Lab Assignment 2: Machine Learning Models

Developer and coordinator: Samson Chota
Co-coordinator: Chris Janssen
Teachers: see course manual

Background and goals

Machine learning models are applicable to scenarios with large, complex datasets. As a case study, we look specifically at neural signals where datasets are complex and often large. Beyond Machine Learning techniques, neuroscientists have developed a multitude of analysis methods to disentangle the interesting from the uninteresting brain signals. To fully appreciate the power and (limits to) applicability of machine learning methods, in this exercise you will take a look at 3 commonly applied analysis methods:

- Part 1: A simple univariate analysis of the BOLD response that allows you to measure if one type of stimulus leads to stronger brain responses than another one.
- Part 2: A little more advanced analysis technique called support vector machine classification (SVM). This method allows you to utilize multivariate brain patterns to distinguish between the types of stimuli that were presented.
- Part 3: Representational similarity analysis (RSA) to compare brain patterns and test specific hypotheses.

This assignment contains some explanation of representational similarity analysis and some tips to help you implement one. You can choose to implement these algorithms in R or Python. Alternatively, you are also welcome to use MATLAB, which also has many relevant packages. Note that you will need to have access to the Statistics and Machine Learning toolbox if you want to use MATLAB.

At the end of this lab session, you can:

1. Process and analyze univariate brain activity patterns using statistics (covered in part 1).
2. Train and test support vector machines to perform multivariate pattern analysis (covered in part 2)
3. Perform Representational similarity analysis to compare multivariate patterns within and between data types. (covered in part 3)

Handing in Answers
The following text contains some explanation, some questions, and some instructions. Questions are clearly labelled and numbered: each group should submit a document answering these questions (with numbered answers). Some questions might require you to include some of the images you make.

Please hand in only a single file as PDF. On Blackboard you can again find a Word template that you can use. This file should contain:

- Your names and student numbers
- The answers to the blackboard questions (including Figures, when asked)

Time Management

- Lab session 1: Finish section 1 and 2
- Lab session 2: Finish section 3

Preparation: Download Data

Engineering: Download dataset
You will start with analyzing the simulated response amplitudes from a single participant. The data contains 100 fMRI recording sites (voxels) in human inferior temporal cortex in response to 88 images. This simulated data follows patterns of responses common in human inferior temporal cortex. This data is stored in the files 'NeuralResponses_S1' and 'NeuralResponses_S2'. For the first and last exercise we will use 'NeuralResponses_S1'. For the 2nd exercise we will use 'NeuralResponses_S2'.

The displayed images have been categorized and ordered by the type of object they contain. 'CategoryVectors' is a table containing binary classifications of the images into various categories. 'CategoryLabels' contains descriptive names for each category. For example, objects 1-44 are animate while 45-88 are inanimate, so element 1 of 'CategoryLabels' is "animate" and column 1 of 'CategoryVectors' has the value 1 (true) for objects 1-44 and value zero (false) for objects 45-88. Load both tables.

Download these files from Blackboard:

- NeuralResponses_S1
- NeuralResponses_S2
- CategoryVectors
- CategoryLabels

Section 1: Response Amplitudes

You will start by analyzing the strength with which different voxels respond to different types of images. The inferior temporal cortex (ITC) responds to all kinds of different objects. The different voxels that are contained in the file "NeuralResponses_S1" stem from different locations in the ITC and are processing different aspects of perceived objects. Some voxels will therefore respond stronger to certain images whereas others will respond weaker.

Engineering: averaging data per category

Load 'NeuralResponses_S1' in your software (R / Python / Matlab). Use the category variable (CategoryVectors + CategoryLabels) to separate neural data in response to images in which an animate objects and in which an inanimate object was shown. Now average the data over all voxels. This should leave you with 44 average voxel values for animate and 44 average voxel values for inanimate objects. **You are now able to answer Blackboard question 1A.**

