CS 307. Logistic Regression, Kalman Filtering

Today’s Goal

- Recall of Naïve Bayes Classifier
  - Variable independence assumption
  - Test is straightforward
  - Performance competitive to most of state-of-the-art classifiers even in presence of violating independence assumption
- Logistic regression
- Kalman filtering

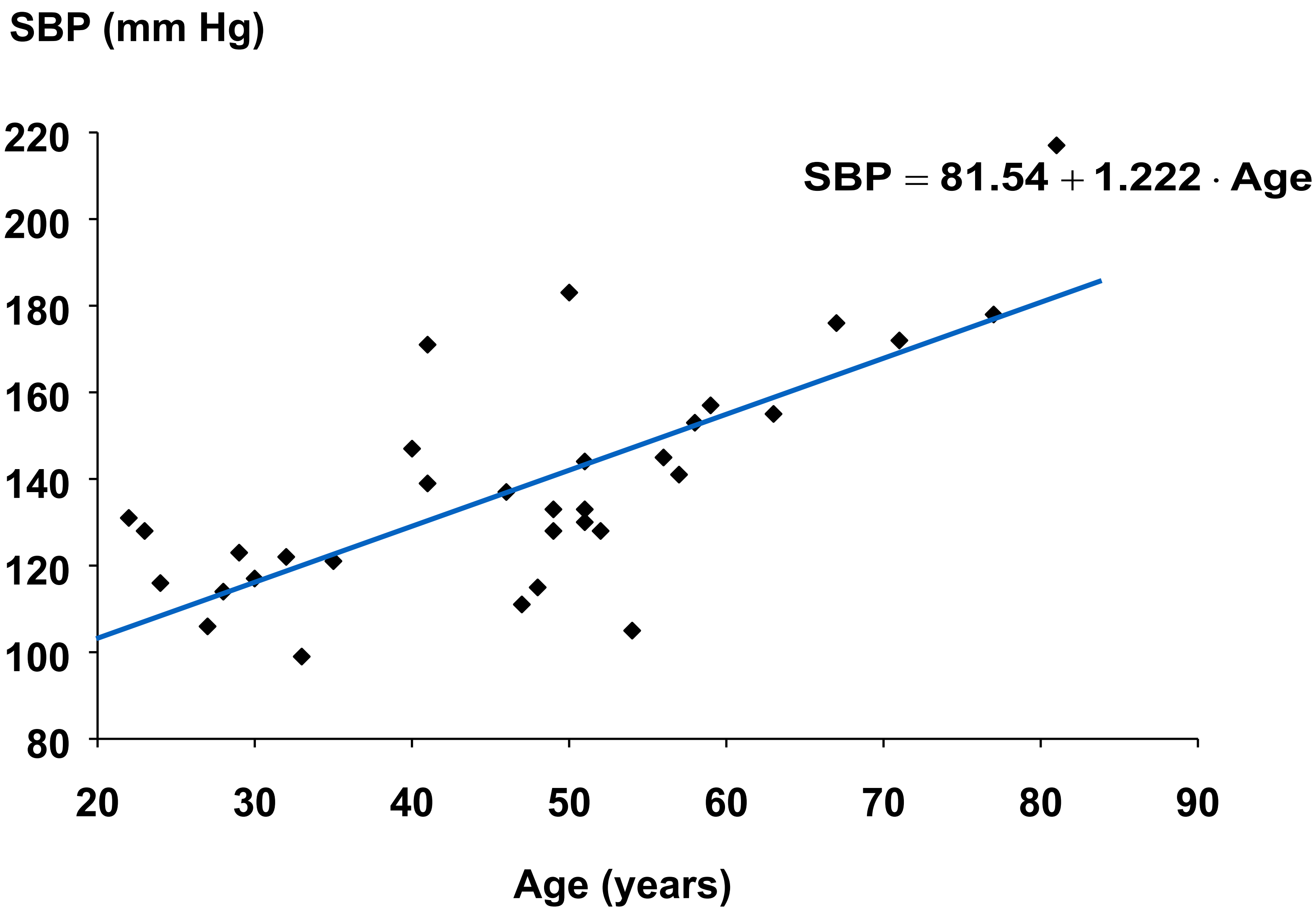
Multivariate analysis

- Machine learning models
  - Linear regression
  - Logistic regression
  - Poisson regression
  - Loglinear model
  - Discriminant analysis
  - ......
- Choice of the tool according to the objectives, the study, and the variables

Simple linear regression

Table 1 Age and systolic blood pressure (SBP) among 33 adult women

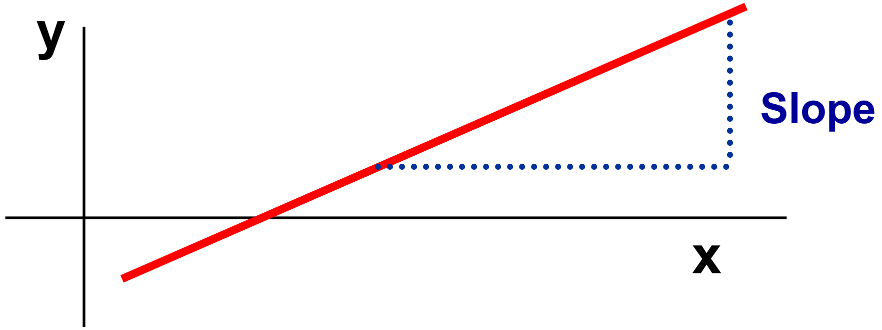
Age	SBP	Age	SBP	Age	SBP
22	131	41	139	52	128
23	128	41	171	54	105
24	116	46	137	56	145
27	106	47	111	57	141
28	114	48	115	58	153
29	123	49	133	59	157
30	117	49	128	63	155
32	122	50	183	67	176
33	99	51	130	71	172
35	121	51	133	77	178
40	147	51	144	81	217



adapted from Colton T. Statistics in Medicine. Boston: Little Brown, 1974

Simple linear regression

- Relation between 2 continuous variables (SBP and age)



$$y = \alpha + \beta_1 x_1$$

- Regression coefficient β_1
  - Measures association between y and x
  - Amount by which y changes on average when x changes by one unit
  - Least squares method

Multiple linear regression

- Relation between a continuous variable and a set of i continuous variables

$$\mathbf{y} = \boldsymbol{\alpha} + \boldsymbol{\beta}_1\mathbf{x}_1 + \boldsymbol{\beta}_2\mathbf{x}_2 + \ldots + \boldsymbol{\beta}_i\mathbf{x}_i$$

- Partial regression coefficients β_i
    - Amount by which y changes on average when x_i changes by one unit and all the other x_is remain constant
    - Measures association between x_i and y adjusted for all other x_i

- Example
    - SBP *versus* age, weight, height, etc

Multiple linear regression

$$\mathbf{y} = \alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_i x_i$$

Predicted	Predictor variables
Response variable	Explanatory variables
Outcome variable	Covariables
Dependent	Independent variables

Logistic regression (1)

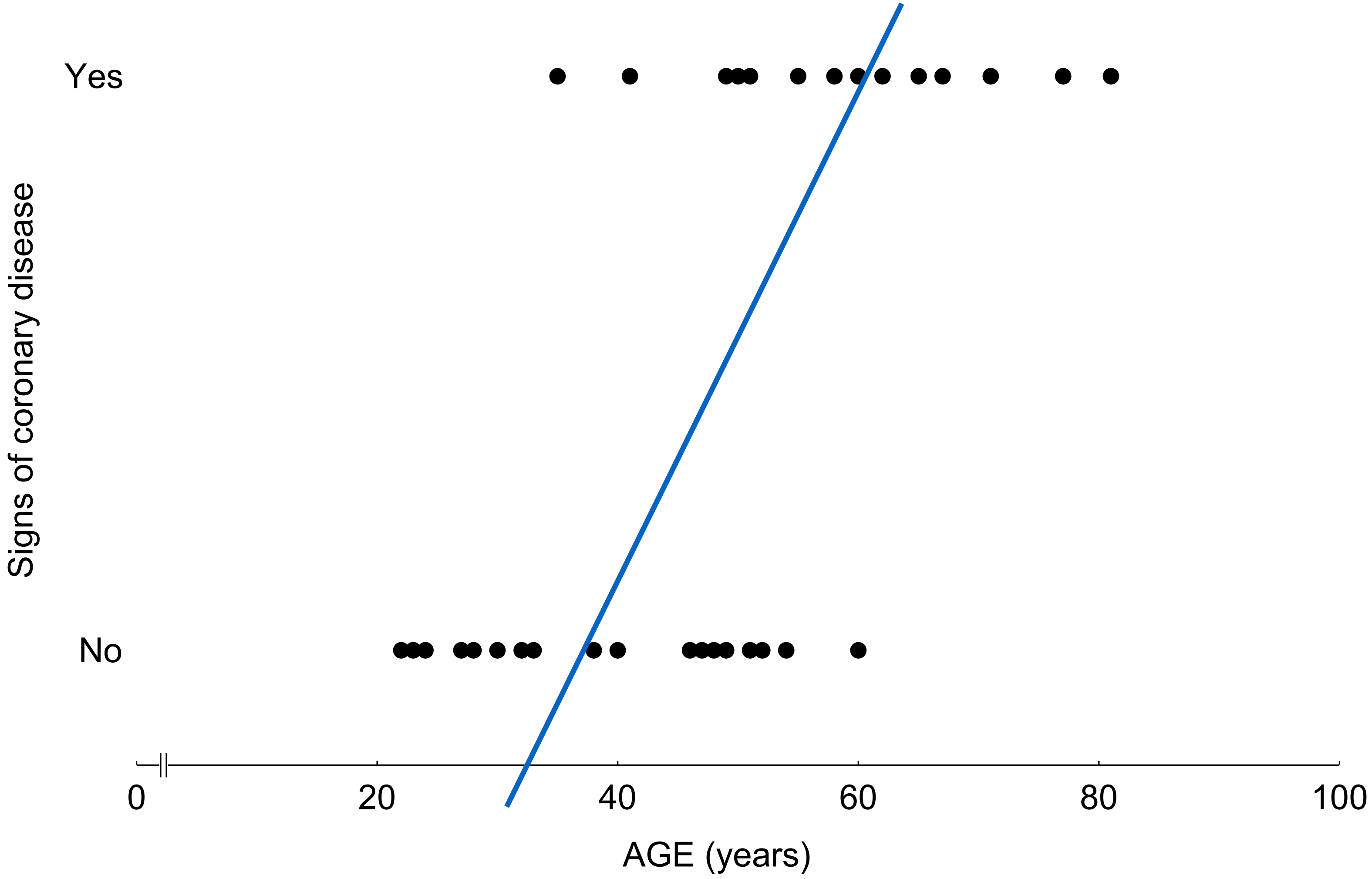
Table 2 Age and signs of coronary heart disease (CD)

Age	CD	Age	CD	Age	CD
22	0	40	0	54	0
23	0	41	1	55	1
24	0	46	0	58	1
27	0	47	0	60	1
28	0	48	0	60	0
30	0	49	1	62	1
30	0	49	0	65	1
32	0	50	1	67	1
33	0	51	0	71	1
35	1	51	1	77	1
38	0	52	0	81	1

How can we analyse these data?

- Compare mean age of diseased and non-diseased
  - Non-diseased: 38.6 years
  - Diseased: 58.7 years ($p<0.0001$)
- Linear regression?

Dot-plot: Data from Table 2

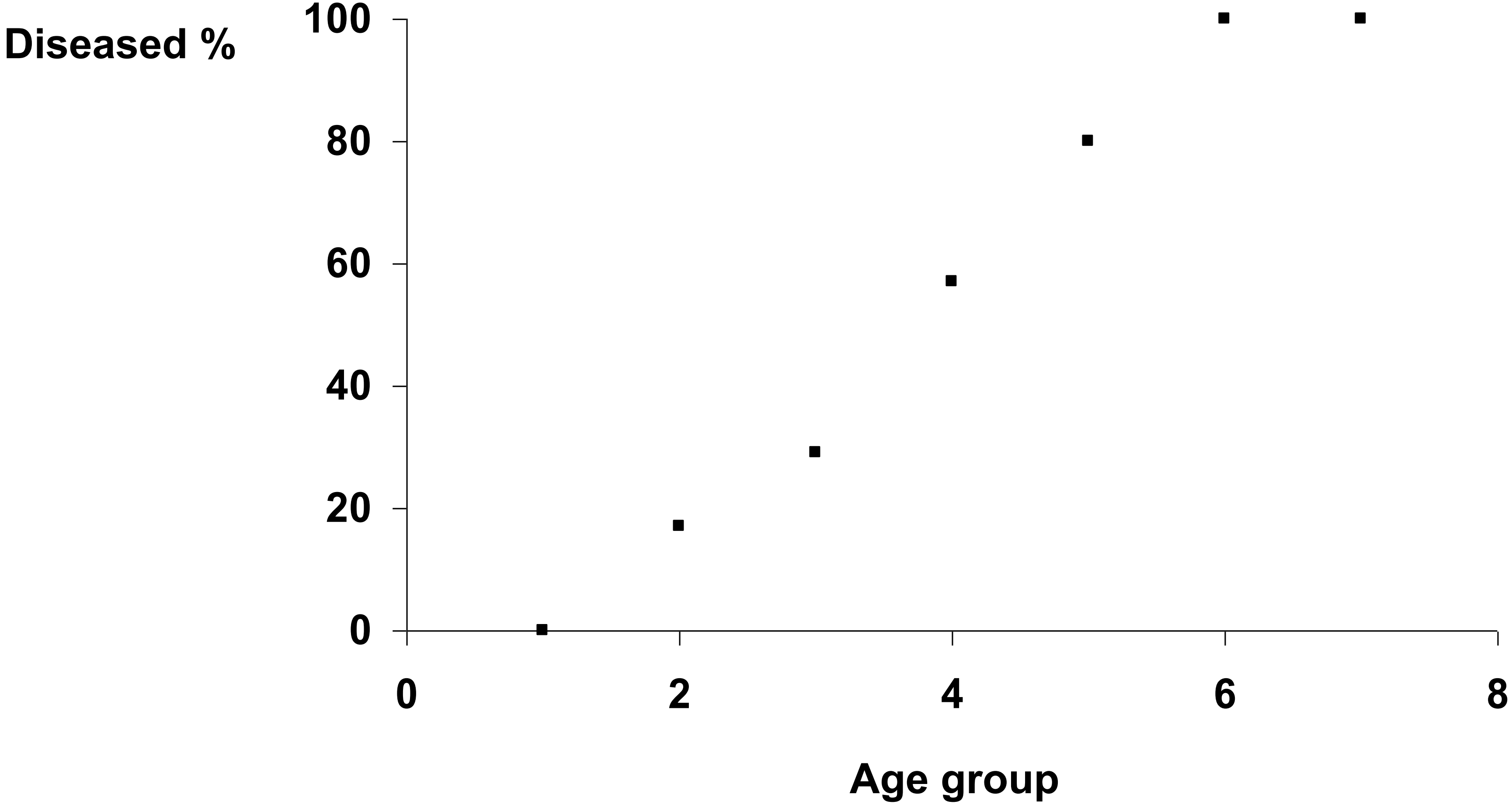


Logistic regression (2)

