# ECE276A: Sensing & Estimation in Robotics
# Lecture 14: Robust Estimation

Lecturer:
Nikolay Atanasov: natanasov@ucsd.edu

Teaching Assistants:
Siwei Guo: s9guo@eng.ucsd.edu
Anwesan Pal: a2pal@eng.ucsd.edu

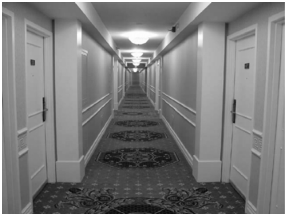
UC San Diego
JACOBS SCHOOL OF ENGINEERING
Electrical and Computer Engineering

# Ground Plane Detection

- **Hyperplane**: a set $\{x \in \mathbb{R}^n \mid \eta^T x = \eta^T x_0\}$, where $\eta \in \mathbb{R}^n$, $\eta \neq 0$ is the **normal vector** and $x_0 \in \mathbb{R}^n$ is any point in the hyperplane so that $b := \eta^T x_0 \in \mathbb{R}$ determines the offset of the hyperplane from the origin.
- The **ground plane** in the world frame is $\{x \in \mathbb{R}^3 \mid \eta_g^T x = 0\}$ with $\eta_g = (0, 0, 1)^T$
- Consider a body frame with position $p \in \mathbb{R}^3$ and orientation $R \in SO(3)$. The set of points in the body frame that belongs to a world-frame plane $\{x \in \mathbb{R}^3 \mid \eta^T x = b\}$ is $\{y \in \mathbb{R}^3 \mid \eta^T(Ry + p) = b\}$.
- Simple ground plane detection: $|\eta_g^T(Ry + p)| \leq \epsilon$ for some small $\epsilon \in \mathbb{R}$.
- **Plane fitting**: to find planes in a point cloud $\{x_i \in \mathbb{R}^3\}$, we need to find parameters $\eta$ and $b$ that fit many of the points $x_i$

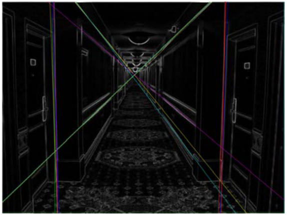
# Line Detection

- Use a similar idea to detect lines $\{y \in \mathbb{R}^2 \mid \eta^T y = b\}$ in an image
- Assume that we have performed edge detection:
  - Convolve $I$ with Sobel/Gaussian filter to get $I_u$ (horizontal edges) and $I_v$ (vertical edges)
  - Gradient magnitude $g(u,v) := \sqrt{I_u(u,v)^2 + I_v(u,v)^2}$ and orientation $\alpha(u,v) := \arctan\left(\frac{I_v(u,v)}{I_u(u,v)}\right)$ (angle with respect to $u$-axis)
  - Threshold the image gradient magnitude $g(u,v)$ to obtain $n$ pixels $y_i$ that *may* describe object boundaries
- To find lines in the image, we need to find parameters $\eta$ and $b$ that fit many of the points $y_i$



Image

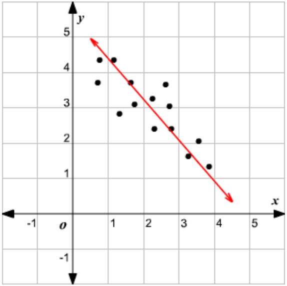
Conv. with Sobel filter



Line features

# Robust Estimation

- How should we:
  - Extract lines from 2-D points (e.g., walls from laser scan, line features in an image)
  - Extract planes from 3-D points (e.g., ground plane or walls from RGB-D images)
  - Match image features (e.g., Harris corners) across images




