Missing Data and Dynamical Systems
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

- **Not missing at random (NMAR)**
  - None of the above
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 learn the coefficients \( a \) using various methods
Rewriting the model

- Linear algebra notation

\[ e = A \cdot x \]

\[
A = \begin{bmatrix}
-a_p & \cdots & -a_1 & 1 & 0 & 0 & \cdots & 0 & 0 \\
0 & -a_p & \cdots & -a_1 & 1 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & -a_p & \cdots & -a_1 & 1 & 0 & 0 \\
0 & \cdots & 0 & 0 & -a_p & \cdots & -a_1 & 1 & 0 \\
0 & 0 & \cdots & 0 & 0 & -a_p & \cdots & -a_1 & 1
\end{bmatrix}
\]
Filling gaps

- Model the input sequence as:

\[ e = A \cdot x = A \cdot \left( U \cdot x_u + K \cdot x_k \right) = A_u \cdot x_u + A_k \cdot x_k \]

- \( x_u \) and \( x_k \) are the unknown and known samples
- \( U \) and \( K \) are repositioning matrices, e.g.:

\[
\begin{bmatrix}
  x_1 \\
  x_2 \\
  x_3 \\
  x_4 \\
\end{bmatrix} =
\begin{bmatrix}
  0 & 0 \\
  1 & 0 \\
  0 & 1 \\
  0 & 0 \\
\end{bmatrix}
\begin{bmatrix}
  x_2 \\
  x_3 \\
\end{bmatrix}
+
\begin{bmatrix}
  1 & 0 \\
  0 & 0 \\
  0 & 0 \\
  0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
  x_1 \\
  x_4 \\
\end{bmatrix}
\]

Unknown

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 \( x_u \) so that we minimize \( e \)
  - i.e. fill in the blanks with least unpredictable samples

\[
A_u \cdot x_u + A_k \cdot x_k = e
\]
Solving for the gaps

- Set the derivative to zero

\[
\frac{\partial e^\top \cdot e}{\partial x_u} = 2e^\top \frac{\partial e}{\partial x_u} = 2 \left( A \cdot x_u + A_k \cdot x_k \right)^\top \cdot A_u = 0
\]

- Solve for unknown samples

\[
\hat{x}_u = -\left( A_u^\top \cdot A_u \right)^{-1} \cdot A_u^\top \cdot A_k \cdot x_k = -A_u^+ \cdot A_k \cdot x_k
\]
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

\[
\begin{bmatrix}
\uparrow & \uparrow & \uparrow & ? & ? & \uparrow & \uparrow & \uparrow \\
\downarrow & \downarrow & \downarrow & ? & ? & \downarrow & \downarrow & \downarrow \\
\end{bmatrix}
\]

- How do we find the close match?
In action

- First iteration
In action

• Second iteration
In action

- Third iteration
In action

• Fourth iteration
In action

• Final iteration
Some examples

Speech

Music

Sound effects
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
The 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 values
    \[ x_u = E\{x\} \quad \text{or} \quad x_u \sim \mathcal{N}(\mu, \sigma^2) \quad \text{or} \ldots \]
  • Perform an SVD approximation of the data
    \[ X \approx U \cdot S \cdot V^\top \]
  • Replace missing data with SVD approximation
    \[ x_u = \left( U \cdot S \cdot V^\top \right)_u \]
• 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

*Input*  |  *SVD*  |  *NMF*
Learning from outside

- Bandwidth expansion
  - Learn model from other recordings

Input

SVD

NMF
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 is what I think is right
  - Variance is an indication of how sure I am
More uncertainty

- My drunken friend’s estimate is unreliable
- Therefore his 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

\[ N(m, s) \propto N(\mu_1, \sigma_1)N(\mu_2, \sigma_2) \]

\[ K = \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \]

\[ m = \mu_1 + K(\mu_2 - \mu_1) \]

\[ s = (1 - K)\sigma_1^2 \]
Serializing this process

• Assume a noisy time series

\[ z_t = z_{t-1} + e_t \]

\[ e_t \sim \mathcal{N} \]

• Can we use that consensus idea for removing noise in successive values?

• What is the model here?
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
- Let's see how we can clean it up
Making a model

• Defining a transition process
  • Looks familiar?

\[
\begin{bmatrix}
    x_t \\
    y_t
\end{bmatrix}
= A \cdot \begin{bmatrix}
    x_{t-1} \\
    y_{t-1}
\end{bmatrix} + e_t
\rightarrow z_t = A \cdot z_{t-1} + e_{t-1}
\]

\[A = I\]

\[e_t \sim \mathcal{N}\]

• In short, the next input will be a random value away from the current input
Starting with it

- Initial estimate will be the first input: $\hat{z}_1 = z_1$

- With a given certainty: $P_1 = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}$

- And two noise models: $Q = \begin{bmatrix} q_x & 0 \\ 0 & q_y \end{bmatrix}$
  $R = \begin{bmatrix} r_x & 0 \\ 0 & r_y \end{bmatrix}$

