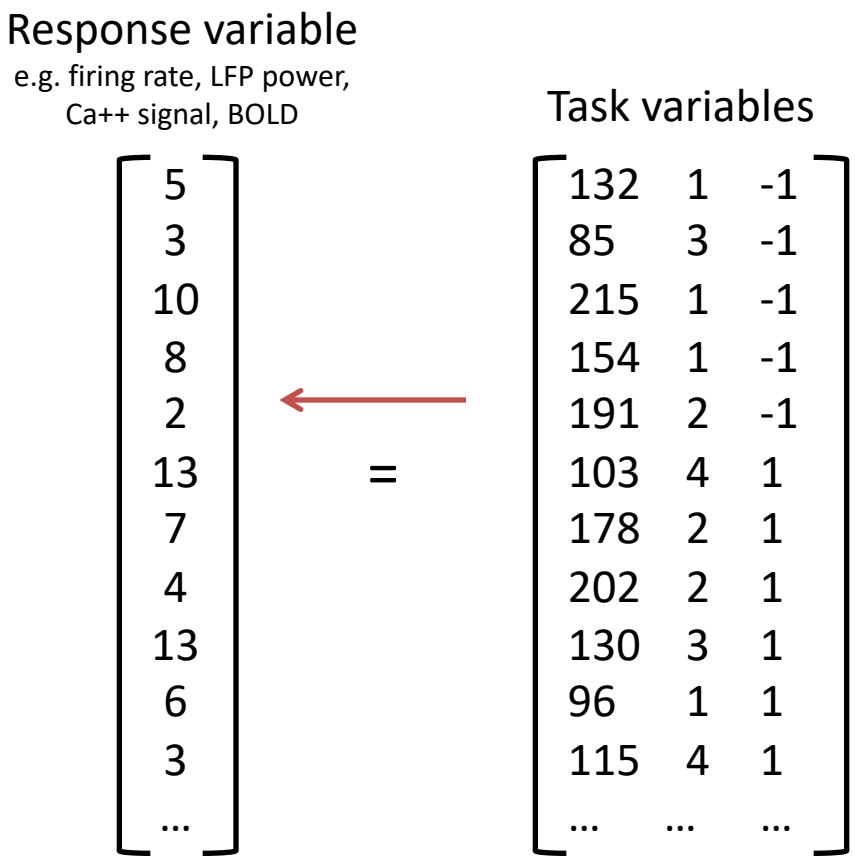
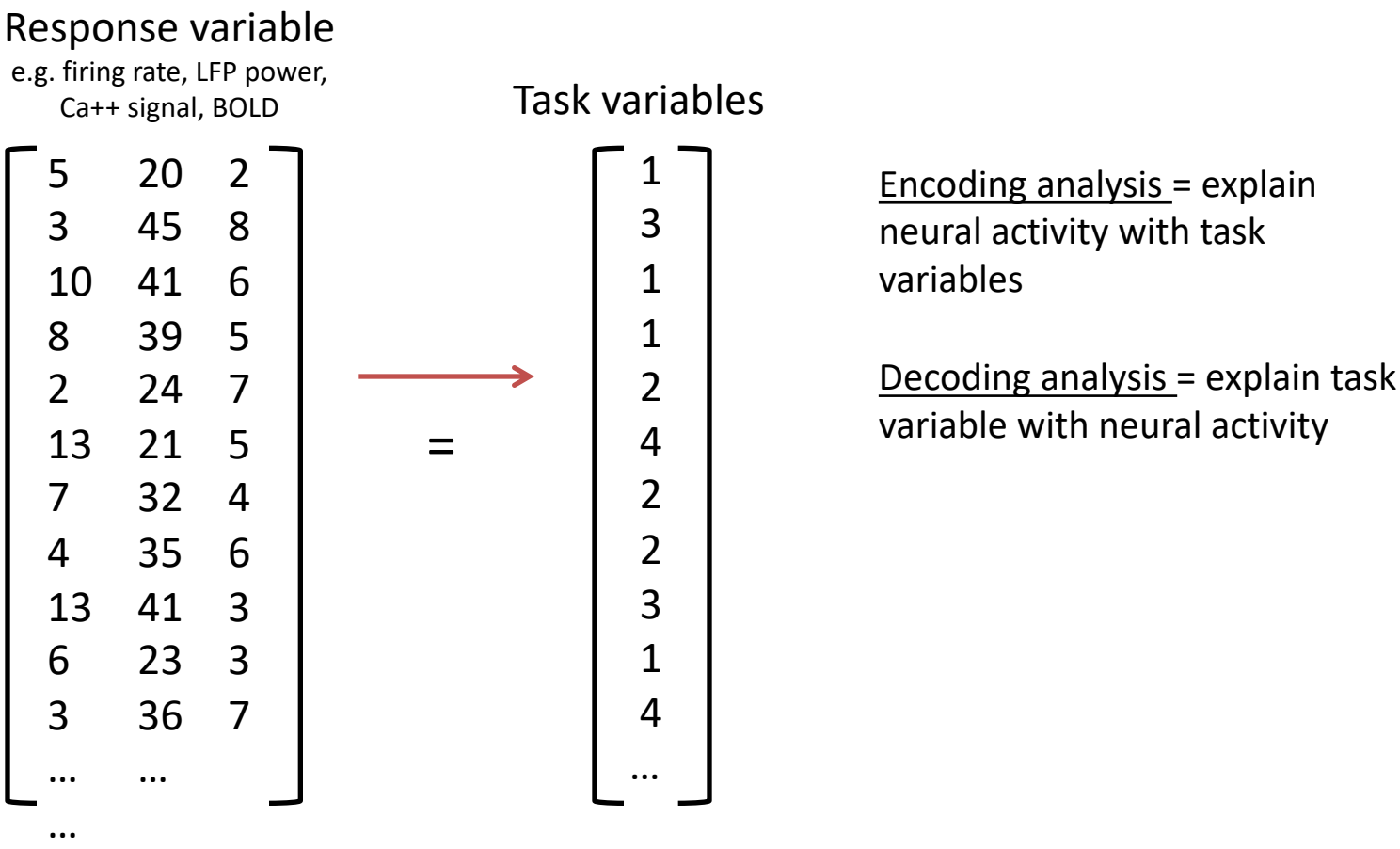
Population codes & decoding

