Applications of graph theory in robust fitting

(Harbin, 2025)

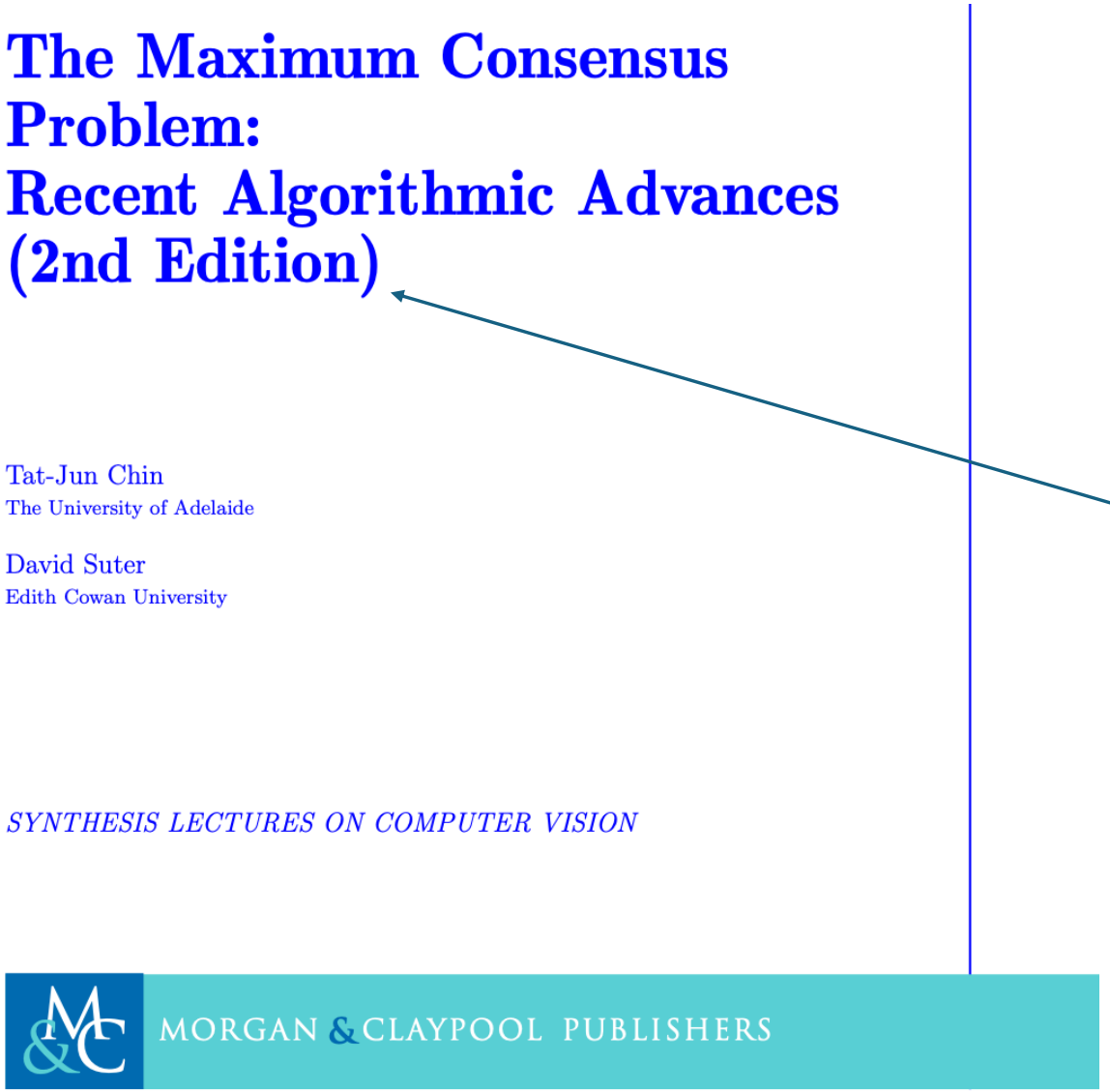
David Suter d.suter@ecu.edu.au

https://ai-ecu.github.io/HIT/HIT2024/index_SoC.html

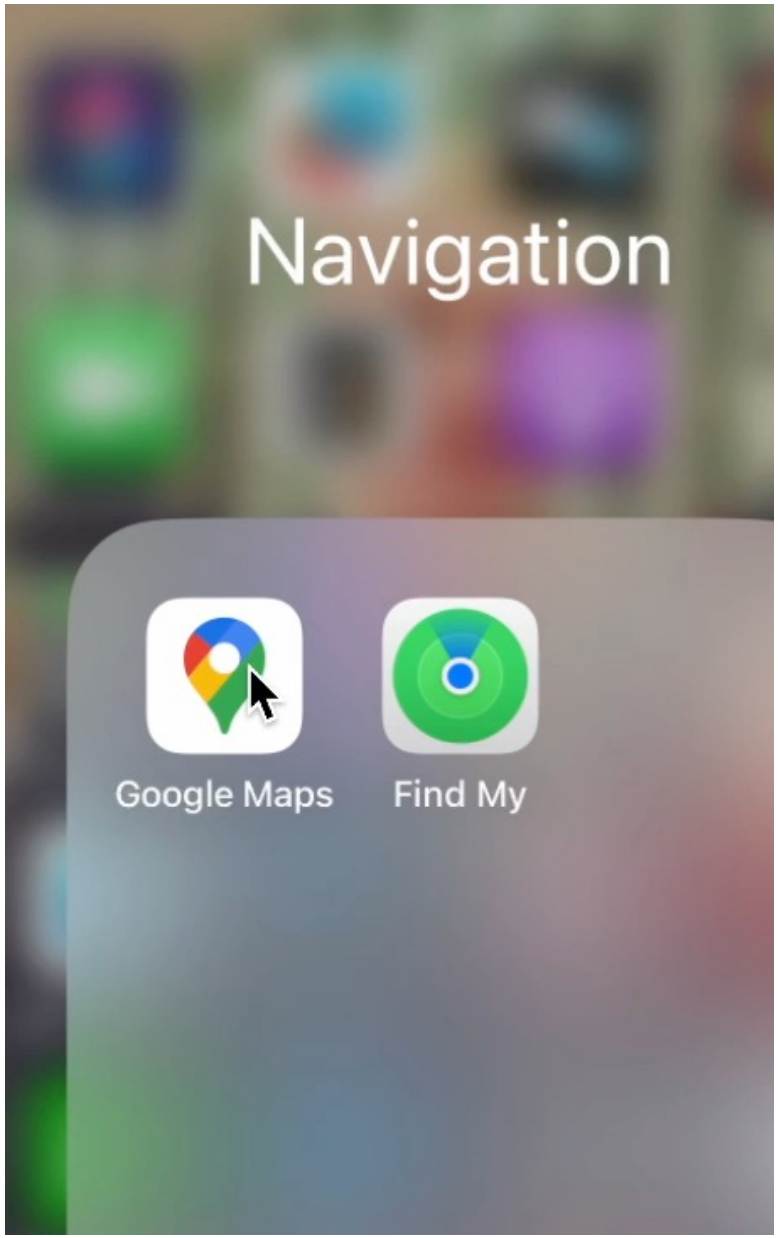
Even though the ideas in this course can be applied more generally – course is influenced by robust fitting

a) by MaxCon criterion

b) To problems in computer vision



Hopefully soon with the publisher – 1st Edition appeared in 2014/15.


