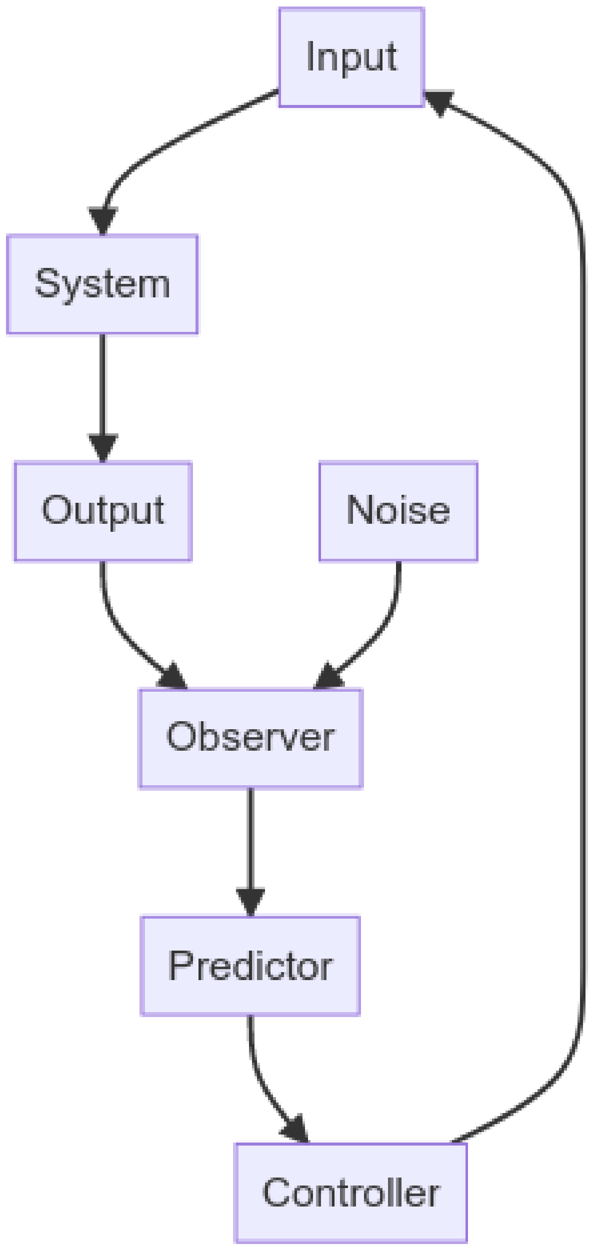
Lecture 1



- Observer = what happened
- Predictor = what will happen
- Control = how to make it happen

Estimate a constant

- There is noise in the system
- $\frac{1}{N}\sum_{k=1}^{N} X_k$
- How big should we make N?

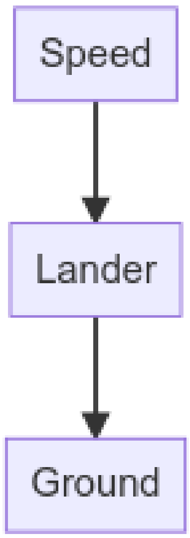
Model

- $X_{m+1} = X_m$
- X = state
- $Y_m = X_m + noise$
- Can assume that the noise has a Gaussian distribution
- We care about the variance

Systems : Models

- $X_{m+1} = A(X_m 1, u_m)$ input
- $y_m = h(X_m)$ output
- This is said to be stochastic in nature

Planet Lander Example:



- $x(t)$ = position = 0
- $\dot{x}(t)$ = speed = 0
- cost |u|
- You estimate to stop from smashing into the ground
- We can then make the problem optimal by minimizing cost

$$\begin{matrix} 0 & 0 & 0 & G \\ 0 & [] & 0 & 0 \\ R & 0 & 0 & 0 \end{matrix}$$

- The observer is a Kalman filter
- Optimal control
- Based on the bellman equation

Probability

- u = universe of all possible outcomes
- A is the set we want
- $P(A) = \frac{A}{u}$
- O <= P(A) <= 1
- B is some other set of outcomes and we want the intersection of B and A
- $P(B|A) = \frac{P(A|B)P(B)}{P(A)}$
- Bayesian reasoning

Lecture 3

Probability Partitioning

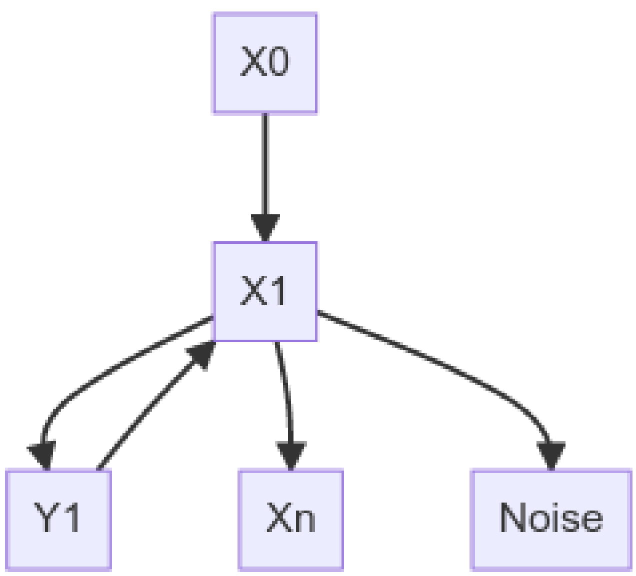
$$\begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}$$

- $A_k \cap A = \phi$
- $VA_k = u$
- $P(c) = \sum P(c|A_k) * A_k$

Hero Example

- 1/100 people is a hero
- 90% accurate, 10% false positive
- 80% accurate, 20% false negative
- P(H | X) = 0.9
- P(H | !X) = 0.2
- P(X) = 0.01
- $P(X|H) = \frac{P(H|X)*P(X)}{P(H)}$
- $P(X|H) = \frac{0.9*0.01}{0.01*0.90.99*0.2} = 0.05 = 5$

Bayesian Reasoning



- $P(X|Y) = \int XP(X|Y)dx$

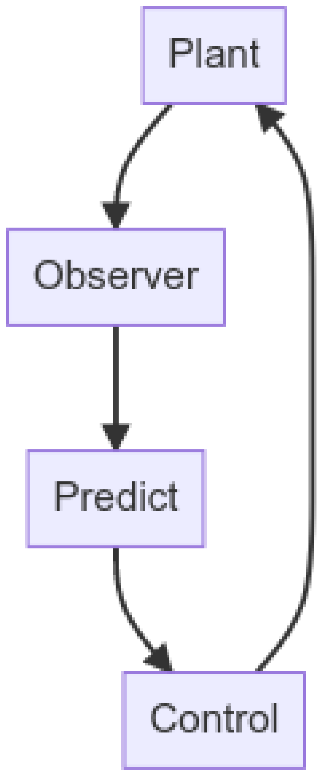
Hallway Robot

- 0 1 2 3 4 5 6 7 8 9
- 0 1 1 0 0 0 1 0 0 0
- $P_1(x) = 0 $\frac{1}{3}$ $\frac{1}{3}$ 0 0 0 $\frac{1}{3}$ 0 0 0
- $P_0(x) = $\frac{1}{7}$ 0 0 $\frac{1}{7}$ $\frac{1}{7}$ $\frac{1}{7}$ 0 $\frac{1}{7}$ $\frac{1}{7}$ $\frac{1}{7}$
- $X_0 = 0.1 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1$
- $P(X|1) = \frac{P(1|X)P(x)}{P(1)}$

Move the robot

- $X_1(x) = 0 0 $\frac{1}{3}$ $\frac{1}{3}$ 0 0 0 $\frac{1}{3}$ 0 0
- $P(X|1) = \frac{P(G|1)P(X)}{P(1)}$

Plant Example



- x' = A(x, u)
- y = u(x)
- X(T) = 0
- $X(\text{inf}) = 0$
- $J = \int_0^{\text{inf}} cost(x, u) dt$
- Arg min J u
- Minimize control input cost that approaches final state

Race Track Example

- Find the series of values to mimize cost
- $Min \sum ||\widetilde{X_k} - X_k||$
- $Min \sum ||Robot - Markers||$
- $\widetilde{X_{k+1}} = A\widetilde{X_k} + Bu_k$

Lecture 2

Control

Models:
* $x' = f(x, u)$
* $y = h(x)$

Discrete time:
* $y'' = -y + u'$ lander
* pos =
* $x_1(m+1) = x_1(m) + \Delta t x_2(m)$
* $x_2(m+1) = x_2(m) + \Delta t(-g + h)$
* $h(x) = x_1$

Linear systems

- $x_{m+1} = Ax_m + Bu_m$
- $y_{m+1} = Cx_m$
- $x_{m+1} = \begin{matrix} 1 & \Delta t \\ 1 & 0 \end{matrix} * \begin{matrix} 0 \\ \Delta t \end{matrix}$

Learning (Supervised)

- $data(x^k, y^k)$
- x^k are uncorrelated
- $F_\theta(x) = y$

- θ = parameter to learn

- Arg min $\theta \sum_{k=1}^{N} ||F_\theta(x^k) - y^k|| =$ error

- Convex shape, where you iterate to find the global minimum
- $x_u = x + \alpha A'(x)$

Tracking Example

- $data(x^k, y^k)$

- Find a control input for a car to follow a path

- $x'' = u'' = F = ma$
-
$$\begin{array}{ccccccc} x_1 & & 1 & 1 & & 0 \\ x_2 & & g & 0 & & 1 \\ y_1 & * & - & - & * & - \\ y_2 & & - & - & & - \end{array}$$

- $x_{m+1} = Ax_m + Bu_m$

- $y_m = Cx_m$

- Arg min $\theta \sum_{k=0}^{N} ||yk - y^k|| =$ error

- u <- u - error

Control Solution / Control Matrix

- $x_{m+1} = Ax_m + Bu_m$

- $y_m = Cx_m$

- x_0

- $x_1 = Ax_0 + Bu_0$

- $x_2 = A^2x_0 + ABu_0 + Bu_1$

- $x_N = A^N x_0 + . . . + B u_{N-1}

- $$\begin{matrix} Ax_0 & CB & u_0 & y^1 \\ Ax_1 & CAB*CB & u_1 & y^2 \\ Ax_2 & +CA^2BCABCB* & u_2 & =y^3 \\ ... & ... & ... & ... \\ A^Nx_C & CA^{N+1}B... & u_{N-1} & y^k \end{matrix}$$

Lecture 4

Linear Algebra

- Vector space
- $V over \Re(C|)$
- $x, y \Xi V$
- $\lambda x' \Xi V$
- $\lambda_1 x + lambda_2 y \Xi V$
- $\lambda_1 lambda_2 \Xi \Re$
- $X \Xi V \exists y x + y = 0$

Apples and Oranges Example

- $A, B \Xi V$
- A = apples, B = oranges
- $\lambda_1 A = \lambda_2 B$
- $\lambda_1 = \lambda_2 = 0$
- This allows you to solve by super position because of the linear independence
- $X \Xi V$
- $\lambda_1(A) + \lambda_2(B) = x = 3A + B$
- $\lambda_1(A + B) + \lambda_2(B) = x = 3A - 2B$

- $\lambda_1(A+B)+\lambda_2(2A+2B) = \text{x} = \text{Not possible}$

- $3A+2B = 3//2$

- $\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} * \begin{matrix} \lambda_1 \\ lambda_2 \end{matrix} = x$

- $\lambda_1 \begin{matrix} 1 \\ 1 \end{matrix} + \lambda_2 \begin{matrix} 0 \\ 1 \end{matrix} = x$

- $\begin{matrix} 1 & 0 \\ 1 & 1 \end{matrix} * \begin{matrix} \lambda_1 \\ \lambda_2 \end{matrix} = (\begin{matrix} 3 \\ 1 \end{matrix})$

Finite Dimension Solution

- $\Re^M = Finite$

- e^{imw_o}

- $X(m) = \sum_{-\inf}^{\inf} \delta(m-k) * x(k)$

- Vector space
  - metric "measure"

