Model Fitting

Fitting data to a model

- Practical computer vision involves lots of fitting of data to models
- Tasks
  - Fit image features to projection of hypothetical shape
    - Fit lines, ellipses, etc.
  - Determine the camera projection matrix
  - Determine motion of objects in video
  - Determine 3-D coordinates of world points
  - … many others
- All these involve fitting data obtained from images to models
- Simplest example of model fitting problem
  - Linear regression

Models

- Have a certain model structure
  - E.g., “linear” “quadratic” “trigonometric” “Gaussian”
- Models have specifiable parameters
- e.g.
- Model Structure Data Parameters

Model	Structure	Data	Parameters
Straight line:	$a x + by + c = 0$	(x_i, y_i)	(a,b,c)
Polynomial:	$y = c_0 + c_1 x + \ldots + c_n x^n$	(x_i, y_i)	$(c_0, c_1, \ldots, c_n)$
Trigonometric			
Gaussian			
“Kernel”			

- More examples on the board
  - higher dimensions
  - Matrix entries …
- In all these we need to find the models and their parameters that best explain the data
- General part of “vision as inference”
- Approaches
  - Based on fitting: today
  - Based on sampling model-space: e.g., Hough transform
  - Combination

Relationships among Variables: Interpretations

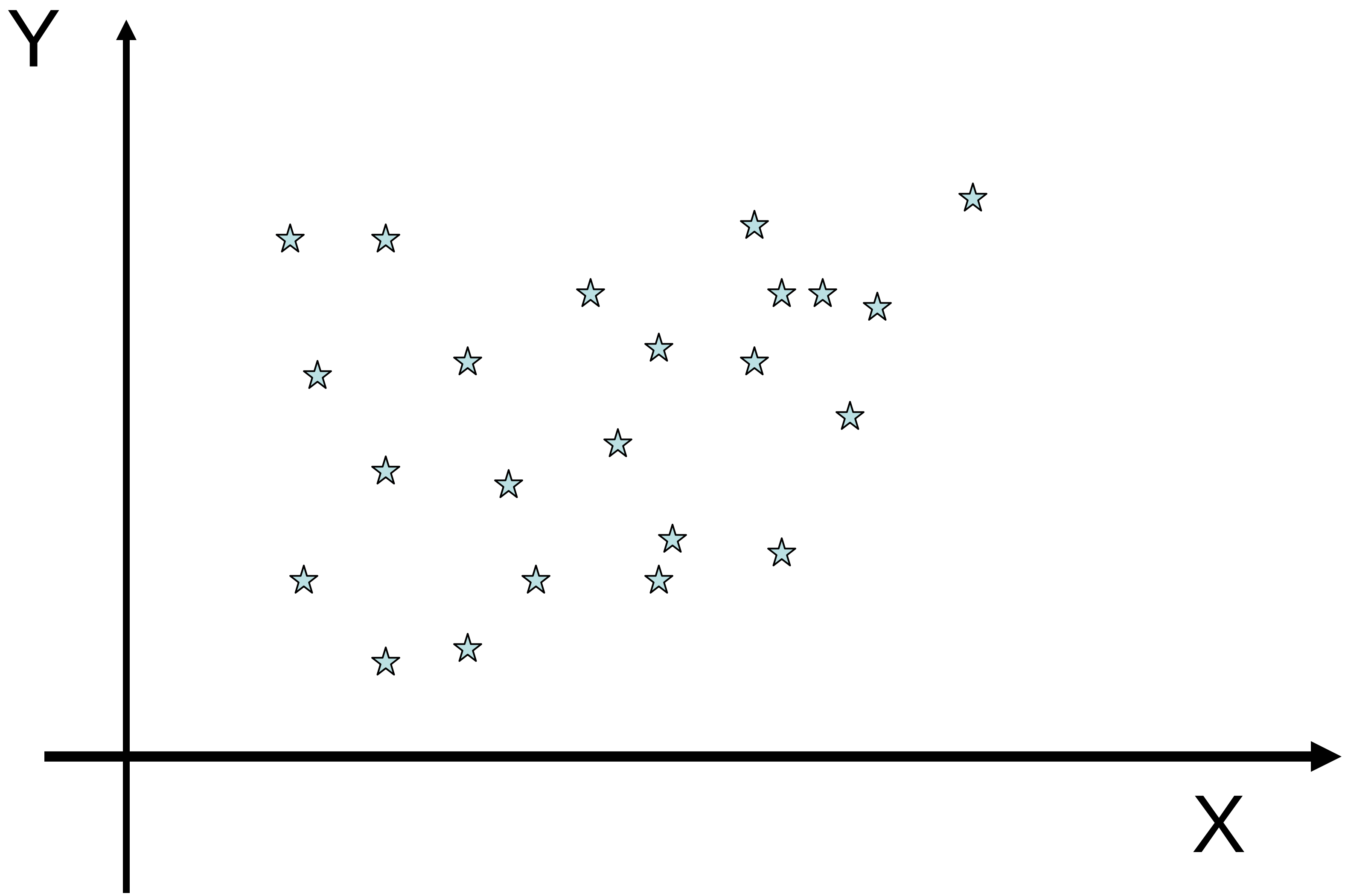
One variable is used to “explain” another variable

X Variable
Independent Variable
Explaining Variable
Exogenous Variable
Predictor Variable