Encoding vs. Decoding



Encoding analysis = explain neural activity with task variables

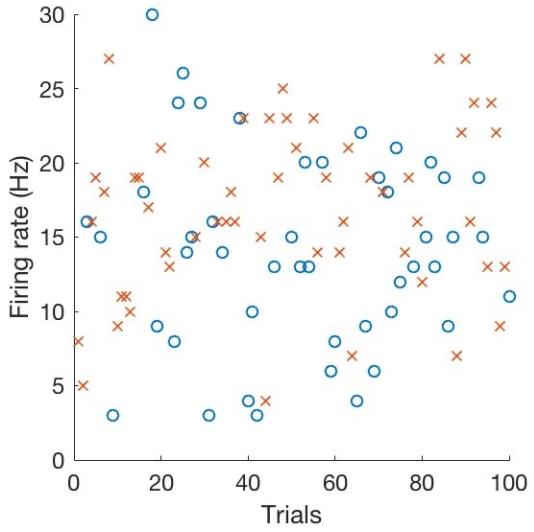
Encoding vs. Decoding



Encoding analysis = explain neural activity with task variables

Decoding analysis = explain task variable with neural activity

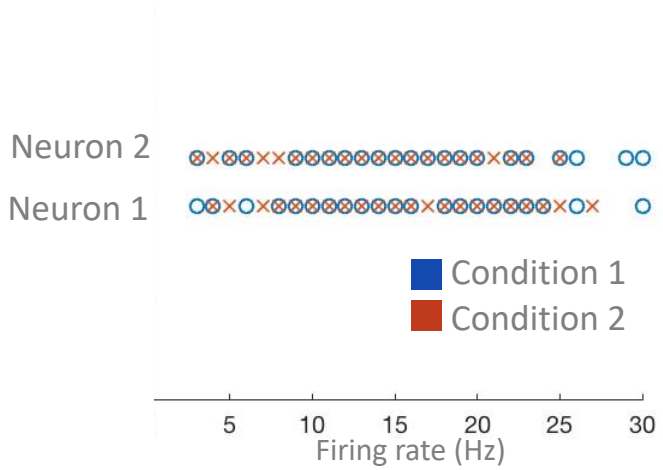
Why populations?

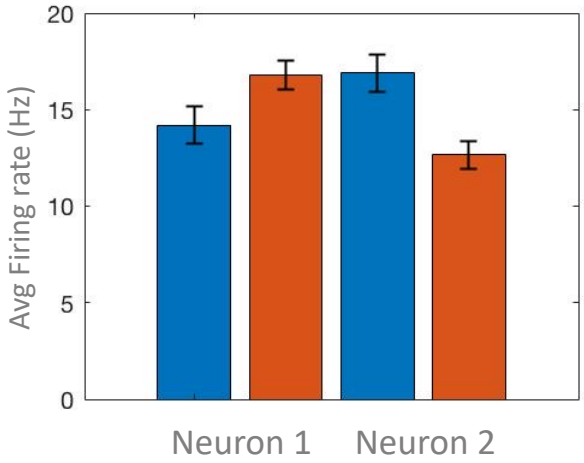


- 1 measure per trial
- each trial has a label (e.g. condition)

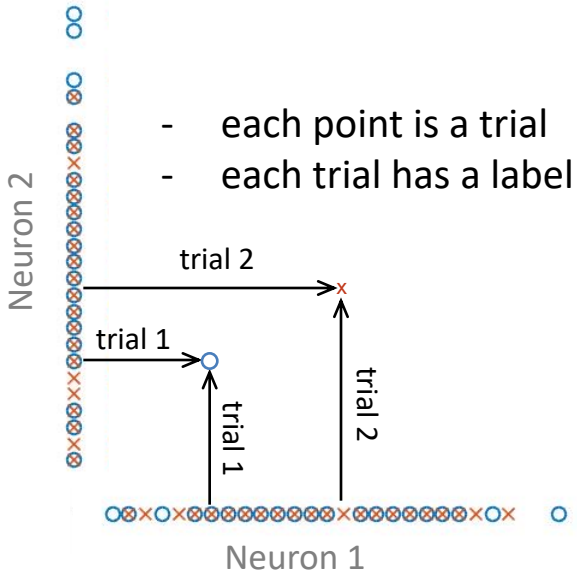
$$\begin{bmatrix} 5 \\ 3 \\ 10 \\ 8 \\ 2 \\ 13 \\ 7 \\ 4 \\ 13 \\ 6 \\ 3 \\ \dots \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \\ 1 \\ 2 \\ 2 \\ 2 \\ 2 \\ 1 \\ 1 \\ 2 \\ \dots \end{bmatrix}$$

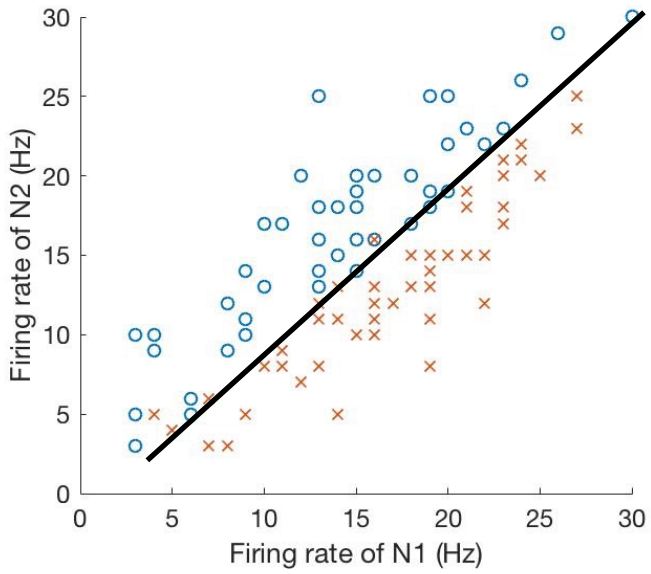
Why populations?



