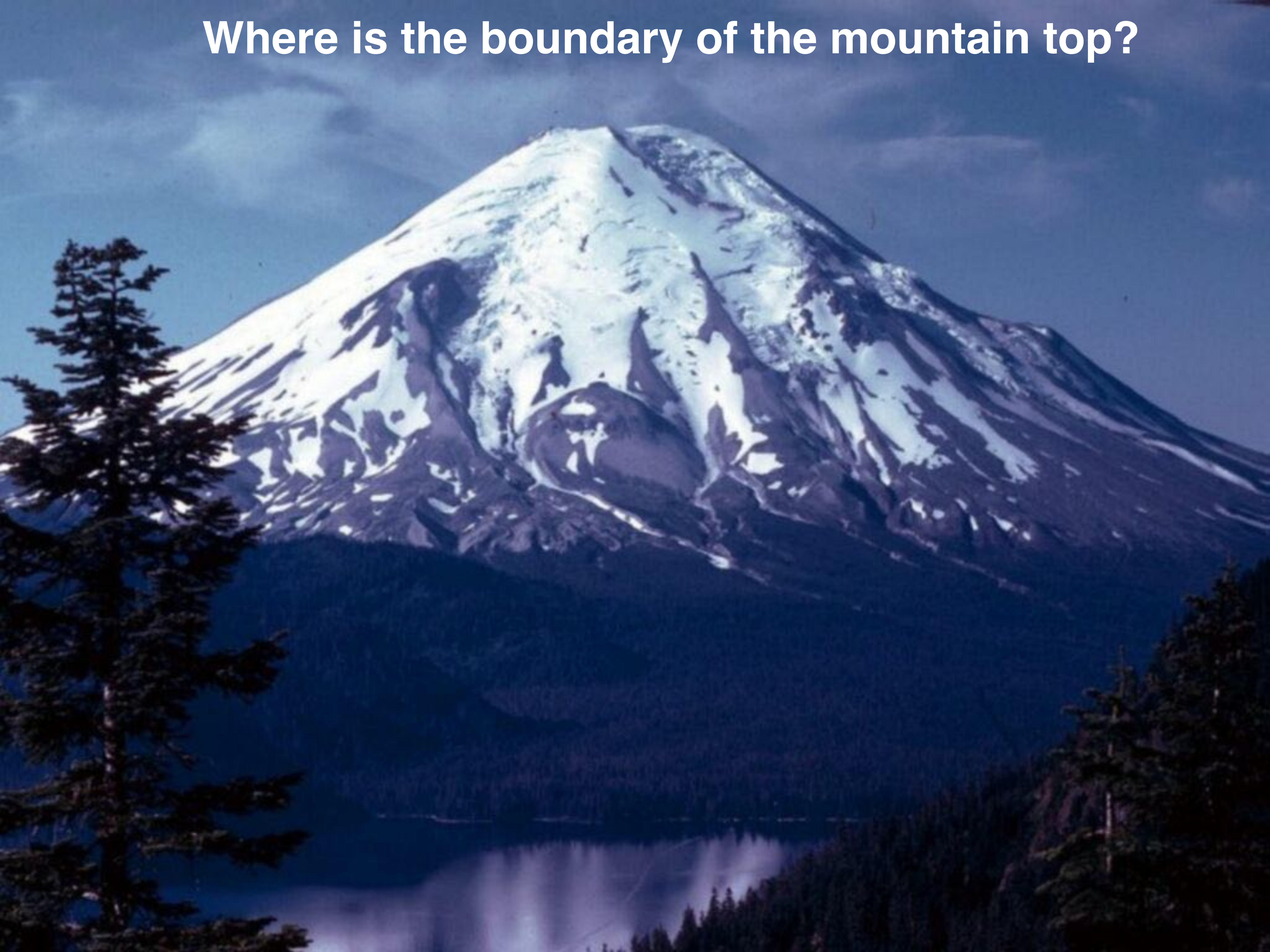
Extracting Lines

Computer Vision

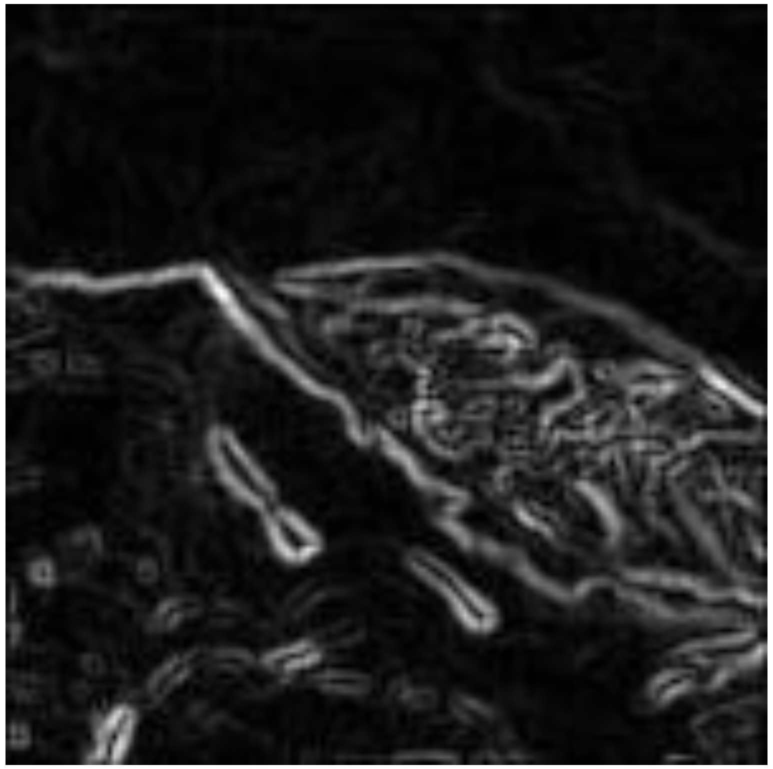
Carnegie Mellon University (Kris Kitani)

Where is the boundary of the mountain top?