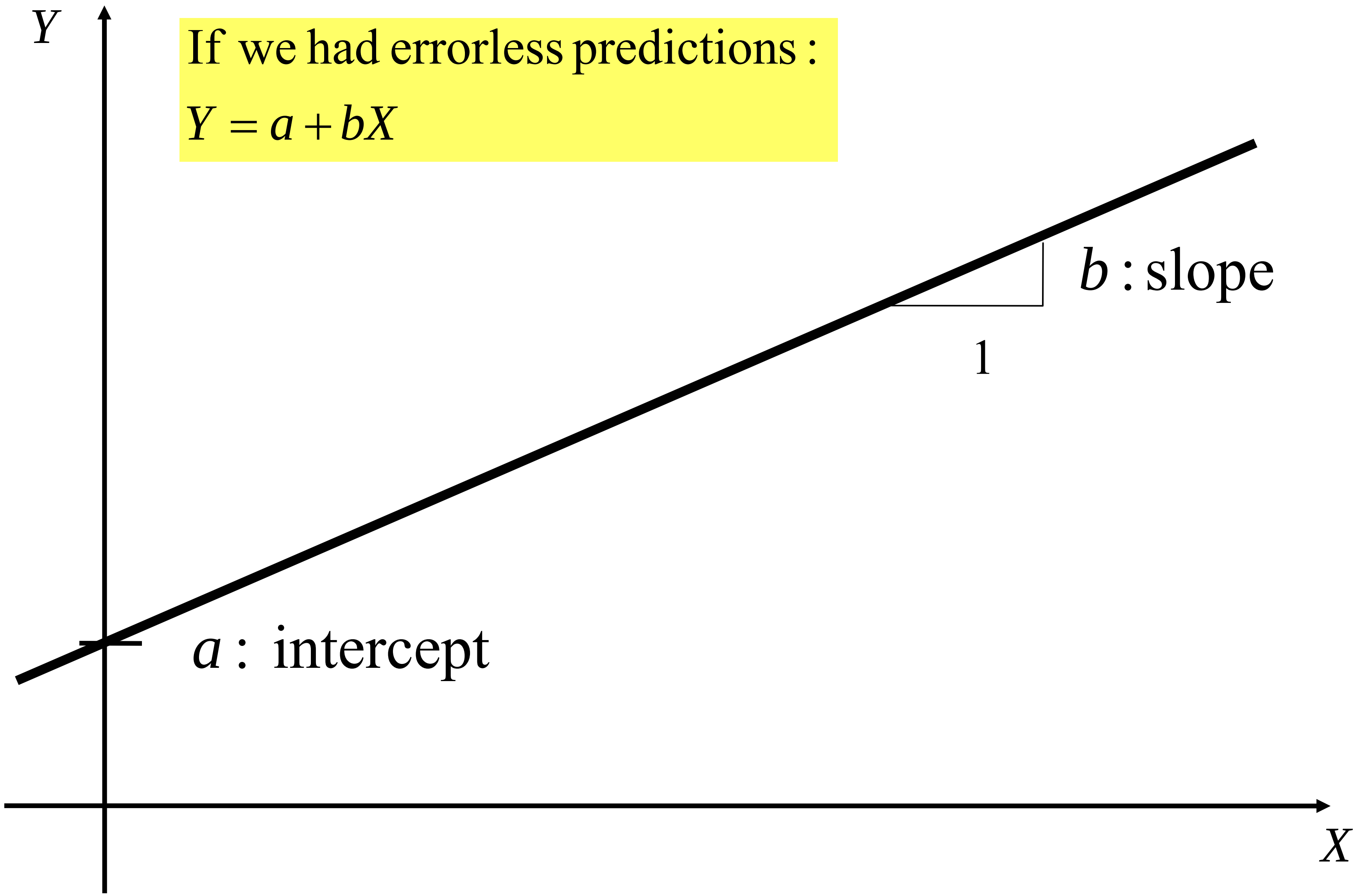
→

Y Variable
Dependent Variable
Response Variable
Endogenous Variable
Criterion Variable

