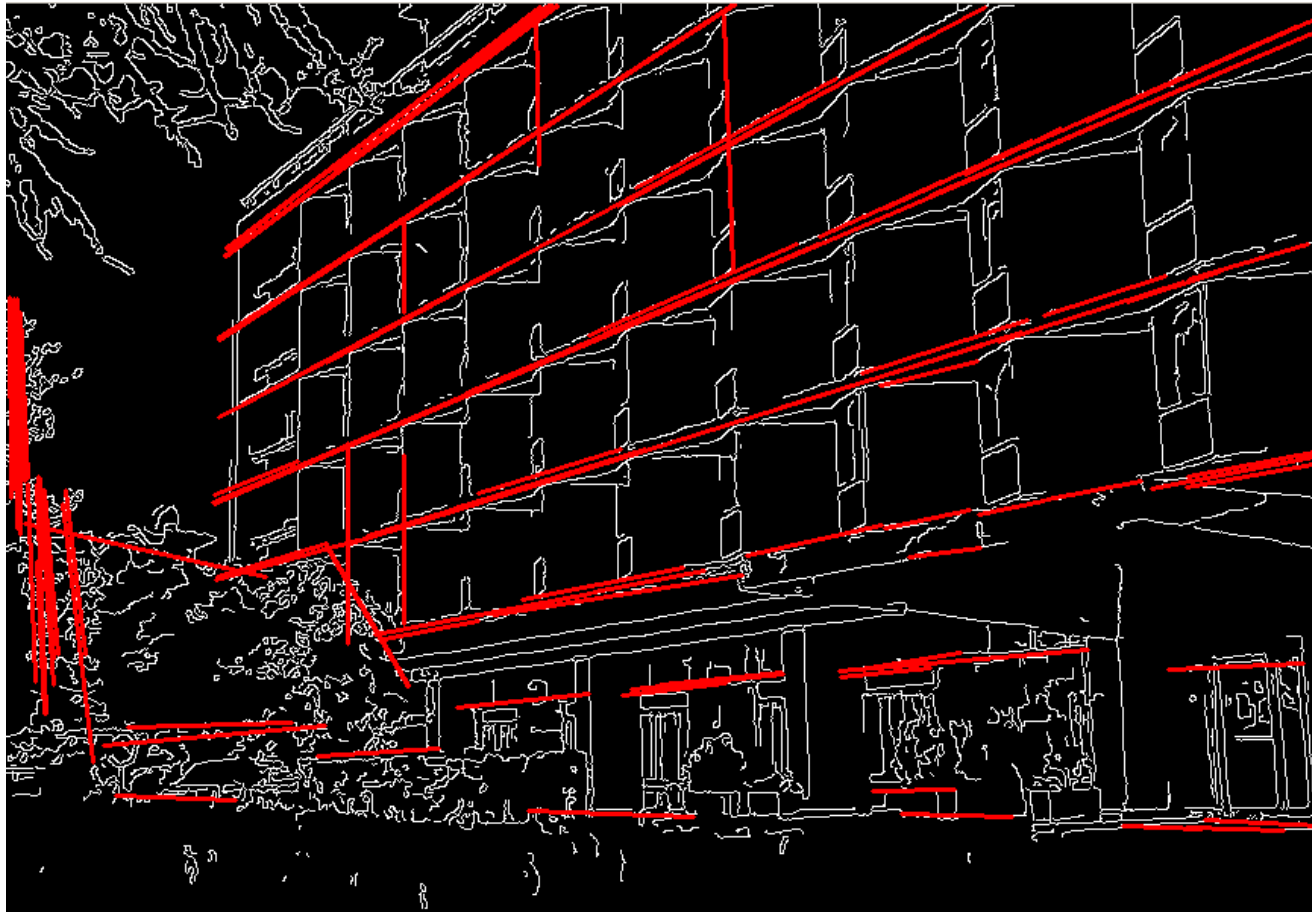


Fitting



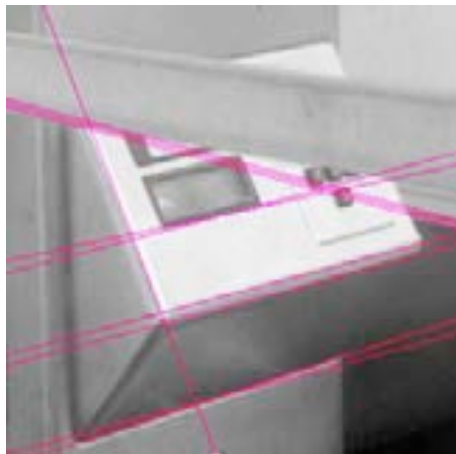
Fitting

- We've learned how to detect edges, corners, blobs. Now what?
- We would like to form a higher-level, more compact representation of the features in the image by grouping multiple features according to a simple model



Fitting

- Choose a *parametric model* to represent a set of features



simple model: lines



simple model: circles



complicated model: car

Fitting: Issues

Case study: Line detection



- **Noise** in the measured feature locations
- **Extraneous data:** clutter (outliers), multiple lines
- **Missing data:** occlusions

Fitting: Overview

- If we know which points belong to the line, how do we find the “optimal” line parameters?
 - Least squares
- What if there are outliers?
 - Robust fitting, RANSAC
- What if there are many lines?
 - Voting methods: RANSAC, Hough transform
- What if we're not even sure it's a line?
 - Model selection (not covered)

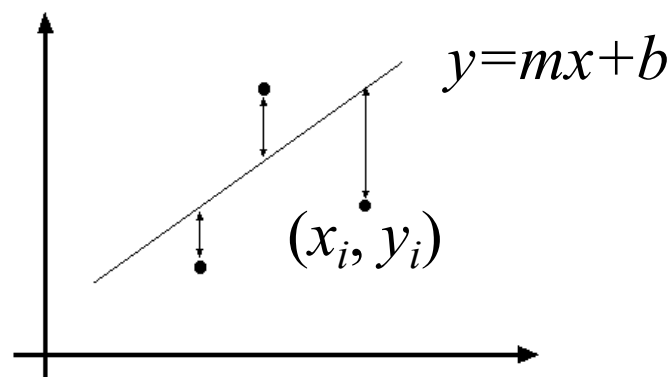
Least squares line fitting

Data: $(x_1, y_1), \dots, (x_n, y_n)$

Line equation: $y_i = m x_i + b$

Find (m, b) to minimize

$$E = \sum_{i=1}^n (y_i - m x_i - b)^2$$



$$E = \|Y - XB\|^2 \quad \text{where} \quad Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \quad X = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \quad B = \begin{bmatrix} m \\ b \end{bmatrix}$$

$$E = \|Y - XB\|^2 = (Y - XB)^T (Y - XB) = Y^T Y - 2(XB)^T Y + (XB)^T (XB)$$

$$\frac{dE}{dB} = 2X^T XB - 2X^T Y = 0$$

<http://faculty.bicmr.pku.edu.cn/~wenzw/bigdata/matrix-cook-book.pdf> sec 2.4

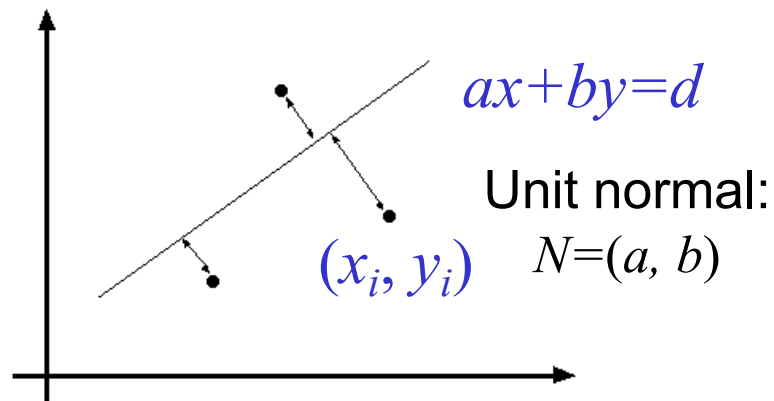
$$X^T XB = X^T Y \quad \text{Normal equations: least squares solution to } XB=Y$$

Problem with “vertical” least squares

- Not rotation-invariant
- Fails completely for vertical lines

Total least squares

Distance between point (x_i, y_i) and line $ax+by=d$ ($a^2+b^2=1$): $|ax_i + by_i - d|$

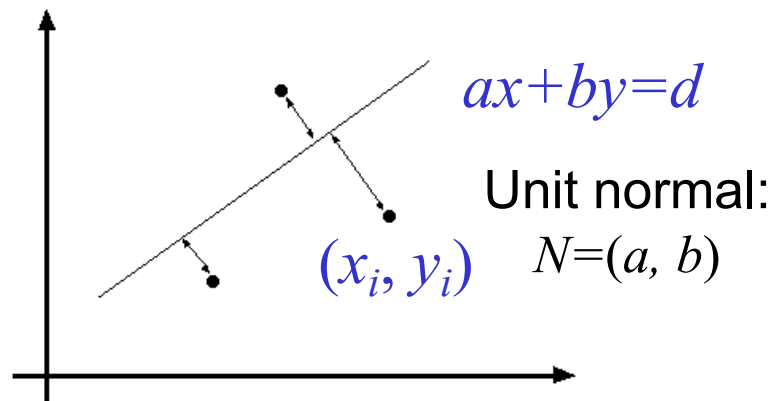


Total least squares

Distance between point (x_i, y_i) and line $ax+by=d$ ($a^2+b^2=1$): $|ax_i + by_i - d|$