- **Least squares**: given $D := \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ determine parameters $\beta \in \mathbb{R}^d$:

$$\min_{\beta} \sum_{i=1}^{n} \left(y_i - \beta^T x_i\right)^2$$

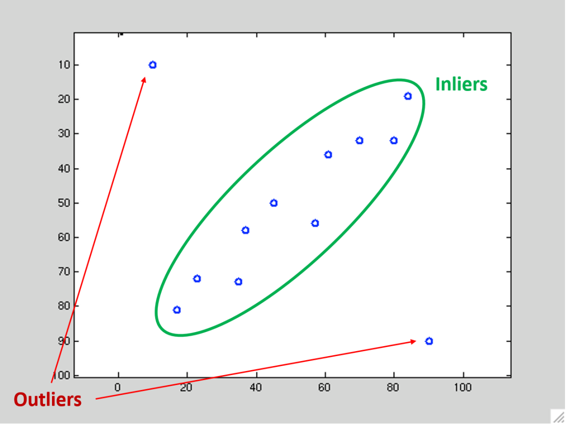
- **Example**: given $D := \{(u_i, v_i)\}_{i=1}^n$ determine line parameters $a, b$ via: $\min_{a,b} \sum_{i=1}^n (au_i + bv_i - 1)_2^2$

- The least squares fit is **sensitive** to noise, outliers, missing data...

- Robotics philosophy: never trust a single point!

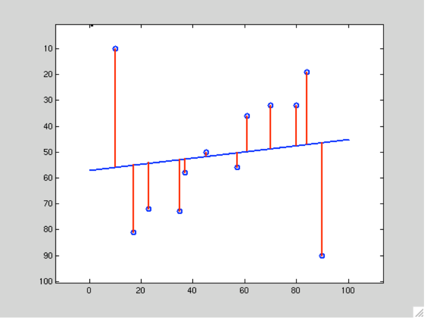
# Outliers

- **Inliers**: points that fit the model
- **Outliers**: points that do not fit the model

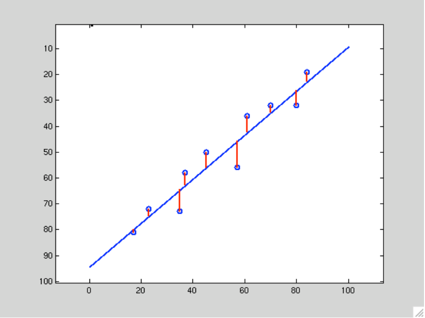


# Problems due to Outliers

- a few outliers can greatly skew the results of least squares estimation




Least squares fit




Robust least squares

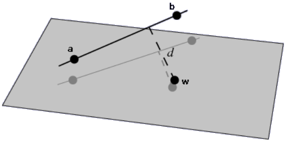
- **Idea**: robust estimation is a two-stage process:
  1. Classify data points as outliers or inliers
  2. Fit the model to the inliers only

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

# Random Sample Consensus (RANSAC)

- **RANSAC line fitting**:
  - Pick two data points at random and generate corresponding line
  - Count the number of inliers (points whose **point-to-line** distance is small)
  - Repeat
  - Pick the line with max number of inliers

- **Point-to-line distance** for point $w \in \mathbb{R}^n$ and the line between points $a, b \in \mathbb{R}^n$:




$$d(w, a \to b) := \frac{\|(b-a) \times (a-w)\|_2}{\|b-a\|_2}$$

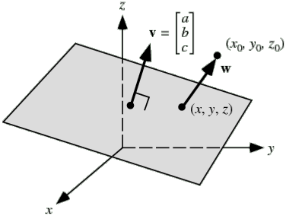
  - Numerator: twice the area of the triangle formed by $a$, $b$, and $w$
  - Denominator: length of the triangle base