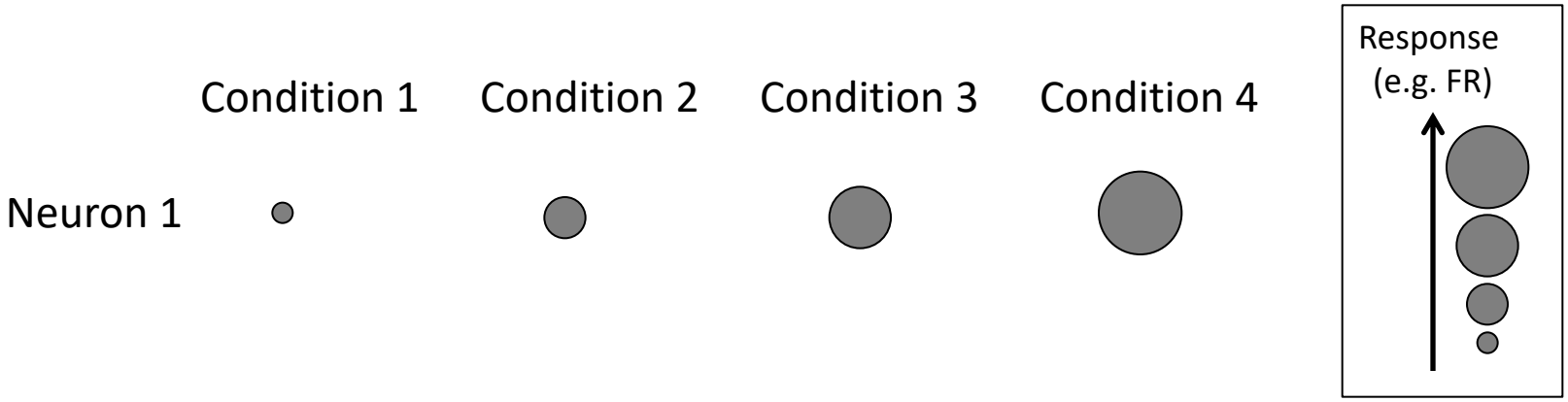


- each point is a trial
- each trial has a label

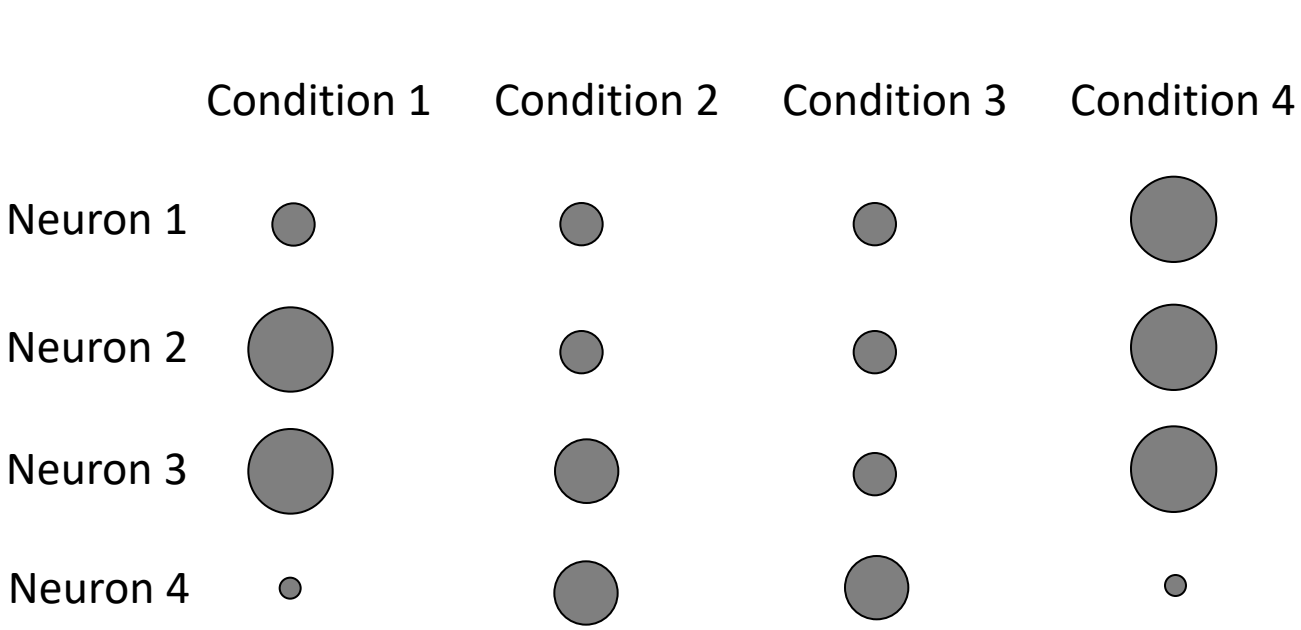


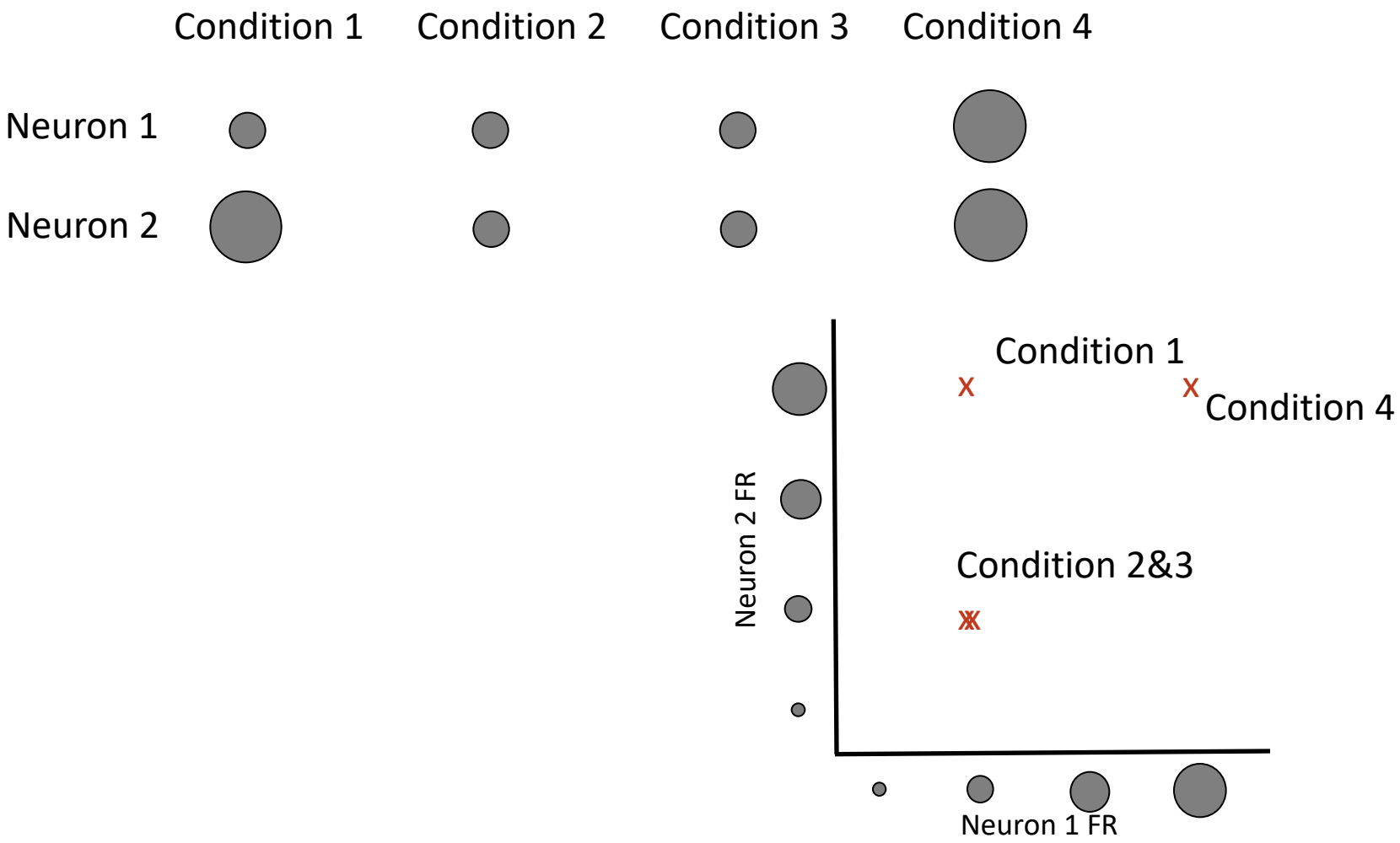


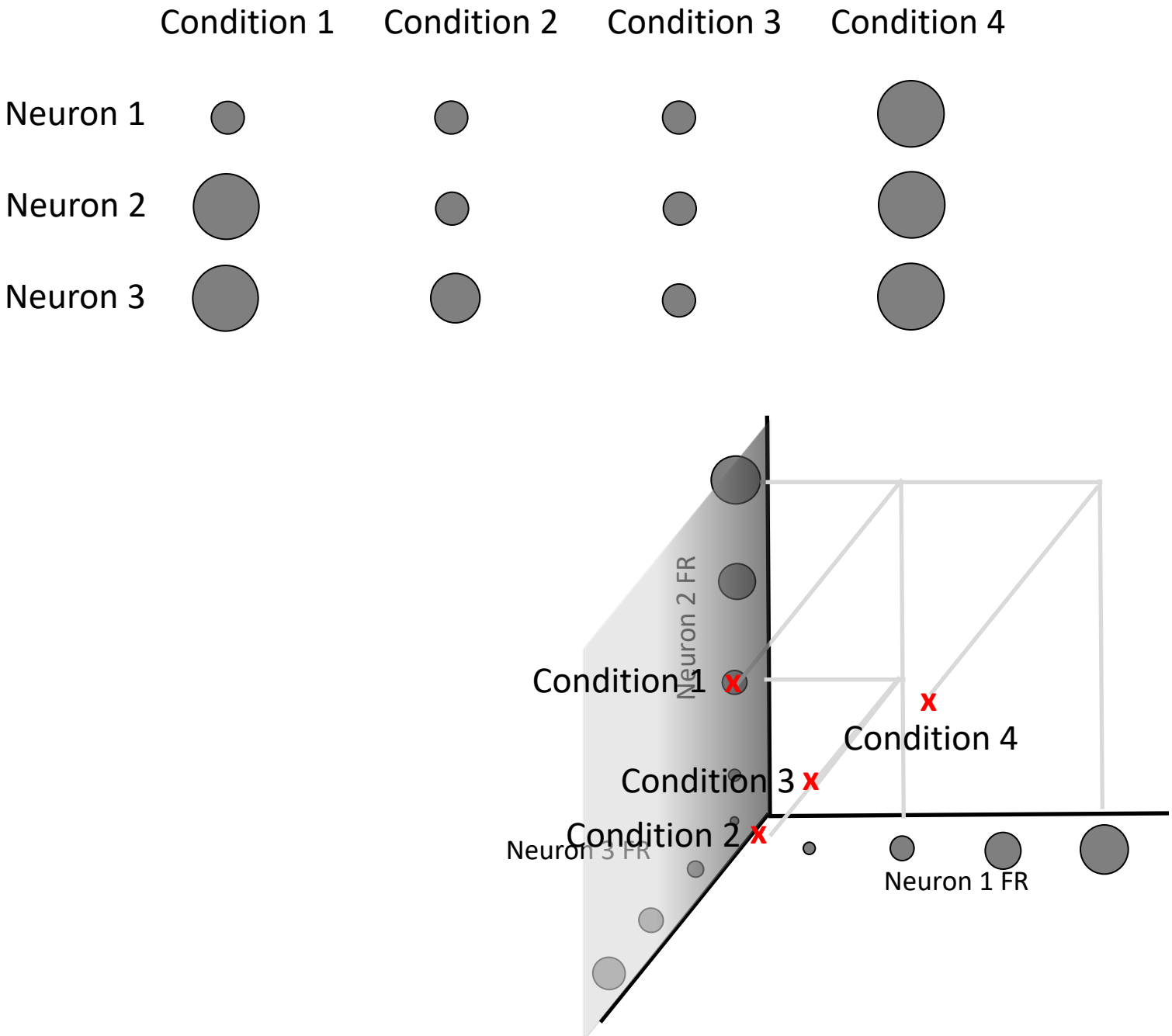
