi_j(d_j) > 0 \forall d_j \in \mathrm{D}_j$
- The unnormalized joint probability of the variables is the product of all the factor function, so that

$$Pr(A = a, B = b, C = c, D = d) = \frac{\phi_1(a,b)\phi_2(b,c)\phi_3(c,d)\phi_4(d,a)}{\sum_{a'}\sum_{b'}\sum_{c'}\sum_{d'}\phi_1(a',b')\phi_2(b',c')\phi_3(c',d')\phi_4(d',a')}$$

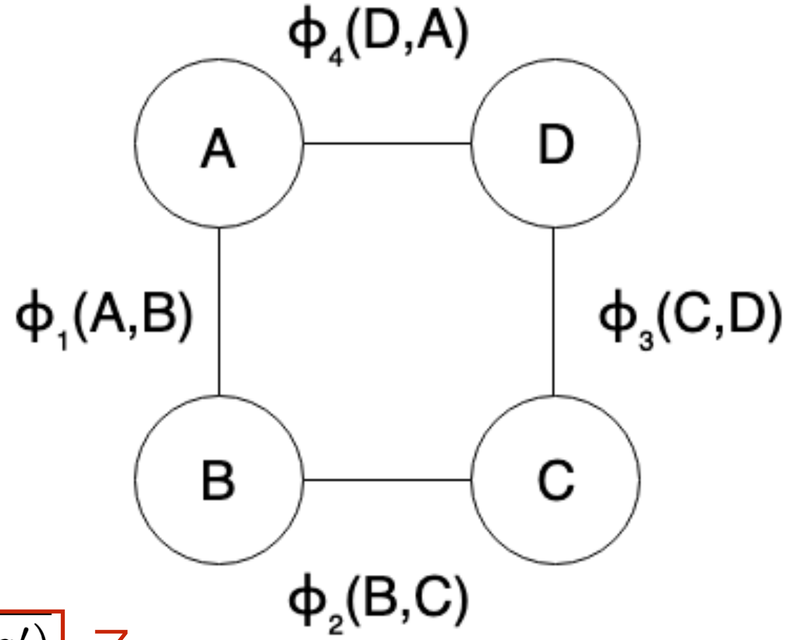
Z



https://towardsdatascience.com/conditional-random-fields-explained-e5b8256da776

Digression: Markov Random Fields (MRFs)

- Represented by a graph $G = (V, E)$
  - nodes represent random variables
  - edges collectively represent the dependencies between them
- The graph can be factorized into J different cliques or factors
  - each governed by a factor function Φ_j with its scope being a subset of random variables D_j where $\Phi_j(d_j) > 0 \forall d_j \in D_j$
- The unnormalized joint probability of the variables is the product of all the factor function, so that



$$Pr(A = a, B = b, C = c, D = d) = \frac{\phi_1(a,b)\phi_2(b,c)\phi_3(c,d)\phi_4(d,a)}{\sum_{a'}\sum_{b'}\sum_{c'}\sum_{d'}\phi_1(a',b')\phi_2(b',c')\phi_3(c',d')\phi_4(d',a')}$$ Z

- Gibb's Notation: represent the joint as a Gibbs distribution by operating on factor functions in log space by using $\beta(d_j) = \log(\phi(d_j))$
  - X is the set of all the random variables in the graph

$$\text{Energy, } E(x) = -\sum_{j=1}^{J} \beta_j(d_j), d_j \subseteq X$$

$$\text{Gibbs, } P(x) = \frac{e^{-E(x)}}{Z}, \text{ where } Z = \sum_{x' \epsilon X} e^{-E(x')}$$

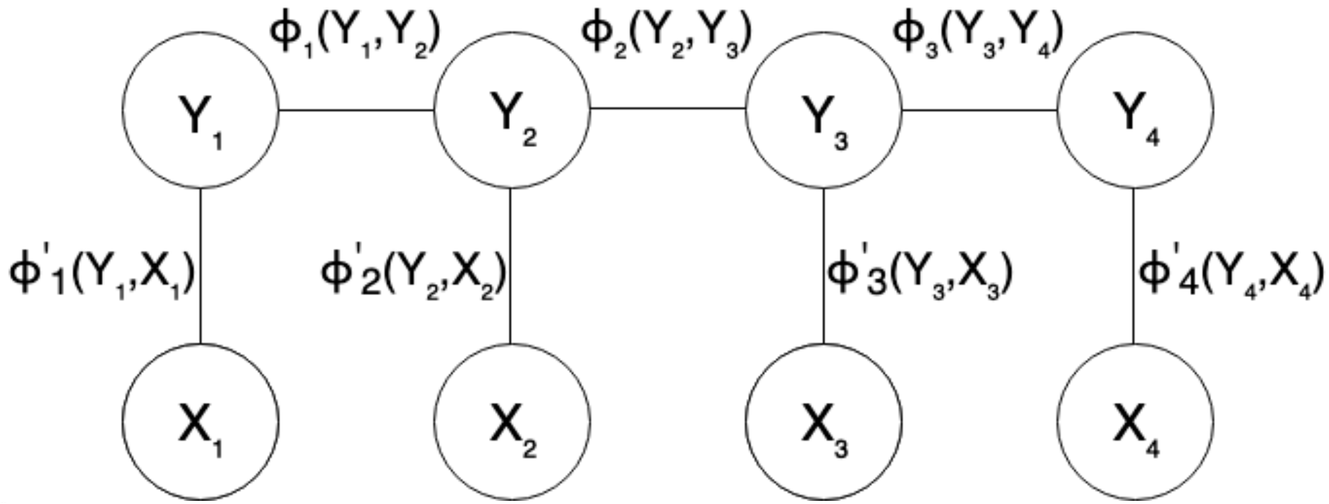
https://towardsdatascience.com/conditional-random-fields-explained-e5b8256da776

