Factor Graphs for Time Series Structured Optimization

Class 23. 20 Nov 2014
Instructor: Gary Overett
**Machine Learning Context**

...according to infallible Wikipedia!

### Machine learning and data mining

#### Problems
- Classification
- Clustering
- Regression
- Anomaly detection
- Association rules
- Reinforcement learning
- Structured prediction
  - Feature learning
  - Online learning
  - Semi-supervised learning
  - Grammar induction

#### Supervised learning
- (classification • regression)
  - Decision trees
  - Ensembles (Bagging, Boosting, Random forest)
  - $k$-NN
  - Linear regression
  - Naive Bayes
  - Neural networks
  - Logistic regression
  - Perceptron
  - Support vector machine (SVM)
  - Relevance vector machine (RVM)

#### Clustering
- BIRCH
- Hierarchical
- $k$-means
- Expectation-maximization (EM)
- DBSCAN
- OPTICS
- Mean-shift

#### Dimensionality reduction
- Factor analysis
- CCA
- ICA
- LDA
- NMF
- PCA
- t-SNE

#### Structured prediction
- Graphical models (Bayes net, CRF, HMM)

#### Anomaly detection
- $k$-NN
- Local outlier factor

#### Neural nets
- Autoencoder
- Deep learning
- Multilayer perceptron
- RNN
- Restricted Boltzmann machine
- SOM
- Convolutional neural network

#### Theory
- Bias-variance dilemma
- Computational learning theory
- Empirical risk minimization
- PAC learning
- Statistical learning
- VC theory

Machine Learning Context

...according to infallible Wikipedia!

Source: http://en.wikipedia.org/wiki/Machine_learning
Where we are exploring today?

### Machine learning and data mining

- **Supervised learning**
  - Classification (decision trees, ensembles, boosting, logistic regression, k-NN, linear regression, naive Bayes)
  - Regression (neural networks, support vector machines, relevance vector machines)

- **Clustering**
  - BIRCH, Hierarchical, k-means
  - Expectation-maximization (EM), DBSCAN, OPTICS, Mean-shift

- **Dimensionality reduction**
  - Factor analysis, CCA, ICA, LDA, NMF, PCA, t-SNE

- **Structured prediction**
  - Graphical models (Bayes net, CRF, HMM)

- **Anomaly detection**
  - k-NN, Local outlier factor

- **Neural nets**
  - Autoencoder, Deep learning, Multilayer perceptron, RNN, Restricted Boltzmann machine, SOM, Convolutional neural network

- **Theory**
  - Bias-variance dilemma, Computational learning theory, Empirical risk minimization, PAC learning, Statistical learning, VC theory

---

What might you know about your “signal(s)”?

• Gaussian Distribution
• Made up of Independent Components
• Face-like
• Clustered
• Sparse
• Locally Smooth (or should be!)
What might you know about your “signal(s)”?

- Gaussian Distribution
- Made up of Independent Components
- Face-like
- Clustered
- Sparse
- Locally Smooth (or should be!)
- Choose or derive your constraints!
The more I know about my signal (or the signal I am trying to estimate) the more constraints I’m able to reasonably apply!
## General vs. Domain Specific Constraints

<table>
<thead>
<tr>
<th>General</th>
<th>Domain Specific</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Gaussian</td>
<td>• N-Gram Speech Model (cat-sat-on-the-?-mat)</td>
</tr>
<tr>
<td>• Independent</td>
<td>• Connectedness (Hand-connected-to-arm etc.)</td>
</tr>
<tr>
<td>• Smooth</td>
<td>• Ballistic Trajectory Prediction</td>
</tr>
<tr>
<td>• Non-Negative</td>
<td>• Vehicle Motion Model (cars-go-forward/backwards not</td>
</tr>
<tr>
<td>• Sparse</td>
<td>sideways)</td>
</tr>
</tbody>
</table>

20 Nov 2014
Consider the following ‘signal’

We want to know the position of a survey vehicle in the real world.