More on population codes



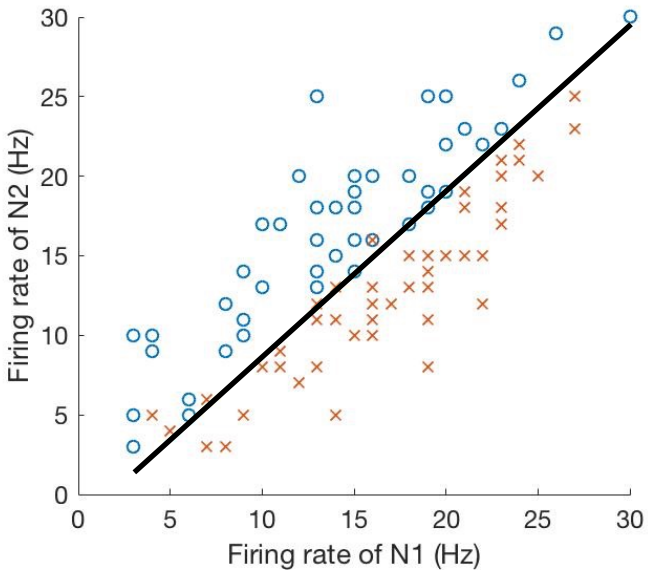
More on population codes




Condition 1
Condition 2
Condition 3
Condition 4
Neuron 1
Neuron 2
Condition 1
Condition 4
Condition 2&3
Neuron 2 FR
Neuron 1 FR


