$$y = mx + b$$

Is My Favorite One-Liner

EECS 16A

Machine learning techniques

Last time: Least Squares

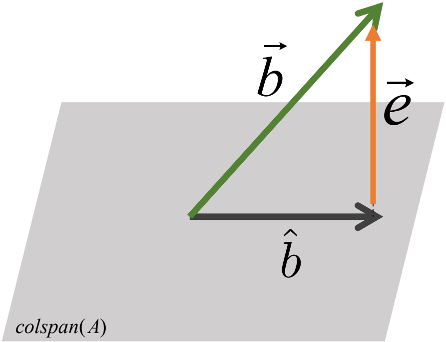
works for:

n > m

overdetermined

(or square n=m)

$$\underset{n\times m}{A}\ \underset{m\times 1}{x} = \underset{n\times 1}{b}$$



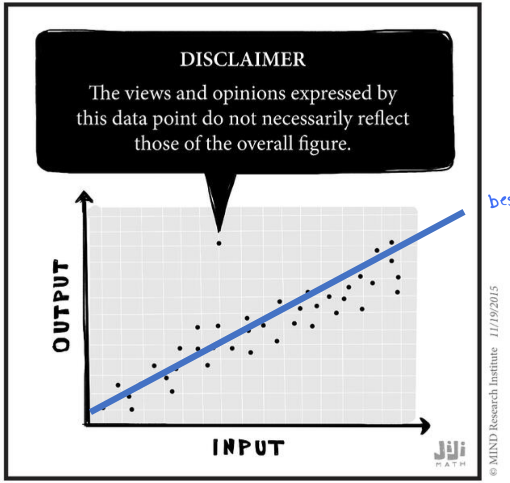
- the least-squares solution "minimally perturbs" b

$$\hat{x} = \underbrace{(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T}_{\text{"pseudo-inverse"}}\vec{b}$$

Least squares is a building block for all signal processing/machine learning!

How do we know when our model is good?

How to avoid fitting to noise?

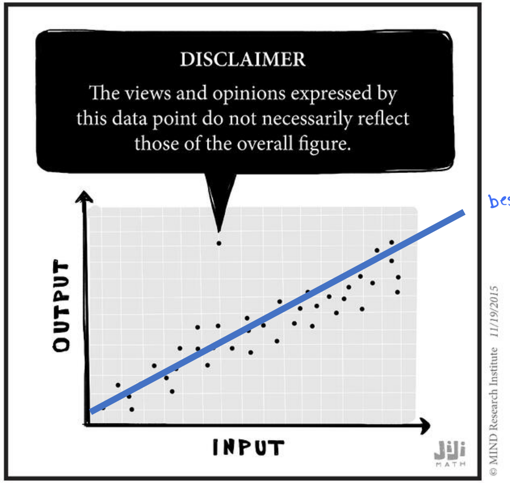


best fit line

$y = \hat{m}x + \hat{c}$

How do we know when our model is good?

How to avoid fitting to noise?



best fit line
$y = \hat{m}x + \hat{c}$

$$
\begin{array}{ccccc}
A & \vec{w} & & \vec{b} \\
\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \\ \vdots & \\ x_n & 1 \end{bmatrix} & \begin{bmatrix} m \\ c \end{bmatrix} & = & \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} \\
n\times 2 & 2\times 1 & & n\times 1
\end{array}
$$

solve for me!

Best estimate for $\hat{\vec{w}} = (A^T A)^{-1} A^T \vec{b}$ least squares sol'n

Demo: 'training-test-lecture-demo.ipynb'

Demo: 'training-test-lecture-demo.ipynb'

↳ fit data to a line, look at norm of errors

↳ fit to higher-order polynomials, but when to stop?

↳ Let's split data into 3 bins: 1) training data - used to fit model (e.g. to find A matrix)

2) test data - report performance on this

often forgotten → 3) validation data - to check model, compute cost on this

Machine Learning: Using data to understand the real world (e.g. making a model)

How do we know whether a model is any good? → testing + validation

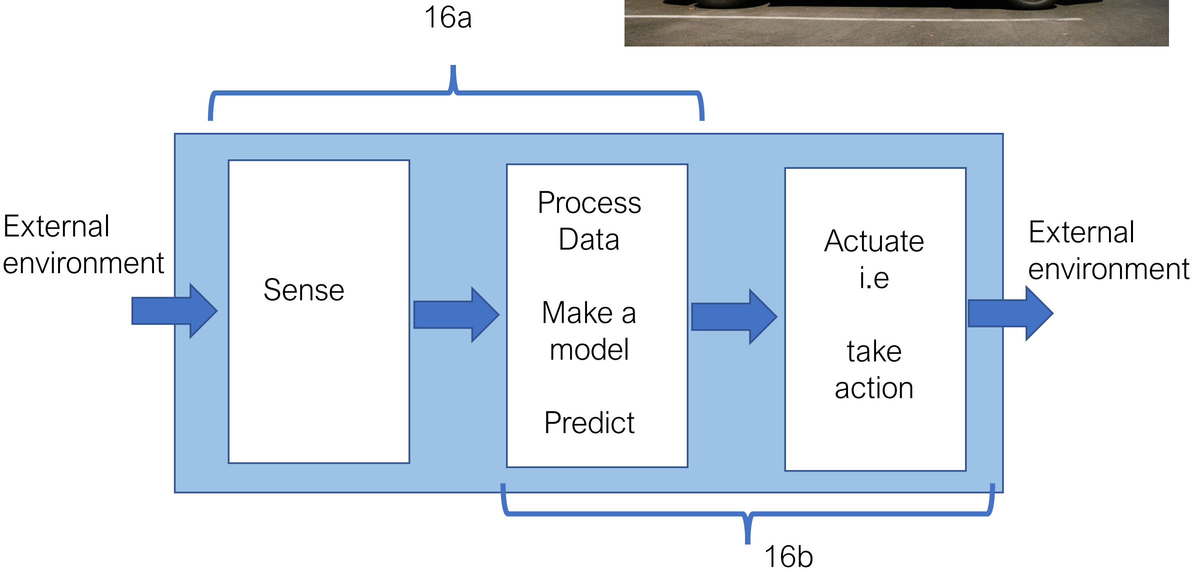
↳ ways a model can go bad: → flaw in setup (e.g. took wrong data)

→ flaw in assumptions (e.g. wrong physics model)