Find (a, b, d) to minimize the sum of squared perpendicular distances

$$E = \sum_{i=1}^n (ax_i + by_i - d)^2$$



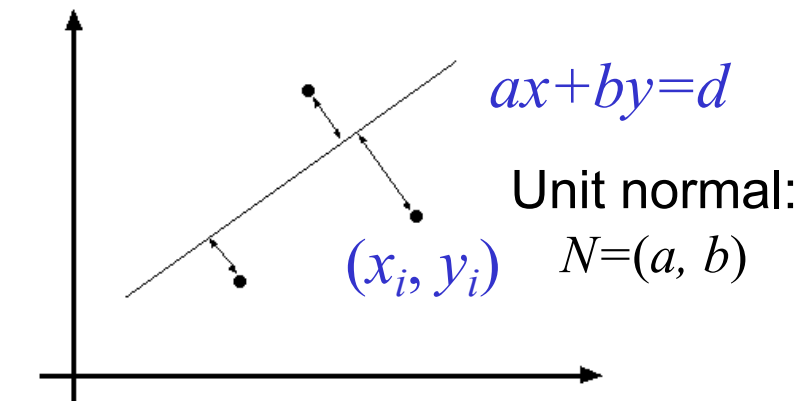
Total least squares

Distance between point (x_i, y_i) and line $ax+by=d$ ($a^2+b^2=1$): $|ax_i + by_i - d|$

Find (a, b, d) to minimize the sum of squared perpendicular distances

$$E = \sum_{i=1}^n (ax_i + by_i - d)^2$$

$$\frac{\partial E}{\partial d} = \sum_{i=1}^n -2(ax_i + by_i - d) = 0$$



$$d = \frac{a}{n} \sum_{i=1}^n x_i + \frac{b}{n} \sum_{i=1}^n y_i = a\bar{x} + b\bar{y}$$

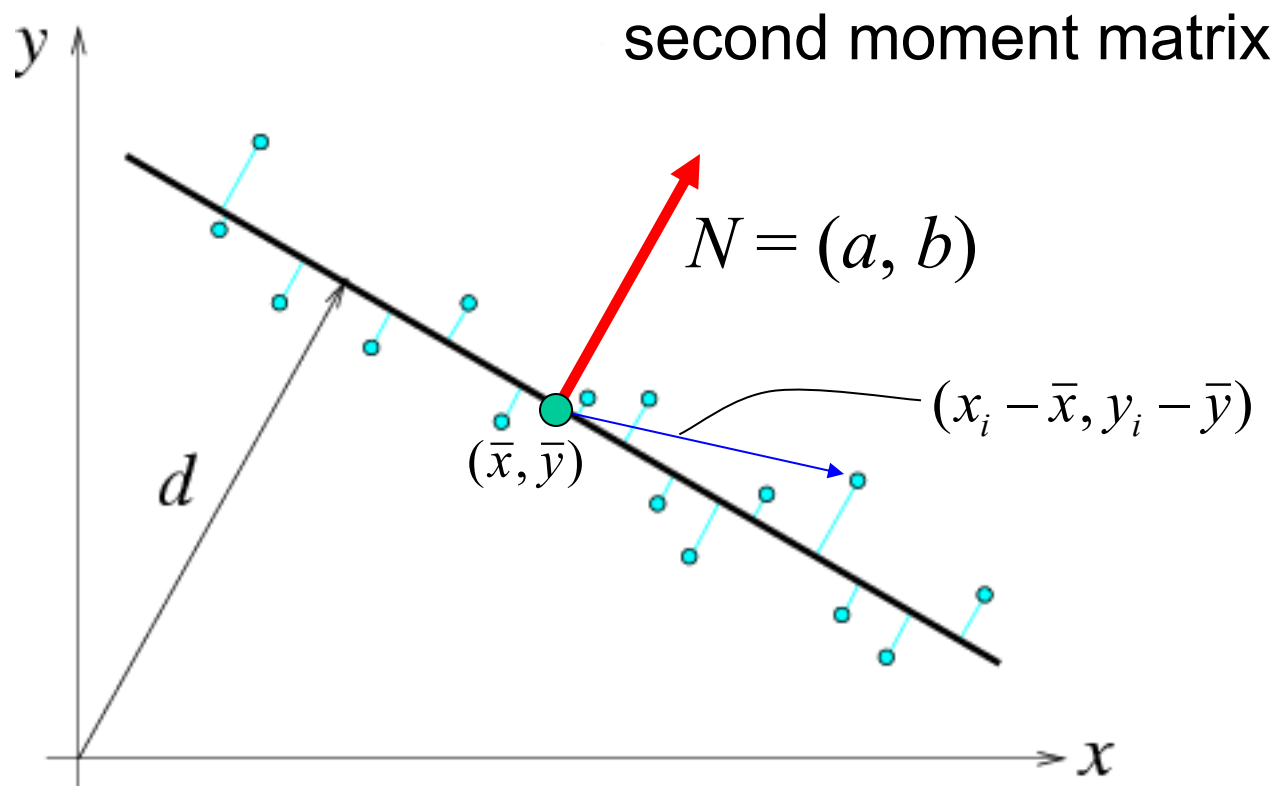
$$E = \sum_{i=1}^n (a(x_i - \bar{x}) + b(y_i - \bar{y}))^2 = \left\| \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \right\|^2 = (UN)^T (UN)$$

$$\frac{dE}{dN} = 2(U^T U)N = 0$$

Solution to $(U^T U)N = 0$, subject to $\|N\|^2 = 1$: eigenvector of $U^T U$ associated with the smallest eigenvalue (least squares solution to homogeneous linear system $UN = 0$)

Total least squares

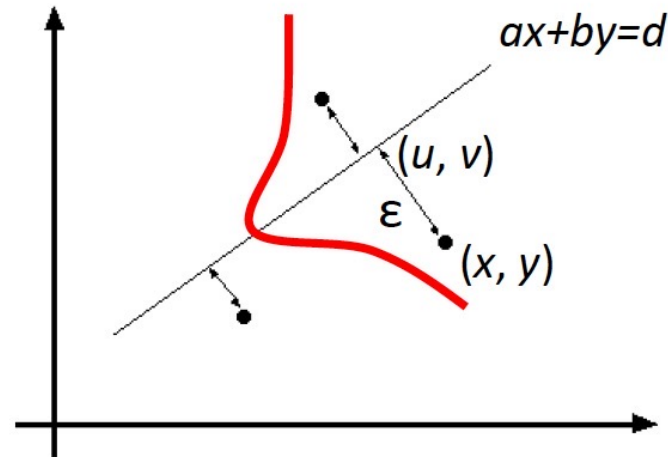
$$U = \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \quad U^T U = \begin{bmatrix} \sum_{i=1}^n (x_i - \bar{x})^2 & \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y}) \\ \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y}) & \sum_{i=1}^n (y_i - \bar{y})^2 \end{bmatrix}$$



Least squares as likelihood maximization

- **Generative model:** line points are sampled independently and corrupted by Gaussian noise in the direction perpendicular to the line

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix} + \varepsilon \begin{pmatrix} a \\ b \end{pmatrix}$$



