Outliers and Robust Estimation

Applied Machine Learning
Derek Hoiem

Dall-E

This class: Robust Estimation

- Robust statistics and quantiles
- Detecting outliers
- Robust fitting
  - Reweighted least squares
  - RANSAC

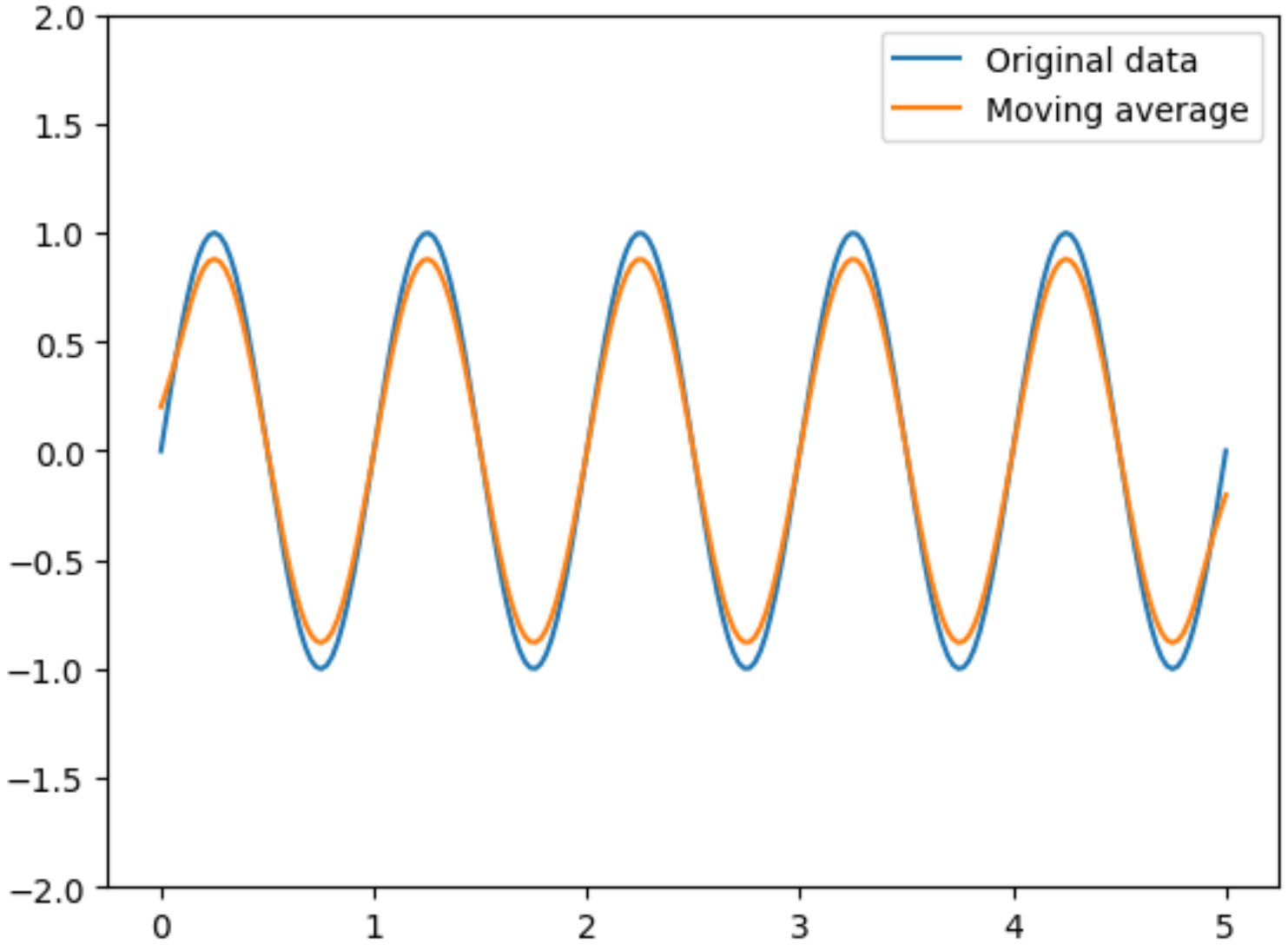
Moving average

Compute the mean value of data within a window

Example: moving average with 3-size window

1	2	3	3	2	1	1	2	3
	2	8/3	8/3	2	4/3	?	2	

(1+1+2)/3=4/3

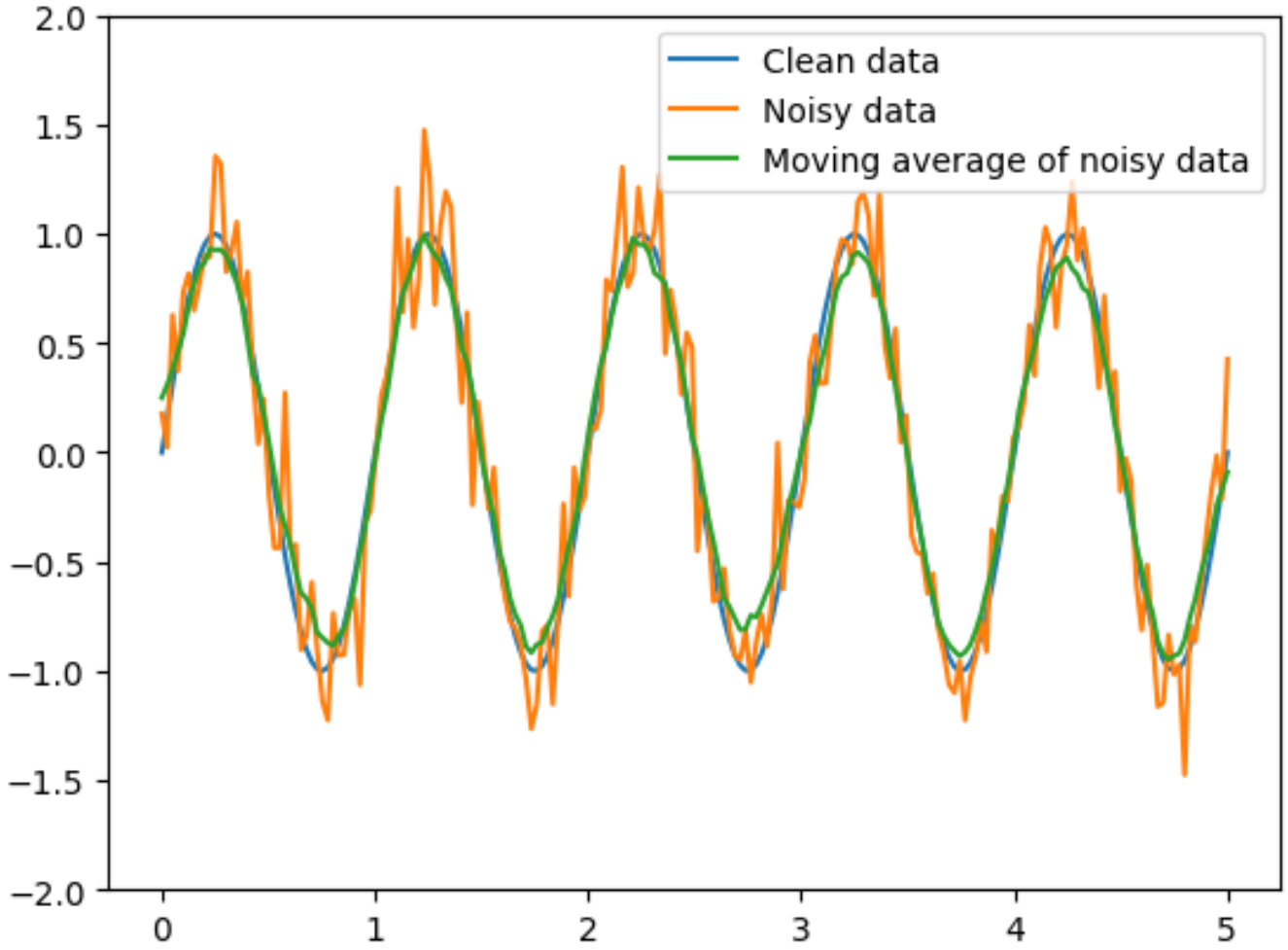


Sinusoidal signal (200 points)
11-element window for moving average

Moving average

Moving average is robust to random additive noise

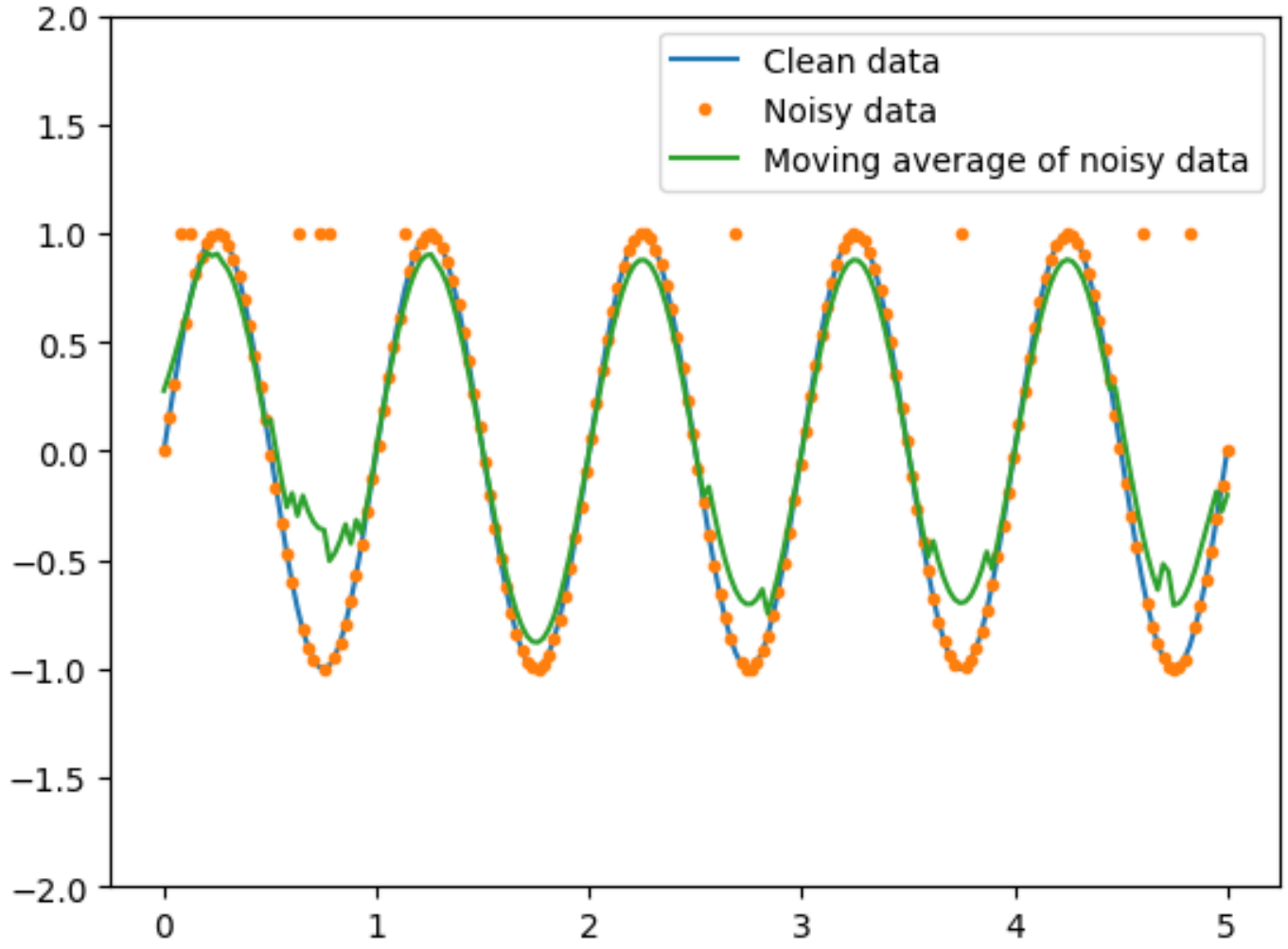
- Noisy signal = clean signal + noise
- If noise has zero mean, then it gets averaged out



Sinusoidal signal + 0.2 stdev noise (200 points)
11-element window for moving average

Moving average

Moving average is not robust to outliers because these can pull the average far in one direction



Sinusoidal signal w/ 5% outliers (replaced with 1)
11-element window for moving average

Why are outliers common?

- Simple noise can sometimes lead to majorly wrong values
