

Model Fitting and Least Squares

Computer Vision I (CSCI 3240U)

Faisal Z. Qureshi

<http://vclab.science.ontariotechu.ca>



Week 6 at a glance

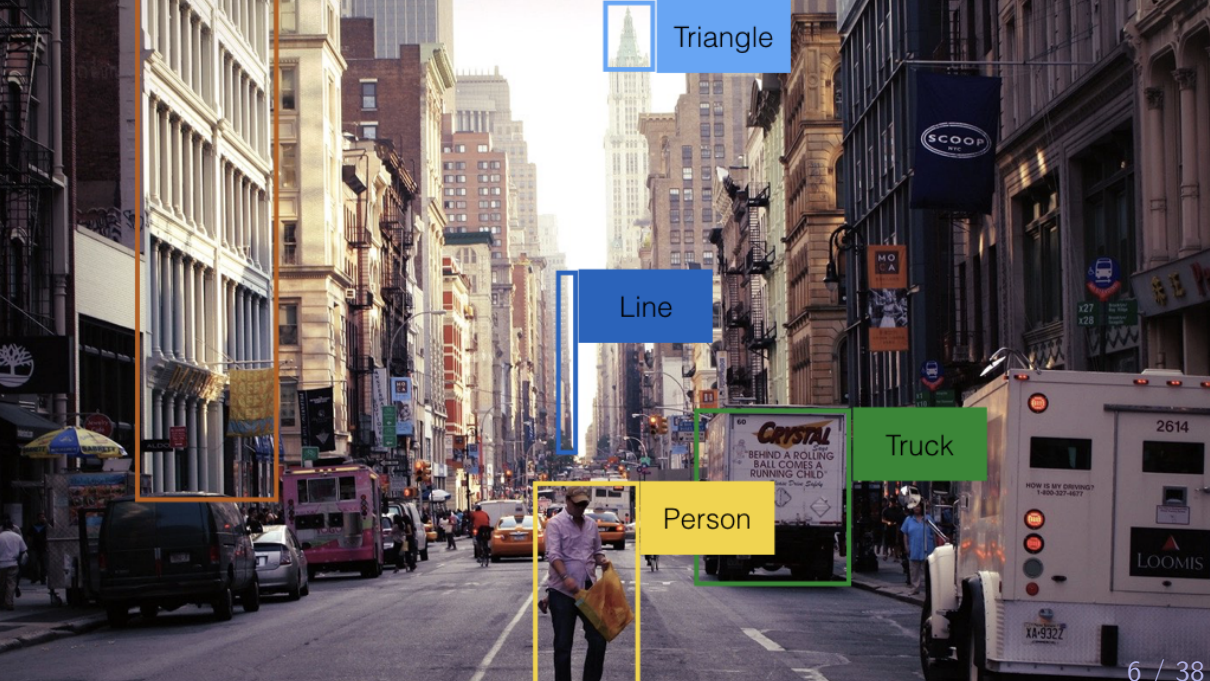
- ▶ **Lecture 1: Ordinary least squares (Sec. 8.1.1, App. A.2)**
 - ▶ Fitting a model to measurements
 - ▶ The normal equations
 - ▶ Two ways OLS fails
- ▶ **Lecture 2: Total least squares (App. A.2)**
 - ▶ Perpendicular distance
 - ▶ Line and plane fitting via SVD
 - ▶ Breakdown point

Reading: Szeliski Sec. 8.1.1, App. A.2. Exercises w06a, w06b. **Lab 5** this week; **Lab 6** builds on this directly.

Lecture 1: Ordinary least squares







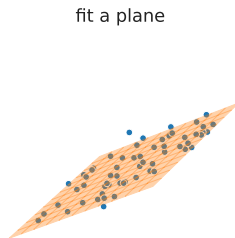
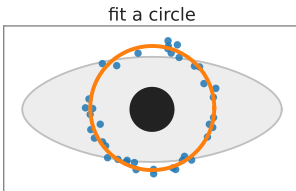
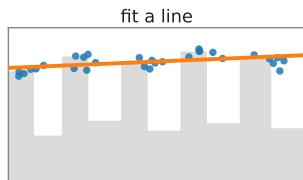
Triangle

Line

Truck

Person

Why fit models?