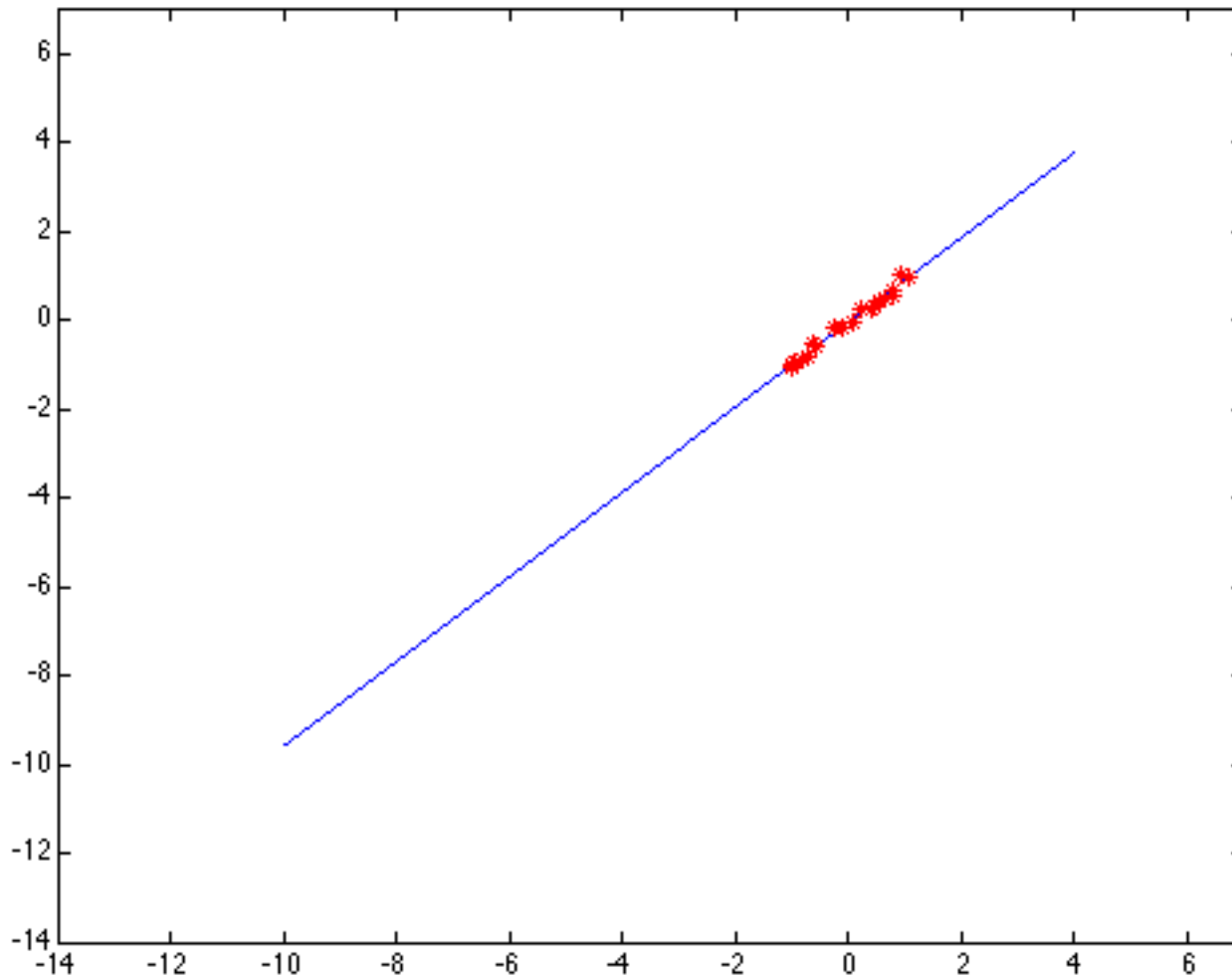
Likelihood of points given line parameters (a, b, d) :

$$P(x_1, y_1, \dots, x_n, y_n \mid a, b, d) = \prod_{i=1}^n P(x_i, y_i \mid a, b, d) \propto \prod_{i=1}^n \exp\left(-\frac{(ax_i + by_i - d)^2}{2\sigma^2}\right)$$

Log-likelihood:
$$L(x_1, y_1, \dots, x_n, y_n \mid a, b, d) = -\frac{1}{2\sigma^2} \sum_{i=1}^n (ax_i + by_i - d)^2$$

Source: S. Lazebnik

Least squares fit to the red points:



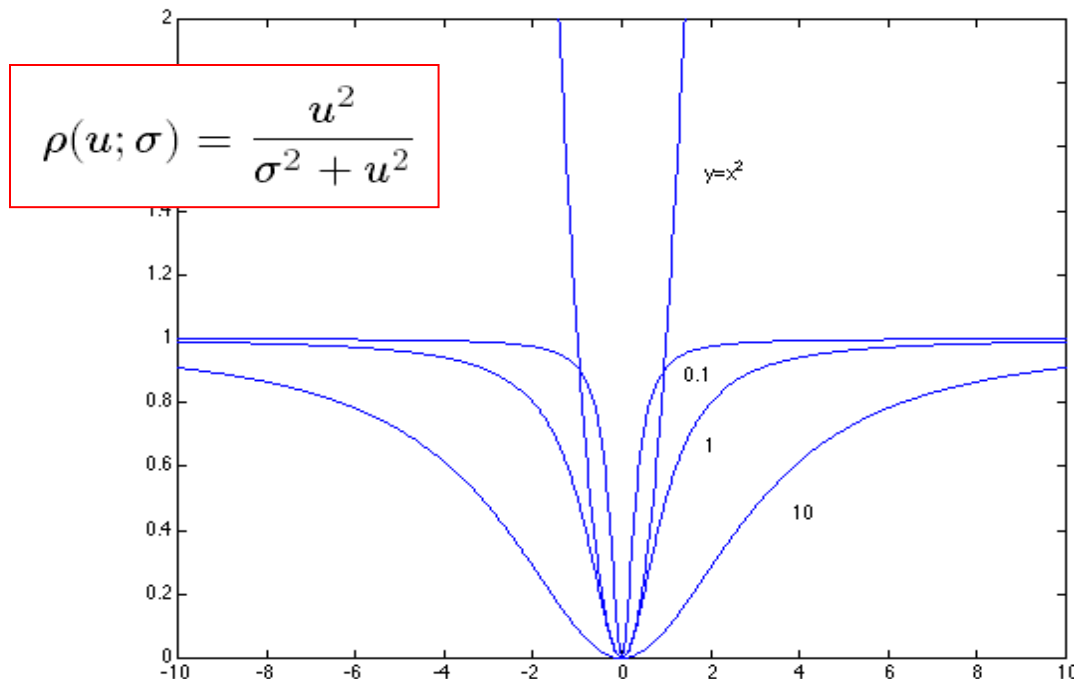
Robust estimators

- General approach: find model parameters θ that minimize

$$\sum_i \rho(r_i(x_i, \theta); \sigma)$$

$r_i(x_i, \theta)$ – residual of i th point w.r.t. model parameters θ

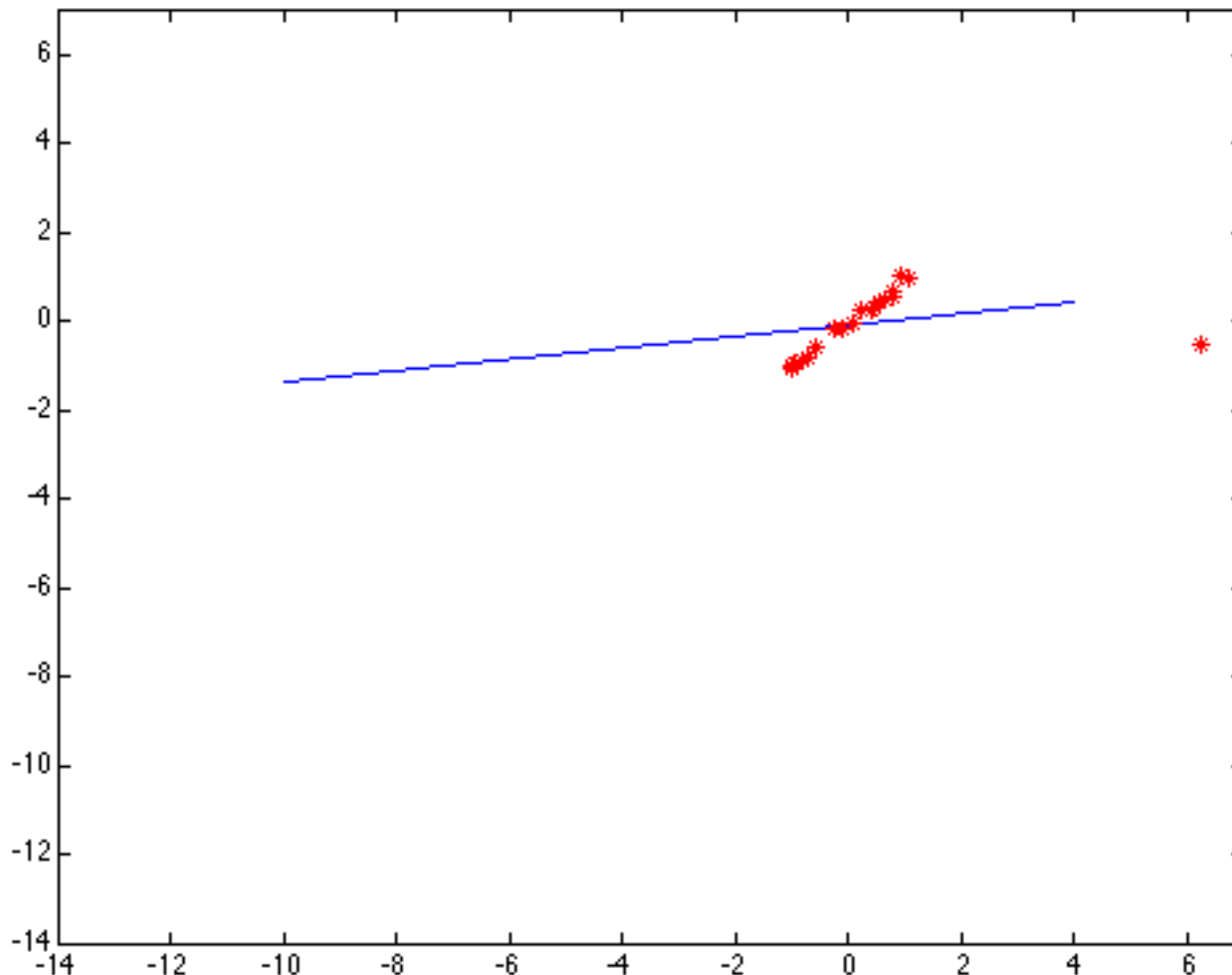
ρ – robust function with scale parameter σ



The robust function ρ behaves like squared distance for small values of the residual u but saturates for larger values of u