Condition 1
Condition 2
Condition 3
Condition 4
Neuron 1
Neuron 2
Neuron 3
Neuron 2 FR
Condition 1
Condition 4
Condition 3
Condition 2
Neuron 3 FR
Neuron 1 FR

Data can be represented in high-dimensional space, in which each feature (e.g. neuron, channel, voxel) is an axis (dimension)



Axes = “features”

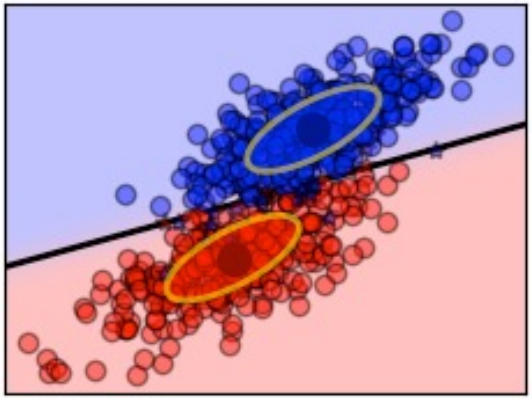
Linear classifiers find the line (2D) or hyperplane (>2D) that best separates the data into categories

Features are weighted to optimize accuracy

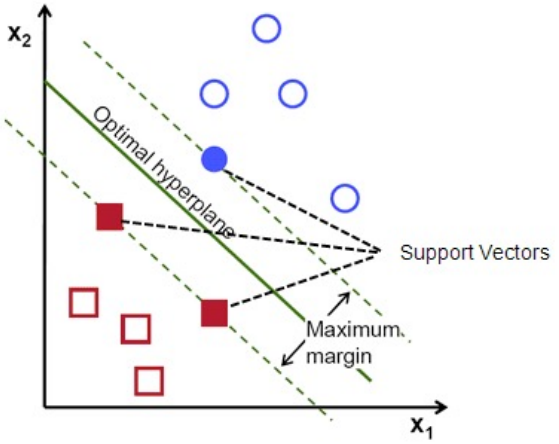
“Supervised” analyses

e.g.
Linear Discriminant Analysis (LDA)
Support Vector Machines (SVM)
Naïve Bayes Classifiers
MVPA

LDA vs. SVM



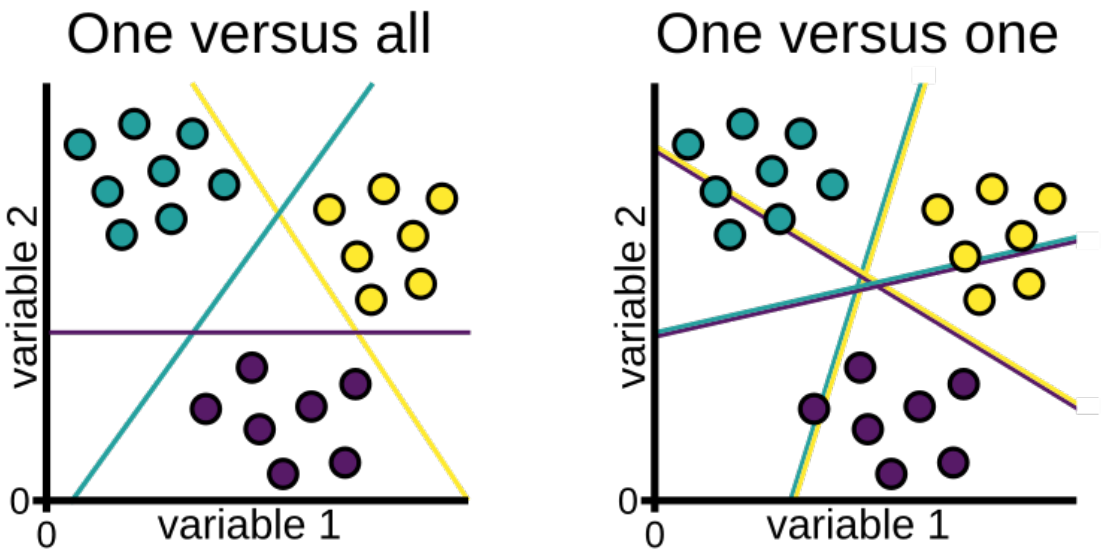
LDA estimates data in each category as a high-dimensional Gaussian



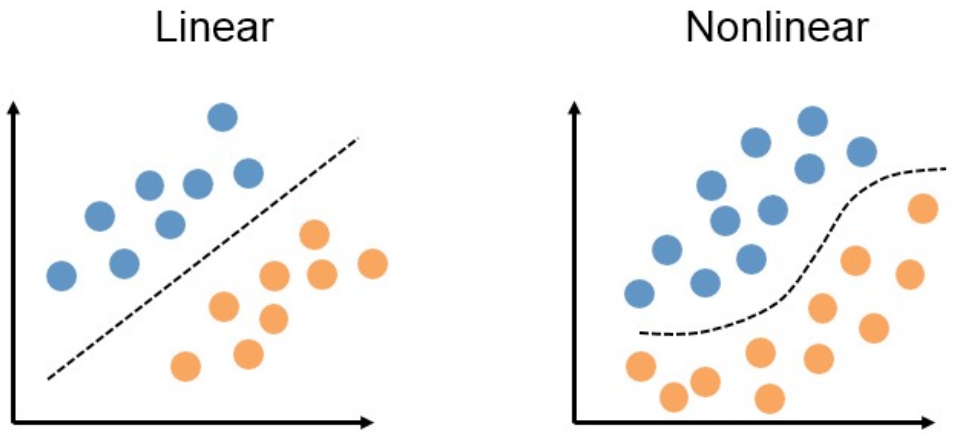
SVM finds data points closest together in high-dimensional space belonging to different categories (support vectors)

Data with >2 categories

Series of binary comparisons between groups



Linear vs. non-linear classifiers

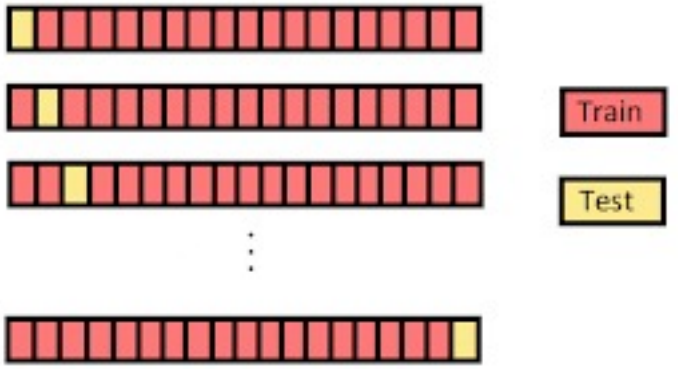


Unless there is a good a-priori reason to use a non-linear classifier, linear is usually best!