  - inner product

  - topology

  - completeness

Metric

- Distance

- $d(x,y) >= 0$

- $d(x,x) = 0$

- $d(x,x) <= d(y,z) + d(z,y)$

- Norms

  - $||X||_2 = sqrt(\sum_{k=1}^{N} X_k^2)$

  - $d(x,y) = ||x-y||_2$

- $||x||_p = (\sum X_k^p)^{\frac{1}{P}}$
- $||x||_2^2 = \sum x_k^2$

Scalar Product

- Inner product
- $< x, y >= \sum x_i y_i$
- $< x, y >= |x||y| cos\alpha$

Linear Functions

- $V = f(x)$
- $V- > \Re$
- $Given f(x) V- > \Re$
- $Linear f(\lambda_1 x + lambda_2 y)$
- $= \lambda_1 f(x) + \lambda_2 f(y)$
- $Exists c \Xi \Re^N$
- $f(x, y) = C^T x$
- $f(x, y) = 2x + 3y$
- $C = \begin{matrix} 2 \\ 3 \end{matrix}$
- Dual allows you to compare contorllers
  - $\Re^N - > \Re$

Linear Transforms

- $X_{m+1} = AX_m = Bu_m$
- $X_m - > 0$

- $\frac{||AX||}{||X|||}$
- $||X^* - x_m|| = ||AX^* - AX||$
- $X^* = fixedpoint$

Lecture 5

Euclidean Space

- $\Re^m$
- m equations
- m < n, x + y = 1 = Space / undeterminant
- m = n there is one solution
- m > n there are more parameters than solutions

Model Fitting

- Model -> Functional
- $F_\theta(x), (x^k, y^k)$
- $\theta, x = parameters$
- $\frac{ArgMin}{\theta} \sum_{k=1}^{N} ||F_\theta(x^k) - y^k||^2$
- $F_\theta(x) = \sum_{m=0}^{N} \theta_m \phi_m(x)$
- $\phi_m(x) = x^m$
- $\phi_m(x) = e^{-iwm}$
- Basis
  - $\lambda_1 \phi_m(x) = \lambda_2 \phi_k(x)$
  - $\lambda_1 = lambda_2 = 0$
  - $m = k$
  - $\phi_0 = \begin{matrix} 1 \\ 0 \end{matrix}$

  - $\phi_1 = \begin{matrix} 0 \\ 1 \end{matrix}$
  - $\begin{matrix} 2 \\ 3 \end{matrix} = 2\phi_0 + 3\phi_1$
- $F_\theta(x^k) = y^k$
- $\theta_0\phi_0(x^1) + \theta_1\phi_1(x^1) + \theta_2\phi_2(x^1) = y^1$
-

$$\begin{matrix} \phi_0(x^1) & phi_1(x^1) & ... & \phi_m(x^1) & \theta_0 & y^1 \\ \phi_0(x^2) & phi_1(x^2) & ... & \phi_m(x^2) * & \theta_1 & * y^2 \\ \phi_0(x^N) & phi_1(x^N) & ... & \phi_m(x^N) & theta_m & y^k \end{matrix}$$

- $||A\theta 0y||2$
- k = horizontal, n = vertical
- N » K, significantly larger

Linear Algebra

- $||x||^2 = X^T X$
- Norm Scalar Product
- $||x||^2 = \sum x_i^2$
- $\sum x_i^2 = variance$
- $\frac{ArgMin}{\theta}||A\theta - y||^2$
- $(A\theta - y)^t(A\theta - y)$
- $\theta^T A^T A\theta - \theta^T A^T y - y^T A\theta + y^T y$
  - $(AB)^T = B^T A^T$
  - $X^T Y = Y^T X$
  - $Y^t A\theta = A^T \theta^T y$
  - $\theta 6T A^T A\theta - 2\theta^T A^T y + y^T y$

- $\frac{d}{d\theta} 2A^T A\theta - 2A^T y = 0$
- $A^T A\theta = A^T y =$ normal equations
- $A\theta = y$

Example

- $F(x) = ax + b$
- (x^k, y^k)
  - $ax^1 + b = y^1$
  - $ax^2 + b = y^2$

$$\begin{matrix} 1 & x^1 \\ 1 & x^2 \\ 1 & x^N \end{matrix} * \begin{matrix} b \\ a \end{matrix} * \begin{matrix} y^1 \\ y^2 \\ y^N \end{matrix}$$

- $A^T A = \begin{matrix} N & \sum x^k \\ \sum x^k & \sum {x^k}^2 \end{matrix}$
- $(\begin{matrix} a & b \\ c & d \end{matrix})^{-1} = \frac{1}{det(A)} * (\begin{matrix} d & -b \\ -c & a \end{matrix})$
- $N \sum (x^k)^2 - (\sum x_k)^2$
- $a = \frac{N(\sum x^k y^k) - (\sum x^k)(\sum y^k)}{N \sum (x^k)^2 - \sum (x_k)^2}$

Example 2

- $f(x) = ax + b$
- (1, 2) (1, 3) (2, 5)
- $\begin{matrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{matrix} * \begin{matrix} b \\ a \end{matrix} = \begin{matrix} 1 \\ 3 \\ 5 \end{matrix}$
- $\begin{matrix} 1 & 1 & 1 \\ 0 & 1 & 2 \end{matrix} * \begin{matrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{matrix} = A^T A$
- $\begin{matrix} 1 & 1 & 1 \\ 0 & 1 & 2 \end{matrix} * \begin{matrix} 1 \\ 3 \\ 5 \end{matrix} = A^T y$

Lecture 6

Model fitting

- $(x^k, y^k) Given$

- k = 1 ... N

- $\frac{Argmin}{\theta} \sum_{k=1} N||F_\theta(x^k) - y^k||^2$

- $F_\theta(x) = \sum_{k=0}^{M} \theta_k]phi_k(x)$

- Linear is a combination of basis elements

- $A_\theta = y$

- $$A = \begin{matrix} \theta_0(x^1)d_1(x^1) & ... & \theta_m(x^1) \\ ... & ... & ... \\ \theta_0(x^N) & ... & \theta_m(x^N) \end{matrix}$$

- $A^T A\theta = A^T Y$

- Numerically use a QR factorization

Plant Model

- $ x_m -> Plant -> y_m$

- not online

- not a filter

- infinite memory filter

- Example Question:
  - (x^k, y^k) -> F(x) = ax$

  - $J_a(x) = \sum_{k=1} N(ax^k - y^k)^2$

  - $= a^2 {x^k}^2 - 2ax^k y^k + {y^k}^2$

- $\frac{d}{da} - 2ax^{k^2} - 2x^k y^k = 0$
- $ax^{k^2} = x^k y^k$
- $a = \frac{\sum x^k y^k}{\sum x^{k^2}}$

Trajectory Example

- $f(x) = ax^2 + bx + c$
- $\frac{d}{dx} = 2ax + b$
- $x = \frac{b}{2a}$ MAX
-

$$\begin{matrix} 1 & x^1 & x^{1^2} & c \\ 1 & x^2 & x^{2^2} * & b \\ 1 & x^3 & x^{3^2} & a \end{matrix}$$

AI

- $(x^k, y^k)
- With model
  - $\frac{ArgMin}{\theta} \sum ||F_\theta(x^k) - y^k||^2$
  - Iterate and update your θ
- Gradient search
  - $\theta_{m+1} = \theta_m + u\frac{dJ}{d\theta}$
  - $J = \sum \frac{1}{2}(F_\theta(x) - y)^2$
  - $F_\theta(x) = ax + b$ <- Line
  - $\frac{dJ}{da} = (ax + b - y)^2 * x$
  - $\frac{dJ}{db} = (ax + b - y)^2 * 1$

Lecture 7

Lecture 8

Random Variables

- $X = randomvariable$
- $X \Xi 1, 2, 3, 4, 5, 6$
- $P(X = z) = \frac{1}{6}$
- P(X) = probability density distribution
- $\sum_x P_x(X = x) = 1$
- $E(x) = \sum_x xP(X = x)$
- X pull samples
- Pull N samples
  - $u(x) = \frac{1}{N} \sum_{k=1}^{N} x_k$
  - Monte Carlo Simulation

Binning

- Pull Samples
- By binning you simulate the probability density function

Continous Case

- $X \Xi \Re$
- $P(0.55 <= X <= 0.6)$
- $\int P(x)dx = 1$
- $\int xP(x)dx = E(x)$
- $E(E(x) - x)^2$
- $= E(E(x)^2 - 2xE(x) + x^2)$

- $E(x)^2 - 2xE(x)E(x) + E(x)^2)$
- $E(x^2) - E(x)^2$
- Var X, E(x) = 0, No DC
- $Var(x) = E((E(x) - x)^2)$
- $= E(x^2) = 0$ - mean
- $PullX_k$
- $Var(x) = \frac{1}{N} \sum x_k^2$

Gauss or Normal Distribution

- $P(x) = \frac{1}{sqrt(2\pi\sigma^2)} e^{\frac{-1}{2} * \frac{(x-u)^2}{\sigma^2}}$
- $\lim \sigma -> 0$
- $P(x) = \delta(x)$
- $f(a) = \int f(x)\delta(x-a)dx$

Several Random Variables

- X, Y
- $Plot(X_k, Y_k)$
- $CO - VAR(x, y) = (E(x) - x|E(y) - y)$
- $CO - VAR(x, x) = VAR(x)$
- E(x) = 0, E(y) = 0
  - Can be accomplished with a DC filter
  - CO-VAR(x,y) = E(x * y)
  - Monte Carlo
    - $\frac{1}{N} \sum x_k y_k$
    - $x_k = cos(w_0 km)$

Lecture 9

Stochastic Process

- $E(x) = \int xP_X(x)dx$
- $E(x) linear$
- $E(\lambda_1 + lambda_2 y)$
- $= \lambda_1 E(x) + \lambda_2 E(y)$
- E(c) = c

Variance

- $VAR(x) = E((E(x) - x)^2)$
- $= E(x^2) - E(x)^2$
- $VAR(x) >= 0, VAR(c) = 0$ bi-linear quadractic
- $VAR(ax) = a^2 VAR(x)$
- $CO - VAR(x,y) = E((E(x) - x) * (E(y) - y))$
- $VAR(x + y) = VAR(X) + VAR(Y) + 2CO - VAR(x,y)$

2 Dimension Example

- $(\begin{matrix} X \\ Y \end{matrix}) * (X \quad Y)^T$
- $= (\begin{matrix} X^2 & YX \\ YX & Y^2 \end{matrix})$
- $= (\begin{matrix} VAR(X) & CO - VAR(X,Y) \\ CO - VAR(X,Y) & VAR(Y) \end{matrix})$ = covariance matrix
- $$\begin{matrix} x \\ y * x \quad y \quad z \\ z \end{matrix} = (\begin{matrix} \sigma_x^2 & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_z^2 \end{matrix})$$
- The above matrix is symmetric

Noise

- $Whitenoise\frac{1}{\epsilon(t)}$
- All frequencies have same probability
- Gaussian noise shaped like gauss
- Pink noise $\frac{12DB}{octave}$

Stochastic (Model Fitting)

- Model Y = aX + b

Equations

- $Y_k = ax_k + b + \epsilon_k$
- $\epsilon_k = N(0, \sigma_x^2)$
- $VAR = \sigma_x^2$
- $$CO - VAR(\epsilon_k, \epsilon_l) = \begin{matrix} 0 & k! = l \\ \sigma_x^2 & k = l \end{matrix}$$

Monte Carlo

- $\frac{1}{N}\sum_{k=1}^{N} y_k$
- $= \frac{1}{N}\sum_{k=1}^{N} aX_k + \epsilon_k$
- $= \frac{1}{N}\sum_{k=1}^{N} aX_k + \frac{1}{N}\sum_{k=1}^{N} \epsilon_k$

Ensemble Averaging

- By adding together all the noise, due to the noise being guassian the noise is equal to 0. So that gives you a meaningful measurement.