Given the following time-series information:
• GPS location data at 1Hz (may suffer dropout)
• Wheel Encoder Odometry Measurements every 2m
• Inertial Sensor Measurements (Gyroscope) every 2m
• Basic Vehicle Geometry (wheel base width etc.)
Vehicle Localization (Pose) Estimation

\[ X = (x, y, \theta) \]

- GPS
- Odometry
- Gyroscope
- Vehicle Motion Constraints
Use case: Vehicle/Camera Pose for Asset Localization
Kalman Filter : Fail!
Kalman Filter

\[ X_{k-1} \quad Z_{k-1} \quad X_k \quad Z_k \]
Certainty Over Time (Odo+GPS)

Credit: “Factor Graphs and GTSAM: A hands on introduction”, Frank Dellaert
Certainty Over Time (Odo only)

Credit: “Factor Graphs and GTSAM: A hands on introduction”, Frank Dellaert
Kalman Filter : Fail!

20 Nov 2014

11755/18979
Survey Vehicle : Signal Behavior

• GPS Suffers Dropout, Noise and Discontinuities
• Odometry is very robust but multiple paths can explain a given odometry
• Gyroscope is has a high precision but ‘drifts’
• Vehicles travel along relatively smooth paths. Normal operation excludes lateral (sideways) motion.
• These constraints suggest a structured (graphical) prediction approach.
Structured Prediction: Factor Graphs

\[ f_1(x_1, x_2; o_1) \]
Factor Graphs

Values – what we want to estimate

\[ f_1(x_1, x_2; o_1) \]
Factor Graphs

Values – what we want to estimate

- Vehicle Pose
- Error Corrected HDD Reader Output
- Image Segmentation Label
- Price of Apple Stock at time t etc.

\[ f_1(x_1, x_2; o_1) \]
Values – what we want to estimate

• Vehicle Pose
• Error Corrected HDD Reader Output
• Image Segmentation Label
• Price of Apple Stock at time $t$ etc.

$$f_1(x_1, x_2; o_1)$$

Factor(s) – a constraint we formulate
Factor Graphs

Values – what we want to estimate

- Vehicle Pose
- Error Corrected HDD Reader Output
- Image Segmentation Label
- Price of Apple Stock at time t etc.

\[ f_1(x_1, x_2; o_1) \]

Factor(s) – a constraint we formulate

- Odometry Constraint
- Error Correcting Code
- Segmentation Prior (Green -> Trees)
- Todays stock price same as yesterday
Factor Graphs

\[ f_1(x_1, x_2; o_1) \]

\[ f_2(x_1; o_2) \]
Factorization

\[ g(x_1, x_2, x_3, x_4, x_5) = f_1(x_1, x_4)f_2(x_1, x_1)f_3(x_2, x_4)f_4(x_2, x_3)f_5(x_3, x_5) \]
Example Factorization: Markov Chain

\[ p(x_1, x_2, x_3) = p(x_1)p(x_2|x_1)p(x_3|x_2) \]
Unifying View of...

• Markov random fields
• Kalman Filters
• HMM’s
• Parity Codes
• Bayesian Networks
Other notations

Original Factor Graph

Forney-Style Factor Graph

Bayesian Network

Markov Random Field

\[ p(u, w, x, y, z) = p(u)p(w)p(x|u, w)p(y|x)p(z|x) \]
• Factors encapsulate constraints being imposed
• Values adjust to become consistent with these constraints
• Usually achieved via “message passing” / gradient descent methods.
Optimization

• Several algorithms are used
• Random vs “Best Guess” initialization
• Gauss-Newton Stepping
• Levenberg-Marquart
• Message Passing Algorithms
• Structure dependent/exploiting optimization strategies
  – ISAM2 (for SLAM problems)
Optimization

- Initialize values to a ‘best guess’/random
- Iterate till convergence
  - Factors messages Values with Error
  - Values messages Factors

\[ X = 7.85 \pm 0.5 \]
Optimization

- Initialize values to a ‘best guess’/random
- Iterate till convergence
  - Factors messages Values with Error
  - Values messages Factors

\[ X = 7.85 \pm 0.5 \]
Optimization

- Initialize values to a ‘best guess’/random
- Iterate till convergence
  - Factors messages Values with Error
  - Values messages Factors

\[ X = 7.85 \pm 0.5 \]
Optimization

- Initialize values to a ‘best guess’/random
- Iterate till convergence
  - Factors messages Values with Error
  - Values messages Factors