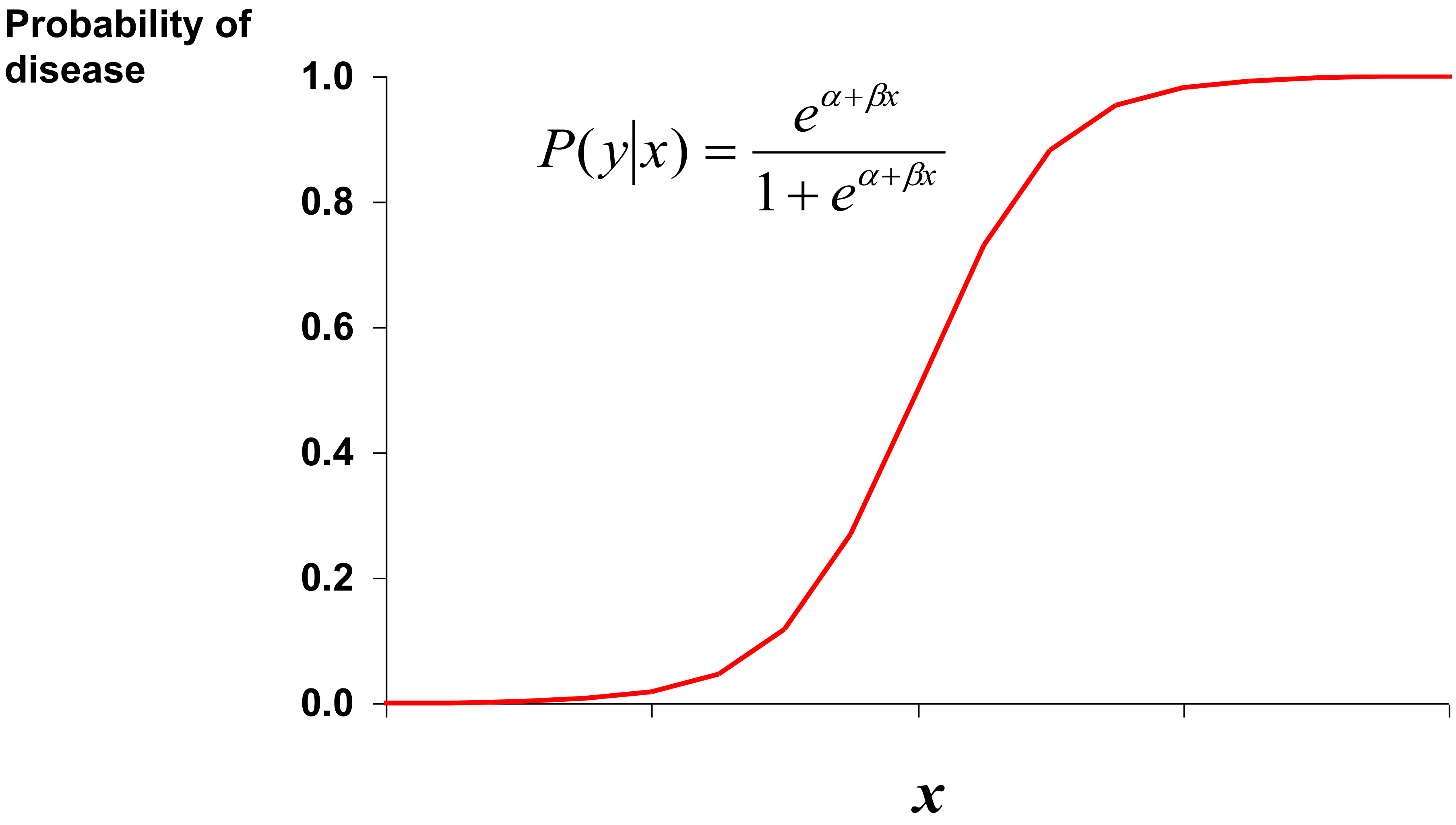
Table 3 Prevalence (%) of signs of CD according to age group

		Diseased	
Age group	**# in group**	**#**	**%**
20 - 29	**5**	**0**	**0**
30 - 39	**6**	**1**	**17**
40 - 49	**7**	**2**	**29**
50 - 59	**7**	**4**	**57**
60 - 69	**5**	**4**	**80**
70 - 79	**2**	**2**	**100**
80 - 89	**1**	**1**	**100**

Dot-plot: Data from Table 3



Logistic function (1)



Transformation

$$P(y|x) = \frac{e^{\alpha+\beta x}}{1+e^{\alpha+\beta x}}$$

$$\frac{P(y|x)}{1-P(y|x)}$$

$$\underbrace{\ln\left[\frac{P(y|x)}{1-P(y|x)}\right]}_{\text{logit of } P(y|x)} = \alpha + \beta x$$

logit of $P(y|x)$

- ✓ **α = log odds of disease in unexposed**
- ✓ **β = log odds ratio associated with being exposed**
- ✓ **e^{β} = odds ratio**

Fitting the data

- Linear regression: Least squares
- Logistic regression: Maximum likelihood
- Likelihood function
  - Estimates parameters α and β
  - Practically easier to work with log-likelihood

$$L(\mathrm{B}) = \ln\left[l(\mathrm{B})\right] = \sum_{i=1}^{n} \left\{ y_i \ln\left[\pi(x_i)\right] + (1 - y_i)\ln\left[1 - \pi(x_i)\right] \right\}$$

Maximum likelihood

- Iterative computing
  - Choice of an arbitrary value for the coefficients (usually 0)
  - Computing of log-likelihood
  - Variation of coefficients’ values
  - Reiteration until maximisation (plateau)

- Results
  - Maximum Likelihood Estimates (MLE) for α and β
  - Estimates of P(y) for a given value of x

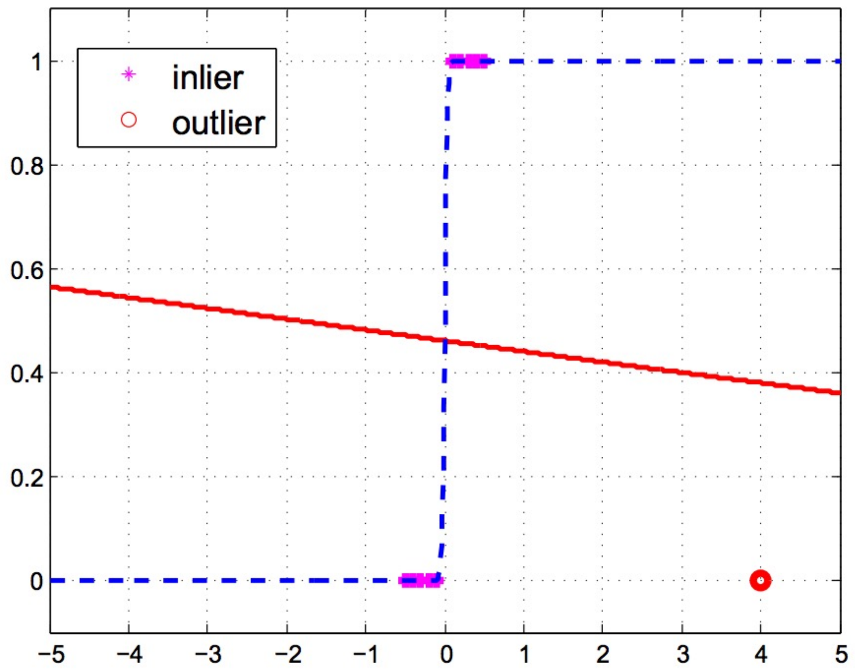
Multiple logistic regression

- More than one independent variable
  - Dichotomous, ordinal, nominal, continuous …

$$\ln\left(\frac{P}{1-P}\right) = \alpha + \beta_1 x_1 + \beta_2 x_2 + \dots \beta_i x_i$$

- Interpretation of β_i
  - Increase in log-odds for a one unit increase in x_i with all the other x_is constant
  - Measures association between x_i and log-odds adjusted for all other x_i

Robust Logistic Regression and Classification



The estimated logistic regression curve (red solid) is far away from the correct one (blue dashed) due to the existence of just one outlier (red circle)

Robust Linear Regression Against Data Poisoning Attack

Main Ideas: a two-phase solution

- Phase 1: Rely on dimension reduction (PCA) to prune non-principal noise in the training data

- Phase 2: In the low-dimensional space, learn a linear model (i.e., PCR)

Main Challenges

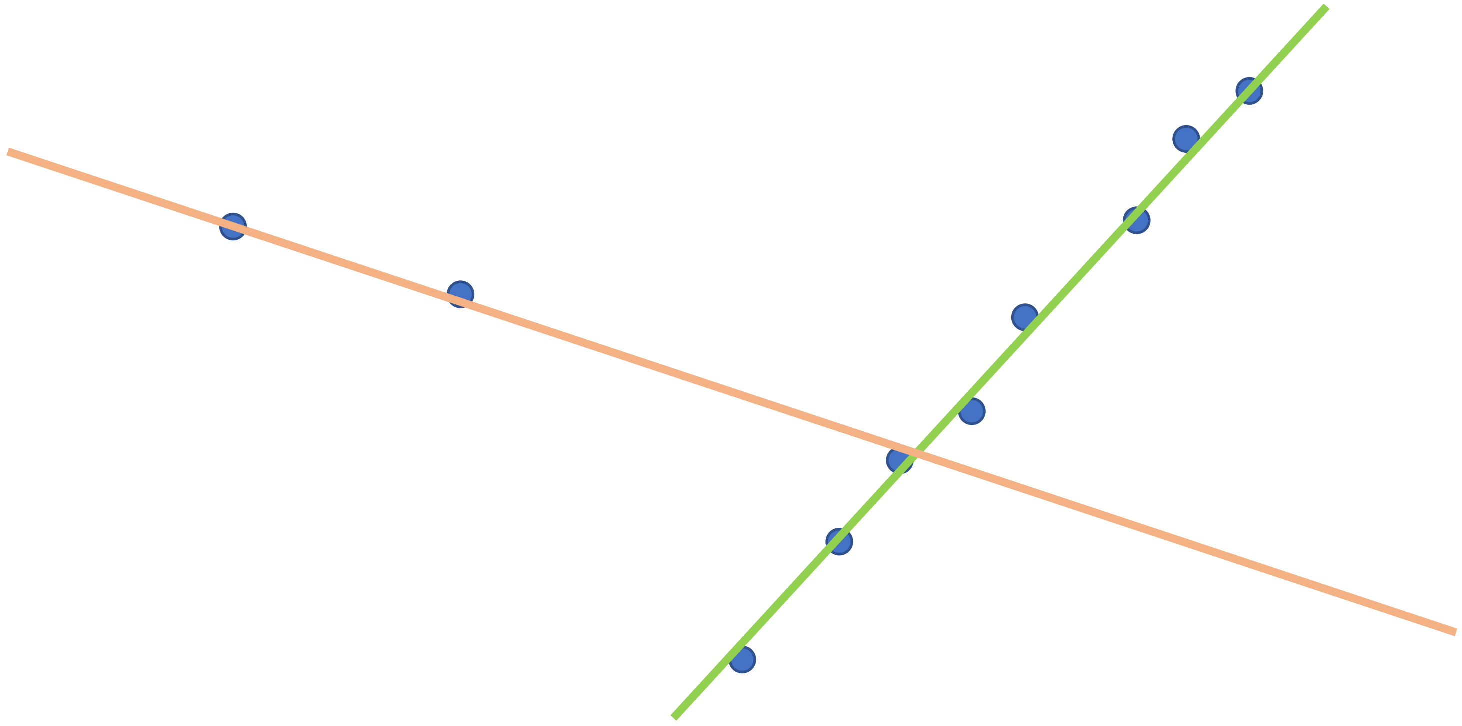
- Both of the **two phases** can be the **target** of the training data poisoning **adversary**

- Have **no assumption** on the ground truth distribution
    - ... except assuming they lie in a low-dimensional manifold

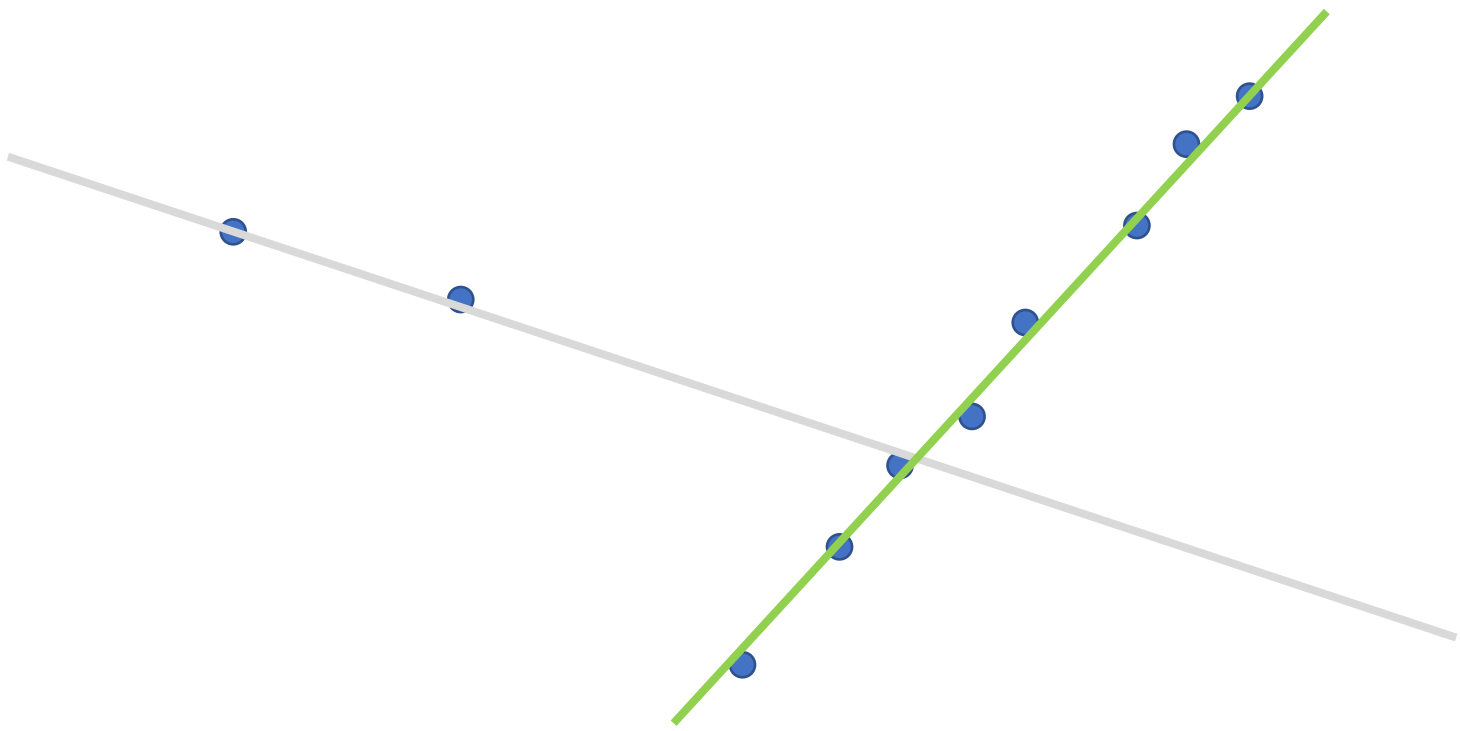
What Can Be Achieved

- Prove a **sufficient and necessary** condition on the **exact sub-space recovery** problem
  - Provides a criteria that the PCA process cannot be poisoned
- A **bound** on the **expected test error** when the training data is poisoned up to γ **poisoning rate**
  - i.e., inject up to γN poisoning samples into the pristine training data of N samples

