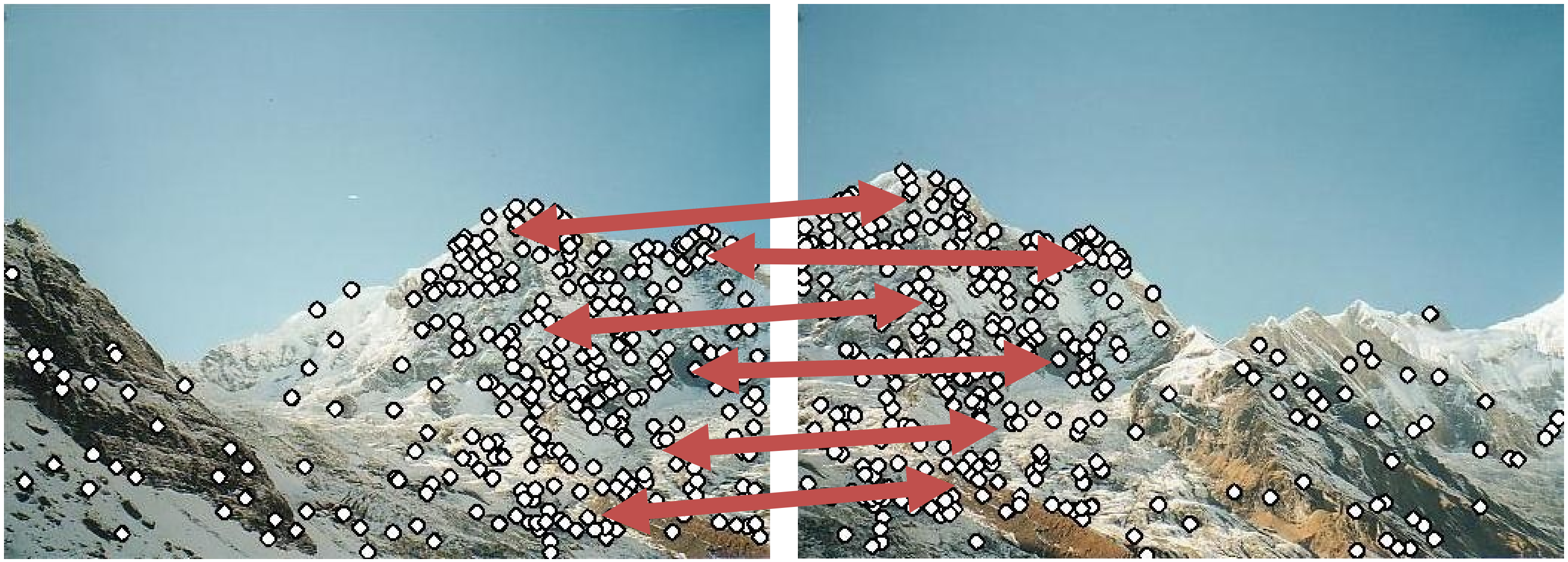
Transformations and Fitting

EECS 442 – David Fouhey and Justin Johnson

Winter 2021, University of Michigan

https://web.eecs.umich.edu/~justincj/teaching/eecs442/WI2021/

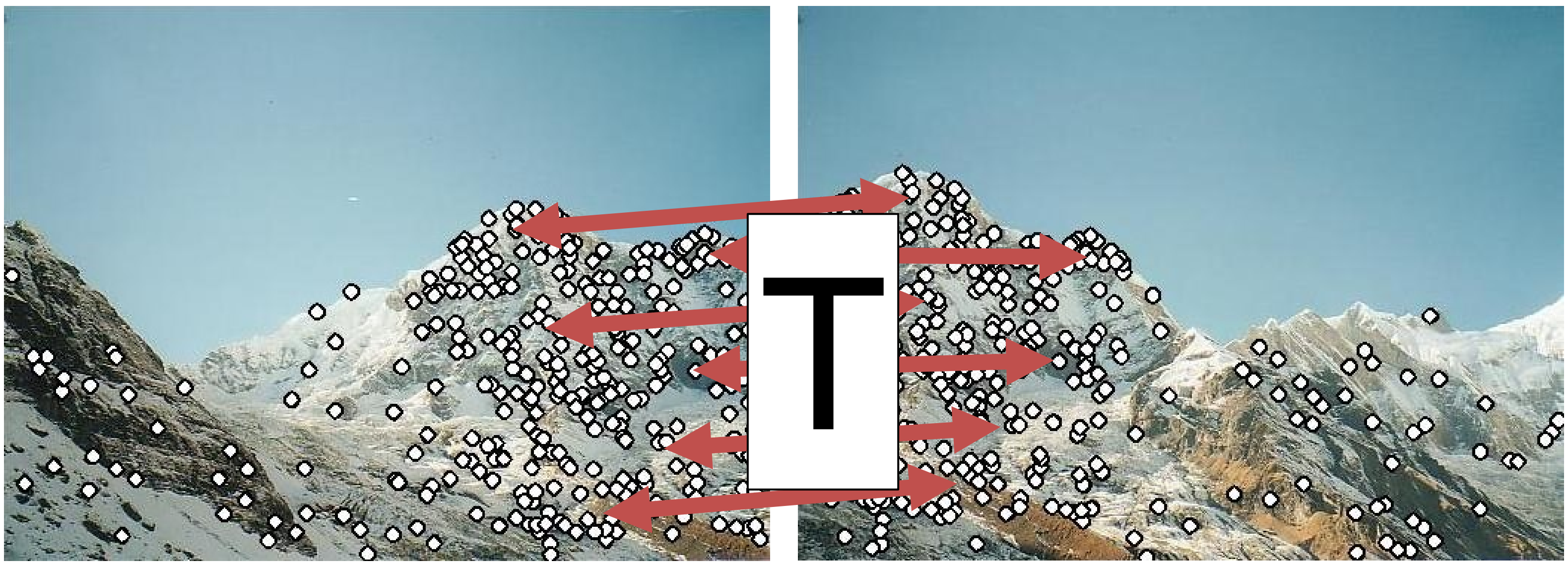
Last Class



1. How do we find distinctive / easy to locate features? *(Harris/Laplacian of Gaussian)*
2. How do we describe the regions around them? *(Normalize window, use histogram of gradient orientations)*

Earlier I promised

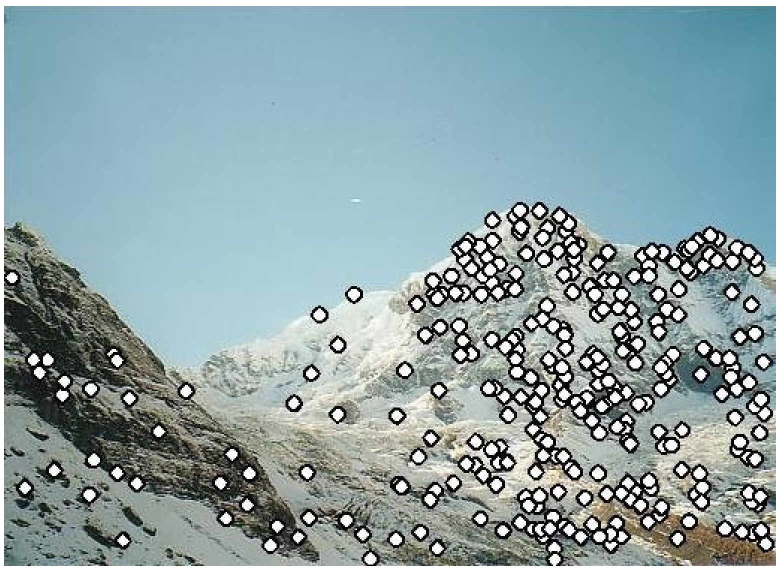
Solving for a Transformation



3: Solve for transformation T (e.g. such that **p1 ≡ T p2**) that fits the matches well

Before Anything Else, Remember

You, with your gigantic brain, see:



The computer sees:

097	097	097	097	097	097	097	097	096	097	097	096	096	096
100	100	100	100	100	100	101	101	102	101	100	100	100	099
105	105	105	105	105	105	105	103	102	102	101	103	104	105
109	109	109	109	109	110	107	118	145	132	120	112	106	103
113	113	113	112	112	113	110	129	160	160	164	162	157	151
118	117	118	123	119	118	112	125	142	134	135	139	139	175
123	121	125	162	166	157	149	153	160	151	150	146	137	168
127	127	125	168	147	117	139	135	126	147	147	149	156	160
133	130	150	179	145	132	160	134	150	150	111	145	126	121
138	134	179	185	141	090	166	117	120	153	111	153	114	126
144	151	188	178	159	154	172	147	159	170	147	185	105	122
152	157	184	183	142	127	141	133	137	141	131	147	144	147
130	147	185	180	139	131	154	121	140	147	107	147	120	128
035	102	194	175	149	140	179	128	146	168	096	163	101	125

You should expect **noise** (not at quite the right pixel) and **outliers** (random matches)

Today

- How do we fit **models** (i.e., a parameteric representation of data that’s smaller than the data) to data?
- How do we handle:
  - **Noise** – least squares / total least squares
  - **Outliers** – RANSAC (random sample consensus)
  - **Multiple models** – Hough Transform (can also make RANSAC handle this with some effort)

Working Example: Lines

- We’ll handle lines as our **models** today since you are more familiar with them than others
- Next class will cover more complex models. I promise we’ll eventually stitch images together
- You can apply today’s techniques on next class’s models

Model Fitting

Need three ingredients

Data: what data are we trying to explain with a model?

Model: what's the compressed, parametric form of the data?

Objective function: given a prediction, how do we evaluate how correct it is?

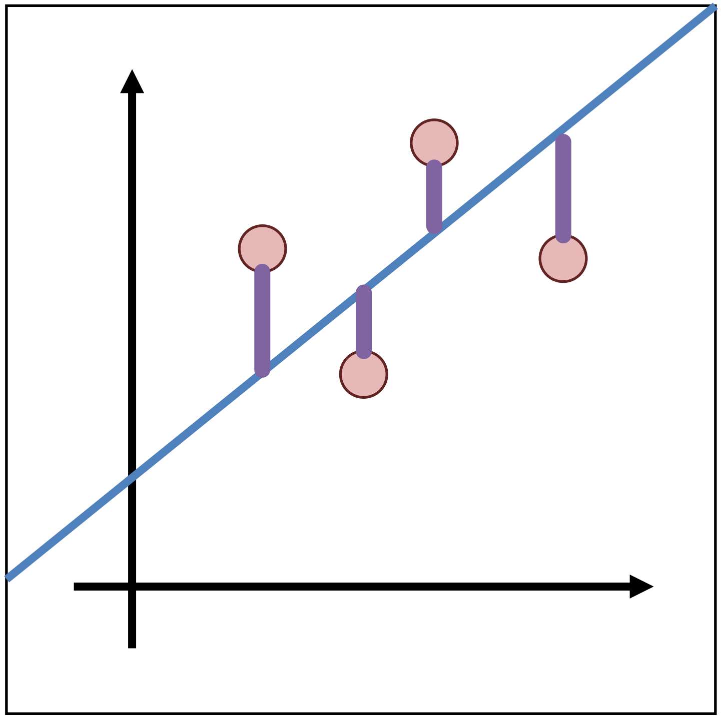
Example: Least-Squares

Fitting a line to data

Data: (x_1,y_1), (x_2,y_2), …, (x_k,y_k)

Model: (m,b) $y_i=mx_i+b$
Or $(\mathbf{w})$ $y_i = \mathbf{w}^T\mathbf{x}_i$

Objective function:
$(y_i - \mathbf{w}^T\mathbf{x}_i)^2$



Least-Squares Setup

$$\sum_{i=1}^{k}\left(y_i - \boldsymbol{w}^T \boldsymbol{x_i}\right)^2 \quad \rightarrow \quad \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$$

$$\boldsymbol{Y} = \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix} \qquad \boldsymbol{X} = \begin{bmatrix} x_1 & 1 \\ \vdots & 1 \\ x_k & 1 \end{bmatrix} \qquad \boldsymbol{w} = \begin{bmatrix} m \\ b \end{bmatrix}$$

Solving Least-Squares

$$||\mathbf{Y} - \mathbf{X}\mathbf{w}||_2^2$$

Where can I find derivatives + matrix expressions and matrix identies?

WolframAlpha computational intelligence.

The Matrix Cookbook

Kaare Brandt Petersen
Michael Syskind Pedersen

Version: November 15, 2012

Wikipedia
The Free Encyclopedia

Solving Least-Squares

$$||\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}||_2^2$$

$$\frac{\partial}{\partial \boldsymbol{w}} ||\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}||_2^2 = 2\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{X}^T\boldsymbol{Y}$$

$$\mathbf{0} = 2\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{X}^T\boldsymbol{Y}$$

Recall: derivative is 0 at a maximum / minimum. Same is true about gradients.

$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{Y}$$

$$\boldsymbol{w} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{Y}$$

Aside: **0** is a vector of 0s. **1** is a vector of 1s.

Two Solutions to Getting W

In One Go

Implicit form
(normal equations)

$$\boldsymbol{X^T X w} = \boldsymbol{X^T Y}$$

Explicit form
(don't do this)

$$\boldsymbol{w} = \left(\boldsymbol{X^T X}\right)^{-1} \boldsymbol{X^T Y}$$

Iteratively

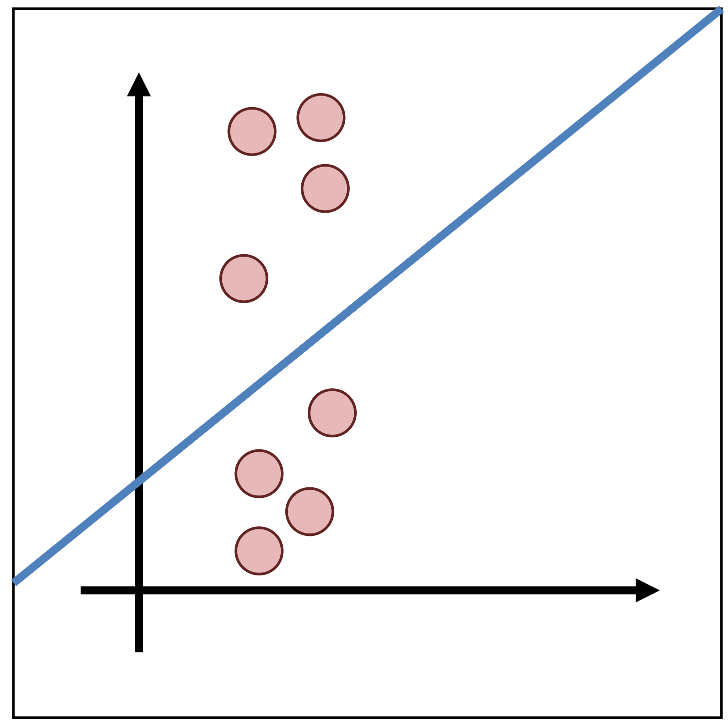
Recall: gradient is also direction that makes function go up the most. **What could we do?**

$$\boldsymbol{w_0} = \boldsymbol{0}$$

$$\boldsymbol{w_{i+1}} = \boldsymbol{w_i} - \boldsymbol{\gamma}\left(\frac{\partial}{\partial \boldsymbol{w}} \|\boldsymbol{Y} - \boldsymbol{Xw}\|_2^2\right)$$

What's The Problem?

- Vertical lines impossible!
- Not rotationally invariant: the line will change depending on orientation of points



Alternate Formulation

Recall: $ax + by + c = 0$

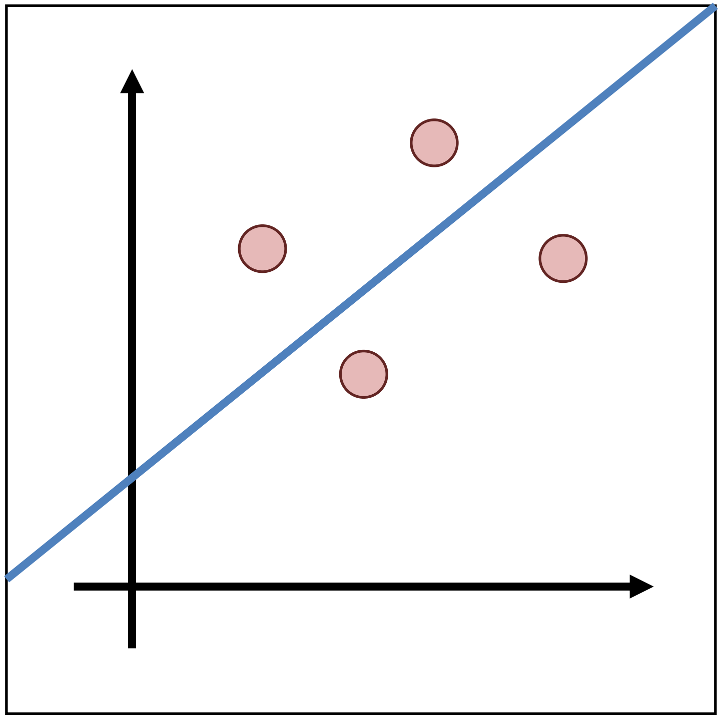
$$\boldsymbol{l}^T \boldsymbol{p} = 0$$

$$\boldsymbol{l} \equiv [a, b, c] \quad \boldsymbol{p} \equiv [x, y, 1]$$

Can always rescale **l**.
Pick another a,b,d so

$$\|\boldsymbol{n}\|_2^2 = \|[a, b]\|_2^2 = 1$$

$$d = -c$$



Alternate Formulation

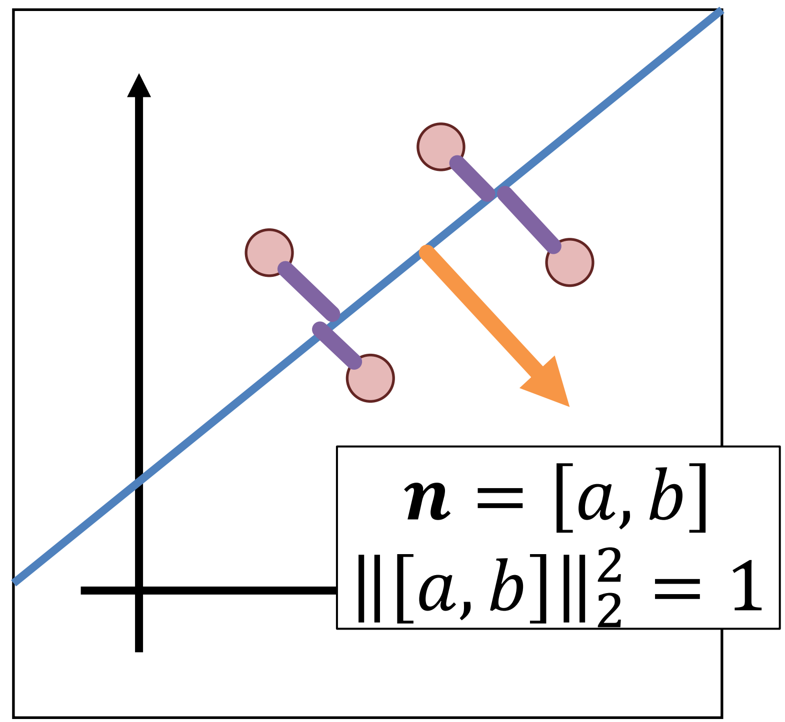
Important part: Any line can be framed in terms of normal **n** and offset d

Now: $ax + by - d = 0$

$$\boldsymbol{n}^{\boldsymbol{T}}[x, y] - d = 0$$

Point to line distance:

$$\frac{\boldsymbol{n}^T[x, y] - d}{\|\boldsymbol{n}\|_2^2} = \boldsymbol{n}^{\boldsymbol{T}}[x, y] - d$$