X = 7.85\pm 0.5
Optimization

- Initialize values to a ‘best guess’/random
- Iterate till convergence
  - Factors messages Values with Error
  - Values messages Factors

\[ X = 7.85 \pm 0.5 \]
Optimization

- Initialize values to a ‘best guess’/random
- Iterate till convergence
  - Factors messages Values with Error
  - Values messages Factors

\[ X = 7.85 \pm 0.5 \]
Optimization, Factorization and Sparsity

\[ J = \text{Measurement Jacobian} \]
Optimization, Factorization and Sparsity

\[ J = \text{Measurement Jacobian} \]
Optimization, Factorization and Sparsity

\[ J = \text{Measurement Jacobian} \]
Vehicle Pose Estimation

\[ P \in \text{SE}(2) \text{, i.e. } P = \begin{bmatrix} p_x \\ p_y \\ p_{\theta} \end{bmatrix} \]
GTSAM

- Georgia Tech Smoothing and Mapping
- State of the Art in SLAM, Bundle Adjustment, ICP
Figure 12: A larger example with about 100 poses and 30 or so landmarks, as produced by gtsam_examples/PlanarSLAMExample_graph.m

Credit: “Factor Graphs and GTSAM: A hands on introduction”, Frank Dellaert
SLAM versus Localization

• No “landmarks”, rather just GPS readings
• Odometry is modeled via similar means
• Vehicle Model is far more constrained in comparison to a typical robotic agent
ARRB Vehicle Introduction

• GPS @ 1Hz
• Gyroscope, Acceleration, Odometry @ 2.0meters
  – 2-axis gyro and 2-D accel leading to somewhat under constrained problem
• Images @ 2.0-5.0 meters (unsynchronised to above)
• Random Noise including non-real-time OS errors so unrestrained arbitrary corruption of the data is possible!
ARRB Vehicle Introduction
ARRB Vehicle Introduction

- GPS @ 1Hz
- Gyroscope, Acceleration, Odometry @ 2.0meters
  - 2-axis gyro and 2-D accel leading to somewhat under constrained problem
- Images @ 2.0-5.0 meters (unsynchronised to above)
- Random Noise including non-real-time OS errors so unrestrained arbitrary corruption of the data is possible!
- This is the kind of great research problem that comes about on the back of greatly incompetent design!
Pose Estimation : GPS Factor

\[ P_1 \xrightarrow{} P_2 \xrightarrow{} P_3 \xrightarrow{} P_4 \xrightarrow{} P_5 \]

\[ F_{G1} \xrightarrow{} F_{G2} \xrightarrow{} F_{G3} \xrightarrow{} F_{G4} \xrightarrow{} F_{G5} \]
Pose Estimation : GPS Factor

\[ F_G(P; G) = \exp \left\{ -\frac{1}{2} \| h_G(P) - G \|^2_\Sigma \right\} \]
Pose Estimation : GPS Factor

\[ F_G(P; G) = \exp \left\{ -\frac{1}{2} \| h_G(P) - G \|_\Sigma^2 \right\} \]

Error : \[ E_G(P) \triangleq h_G(P) - G \]
Pose Estimation: GPS Factor

\[ F_G(P; G) = \exp \left\{ -\frac{1}{2} \| h_G(P) - G \|^2_\Sigma \right\} \]

Error: \[ E_G(P) \triangleq h_G(P) - G \]

Jacobian: \[ H_G = \begin{bmatrix}
\frac{\delta E_{Gx}}{\delta P_x} & \frac{\delta E_{Gx}}{\delta P_y} & \frac{\delta E_{Gx}}{\delta P_\theta} \\
\frac{\delta E_{Gy}}{\delta P_x} & \frac{\delta E_{Gy}}{\delta P_y} & \frac{\delta E_{Gy}}{\delta P_\theta}
\end{bmatrix} \]
Pose Estimation : GPS Factor

\[ F_G(P; G) = \exp \left\{ -\frac{1}{2} \|h_G(P) - G\|_2^2 \right\} \]

Error : \( E_G(P) \triangleq h_G(P) - G \)

Jacobian : \( H_G = \begin{bmatrix}
\frac{\delta E_G x}{\delta P x} & \frac{\delta E_G x}{\delta P y} & \frac{\delta E_G x}{\delta P \theta} \\
\frac{\delta E_G y}{\delta P x} & \frac{\delta E_G y}{\delta P y} & \frac{\delta E_G y}{\delta P \theta}
\end{bmatrix} \)
Pose Estimation: GPS Factor

\[ F_G(P; G) = \exp \left\{ -\frac{1}{2} \| h_G(P) - G \|_\Sigma^2 \right\} \]