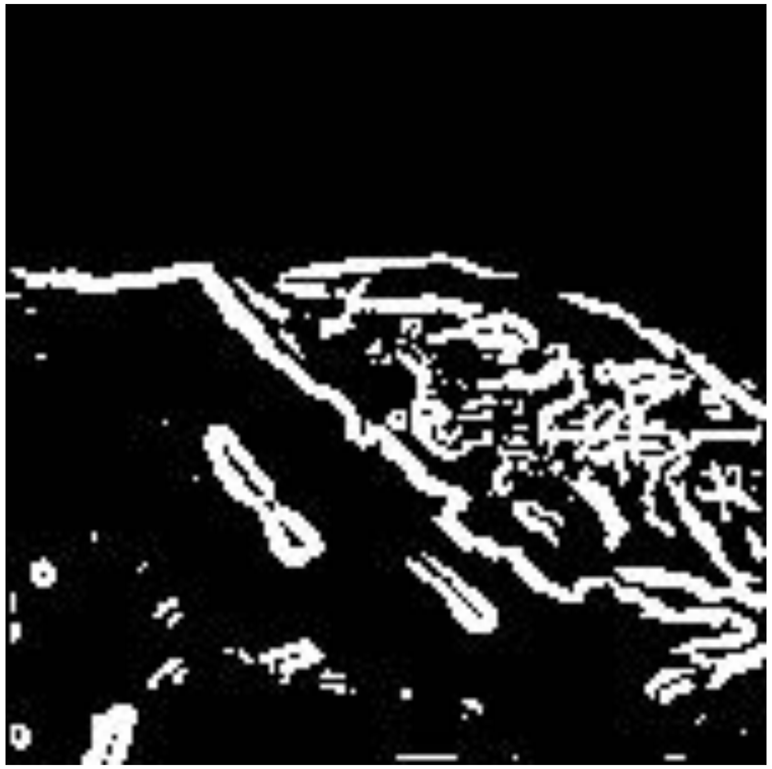


Lines are hard to find

Original image



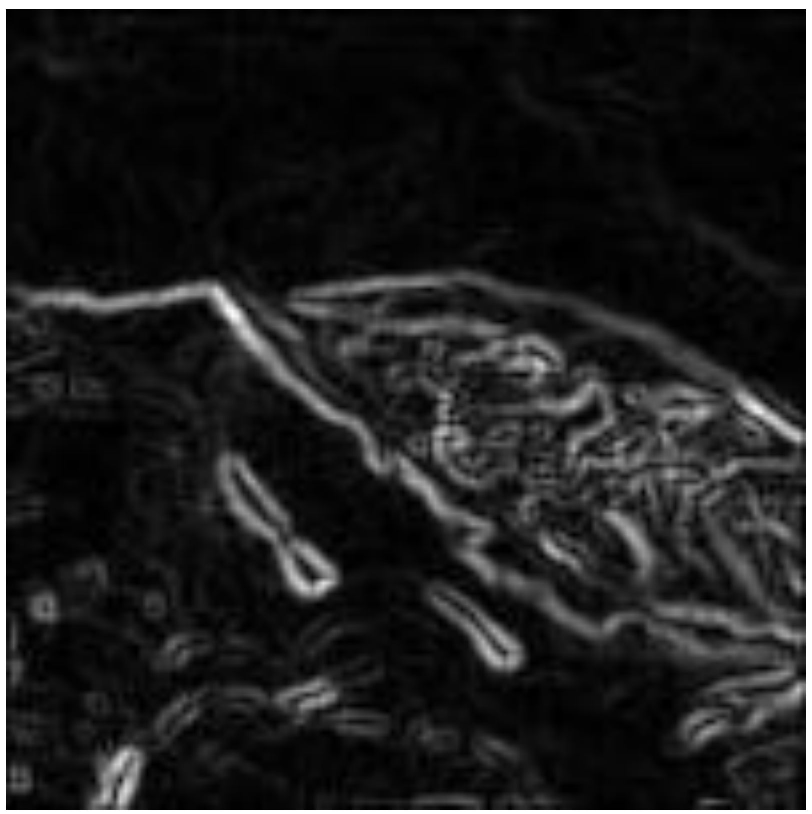
Edge detection



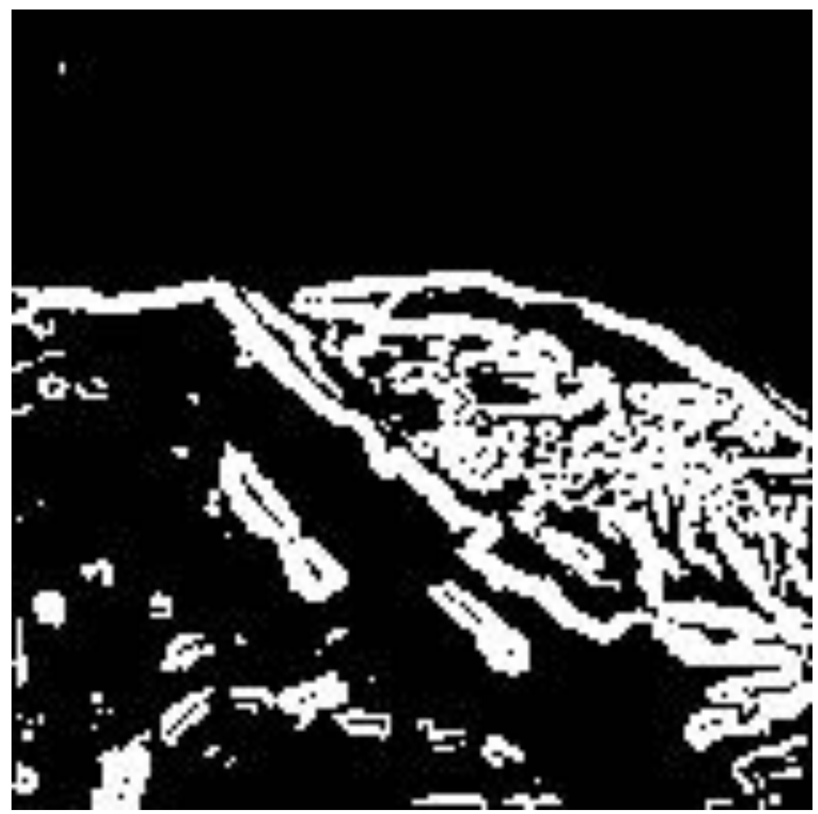
Thresholding

Noisy edge image
Incomplete boundaries

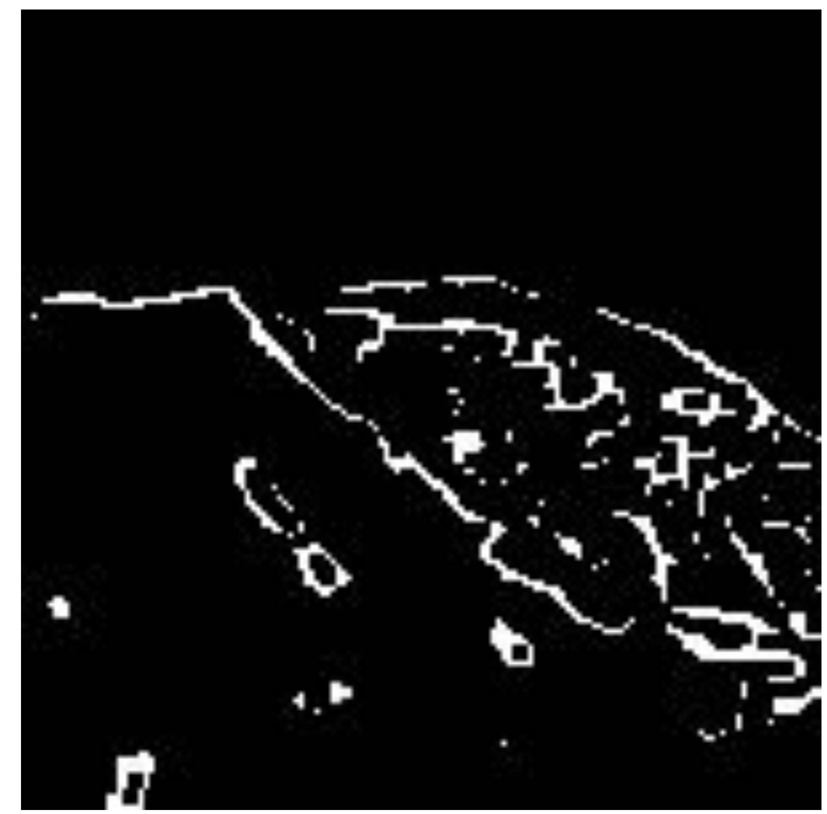
idea #1: morphology



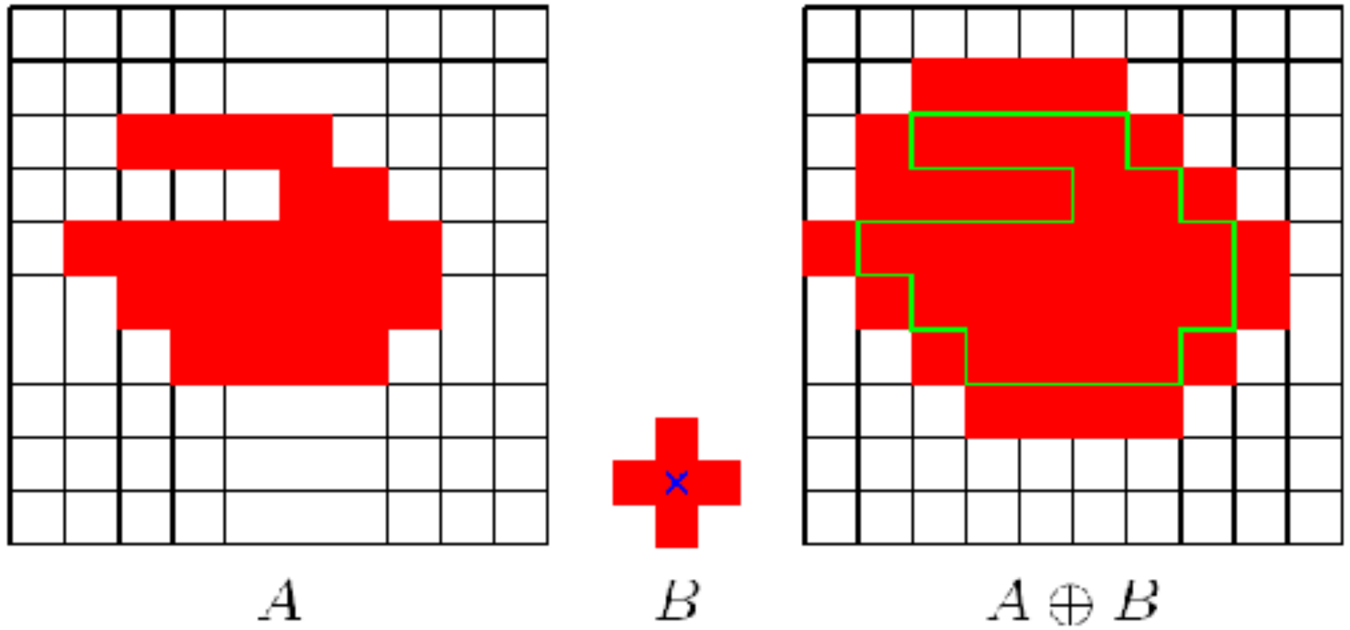
Sobel



