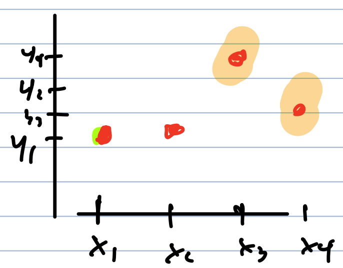
Section 6.8

More applied stuff. Will mix least-squares techniques of 6.5, 6.6 with abstract inner products from 6.7.

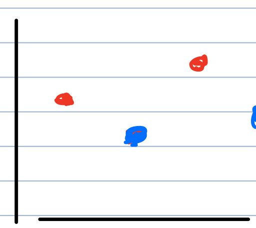
Scenario: (In $\mathbb{R}^n$)

Back to data points and finding line that best explains data:



Data may be measurements/observations.

Maybe some measurements/observations are more reliable than others.



Maybe I know somehow that blue points are very accurate. Red points not very accurate.

Shouldn't I account for that somehow when I make my least-squares line? Yes. Is that possible? Also yes

Back up a bit.

$\vec{y}$ vector of your observed data (just the y-coordinates)

$$\vec{y} = \begin{vmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{vmatrix}$$

$\hat{y}$ vector of your <u>predicted</u> y-coordinates (which we haven't found yet)

$$\hat{y} = \begin{vmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_n \end{vmatrix}$$

Recall in least squares problems want to minimize sum of squares

$$SS(E) = (y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2 + \dots (y_n - \hat{y}_n)^2$$

sum of squares error

Note that $SS(E)$ just $\|y - \hat{y}\|^2$ (using <u>usual</u> idea of length in $\mathbb{R}^n$)

We want to <u>minimize</u> error. But recall some observations more trustworthy than others

To account for this we assign <u>relative</u> weights to each output/measurement.

Look at your <u>observed</u> y-values. Lets say we are in $\mathbb{R}^5$

$$\vec{y} = \begin{vmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{vmatrix}$$

What if I know that measurements of y_3, y_5 are 3 times as reliable as rest?

$$\begin{vmatrix} y_1 \\ y_2 \\ 3y_3 \\ y_4 \\ 3y_5 \end{vmatrix}$$

I multiply (weight) these values by 3.

$$\underset{W}{\begin{vmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{vmatrix}} \underset{\vec{y}}{\begin{vmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{vmatrix}}$$

Note same vector achieved by multiplying original $\vec{y}$ by matrix W

<u>Now</u> we want to minimize <u>weighted</u> squared error

$$\text{Weighted } SS(E) = w_1^2(y_1 - \hat{y}_1)^2 + \dots + w_n^2(y_n - \hat{y}_n)^2$$

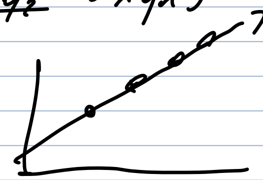
(where $w_1 \dots w_n$ are diagonal entries on W, our weights)

Note that:

$$w_1^2(y_1-\hat{y}_1)^2 + \dots + w_n^2(y_n-\hat{y}_n)^2$$

$$= (\underline{w_1 y_1} - \underline{w_1 \hat{y}_1})^2 + \underline{\dots} \; (\underline{w_n y_n} - w_n\hat{y}_n)^2$$

$$= \| W\vec{y} - W\hat{y} \|^2$$



Remember, ideally our observed data would be exactly equal to our predicted data

$$\hat{y} = \vec{y}$$

And our predicted data is linear (i.e. $\hat{y} = Ax$ for our known inputs)

$$X\vec{\beta} = \hat{y} = \vec{y}$$

If all these are equal, should have

$$\bigstar \; WX\vec{\beta} = W\hat{y} = W\vec{y}$$

(our weighted system) $\underbrace{W}_{\text{matrix}} X \vec{\beta} = W\vec{y}$ ← weighted observations

vector that gives line

But again, in real situation these things won't be equal. Know want to find least squares solution. But solution to weighted system.

$$(WX)\vec{\beta} = W\vec{y} \qquad \text{(No solution)}$$

$$(WX)^T WX\vec{\beta} = (WX)^T W\vec{y} \qquad \text{Least squares system}$$

/

From there, solve normally. The solution / line we find will be a line that favors certain points more than others, according to relative weights we have given.