Error: \[ E_G(P) \triangleq h_G(P) - G \]

Jacobian: \[ H_G = \begin{bmatrix}
\frac{\delta E_{Gx}}{\delta P_x} & \frac{\delta E_{Gx}}{\delta P_y} & 0 \\
\frac{\delta E_{Gy}}{\delta P_x} & \frac{\delta E_{Gy}}{\delta P_y} & 0 
\end{bmatrix} \]
Pose Estimation: GPS Factor

\[ F_G(P; G) = \exp \left\{ -\frac{1}{2} \| h_G(P) - G \|^2_\Sigma \right\} \]

Error: \( E_G(P) \triangleq h_G(P) - G \)

Jacobian: \( H_G = \begin{bmatrix}
\frac{\delta E_{G_x}}{\delta P_x} & \frac{\delta E_{G_x}}{\delta P_y} & 0 \\
\frac{\delta E_{G_y}}{\delta P_x} & \frac{\delta E_{G_y}}{\delta P_y} & 0 
\end{bmatrix} \)
**Pose Estimation : GPS Factor**

\[ F_G(P; G) = \exp \left\{ -\frac{1}{2} \| h_G(P) - G \|^2_\Sigma \right\} \]

**Error :** \( E_G(P) \triangleq h_G(P) - G \)

**Jacobian :** \( H_G = \begin{bmatrix} \frac{\delta E_{Gx}}{\delta P_x} & 0 & 0 \\ 0 & \frac{\delta E_{Gy}}{\delta P_y} & 0 \end{bmatrix} \)
Pose Estimation: GPS Factor

\[ F_G(P; G) = \exp \left\{ -\frac{1}{2} \| h_G(P) - G \|^2_\Sigma \right\} \]

Error: \( E_G(P) \triangleq h_G(P) - G \)

Jacobian: \( H_G = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \)
Pose Estimation: GPS Factor

\[ F_G(P; G) = \exp \left\{ -\frac{1}{2} \| h_G(P) - G \|^2_\Sigma \right\} \]

Error: \( E_G(P) \triangleq h_G(P) - G \)

Jacobian: \( H_G = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \)

Covariance: \( \Sigma_{E_G} = \begin{bmatrix} \sigma_x \\ \sigma_y \end{bmatrix} = \begin{bmatrix} 10m \\ 10m \end{bmatrix} \)
Pose Estimation : GPS Factor

\[ F_G(P; G) = \exp \left\{ -\frac{1}{2} \| h_G(P) - G \|^2 \sigma \right\} \]

Error : \( E_G(P) \triangleq h_G(P) - G \)

Jacobian : \( H_G = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \)

Covariance : \( \Sigma_{E_G} = \begin{bmatrix} \sigma_x \\ \sigma_y \end{bmatrix} = \begin{bmatrix} 10m \\ 10m \end{bmatrix} \)

Once you have defined all these you are ready to code your Factor in GTSAM
Pose Estimation: GPS Discontinuity

$P_1 \rightarrow P_2$

$F_{G1} \rightarrow F_{G2}$

$P_3 \rightarrow P_4 \rightarrow P_5$

$F_{G3} \rightarrow F_{G4} \rightarrow F_{G5}$
Pose Estimation : Vehicle Model Factor

\[ F_{V1} \rightarrow P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4 \rightarrow P_5 \]

\[ F_{G1} \rightarrow F_{G2} \rightarrow F_{G3} \rightarrow F_{G4} \rightarrow F_{G5} \]
Pose Estimation : Vehicle Model Factor

Prediction with binary factor requires more than just pose information. Therefore we augment $P$ with additional variables to track; time, speed and steering angle.

$$P = \begin{bmatrix} x \\ y \\ \theta \\ t \\ s \\ \omega \end{bmatrix}$$
Pose Estimation: Vehicle Model Factor

Prediction with binary factor requires more than just pose information. Therefore we augment $P$ with additional variables to track; time, speed and steering angle.