# Random Sample Consensus (RANSAC)

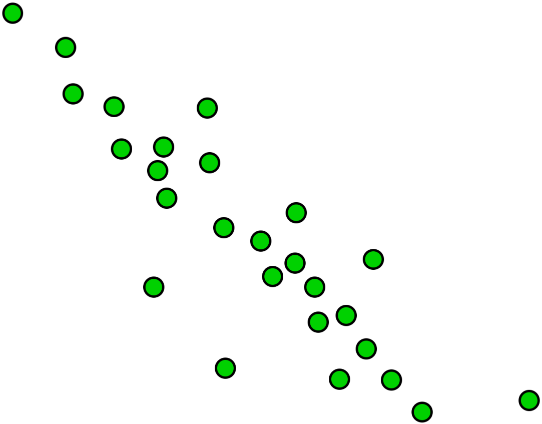
- **RANSAC plane fitting**:
  - Pick 3 data points at random and generate corresponding plane
  - Count the number of inliers (points whose **point-to-plane** distance is small)
  - Repeat
  - Pick the plane with max number of inliers

- **Point-to-plane distance** for point $w \in \mathbb{R}^n$ and the plane $v^T(x-a)=0$ through point $a \in \mathbb{R}^n$ with normal $v \in \mathbb{R}^n$:

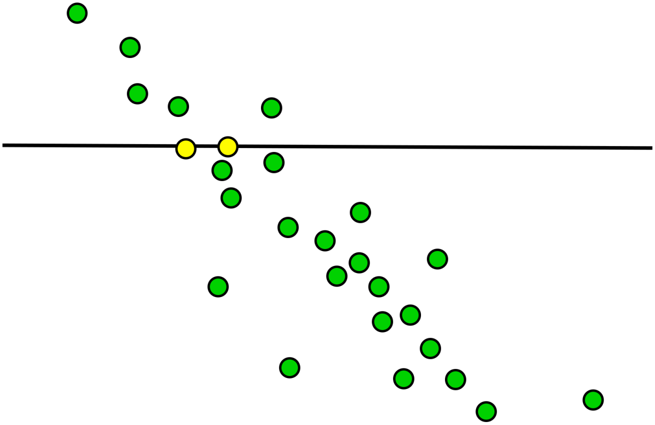
$$d(w, v^T(x-a)=0) := \frac{|v^T(w-a)|}{\|v\|_2}$$



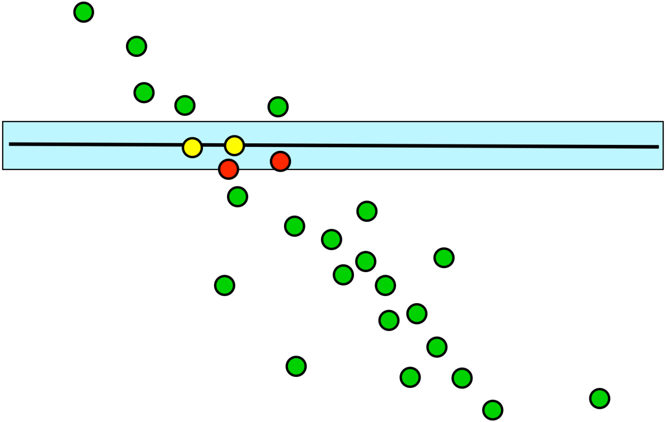
# Random Sample Consensus (RANSAC)