  - E.g. in estimating point clouds, slight errors in estimating corresponding pixels can lead to large errors in 3D point estimates
- Data may have missing values that are filled with constants
  - E.g. unknown salaries may be filled with “0”
- Data may have incorrectly entered values
  - E.g. some salaries are entered in thousands or some entries had typos
- Naturally occurring processes are not fully modeled
  - E.g. stocks could split or merge, or a company could go bankrupt, leading to misleading or exceptional price changes
  - Sensors may be occasionally blocked by another object, or briefly output erroneous values
- Values could be correct but non-representative
  - E.g. average net worth of Harvard drop-outs is very high due to Bill Gates ($110B), Mark Zuckerburg ($79B), and Bom Kim ($2.8B)

Median is more robust

Moving median: return median within each window

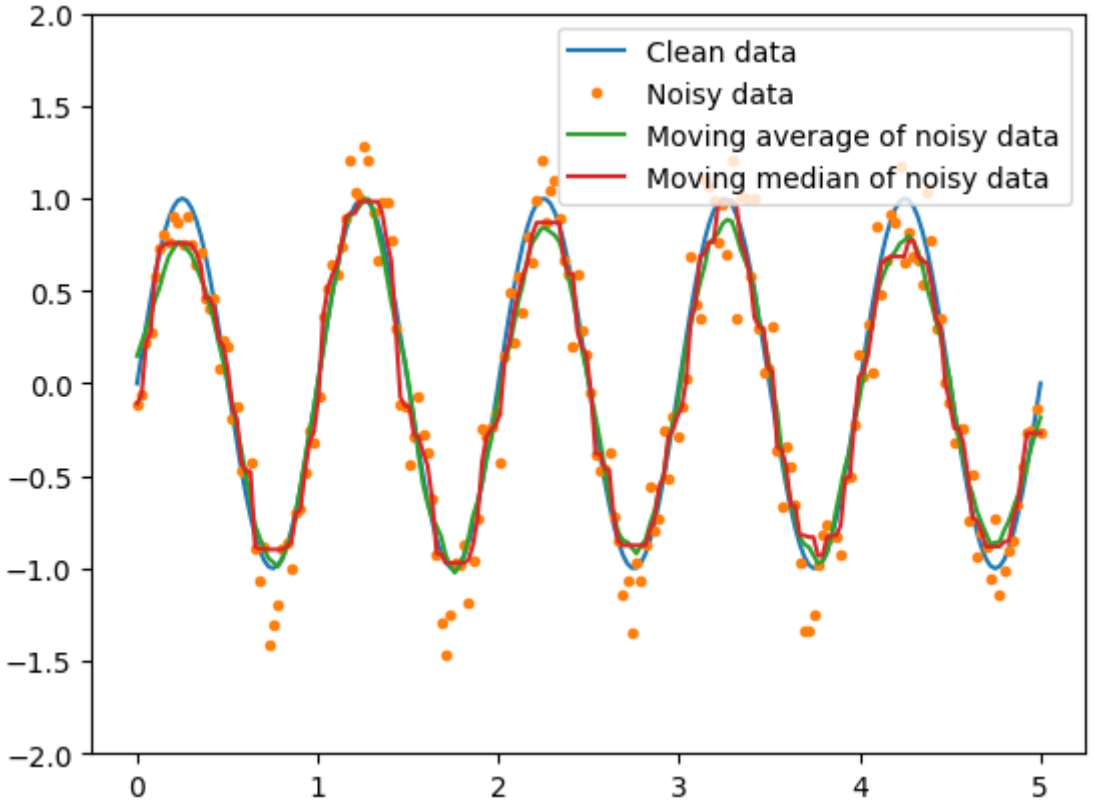
- Mean(1, 2, 7)=3
- Median(1, 2, 7) =2
- Mean(1, 2, 96) = 33
- Median(1, 2, 96) = 2

data	1	2	3	3	2	1	1	2	3
mean		2	8/3	8/3	2	4/3	4/3	2	
median		2	3	3	2	1	1	2	

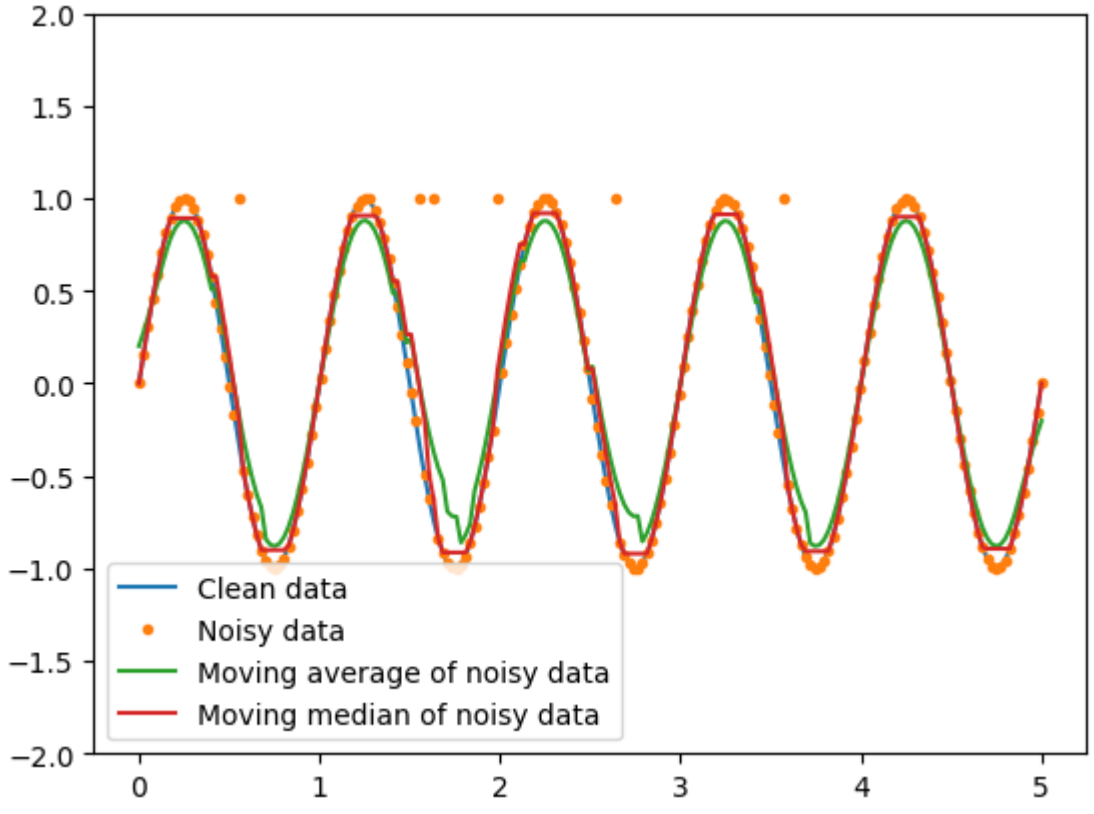
data	1	2	-50	3	2	1	200	2	3
mean		47/3	15	15	2	203/3	203/3	205/3	
median		2	3	2	2	?	?	?	

