CS545 – Machine Learning for Signal Processing

Missing Data and Dynamical Systems

11 October 2023

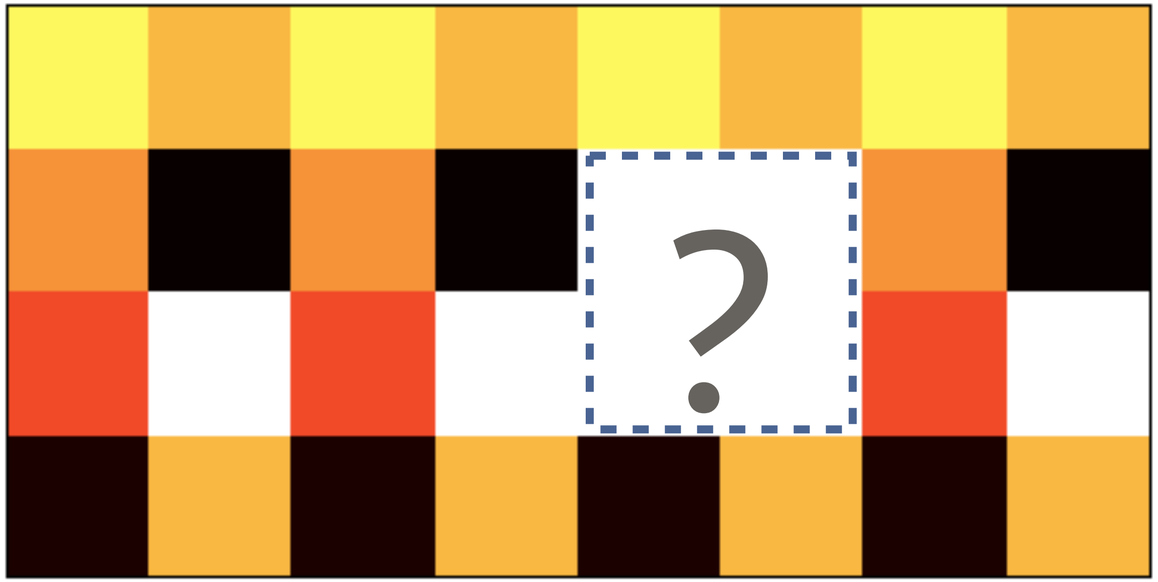
Today's lecture

- Dealing with missing data
- Tracking and linear dynamical systems

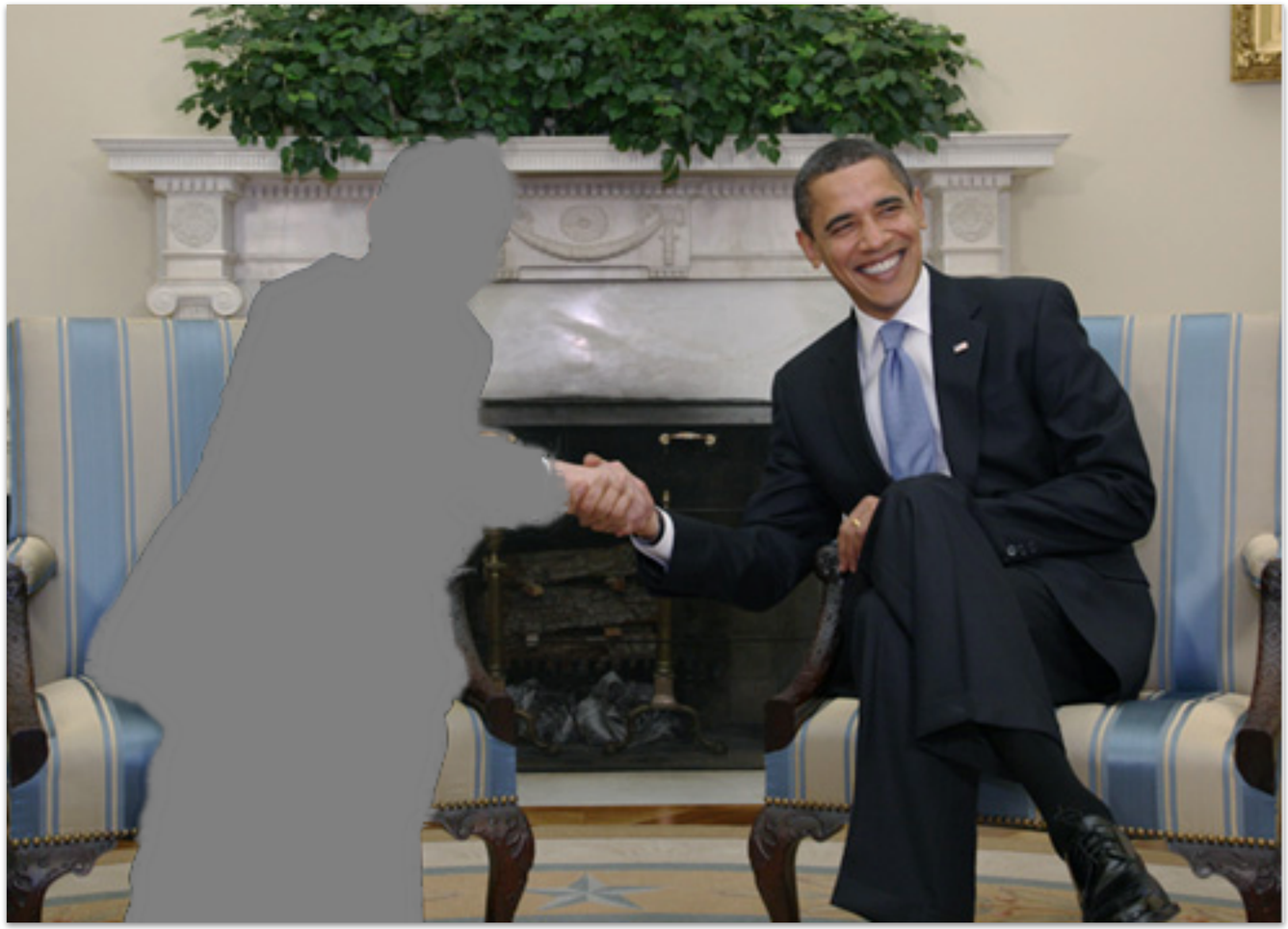
Missing data

- You don’t always have all your data!
  - e.g. cell phone drops, gaps in pictures, …
- How do we deal with that?

A simple case

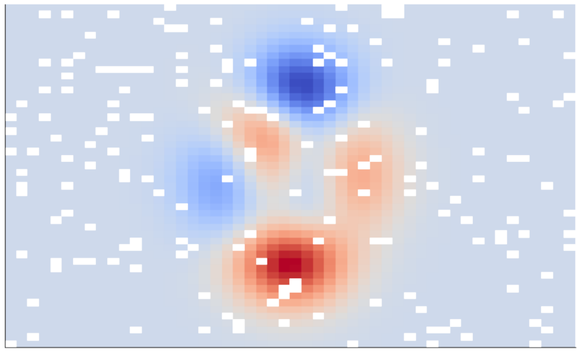


A real-world case!

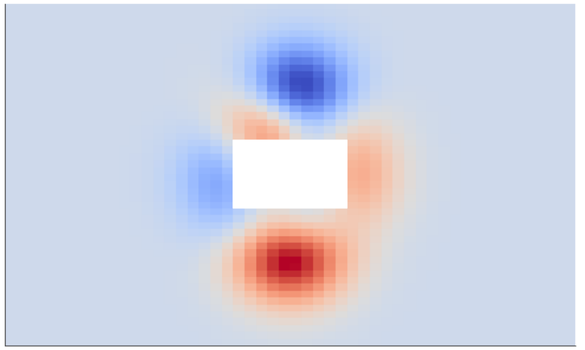


General classification of cases

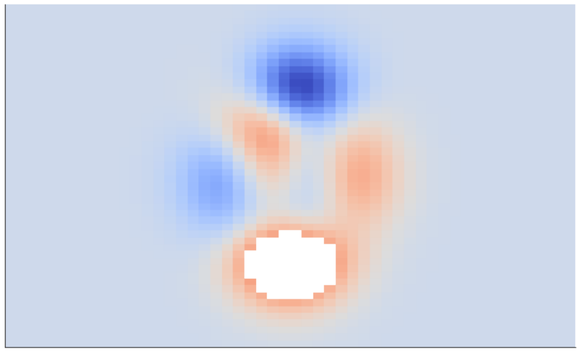
- Missing completely at random (MCAR)
  - "Missingness" is really random



- Missing at random (MAR)
  - "Missingness" depends on some variable
    - But not on the missing data values
    - Typical problem with questionnaires

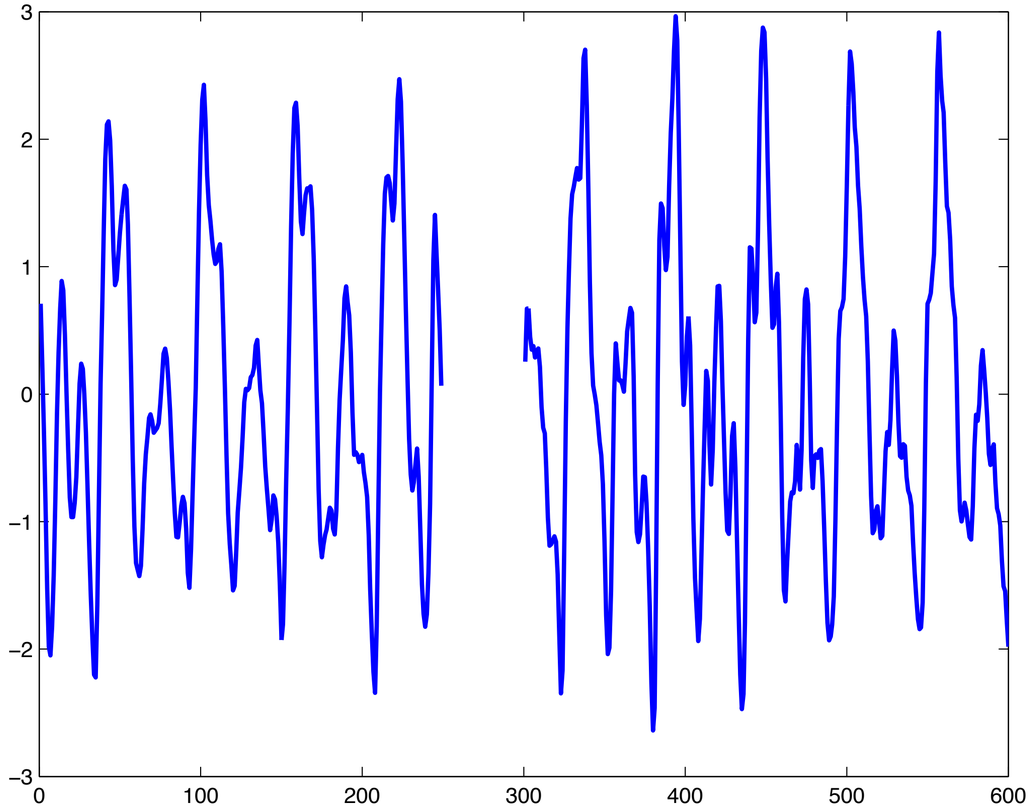


- Not missing at random (NMAR)
  - None of the above
  - Often depending on the data values



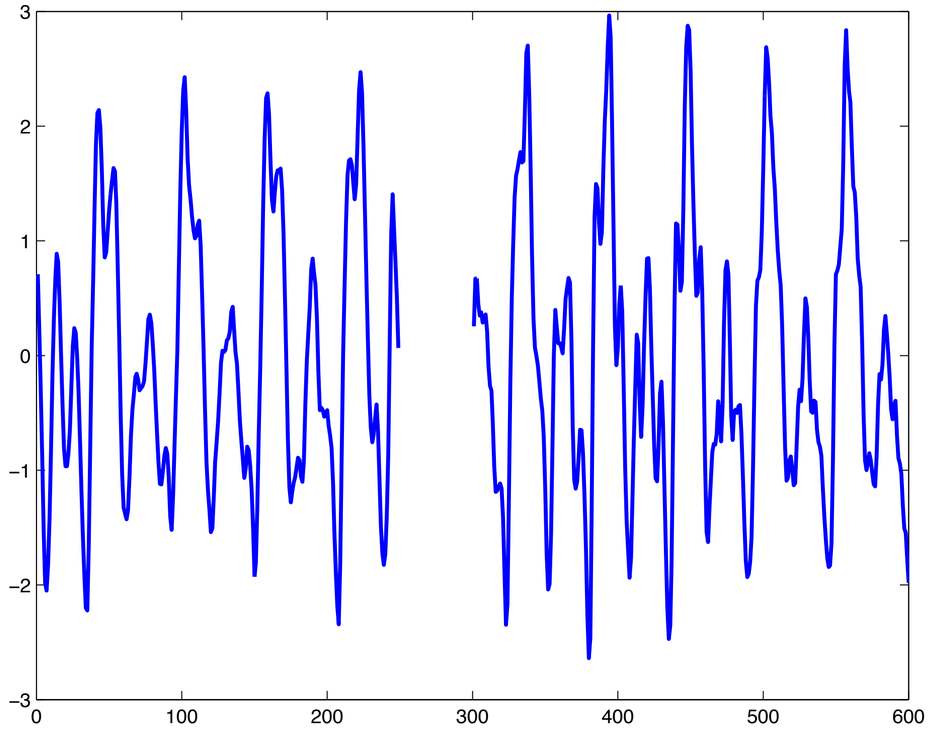
Starting with 1D

- Assume a small gap in 1D data
  - Can we find the missing values?