Navigation
Google Maps
Find My

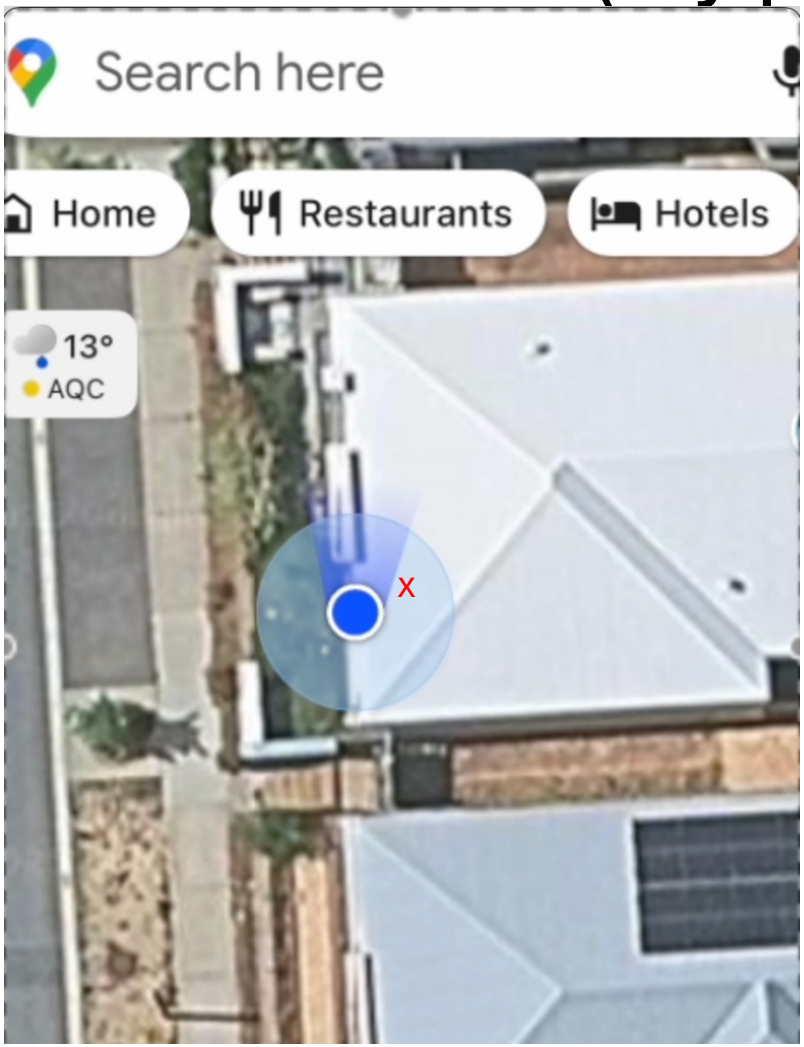
Where am I? 2D location estimation

Where am I? 2D location estimation

Where am I? 2D location estimation

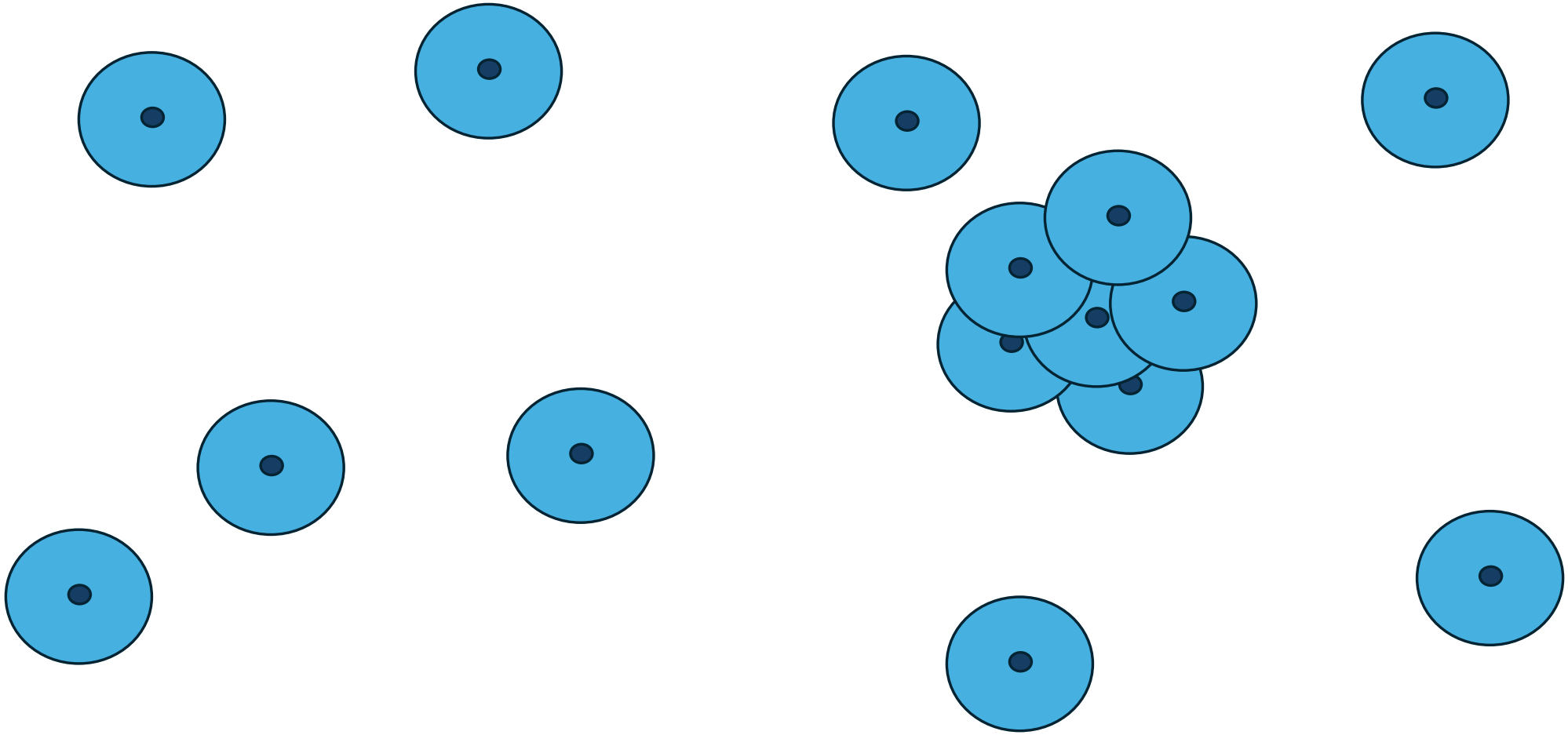
Where am I? 2D location estimation

Where was I? (My phone)

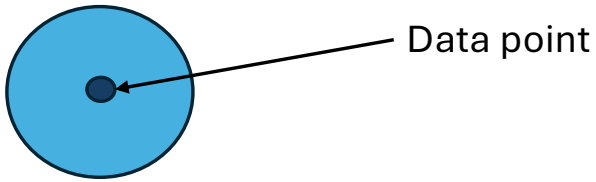


(my best guess from knowledge of where desk is and interior of house)

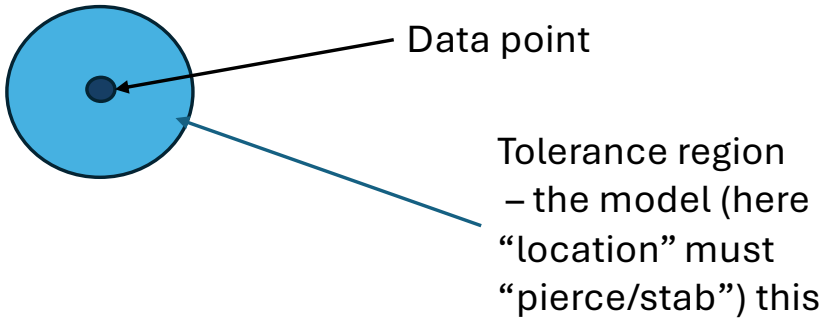
2D location estimation – model problem



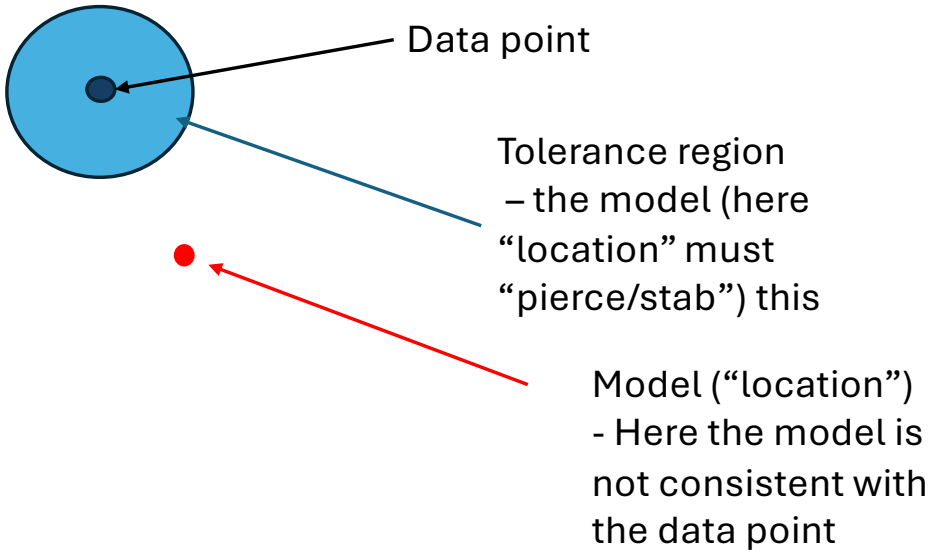
Models/hypotheses, data, and metrics (tolerances)



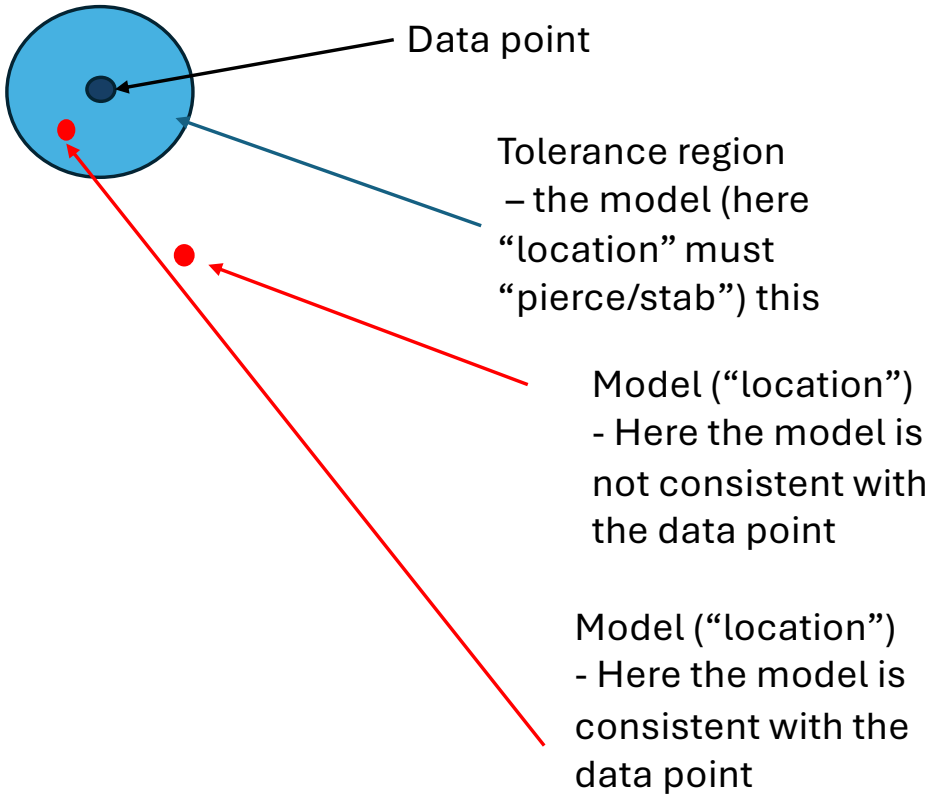
Models/hypotheses, data, and metrics (tolerances)



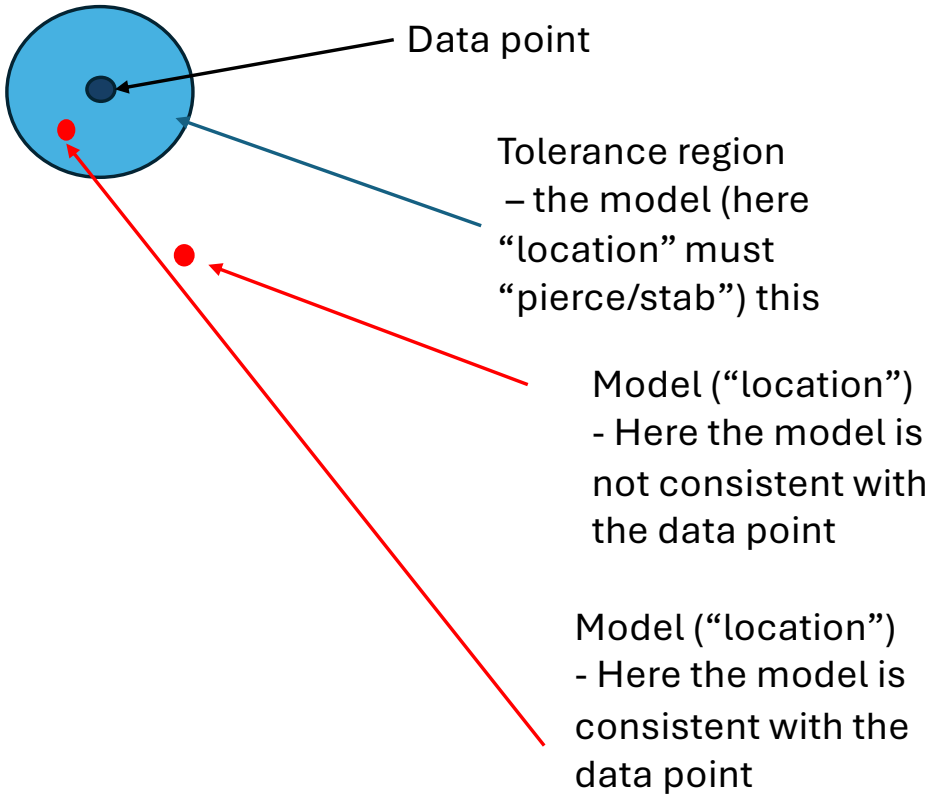
Models/hypotheses, data, and metrics (tolerances)