Threshold



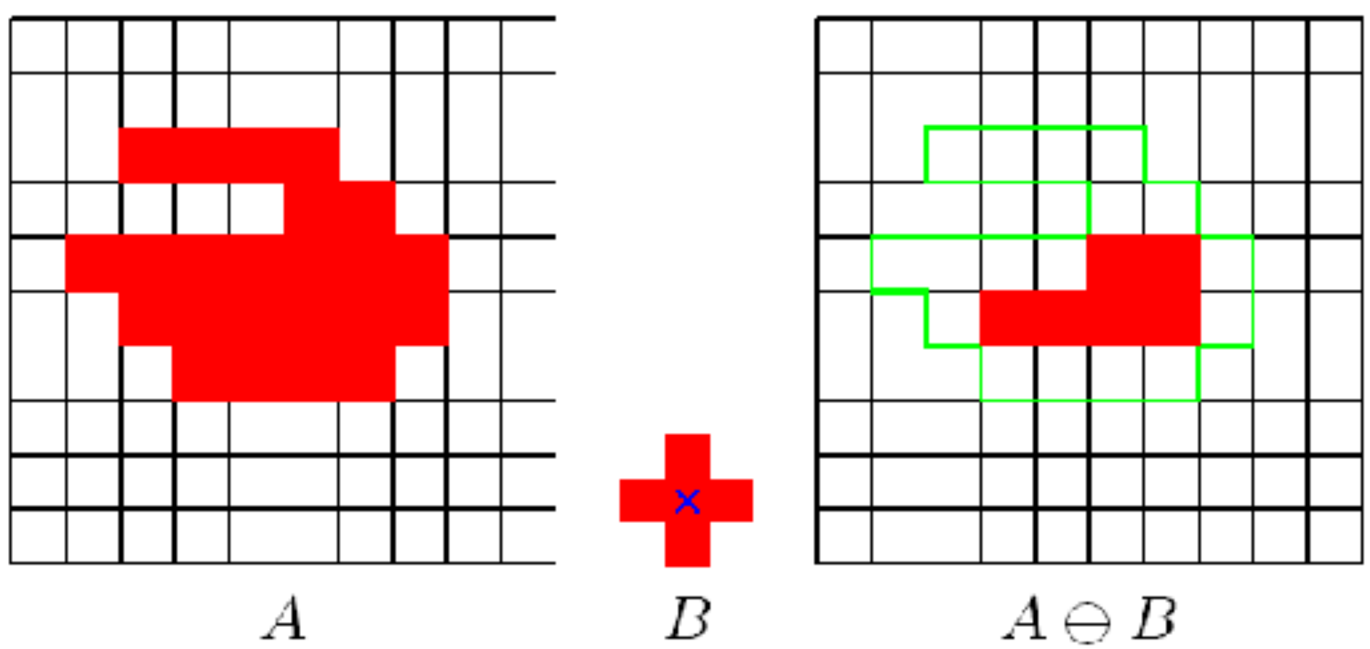
Morphology
(shrink, expand, shrink)



Dilation
operator

If filter response > 0, set to 1

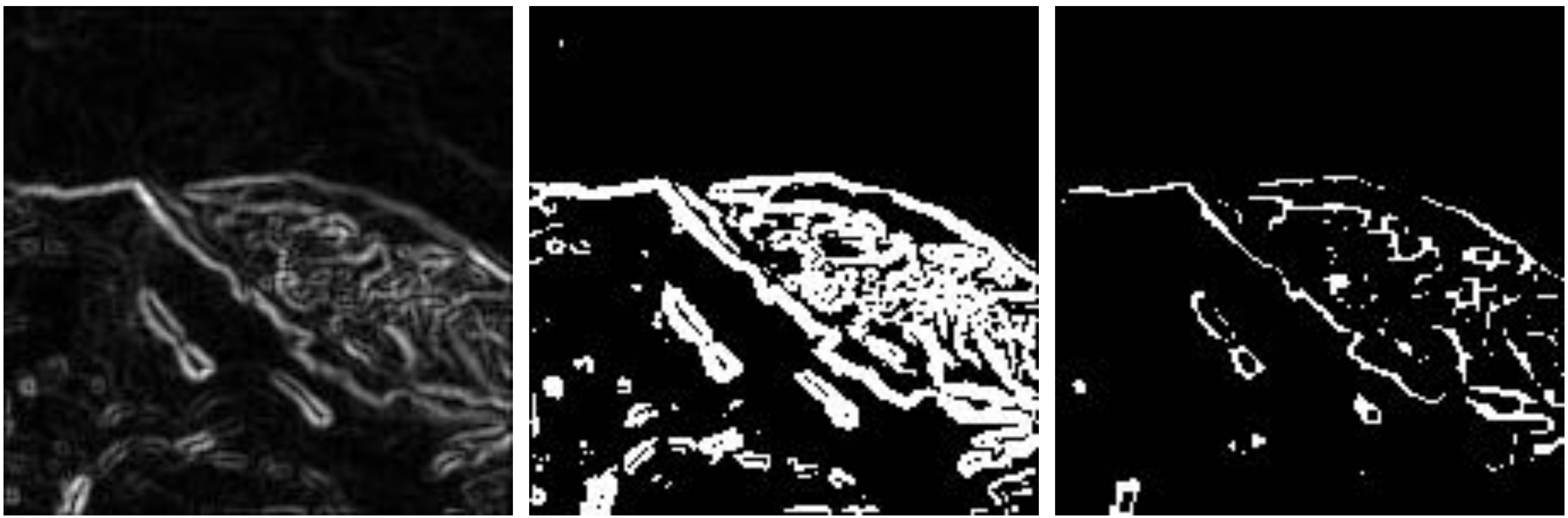
'expands'



Erosion
operator

If filter response is MAX, set to 1

'shrinks'



Sobel

Threshold

Morphology
(shrink, expand, shrink)

What are some problems with the approach?