The rest of this course is variations on a single move: **noisy measurements in, a model out**. A line through edge pixels, a circle through iris samples, a plane through a point cloud.

The situation

Weeks 3–4 gave us *measurements*: edge pixels, gradient directions, candidate points.

Now we want to turn measurements into **structure**: a line, a plane, a homography, a camera pose.

Every one of these is the same problem:

Given noisy measurements and a parametric model, find the parameters that best explain the measurements.

Two questions to answer each time: **what does “best” mean**, and **what happens when some measurements are simply wrong?**

Fitting a line: the setup

Model $y = mx + c$, two parameters. Data (x_i, y_i) , $i = 1, \dots, n$.

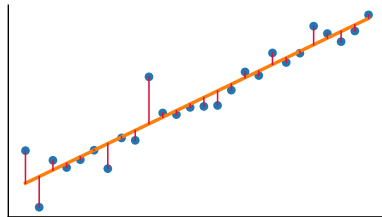
Define the **residual** at point i :

$$r_i = y_i - (mx_i + c)$$

Least squares chooses parameters minimising

$$E(m, c) = \sum_{i=1}^n r_i^2 = \sum_{i=1}^n (y_i - mx_i - c)^2$$

vertical residuals minimised by OLS



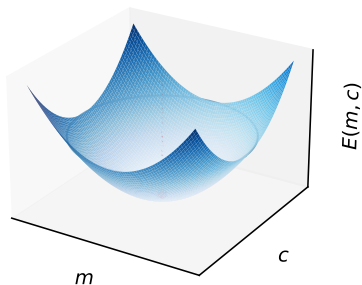
Why squares?

- ▶ **Differentiable**, so we can solve in closed form. $|r|$ is not.
- ▶ Gives the **maximum-likelihood** estimate when the noise is zero-mean Gaussian and independent.
- ▶ Penalises large residuals heavily.

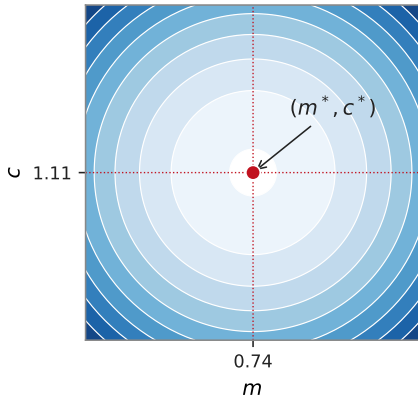
That last property is a liability, not a feature. A residual of 100 contributes 10,000, so one bad measurement can outvote a hundred good ones. We return to this at the end of the lecture.

The cost is a bowl

bowl-shaped error surface



contours: minimum at the centre



$E(m, c)$ is quadratic in m and c : one minimum, a smooth descent to it from every direction. This is why setting the two partial derivatives to zero finds the answer, and why no iteration is needed.

Solving it

Set the derivatives to zero:

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

which gives the closed form

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

- ▶ The fitted line always passes through the **centroid** $(\bar{x}, \bar{y})$.
- ▶ No iteration is needed.

The step-by-step derivation is in the appendix at the end of these slides.

Matrix form

Stack the equations. With $\theta = (m, c)^\top$,

$$\mathbf{A} = \begin{pmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

minimise $\|\mathbf{A}\theta - \mathbf{b}\|^2$.

Setting the gradient to zero gives the **normal equations**

$$\mathbf{A}^\top \mathbf{A} \theta = \mathbf{A}^\top \mathbf{b} \quad \implies \quad \theta = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b}$$

In practice, do not form $(\mathbf{A}^\top \mathbf{A})^{-1}$. Use `np.linalg.lstsq`, which is more stable.

