20:20

EECS 16A

Discussion 14A Notes

1 Polynomial Fitting

Suppose we have a function $y(x)$ which has been sampled as shown right.

x	y
0.0	24.0
0.5	6.61
1.0	0.0
1.5	-0.95
2.0	0.07
2.5	0.73
3.0	-0.12
3.5	-0.83
4.0	-0.04
4.5	6.42

The general fit we will choose is:

$$y(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4$$

a) What are the unknowns here?

Unknowns: $\vec{a} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}$ ✓

In this case the 'x' values are part of the data; they're known!

(Technically we've "sampled" at various 'x' values; point is still valid)

b) Can you write out an equation for the 1st expression $\overset{x}{0.0}$ $\overset{y}{24.0}$?

$$y(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4$$

$$\underbrace{24 = a_0} + \cancel{a_1 0} + \cancel{a_2 0} + \cancel{a_3 0} + \cancel{a_4 0}$$

First equation is pretty straight forward! Corresponds to y-intercept.

c] Write out the remaining equations!

x	y
0.0	24.0
0.5	6.61
1.0	0.0
1.5	-0.95
2.0	0.07
2.5	0.73
3.0	-0.12
3.5	-0.83
4.0	-0.04
4.5	6.42

$$y(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4$$

From Above!

$$24 = a_0$$

$$6.61 = a_0 + a_1\left(\tfrac{1}{2}\right) + a_2\left(\tfrac{1}{2}\right)^2 + a_3\left(\tfrac{1}{2}\right)^3 + a_4\left(\tfrac{1}{2}\right)^4$$

$$0.0 = a_0 + a_1\cancel{(1)} + a_2\cancel{(1)^2} + a_3\cancel{(1)^3} + a_4\cancel{(1)^4}$$

$$-0.95 = a_0 + a_1(1.5) + a_2(1.5)^2 + a_3(1.5)^3 + a_4(1.5)^4$$

$$\vdots$$

$$6.42 = a_0 + a_1(4.5) + a_2(4.5)^2 + a_3(4.5)^3 + a_4(4.5)^4$$

D

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & \frac{1}{2} & \left(\frac{1}{2}\right)^2 & \left(\frac{1}{2}\right)^3 & \left(\frac{1}{2}\right)^4 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1.5 & 1.5^2 & 1.5^3 & 1.5^4 \\ & & \vdots & & \\ 1 & 4.5 & 4.5^2 & 4.5^3 & 4.5^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} 24 \\ 6.61 \\ 0.0 \\ -0.95 \\ \vdots \\ 6.42 \end{bmatrix}$$

Not feasible by hand, but we can find the "$\vec{a}$" coefficients using our least-squares formula that best fits y(x) to the data!