Engineering: make your own t-test function

Implement code to conduct a **paired** t-test (i.e., write a function for this), to check if the average response amplitude is **different** between animate and inanimate images. To do this, you only need to turn the following formula into code:

$$t = \frac{m}{s/\sqrt{N}}$$

M and *s* are the mean and the standard deviation of the **difference between** our measured voxels' responses to animate and inanimate images. *N* is the number of observations per category. The resulting t value can be compared to the values in a t-table found here: (https://www.ttable.org/) and will tell us the corresponding p-value. For this the only thing left to do is to calculate the degrees of freedom (df = N-1) required for the lookup table. **You can now answer Blackboard question 1B.**

Engineering: analyzing individual voxels

Now, you will look at the responses of some individual voxels. Take the original data, however this time average over animate and inanimate images (respectively) instead of voxels. This should leave you with two arrays of size 100. Now subtract the average voxel responses of inanimate images from animate images (do this for each voxel separately). **You can now answer Blackboard question 1C and 1D**

Self-check (1):

Compare your averaged voxel responses to the individual voxel responses. How and why do some of the individual voxel responses differ from the effects that we see in the average?

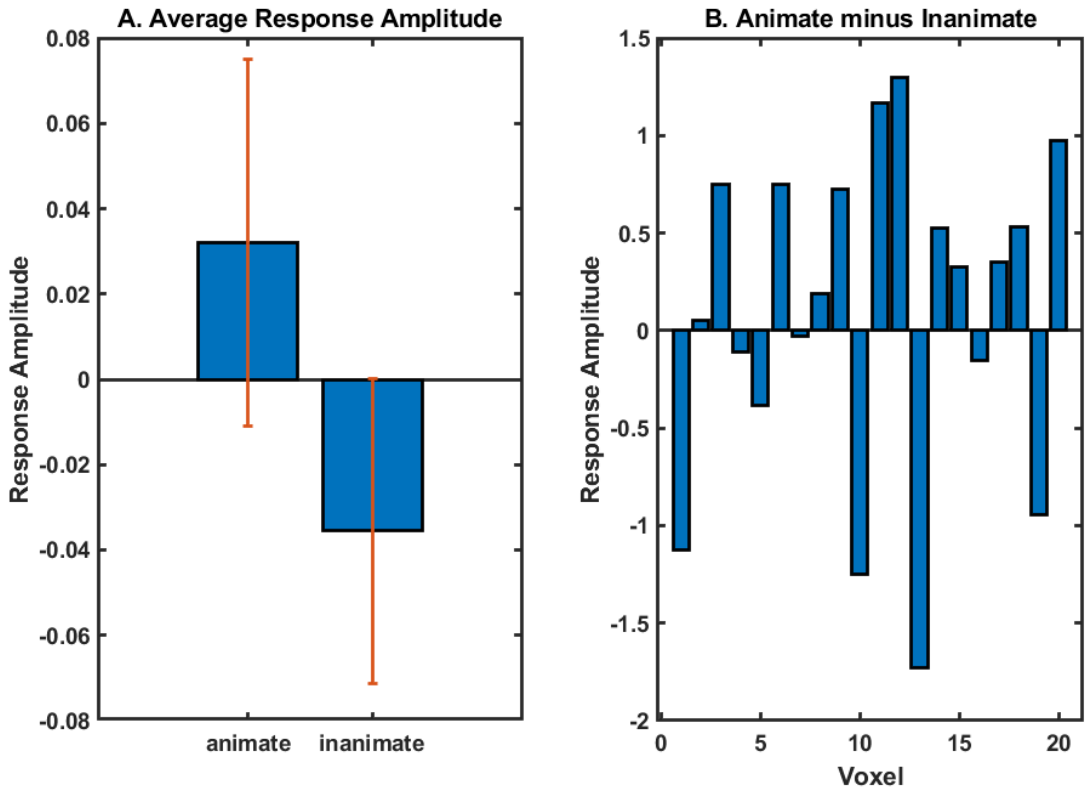

Figure 1 A and B

Blackboard question 1 (1 point total)

- **A.** Plot the average voxel response for animate and inanimate images using a bar plot. Add error bars to indicate the standard error of the mean (sem). Your plot should look similar in style to Figure 1A. Include your figure in your answer document.
- **B.** Report the t-value and p-value in your answer document of the difference between animate and inanimate results. In addition, answer if there is a significant difference between animate and inanimate condition.
- **C.** Use a bar plot to visualize the difference values of the first 20 voxels. Your plot should look similar in style to Figure 1 B. Include your visualization in the answer document.
- **D.** Why do some of the voxels show a positive and some a negative value? What does this tell us about the difference between conditions on the level of the group average versus the difference between conditions in single voxel responses? Explain your reasoning in 3-5 sentences.

