CogSci 109: Lecture 11

Monday Oct 29, 2007

Return to classes - changes in the course plan, basic fits - regression, linear least squares

Outline for today

- Announcements
  - **Addressing the devastating fires**
    - How is our plan changing?
    - How is our plan staying the same?
- Outline for today
  - **Reminder custom colormap demo**
  - **Basic data fits**
    - Least squares minimization
    - Linear models and regression
      - **Introduction**
      - **Examples**
      - **Matlab implementation**

Announcements

- Wildfires in San Diego
- Hw 3 due date is today, but it was going to be due Monday of last week, so you should at least have been mostly done
  - **If you have special needs in terms of time, come speak with me after class or in office hours**
  - **If you are considering dropping, please discuss it with me first**
- Reading for least squares and other fits

Update: the big picture

- Where we are
  - 4 parts of the course
    - We discussed data
      - What is it, how do we manipulate it, matlab implementation
      - Filtering
      - Computing basic statistics
    - We discussed basic visualization
      - Plotting data (2d, 3d, colormaps)

Update: the big picture (II)

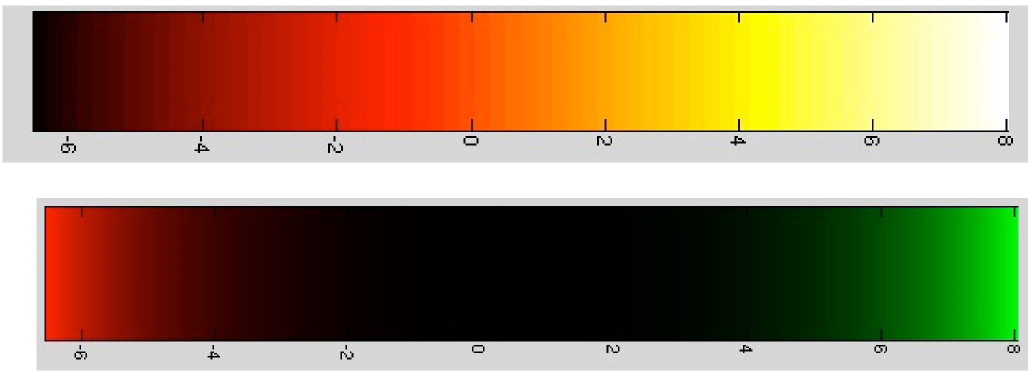
- Where we're going
  - We will now cover
    - **Modeling**
      - what is modeling?
    - **Error analysis**
      - How good is your model?

Update: the big picture (III)

- Where we're going (continued)
  - **What we're going to cover**
    - Basic models
      - **Linear fits, nonlinear fits**
      - **Regression**
      - **Relationship to machine learning**
      - **Interpolation/extrapolation (also data analysis methods)**
    - Advanced models and modeling methods
      - **Fitting models with optimization methods**
      - **Artificial neural networks**
      - **AI**
    - Communicating results
      - **This has been integrated and will continue to be integrated**
      - **Proper forms of inserting figures and tables in scientific communications**
      - **Format in homeworks is designed to teach proper communication methodology**

Creating color maps - review and expansion

- What if I want to examine the boundaries of my data?
  - **I only want to see the extremes**
  - **We can create a custom color map!**



Creating the color map (r,g,b) components

- To create a custom color map we need to make a matrix which is Dim nx3, range [0,1]
- Each column is the range of either red, green, blue
- Writing it by hand:

$$\boxed{M = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}}$$

- Typing it into a matlab variable:

M = [1 0 0; 0 0 0; 0 0 0; 0 0 0; 0 1 0];

Now what?