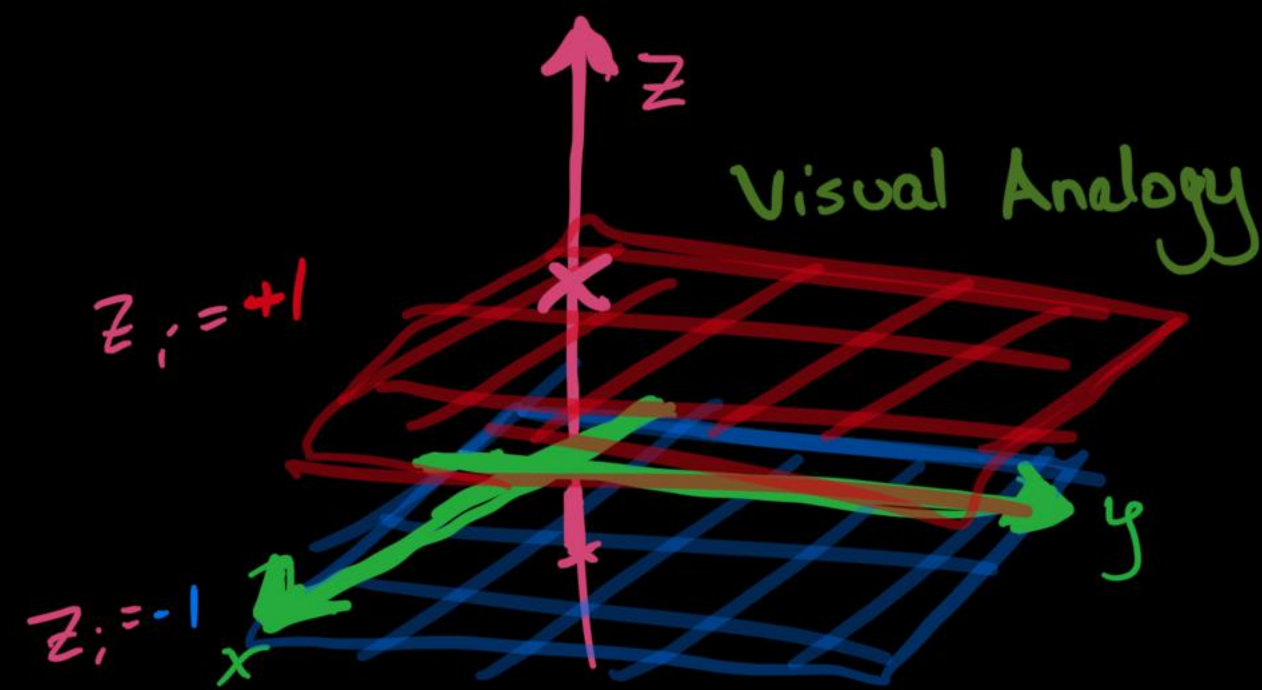
Same as $\vec{a}$ in lecture

$$\vec{a}' = (D^T D)^{-1} D^T \vec{y} \quad \checkmark$$

d] Identify the solution $\vec{a}$ using Python notebook!

$$y(x) = 24 - 50x + 35x^2 - 10x^3 + 1x^4 \qquad \vec{a} = \begin{bmatrix} 24 \\ -50 \\ 35 \\ -10 \\ 1 \end{bmatrix}$$

(See Notebook!)



② Finding Classifiers:

There are data $[x_i \; y_i]$, and for each point there is a label l_i.

x_i	y_i	l_i
-2	1	-1
-1	1	1
1	1	1
2	1	-1

(x_i, y_i, z_i)

- Okay, so there are some new terms here. To ease into the problem, you can imagine l_i as a 3rd dimension z_i. In this context we are just playing the same game as above, but with a 2D function like this: $z(x,y)$ in which we have sampled x_i and y_i at various points and would like to fit to $z(x,y)$.

- But more generally, we could expand to MANY more dimensions! Another cool context: Machine Learning on hand-writing
- Instead of x and y, let there be $p^{(1)}$ and $p^{(2)}$, and keep going $p^{(3)}, p^{(4)}, \dots, p^{(1280 \times 720)}$ where $p^{(j)}$ is the j^{th} pixel of an image.

  Now the l "label" would represent which letter is in the image. So if it's an 'a' then $l=1$, for 'b' then $l=2$, and so on.

  In summary...

$\vdots$	$\vdots$	•••	$\vdots$	$\vdots$
$P_i^{(1)}$	$P_i^{(2)}$	•••	$P_i^{(921600)}$	l_i
$\vdots$	$\vdots$	•••	$\vdots$	$\vdots$

2nd pixel value of ith image

Which letter is in the ith image.

In this context, getting a fit for $l(p^{(1)}, p^{(2)}, \dots, p^{(921600)})$ is like building scheme to predict the letter l from a brand new image!!!

Warning: Our problem here differs from machine learning because we test specific classifiers: eg. $\ell = \alpha x + \beta y + \gamma$

or

$= \alpha x + \beta x^2 + \gamma$

and find the optimal parameters given our classifiers.