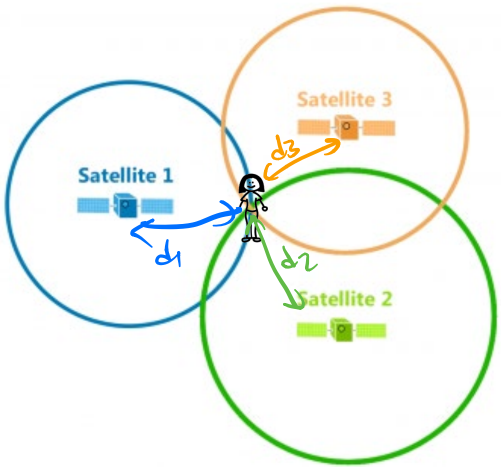
Machine learning

- Using data to understand the real world (e.g. make a model!)
- Figure out whether model is any good (testing and validation)
  - Models can be bad either because of a flaw in setup (e.g. wrong data) or a flaw in assumptions (e.g. wrong physics model)
- To learn more, take:
  - EECS16B
  - EECS127 Optimization
  - CS188
  - CS189

Example application: self-driving cars

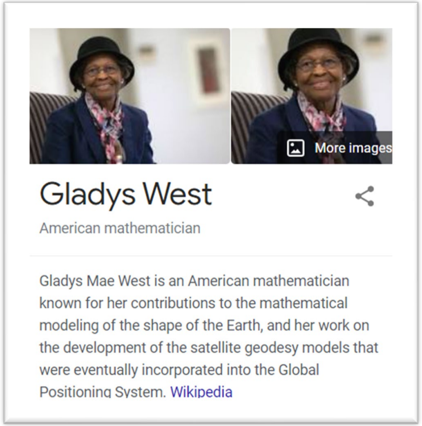


Back to GPS (Global Positioning System)



Find my coordinates $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

from known satellite positions

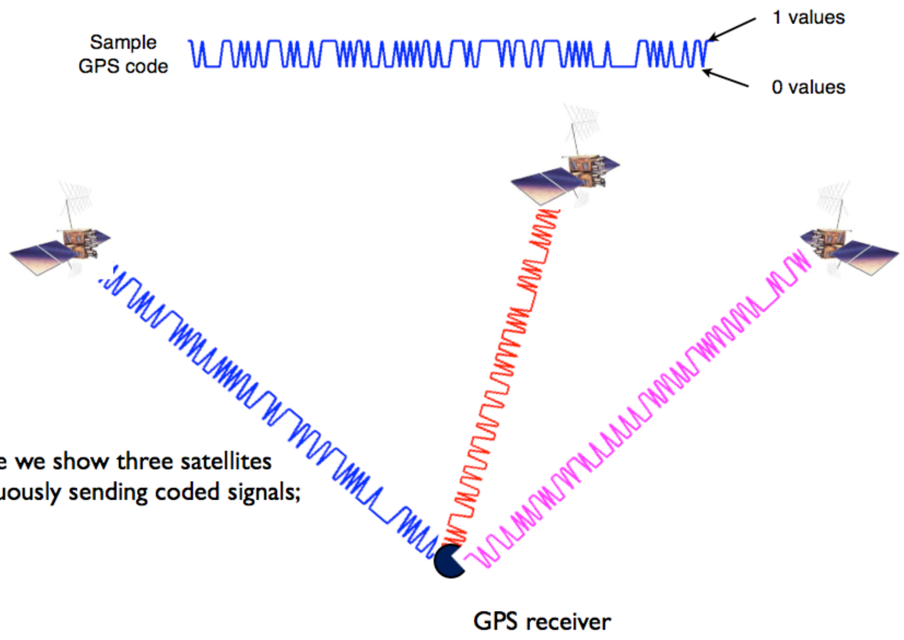


Which signals make good songs?

- Shifted versions of self are not very correlated
- Songs for each satellite/beacon are not very correlated

Can I achieve this with just 1’s and 0’s?

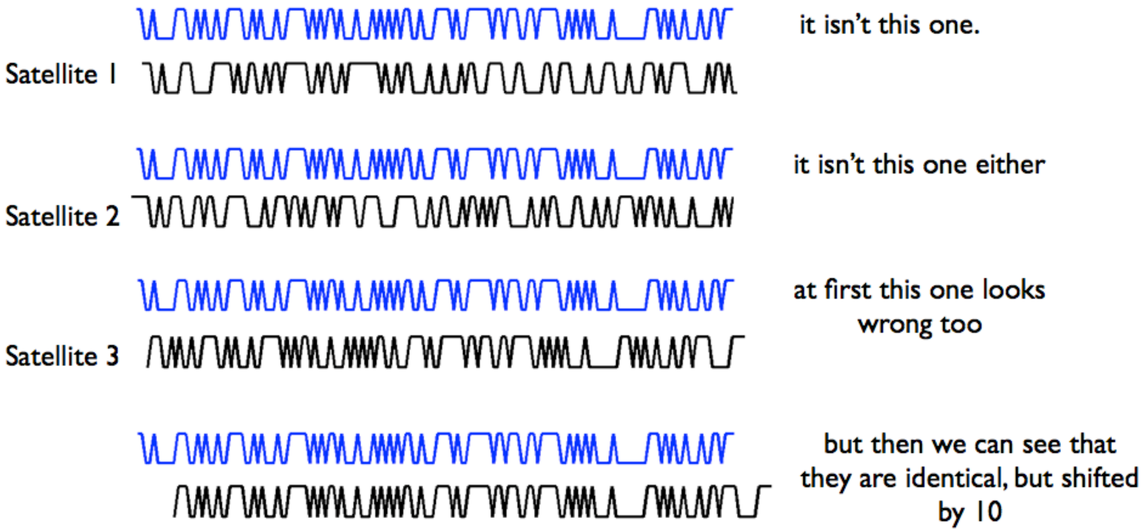
Example of binary GPS "songs"



https://spotlight.unavco.org/how-gps-works/gps-basics/decoding-the-gps-signal.html

Example of binary GPS "songs"

Here the receiver compares the blue coded signal to all the known codes.

