deep imaging

Lecture 5: A gentle introduction to optimization

Machine Learning and Imaging

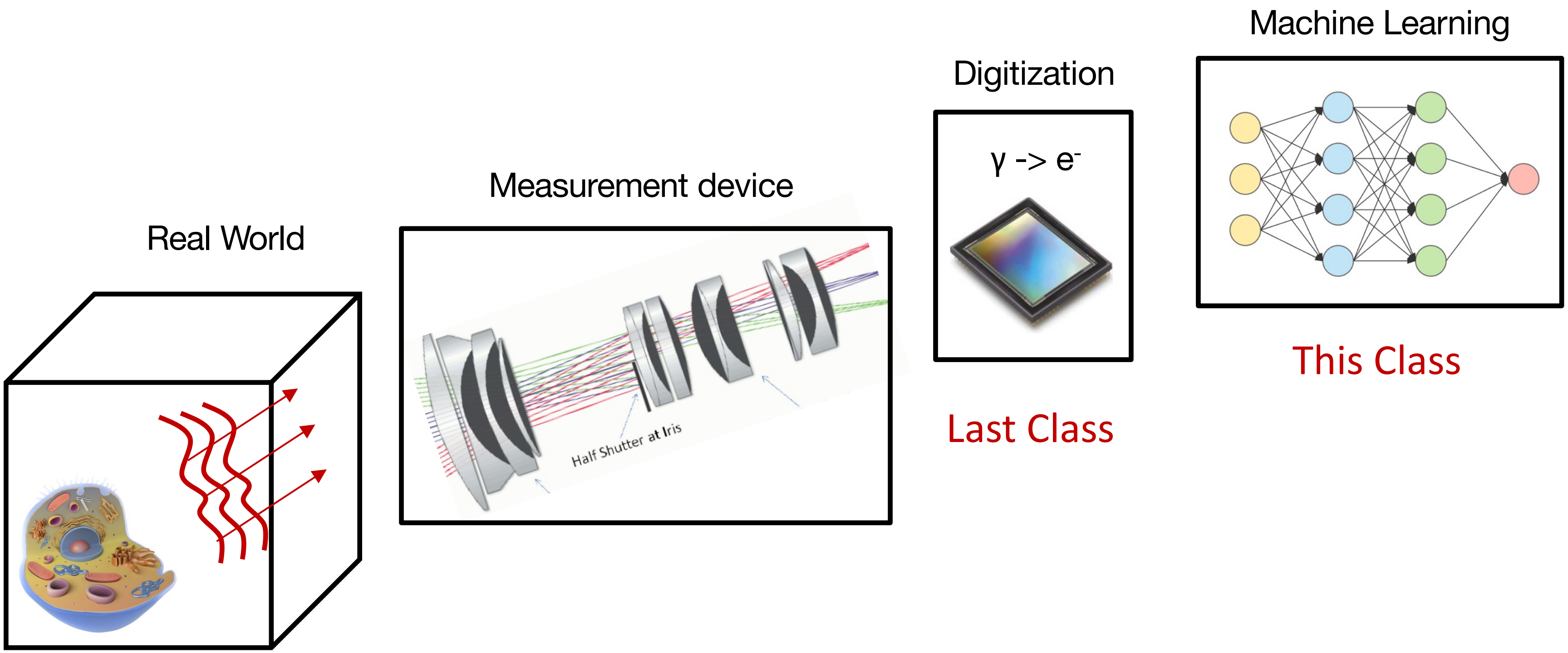
BME 548L
Roarke Horstmeyer

Announcements

- Lab today/Wednesday by me, new lab notebooks for next week released soon
- Homework #1 will be assigned by Wednesday (we'll send out an announcement email)
- Anticipated due date: Wed Feb 14

deep imaging

ML+Imaging pipeline



Mathematical Optimization: "Selection of a best element (with regard to some criterion) from a set of available alternatives"

deep imaging

Mathematical Optimization: "Selection of a best element (with regard to some criterion) from a set of available alternatives"

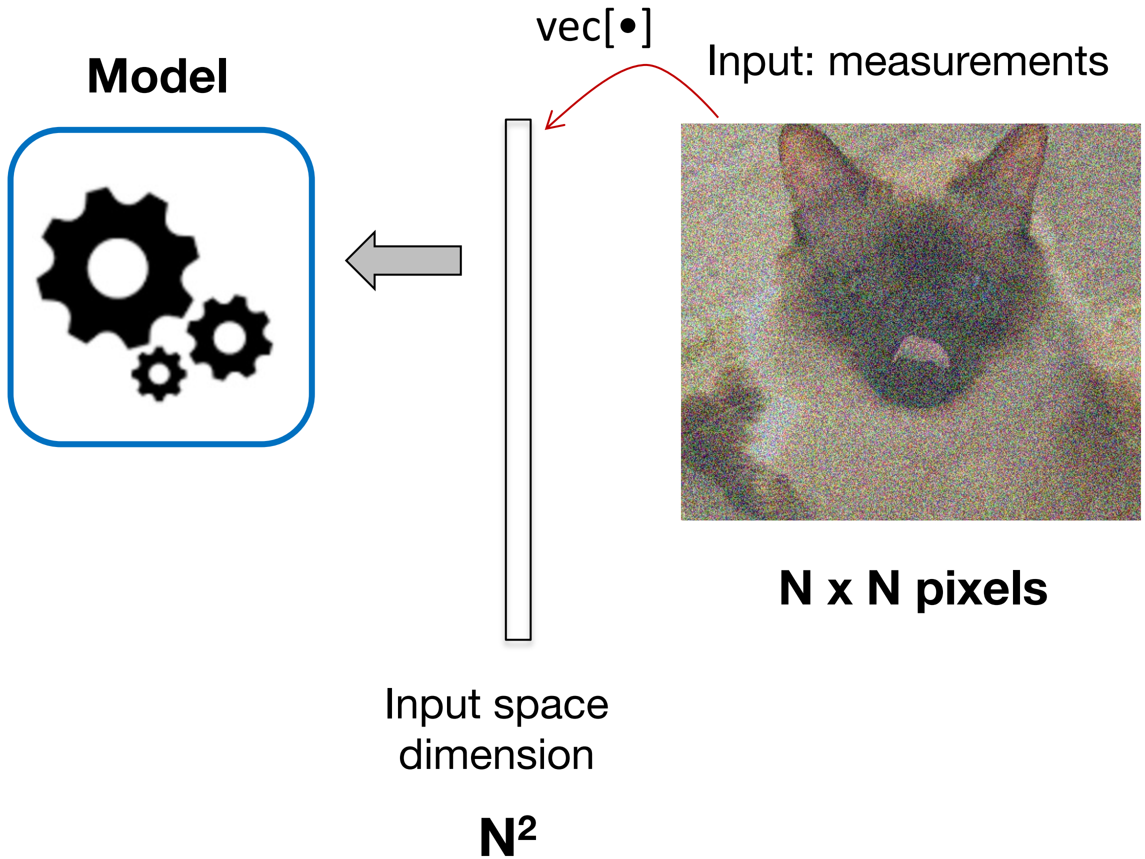
3 elements:

1) Your desired output (a better image, a clean signal, a classification of "cat" or "dog", etc.)

2) A model of what you are looking for - how you form the desired output from your measured data

3) A cost function, to measure how close you're getting to the answer (the cost function minimum)

Generalized optimization pipeline



Generalized optimization pipeline

vec[•]

Input: measurements

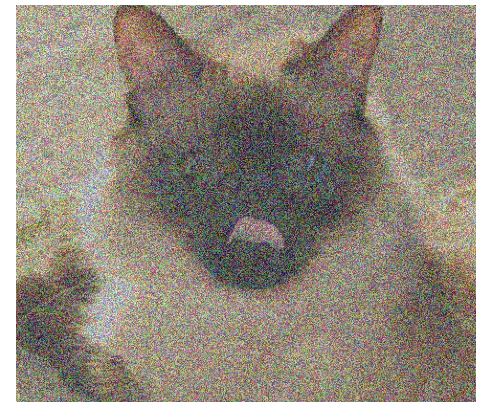
Output

Model

Cost function

0.47

$$f(x)$$



N x N pixels

Performance measure

1 number!

Output space dimension

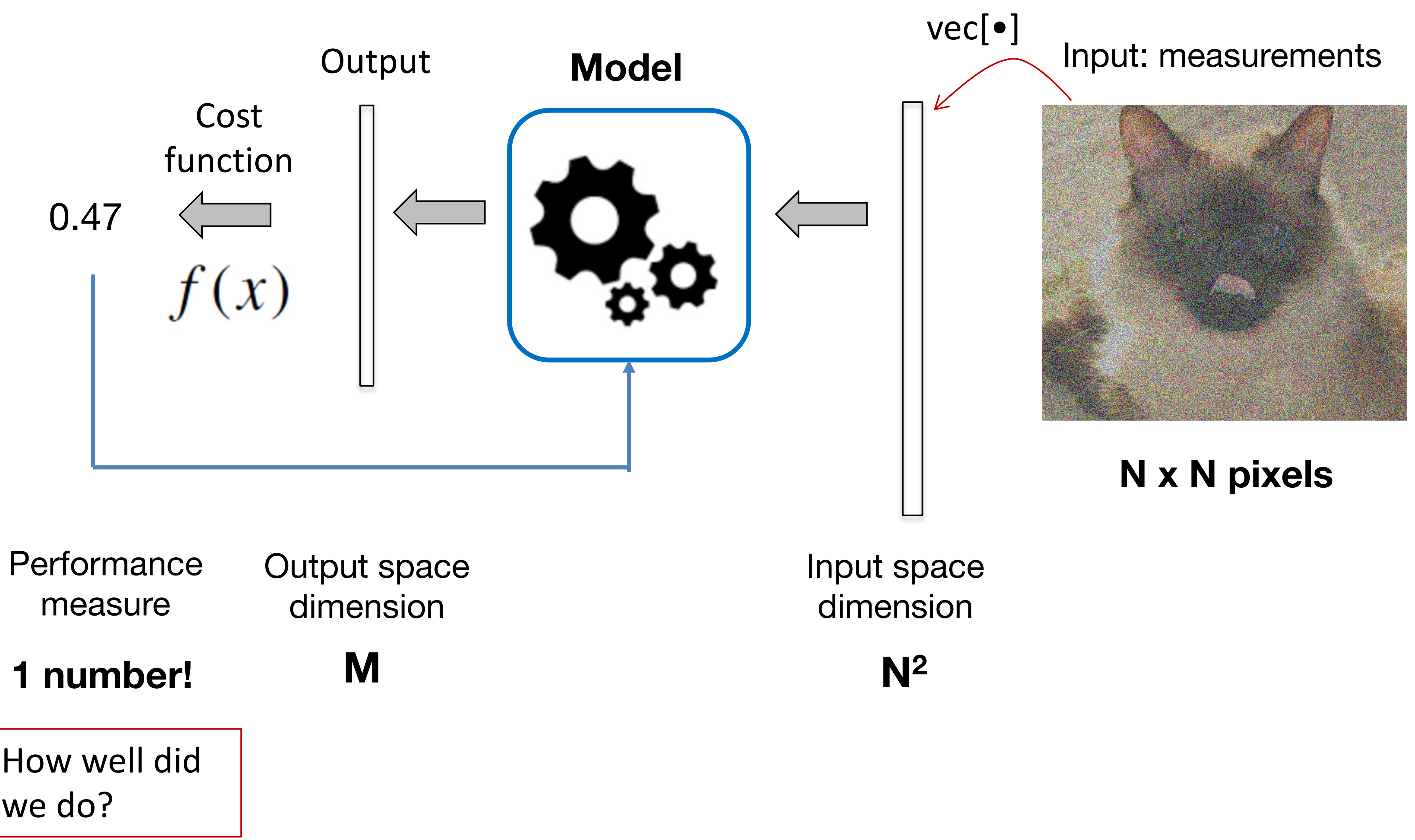
M

Input space dimension

N^2

How well did we do?

Machine learning: update model to decrease error



De-noising: "What is the closest image to what I detected, except without so many fluctuations"?