Maximum Likelihood (Estimator)

- $Y_k = aX_k + b$ -> Guass

- $P(Y_k - aX_k b)$ -> Want to maximize the probability
- $L = \prod_{k=1}^{N} P(y_k - ax_k - b)$
- $\frac{MaxL}{a,b} = \prod P(y_k - ax_k - b)$
- $Y_k - aX_k - b = \epsilon_k$
- $\epsilon_k = N(0, \sigma_x^2)$
- $\prod e^{\frac{\frac{-1}{2}(y_k - ax_k - b)^2}{\sigma_x^2}}$
- Maximize L
- First pull log
- Max(f) = Max(log(f))
- $e^{\sum_{k=1}^{N} \frac{\frac{-1}{2}(y_k - ax_k - b)^2}{\sigma_x^2}}$
- Pull Log
- $\frac{-1}{2sigma_x^2} \sum (y_k - ax_k - b)^2$ -> same as before

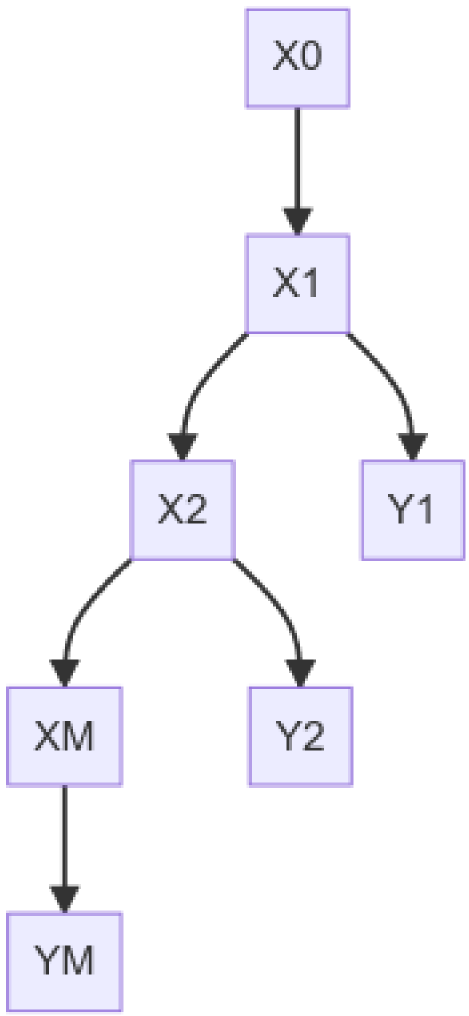
Stochastic System

- $x_{m+1} = f(x_m) = \epsilon_k$
- $y_m = h(x_m) + u_k$

Lecture 10

Stochastic Systems

- Markov Chains



- $x_{m+1} = f(x_m)$ -> deterministic
- $x_{m+1} = f(x_m) + \epsilon_m$
- $P(x_{m+1}|x_m)$
- $E(x_{m+1} = \int xP(x_{m+1})x_m dx)$
- $P(y_m|x_m)$

State Space Control

- $x' = f(x, u)$
- $y = h(x)$
- These equations are constraints on the system

Performance Measurement

- $J = \int_0^{\inf} g(x, x', u)dt =$ cost function

Particle Example

- f(x)
- g(x) = 0
- $L = g(x, u) + \lambda f(x - u) - x'$
- $\frac{dL}{dx}$
- $\frac{dL}{d\lambda}$
- $x_{m+1} = Ax_m + Bu_m + \epsilon_m$
- $y_m = Cx_m$
- $J = \sum_{m=0}^{T-1}(x_m^T Q x_m + u_m^T R u_m) + x_T^T Q_T x_T$
- Infinite horizon -> for stability, T = inf
- Finite -> positioning
- $x^T y = scalarproduct = x^T Q y$
- $x^T x = ||x||$

Optimal Control

Controlability

- $x_{m+1} = (\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}) x_m + (\begin{matrix} 1 \\ 1 \end{matrix}) u_m$

- $y(u) = (1, 1)x_m$
- Pick some x such that, go from 0 to any x
- (2, 2) works but (2, 3) doesn't because the system only works on the diagonal
- $x_{m+1} = Ax_m + B_m$
  - $x_0 = \binom{0}{0}$
  - $x_1 = Bu(0)$
  - $x_2 = ABu(0) + Bu(1)$
  - $B, AB, A^2B, ...$
  - If the matrix is full rank it is controllable

Observable

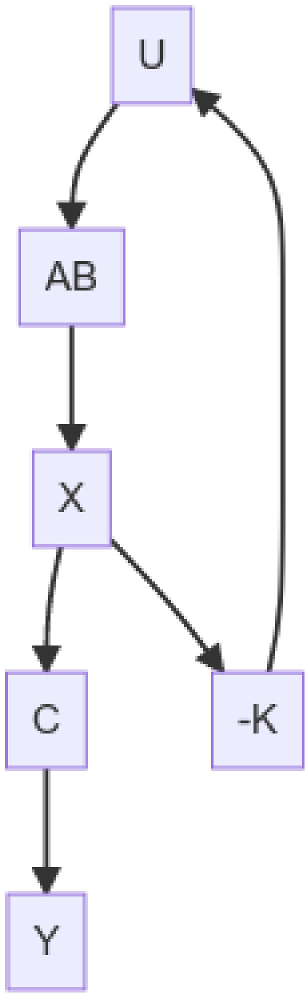
- Reconstruct x(0) from observing $x_1, x_2, x_3, ...x_m$
- $x_{m+1} = (\begin{matrix} 1 & 0 \\ 1 & 0 \end{matrix})x + \binom{1}{1}u$
- y(u) = (0, 1) x
- This system is not observable because you can't view all states

Lecture 11

State Space Control

- $x_{m+1} = Ax_m + Bu_m$
- $y_m = Cx_m$
  - No noise
  - Controllable
  - Observable

- Full state feedback control



- $u_m = -kx_m$
- $x_{m+1} = Ax_m - BKx_m$
- $= (A - BK|x_m)$
- We get to pick the value of K

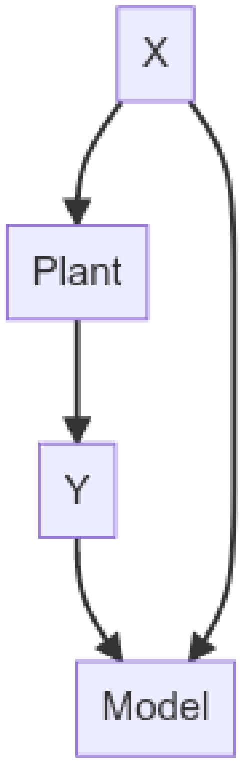
- $x_m = 0$
- $x_{m+1} = A * 0 - KB0$
- $x_{m+1} = (A - BK)x_m$
- $||x_{m+1}|| = ||(A - BK)x_m||$
- $||x_{m+1}|| <= ||A - BK||||x_m||$

Control Example

- $y'' + y' + y = u$
- $x_{m+!} = Ax_m$
- Want to control to y*
- $u = K(x^* - x)$

Observer

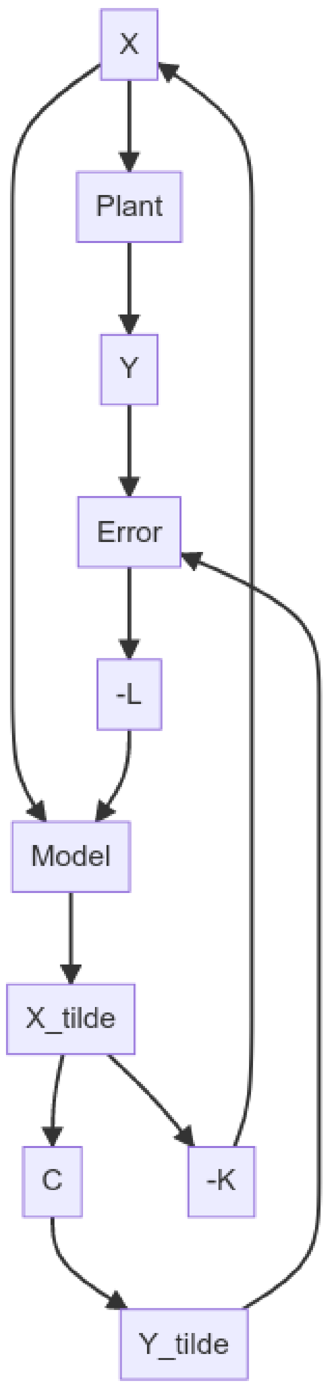
- Estimates the current state



- Model means that you know A, B, C
- $\widetilde{x_{m+1}} = A\widetilde{x_m} + Bu_m$
- As long as you know X_0
- $\widetilde{x_0} = x_0$
- $\widetilde{x_1} = Ax_m + Bu_m$
- This is open loop control

Construct an observer

- Control observation error to zero
- $\widetilde{x_{m+1}} = A\widetilde{x_m} + Bu_m + L(y_m - \widetilde{y_m})$


X
Plant
Y
Error
-L
Model
X_tilde
C
-K
Y_tilde

Plant

- $x_{m+1} = Ax_m + Bu_m$
- $y_m = Cx_m$

Model

- $\widetilde{x_{m+1}} = A\widetilde{x_m} + Bu_m + L(y_m - \widetilde{y_m})$
- $\widetilde{y_m} = C\widetilde{x_m}$

Error Analysis

- $error = x_{m+1} - \widetilde{x_{m+1}}$
- $Ax_m = Bu_m - A\widetilde{x_m} - Bu_m - L(y_m - \widetilde{y_m}) - cx_m + c\widetilde{x_m}$
- $= (A - Lc)(x_m - \widetilde{x_m})$
- Pick L so poles are in the unit circle
- (A - Bk) = luneberg observer

Lecture 12

Observer

- Run a second system in parallel and try to estimate the internal state
- $\widetilde{x_{m+1}} = A\widetilde{x_m} + Bu_m = L(y_m - \widetilde{y_m})$
- Pick the L such that it goes ot 0

Plant

- $x_{m+1} = Ax_m + Bu_m$
- $u_m = -kx_m$
- $= -k\widetilde{x_m}$
- $= x_m - e_m$ where $e_m = x_m - \widetilde{x_m}$

- e_m approaches 0, x_m approaches 0

Learning and Fusion

- $\widetilde{x_{m+1}} = A\widetilde{x_m} + L(y_m\widetilde{y_m})$

- Want a finite memory filter because we don't want to remember all previous values.
- Predict -> Observe -> Update
- $S - N = \frac{1}{N}\sum_{k=1}^{N} y_k$

- $S - N = \frac{1}{N}\sum_{k=1}^{N-1} y_k + \frac{1}{N}y_N$

- $\frac{N-1}{N}\frac{1}{N-1}\sum_{k=1}^{N-1} y_k + \frac{1}{N}y_N$

- $S_{N-1} + \frac{1}{N}(y_m - S_{N-1})$

Example

- Everything is gauss noise

- $VAR(x) = \sigma_x^2$

- How do you use fuse 2 guassian observations

- x,y -> E(x), E(y)
  - Choose 0 <= K <= 1

  - $kx + (1-k)y$

  - $a = k, b = (1-k)$

  - Want to minimize the variance

  - $VAR(kx + (1-k)y)$

  - $E((E(ax+by) - ax - by)^2)$

  - $E((aE(x) + bE(y) - ax - by)^2)$

  - $(a(E(x) - x) + b(E(y) - y))^2$

  - $VAR(ax+by) = a^2VAR(x) + b^2VAR(y) + 2abCO - VAR(x,y)$

- $k^2\sigma_x^2 + (1-k)^2\sigma_y^2$
- Minimize
  - $k^2\sigma_x^2 + \sigma_y^2 - k\sigma_y^2 + k^2\sigma_y^2$
  - $\frac{d}{dk} = 2k\sigma_x^2 - \sigma_y^2 + 2k\sigma_y^2 = 0$
  - $k = \frac{\sigma_y^2}{\sigma_x^2 + \sigma_y^2}$

Lecture 13

Predictor & Observer

- Use the prediction and observation to determine state
- $P(x_m|x_{m-1})$ = prediction
- $P(x_m|y_m)$ = measurement of state
- Guassian assumption
- $E(x_m) = \widetilde{x}_m$
- $VAR(x_m) = P_m$
- $Given \widetilde{x}_{m-1} P_{m-1}$
- $Predict \widetilde{x}_m^+ P_m^+$
- $Observe y_m$
- Update to get $\widetilde{x}_m P_m$

