Matlab, part 3: Matrix math + 3D matrices

Matrices and weighted sums

\[
\begin{bmatrix}
1 & 4 & 1 & 0 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 \\
1 \\
0 \\
0 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & 1 & 0 & 1 & 1 & 1 \\
-1 & 0 & 1 & 0 & 1 & 1 \\
\end{bmatrix}
\]

Left Matrix columns times Right matrix numbers

\[
\begin{bmatrix}
v_1 & v_2 & v_3 \\
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
\end{bmatrix}
= xv_1 + yv_2 + zv_3
\]

Assuming right matrix is a single column
In general, # of left matrix columns must equal # of right matrix rows

Matrices in matlab

```
A= [1 2; 3 4];
b= [4; 5];

What is A*b?     [14 ; 32 ]
```

Transpose: [4; 5] == [4 5]'
a' flips rows and columns
"Solving for x"

\[ A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad c = \begin{bmatrix} -2 \end{bmatrix} \]

\[ Ax = c \]

What is x?

\[ x = A^{-1}c = \begin{bmatrix} 1 \\ 2 \end{bmatrix}^{-1} \begin{bmatrix} -2 \end{bmatrix} = \begin{bmatrix} 2 \end{bmatrix} \]

\[ \text{inv}(A) \ast c \]

LOC localizer: experimental design

Each second:
- new object OR
- new noise OR
- “blank screen” (fixation)

Building the voxel response

Voxel response neuroData
at each second neural response to stimuli
timesObj, at each second:
- 0 for no-object,
- 1 for yes-object
- Drift and offset

Design matrix \( M \)
- On/off information \( O \)
- Constant offset \( C \)
- Linear drift \( L \)

Measured voxel output \( v \)

\[ v = [v_{t=1}, v_{t=2}, ..., v_{t=93}]^T \]

\[ M = \begin{bmatrix} O & C & L \\ \end{bmatrix} \]

\[ M \begin{bmatrix} \beta_o \\ \beta_c \\ \beta_L \end{bmatrix} = v \quad B = M^{-1}v \]
Matlab code

% Want: k1*O+k2*C+k3*L=sigOut1;
% I.e.: M*kVec = sigOut1;
% Define M "design matrix"
M=[ blockPatt; driftL; driftC];
% M is 3 x 93, we want 93 x 3 matrix
Mtrans=M';
% inv only works on square mat -
% num rows=num cols,
% pinv on any matrix
x=pinv(Mtrans)*sigOut1;

More Matlab code

% Find patterns in sigOut1
M= [ blockPatt; driftL; driftC];
Mtrans=M';
x=pinv(Mtrans)*sigOut1;

% Patt-correct time blocks,
% Patt2-unrelated time blocks
M=[blockPatt;driftL;driftC; blockPatt2];
Mtrans=M';
x4=pinv(Mtrans)*sigOut1;

Matrices in n dimensions

x=[1 2 3; 4 5 6]
[1 2 3]
[4 5 6]
y(:,:,1)=[1 2; 3 4]
y(:,:,2)=[5 6; 7 8]
y(:,:,3)=[9 10; 11 12]
size(y) -> [2 2 3]

Typical brain data : location of neuron (x,y,z) + time

Heat-maps

imagesc(Data) – view 2D matrix of scaled data as image
• Red/yellow is highest value, blue is lowest value

Visualize a 2D slice of brain data (size(brainData) -> 128x128x88)
slice=squeeze(brainData(:,:,20)) -> slice 20 of brain
imagesc(slice)
figure -> opens new plotting window

subplot(r,c,i) -> creates grid of plots with
• r rows
• c columns
• fill in position i

currSlice = squeeze(brainData(:,:,10));
subplot(1,3,1); imagesc(currSlice);
currSlice = squeeze(brainData(:,:,20));
subplot(1,3,2); imagesc(currSlice);
currSlice = squeeze(brainData(:,:,30));
subplot(1,3,3); imagesc(currSlice);

squeeze out 1-entry dimensions

currSlice = squeeze(brainData(:,60,:));
% currSlice has size 128x1x88
% won't be plotted by imagesc
% - expects 2D matrix

currSlice = squeeze(currSlice);
% now currSlice has size 128x88

Distribution of matrix values
S1brainVec = S1brain(:);
% convert entries to 1 long vector
hist(S1brainVec)
% see distribution of values
% see most values below 400,
% few outliers
S1brain(find(S1brain>400)) = 400;
% pull outliers down to 400
imagesc(S1brain(:,:,40)) %replot

With looping
for i=1:8,
currSlice = squeeze(brainData(:,:,i*10));
subplot(1,3,i), imagesc(currSlice);
end;
Scaling vs. not-scaling

`imagesc(Data)` – view 2D matrix of scaled data as image
- Red (or yellow) is highest value, blue is lowest value

`image(Data)` – view 2D matrix of data as image
- Red (or yellow) is 64 or higher, blue is 0 or lower

\[ \text{slice} = \text{squeeze}(	ext{brainData}(:,:,10)); \]
\[ \text{figure; imagesc(slice); vs image(slice)} \]