CogSci 109: Lecture 8

Tuesday Oct 17, 2006
Color examples and basic fits

Announcements

- Hw 2 extended to Thurs Oct 19
- Hints page
- Reading for least squares and other fits
- Chancellor's challenge

Aside: You don't have to know everything to understand things

- Your equations and math concepts are ways of expressing the underlying reality
- But the reality is expressed THROUGH that
- The true reality is its own thing, but the graphs, models, equations and so on become expressions of the underlying truth, yielding potential insights
- Do not judge yourself or compare, do not tell yourself you are 'not a math/computer person,' inferior to another, nor should you feel superior
  - Address the reality of the moment
  - If you don't know something, read about it, think, ponder, experience, but do not judge

Update: the big picture

- Where we are
- Where we're going

Homework hints

- Reminder - Hertz
  - abbreviated Hz = cycles/second
- The hints page - check for your answer there
- Problems with crashing?
  - At the command prompt type

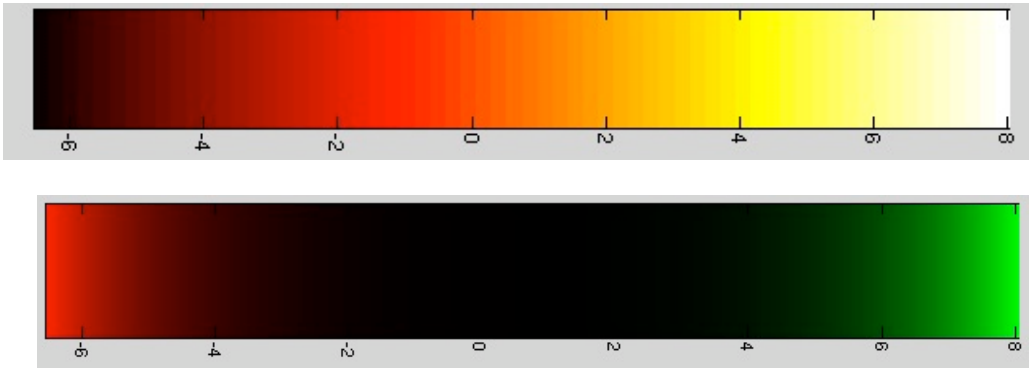
```
opengl neverselect
```

More motivation for math and programming

- Why so much math? How does that relate to programming again?
  - Math provides a

Some specific matlab color map examples

- 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
- We could write it by hand:

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

- Type 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:

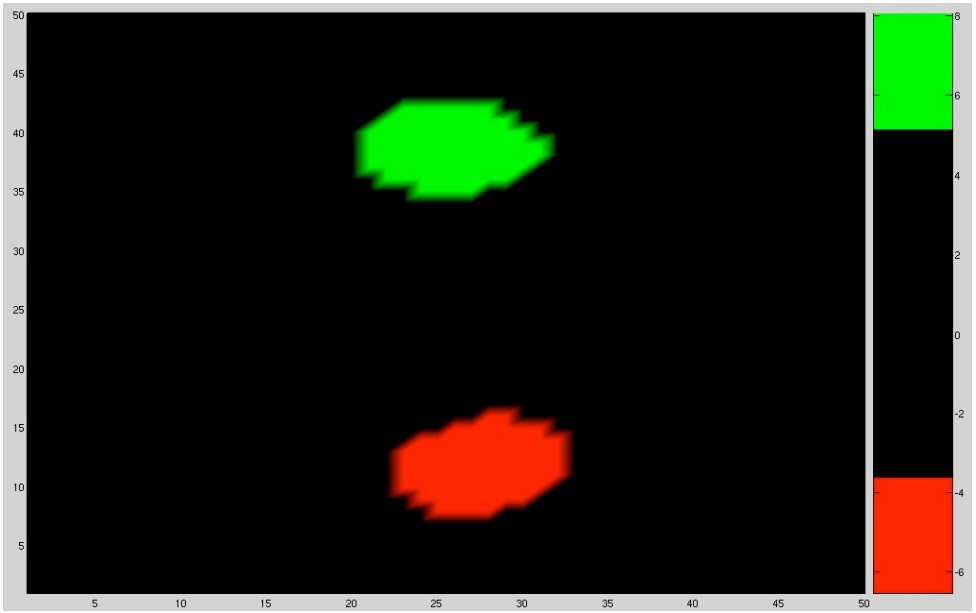
```
X = peaks(50);
```

- 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

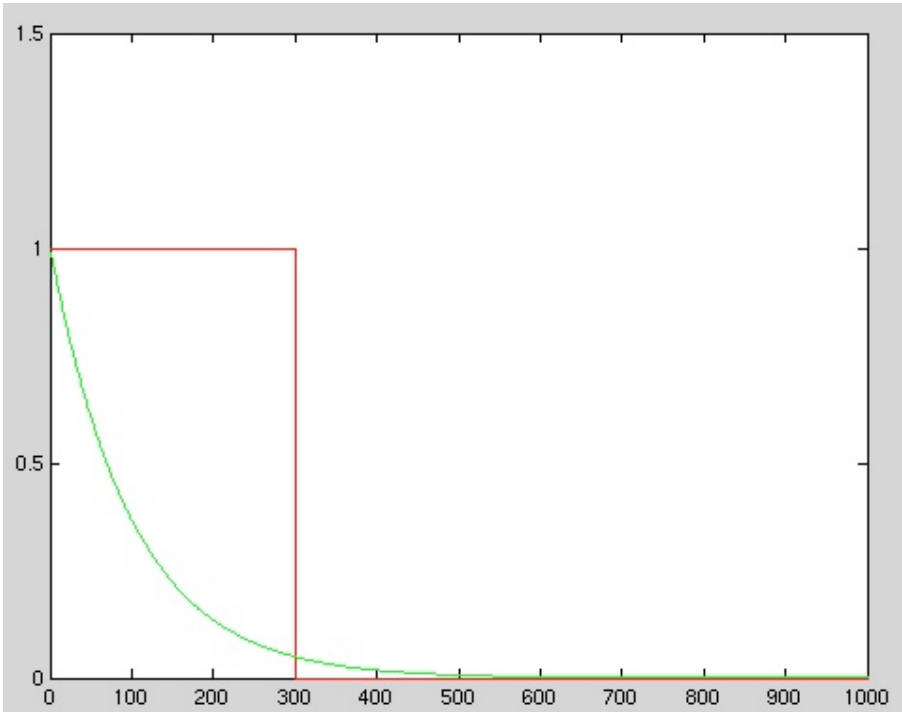
    range(0,1) -> range(min(data), max(data))

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:)

$$r = \exp(-x)$$

$$x = 0:.01:10$$



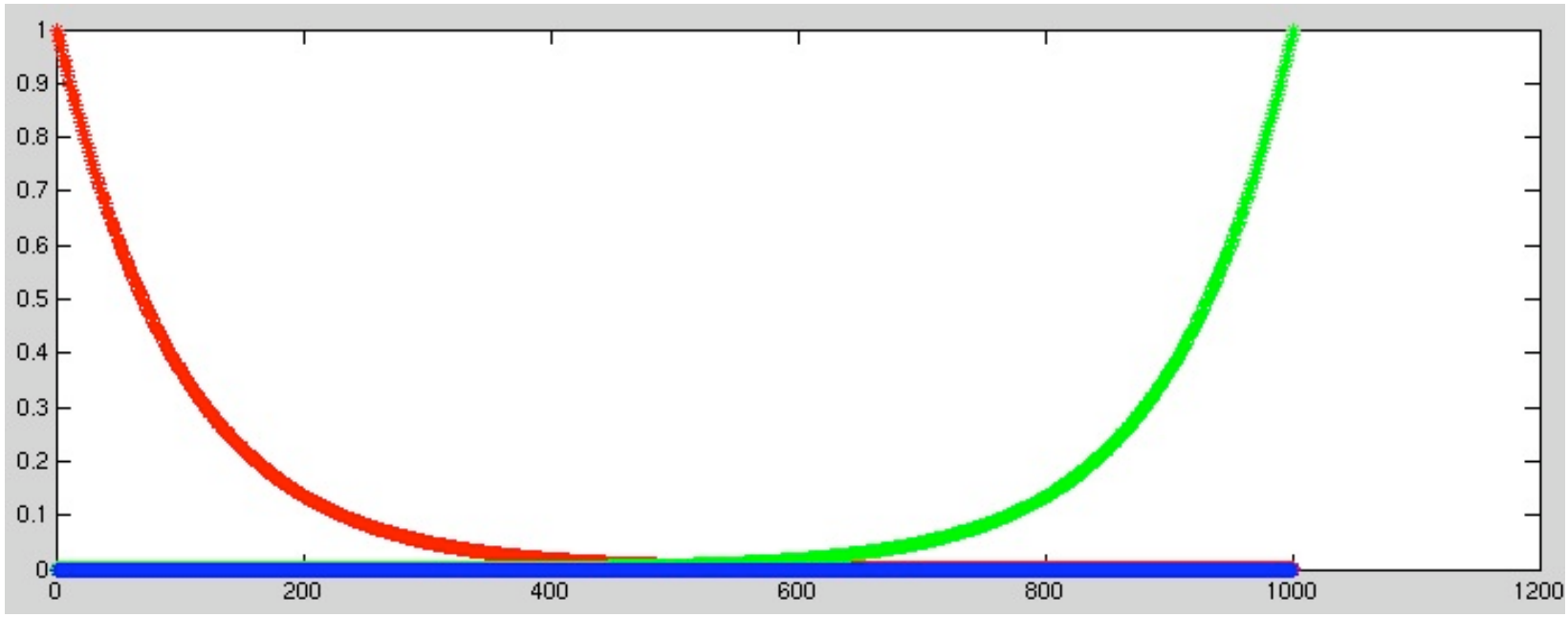
The final smooth color map

- And equations:
  - Decreasing:

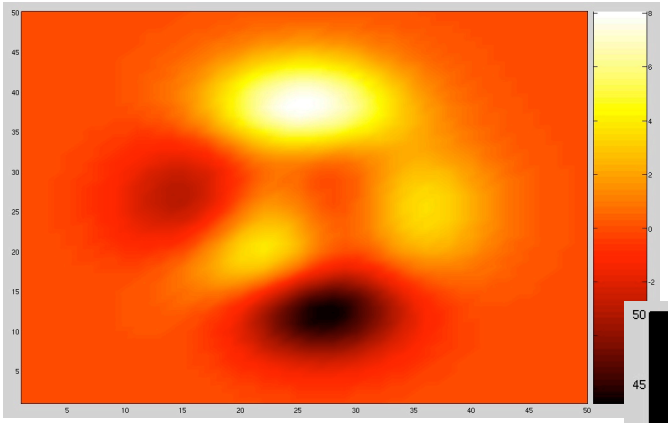
$$r = \exp(-x)$$
$$x = 0:.01:10$$

-Increasing:

$$g = \frac{\exp(x)}{\max[\exp(x)]}$$
$$x = 0:.01:10$$

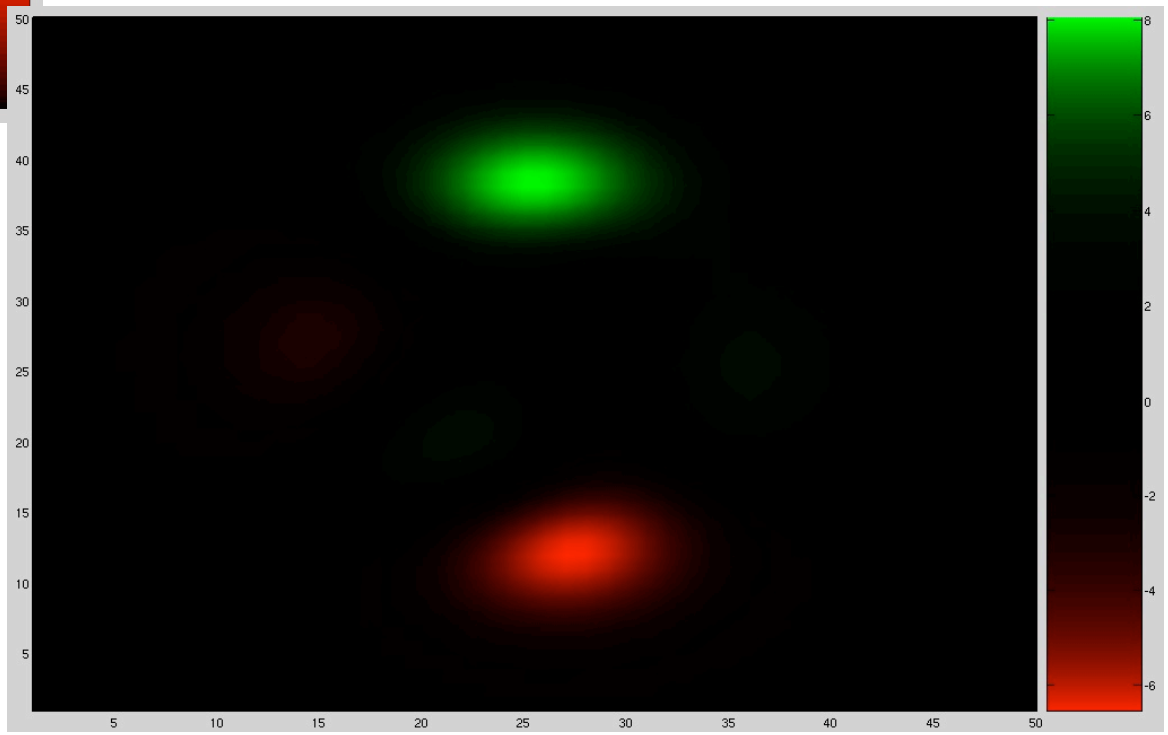


Results of our custom color map



<- Using the built-in ‘hot’ color map

Using our color map->



Matlab implementation…

- To matlab…

Covariance clarifications

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

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

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

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

- Which is of the basic form

$$Ax = b$$

- We want to find

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

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

How to solve Ax=b in matlab

- In matlab this can be solved for with the \ operator
- A\B is the matrix division of A into B
  - 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)
```

Correlation

Correlation as a measure of fit

- Closer to 0 worse fit
- Closer to 1 better fit

But what if the data isn't linear?

- Nonlinear least squares
  - Polynomial fit
  - Exponential fit