$$P = \begin{bmatrix} x \\ y \\ \theta \\ t \\ s \\ \omega \end{bmatrix}$$

$$h_x^*(P_1, P_2) = \frac{1}{\omega_1} \sin (\omega_1 s_1(t_2 - t_1))$$

$$h_y^*(P_1, P_2) = \frac{1}{\omega_1} \left(1 - \cos (\omega_1 s_1(t_2 - t_1)) \right)$$

$$h_\theta^*(P_1, P_2) = \omega_1 s_1(t_2 - t_1)$$

$$h_s(P_1, P_2) = s_2 - s_1$$

$$h_\omega(P_1, P_2) = \omega_2 - \omega_1$$
Pose Estimation : Vehicle Model Factor

Prediction with binary factor requires more than just pose information. Therefore we augment P with additional variables to track; time, speed and steering angle.

\[
P = \begin{bmatrix}
x \\
y \\
\theta \\
t \\
s \\
\omega \\
\end{bmatrix}
\]

\[
h_x^*(P_1, P_2) = \frac{1}{\omega_1} \sin (\omega_1 s_1 (t_2 - t_1))
\]

\[
h_y^*(P_1, P_2) = \frac{1}{\omega_1} \left(1 - \cos (\omega_1 s_1 (t_2 - t_1))\right)
\]

\[
h_\theta^*(P_1, P_2) = \omega_1 s_1 (t_2 - t_1)
\]

\[
h_s(P_1, P_2) = s_2 - s_1
\]

\[
h_\omega(P_1, P_2) = \omega_2 - \omega_1
\]

Pose Prediction Model
Prediction with binary factor requires more than just pose information. Therefore we augment P with additional variables to track: time, speed and steering angle.

\[
P = \begin{bmatrix}
x \\
y \\
\theta \\
t \\
s \\
\omega \\
\end{bmatrix}
\]

\[
h^*_x(P_1, P_2) = \frac{1}{\omega_1} \sin(\omega_1 s_1 (t_2 - t_1))
\]

\[
h^*_y(P_1, P_2) = \frac{1}{\omega_1} \left(1 - \cos(\omega_1 s_1 (t_2 - t_1))\right)
\]

\[
h^*_\theta(P_1, P_2) = \omega_1 s_1 (t_2 - t_1)
\]

\[
h_s(P_1, P_2) = s_2 - s_1
\]

\[
h_\omega(P_1, P_2) = \omega_2 - \omega_1
\]

Pose Prediction Model
Pose Estimation: Vehicle Model Factor

Prediction with binary factor requires more than just pose information. Therefore we augment P with additional variables to track; time, speed and steering angle.

\[
P = \begin{bmatrix} x \\ y \\ \theta \\ t \\ s \\ \omega \end{bmatrix}
\]

\[
h_x^*(P_1, P_2) = \frac{1}{\omega_1} \sin(\omega_1 s_1 (t_2 - t_1))
\]

\[
h_y^*(P_1, P_2) = \frac{1}{\omega_1} \left(1 - \cos(\omega_1 s_1 (t_2 - t_1))\right)
\]

\[
h_{\theta}(P_1, P_2) = \omega_1 s_1 (t_2 - t_1)
\]

\[
h_s(P_1, P_2) = s_2 - s_1
\]

\[
h_\omega(P_1, P_2) = \omega_2 - \omega_1
\]
Prediction with binary factor requires more than just pose information. Therefore we augment $P$ with additional variables to track; time, speed and steering angle.

$$P = \begin{bmatrix} x \\ y \\ \theta \\ t \\ s \\ \omega \end{bmatrix}$$

\[
h^*_x(P_1, P_2) \\
h^*_y(P_1, P_2) \\
h^*_\theta(P_1, P_2) \\
h_s(P_1, P_2) \\
h_\omega(P_1, P_2)
\]

And compute the $5 \times 6$ Jacobians for $P_1$ & $P_2$!
Pose Estimation: Vehicle Model Factor

Prediction with binary factor requires more than just pose information. Therefore we augment $P$ with additional variables to track time, speed and steering angle.

$$P = \begin{bmatrix} x \\ y \\ \theta \\ t \\ s \\ \omega \end{bmatrix}$$

$$h_x^*(P_1, P_2)$$
$$h_y^*(P_1, P_2)$$
$$h_\theta^*(P_1, P_2)$$
$$h_s(P_1, P_2)$$
$$h_\omega(P_1, P_2)$$

And compute the $5 \times 6$ Jacobians for $P_1$ & $P_2$!