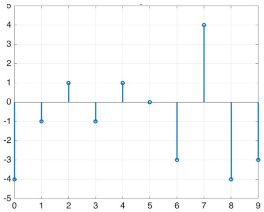


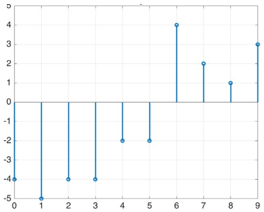
https://spotlight.unavco.org/how-gps-works/gps-basics/decoding-the-gps-signal.html satellites / beacons

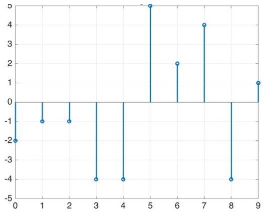
signal / signature

3 transmitters 'on', each with a 'song'

Example:







$$\vec{s}_1 = \begin{bmatrix} -4 \\ -1 \\ 1 \\ -1 \\ 1 \\ 0 \\ -3 \\ 4 \\ -4 \\ -3 \end{bmatrix}$$

song for satellite 1

$$\vec{s}_2 = \begin{bmatrix} -4 \\ -5 \\ -4 \\ -4 \\ -2 \\ -2 \\ 4 \\ 2 \\ 1 \\ 3 \end{bmatrix}$$

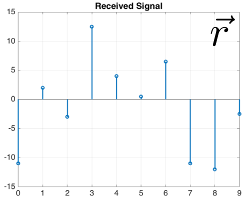
song for satellite 2

$$\vec{s}_3 = \begin{bmatrix} -2 \\ -1 \\ -1 \\ -4 \\ -4 \\ 5 \\ 2 \\ 4 \\ -4 \\ 1 \end{bmatrix}$$

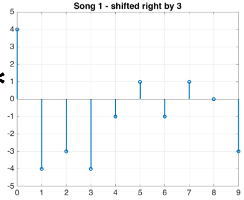
song for satellite 3

Receiver sees sum of weighted, shifted songs

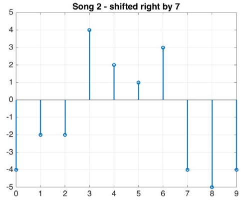
the weights on songs correspond to 'volume' at reciever



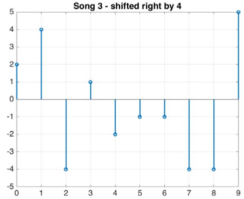
$=$ -1*



$+$ 2*



$+$ 0.5*



measurement

$$\vec{r} = \alpha_1 \vec{s}_1[n-k_1] + \alpha_2 \vec{s}_2[n-k_2] + \alpha_3 \vec{s}_3[n-k_3] + \vec{n}$$

weights shifts things to solve for

$$\underbrace{\begin{bmatrix} \vec{r} \end{bmatrix}}_{10\times 1} = \underbrace{\begin{bmatrix} \vec{s}_1[n-k_1] & s_2[n-k_2] & s_3[n-k_3] \end{bmatrix}}_{10\times 3} \underbrace{\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}}_{3\times 1} + \underbrace{\begin{bmatrix} \vec{n} \end{bmatrix}}_{10\times 1}$$

Now it's in the form $\vec{r} = A\vec{x} + \vec{n}$

From $\vec{r}$ we want to find which songs were received, how they were shifted, and their weights.

How to solve for GPS coordinates:

1. ***Identify*** which satellites are 'on'

Identify which satellites are transmitting

↳ calculate cross-correlations of $\vec{r}$ for each satellite signal $\{\vec{s}_1, \vec{s}_2, \vec{s}_3, \vec{s}_4\}$

↳ peaks in cross-correlations are used to detect which satellites are 'on'

2. Find the ***delay/shift*** for each satellite

Find the delay/shift for each one

3. Use shifts to find ***distances*** to each satellite

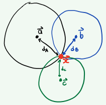
Use shifts to find distances to each satellite

↳ convert shift/delay to time: Ex: time = 2 sample shift × 2 seconds/sample = 2s

↳ distance = velocity × time

4. ***Trilateration*** to find my coordinates

Trilateration find coordinates $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ from distances d_A, d_B, d_C to 3 sattelites with positions $\vec{a}, \vec{b}, \vec{c}$



$$\|\vec{x}-\vec{a}\|^2 = d_A^2$$
$$\|\vec{x}-\vec{b}\|^2 = d_B^2$$
$$\|\vec{x}-\vec{c}\|^2 = d_C^2$$

Distances → Position

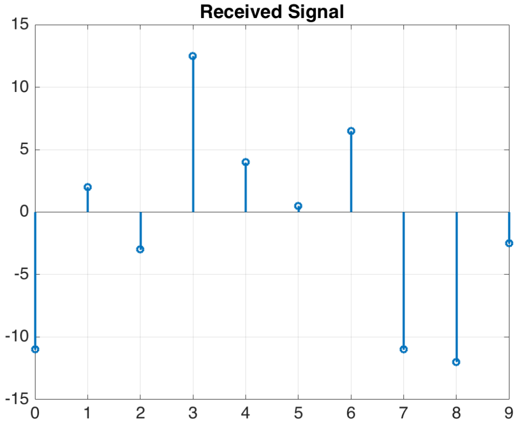
$$\underbrace{\begin{bmatrix} -2a_1+2b_1 & -2a_2+2b_2 \\ -2a_1+2c_1 & -2a_2+2c_2 \end{bmatrix}}_{\substack{\text{knowns} \\ A}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{\substack{\text{to solve for} \\ \vec{x}}} = \underbrace{\begin{bmatrix} d_A^2 - d_B^2 - \|\vec{a}\|^2 + \|\vec{b}\|^2 \\ d_A^2 - d_C^2 - \|\vec{a}\|^2 + \|\vec{c}\|^2 \end{bmatrix}}_{\substack{\text{knowns} \\ \vec{b}}}$$

Linear system

to estimate position use least squares: $\hat{x} = (A^T A)^{-1} A^T \vec{b}$ ← Pseudo-inverse

The things we know

The received signal (r)

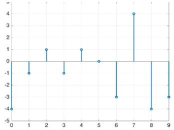


$$\vec{r} = \begin{bmatrix} -11 \\ 2 \\ -3 \\ 12 \\ 4 \\ 0.5 \\ 6.5 \\ -11 \\ -12 \\ -2.5 \end{bmatrix}$$

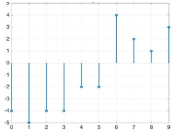
Sparsity level, k

In this example, k=3

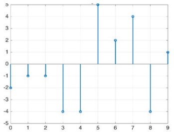
All possible songs



song 1

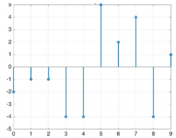


song 2



song 3

⋮



song m

The things we want to know

weights

shifts

$$\vec{r} = \alpha_1 \vec{s}_1[n - k_1] + \alpha_2 \vec{s}_2[n - k_2] + \alpha_3 \vec{s}_3[n - k_3] + \alpha_4 \vec{s}_4[n - k_4] + \ldots + \alpha_m \vec{s}_m[n - k_m] + \vec{n}$$