# Random Sample Consensus (RANSAC)

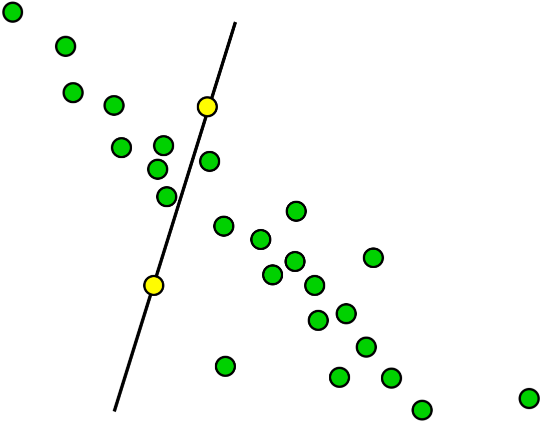


# Random Sample Consensus (RANSAC)

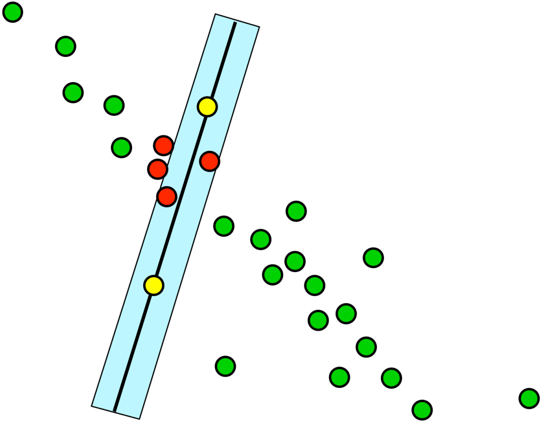


**Count = 4**

# Random Sample Consensus (RANSAC)

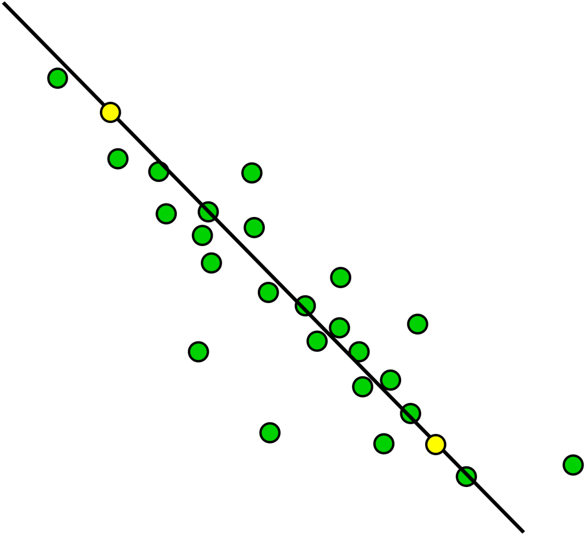


# Random Sample Consensus (RANSAC)

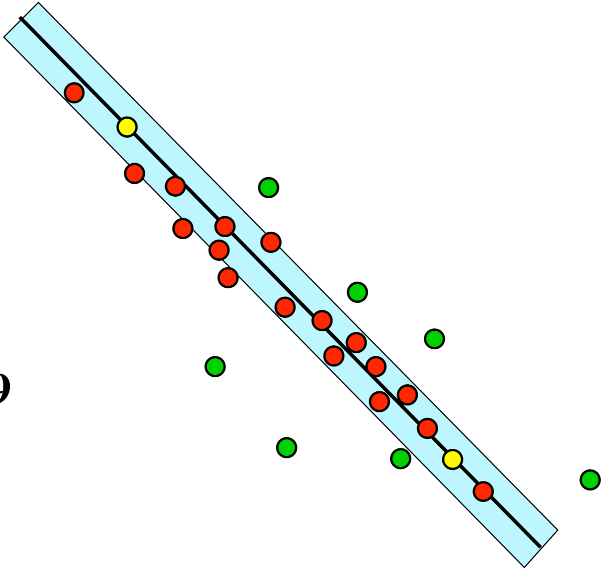


**Count = 6**

# Random Sample Consensus (RANSAC)

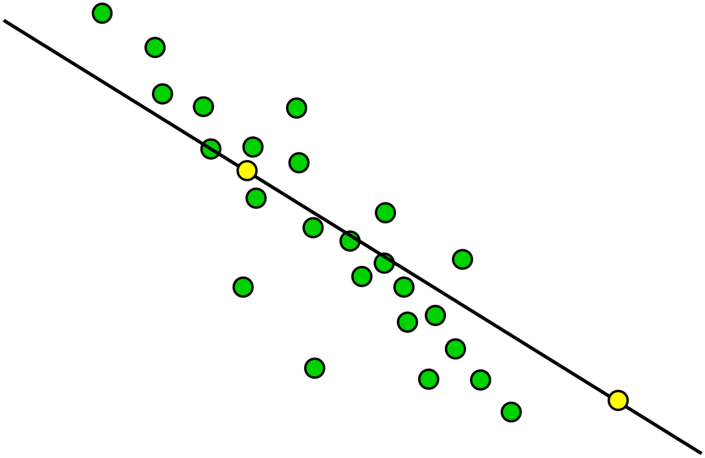


# Random Sample Consensus (RANSAC)

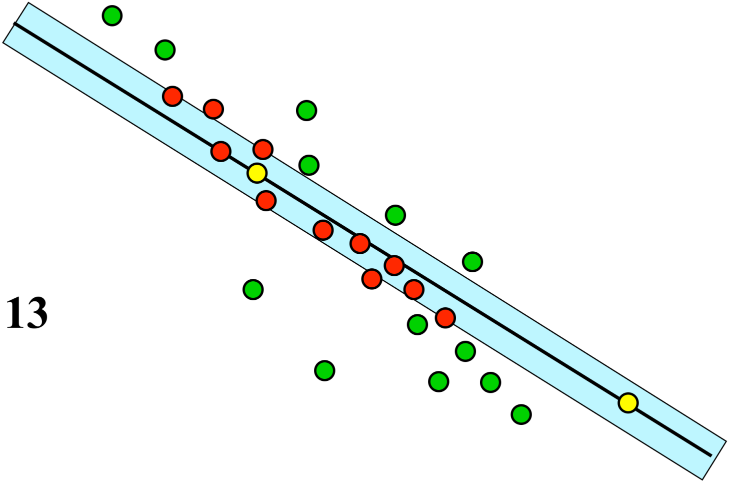


**Count = 19**

# Random Sample Consensus (RANSAC)

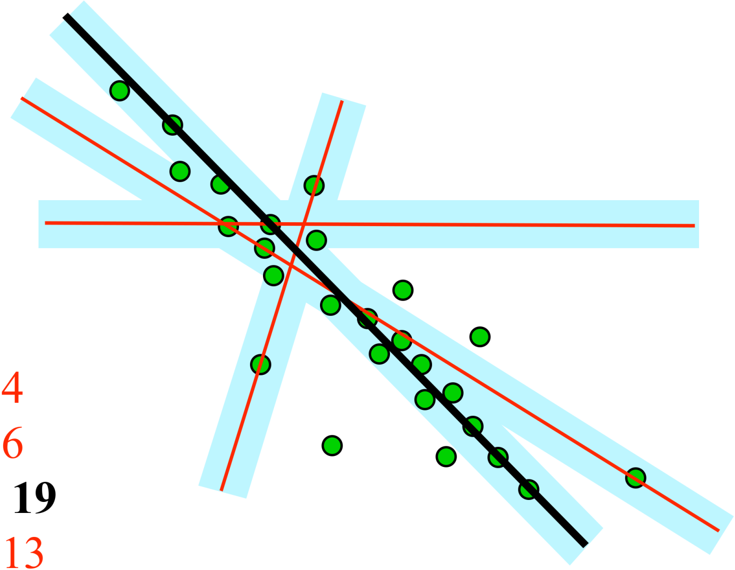


# Random Sample Consensus (RANSAC)



**Count = 13**

# Random Sample Consensus (RANSAC)



Count = 4

Count = 6

**Count = 19**

Count = 13

# Random Sample Consensus (RANSAC)

- **Termination criteria**: how many times should we repeat the RANSAC procedure?

$$1 - (1 - (1 - e)^S)^N = p \quad \Rightarrow \quad \boxed{N = \frac{\log(1 - p)}{\log(1 - (1 - e)^S)}}$$

- $p =$ desired probability for a good sample
- $N =$ number of RANSAC repetitions
- $S =$ number of points in a sample (e.g., 2 for a line)
- $e =$ probability that a point is an outlier
- $(1 - e) =$ probability of an inlier
- $(1 - e)^S =$ probability of $S$ inliers
- $1 - (1 - e)^S =$ probability that one or more points in the sample are outliers
- $(1 - (1 - e)^S)^N =$ probability that all $N$ samples contain outliers
- $1 - (1 - (1 - e)^S)^N$ probability that at least one sample does not contain outliers