Kalman Filter

- Scalar
- $x_{m+1} = ax_m + v_m$
- $y_m = cx_m + q_m$
- $N(0, \sigma_v^2)$

- $N(0, \sigma_q^2)$
- $\widetilde{x}_m = E(x_m)$
- $P_m = VAR(x_m - \widetilde{x}_m)$

1. Predict -> given $\widetilde{x}_{m-1} P_{m-1}$
   - $\widetilde{x}_m^+ = A\widetilde{x}_{m-1}$
   - $\widetilde{P}_m^+ = VAR(x_m - \widetilde{x}_m^+)$
   - $= VAR(Ax_{m-1} + v_m - A\widetilde{x}_{m-1})$
   - $= a^2 VAR(x_{m-1} - \widetilde{x}_{m-1}) + VAR(v_m) + 2CO - VAR(x_{m-1}, -\widetilde{x}_{m-1}, v_m)$
   - $= a^2 P_{,-1} + \sigma_v^2$
   - $\widetilde{x}_m^+ = a\widetilde{x}_{m-1}$
   - $P_m^+ = a^2 P_{m-1} + \sigma_v^2$
2. Observe y_m
   - $\widetilde{x}_m = \widetilde{x}_m^+ + k(y_m - \widetilde{y}_m)$
   - $\widetilde{x}_m = \widetilde{x}_m^+ + kcx_m - kc\widetilde{x}_m^+ + kq_m$
   - $P_m = VAR(x_m - \widetilde{x_m}) = E((x_m - \widetilde{x}_m)^2)$
   - $x_m - \widetilde{x}_m = x_m - \widetilde{x}_m^+ - k(cx_m + q - \widetilde{y}_m)$
   - $x_m - \widetilde{x}_m^+ - kcx_m - kc\widetilde{x}_m^+ - kq_m$
   - $(1 - kc)(x_m - \widetilde{x}_m^+) - kq_m$
   - $(ax + b)2$
     - $a = (1 - kc)$
     - $a = (x_m - \widetilde{x}_m^+)$
     - $k = -kq_m$

- $E((1 - kc)^2 (x_m\ widetildex_m^+) + k^2 q_m^2 + 2(a - kc)(x - \widetilde{x}_m^+))$

- $(1-kc)^2E((x_m-\widetilde{x}_m^+)^2)=k^2E(q_m^2)$
- $(1-2kc+k^2c^2)P_m^+ + k^2\sigma_q^2$
- $\frac{d}{dk}=-2cP_m^+ + 2kc^2P_m^+ + 2k\sigma_q^2=0$
- $k=\frac{cP_m^+}{c^2P_m^+ + \sigma_q^2}$
- $\widetilde{x}_m=\widetilde{x}_m^+ + k(y_m-c\widetilde{x}_m^+)$
- $P_m=(1-kc)P_m^+$

Scalar Kalman

- $Givena, c, \sigma_v^2 = modeluncertainty, \sigma_q^2 = observationuncertainty$
- Start at $\widetilde{x}_0$ with P_0
- Predict
  - $\widetilde{x}_m^+ = ax_{m-1}$
  - $P_m^+ = a^2P_{m-1}+\sigma_v^2$
- Update y_m
  - $\widetilde{x}_m=\widetilde{x}_m^+ + k(y_m-c\widetilde{x}_m^+)$
  - $P_m=(1-kc)P_m^+$
  - $k=\frac{cP_m^+}{c^2P_m^+ + \sigma_q^2}$

Lecture 14

Kalman

- $x_{m+1}=Ax_m=w_m$
- $y_m=cx_m+v_m$
- $E(w)=0, E(v)=0$
- $VAR(w)=Q$
- XX^T = CO-VAR matrix

- $E(ww^T) = Q, E(vv^T) = R$

Observer

- $\widetilde{x}_m = E(x_m)$
- $P_m = VAR(x_m - E(x_m)$
- $= E((x_m - \widetilde{x}_m)(x_m - \widetilde{x}_m)^T)$

Predict

- $\widetilde{x}_m = A\widetilde{x}_{m-1}$
- $P_m^+ = E((x_m - \widetilde{x}_m^+)(x_m - \widetilde{x}_m^+)^T)$
- $E(A()()^T A^T + 2A()v_m^T + v_m v_m^T)$
- $E(A()()^T A^T = v_m v_m^T)$
- $P_m^+ = AP_{m-1}A^T + Q$
- $\widetilde{x}_m^+ = A\widetilde{x}_{m-1}$

Update

- Observe y_m
- $\widetilde{x}_m = \widetilde{x}_m^+ = k(y - m - c\widetilde{x}_m^+)$
- $\widetilde{x}_m = \widetilde{x}_m^+ + kcx_m + kv_m - kc\widetilde{x}_m^+$
- $= (I + kc)(x_m - \widetilde{x}_m^+) + kv_m$
- Minimize P choosing K
- $E((x_m - \widetilde{x}_m)(x_m - widetildex_m)^T)$
- $\widetilde{x}_m = \widetilde{x}_m^+ + kcx_m + kv_m - kc\widetilde{x}_m^+$
- $x_m - \widetilde{x}_m = x_m - \widetilde{x}_m^+ - kcx_m - kv_m + kc\widetilde{x}_m^+$
- $(I - kc)(x_m - \widetilde{x}_m^+) - kv_m$

- $aa^T + ba^T = ab^T + bb^T$
- $(I - kc)(x_m - \widetilde{x}_m^+)(x_m - \widetilde{x}_m^+)^T(I - kc)^T + kv_m v_m^T k^T$

Kalman Steps

- Given $\widetilde{x}_0, P_0, Q, R$
- $\widetilde{x}^+(m) = A\widetilde{x}(m-1)$
- $P^+(m) = AP(m-1)A^T + Q$
- Update with y(m)
  - $k = \frac{P^+(m)C^T}{cP^+(m)C^T+R}$
  - $\widetilde{x}_m = \widetilde{x}_m^+ + k(y(m) - c\widetilde{x}^+(m))$
  - $P(m) = (1 - kc)P^+(m)$

Lecture 15

Bayesian Reasoning

Linear Least Sqaures

State Space Control Principles

- Controllable
- Observable
- Luneberg observer

Regression

Expected Value, Variance

- Matrix version
- Scalar version

Fusion (optimal linear mix) + Scalar Kalman

How to Practice

Bayesian Reasoning Example

- P(A | B) -> B has happened
- $P(A|B) = \frac{P(A \cap B)}{P(B)}$
- $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$
- Partitioning is when you split into sections
  - $\cup B_k = u$
  - $B_k \cap b_l =$ Disjoint
  - $P(k) = \sum P(A|B_k)P(K_k)$

Linear Least Squares Example

- (x^k, y^k)
- Linear Model F(x) = $\sum \alpha_k J_k(x)$
- $\frac{Minimize}{\alpha_1 \alpha_n} \sum_{k=1}^{M} ||F(x^k) - y^k||$
- Given (x^k, y^k), k=1 … N
- Model is f(x) = aX
- $\begin{bmatrix} x^1 \\ x^2 \\ x^N \end{bmatrix} * \begin{bmatrix} a \end{bmatrix} = \begin{bmatrix} y^1 \\ y^2 \\ y^N \end{bmatrix}$
- $A^T Ax = A^T y$
- $\begin{bmatrix} x^1 & ... & x^n \end{bmatrix} * \begin{bmatrix} x^1 \\ ... \\ x^N \end{bmatrix} a = \begin{bmatrix} x^1 & ... & x^n \end{bmatrix} y$
- $\sum x_k^2 a = \sum x_k y_k$

- $a = \frac{\sum x_k^2}{\sum x_k y_k}$

Probability Example

- Random Var X -> x = scalar
- Random Vector (X_1, X_2) = matrix
- E(x) -> Definition u(x) = $\frac{1}{N} \sum x^k$
- $VAR(x) = E((E(x) - x)^2)$
- $CO - VAR(x, y) = E((E(x) - x)(E(y) - y))$

Scalar Example

- x is a random variable, C is a constant
- VAR(x + C) = VAR(x)
- f(x) is a linear mapping
- $\Re - > \Re$
- I know the variance of x.
- What is the VAR(f(x))
- f(x) = ax
- VAR(ax) = aˆ2 VAR(x)

Vector Example

- $VAR(X) = E((E(x) - x)(E(x) - x)^T)$
- $(AB)^T = B^T A^T$
- $X^T y = y^T X$

State Space Control Example

- Controllable, Observable
- $x_{m+1} = Ax_m + Bu_m$
- $y_m = Cx_m$

Lecture 16

Kalman

State x_m
Measuring y_m
Everything is Gauss
Estimate $\widetilde{x}_m = E(x_m)$
$P(m) = VAR(x_m - \widetilde{x_m})$

Model

$x_m = Ax_{m-1} + q_m$
$y_m = (x_M + something_m)$

$\widetilde{x}_1^+$
$\widetilde{x}_m^+ = A\widetilde{x}_{m-1}$
$P_m^+ = AP_{m-1}A^T + Q$

We want to minimize the prediction error

Predict: $\widetilde{x}_m^+ - > C\widetilde{x}_m^+ \widetilde{y}_m$

$\widetilde{x}_m = \widetilde{x}_m^+ + k_m(y_m - \widetilde{y}_m^+)$
$k = \frac{\text{model uncertainty}}{\text{uncertainty in model + uncertainty in measurement}}$
$k = \frac{P_m^+ C^+}{P_m^+ C^+ + R}$

Simple Example

A, B, C, Q, R
AB for the model
C for the robot
A = Matrix
C = output
Q = model uncertainty (one of these two is wrong)

R = model uncertainty

$y'' = \alpha_1 y' + \alpha_0 y = 0$
$system = \begin{bmatrix} y \\ y1 \end{bmatrix}$
$\begin{bmatrix} 1 & 0 \\ -\alpha_0 & -\alpha_1 \end{bmatrix}$
$alpha_1, alpha_0 > 0$
Spiral -> creates a spiral when you plot y and y_!, when you plot in state space you get a spiral

y" = -g
y'= v
v' = -g

g is constant!, $\dot{g} = 0$
y' = 0
v' = -0
g' = 0
$\begin{bmatrix} y \\ y_1 \\ y_2 \end{bmatrix}$
$y' = \begin{bmatrix} 0 & 10 & \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$
(I + hA)

y" = -g + noise
g = 3.1

Parameter Tuning

Q and R
Measuring α and distance d
y" = -g
If the kalman equation doesn't converge and continues straight you need to add a state variable
d sin(d) $\begin{bmatrix} x & \dot{x} \\ y & \dot{y} \\ \theta & \dot{\theta} \end{bmatrix}$
$\dot{x} = vcos(\theta)$
$\dot{y} = vsin(\theta)$
Landmark
As a robot you measure the distance and direction to all the landmarks

Lecture 17

Extended Kalman Filter

Simple Robot

Differential drive robot

$$\text{state} = \begin{bmatrix} x \\ y \\ \theta \\ \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{bmatrix}$$

There are no direct sensors to measure speed
All of these are found in a typically IMU

$$\text{Robot} = \begin{bmatrix} \dot{x} = vcos(B) \\ \dot{y} = vsin(B) \\ \dot{\theta} \end{bmatrix}$$

$$\text{World} = \begin{bmatrix} \dot{x} = vcos(\theta + \frac{B}{2}) \\ \dot{y} = vsin(\theta + \frac{B}{2}) \\ \dot{\theta} = B \end{bmatrix}$$

$-\pi \leq \theta \leq \pi$

$d = \sqrt{(x - x_m)^2 (y - y_m)^2}$

$Actan2(x, y)$

$x_{m+1} - f(x_m, u_m)$

$y_m = h(x_m)$

Predict

$\widetilde{x}_m^+ = f(\widetilde{x}_{m-1}, u_{m-1}) = P_m^+$

Update

$f(x+h) = f(x) + hf(x)$
$= (A + hJ_A)x$

$$J_f(x) = \begin{bmatrix} \frac{df_1}{dx_1} & \frac{df_1}{dx_2} & ... & \frac{df_1}{dx_m} \\ ... & ... & ... & \\ \frac{df_m}{dx_1} & ... & \frac{df_m}{dx_m} & \end{bmatrix}$$

$$f(x_1...x_m) = \begin{bmatrix} f_1(x1...xm) \\ ... \\ f_m(x1...xm) \end{bmatrix}$$