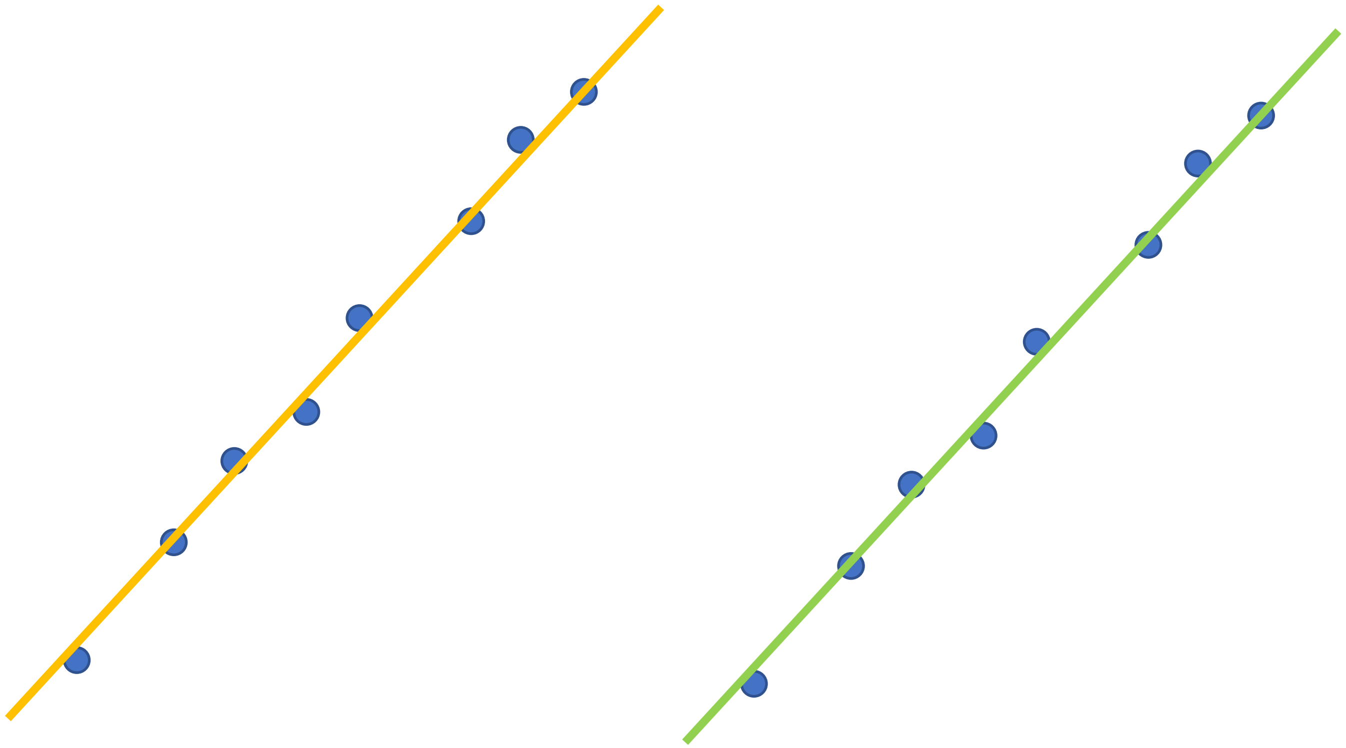
Which line fits the data better?



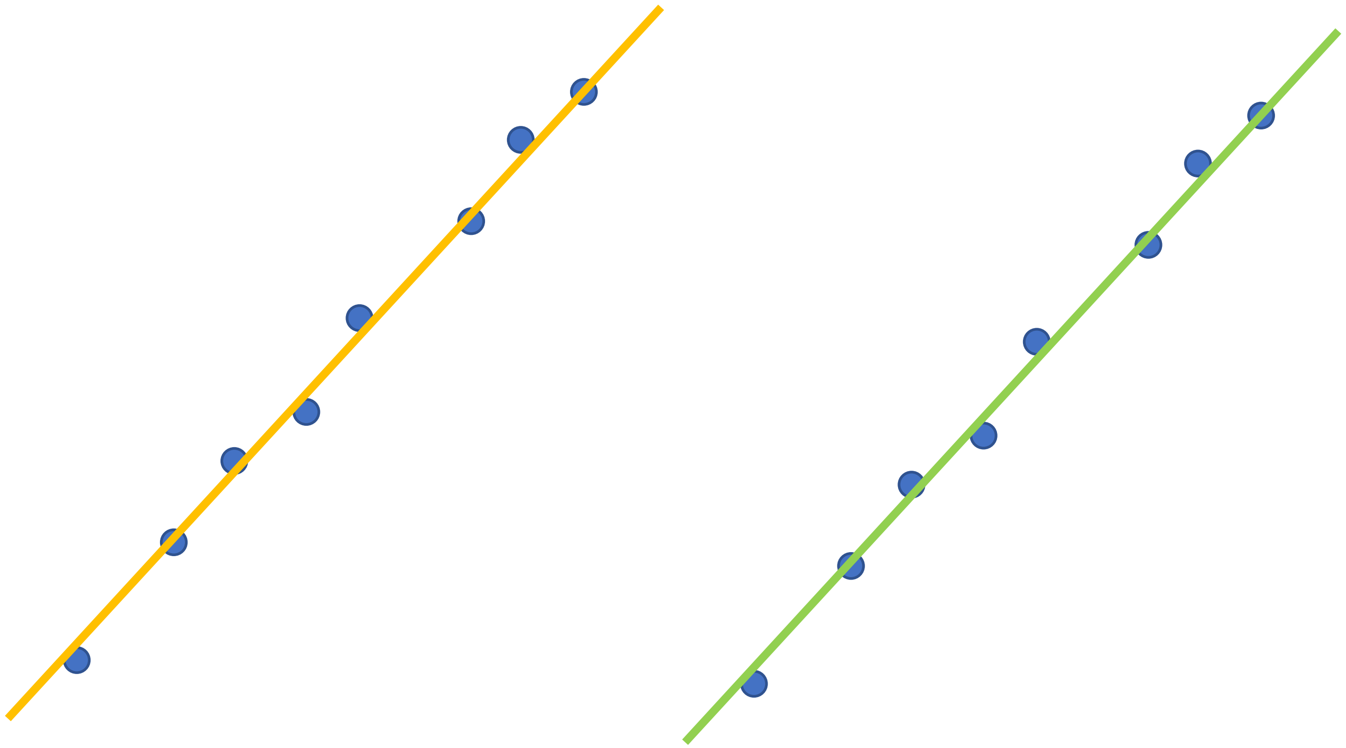
Answer: democracy!



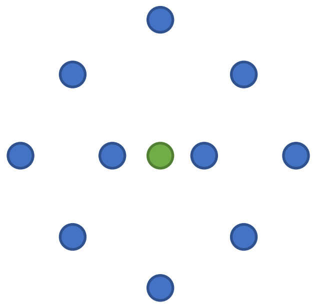
What about now?



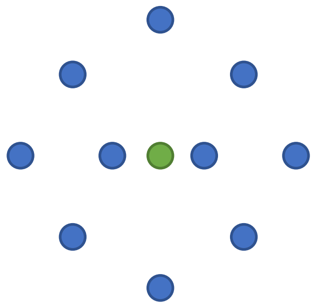
Observation 1: When $\gamma \geq 1$, it is impossible to distinguish the poisoning samples from the pristine ones



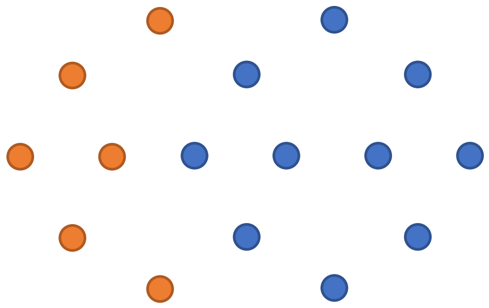
What is the mean of the data distribution?



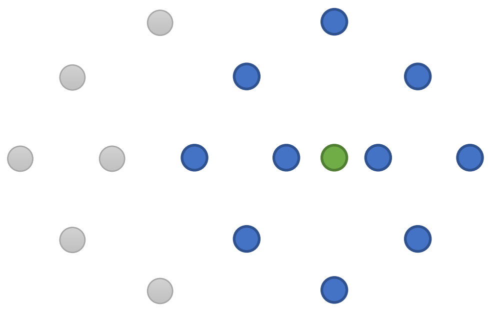
How can a data poisoning adversary efficiently fool the mean estimator?



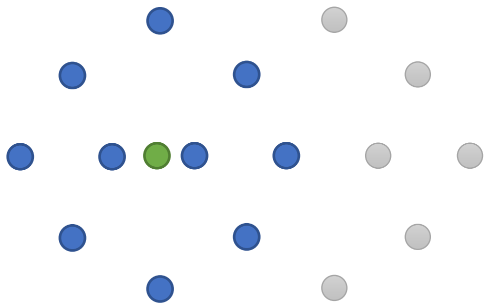
Answer: leveraging the pristine data!



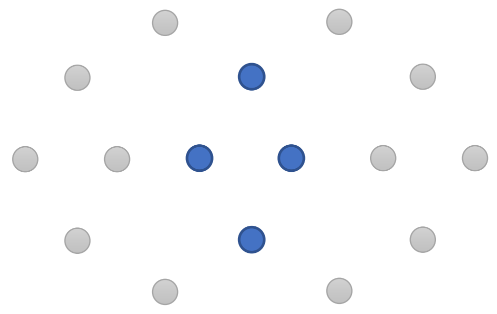
Answer: leveraging the pristine data



Answer: leveraging the pristine data



Observation 2: the data poisoning adversary can fool a machine learning algorithm if and only if there is a portion of the pristine data that he can leverage



Sub-space Recovery Problem

- Problem Definition 1 (Subspace Recovery). Design an algorithm $\mathcal{L}_{recovery}$, which takes as input X, and returns a set of vectors B that form the basis of $X_\star$

- Notation:
  - X: observed (poisoned) feature matrix
  - $X_\star$: the pristine feature matrix
  - X_0: the pristine feature matrix with noise
    - $X_0 = X_\star + noise$

Noise Residual and sub-matrix Residual

- Noise residual $NR(X_0)$ optimizes

$$\min_{X'} ||X_0 - X'||$$
$$\text{s.t.}\ \text{rank}(X') \leq k$$

- Sub-matrix residual $SR(X_0)$ optimizes

$$\min_{I,B,U} \left|\left|X_0^I - U\bar{B}\right|\right|$$
$$\text{s.t.}\ \text{rank}(\bar{B}) = k, \bar{B}\bar{B}^T = I_k, X_\star \bar{B}^T \bar{B} \neq X_\star$$
$$I \subseteq \{1,2,\dots,n\}, |I| = (1-\gamma)N$$

Sufficient and Necessary Condition

- Theorem. If $SR(X_0) \leq NR(X_0)$, then no algorithm solves problem 1 with a probability greater than 1/2.
- If $SR(X_0) > NR(X_0)$, then Algorithm 2 solves problem 1.

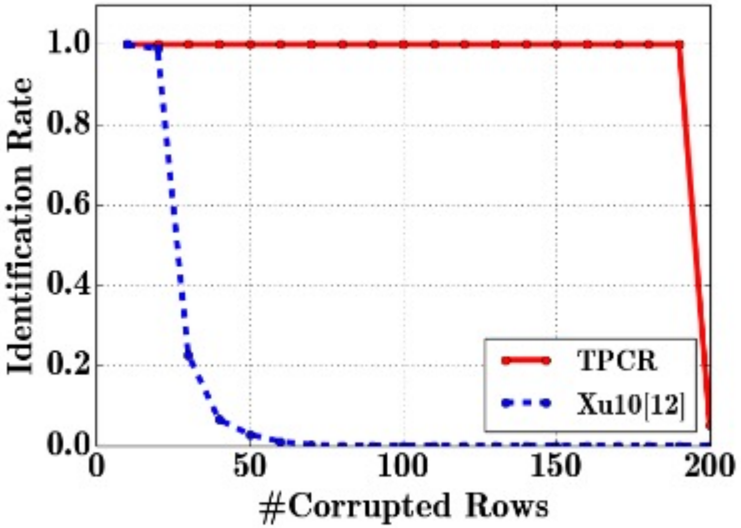
Algorithm 2 Exact recovery algorithm for Problem 1

Solve the following optimization problem and get $\mathscr{I}$.

$$\begin{aligned} &\min_{\mathscr{I}, L} ||\mathbf{X}^{\cdot\mathscr{I}} - L|| \\ &\text{s.t. } \operatorname{rank}(L) \leq k, \mathscr{I} \subseteq \{1, \ldots, n + n_1\}, |\mathscr{I}| = n \end{aligned} \quad (3)$$

return a basis of $\mathbf{X}^{\cdot\mathscr{I}}$.

Sub-space recover experiments (synthetic data)



Takeaways

Message 1. The **poisoning attacker** can **leverage pristine data** distribution to construct strong attacks

Message 2. When the **poisoning ratio** is not sufficiently large, we can **bound the loss** on the computed estimator.