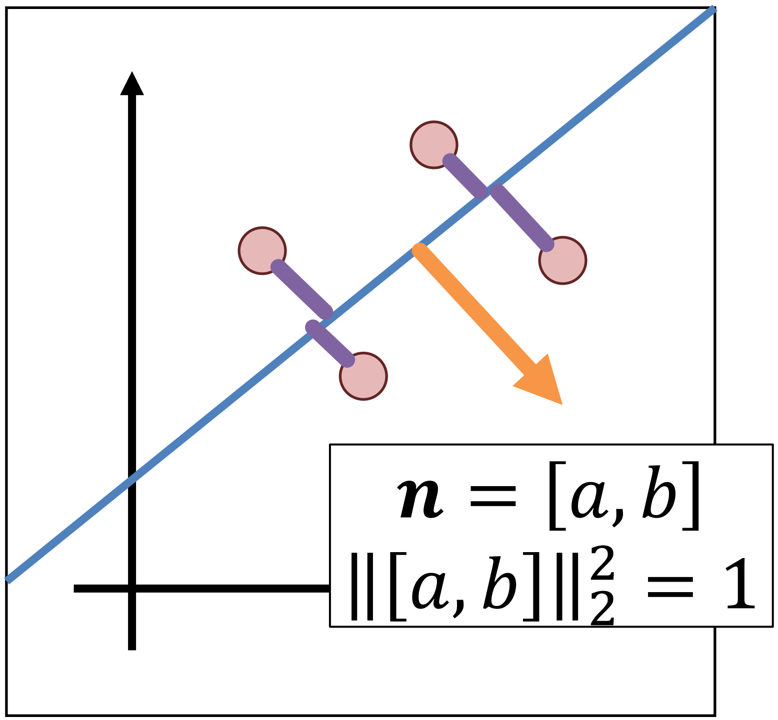
Total Least-Squares

Fitting a line to data

Data: (x_1,y_1), (x_2,y_2), …, (x_k,y_k)

Model: $(\mathbf{n},d)$, $||\mathbf{n}||^2 = 1$
$\mathbf{n}^T[x_i,y_i]-d=0$

Objective function:
$(\mathbf{n}^T[x_i,y_i]-d)^2$



Total Least Squares Setup

Figure out objective first, then figure out ||n||=1

$$\sum_{i=1}^{k} \left(\boldsymbol{n}^T [x, y] - d\right)^2 \longrightarrow \|\boldsymbol{X}\boldsymbol{n} - \mathbf{1}d\|_2^2$$

$$\boldsymbol{X} = \begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_k & y_k \end{bmatrix} \quad \mathbf{1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \quad \boldsymbol{n} = \begin{bmatrix} a \\ b \end{bmatrix} \quad \boldsymbol{\mu} = \frac{1}{k}\mathbf{1}^T\boldsymbol{X}$$

The mean / center of mass of the points: np.sum(X,axis=0). We'll use it later

Total Least Squares Setup

Want to make sure that the following is minimized:

$$\|\boldsymbol{X}\boldsymbol{n} - \mathbf{1}d\|_2^2$$

Won't derive, but can show that whenever you find the **n**, and d that minize the objective, d = $\boldsymbol{\mu}\boldsymbol{n}$. (at back of slides if you're curious.)

$$\boldsymbol{X} = \begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_k & y_k \end{bmatrix} \quad \mathbf{1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \quad \boldsymbol{n} = \begin{bmatrix} a \\ b \end{bmatrix} \quad \boldsymbol{\mu} = \frac{1}{k}\mathbf{1}^T\boldsymbol{X}$$

The mean / center of mass of the points: np.sum(X,axis=0). We'll use it later

Solving Total Least-Squares

$$\|\boldsymbol{X}\boldsymbol{n} - \mathbf{1}d\|_2^2 = \|\boldsymbol{X}\boldsymbol{n} - \mathbf{1}\boldsymbol{\mu}\boldsymbol{n}\|_2^2 \qquad d = \boldsymbol{\mu}\boldsymbol{n}$$

$$= \|(\boldsymbol{X} - \mathbf{1}\boldsymbol{\mu})\,\boldsymbol{n}\|_2^2$$

Objective is then:

$$\arg\min_{\|\boldsymbol{n}\|=1} \left\|(\boldsymbol{X} - \mathbf{1}\boldsymbol{\mu})\,\boldsymbol{n}\right\|_2^2$$

The thing that makes the expression smallest

Homogeneous Least Squares

$$\underset{\|\boldsymbol{v}\|_2^2=1}{\arg\min} \|\boldsymbol{A}\boldsymbol{v}\|_2^2 \longrightarrow$$ Eigenvector corresponding to smallest eigenvalue of A^TA

Why do we need $||v||^2 = 1$ or some other constraint?

Applying it in our case:

$$\boldsymbol{n} = \text{smallest\_eigenvec}((\boldsymbol{X} - \mathbf{1}\boldsymbol{\mu})^{\boldsymbol{T}}(\boldsymbol{X} - \mathbf{1}\boldsymbol{\mu}))$$

Note: technically homogeneous only refers to ||Av||=0 but it's common shorthand in computer vision to refer to the specific problem of ||v||=1

Details For ML-People

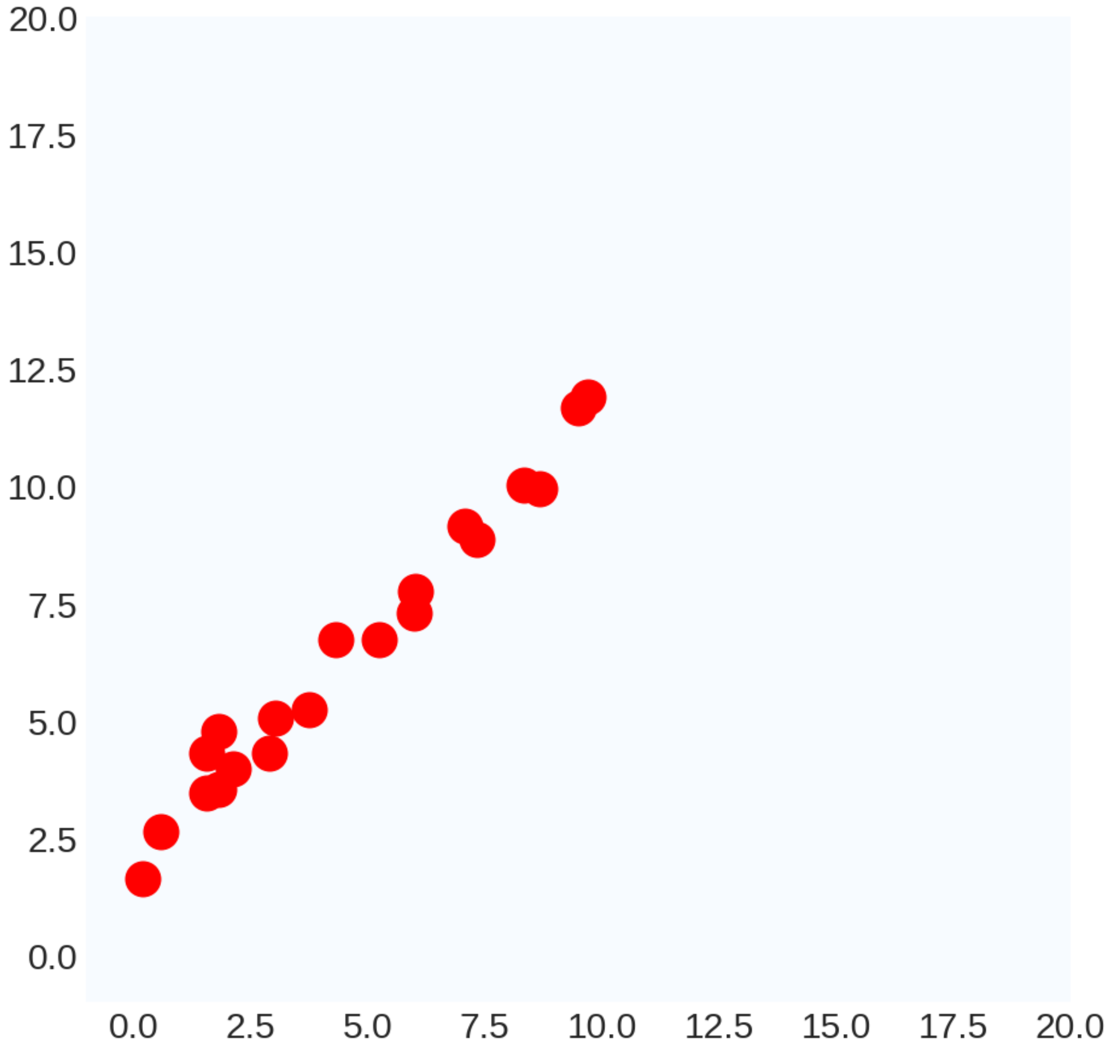
Matrix we take the eigenvector of looks like:

$$(\boldsymbol{X} - \mathbf{1}\boldsymbol{\mu})^T(\boldsymbol{X} - \mathbf{1}\boldsymbol{\mu}) = \begin{bmatrix} \sum_i (x_i - \mu_x)^2 & \sum_i (x_i - \mu_x)(y_i - \mu_y) \\ \sum_i (x_i - \mu_x)(y_i - \mu_y) & \sum_i (y_i - \mu_y)^2 \end{bmatrix}$$

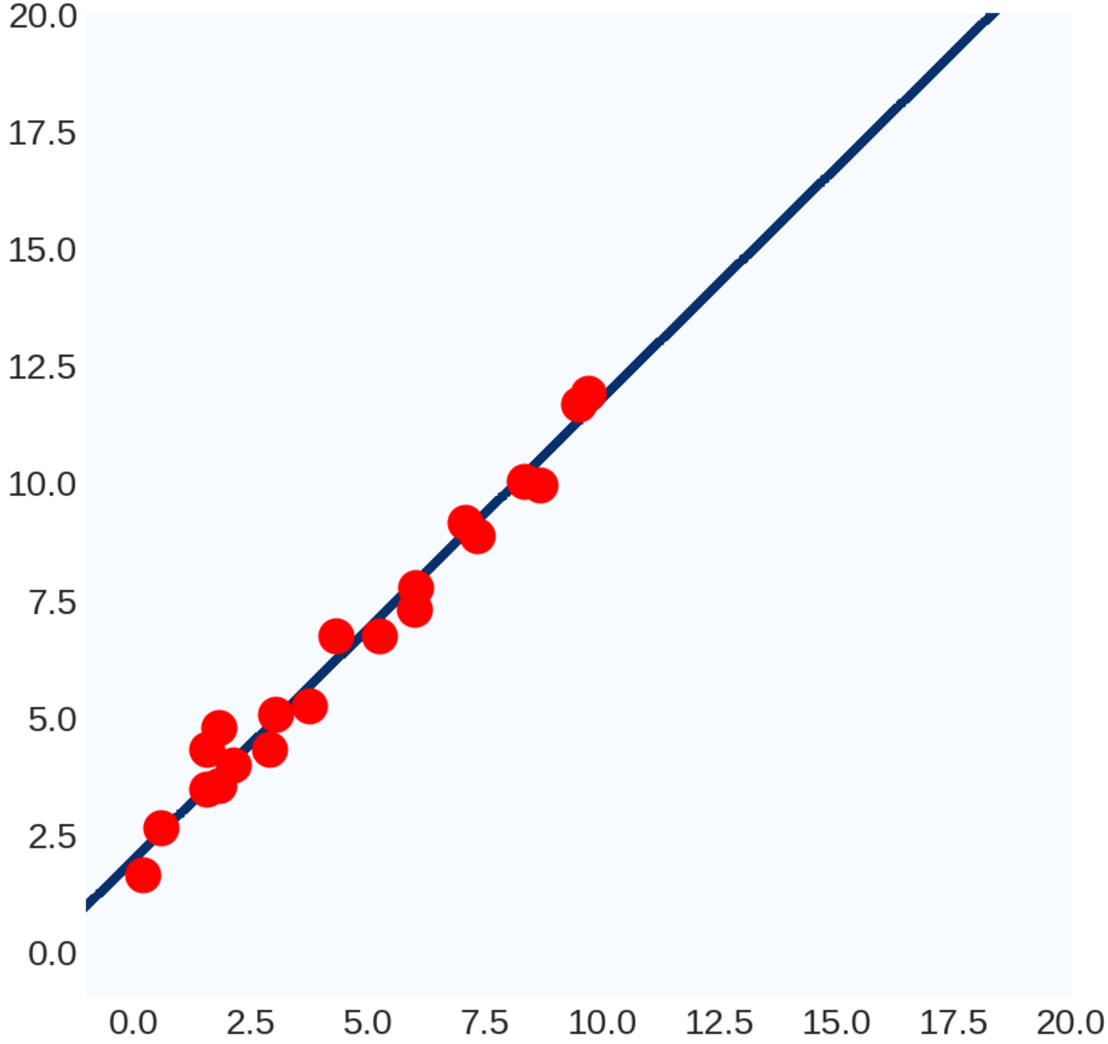
This is a scatter matrix or scalar multiple of the covariance matrix. We're doing PCA, but taking the least principal component to get the normal.

Note: If you don't know PCA, just ignore this slide; it's to help build connections to people with a background in data science/ML.

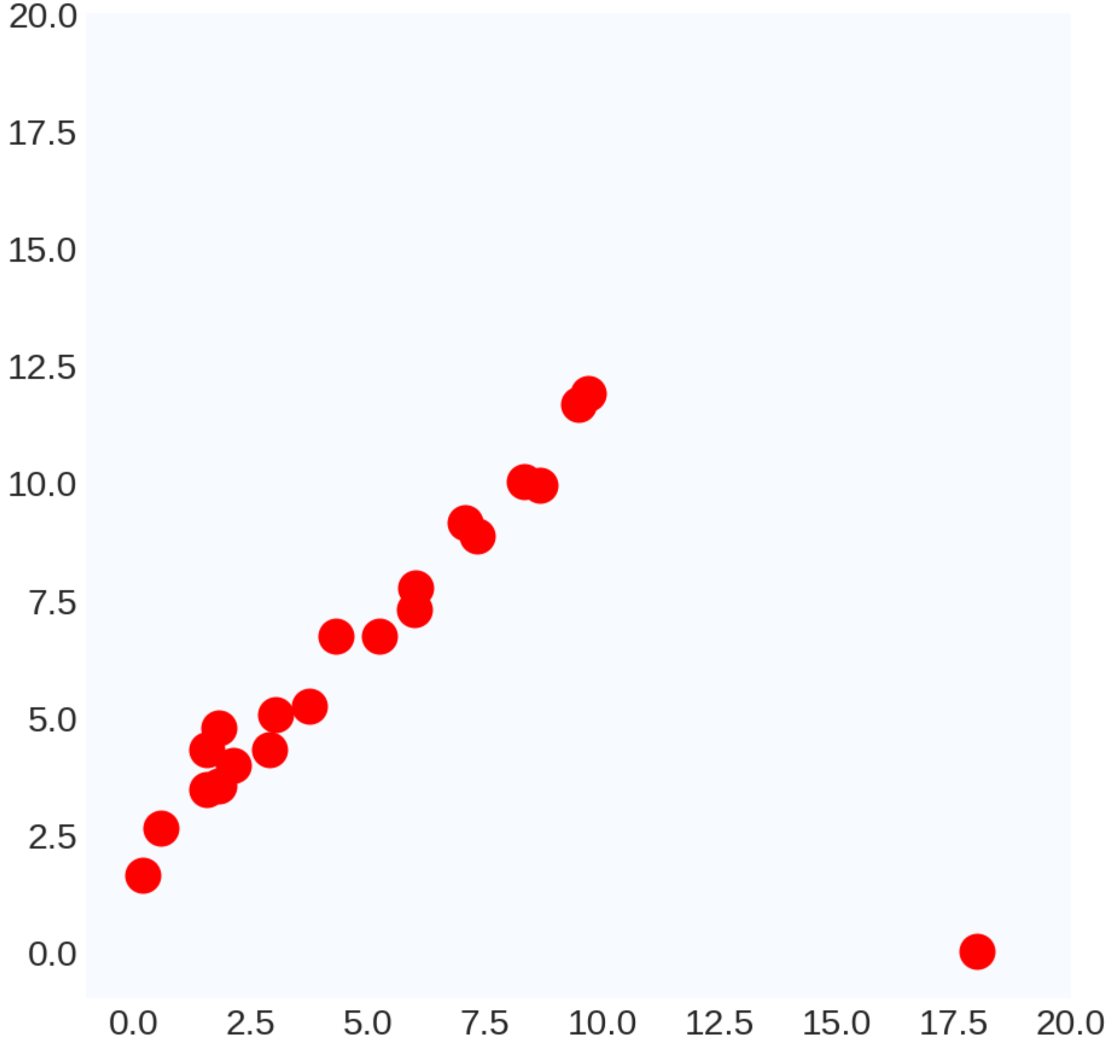
Running Least-Squares



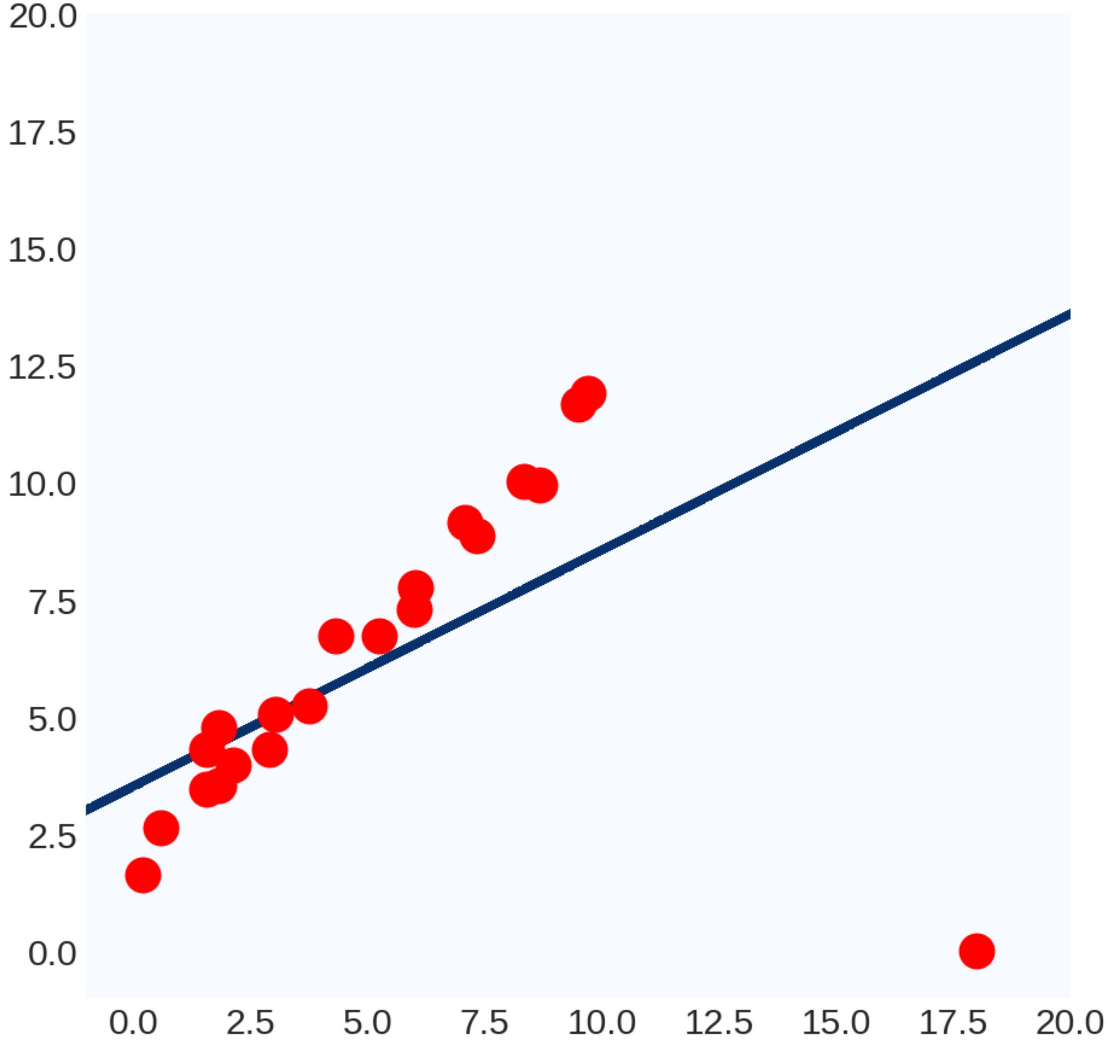
Running Least-Squares



Ruining Least Squares



Ruining Least Squares



Ruining Least Squares

Way to think of it #1:

$$||\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}||_2^2$$

100^2 >> 10^2: least-squares prefers having no large errors, even if the model is useless overall