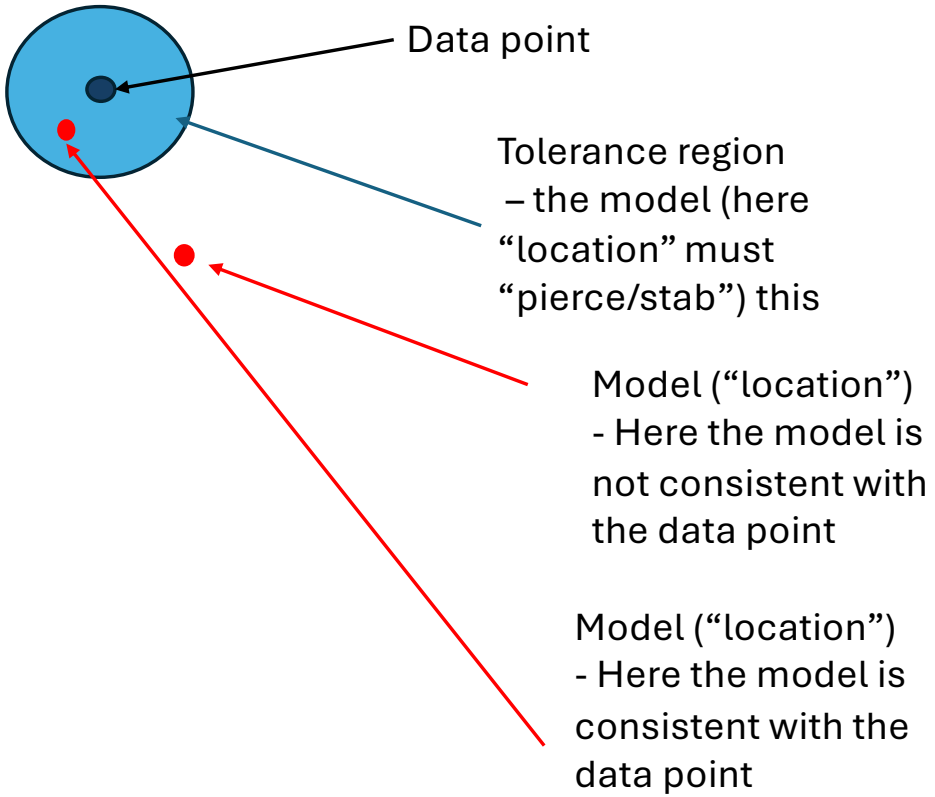
Models/hypotheses, data, and metrics (tolerances)



Models/hypotheses, data, and metrics (tolerances)



Models/hypotheses, data, and metrics (tolerances)



Tolerance region (around a data point) is the set of all models consistent with the data point

Models/hypotheses, data, and metrics (tolerances)

Model – “location” - a point (here in 2D)
Data – “measurement” – here a point in 2D
Metric – how to measure “within tolerance” – here Euclidean metric for 2D
Tolerance region – region around a data point that the model must “pierce/stab”

Can generalize this to other “models”, other data types, other metrics, and other interpretations of “pierce/stab”.

You get a “zoo” of problem-types – this course is about this zoo…..☺

The zoo - of fitting within tolerance problems

- I am biased towards the zoo that can be found in computer vision
- But essentially that's much the same zoo as you find anywhere (perhaps with some more exotic "animals") ☺

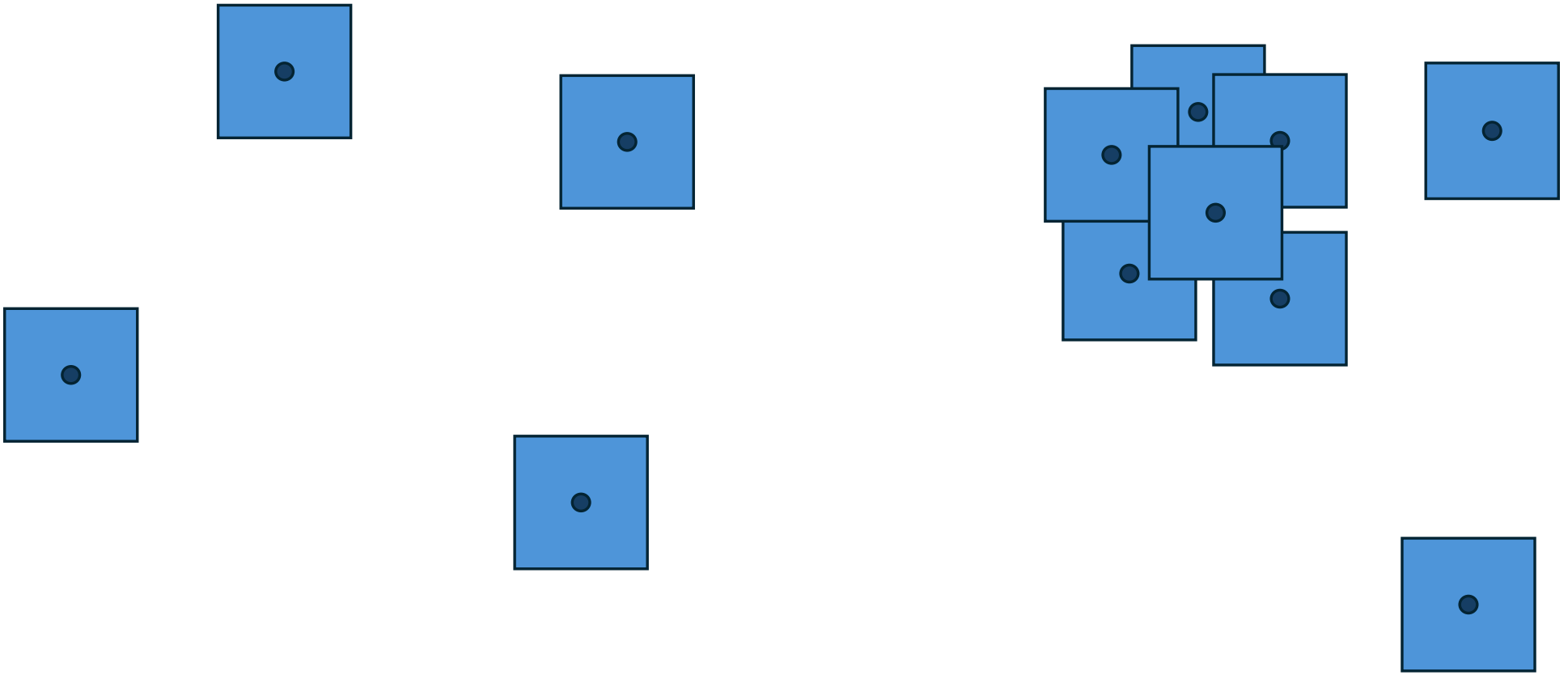
- Let's meet some

Models/hypotheses, data, and metrics (tolerances)

- Data? (points in 2D)
- Metric? (well several are used but the most common are:
  - vertical (y) distance (a vertical interval of some size around each data point)
  - Euclidean distance (circles of some radius around each data point)
  - (probably not very common) Manhattan distance (squares of some size about each data point)
- Model? Line in 2D
- Notion of "pierce/stab" – line must pass through tolerance region

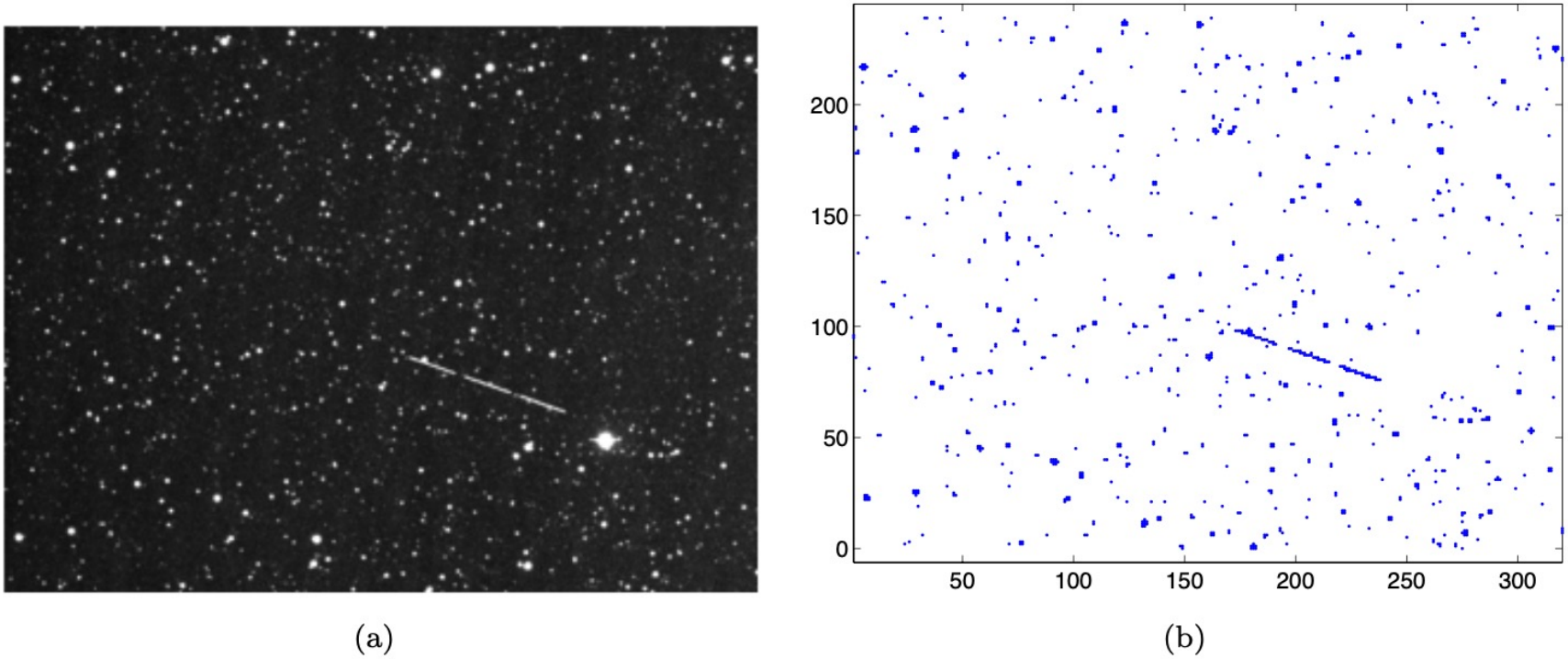
- Note: here we see the "location" examples have a special "quirk" the models and the data are both points. This allows the tolerance region to be both the thing that must be stabbed by the consistent models AND the set of all such models. In general – these are two different things and are usually even in separate spaces "data space" and "model space" (sometimes called parameter space).
- If you want to be even more technical – there are usually different ways to parameterise the model and thus different parameter spaces you might consider……..the zoo gets even more variety (and strange/exotic…and interesting ☺)

2D location estimation – Manhattan distance



(Maybe more appropriate for images/rectangular grid data..”to the nearest pixel”)

"Locate" a line.....