Machine learning in some sense assumes far more general classifiers that can have non-linear dependence on their parameters. It also uses methods other than least-squares, you might learn these later on!!!

a] Model 1: $\ell_i = \alpha x_i + \beta y_i + \gamma$

$$\begin{aligned}
-1 &= \alpha(-2) + \beta(1) + \gamma \\
1 &= \alpha(-1) + \beta(1) + \gamma \\
1 &= \alpha(1) + \beta(1) + \gamma \\
-1 &= \alpha(2) + \beta(1) + \gamma
\end{aligned}
\quad \Rightarrow \quad
\underbrace{\begin{bmatrix} -2 & 1 & 1 \\ -1 & 1 & 1 \\ 1 & 1 & 1 \\ 2 & 1 & 1 \end{bmatrix}}_{D}
\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix}
=
\begin{bmatrix} -1 \\ 1 \\ 1 \\ -1 \end{bmatrix}$$

Parameter Vector $\vec{V}$

output vector $\vec{w}$

Least squares solution! $\longrightarrow$ $\vec{V}' = (D^T D)^{-1} D^T \vec{w}$

Oh no! D has linearly-dependent columns

This means $D^T D$ does <u>not</u> have an inverse and we can't use our formula :(.

* What's going on??

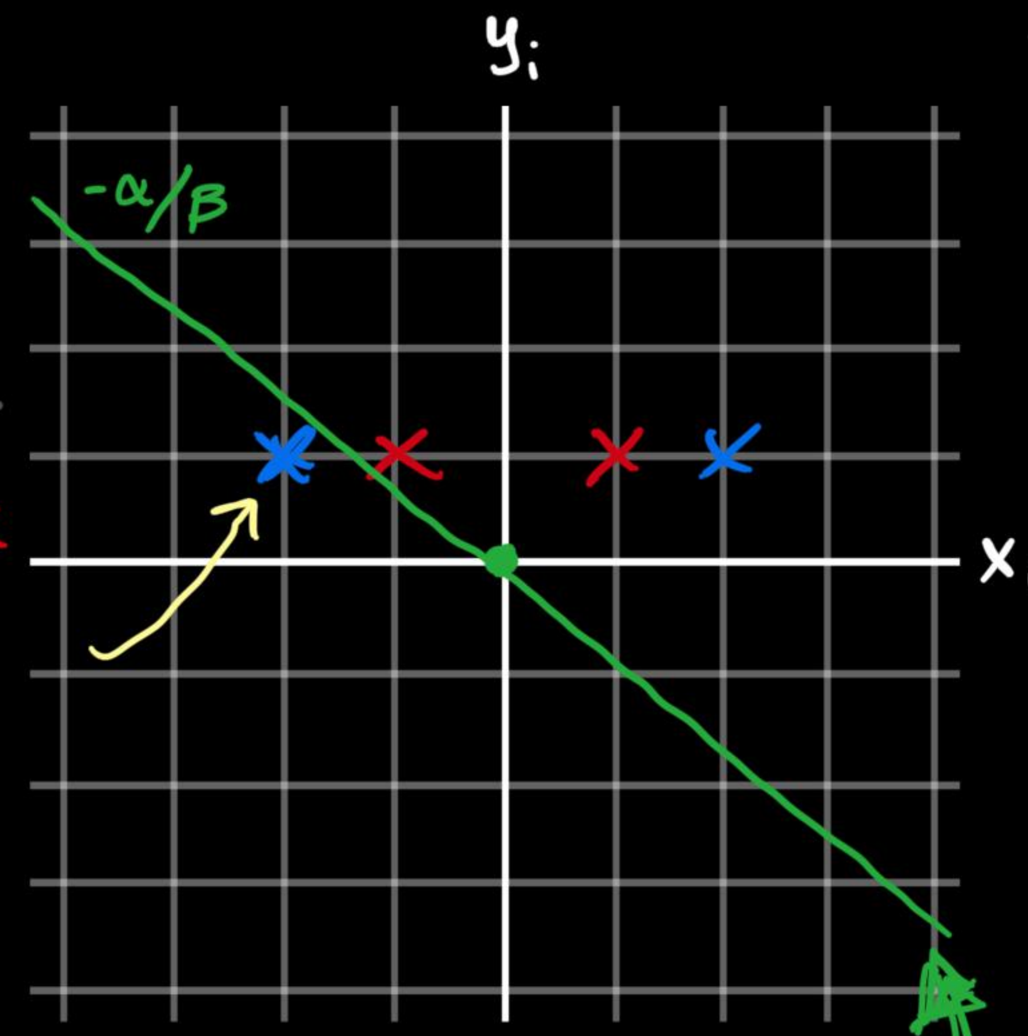
Basically β and γ have identical roles in the model due to how y_i is sampled. Thus there is <u>no unique</u> least-squares solution.