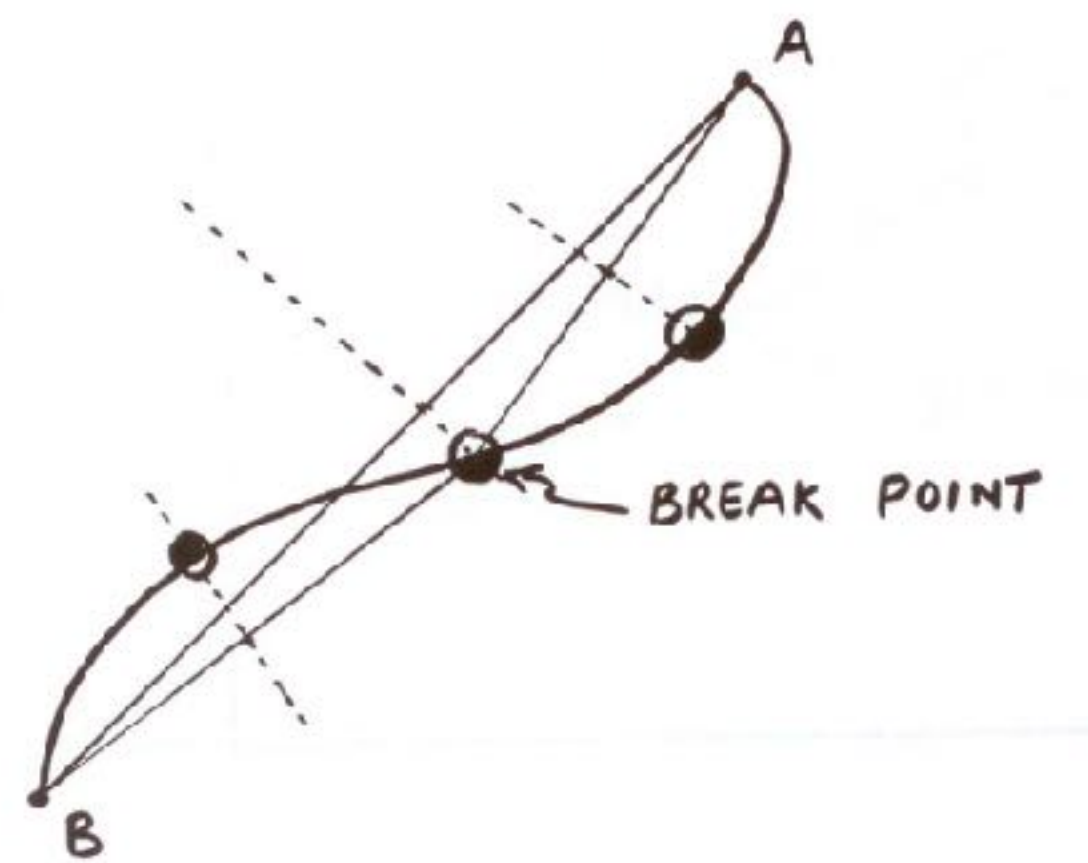
idea #2: breaking lines

Divide and Conquer:

Given: Boundary lies between points A and B

Task: Find boundary

```
Connect A and B with Line
  Find strongest edge along line bisector
  Use edge point as break point
Repeat
```



What are some problems with the approach?

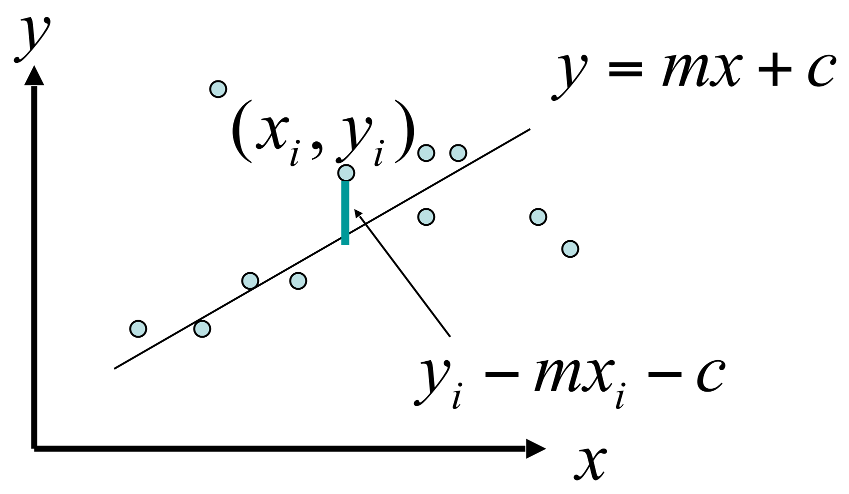
idea #3: line fitting

Given: Many (x_i, y_i) pairs

Find: Parameters (m, c)

Minimize: Average square distance:

$$E = \sum_i \frac{(y_i - mx_i - c)^2}{N}$$

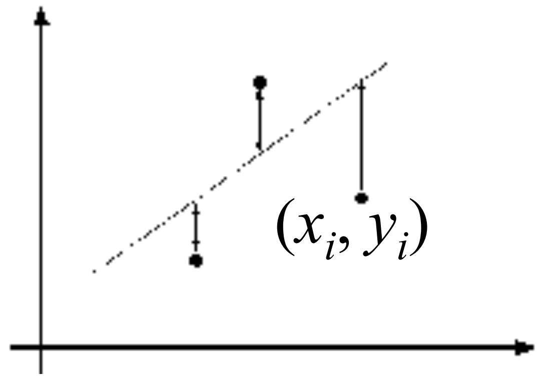


How do you minimize this objective?

(recall high school calculus)

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

Line equation: $y_i = m\,x_i + b$



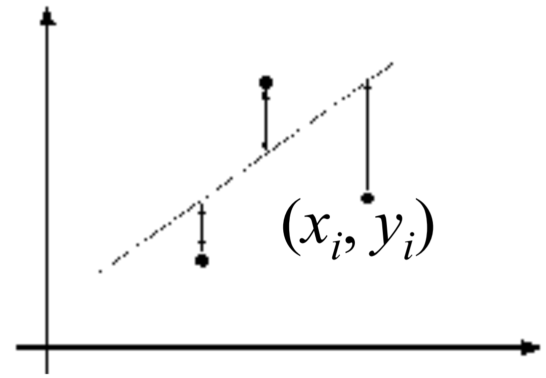
Find (m, b) to minimize $\boxed{E = \sum_{i=1}^{n} (y_i - m x_i - b)^2}$

You can stack the data as vectors

$$Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad X = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \qquad B = \begin{bmatrix} m \\ b \end{bmatrix}$$

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

Line equation: $y_i = m\, x_i + b$



Find (m, b) to minimize $\boxed{E = \sum_{i=1}^{n} (y_i - m x_i - b)^2}$

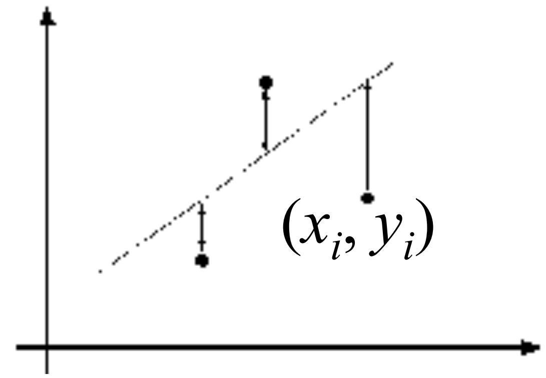
$$Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad X = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \qquad B = \begin{bmatrix} m \\ b \end{bmatrix}$$