(a) (b)

Figure 1.1: Estimating satellite trajectory by finding linear streaks. (a) Input image containing an orbiting satellite obtained via telescope (courtesy of Defence Science Technology Group, Australia's primary defence science research organisation). (b) A set of points obtained by intensity thresholding, removal of large blobs and centroiding. Observe that there exist a significant number of outliers, *i.e.*, points not lying on the target line.

Sub-zoo of line fitting in 2D

- Ways to parameterise lines
  - m,b y=mx+b 2D parameter space – can't represent vertical lines (ok for regression – so to some extent this gets entangled with choice of metric)
  - R,θ x =r cos(θ) y=r sin(θ)
  - ax+by+c=0
- Metrics – Euclidean, Manhattan……
- Piercing/stabbing – line goes through region defined by metric

(note: as can be complicated – the piercing/stabbing can also be considered by intersection of regions in model space….rather than line stabbing in "data space")

But what does this have to do with AI, Machine Learning, Signal and Image Processing???

(Permission to use photo via Daniel Barath)

Image 1 (x,y,1) <-> Image 2 (x’,y’,1)

(x’,y’,w)^T=H(x,y,1)^T
H – 3x3 homography matrix

(image stitching)

Other “models” R,t (point cloud aignment)
F (3x3 matrix relating geometry of stereo vision)
$x^T Fx=0$
Etc.

Fundamental Matrix estimation

(Common pinhole camera assumption)

$x^T Fx'=0$ for matching pair of points $x=(x_1,x_2,1)$, $x'=(x'_1,x'_2,1)$
Note F is a 3x3 matrix BUT it is scale independent (hence 1 less degree of freedom) and det(F)=0 (hence another reduction in degrees of freedom).
If you "know" F (can extract from matching pairs) then you can derive the relative camera poses (up to a projective transformation). You can then usually use other information to recover Euclidean geometry – up to a scale. It has 7 parameters, effectively, but a nonlinear problem if posed as such.

If you have camera calibration information (or can guess reasonable camera calibration parameters then you work in terms of the Essential matrix – which has the same constraint eqn: $x^T Ex'=0$ where again, the matrix E is singular (it has two equal singular values and one zero singular values). Effectively 5 degrees of freedom. But again nonlinear.

In short, there are in fact a multitude of versions of camera geometry formulations depending on whether one wants to use a "linearized" version (and hence suffer issues that full constraints are not enforced and suffer heteroscedastic noise issues, and also needs "more points" to uniquely determine) or use a nonlinear version (which requires less points, may be statistically much better, but requires complex polynomial solvers).

Fundamental Matrix estimation

(Common pinhole camera assumption)

Here we only focus/illustrate the linearized version(s)

a) Because they are in fact the popular
b) Because they are simpler to understand/analyse (at least from more points of view)
c) Because they more naturally fit a progression of "hyperplane fitting" problems from 1 D location estimation, 1D subspace in 2D regression, affine line fitting in 2D, plane fitting in 3D….and other linearized (or already linear) model fitting
d) With my focus on understand the hypergraph classes associated with computer vision problems – they are *probably* the easiest to understand.

The essence is to simply expand the constraint equation: giving a bilinear constraint; and treating each blinear term as a separate "data variable".
e.g. $x_1x'_1 f_{1,1}$+……+1 $f_{3,3}$ = 0. 9 terms, four linear (one in each of x_1,x_2,x'_1,x'_2), one constant (in terms of the x's) and four homogeneous bilinear ($x_1,x'_1,x_2x'_1,x_1x'_2,x_2x'_2$).

If ALL of the matches were correct AND accurate (AND not degenerate), every set of 9 matches would supply an independent constraint on the 9 entries of F and we would have a correct F (similar story for E) by solving linear equations.

Fundamental Matrix estimation

(Common pinhole camera assumption)

BUT even of the matches were correct, they certainly are not infinitely accurate (small noise) and moreover we haven't enforced det(F)=0 nor used the fact that the scale of F is indeterminate. Both can cause problems (complex interactions with small noise) – and that even assuming the matches are correct!

There have been many variants proposed but a common one is the 8-point algorithm.
Fix ONE of the entries of F to be 1 (which we can do, since the scale is arbitrary - SO LONG as we are not unlucky to fix an entry that *should be* zero.(!)
And solve and then project onto rank 2.

It "can go wrong" for multiple reasons – (picking the wrong F entry to fix, degenerate matching points, inaccurate matches, totally erroneous matches) and the latter is where I have taken most interest – outliers! Various fixes have been proposed (including the "obvious": try multiple times varying different F entries to fix, detecting degenerate configurations, least squares refinement – particularly after using RANSAC (the robust fitting method that introduced the MaxCon criterion for robust fitting!) to remover the biggest issue (typically) – outliers!

The main message for our purposes though is that the above framework more or less treats F estimation as hyperplane fitting in 9D.

Fundamental Matrix estimation

(Common pinhole camera assumption – 8-point linearized method assumed)

Every 8D hyperplane is given by 8 points in general position (ignore for now degeneracy).
But you will always “get an answer” – just as fitting a line to two points in 2D is not a really sensible thing to do when the data has noise.

If you know something about the likely noise size (tolerance to your data), then it is more sensible to fit to 9 or more points and outliers are unlikely (statistically) to fit within tolerance to many other selections of 8 points.

That is why I am interested in up to 9-uniform hypergraphs (and mostly linear models).

Sub-zoo of “Geometric Computer Vision”

- Ways to parameterise homographies, FM, rigid motion, - its complicated – maybe later in the course we’ll illustrate
- Metrics – Euclidean, Manhattan, algebraic……again complicated!
- Piercing/stabbing – complicated…

(data – in most cases the single data point is really a PAIR of matched data – (x,y)(x’y’) is image to image matching; (x,y,z)(x’y’z’) for point cloud to point cloud – so for example a point in R^4, or R^6 for example)

But they are all instances of some “geometric intersection (hyper)graph MaxClique problem.”

Robust Fitting (MaxCon) – Fitting within tolerance
(Given a model – line, plane, hyperplane,... and data points, what is the model that fist more points than any other model? Alternatively, what is the largest subset of points within tolerance of a model)

Problem	(Hyper)Graph class (every instance is a member of this class)
1D location estimation (equal tolerances/unequal tolerances)	Unit interval/Interval
1D subspace (in 2D) regression	Interval
1D line regression in 2D (affine line fitting in 2D)	3-uniform hypergraph (some subclass of – WHICH?) Intersection of ”fat” lines in 2D – but what class is that?
(2D) plane regression in 3D	4-uniform hypergraph (some subclass of – WHICH?) Intersection of fat planes in 3D
..... (7D) hyperplane regression in 8D (linear model/approximation of some projective geometry in computer vision)	9-uniform hypergraph (some subclass of – WHICH?) Intersection of fat hyperplanes in 3D

Note: the above are just *SOME* examples of the models. Some use geometric distance instead of regression distance – some are fitting “element of a group” (rigid body registration), some are in projective space and some “strange ”distance”).

In Computer Vision the most commonly used robust fitting criterion is MaxCon

Back to the “simple cases”..understand them first!

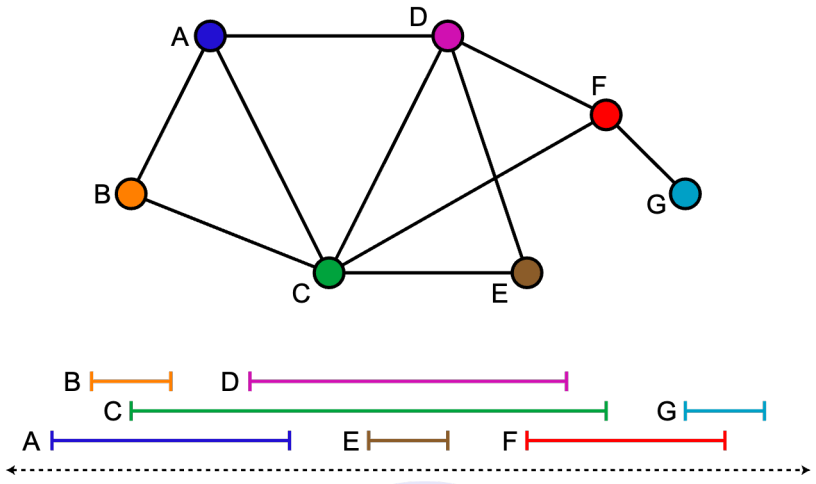
1D location – Interval Graphs – most simple!

Graphs are relationships between PAIRS of items

Hypergraphs are relationships between ANY NUMBER of items

Geometric (Hyper)Graph arise through GEOMETRIC relationships of Items

- Perhaps simplest example is an interval graph
  - Items (vertices in (hyper)graph)
  - Relationship (pairwise) intersection – "overlap"
  - Applications – scheduling, and more!