# Number of RANSAC Samples

- Choose $N$ so that with probability $p = 0.99$ at least one sample is outlier-free

| | 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 |

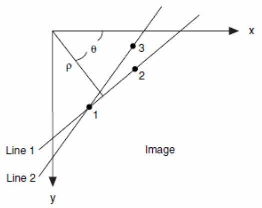
- **Example**: line-fitting with 12 points of which 2 are outliers, i.e., $e = 2/12 = 20\%$, since $S = 2$ points are needed per sample, $N = 5$ gives a 99% chance of obtaining an outlier-free sample. This is in contrast to $N = 66$ samples needed to try every pair of points.

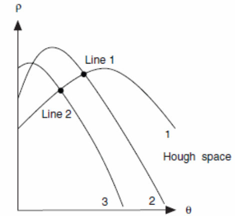
# Hough Transform

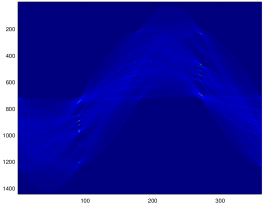
- **Voting scheme**: each data point $\mathbf{x}_i$ votes for several parameters $\beta$ that are consistent with it
- Used to find parametric curves, e.g., line, polynomial, circle, ellipse, etc.
- Handles missing and occluded data
- **Idea**: accumulate the consistent parameters $\beta$ for each data sample $(\mathbf{x}_i, y_i)$ in parameter space $\mathcal{B}$
  - The space $\mathcal{B}$ is discretized into a set $\mathcal{A}$ (**accumulator**)
  - Each training point $(\mathbf{x}_i, y_i)$ votes for the consistent cells in $\mathcal{A}$, i.e., $\beta_j$ that satisfy $y_i = \beta_j^T \mathbf{x}_i$
- The discretization of the accumulator makes the algorithm computationally demanding for high dimensional curves
- The lengths and positions of the curves cannot be determined

# Hough Transform

- Line fitting for 2-D image features $\{u_i, v_i\}$
- **Normal equation** of line: $u \cos\theta + v \sin\theta = \rho$
  - $\theta$ – angle of the line normal wrt the origin
  - $\rho$ – distance to the line along the normal
- **Accumulator**:
  - Discretize the $(\rho, \theta)$ space, e.g., $\theta \in [-90^\circ, 90^\circ]$ and $\rho \in [-N\sqrt{2}, N\sqrt{2}]$ for an $N \times N$ image.
  - Given $(u_i, v_i)$, add 1 to each consistent $(\rho, \theta)$ cell, e.g., for each $\theta_j$ increment all $\rho$ such that $\rho = u_i \cos\theta_j + v_i \sin\theta_j$
  - Repeat for every $(u_i, v_i)$
  - The most likely line hypotheses correspond to the max locations in the accumulator $\mathcal{A}[\rho, \theta]$



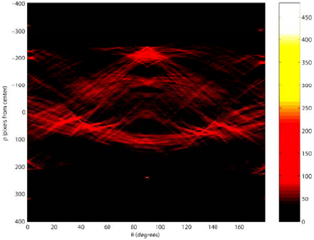



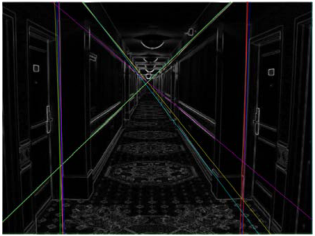
# Hough Transform

- **Line detection example**: 20 most prominent lines in a natural scene, preprocessed by convolution with a Sobel kernel and thresholding:

Conv. with Sobel filter




Accumulator



Line features

- The same idea can be used for other curves but since more parameters are needed to describe them, the accumulator needs to be higher dimensional