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

Objective function in matrix form…

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

Line equation: $y_i = m\, x_i + b$



Find (m, b) to minimize $\boxed{E = \sum_{i=1}^{n} (y_i - m x_i - b)^2}$

$$Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad X = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \qquad B = \begin{bmatrix} m \\ b \end{bmatrix}$$

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

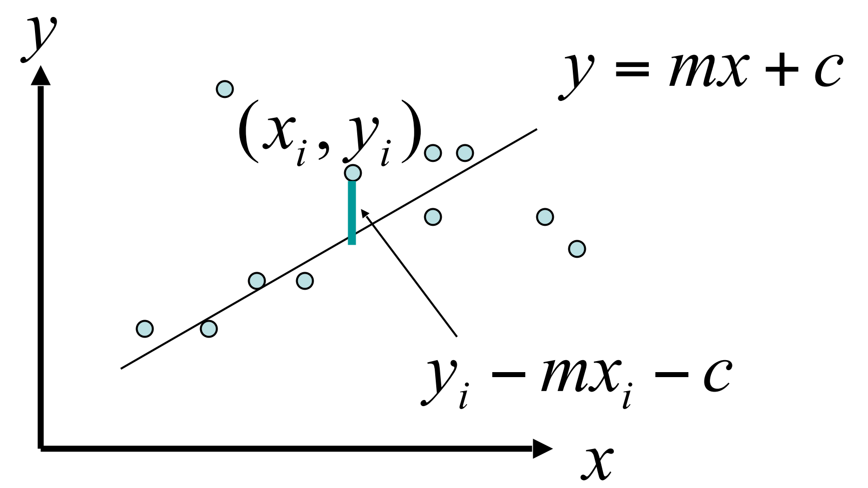
$$\frac{dE}{dB} = 2X^T XB - 2X^T Y = 0 \qquad \boxed{X^T XB = X^T Y}$$

Normal equations: least squares solution to $XB=Y$

idea #3: line fitting

Given: Many (x_i, y_i) pairs

Find: Parameters (m, c)



Minimize: Average square distance:

$$E = \sum_i \frac{(y_i - mx_i - c)^2}{N}$$

Using:

$$\frac{\partial E}{\partial m} = 0 \quad \& \quad \frac{\partial E}{\partial c} = 0$$

Note:

$$\bar{y} = \frac{\sum_i y_i}{N} \qquad \bar{x} = \frac{\sum_i x_i}{N}$$

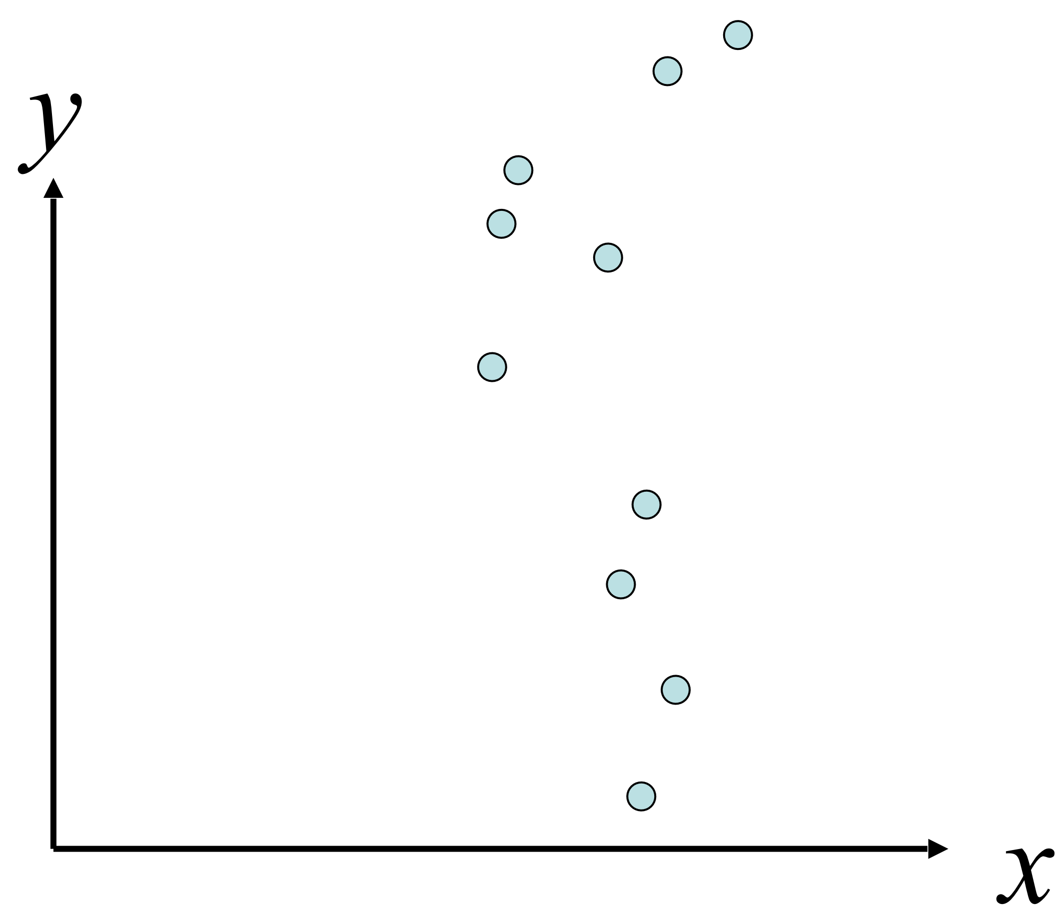
$$c = \bar{y} - m\bar{x}$$

$$m = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}$$

What are some problems with the approach?