Robot Example

$state = \begin{bmatrix} x \\ y\theta \end{bmatrix}$

$control = \begin{bmatrix} v \\ steer \end{bmatrix}$

$x_{m+1} = \begin{bmatrix} x_m + dtcos(\theta_m) \\ y_m + dtsin(\theta_m) \\ \theta_m + dtB \end{bmatrix} \begin{bmatrix} f_x \\ f_y \\ f_\theta \end{bmatrix}$

$J_f = \begin{bmatrix} 1 & 0 & -dtvsin(\theta) \\ 0 & 1 & vcos(\theta) \\ 0 & 0 & dtv \end{bmatrix}$

$\widetilde{x}_{m+1} = f(x(state), \genfrac{}{}{0pt}{}{V}{B}(control))$

Q = modelling uncertainty
Noise in V means that noise is not square but is only in direction of driving
Geometry of the noise differs from the state space

$$J_A Control = \begin{bmatrix} \frac{df}{dv} & \frac{df}{d\beta} \end{bmatrix}$$

$$J_A Control = \begin{bmatrix} dtcos(\theta) & 0 \\ dtsin(\theta) & 0 \\ 0 & dt \end{bmatrix}$$

Marker

$h = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$

$$\begin{bmatrix} \sqrt{(x_m - x)^2 + (y_m - y)^2} \\ atan(\frac{y_m - y}{x_m - x}) - \theta \end{bmatrix}$$

$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$ <-> $\begin{bmatrix} x_m \\ y_m \end{bmatrix}$ Computes the distance and the angle between the location and the marker

$J_f(x, y, \theta)$

Lecture 18

Particle Filter

Markov Model (Chains)
x_0 -> x_1 -> x_2 -> x_3
y_1 -> y_2 -> y_3

$P(x_k|x_{k-1}))$ Dynamics $f(x_k|x_{k-1})$
$P(y_k|x_k))$ Output $g(y_k|x_k)$

We want:
$P(x_k|y_1...y_k)$ this one has the current state, so y_k
$P(x_k|y_1...y_{k-1})$ this one is a prediction, so y_{k-1}

Facts "Markov Property"

$P(x_k|x_{k-1}, x_{k-2}, x_0 = P(x_k|x_{k-1})$ only depends on previous states
$P(y_K|x_k...x_0) = P(y_k|x_k)$ means the covariance is 0, and current position is only based on current state
$P(x_k|y_1...y_k) = \frac{P(x_k \cap y_1...y_k)}{P(y_1,...,y_k)}$ $\cap$ can be replaced with a ,

Dynamic

$P(x_k|y_1...y_k) = P(x_{k-1}, y_1...y_{k-1}) * f(x_k|x_{k-1}) * g(y_k|x_k)$
$P(x_k|y_1...y_k) = P(x_{k-1}, y_1...y_{k-1}) * \frac{f(x_k|x_{k-1}) * g(y_k|x_k)}{P(y_k|y_1...y_{k-1})}$
We normailze the above equation given the observation.

Monte Carlo Simulation

$E(x) = \int xP(x)dx \approx \frac{1}{N}\sum x_k$
x_k are pulled from P(x)
$E(f(x)) = \int f(x)P(x)dx$

Trick

$E(f(x)) = \int f(x)p(x)\frac{\pi(x)}{\pi(x)}dx$
$E(f(x)) = \int f(x)\frac{p(x)}{\pi(x)}\pi(x)dx$ This allows you to pick a new distribution (typically Gaussian to make your life easy)

Motion

Sequential Important Resampling
$f(x)$ $E(f(x)) = \int f(x)p(x)dx$
$E(f(x)) = \int f(x)\frac{p(x)}{\pi(x)}\pi(x)dx$
$E(f(x)) \approx \sum f(x_k)w_k dx$ $\sum w_k = 1$ $\pi(x)$ drops out because you are pulling samples for Monte Carlo

Update:
$\int f(x)\frac{p(x|y)}{\pi(x)}\pi(x)dx$
$\int f(x)\frac{p(x|y)p(x)}{p(y)\pi(x)}\pi(x)dx$

Particles

(x^k, w^k) $k = 1...N$

#Lecture 19

Sequential Importance Resampling (Sequential Monte Carlo)

$P(x_k|y_1...y_k)$

$E(f(x)) = \int f(x)p(x)dx$

$\approx \frac{1}{N}\sum w_k x_k$

P(x) is gauss: $\begin{bmatrix} \mu = expectedvalue \\ \sigma^2 = variance \end{bmatrix}$

Particles

Camel Shaped Graph

PDF = probability density function

$P = (x_k, w_k)$

$\sum w_k = 1$

$P(x) = \sum w_k \delta(x - x_k)$

$E(f) = f(x_k)$

Example 2

Parabola

$w_k = \frac{1}{N}$

Because you're particles aren't distributed as they are supposed to be you have to resample

Resample

$$w_1 = 0.1$$
$$w_2 = 0.5$$
$$w_3 = 0.1$$
$$w_4 = 0.3$$

CDF = cummulative density function

$CDF(x) = \sum w_k$ where $w_k < x$

[0.1 0.5 0.1 0.3]

1. Randomly select a number
2. Use the output to select a weight and clone it
3. Repeat many many times and you will have a PD equal to the original function

You would compute the CDF and then do your selection to place particles where you want

Spinning may not be ideal if you lack samples or by chance the selection is skewed

Chose the smallest one and then iterate through using the smallest one

Markov chain with our states

Want to compute $P(x_k|y_1...y_k)$

$P(x_k|x_{k-1}...x_0) = P(x_k|x_{k-1})$ -> f

$P(y_k|x_k...x_0) = P(y_k|x_k)$ -> g

$P(x_K, y_1...y_k) = P(x_{k-1}, y_1...y_{k-1}) -> f(x_k|x_{k-1})$ and $g(y_k)$

$P(x_k|y_1...y_k) = P(x_{k-1}|y_1...y_{k-1})$

Use the Bayesian
$\frac{f(x_k|x_{k-1})g(y_k|x_k)}{P(y_k|y_1...y_k)}$

You can ask the clone to solve for g(stuff)

$E(f(x)) = \int f(x)p(x)dx$

$= \int f(x)p(x)\frac{\pi(x)}{\pi(x)}dx$

$= \int f(x)\frac{p(x)}{\pi(x)}\pi(x)dx$ Rearranged so we can pull from pi(x) or any probability distribution

(x_k, w_k)

Initially all particles are evenly distributed -> $w_k = \frac{1}{N}$

1. Compute Dynamics
   x_k <- f(x_k)
2. Observe [y_k] <-
   $w_k < -w_k * \pi(y_m - \widetilde{y_k})$
   Where pi is a normal distributino $N(y_k - \widetilde{y_{k-1}}, \sigma)$
   Rank each particle according to the weight and the likelihood of it's observation being true Use an exponential function for the assignin the weights

3. Normalize $w_k < -\frac{w_k}{\sum w_k}$
4. Resample with noise (may normalize again)
5. Goto 1

#Lecture 20

Bellman Equation

Discrete Time
Discrete State Space

a is part of set A which are actions
reward = R

0.1	0.8	0.1
0	*Robot*	0
0	0	0

$x_{m+1} = Ax_m + \epsilon_m$

Policy = Controller
$\pi(S, a)$ probability of taking action a in state s.

Markov Model

States S_i Actions A_i
Transitions $S_i - a - > S_k$
In state S action A leads to S^1

$\bigvee S \sum_a P^a_{ss^1} = 1$

Markov Chains (Rewards)

A_i -> A_{i+1} -> A_{i+2} -> A_{i+m-1}

S_i -> S_{i+1} -> S_{i+2} -> S_{i+m}

R = reward
A = action
S = state

$R_i = r_{i+1} + r_(i+2) + ... $

$R_i = \sum_{k=0}^{\inf} r_{i+k+1}$

Discounted Reward

$0 \leq \gamma \leq 1$

$R_i = \sum_{k=0}^{\inf} \gamma^k r_{i+k+1}$

$R_i = r_{i+1} + \gamma r_{i+1} + \gamma^2 r_{i+2}$

$V(s) = E(r_i | s_i = s)$

$Q(s, a) = E(r_i | s_i = s_1, a_i = a)$

$R_i = \sum_{k=0}^{\inf} \gamma^k r_(i + k + 1)$

$R_i = r_{i+1} + \sum_{k=1}^{\inf} \gamma^k r_(i + k + 1)$

$R_i = r_{i+1} + \gamma \sum_{k=0}^{\inf} \gamma^k r_(i + k + 2)$

Immediate reward + the discounted future reward

$V(s) = E(r_{i+1} + \gamma \sum_{k=0}^{\inf} r_{i+k+2} | s_i = s)$

Notation

$P_{ss^1}^a = P_r(s_{i+1} = s^1 | s_i = s_1, a_i = a)$

$R_{s s^1}a = E(r_{i+1} | s_i=s_1, a_i = a, s_{i+1} = s^1)

Policy

$\pi(s, a)$

$V(s)^\pi = E_\pi(\sum_{k=0}^{\inf} \gamma^k r_{i+k+1} | s_i = s)$

$= E_\pi(r_{i+1} + \gamma \sum_{k=0}^{\inf} \gamma^k r_{i+k+2} | s_i = s)$

Immediate reward

$E_\pi(r_{i+1} | s_i = s) = \sum_a \pi(s, a) \sum_{s^1} P_{ss^1}^a r_{ss^1}^a$

Delayed Reward

$E_\pi(\gamma \sum_{k=0}^{\inf} \gamma^k r_{i+k+2} | s_i = s)$

$= \sum_a \pi(s, a) \sum P_{ss^1}^a \gamma E(\sum \gamma^k r_{i+k+2} | s_i = s)$

Bellman Equation

$V(s)^\pi = \sum \pi(s, a) \sum P_{ss^1}^a + \gamma V(s^1)^\pi$

Expected reward under state s with policy pi is all possible choices of the policy plus all possible choices by non determinism of the system immediate robot + discounted future reward.

$Q^\pi(s, a) = \sum_{s^1} P_{ss^1}^a (R_{ss^1}^a + \gamma \sum \pi(s^1, a^1)) Q^\pi(s^1, a^1))$

Optimal π

Maximize $V(s)^\pi$

Lecture 21

Markov Chains

$R_i \sum_{k=0}^{\inf} \gamma^k r_{i+k+1}$

$P^a_{ss^1} = P_R(s_{i+1} = s^1 | s_i = s, a_i = a)$

$R^a_{ss^1} = E(r_{i+1} | s_i = s_1, a_i = a, s_{i+1} = s^1)$

Policy

$\pi(s, a)$ probability of taking action a in state s

$V(s)^\pi = E_\pi(R_i | s_i = s)$

$Q(s)^\pi = E_\pi(R_i | s_i = s, a_i = a)$

$V(s)^\pi = \sum_a \pi(a|s) Q^\pi(s, a)$

Potential at this state = sum of the probability of taking an action and the reward for taking that action

Bellman Equation

$V(s)^\pi = \sum \pi(s, a) \sum P^a_{ss^1}(R^a_{ss^1} + \gamma V(s^1)^\pi)$

$Q(s, a)^\pi = \sum P^a_{ss^1}(R^a_{ss^1} + \gamma \sum_{a^1} \pi(s^1, a^1) Q^\pi(s^1, a^1))$

$V(s)^* = MAX \sum P^a_{ss^1}(R^a_{ss^1} + \gamma V(s^1)^*)$

Greedy = want to maximize your problem

Optimal Policy

π_1, π_2 -> $V(s)^{\pi_1} \geq V(s)^{\pi_2}$

$V(s)^* = MAX \pi of V(s)^\pi$

$\pi(s, a) = \begin{matrix} 1 & MAXV(s)^* \\ 0 & \end{matrix}$

- There is an optimal solution
- It is not unique
- How do I find it
- Dynamic programming (optimization)
- Value iteration
- Reinforcement learning

