# Orthogonal Projections and Overdetermined Linear Systems

Szymon Rusinkiewicz

COS 302, Fall 2020

PRINCETON UNIVERSITY

# Orthogonal Projections and Overdetermined Linear Systems

The technique of linear least squares will crop up many times during this course.

Today: study it from the point of view of *overdetermined* linear systems.

# Overdetermined Linear Systems

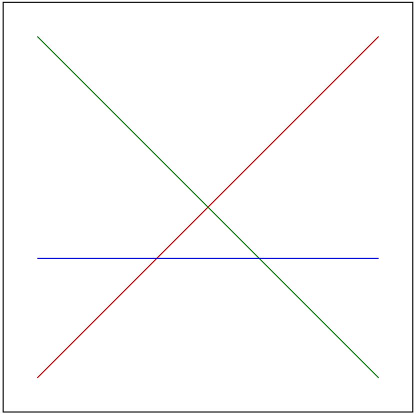
$$\begin{aligned} 2x - y &= -4 \\ 2x + y &= \phantom{-}4 \\ 2y &= \phantom{-}2 \end{aligned}$$

Overdetermined systems can't be solved. So why care about them?

- Measurements are noisy (e.g., imperfect sensors)
- Measurements are fundamentally uncertain (e.g., human preferences)
- Linear model is too simple, but used anyway

Lots of data, fit to an overdetermined model, can lead to accurate predictions.

# Overdetermined Linear Systems



$$\begin{aligned} 2x - y &= -4 \\ 2x + y &= 4 \\ 2y &= 2 \end{aligned}$$

This system can't be solved.
But intuitively, there should be a "compromise" solution that *almost* satisfies the equations…

# Overdetermined Linear Systems

Write as a matrix equation:

$$\begin{bmatrix} 2 & -1 \\ 2 & 1 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -4 \\ 4 \\ 2 \end{bmatrix}$$

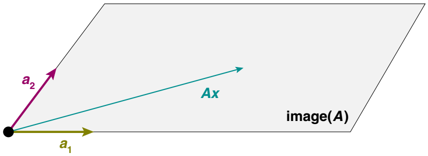
$$\boldsymbol{A} \quad \boldsymbol{x} = \boldsymbol{b}$$

Now, think of the intuition behind the linear transformation $\boldsymbol{A}$.
Its *columns* tell us where the $x$ and $y$ axes are sent.

$$\begin{bmatrix} 2 & -1 \\ 2 & 1 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -4 \\ 4 \\ 2 \end{bmatrix}$$

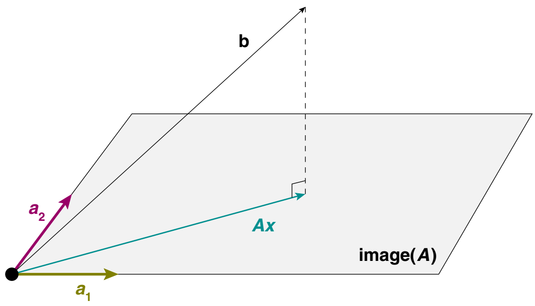
# Solving Overdetermined Systems

$$\underbrace{\begin{bmatrix} 2 & -1 \\ 2 & 1 \\ 0 & 2 \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} x \\ y \end{bmatrix}}_{\boldsymbol{x}} = \underbrace{\begin{bmatrix} -4 \\ 4 \\ 2 \end{bmatrix}}_{\boldsymbol{b}}$$




- Think of $A$ as a linear mapping from a 2-dimensional space to a 3-dimensional space.
- The set of points reachable by that mapping is a 2-D subset of the 3-D space. (i.e., the linear mapping is not *surjective*)
- The columns of $A$, namely $\boldsymbol{a}_1 = \begin{bmatrix} 2 \\ 2 \\ 0 \end{bmatrix}$ and $\boldsymbol{a}_2 = \begin{bmatrix} -1 \\ 1 \\ 2 \end{bmatrix}$, span that 2-D subspace. (i.e., they are a basis for the *image* of $A$).

# Orthogonal Projections




- The point $\boldsymbol{b}$ lies in the 3-D space, but not (in general) in that 2-D subspace. Our strategy will be to *project* $\boldsymbol{b}$ into the 2-D subspace.