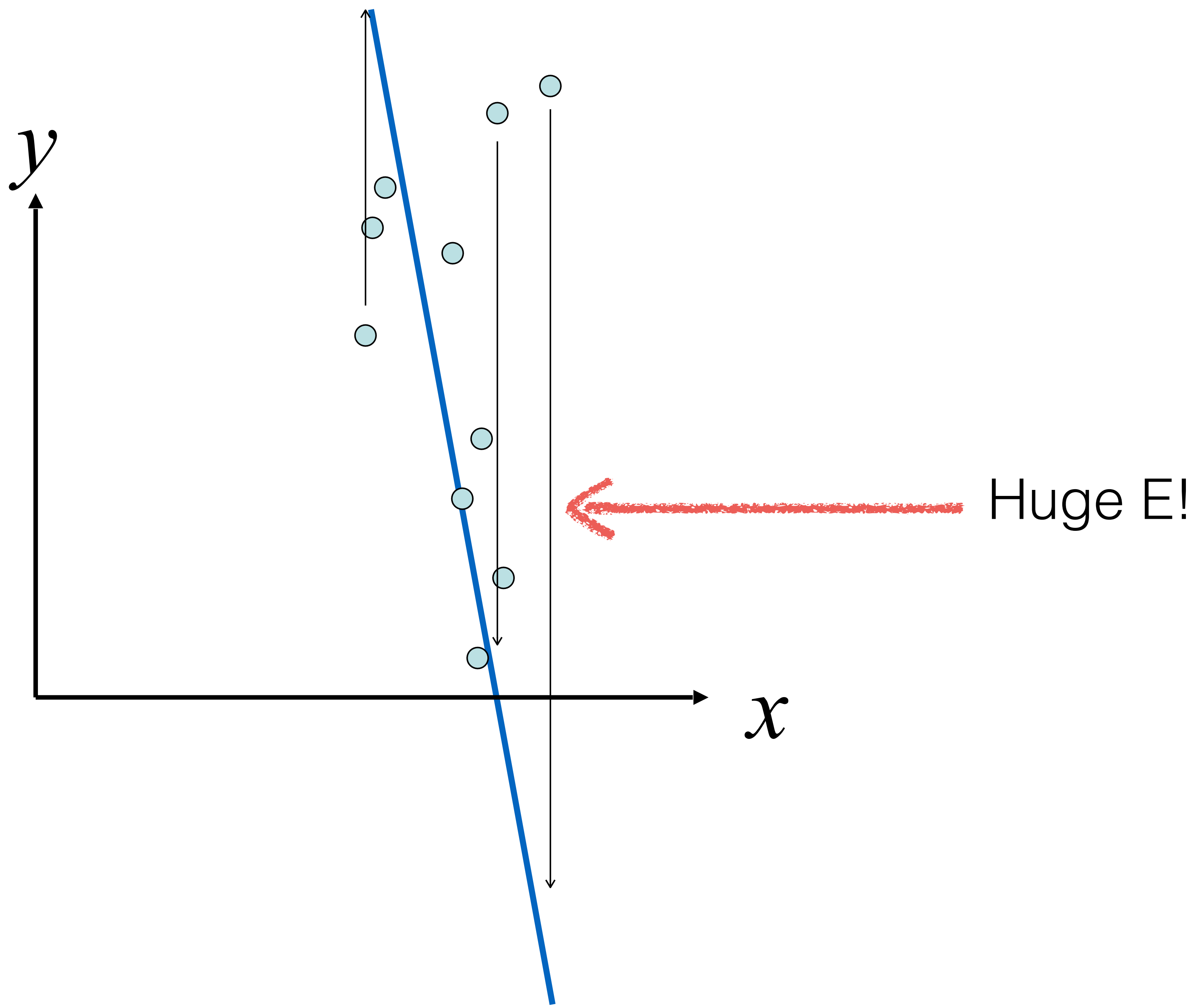
Problems with parameterizations

Where is the line that minimizes E? $E = \sum_{i=1}^{n}(y_i - mx_i - b)^2$



Problems with parameterizations

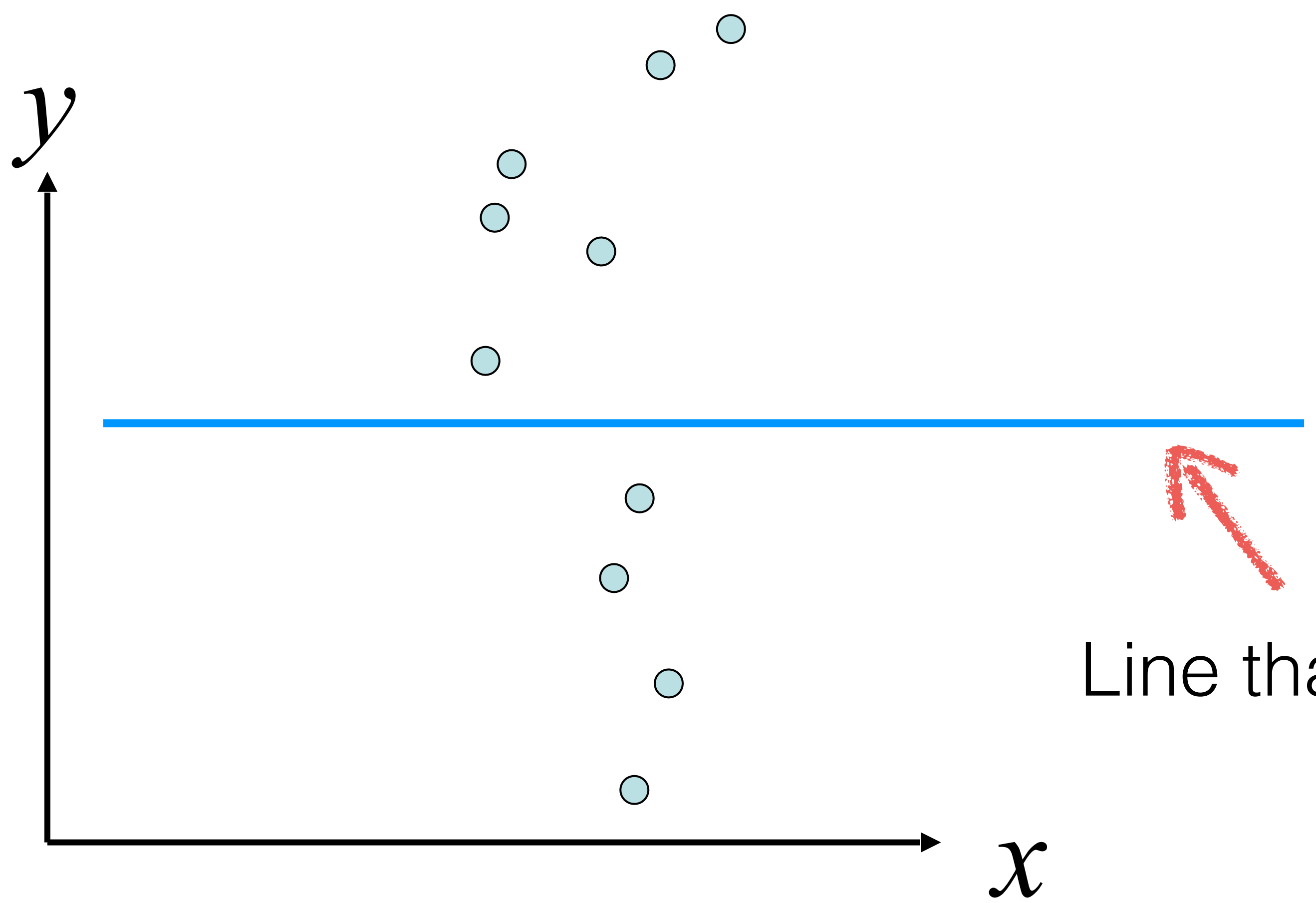
Where is the line that minimizes E? $\boxed{E = \sum_{i=1}^{n} (y_i - m x_i - b)^2}$



Problems with parameterizations

Where is the line that minimizes E?