The step-by-step derivation is in the appendix at the end of these slides.

Generalising: it need not be a line

The same machinery fits anything **linear in the parameters**:

$$\hat{y} = \mathbf{w}^\top \mathbf{x} + b$$

- ▶ d features $\Rightarrow d + 1$ parameters (do not forget the bias).
- ▶ Mean squared error:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N (y^{(i)} - \hat{y}^{(i)})^2$$

Polynomials are still *linear in the parameters*: fit $y = ax^2 + bx + c$ by putting x^2 , x , 1 in the columns of $\mathbf{A}$. “Linear model” refers to the parameters, not to the shape of the curve.

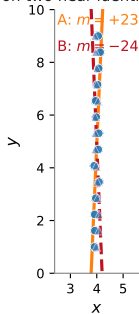
This is exercise w06a.

Failure 1: the parameterisation

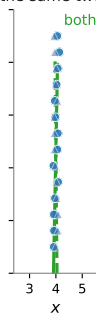
$y = mx + c$ cannot represent a **vertical line**: $m \rightarrow \infty$.

- ▶ A vertical line simply is not in the model. Near-vertical data gives huge, unstable m : two near-identical scatters produce very different fits.
- ▶ The choice of which axis is “ y ” is arbitrary; fitting x against y gives a *different* answer on the same points.

OLS on two near-identical scatters



TLS on the same two scatters



That asymmetry should bother you. The fix is next lecture.

Failure 2: outliers

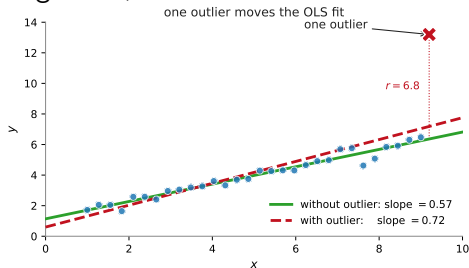
Take a clean line and add **one** grossly wrong point.

- ▶ The squared penalty makes the fit swing towards the outlier.
- ▶ With n good points and one bad one, the bad one can still dominate.

Formally: least squares has a **breakdown point of zero**. A single arbitrarily bad measurement can move the estimate arbitrarily far.

And in vision, outliers are not rare: a feature matcher routinely produces 30–50% wrong matches.

This is not an edge case; it is the normal case.

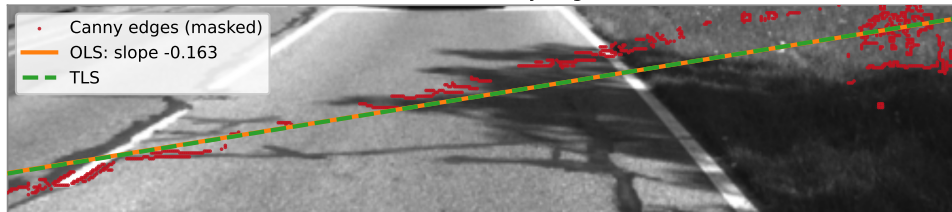


Lecture 1 recap

- ▶ Model fitting: choose parameters minimising a cost over residuals.
- ▶ Squared residuals give a closed form and the ML estimate under Gaussian noise.
- ▶ The fit passes through the centroid; use `lstsq`, not the explicit inverse.
- ▶ $y = mx + c$ cannot represent vertical lines and is asymmetric in x and y .
- ▶ One outlier can destroy the estimate: **breakdown point zero**.

On real data: Canny edges to a lane line

KITTI um 000032: line fitted to Canny edges of the lane divider



The next-week loop, previewed: Canny picks edge pixels; OLS or TLS turns them into a line. Here the two fits agree to a hundredth of a degree because the line is well-behaved. Lab 6 asks you to do the same on lane markings.