2 2 3

Moving median vs. moving mean

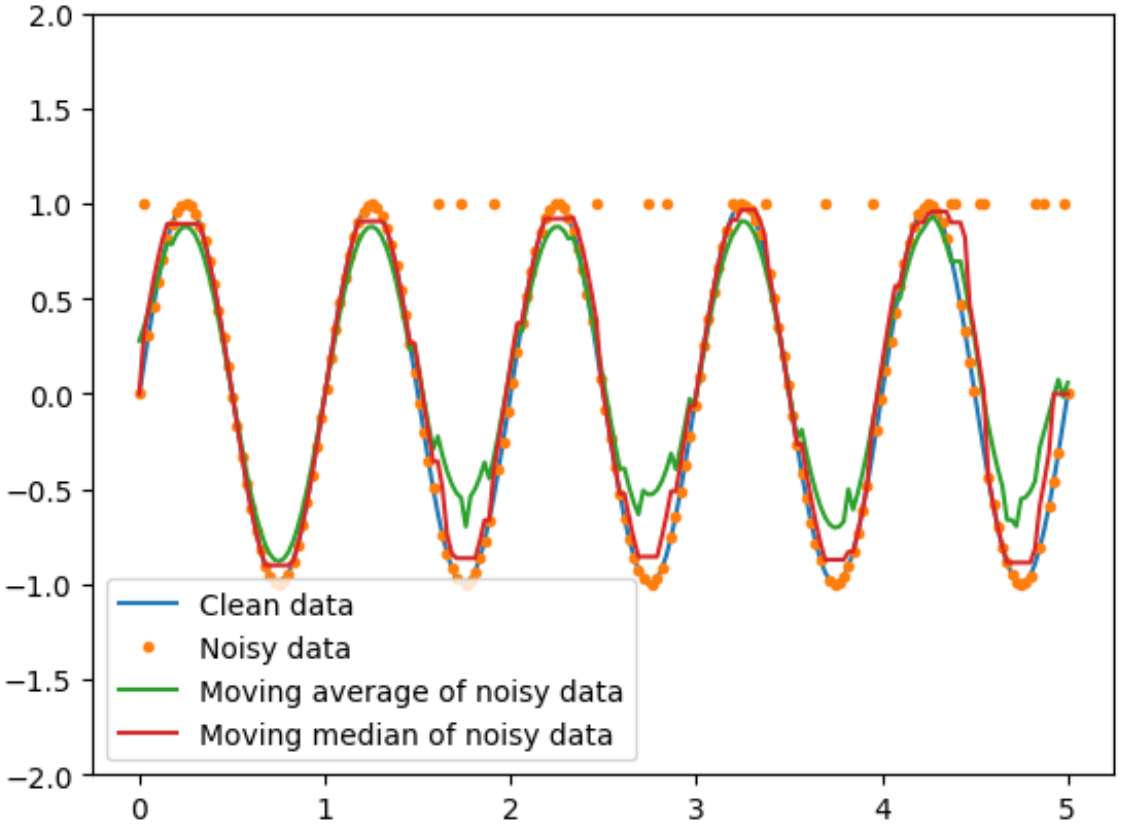


Gaussian noise

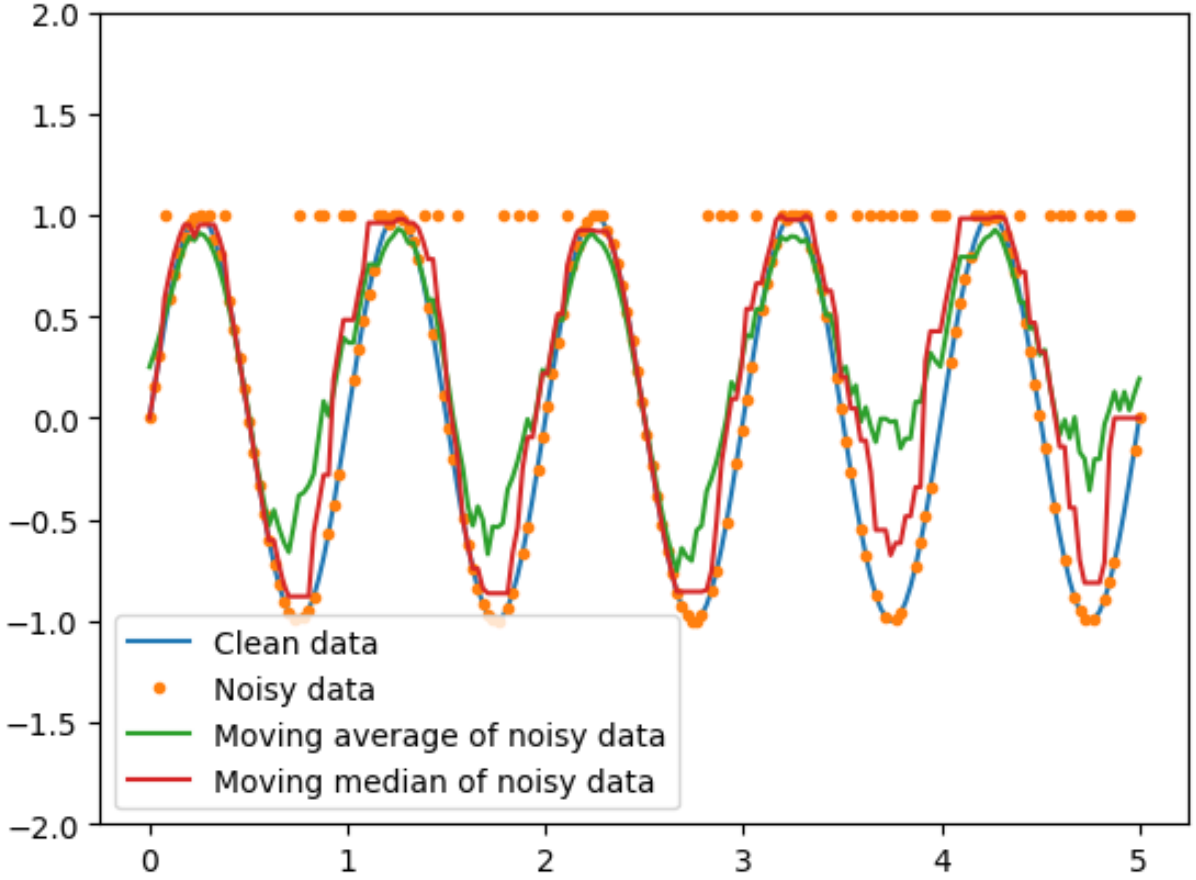


5% Outlier noise

Moving median vs. moving mean



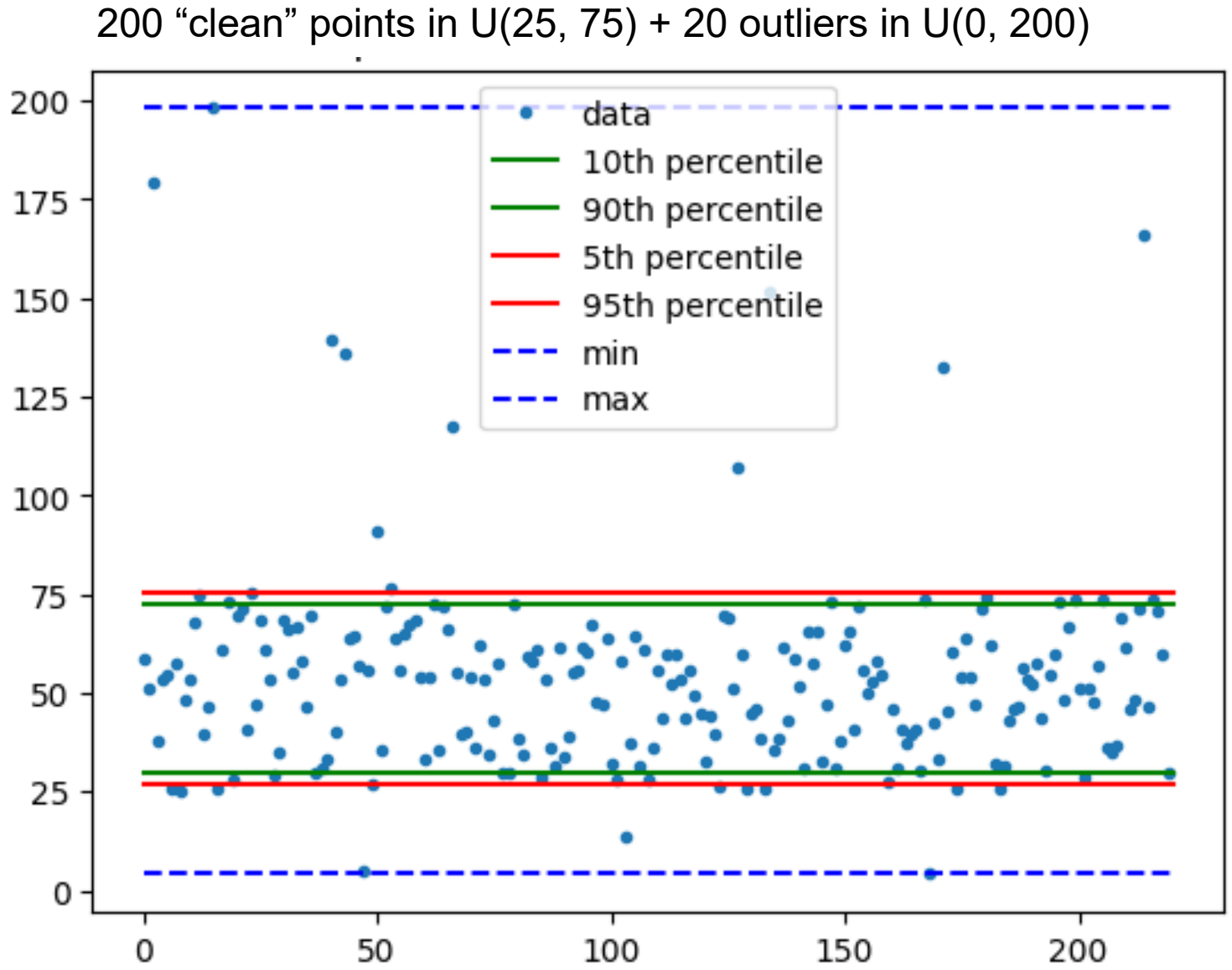
10% outlier noise



25% outlier noise

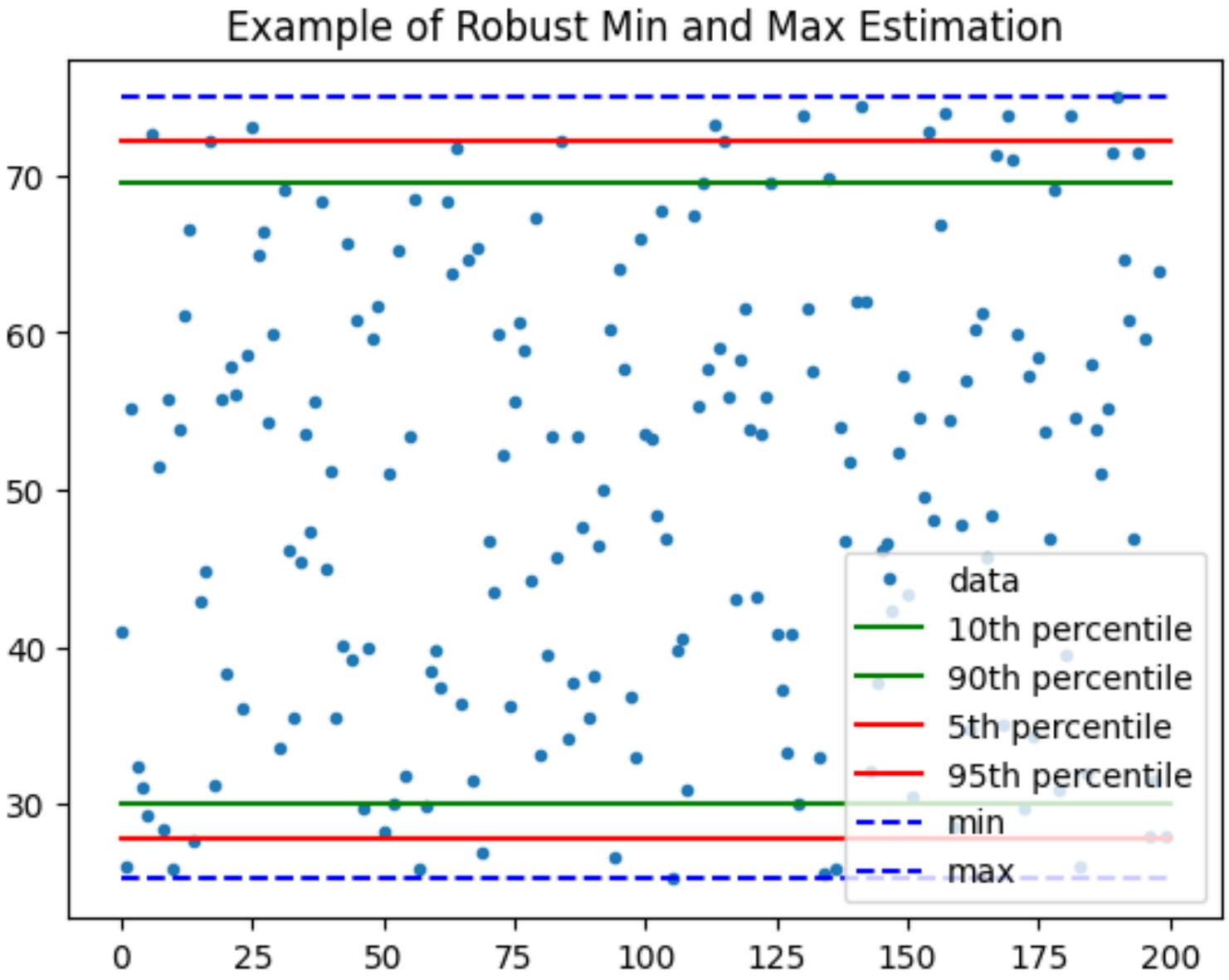