→ go to pptx. slides for animations

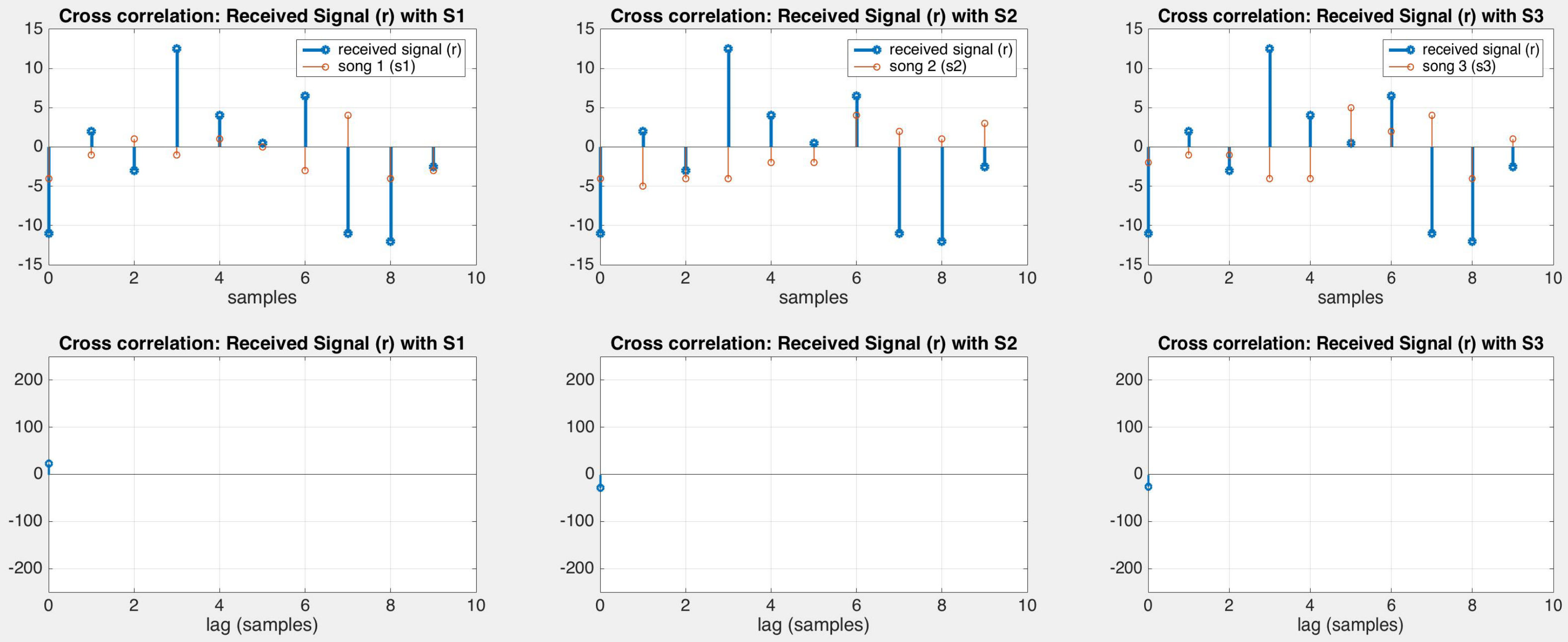
New fancy algorithm:

Orthogonal Matching Pursuit (OMP)

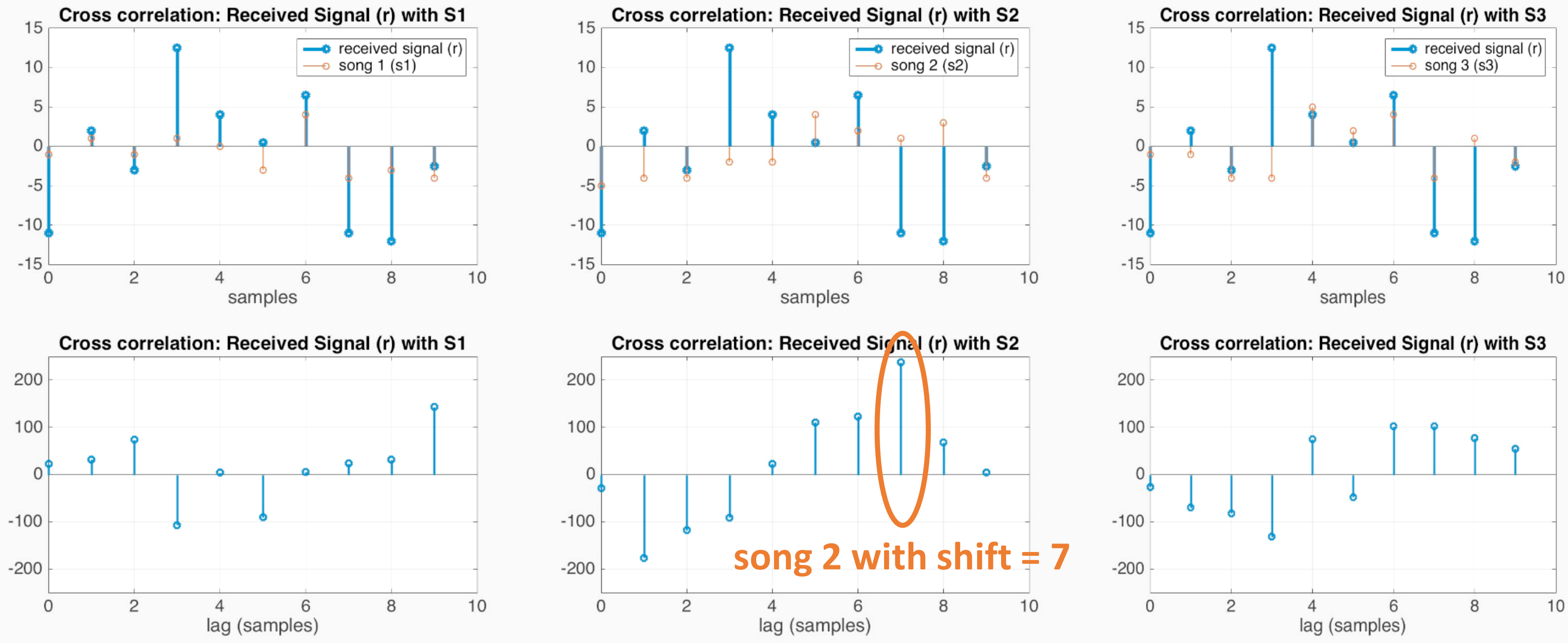
Iterative: will go through same process many times