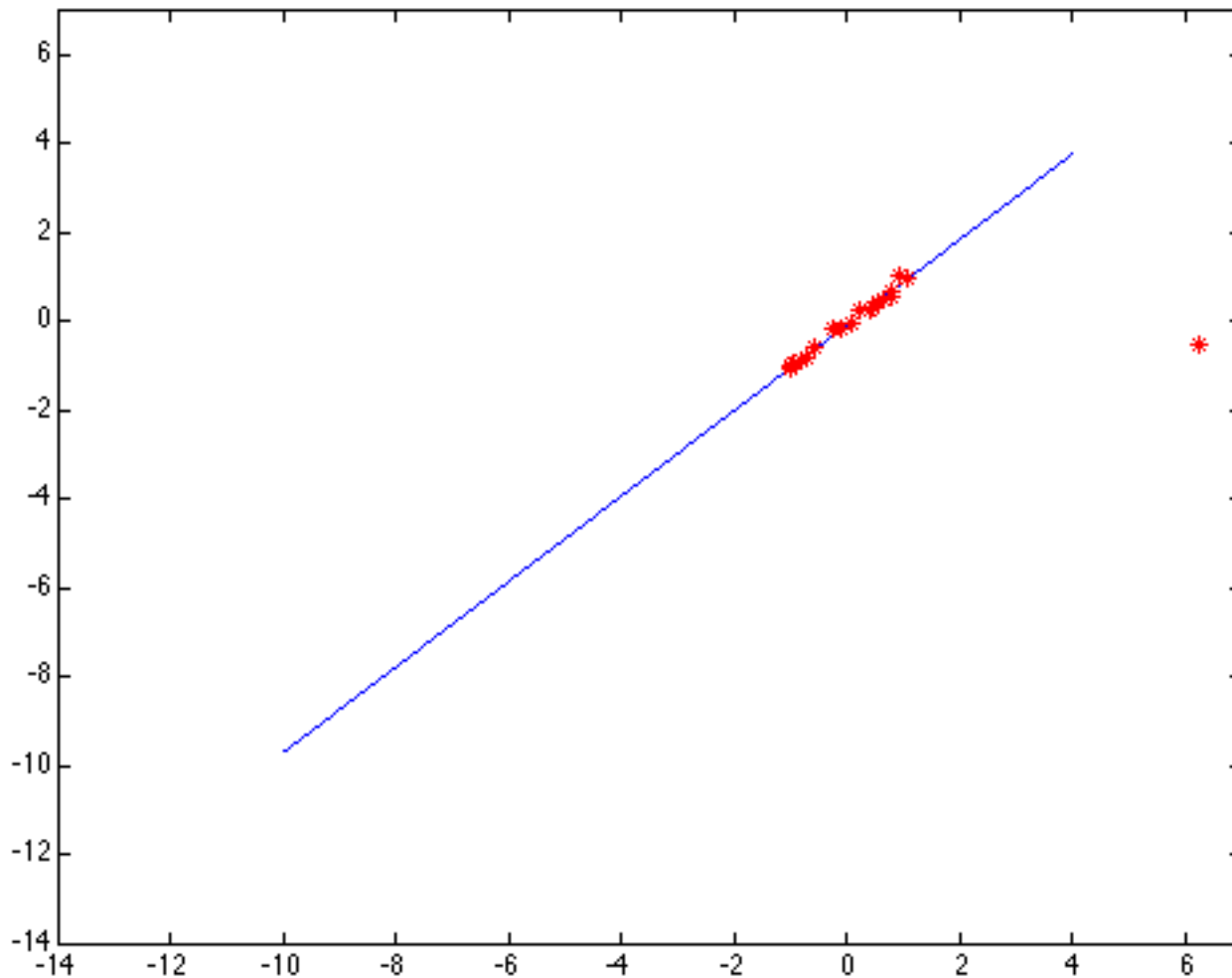
Least squares: Robustness to noise

Least squares fit with an outlier:



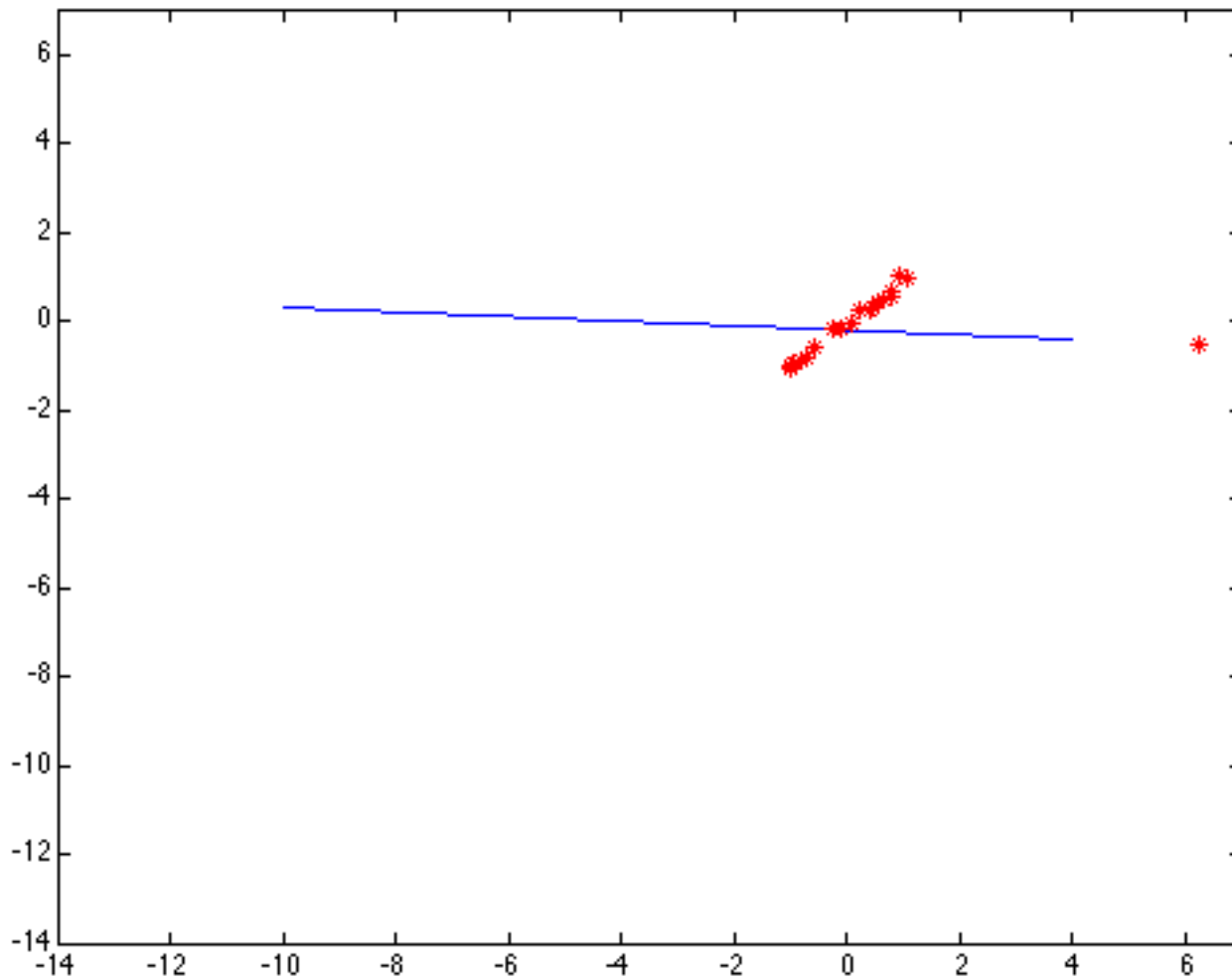
Problem: squared error heavily penalizes outliers