Kalman Filtering

- Follow a point
- Follow a template
- Follow a changing template
- Follow all the elements of a moving person, fit a model to it.
- What are the dynamics of the thing being tracked?
- How is it observed?

$$\hat{X}_{n|n} = E(X_n | Y_1, Y_2, \ldots, Y_n) \;\; , \quad \hat{X}_{n+1|n} = E(X_{n+1} | Y_1, Y_2, \ldots, Y_n)$$

$$\hat{X}_{n|n} \xrightarrow{Prediction} \hat{X}_{n+1|n} \xrightarrow{Correction} \hat{X}_{n+1|n+1}$$

Three main issues in tracking

- **Prediction:** we have seen $\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1}$ — what state does this set of measurements predict for the i'th frame? to solve this problem, we need to obtain a representation of $P(\boldsymbol{X}_i | \boldsymbol{Y}_0 = \boldsymbol{y}_0, \ldots, \boldsymbol{Y}_{i-1} = \boldsymbol{y}_{i-1})$.
- **Data association:** Some of the measurements obtained from the i-th frame may tell us about the object's state. Typically, we use $P(\boldsymbol{X}_i | \boldsymbol{Y}_0 = \boldsymbol{y}_0, \ldots, \boldsymbol{Y}_{i-1} = \boldsymbol{y}_{i-1})$ to identify these measurements.
- **Correction:** now that we have $\boldsymbol{y}_i$ — the relevant measurements — we need to compute a representation of $P(\boldsymbol{X}_i | \boldsymbol{Y}_0 = \boldsymbol{y}_0, \ldots, \boldsymbol{Y}_i = \boldsymbol{y}_i)$.

Simplifying Assumptions

- **Only the immediate past matters:** formally, we require

$$P(\boldsymbol{X}_i|\boldsymbol{X}_1,\ldots,\boldsymbol{X}_{i-1}) = P(\boldsymbol{X}_i|\boldsymbol{X}_{i-1})$$

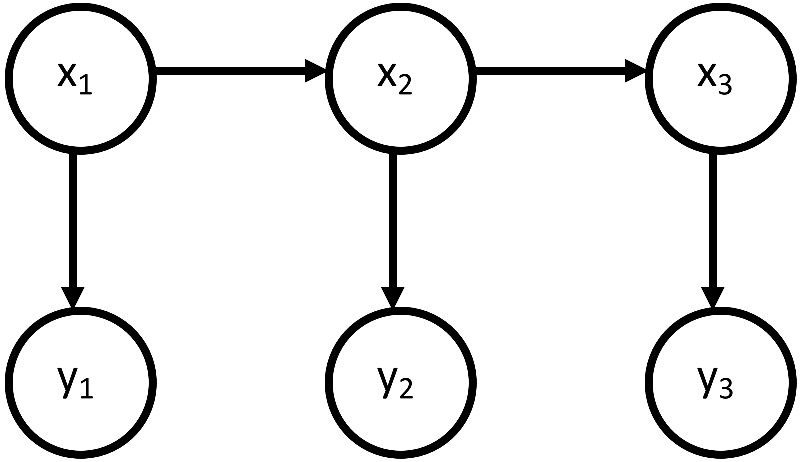
  This assumption hugely simplifies the design of algorithms, as we shall see; furthermore, it isn't terribly restrictive if we're clever about interpreting $\boldsymbol{X}_i$ as we shall show in the next section.

- **Measurements depend only on the current state:** we assume that $\boldsymbol{Y}_i$ is conditionally independent of all other measurements given $\boldsymbol{X}_i$. This means that

$$P(\boldsymbol{Y}_i,\boldsymbol{Y}_j,\ldots\boldsymbol{Y}_k|\boldsymbol{X}_i) = P(\boldsymbol{Y}_i|\boldsymbol{X}_i)P(\boldsymbol{Y}_j,\ldots,\boldsymbol{Y}_k|\boldsymbol{X}_i)$$

  Again, this isn't a particularly restrictive or controversial assumption, but it yields important simplifications.

Kalman filter graphical model and corresponding factorized joint probability



$$P(x_1, x_2, x_3, y_1, y_2, y_3) =$$
$$P(x_1)P(y_1 \mid x_1)P(x_2 \mid x_1)P(y_2 \mid x_2)P(x_3 \mid x_2)P(y_3 \mid x_3)$$

Tracking as induction

- Make a measurement starting in the 0^{th} frame
- Then: assume you have an estimate at the ith frame, after the measurement step.
- Show that you can do prediction for the i+1th frame, and measurement for the i+1th frame.

Prediction step

Prediction

Prediction involves representing

$$P(\boldsymbol{X}_i | \boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1})$$

given

$$P(\boldsymbol{X}_{i-1} | \boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1}).$$

Our independence assumptions make it possible to write

$$\begin{aligned}
P(\boldsymbol{X}_i | \boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1}) &= \int P(\boldsymbol{X}_i, \boldsymbol{X}_{i-1} | \boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1}) d\boldsymbol{X}_{i-1} \\
&= \int P(\boldsymbol{X}_i | \boldsymbol{X}_{i-1}, \boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1}) P(\boldsymbol{X}_{i-1} | \boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1}) d\boldsymbol{X}_{i-1} \\
&= \int P(\boldsymbol{X}_i | \boldsymbol{X}_{i-1}) P(\boldsymbol{X}_{i-1} | \boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1}) d\boldsymbol{X}_{i-1}
\end{aligned}$$

Correction step

Correction

Correction involves obtaining a representation of

$$P(\boldsymbol{X}_i|\boldsymbol{y}_0, \ldots, \boldsymbol{y}_i)$$

given

$$P(\boldsymbol{X}_i|\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1})$$

Our independence assumptions make it possible to write

$$\begin{aligned}
P(\boldsymbol{X}_i|\boldsymbol{y}_0, \ldots, \boldsymbol{y}_i) &= \frac{P(\boldsymbol{X}_i, \boldsymbol{y}_0, \ldots, \boldsymbol{y}_i)}{P(\boldsymbol{y}_0, \ldots, \boldsymbol{y}_i)} \\
&= \frac{P(\boldsymbol{y}_i|\boldsymbol{X}_i, \boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1})P(\boldsymbol{X}_i|\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1})P(\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1})}{P(\boldsymbol{y}_0, \ldots, \boldsymbol{y}_i)} \\
&= P(\boldsymbol{y}_i|\boldsymbol{X}_i)P(\boldsymbol{X}_i|\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1})\frac{P(\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1})}{P(\boldsymbol{y}_0, \ldots, \boldsymbol{y}_i)} \\
&= \frac{P(\boldsymbol{y}_i|\boldsymbol{X}_i)P(\boldsymbol{X}_i|\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1})}{\int P(\boldsymbol{y}_i|\boldsymbol{X}_i)P(\boldsymbol{X}_i|\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1})d\boldsymbol{X}_i}
\end{aligned}$$

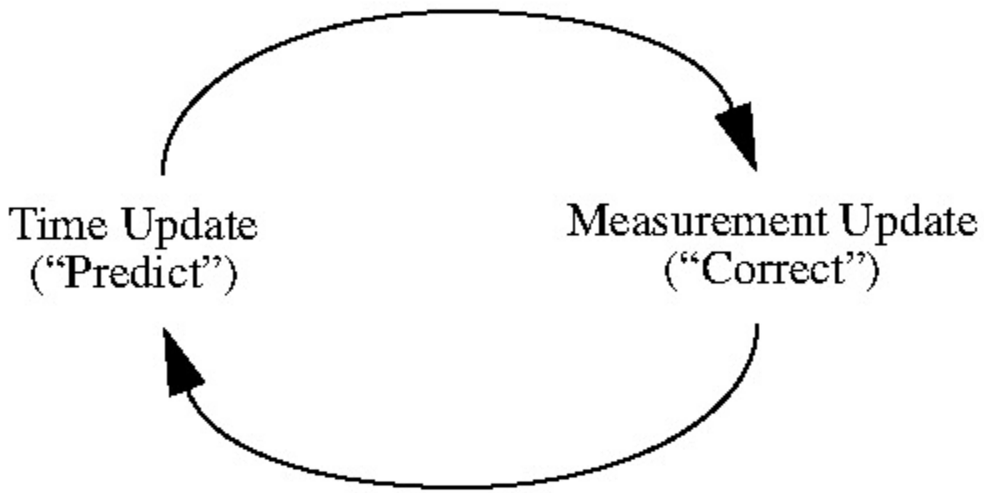
The Kalman Filter

- Key ideas:
  - Linear models interact uniquely well with Gaussian noise - make the prior Gaussian, everything else Gaussian and the calculations are easy
  - Gaussians are really easy to represent --- once you know the mean and covariance, you're done

Recall the three main issues in tracking

- **Prediction:** we have seen $\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1}$ — what state does this set of measurements predict for the i'th frame? to solve this problem, we need to obtain a representation of $P(\boldsymbol{X}_i | \boldsymbol{Y}_0 = \boldsymbol{y}_0, \ldots, \boldsymbol{Y}_{i-1} = \boldsymbol{y}_{i-1})$.
- **Data association:** Some of the measurements obtained from the i-th frame may tell us about the object's state. Typically, we use $P(\boldsymbol{X}_i | \boldsymbol{Y}_0 = \boldsymbol{y}_0, \ldots, \boldsymbol{Y}_{i-1} = \boldsymbol{y}_{i-1})$ to identify these measurements.
- **Correction:** now that we have $\boldsymbol{y}_i$ — the relevant measurements — we need to compute a representation of $P(\boldsymbol{X}_i | \boldsymbol{Y}_0 = \boldsymbol{y}_0, \ldots, \boldsymbol{Y}_i = \boldsymbol{y}_i)$.

The Kalman Filter



Example: The Kalman Filter in 1D

- Dynamic Model

$$x_i \sim N(d_i x_{i-1}, \sigma_{d_i}^2)$$

$$y_i \sim N(m_i x_i, \sigma_{m_i}^2)$$

- Notation

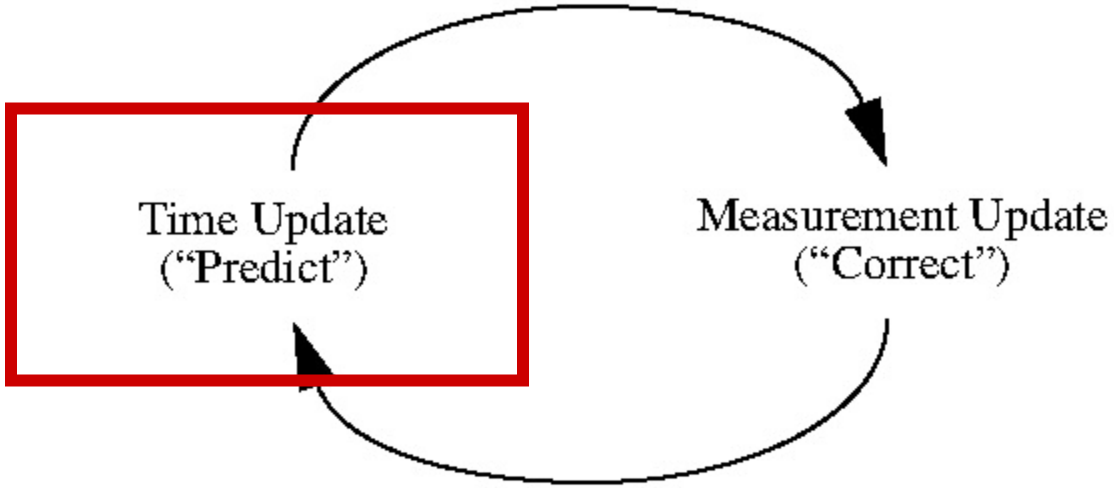
mean of $P(X_i | y_0, \ldots, y_{i-1})$ as $\overline{X}_i^-$ ← Predicted mean

mean of $P(X_i | y_0, \ldots, y_i)$ as $\overline{X}_i^+$ ← Corrected mean

the standard deviation of $P(X_i | y_0, \ldots, y_{i-1})$ as σ_i^-

of $P(X_i | y_0, \ldots, y_i)$ as σ_i^+

The Kalman Filter



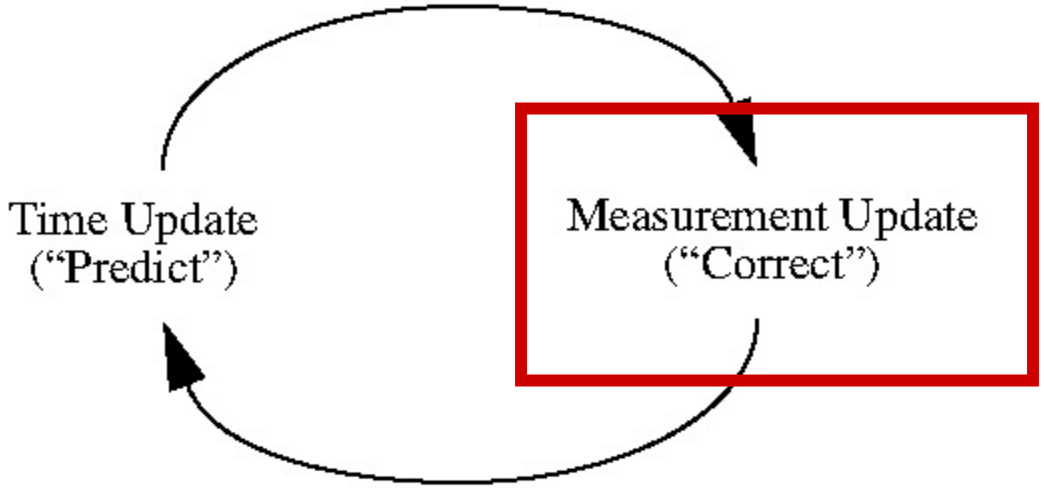
Prediction for 1D Kalman filter

- The new state is obtained by $x_i \sim N(d_i x_{i-1}, \sigma_{d_i}^2)$
  - multiplying old state by known constant
  - adding zero-mean noise

- Therefore, predicted mean for new state is
  - constant times mean for old state
- Old variance is normal random variable
  - variance is multiplied by square of constant
  - and variance of noise is added.

$$\overline{X}_i^- = d_i \overline{X}_{i-1}^+ \qquad (\sigma_i^-)^2 = \sigma_{d_i}^2 + (d_i \sigma_{i-1}^+)^2$$

The Kalman Filter



Measurement update for 1D Kalman filter

$$x_i^+ = \left(\frac{\overline{x}_i^- \sigma_{m_i}^2 + m_i y_i (\sigma_i^-)^2}{\sigma_{m_i}^2 + m_i^2 (\sigma_i^-)^2} \right)$$

$$\sigma_i^+ = \sqrt{\left(\frac{\sigma_{m_i}^2 (\sigma_i^-)^2}{(\sigma_{m_i}^2 + m_i^2 (\sigma_i^-)^2)} \right)}$$

Notice:

- if measurement noise is small, we rely mainly on the measurement;
- if it’s large, mainly on the prediction
- σ does not depend on y

Dynamic Model:

$$x_i \sim N(d_i x_{i-1}, \sigma_{d_i})$$

$$y_i \sim N(m_i x_i, \sigma_{m_i})$$

Start Assumptions: $\overline{x}_0^-$ and σ_0^- are known
Update Equations: Prediction

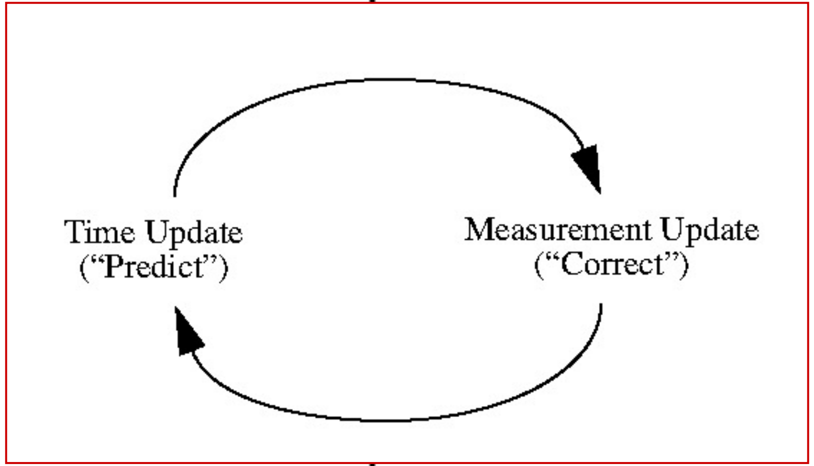
$$\overline{x}_i^- = d_i \overline{x}_{i-1}^+$$

$$\sigma_i^- = \sqrt{\sigma_{d_i}^2 + (d_i \sigma_{i-1}^+)^2}$$

Update Equations: Correction

$$x_i^+ = \left(\frac{\overline{x}_i^- \sigma_{m_i}^2 + m_i y_i (\sigma_i^-)^2}{\sigma_{m_i}^2 + m_i^2 (\sigma_i^-)^2} \right)$$

$$\sigma_i^+ = \sqrt{\left(\frac{\sigma_{m_i}^2 (\sigma_i^-)^2}{(\sigma_{m_i}^2 + m_i^2 (\sigma_i^-)^2)} \right)}$$



What happens if the x dynamics are given a non-zero variance?