Lecture 2: Total least squares

Fixing the asymmetry

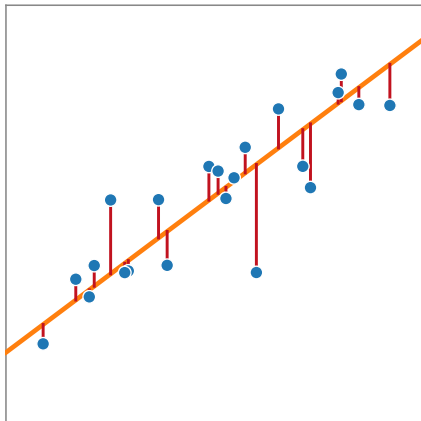
OLS minimises **vertical** distance, errors in y only.

But an edge pixel has error in **both** coordinates. There is no reason to privilege y .

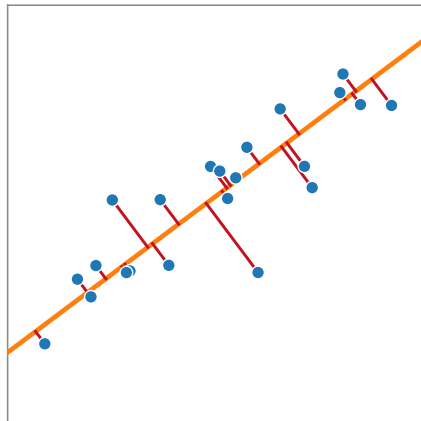
Instead, minimise the **perpendicular** distance to the line. This is **total least squares** (also: orthogonal regression).

OLS versus TLS residuals

OLS: vertical distance



TLS: perpendicular distance



A better line representation

Write the line implicitly:

$$ax + by + c = 0, \quad a^2 + b^2 = 1$$

- ▶ $\mathbf{n} = (a, b)$ is the **unit normal**.
- ▶ With the constraint, $|ax_i + by_i + c|$ is the perpendicular distance from (x_i, y_i) to the line.
- ▶ Vertical lines are perfectly representable: $a = 1, b = 0$.
- ▶ The constraint is needed because the equation is homogeneous: scaling (a, b, c) changes nothing.

Solving TLS

Minimise

$$E = \sum_i (ax_i + by_i + c)^2 \quad \text{subject to} \quad a^2 + b^2 = 1$$

Setting $\partial E / \partial c = 0$ gives $c = -(a\bar{x} + b\bar{y})$: again the line passes through the centroid.
Substituting back,

$$E = \sum_i (a(x_i - \bar{x}) + b(y_i - \bar{y}))^2 = \|\mathbf{M}\mathbf{n}\|^2$$

where $\mathbf{M}$ has the **centred** points as rows.

Minimising $\|\mathbf{M}\mathbf{n}\|$ subject to $\|\mathbf{n}\| = 1$:

$\mathbf{n}$ is the right singular vector of $\mathbf{M}$ for the SMALLEST singular value.

The step-by-step derivation is in the appendix at the end of these slides.

The recipe

1. Compute the centroid $\bar{\mathbf{p}}$.
2. Build $\mathbf{M}$ with rows $\mathbf{p}_i - \bar{\mathbf{p}}$.
3. SVD: $\mathbf{M} = U\Sigma V^\top$.
4. $\mathbf{n} = (a, b)$ is the **last** column of V .
5. $c = -\mathbf{n} \cdot \bar{\mathbf{p}}$.

Equivalently, $\mathbf{n}$ is the eigenvector of the scatter matrix $\mathbf{M}^\top \mathbf{M}$ with the smallest eigenvalue: the direction of **least variance**.

Smallest, not largest. This is the most common error in this material, and it recurs in Week 9 when we solve $\mathbf{A}\mathbf{h} = \mathbf{0}$.