Key observations

- Temporal structure
  - The signal is somewhat periodic
- We could predict the future from the past
- How do we formalize this?



A predictive model

- Predict current sample from the past
  - Using a weighted average of preceding samples

$$x_t = \sum a_i x_{t-i} + e_t$$

- Autoregressive (AR) model
  - We can easily learn the coefficients a using various methods

Rewriting the model

- Linear algebra notation

$$\mathbf{e} = \mathbf{A} \cdot \mathbf{x}$$

$$\mathbf{A} = \begin{bmatrix} -a_p & \cdots & -a_1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & -a_p & \cdots & -a_1 & 1 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \cdots & 0 & 0 & -a_p & \cdots & -a_1 & 1 & 0 & 0 \\ 0 & \cdots & 0 & 0 & -a_p & \cdots & -a_1 & 1 & 0 \\ 0 & 0 & \cdots & 0 & 0 & -a_p & \cdots & -a_1 & 1 \end{bmatrix}$$

Filling gaps

- Model the input sequence as:

$$\mathbf{e} = \mathbf{A} \cdot \mathbf{x} = \mathbf{A} \cdot \left(\mathbf{U} \cdot \mathbf{x}_u + \mathbf{K} \cdot \mathbf{x}_k \right) = \mathbf{A}_u \cdot \mathbf{x}_u + \mathbf{A}_k \cdot \mathbf{x}_k$$

  - $\mathbf{x}_u$ and $\mathbf{x}_k$ are the unknown and known samples
  - $\mathbf{U}$ and $\mathbf{K}$ are repositioning matrices, e.g.:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \cdot \underbrace{\begin{bmatrix} x_2 \\ x_3 \end{bmatrix}}_{\textit{Unknown}} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \cdot \underbrace{\begin{bmatrix} x_1 \\ x_4 \end{bmatrix}}_{\textit{Known}}$$

Isolating the unknowns

- Final form isolates unknowns as a variable
  - These are values that should conform to the model
    - Assuming they aren't missing at random!

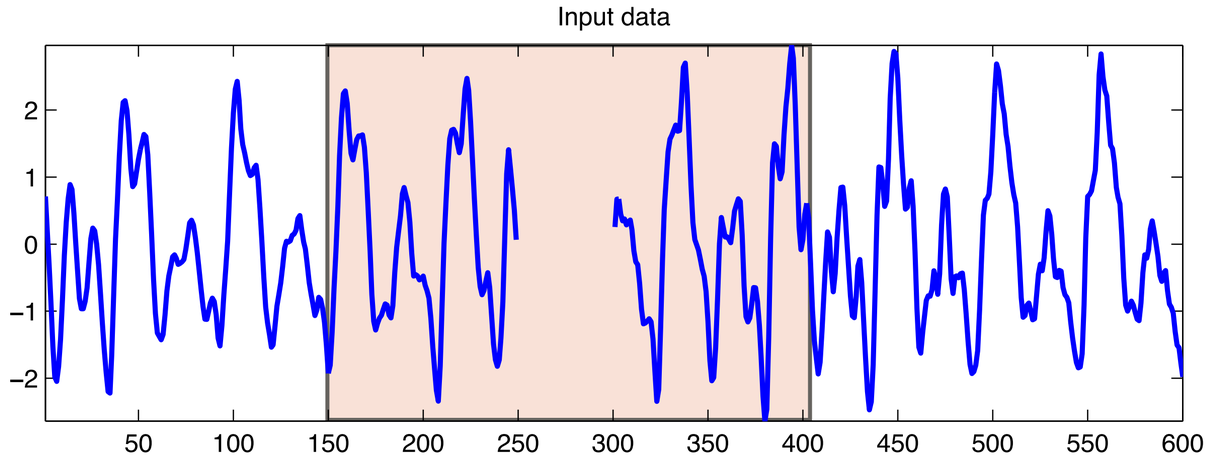
- We thus want to find $\mathbf{x}_u$ so that we minimize $\mathbf{e}$
  - i.e. fill in the blanks with least unpredictable samples

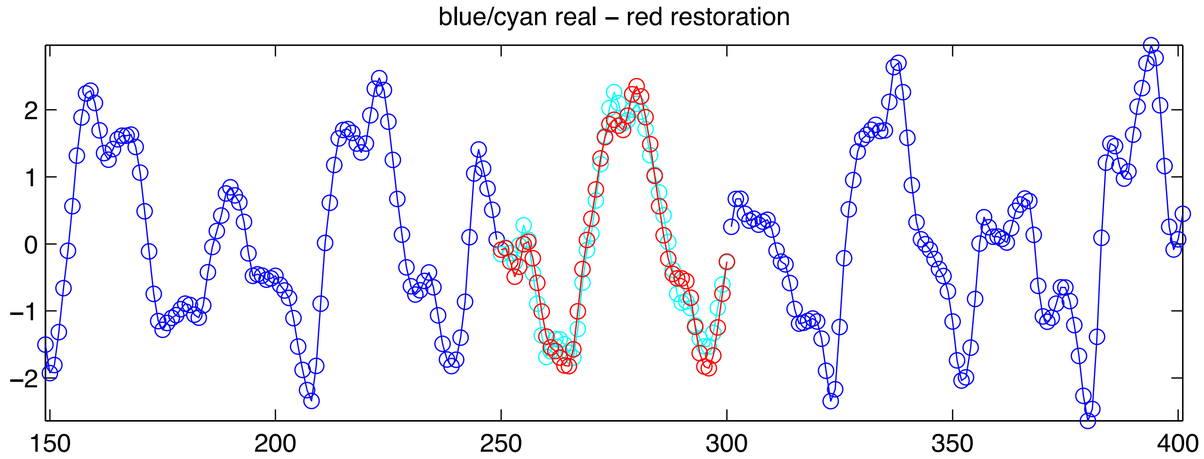
$$\mathbf{A}_u \cdot \mathbf{x}_u + \mathbf{A}_k \cdot \mathbf{x}_k = \mathbf{e}$$

$$\Rightarrow \mathbf{x}_u = -\mathbf{A}_u^+ \cdot \mathbf{A}_k \cdot \mathbf{x}_k$$

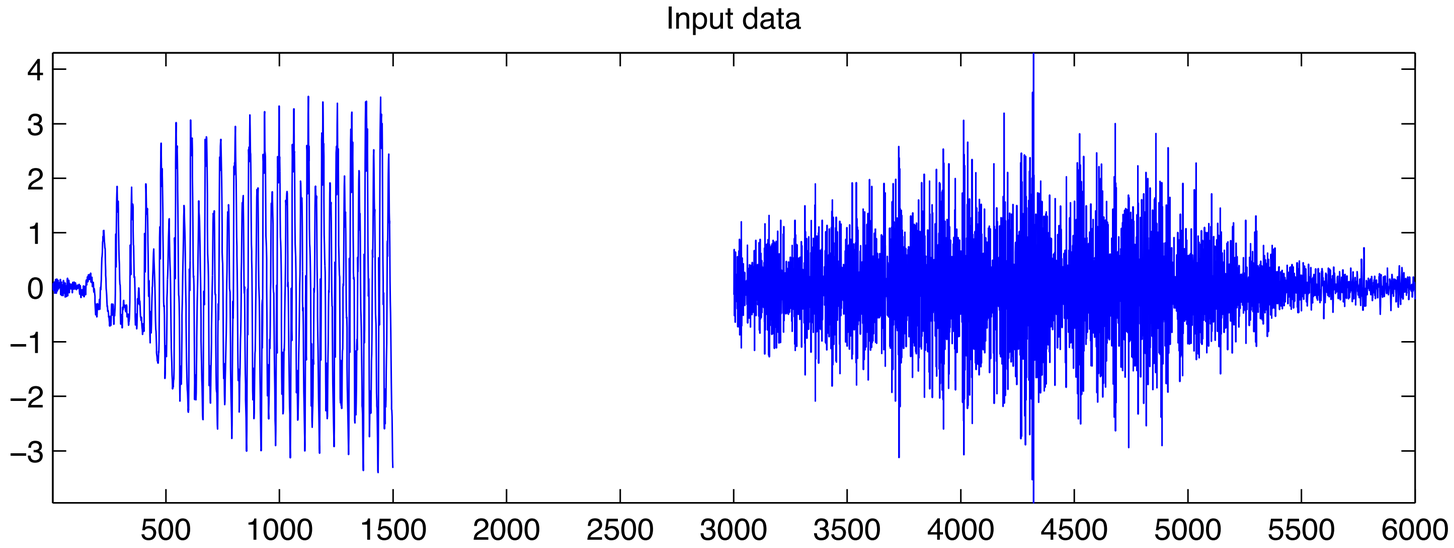
Results

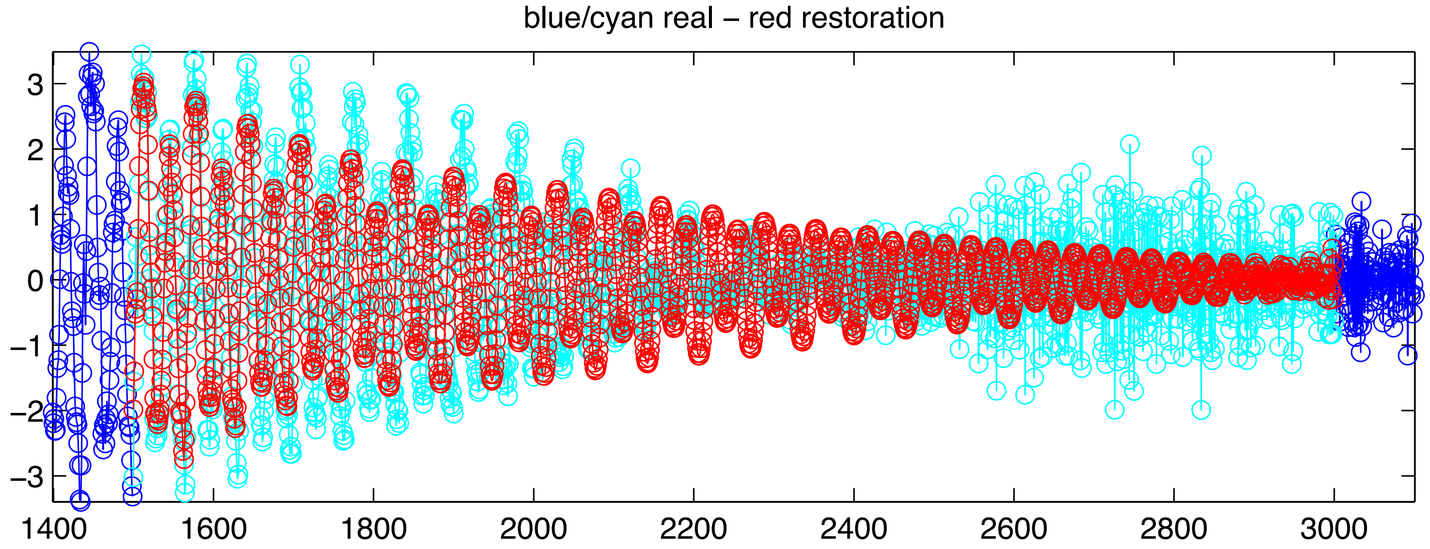
- Works pretty well!