Dynamic Model:

$$x_i \sim N(d_i x_{i-1}, \sigma_{d_i})$$

$$y_i \sim N(m_i x_i, \sigma_{m_i})$$

Start Assumptions: $\overline{x}_0^-$ and σ_0^- are known

Update Equations: Prediction

$$\overline{x}_i^- = d_i \overline{x}_{i-1}^+$$

$$\sigma_i^- = \sqrt{\sigma_{d_i}^2 + (d_i \sigma_{i-1}^+)^2}$$

Update Equations: Correction

$$x_i^+ = \left(\frac{\overline{x}_i^- \sigma_{m_i}^2 + m_i y_i (\sigma_i^-)^2}{\sigma_{m_i}^2 + m_i^2 (\sigma_i^-)^2} \right)$$

$$\sigma_i^+ = \sqrt{\left(\frac{\sigma_{m_i}^2 (\sigma_i^-)^2}{(\sigma_{m_i}^2 + m_i^2 (\sigma_i^-)^2)} \right)}$$

Kalman filter model

$$d_i = 1,\ m_i = 1,\ \sigma_{d_i} = 1,\ \sigma_{m_i} = 1$$

Initial conditions

$$\overline{x}_0^- = 0 \qquad \sigma_0^- = \infty$$

Iteration	0	1	2
$\overline{x}_i^-$	0	y_0	$\frac{y_0 + 2y_1}{3}$
$\overline{x}_i^+$	y_0	$\frac{y_0 + 2y_1}{3}$	$\frac{y_0 + 2y_1 + 5y_2}{8}$
σ_i^-	∞	$\sqrt{2}$	$\sqrt{\frac{5}{3}}$
σ_i^+	1	$\sqrt{\frac{2}{3}}$	$\sqrt{\frac{5}{8}}$

Linear dynamic models

- A linear dynamic model has the form

$$\mathbf{x}_i = N\left(\mathbf{D}_{i-1}\mathbf{x}_{i-1}; \Sigma_{d_i}\right)$$

$$\mathbf{y}_i = N\left(\mathbf{M}_i\mathbf{x}_i; \Sigma_{m_i}\right)$$

- This is much, much more general than it looks, and extremely powerful

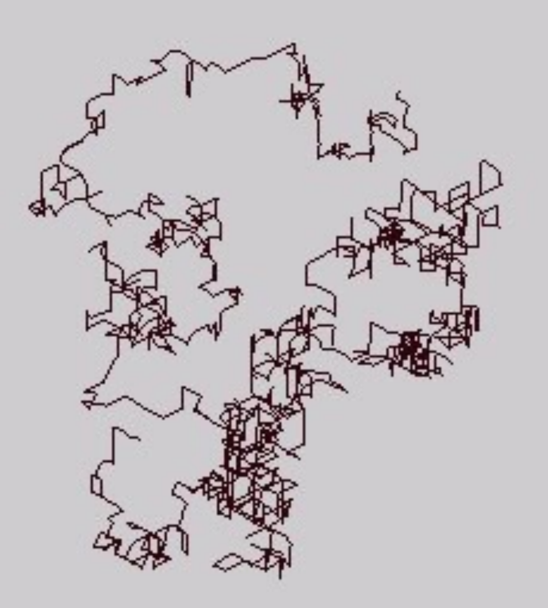
Examples of linear state space models

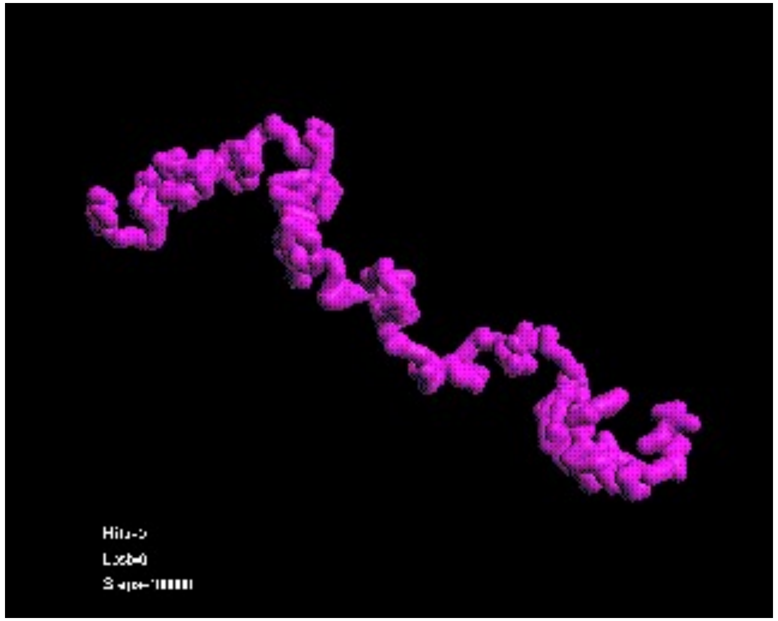
$$\mathbf{x}_i = N\left(\mathbf{D}_{i-1}\mathbf{x}_{i-1};\Sigma_{d_i}\right)$$

$$\mathbf{y}_i = N\left(\mathbf{M}_i\mathbf{x}_i;\Sigma_{m_i}\right)$$

- Drifting points
  - assume that the new position of the point is the old one, plus noise

D = **Identity**





Constant velocity

$$\mathbf{x}_i = N(\mathbf{D}_{i-1}\mathbf{x}_{i-1};\Sigma_{d_i})$$

$$\mathbf{y}_i = N(\mathbf{M}_i\mathbf{x}_i;\Sigma_{m_i})$$

- We have

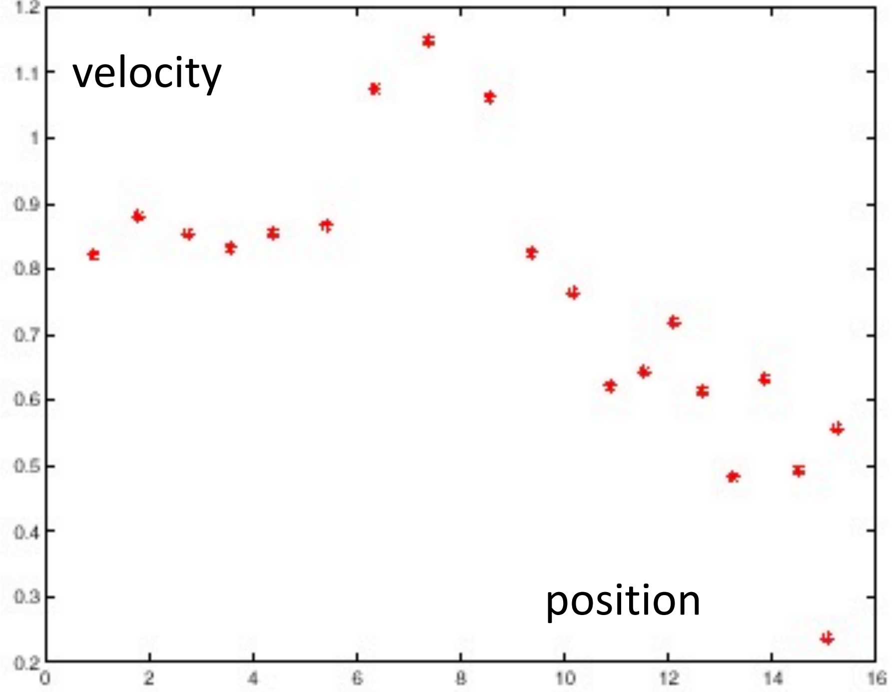
$$u_i = u_{i-1} + \Delta t v_{i-1} + \varepsilon_i$$
$$v_i = v_{i-1} + \varsigma_i$$

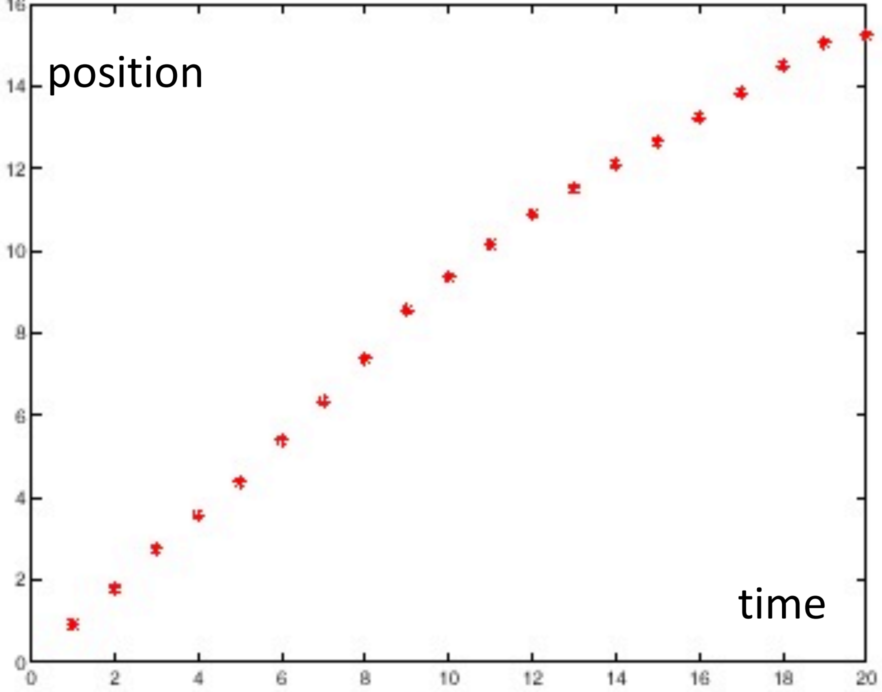
- Stack (u, v) into a single state vector

$$\begin{pmatrix} u \\ v \end{pmatrix}_i = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}_{i-1} + \text{noise}$$

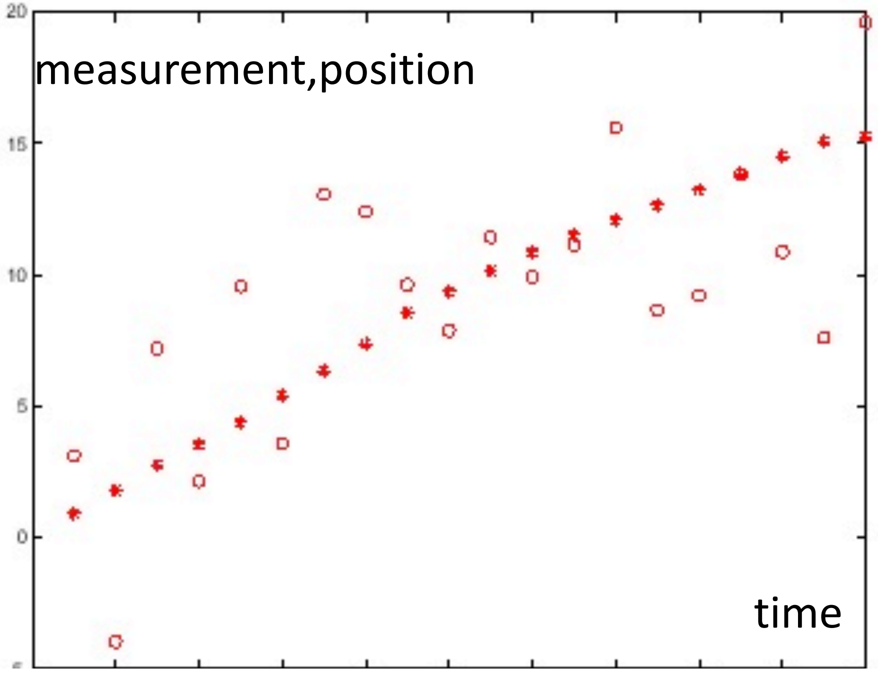
  - which is the form we had above

$\mathbf{x}_i$ **$\mathbf{D}_{i-1}$** **$\mathbf{x}_{i-1}$**





Constant Velocity Model



Constant acceleration

$$\mathbf{x}_i = N\left(\mathbf{D}_{i-1}\mathbf{x}_{i-1}; \Sigma_{d_i}\right)$$

$$\mathbf{y}_i = N\left(\mathbf{M}_i\mathbf{x}_i; \Sigma_{m_i}\right)$$

- We have

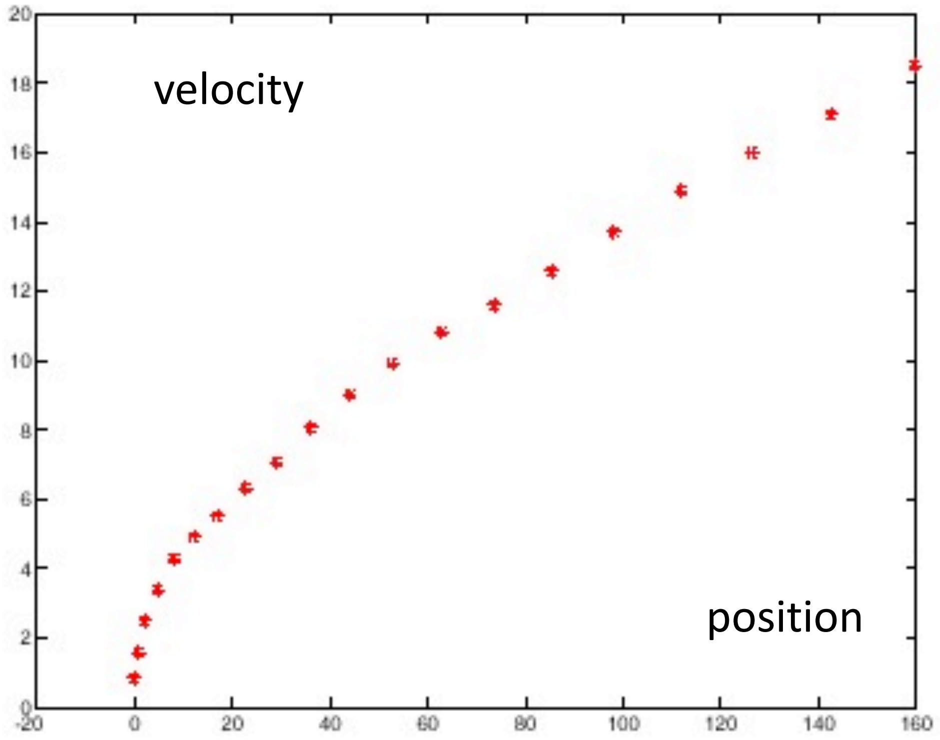
$$u_i = u_{i-1} + \Delta t v_{i-1} + \varepsilon_i$$
$$v_i = v_{i-1} + \Delta t a_{i-1} + \varsigma_i$$
$$a_i = a_{i-1} + \xi_i$$