Problem break

Steps for weighted problems

Have data points $(x_1, y_1) \dots (x_n, y_n)$

① Make your usual set-up from 6.6.

$$[X \mid \vec{y}] = \left[\begin{array}{cc|c} 1 & x_1 & y_1 \\ 1 & x_2 & \vdots \\ \vdots & \vdots & \vdots \\ 1 & x_n & y_n \end{array}\right]$$

design matrix

y-values of data

② Decide on relative weights

Ex If y_3, y_4 are 1.5 times as important as rest then

$w_3, w_4 = 1.5$

and rest of w's $= 1$

③ Make diagonal matrix W with weights on diagonal

④ Multiply everything by W on the left

$$\left[\, WX \;\middle|\; W\vec{y} \,\right] = \left[\; \underbrace{\begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & & \\ \vdots & & \ddots & \\ & & & w_n \end{bmatrix}}_{W} \underbrace{\begin{bmatrix} 1 & x_1 \\ 1 & \vdots \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}}_{X} \;\middle|\; \underbrace{\begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & & \\ \vdots & & \ddots & \\ & & & w_n \end{bmatrix}}_{W} \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}}_{\vec{y}} \;\right]$$

$$\downarrow \qquad \downarrow$$

$$\left[\, B \;\middle|\; \vec{c} \,\right] \rightarrow \left[\, B^T B \;\middle|\; B^T \vec{c} \,\right]$$

⑤ Now we have new system, still doesn't have exact solution

So now solve using Least Squares from 6.5/6.6

⑥ Results/solutions still be β_0, β_1 giving you weighted line

$$y = \beta_0 + \beta_1 x$$

Machine Learning

Neural networks etc.

start with system of data points as above

Pick/guess at weights data. Solve your system to give you line/function that explains the data.

Then you have algorithm that takes in new data, re-evaluates/updates your weights, solves again for a <u>new</u> function.

<u>Connection to inner products:</u>

Recall we wanted to minimize weighted squared error $\|Wy - W\hat{y}\|^2$ ☆

$$= (w_1 y_1 - w_1 \hat{y}_1)^2 + (w_2 y_2 - w_2 \hat{y}_2)^2 + \dots$$

$$\langle y-\hat{y}, y-\hat{y}\rangle \; ☆ \; = w_1^2 (y_1 - \hat{y}_1)^2 + w_2^2 (y_2 - \hat{y}_2) + \dots$$

Recall one of inner products on $\mathbb{R}^n$ we saw, for chosen positive numbers $w_1^2 \dots w_n^2$

$$\langle u, v\rangle = w_1^2 u_1 v_1 + w_2^2 u_2 v_2 + \dots w_n^2 u_n v_n$$

Then ☆ is exactly:

$$\langle y-\hat{y}, y-\hat{y}\rangle = w_1^2 (y_1 - \hat{y}_1)^2 + \dots$$

The weights (squared) give us this inner product on $\mathbb{R}^n$, and can view our goal as trying to minimize the inner product

Skip Trend Analysis because it's not explained well.

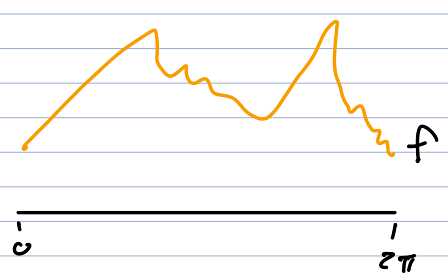
Next Application: Fourier Series!

May have mentioned we can approximate any continuous function on closed interval with polynomials.