Choosing the scale: Just right



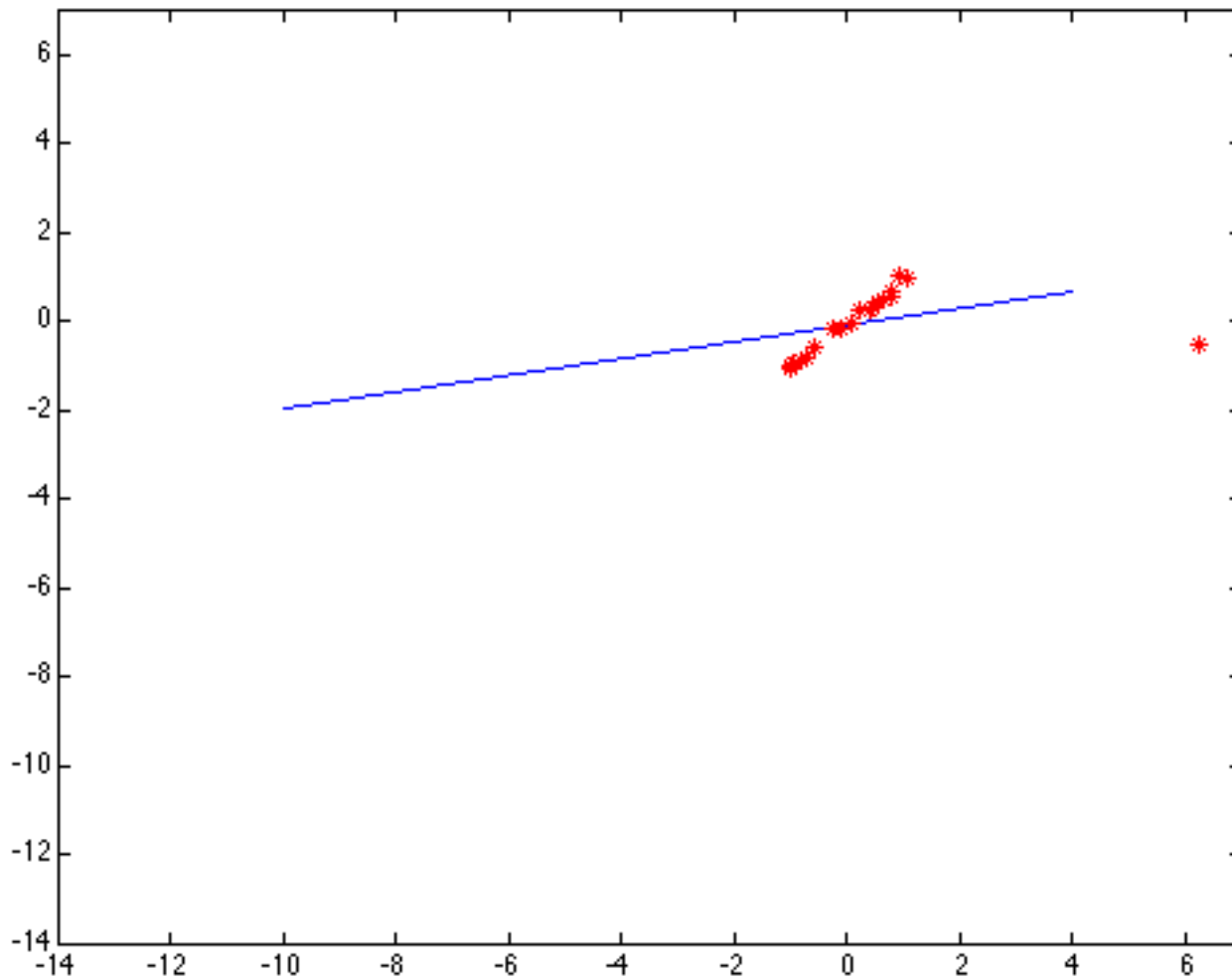
The effect of the outlier is minimized

Choosing the scale: Too small



The error value is almost the same for every point and the fit is very poor

Choosing the scale: Too large



Behaves much the same as least squares

Robust estimation: Details

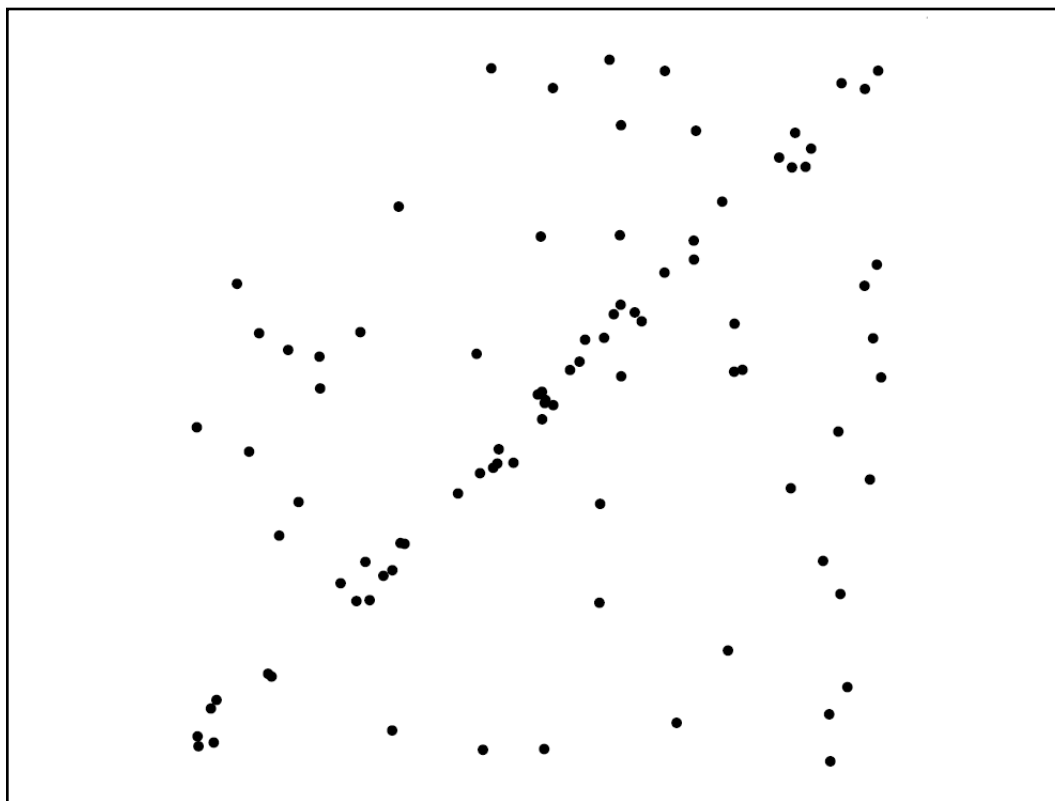
- Robust fitting is a nonlinear optimization problem that must be solved iteratively
- Least squares solution can be used for initialization
- Scale of robust function should be chosen adaptively based on median residual

RANSAC

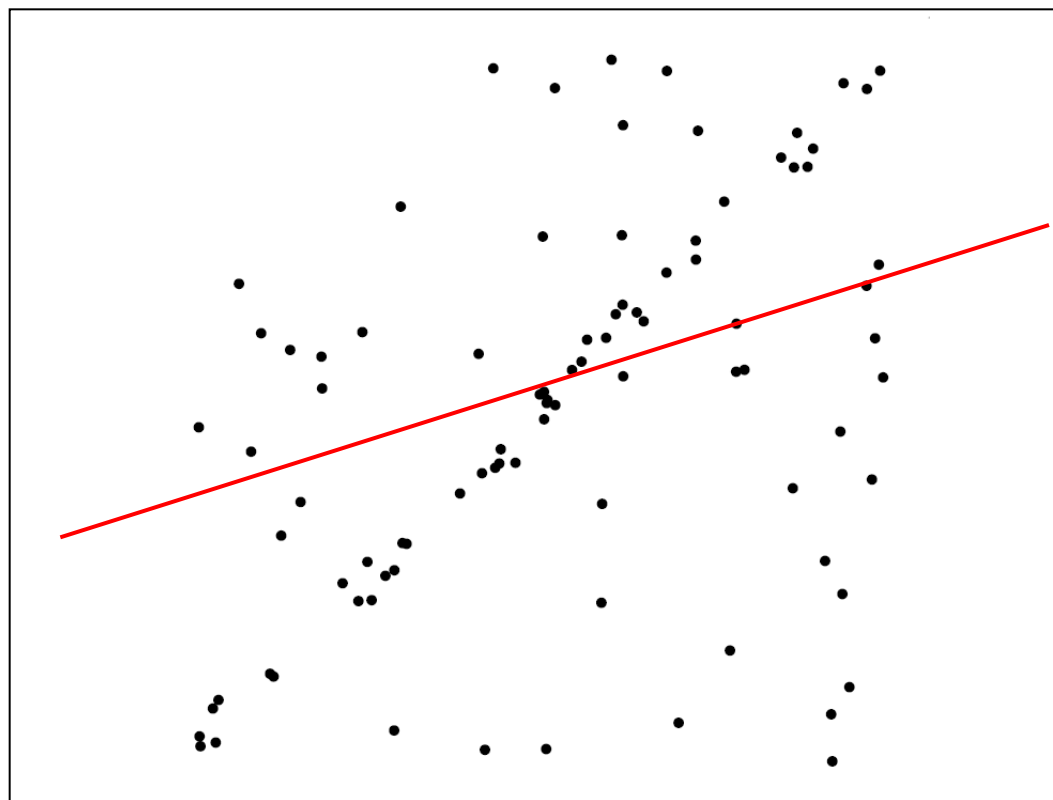
- Robust fitting can deal with a few outliers – what if we have very many?
- Random sample consensus (RANSAC):
Very general framework for model fitting in the presence of outliers
- Outline
 - Choose a small subset of points uniformly at random
 - Fit a model to that subset
 - Find all remaining points that are “close” to the model and reject the rest as outliers
 - Do this many times and choose the best model