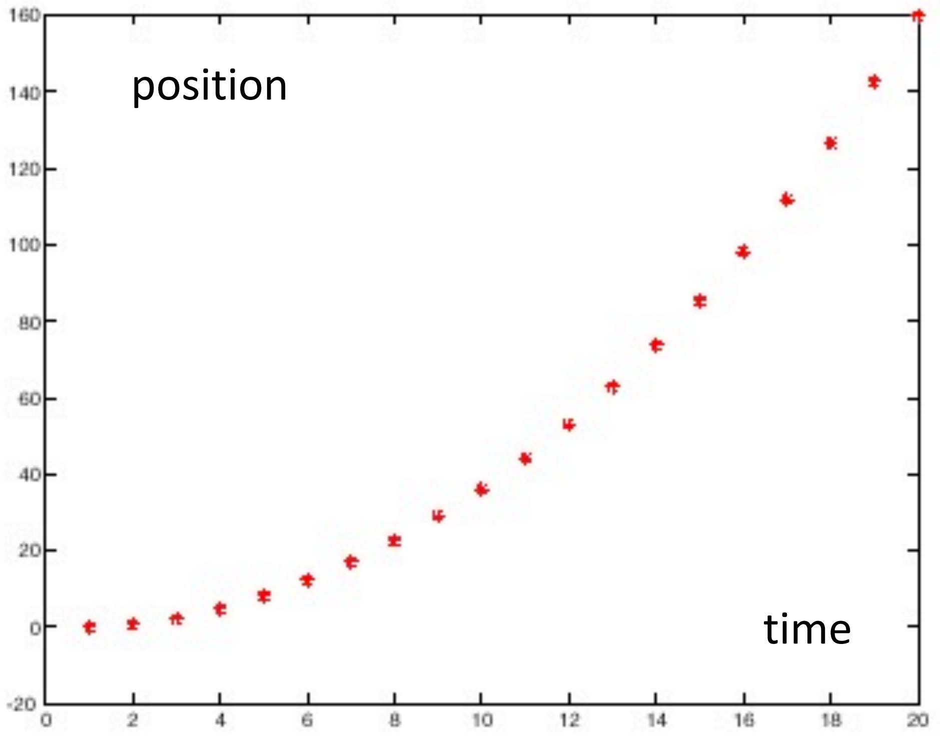
  - (the Greek letters denote noise terms)
- Stack (u, v) into a single state vector

$$\begin{pmatrix} u \\ v \\ a \end{pmatrix}_i = \begin{pmatrix} 1 & \Delta t & 0 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \\ a \end{pmatrix}_{i-1} + \text{noise}$$

  - which is the form we had above

$\mathbf{x}_i$ $\mathbf{D}_{i-1}$ $\mathbf{x}_{i-1}$





Constant Acceleration Model

Periodic motion

$$\mathbf{x}_i = N\left(\mathbf{D}_{i-1}\mathbf{x}_{i-1};\Sigma_{d_i}\right)$$
$$\mathbf{y}_i = N\left(\mathbf{M}_i\mathbf{x}_i;\Sigma_{m_i}\right)$$

Assume we have a point, moving on a line with a periodic movement defined with a differential eq:

$$\frac{d^2p}{dt^2} = -p$$

can be defined as

$$\frac{d\boldsymbol{u}}{dt} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \boldsymbol{u} = \mathcal{S}\boldsymbol{u}$$

with state defined as stacked position and velocity u=(p, v)

Periodic motion

$$\mathbf{x}_i = N\left(\mathbf{D}_{i-1}\mathbf{x}_{i-1};\Sigma_{d_i}\right)$$

$$\mathbf{y}_i = N\left(\mathbf{M}_i\mathbf{x}_i;\Sigma_{m_i}\right)$$

$$\frac{d\boldsymbol{u}}{dt} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \boldsymbol{u} = \mathcal{S}\boldsymbol{u}$$

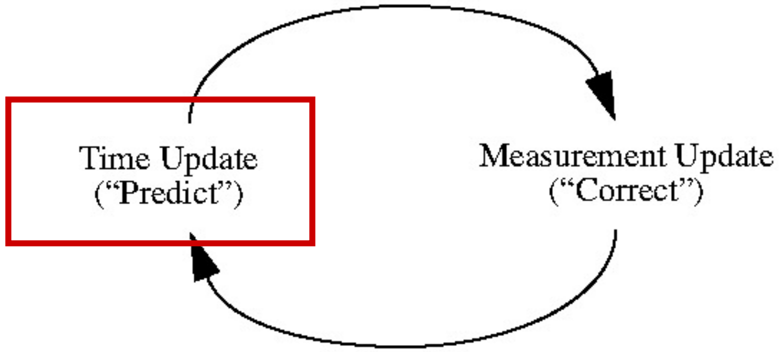
Take discrete approximation….(e.g., forward Euler integration with Δt stepsize.)

$$\begin{aligned} \boldsymbol{u}_i &= \boldsymbol{u}_{i-1} + \Delta t \frac{d\boldsymbol{u}}{dt} \\ &= \boldsymbol{u}_{i-1} + \Delta t \mathcal{S} \boldsymbol{u}_{i-1} \\ &= \begin{pmatrix} 1 & \Delta t \\ -\Delta t & 1 \end{pmatrix} \boldsymbol{u}_{i-1} \end{aligned}$$

$\mathbf{x}_i$ $\mathbf{D}_{i-1}$ $\mathbf{x}_{i-1}$

n-D Prediction

Generalization to n-D is straightforward but more complex.



Prediction:

- Multiply estimate at prior time with forward model:

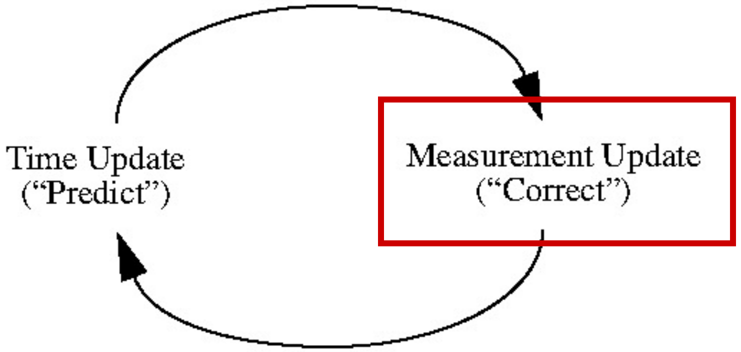
$$\overline{\boldsymbol{x}}_i^- = \mathcal{D}_i \overline{\boldsymbol{x}}_{i-1}^+$$

- Propagate covariance through model and add new noise:

$$\Sigma_i^- = \Sigma_{d_i} + \mathcal{D}_i \sigma_{i-1}^+ \mathcal{D}_i$$

n-D Correction

Generalization to n-D is straightforward but more complex.



Correction:

- Update *a priori* estimate with measurement to form *a posteriori*

n-D correction

Find linear filter on innovations

$$\overline{\boldsymbol{x}}_i^+ = \overline{\boldsymbol{x}}_i^- + \mathcal{K}_i \left[\boldsymbol{y}_i - \mathcal{M}_i \overline{\boldsymbol{x}}_i^- \right]$$

which minimizes *a posteriori* error covariance:

$$E\left[\left(x - \overline{x^+}\right)^T \left(x - \overline{x^+}\right)\right]$$

K is the *Kalman Gain* matrix. A solution is

$$\mathcal{K}_i = \Sigma_i^- \mathcal{M}_i^T \left[\mathcal{M}_i \Sigma_i^- \mathcal{M}_i^T + \Sigma_{m_i} \right]^{-1}$$

Kalman Gain Matrix

$$\overline{\boldsymbol{x}}_i^+ = \overline{\boldsymbol{x}}_i^- + \mathcal{K}_i \left[\boldsymbol{y}_i - \mathcal{M}_i \overline{\boldsymbol{x}}_i^- \right]$$

$$\boxed{\mathcal{K}_i = \Sigma_i^- \mathcal{M}_i^T \left[\mathcal{M}_i \Sigma_i^- \mathcal{M}_i^T + \Sigma_{m_i} \right]^{-1}}$$

As measurement becomes more reliable, K weights residual more heavily,

$$\lim_{\Sigma_m \to 0} K_i = M^{-1}$$

As prior covariance approaches 0, measurements are ignored:

$$\lim_{\Sigma_i^- \to 0} K_i = 0$$

Dynamic Model:

$$\boldsymbol{x}_i \sim N(\mathcal{D}_i \boldsymbol{x}_{i-1}, \Sigma_{d_i})$$

$$\boldsymbol{y}_i \sim N(\mathcal{M}_i \boldsymbol{x}_i, \Sigma_{m_i})$$

Start Assumptions: $\overline{\boldsymbol{x}}_0^-$ and Σ_0^- are known
Update Equations: Prediction

$$\overline{\boldsymbol{x}}_i^- = \mathcal{D}_i \overline{\boldsymbol{x}}_{i-1}^+$$

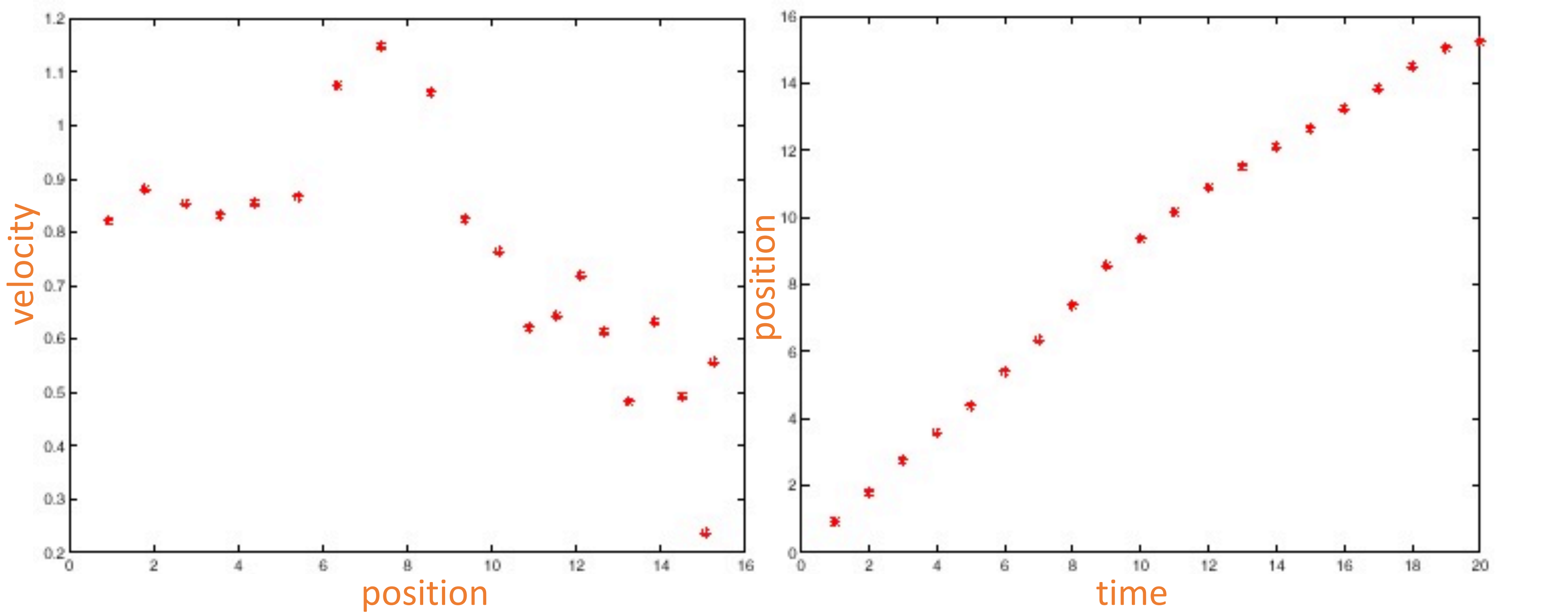
$$\Sigma_i^- = \Sigma_{d_i} + \mathcal{D}_i \sigma_{i-1}^+ \mathcal{D}_i$$

