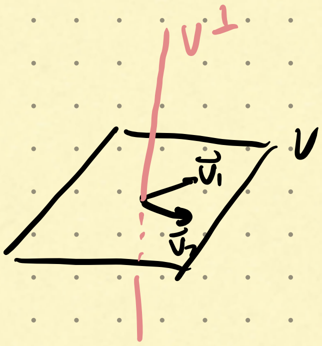
§5.4

- formula for $V^\perp$
- data fitting
- approx. solutions
- data fitting II

---

let $V = \text{span}\{\vec{v}_1, \dots, \vec{v}_n\} \subseteq \mathbb{R}^m$. Recall: $V^\perp = \{\vec{x} \in \mathbb{R}^m \mid \vec{x} \text{ is ortho to } \vec{v} \ \forall \vec{v} \in V\}$

"orthogonal complement"



Note: $\vec{x} \in V^\perp \iff \vec{x}$ is ortho to $\vec{v}_i$ for $i = 1, \dots, n$

$\iff \vec{x} \cdot \vec{v}_i = 0$ for $i = 1, \dots, n$

$$\iff \begin{bmatrix} - & \vec{v}_1^T & - \\ & \vdots & \\ - & \vec{v}_n^T & - \end{bmatrix} \vec{x} = \vec{0}$$

Note $V = \text{im} \begin{bmatrix} | & & | \\ \vec{v}_1 & \dots & \vec{v}_n \\ | & & | \end{bmatrix}$

Formula: $\vec{x} \in \left(\text{im}\begin{bmatrix} | & & | \\ \vec{v}_1 & \dots & \vec{v}_n \\ | & & | \end{bmatrix}\right)^\perp \iff \vec{x} \in \ker \begin{bmatrix} - & \vec{v}_1^T & - \\ & \vdots & \\ - & \vec{v}_n^T & - \end{bmatrix}$

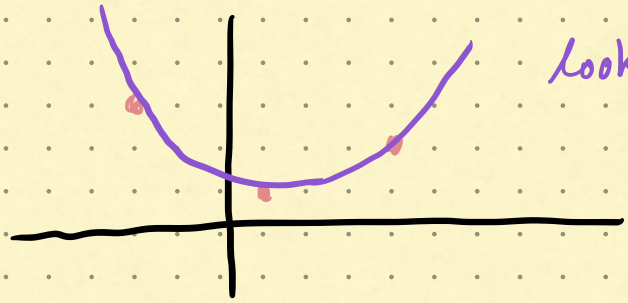
$$(\text{im } A)^\perp = \ker(A^T)$$

eg. $V = \text{span}\left(\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}\right)$. Then $V^\perp = \ker \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$

Data fitting (least squares regression)

The problem of data fitting: Given three data pts $(-2, 3)$, $(1, 1)$, $(4, 2)$.

Want to find the equation of a parabola going thru these pts



looking for $f(x) = c_0 + c_1 x + c_2 x^2$

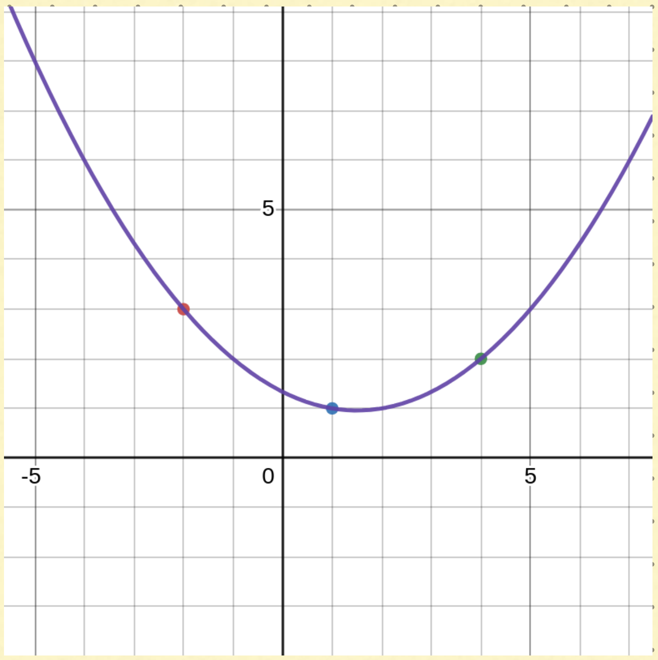
Plug in data pts:

$$c_0 + c_1(-2) + c_2(-2)^2 = 3$$
$$c_0 + c_1(1) + c_2(1)^2 = 1$$
$$c_0 + c_1 \cdot 4 + c_2 \cdot 4^2 = 2$$