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 line fitting example

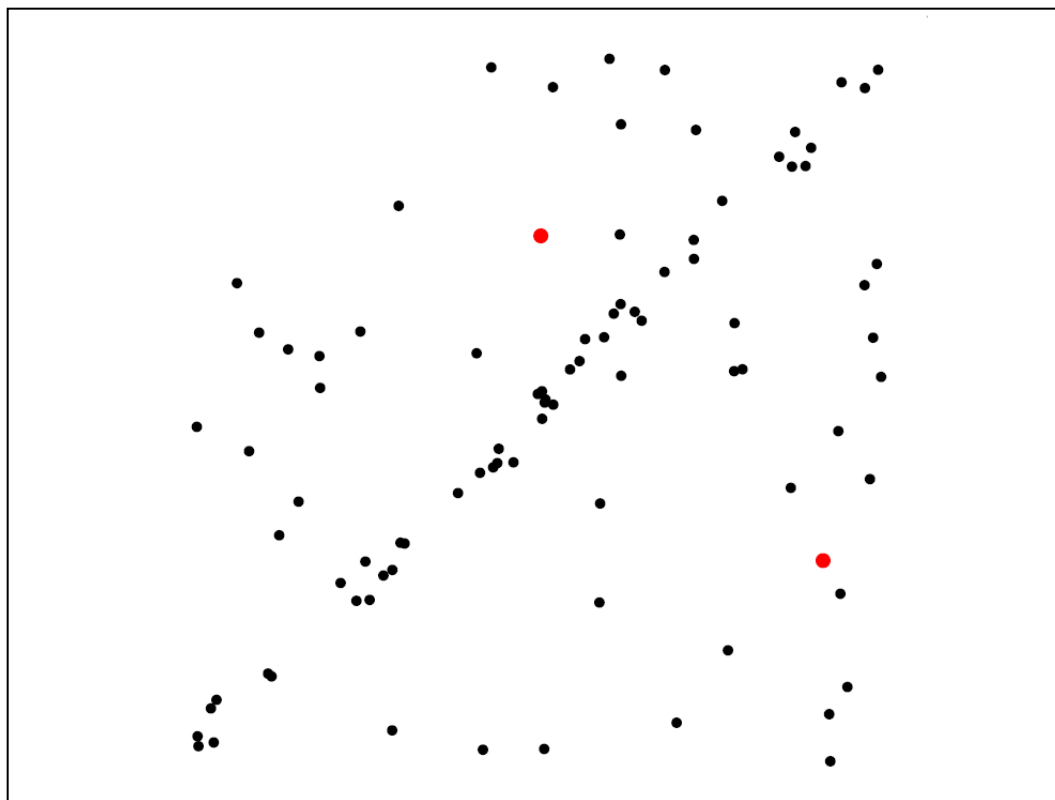


RANSAC for line fitting example



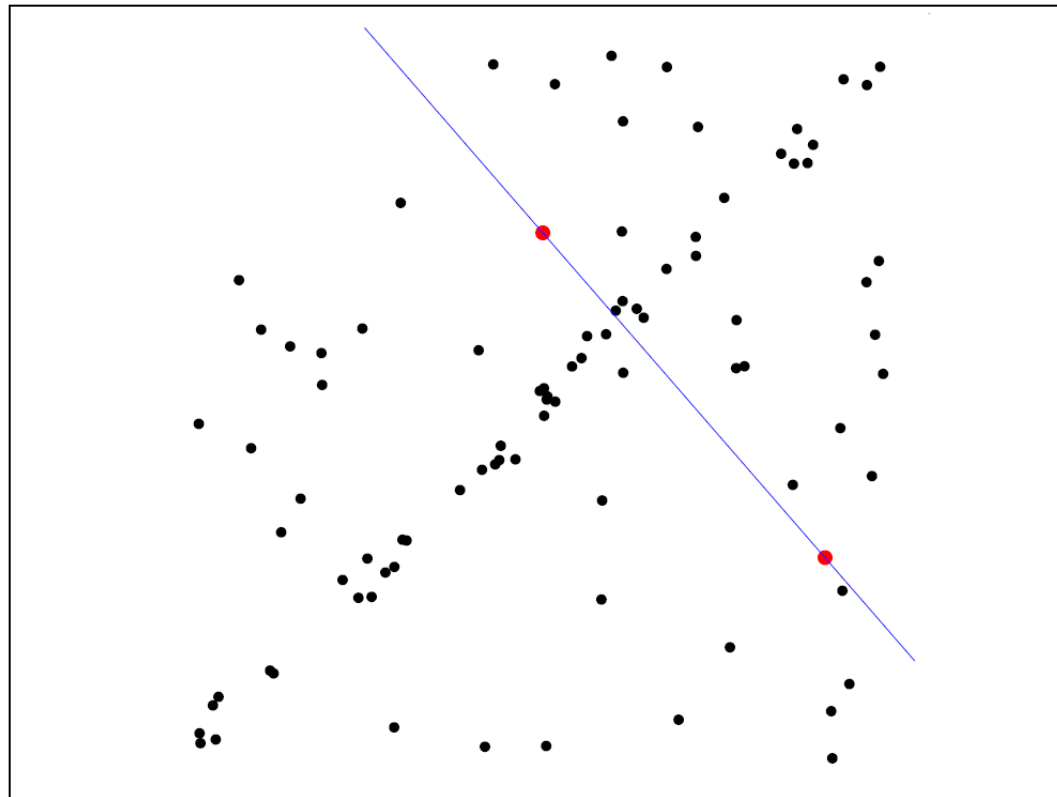
Least-squares fit

RANSAC for line fitting example



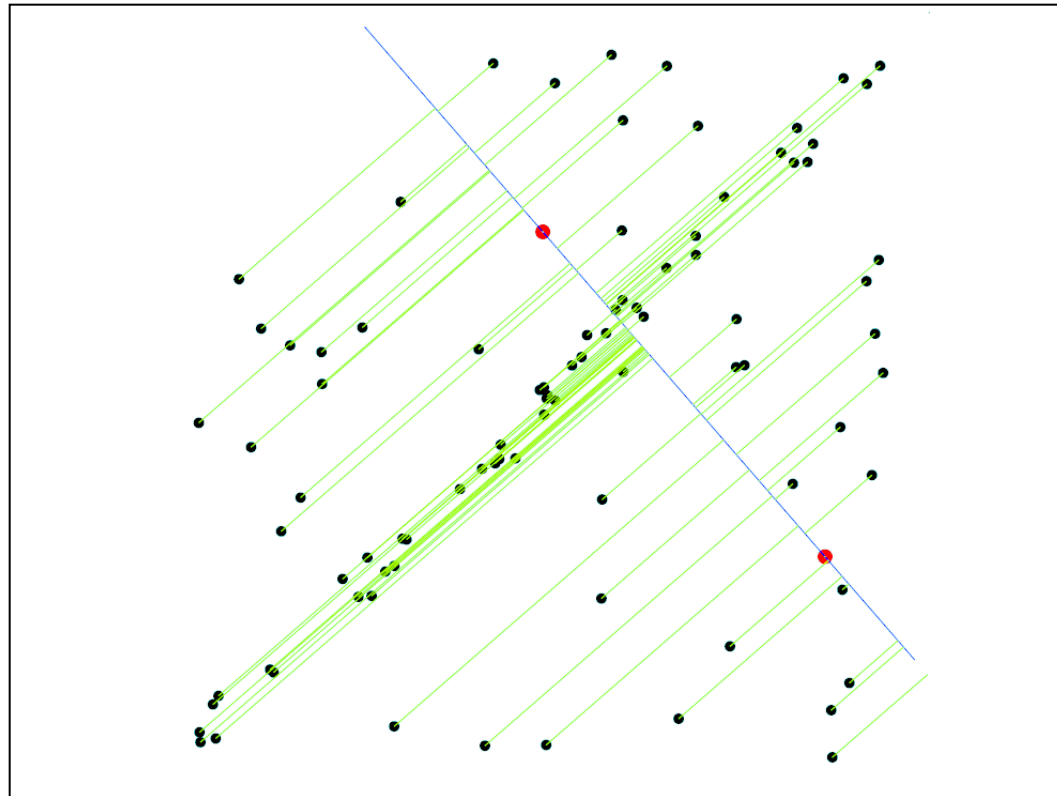
1. Randomly select minimal subset of points

RANSAC for line fitting example



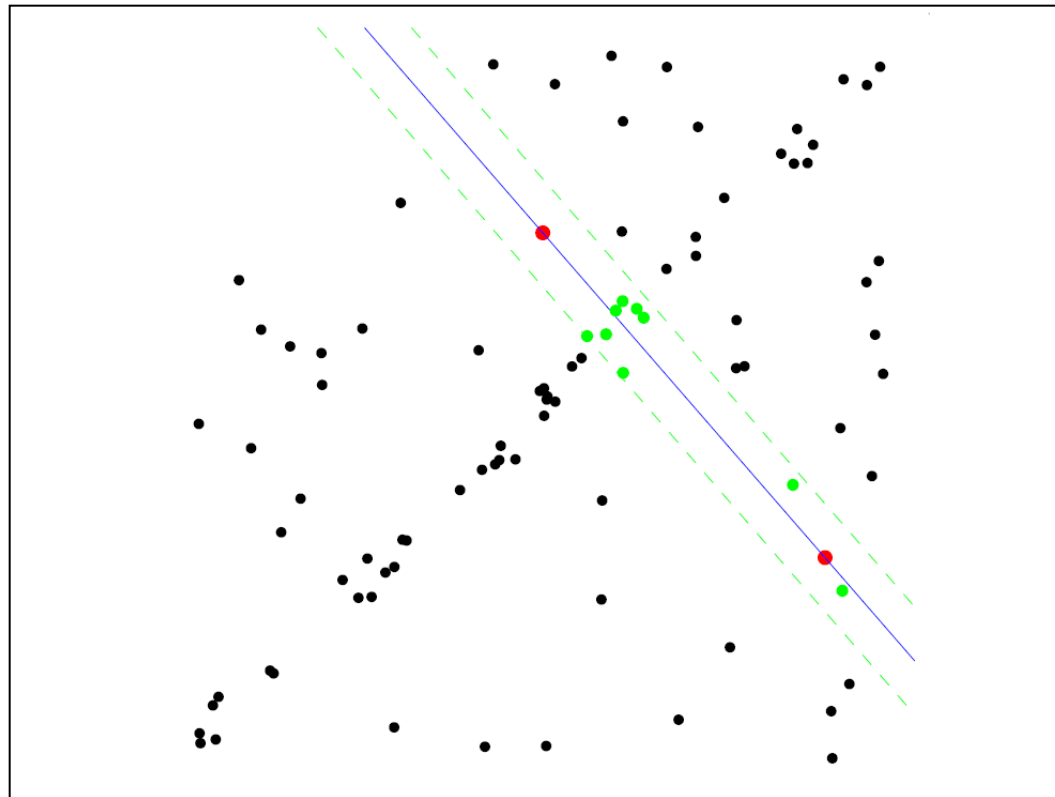
1. Randomly select minimal subset of points
2. Hypothesize a model