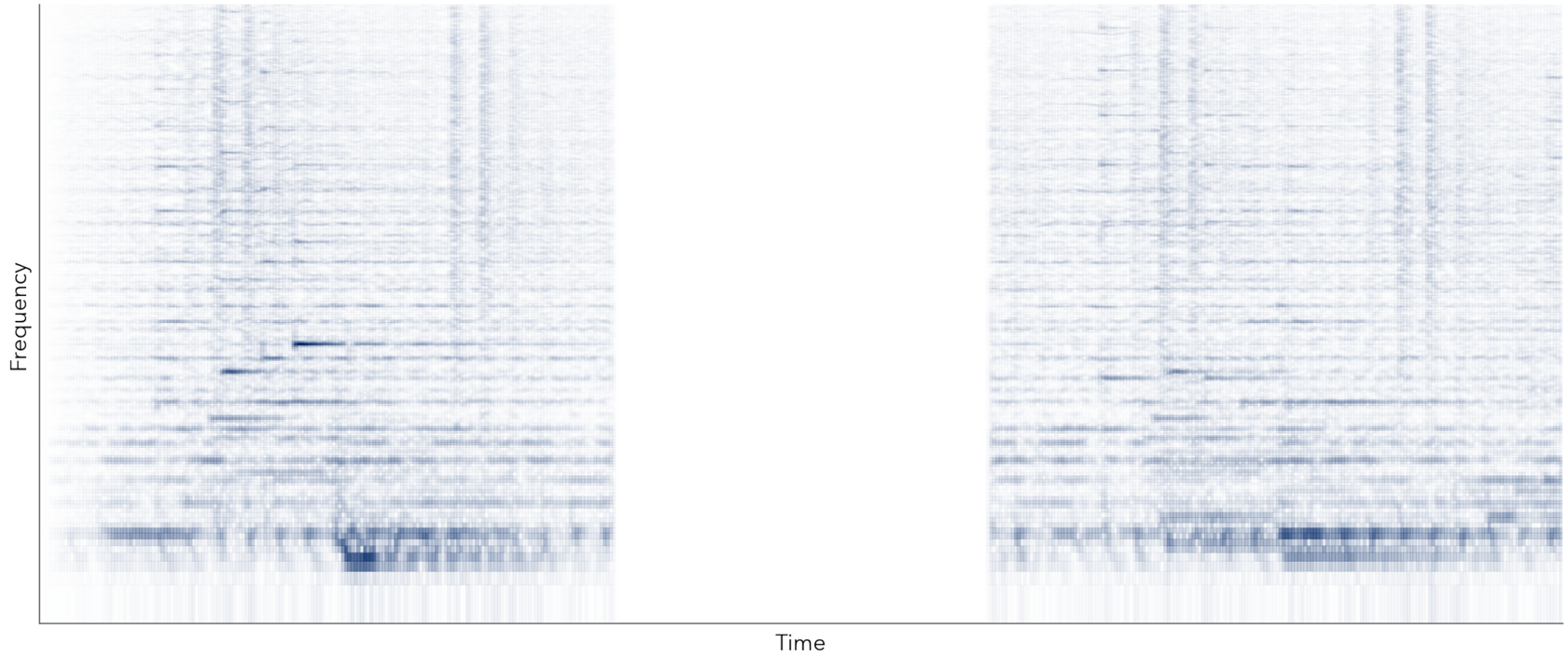
But not for large windows!





Looking for a new idea

- We should move away from the sample level, and look at a coarser scale
  - e.g. a time-frequency view



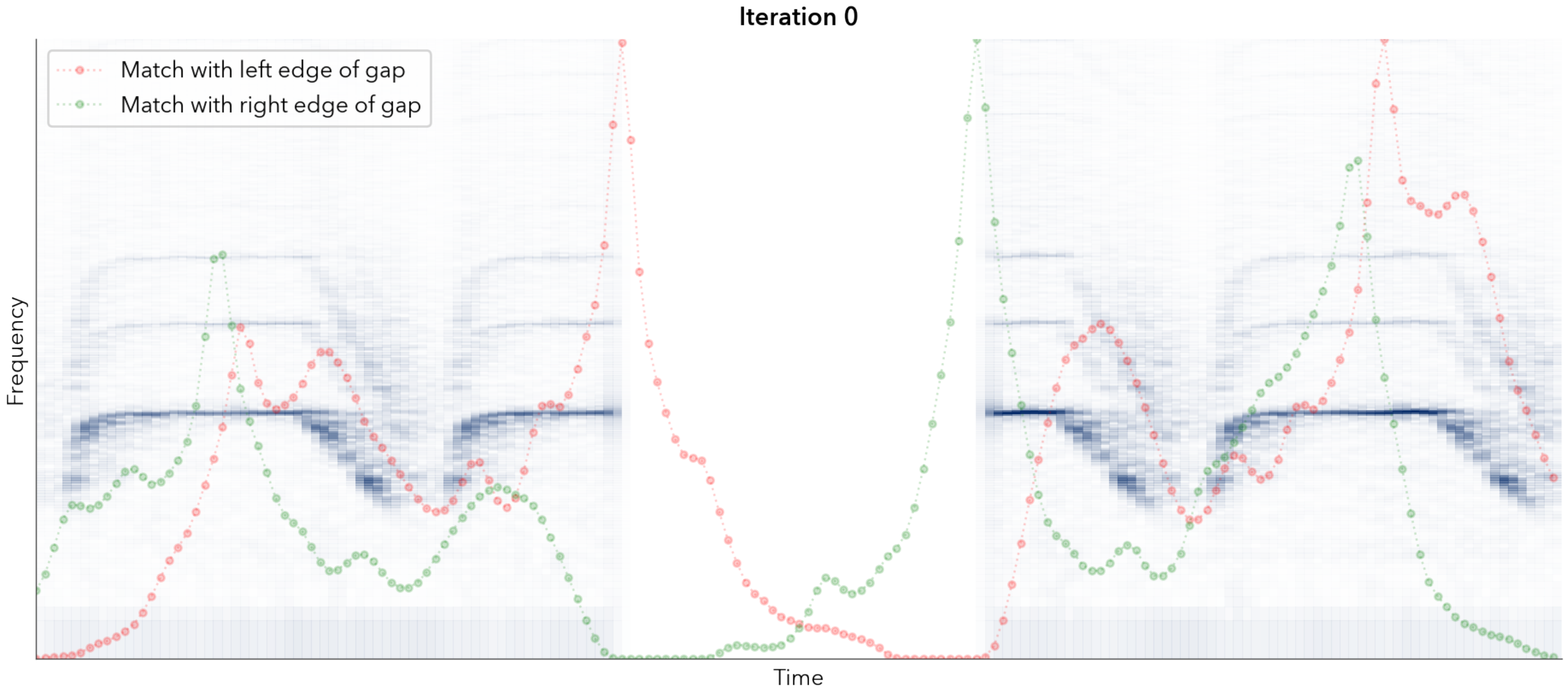
A simpler idea

- Find sections most similar to the gap edges
  - Replace missing sections with their neighbors

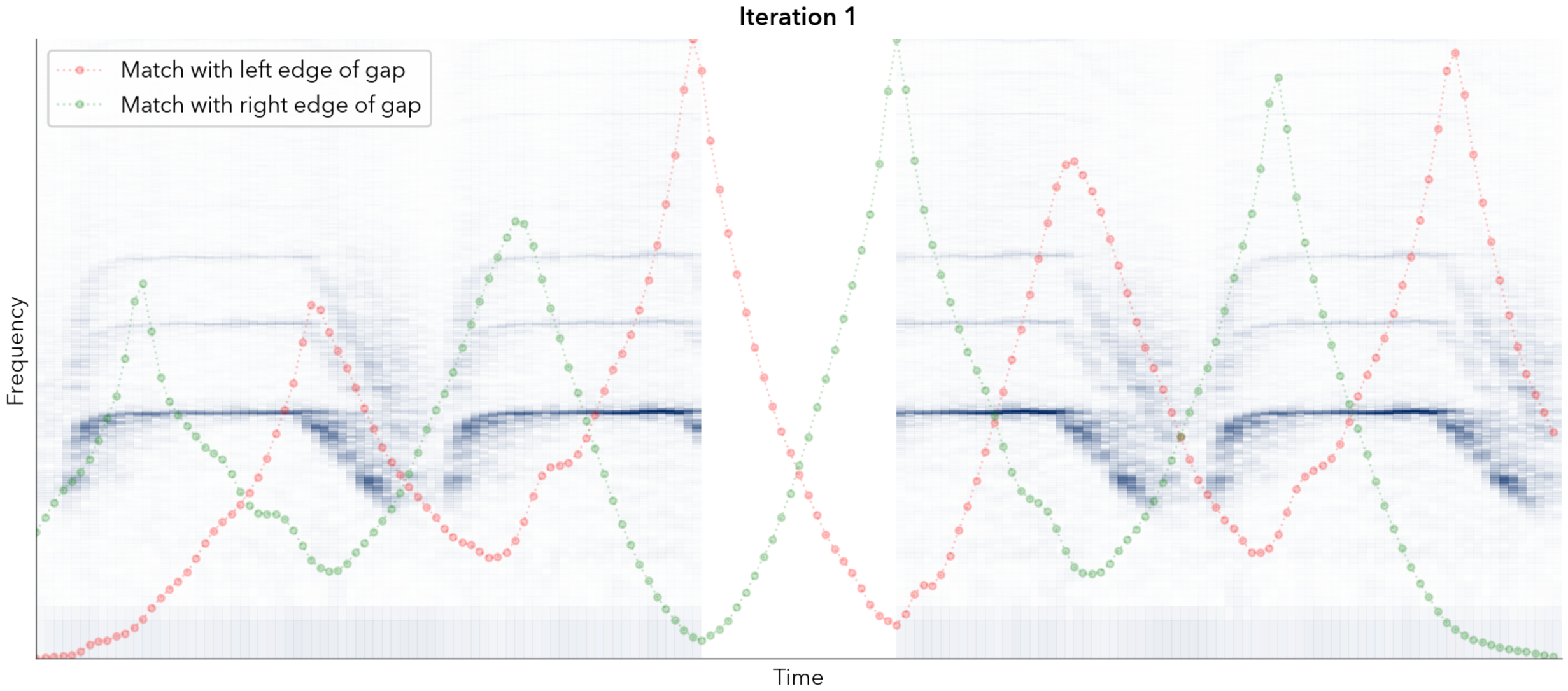
$$\begin{bmatrix} \uparrow & \uparrow & \uparrow & ? & ? & \uparrow & \uparrow & \uparrow \\ \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & ? & ? & \mathbf{x}_6 & \mathbf{x}_7 & \mathbf{x}_8 \\ \downarrow & \downarrow & \downarrow & ? & ? & \downarrow & \downarrow & \downarrow \end{bmatrix}$$

- How do we find the close match?