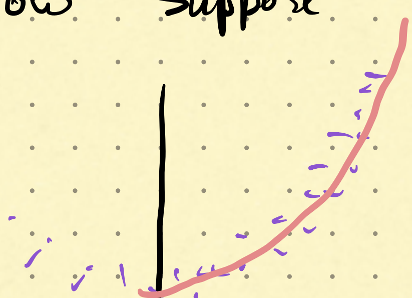
Matrix eqn: $\begin{bmatrix} 1 & -2 & 4 \\ 1 & 1 & 1 \\ 1 & 4 & 16 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix}$

Solve for c_0, c_1, c_2: $c_0 = 4/3$, $c_1 = -1/2$, $c_2 = 1/6$

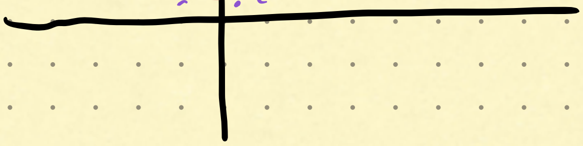
So $f(x) = 4/3 - \frac{1}{2}x + \frac{1}{6}x^2$



Now suppose we N data pts $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$



Q which parabola "best fits" these data?



We can try plugging in our data pts and solving as before:

$$c_0 + c_1 x_1 + c_2 x_1^2 = y_1$$
$$c_0 + c_1 x_2 + c_2 x_2^2 = y_2$$
$$\vdots$$
$$c_0 + c_1 x_N + c_2 x_N^2 = y_N$$

$$\rightarrow \begin{bmatrix} 1 & x_1 & x_1^2 \\ & \vdots & \\ 1 & x_N & x_N^2 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

<u>Q</u> do you expect this system of eqs to have a solution?

<u>A</u> Probably not! Having a solution means our parabola goes thru every single data pt, which is not realistic.

<u>Big Q</u>: how can we find a good, <u>approximate</u> solution?

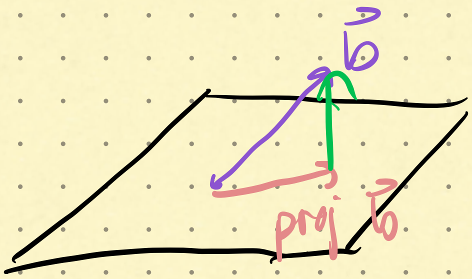
<u>Ans</u>: orthogonal projection!

§ Approx. Solutions

let $A = m\times n$ matrix, let $\vec{b} \in \mathbb{R}^m$

Suppose $\vec{b} \notin \text{im } A$. So we can't solve $A\vec{x} = \vec{b}$

We want to find $\vec{x}$ s.t. $A\vec{x} \approx \vec{b}$



im A

Idea: $\vec{x} \in \mathbb{R}^n$ solves $A\vec{x} \approx \vec{b}$

if $A\vec{x} = \underbrace{\text{proj}_{\text{im } A}(\vec{b})}$

the vector on im A closest to $\vec{b}$

This $\vec{x}$ is called the "least squares solution" to $A\vec{x} = \vec{b}$

Formula: $A\vec{x} = \text{proj}_{\text{im} A}(\vec{b}) \iff A^T A\vec{x} = A^T\vec{b}$

"normal equation"

why? $A\vec{x} = \text{proj}_{\text{im} A}(\vec{b}) \iff (\vec{b} - A\vec{x})$ is ortho to im A