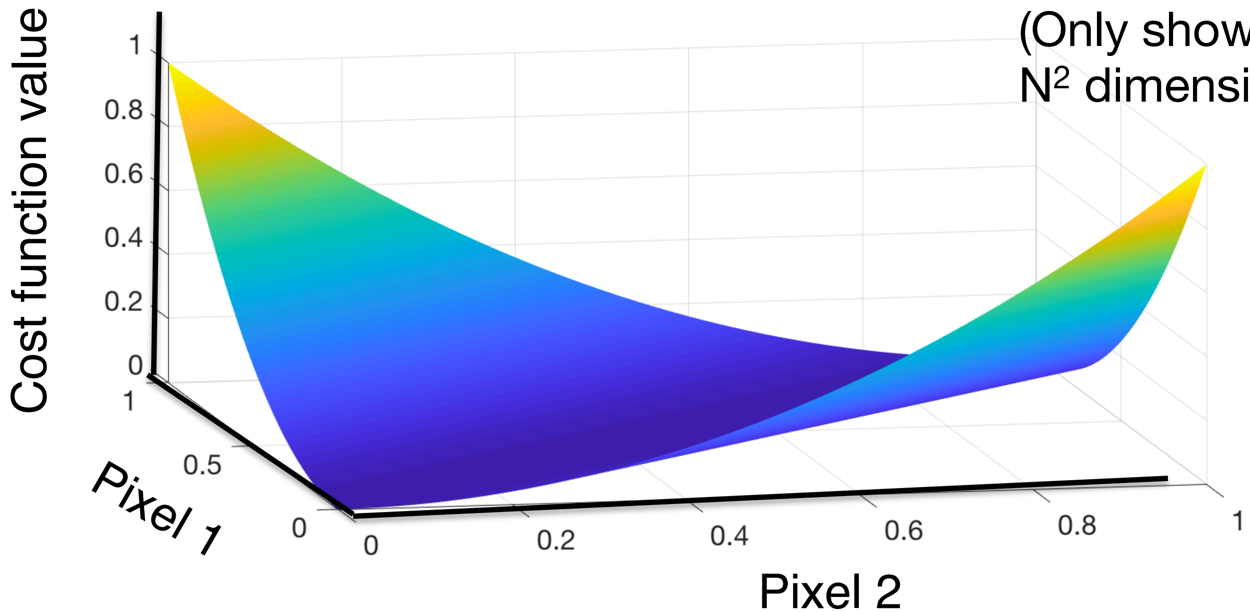
Input dimension: N x N image

Output dimension: N x N image

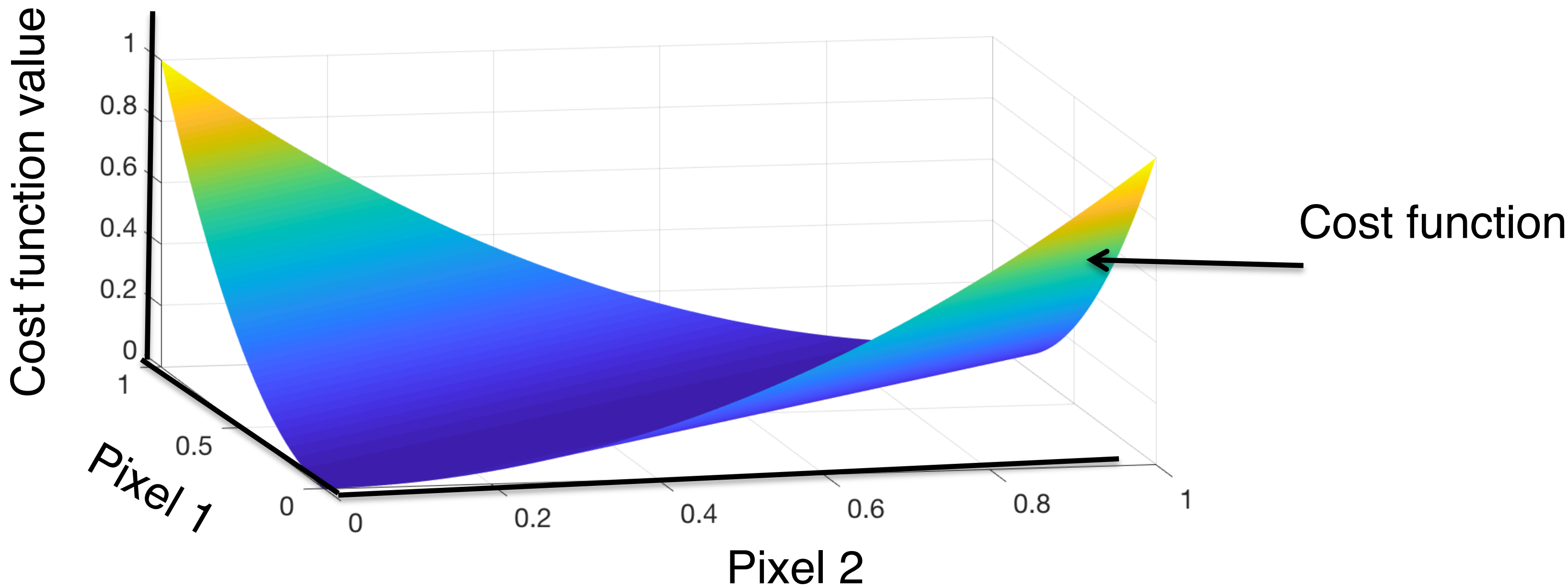
De-noising: "What is the closest image to what I detected, except without so many fluctuations"?

Cost function: "Don't let nearby pixels vary around too much"

(Only showing 2 of the N^2 dimensions)

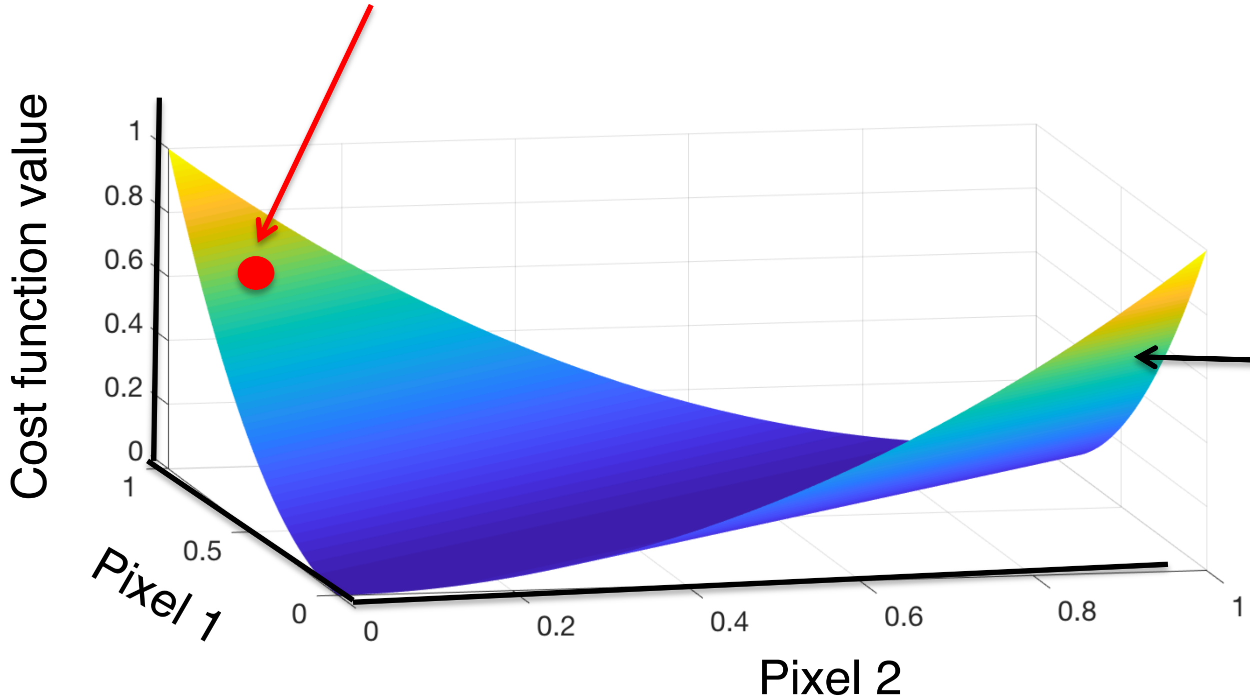


De-noising: "What is the closest image to what I detected, except without so many fluctuations"?



De-noising: "What is the closest image to what I detected, except without so many fluctuations"?

Start with a guess for desired output



Cost function

Noisy image

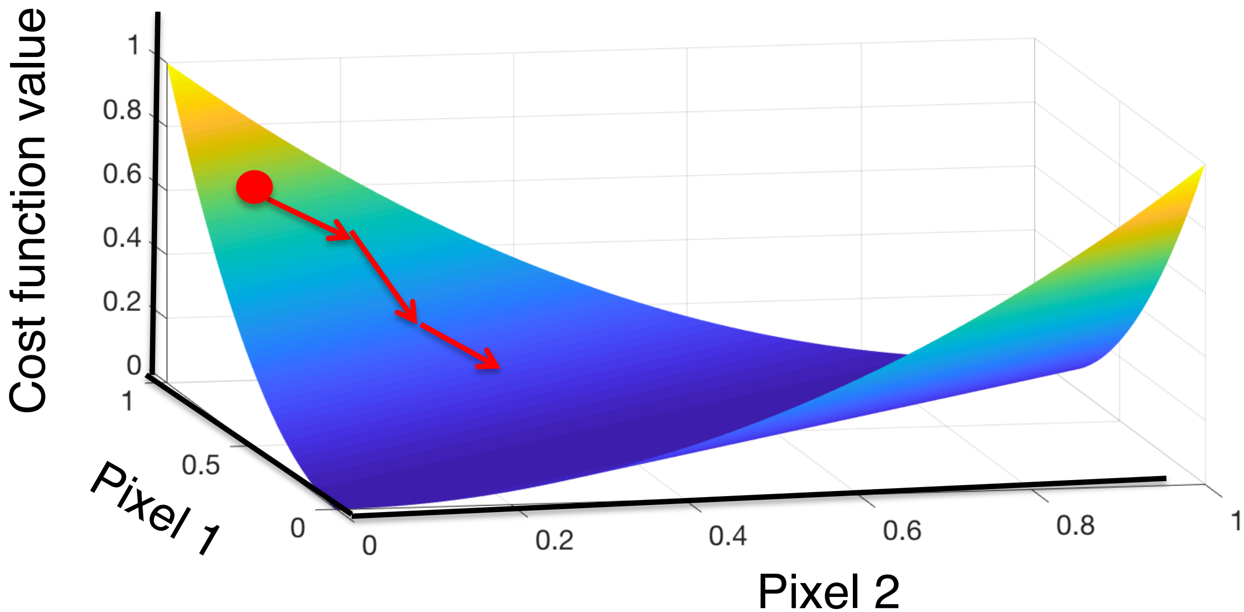
Pixel value 1

Pixel value 2

⋮

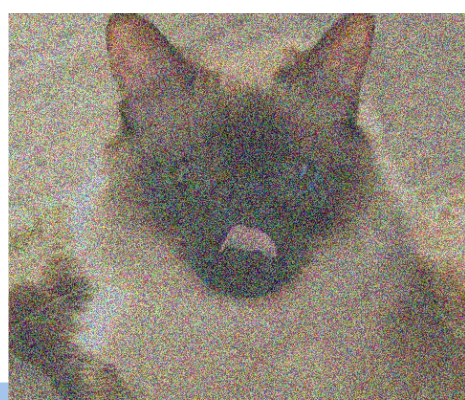
deep imaging

"Descend" to minimize cost function
(tweak values of each pixel)



Note: This part computers are really good at!

Noisy image

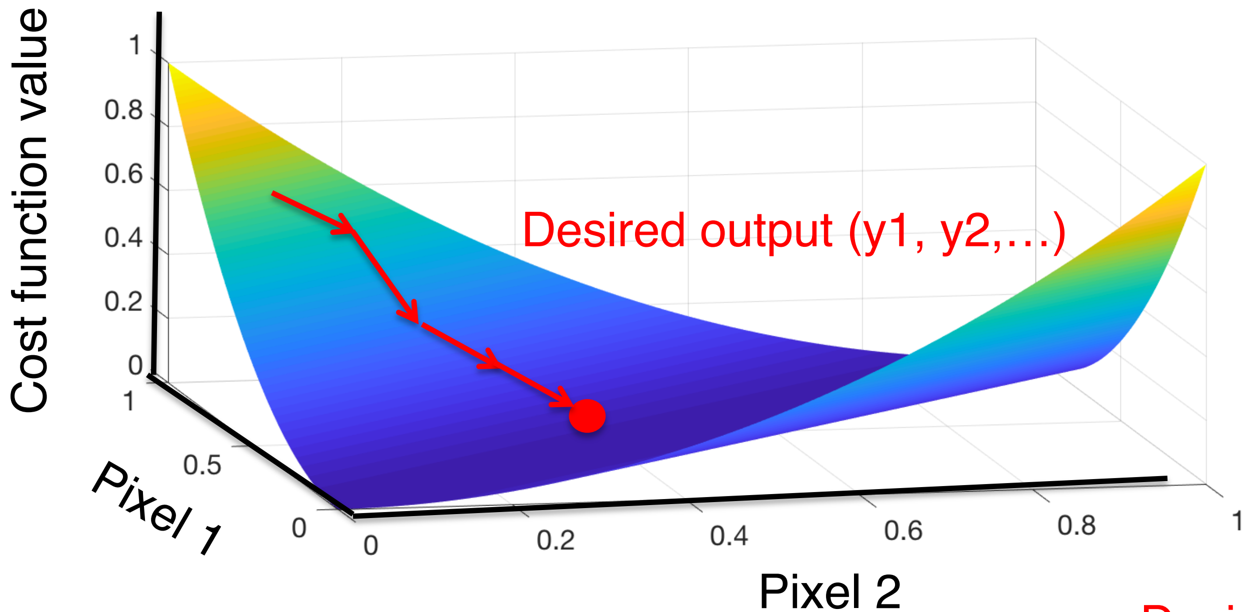


Pixel value 1

Pixel value 2

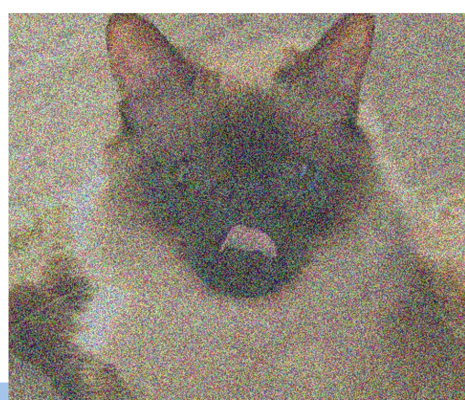
⋮

"Descend" to minimize cost function
(tweak values of each pixel)



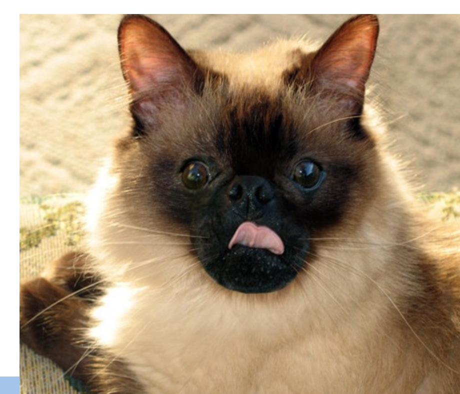
Note: This part computers are really good at!