(Image source – wikepedia)

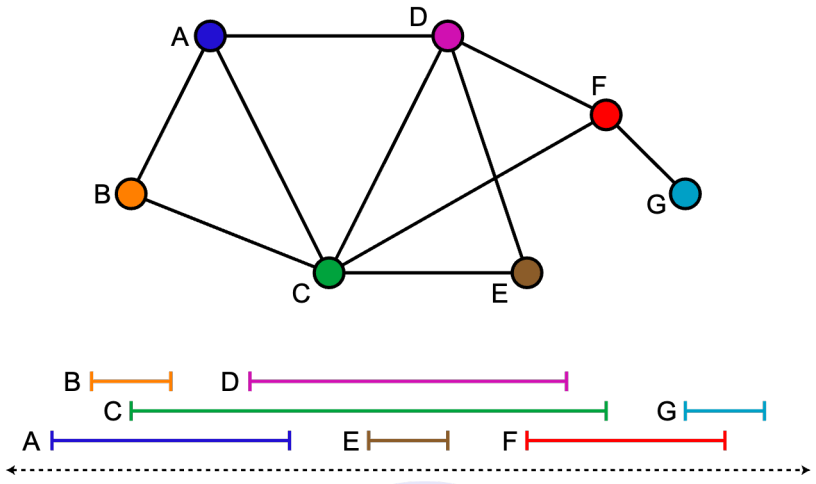
Simple Example – Interval Graphs

Graphs are relationships between PAIRS of items

Hypergraphs are relationships between ANY NUMBER of items

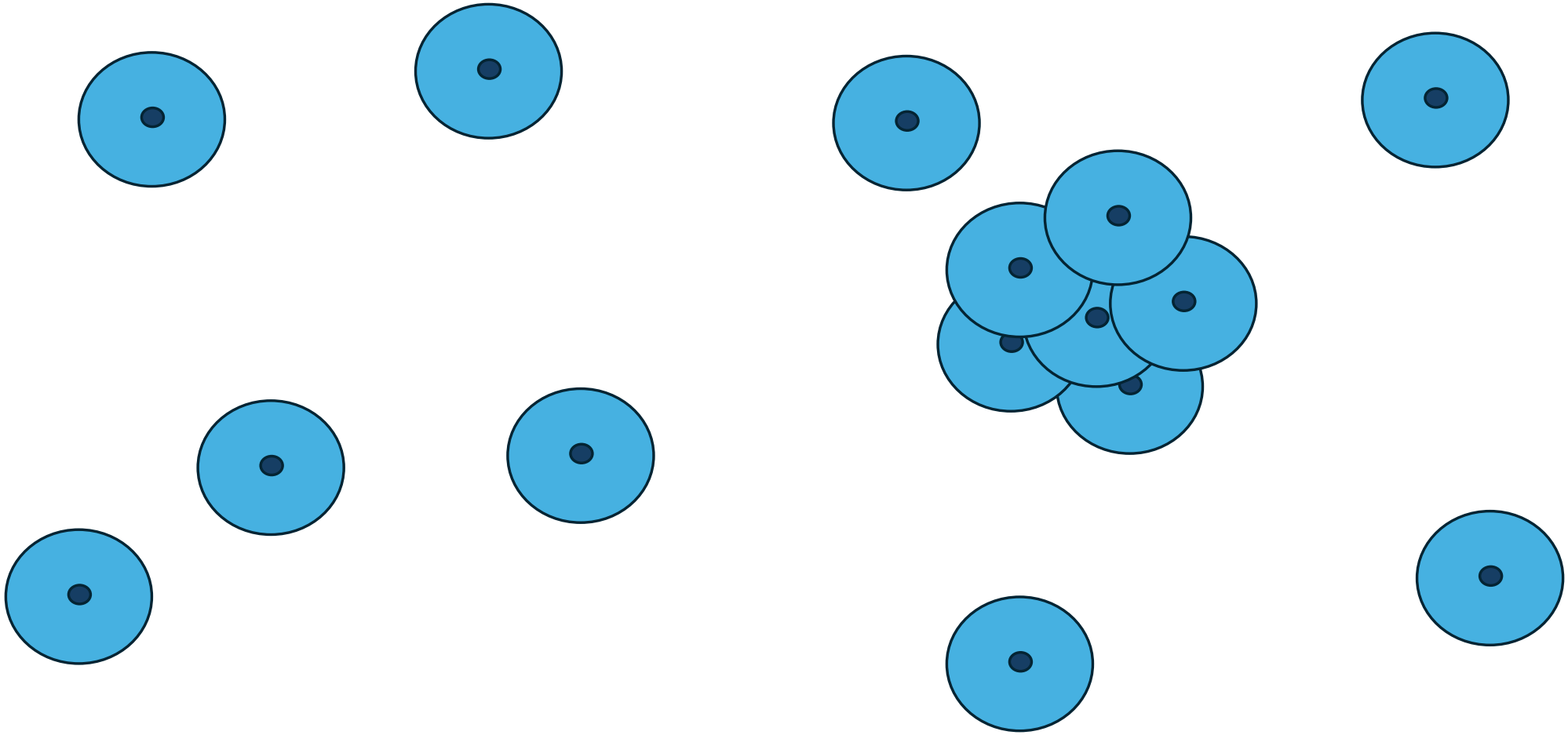
Geometric (Hyper)Graph arise through GEOMETRIC relationships of Items

- In scheduling applications – Independent sets (subsets of vertices not containing an edge) represent events that can be scheduled without overlap.
- Sometimes overlap is bad (conflicts – classes that cannot be at overlapping times for students) – other times good (allowing for passengers to go from train to train, plane to plane)

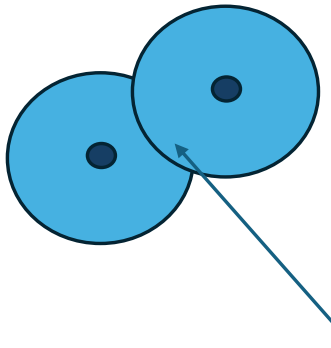


(Image source – wikepedia)

2D location estimation – model problem



2D location estimation – model problem



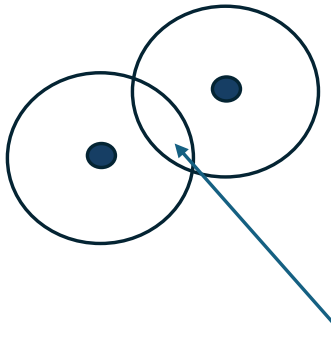
Common models – models they are with are in the intersection

2D location estimation – model problem

So…we can take every pair of data points and decide if they “agree” or not.
This would form a graph. Each vertex in the graph is a data point. There is an edge between each two “if they agree” and no edge otherwise.

So models “supported by the data” are associated with many “data points that agree” – clusters/cliques in graphs.

MaxCon (maximum concensus) is essentially clique finding.



Common models – models they are with are in the intersection

Maximum Clique in Unit Disk graphs

- One has to distinguish two situations:
  - One has the data – then one can use the “data space” geometry/information
  - One has only the graph (it is usually quite hard to then derive a data configuration that would generate that graph – so one has to use ”pure” graph algorithms
- For most of the problems/scenarios I’m interested in – one has the data. Moreover, for hypergraphs (later) especially, one doesn’t even want to construct the (hyper)graph (though going from data to (hyper)graph is usually much easier than the other way around!)

MaxClique on Unit Disk graphs given the data

- A fairly recent survey of state of the art algorithms can be found in https://arxiv.org/pdf/2506.21926 (June 27 2025).
  - (1990) $O(N^{4.5})$ algorithm
  - (1991) $O(N^{3.5} \log N)$ algorithm
  - (2007) $O(N^{3} \log N)$ algorithm
  - (2023) $O(N^{2.5} \log N)$ algorithm
  - (2025) $O(N^{7/3} \log N)$ algorithm
- What if the best one can ever do? That's yet to be determined!

What if you are NOT given the data, and what about if the graph is NOT a unit disk graph (you want to find max clique in a given graph)..

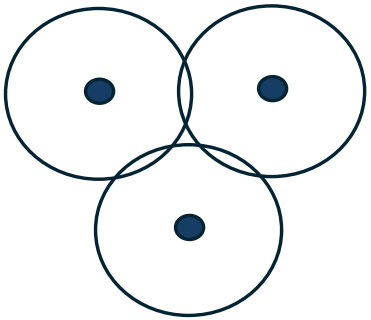
- The naive algorithm is to test every subset of vertices to see if is a clique and to return the largest such. This is a terrible algorithm for "large" n – as the number of subsets is 2^N (so the runtime of the naïve algorithm is roughly $p(N)2^N$ for some (uninteresting for very large N) polynomial p(N).
- It has been known since the 1970's (e.g., 1976 Stanford technical report of Tarjan) that one can *in principle* shave the exponent (that technical report shaves to N/3) – but as that report admits, this only "buys" a factor of 3 in the size of the graphs one can handle and at the expense of a complicated algorithm – it's more a theoretical than a practical algorithm.

What if you are NOT given the data, and what about if the graph is NOT a unit disk graph (you want to find max clique in a given graph)..

- As we saw, we can generally do much better if we know the class of graphs (unit disk graph) AND have the data that gave rise to that graph – somewhere between $O(N^2)$ and $O(N)$ in that case.
- But how much of the gain comes from the restricted graph class and how much comes from having access to the data that generated the graph?
- This set of notes is around those sort of questions…

MaxClique on Unit Disk graphs given the data

Note: there are subtley different versions of what one might mean by MaxClique – for Unit Disk tolerances (and more generally for our "zoo" of problems….). It has to do with what is known as the Helly number/Helly's theorem.



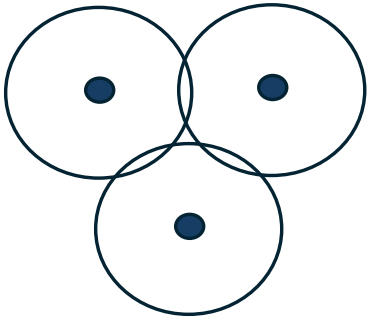
Pairwise agreement amongst all pairs of 3 data points.

But is there (at least one) MODEL they all agree on??

MaxClique on Unit Disk graphs given the data

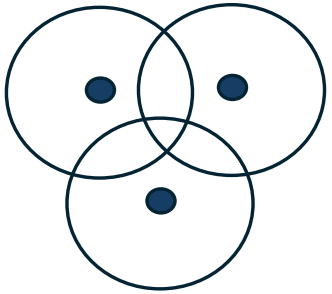
Three (or more) data points that only *pairwise* agree may not agree at each triple (or more) on a *common* model.

Helly: if all (d+1)-sized subsets of a set of convex bodies in R^d intersect, then they ALL intersect (at a common point – or set of common points). Here d=2 so (2+1)-set intersections are what you need.



Pairwise agreement amongst all pairs of 3 data points.