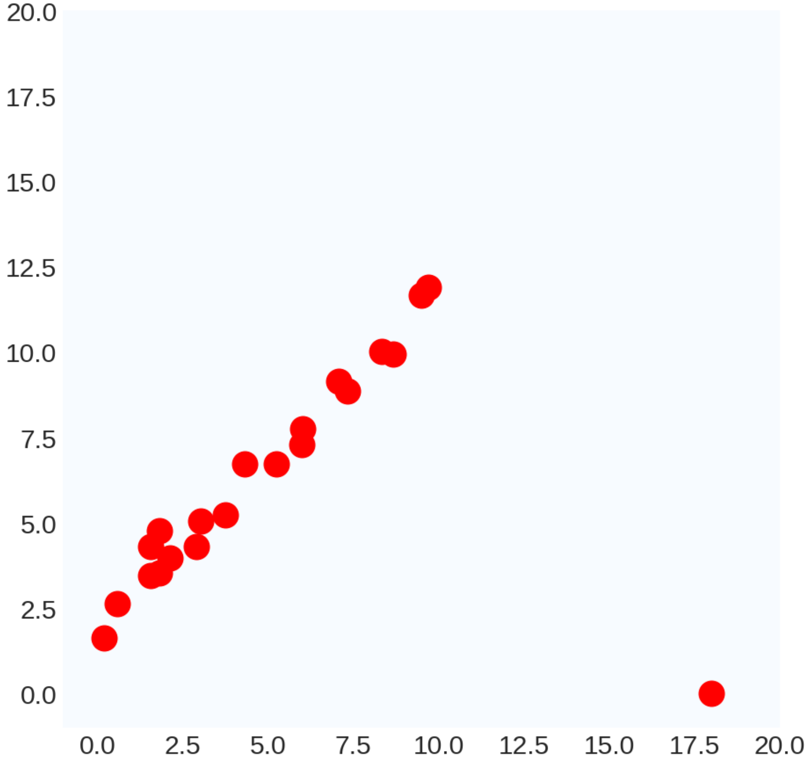
Way to think of it #2:

$$\boldsymbol{w} = \underline{\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T}\boldsymbol{Y}$$

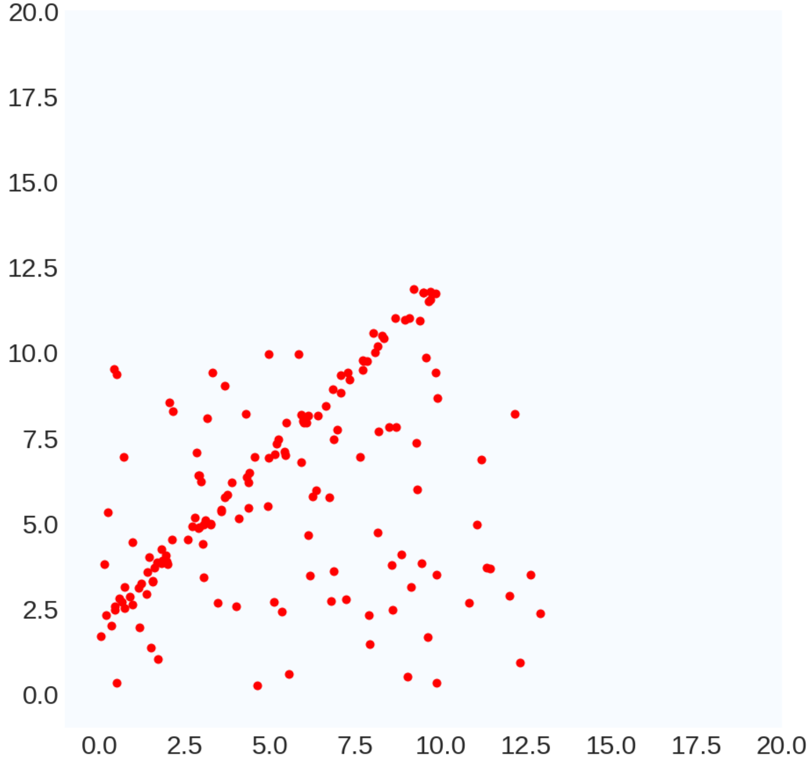
Weights are a linear transformation of the output variable: can manipulate W by manipulating Y.

Outliers in Computer Vision

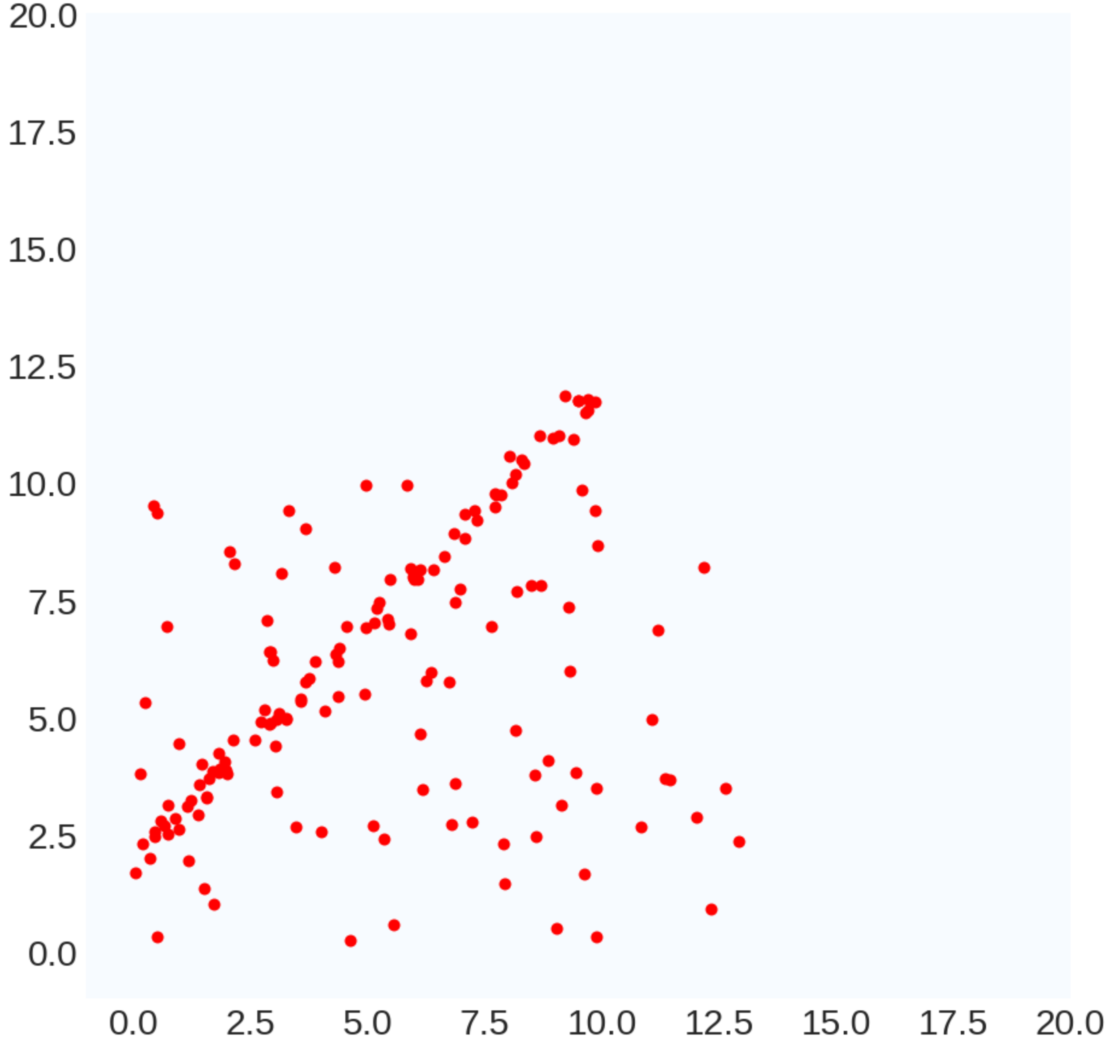
Single outlier: *rare*



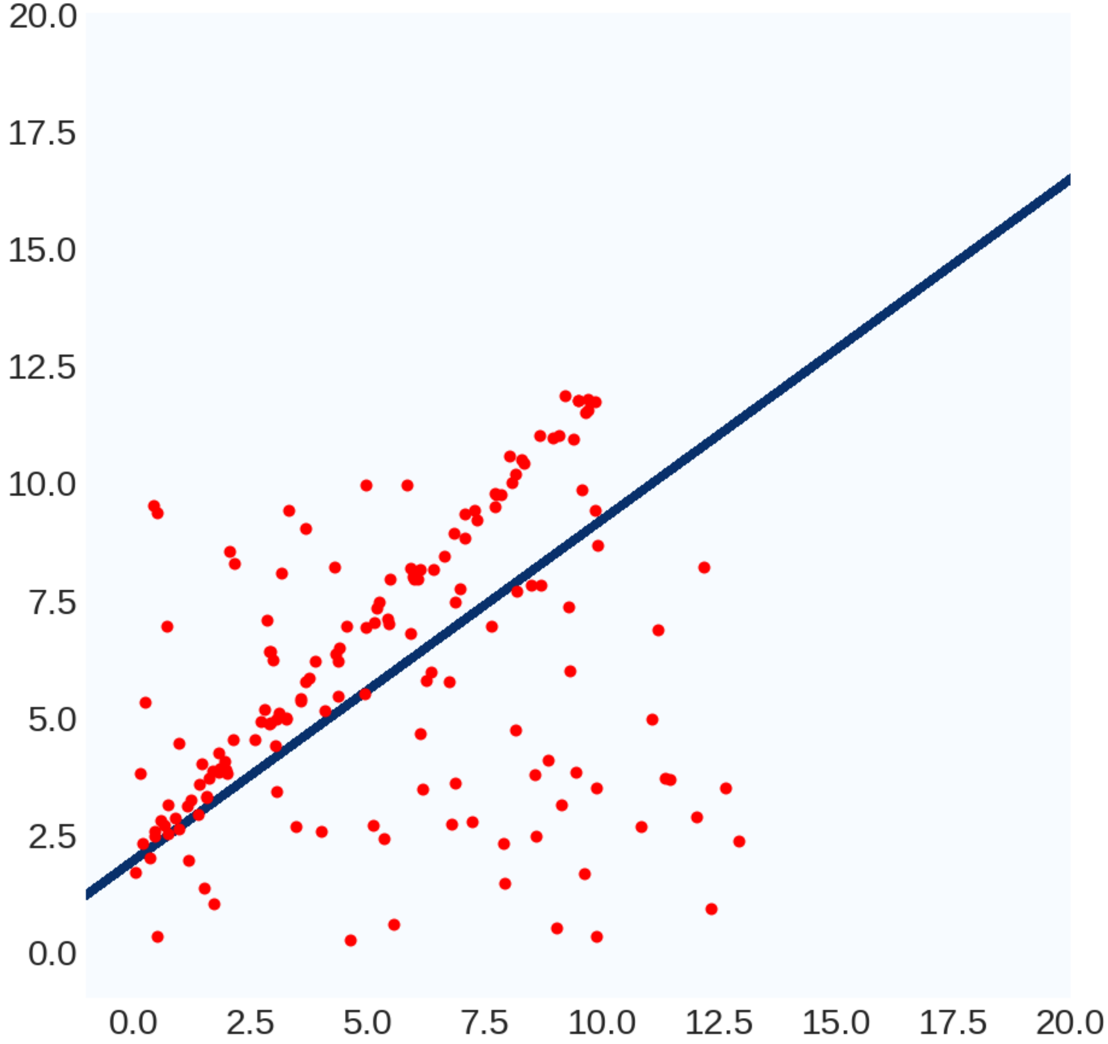
Many outliers: *common*



Ruining Least Squares Continued



Ruining Least Squares Continued



A Simple, Yet Clever Idea

- *What we really want*: model explains **many** points **“well”**
- *Least Squares*: model makes as few big mistakes as possible over the entire dataset

- *New objective*: find model for which error is “small” for as many data points as possible
- *Method*: RANSAC (**RA**ndom **SA**mple **C**onsensus)

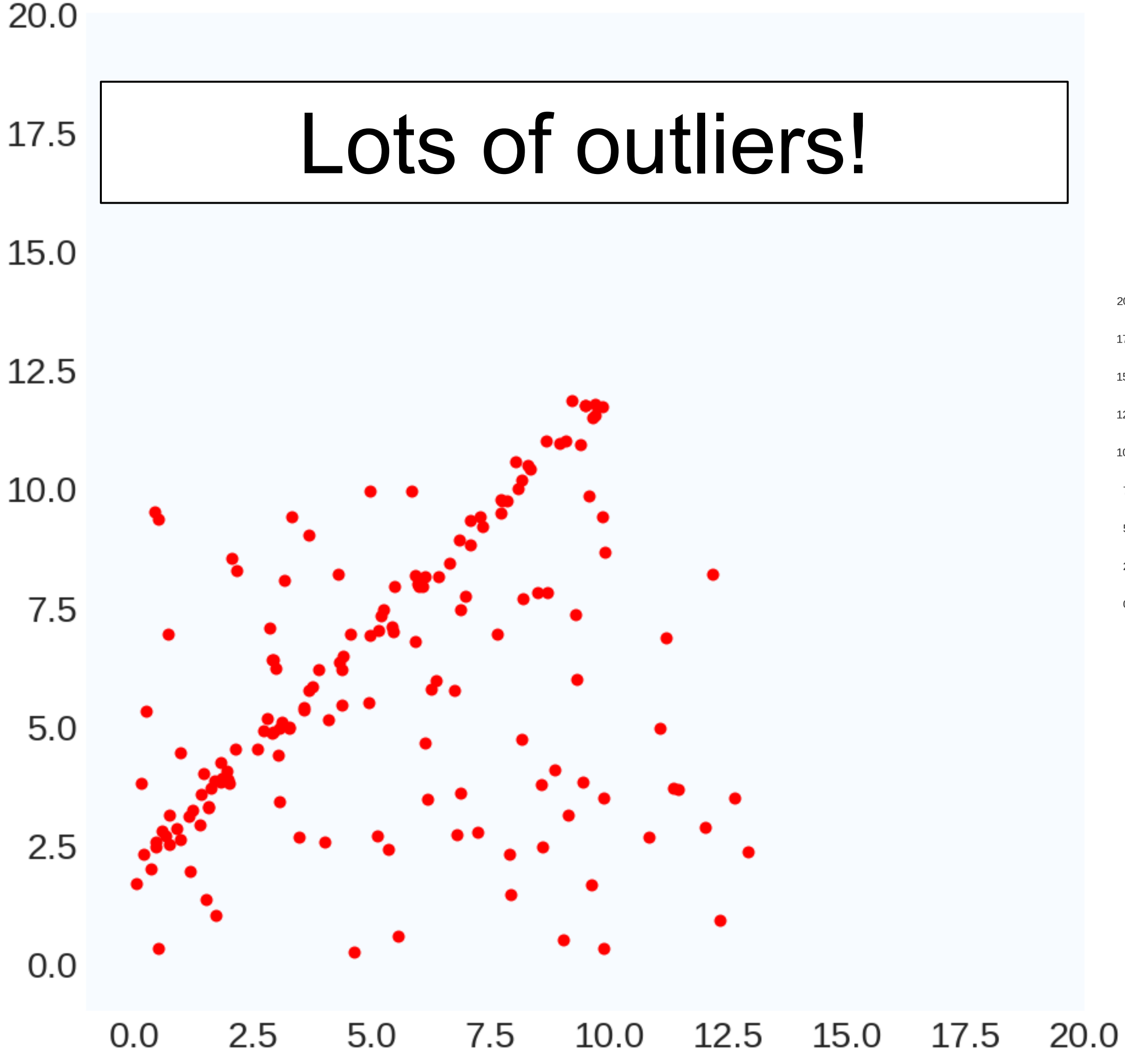
M. A. Fischler, R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. of the ACM, Vol 24, pp 381-395, 1981.

RANSAC For Lines

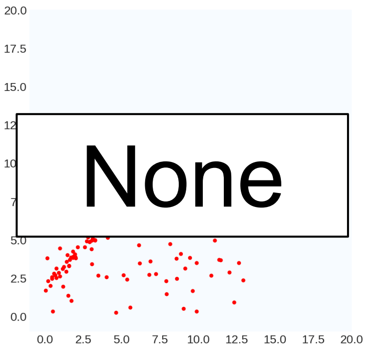
```
bestLine, bestCount = None, -1
for trial in range(numTrials):
        subset = pickPairOfPoints(data)
        line = totalLeastSquares(subset)
        E = linePointDistance(data,line)
        inliers = E < threshold
        if #inliers > bestCount:
                bestLine, bestCount = line, #inliers
```

Running RANSAC

Trial #1



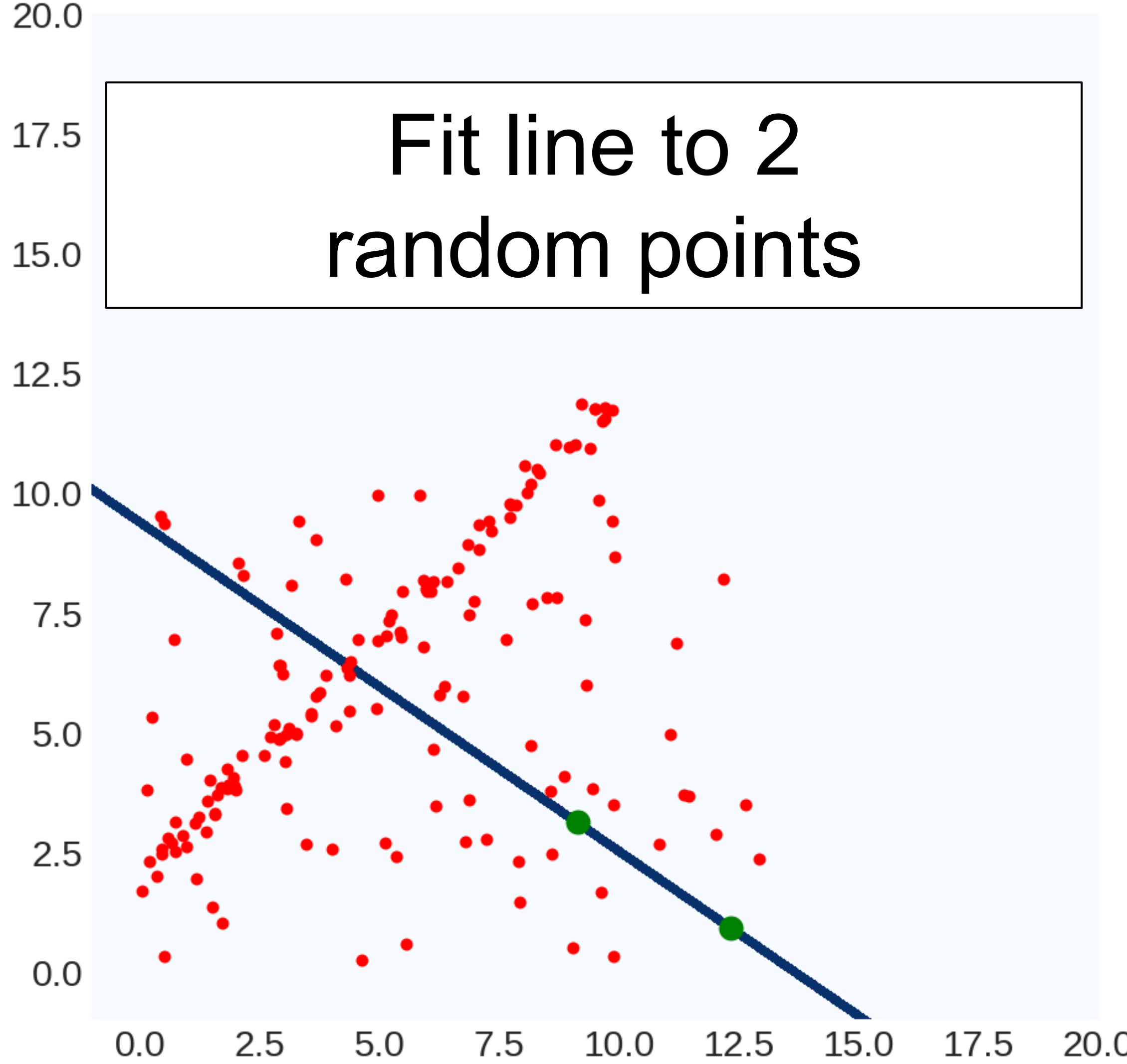
Best Model:



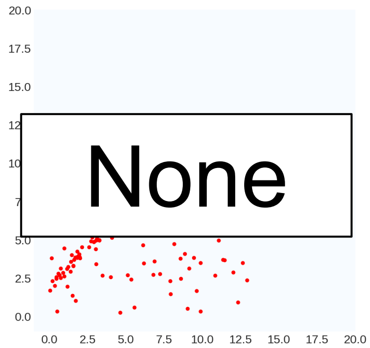
Best Count: -1

Running RANSAC

Trial #1



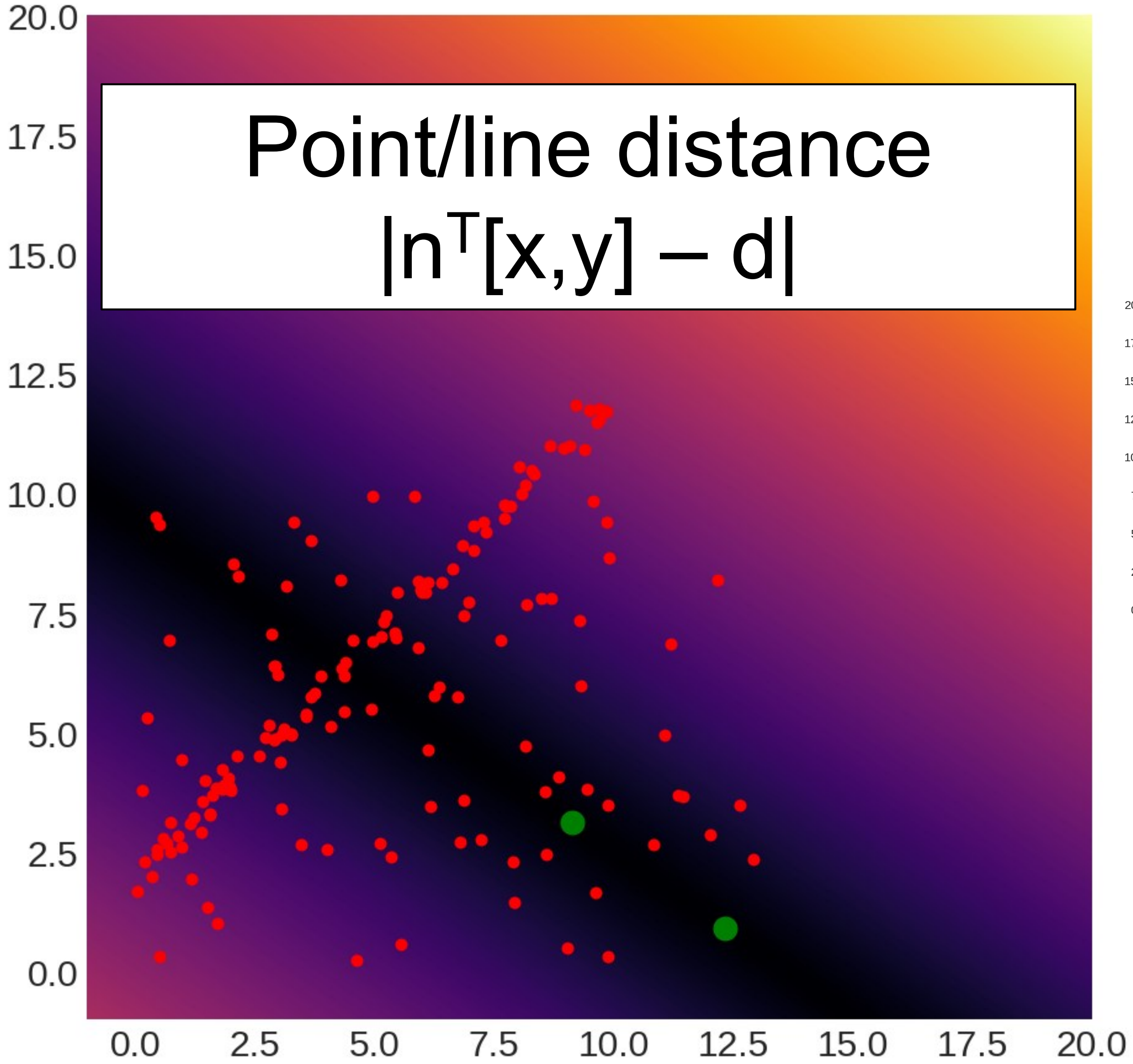
Best Model:



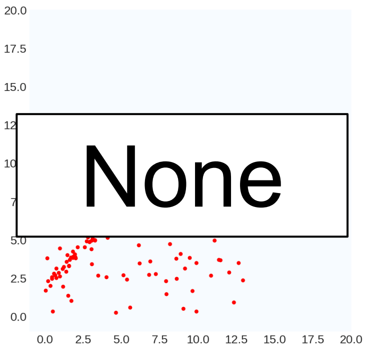
Best Count: -1

Running RANSAC

Trial #1



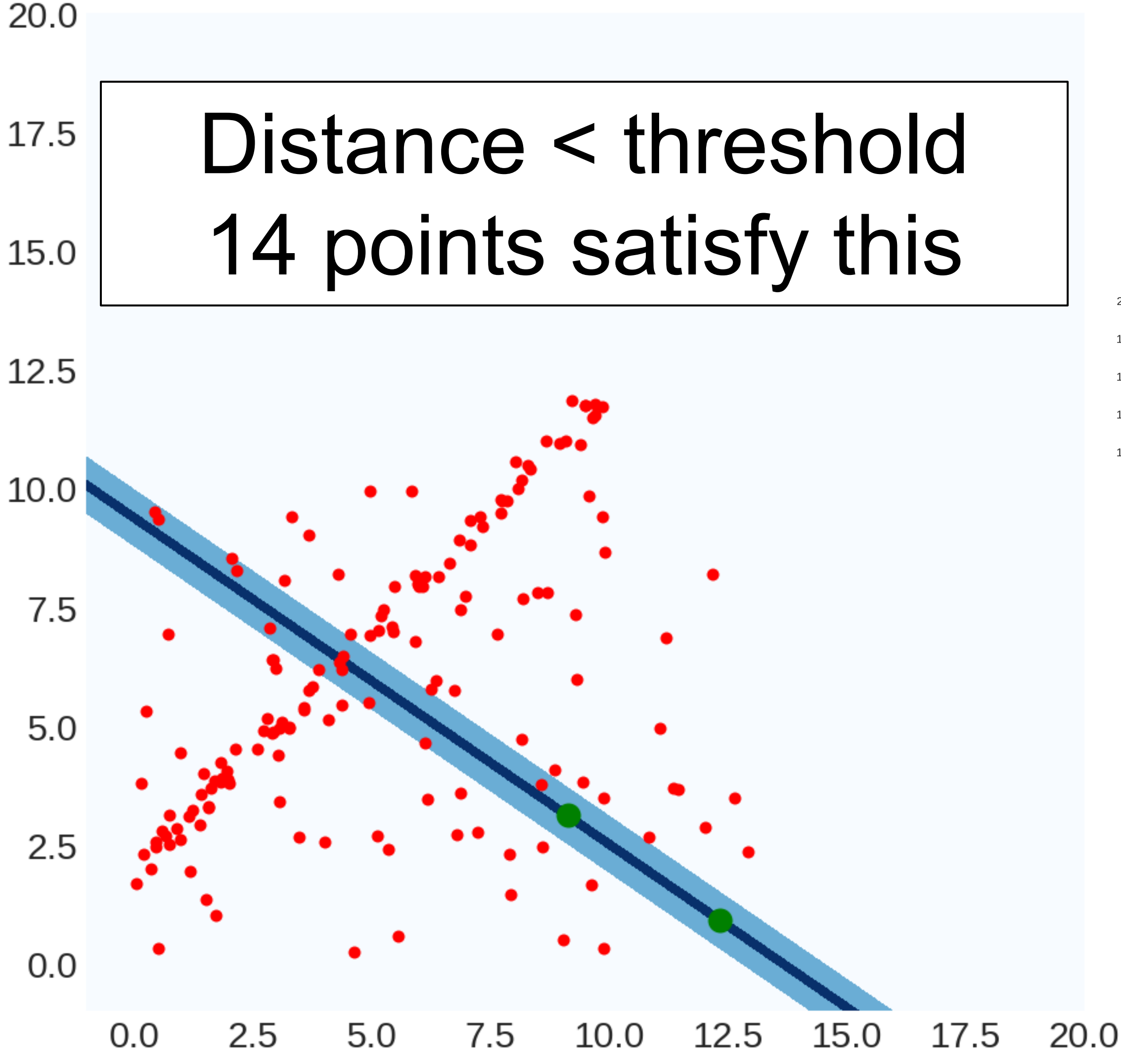
Best Model:



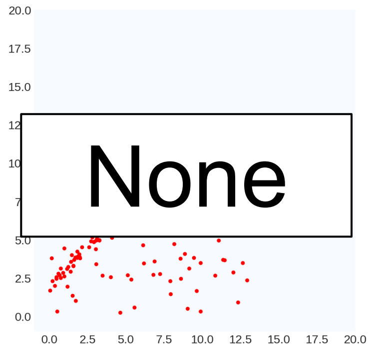
Best Count: -1

Running RANSAC

Trial #1



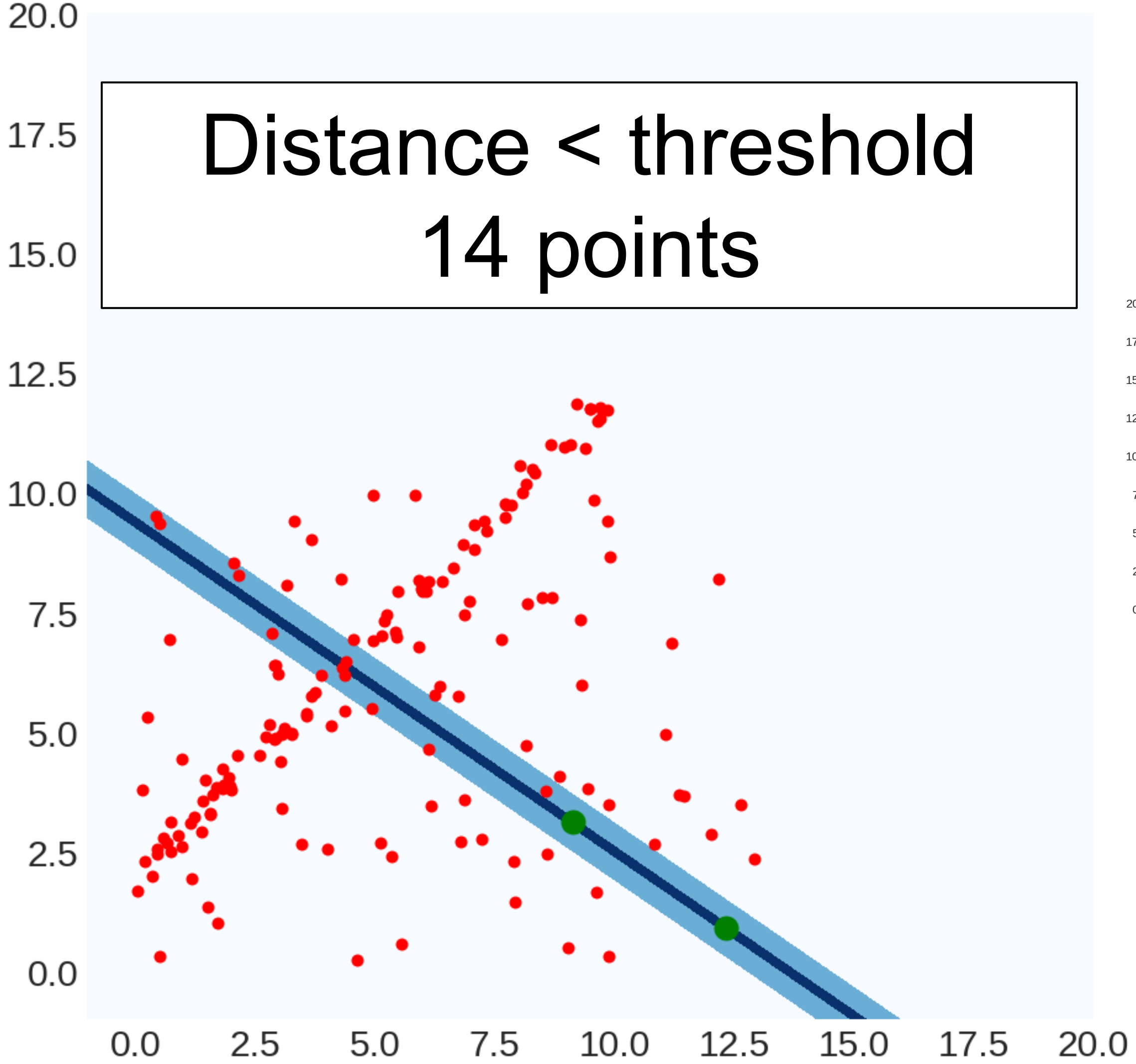
Best Model:



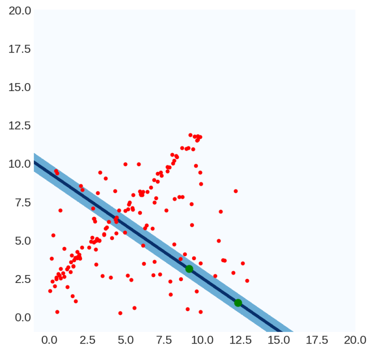
Best Count: -1

Running RANSAC

Trial #1



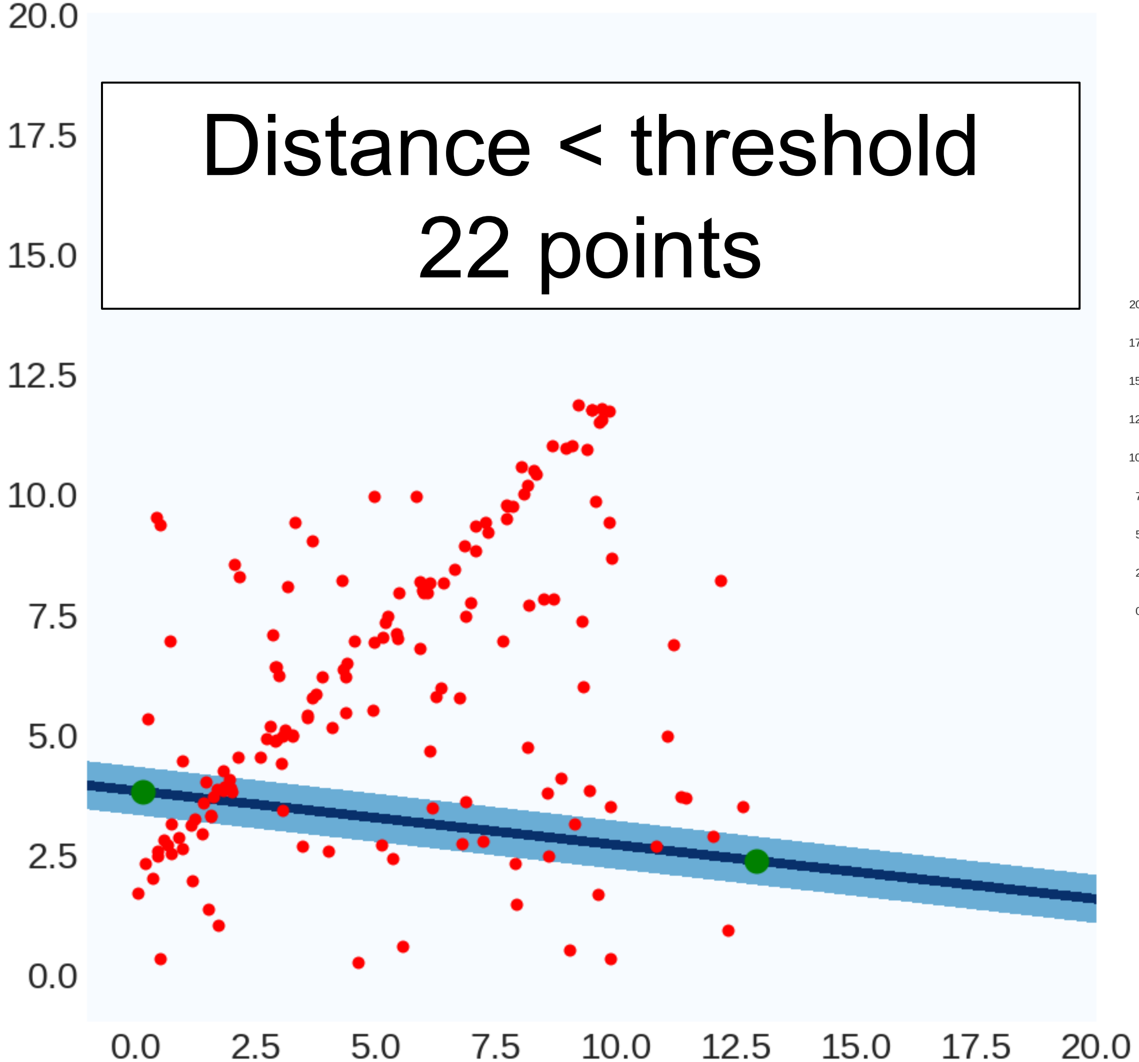
Best Model:



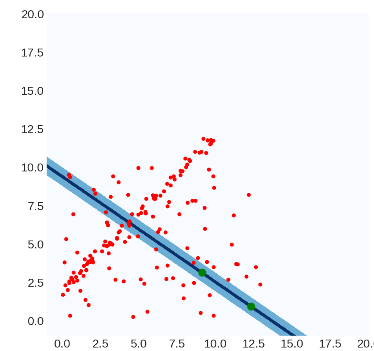
Best Count: 14

Running RANSAC

Trial #2



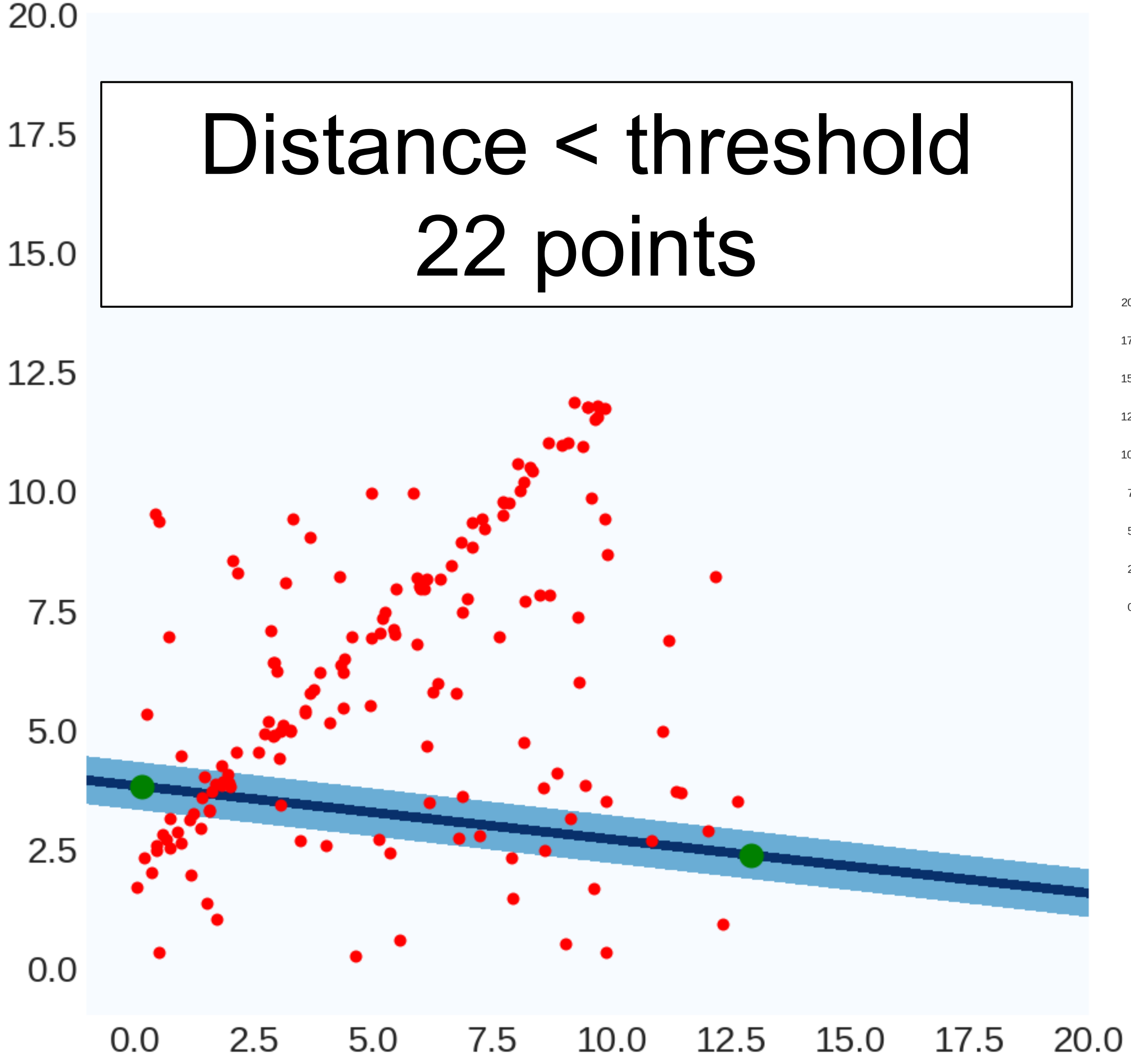
Best Model:



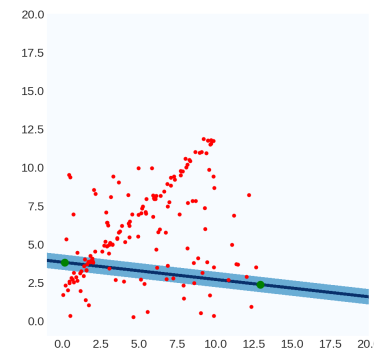
Best Count: 14

Running RANSAC

Trial #2



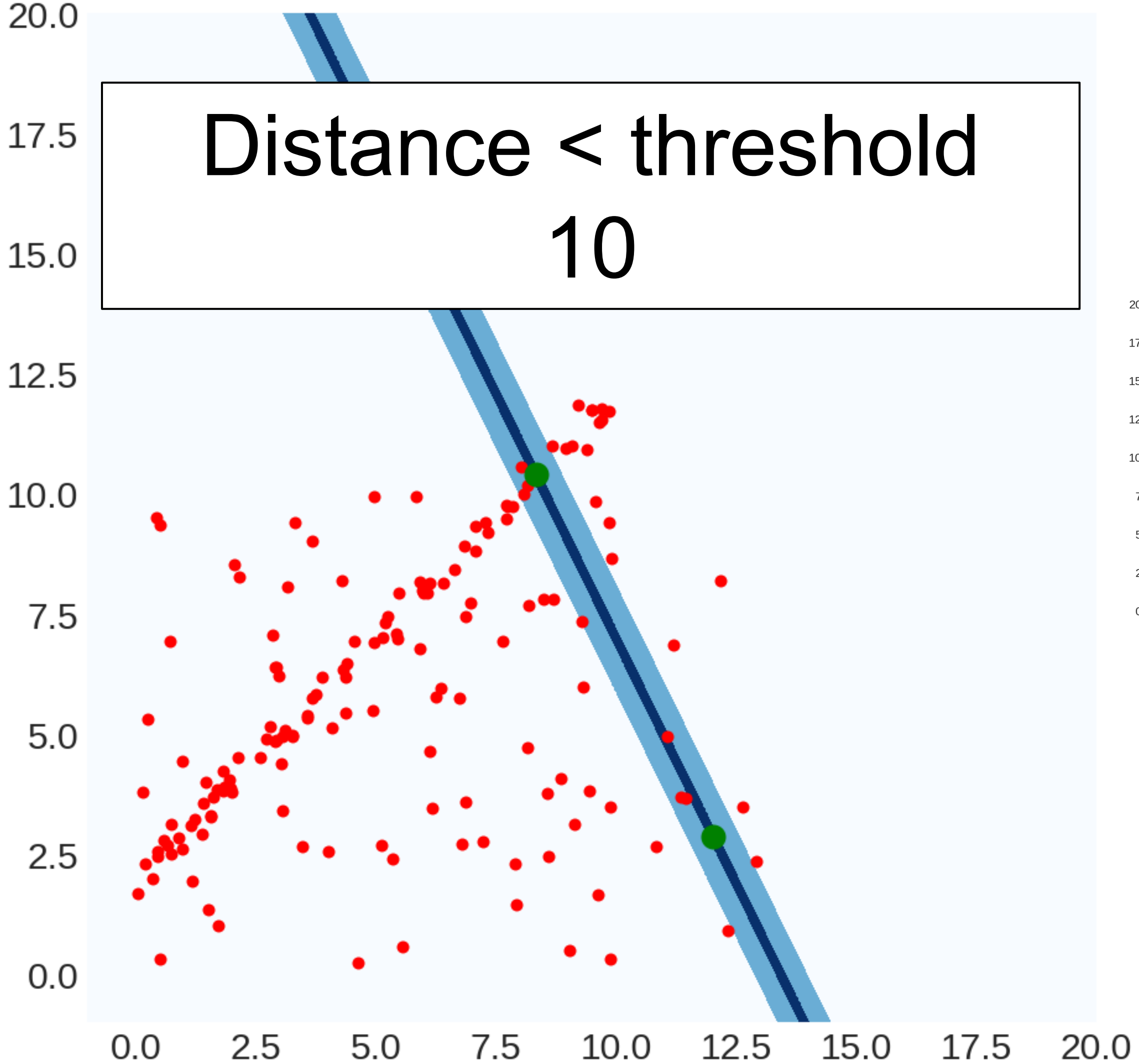
Best Model:



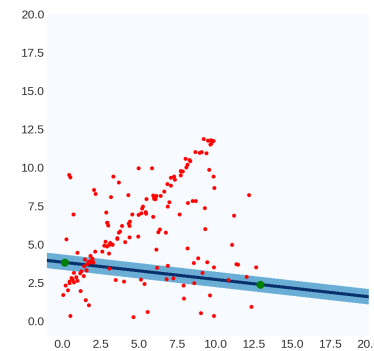
Best Count: 22

Running RANSAC

Trial #3



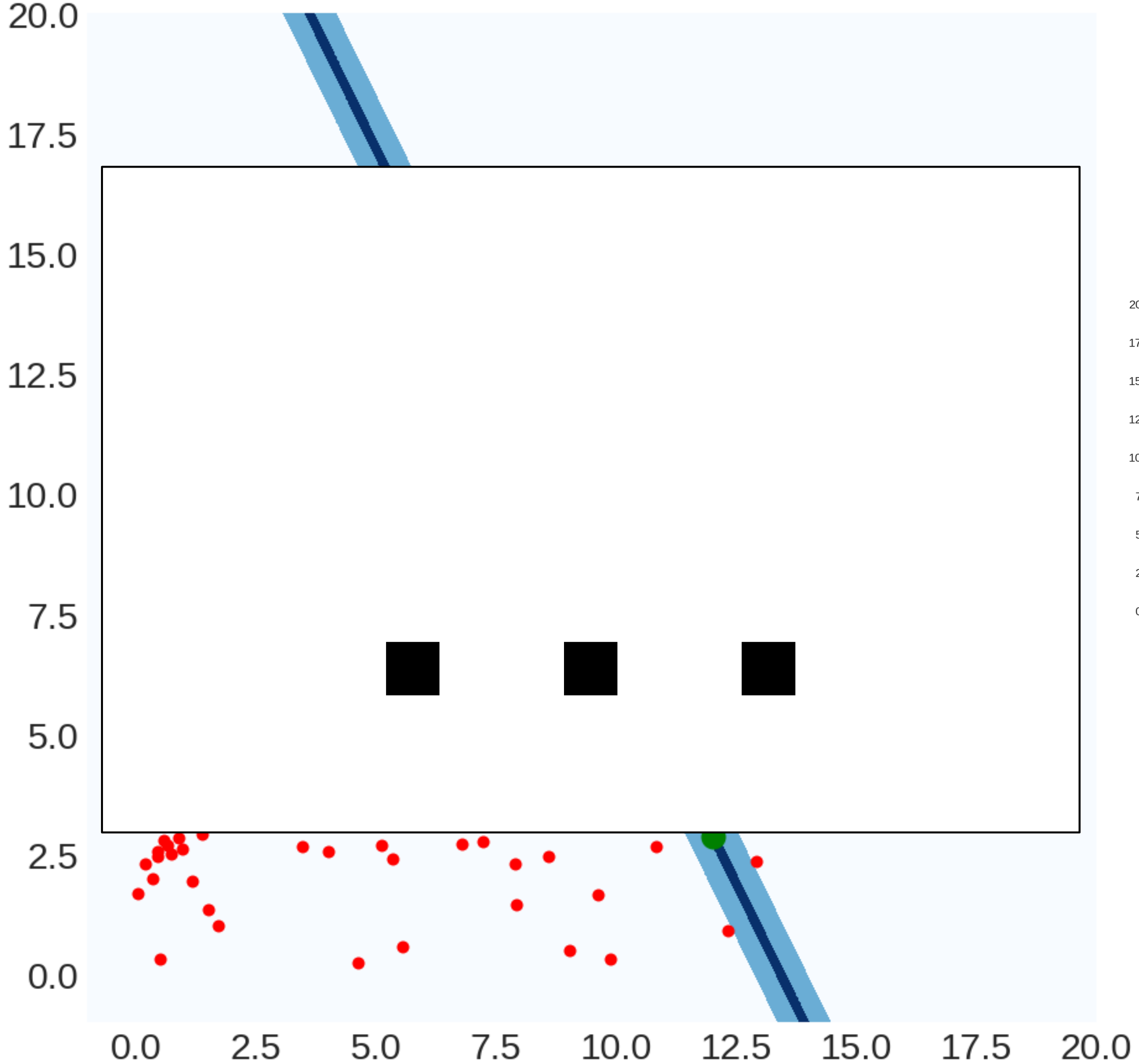
Best Model:



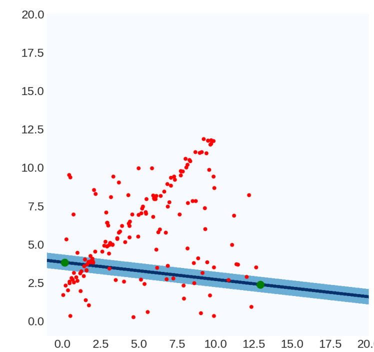
Best Count: 22

Running RANSAC

Trial #3



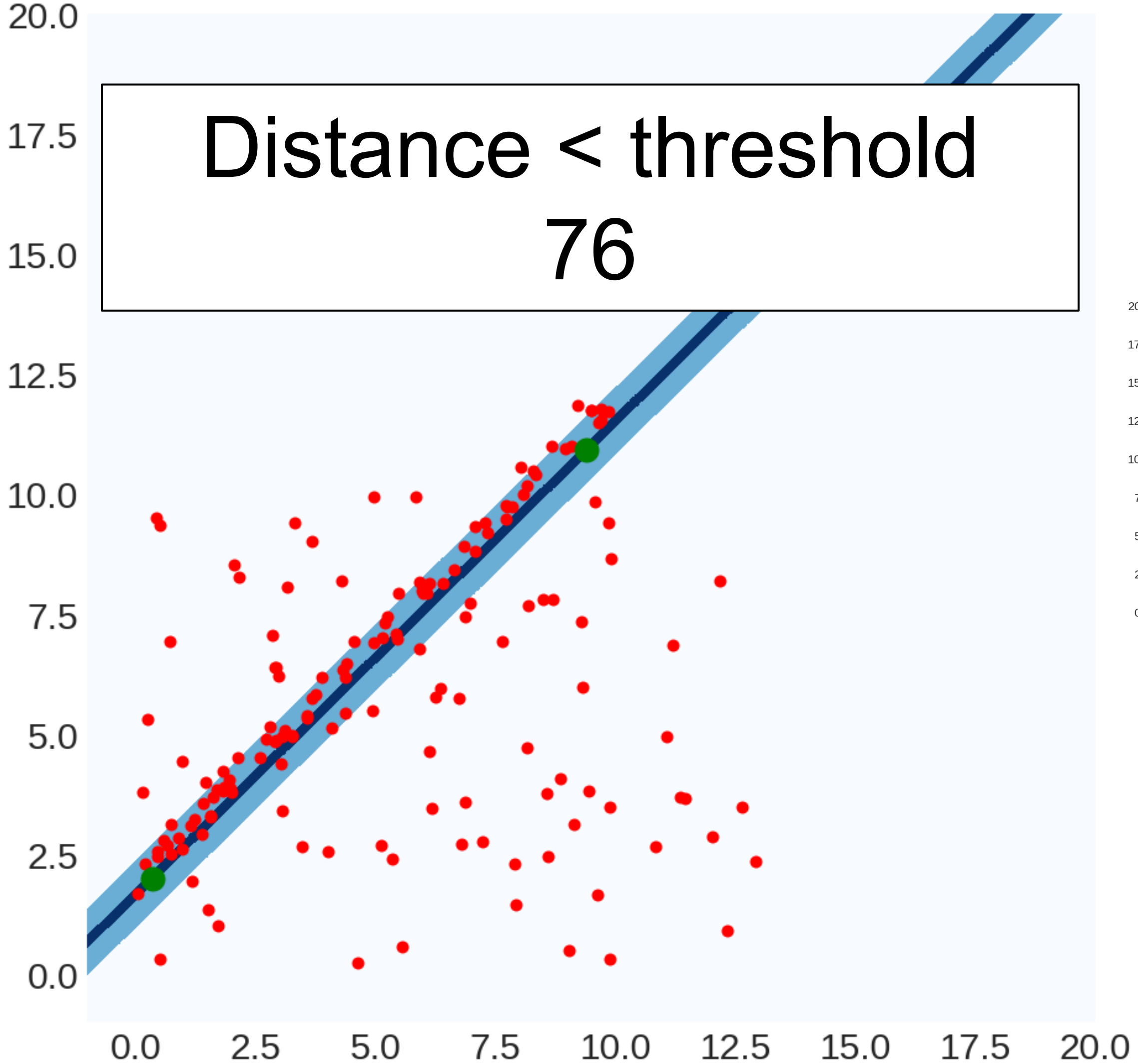
Best Model:



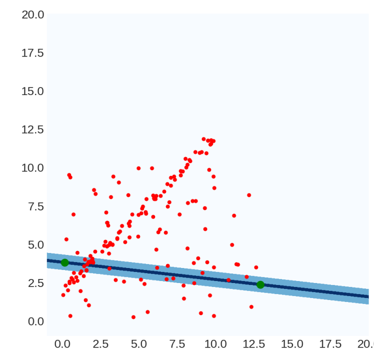
Best Count: 22

Running RANSAC

Trial #9



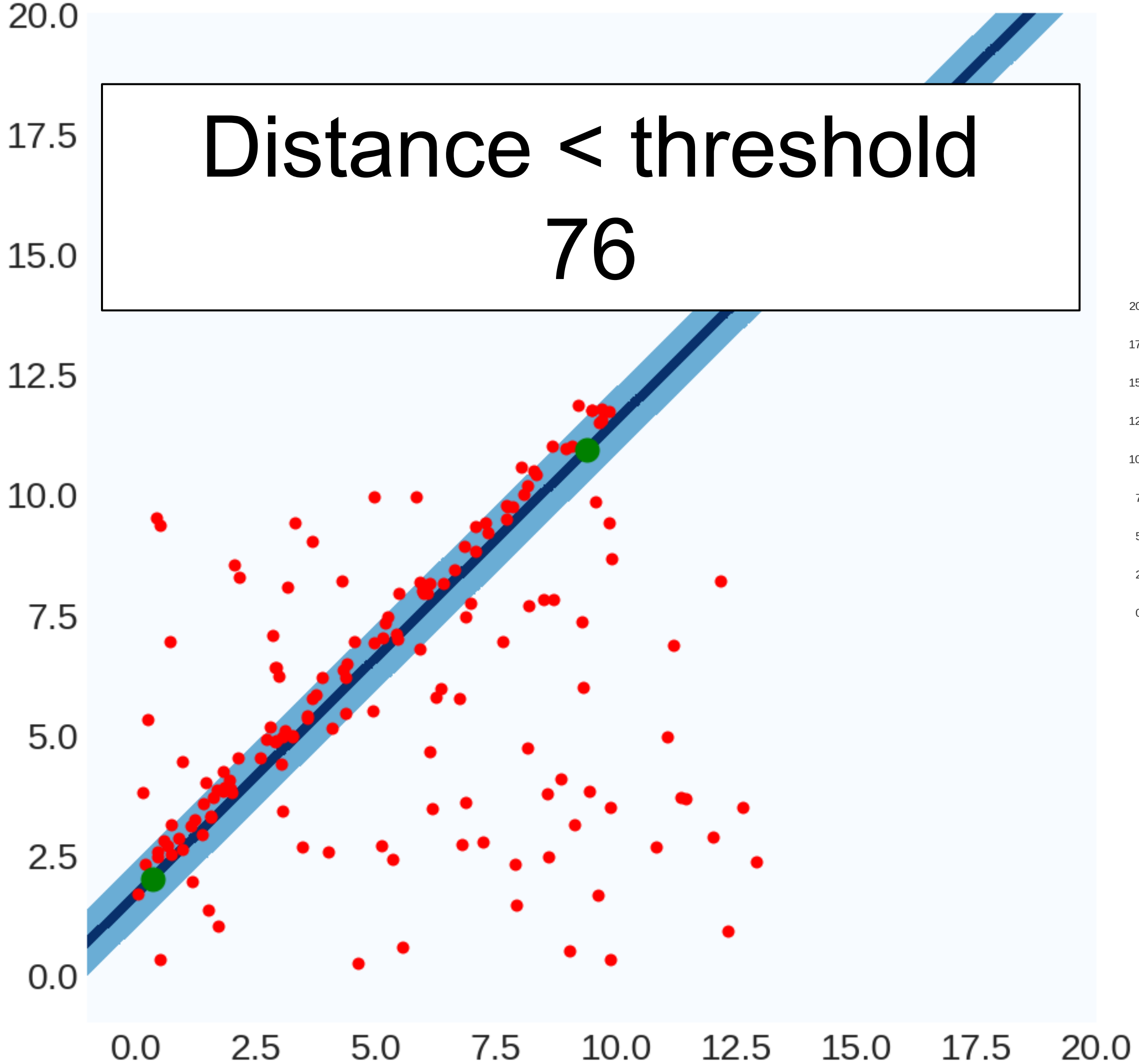
Best Model:



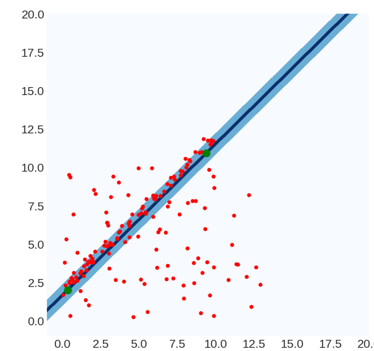
Best Count: 22

Running RANSAC

Trial #9



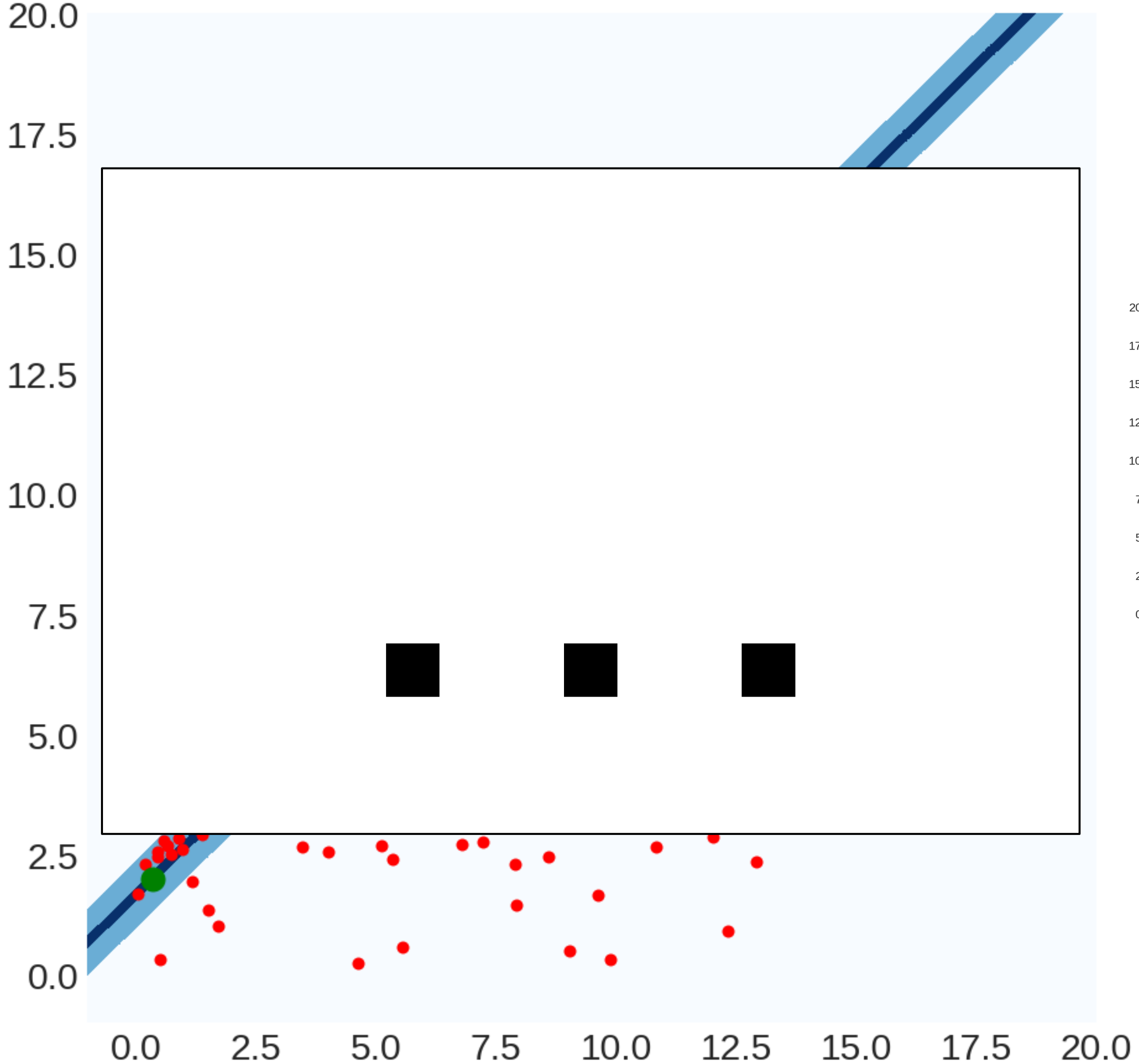
Best Model:



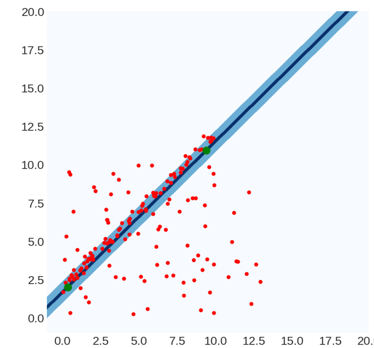
Best Count: 76

Running RANSAC

Trial #9



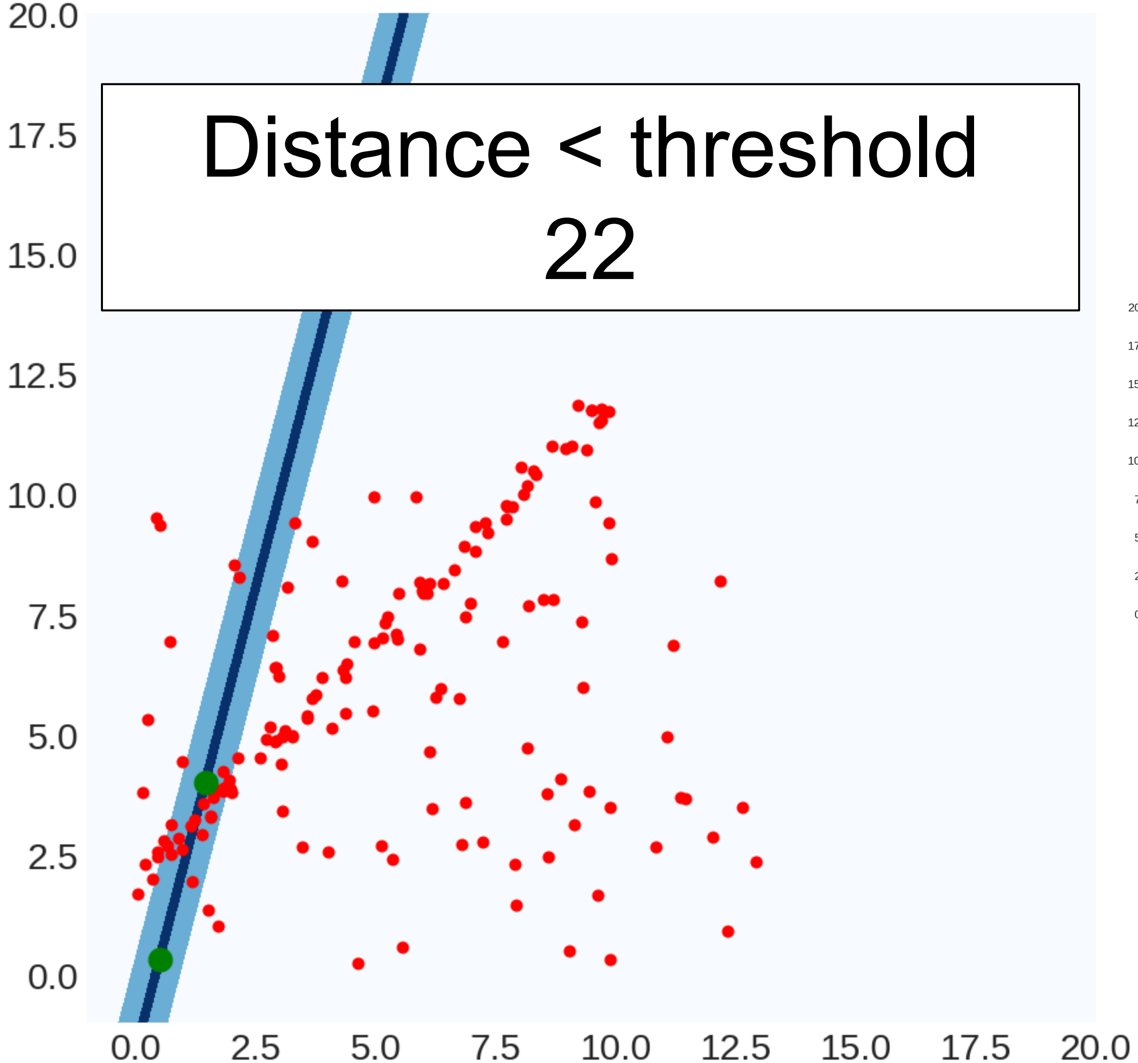
Best Model:



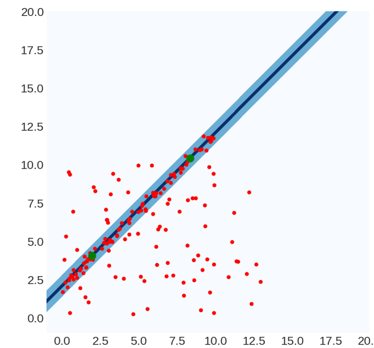
Best Count: 76

Running RANSAC

Trial #100

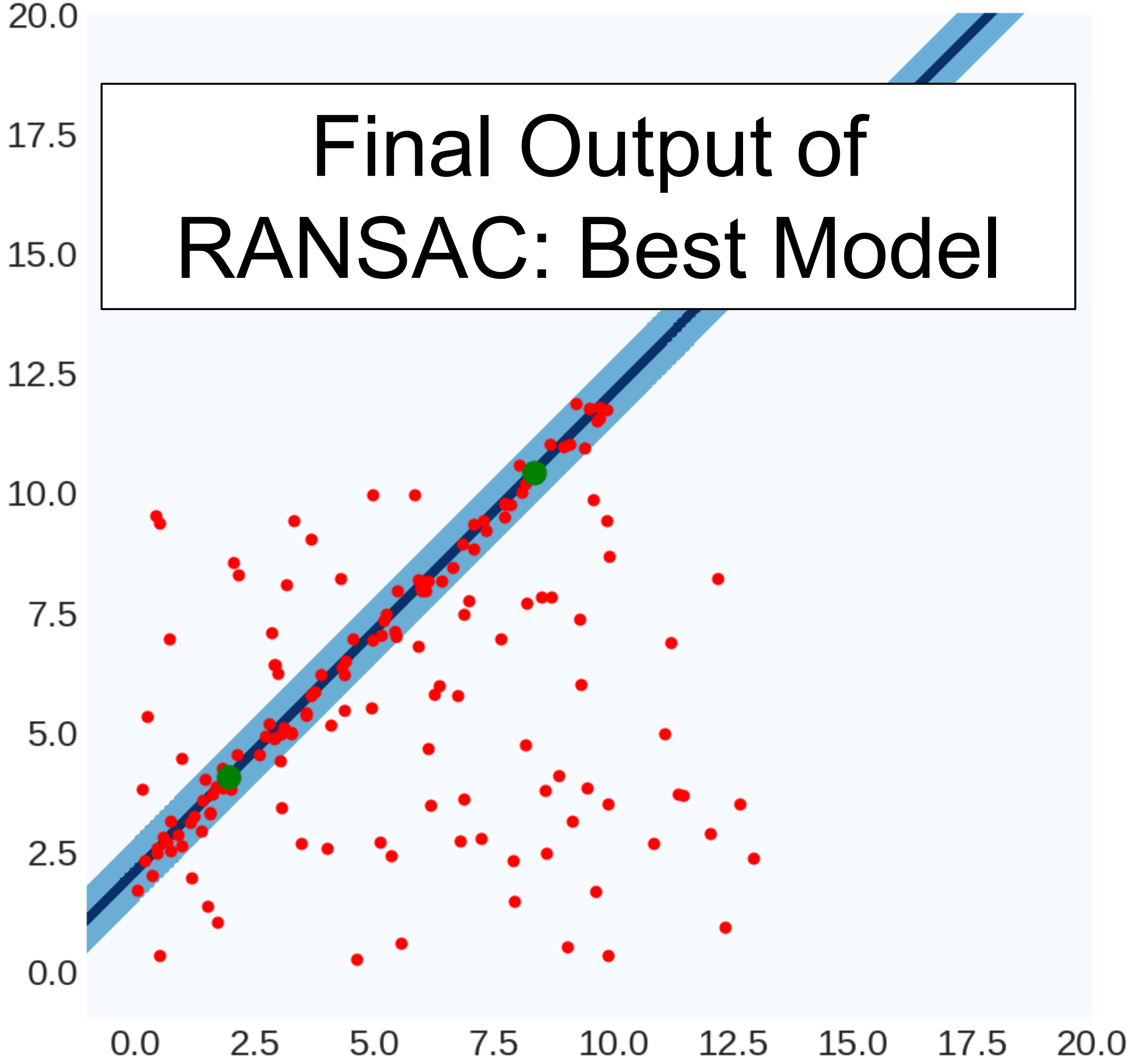


Best Model:



Best Count: 85

Running RANSAC



RANSAC In General

```
best, bestCount = None, -1
for trial in range(NUM_TRIALS):
        subset = pickSubset(data,SUBSET_SIZE)
        model = fitModel(subset)
        E = computeError(data,line)
        inliers = E < THRESHOLD
        if #(inliers) > bestCount:
                best, bestCount = model, #(inliers)
(often refit on the inliers for best model)
```

Parameters – Num Trials

Suppose
- r is the fraction of outliers (e.g., 80%)
- we pick s points (e.g., 2)
- we run RANSAC N times (e.g., 500)

What's the probability of picking a sample set with no outliers?

$$\approx (1 - r)^s$$ **(4%)**

What's the probability of picking a sample set with any outliers?

$$1 - (1 - r)^s$$ **(96%)**

Parameters – Num Trials

Suppose r is the fraction of outliers (e.g., 80%)
we pick s points (e.g., 2)
we run RANSAC N times (e.g., 500)

What's the probability of picking a sample set with any outliers?

$$1-(1-r)^s$$ **(96%)**

What's the probability of picking only sample sets with outliers?

$$(1-(1-r)^s)^N$$ **(10^{-7}% N=500)**

(13% N=50)

What's the probability of picking any set with inliers?

$$1-(1-(1-r)^s)^N$$

Parameters – Num Trials

P($157M Jackpot):
1 / 302,575,350

Death by vending machine

P(Death):
≈1 / 112,000,000

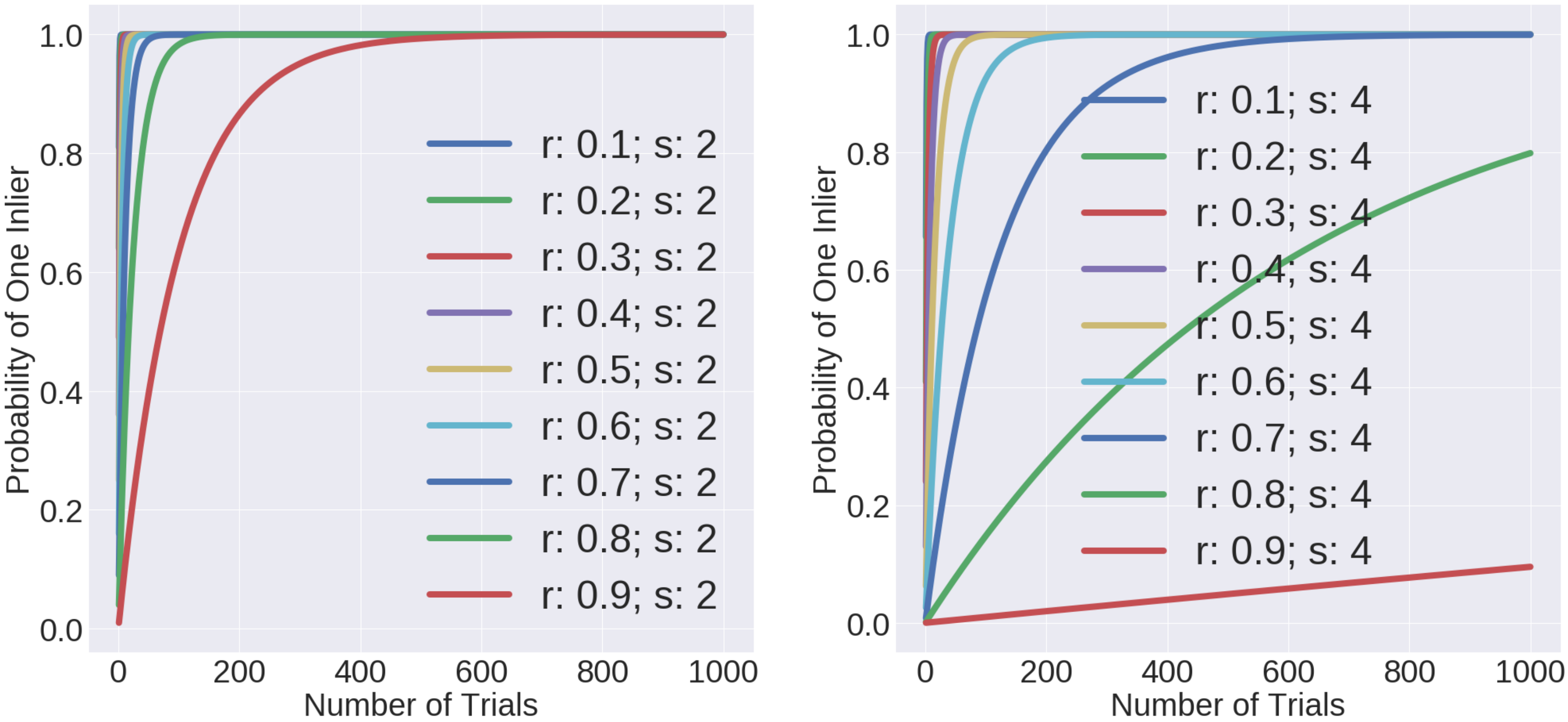
RANSAC fails to fit a line with 80% outliers after trying only 500 times

P(Failure):
1 / 731,784,961

Odds/Jackpot amount from 2/7/2019 megamillions.com, unfortunate demise odds from livescience.com

Parameters – Num Trials

Suppose *r* is the fraction of outliers (e.g., 80%)
we pick *s* points (e.g., 2)
we run RANSAC *N* times (e.g., 500)



Parameters – Subset Size

- Always the smallest possible set for fitting the model.
- Minimum number for lines: 2 data points
- Minimum number for 3D planes: **how many?**

- **Why the minimum intuitively?**
- You’ll find out more precisely in homework 3.