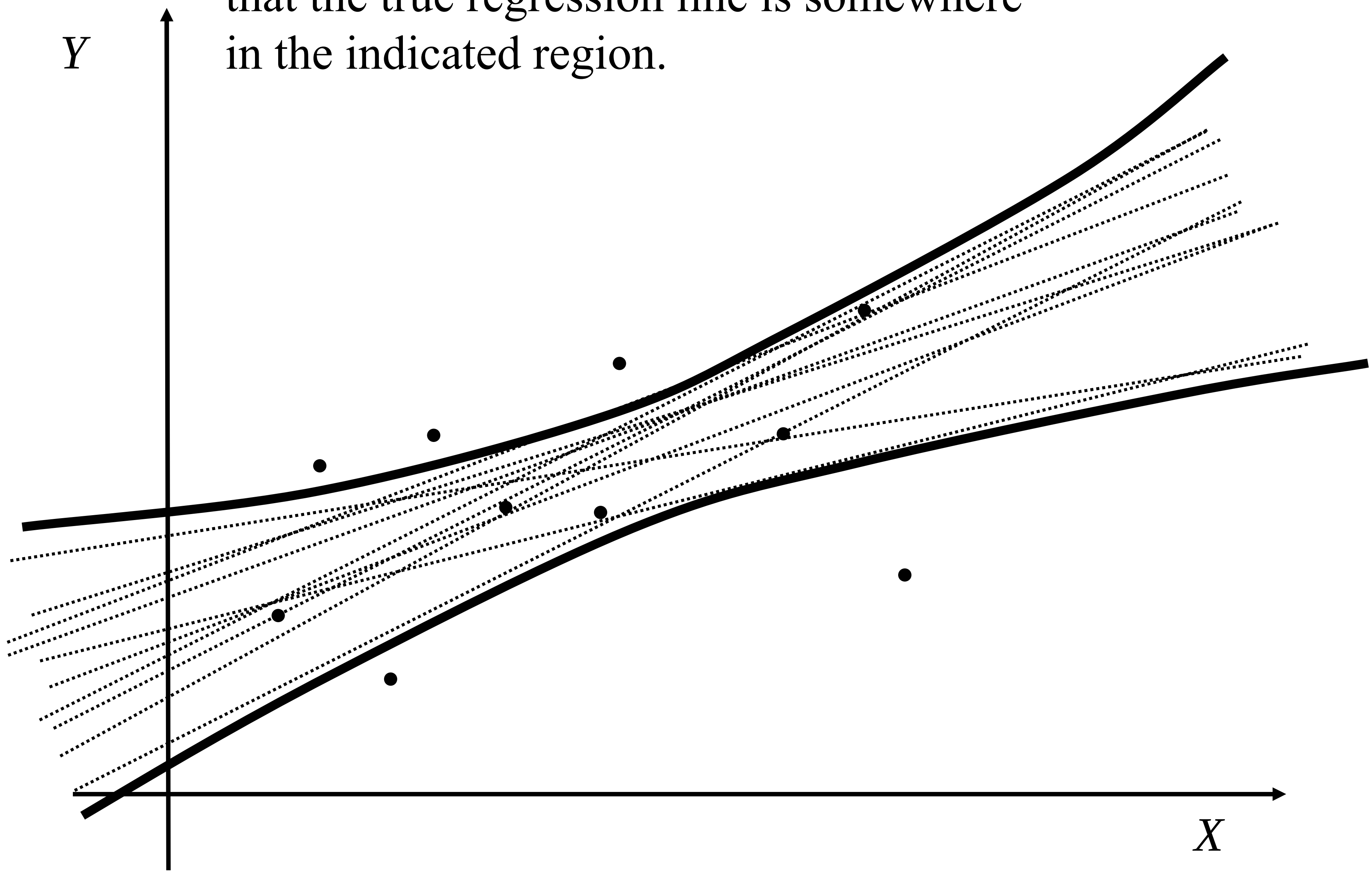
Scatter Plots



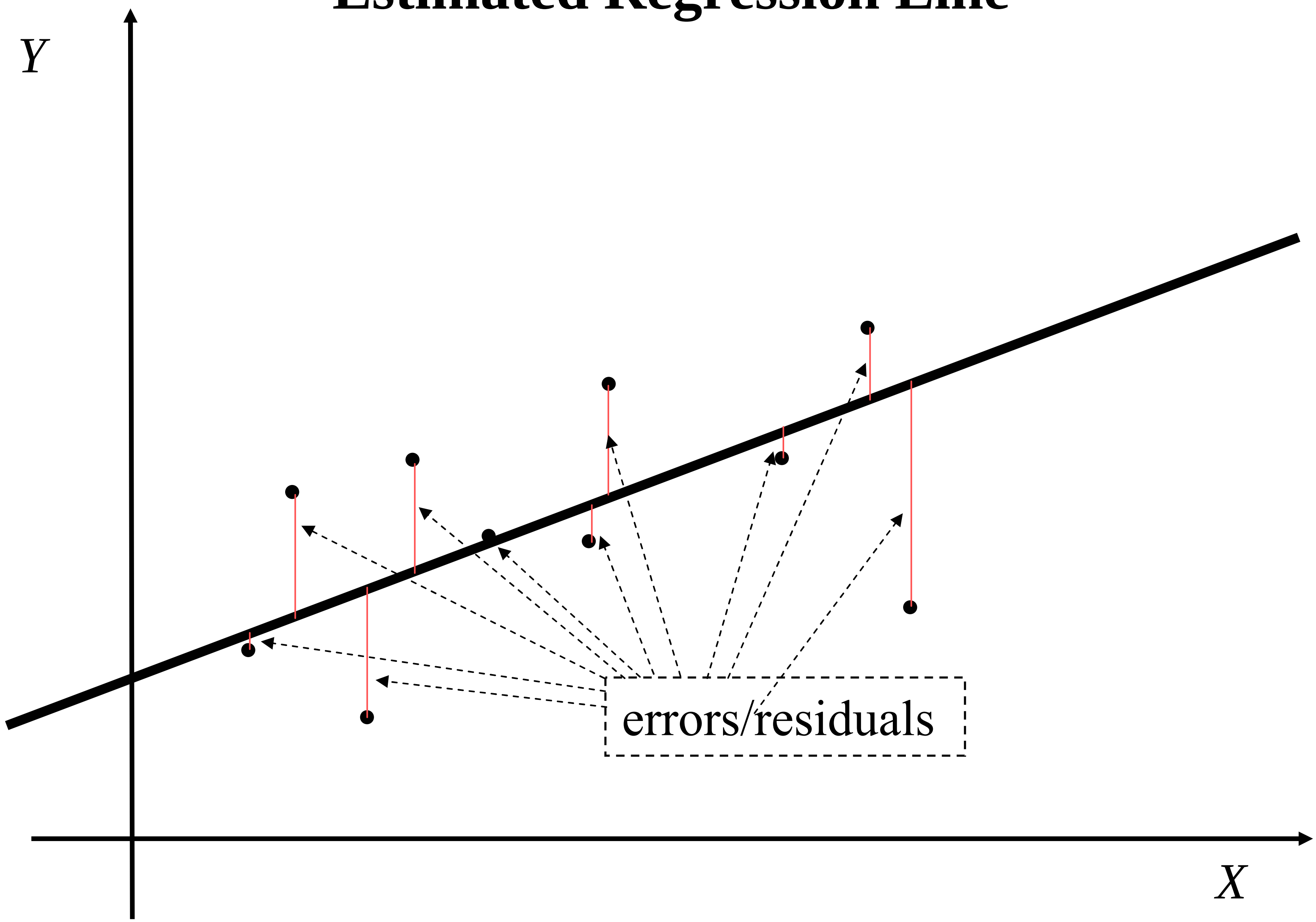
Simple Least-Squares Regression



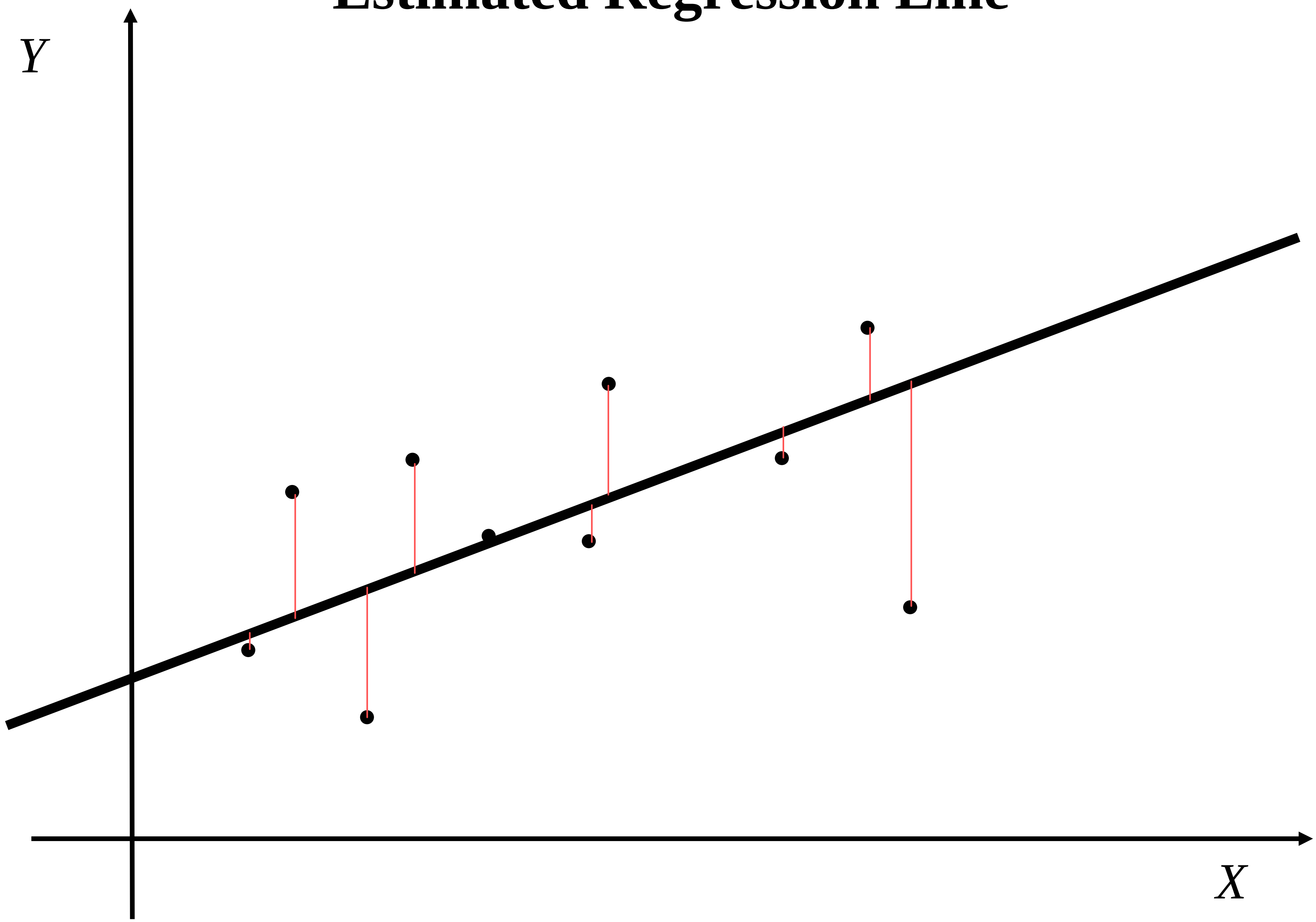
We will end up being reasonably confident that the true regression line is somewhere in the indicated region.