Robust Min and Max Estimation

- Min of range
  - True: 25
  - Min data: 4
  - 5th pct: 27
  - 10th pct: 30
- Max of range
  - True: 75
  - Max data: 198
  - 95th pct: 76
  - 90th pct: 72

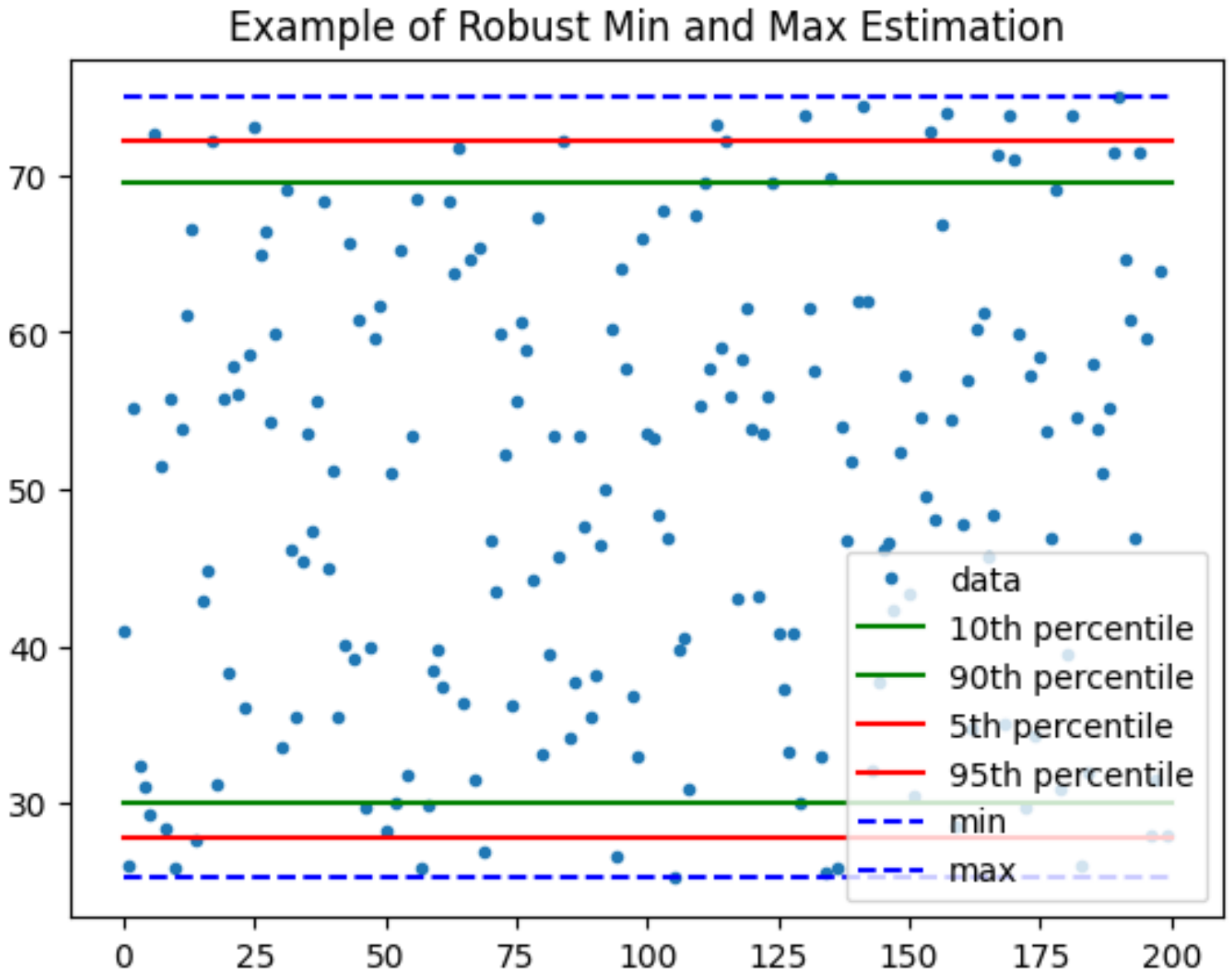


Robust Min and Max in clean data

- (min, 5th, 10th) = (25, 28, 30)
- (max, 95th, 90th) = (75, 72, 70)



