Data Association

We've mostly assumed that we know which feature is which. What if we don't?

\[ \text{estimates of feature position} \]

\[ a \& b \text{ are obs without known feature association} \]

Euclidean nearest-neighbor

- match each obs to closest landmark,
- easy & fast
- doesn't use uncertainties
- could a \& b both be assoc. w the same feature?
  - sometimes, this is sensible. Depends on sensor properties.
  - when it's not, use greedy matching or stable marriage.

Is it a new landmark?

\[ \text{distance/err to best landmark} \]

associate, unsure, do nothing, create new landmark
Data Association (Incremental)

Assume we've been SLAMMING for a while:

\[
\begin{bmatrix} J_1 \\ J_2 \end{bmatrix} \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \quad \text{with} \quad \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}
\]

We now receive a batch of observations. For any particular data association choice, we get more rows:

\[
\begin{bmatrix} J_1 \\ J_2 \end{bmatrix} \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \quad \text{with} \quad \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}
\]

What is the \( \chi^2 \) of our posterior? Easy! Solve:

\[
\begin{bmatrix} J_1 \\ J_2 \end{bmatrix} \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \quad \text{with} \quad \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}
\]

which gives:

\[
x = \left( J_1^T w_1 J_1 + J_2^T w_2 J_2 \right)^{-1} \left( J_1^T w_1 b_1 + J_2^T w_2 b_2 \right)
\]

And \( \chi^2 \) is:

\[
\begin{align*}
\chi^2 &= x^T J_1^T w_1 J_1 x - 2x^T J_1^T w_1 b_1 + b_1^T w_1 b_1 \\
&\quad + x^T J_2^T w_2 J_2 x - 2x^T J_2^T w_2 b_2 + b_2^T w_2 b_2 \\
&= x^T \left( J_1^T w_1 J_1 + J_2^T w_2 J_2 \right) x - 2x^T \left( J_1^T w_1 b_1 + J_2^T w_2 b_2 \right) + b_1^T w_1 b_1
\end{align*}
\]

Be clever in substituting to preserve symmetry...

\[
\chi^2 = (A^{-1} B)^T A (A^{-1} B) - 2 (A^{-1} B)^T B + b_1^T w_1 b_1
\]

\( A^{-1} \) is symmetric...

\[
= B^T A^{-1} B - 2 B^T A^{-1} B + b_1^T w_1 b_1
\]

\( \Rightarrow -B^T A^{-1} B + b_1^T w_1 b_1 \) doesn't depend on \( x \)...

\( A^{-1} = \text{cov. of posterior}, \quad A^{-1} = \text{cov. of posterior} \)

\[ + R^T w_1 b_1, \quad R = \text{residual}\]
We now have a search problem: find data associations that minimize $\chi^2$ of posterior $-B^T A^{-1} B$.

(remember: $J_2$ encodes data association.)

Suppose we have $F$ features in our map, and we observe $Z$ of them. How big is our search space?

$O(Z^F)$ (actually $Z^{F+1}$, but we’ll come back to that.)

"Interpretation Tree"

Simple recursive implementation.

> $A^*$ heuristic?

$h(n) = O$ (assume no cost for subsequent assignments).

> Can we ever prune branches if $\chi^2$ is really improbable?

Uh-Oh #1
Where do new features come from?

Easy! Increase branching factor by 1: possibility that $Z_i$ is a new feature.

Uh-Oh #2
What is the ML data association?

> Every observation is a new feature has no error!
Pose problem a bit differently:

- What is the largest # non-null data associations we can make such that \(X^2\) doesn't exceed a threshold?

Implementation

\[
\text{besth} = \{3\}
\]

\[
\text{JCBB(depth, graph, obs, h)}
\]

\[
\text{}\text{if} \quad \text{depth} = \text{length}(\text{obs}) \quad \text{leaf?}
\]

\[
\text{if} \quad |h| > |\text{besth}|
\]

\[
\text{besth} = h
\]

\[
\text{return.}
\]

\[
\text{for i = 1 : length(graph.features)}
\]

\[
X^2 = \text{compute-chi2(graph, obs, h u \{i \mid \text{depth} \Rightarrow \{i\} \}}
\]

\[
\text{if} \quad X^2 < \text{chi2(DoF)}
\]

\[
\text{JCBB(depth+1, graph, obs, h u \{i \mid \text{depth} \Rightarrow \{i\} \}}
\]

\[
\text{JCBB(depth+1, graph, obs, h \in \{i \mid \text{depth} \Rightarrow \{i\} \}}
\]

Many optimizations possible:

- Do a cheap "individual compatibility" test before computing the slow test, (i.e., could \(z_i \Rightarrow x_i\) with no other data assoc in \(h\)?)

- Don't recurse down new feature path if \(|\text{obs}|-\text{depth} + |h| < |\text{besth}|\)

- Compute-chi2 requires inverting a growing matrix... incremental "trick" can help.

Runtime? still exponential in principle, though culling is often very effective.
Quasi-probabilistic method leaves something to be desired
> why is maximizing # of non-null hypotheses the right thing to do?

Alternative: suppose we have:

\[ p(z \text{ is a new feature}) \]

Now we can appropriately "penalize" new features. New features cost us some constant \( \chi^2 \) amount.

This method works well with FastSLAM too!