But is there a (at least one) MODEL they all agree on??

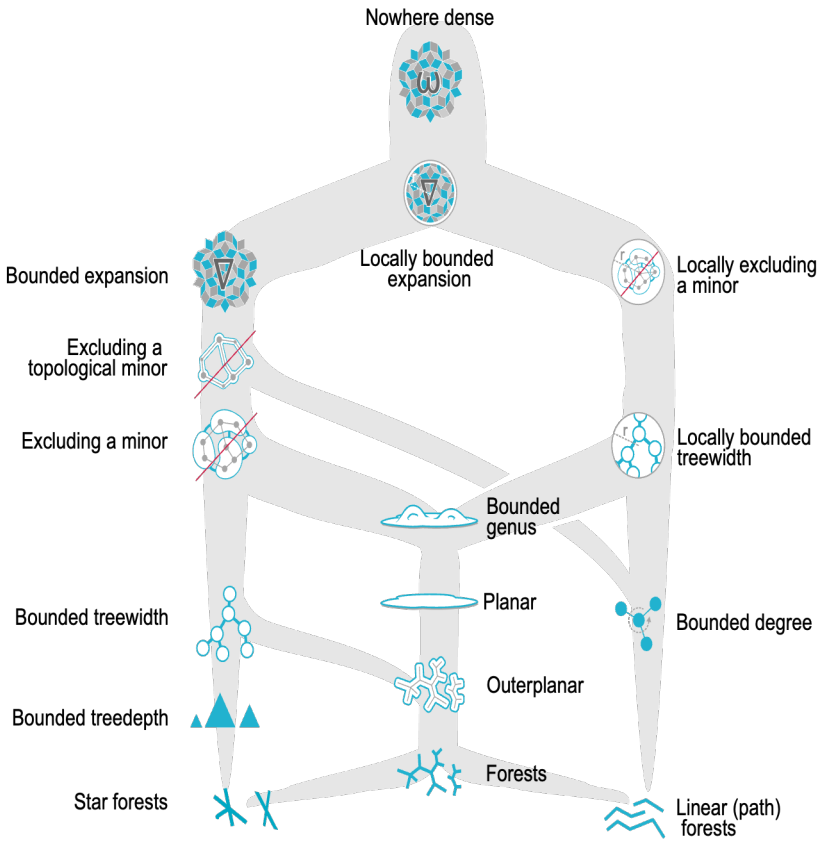


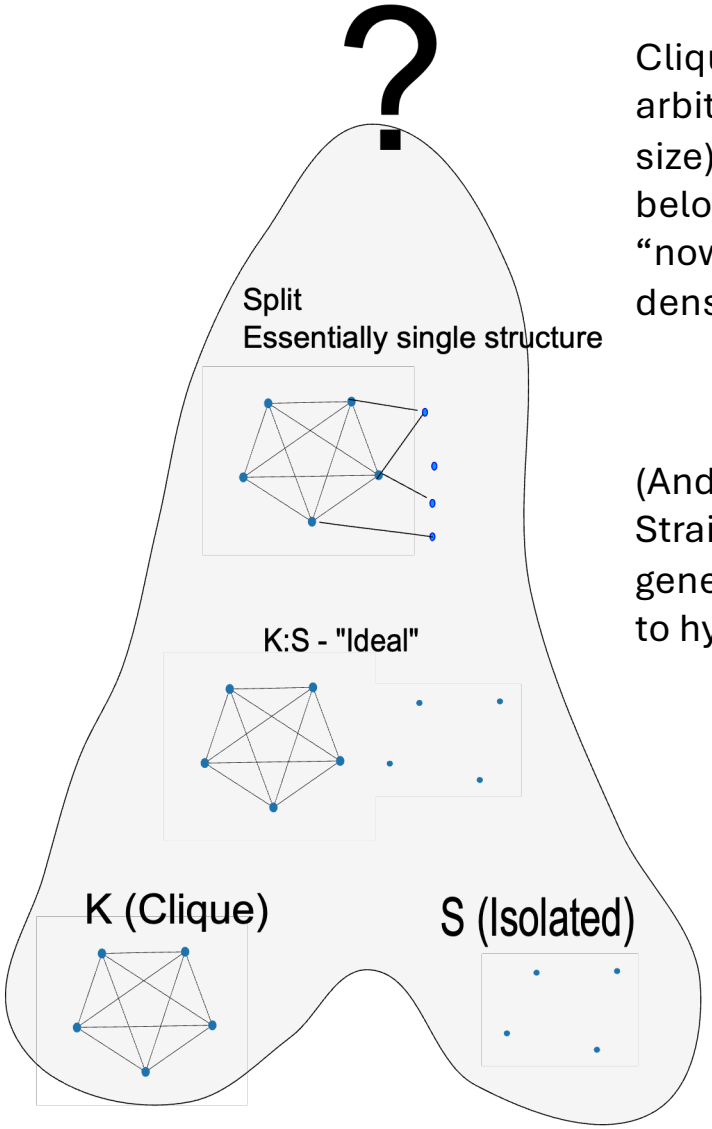
3-wise agreement

Helly's theorem states that you need to enforce 3-wise agreements to ensure that ALL data in some subset "agree" – stronger notion of "agree". Leads to **hyper**graph versions of unit disk "agreement" in R^2

MaxCon fitting/segmentation and Graphs

- MaxCon is equivalent to Max-Clique (of the complement of the graph of minimal sized infeasible subsets) or, if you prefer, equivalent to the Max-Independent Set (of the graph of minimal sized infeasible subsets).
- BUT MaxCon comes with “data space” oracles – L_infinity fit -> oracle of feasibility (of any set of vertices)
- So…does the complexity of the underlying graph still reflect the complexity (in some way – relative ordering of complexity) of data?





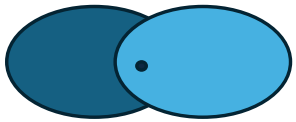
Cliques (of arbitrary large size) cannot belong to “nowhere dense” classes.

(And Straightforward generalizations to hypergraphs)

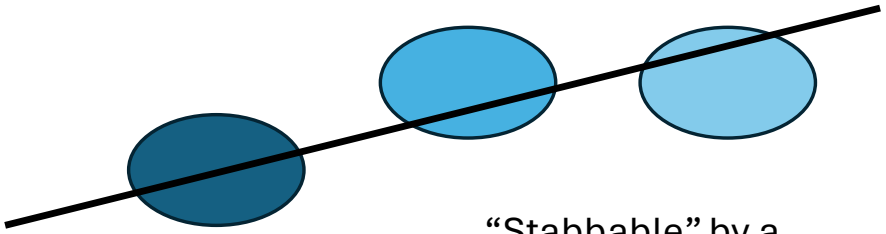
Generalizing Overlap/Intersection –stabbing etc
Universe of Geometric Hypergraphs

Intersection = "stabbable" by points. If sets of points in R^d intersect then they have at least one point in common – there is one point that "stabs" every set.

Generalise – "stabbable" by points -> stabbable by X (X could be lines – having at least one point in common becomes they have a common line that intersects them all)



Intersect <-> at least one point stabs them both



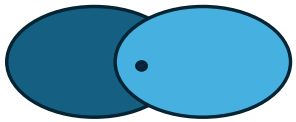
"Stabbable" by a common line

More technical/common terms – transversal/hitting set. Range spaces….etc.

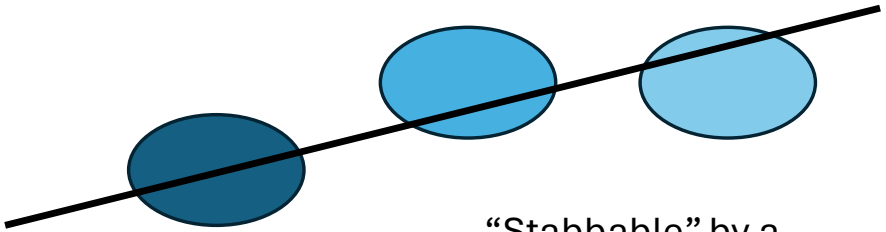
Generalizing Overlap/Intersection –stabbing etc
Universe of Geometric Hypergraphs

Note – any PAIR of disks is ALWAYS stabbable by a single line. So that is an "information free" event.

Information only exists when 3 or more discs are stabble by a line. So this is not a "graph" setting – but a HYPERgraph setting.



Intersect <-> at least one point stabs them both



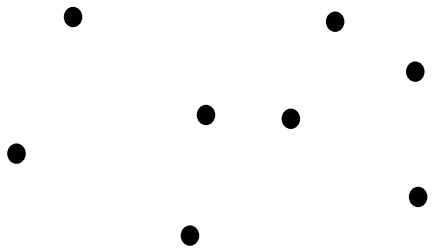
"Stabbable" by a common line

More technical/common terms – transversal/hitting set. Range spaces….etc.

But what does this have to do with AI, Machine Learning, Signal and Image Processing???

Robust fitting is essentially like "fitting within tolerances". Outliers "don't belong to the fit" because they cannot be fit within tolerance.

Tolerances can be pictured as geometric shapes around the data points. The models must "stab" these geometric shapes to be within tolerance of the data points.



Data Points

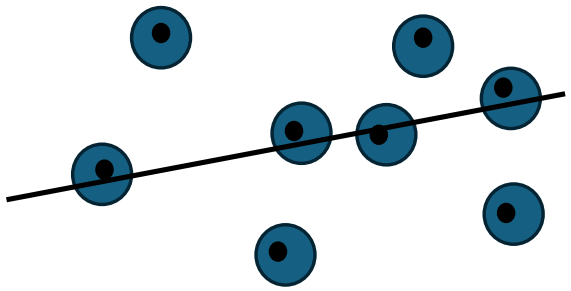
Robust Fitting (MaxCon) – Fitting within tolerance
(Given a model – line, plane, hyperplane,... and data points, what is the model that fist more points than any other model? Alternatively, what is the largest subset of points within tolerance of a model)

Problem	(Hyper)Graph class (every instance is a member of this class)
1D location estimation (equal tolerances/unequal tolerances)	Unit interval/Interval
1D subspace (in 2D) regression	Interval
1D line regression in 2D (affine line fitting in 2D)	3-uniform hypergraph (some subclass of – WHICH?) Intersection of ”fat” lines in 2D – but what class is that?
(2D) plane regression in 3D	4-uniform hypergraph (some subclass of – WHICH?) Intersection of fat planes in 3D
..... (7D) hyperplane regression in 8D (linear model/approximation of some projective geometry in computer vision)	9-uniform hypergraph (some subclass of – WHICH?) Intersection of fat hyperplanes in 3D

Note: the above are just *SOME* examples of the models. Some use geometric distance instead of regression distance – some are fitting “element of a group” (rigid body registration), some are in projective space and some “strange ”distance”).