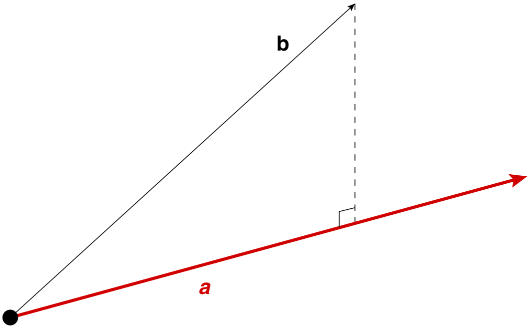
# Orthogonal Projections

Recall that the orthogonal projection of a point $\boldsymbol{b}$ onto direction $\boldsymbol{a}$ is

$$\left(\frac{\boldsymbol{a}}{||\boldsymbol{a}||} \cdot \boldsymbol{b}\right) \frac{\boldsymbol{a}}{||\boldsymbol{a}||} = \frac{\boldsymbol{a}^\top \boldsymbol{b}}{\boldsymbol{a}^\top \boldsymbol{a}} \boldsymbol{a} = \lambda \boldsymbol{a}$$

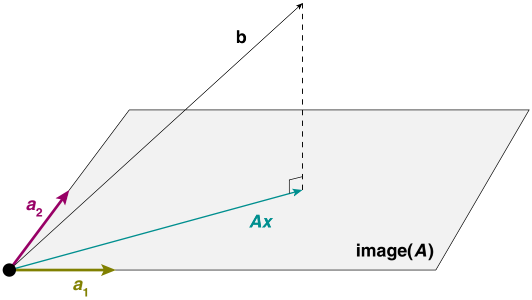
Let's write this as a linear equation for $\lambda$:

$$\boldsymbol{a}^\top \boldsymbol{a}\, \lambda = \boldsymbol{a}^\top \boldsymbol{b}$$



# Orthogonal Projections

We now want to project onto a space spanned by several directions:



# Orthogonal Projections

We write down the conditions for projection onto the two directions:

$$\begin{aligned} \boldsymbol{a}_1^\top \boldsymbol{a}_1\ \lambda_{\boldsymbol{a}_1} &= \boldsymbol{a}_1^\top \boldsymbol{b} \\ \boldsymbol{a}_2^\top \boldsymbol{a}_2\ \lambda_{\boldsymbol{a}_2} &= \boldsymbol{a}_2^\top \boldsymbol{b} \end{aligned}$$

Or,

$$\boldsymbol{A}^\top \boldsymbol{A} \begin{bmatrix} \lambda_{\boldsymbol{a}_1} \\ \lambda_{\boldsymbol{a}_2} \end{bmatrix} = \boldsymbol{A}^\top \boldsymbol{b}$$

But $\lambda_{\boldsymbol{a}_1}$ and $\lambda_{\boldsymbol{a}_2}$ are just the amounts of $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ in the projection — i.e., our original $x$ and $y$.

# Solving Overconstrained Systems

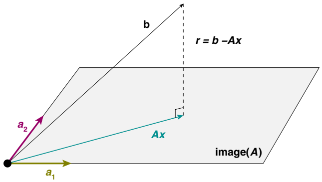
So, we get what are known as the normal equations of the original overconstrained linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$:

$$\boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^\top \boldsymbol{b}$$

(Notice, by the way, that $\boldsymbol{A}^\top \boldsymbol{A}$ is SPD. This will be important later.)

# Solving Overconstrained Systems

Alternative derivation: consider the residual $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$:




$\boldsymbol{r}$ must be perpendicular to image($\boldsymbol{A}$), so must be perpendicular to $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$:

$$\begin{aligned} \boldsymbol{r} \cdot \boldsymbol{a}_1 &= 0 \\ \boldsymbol{a}_1^\top(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}) &= 0 \\ \boldsymbol{a}_1^\top \boldsymbol{A}\boldsymbol{x} &= \boldsymbol{a}_1^\top \boldsymbol{b} \end{aligned}$$

$$\begin{aligned} \boldsymbol{r} \cdot \boldsymbol{a}_2 &= 0 \\ \boldsymbol{a}_2^\top(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}) &= 0 \\ \boldsymbol{a}_2^\top \boldsymbol{A}\boldsymbol{x} &= \boldsymbol{a}_2^\top \boldsymbol{b} \end{aligned}$$