RANSAC for line fitting example



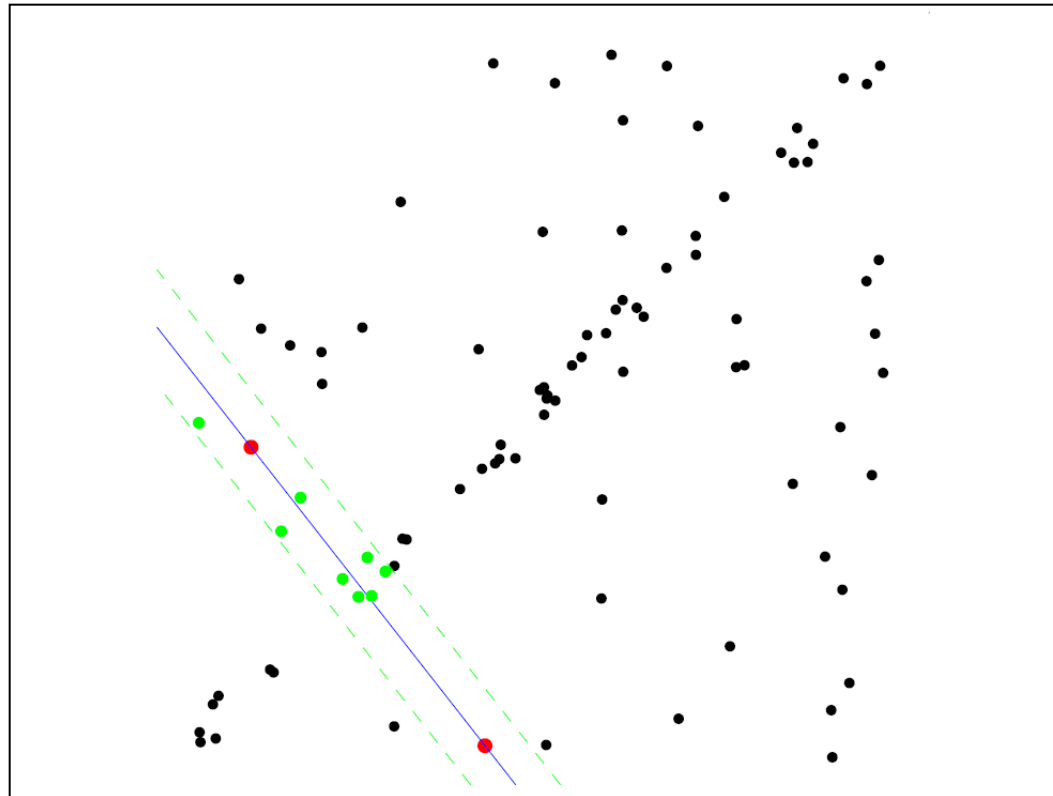
1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function

RANSAC for line fitting example



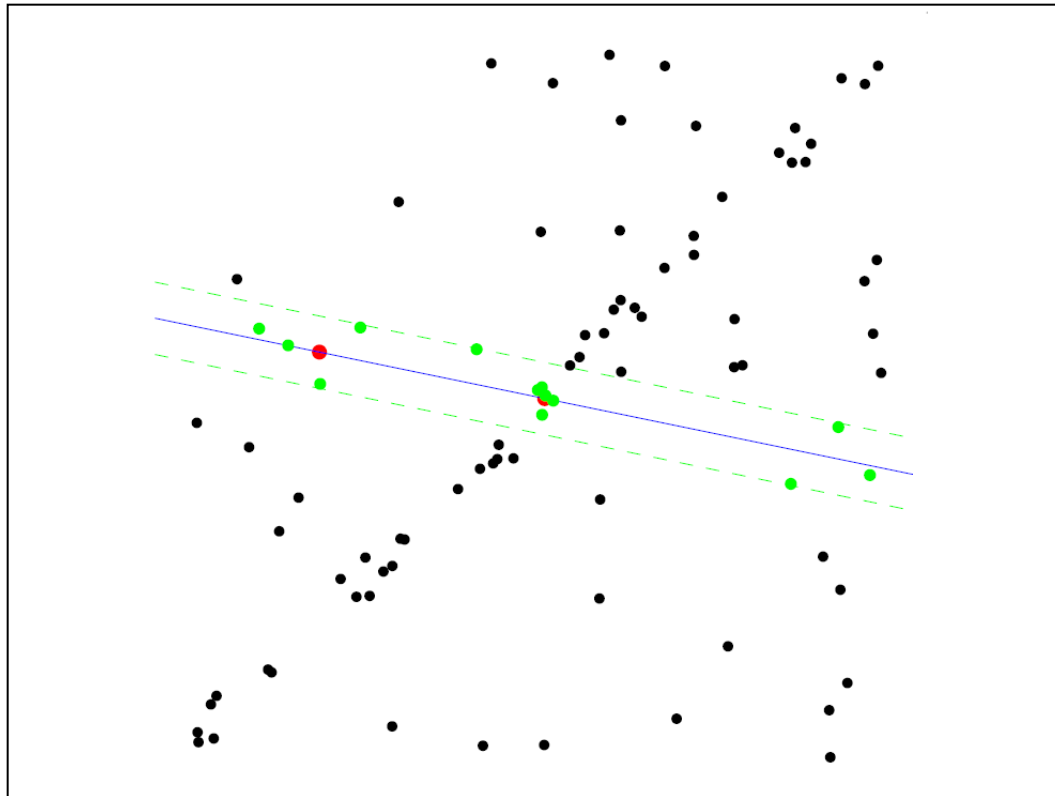
1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model

RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

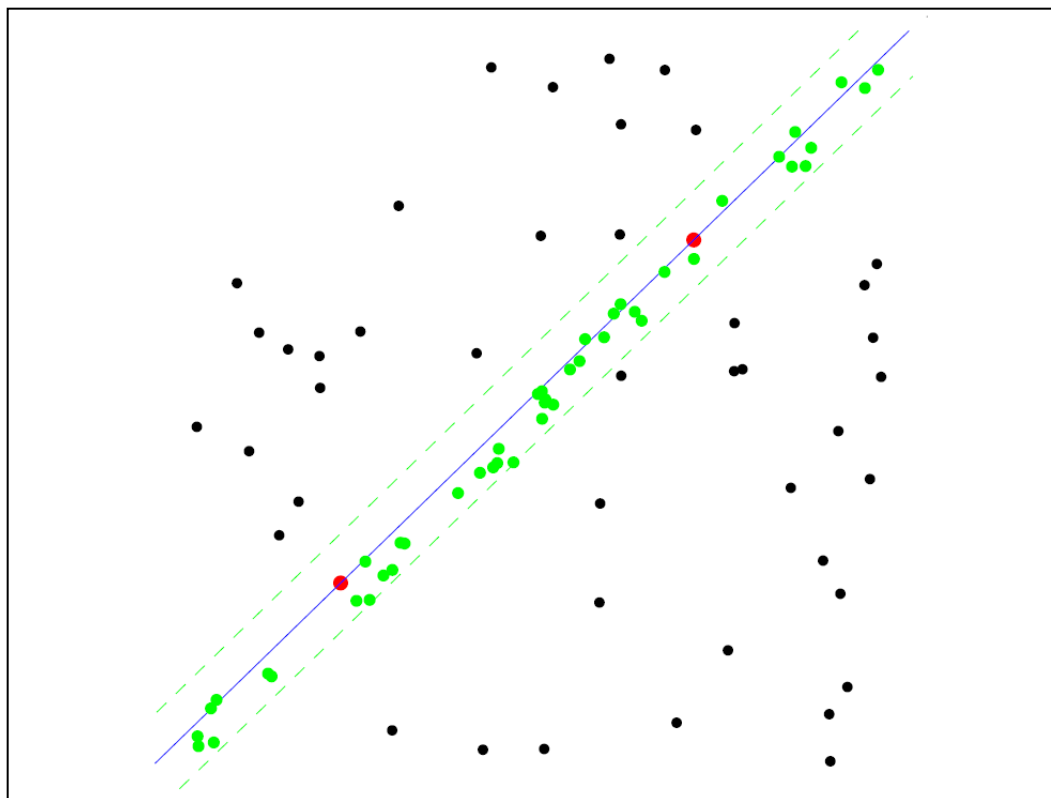
RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