Practical use of linear classifiers: Cross-validation

- Tests the performance of a classifier
- Separate *training* and *testing* data sets

Leave-one-out cross-validation



Practical use of linear classifiers: Cross-validation

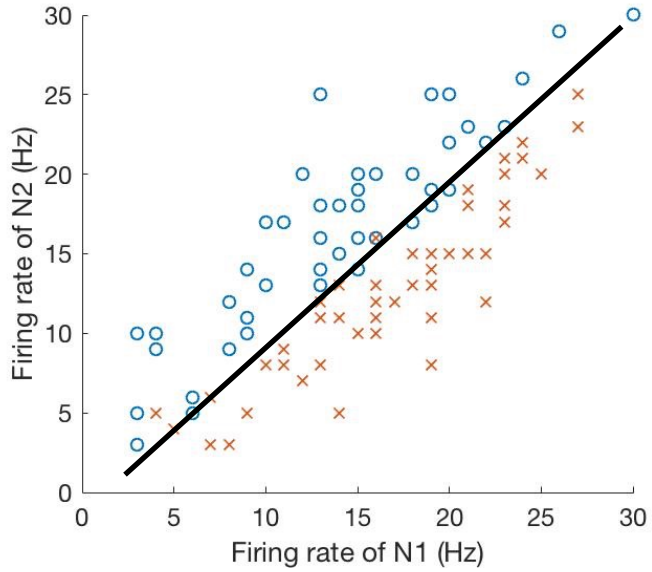
- Tests the performance of a classifier
- Separate *training* and *testing* data sets

K-fold cross-validation

Practical use of linear classifiers: Cross-validation

- Tests the performance of a classifier
- Separate *training* and *testing* data sets

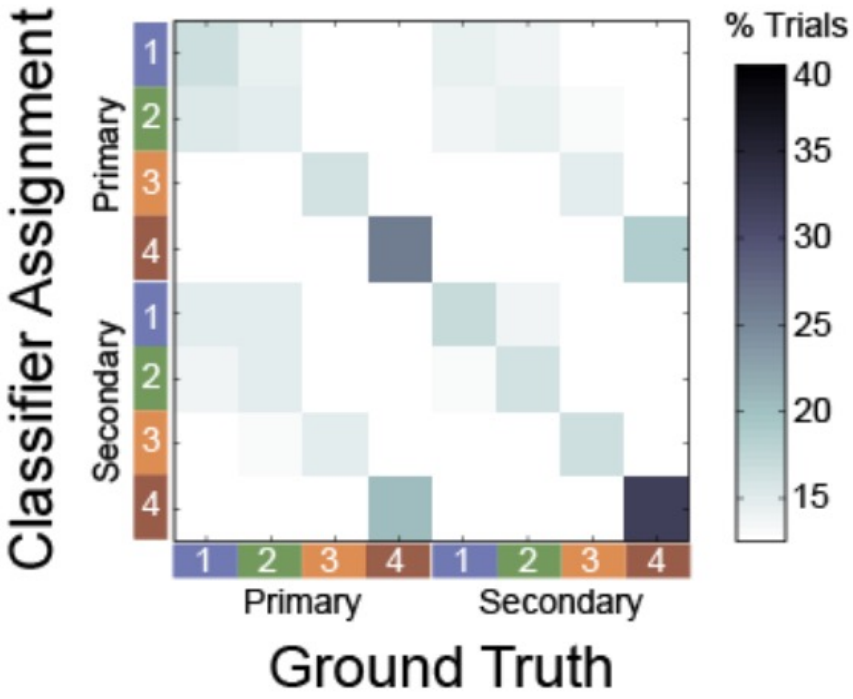
Outputs:



Each *test* data point gets
- a category assignment (e.g. red/blue)
- a measure of category fit (posterior probability)

Practical use of linear classifiers: Confusion matrix

- Displays the results of a classifier as a matrix of data points of each category, classified as each category



Accurate classifications on the diagonal
Note the off-diagonal misclassifications

8 categories -> chance = 12.5%*

*if there are ~equal number of observations per category
i.e. there are equal *prior* probabilities

Decoding continuous data

1. GLMs can be fit and used to predict new data

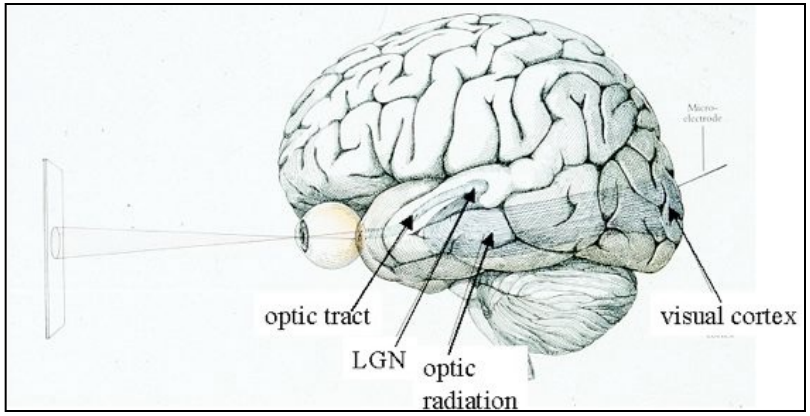
$$Y_i = \beta_0 + \beta_1 X_{1i} + \cdots + \beta_k X_{ki} + \varepsilon_i$$

- *regularization* can improve fits by down-weighting less predictive features (e.g. LASSO, ridge regression, elastic net)

2. Bayesian decoders can also be used

Limits of decoding approaches

Classifiers / decoders tell you *that* information is present not *why*



Classifiers / decoders are *supervised* -> only find information they are trained to find