Conditional Random Field Model

- Assume a MRF divides into two sets of random variables Y (unobserved) and X (observed) respectively
- CRF ia special case of this MRF wherein the graph satisfies the property:

  "When we condition the graph on X globally i.e. when the values of random variables in X is fixed or given, all the random variables in set Y follow the Markov property $p(Y_u|X,Y_v, u\neq v) = p(Y_u|X,Y_x, Y_u \sim Y_x)$, where $Y_u \sim Y_x$ signifies that Y_u and Y_x are neighbors in the graph."

  - A variable's neighboring nodes or variables are also called the Markov Blanket of that variable
- One such graph that satisfies the above property is the chain-structured graph shared below:

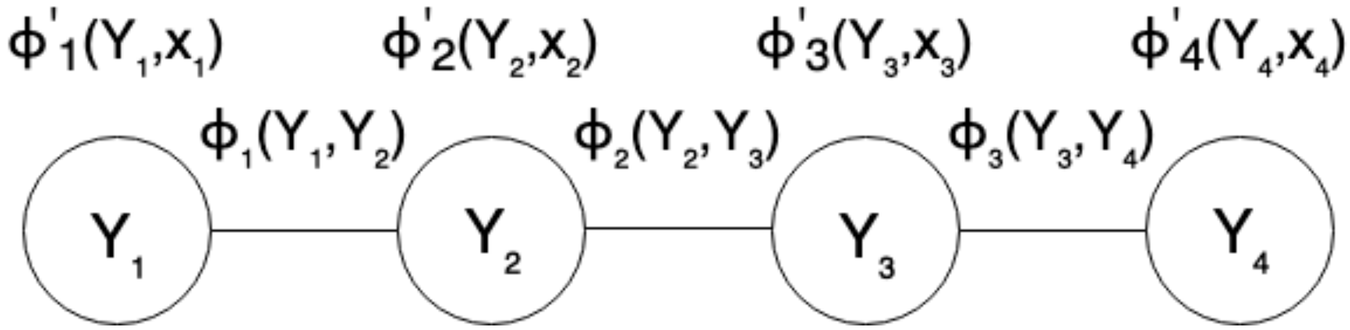


https://towardsdatascience.com/conditional-random-fields-explained-e5b8256da776

Making Inferences with CRF

- CRF is a discriminative model i.e. it models the conditional probability P(Y|X)
  - i.e. X is always given or observed