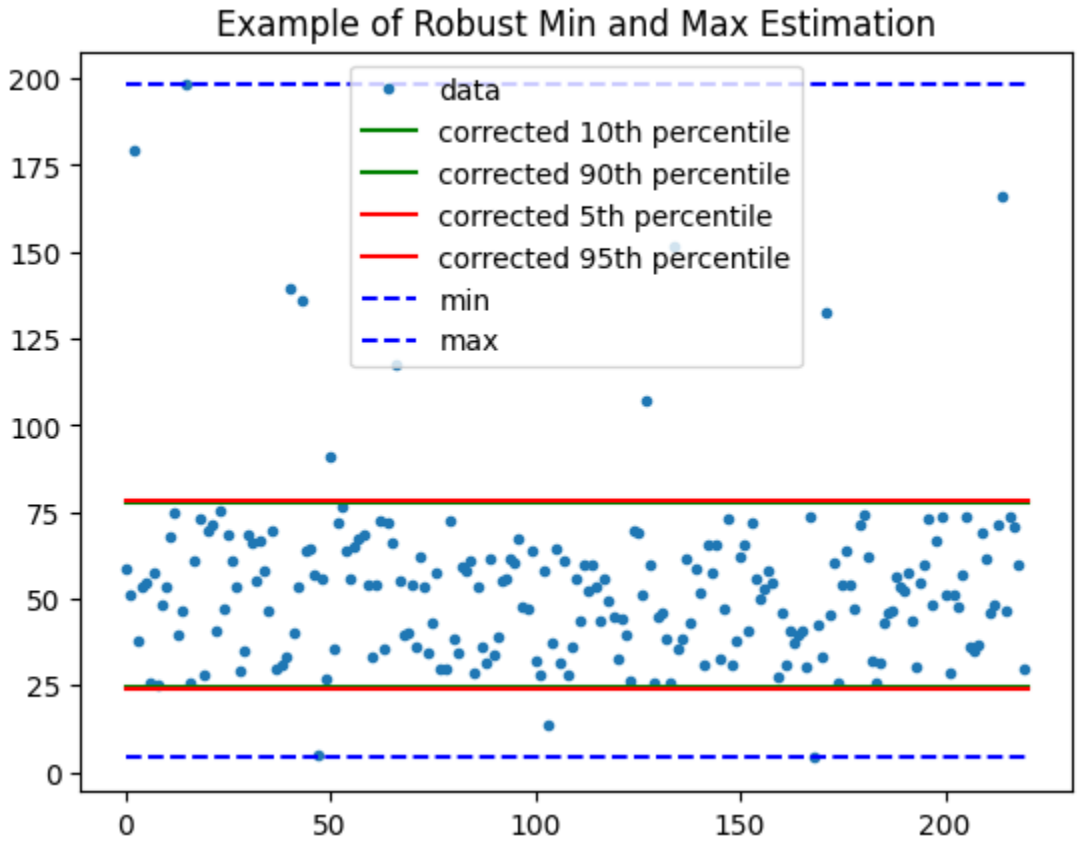
Percentiles give consistent under-estimate of range when data is clean. Can we correct for this?

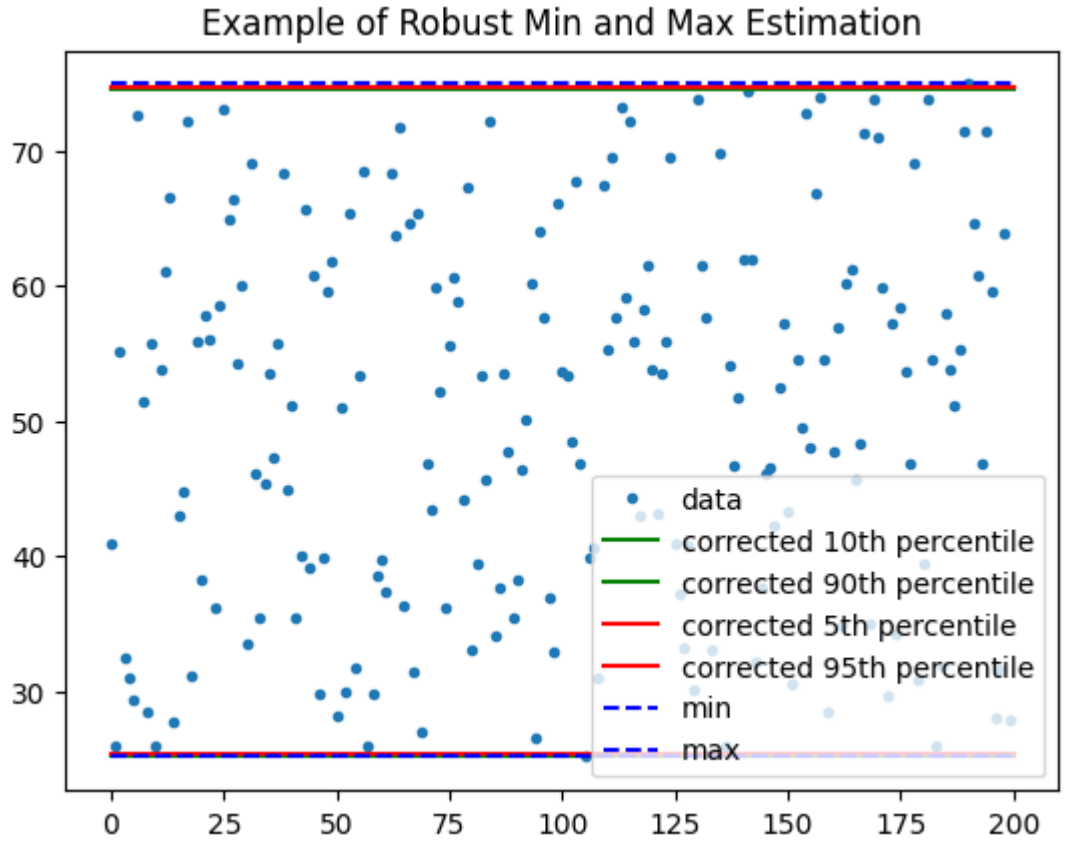


Robust Min and Max, Corrected

- Corrected by assuming that distribution is uniform, so e.g.

$$\max(x) \approx pct(x, 90) + \big(pct(x, 90) - pct(x, 10)\big) * 0.1/0.8$$





Detecting outliers

Outlier detection involves identifying which data points are distractors, not just robustly estimating statistics

If we can detect and remove outliers, we can use any method for further analysis

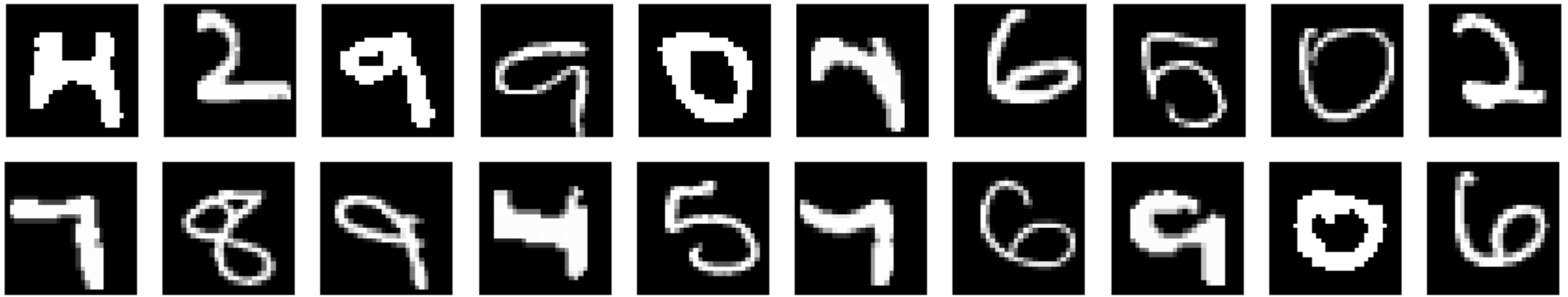
How might we detect outliers with PCA and/or Gaussian Mixture Model?

Detecting outliers: low probability data points

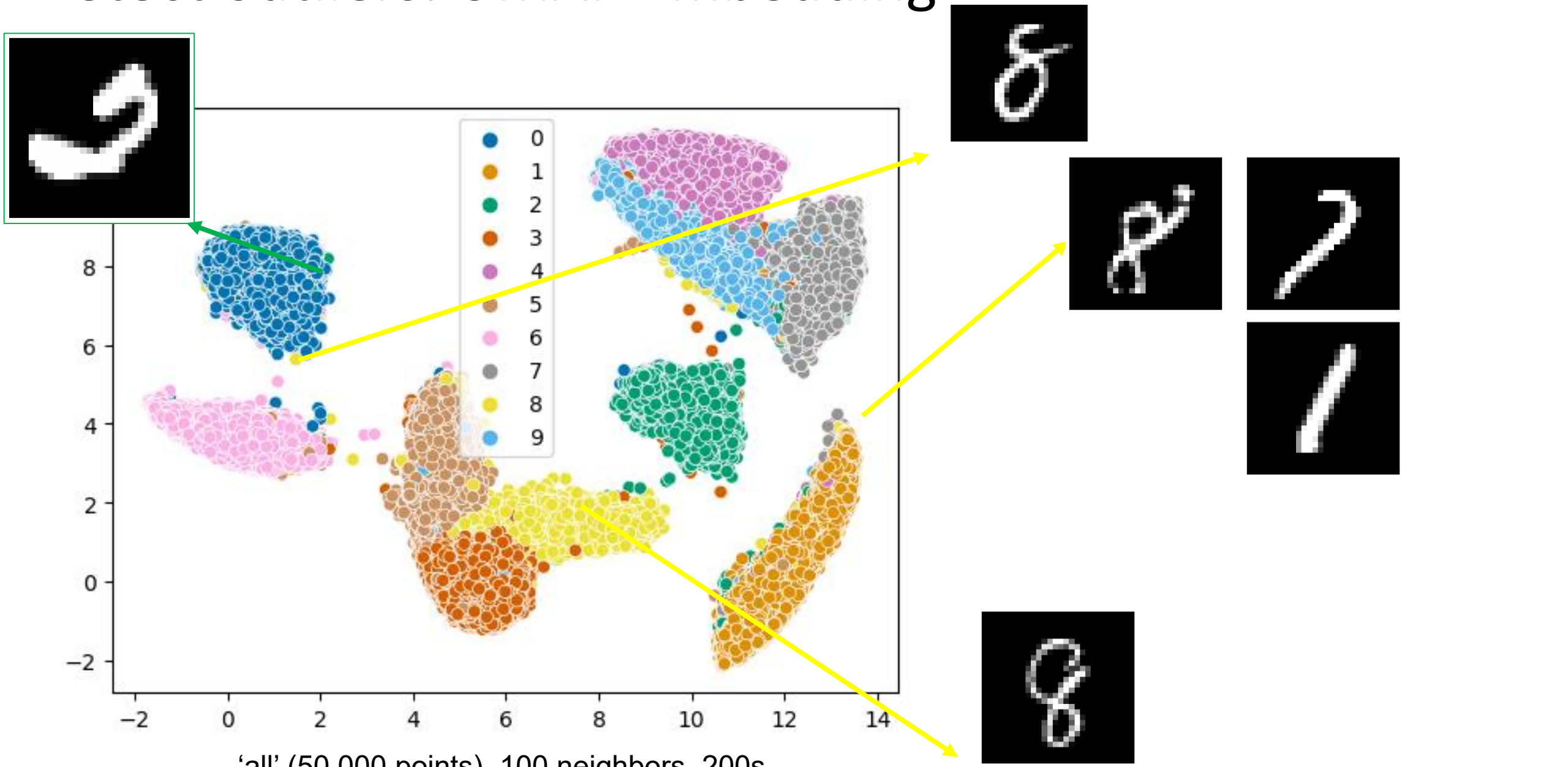
- Estimate 40 component diagonal GMM
- Find 20 lowest probability examples (only considering features, not labels)