We can set $\beta = 0$ or $\gamma = 0$ in our model and then the machiny should work :) (in this case $\vec{v} = \begin{bmatrix} \alpha \\ \gamma \end{bmatrix}$ or $\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$)

b] Plot the labels here:

(See the header diagram for aid)

x_i	y_i	ℓ_i	
-2	1	-1	x
-1	1	1	x
1	1	1	
2	1	-1	



If we'd like to predict how to label new input data $[x, y] \to \ell$ as either $\ell = -1$ or $\ell = +1$, then it becomes a game of checking if $\ell < 0$ or $\ell > 0$. The dividing case of $\ell = 0$ can be visually shown in 2D: $\ell = \alpha x + \beta y \equiv 0$

$$\Downarrow$$

$$y(x) = -\left(\frac{\alpha}{\beta}\right)x$$

However, there is <u>no</u> line we can draw through the origin that parts x and x. This means our classifier is too limited to predict our current data <u>regardless</u> of the α, β parameters we choose.

We need a better classifier...

c] Model 2: $\ell_i = \alpha x_i + \beta x_i^2$

Forget y_i !!! Choose new term

$$-1 = \alpha(-2) + \beta\,\cancel{(-2)^2} \to 4$$
$$1 = \alpha(-1) + \beta\,\cancel{(-1)^2} \to 1$$
$$1 = \alpha(1) + \beta\,\cancel{(1)^2} \to 1$$
$$-1 = \alpha(2) + \beta\,\cancel{(2)^2} \to 4$$

x_i	y_i	ℓ_i	
-2	1	-1	x
-1	1	1	x
1	1	1	x
2	1	-1	x

$$\begin{bmatrix} -2 & 4 \\ -1 & 1 \\ 1 & 1 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 1 \\ -1 \end{bmatrix}$$

d] Plot the labels here:

Same game as before, but now the vertical axis is x_i^2.