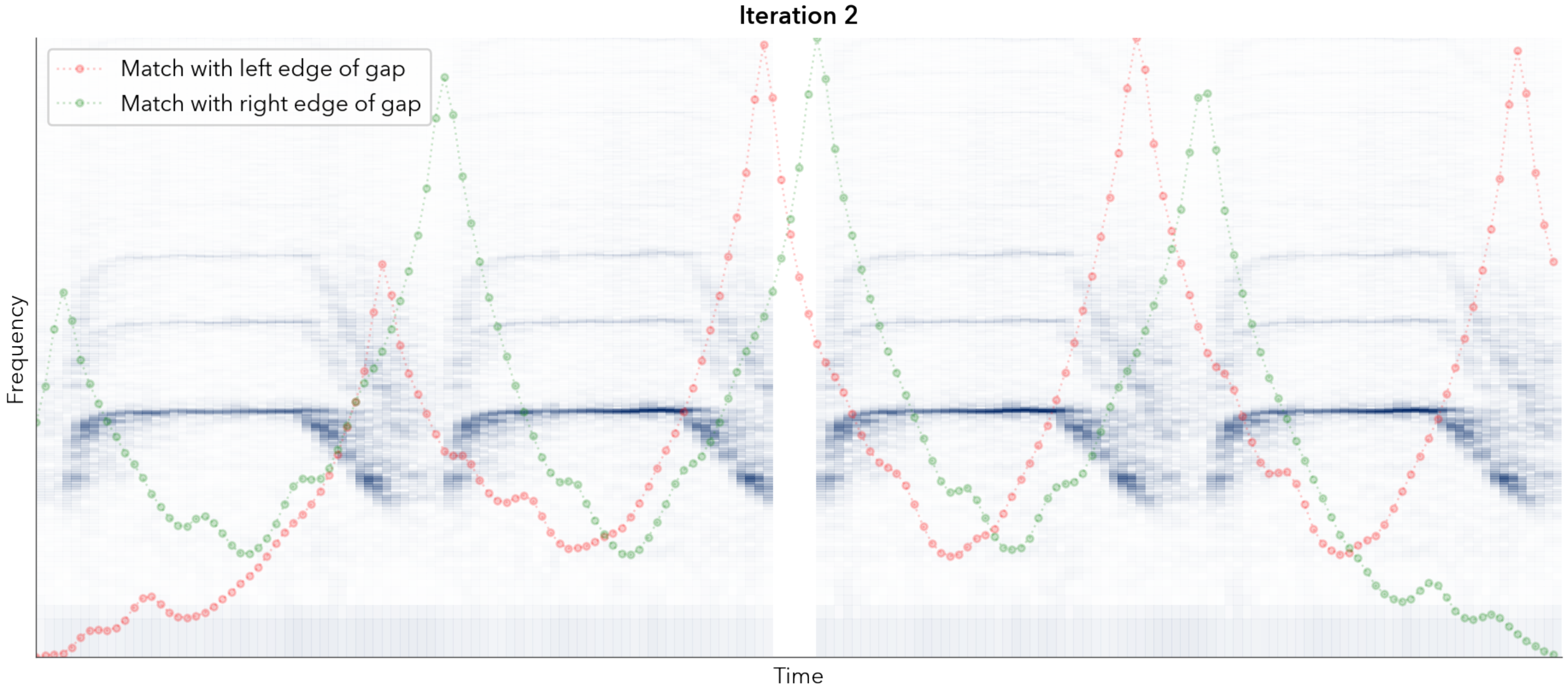
In action


Iteration 0
Match with left edge of gap
Match with right edge of gap
Frequency
Time

In action


Iteration 1
Match with left edge of gap
Match with right edge of gap
Frequency
Time

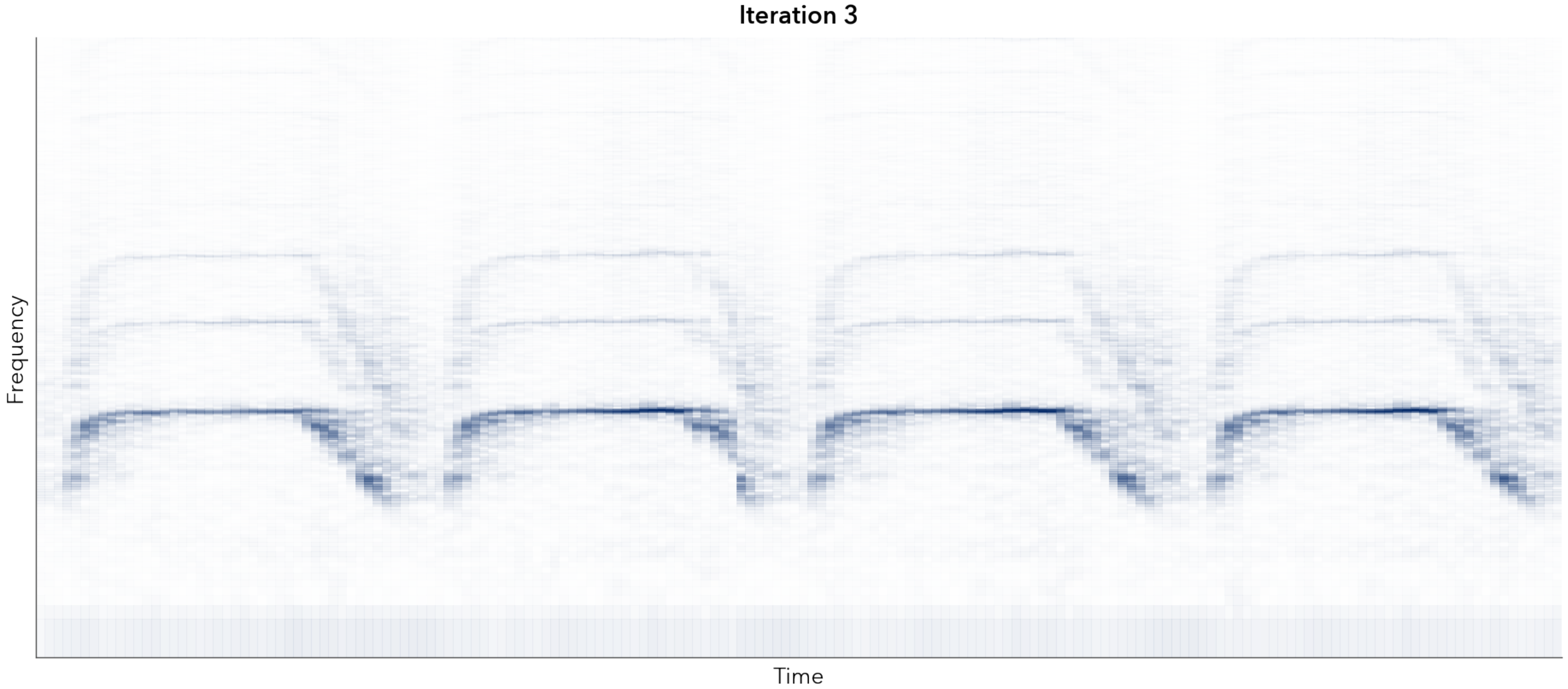
In action


Iteration 2
Match with left edge of gap
Match with right edge of gap
Frequency
Time

In action

Input

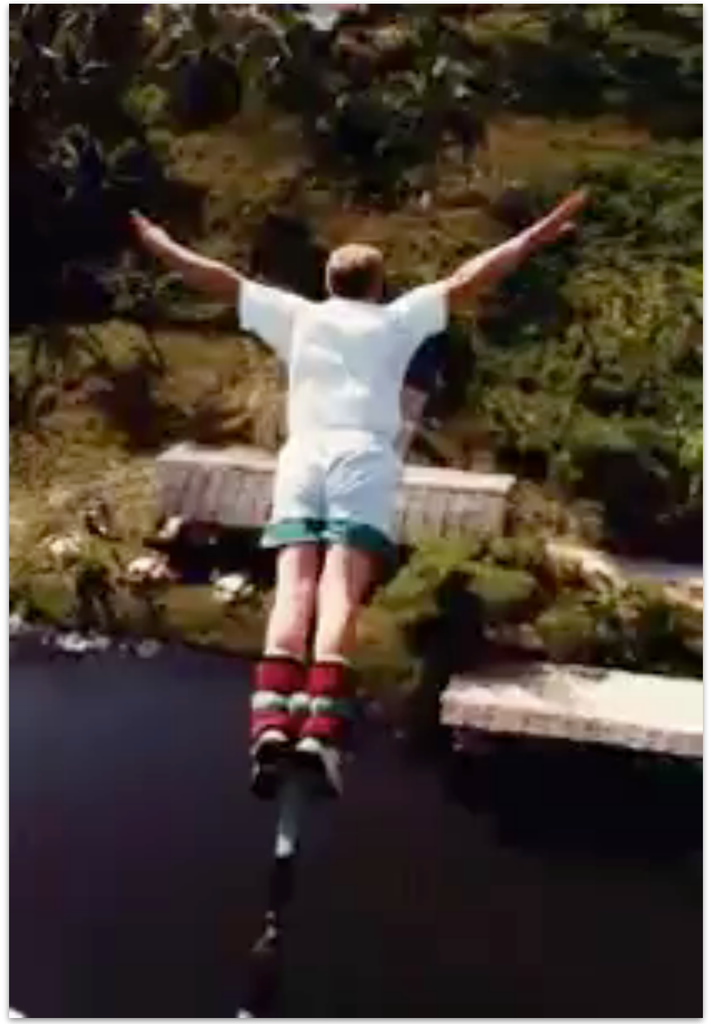
Output



Some examples

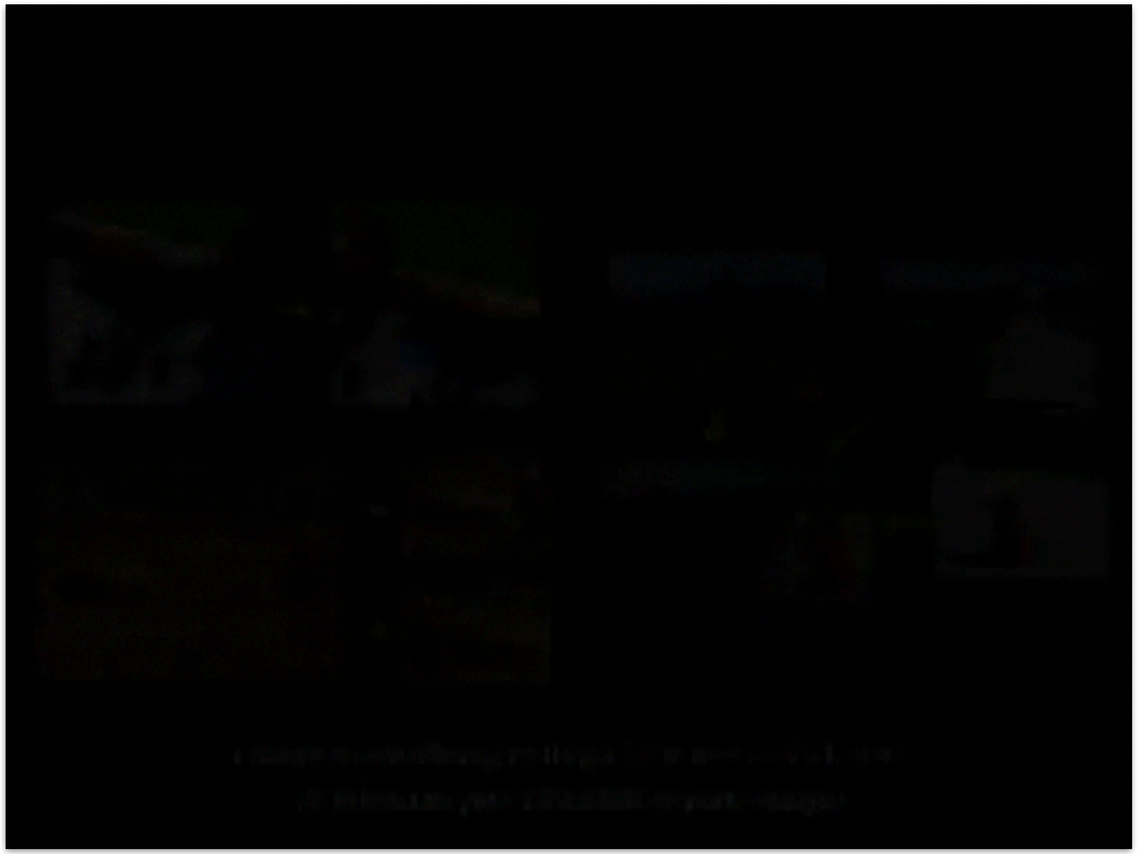
What about images?

- Basic idea is the same
  - Find a neighbor and blend it
- A bit more complicated
  - We need to blend carefully, we can’t just add
    - Poisson blending
- Also more computationally intensive (2d search!)



A real-world example

- Inpainting, recomposing, warping the truth!



PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing
Connelly Barnes, Eli Shechtman, Adam Finkelstein, Dan B Goldman

A global statistical viewpoint

- Nearest-neighbor search was local
  - We ignore global data structure
- Missing data should conform to a statistical model of the input
  - i.e. they shouldn't be outliers

SVD-based imputation

- Simple global approach
  - Replace missing data with some rough guesses

$$x_u = E\{x\} \quad \text{or} \quad x_u \sim \mathcal{N}(\mu, \sigma^2) \quad \text{or} \ldots$$

  - Perform a low-rank SVD approximation of the data

$$\mathbf{X} \approx \mathbf{U} \cdot \mathbf{S} \cdot \mathbf{V}^\top$$

  - Replace missing data with SVD approximation

$$\mathbf{x}_u = \left(\mathbf{U} \cdot \mathbf{S} \cdot \mathbf{V}^\top \right)_u$$

  - Repeat!

Why?