The same recipe fits a plane

For 3D points and a plane $ax + by + cz + d = 0$ with $\|(a, b, c)\| = 1$:

1. Centroid.
2. $\mathbf{M}$ is $n \times 3$ with centred points as rows.
3. SVD; the normal is the singular vector for the smallest singular value.
4. $d = -\mathbf{n} \cdot \bar{\mathbf{p}}$.

This is exercise w06b, and it is genuinely useful: fit a plane to a LiDAR or stereo point cloud to find the **ground surface**. Points well above it are obstacles.

You will use exactly this in Lab 9 to check your stereo depths against KITTI's road plane.

Does TLS fix outliers?

No.

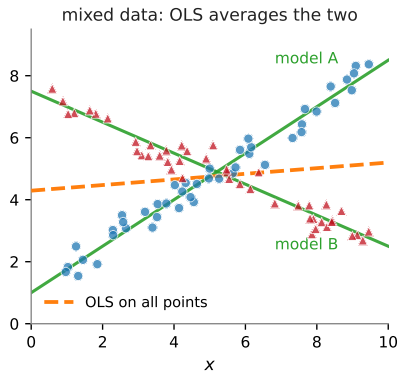
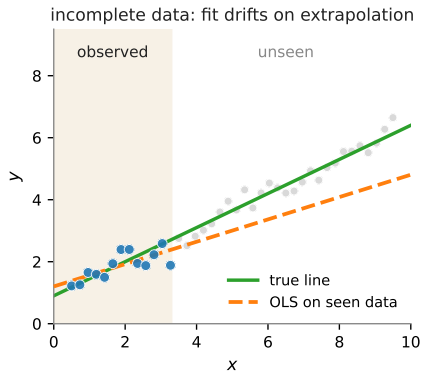
TLS changes the *distance metric*, perpendicular instead of vertical. It is still a least-squares estimator, still minimising a sum of squares, and still has **breakdown point zero**.

One arbitrarily bad point still destroys it.

This is the question students most often get wrong. Be clear about what TLS does and does not buy you:

- ▶ Fixes: the vertical-line failure, the x/y asymmetry.
- ▶ Does not fix: outliers.

Two more things least squares cannot do



- ▶ **Incomplete data.** Fit is only valid where you have samples; extrapolation drifts.
- ▶ **Mixed data.** Two populations from two different models are averaged into one line that fits neither. Hough handles this next week.

What we actually need

For real data we need an estimator that:

- ▶ ignores a substantial fraction of grossly wrong measurements, and
- ▶ uses all the good ones.

Least squares cannot do this, because every point contributes to the cost no matter how absurd it is.

The answer, **RANSAC**, is next week. The idea is to stop trying to fit all the data at once and instead *hypothesise* a model from a minimal sample, then *count* how many measurements agree.

Week 6 recap

- ▶ Fitting = minimising a cost over residuals.
- ▶ OLS: closed form, ML under Gaussian noise, passes through the centroid.
- ▶ OLS fails on vertical lines and is asymmetric in x, y .
- ▶ TLS: perpendicular distance, implicit line, solved by SVD, taking the **smallest** singular vector.
- ▶ The same recipe fits planes in 3D.
- ▶ **Neither** OLS nor TLS tolerates outliers: breakdown point zero.

This week's work

- ▶ **Reading:** Szeliski Sec. 8.1.1, Appendix A.2.
- ▶ **Exercises:** w06a (linear models and MSE), w06b (plane fitting).
- ▶ **Lab 5:** image pyramids and scale. Decide which pyramid level you will hand to next week's line fitting.

Next week: robustness. RANSAC, and the Hough transform.

Appendix: worked derivations

The remaining slides show, step by step, the closed-form solutions that Lectures 1 and 2 assert:

- ▶ **OLS, scalar form:** $\partial E / \partial c$ and $\partial E / \partial m$ give the two formulas for c and m .
- ▶ **OLS, matrix form:** $\nabla_{\theta} \|\mathbf{A}\theta - \mathbf{b}\|^2 = \mathbf{0}$ gives the **normal equations**.
- ▶ **TLS:** $\partial E / \partial c = 0$, then substitution to $\|\mathbf{M}\mathbf{n}\|^2$, and the SVD argument that picks the smallest singular vector.