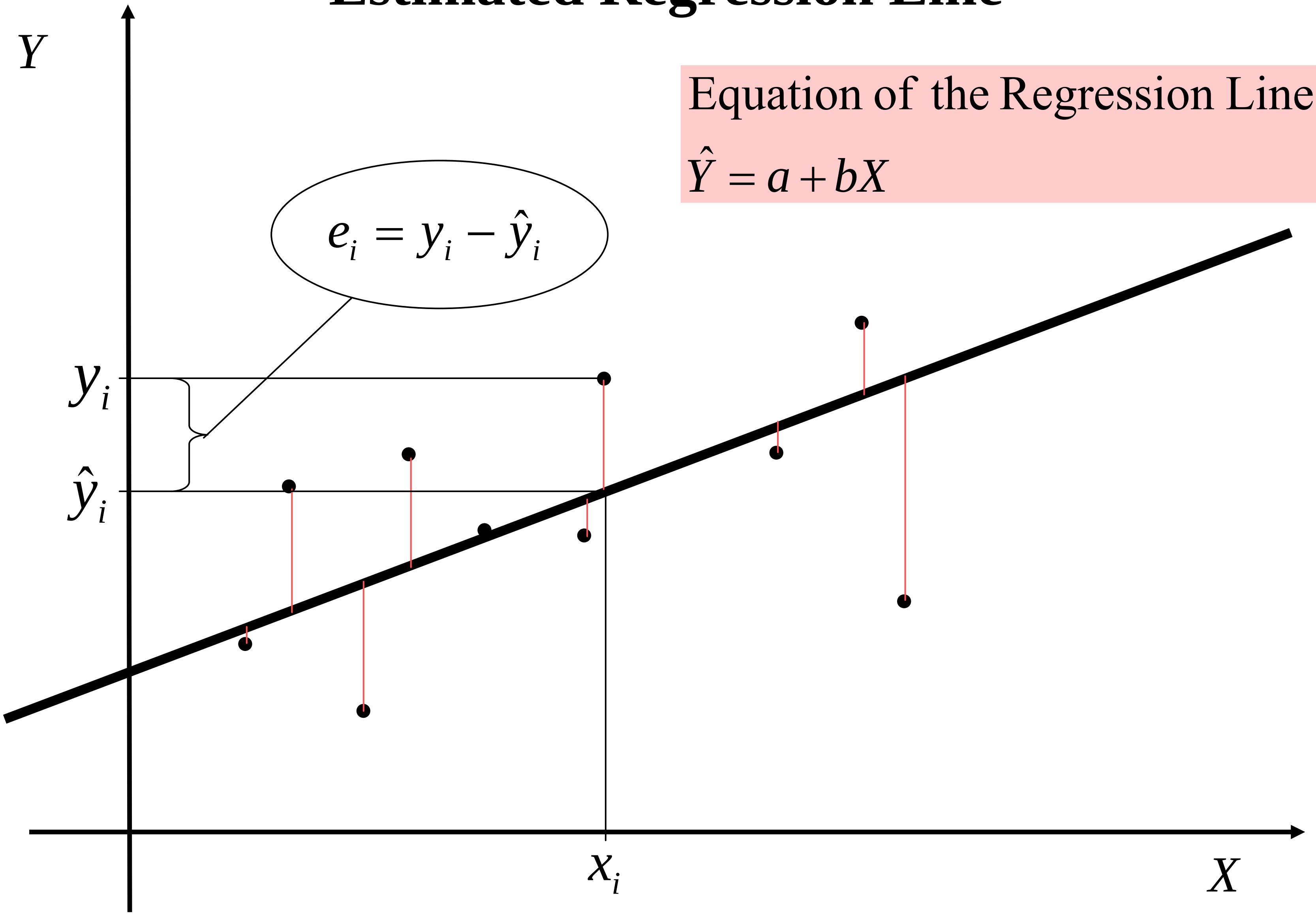
Estimated Regression Line



Estimated Regression Line



Estimated Regression Line

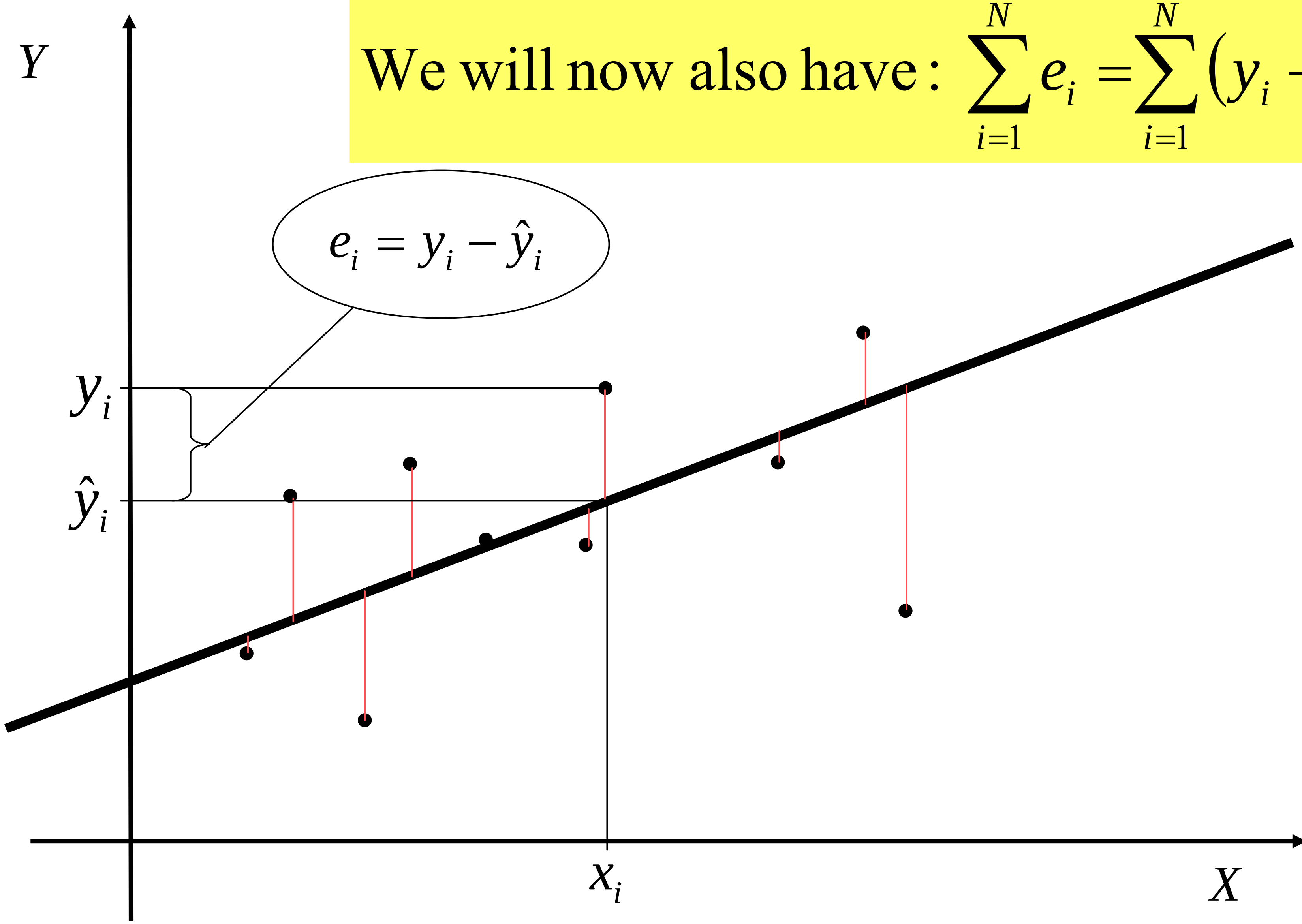


Equation of the Regression Line:

$$\hat{Y} = a + bX$$

$$\text{Remember}: \sum_{i=1}^{N}\left(y_i - \bar{y}\right) = 0$$

$$\text{We will now also have}: \sum_{i=1}^{N} e_i = \sum_{i=1}^{N}\left(y_i - \hat{y}_i\right) = 0$$



How do we find a and b?

In Least-Squares Regression:

Find a, b to minimize the sum of squared errors/residuals

$$\sum_{i=1}^{N} (e_i)^2 = \sum_{i=1}^{N} \left(y_i - [bx_i + a]\right)^2$$

In Least-Squares Regression:

$$b = \frac{\sum_{i=1}^{N}\left(X_i - \overline{X}\right)\left(Y_i - \overline{Y}\right)}{\sum_{i=1}^{N}\left(X_i - \overline{X}\right)^2}, \qquad a = \overline{Y} - b\overline{X}$$

Computational Formula

$$b = \frac{N\sum_{i=1}^{N} X_i Y_i - \left(\sum_{i=1}^{N} X_i\right)\left(\sum_{i=1}^{N} Y_i\right)}{N\sum_{i=1}^{N} X_i^2 - \left(\sum_{i=1}^{N} X_i\right)^2}$$

Models

- Have a certain model structure