But what does this have to do with AI, Machine Learning, Signal and Image Processing???

Tolerances can be pictured as geometric shapes around the data points. The models must "stab" these geometric shapes to be within tolerance of the data points.

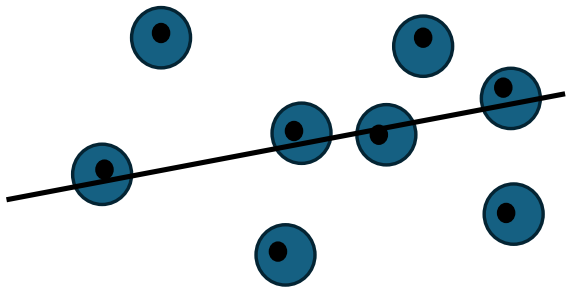


Data Points – with tolerances

A line that fits 4 points within tolerance – ie., line stabbing disks

But what does this have to do with AI, Machine Learning, Signal and Image Processing???

Maximum Consensus Robust fitting (MaxCon) – find the model that fits the most data points within tolerance.

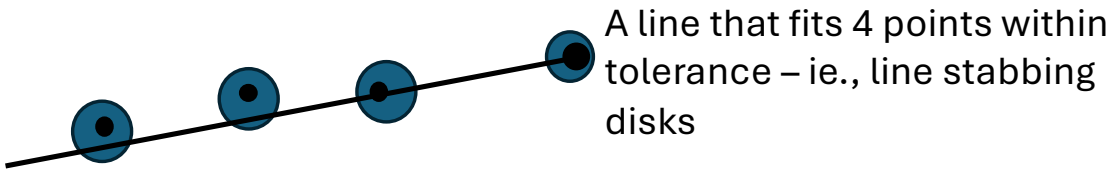


A line that fits 4 points within tolerance – ie., line stabbing disks

Data Points – with tolerances

Example (robust) line fitting

Most simple situation – one clique, no outliers (one "structure", one "cluster"). So simple – there is no need for robust fitting!

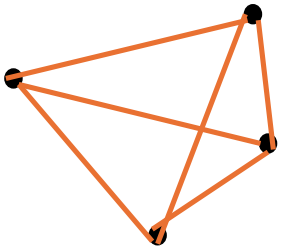


Complexity charcterization

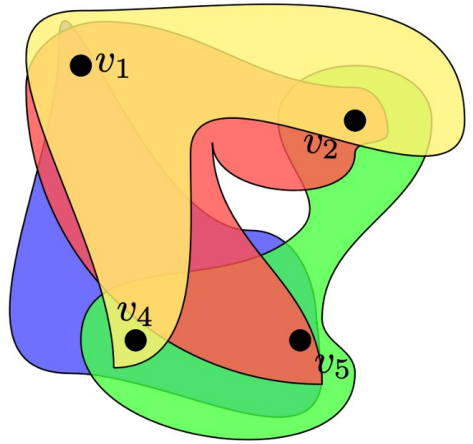
Clique – “level 0 complexity”

All four data points are feasible…no outliers.

Single structure – no outliers



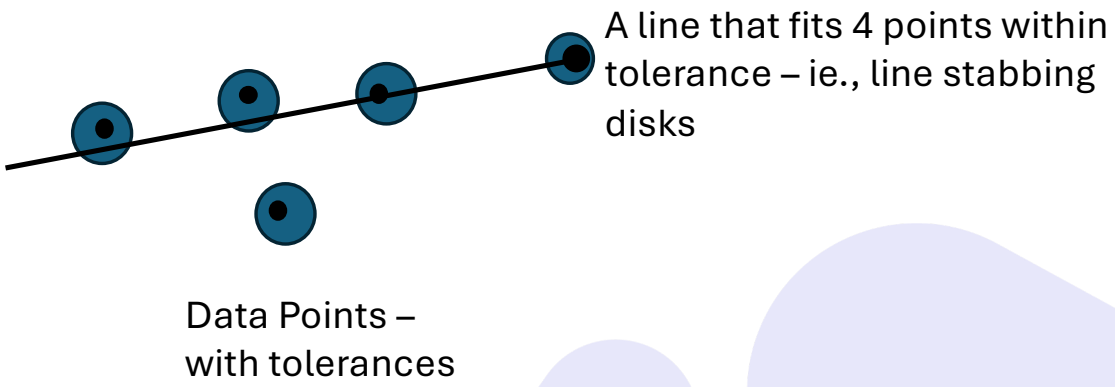
K_4



A 3-uniform hypergraph containing a 4-clique

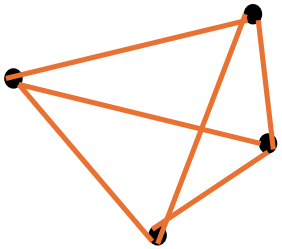
Example (robust) line fitting

Next most simple – one clique and isolated vertices (one “structure” and “pure” outliers)



Complexity charcterization

Clique with isolated vertices – “level 1 complexity”



Single structure –
with few outliers.

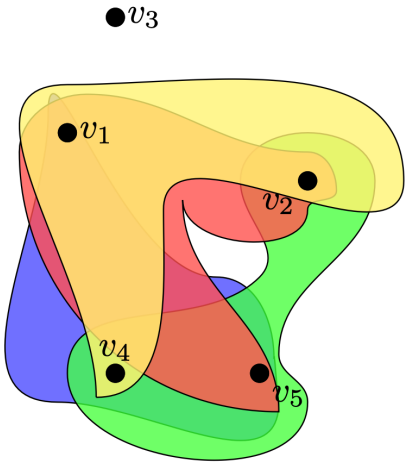
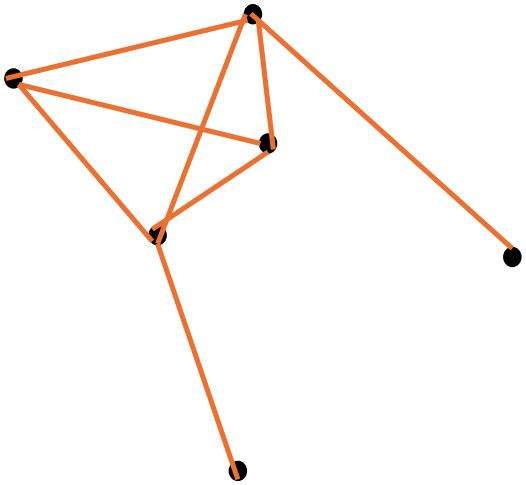


Figure 5.1: A 3-uniform hypergraph containing a 4-clique

K:S – clique and
independent/stable set
= complement of complete
split

Complexity charcterization

Clique with set of independent vertices and some edges between – “level 2 complexity”



K:S aka “Split Graph”

The edges between some of the outliers and inliers are “accidental alignments”

Estimates of Location in 1D

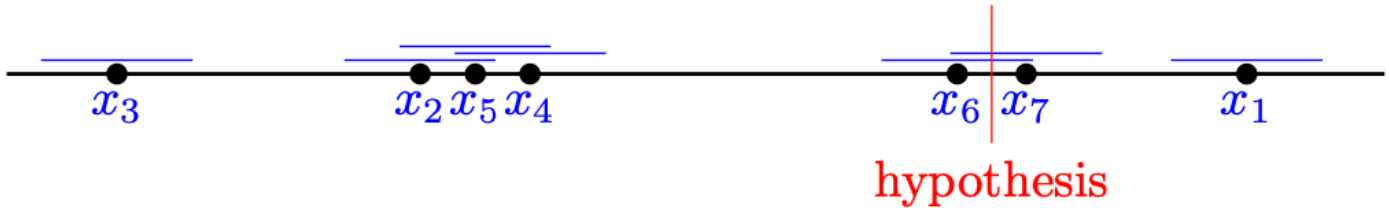
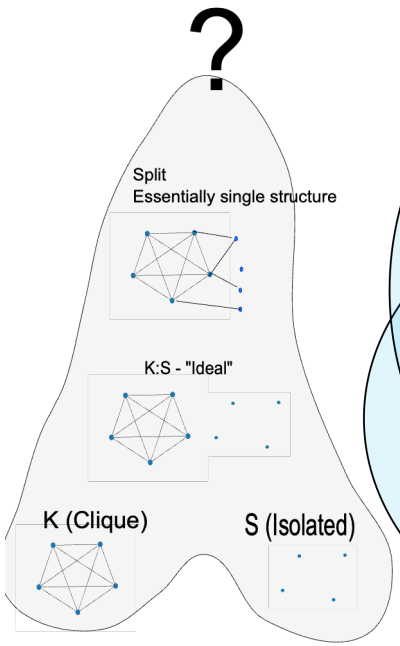