Noisy image



Pixel value 1
Pixel value 2
⋮

Desired output



y1
y2
⋮

Optimization pipeline for denoising

deep imaging

vec[•]

Input: measurements

N x N pixels

Output

Model

Cost function

0.47

$f(y)$

"Neighboring pixels shouldn't vary too rapidly"

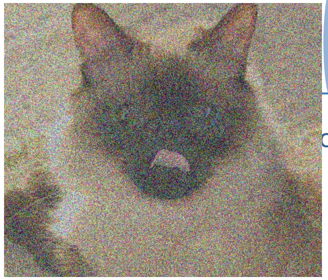
Performance measure

Output space dimension **N^2**

Input space dimension **N^2**

Mean-squared error

Input image



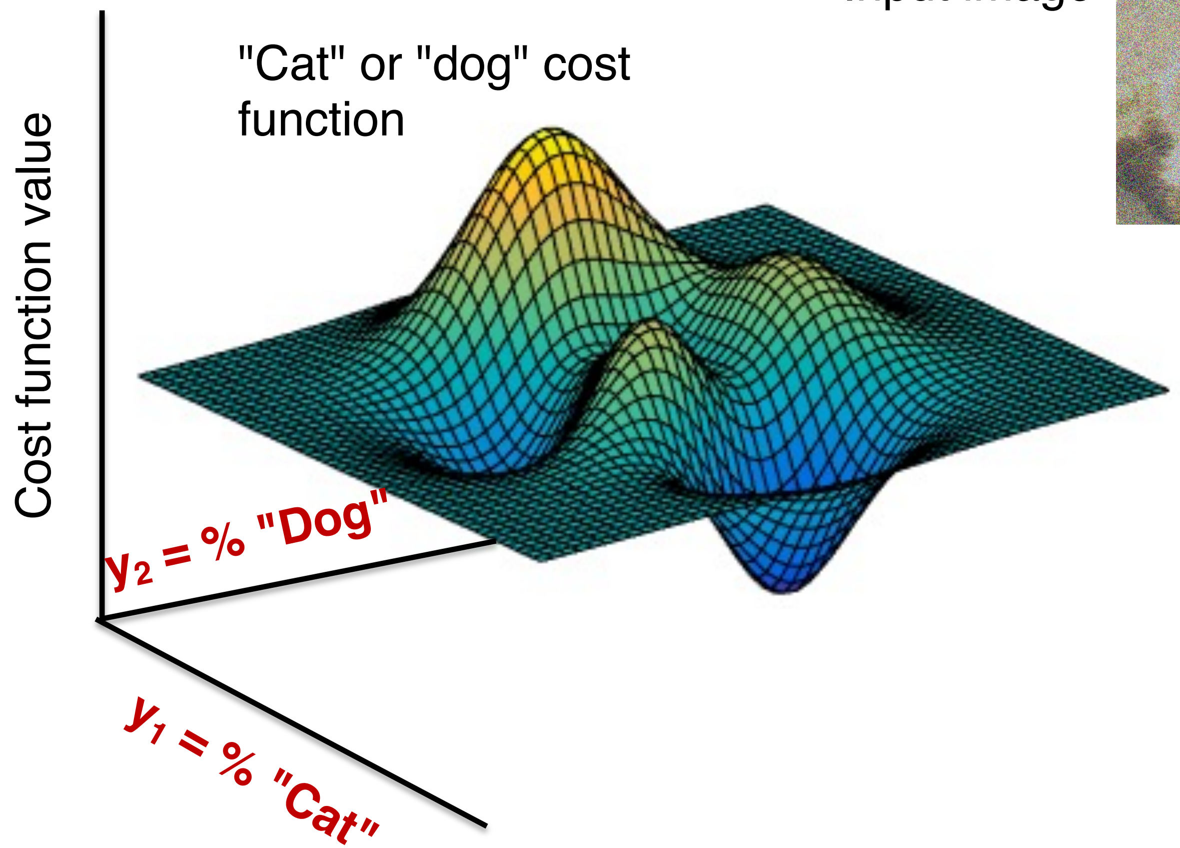
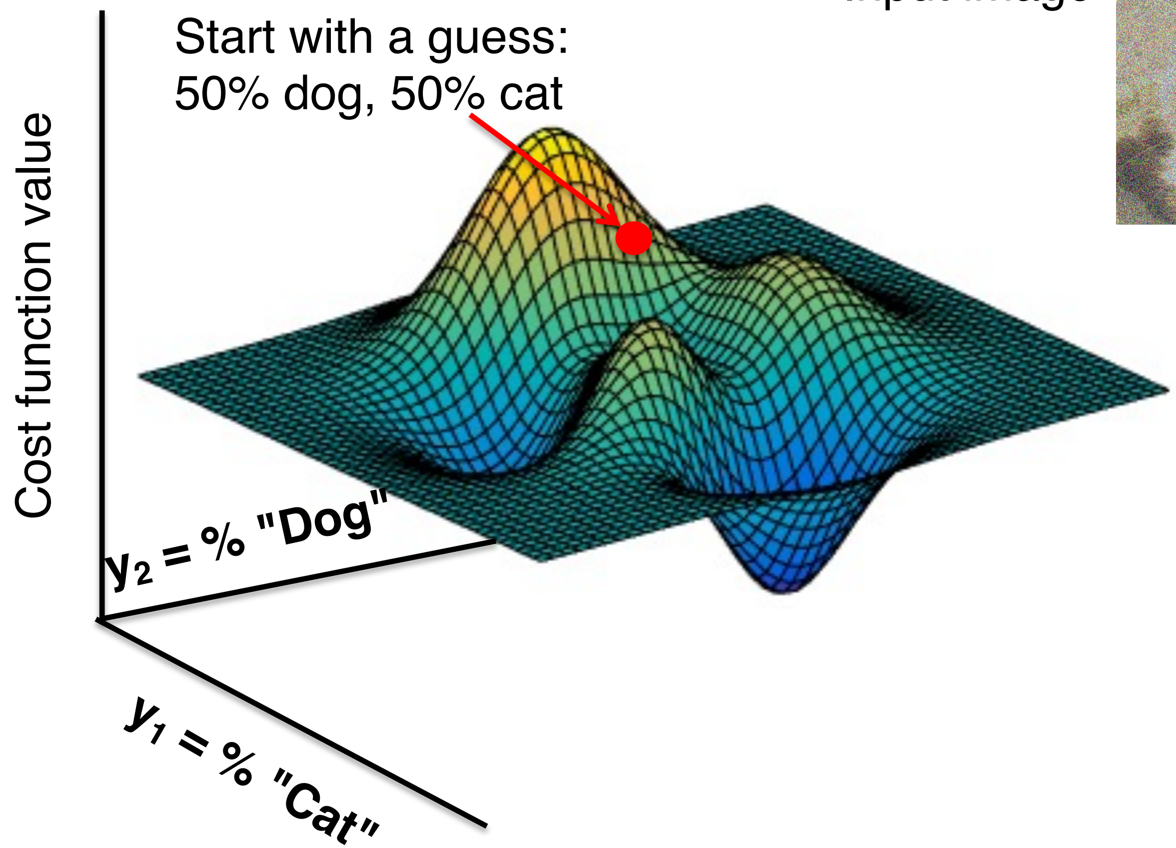
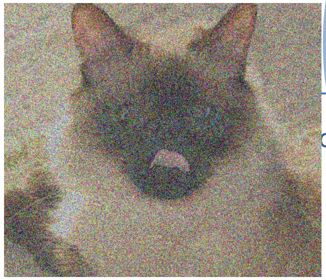


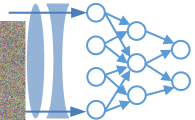
Image Classification Problem: "Is the image of a dog or a cat"?

Output dimensions now *not* image pixels, but instead some “decision” axes

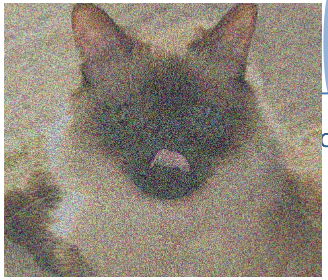

Start with a guess:
50% dog, 50% cat
Cost function value
y_2 = % "Dog"
y_1 = % "Cat"

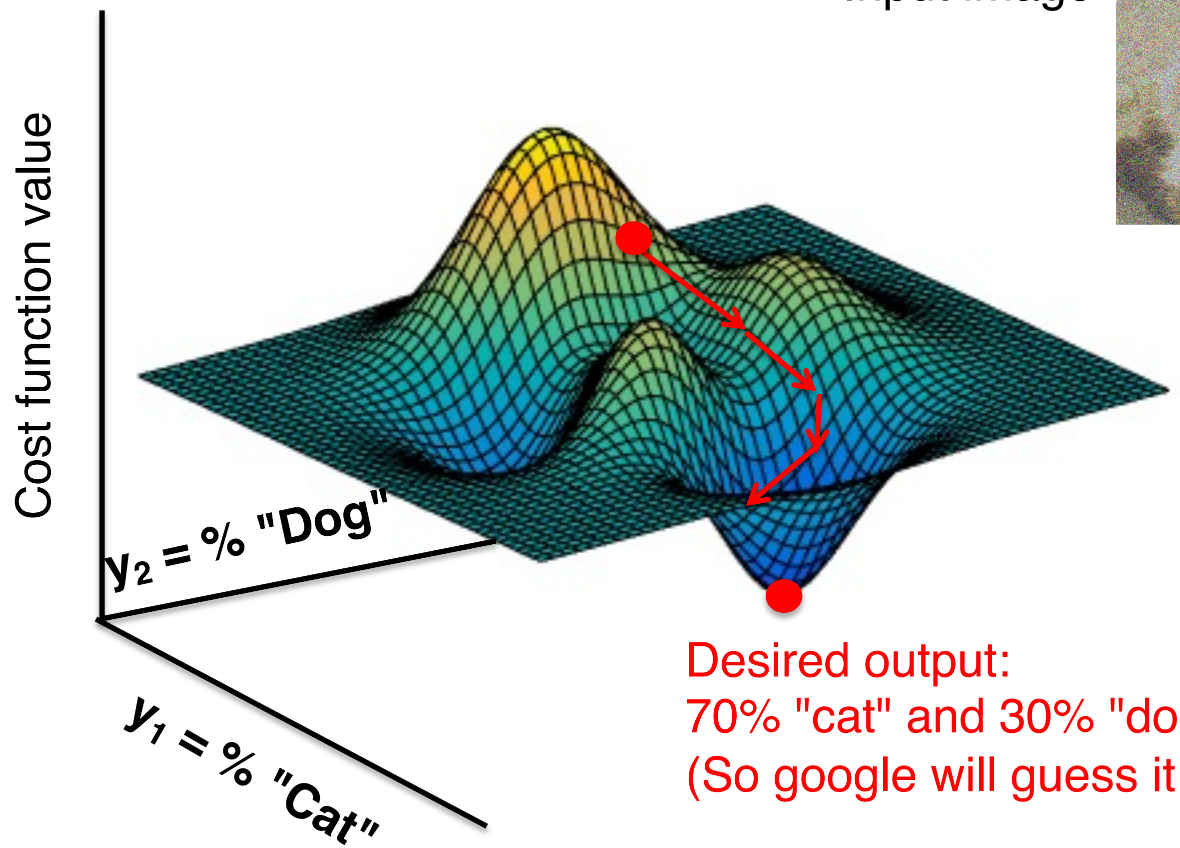
Input image




deep imaging

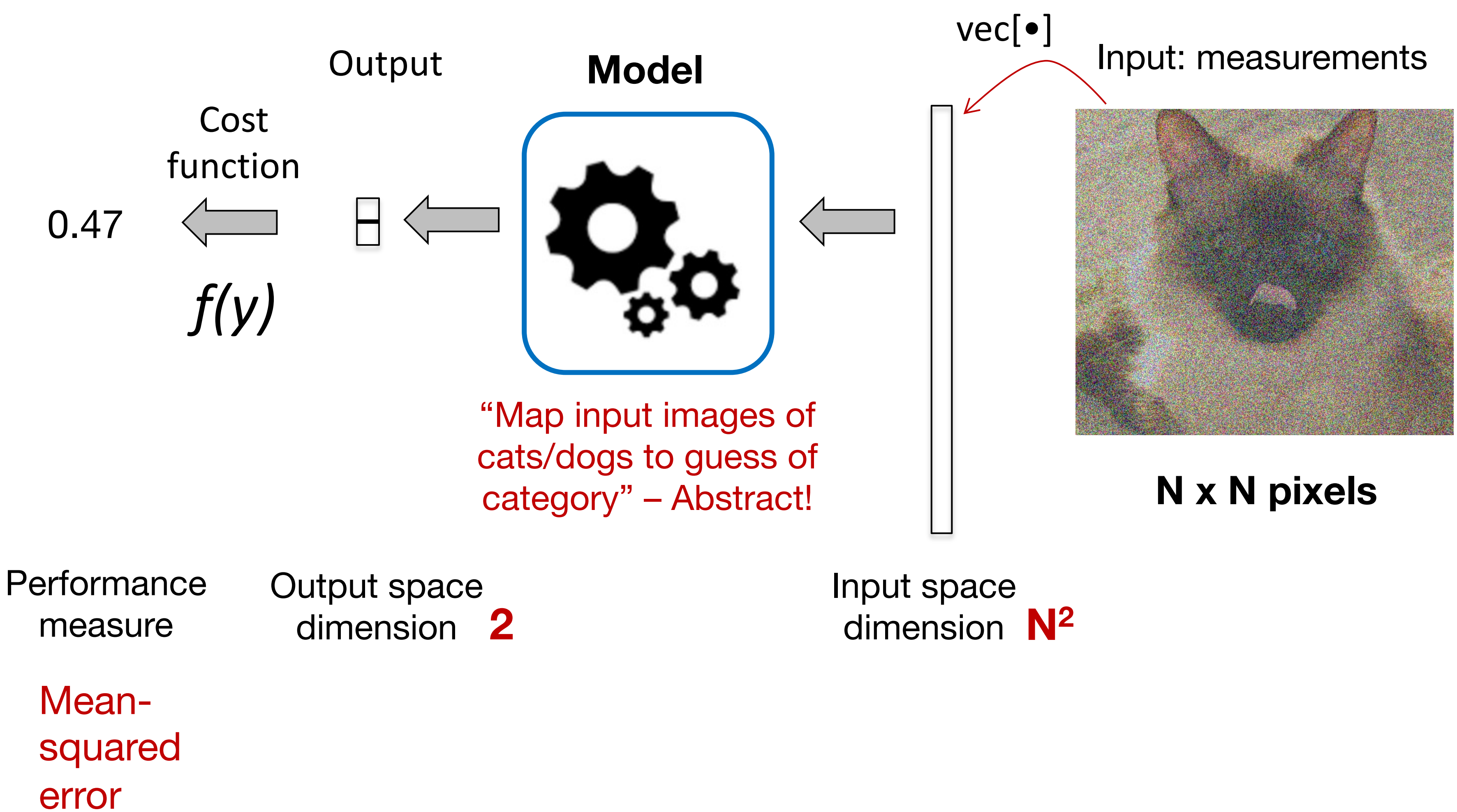
Input image





Desired output:
70% "cat" and 30% "dog"
(So google will guess it is likely a cat)

Optimization pipeline for classification



A simple example: spectral unmixing

(For whatever reason, whenever I get confused about optimization, I think about this example...it's a good one)

A simple example: spectral unmixing

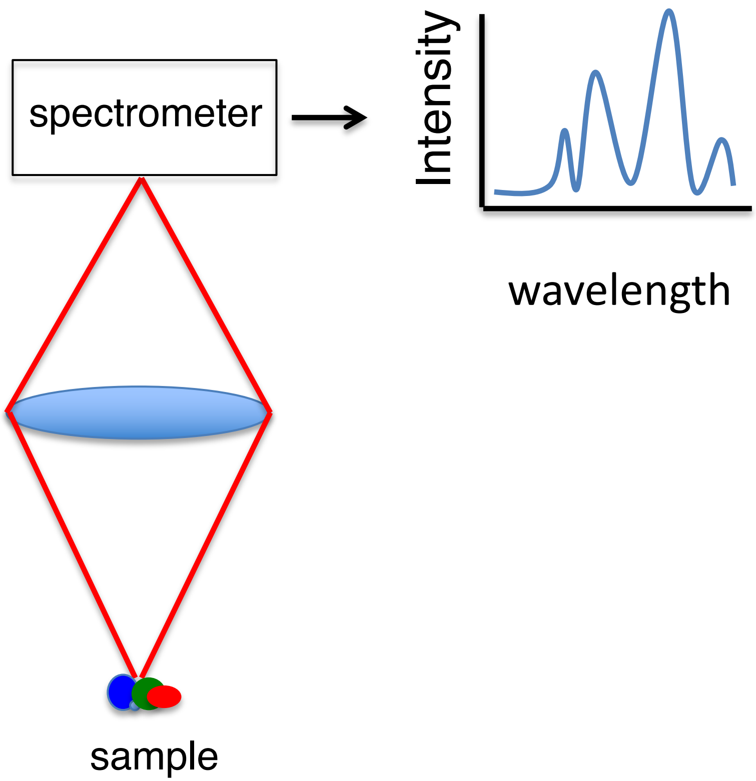
(For whatever reason, whenever I get confused about optimization, I think about this example...it's a good one)

The setup:

- measure the color (spectral) response of a sample (e.g., how much red, green and blue there is, or several hundred measurements of its different colors).