Skip these if you are comfortable with the results; return here when you implement Lab 6.

Aside: OLS, the derivative in c

Start from

$$E(m, c) = \sum_{i=1}^n (y_i - mx_i - c)^2.$$

Chain rule on each squared term:

$$\frac{\partial E}{\partial c} = \sum_{i=1}^n 2(y_i - mx_i - c) \cdot (-1) = -2 \sum_{i=1}^n (y_i - mx_i - c).$$

Set it to zero and divide by -2 :

$$\sum_i y_i - m \sum_i x_i - nc = 0 \quad \implies \quad \boxed{c = \bar{y} - m \bar{x}}$$

That is why the fit passes through the centroid: solving one equation forces the line through $(\bar{x}, \bar{y})$ before we even touch m .

Aside: OLS, the derivative in m

Same E ; differentiate with respect to m :

$$\frac{\partial E}{\partial m} = \sum_i 2(y_i - mx_i - c) \cdot (-x_i) = -2 \sum_i x_i (y_i - mx_i - c).$$

Set it to zero, expand:

$$\sum_i x_i y_i - m \sum_i x_i^2 - c \sum_i x_i = 0.$$

Substitute $c = \bar{y} - m\bar{x}$ from the previous slide and use $\sum_i x_i = n\bar{x}$:

$$\sum_i x_i y_i - m \sum_i x_i^2 - (\bar{y} - m\bar{x}) n\bar{x} = 0.$$

Collect m on the left, and use $\sum_i x_i^2 - n\bar{x}^2 = \sum_i (x_i - \bar{x})^2$,
 $\sum_i x_i y_i - n\bar{x}\bar{y} = \sum_i (x_i - \bar{x})(y_i - \bar{y})$:

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

Aside: OLS, the normal equations

Same problem in matrix form. With $\boldsymbol{\theta} = (m, c)^\top$, $\mathbf{A} \in \mathbb{R}^{n \times 2}$ having rows $(x_i, 1)$, and $\mathbf{b} = (y_1, \dots, y_n)^\top$:

$$E(\boldsymbol{\theta}) = \|\mathbf{A}\boldsymbol{\theta} - \mathbf{b}\|^2 = (\mathbf{A}\boldsymbol{\theta} - \mathbf{b})^\top (\mathbf{A}\boldsymbol{\theta} - \mathbf{b}).$$

Expand the product:

$$E(\boldsymbol{\theta}) = \boldsymbol{\theta}^\top \mathbf{A}^\top \mathbf{A} \boldsymbol{\theta} - 2 \boldsymbol{\theta}^\top \mathbf{A}^\top \mathbf{b} + \mathbf{b}^\top \mathbf{b}.$$

Differentiate with respect to $\boldsymbol{\theta}$. Two standard vector identities do the work:

$\nabla(\boldsymbol{\theta}^\top \mathbf{M} \boldsymbol{\theta}) = 2 \mathbf{M} \boldsymbol{\theta}$ for symmetric $\mathbf{M}$, and $\nabla(\boldsymbol{\theta}^\top \mathbf{v}) = \mathbf{v}$:

$$\nabla_{\boldsymbol{\theta}} E = 2 \mathbf{A}^\top \mathbf{A} \boldsymbol{\theta} - 2 \mathbf{A}^\top \mathbf{b}.$$

Set it to zero and cancel the 2:

$$\boxed{\mathbf{A}^\top \mathbf{A} \boldsymbol{\theta} = \mathbf{A}^\top \mathbf{b}} \implies \boldsymbol{\theta}^* = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b} \text{ if } \mathbf{A}^\top \mathbf{A} \text{ is invertible.}$$

Never form $(\mathbf{A}^\top \mathbf{A})^{-1}$ in code. The condition number of $\mathbf{A}^\top \mathbf{A}$ is the square of that of $\mathbf{A}$, so `np.linalg.lstsq` (which uses SVD or QR) is dramatically more stable.