  - E.g., “ straight line” “parabolic” “trigonometric” “Gaussian”
- Models have specifiable parameters
- e.g.
- Model Structure Data Parameters

Model	Structure	Data	Parameters
Straight line:	$a\,x + by + c = 0$	(x_i, y_i)	(a,b,c)
Polynomial:	$y=c_0+c_1\,x+ \ldots +c_n\,x^n$	(x_i, y_i)	$(c_0, c_1, \ldots, c_n)$
Trig.:	$y=c_0+c_1 \sin x+ \ldots +c_n \sin nx$	(x_i, y_i)	$(c_0, c_1, \ldots, c_n)$
Gaussian	$y=c_0 \exp(-(x-\mu)^2/\sigma^2)$	(x_i, y_i)	(c_0, σ, μ)
“Kernel/RBF”	$y=\sum_i a_i\, k(x,x_i)$	(x_i, y_i)	(a_i) and (x_i)

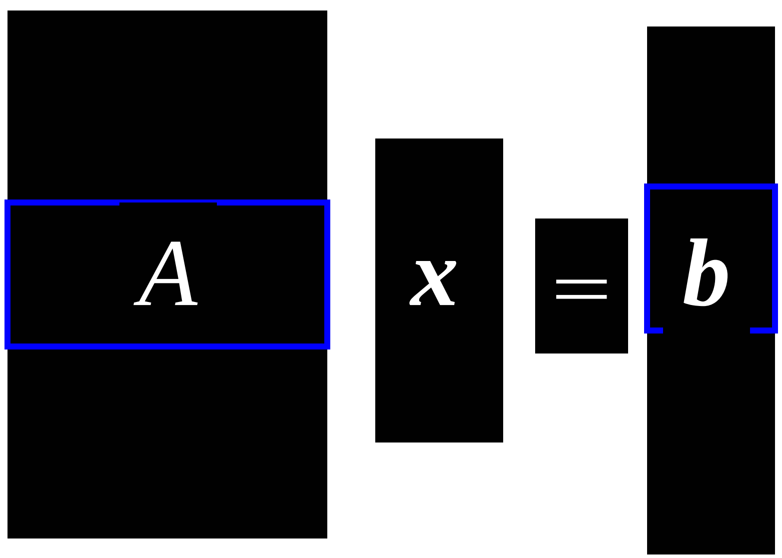
Linear models

- All models considered are “linear”
- (does not mean straight lines)
- Means that we can separate the “structure” and “parameters” as a matrix-vector product
- “structure” forms a matrix and “parameters” a vector
- Goal of model fitting: find the parameters
- Question: Is the Gaussian model a linear one?

Linear Systems

Square system:

- unique solution
- Gaussian elimination

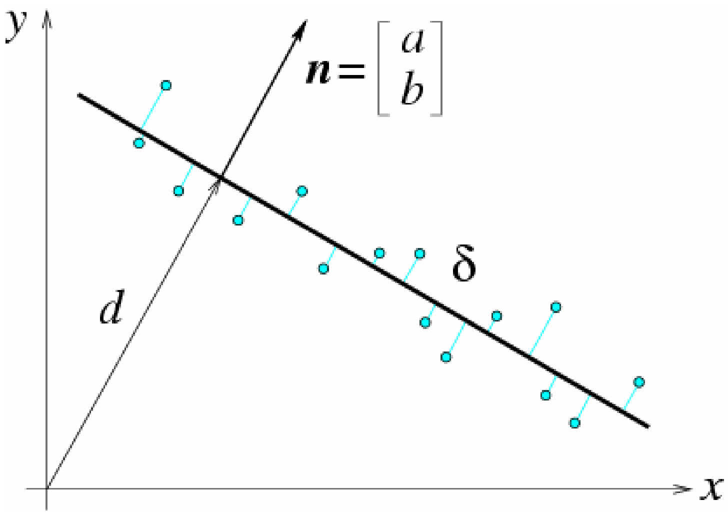


Rectangular system ??