Dynamic Programming

$V(s)^m + 1 = MAXa \sum P^a_{ss^1}(R^a_{ss^1} + \gamma V(s^1)^m)$

A Star

Can use A* for path finding. Start with the goal and everything set to 0. Work backwards from the goal and subtract 1. Everytime you can replace with a value that is higher than current you do. Does not matter about the start

Lecture 22

Bellman Equation

$V(s)^\pi = \sum_a \pi(s,a) \sum_{s^1} P^a_{ss^1}(R_{ss^1} + \gamma V(s^1)^\pi)$

$V(s)^\pi = \sum_a \pi(s,a) Q^\pi(s,a)$

$Q^\pi(s,a) = \sum_{s^1} P^a_{ss^1}(R^a_{ss^1} + \gamma V(s^1)^\pi)$

$\pi(s,a) = MAXaQ(s,a)$ <- Greedy policy

Value Iteration

$V(s)^{\pi_1} \geq V(s)^{\pi^2}$

$V(s)^*$

$V(s)^m - > V(s)^{m+1} - > V(s)^{m+2}$

$V(s)^{m+1} = V(s)^m + MAXa(R^a_{ss^1} + \gamma V(s^1)^m)$

Assumptions: - Model is known - Online learning

Model less Value Iteration

$s_1, s_2, s_3, s_4, ..., s_N$

$V(s_{m+1}) = V(s_m) + \gamma(V(s_{m-1}) - V(s_m))$

The problem is that we have to wait for the agent to get the reward

Temporal Difference Learning

$V(s_m) < -V(s_m) + m(R + \gamma(V(s_{m-1}) - V(s_m)))$

Q learning

Learning based on actions

$Q(s, a) < -Q(s, a) + m(R + \gamma MAX$ a $(Q(s^1, a^1) - Q(s, a)))$

Lecture 23

Control, Optimal, Predictive

System
$x' = f(x, u)$

$y = h(x)$

$x_{m+1} = Ax_m + Bu_m$

$y_m = Cx_m$

- Stability
- Control of finite position and final time
- Optimality -> Cost
- Constraints
- Predictive, Horizons

Stability

BIBO stability
y" + y' + y = u
Inward spiral and we want to end up center
Asymptotically stable
lim y(t) -> 0 as t-> infinity

L(x) is a monotonic function (gets better every step) [Liapunov]
Your solutions must always be below L(x) $||x(t)|| \leq L(x)$
lim L(t) -> 0 as t-> infinity

Quadratic Forms

$V(x) = X^T PX =$ general scalar product -> always positive
$X^T(PX) = (PX)^T X$

$X^T P^T X$

$P = P^T$

Symetric positive definition $P \geq 0$

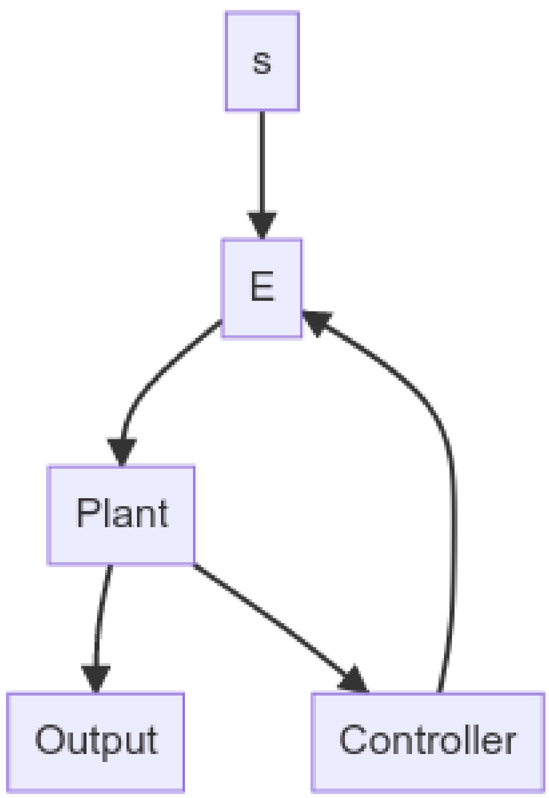
General Cost Problem

$\dot{x} = f(x, u)$
$y = h(x)$
$J = \int_0^T g(x, u)dt$

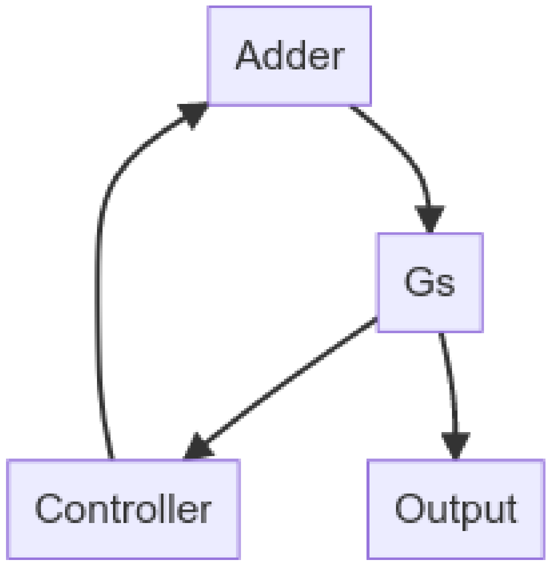
$x_{m+1} = Ax_m + Bu_m$
$y_m = CX_m$
$J = \sum_{m=0}^{N-1} x_m^T Q x_m + u_m^T R u_m + x_n^T Q_F n_n$

Find u_m to minimize J subject to $x_{m+1} = Ax_m + Bu_m$

Control to 0
Control to a set point



E = k(s-x)



$G = \frac{G(s)}{1-kG(s)}$ $G(0) = 0$ $k(r-x)$

Cannot control in steady state because G(s) goes to 0 so K have no effect

$\frac{1}{s}$ needed to cancel out the 0

PID control needs the I to learn the proper state to hold the controller when the system goes to 0

$x_{m+1} = AX$

$\widetilde{u}_{m+1} = u_m + u$

Lecture 24

Optimal Control

Observable, Controllable

System: $x_{m+1} = f(x_m, u_m)$

$J = \sum_{m=0}^{N} Cost(x_m, u_m)$

Finite Horizon x(0) = x_0 x(N)

Minimize J subject to $u_0 ... u_{N-1}$ system

1D for today

$x_{m+1} = ax_m + bu_m$

c = identity

$J_N = \sum_{m=0}^{N-1} qx_m^2 + ru_m^2 + q_T x_N^2$

weight of control + weight of state + terminal cost

r = reward

Bellman Equation

$V^*(s) = MAX$ a $\sum P_{ss'}^a (R_{ss'}^a + \gamma V(s'))$

Greedy policy

$V_m = MIN\ u_m\ (cost(x_m, u_m) + V_{m-1} f(x_m, u_m))$

V_N is the cost go N steps

$V_m = P_m x_m^2$ ——— $(x_m^T P_m x_m)$ in matrices

$V_0 = q_T x_T^2$ you can assume it is a quadratic function

$V_m = MIN\ u_m\ (cost(x_m, u_m) + V_{m-1} f(x_m, u_m))$

$V_m = P_m x_m^2$ and $f(x_m, u_m) = ax_m + bu_m$

$V_m = MIN\ u_m\ (qx_m^2 + ru_m^2 + P_{m-1}(ax_{m-1} + bu_{m-1}))$

Not using the indices $\frac{dV_m}{du_m} = 2ru + 2P(ax + bu_m * b)$

$u = \frac{abP_{m-1}}{r+b^2P_{m-1}} X = kX$

$= -a \frac{bP_{m-1}}{b^2 P_{m-1} + r}$

These problems are dual, solve one and you solve the other. If you observe you control error to 0 and when you control you control the plant to 0. b = control matrix, c = output matrix.

$V_m(s) = P_m x^2$ ——- $u = -kX$

$= qx^2 + rk^2x^2 + P_{m-1}(ax - bkx)^2$

$= qx^2 + rk^2x^2 + P_{m-1}(a - bk)^2 x^2$

Ricatti Equation

$P_m = q + rk^2 + P_{m-1}(a - bk)^2$

$P_0 = q_T$

$u = -kX$

$x_{m+1} = Ax_m + Bu_m$

$J_N = \sum_{m=0}^{N-1} x_m^T Q x_m + u_m^T R u_m + x_N^T + Q_T x_N$

x_N = control

Observe = x(0) from N observations of the system

Observe –>

Control <–

Optimal control is done by working backward from your end point

Lecture 25

Linear Quadratic Regulator

$x_{m+1} = x_m + u_m$ is an integrator

$J = \sum q x_m^2 + r u_m^2 + q - T x_N^2$

$V^m(s)$ cost m steps to go at state s

$V^m(x) = MIN$ u $(cost(x,u) + V^{m-1} f(x,u))$

$x_{m+1} = Ax_m + Bu_m$

$J = \sum_{m=0}^{N-1} x_m^T Q x_m + u_m^T R u_m + x_N^T Q_T x_N$

$V^m(x) = x^T P_m x$

$V^0(x) = x^T Q_T x$

$P^0 = Q_T$

$V^m(x) = x^T Q x + u^T R u + V^{m-1}(Ax + Bu)$

Minimize u

$V^m(x) = x^T Q x + u^T R u + (Ax + Bu)^T P_m (Ax + Bu)$

$V^m(x) = x^T Q x + u^T R u + x^T A^T P_m A x + x^T A^T P_m B u + u^T B^T P_m A x + u^T B^T P_m B u$

$V^m(x) = x^T Q x + u^T R u + x^T A^T P_m A x + 2u^T B^T P_m A x + u^T B^T P_m B u$

$\frac{d}{du} = 2Ru + 2B^T P A x + 2B^T P B u = 0$

$u = \frac{-B^T PA}{B^T PB + R} X = kX$

$u = -(R + B^T P_m B)^{-1} B^T P A$ –> have to be careful of singular

$x^T P_{m+1} x = x^T Q x + u^T R u + (Ax + Bu)^T P_m (Ax + Bu)$

$x^T P_{m+1} x = x^T Q x + x^T k^t R k x + (Ax + Bu)^T P_m (Ax + Bu)$

$u = -kx$

$(Ax - Bkx)^T P_m (Ax - Bkx)$

$x^T (A - Bk)^T P_m (A - Bk) x$

$P_{m+1} = Q + k^T Rk + (A - Bk)^T P_m (A - Bk)$

How to use it

Start with k0 and p0 at the end and work backward storing the state.