- **Ellipse**: need 5D accumulator $\theta = \{a, b, c, u_0, v_0\}$

$$a(u - u_0)^2 + 2b(u - u_0)(v - v_0) + c(v - v_0)^2 = 1$$

# Outlier Rejection for Least Squares

- All least-squares problems correspond to Gaussian MLE inference:

$$\arg\max_{\beta} \prod_{i=1}^{n} \phi(y_i; \beta^T \mathbf{x}_i, \sigma^2) = \arg\max_{\beta} \sum_{i=1}^{n} \log\exp\left(-\frac{1}{2\sigma^2}(y_i - \beta^T \mathbf{x}_i)^2\right)$$

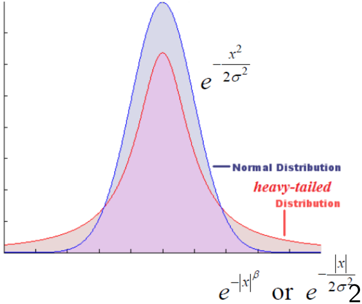
- To place less weight on outliers, choose a distribution with a **heavy tail** (slowly decaying), e.g., $\exp(-f(x))$, where $f(x)$ is the **error measure**:

$$\arg\max_{\beta} \sum_{i=1}^{n} \log\exp\Big(-f(y_i - \beta^T \mathbf{x}_i)\Big) = \arg\min_{\beta} \sum_{i=1}^{n} f(y_i - \beta^T \mathbf{x}_i)$$

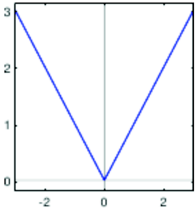
- **Huber loss**: frequently used in practice

$$f(x) = \begin{cases} \frac{x^2}{2} & \text{for } |x| \le \epsilon \\ \epsilon\left(|x| - \frac{\epsilon}{2}\right) & \text{otherwise} \end{cases}$$

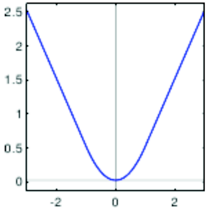
- Several others: Tukey, Cauchy, Blake-Zisserman, Corrupted Gaussian, etc.



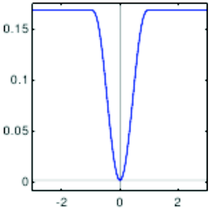
# Outlier Rejection for Least Squares



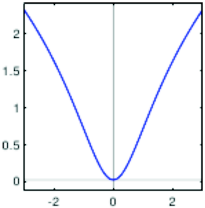

L1 norm



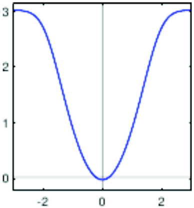

Huber



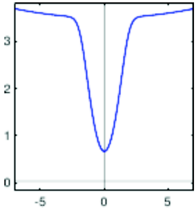

Tukey




Cauchy




Blake-Zisserman




Corrupted G.

# Iteratively Reweighted Least Squares (IRLS)

- Nonlinear least squares: $\min_{\beta} \sum_{i=1}^{n} f(y_i - \beta^T \mathbf{x}_i)$
- If $f(x)$ is below $|x|$, the problem is **not convex**
- **Idea**: construct a tight upper bound using a quadratic function
  - E. update $w_i$ to get a tight upper bound: $f(y_i - \beta^T x_i) \leq w_i (y_i - \beta^T \mathbf{x}_i)^2$
  - M. update $\beta$ by $\min_{\beta} \sum_i w_i (y_i - \beta^T \mathbf{x}_i)^2$

Example: $\min_{\beta} \sum_i |y_i - \mathbf{x}_i^T \beta|^p$

Initialize: $w_i^{(0)} = 1$

M-step: $\beta^{(t)} = \arg\min_{\beta} \sum_i w_i^{(t)} |y_i - \mathbf{x}_i^T \beta|^2$

E-step: $w_i^{(t+1)} = |y_i - \mathbf{x}_i^T \beta^{(t)}|^{p-2}$