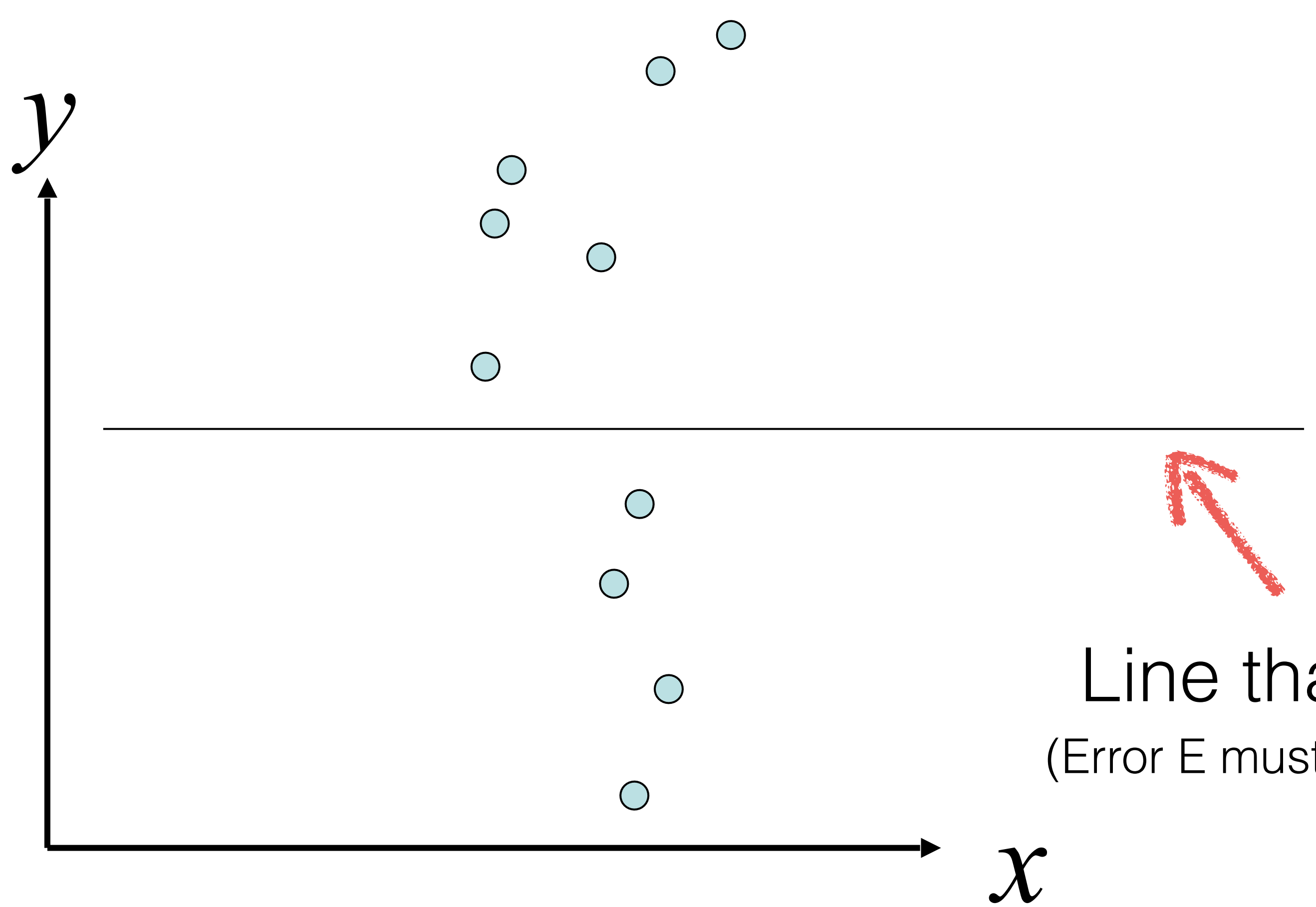
$$\boxed{E = \sum_{i=1}^{n} (y_i - mx_i - b)^2}$$



Line that minimizes E!!

Problems with parameterizations

Where is the line that minimizes E?

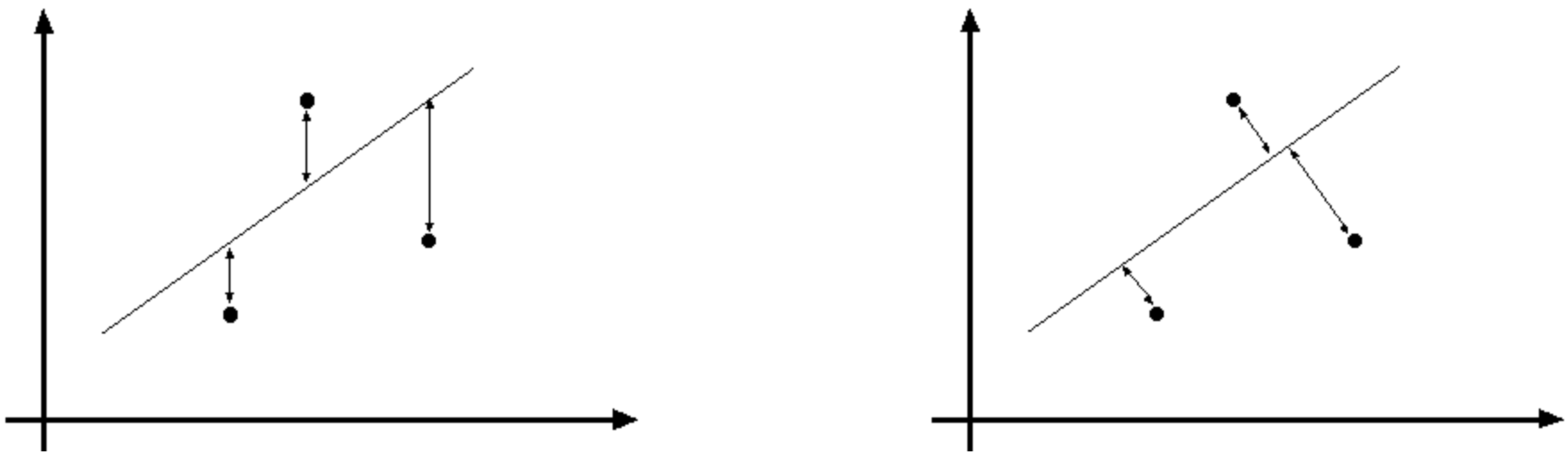


Line that minimizes E!!

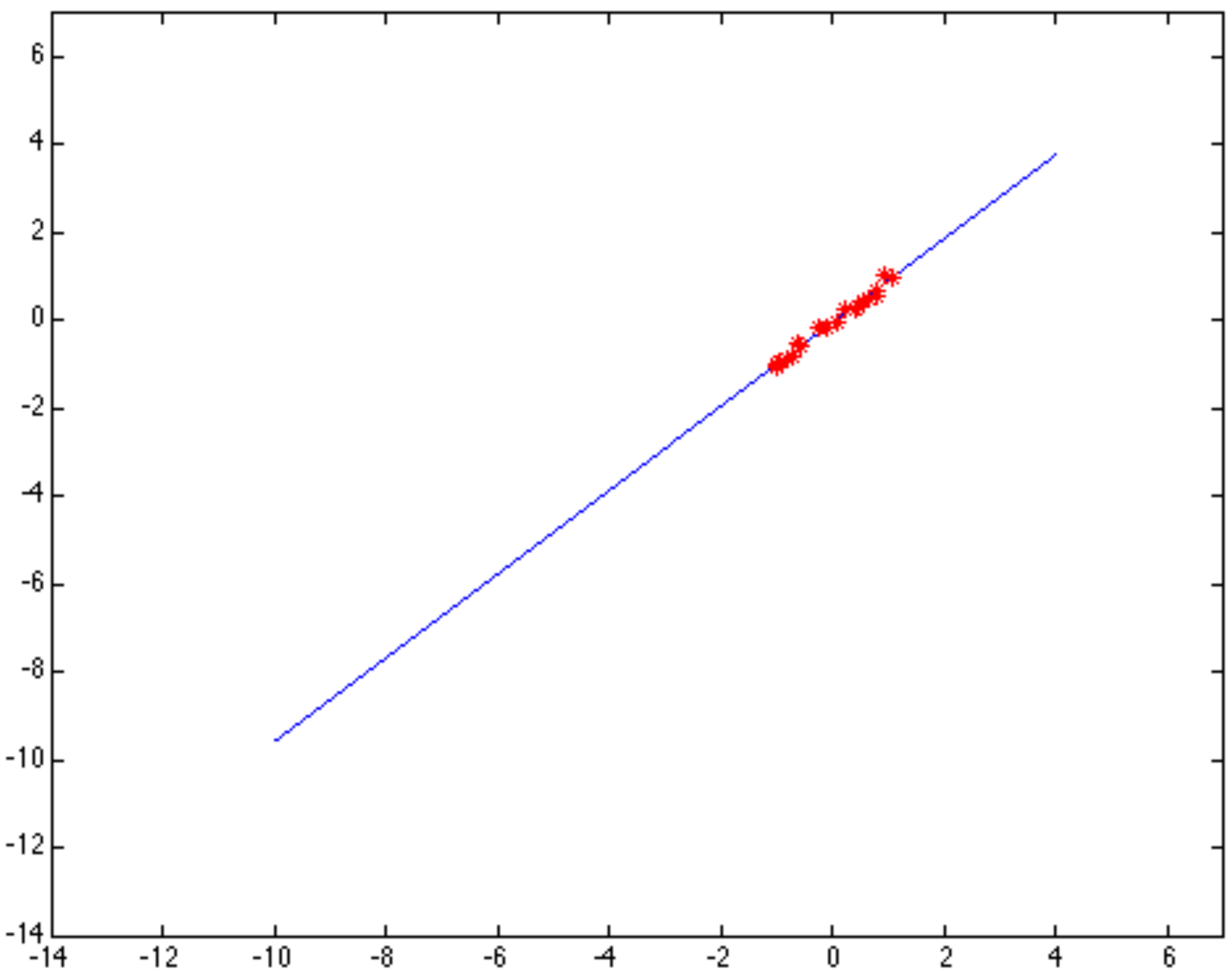
(Error E must be formulated carefully!)