- We create our plot, let's create some data:

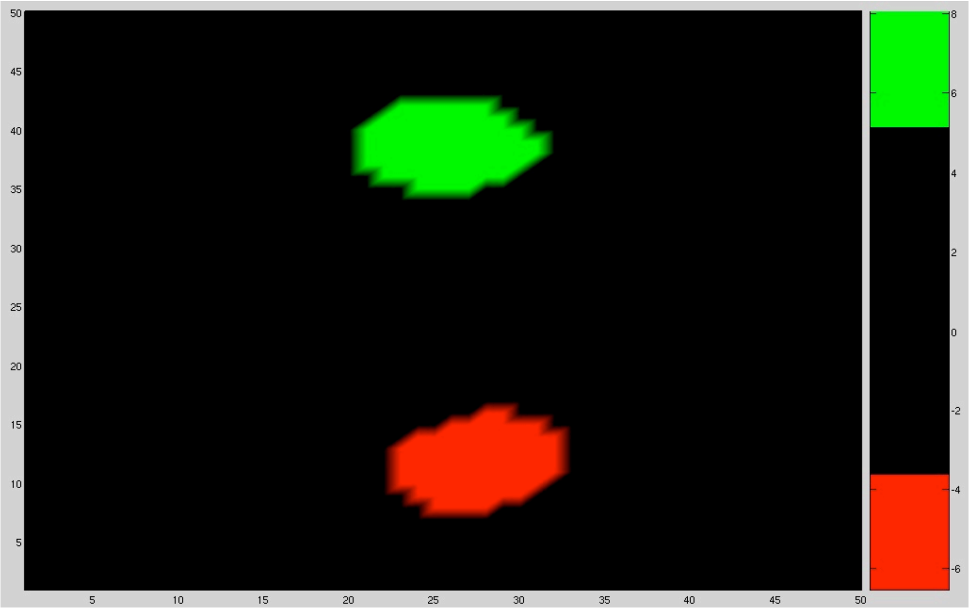
```
X = peaks(50);
```

- And plot it using *pcolor:*

```
pcolor(X)
colormap(M)
```

Here's what we get…

- As you can see this can be very useful for feature detection
- But let's say we want to make a smooth map, how do we do that?



Creating smooth color map functions

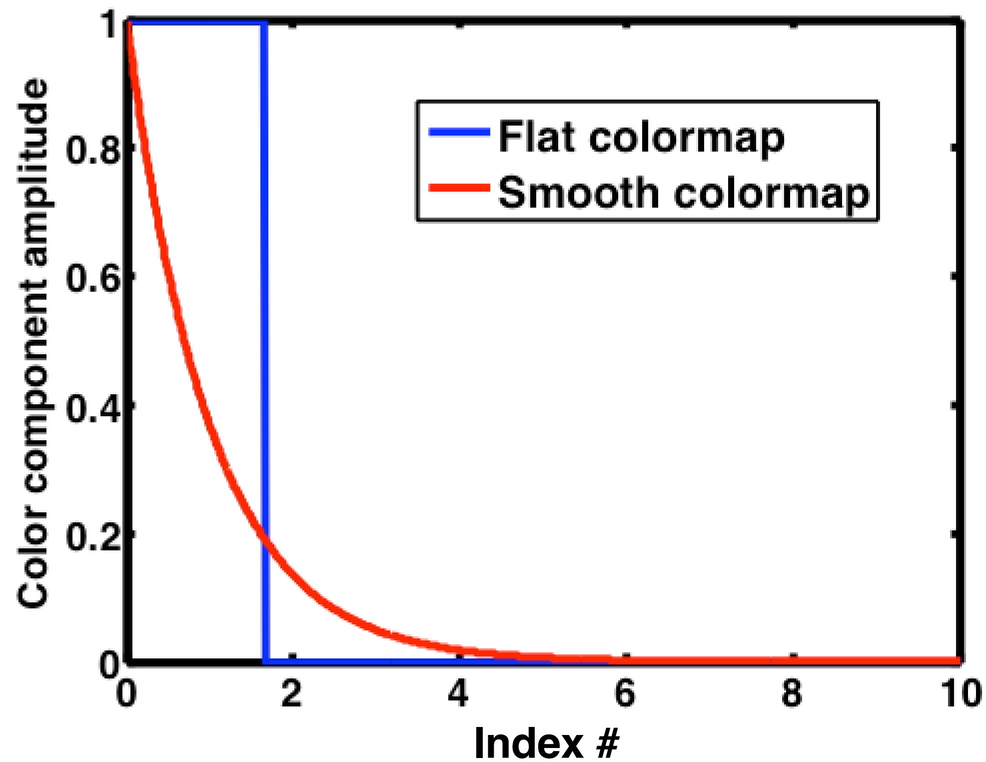
- Instead of typing the matrix in manually, let's construct the functions we need to make transitions smooth from one color to the next
- Create many values in between 0 and 1
- Two things of note
  - **The length of your colormap array is up to you, the more numbers and the smaller the transitions, the more smooth the colors look (crayons vs. airbrushing)**
  - **The colors are mapped so the**