Star at a value, multiply by the ks and ps to solve

Linear Quadratic Regulator + Kalman (Linear Guassian Regulator)

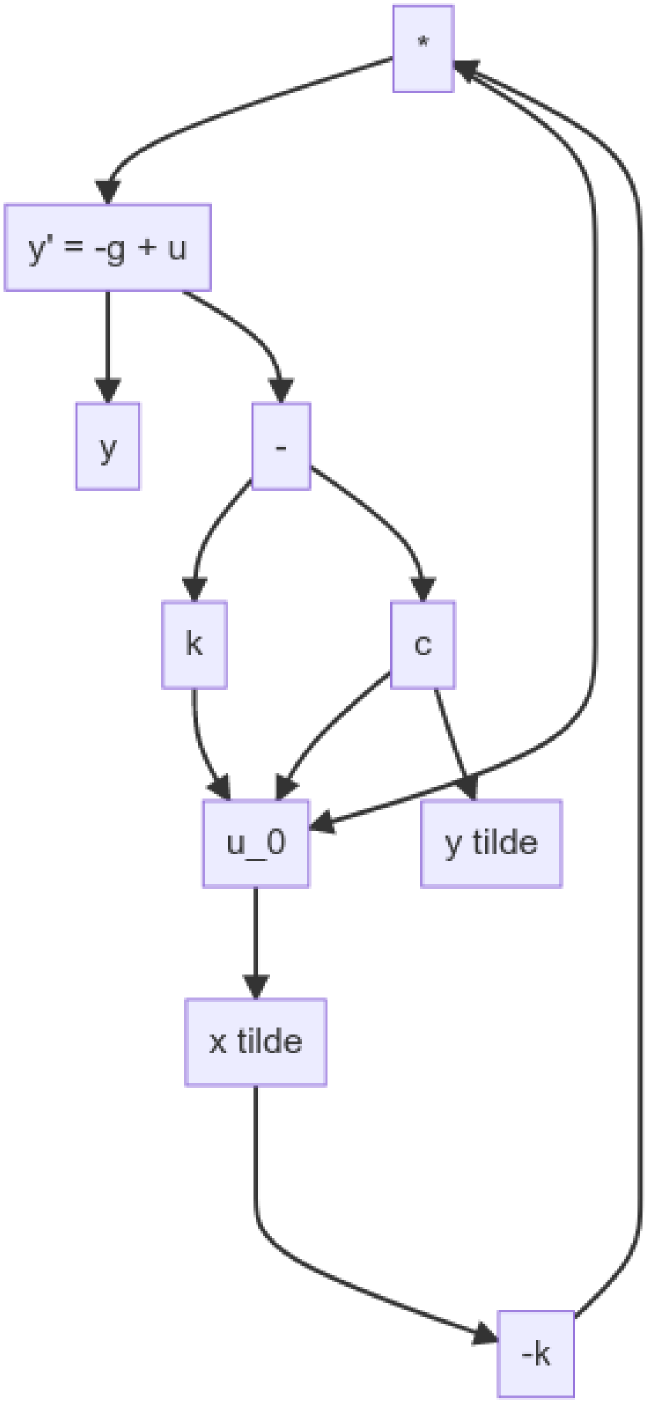
Moon lander problem

y" = -g + u

$$\begin{bmatrix} y \\ y' \\ g \end{bmatrix}$$

x(T) = 0 —— x'(T) = 0

y' = -g + u is a plant hidden to the controller y = output = height


*
y' = -g + u
y
-
k
c
u_0
y tilde
x tilde
-k

-k is stored

Lecture 26

Multi Predictive Control

x(t) = past

m = current

H = horizon

We are looking to compute the optimal for $u_m ... u_m + H$

MDC Implementation

If you an look into the future and see something better you can always go there even if it increases the cost to get there.

VFH^+ means you have precomputed maps to tell you what to do when driving (lunar rover)

Optimize MDC

$x_{m+1} = Ax_m + Bu_m$

$J = \sum_{m=0}^{N-1} x_m^T Q x_m + u_m^T R u_m + x_N^T Q_F x_N$

$x_0, \begin{bmatrix} x_1 & x_2 & ... & x_N \\ u_0, & u_1, & ... & u_{N-1} \end{bmatrix}$ 2N

$x_1 = Ax_0 + Bu_0$

$x_2 = A^2 x_0 + ABu_0 + Bu_1$

$x_3 = A^3 x_0 + A^2 Bu_0 + ABu_1 + Bu_2$

$$u = \begin{bmatrix} u_0 \\ u_1 \\ ... \\ u_{N-1} \end{bmatrix}$$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ ... \\ x_{N-1} \end{bmatrix}$$

$$\bar{Q} = \begin{bmatrix} Q & & \\ & ... & \\ & & Q_F \end{bmatrix}$$

$$\bar{R} = \begin{bmatrix} R & & \\ & ... & \\ & & R_F \end{bmatrix}$$

$x_0^T Q x_0 + \bar{x}^T \bar{Q} \bar{x} + \bar{u}^T + \bar{R} + \bar{u}$

$$\bar{x} = \begin{bmatrix} A \\ A^2 \\ ... \\ A^N \end{bmatrix} x_0 + \begin{bmatrix} B & & \\ AB & B & \\ A^2B & AB & B \\ ... & ... & ... \end{bmatrix} u$$

T = A matrix, S = AB matrix

$(Tx_0 + S\bar{u})^T Q(Tx_0 + S\bar{u}) + \bar{u}^T \bar{R}\bar{u} + x_0^T Q_F x_0$

$\bar{u}^T S^T QS\bar{u} + 2\bar{u}^T S^T \bar{Q} T x_0 + x_0^T T^T \bar{Q}^T T x_0 + \bar{u}^T \bar{R}\bar{u} + x_0^T Q_F x_0$

$\frac{dJ}{du} = 2(\bar{R} + S^T \bar{Q} S)u + 2S^T \bar{Q} T x_0 = 0$

$\frac{dJ}{du} = H\bar{u} + Fx_0$

$H\bar{u} = -Fx_0$

$u = -(H^T H)^{-1} H^T F x_0$

You always assume convex quadratic so minimum is first derivative. Convex is quadratic with constraints.

$V(x) = x^T Qx$ subject to $F_k \leq 0$

This has been shown to be very robust

Use a piecewise function

$u(k) = \begin{matrix} u_k & \tau_i < \tau i + 1 \\ 0 & \end{matrix}$

Hybrid system is a mix of continuous and discrete time

$y' = u \begin{matrix} -1 \\ 0 \\ 1 \end{matrix}$

Lecture 27

AI

SVM = support vector machines CNN = convolutional neural networks

AI Topics

- Classify
- Regression "Learning Function - Model Fitting"
- Symbolic "Expert Systems"

Learning

- Supervised learning (By example)
- Unsupervised learning (On your own)
- Re-enforcement learning (Bellman)

Measurement of success

(x^k, y^k) k = samples

x^k = data, y^k = class

Classifier cl

$cl(x^k) = y^k$

w = parameters

$ARGMIN$ w $\frac{1}{2}\sum_{k=0}^{M}||cl_w(x^k) - y^k||^2$

x = predicted, y = expected

Can use gradient descent to solve

Bayesian Classification

(x^k, y^k) k = samples

$P(y^k|x^k) = \frac{P(x^k|y^k)P(x^k)}{y^k}$

What is the probability it is a monkey = # of monkeys * probability of seeing a monkey / all monkeys

(x^k, y^k) = training data (80%)

$(\widetilde{x}^k, \widetilde{y}^k)$ = testing data (20%)

$P(y^k | x^k, x_1 . . .)

Linear Separable

Using different transformations to create data that you can linearly separate.

Class 1, Class 2

$P(1|x)$ —- $P(2|x)$ — Bayesian Classifier

$N(m_1|\sigma_1) - - - -N(m_2|\sigma_2)$

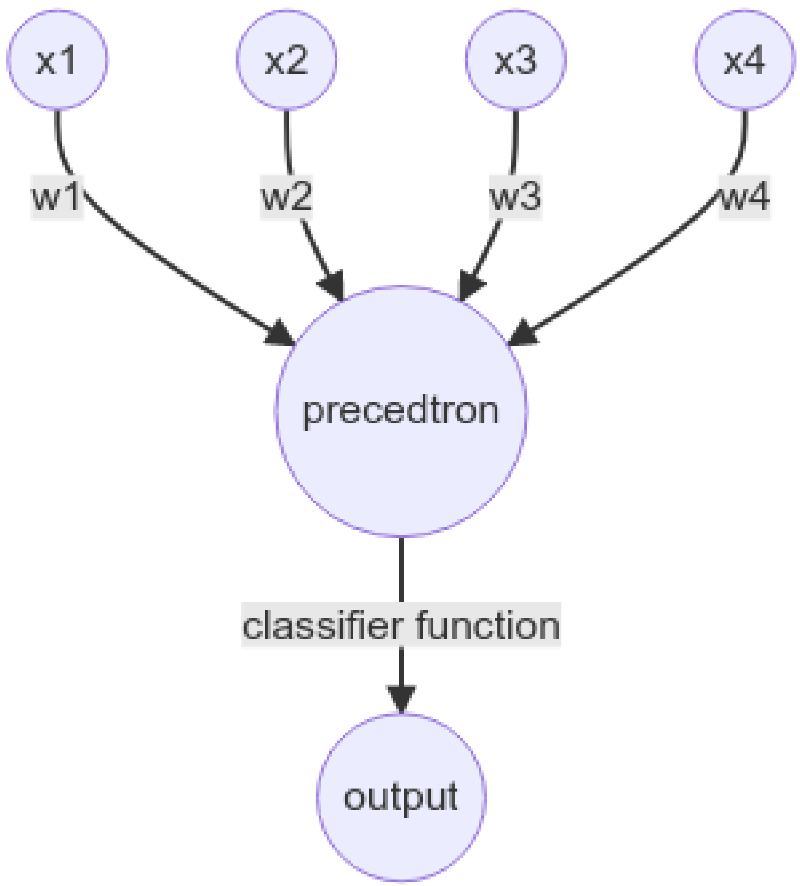
$P(1|x) = p(2|x)$

$(x^k|y^k)$

$$\begin{matrix} & > 0 \\ w^T x^k + b & 1 \\ & < 0 \end{matrix}$$

$x \epsilon R^m$ — $w \epsilon R^m$

$x^{\text{sample number}}_{\text{dimension of vector}}$



$L = \frac{1}{2} \sum_{k=1}^{M} |f(w^T x^k + b) - y^k|^2$

L = loss function

$ ARGMIN w $

$\frac{dL}{dw} = \sum (f(w^T x^k + b) - y^k)$

$w < -w + \delta \frac{dL}{dw}$

δ = learning rate

Lecture 28

Artificial Neural Networks (ANN)

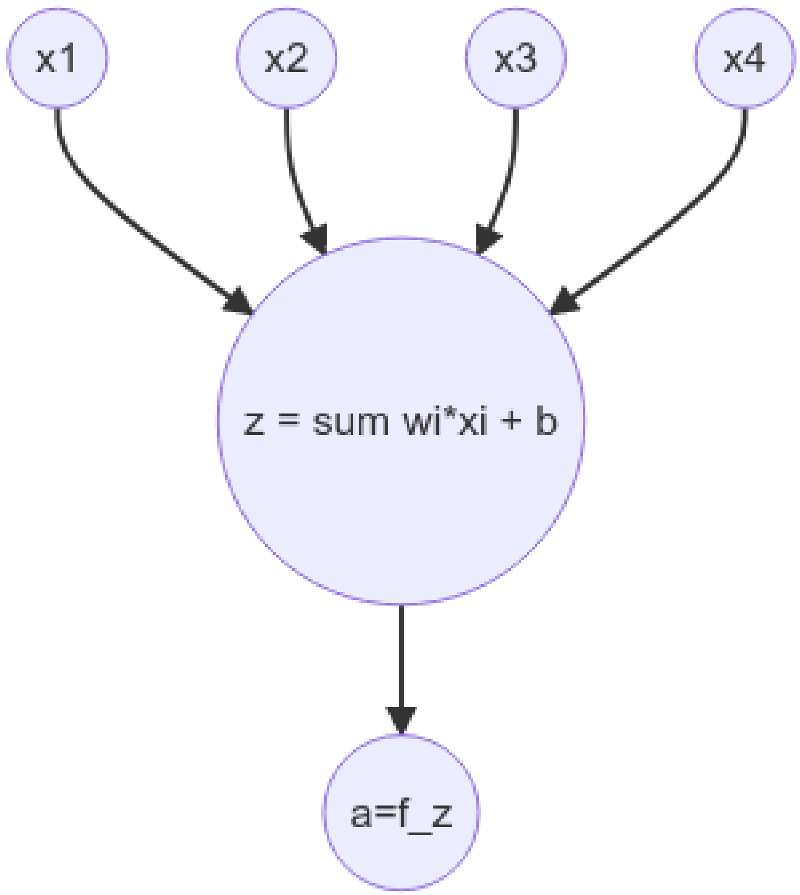
Back propagation = training method used for neural networks

Supervised Learning

- Classify
- Regression

(x^k, y^k)

$L = \frac{1}{2} \sum (cl(x^k) - y^k)^2$



R^N $->$ R

a = output

$-kx_n^k \leq 1$

$f(x) = \frac{1}{1+e^{-z}}$ is the sigmoid function

$L = \frac{1}{2}\sum(f(w^T x^k + b) - y^k)^2$

ARGMIN L with respect to w_k, b

$$\frac{dL}{d\theta} = (f(w^T x^k + b) - y^k) * f'(w^T x^k + b) * \begin{array}{cc} \theta = w_k & x_k \\ \theta = b & 1 \end{array}$$

Error * Z (Gain, how sensitive it is to changes on weights and x)

$w_m < -- w_m + \delta \frac{dL}{dw_k}$

$[x^k, y^k]$

We do this in batches (0 –> N) or epochs.