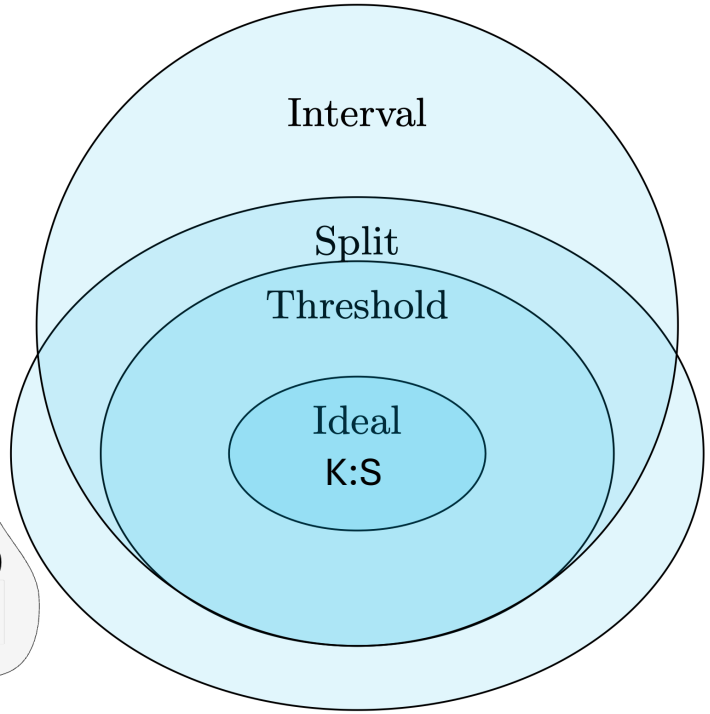
Figure 7.1: 1D estimates of location

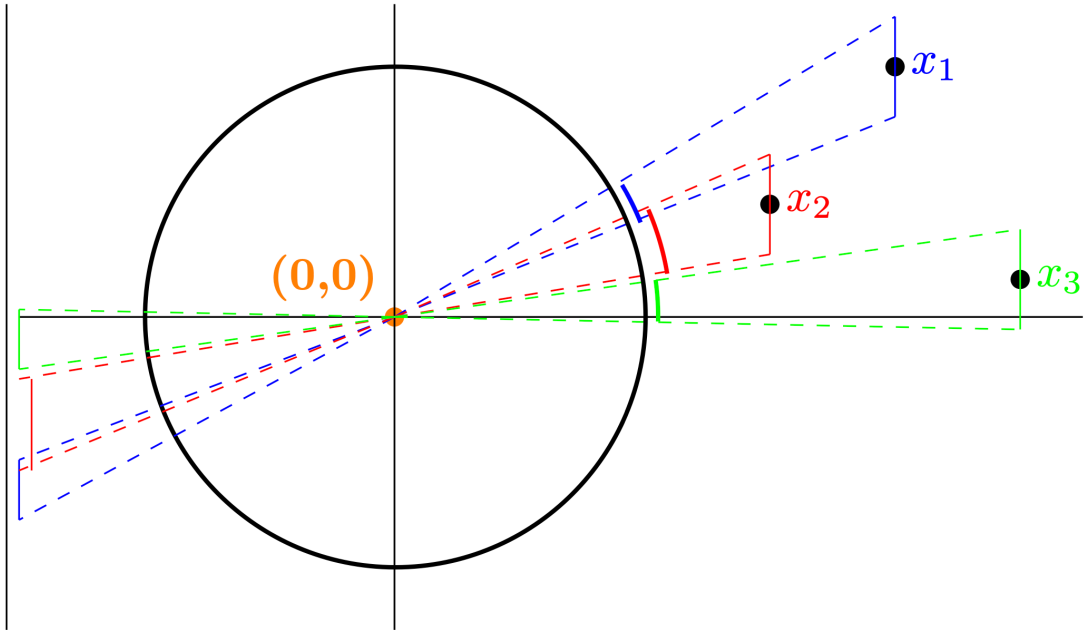
This is essentially the definition of unit interval graphs!

So we can immediately relate to the structural graph theory.

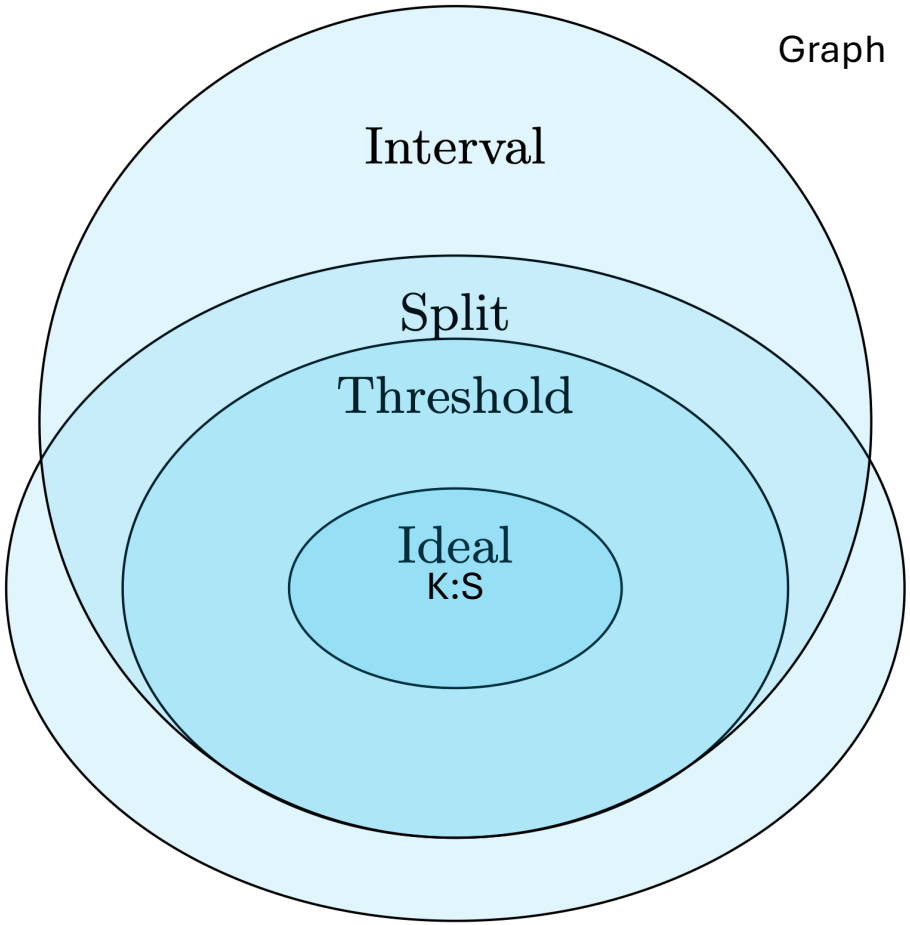
Complexity characterization – robust linear subspace in 2D (line) regression







Complexity characterization – robust fitting



Graph

1D location,
1D subspace in 2D

Hypergraph

?

(affine) line fitting in 2D, Homography fitting, Fundamental Matrix fitting, Rigid Motion fitting.....

MaxCon fitting/segmentation and Graphs

- MaxCon is equivalent to Max-Clique (of the complement of the graph of minimal sized infeasible subsets) or, if you prefer, equivalent to the Max-Independent Set (of the graph of minimal sized infeasible subsets).
- BUT MaxCon comes with “data space” oracles – L_infinity fit -> oracle of feasibility (of any set of vertices)
- So…does the complexity of the underlying graph still reflect the complexity (in some way – relative ordering of complexity) of data?

Robust Fitting (MaxCon) – Fitting within tolerance
(Given a model – line, plane, hyperplane,... and data points, what is the model that fist more points than any other model? Alternatively, what is the largest subset of points within tolerance of a model)

Problem	(Hyper)Graph class (every instance is a member of this class)
1D location estimation (equal tolerances/unequal tolerances)	Unit interval/Interval
1D subspace (in 2D) regression	Interval
1D line regression in 2D (affine line fitting in 2D)	3-uniform hypergraph (some subclass of – WHICH?) Intersection of ”fat” lines in 2D – but what class is that?
(2D) plane regression in 3D	4-uniform hypergraph (some subclass of – WHICH?) Intersection of fat planes in 3D
..... (7D) hyperplane regression in 8D (linear model/approximation of some projective geometry in computer vision)	9-uniform hypergraph (some subclass of – WHICH?) Intersection of fat hyperplanes in 3D

Note: the above are just *SOME* examples of the models. Some use geometric distance instead of regression distance – some are fitting “element of a group” (rigid body registration), some are in projective space and some “strange ”distance”).

(tw,ω)-bounded class

"The only reason treewidth is unbounded for the class, is the presence of a large clique". Trivial example – class K_n

A hereditary class of graphs C is said to be (tw,ω)-bounded if there exists a function g such that the treewidth of any graph $G \in C$ is at most $g(\omega(G))$.

So clearly is a characterization of some of the lower hierarchy of

$(K_{n-k}, S_k) \longrightarrow$ Interval

How much of the hierarchy?

Version for hypergraphs?

Treewidth of interval graphs

For any interval graph its clique tree = tree decomposition = path decomposition

pathwidth = treewidth for interval graphs

pw(G) = |largest clique| – 1 when G is an interval graph

= tw(G)

Hence interval graphs are (tw,ω)-bounded with g(ω(G))= ω(G))+1

Twin width

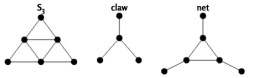
- A twin pair u and v are vertices such that N(u)\v=N(v)\u
- Twin width is a measure that captures how close to being composed of “twins” a graph is
- Graphs of twin-width 0 are EXACTLY the class of co-graphs
  - Cograph (aka P_4 –free) intersect interval, is the same as cograph intersect chordal
  - Infact, (cograph intersect interval)=(cograph intersect chordal)=quasi threshold = trivially perfect = intersection graph of nested intervals (no interval partially overlaps another interval)
  - Which implies that the interval graphs of twin-width 0 are exactly the quasi-threshold (aka trivially perfect) graphs (aka (C_4 ,P_4)-free)

Computational Considerations

Any algorithm that SOLELY exploits twin-width=0 will not distinguish between the subclasses of quasi-threshold (will show the same performance scaling), BUT will distinguish between quasi-threshold and any larger class (including those Interval graphs that are outside of quasi-threshold).

Empty/$\overline{K}_n$	Complete/ Clique K_n	$\overline{K}_2$ or ($2K_1$)	K_2	Constant speed (linear with n)
$\overline{\text{Complete Split}}$ (Ideal)	Complete Split	($2K_2$,P_3)	(C_4,$\overline{P}_3$)	Exponentional <factorial
threshold		($2K_2$,C_4,P_4)		factorial
split		($2K_2$,C_4,C_5)		Super factorial
Quasi-threshold		(C_4,P_4)		factorial
Unit Interval		**(C_{n+4},S_3,claw,net)**		**factorial**
Interval		**(C_{n+4},T_2,X_{31},XF_2^{n+1},XF_3^n)**		**factorial**

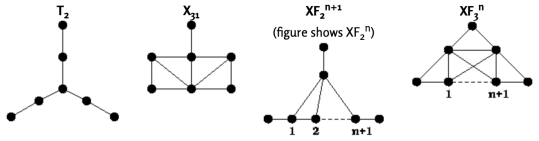
Speeds are in terms of *Labelled* graphs



Split-interval: note T_2 is not a split graph – so automatically excluded by being split, C_{n+4} , X_{31} , fork are also not split, (but net is), $XF^{n+1}{}_2$ is not split for n>1, S_3 and rising sun are split, but $XF^n{}_3$ is not split for n>1

So…split-interval are the graphs that exclude ($2K_2$,C_4,C_5,S_3 ,rising sun, net)

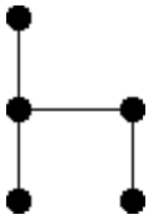
Interval-threshold – this intersection is just threshold – as can be seen by the fact that all excluded induced subgraphs of interval are not threshold graphs – so already excluded.



$XF^0{}_2$=fork, $XF^1{}_2$=net

$XF^0{}_3$=S_3, $XF^1{}_3$=rising sun

fork = chair g6: DiC



rising sun g6: F~p`_