- underconstrained: infinity of solutions
- overconstrained: no solution

Minimize $|Ax-b|^2$

Example: Line Fitting with errors in both coordinates



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

Minimize

$$\frac{\partial E}{\partial d} = 0 \Longrightarrow d = \sum_{i=1}^{n} ax_i + by_i = a\bar{x} + b\bar{y}$$

- Minimize *E* with respect to *a*,*b*:

$$E = \sum_{i=1}^{n} [a(x_i - \bar{x}) + b(y_i - \bar{y})]^2 = |\mathcal{U}\boldsymbol{n}|^2$$

$$\mathcal{U} = \begin{pmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \dots & \dots \\ x_n - \bar{x} & y_n - \bar{y} \end{pmatrix}$$

Least Squares for more complex models

- Number of equations and unknowns may not match
- Data may have noise
- Look for solution by minimizing some cost function
- Simplest and most intuitive cost function: $||\mathbf{Ax} - \mathbf{b}||$
- Define for each data point x_i a residual r_i
- *Minimize* $\sum_i r_i \, r_i$ with respect to x_1
- $r_i \, r_i = \sum_j (A_{ij} x_j - b_i). \sum_k (A_{ik} x_k - b_i)$

$$\frac{\partial}{\partial x_l}(A_{ij}x_j - b_i)\bullet(A_{ik}x_k - b_i) = 0$$

$$(A_{ij}\delta_{jl})\bullet(A_{ik}x_k - b_i) + (A_{ij}x_j - b_i)\bullet(A_{ik}\delta_{kl}) = 0$$

$$A_{il}\bullet(A_{ik}x_k - b_i) + (A_{ij}x_j - b_i)\bullet A_{il} = 2\left(A_{il}A_{ik}x_k - A_{il}b_i\right) = 0$$

$$A_{il}A_{ik}x_k = A_{il}b_i$$

SVD and Pseudo-Inverse

- $\mathbf{Ax}=\mathbf{b}$ **A** is $m \times n$, **x** is $n \times 1$ and **b** is $m \times 1$.
- $\mathbf{A}=\mathbf{USV}^t$ where **U** is m×m, **S** is m×n and **V** is $n \times n$
- $\mathbf{USV}^t\,\mathbf{x}=\mathbf{b}$. So $\mathbf{SV}^t\,\mathbf{x}=\mathbf{U}^t\mathbf{b}$
- If **A** has rank *r*, then *r* singular values are significant

  $\mathbf{V}^t\mathbf{x}= \mathrm{diag}(\sigma_1^{-1},\ldots,\sigma_r^{-1},0,\ldots,0)\mathbf{U}^t\mathbf{b}$

  $\mathbf{x}= \mathbf{V}\mathrm{diag}(\sigma_1^{-1},\ldots,\sigma_r^{-1},0,\ldots,0)\mathbf{U}^t\mathbf{b}$

$$\mathbf{x}_r = \sum_{i=1}^{r} \frac{\mathbf{u}_i^t\mathbf{b}}{\sigma_i}\mathbf{v}_i \qquad \sigma_r > \varepsilon, \quad \sigma_{r+1} \le \varepsilon$$

- Pseudoinverse $\mathbf{A}^+=\mathbf{V}\,\mathrm{diag}(\sigma_1^{-1},\ldots,\sigma_r^{-1},0,\ldots,0)\,\mathbf{U}^t$
  - $\mathbf{A}^+$ is a $n \times m$ matrix.
  - If rank (**A**) $=n$ then $\mathbf{A}^+=(\mathbf{A}^t\mathbf{A})^{-1}\mathbf{A}$
  - If **A** is square $\mathbf{A}^+=\mathbf{A}^{-1}$

Well posed problems

- Hadamard postulated that for a problem to be "well posed"

1. Solution must exist
2. It must be unique
3. Small changes to input data should cause small changes to solution

- Many problems in science and computer vision result in "ill-posed" problems.
  - Numerically it is common to have condition 3 violated.
  - Recall from the SVD $\mathbf{x} = \sum_{i=1}^{n} \frac{\mathbf{u}_i^t \mathbf{b}}{\sigma_i} \mathbf{v}_i \qquad \sigma_r > \varepsilon, \quad \sigma_{r+1} \le \varepsilon$

  If the σ are close to zero small changes in the "data" vector **b** cause big changes in **x**.

- Converting ill-posed problem to well-posed one is called *regularization*.

Regularization

- Pseudoinverse provides one means of regularization
- Another is to solve $(\mathbf{A}+\varepsilon\mathbf{I})\mathbf{x}=\mathbf{b}$

$$\mathbf{x} = \sum_{i=1}^{n} \frac{\sigma_i}{\varepsilon + \sigma_i^2} (\mathbf{u}_i^t \mathbf{b}) \mathbf{v}_i$$

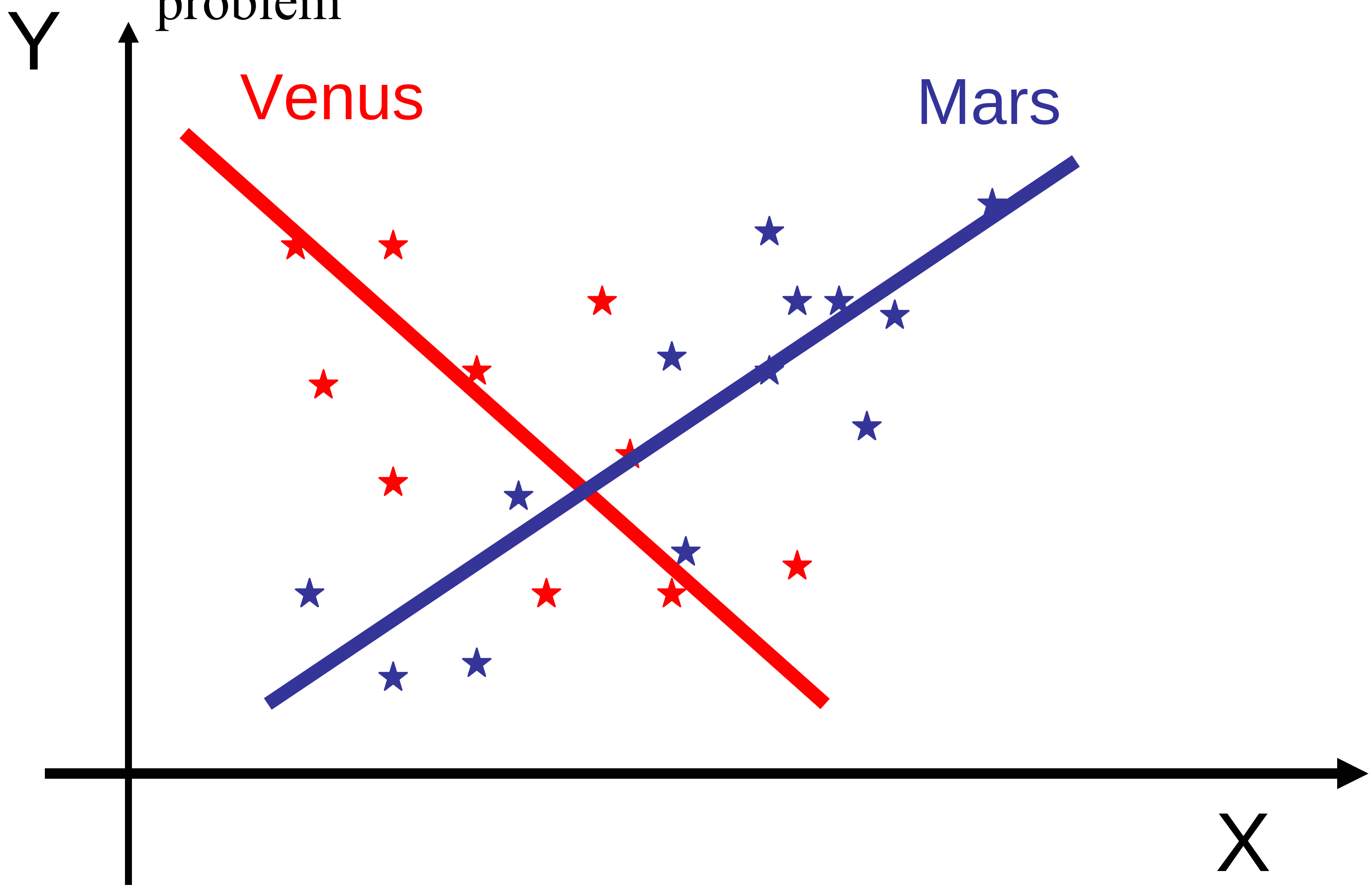
- Solution of the regular problem requires minimizing of $\|\mathbf{Ax}-\mathbf{b}\|^2$
- Solving this modified problemcorresponds to minimizing