# Solving Overconstrained Systems

- We get the same normal equations:

$$A^\top A\boldsymbol{x} = A^\top \boldsymbol{b}$$

- In ideal-perfectly-accurate-math land, we could solve this by multiplying by the inverse of $A^\top A$:

$$\boldsymbol{x} = (A^\top A)^{-1} A^\top \boldsymbol{b}$$

  The quantity $(A^\top A)^{-1} A^\top$ is called the *pseudoinverse* of $A$.

- But with roundoff-prone computer math, we don't do that.
  Solve the normal equations or, better yet, use SVD (next week!)

# Solving Overconstrained Systems

Back to our original problem:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

$$\begin{bmatrix} 2 & -1 \\ 2 & 1 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -4 \\ 4 \\ 2 \end{bmatrix}$$

Solve via:

$$\boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^\top \boldsymbol{b}$$

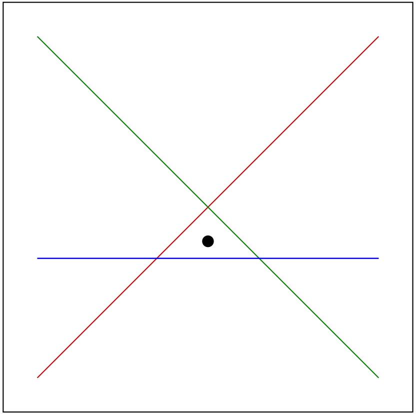
$$\begin{bmatrix} 2 & 2 & 0 \\ -1 & 1 & 2 \end{bmatrix} \begin{bmatrix} 2 & -1 \\ 2 & 1 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 & 2 & 0 \\ -1 & 1 & 2 \end{bmatrix} \begin{bmatrix} -4 \\ 4 \\ 2 \end{bmatrix}$$

# Solving Overconstrained Systems

$$\begin{bmatrix} 4 & 0 \\ 0 & 6 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 12 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$$

# Solving Overconstrained Systems



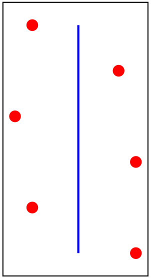
Note how the solution “splits the difference” between the three lines.

We will make this explicit later in the semester when we see other ways of deriving this procedure.

# Special Case: Constant

Let's say you want to solve the overdetermined system

$$\begin{aligned} x &= 2 \\ x &= 7 \\ x &= 1 \\ x &= 8 \\ x &= 2 \\ x &= 8 \end{aligned}$$



This is an overdetermined system of 6 equations in 1 variable.

# Special Case: Constant

Write as matrix equation:

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} 2 \\ 7 \\ 1 \\ 8 \\ 2 \\ 8 \end{bmatrix}$$

# Special Case: Constant

Solve normal equations:

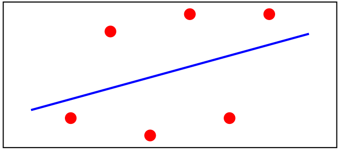
$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 7 \\ 1 \\ 8 \\ 2 \\ 8 \end{bmatrix}$$

$$x = \frac{2 + 7 + 1 + 8 + 2 + 8}{6}$$

Solution is the *mean* of the values!

# Special Case: Line

Let's say you want to fit a line $y = a + bx$ to a set of datapoints $(x_i, y_i)$:



Your system of equations is:

$$
\begin{aligned}
a + bx_1 &= y_1 \\
a + bx_2 &= y_2 \\
&\vdots
\end{aligned}
$$

# Special Case: Line

Write as a matrix equation:

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ \vdots & \vdots \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \end{bmatrix}$$

and solve the resulting normal equations for $a$ and $b$.

(You'll see ugly formulas out there involving lots of nasty summations, but much simpler to remember the general principle.)

# Line Fitting Caveats

- Single outlier can have large effect on best-fit line

- This minimizes "vertical" distance to line: not always what you want