Start with iteration 1…

Cross-correlate $\vec{r}$ with all songs



Find song/shift combo with max correlation



What's the best approx. of r with only $\vec{s}_2[n-7]$?

song 2 with shift = 7

$\vec{s}_2[n-7]$

received signal

$$\begin{bmatrix} -4 \\ -2 \\ -2 \\ 4 \\ 2 \\ 1 \\ 3 \\ -4 \\ -5 \\ -4 \end{bmatrix} \alpha_2 = \begin{bmatrix} -11 \\ 2 \\ -3 \\ 12.5 \\ 4 \\ 0.5 \\ 6.5 \\ -11 \\ -12 \\ -2.5 \end{bmatrix}$$

A x = r

What's the best approx. of r with only $\vec{s}_2[n-7]$?

song 2 with shift = 7

$\vec{s}_2[n-7]$

$$\begin{bmatrix} -4 \\ -2 \\ -2 \\ 4 \\ 2 \\ 1 \\ 3 \\ -4 \\ -5 \\ -4 \end{bmatrix} \alpha_2 = \begin{bmatrix} -11 \\ 2 \\ -3 \\ 12.5 \\ 4 \\ 0.5 \\ 6.5 \\ -11 \\ -12 \\ -2.5 \end{bmatrix}$$

received signal

A x = r

Use least squares to find the weight:

$\alpha_2 = (A^T A)^{-1} A^T \vec{r}$

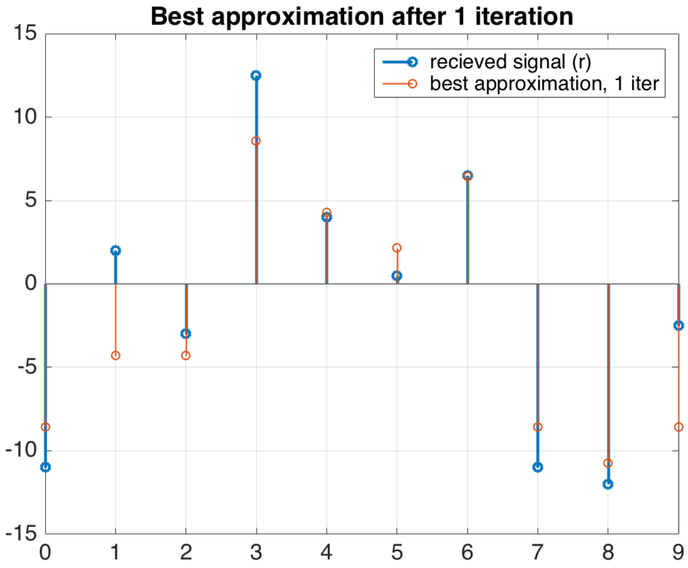
= 2.14 ← good first guess

(the actual coefficient was 2)

How did we do? Find best approx. to r

$\vec{s}_2[n-7]$

$$\hat{\vec{r}} = A\hat{x} = \begin{bmatrix} -4 \\ -2 \\ -2 \\ 4 \\ 2 \\ 1 \\ 3 \\ -4 \\ -5 \\ -4 \end{bmatrix} 2.14$$