- Known data will dominate statistics
  - Assuming they are enough!
- Each approximation will try to conform to the global statistics of the input
  - i.e. made up values will not be random
- With each new iteration, that conformance to global statistics will increase

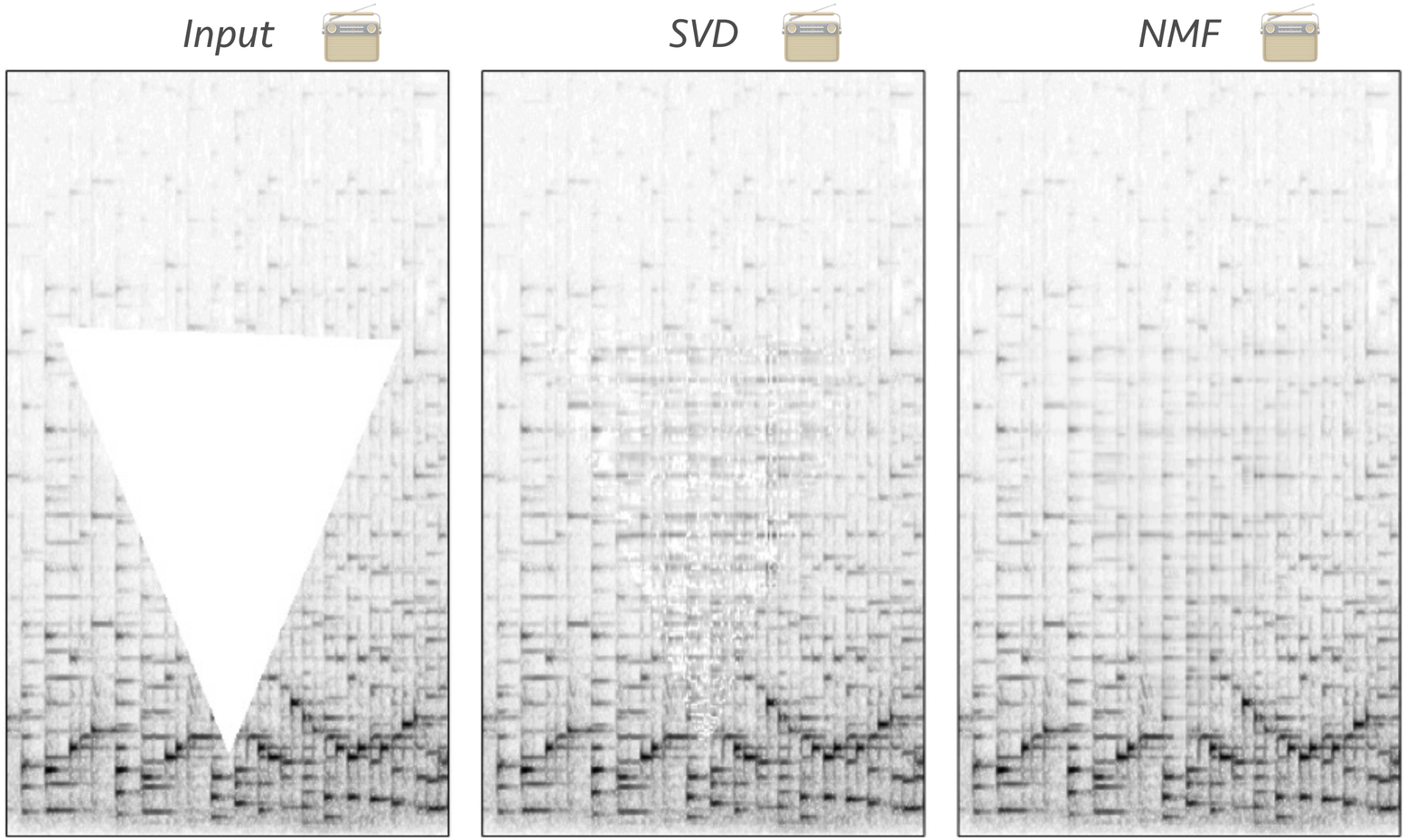
Variations

- We don’t have to use an SVD
  - e.g. our data might be non-negative
    - So use NMF, or probabilistic models, or whatever makes sense

- Any global statistical model would work
  - Just make sure that it fits the data well

Example

- Learn from input and fill-in missing values



Tracking

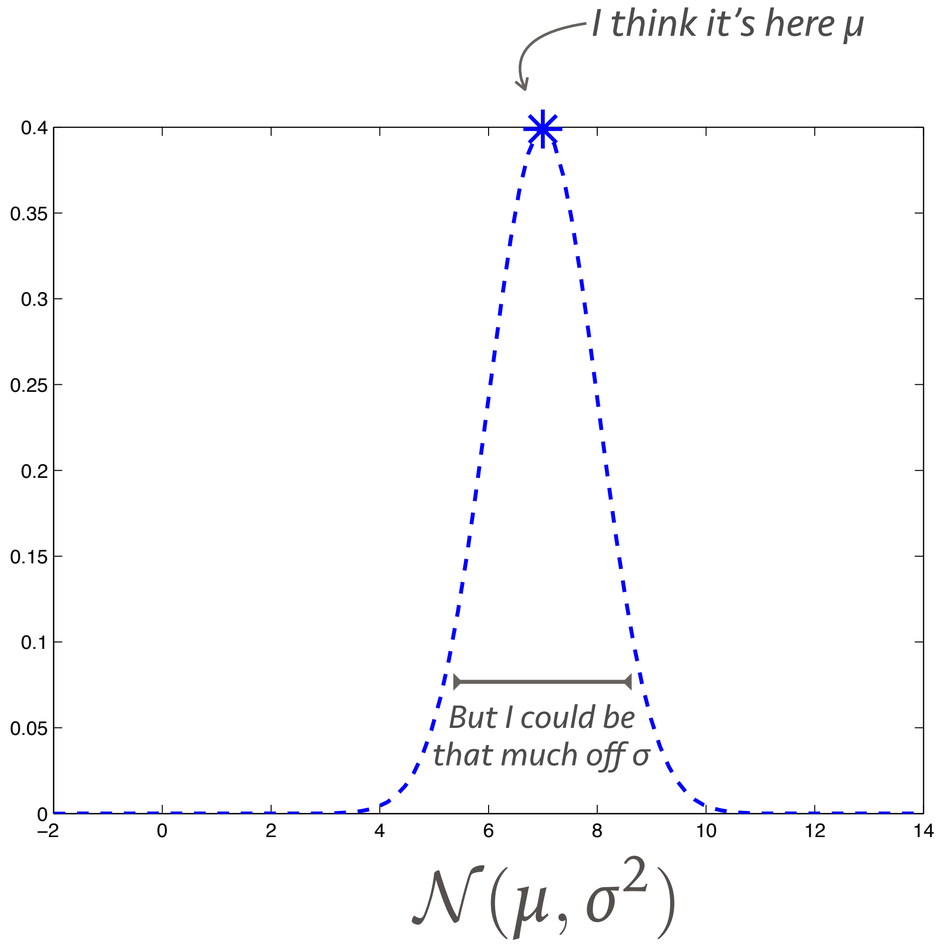
- Tracking things that evolve in time
  - missiles, faces, finances, etc ...
- A time-series model again

A simple problem

- I'm looking at a star which is about "there"
  - So does my drunken friend
- How do we consolidate our observations?

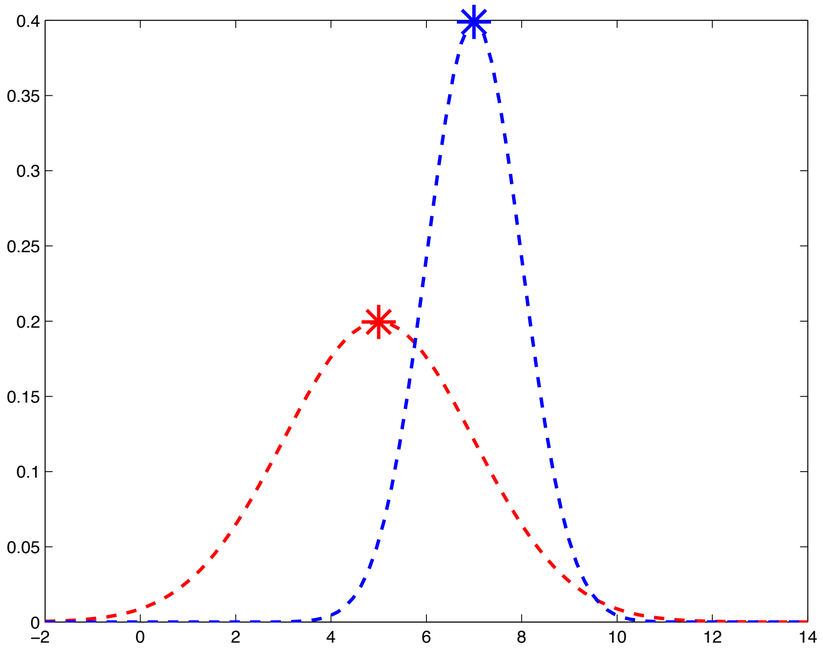
An uncertain estimate

- My observation is a Gaussian that helps me account for some uncertainty
  - Mean μ is what I think is right
  - Variance σ is an indication of how sure I am



More uncertainty

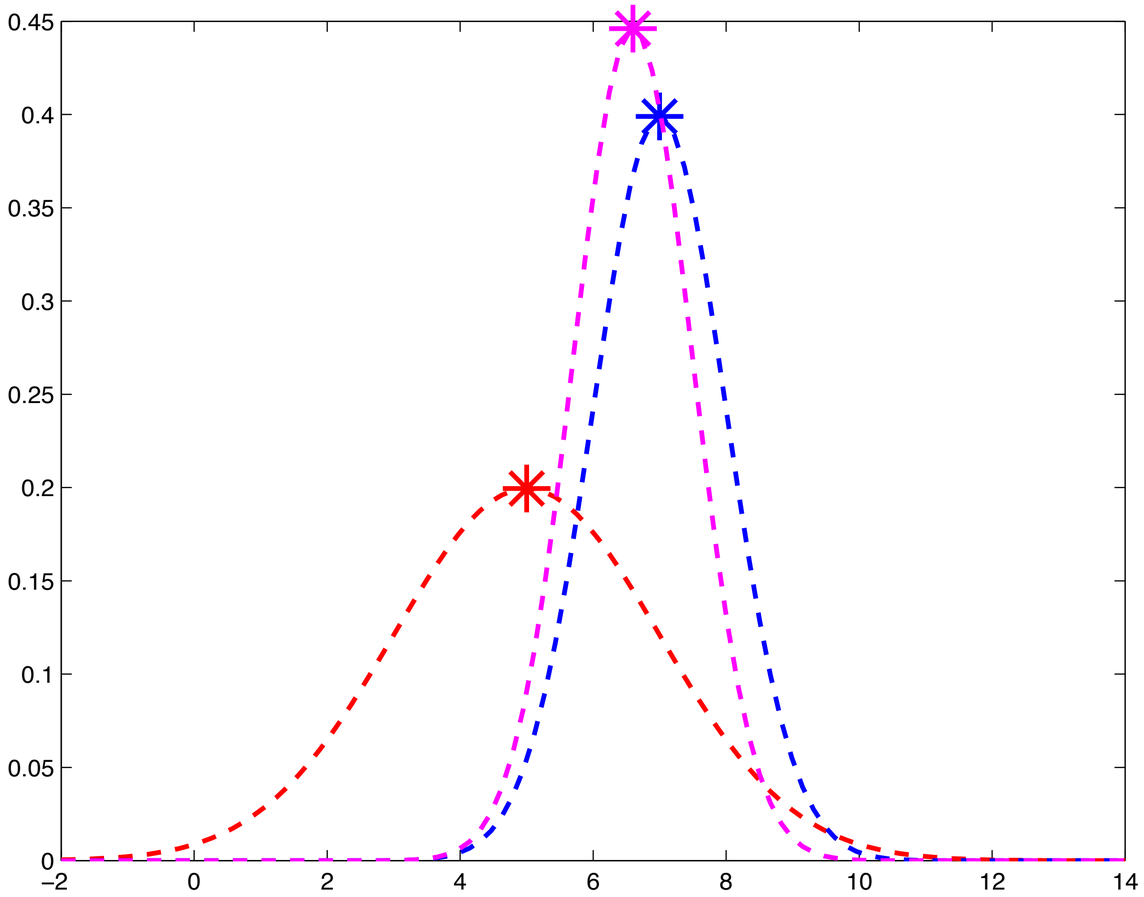
- My drunken friend's estimate is unreliable
  - Therefore my friend's variance is higher



$$\mathcal{N}(\mu_1, {\sigma_1}^2)$$
$$\mathcal{N}(\mu_2, {\sigma_2}^2)$$
$$\sigma_2 > \sigma_1$$