Update Equations: Correction

$$\mathcal{K}_i = \Sigma_i^- \mathcal{M}_i^T \left[\mathcal{M}_i \Sigma_i^- \mathcal{M}_i^T + \Sigma_{m_i}\right]^{-1}$$

$$\overline{\boldsymbol{x}}_i^+ = \overline{\boldsymbol{x}}_i^- + \mathcal{K}_i \left[\boldsymbol{y}_i - \mathcal{M}_i \overline{\boldsymbol{x}}_i^-\right]$$

$$\Sigma_i^+ = \left[Id - \mathcal{K}_i \mathcal{M}_i\right] \Sigma_i^-$$

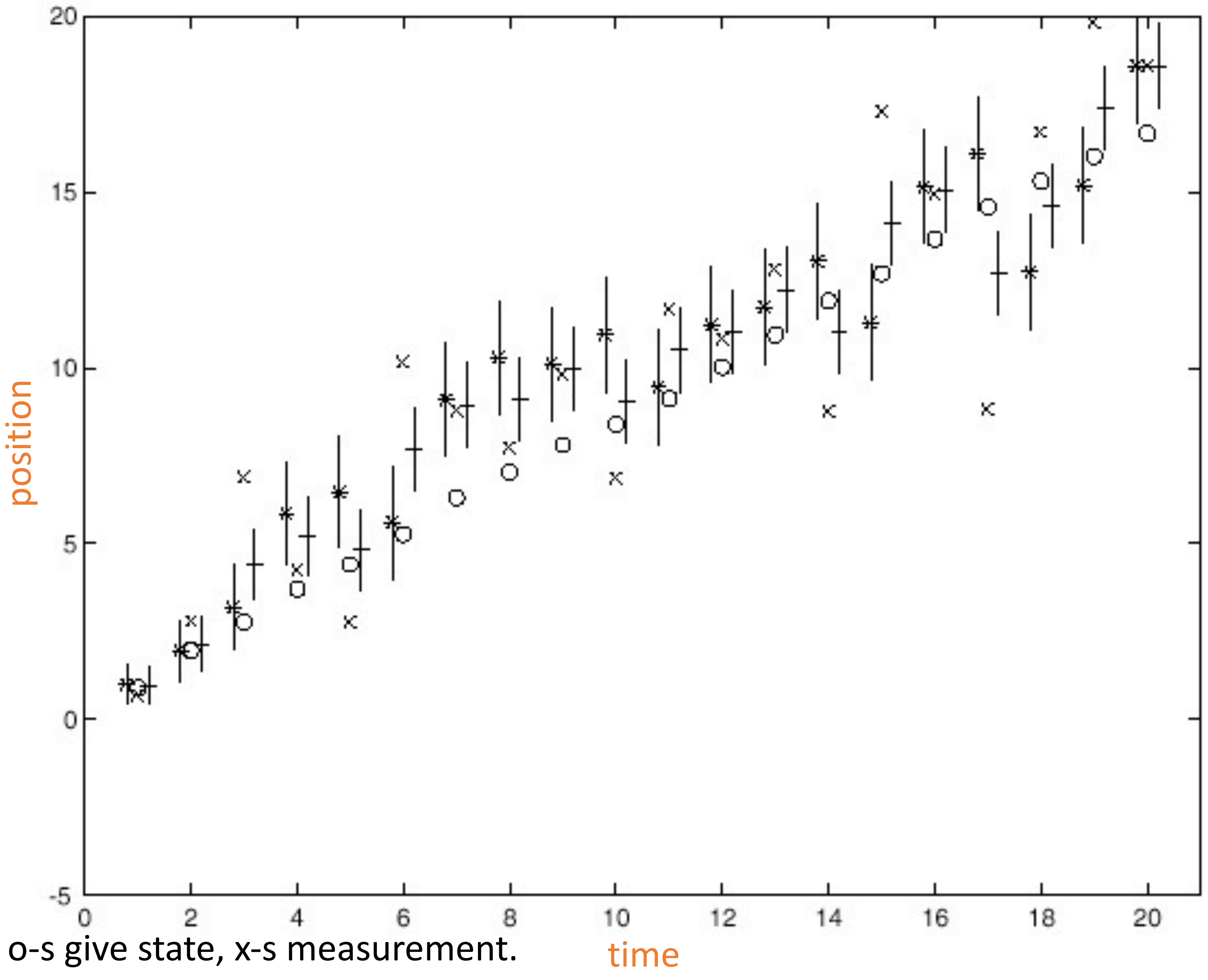


Constant Velocity Model

Smoothing

- Idea
  - We don’t have the best estimate of state - what about the future?
  - Run two filters, one moving forward, the other backward in time.
  - Now combine state estimates
    - The crucial point here is that we can obtain a smoothed estimate by viewing the backward filter’s prediction as yet another measurement for the forward filter

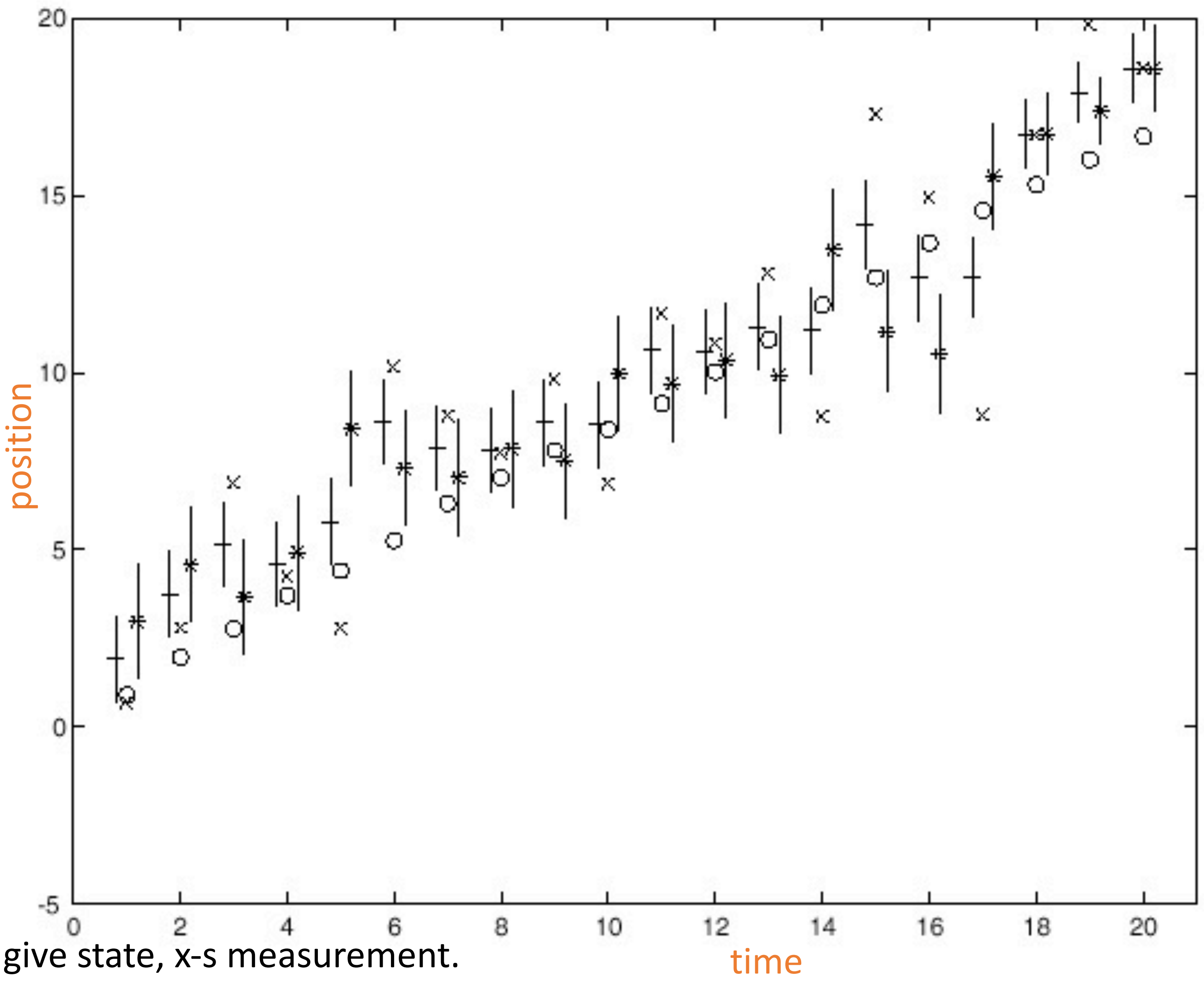
Forward estimates.



The o-s give state, x-s measurement.

The *-s give $\overline{\boldsymbol{x}}_i^-$, +-s give $\overline{\boldsymbol{x}}_i^+$, vertical bars are 3 standard deviation bars

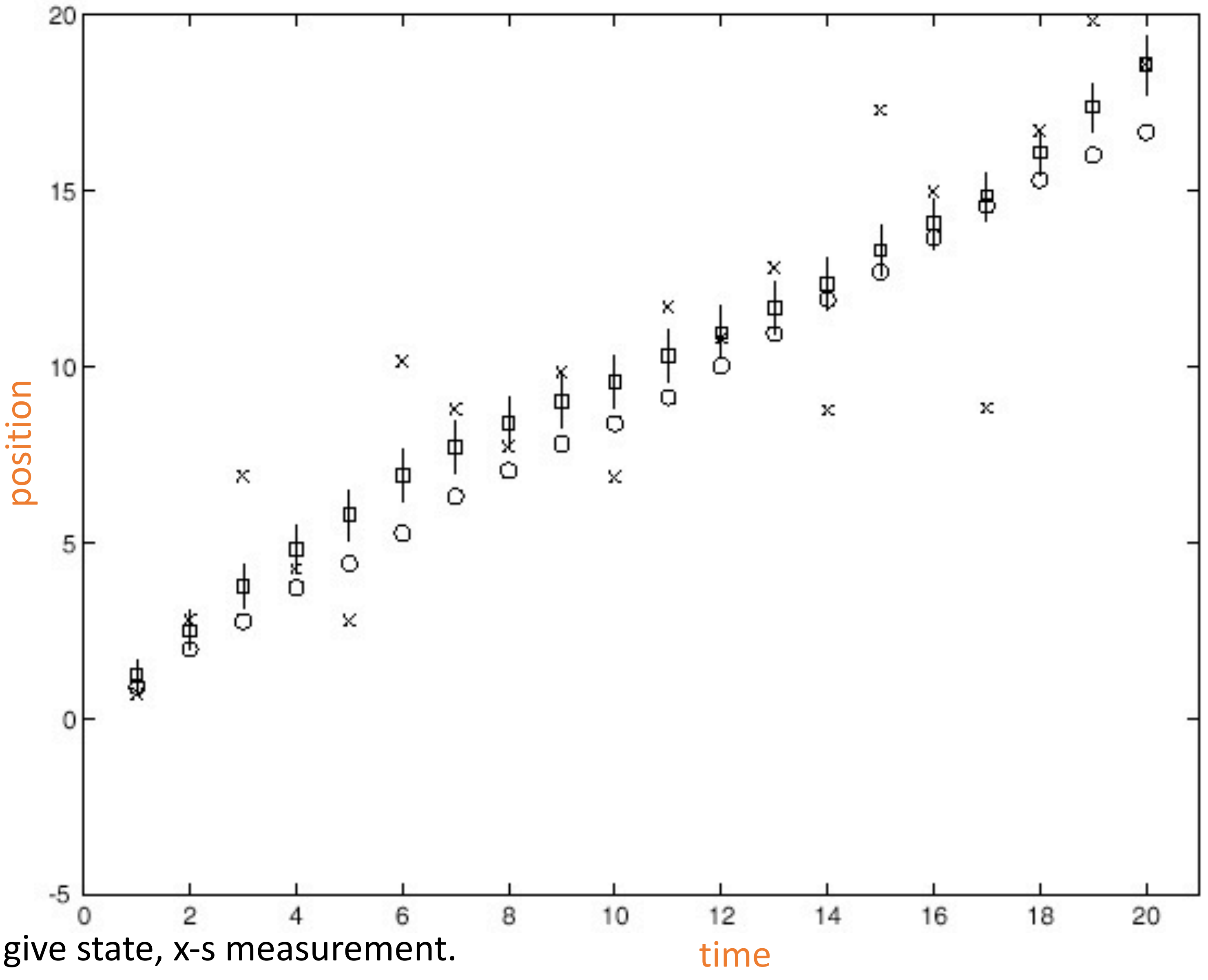
Backward estimates.



The o-s give state, x-s measurement.

The *-s give $\overline{x}_i^-$, +-s give $\overline{x}_i^+$, vertical bars are 3 standard deviation bars

Combined forward-backward estimates.

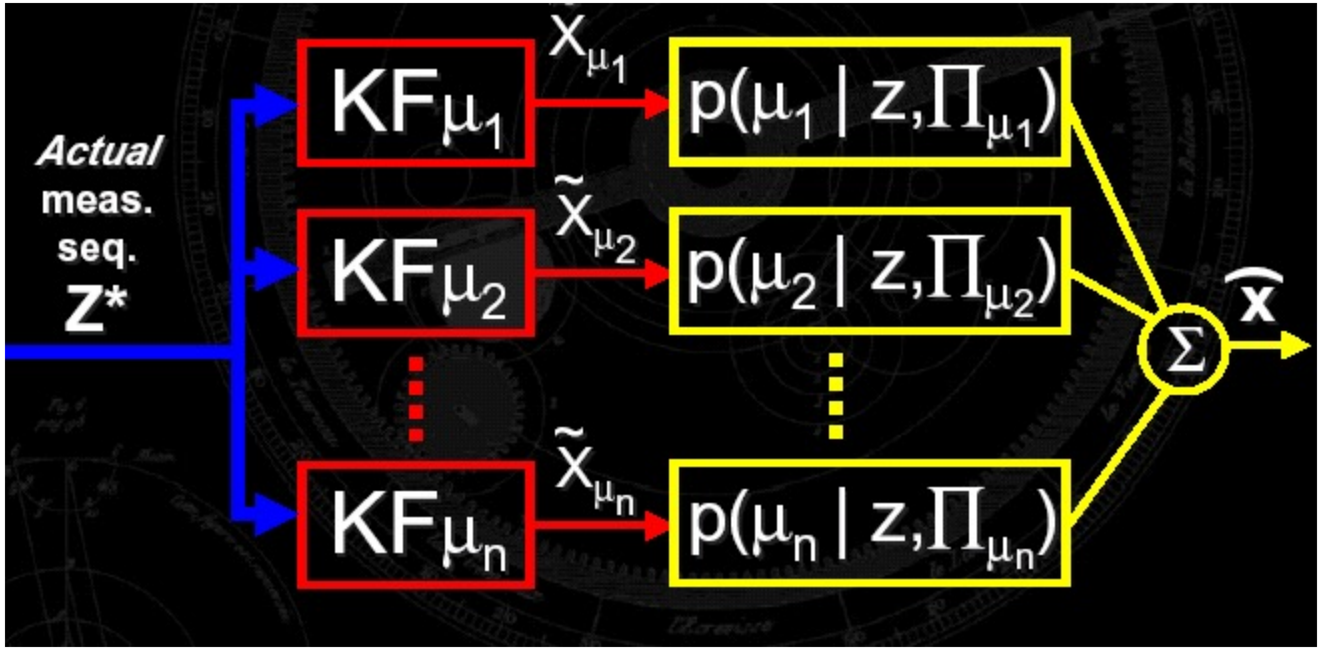


The o-s give state, x-s measurement.