Parameters – Threshold

- No magical threshold

RANSAC Pros and Cons

Pros

1. Ridiculously simple
2. Ridiculously effective
3. Works in general

Cons

1. Have to tune parameters
2. No theory (so can’t derive parameters via theory)
3. Not magic, especially with lots of outliers

List credit: S. Lazebnik

Hough Transform

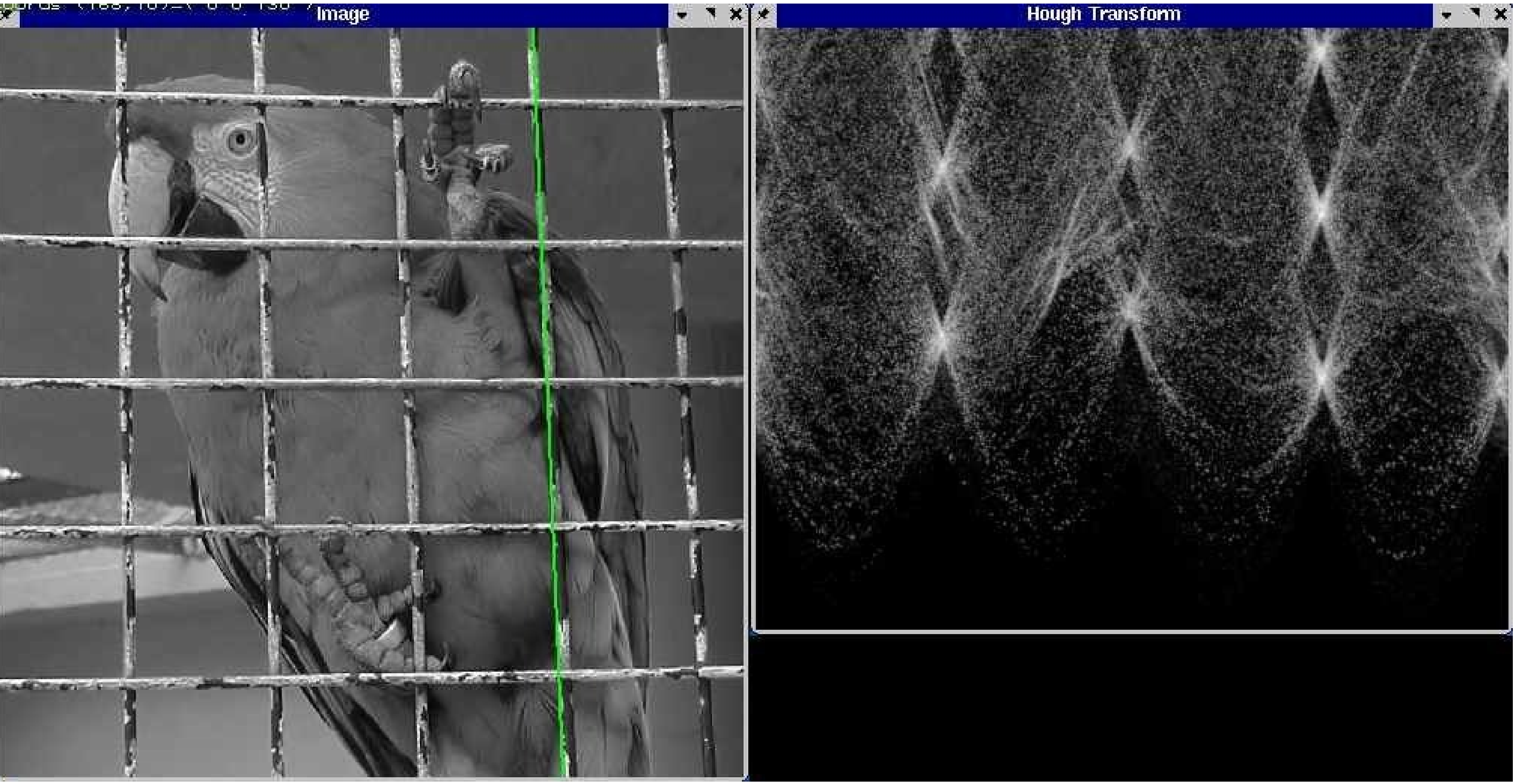
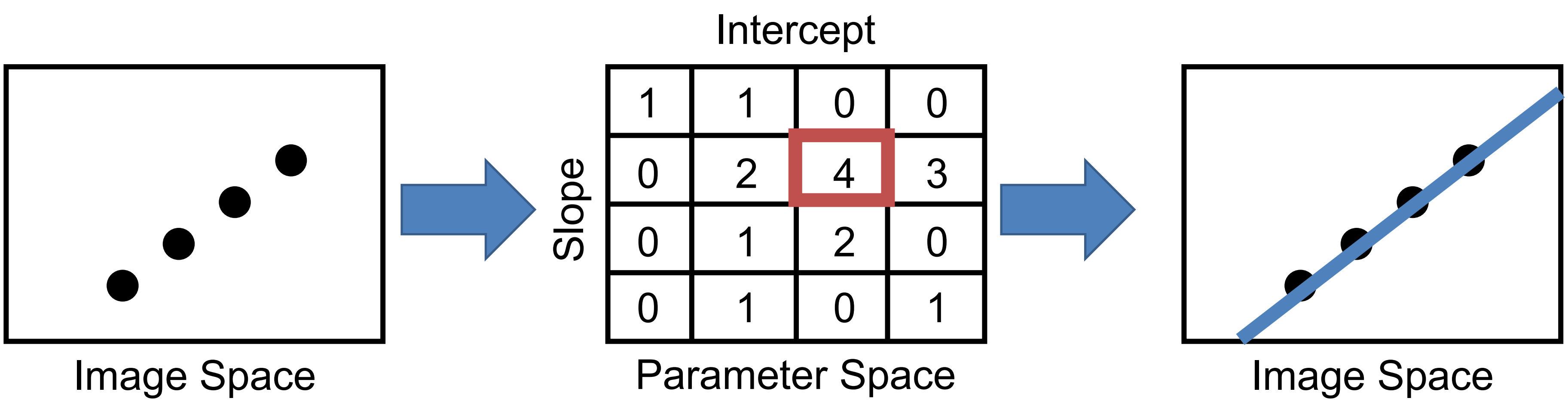


Image credit: S. Lazebnik

Hough Transform

1. Discretize space of parametric models
2. Each pixel votes for all compatible models
3. Find models compatible with many pixels



P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures,* Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

Slide design credit: S. Lazebnik

Hough Transform

Line in image = point in parameter space

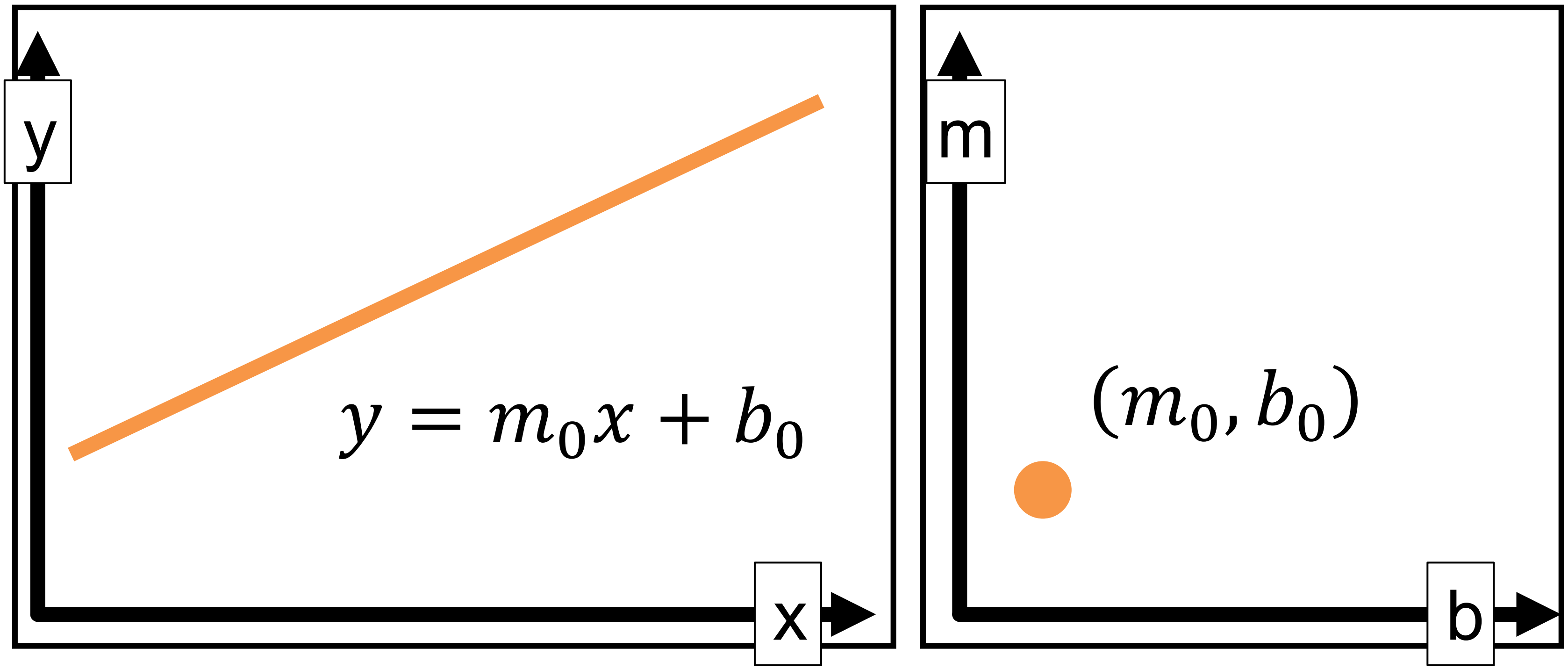


Image Space

Parameter Space

Diagram is remake of S. Seitz Slides; these are illustrative and values may not be real

Hough Transform

Point in image = line in parameter space

All lines through the point: $b = x_0 m + y_0$

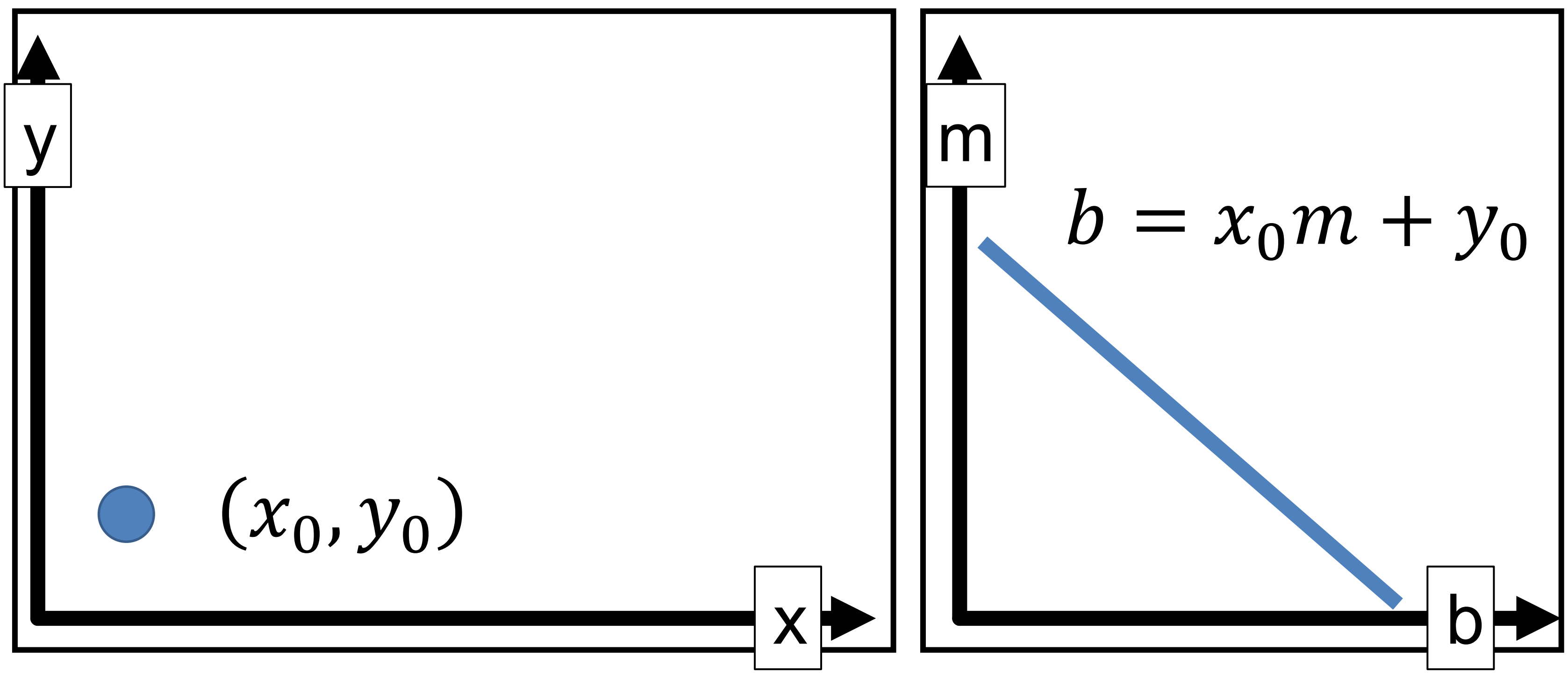


Image Space

Parameter Space

Diagram is remake of S. Seitz Slides; these are illustrative and values may not be real

Hough Transform

Point in image = line in parameter space

All lines through the point: $b = x_1 m + y_1$

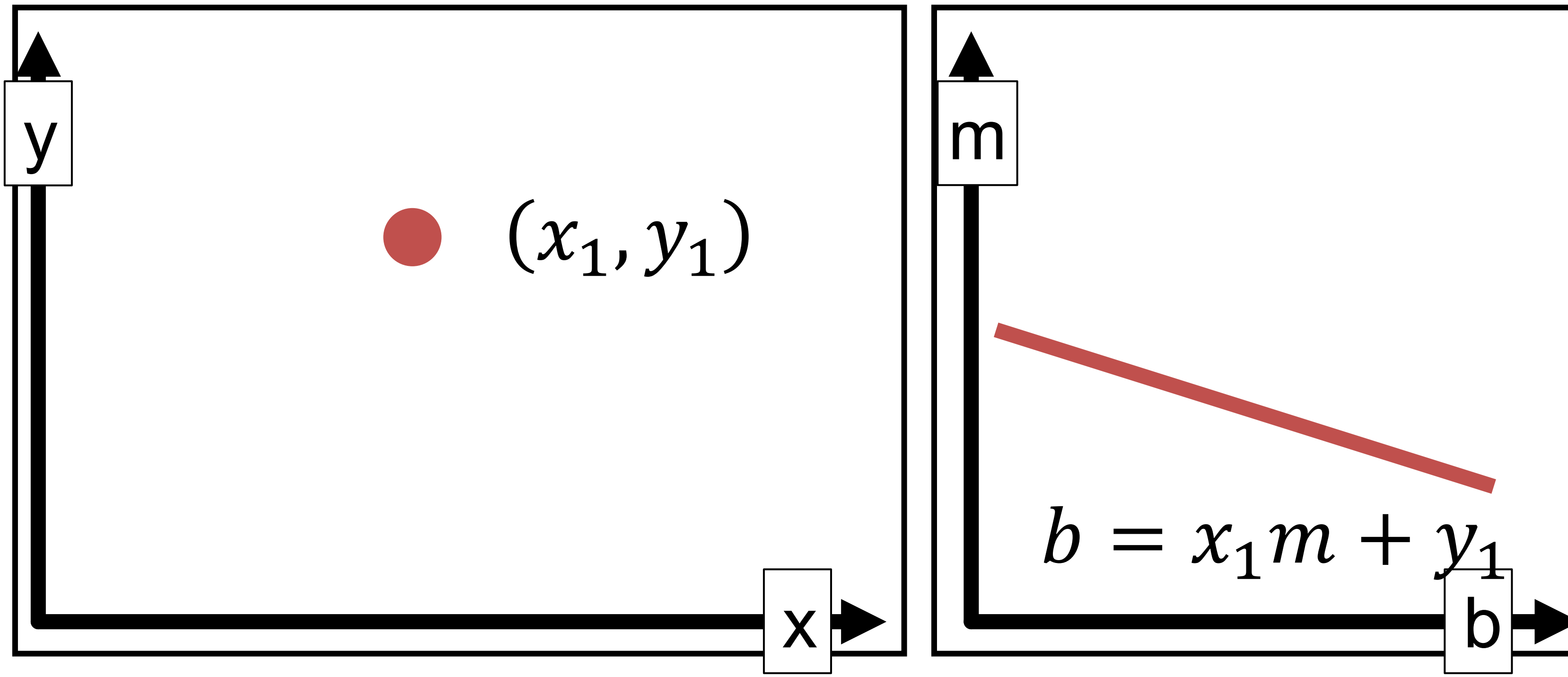


Image Space

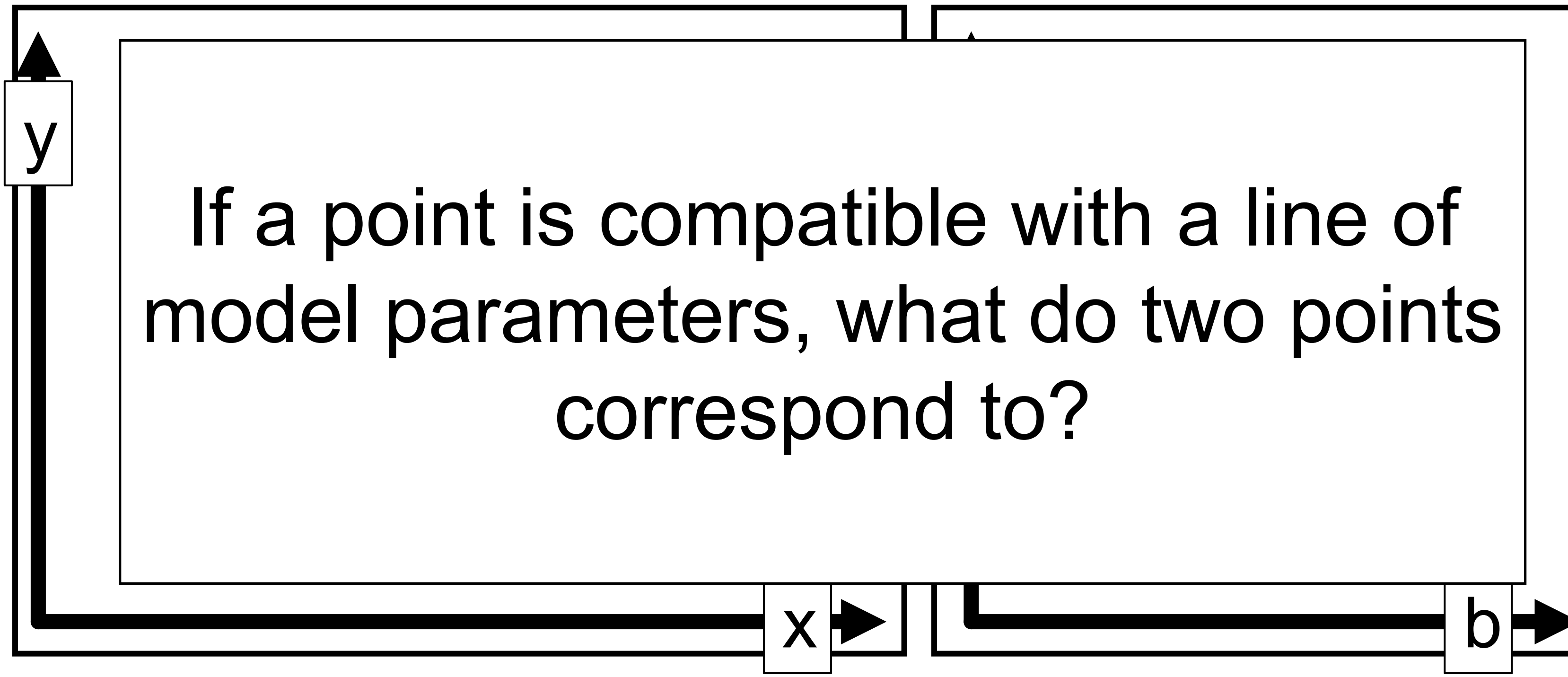
Parameter Space

Diagram is remake of S. Seitz Slides; these are illustrative and values may not be real

Hough Transform

Point in image = line in parameter space

All lines through the point: $b = x_1 m + y_1$



If a point is compatible with a line of model parameters, what do two points correspond to?