$$H_{P_i} = \begin{bmatrix} \frac{\delta h_x^*}{\delta x_i} & \frac{\delta h_x^*}{\delta y_i} & \frac{\delta h_x^*}{\delta \theta_i} & \frac{\delta h_x^*}{\delta t_i} & \frac{\delta h_x^*}{\delta s_i} & \frac{\delta h_x^*}{\delta \omega_i} \\
\frac{\delta h_y^*}{\delta x_i} & \frac{\delta h_y^*}{\delta y_i} & \frac{\delta h_y^*}{\delta \theta_i} & \frac{\delta h_y^*}{\delta t_i} & \frac{\delta h_y^*}{\delta s_i} & \frac{\delta h_y^*}{\delta \omega_i} \\
\frac{\delta h_\theta^*}{\delta x_i} & \frac{\delta h_\theta^*}{\delta y_i} & \frac{\delta h_\theta^*}{\delta \theta_i} & \frac{\delta h_\theta^*}{\delta t_i} & \frac{\delta h_\theta^*}{\delta s_i} & \frac{\delta h_\theta^*}{\delta \omega_i} \\
\frac{\delta h_s}{\delta x_i} & \frac{\delta h_s}{\delta y_i} & \frac{\delta h_s}{\delta \theta_i} & \frac{\delta h_s}{\delta t_i} & \frac{\delta h_s}{\delta s_i} & \frac{\delta h_s}{\delta \omega_i} \\
\frac{\delta h_\omega}{\delta x_i} & \frac{\delta h_\omega}{\delta y_i} & \frac{\delta h_\omega}{\delta \theta_i} & \frac{\delta h_\omega}{\delta t_i} & \frac{\delta h_\omega}{\delta s_i} & \frac{\delta h_\omega}{\delta \omega_i} \end{bmatrix}$$

in code!
**Pose Estimation : Vehicle Model Factor**

Prediction with binary factor requires more than just pose information. Therefore we augment $P$ with additional variables to track; time, speed and steering angle.

\[
P = \begin{bmatrix}
x \\ y \\ \theta \\ t \\ s \\ \omega \\
\end{bmatrix}
\]

\[
h^*_x(P_1, P_2) \\
h^*_y(P_1, P_2) \\
h^*_\theta(P_1, P_2) \\
h_s(P_1, P_2) \\
h_\omega(P_1, P_2)
\]

And compute the $5 \times 6$ Jacobians for $P_1 \& P_2$!

or since GTSAM 2.3 use numerical derivatives!
Pose Estimation: GPS+Vehicle...
Pose Estimation: GPS+Vehicle...
Pose Estimation: GPS+Vehicle...

Indeterminate System without time being constrained.
Pose Estimation: GPS + Vehicle + Time
Pose Estimation: GyroOdo Factor (R)

R constrains x, y, θ in P₁ & P₄
Pose Estimation: GyroOdo Factor (R)
Pose Estimation : Stationary Factor (S)
Optimization

- Levenberg-Marquardt, iSAM, iSAM2, others
- Initialization sensitivities can require somewhat complex ‘rules of thumb’ to avoid undesired minima
- Speed can be an issue
- Without ground truth data or manual checking its always possible there are unpleasant surprises in the optimized output
Results: “Poser”
Results: “Poser” - more than 2 mile dropout
Results: “Poser” - tested on > 40,000 miles
Results: “Poser” - tested on > 40,000 miles

2 known fail cases :(
GICP : Kinect Sensor

https://www.youtube.com/watch?v=TY99Y_I_egg
Factor Graphs: Round Up

• Factor Graphs are great for expressing formally defined relationships (e.g. in SLAM we ‘know’ how GPS positions should relate to real world locations)

• Highly General Tool – but its useful for building specific models/capturing prior knowledge you can formalize

• Contrast with Neural Networks – both are graphical models but Neural Networks work to discover the unknown (or you are lazy) relationships as opposed to known ones.
Resources

• GTSAM - [https://collab.cc.gatech.edu/borg/download](https://collab.cc.gatech.edu/borg/download)

• “Factor Graphs and GTSAM: A hands on introduction”, Frank Dellaert

• Michael Kaess – GTSAM contributor now at CMU-RI (see [https://www.ri.cmu.edu/video_view.html?video_id=129&menu_id=387](https://www.ri.cmu.edu/video_view.html?video_id=129&menu_id=387))

• G2O – an alternative library for similar factor graph optimizations