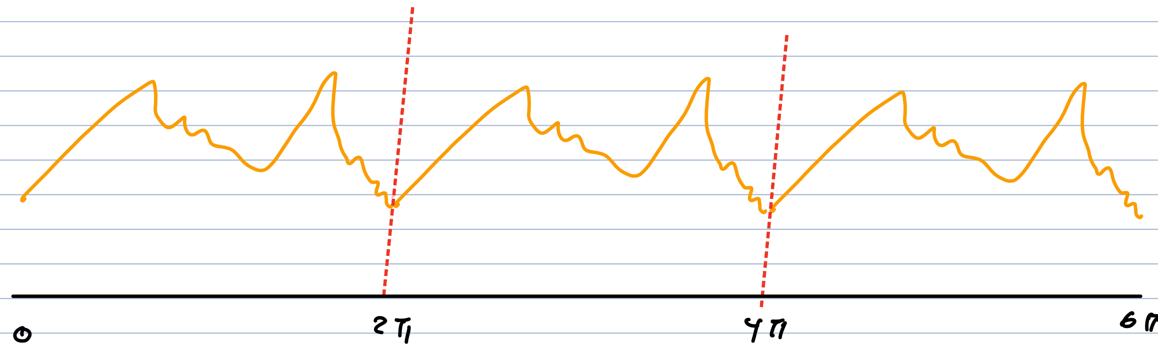
Instead of polynomials, could also use sin, cos functions such as sin(t), sin(2t), cos(4t), etc.

This is great for physics which deals with a lot of different waves.

Waves usually repeat, so it is enough to consider a closed interval. For simplicity, let it be $[0, 2\pi]$



Audio signal



If we can approximate it on $[0, 2\pi]$ we have whole wave.

Assume function that <u>really</u> describes wave on $[0, 2\pi]$ is f.

Will approximate signal on $[0, 2\pi]$ using trigonometric polynomial, something of form

$$\frac{a_0}{2}(1) + \underbrace{a_1 \cos(t) + a_2 \cos(2t) + \dots a_n \cos(nt)}_{\text{Cosines}} + \underbrace{b_1 \sin(t) + \dots \, b_n \sin(nt)}_{\text{sines}}$$

(Span of)
Trig functions $\{1, \sin(t), \cos(t), \sin(2t), \cos(2t) \dots \cos(nt)\}$
form subspace of $\underline{C[0, 2\pi]} \leftarrow$

Every trig polynomial (up to order n) is in this subspace

So if we want to approximate f with one of these trig polynomials, should choose best approximation, aka projection of f onto subspace

$\text{proj}_W f$ where $W = \{1, \sin(t), \cos(t), \dots \cos(nt)\}$

To do this, need basis of subspace to be orthogonal.

Great! $\{1, \sin(t), \cos(t), \dots \sin(nt), \cos(nt)\}$ are all orthogonal to each other on $[0, 2\pi]$; w.r.t the L^2 inner product

Assume projection of f (signal's true function) on to W (subspace of trig functions) is:

$\text{proj}_W f$

$$\frac{a_0}{2} + a_1 \cos(t) + a_2 \cos(2t) + \dots a_n \cos(nt) + b_1 \sin(t) + \dots b_n \sin(nt)$$

Best Approximation of f onto subspace W is $\text{proj}_W f$

If $\{1, \cos(t), \dots \cos(nt), \sin(t), \dots \sin(nt)\}$

$$\text{proj}_W f = \underline{\text{proj}_1 f} + \underline{\text{proj}_{\cos t} f} + \dots \text{proj}_{\sin(nt)} f$$

$$= \frac{a_0}{2}(1) + a_1 \cos(t) + \dots \quad b_n \sin(nt)$$

$$= \frac{\langle f, 1 \rangle}{\langle 1, 1 \rangle} 1 + \frac{\langle f, \cos(t) \rangle}{\langle \cos(t), \cos(t) \rangle} \cos(t) + \dots$$

$$= \frac{\int_0^{2\pi} (f) 1 \, dx}{\int_0^{2\pi} 1 \, dx} (1) + \frac{\int_0^{2\pi} f \cos(x) \, dx}{\int_0^{2\pi} \cos(x) \cos(x) \, dx} \cos(x) + \dots$$

Usually we know f is but is super complicated, approximate it with sin, cos to make it simple, store on disc.