- You know that the sample can only contain 9 different fluorophores.

- What % of each fluorophores is in your sample?



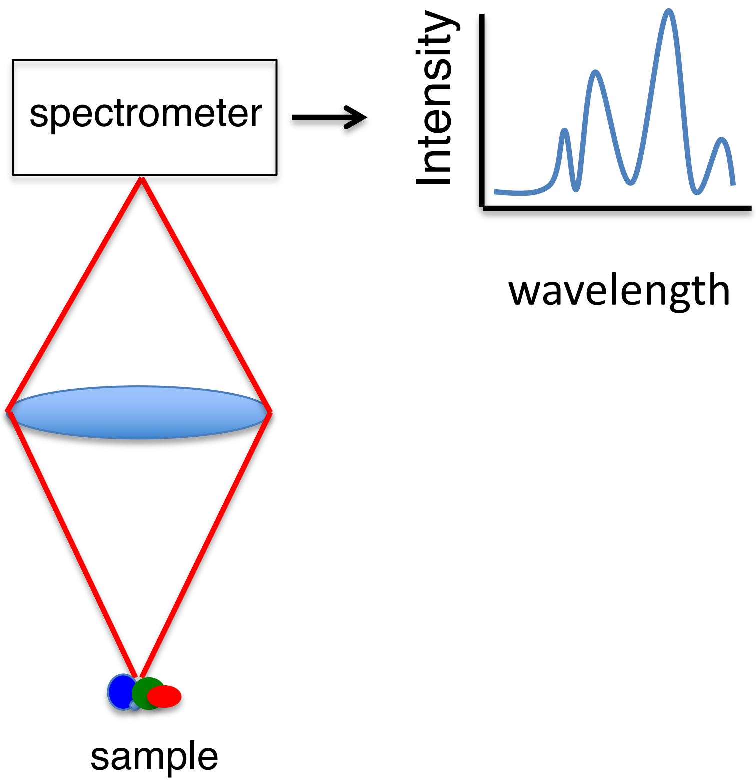
A simple example: spectral unmixing

3 elements of optimization:

1) Desired output

2) The model

3) The cost function



A simple example: spectral unmixing

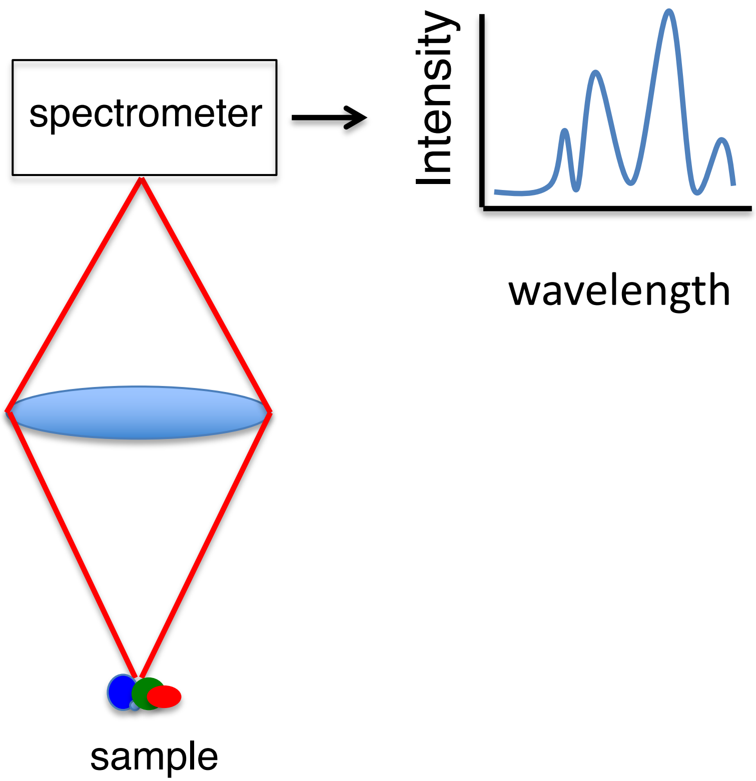
3 elements of optimization:

1) Desired output

What % of each of the 9 fluorophores

2) The model

3) The cost function



A simple example: spectral unmixing

3 elements of optimization:

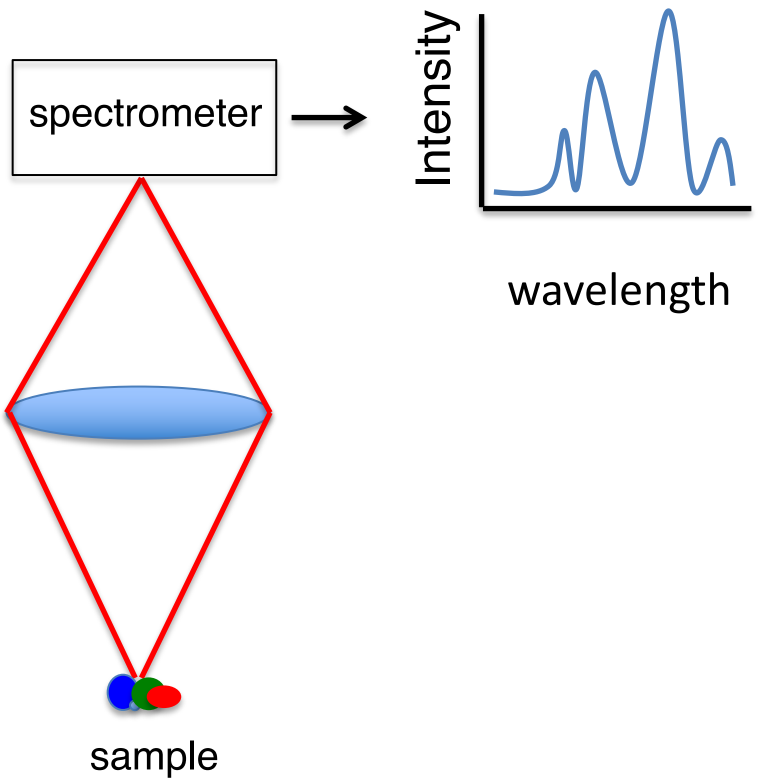
1) Desired output

What % of each of the 9 fluorophores

2) The model

"Dictionary" of the 9 different spectra

3) The cost function



A simple example: spectral unmixing

3 elements of optimization:

1) Desired output

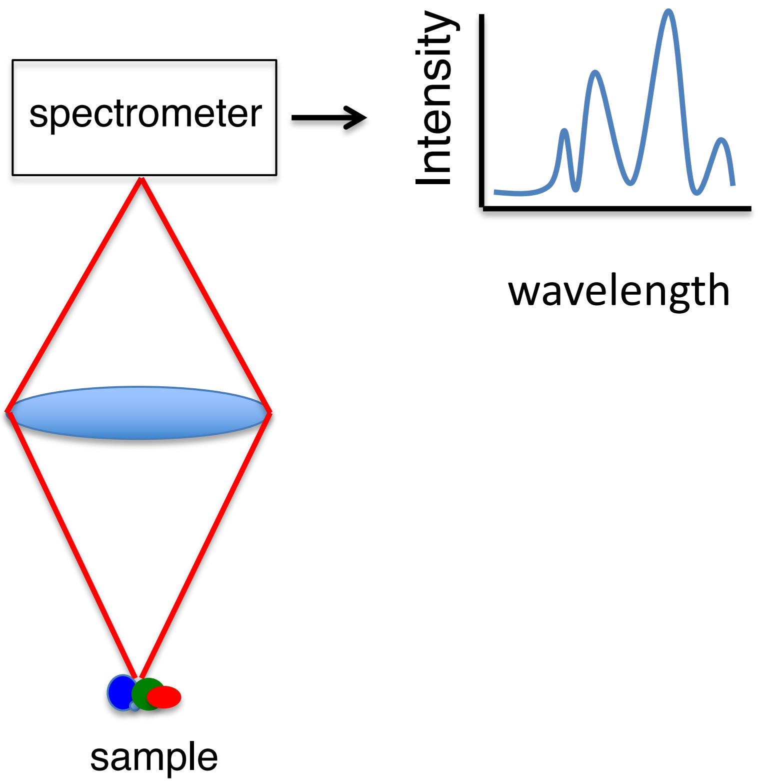
What % of each of the 9 fluorophores

2) The model

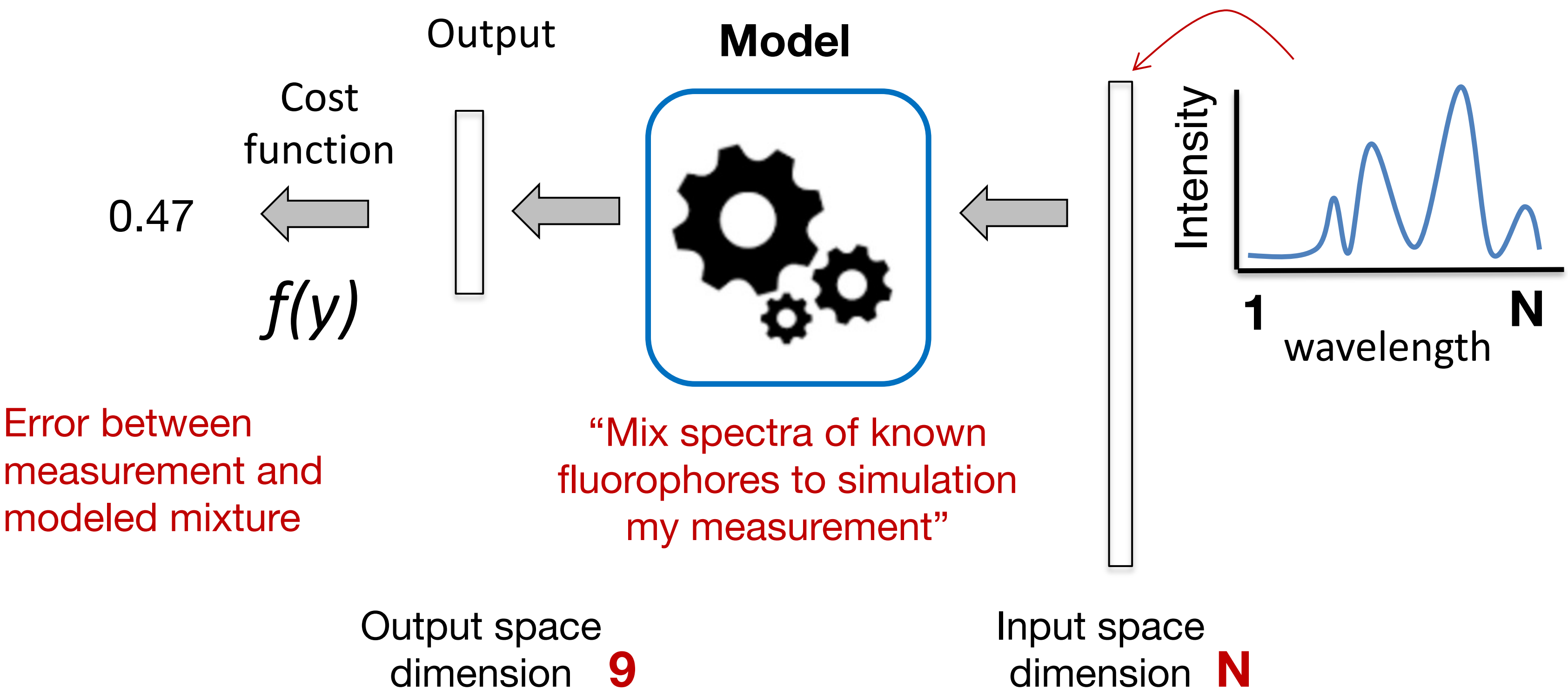
"Dictionary" of the 9 different spectra

3) The cost function

Minimum mean squared error (to start)