Detect outliers: PCA Reconstruction Error

- Compress to 100 PCA coefficients
- Reconstruct, and measure reconstruction error
- Show examples with highest reconstruction error



Detect outliers: UMAP Embedding

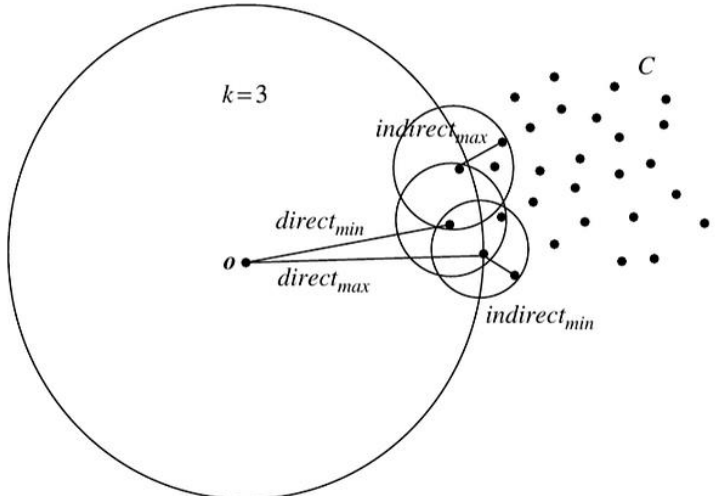


‘all’ (50,000 points), 100 neighbors, 200s

Identifying outliers based on density

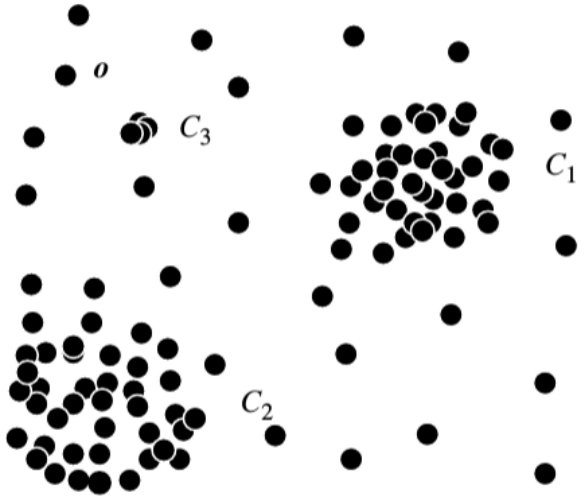
Based on neighborhood density:

1. Compute average density based on some samples, e.g. inverse density is average distance to K neighbors
2. If point has much lower density than its neighbors, it is an outlier



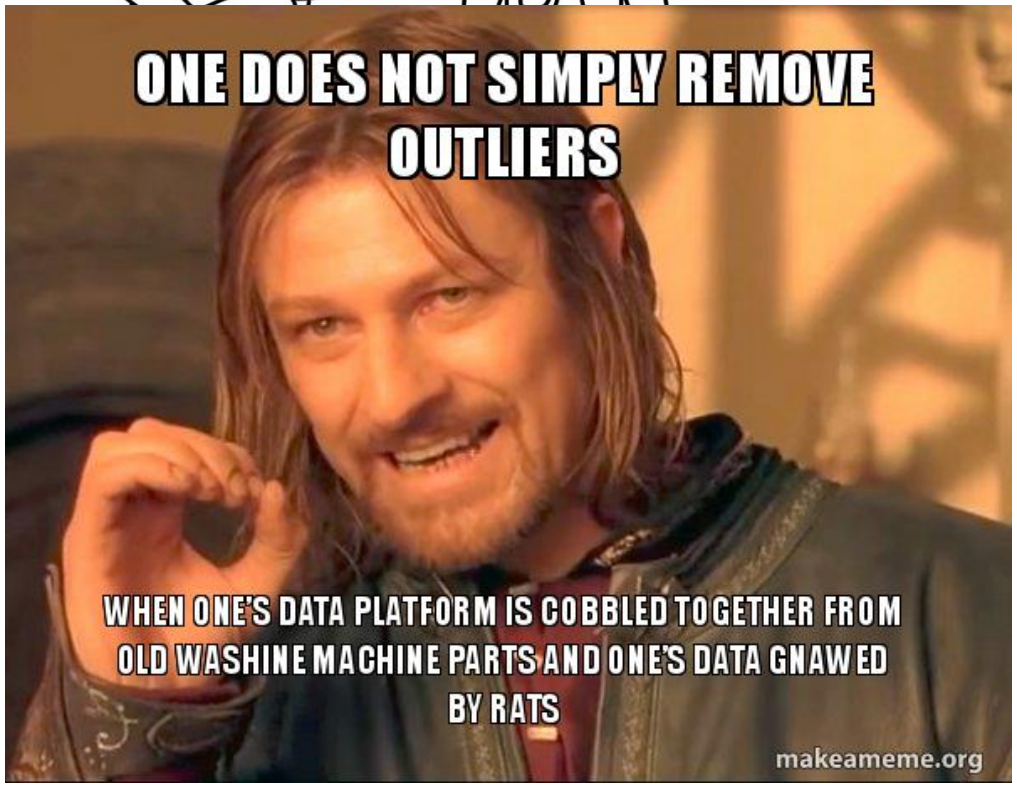
Based on a radius:

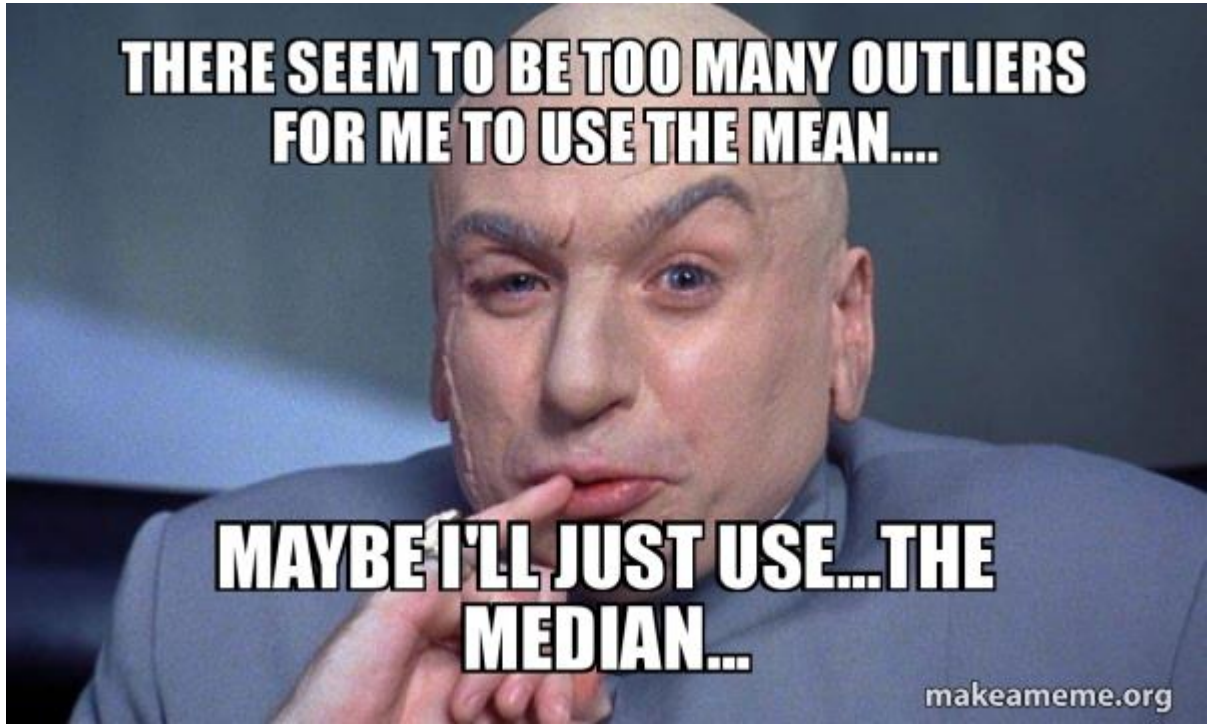
1. Compute average number of points within some radius of another point
2. Any points that have much lower density than average are outliers



Figs: https://towardsdatascience.com/density-based-algorithm-for-outlier-detection-8f278d2f7983

2 minute break





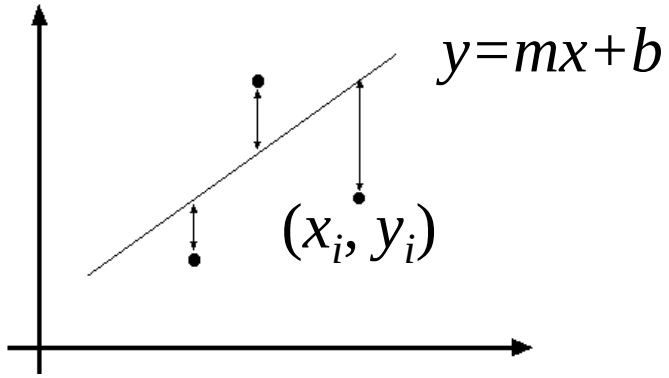
Robust estimation

- Fitting data with robustness to outliers

Least squares line fitting

- Data: $(x_1, y_1), \ldots, (x_n, y_n)$
- Line equation: $y_i = m x_i + b$
- Find (m, b) to minimize