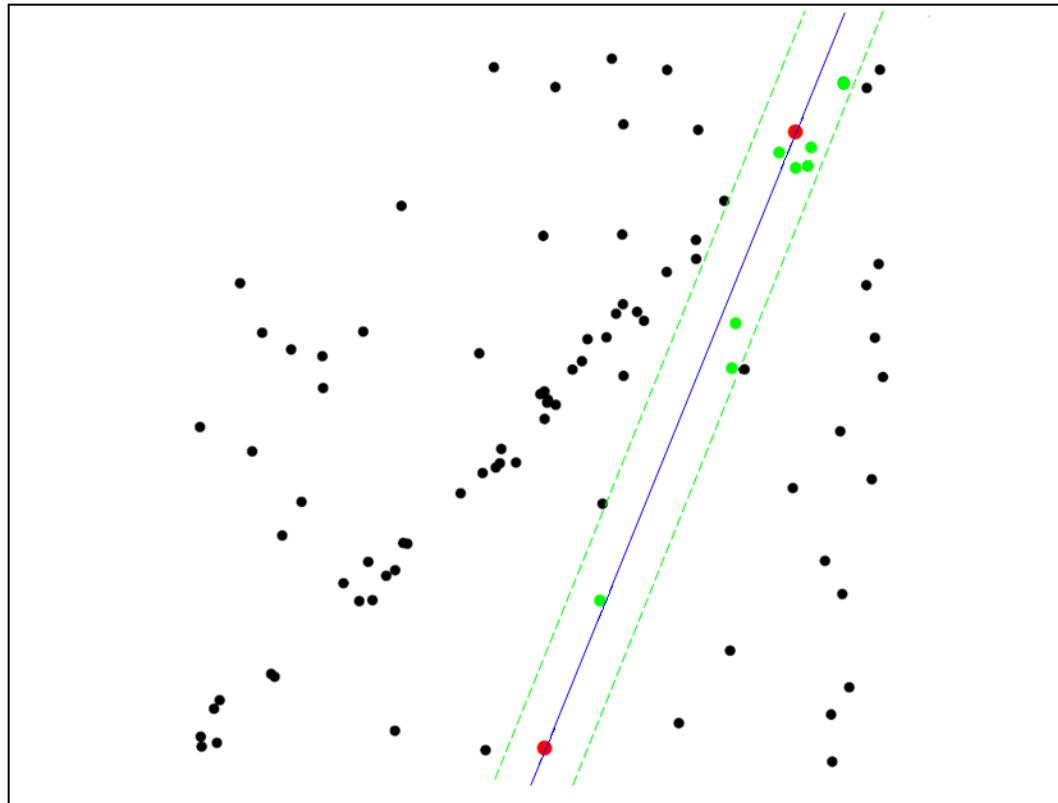
RANSAC for line fitting example

Uncontaminated sample



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

RANSAC for line fitting

Repeat N times:

- Draw s points uniformly at random
- Fit line to these s points
- Find *inliers* to this line among the remaining points (i.e., points whose distance from the line is less than t)
- If there are d or more inliers, accept the line and refit using all inliers

Choosing the parameters

- Initial number of points **s**
 - Typically minimum number needed to fit the model
- Distance threshold **t**
 - Choose **t** so probability for inlier is p (e.g. 0.95)
 - Zero-mean Gaussian noise with std. dev. σ : $t^2 = 3.84\sigma^2$
- Number of samples **N**
 - Choose **N** so that, with probability p , at least one random sample is free from outliers (e.g. $p=0.99$) (outlier ratio: e)

Choosing the parameters

- Initial number of points **s**
 - Typically minimum number needed to fit the model
- Distance threshold **t**
 - Choose **t** so probability for inlier is p (e.g. 0.95)
 - Zero-mean Gaussian noise with std. dev. σ : $t^2 = 3.84\sigma^2$
- Number of samples **N**
 - Choose **N** so that, with probability p , at least one random sample is free from outliers (e.g. $p=0.99$) (outlier ratio: e)

$$\left(1 - (1 - e)^s\right)^N = 1 - p$$

$$N = \log(1 - p) / \log(1 - (1 - e)^s)$$

s	proportion of outliers e						
	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

Choosing the parameters

- Initial number of points **s**
 - Typically minimum number needed to fit the model
- Distance threshold **t**
 - Choose **t** so probability for inlier is p (e.g. 0.95)
 - Zero-mean Gaussian noise with std. dev. σ : $t^2=3.84\sigma^2$
- Number of samples **N**
 - Choose **N** so that, with probability p , at least one random sample is free from outliers (e.g. $p=0.99$) (outlier ratio: e)

$$\left(1 - (1 - e)^s\right)^N = 1 - p$$

$$N = \log(1 - p) / \log\left(1 - (1 - e)^s\right)$$

Choosing the parameters

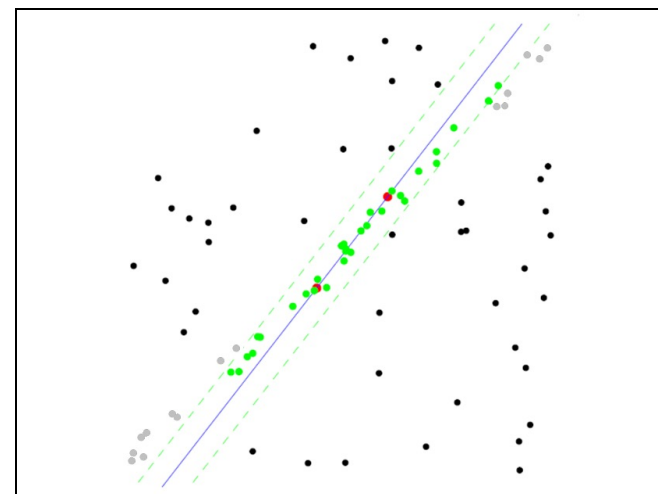
- Initial number of points **s**
 - Typically minimum number needed to fit the model
- Distance threshold **t**
 - Choose **t** so probability for inlier is p (e.g. 0.95)
 - Zero-mean Gaussian noise with std. dev. σ : $t^2 = 3.84\sigma^2$
- Number of samples **N**
 - Choose **N** so that, with probability p , at least one random sample is free from outliers (e.g. $p=0.99$) (outlier ratio: e)
- Consensus set size **d**
 - Should match expected inlier ratio

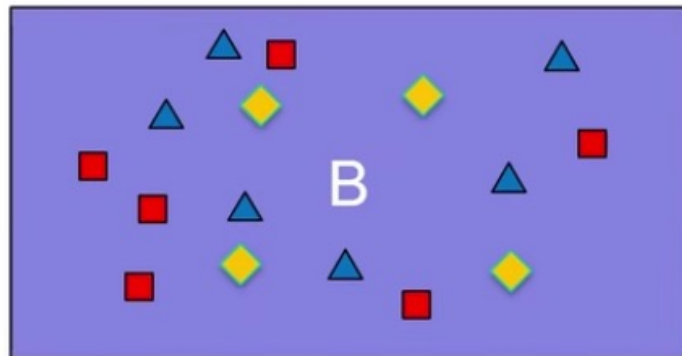
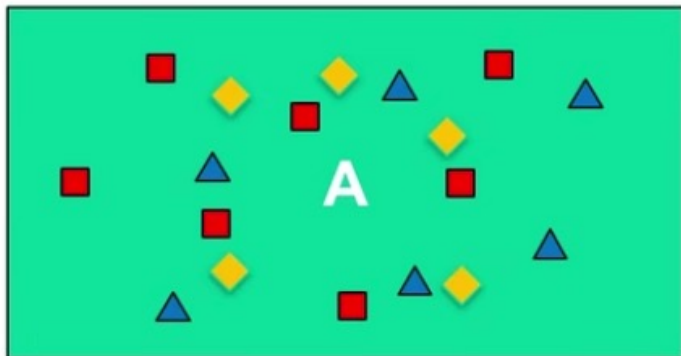
Adaptively determining the number of samples

- Outlier ratio e is often unknown a priori, so pick worst case, e.g. 50%, and adapt if more inliers are found, e.g. 80% would yield $e=0.2$
- Adaptive procedure:
 - $N=\infty$, $sample_count=0$
 - While $N > sample_count$
 - Choose a sample and count the number of inliers
 - If inlier ratio is highest of any found so far, set $e = 1 - (\text{number of inliers})/(\text{total number of points})$
 - Recompute N from e :
$$N = \log(1 - p) / \log(1 - (1 - e)^s)$$
 - Increment the $sample_count$ by 1

RANSAC pros and cons

- Pros
 - Simple and general
 - Applicable to many different problems
 - Often works well in practice
- Cons
 - Lots of parameters to tune
 - Doesn't work well for low inlier ratios (too many iterations, or can fail completely)
 - Can't always get a good initialization of the model based on the minimum number of samples





$$\begin{bmatrix} x_a \\ y_a \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x_b \\ y_b \\ 1 \end{bmatrix}$$

Fitting: Review

- ✓ If we know which points belong to the line, how do we find the “optimal” line parameters?
 - ✓ Least squares
- ✓ What if there are outliers?
 - ✓ Robust fitting, RANSAC
- What if there are many lines?
 - Voting methods: RANSAC, Hough transform