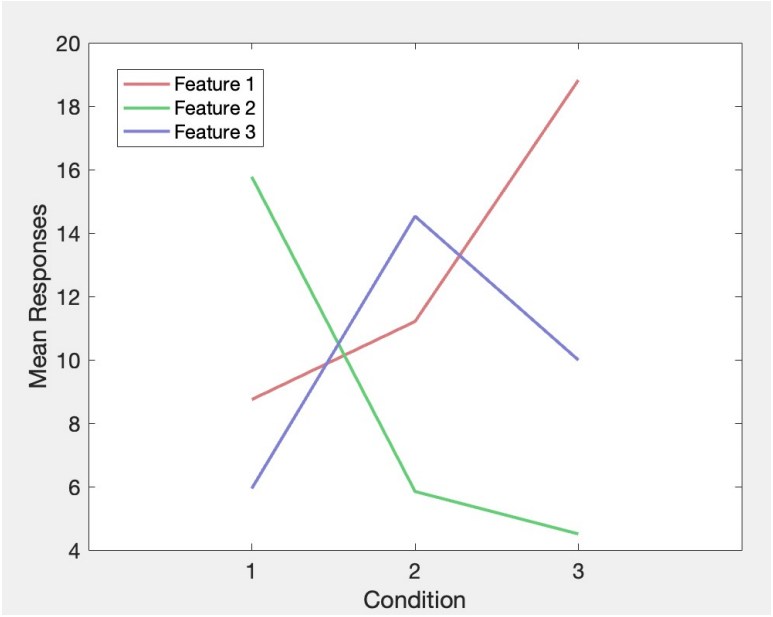
Demo

1. Load data:
   Neurons.txt = matrix of 900 samples (e.g. trials) x 3 features (e.g. neurons)
   Conds.txt = vector of labels (e.g. trial conditions) 900 x 1

Demo

1. Load data:
   Neurons.txt = matrix of 900 samples (e.g. trials) x 3 features (e.g. neurons)
   Conds.txt = vector of labels (e.g. trial conditions) 900 x 1

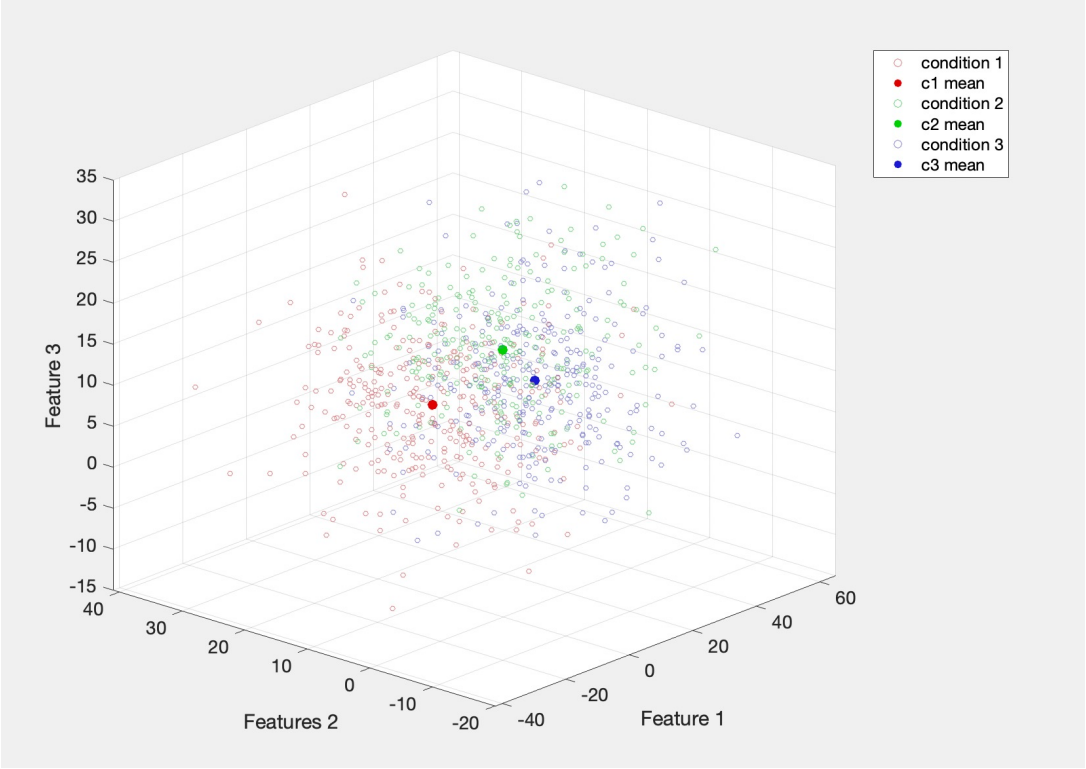
2. Plot the mean responses of each neuron to each condition



Demo

1. Load data:
   Neurons.txt = matrix of 900 samples (e.g. trials) x 3 features (e.g. neurons)
   Conds.txt = vector of labels (e.g. trial conditions) 900 x 1

2. Plot the mean responses of each neuron to each condition
3. Plot all responses in “feature space”



Demo

4. Use a classifier to decode conditions from the three features

    4a. Separate the data into training and testing sets

    (e.g. use 5-fold cross-validation: shuffle the trial order and separate into 5 folds while maintaining the trial indices so they are matched to their correct labels)

Demo

4. Use a classifier to decode conditions from the three features

    4a. Separate the data into training and testing sets

    (e.g. use 5-fold cross-validation: shuffle the trial order and separate into 5 folds while maintaining the trial indices so they are matched to their correct labels)

```
numtrials = length(conditions);
folds = 5;
trialorder = randperm(numtrials); %I'm randomizing trial order within folds -
%this is a good practice, in case there's anything that changes over
%consecutive trials, but it's not required

%this section finds the trial indices (inds) that will be in each fold of data
foldstart = 0;
inds = [];
for k=1:folds
    foldstart = foldstart+1;
    inds(:,k) = trialorder(foldstart:foldstart+numtrials/folds-1);
    foldstart = foldstart+numtrials/folds-1;
end
clear foldstart
```

Demo

4. Use a classifier to decode conditions from the three features

4a. Separate the data into training and testing sets
(e.g. use 5-fold cross-validation: shuffle the trial order and separate into 5 folds while maintaining the trial indices so they are matched to their correct labels)

4b. For each fold, train the classifier on all but one fold, and test on the held-out data

```
%Now for each fold, identify the training and test data, and run a
%classifier and find the predicted categories
PredClass = [];
ActualClass = [];
for k= 1:folds
    test = neurons(inds(:,k),:); %this is the test data
    testcond = conditions(inds(:,k)); %these are the condition identities of the test data
    train = neurons;
    train(inds(:,k),:) = []; %remove the test data from the training set
    traincond = conditions;
    traincond(inds(:,k)) = []; %also remove the test data conditions from the training conditions
    lda = fitcdiscr(train,traincond); %this trains an LDA classifier and saves it as a variable
    ldaClass = predict(lda,test); %this uses the trained classifier (lda) to predict new data
    PredClass = [PredClass;ldaClass]; %these are the predicted categories
    ActualClass = [ActualClass;testcond]; %these are the actual categories
end
```