$\iff A^T(\vec{b} - A\vec{x}) = 0$

$\iff A^T\vec{b} = A^T A\vec{x}$

---

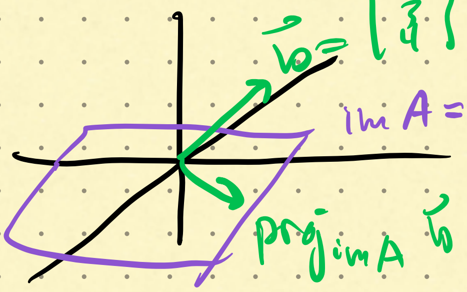
ex Find the least squares solution to

$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}\vec{x} = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$ Multiply both sides by

$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$

$$\begin{bmatrix}1&0&0\\0&1&0\end{bmatrix}\begin{bmatrix}1&0\\0&0\\0&1\end{bmatrix}\vec{x} = \begin{bmatrix}1&0&0\\0&1&0\end{bmatrix}\begin{bmatrix}2\\3\\4\end{bmatrix}$$

$$\rightarrow \begin{bmatrix}1&0\\0&1\end{bmatrix}\vec{x} = \begin{bmatrix}2\\3\end{bmatrix} \rightarrow \boxed{\vec{x} = \begin{bmatrix}2\\3\end{bmatrix}}$$



$\vec{b} = \begin{bmatrix}2\\3\\4\end{bmatrix}$

$\text{im}\, A =$ xy-plane in $\mathbb{R}^3$

$\text{proj}_{\text{im} A}\, \vec{b} = \begin{bmatrix}2\\3\\0\end{bmatrix} = A \cdot \begin{bmatrix}2\\3\end{bmatrix}$ least squares soln.

Back to data fitting

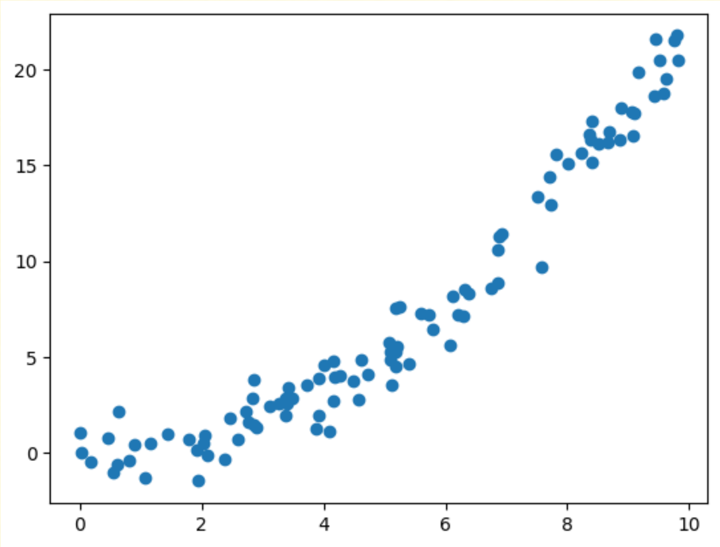
data pts $(x_1, y_1), \ldots, (x_N, y_N)$

want to approximately solve

$$\begin{bmatrix}1 & x_1 & x_1^2\\ & \vdots & \\ 1 & x_N & x_N^2\end{bmatrix}\begin{bmatrix}c_0\\c_1\\c_2\end{bmatrix} = \begin{bmatrix}y_1\\ \vdots\\ y_N\end{bmatrix}$$

So, we should solve the system

$$\begin{bmatrix}1 & \cdots & 1\\ x_1 & & x_N\\ x_1^2 & & x_N^2\end{bmatrix}\begin{bmatrix}1 & x_1 & x_1^2\\ & \vdots & \\ 1 & x_N & x_N^2\end{bmatrix}\begin{bmatrix}c_0\\c_1\\c_2\end{bmatrix} = \begin{bmatrix}1 & \cdots & 1\\ x_1 & & x_N\\ x_1^2 & & x_N^2\end{bmatrix}\begin{bmatrix}y_1\\ \vdots\\ y_N\end{bmatrix}$$



In [8]:
```
# Solve the "normal equation" we saw in class: A^T A x = A^
A_transpose_A = np.matmul(Amatrix.transpose(),Amatrix)
A_transpose_y = np.matmul(Amatrix.transpose(),yarray)
coeffs = np.linalg.solve(A_transpose_A,A_transpose_y)
```

In [9]:
```
coeffs
```

Out[9]:
```
array([ 0.13172747, -0.21136219,  0.24281359])
```