Summary section 1

In this exercise you learned that

1. Quantifying the amplitude of neural responses allows you to test if certain parts of the brain respond stronger to certain stimuli than others.
2. You can statistically compare the responses from a specific group of images to another group using paired t-tests.
3. Responses of individual voxels can reveal patterns in the data that are not visible in the average.
4. More generally: when there is a clear analysis plan (often: informed by theory), conventional statistics can help you answer specific research questions. More advanced methods (such as machine learning) might not be needed.

Section 2: Support Vector machines

In the previous section you have looked at average response amplitudes of individual voxels. You might have noticed that the responses show quite some variation across voxels. You can make use of these variations in voxel responses by using a machine learning technique called Support Vector Machines (SVM). Support Vector Machines can be trained to distinguish between different classes of samples that contain several features (e.g., voxel responses). We call these samples multivariate (containing multiple variables).

An intuitive (but not entirely correct) way to understand how an SVM accomplishes this, is by finding the value for every voxel that best separates responses between one class and another. In reality this is a bit more complex: the SVM does this for all features (voxels) simultaneously with the help of a *hyperplane* (reflecting the relationship between all variables). In the figure below this is visualized for values from a single Voxel (figure 2A) but also for values from 2 or more voxels (figure 2B). The dotted line represents the estimated border of the SVM and is called the hyperplane. As you can see, the dimensionality of the hyperplane increases with the number of variables/dimensions. For a single variable, the hyperplane is a point. For two variables the hyperplane is a line.

Self-check (2): For three variables the hyperplane is a …?

After training, when the SVM makes predictions on new data (test data), values on one side of the hyperplane are classified as category A and values on the other side are classified as category B.