Demo

4. Use a classifier to decode conditions from the three features

4a. Separate the data into training and testing sets
(e.g. use 5-fold cross-validation: shuffle the trial order and separate into 5 folds while maintaining the trial indices so they are matched to their correct labels)

4b. For each fold, train the classifier on all but one fold, and test on the held-out data

4c. Find the accuracy of the decoder
i.e. what proportion (or %) of the time does the predicted label = actual label?

```
>> Accur = length(find(PredClass==ActualClass))/length(ActualClass);
chance = 1/length(unique(conditions));
disp(['Overall Accuracy = ' num2str(Accur*100) '% Chance = ' num2str(chance*100) '%'])
Overall Accuracy = 63.4444% Chance = 33.3333%
```

Demo

4. Use a classifier to decode conditions from the three features

4a. Separate the data into training and testing sets
(e.g. use 5-fold cross-validation: shuffle the trial order and separate into 5 folds while maintaining the trial indices so they are matched to their correct labels)

4b. For each fold, train the classifier on all but one fold, and test on the held-out data

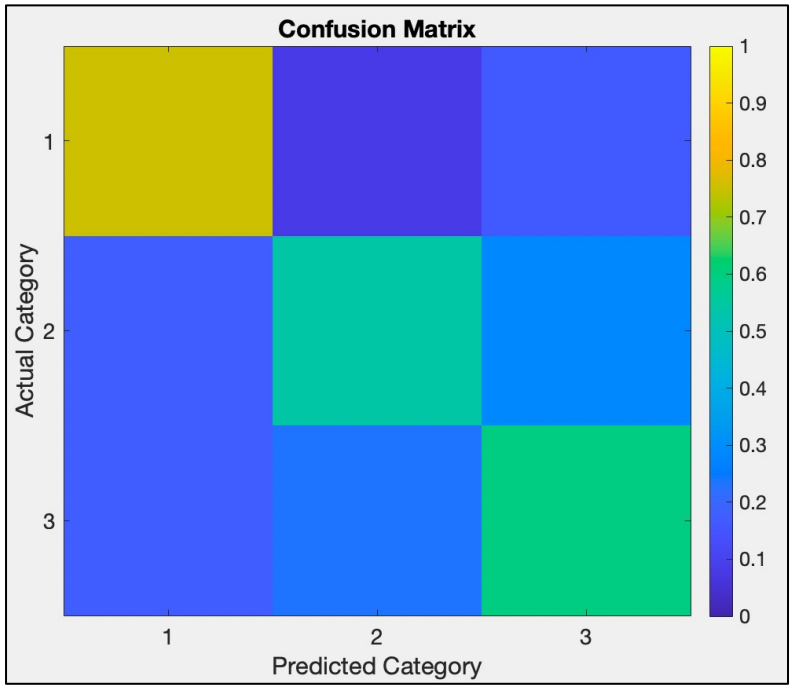
4c. Find the accuracy of the decoder
i.e. what proportion (or %) of the time does the predicted label = actual label?

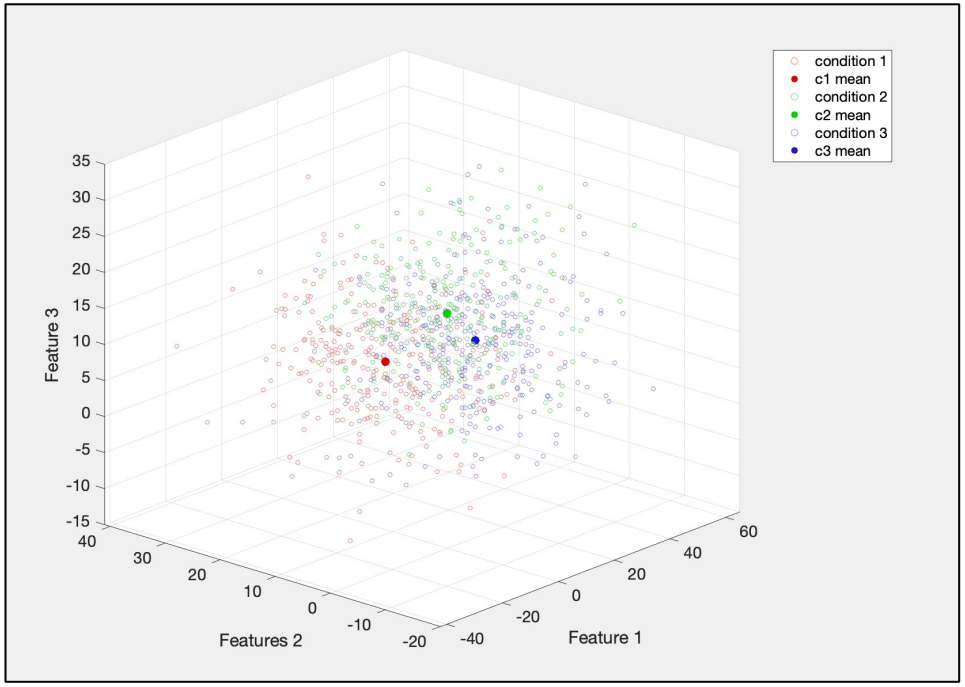
4d. Find the decoder accuracy within each category

```
>> CatAccur = [];
for k = 1:max(conditions)
    CatAccur(k) = length(find(PredClass==ActualClass & ActualClass==k))/length(find(ActualClass==k));
end
disp(['Category Accuracies = ' num2str(CatAccur.*100)])
Category Accuracies = 75.3125      53.9326      59.4249
```

Demo

5. Make a confusion matrix of the decoder's output





Homework #8

You have two different data sets: features1 and features2. Each includes 1200 observations (e.g. trials) of 10 different neural signals. Each observation belongs to a condition (1 to 4). The data can be loaded from the following files:

features1.txt
features2.txt
conditions.txt

1. First, plot the mean response to each condition (1 to 4) for each of the 10 neurons in features1 and features2. (You can overlay them in the same panel if you like). What is the main difference between these populations?

2. Using features1, use any linear classifier to decode condition. Test your classifier with 10-fold cross-validation.
   Report: (a) the overall accuracy (i.e. the overall proportion of observations that were correctly classified)
   (b) the accuracy for each condition (i.e. the proportion of observations within each condition that were correctly classified)

3. Run your code again (repeat the steps in 2) with features2 and report the same results