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



$$E = \sum_{i=1}^{n} \left(\begin{bmatrix} x_i & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - y_i \right)^2 = \left\| \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \right\|^2 = \|\mathbf{Ap} - \mathbf{y}\|^2$$

$$= \mathbf{y}^T \mathbf{y} - 2(\mathbf{Ap})^T \mathbf{y} + (\mathbf{Ap})^T (\mathbf{Ap})$$

$$\frac{dE}{dp} = 2\mathbf{A}^T \mathbf{Ap} - 2\mathbf{A}^T \mathbf{y} = 0$$

$$\mathbf{A}^T \mathbf{Ap} = \mathbf{A}^T \mathbf{y} \Rightarrow \mathbf{p} = \left(\mathbf{A}^T \mathbf{A} \right)^{-1} \mathbf{A}^T \mathbf{y}$$

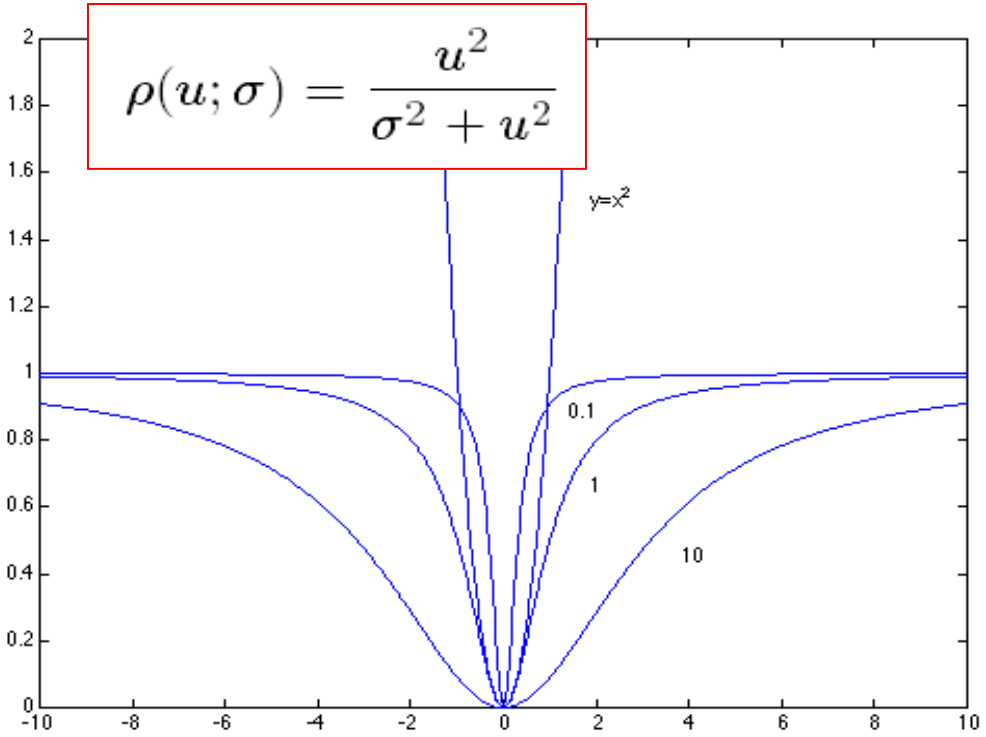
Modified from S. Lazebnik

Robust least squares (to deal with outliers)

General approach:

minimize $\sum_{i} \rho\left(u_i\left(x_i, \theta\right); \sigma\right)$ $\qquad u^2 = \sum_{i=1}^{n} (y_i - mx_i - b)^2$

$u_i(x_i, \theta)$ – residual of ith point w.r.t. model parameters ϑ
ρ – robust function with scale parameter σ



The robust function ρ

- Favors a configuration with small residuals
- Constant penalty for large residuals

Slide from S. Savarese

Robust Estimator

1. Initialize: e.g., choose θ by least squares fit and
   $$\sigma = 1.5 \cdot \text{median}(error)$$
2. Choose params to minimize: $$\sum_i \frac{error(\theta, data_i)^2}{\sigma^2 + error(\theta, data_i)^2}$$
   – E.g., numerical optimization
3. Compute new $$\sigma = 1.5 \cdot \text{median}(error)$$
4. Repeat (2) and (3) until convergence

Iteratively Reweighted Least Squares (IRLS)

Goal solve an optimization involving a robust norm, e.g. p=1

$$\arg\min_{\beta} \sum_{i=1}^{n} \left| y_i - f_i(\boldsymbol{\beta}) \right|^p$$

1. Initialize weights w to 1
2. Solve for parameters β that minimize weighed squared error

$$\boldsymbol{\beta}^{(t+1)} = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} w_i^{(t)} |y_i - X_i \boldsymbol{\beta}|^2 = (X^{\mathrm{T}} W^{(t)} X)^{-1} X^{\mathrm{T}} W^{(t)} \mathbf{y}$$

3. Assign new weights based on error

$$w_i^{(t)} = \left| y_i - X_i \boldsymbol{\beta}^{(t)} \right|^{p-2}$$

RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in ‘81.

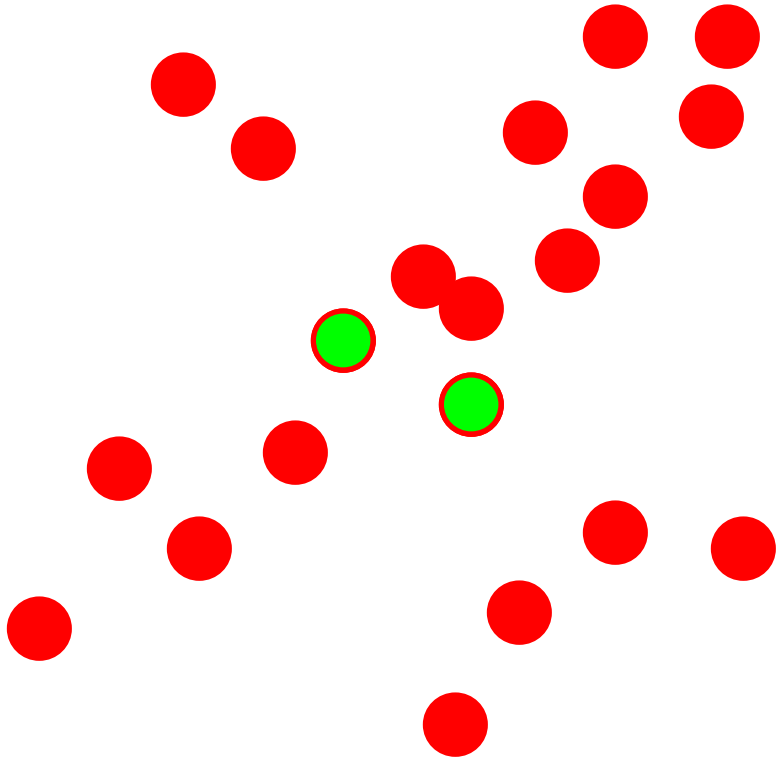
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

Line fitting example



Algorithm:

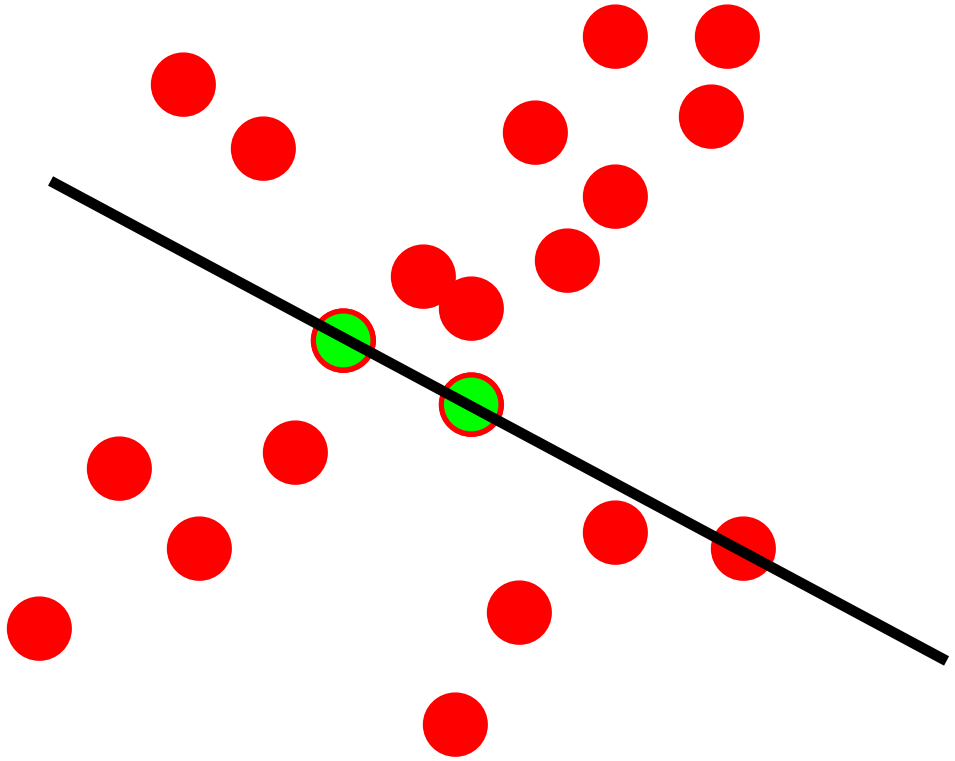
1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Illustration by Savarese