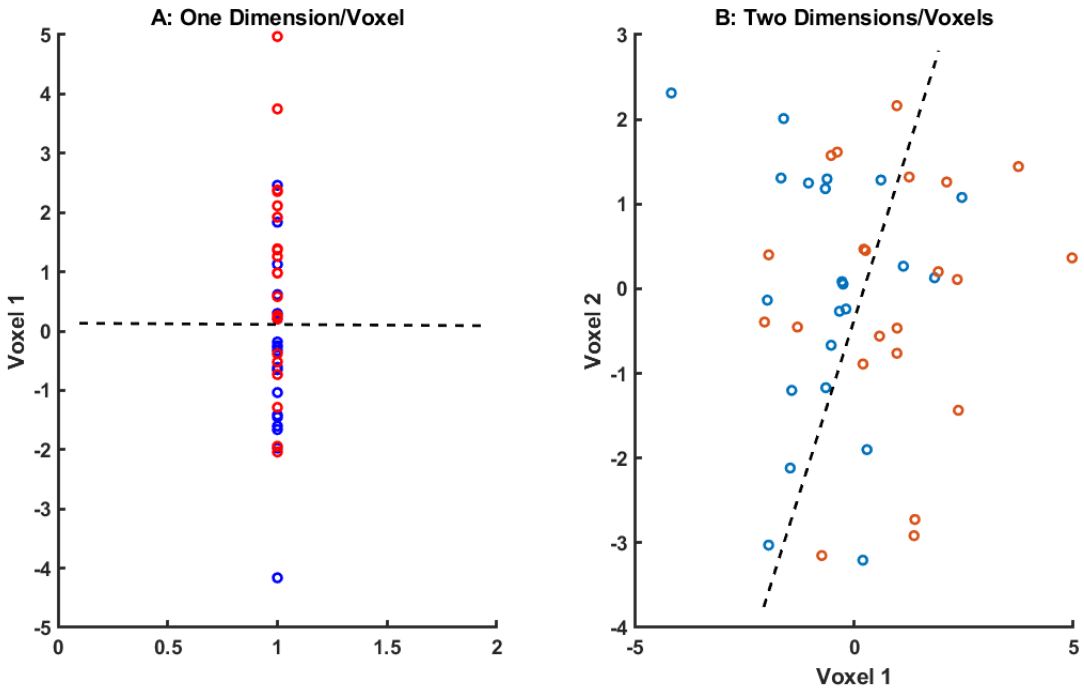

Figure 2: Support Vector Machines

Engineering: implementing an SVM

If you are using MATLAB you can refer to this page for help:
https://nl.mathworks.com/help/stats/fitcsvm.html
MATLAB SVM Code:

```
SVMModel = fitcsvm([training_set_A; training_set_B],[train_labels_A, train_labelsB]);
[predicted_label,score] = predict(SVMModel,[test_set_A; test_set_B]);
```

Note that you will need to have access to the Statistics and Machine Learning toolbox.

If you are using R you can refer to this page for help:
https://www.datacamp.com/tutorial/support-vector-machines-r
Install package e1071 (this contains the SVM function)
R SVM Code:

```
dat = data.frame([rbind(training_set_A,training_set_B), trainlabels = as.factor(training_labelsAB))
svmfit = svm(trainlabels ~ ., data = dat, kernel = "linear", cost = 10, scale = FALSE)
predicted = predict(svmfit, test)
```

If you are using Python you can refer to this page for help:
https://scikit-learn.org/stable/modules/svm.html
Install the sci-kit toolbox
Python SVM Code:

```
clf = svm.SVC()
clf.fit(training_dataAB, training_labels_AB)
prediction = clf.predict(test_dataAB)
```

This time you will load the data of another participant located in **NeuralResponses_S2**. Start by splitting the data into data that is in response to animate and to inanimate images. Your resulting matrices should have the following shapes (inanimate: 44 images by 100 voxels; animate: 44 images by 100 voxels). You will use these two types of images to train a linear support vector machine to differentiate between animate and inanimate voxel responses. Use the first 22 images of each condition as training set for your SVM. You will use the last 22 images from each condition as the test set. Dividing your data into training and test sets for machine learning applications is incredibly important as it helps to avoid overfitting. In a nutshell, a model overfits if it learns too many useless intricacies of the specific training set and is not able to apply the learned patterns to new data. If you are interested, you can find more on overfitting here: *https://elitedatascience.com/overfitting-in-machine-learning*

Your training and test sets should have the following shape:

Training set animate:	22 images * 100 voxels	label = 1
Training set inanimate:	22 images * 100 voxels	label = -1
Test set animate	22 images * 100 voxels	label = 1
Test set inanimate	22 images * 100 voxels	label = -1

Now train your support vector machine to differentiate between animate and inanimate images using only the training set. Make sure to provide your SVM with the correct array of labels (1 for animate and -1 for inanimate).

Now test your support vector machine on the test set. The SVM will provide a label (1 or -1) for every image in the test set (22 animate and 22 inanimate). Compare the predicted labels of your SVM with the actual labels of the images in the test set. **You can now answer Blackboard questions 2A and 2B.**

Now extract the weights that your support vector machine has learned for the first 20 voxels. You can extract these weights using this code in R:
weights = t(svmfit$SV) %*% svmfit$coefs
Plot them against the voxel responses from figure 1B using a scatter plot. Your graph should look like figure 3. **You can now answer Blackboard question 2C.**

Now calculate the Pearson correlation value r for these two variables. You can calculate Pearson's correlation coefficient for a single pair using the function:

- R:'cor'
- MATLAB: corrcoef
- Python: np.corrcoef

You can now answer Blackboard questions 2D and 2E.