Consensus estimate

- The consensus estimate is proportional to a Gaussian



$$\mathcal{N}(m,s) \propto \mathcal{N}(\mu_1,\sigma_1)\mathcal{N}(\mu_2,\sigma_2)$$

$$K = \frac{\sigma_1^2}{\sigma_1^2+\sigma_2^2}$$

$$m = (1-K)\mu_1 + K\mu_2 = \mu_1 + K(\mu_1+\mu_2)$$

$$s = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2} = (1-K)\sigma_1^2$$

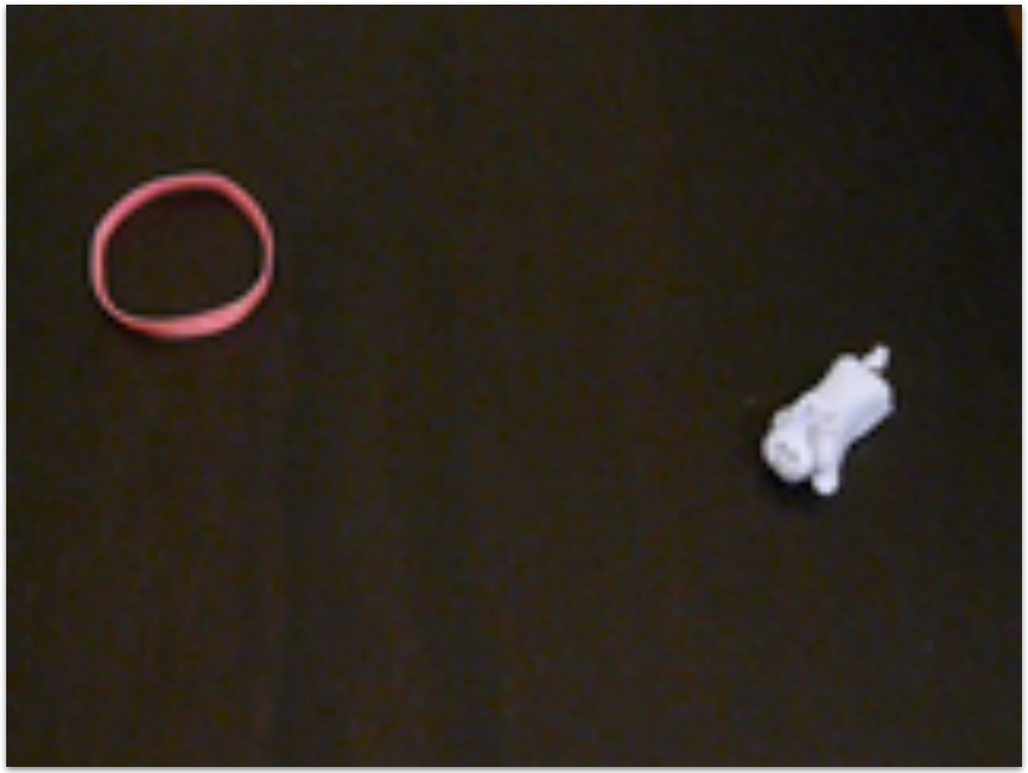
Serializing this process

- Can we use that consensus idea for refining estimates from successive measurements?

- If we have a target to track, can we use this to make a better estimate over time?

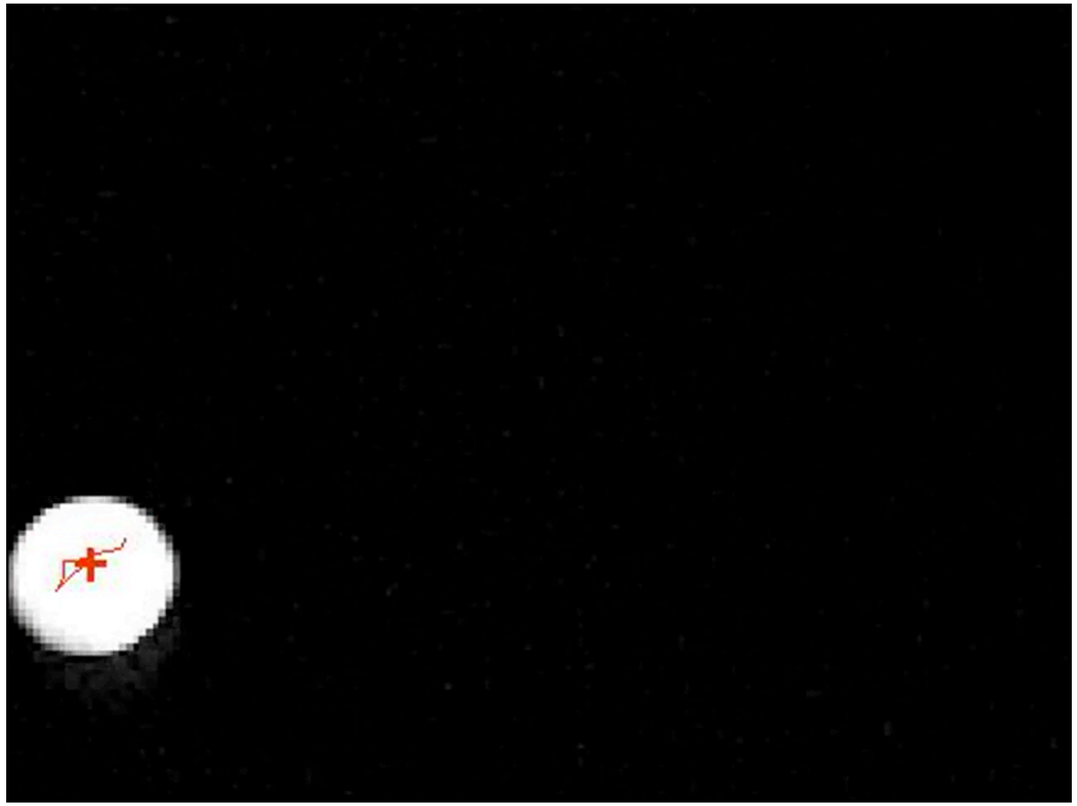
Example case

- Track the position of a moving ball



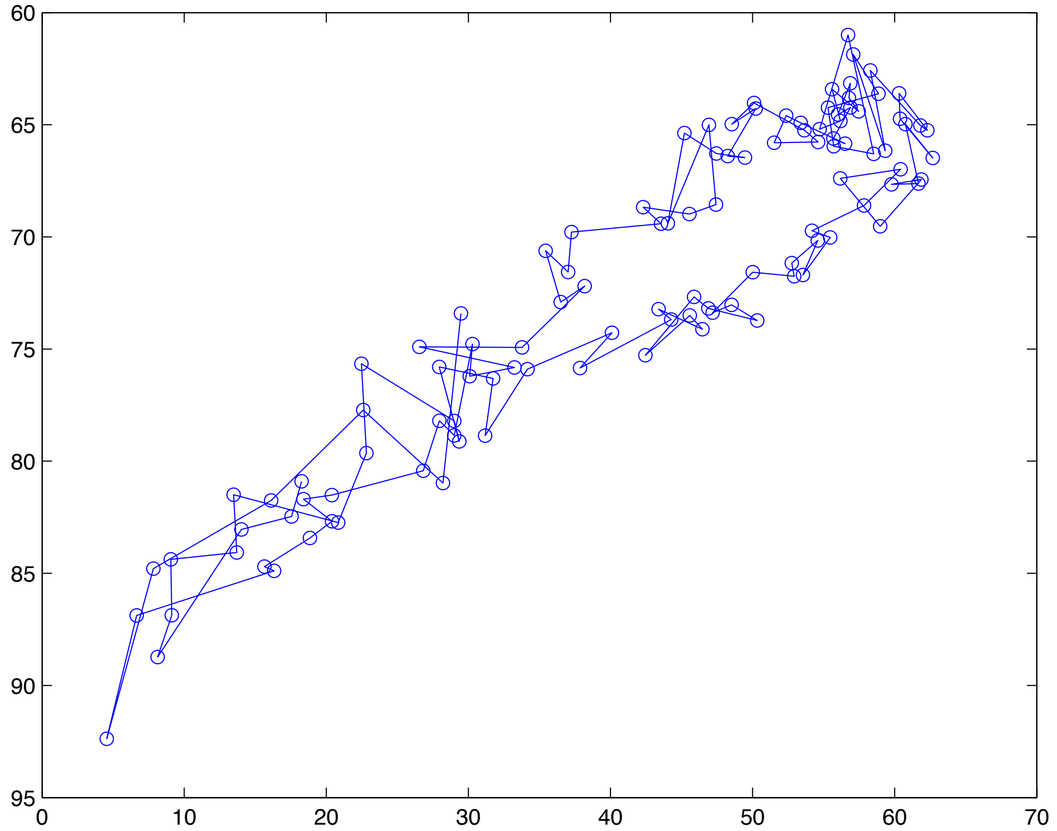
Some simple processing

- Background removal
  - Subtract constant part, find center of remaining pixels



The problem

- The position estimate is very noisy
  - Lets see how we can clean it up



Making a model

- Defining a transition process

$$\begin{bmatrix} x_t \\ y_t \end{bmatrix} = \mathbf{A} \cdot \begin{bmatrix} x_{t-1} \\ y_{t-1} \end{bmatrix} + \mathbf{e}_t \rightarrow \mathbf{z}_t = \mathbf{A} \cdot \mathbf{z}_{t-1} + \mathbf{e}_{t-1}$$

Looks familiar?

$$\mathbf{A} = \mathbf{I}$$

$$\mathbf{e}_t \sim \mathcal{N}$$

  - In short, the next input will be a Gaussian random value away from the current input (i.e. not too far)

Starting with it

- Initial estimate will be the first input: $\hat{\mathbf{z}}_1 = \mathbf{z}_1$
- With a given certainty: $\mathbf{P}_1 = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}$
- And two noise models: $\mathbf{Q} = \begin{bmatrix} q_x & 0 \\ 0 & q_y \end{bmatrix}$ $\mathbf{R} = \begin{bmatrix} r_x & 0 \\ 0 & r_y \end{bmatrix}$

Process noise
(how much the data changes)

Measurement noise
(how reliable my measurements are)

Setting up sequential estimates

- Predict future value

$$\mathbf{z}_t^- = \hat{\mathbf{z}}_{t-1} \longleftarrow \textit{We don't expect change other than noise}$$

$$\mathbf{P}^- = \mathbf{P}_{t-1} + \mathbf{Q} \longleftarrow \textit{And we accumulate the uncertainty}$$