Why batches

Control was a smooth concave shape, but in supervised learning there are lots of local minima which could trap our response.

$$
\begin{array}{ccccc}
a_k^0 = x_k & O & O & O & x_n \\
O & O & O & O & O \\
O & O & O & O & O \\
O & O & O & O & O \\
O & O & O & O & O \\
a_1^L & O & O & O & O
\end{array}
$$

L + 1 layers last layer is y middle layers are called hidden layers

$a_i^l = f(\sum_k w_{ik}^l a_k^{l-i} + b_i)$ value of 1 neuron at layer l at position i

$a_i^0 = x_i = \text{top}$

$y_i = a_i^L = \text{output}$

$J = \frac{1}{2} \sum (a_i^L - y_i)^2$

$\frac{dJ}{d\theta} = (a_i^L - y_i) \frac{da_i^L}{d\theta}$

$a_i^L = f(w^T)$

$\frac{da_i^L}{d\theta} = f'(z_i^L) * \frac{\sum w_{ik}^l a_k^{l-i} + b_i^l}{d\theta}$

$$
\theta = \begin{array}{cc}
w_{ik}^l & a_k^{l-1} \\
b_i^l & 1 \\
w^{l-1} & \frac{da_k^{l-1}}{du}
\end{array}
$$

$\delta_i^L = (y_i - a_i^l) f'(z_i^l)$

$\delta_i^l = f'(z_i^l) \sum w_{ik}^l \delta_k^{l-1}$

Lecture 29

Neural Networks

Derived back propagation last class

$f : R --> R$

$NT(x) - f(x) < \epsilon$

To prove this:

$f \cap \sum_{k=0}^{N} \alpha_k x^k$

(x^k, y^k) split into training and testing (60/40)

Initialize with random weights

Confusion table:

	1	0
$50C1$	45	5
$50C2$	5	30

45 C1 correct, 30 C2 correct, 5 incorrect C2, 5 incorrect C1.

---

sin(2t) sin(3t)

Need to make data that is not highly correlated

Data -> Feature Extraction -> Classify (ANN, SVM)

Deep learning does not need feature extraction

For CNN you create a 2D kernel which you run over the data to identify features. The amount you move over is called the stride.

Too many kernels causes an explosion and there is a lot of redundant data that needs to be removed.

You can use polling to reduce the amount of neurons you need to train.

Autoencoder

Encoder with a small connection to the decoder.

This allows you to direct the training and create meaningful features

Lecture 30

You start with a smaller input and when you feature extraction the information you have blows up in size. Afterwards you classify

Usually you need to do some enhancing before running over data

convolution -> reduction (polling, reduction layers)

Recurrent neural networks carry output neurons to the next generation for training

(x^k, y^k)

$$\begin{matrix} good \\ cylinder \\ etc \end{matrix}$$

Worked well in fourier due to crank shaft sensor (hall with one tooth missing)

Lecture 31

Non Neural AI Classify

(x^k, y^k) $y^k = +1 or - 1$

k = 1 . . . K

Data –> Features –> Decide (CNN)

Support Vector Machines (SVM)

$w^T x^k + b =$ the variance

$(w^T x^k + b)y^k \geq 0$

Margin is the space between the two lines

Decision line, want the largest maximal margin

Maximal Margin Classifier

$W^T x^k + b)y^k = \gamma^k$

Maximize -> MIN γy^k

$||w|| = 1$

$(w^T x^k + b)y^k = \gamma^k$

$(\frac{w^T}{||w||} x^k + b)y^k = \gamma^k$

Gamma = margin

$y^k(w^T x^k + b) \geq 1$

MIN $||w||^2$ w, k

Can be solved via a standard optimizer

Soft Margin

$y^k(w^T x^k + b) \geq 1$

Instead we use

$y^k(w^T x^k + b) = 1 - \sigma_k$

$MIN||w||^2 + \sum \sigma_k$

$\sigma_k \geq 0$

Allows for exceptions

---

k = number of samples

n = dimension

N »> k

We only care about the points on the line. Support vectors are the only thing that matter.

$w = \sum a_m y^m x^m$

Most a_k are 0

$y^k(w^T x^k + b) = ...$

$y^k(\sum a_m y^m x^{mT} x^k + b)$

Data –> Maximize the variance –> classify

Principle Component Analysis

$\sum_{k=1}^{K} ||x^k - VV^T x^k||$

Lecture 32

Intelligent Systems

Data from sensors -> features -> classify / regression / tracking

Signal processing

Feature extraction

Reduction (PCA)

$\sum_{k=1}^{K} ||x^k - V_m V_m^T x^k||$

$V_1 ... V_N$ Components

Support Vector Machines

$()w^T x^k + b)y^k = /gamma^k$

$y^k = +1/-1$

Clustering

(x^k) —> unsupervided

Picking groups of things

2^m

K Mean Clustering

Need to know how many you want

1. Pick random k points –> centres
2. For each point assign to cluster by distance
3. Recompute centers
4. Go to 2

Best criteria?

$I(c)$ information contained in cluster C

Hierarchical Clustering

All pairs All triples All ...

Entropy = $\sum p_k - log(p_k)$

Data Association

Hungarian algorithm

"Effectively"

Association

x^k –> l

X_{kl}

$J = \sum_k \sum_i X_{kl}||x^i - u_k||$

$\frac{dJ}{du_k} = \frac{1}{2} \sum X_{ki}||x^i - u_k|| - 1 = 0$

$u_k = \frac{\sum_i X_{ki} x^i}{\sum_i X_{ki}}$

Lecture 33

Linear Least Squares (LLS)

(x^k, y^k) –> some data

This is a regression problem

$\phi_\theta(x^k) = y^k$

Least Square

$ARGMIN\ \theta\ \sum_{k=1}^{K} ||y_\theta(x^k) - y^k||^2$

Linear

$\phi_\theta(x) = \sum_{m=0}^{N} \theta_m \phi_m(x)$

$\phi_m(x)$ = basis functions –> orthogonal family (1, x, x^2 ...)

How to solve the problem

Matrix

$$\begin{bmatrix} \phi_0(x^1) & ... & \phi_N(x^1) \\ ... & ... & ... \\ \phi_0(x^k) & ... & \phi_N(x^k) \end{bmatrix} \begin{bmatrix} \theta_0 \\ ... \\ \theta_N \end{bmatrix} = \begin{bmatrix} y^1 \\ ... \\ y^k \end{bmatrix}$$

$\phi_k = x^k$ –> 1, x, x^2 ...

$A\phi = y$

- If matrix is square you can solve
- More variables (rows) than equations (columns), no unique solution

$A^T A\theta = A^T y$

$\theta = (A^T A)^{-1} A^T y$

Perfect mathematically but not numerically. QR is used on the computer

Model Fitting Example Don't just pick the maximum value. Fit points to a model. For one unique point just use $ax^2 + bx + c$

Gradient Descent $ARGMIN\ \theta\ \sum_{k=1}^{K} ||y_\theta(x^k) - y^k||^2$

$f(w^T x + b) = \phi$

$\frac{dL}{d\theta} = 2 \sum ||\theta_\phi(x) - y'|| * \phi'_\theta(x)$

$\theta_{m+1} = \theta_m + \gamma \frac{dL}{d\theta}$

Steepest Descent * $\frac{dL}{d\theta}$ = gradient * γ = learning rate

Maximum Likelihood LLS

$P(y|x)$

Want to estimate the probability distribution and not the function.

(x^k, y^k)

$MAX\pi_{k=1}^{K} P(y^k|x^k)$

$P(y|x) = N(\mu, \sigma^2)$

- N = normal or gauss
- u = expected value
- σ = variance
- σ^2 = standard deviation

$\frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(y-\mu)^2}{\sigma^2}}$

Statistics

A = is a set u = universe

$P(A) = \frac{|A|}{|u|}$

Random Variable

Discrete u = [a, b, c, d] –> equadistant

You pull samples from u –> $\bar{x}$

$P_{\bar{x}}(x) = \frac{1}{4}$

u = [a, a, b]

$P_{\bar{x}}(x = a) = \frac{2}{3}$ $P_{\bar{x}}(x = b) = \frac{1}{3}$

Continous $\bar{x} \varepsilon R$

Cannot do x==0, have to do $abs(x) < \epsilon$

$P_{\bar{x}}(\alpha \leq x \leq \beta) = \int_{\alpha}^{\beta} x P_{\bar{x}}(x) dx$

P = PDF

$\int_{-\inf}^{\inf} P_{\bar{x}}(x) dx = 1$

$\int x P(x) dx = E(x)$ –> 1st momentum

$E(x) = 0$

$\int x^2 P(x) dx = VAR(x)$ –> 2nd momentum

$E((x - E(x))^2)$

CO-VAR(x,y)

$E((x - E(x)) (y- E(y)))

$\bar{x} \varepsilon R^m$

$x = \begin{bmatrix} x_1 \\ x_2 \\ ... \\ x_N \end{bmatrix}$

From previous midterm $\quad P_{\bar{x}}(x) = P_{x_1}(x) * P_{x_3}(x) * P_{x_3}(x)$

$$\begin{bmatrix} x_1 \\ x_2 \\ ... \\ x_N \end{bmatrix} \begin{bmatrix} \sigma_{1,1}^2 & \sigma_{1,2}^2 & ... & \sigma_{1,N}^2 \\ ... & ... & ... & ... \\ \sigma_{N,1}^2 & \sigma_{N,2}^2 & ... & \sigma_{N,N}^2 \end{bmatrix}$$

$\sigma_{i,j} = COVAR(x_i, x_j)$

Lecture 34

Fusion

$\bar{x}$ $\bar{y}$

Can be correlated or uncorrelated based on CO-VAR(x,y)

Assume that x and y are independent observations of the same thing

Assume that the noise is uncorrelated

We work in a gaussian world and everything is gaussian distributed

$hx + (1 - h)y$

Linear mix

Min var

$\frac{\sigma_x^2}{\sigma_x^2 + \sigma_y^2}$

x, y = gauss

x + y = guass

x * y = gauss

Kalman Example

Plant = Ax + Bu y = Cx

u = input

y = output

u feeds into $A\widetilde{x} + Bu$ which outputs x tilde

x tilde feeds into $C\widetilde{x}$ which outputs y tilde

y - y tilde gets fed through a gain L (error feedback)

L feed back into $A\widetilde{x} + Bu$

x tilde goes to -K and feeds back to before u (control)

Can precompute Ks going backwards

17:53 in lecture 34 for diagram

Optimal (Reinforcement)

Markov chains

$P(x_{m+1}|x_m)$ –> f (only dependent on previous state)

$P(y_m|x_m)$ –> h (only dependent on current state)

Use Cost or Reward

$J = \int_0^t COST(x,u)dt$ have used $X^TQx + u^TRU$

x' = f(x,u)

s_i = state

a_i = action

state = x, action = u

$P^a_{ss'}$ = probability something happens

$R^a_{ss'}$ = reward for taking an action

γ = discount, bigger = care more about future, and small = care less

V = expected reward

$V(s) = \sum_a \sum_{s'} P^a_{ss'}(R^a_{ss'} + \gamma V(s'))$

Potential in a state is all possible actions and all states from these actions which is the probability of an action times the reward for that action plus the future discounted reward

Policy iteration example You are given a diagram with set rewards for certain locations.

$R^a_{ss'}$ = -1 or 100 if s' = qf

$V^{m+1}(s) = MIN$ u $(COST(s,u) + \gamma V^m(f(x,u)))$

$x_{m+1} = Ax_m + Bu_m$

$x^TQx + u^TRu + x^TQx$

Bayesian

A = set B = set u = universe

Problem with bayesian is you need to know about the universe

$P(B|A) = \frac{P(A \cap B)}{P(A)}$

$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$

If you compare B_1, B_2, B_3 you do not need to normalize

$x_0 = [0.1, 0.1, ...0.1]$ for 10

$P(G|X) = [0, 0, 0.3, 0.3, 0, 0, 0.3]$ for 10