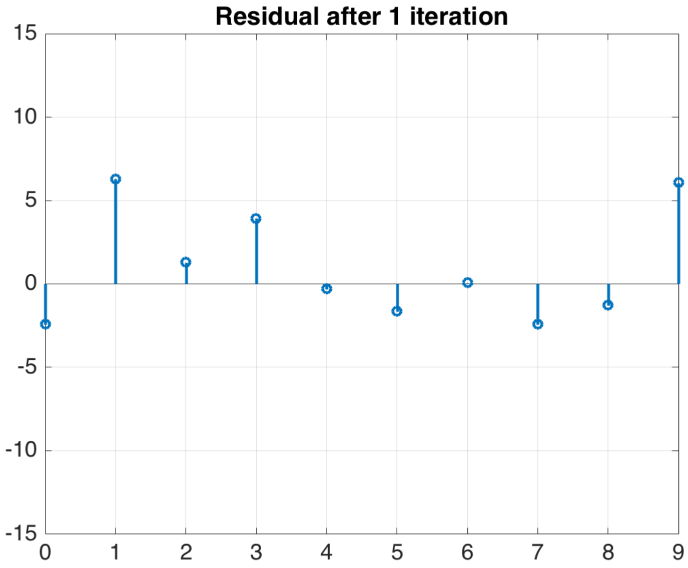
Calculate the residual error

received signal

best approximation of received signal

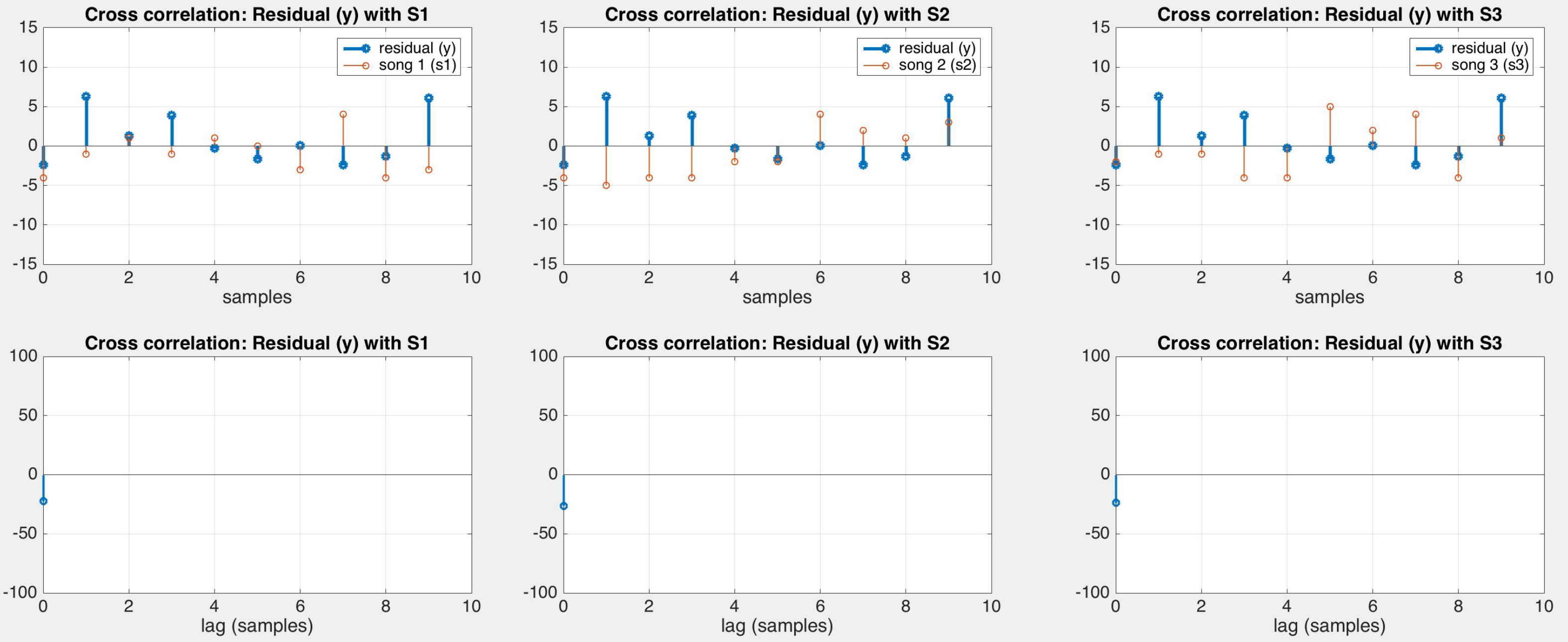
Residual: $\vec{y} = \vec{r} - \hat{\vec{r}}$

$$= \vec{r} - A\vec{x}$$



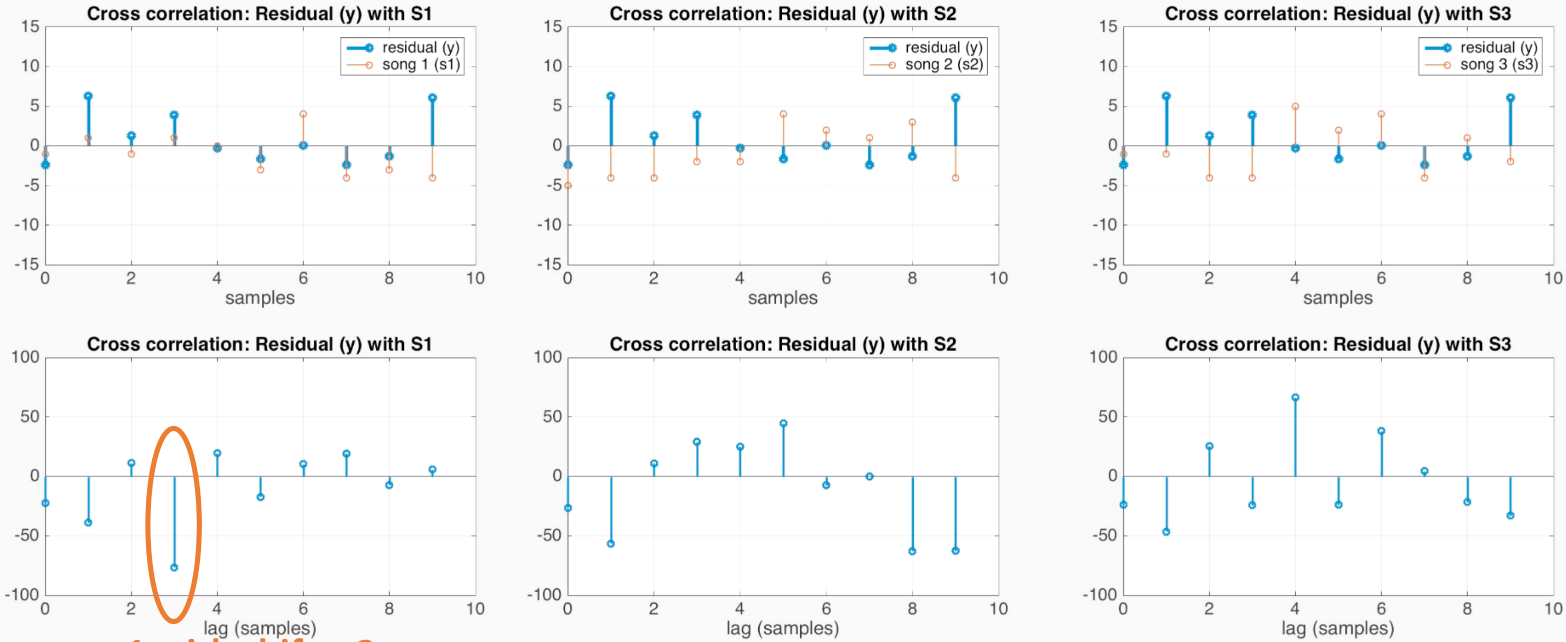
Rinse and repeat (iteration 2)

Cross-correlate $\vec{y}$ with all songs



Find song/shift combo with max correlation

note we really care abt (absolute value)



song 1 with shift = 3

↳ negative peak suggests $a_1 < 0$

What's the best approx. of r with only $\vec{s}_2[n-7]$, $\vec{s}_1[n-3]$?

$$\begin{array}{c} \text{song 2 with shift = 7} \quad \text{song 1 with shift = 3} \\ \underbrace{\begin{bmatrix} -4 & 4 \\ -2 & -4 \\ -2 & -3 \\ 4 & -4 \\ 2 & -1 \\ 1 & 1 \\ 3 & -1 \\ -4 & 1 \\ -5 & 0 \\ -4 & -3 \end{bmatrix}}_{\mathbf{A}} \end{array} \underbrace{\begin{bmatrix} \alpha_2 \\ \alpha_1 \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} -11 \\ 2 \\ -3 \\ 12.5 \\ 4 \\ 0.5 \\ 6.5 \\ -11 \\ -12 \\ -2.5 \end{bmatrix}}_{\mathbf{r}} \quad \text{received signal}$$

A **x =** **r**

What's the best approx. of r with only $\vec{s}_2[n-7]$, $\vec{s}_1[n-3]$?