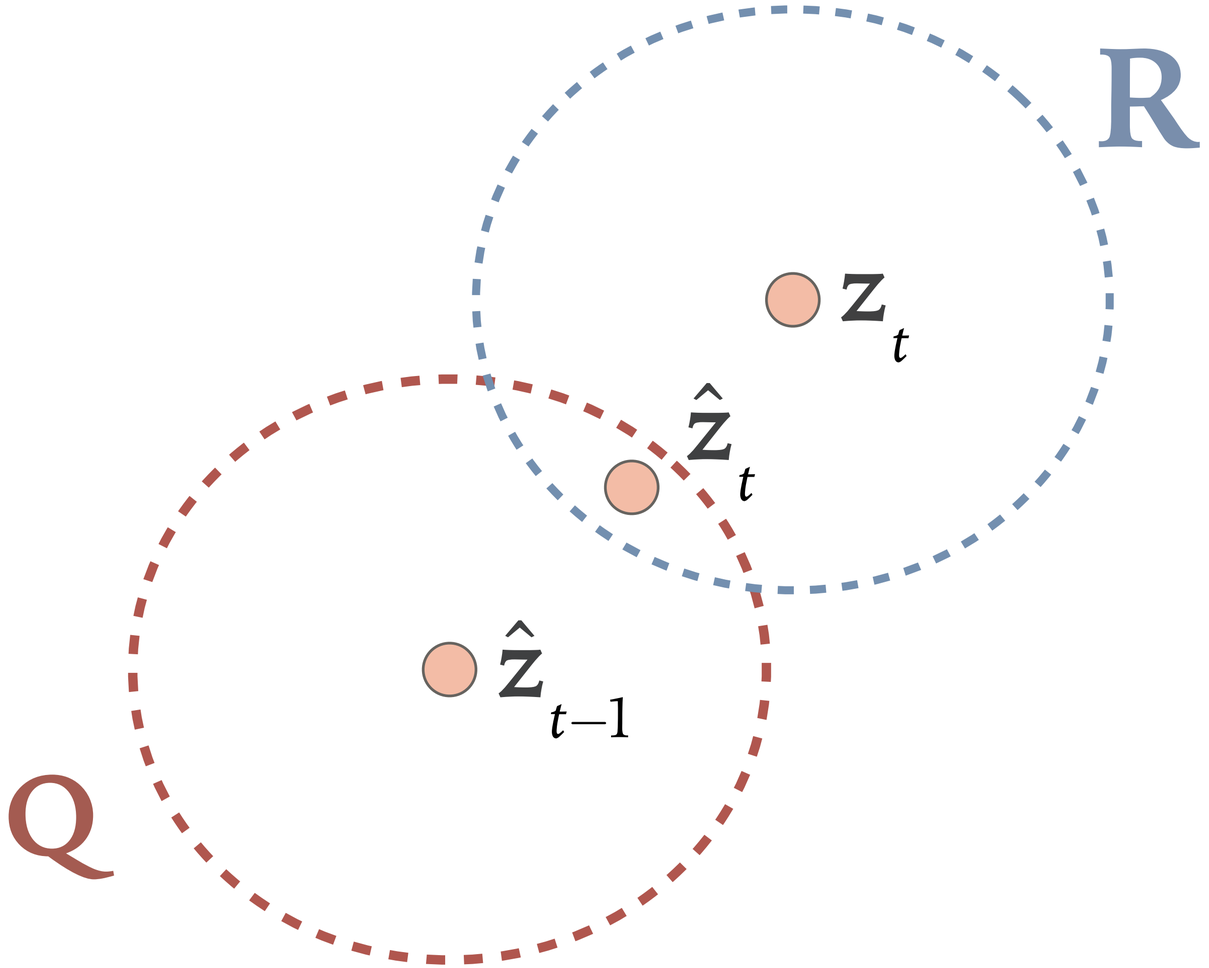
- Correct that estimate given new measurement

$$\mathbf{K} = \mathbf{P}^- \cdot \left(\mathbf{P}^- + \mathbf{R}\right)^{-1} \longleftarrow \textit{Get uncertainty of a consensus estimate}$$

$$\hat{\mathbf{z}}_t = \mathbf{z}_t^- + \mathbf{K} \cdot \left(\mathbf{z}_t - \mathbf{z}_t^-\right) \longleftarrow \textit{And the estimate itself}$$

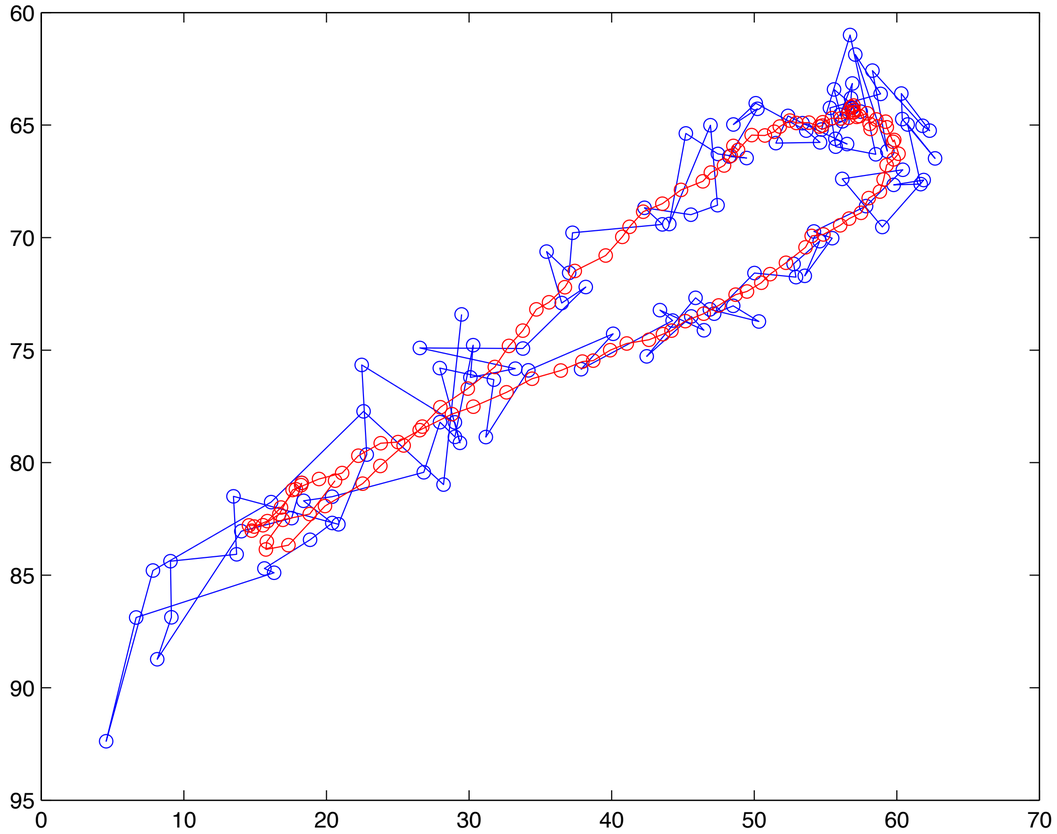
$$\mathbf{P}_t = \left(\mathbf{I} - \mathbf{K}\right) \cdot \mathbf{P}^- \longleftarrow \textit{Get new uncertainty}$$

Single step



Example output

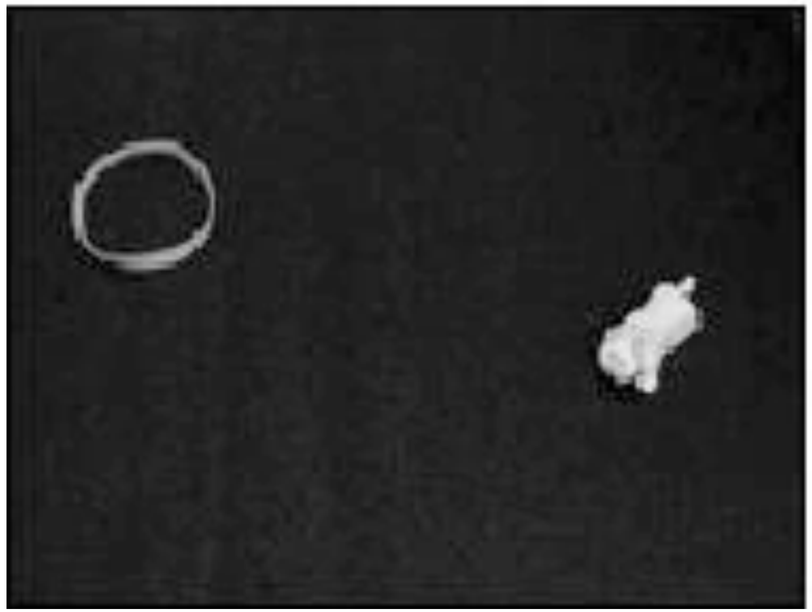
- Estimate is closer to ground truth



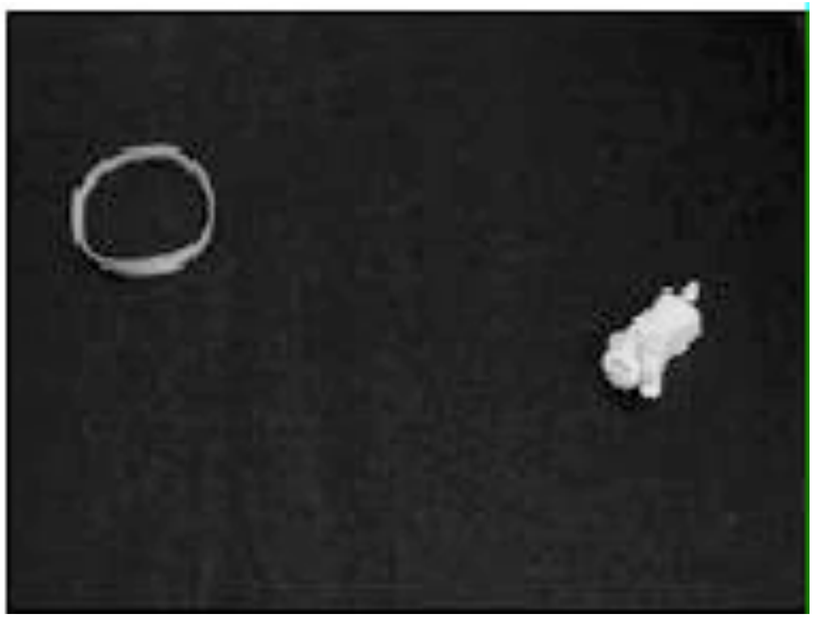
Example output

- Tracking is more smooth
  - But it lags a bit (why?)

Noisy input



Filtered output



Making the model better

- Adding velocity information
  - Internal state representation

$$\mathbf{z}_t = \begin{bmatrix} x_t & y_t & \frac{dx}{dt} & \frac{dy}{dt} \end{bmatrix}^\top$$

- Transition forces constant velocity

$$\mathbf{z}_t = \mathbf{A} \cdot \mathbf{z}_{t-1} + \mathbf{e}_t = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ dx/dt \\ dy/dt \end{bmatrix} + \mathbf{e}_t$$

Measurements

- What we measure is position only:

$$\underset{\text{Observation}}{\mathbf{w}_t} = \mathbf{H} \cdot \underset{\text{State}}{\mathbf{z}_t} + \underset{\text{State noise}}{\mathbf{v}_t} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \cdot \mathbf{z}_t + \mathbf{v}_t$$

- Internal state is not same as the measurement
  - And it has it's own noise

The Kalman filter

- Predict future value

$$\mathbf{z}_t^- = \mathbf{A} \cdot \hat{\mathbf{z}}_{t-1} \quad \longleftarrow \quad \textit{Change state according to model}$$

$$\mathbf{P}^- = \mathbf{A} \cdot \mathbf{P}_{t-1} \cdot \mathbf{A}^\top + \mathbf{Q} \quad \longleftarrow \quad \textit{Accumulate the uncertainty}$$

- Correct that estimate given a new observation $\mathbf{w}_t$

$$\overset{4\times2}{\mathbf{K}} = \overset{4\times4}{\mathbf{P}^-} \cdot \overset{4\times2}{\mathbf{H}^\top} \cdot \left(\overset{2\times4}{\mathbf{H}} \cdot \overset{4\times4}{\mathbf{P}^-} \cdot \overset{4\times2}{\mathbf{H}^\top} + \overset{2\times2}{\mathbf{R}} \right)^{-1} \quad \longleftarrow \quad \textit{Get uncertainty of a consensus estimate}$$

$$\hat{\mathbf{z}}_t = \mathbf{z}_t^- + \mathbf{K} \cdot \left(\mathbf{w}_t - \mathbf{H} \cdot \mathbf{z}_t^- \right) \quad \longleftarrow \quad \textit{And the estimate itself}$$

$$\mathbf{P}_t = \left(\mathbf{I} - \mathbf{K} \cdot \mathbf{H} \right) \cdot \mathbf{P}^- \quad \longleftarrow \quad \textit{Get new uncertainty}$$

More elaborate extensions

- Add acceleration, etc ...
  - More complex internal state
  - More accurate models given various inputs
  - Nonlinear version (extended and unscented Kalman filter)
- Use this to track moving processes
  - Missile over an ocean
  - Finger over a touchscreen
  - Cars on the highway
  - Head movements of a VR headset
  - ...