    range(0,1) -> range(min(data), max(data))

Looking at smooth transitions

- Comparison after matching the number of values in the simple color variation (1 -> 0) vs. a smooth function from 1->0
- Uses the equation…
  - **(for Decreasing:)**

$$\boxed{\begin{aligned} r &= \exp(-x) \\ x &= 0 : .01 : 10 \end{aligned}}$$



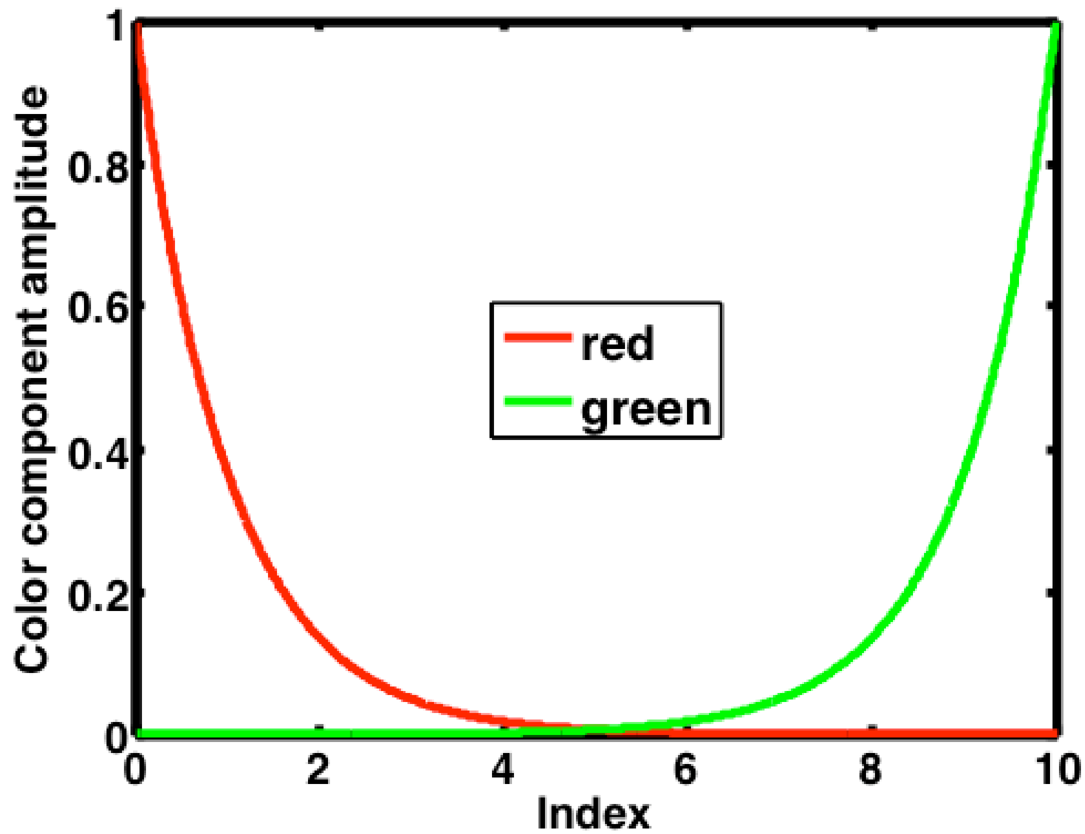
The final smooth color map

- And equations:
  - **Decreasing:**

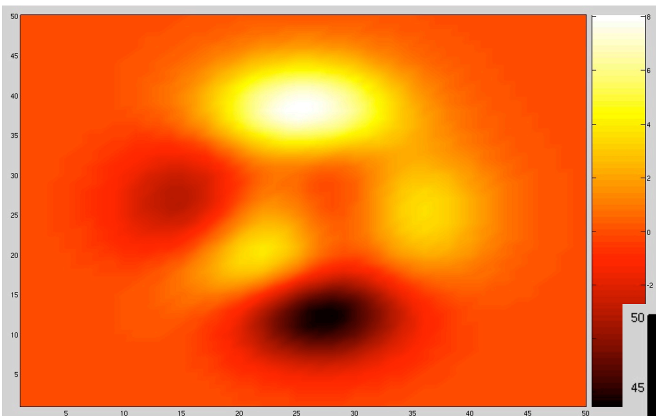
$$\boxed{\begin{aligned} r &= \exp(-x) \\ x &= 0:.01:10 \end{aligned}}$$

  - **Increasing:**

$$\boxed{\begin{aligned} g &= \frac{\exp(x)}{\max[\exp(x)]} \\ x &= 0:.01:10 \end{aligned}}$$