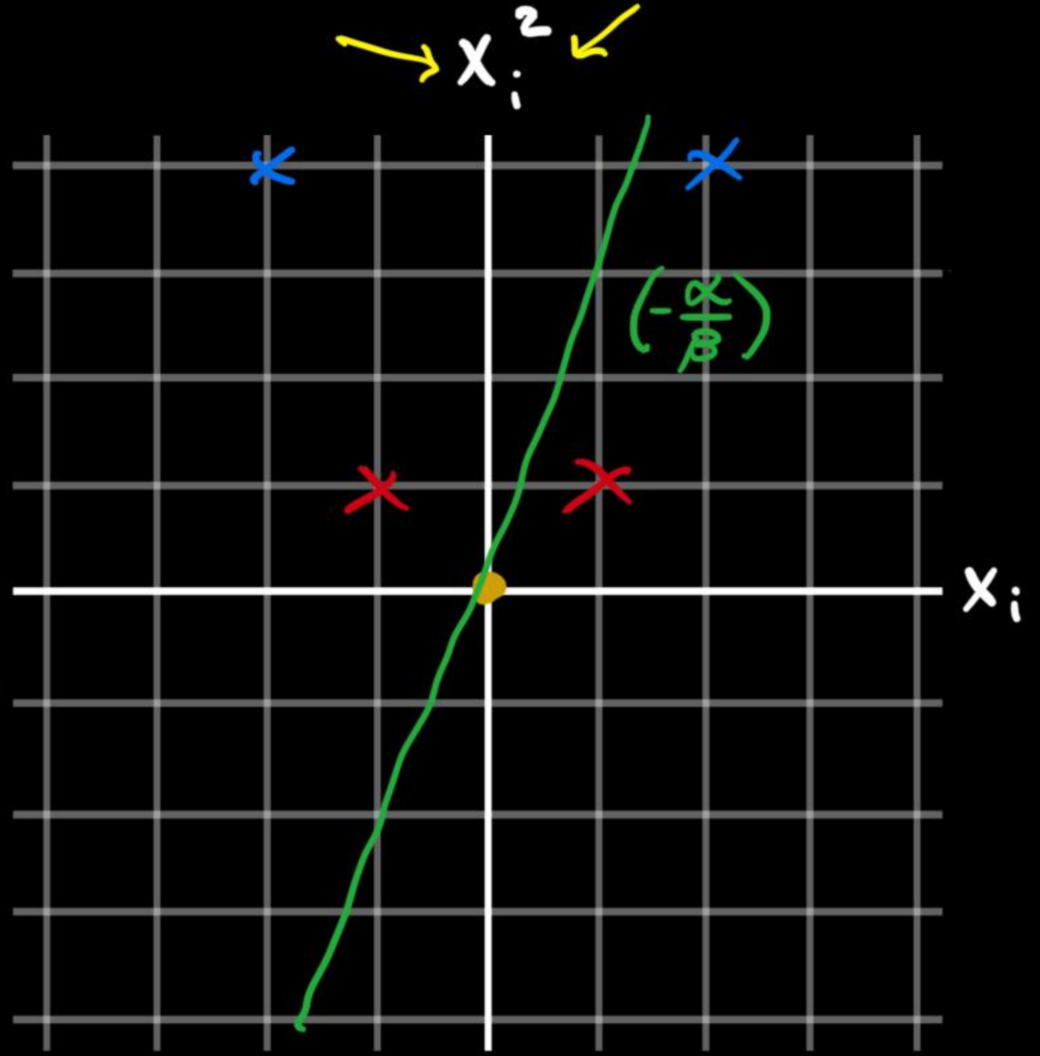
$$\ell = \alpha x + \beta q \equiv 0$$

$q = x^2$
use a different name so we can plot in 2D, although this model is only 1D.

$$\Downarrow$$

$$q = -\left(\frac{\alpha}{\beta}\right) x$$

Still an issue, :-
but we're getting closer!



It looks possible now, but not from the origin. We need to bring back γ...