Now you will perform the same analysis, but this time, you classify human and non-human images from the animate image set. Again, split both classes into training set and test set (first 10 images: training set, last 10 images: test set). There are 4 more human than non-human images in your dataset. We do not want to include these extra images because this can bias our SVM to predict a human image more often. Therefore you need to remove the last 4 datasets from the human dataset. **You can now answer Blackboard question 2F.**

Blackboard question 2 (1 point)

A. Calculate and report the accuracy of your SVM in %.
B. Explain: Why is the predictions not perfect? Provide your answer in at most 5 sentences (can be shorter!).
C. Include a picture of your scatter plot.
D. Report the pearson correlation coefficient that you calculated.
E. Take a look at this page: (https://libguides.library.kent.edu/spss/pearsoncorr) to find out if the correlation coefficient indicates a strong, moderate or weak correlation. Explain what the cause might be of this (strong, moderate or weak) effect in 3-5 sentences
F. Provide the decoding accuracy *and* explain whether and how this gives confidence / support that the SVM learned to differentiate between the two categories of trials. Provide your answer in 3-5 sentences.

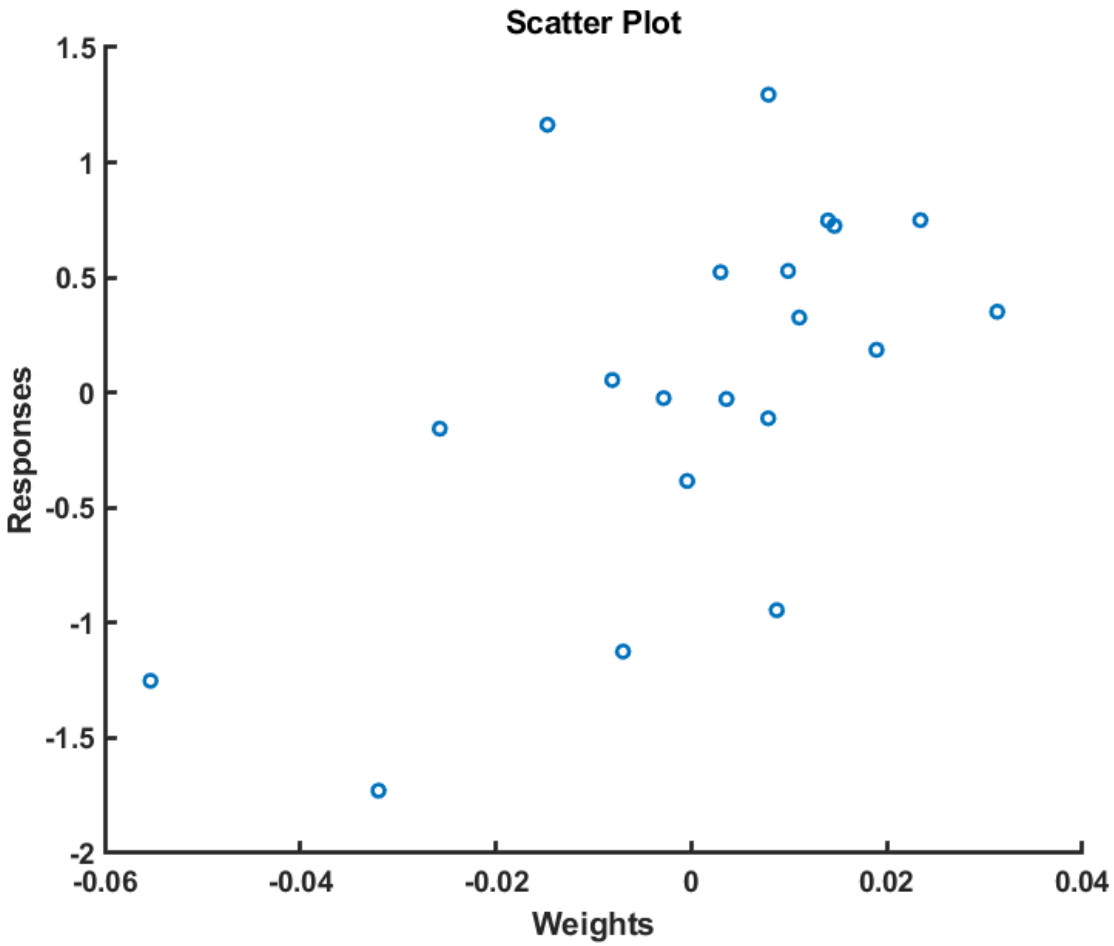

Figure 3

Summary Exercise 2

In this exercise you learned that

1. Support Vector machines can be used to classify multivariate patterns that are not revealed by univariate responses.
2. The support vector weights are used to differentiate between classes using a combination of all voxels.
3. The parameters of your training and test set are very important for efficient training.
4. You can use cross-validation to make better use of the dataset and statistically estimate the variance in our classification.