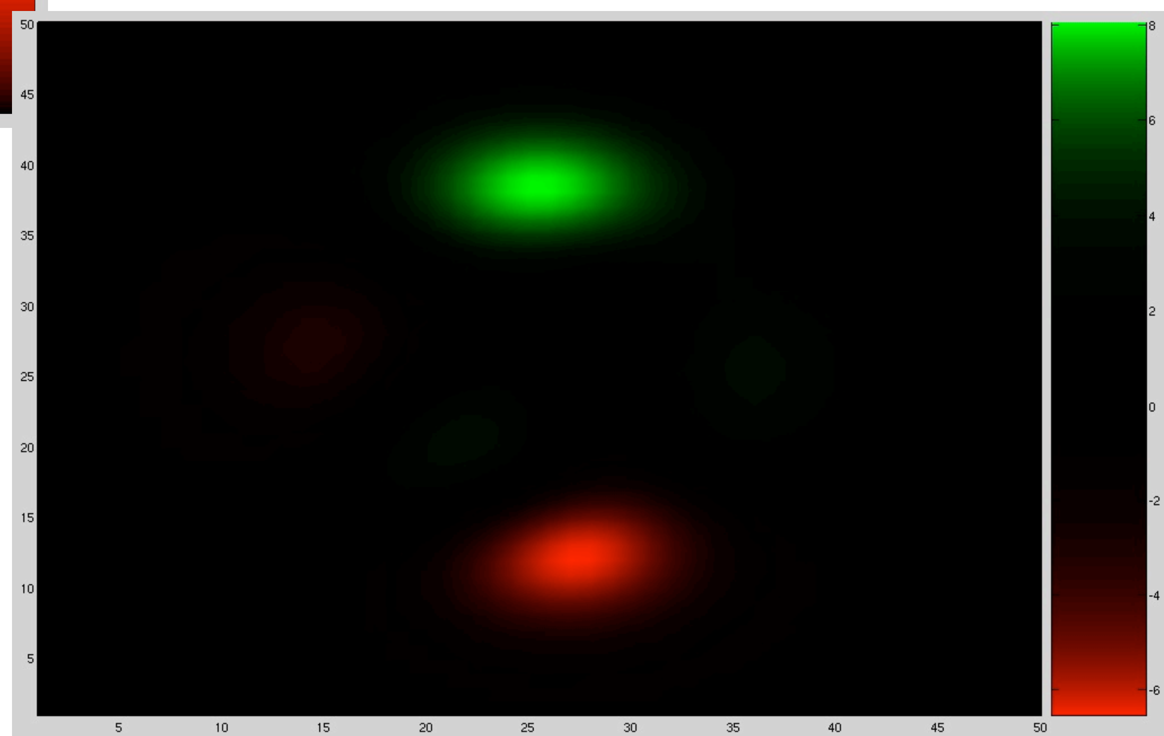


Results of our custom color map



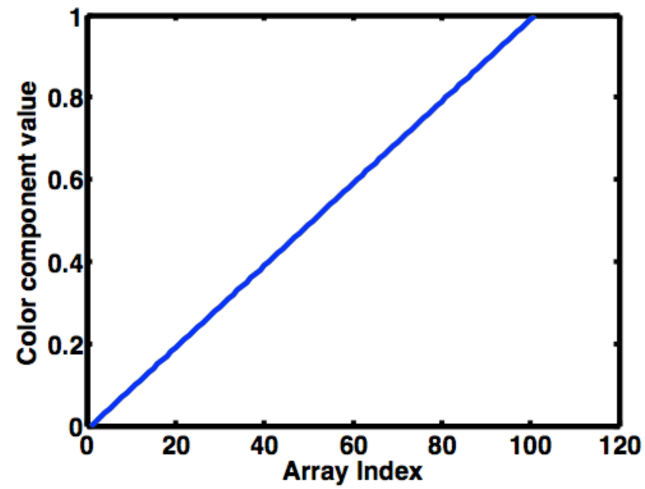
<- Using the built-in ‘hot’ color map

Using our color map->

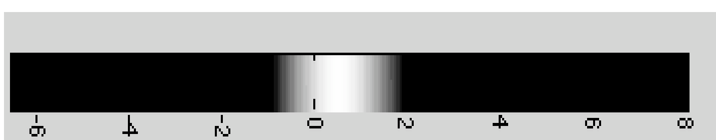


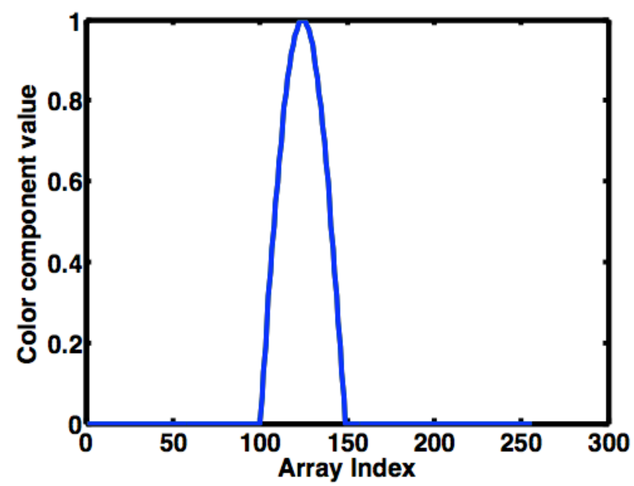
Other plots vs. custom color maps

- Grayscale?



- Compressed?





Matlab implementation…

- To matlab…

Part III: Models and the modeling process

Linear least squares

- You're probably all familiar with linear regression -- fitting a line to a bunch of data.
- more formally fitting $\boldsymbol{y = mx + b}$ for paired x,y data (can also do multidimensional)
- Let’s see how it’s done mathematically