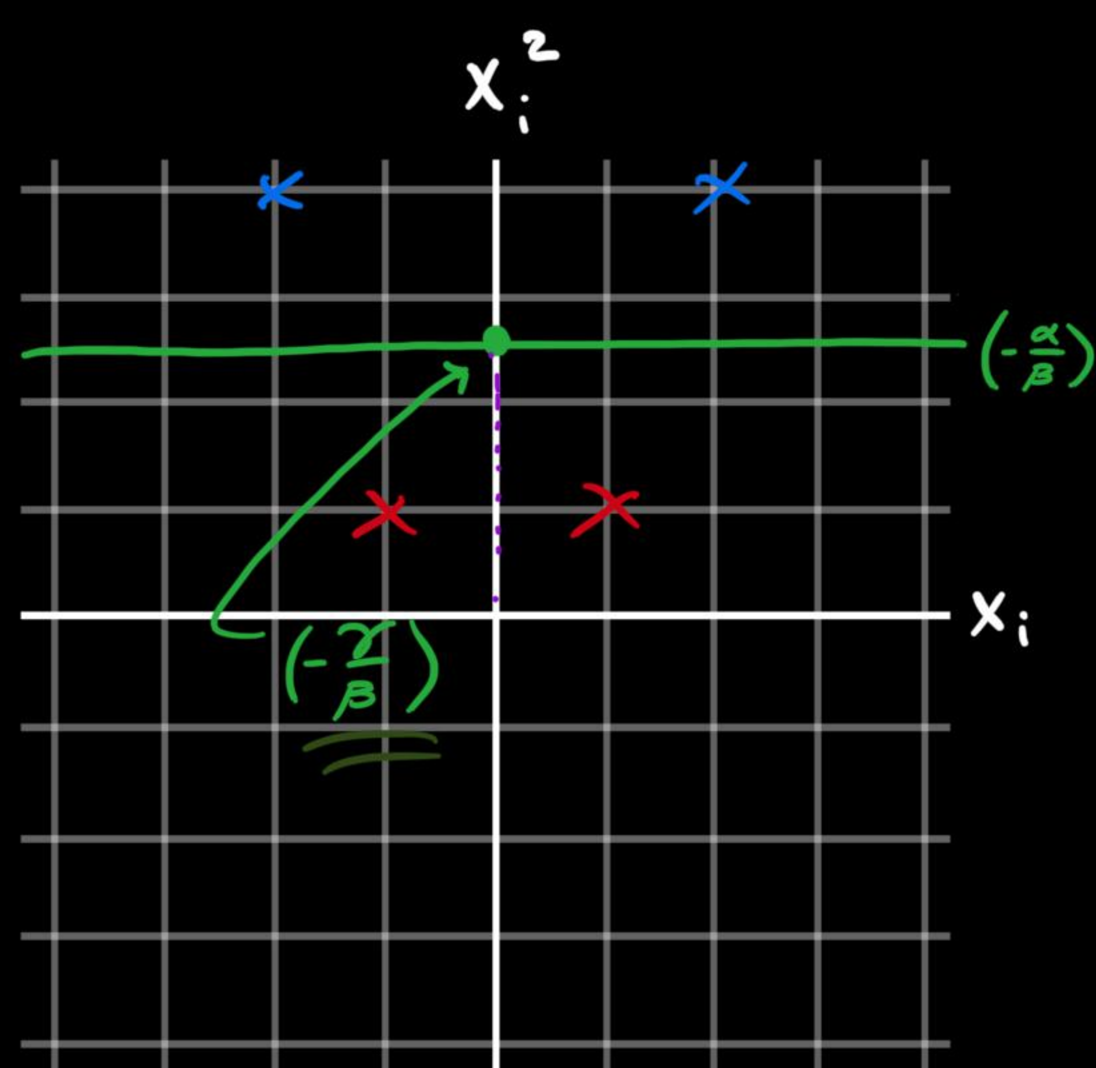
e) Model 3: $\ell_i = \alpha x_i + \beta x_i^2 + \gamma$

Do you expect better performance here than in (c)?

$$-1 = \alpha(-2) + \beta\,\cancel{(-2)^2} \rightarrow 4 + \gamma$$
$$1 = \alpha(-1) + \beta\,\cancel{(-1)^2} \rightarrow 1 + \gamma$$
$$1 = \alpha(1) + \beta\,\cancel{(1)^2} \rightarrow 1 + \gamma$$
$$-1 = \alpha(2) + \beta\,\cancel{(2)^2} \rightarrow 4 + \gamma$$

x_i	~~y_i~~	ℓ_i	
-2	~~1~~	-1	x
-1	~~1~~	1	x
1	~~1~~	1	x
2	~~1~~	-1	x

$$\begin{bmatrix} -2 & 4 & 1 \\ -1 & 1 & 1 \\ 1 & 1 & 1 \\ 2 & 4 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 1 \\ -1 \end{bmatrix}$$



$$\ell = \alpha x + \beta q + \gamma \equiv 0$$

$q = x^2$

$$q(x) = \left(-\frac{\alpha}{\beta}\right)x + \left(-\frac{\gamma}{\beta}\right)$$

Graphically there is now a clear way to parameterize our classifier so it correctly assigns labels to our current data! :)

Note: While the graph above suggests there is not a unique selection for α, β, γ (there is not, in fact), the least-squares method will give us unique parameters.

The catch here is the whole "$\ell = -1$ if $\ell < 0$ and $\ell = +1$ if $\ell > 0$" lousiness.