Use this instead: $$E = \frac{1}{N} \sum_i (\rho - x_i \cos\theta + y_i \sin\theta)^2$$ I'll explain this later …

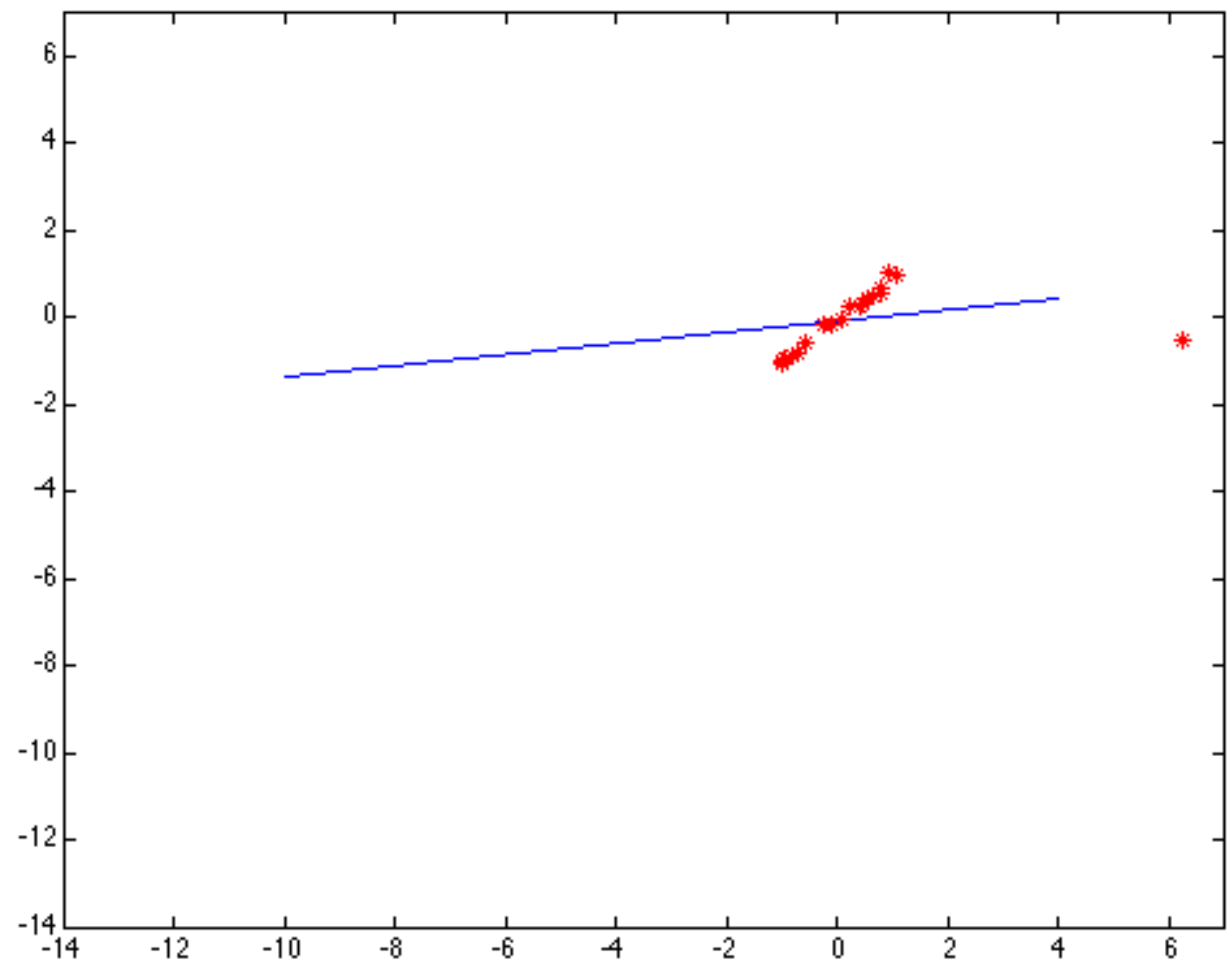
Line fitting is easily setup as a maximum likelihood problem
… but choice of model is also important



Problems with noise



Least-squares error fit

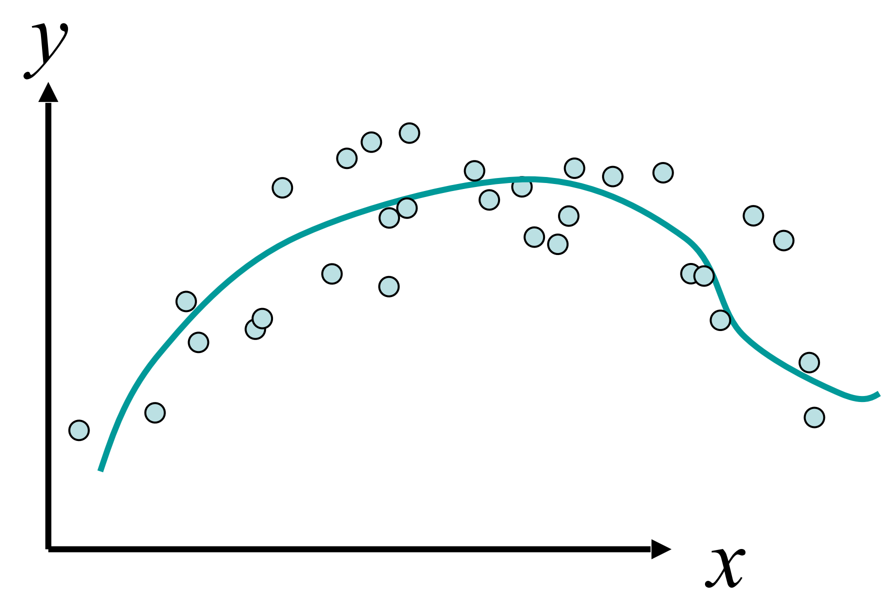


Squared error heavily penalizes outliers

idea #4: curve fitting

Find Polynomial: $y = f(x) = ax^3 + bx^2 + cx + d$

that best fits the given points (x_i, y_i)



Minimize: $$\frac{1}{N}\sum_i [y_i - (ax_i^3 + bx_i^2 + cx_i + d)]^2$$

Using: $$\frac{\partial E}{\partial a} = 0 \quad , \quad \frac{\partial E}{\partial b} = 0 \quad , \quad \frac{\partial E}{\partial c} = 0 \quad , \quad \frac{\partial E}{\partial d} = 0$$

Note: $f(x)$ is LINEAR in the parameters (a, b, c, d)

What are some problems with the approach?

Model fitting is difficult because…

- **Extraneous data:** clutter or multiple models
  - We do not know what is part of the model?
  - Can we pull out models with a few parts from much larger amounts of background clutter?
- **Missing data:** only some parts of model are present
- **Noise**

- Cost:
  - It is not feasible to check all combinations of features by fitting a model to each possible subset

So what can we do?