Section 3: The basic RSA (Representational Similarity Analysis)

Representational similarity analysis (RSA) is a data analysis method with broad applications to neuroscience. It is increasingly popular largely because it gives researchers the ability to compare patterns of responses across several types of data. For neuroscience specifically, this could be for example: fMRI recordings, recordings from large numbers of individual neurons, behavioral responses, and computational model unit responses.

The objective of this part of the assignment is to implement an RSA and compare patterns of representational similarity across different types of neuroscience data. RSA is a good machine learning technique here because it is widely applicable to many types of data yet only moderately complex.

There are various ways to quantify dissimilarity. In class we discussed using a Euclidian distance and 1-correlation, which both have advantages and disadvantages. Here, you should use 1-correlation, where 'correlation' is Pearson's correlation coefficient (confusingly called 'r').

The main difference between 1-correlation and Euclidian distance is that for Euclidian distances, responses to one object may be larger overall than responses to another object (i.e., their mean responses may differ). By comparison, the 1-correlation measure is not affected by these differences in mean response, while they can have a large effect on Euclidian distance.

Self-check (3): benefits of different analyses
What might be some reasons why each method may be more suitable for different data types or research questions? In other words, in which situations is it best to ignore differences in mean response, and when are they important to consider?

Load 'NeuralResponses_S1'. Now calculate the representational dissimilarity of each object pair, constructing a representational dissimilarity matrix (RDM). Take the responses of each object (across all recording sites) and calculate the dissimilarity from the responses to each other object. You will need to repeat this for every possible pair to produce an RDM. You know already how to calculate Pearson's correlation coefficient from exercise 2 (though you may be able to find a more efficient approach to compare all pairs of recordings). Remember that the RDM should contain measures of dissimilarity (i.e., 1-correlation) rather than similarity (i.e., correlation). Plot your RDM using an image plot similar to figure 4 A. Pay attention to the range of values in the color axis. **This is Blackboard question 3A.**

You can then use each column of CategoryVectors to construct RDMs representing specific hypotheses about the data. Each of these RDMs gives the pattern of dissimilarity predicted by a hypothesized response to the category described by the column. For example, the RDM from column 1 (animate/inanimate) predicts high

dissimilarity of responses when one object is animate and the other is inanimate, but low dissimilarity when both objects are animate or both are inanimate. You can also do this for the other columns (e.g., human/non-human).

Note here that in the example above of animate/inanimate, the value in this RDM reflects whether the pair have the same animacy state, NOT whether the pair are both animate: if both objects are inanimate, they share an animacy state and should have a similar representation. This is a very important principle of RSA: similarity reflects the **similarity** of the stimulus category, not the stimulus category itself.

With that knowledge, one can now ask whether objects with similar animacy category are represented more similarly (i.e., produce more similar response patterns) than objects with different animacy category.
For column 1 of CategoryVectors (i.e., labels for animate objects), compare each pair of values, filling the RDM with '0' where an object pair has the same state of animacy, and '1' where the two objects have a different state of animacy. This forms a "mask" that you can now use to select different groups of pairs.
Now, use a two-sided unpaired t-test to compare the dissimilarity of pairs with the same animacy against the dissimilarity of pairs with different animacy. Report your p-value in the answer file. Also create a bar plot showing the average correlation values between images from the same and images from different animacy category with error bars (sem). It should look like figure 4B. Remember to exclude pairs where an object is compared to itself. Also remember that each dissimilarity measure is included twice in the RDM but should only be included once in the t-test.
You can now answer Blackboard questions 3B and 3C.

Self-check (4):
Compare the resulting t-statistics and p-values to those for your original t-test of average response amplitudes for animate and inanimate images. Think about why they are different and what the two individual tests can tell us about the data in the two conditions. You don't need to include this in your answer file.