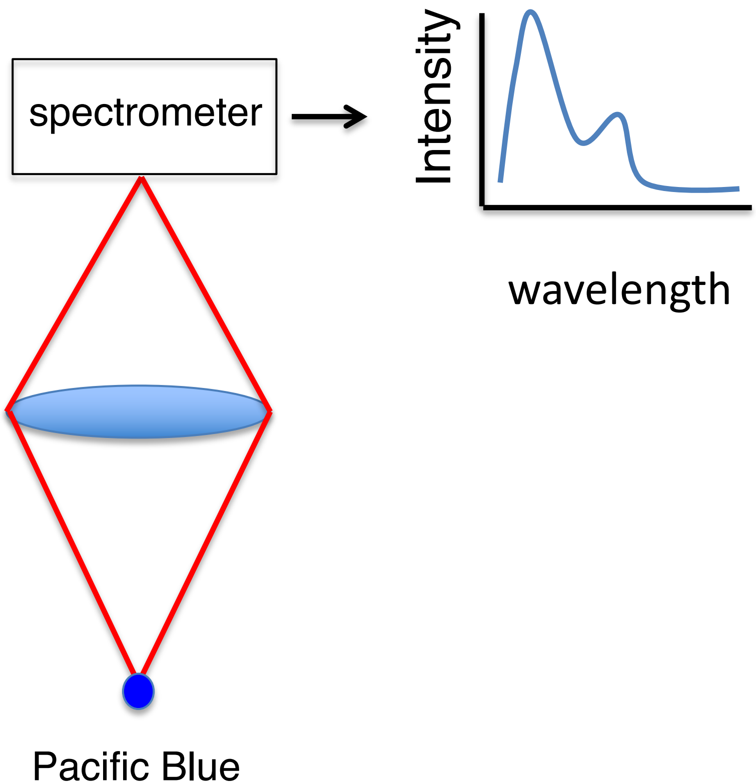


Optimization pipeline for spectral unmixing



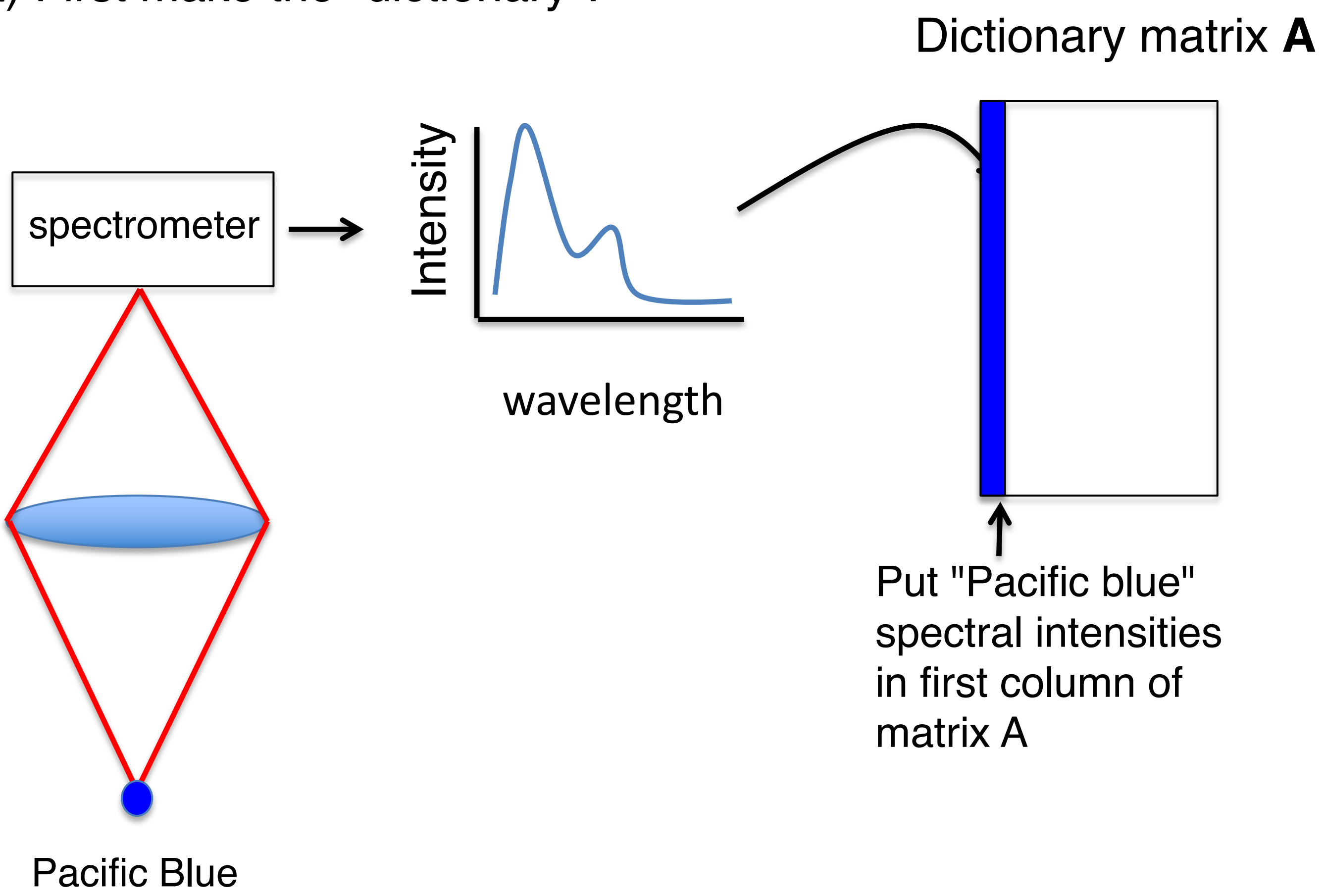
Mathematical model for spectral unmixing

a) First make the "dictionary":



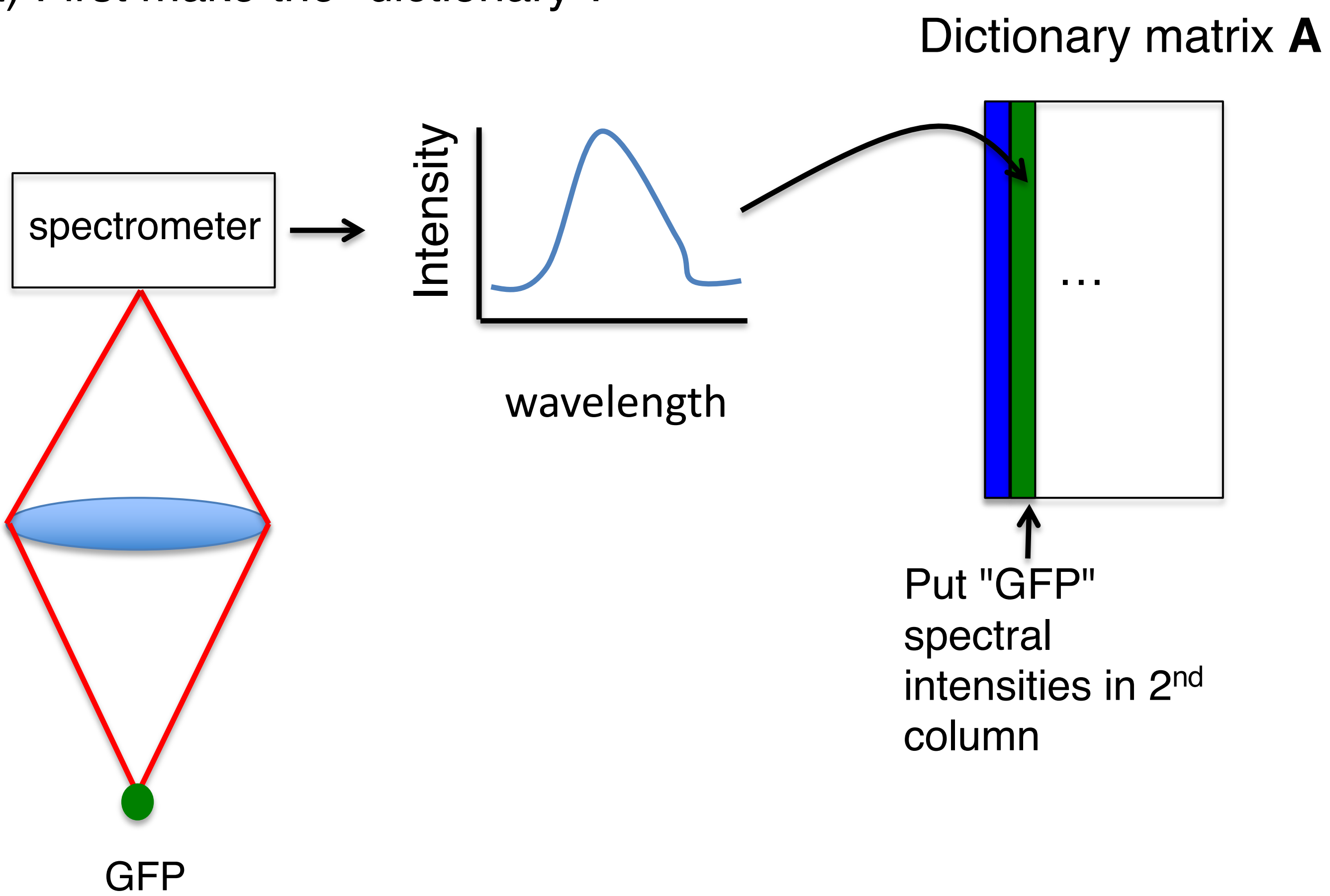
Mathematical model for spectral unmixing

a) First make the "dictionary":



Mathematical model for spectral unmixing

a) First make the "dictionary":

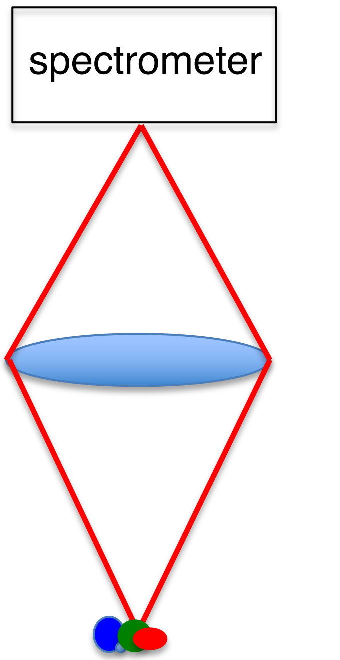


deep imaging

Mathematical model for spectral unmixing

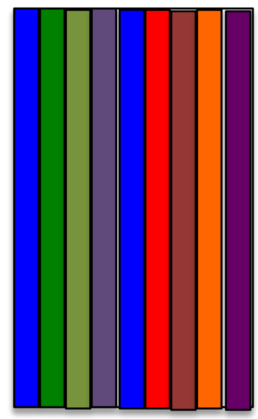
b) Model the unknown sample %'s

(the desired output)



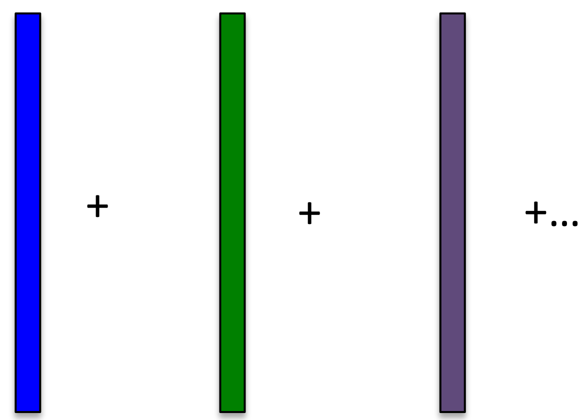
Unknown sample y

Dictionary matrix **A**



9 possible spectra

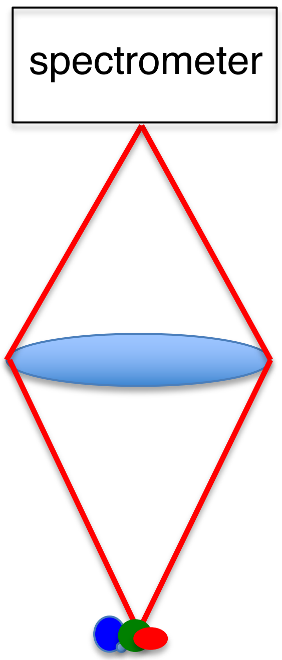
Some mixture…



Mathematical model for spectral unmixing

b) Model the unknown sample %'s

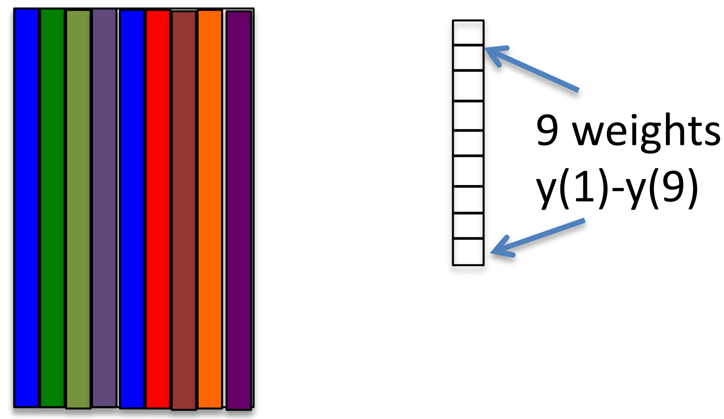
(the desired output)



Unknown sample y

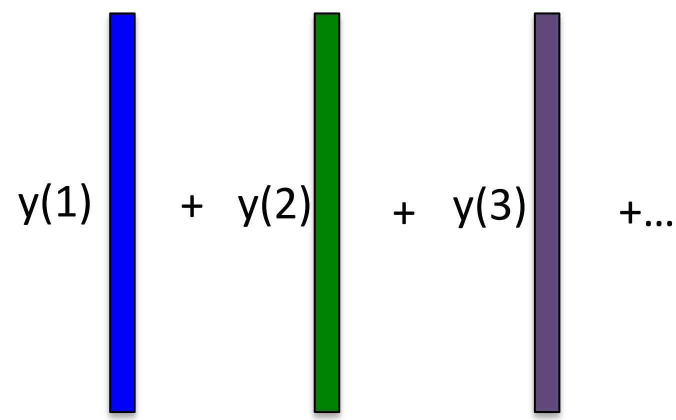
Dictionary matrix **A**

Unknown sample %'s **y**



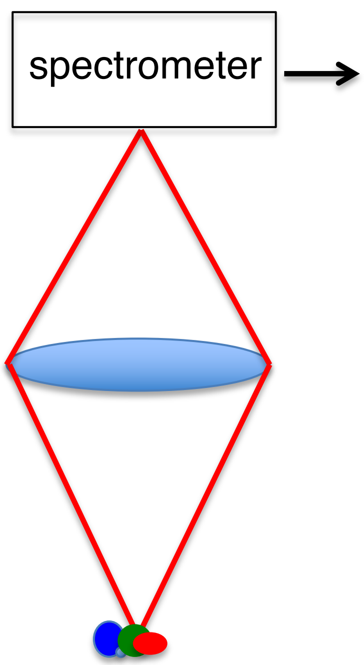
9 possible spectra

Each weight in x is percentage:



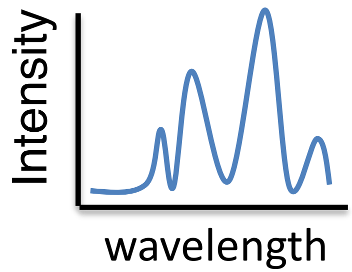
Mathematical model for spectral unmixing

b) Model the unknown sample %'s

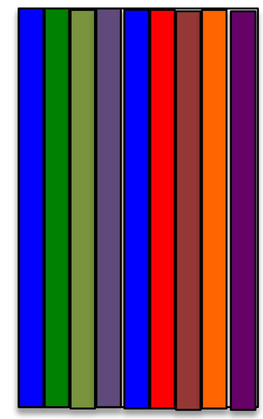


Unknown sample y

New measurement x

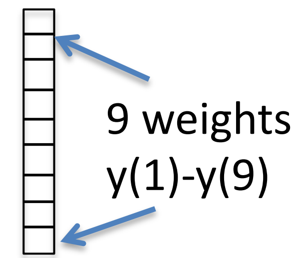


Dictionary matrix **A**

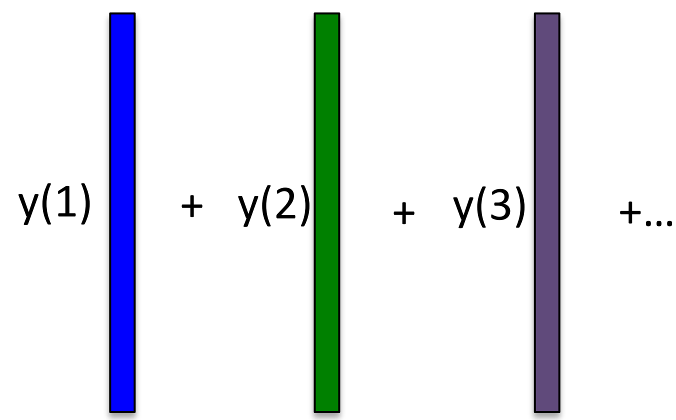


9 possible spectra

Unknown sample %'s **y**

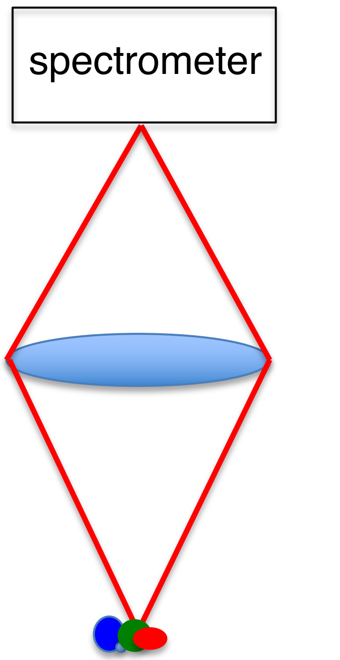


Each weight in x is percentage:

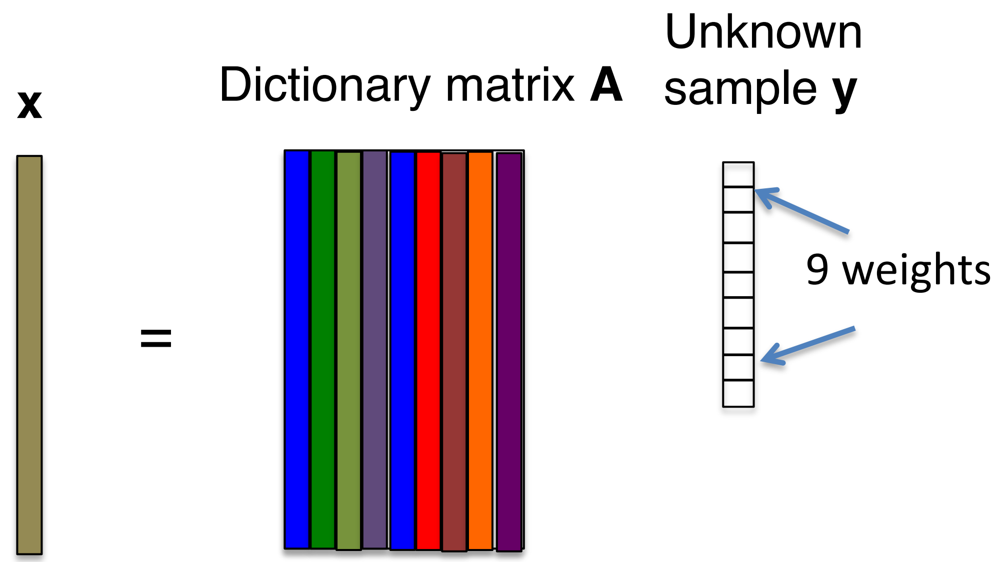


Mathematical model for spectral unmixing

deep imaging



Unknown sample y

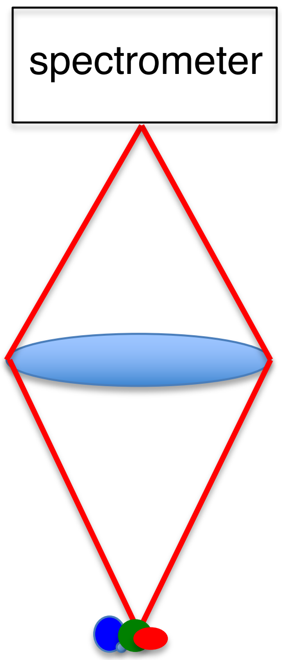


Matrix equation: **x=Ay**

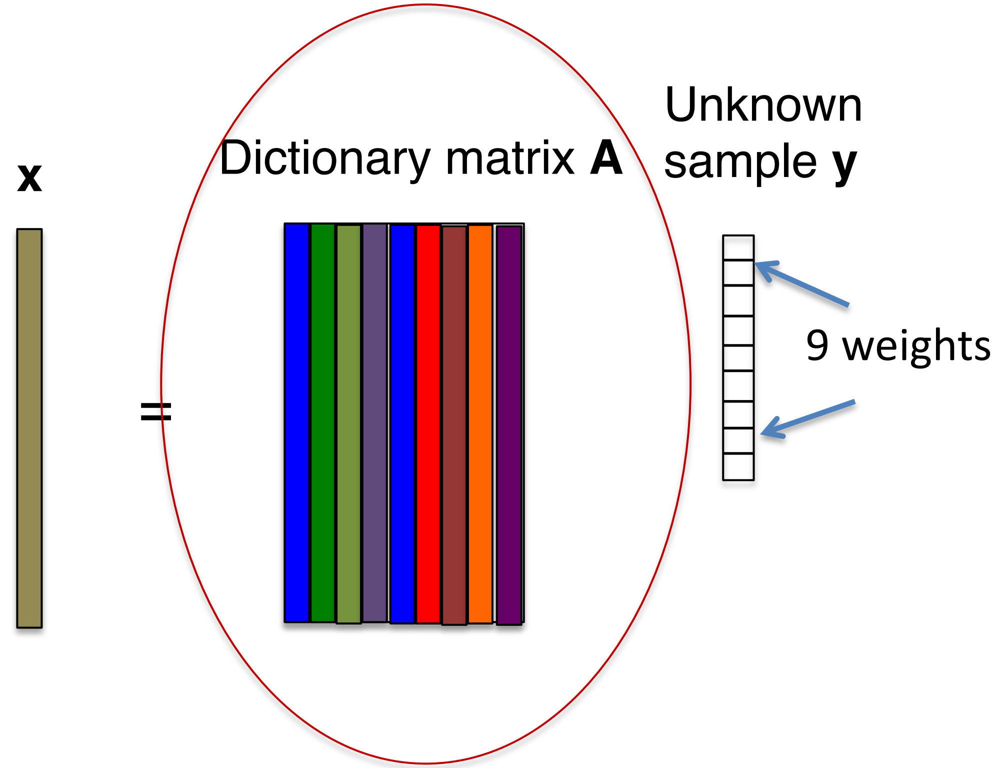
This is referred to as a “forward” model

Goal: Given A and x, find y

Mathematical model for spectral unmixing



Unknown sample y



Data in A can be thought of , in some sense, as "training data"

Cost function for spectral unmixing

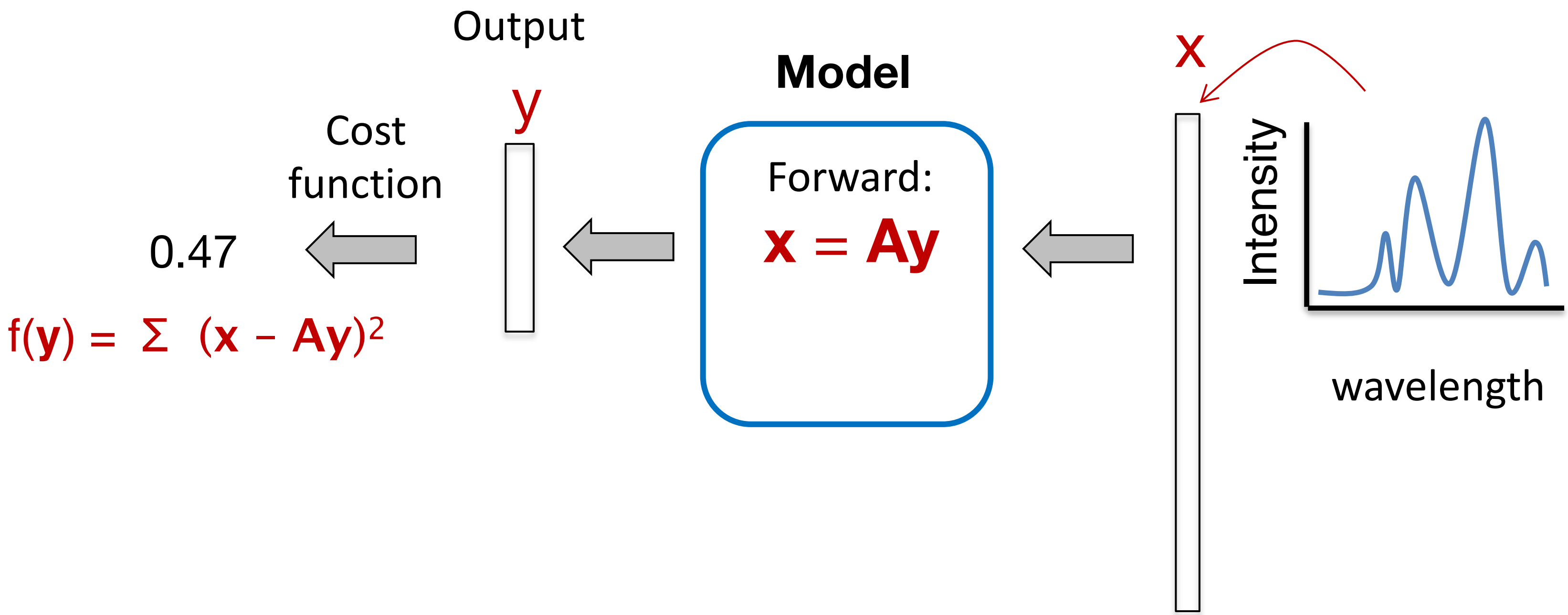
deep imaging

$\mathbf{x} = \mathbf{A}\mathbf{y}$ won't always be true, due to noise (actually, $\mathbf{x} = \mathbf{A}\mathbf{y}+\mathbf{n}$)

Common cost function is minimum mean-squared error:

$$\text{Cost function } f(\mathbf{y}) = \underbrace{\Sigma}_{\text{spectral measurements}} (\mathbf{x} - \mathbf{A}\mathbf{y})^2$$

Optimization pipeline with input and output variables



Cost function for spectral unmixing

deep imaging

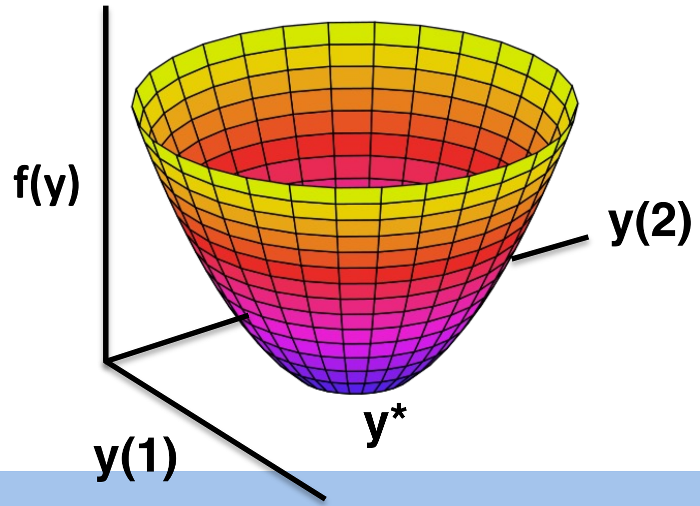
$\mathbf{x} = \mathbf{Ay}$ won't always be true, due to noise (actually, $\mathbf{x} = \mathbf{Ay}+\mathbf{n}$)

Common cost function is minimum mean-squared error:

$$\text{Cost function } f(\mathbf{y}) = \sum_{\text{spectral measurements}} (\mathbf{x} - \mathbf{Ay})^2$$

Find mixture **y** of known spectra **A** that is as close as possible to measurement **x**

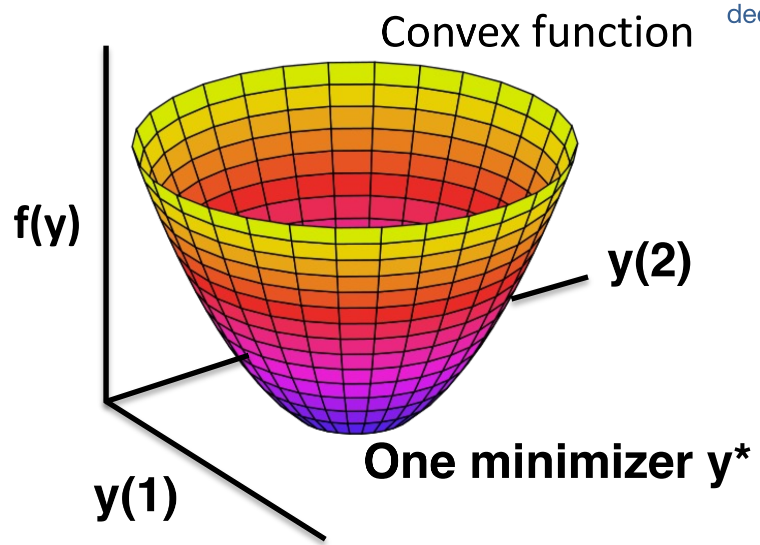
$$\mathbf{y}^* = \text{minimize } f(\mathbf{y})$$



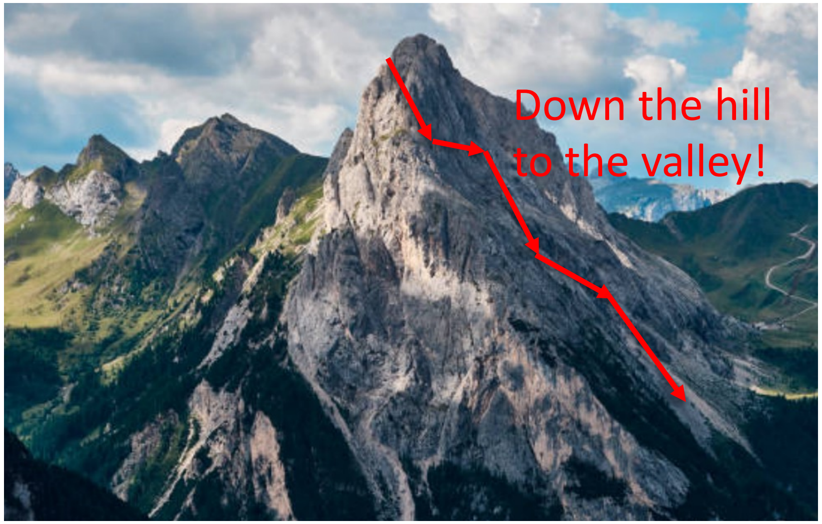
Cost function for spectral unmixing

$$f(\mathbf{y}) = \Sigma \ (x - \mathbf{A}\mathbf{y})^2$$

spectral
measurements



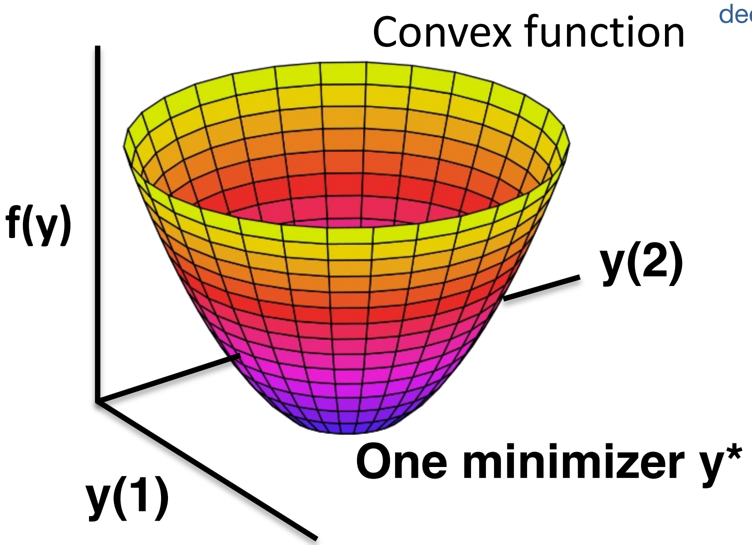
f(**y**) is convex, so finding **y*** is easy via its gradient:



Cost function for spectral unmixing

$$f(\mathbf{y}) = \Sigma \ (x - \mathbf{Ay})^2$$

spectral
measurements



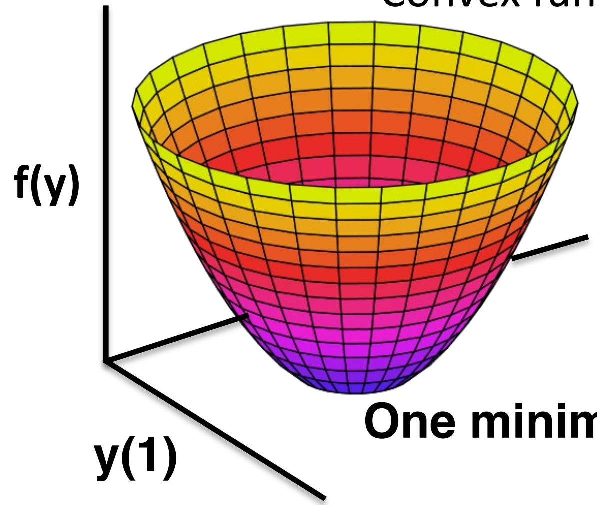
f($\mathbf{y}$) is convex, so finding $\mathbf{y^*}$ is easy via its gradient:

$$d/d\mathbf{y} \ f(\mathbf{y}) = d/d\mathbf{y} \ \ \Sigma \ (\mathbf{x} - \mathbf{Ay})^2$$

Cost function for spectral unmixing

$$f(\mathbf{y}) = \Sigma\ (x - \mathbf{Ay})^2$$

spectral
measurements



f(**y**) is convex, so finding **y*** is easy via its gradient:

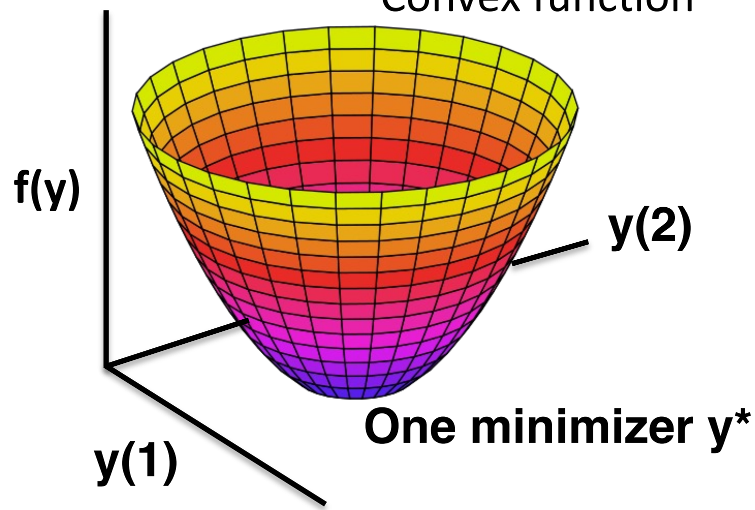
$$d/d\mathbf{y}\ f(\mathbf{y}) = d/d\mathbf{y}\ \ \Sigma\ (\mathbf{x} - \mathbf{Ay})^2$$

$$df/d\mathbf{y} = \Sigma\ d/d\mathbf{y}\ (\ \mathbf{x} - \mathbf{Ay})^2$$

Cost function for spectral unmixing

$$f(\mathbf{y}) = \Sigma \ (x - \mathbf{Ay})^2$$

spectral measurements



f(**y**) is convex, so finding **y*** is easy via its gradient:

$$d/d\mathbf{y}\ f(\mathbf{y}) = d/d\mathbf{y}\ \ \Sigma\ (\mathbf{x} - \mathbf{Ay})^2$$

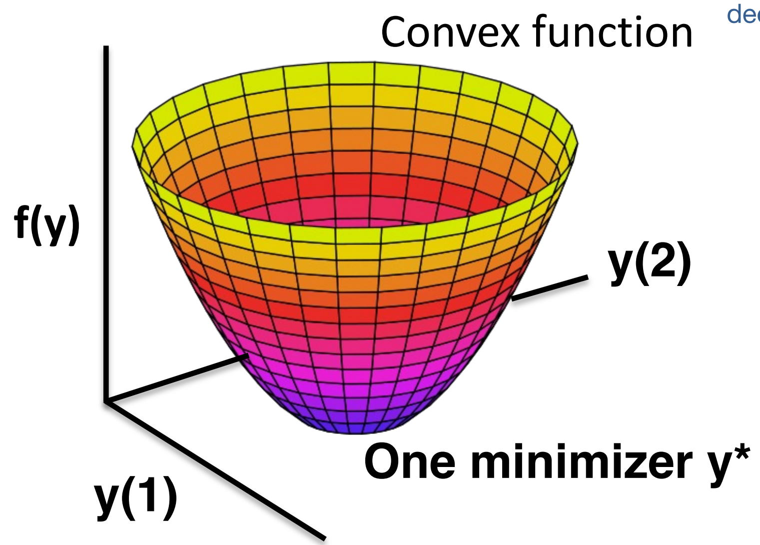
$$df/d\mathbf{y} = \Sigma\ d/d\mathbf{y}\ (\ \mathbf{x} - \mathbf{Ay})^2$$

$$df/dy[j] = \Sigma\ -2\ \mathbf{a}(:,j) * (\ \mathbf{x} - \mathbf{Ay})$$

Cost function for spectral unmixing

$$f(\mathbf{y}) = \Sigma \ (x - \mathbf{Ay})^2$$

spectral measurements



f($\mathbf{y}$) is convex, so finding $\mathbf{y}^*$ is easy via its gradient:

$$d/d\mathbf{y}\ f(\mathbf{y}) = d/d\mathbf{y}\ \ \Sigma\ (\mathbf{x} - \mathbf{Ay})^2$$

$$df/d\mathbf{y} = \Sigma\ d/d\mathbf{y}\ (\ \mathbf{x} - \mathbf{Ay})^2$$

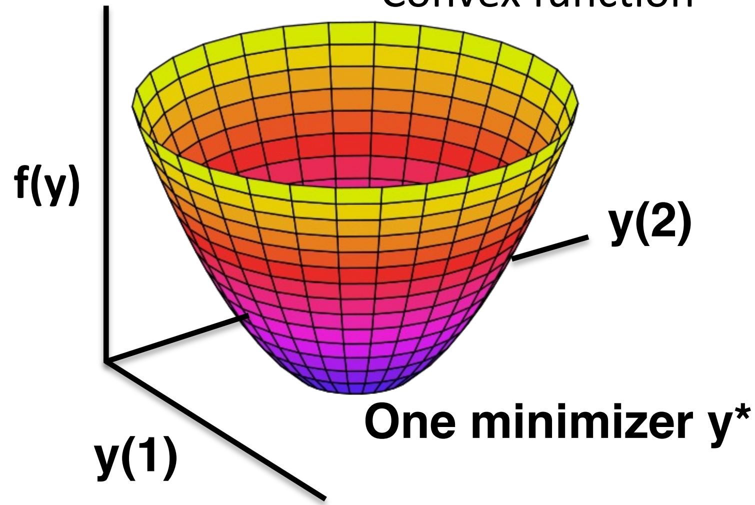
$$df/dy[j] = \Sigma\ -2\ \mathbf{a}(:,j) * (\ \mathbf{x} - \mathbf{Ay})$$

$$df/d\mathbf{y} = -2\ \mathbf{A}^T(\ \mathbf{x} - \mathbf{Ay})$$

Cost function for spectral unmixing

$$f(\mathbf{y}) = \Sigma \; (x - \mathbf{Ay})^2$$

spectral
measurements



Method 1: *Gradient descent* – follow gradient downhill to solution **y***

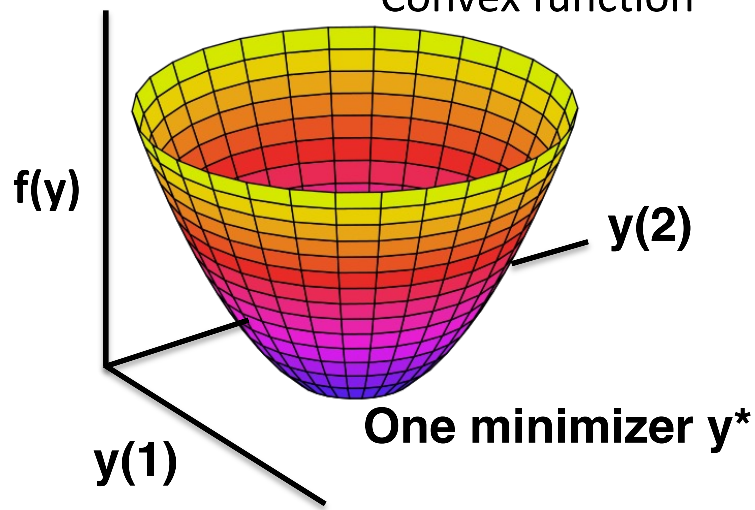
Algorithm 4.1 An algorithm to minimize $f(y) = \frac{1}{2}||\boldsymbol{A}y - x||_2^2$ with respect to y using gradient descent, starting from an arbitrary value of y.

Set the step size (ϵ) and tolerance (δ) to small, positive numbers.
while $||\boldsymbol{A}^\top \boldsymbol{A}y - \boldsymbol{A}^\top x||_2 > \delta$ **do**
  $y \leftarrow y - \epsilon\left(\boldsymbol{A}^\top \boldsymbol{A}y - \boldsymbol{A}^\top x\right)$
end while

Cost function for spectral unmixing

$$f(\mathbf{y}) = \Sigma \ (x - \mathbf{A}\mathbf{y})^2$$

spectral measurements



Method 2: *Direct solution* – set derivative to 0 to find **y*** directly

$$df/d\mathbf{y} = \mathbf{A}^T(\ \mathbf{x} - \mathbf{A}\mathbf{y}^*) = 0$$ ⟵ y* is where gradient of f(y) is zero

$$\mathbf{A}^T\ \mathbf{x} = \mathbf{A}^T\mathbf{A}\mathbf{y}^*$$ ⟶

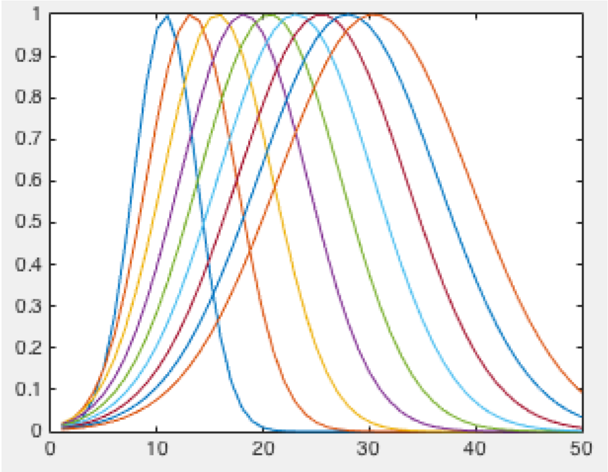
$$\boxed{(\mathbf{A}^T\mathbf{A})^{-1}\ \mathbf{A}^T\ \mathbf{x} = \mathbf{y}^*}$$ "Moore-Penrose Pseudo-inverse"

(Note: setting gradient to 0 and solving is hard to do for non-linear problems…)

Example unmixing with the pseudo-inverse

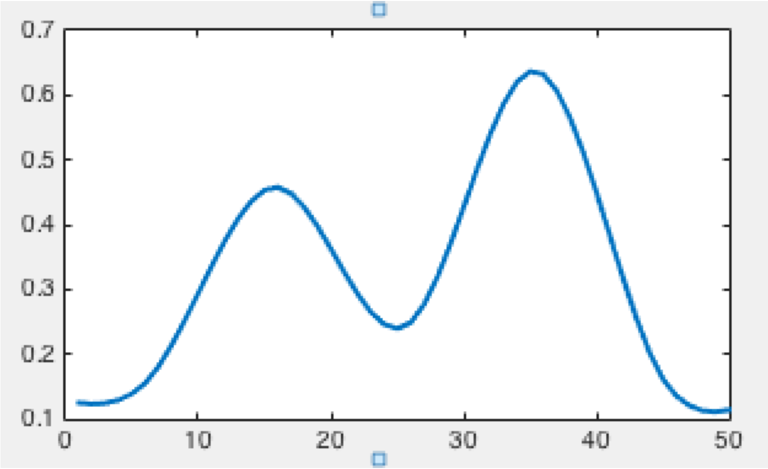
Moore-Penrose Pseudo-inverse: $\boxed{y^* = (A^T A)^{-1} A^T x}$

Example dictionary A
9 spectra



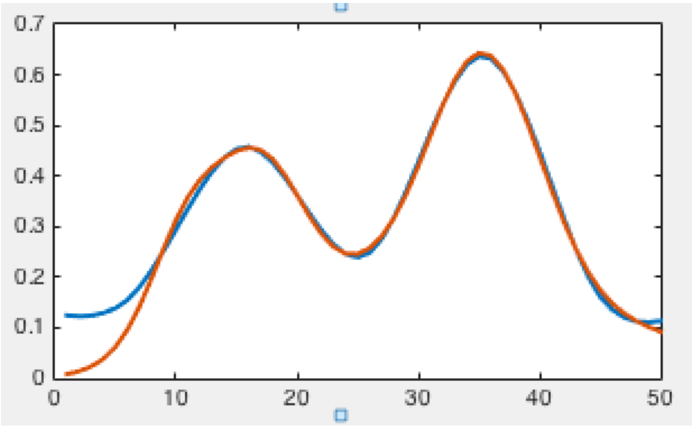
Example y, compute Ay

Example detected spectra x



Compute pseudo-inverse, x*=Ay* is red curve:

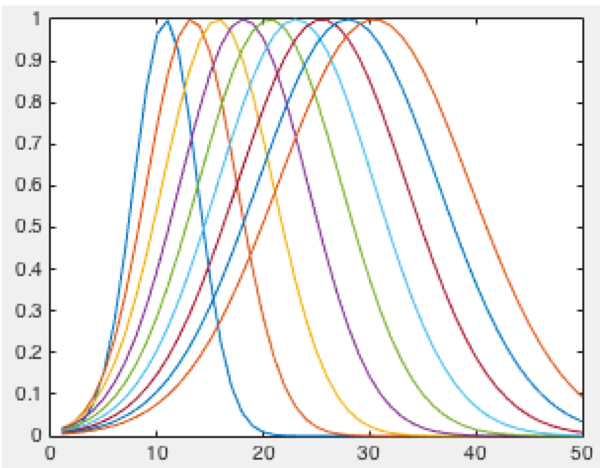
Good fit!