$$\begin{array}{cc} \text{song 2 with shift = 7} & \text{song 1 with shift = 3} \end{array}$$

$$\underbrace{\begin{bmatrix} -4 & 4 \\ -2 & -4 \\ -2 & -3 \\ 4 & -4 \\ 2 & -1 \\ 1 & 1 \\ 3 & -1 \\ -4 & 1 \\ -5 & 0 \\ -4 & -3 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \alpha_2 \\ \alpha_1 \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} -11 \\ 2 \\ -3 \\ 12.5 \\ 4 \\ 0.5 \\ 6.5 \\ -11 \\ -12 \\ -2.5 \end{bmatrix}}_{\mathbf{r}}$$

received signal

Use least squares to find the weights:

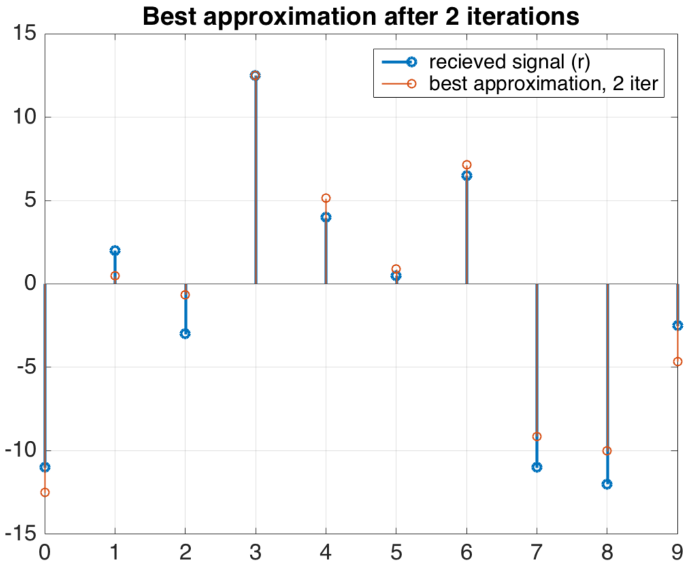
$$\vec{x} = (A^T A)^{-1} A^T \vec{r}$$

$$= \begin{bmatrix} 2.00 \\ -1.12 \end{bmatrix} \begin{array}{l} \leftarrow \text{correct!} \\ \leftarrow \text{some error} \end{array}$$

(the actual coefficients were 2,-1)

How did we do? Find best approx. to r

$$\hat{\vec{r}} = A\hat{x} = \begin{array}{c} \begin{array}{cc} \text{song 2 with} & \text{song 1 with} \\ \text{shift = 7} & \text{shift = 3} \end{array} \\ \begin{bmatrix} -4 & 4 \\ -2 & -4 \\ -2 & -3 \\ 4 & -4 \\ 2 & -1 \\ 1 & 1 \\ 3 & -1 \\ -4 & 1 \\ -5 & 0 \\ -4 & -3 \end{bmatrix} \end{array} \begin{bmatrix} 2.00 \\ -1.12 \end{bmatrix}$$