  - Process noise (how much the data changes)
  - Measurement noise (how reliable my measurements are)
Setting up sequential estimates

- **Predict future value**
  \[
  z_t^- = \hat{z}_{t-1} \\
  P_t^- = P_{t-1} + Q
  \]
  We don’t expect change other than noise

- **Correct that estimate given new input**
  \[
  K = P^- \cdot \left( P^- + R \right)^{-1} \\
  \hat{z}_t = z_t^- + K \cdot \left( z_t - z_t^- \right) \\
  P_t = \left( I - K \right) \cdot P^-
  \]
  Get uncertainty of a consensus estimate
  And the estimate itself
  Get new uncertainty
Single step

\[ \hat{Z}_t, Z_t, \hat{Z}_{t-1} \]
Example output

- Estimate is closer to ground truth
Example output

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

- Adding velocity information
  - Internal state representation
    \[
    z_t = \begin{bmatrix}
    x_t & y_t & \frac{dx}{dt} & \frac{dy}{dt}
    \end{bmatrix}^T
    \]

- Transition forces constant velocity
  \[
  z_t = A \cdot z_{t-1} + e_t = \begin{bmatrix}
  1 & 0 & dt & 0 \\
  0 & 1 & 0 & dt \\
  0 & 0 & 1 & 0 \\
  0 & 0 & 0 & 1
  \end{bmatrix}
  \begin{bmatrix}
  x_{t-1} \\
  y_{t-1} \\
  \frac{dx}{dt} \\
  \frac{dy}{dt}
  \end{bmatrix}
  \]
Measurements

• What we measure is position only:

\[ w_t = H \cdot z_t + v_t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \cdot z_t + v_t \]

• Internal state is not same as measurement
  • And it has it’s own noise
The Kalman filter

- Predict future value
  \[ z^- = A \cdot \hat{z}_{t-1} \]  \[ P^- = A \cdot P_{t-1} \cdot A^\top + Q \]  \text{Change state according to model}  
  \text{Accumulate the uncertainty}

- Correct that estimate given new input
  \[ K = P^- \cdot H^\top \left( H \cdot P^- \cdot H^\top + R \right)^{-1} \]  \text{Get uncertainty of a consensus estimate}
  \[ \hat{z}_t = z^- + K \left( w_t - H \cdot z^- \right) \]  \text{And the estimate itself}
  \[ P_t = \left( I - K \cdot H \right) \cdot P^- \]  \text{Get new uncertainty}
More elaborate extensions

• Add acceleration, etc ...
  • More complex internal state
  • More accurate models given various inputs

• Use this to track moving processes
  • Missile over an ocean
  • Finger over a touchscreen
  • Cars on the highway
  • ...

And yet more extensions

- Non-linear processes
  \[ z_t = f(z_{t-1}) + e_t, \quad w_t = h(w_{t-1}) + v_t \]
  - Extended Kalman filter
  - Unscented Kalman filter (for very nonlinear)

- Particle filters
  - Sampling based method
  - Bypasses Gaussian assumptions
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 user activity (sitting, walking, going upstairs, etc).
  - Each activity has a unique temporal pattern
The VAR model \((\text{Vector AutoRegression})\)

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

\[
\begin{bmatrix}
    x_{t+1} \\
    y_{t+1} \\
    z_{t+1}
\end{bmatrix}
= A \cdot
\begin{bmatrix}
    x_t \\
    y_t \\
    z_t \\
    1
\end{bmatrix} + e_t
\]

- Matrix \(A\) should be unique to each class
  - Captures the temporal structure of each class
  - Can easily estimate \(A\) 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**

\[ A = 0 \Rightarrow x = g\left(\mathcal{N}(0, Q)\right) \]
\[ y = C \cdot x + \mathcal{N}(0, R) \]

**PCA**

\[ A = 0 \Rightarrow x = \mathcal{N}(0, Q) \]
\[ y = C \cdot x + \mathcal{N}(0, R) \]

**LDS**

\[ x_{t+1} = A \cdot x + \mathcal{N}(0, Q) \]
\[ y_t = C \cdot x_t + \mathcal{N}(0, R) \]

**HMM**

\[ x_{t+1} = \text{WTA}\left[A \cdot x + \mathcal{N}(0, Q)\right] \]
\[ y_t = C \cdot x_t + \mathcal{N}(0, R) \]

**GMM / k-means / VQ**

\[ A = 0 \Rightarrow x = \text{WTA}\left[\mathcal{N}(\mu, Q)\right] \]
\[ y = C \cdot x + \mathcal{N}(0, R) \]
So it’s all really 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
  • Monophonic signal separation methods
Reading

- Missing data in audio
  - http://www-sigproc.eng.cam.ac.uk/~sjg/springer/index.html

- Inpainting (and more) in images

- The Kalman filter

- LDS