An example

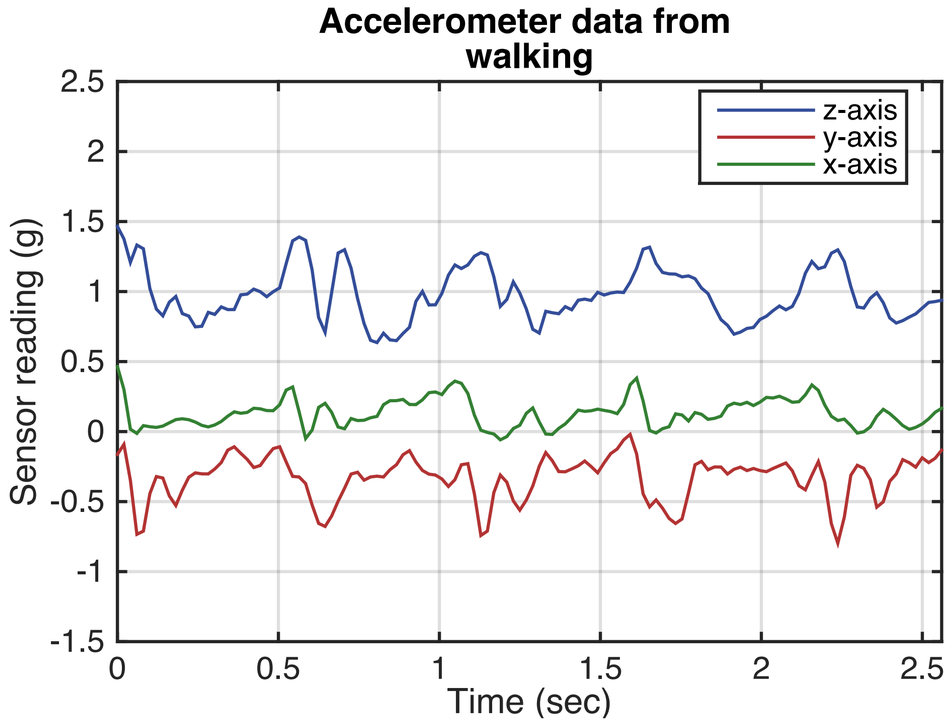
In the real world

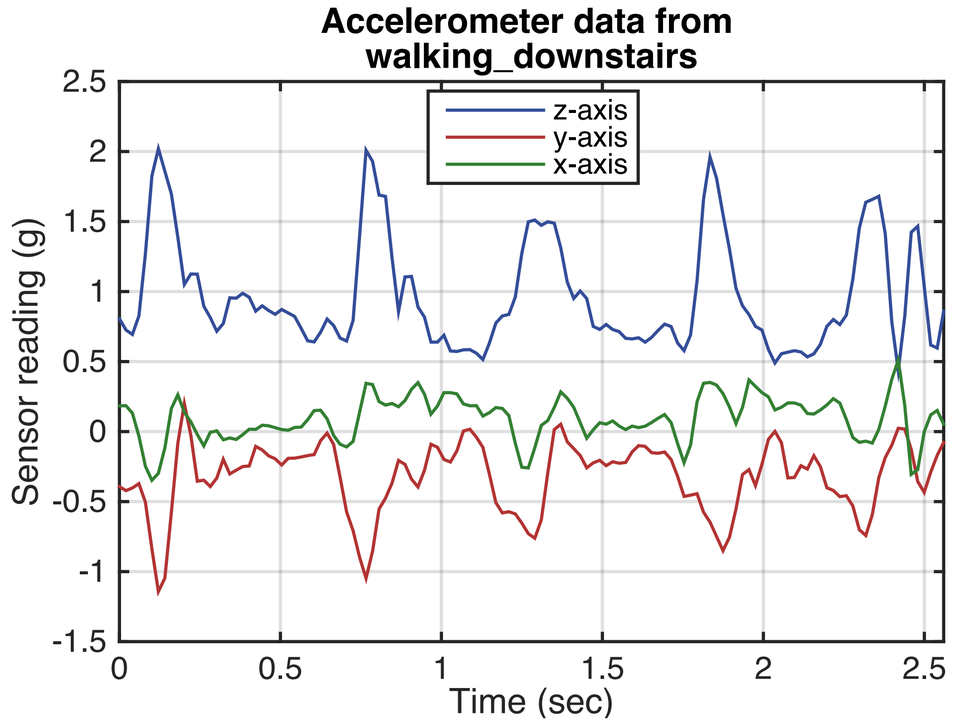
Using dynamical systems to classify

- We can also classify sequences using dynamical models
  - So far we only did prediction, tracking, and smoothing
- Dynamical models can be fit on training sequences
  - And evaluated on new sequences that we want to classify
- Goodness of fit (for now) can be used for classification

An example

- 3D Accelerometer data from a cell phone
  - Classes are the user's activity (sitting, walking, going upstairs, etc)
    - Each activity has a unique temporal pattern





The VAR model (Vector AutoRegression)

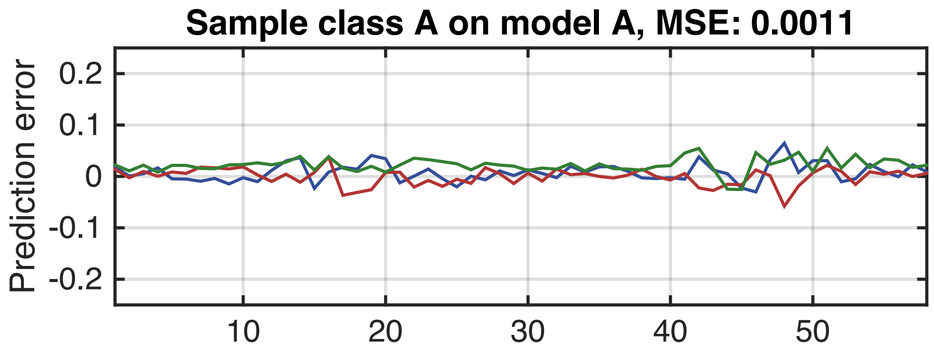
- We can make a linear model to predict future samples using the past

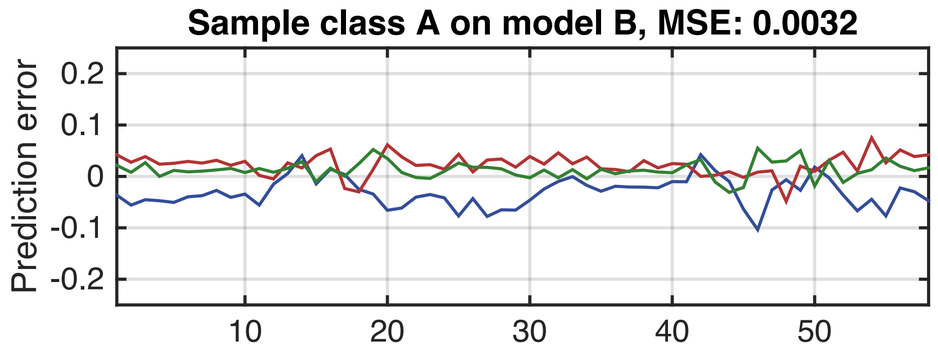
$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ z_{t+1} \end{bmatrix} = \mathbf{A} \cdot \begin{bmatrix} x_t \\ y_t \\ z_t \\ 1 \end{bmatrix} + \mathbf{e}_t$$

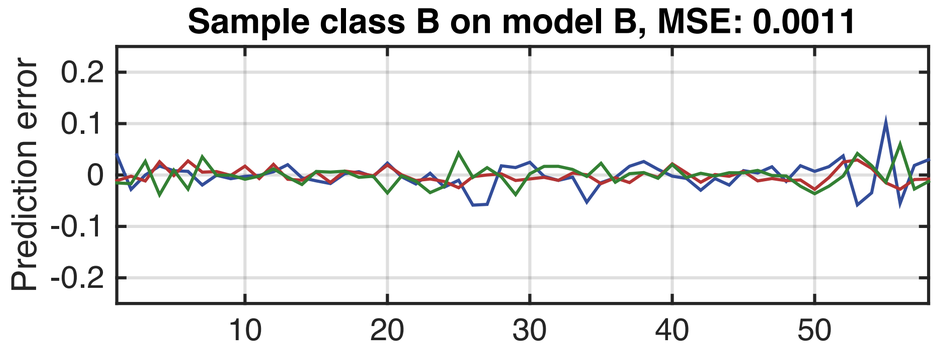
- Matrix $\mathbf{A}$ should be unique to each class
  - Captures the temporal structure of each class
  - Can easily estimate $\mathbf{A}$ with e.g. least squares

Predicting new inputs

- Get each model's prediction error on new samples
  - Model with least error belongs to the same class as input
    - Can also reformulate probabilistically to get likelihoods instead







One big happy family

ICA

$$\mathbf{A} = 0 \Rightarrow \mathbf{x} = \mathrm{g}\left(\mathcal{N}(0, \mathbf{Q})\right)$$

$$\mathbf{y} = \mathbf{C} \cdot \mathbf{x} + \mathcal{N}(0, \mathbf{R})$$

PCA

$$\mathbf{A} = 0 \Rightarrow \mathbf{x} = \mathcal{N}(0, \mathbf{Q})$$

$$\mathbf{y} = \mathbf{C} \cdot \mathbf{x} + \mathcal{N}(0, \mathbf{R})$$

LDS

$$\mathbf{x}_{t+1} = \mathbf{A} \cdot \mathbf{x} + \mathcal{N}(0, \mathbf{Q})$$

$$\mathbf{y}_t = \mathbf{C} \cdot \mathbf{x}_t + \mathcal{N}(0, \mathbf{R})$$

HMM

$$\mathbf{x}_{t+1} = \mathrm{WTA}\left[\mathbf{A} \cdot \mathbf{x} + \mathcal{N}(0, \mathbf{Q})\right]$$

$$\mathbf{y}_t = \mathbf{C} \cdot \mathbf{x}_t + \mathcal{N}(0, \mathbf{R})$$

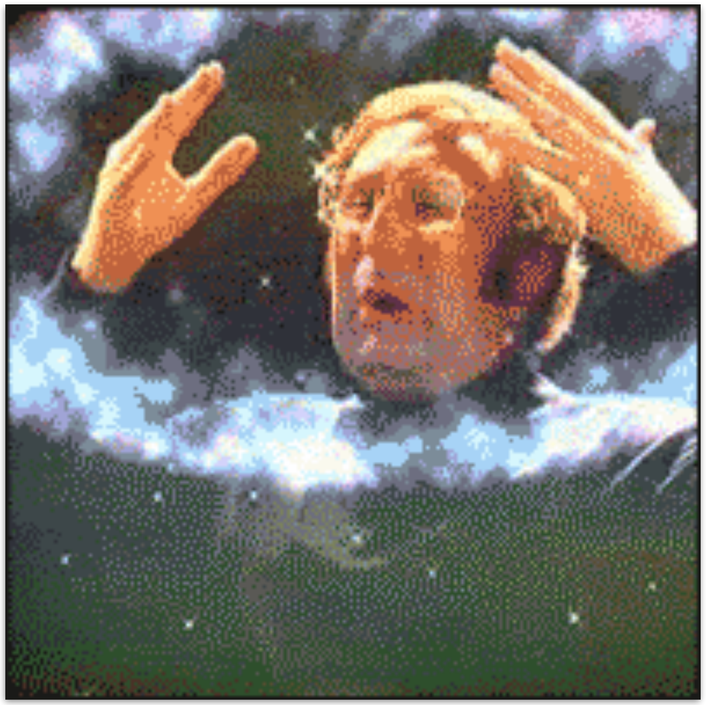
k-means / GMM

$$\mathbf{A} = 0 \Rightarrow \mathbf{x} = \mathrm{WTA}\left[\mathcal{N}(\mu, \mathbf{Q})\right]$$

$$\mathbf{y} = \mathbf{C} \cdot \mathbf{x} + \mathcal{N}(0, \mathbf{R})$$

So everything so far has been the same model!

- Like I said in the beginning of this class, DSP and ML are pretty much the same equation over and over



Recap

- Missing data techniques
  - AR models → using temporal dynamics
  - NN, SVD, NMF → using context information
- Tracking and prediction
  - Kalman filter *et al.*
- The broader Linear Dynamical System family

Next week

- Source separation
  - Array methods using machine learning
  - Multi-channel signal separation methods

Reading

- Missing data in audio
  - http://www-sigproc.eng.cam.ac.uk/~sjg/springer/index.html
  - http://paris.cs.illinois.edu/pubs/smaragdis-jsps10.pdf
- Inpainting (and more) in images
  - http://www.cs.princeton.edu/gfx/pubs/Barnes_2009_PAR/index.php
- The Kalman filter
  - http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf
- LDS
  - http://cs.nyu.edu/~roweis/papers/lds.pdf

Date night!

- If you do not have a project partner (or want more partners) hang around in the classroom for a few more minutes so you get to meet each other