RANSAC

Line fitting example



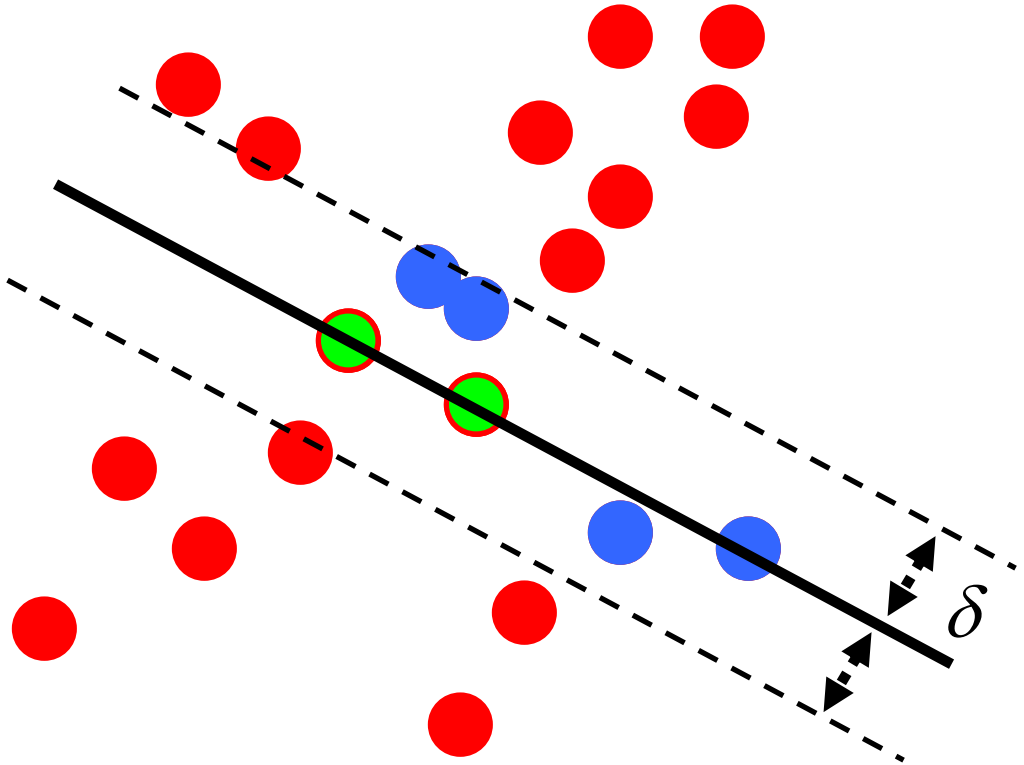
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

Line fitting example



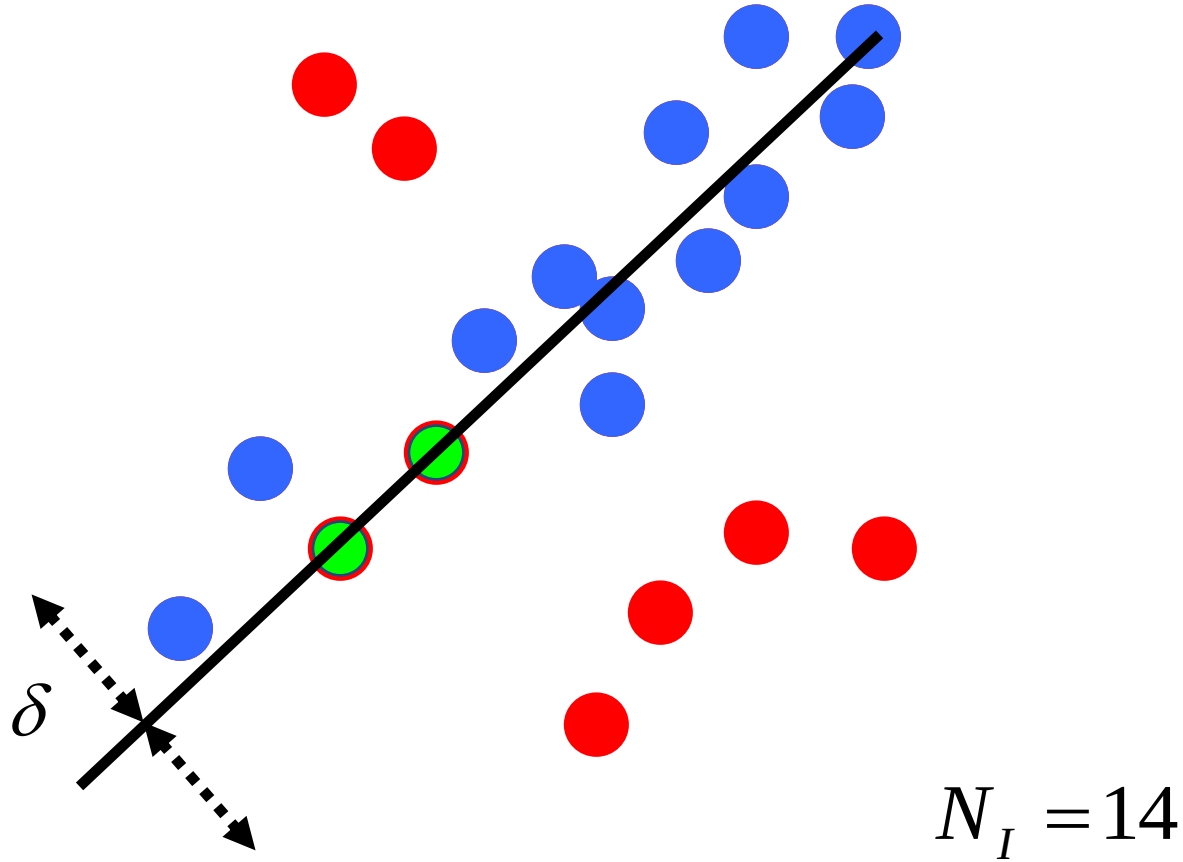
$$N_I = 6$$

Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC



$N_I = 14$

Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

How to choose parameters?

- 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*)
- Number of sampled points *s*
  - Minimum number needed to fit the model
- Distance threshold δ
  - Choose δ so that a good point with noise is likely (e.g., prob=0.95) within threshold
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$

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

	proportion of outliers *e*						
s	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

Advanced algorithms automatically find N and δ

modified from M. Pollefeys

RANSAC conclusions

Good

- Robust to outliers
- Advanced forms can automatically estimate thresholds and number of iterations

Bad

- Computational time grows quickly with fraction of outliers and number of parameters

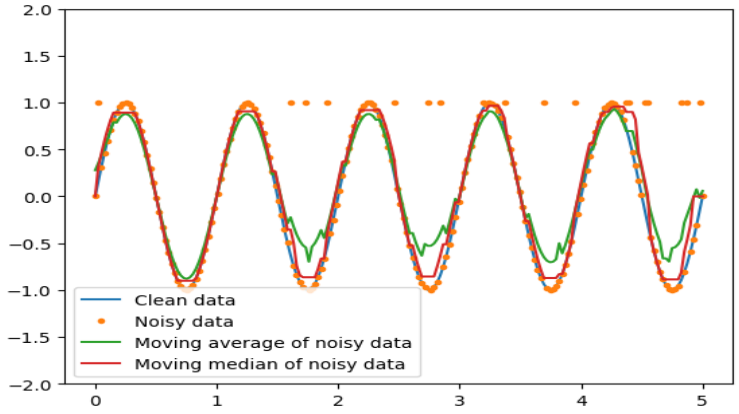
Matlab demo

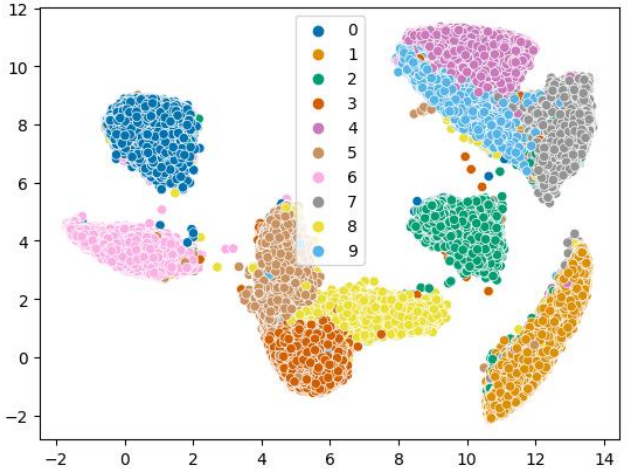
Things to remember

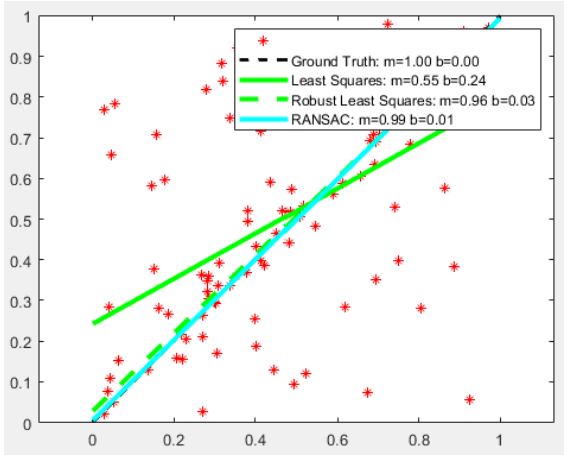
Median and quantiles are robust to outliers, while mean/min/max aren't

Outliers can be detected as low probability points, low density points, poorly compressible points, or through 2D visualizations

Least squares is not robust to outliers. Use RANSAC or IRLS or robust loss function instead.







Next class

- Decision trees