Aside: TLS, the derivative in c

Start from the constrained problem

$$E = \sum_{i=1}^n (ax_i + by_i + c)^2 \quad \text{subject to} \quad a^2 + b^2 = 1.$$

The constraint involves only a and b , so treat c as free. Chain rule:

$$\frac{\partial E}{\partial c} = \sum_i 2(ax_i + by_i + c) \cdot 1 = 2 \sum_i (ax_i + by_i + c).$$

Set it to zero, use $\sum_i x_i = n\bar{x}$ and $\sum_i y_i = n\bar{y}$:

$$a n\bar{x} + b n\bar{y} + nc = 0$$

$\implies$

$$c = -(a\bar{x} + b\bar{y})$$

The TLS line, like the OLS line, passes through the centroid $(\bar{x}, \bar{y})$; the derivation for c is one line shorter because the constraint hides in (a, b) .

Aside: TLS, substituting c

Put $c = -(a\bar{x} + b\bar{y})$ back into E :

$$E = \sum_i \left(ax_i + by_i - a\bar{x} - b\bar{y} \right)^2 = \sum_i \left(a(x_i - \bar{x}) + b(y_i - \bar{y}) \right)^2.$$

Each summand is now the *inner product* of the row $(x_i - \bar{x}, y_i - \bar{y})$ with $\mathbf{n} = (a, b)^\top$.
Stack the rows:

$$\mathbf{M} = \begin{pmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{pmatrix}, \quad \mathbf{n} = \begin{pmatrix} a \\ b \end{pmatrix}.$$

Then the i -th entry of $\mathbf{M}\mathbf{n}$ is exactly the term inside each squared bracket above, so

$$\boxed{E = \|\mathbf{M}\mathbf{n}\|^2 \quad \text{subject to} \quad \|\mathbf{n}\| = 1}$$

with $\mathbf{M}$ the matrix of **centred** points as rows.

Aside: TLS via SVD

Take the SVD $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ with $\mathbf{\Sigma} = \text{diag}(\sigma_1, \sigma_2)$ and $\sigma_1 \geq \sigma_2 \geq 0$.

$\mathbf{U}$ has orthonormal columns and $\mathbf{V}$ is orthogonal, so

$$\|\mathbf{M}\mathbf{n}\|^2 = \|\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top \mathbf{n}\|^2 = \|\mathbf{\Sigma}\mathbf{V}^\top \mathbf{n}\|^2.$$

Write $\mathbf{y} = \mathbf{V}^\top \mathbf{n}$; the constraint $\|\mathbf{n}\| = 1$ becomes $\|\mathbf{y}\| = 1$.

Then

$$\|\mathbf{\Sigma}\mathbf{y}\|^2 = \sigma_1^2 y_1^2 + \sigma_2^2 y_2^2 \quad \text{minimised over} \quad y_1^2 + y_2^2 = 1.$$

With $\sigma_1 \geq \sigma_2$, put all the unit mass on the smallest singular value: $\mathbf{y} = (0, 1)^\top$.

Undo the change of variables, $\mathbf{n} = \mathbf{V}\mathbf{y}$:

$\mathbf{n} = \mathbf{v}_2$, the right singular vector for the smallest singular value.

Equivalently, $\mathbf{n}$ is the eigenvector of $\mathbf{M}^\top \mathbf{M}$ with the smallest eigenvalue, since the squared singular values are its eigenvalues. This is the pattern that recurs in Week 9 when solving $\mathbf{A}\mathbf{h} = \mathbf{0}$.

Copyright and License

©Faisal Z. Qureshi



This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.