Image Space

Parameter Space

Diagram is remake of S. Seitz Slides; these are illustrative and values may not be real

Hough Transform

Line through two points in image = intersection of two lines in parameter space (i.e., solutions to both equations)

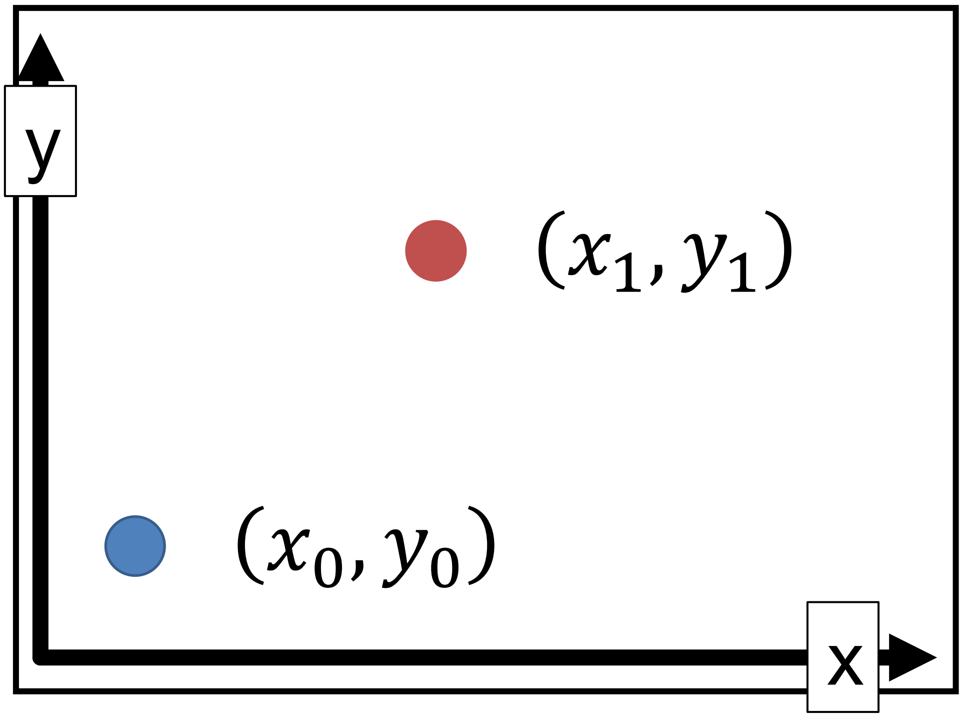
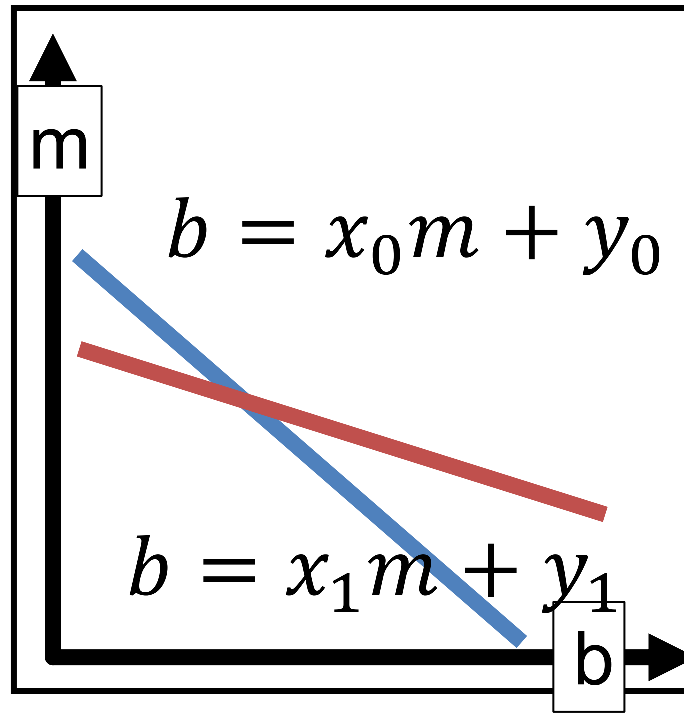


Image Space



Parameter Space

Diagram is remake of S. Seitz Slides; these are illustrative and values may not be real

Hough Transform

Line through two points in image = intersection of two lines in parameter space (i.e., solutions to both equations)

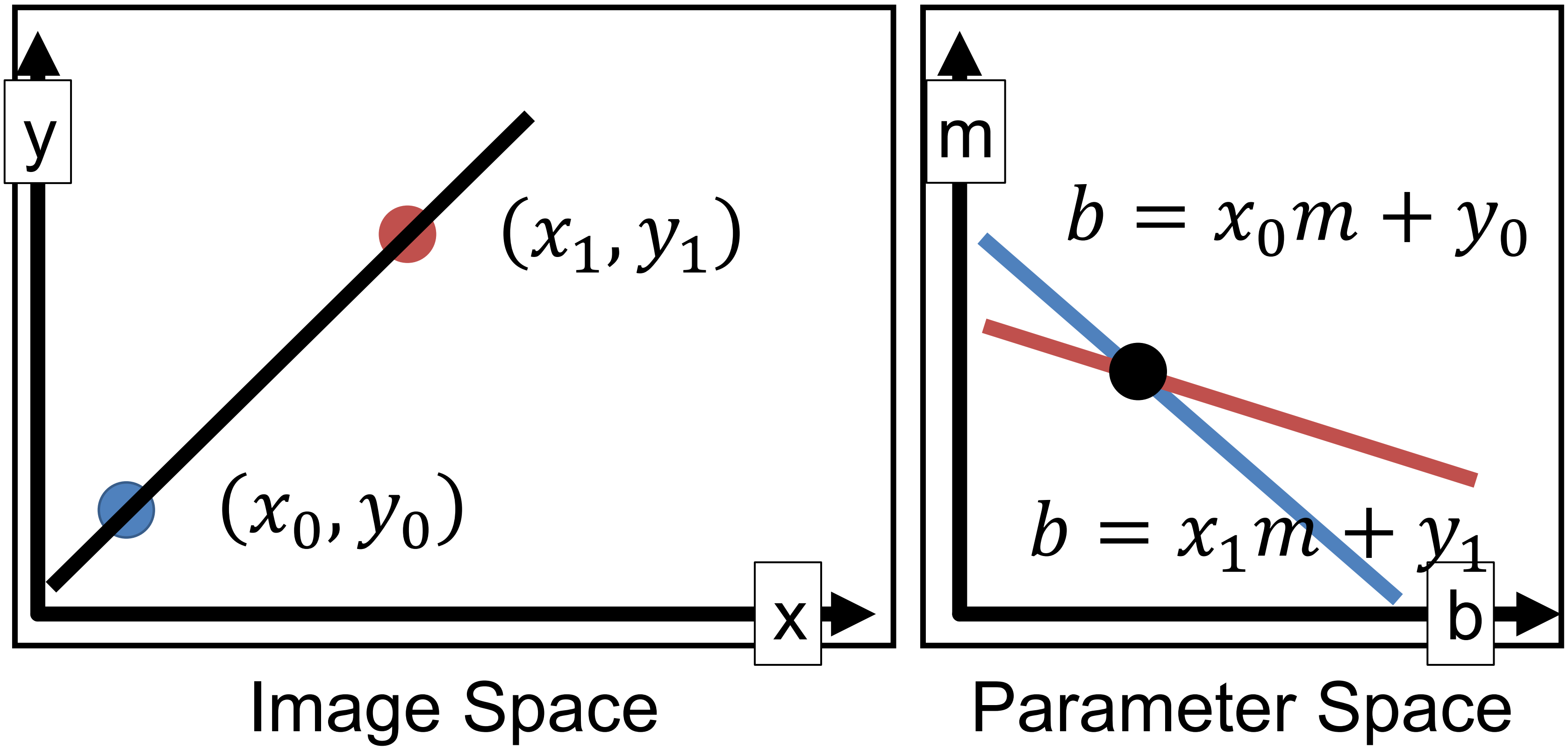
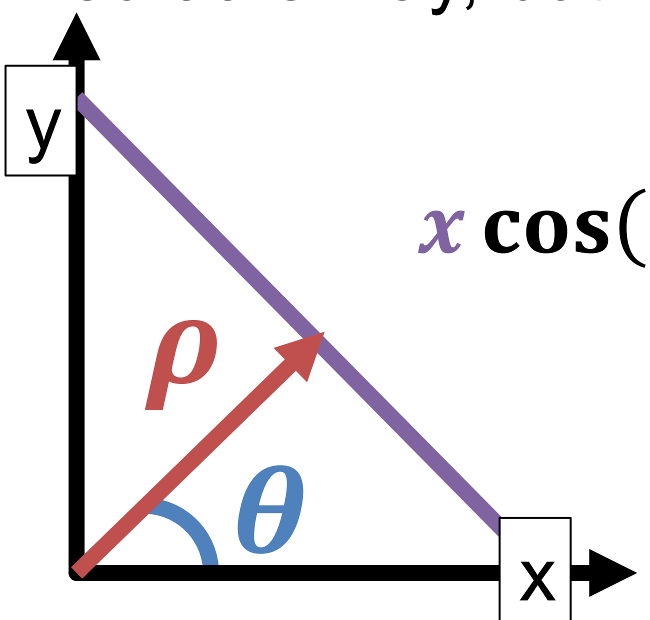


Diagram is remake of S. Seitz Slides; these are illustrative and values may not be real

Hough Transform

- *Recall*: m, b space is awful
- ax+by+c=0 is better, but *unbounded*
- Trick: write lines using angle + offset (normally a mediocre way, but makes things bounded)



$$x\,\mathbf{cos}(\theta) + y\,\mathbf{sin}(\theta) = \rho$$

Diagram is remake of S. Seitz Slides; these are illustrative and values may not be real

Hough Transform Algorithm

Remember: $x\cos(\theta) + y\sin(\theta) = \rho$

```
Accumulator H = zeros(?,?)
For x,y in detected_points:
     For θ in range(0,180,?):
          ρ = x cos(θ) + y sin(θ)
          H[θ, ρ] += 1
#any local maxima (θ, ρ) of H is a line
#of the form ρ = x cos(θ) + y sin(θ)
```

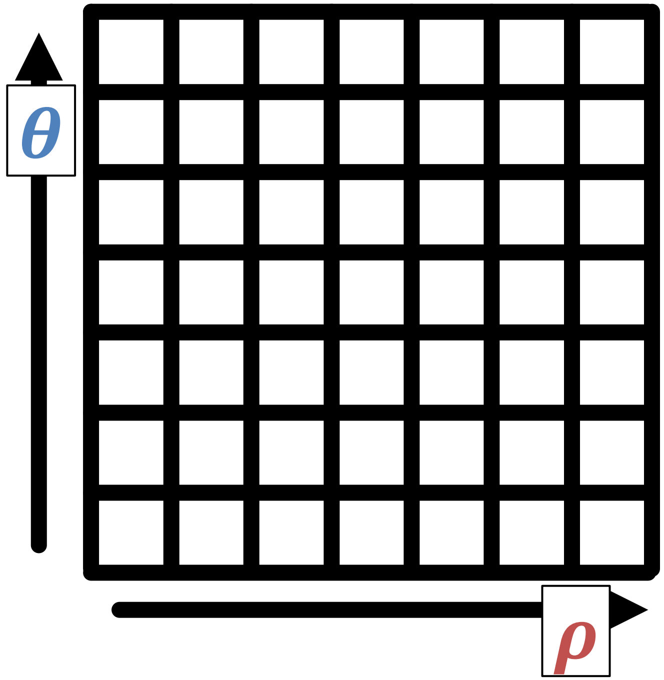


Diagram is remake of S. Seitz slides

Example

Points (x,y) -> sinusoids

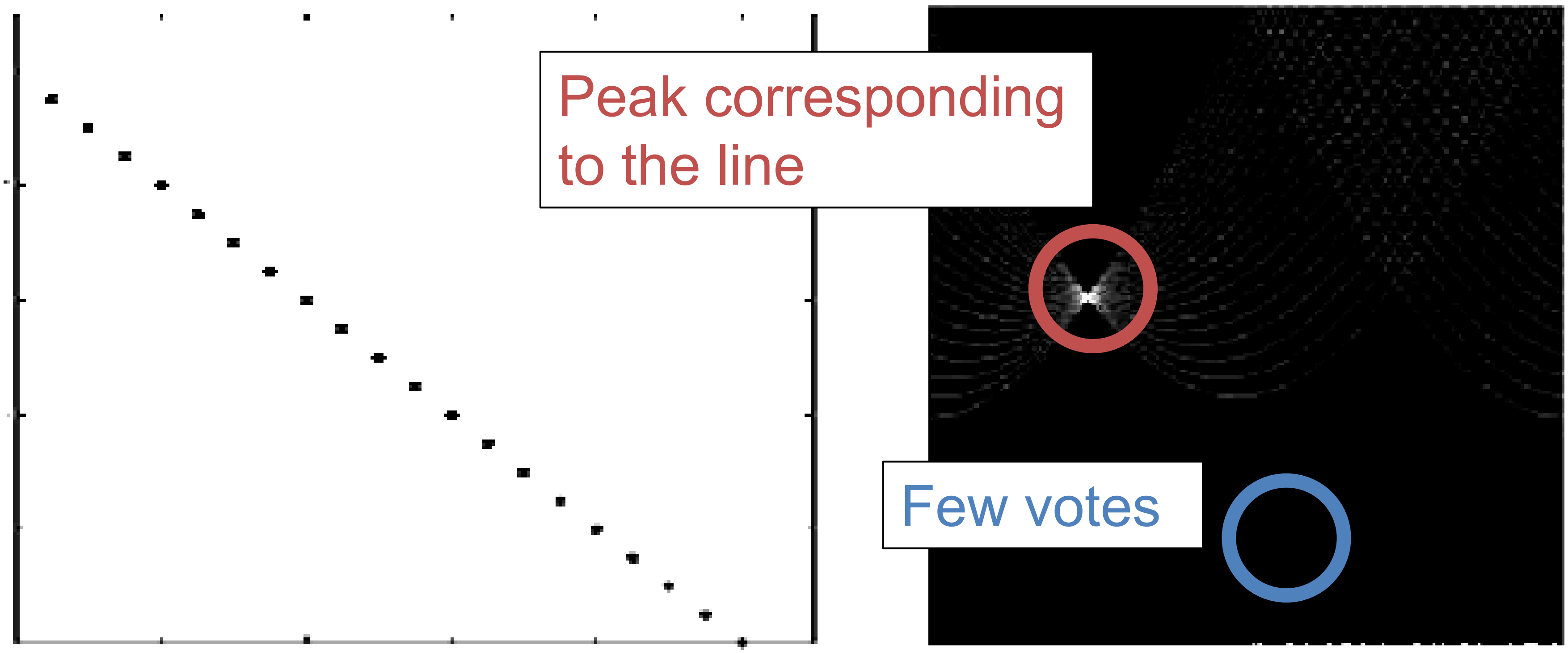


Image Space

Parameter Space

Slide Credit: S. Lazebnik

Hough Transform Pros / Cons

Pros

1. Handles **multiple** models
2. Some robustness to noise
3. In principle, general

Cons

1. Have to bin ALL parameters: exponential in #params
2. Have to parameterize your space nicely
3. Details really, really important (a working version requires a lot more than what I showed you)

Slide Credit: S. Lazebnik

Next Time

- What happens with fitting more complex transformations?

Details for the Curious

Least Squares

Derivation for the Curious

$$\|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 = (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w})$$

$$= \boldsymbol{Y}^T\boldsymbol{Y} - \boldsymbol{2}\boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{Y} + (\boldsymbol{X}\boldsymbol{w})^T\boldsymbol{X}\boldsymbol{w}$$

$$\frac{\partial}{\partial \boldsymbol{w}} (\boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{X}\boldsymbol{w}) = 2\left(\frac{\partial}{\partial \boldsymbol{w}}\boldsymbol{X}\boldsymbol{w}^T\right)\mathbf{Xw} = \mathbf{2X^TXw}$$

$$\frac{\partial}{\partial \boldsymbol{w}} \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 = 0 - 2\boldsymbol{X}^T\boldsymbol{Y} + 2\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w}$$

$$= 2\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{X}^T\boldsymbol{Y}$$

Total Least Squares

- In the interest of less material better, I’m giving that d = $\boldsymbol{\mu n}$.
- This can be derived by solving for d at the optimum in terms of the other variables.

Solving Total Least-Squares

$$\|\boldsymbol{Xn} - \mathbf{1}d\|_2^2 = (\boldsymbol{Xn} - \mathbf{1}d)^T(\boldsymbol{Xn} - \mathbf{1}d)$$

$$= (\boldsymbol{Xn})^{\boldsymbol{T}}(\boldsymbol{Xn}) - 2d\mathbf{1}^{\boldsymbol{T}}\boldsymbol{Xn} + d^{\mathbf{2}}\mathbf{1}^{\boldsymbol{T}}\mathbf{1}$$

First solve for d at optimum (set to 0)

$$\frac{\partial}{\partial d}\|\boldsymbol{Xn} - \mathbf{1}d\|_2^2 = 0 - 2\mathbf{1}^{\boldsymbol{T}}\boldsymbol{Xn} + 2dk$$

$$0 = -2\mathbf{1}^{\boldsymbol{T}}\boldsymbol{Xn} + 2dk \Longrightarrow 0 = -\mathbf{1}^{\boldsymbol{T}}\boldsymbol{Xn} + dk$$

$$\Longrightarrow d = \frac{1}{k}\mathbf{1}^{\boldsymbol{T}}\boldsymbol{Xn} = \boldsymbol{\mu n}$$

Common Fixes

Replace Least-Squares objective

Let $\boldsymbol{E} = \boldsymbol{Y} - \boldsymbol{X}\boldsymbol{W}$

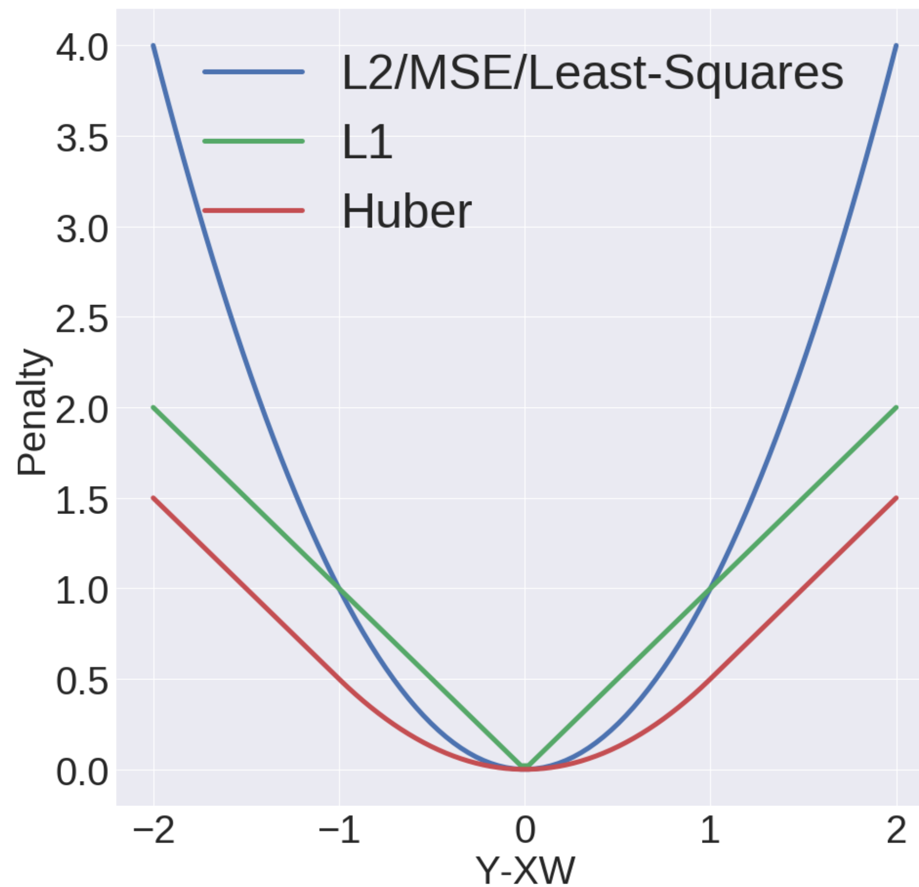
LS/L2/MSE: $\boldsymbol{E}_i^2$

L1: $|\boldsymbol{E}_i|$

Huber:

$|\boldsymbol{E}_i| \le \delta$: $\frac{1}{2}\boldsymbol{E}_i^2$

$|\boldsymbol{E}_i| > \delta$: $\delta\left(|\boldsymbol{E}_i| - \frac{\delta}{2}\right)$



Issues with Common Fixes

- Usually complicated to optimize:
  - Often no closed form solution
  - Typically not something you could write yourself
  - Sometimes not convex (local optimum is not necessarily a global optimum)
- Not simple to extend more complex objectives to things like total-least squares
- Typically don't handle a ton of outliers (e.g., 80% outliers)