$$\|\mathbf{Ax}-\mathbf{b}\|^2 + \varepsilon\|\mathbf{x}\|^2$$

- Philosophy – pay a "penalty" of $O(\varepsilon)$ to ensure solution does not blow up.
- In practice we may know that the data has an uncertainty of a certain magnitude … so it makes sense to optimize with this constraint.
- Ill-posed problems are also called "ill-conditioned"

Data comes from multiple models

- If data were labeled we are home …
  - Small amount of mislabeled data will not cause too much problem



- If there are many features and many models, then we need to do something different
- One approach … scan the space of parameters
  - Hough transform
- More generally use a “Voting” algorithm

RANSAC: Random Sample Consensus

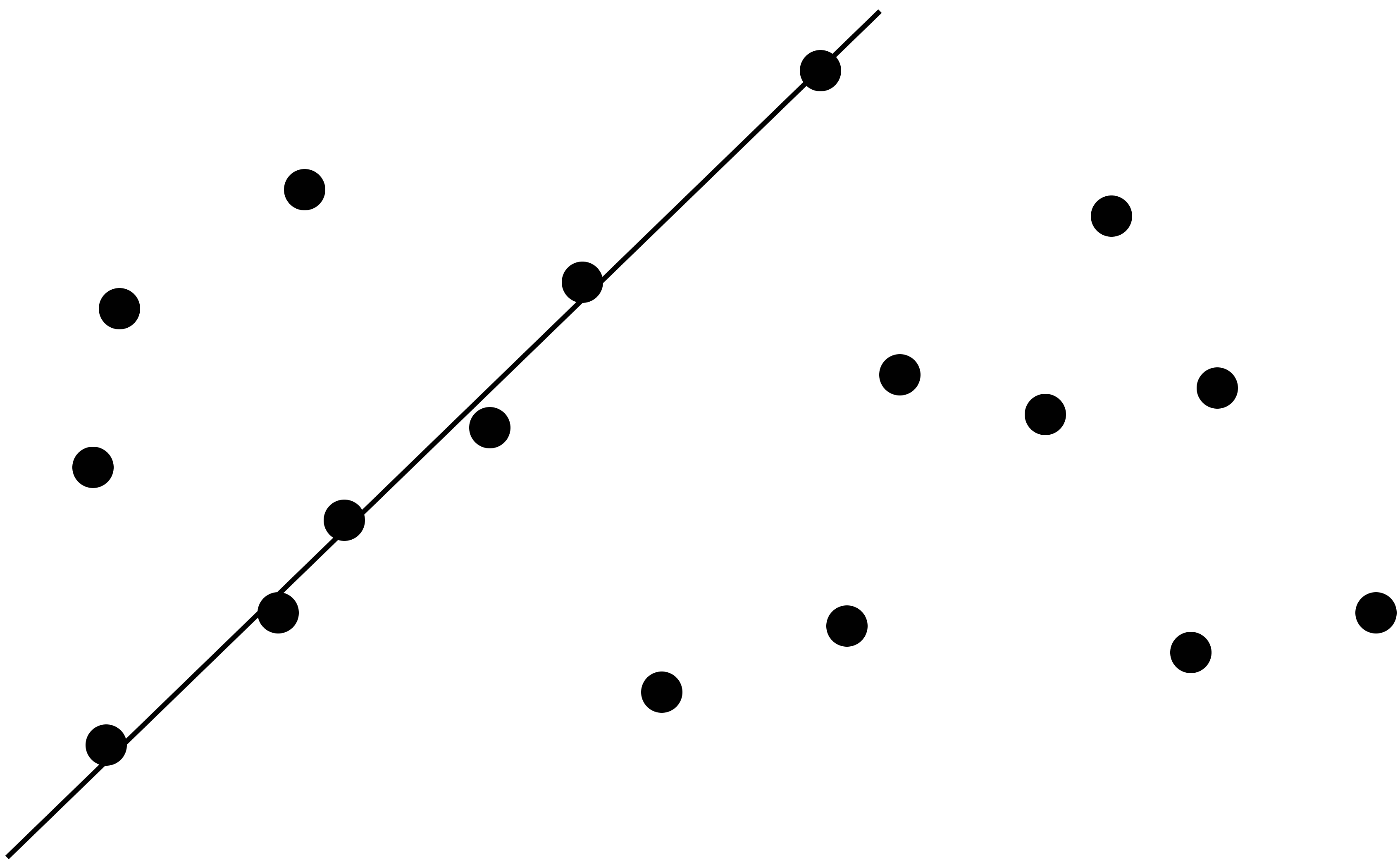
- Generate a bunch of reasonable hypotheses.
- Test to see which is the best.
- RANSAC for line fitting:
- Decide how good a line is:
  - Count number of points within ε of line.
    - Parameter ε measures the amount of noise expected.
  - Other possibilities. For example, for these points, also look at how far they are.
- Pick the best line.