Calculate the residual

received signal

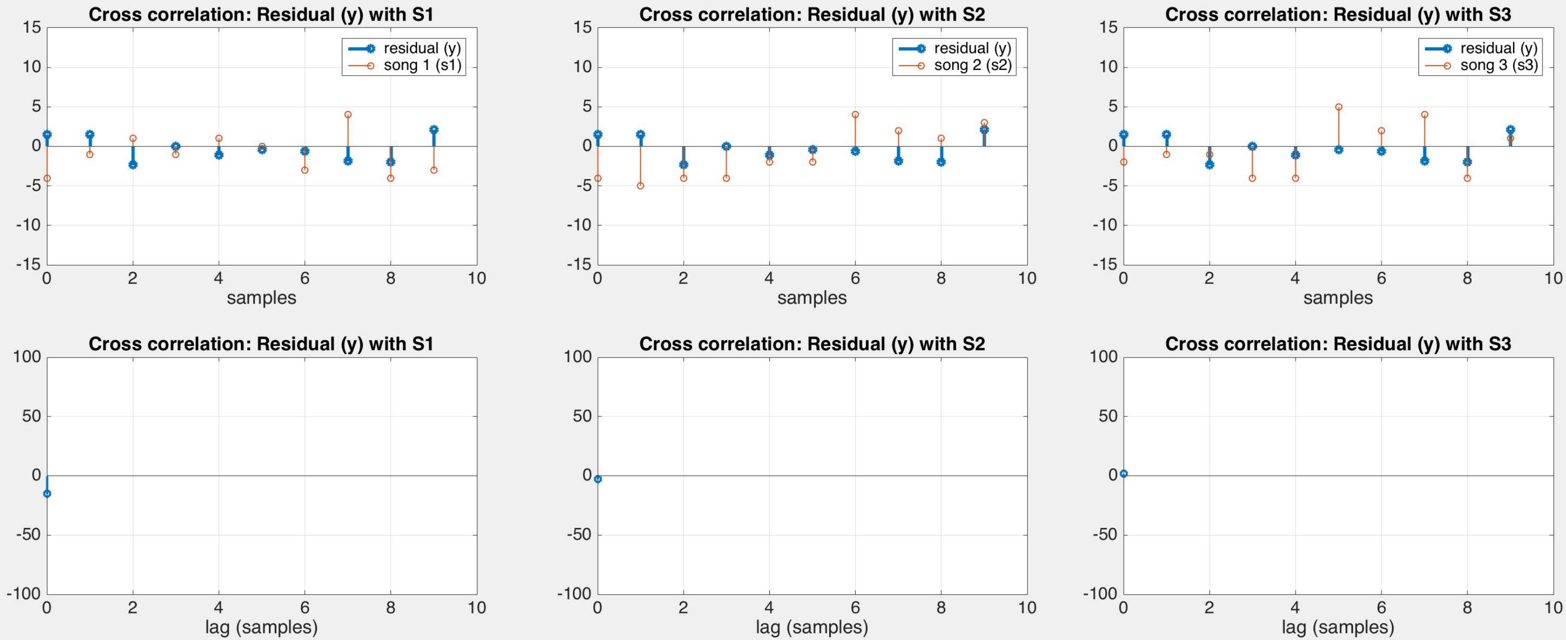
best approximation of received signal

Residual: $\vec{y} = \vec{r} - \hat{\vec{r}}$

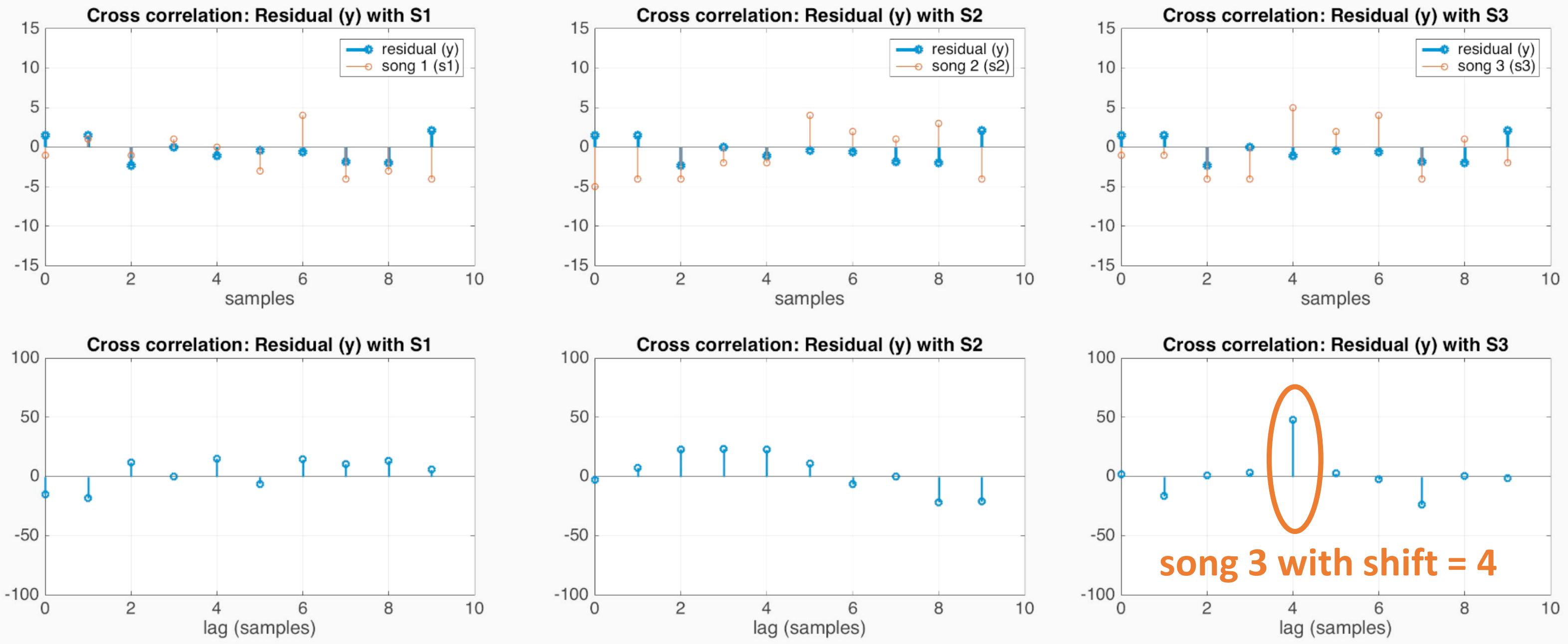
$= \vec{r} - A\vec{x}$

Rinse and repeat (iteration 3)

Cross-correlate $\vec{y}$ with all songs



Find song/shift combo with max correlation



What's the best approx. of r with $\vec{s}_2[n-7]$, $\vec{s}_1[n-3]$, $\vec{s}_3[n-4]$?

$$\underbrace{\begin{bmatrix} -4 & 4 & 2 \\ -2 & -4 & 4 \\ -2 & -3 & -3 \\ 4 & -4 & 1 \\ 2 & -1 & -2 \\ 1 & 1 & -1 \\ 3 & -1 & -1 \\ -4 & 1 & -4 \\ -5 & 0 & -4 \\ -4 & -3 & 5 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} -11 \\ 2 \\ -3 \\ 12.5 \\ 4 \\ 0.5 \\ 6.5 \\ -11 \\ -12 \\ -2.5 \end{bmatrix}}_{\mathbf{r}}$$

received signal

A x = r

Use least squares to find the weights:

$$\vec{x} = (A^T A)^{-1} A^T \vec{r}$$

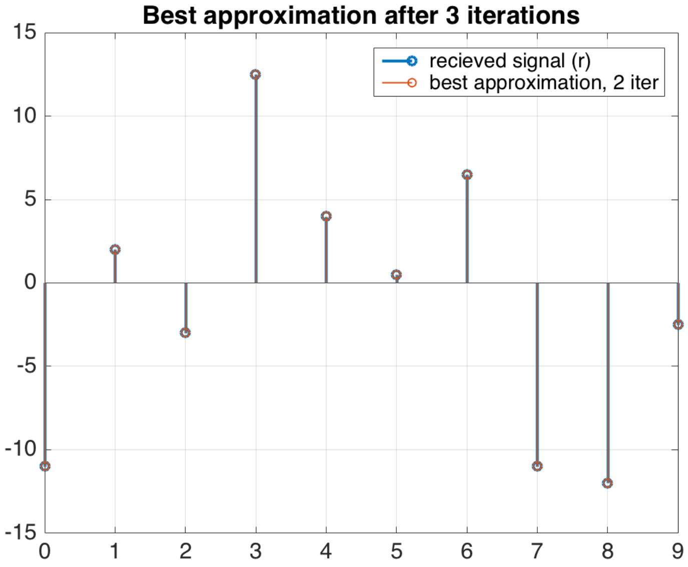
$$= \begin{bmatrix} 2.0 \\ -1 \\ 0.5 \end{bmatrix}$$

will only be exactly correct w/ zero noise!

(these are correct!)

How did we do? Find best approx. to r

$$\hat{\vec{r}} = A\hat{x} = \begin{bmatrix} \vec{s}_1^{(7)} & \vec{s}_2^{(3)} & \vec{s}_3^{(4)} \end{bmatrix} \begin{bmatrix} 2.0 \\ -1 \\ 0.5 \end{bmatrix}$$



Calculate the residual error

received signal

best approximation of received signal

Residual: $\vec{y} = \vec{r} - \hat{\vec{r}}$

$= \vec{0}$

No more error! We're done!

How to decide when to stop if there's noise?

Stop when either:
1) finished k iterations (sparsity), or
2) norm of residual is lower than some threshold value

can choose threshold smartly if you know noise statistics

Yay!


EASY
PEASY
LEMON
SQUEEZY


When someone tells me it's easy peasy lemon squeezy, but for me it's always stressy, depressy, lemon zesty