Let's start by considering an easier question…

- We have 2 points, and want to fit a line to them
- (1,2) , (3,4)
- How would you solve this problem?
- We want y=mx+b (we need **m** and **b)**
  - **Substitute each point in**

$$\boxed{\begin{aligned} 2 &= m(1) + b \\ 4 &= m(3) + b \end{aligned}}$$

Example continued

- And solve for **b** first, then **m**

$$b = 2 - m$$

$$4 = 3m + 2 - m$$

$$4 = 2m + 2$$

$$m = 1$$

$$b = 2 - m$$

$$b = 1$$

Example continued

- We have two equations and two unknowns (m, b)
- This can be written compactly as

$$\boxed{\begin{bmatrix} 1 & 1 \\ 3 & 1 \end{bmatrix} \begin{Bmatrix} m \\ b \end{Bmatrix} = \begin{Bmatrix} 2 \\ 4 \end{Bmatrix}}$$

- Which is of the basic form

$$\boxed{Ax = b}$$

- We want to find

$$\boxed{x = A^{-1} b}$$

Solving Ax=b

- Solving for $\boxed{x = A^{-1}b}$ involves computing the inverse of the A matrix
  - **Insiwhatsitz? Don't worry…inverses are a way to make life easier**
- There are several methods, and you can solve for arbitrarily sized problems (ie what if we want to find 100 variables? Not fun by hand:(Let's use a computer to do it for us!!!:)
  - **Gaussian elimination (what you learned in linear algebra class)**
    - Don't worry you won't have to do it by hand in this class!
  - **Thomas algorithm, etc (and other more efficient methods computationally)**
  - **Matlab has gaussian elimination built-in nicely of course**

We need to remind ourselves of matrix inversion

- What is an inverse of a matrix?
- Rotation example
  - **If a vector is rotated by multiplying it by a rotation matrix, then multiplying the rotated vector by the inverse rotates the vector back to its original orientation**
  - **Side note - a matrix times its inverse yields the identity matrix**
    - You can test for a matrix being the inverse of another matrix by multiplying the two, and see how close do you get to the identity matrix?

$$\boxed{AA^{-1} = I} \qquad \boxed{A^{-1}A = I} \qquad \boxed{AI = A} \qquad \boxed{IA = A}$$

      - **Look up more of the definition details…see references on site**
    - Homework problem, one matrix plot is an example…which could it be? Hmm…what special matrices have we just mentioned? Hmmm…how could I IDENTIFY this matrix? Hmmm…
- Dating example

Solving Ax=b

- We compute the solution of our canonical problem by

$$A^{-1}A = I$$
$$Ix = x$$

Recall that…

$$\begin{aligned} Ax &= b \\ A^{-1}Ax &= A^{-1}b \\ Ix &= A^{-1}b \\ x &= A^{-1}b \end{aligned}$$

How to solve Ax=b in matlab

- In matlab this can be solved for with the \ operator
- A\B is the matrix division of B into A
  - **roughly the same as INV(A)*B**
  - **computed in a different way.**
    - If A is an N-by-N matrix and B is a column vector with N components, or a matrix with several such columns, then X = A\B is the solution to the equation A*X = B computed by Gaussian elimination.
- Doing it in matlab:

```
mb= [1 1; 3 1]\[2;4];    %(left matrix divide)
```

Derivation of linear least squares

- <on board>

Another example in matlab

- consider (1,2) (3,4) (2, 3.5)

```
x=[1 3 2]
y=[2 4 3.5]
plot(x,y,'*')
```

- 1m +b =2
- 3m +b =4

- 2m +b = 3.5

Example continued

```
A=[1 1; 3 1; 2 1]
y= [ 2; 4; 3.5]
```

- if we use the m=1, b=1 solution to the first two it doesn't fit the third
- e.g. 3 equations and 2 unknowns
- This is what is known as an overconstrained problem. People commonly like to find the solution that minimizes the mean square error

Example continued

- This means we want to find the solution that minimizes

  \sum_{(x,y) pairs} (y-mx-b)^2

- Matlab again solves this with

```
mb=A\y
hold on
newA=[0 1; 5 1]
plot([0 5],newA*mb)
```