Example unmixing with the pseudo-inverse

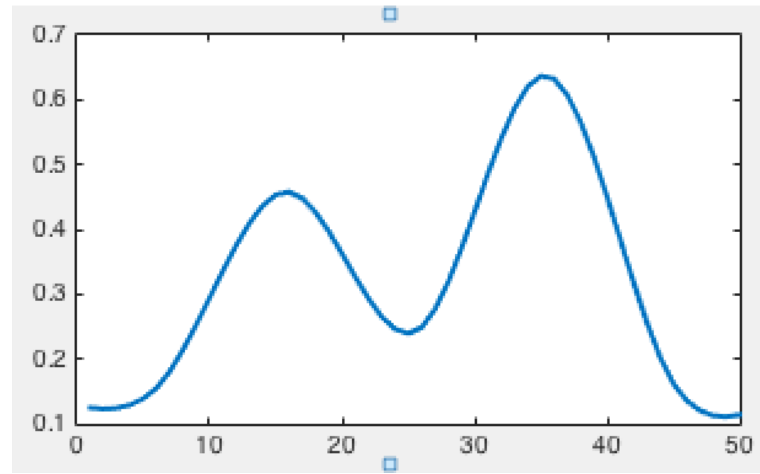
Moore-Penrose Pseudo-inverse: $\boxed{y^* = (A^T A)^{-1} A^T x}$

Example dictionary A
9 spectra



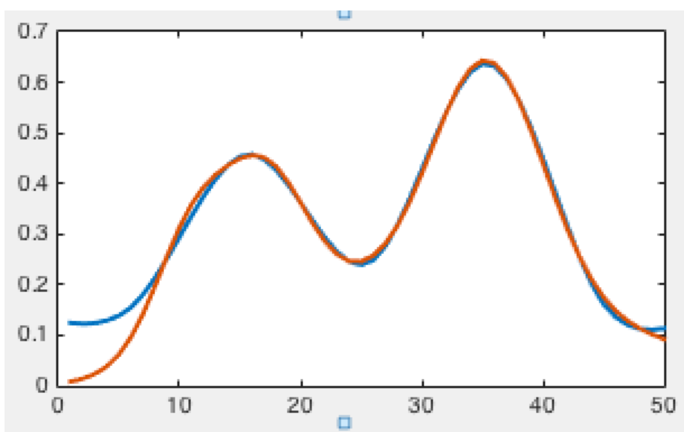
Example y, compute Ay

Example detected spectra x



Compute pseudo-inverse, x*=Ay* is red curve:

Good fit!



PROBLEM:

y* = [0.2, -1.1, -1.6, …]

Solution has negative weights!

Not physically possible…

Example unmixing with the pseudo-inverse

deep imaging

Moore-Penrose Pseudo-inverse: $\boxed{\mathbf{y}^* = (\mathbf{A}^T\mathbf{A})^{-1}\,\mathbf{A}^T\,\mathbf{x}}$

```
n = 50; %number of pixels
m = 9; %number of spectral
A=zeros(n,m); %known dictionary of spectra
for j=1:m
    A(:,j) = exp(-(linspace(-1,1,n)+.5-.1*j+.2).^2/(.03*j));
end
%Simulate some spectra
b = imresize(rand([5,1]),[n 1]);
x_opt = A\b;
%Show results
figure;plot(b); hold all; plot(A*x_opt);
```

Pseudo-inverse = one line

Spectral un-mixing with a positivity constraint

deep imaging

Option 1: Add a constraint

Minimize $f(\mathbf{y}) = \Sigma\ (\mathbf{x} - \mathbf{A}\mathbf{y})^2$ — Convex cost function

Subject to $\mathbf{y} >= 0$ — Convex constraint

*When you have constraints, can use **CVX**, convex toolbox for Matlab

http://cvxr.com/cvx/

Spectral un-mixing with a positivity constraint

deep imaging

Option 1: Add a constraint

```
%%%%%%%%%%%%%%
addpath '/users/Roarke/Documents/Matlab/cvx'; cvx_setup;
cvx_begin
    variable xc(m);
    minimize( norm(A*xc-b) );
    subject to
        xc >= 0;
cvx_end
%Show results
figure;plot(b); hold all; plot(A*xc);
%%%%%%%%%%%%%%
```

Spectral un-mixing with a positivity constraint

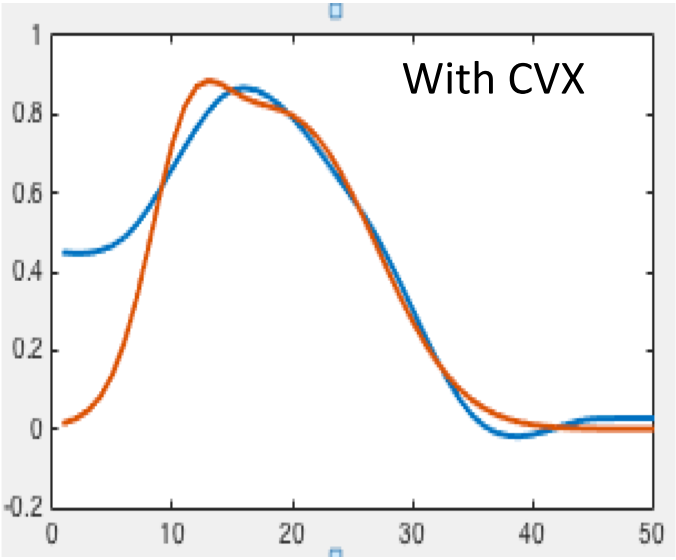
Option 1: Add a constraint

Minimize $f(\mathbf{y}) = \Sigma\ (\mathbf{x} - \mathbf{Ay})^2$ — Convex cost function

Subject to $\mathbf{y} >= 0$ — Convex constraint

*When you have constraints, can use **CVX**, convex toolbox for Matlab

http://cvxr.com/cvx/



y*

0.3612
0.2238
0.0006
0.0000
0.7336
0.0000
0.0000
0.0000
0.0000

Spectral un-mixing with a positivity constraint

deep imaging

Option 1: Add a constraint

Minimize f(**y**) = Σ (**x** – **Ay**)^2 — Convex cost function

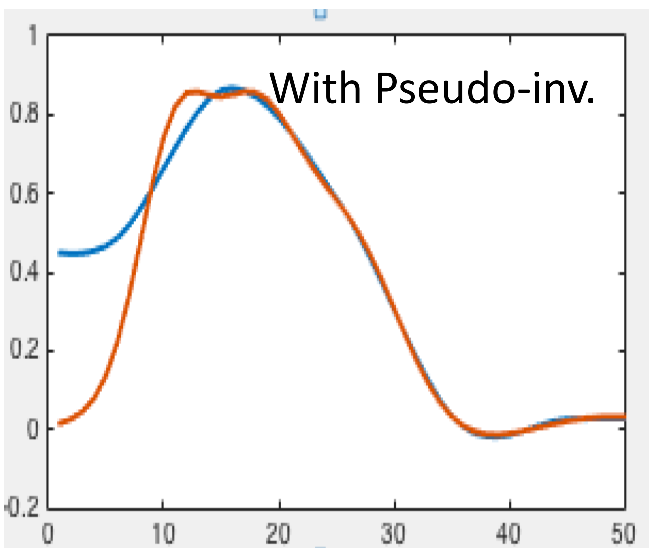
Subject to **y** >= 0 — Convex constraint

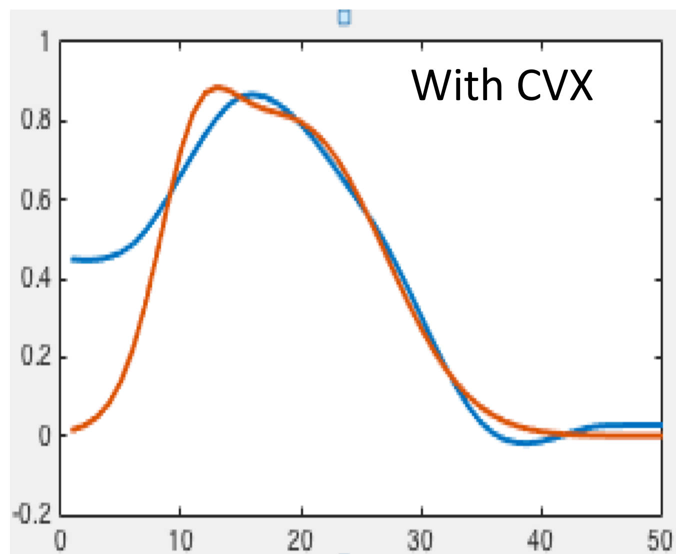
*When you have constraints, can use **CVX**, convex toolbox for Matlab

http://cvxr.com/cvx/

y*

0.6683
-1.5880
5.7848
-11.7459
17.9304
-20.1231
18.3572
-10.7984
2.8557





y*

0.3612
0.2238
0.0006
0.0000
0.7336
0.0000
0.0000
0.0000
0.0000

Spectral un-mixing with a positivity constraint

deep imaging

Option 2: Modify cost function

Minimize $f(\mathbf{z}) = \Sigma \ (\mathbf{x} - \mathbf{A}\mathbf{z}^2)^2$

$\mathbf{z}^2 = \mathbf{y}$ is dummy variable, will change cost function and gradient

*When you don't have constraints but can find the gradient, use **Minfunc**

https://www.cs.ubc.ca/~schmidtm/Software/minFunc.html

Spectral un-mixing with a positivity constraint

deep imaging

Option 2: Modify cost function

Minimize $f(\mathbf{z}) = \Sigma \; (\mathbf{x} - \mathbf{A}\mathbf{z}^2)^2$

$\mathbf{z}^2 = \mathbf{y}$ is dummy variable, will change cost function and gradient

```
%%%%%%%%%%%%%
%3. Minfunc
addpath '/users/Roarke/Documents/Matlab/minFunc_2012';
startVec = ones(m,1);
spectrum_anonymous = @(startVec)spectrum(startVec, b, A);
%Evaluate with minfunc
[xm, msevalue, moreinfo] = minFunc(@(startVec)spectrum_anonymous(startVec), startVec, options);
figure;plot(b); hold all; plot(A*abs(xm).^2);
```

Spectral un-mixing with a positivity constraint

deep imaging

Option 2: Modify cost function

Minimize $f(\mathbf{z}) = \Sigma\ (\mathbf{x} - \mathbf{Az}^2)^2$

$\mathbf{z}^2 = \mathbf{y}$ is dummy variable, will change cost function and gradient

```
%%%%%%%%%%%%%
%3. Minfunc
addpath '/users/Roarke/Documents/Matlab/minFunc_2012';
startVec = ones(m,1);
spectrum_anonymous = @(startVec)spectrum(startVec, b, A);
%Evaluate with minfunc
[xm, msevalue, moreinfo] = minFunc(@(startVec)spectrum_anonymous(startVec), startVec, options);
figure;plot(b); hold all; plot(A*abs(xm).^2);
```

```
function [err_function, grad_function] = spectrum(input_vec, b, A)

%for direct pseudo-inverse - no constraints or dummy
%err_function = norm(A*input_vec - b);
%grad_function = A'*(A*input_vec - b);

err_function = norm(A*abs(input_vec).^2 - b);
grad_function = A'*((A*abs(input_vec).^2 - b) .* conj(A*input_vec));
```

Spectral un-mixing with a positivity constraint

Option 2: Modify cost function

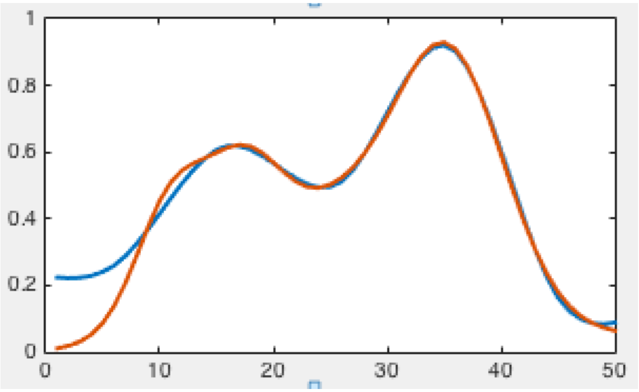
Minimize $f(\mathbf{z}) = \Sigma \ (\mathbf{x} - \mathbf{A}\mathbf{z}^2)^2$

$\mathbf{z}^2 = \mathbf{y}$ is dummy variable, will change cost function and gradient

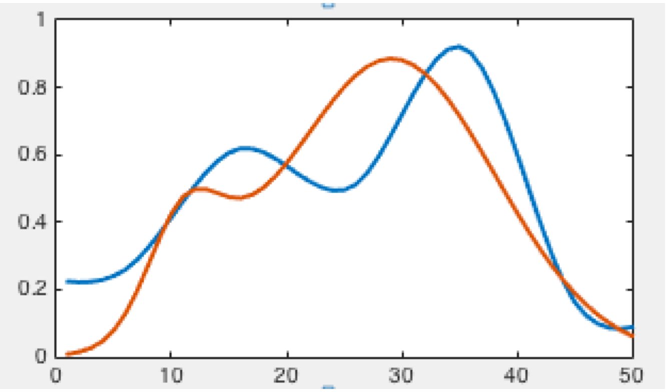
*When you don't have constraints but can find the gradient, use **Minfunc**

https://www.cs.ubc.ca/~schmidtm/Software/minFunc.html

Pseudo-inverse



Dummy variable w/ minfunc



y*

0.5013
0.3345
0.1811
0.0367
0.0132
0.0705
0.2626
0.5080
0.7539

Not working too well, gradient could be wrong?

Additional features that are commonly encountered

deep imaging

1) Sometimes see solutions where x values get really big

Fix this with a "*regularizer*":

$$\text{Minimize } f(\mathbf{x}) = \Sigma \; (\mathbf{x} - \mathbf{A}\mathbf{y})^2 + \boxed{C^{*}\Sigma \; (y)^2}$$

"Don't let y vary too much"

Choose constant C appropriately

Additional features that are commonly encountered

deep imaging

1) Sometimes see solutions where x values get really big

Fix this with a "*regularizer*":

$$\text{Minimize } f(\mathbf{x}) = \Sigma\ (\mathbf{x} - \mathbf{Ay})^2 + \boxed{C^*\Sigma\ (y)^2}$$

"Don't let y vary too much"

Choose constant C appropriately

2) If you think your signal is "sparse", then it probably has mostly zeros.
Can include this in your model with an "L1" cost function:

$$\text{Minimize } f(\mathbf{y}) = \Sigma\ |\ \mathbf{x} - \mathbf{Ay}\ |$$

- An extremely simple modification with pretty strong implications