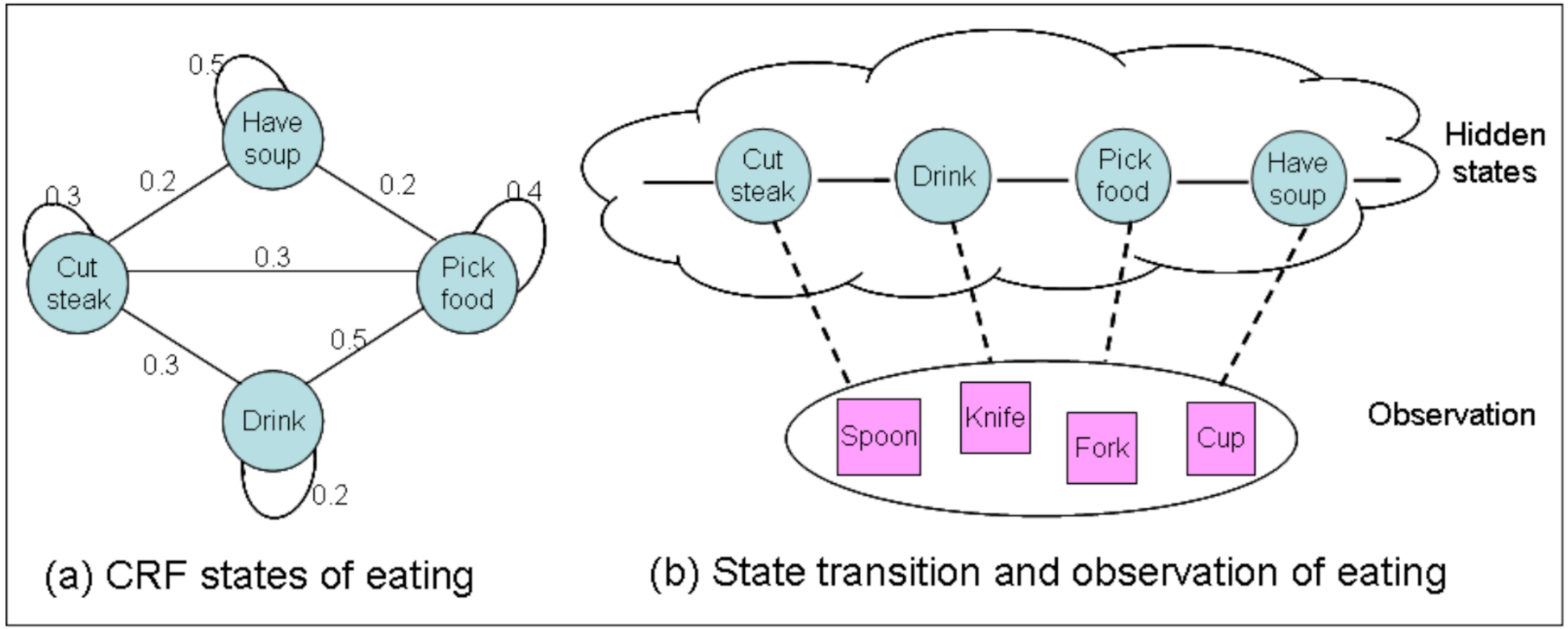
- Therefore the graph ultimately reduces to a simple chain



- As we condition upon X and we are trying to find the corresponding Y_i for every X_i , X and Y are also called the evidence (sensor observations) and label variables respectively
- Training problem: maximizing the log likelihood wrt all model parameters
  - See “Conditional Random Fields Explained” by Aditya Prasad @ the URL in left bottom footer
  - Many off-the-shelf packages

https://towardsdatascience.com/conditional-random-fields-explained-e5b8256da776

CRF Version of the Eating Activity Example



Kim, Eunju, Sumi Helal, and Diane Cook. "Human activity recognition and pattern discovery."

Practical Resources

- **scikit-learn** (https://scikit-learn.org) provides many general classification and regression algorithms
- **Pomegranate** package for Python
  - Many probabilistic models: HMM, Naive Bayes, and various others…
  - Schreiber, Jacob. "Pomegranate: fast and flexible probabilistic modeling in python." The Journal of Machine Learning Research 18, no. 1 (2017): 5992-5997.
    - https://jmlr.org/papers/volume18/17-636/17-636.pdf
  - https://github.com/jmschrei/pomegranate and https://pomegranate.readthedocs.io
- **pgmpy** python library for working with Probabilistic Graphical Models.
  - Many probabilistic graphs models: Dynamic Bayesian Networks, Naive Bayes etc.
  - https://github.com/pgmpy/pgmpy and http://pgmpy.org
- **Fun with Hidden Markov Models** in PyTorch (by Loren Lugosch)
  - Based on: Rabiner, Lawrence R. "A tutorial on hidden Markov models and selected applications in speech recognition." Proceedings of the IEEE 77, no. 2 (1989): 257-286.
    - https://www.cs.cmu.edu/~cga/behavior/rabiner1.pdf
  - https://colab.research.google.com/drive/1IUe9lfoIiQsL49atSOgxnCmMR_zJazKI
- **python-crfsuite** python binding for CRFSuite
  - An implementation of Conditional Random Fields (CRFs) for labeling sequential data
  - https://github.com/chokkan/crfsuite

Practical Resources (contd.)

- **FilterPy** package for Python
  - Python library that implements a number of Bayesian filters, most notably Kalman filters
  - https://github.com/rlabbe/filterpy
  - Excellent related interactive book: **Kalman and Bayesian Filter in Python**
    - https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python/
- **pykalman** library for Python
  - Kalman Filter, Kalman Smoother, and EM
  - https://pykalman.github.io and https://github.com/pykalman/pykalman
- **TinyEKF** Lightweight C/C++ Extended Kalman Filter with Python
  - https://github.com/simondlevy/TinyEKF
- **tsBNgen** for generating synthetic time series data in Python (from Prof. Pottie's group)
  - Python package to generate time series data based on an arbitrary Bayesian Network structures
  - Tadayon, Manie, and Greg Pottie. "tsBNgen: A Python Library to Generate Time Series Data from an Arbitrary Dynamic Bayesian Network Structure." arXiv preprint arXiv:2009.04595 (2020).
    - https://arxiv.org/pdf/2009.04595.pdf
  - https://github.com/manitadayon/tsBNgen#Features