Algorithm 15.4: RANSAC: fitting lines using random sample consensus

Determine:
  n — the smallest number of points required
  k — the number of iterations required
  t — the threshold used to identify a point that fits well
  d — the number of nearby points required
    to assert a model fits well
Until k iterations have occurred
  Draw a sample of n points from the data
    uniformly and at random
  Fit to that set of n points
  For each data point outside the sample
    Test the distance from the point to the line
      against t; if the distance from the point to the line
      is less than t, the point is close
  end
  If there are d or more points close to the line
    then there is a good fit. Refit the line using all
    these points.
end
Use the best fit from this collection, using the
  fitting error as a criterion

(Forsyth & Ponce)

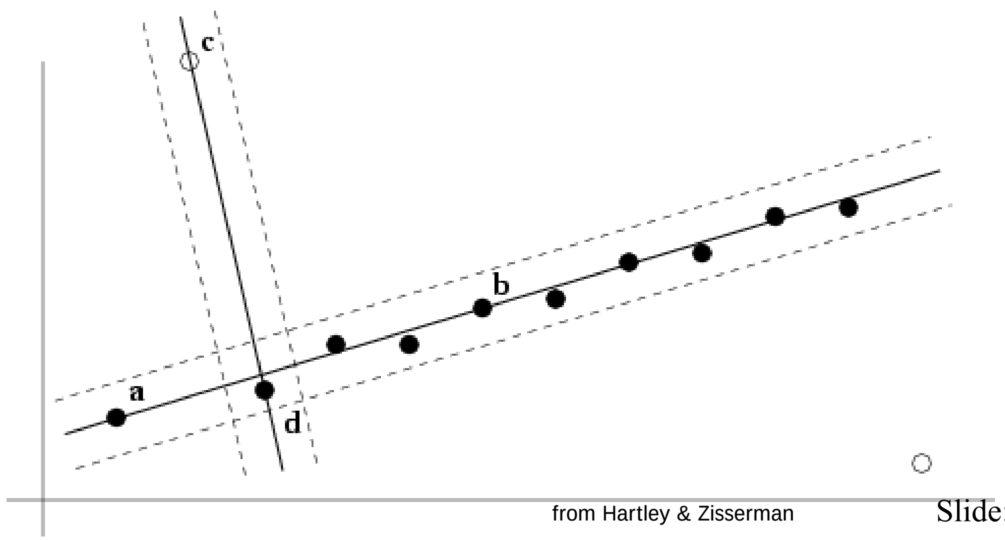
Line Grouping Problem



RANSAC (RANdom SAmple Consensus)

1. Randomly choose minimal subset of data points necessary to fit model (a *sample*)
2. Points within some distance threshold t of model are a *consensus set*. Size of consensus set is model's *support*
3. Repeat for N samples; model with biggest support is most robust fit
   – Points within distance t of best model are inliers
   – Fit final model to all inliers

Two samples
and their supports
for line-fitting



from Hartley & Zisserman

Slide: Christopher Rasmussen

Algorithm 15.4: RANSAC: fitting lines using random sample consensus

Determine:
  n — the smallest number of points required
  k — the number of iterations required
  t — the threshold used to identify a point that fits well
  d — the number of nearby points required
    to assert a model fits well
Until k iterations have occurred
  Draw a sample of n points from the data
    uniformly and at random
  Fit to that set of n points
  For each data point outside the sample
    Test the distance from the point to the line
      against t; if the distance from the point to the line
      is less than t, the point is close
  end
  If there are d or more points close to the line
    then there is a good fit. Refit the line using all
    these points.
end
Use the best fit from this collection, using the
  fitting error as a criterion

RANSAC: How many samples?

How many samples are needed?

Suppose *w* is fraction of inliers (points from line).

n points needed to define hypothesis (2 for lines)

k samples chosen.

Probability that a single sample of n points is correct:

$$w^n$$

Probability that all samples fail is:

$$(1-w^n)^k$$

Choose *k* high enough to keep this below desired failure rate.

RANSAC: Computed k ($p = 0.99$)

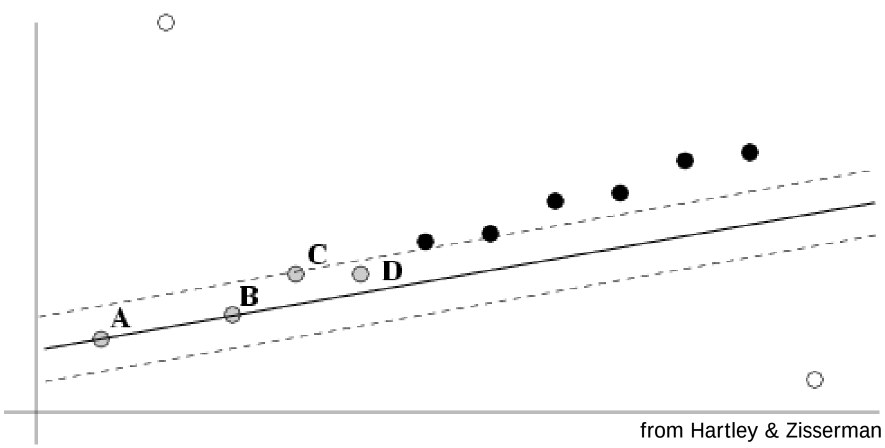
Sample size	Proportion of outliers						
n	**5%**	**10%**	**20%**	**25%**	**30%**	**40%**	**50%**
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

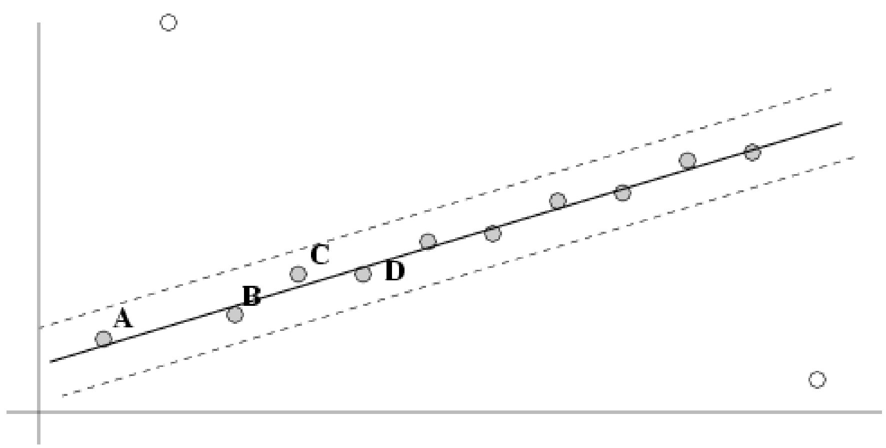
adapted from Hartley & Zisserman

Slide credit: Christopher Rasmussen

After RANSAC

- RANSAC divides data into inliers and outliers and yields estimate computed from minimal set of inliers
- Improve this initial estimate with estimation over all inliers (e.g., with standard least-squares minimization)
- But this may change inliers, so alternate fitting with re-classification as inlier/outlier





Slide credit: Christopher Rasmussen