The *-s give $\overline{x}_i^-$, +-s give $\overline{x}_i^+$, vertical bars are 3 standard deviation bars

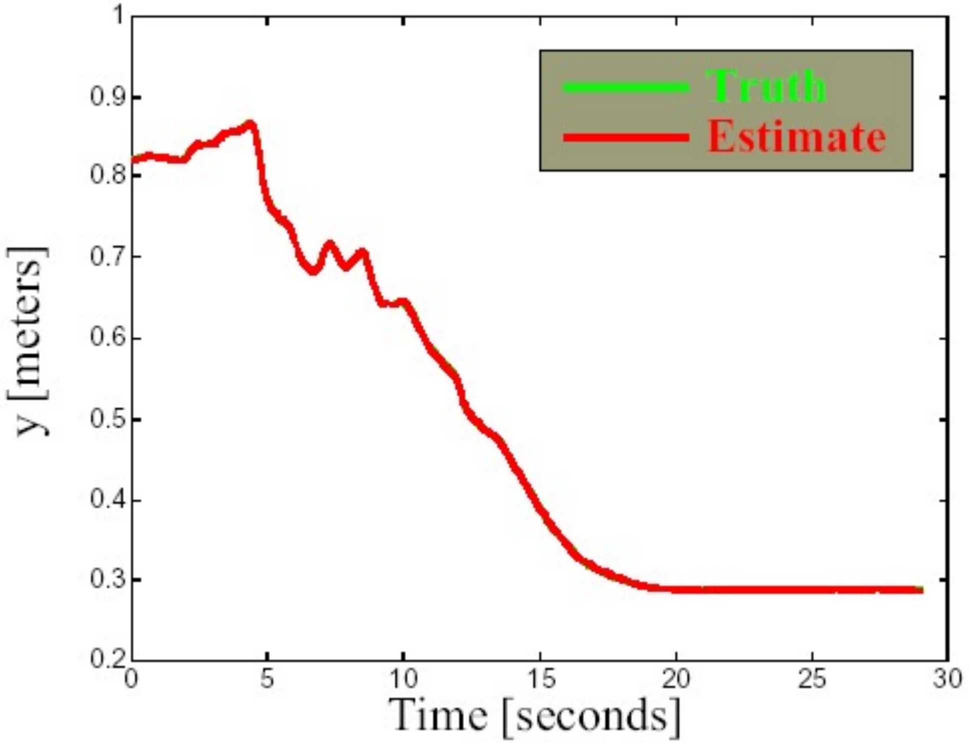
Multiple model filters

Test several models of assumed dynamics



MM estimate

Two models: Position (P), Position+Velocity (PV)

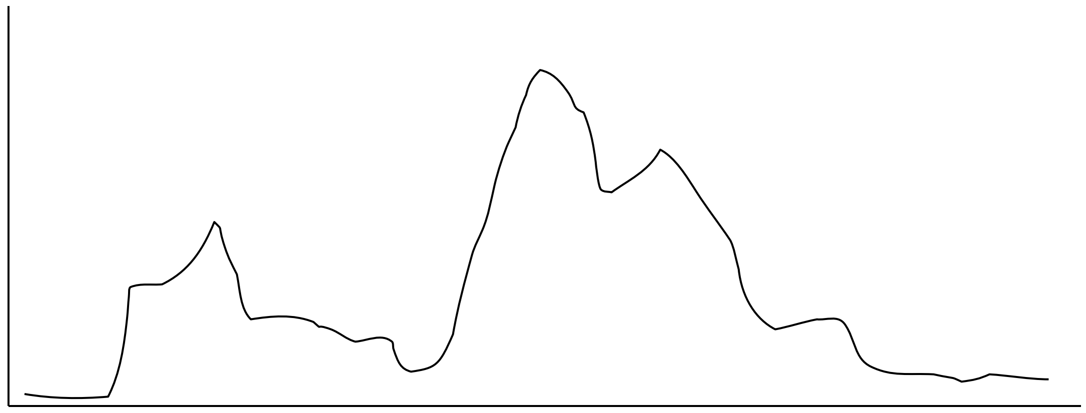


Non-toy image representation

Phase of a steerable quadrature pair (G2, H2). Steered to 4 different orientations, at 2 scales.

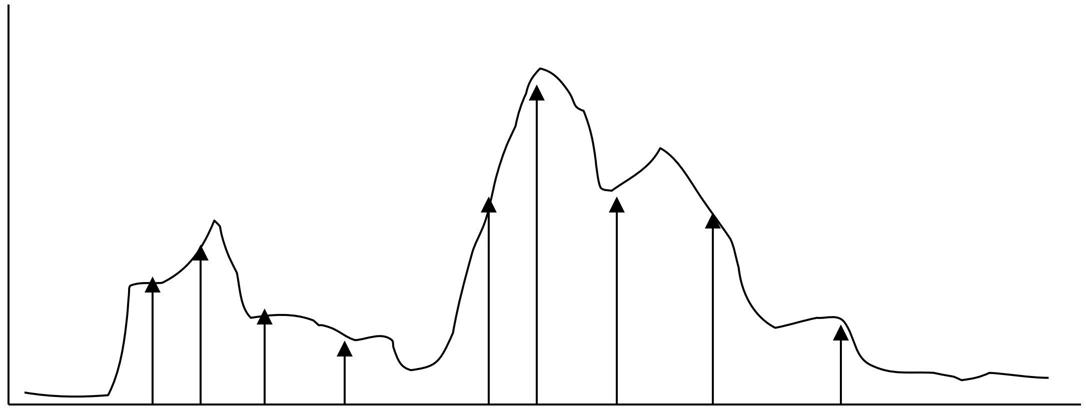
Representing Distributions using Weighted Samples

Rather than a parametric form, use a set of samples to represent a density:



Representing Distributions using Weighted Samples

Rather than a parametric form, use a set of samples to represent a density:



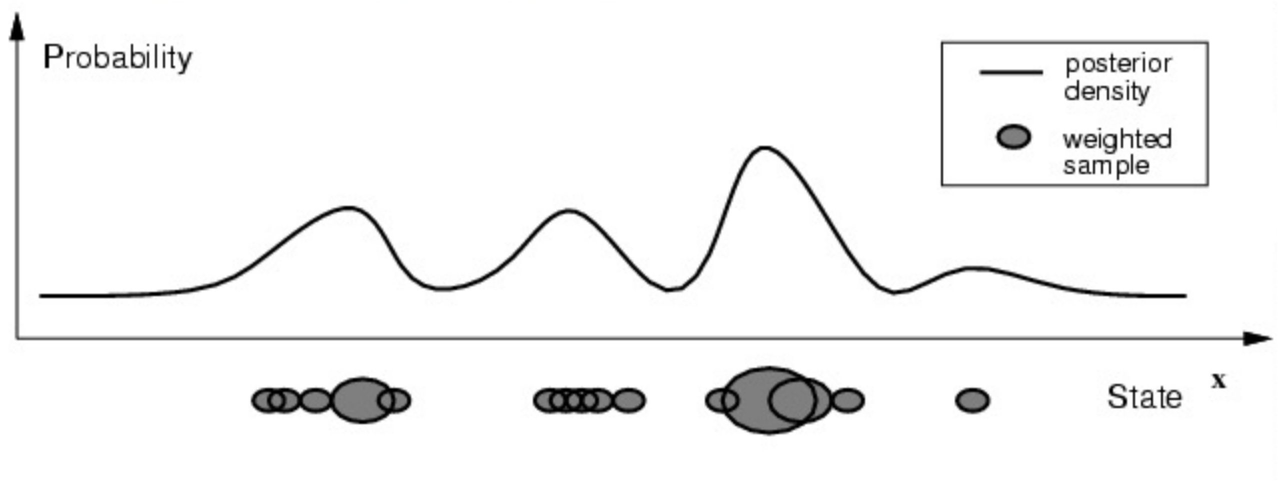
$$\{(\boldsymbol{u}^i, w^i)\}$$

Sample positions

Probability mass at each sample

This gives us two knobs to adjust when representing a probability density by samples: the locations of the samples, and the probability weight on each sample.

Representing distributions using weighted samples, another picture



[Isard 1998]

Sampled representation of a probability distribution

Represent a probability distribution

$$p_f(\boldsymbol{X}) = \frac{f(\boldsymbol{X})}{\int f(\boldsymbol{U})d\boldsymbol{U}}$$

by a set of N weighted samples

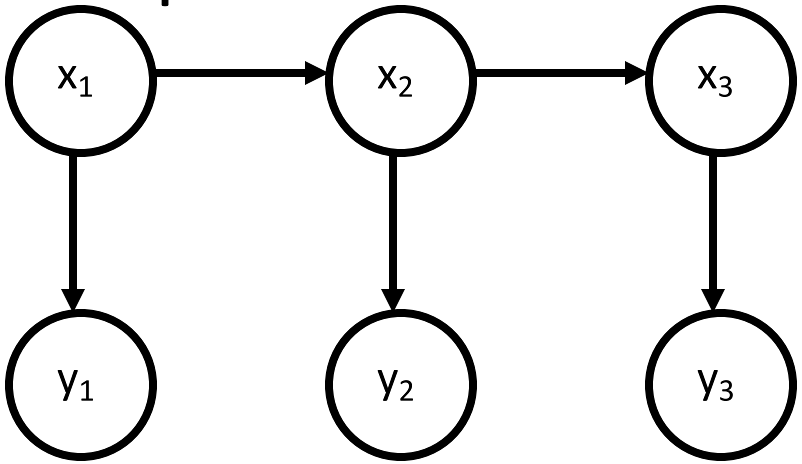
$$\{(\boldsymbol{u}^i, w^i)\}$$

where $\boldsymbol{u}^i \sim s(\boldsymbol{u})$ and $w^i = f(\boldsymbol{u}^i)/s(\boldsymbol{u}^i)$.

You can also think of this as a sum of dirac delta functions, each of weight w:

$$p_f(x) = \sum_i w^i \delta(x - u^i)$$

Tracking, in particle filter representation



$$p_f(x) = \sum_i w^i \delta(x - u^i)$$

$$P(x_n | y_1 \ldots y_n) = k\, P(y_n | x_n) \int dx_{n-1} P(x_n | x_{n-1}) P(x_{n-1} | y_1 \ldots y_{n-1})$$

Update step

Prediction step

Applications

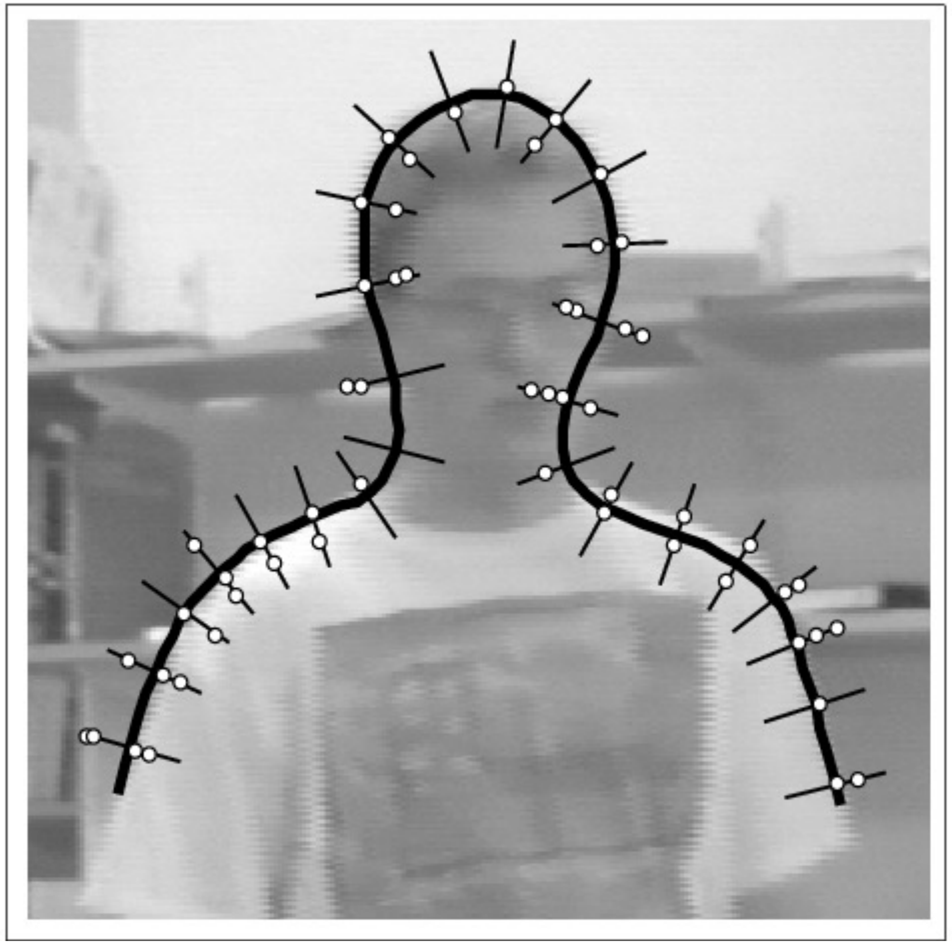
Tracking

- hands
- bodies
- Leaves

What might we expect?

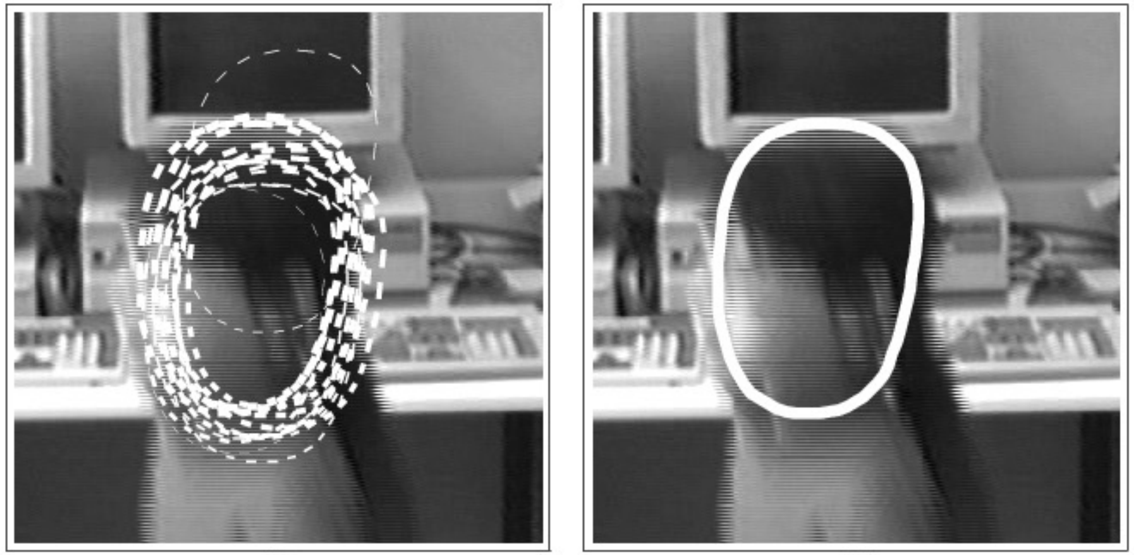
Reliable, robust, slow

Contour tracking



[Isard 1998]

Head tracking



(a) Picture of the states represented by the top weighted particles

(b) The mean state

[Isard 1998]

Leaf tracking

[Isard 1998]