So far, you have used fMRI response amplitudes to construct and test RDMs. However, the same approach can also be used to compare different types of data. Here you will utilize this to test if behavioral similarity ratings are similar to neural similarity. The table 'BehaviourRDM' contains an RDM constructed from responses of human observers asked to arrange the objects by their similarity. Load this table and make an image of this RDM. You may see that some patterns are like those in the neural data, while some are not found in the neural data. This suggests that humans use other information, together with neural responses from inferior temporal cortex, when judging the similarity between objects.
Nevertheless, at least some of our behavior seems to be predicted by these neural responses. **You can now answer Blackboard question 3D and 3E.**

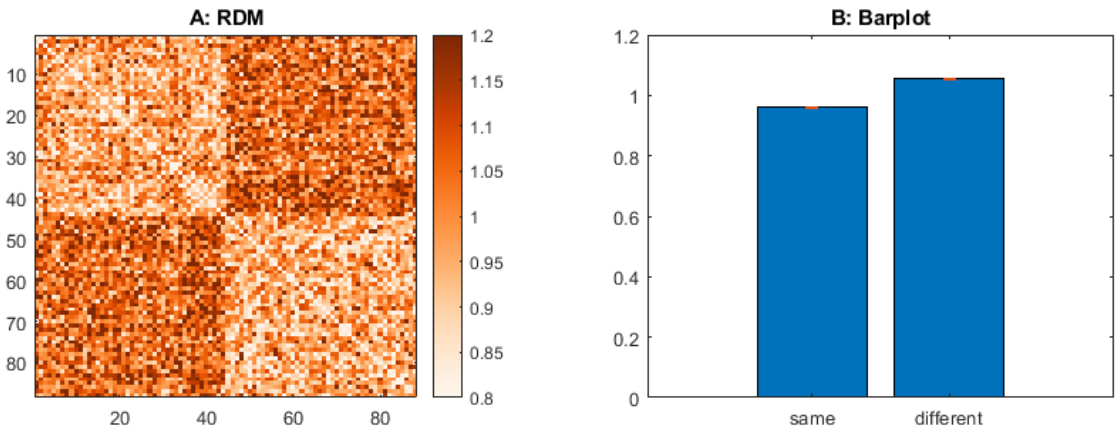

Figure 4 A and B

Blackboard question 3 (1 point)

A. Submit an image of your RDM. It should be similar in style to Figure 4A.
B. Submit the t-value and p-value of the two-sided unpaired t-test to compare the dissimilarity of pairs with the same animacy against the dissimilarity of pairs with different animacy.
C. Submit a picture of the associated barplot of this data. In style, it should be similar to Figure 4B (but values might be different).
D. How well does the *behavioral* RDM correlate to the fMRI RDM when using (1) all categories, (2) only animate, or (3) only inanimate categories? Report the correlation coefficient for all three cases.
E. What is an explanation for the results of the correlations in 3D?

When you are done with your questions, make a PDF file from your answer sheet and have 1 group member upload it on behalf of the entire group. If you are up for a challenge, consider completing the bonus questions.

Summary Section 3

In this section you have learned

1. When it is useful to utilize differences in response amplitudes in a dataset and when it is important to ignore them
2. Representational similarity analysis can be used to test how similar neural responses are between conditions/types of stimuli.

Bonus questions (at most 0.5 points)

If you want even more challenge, you can work on one of the following bonus options. Please note that these assignments will be marked strict, so merely trying will not give points. You can get at most 0.5 points for the entire bonus question section.

Bonus Question option A: LOOCV

When using an SVM to classify between human and nonhuman images we might have found a reasonable classification accuracy (or not). But how do we know if our SVM has truly *learned* a useful pattern? What if our classification accuracy was 51%? Would we be confident to say that we could differentiate between the two types of images based on our neural data? Moreover, we might have gotten unlucky (or lucky) with the noise in our data or the split between training and test set. A high decoding accuracy might incline us to say that our decoder performs well but we need to perform proper statistical tests to be certain. More precisely we need a way to estimate the variance in our dataset and the decoder estimates. You are already familiar with a technique that comes in handy here: leave-one-out cross-validation.

Perform leave-one-out cross-validation to classify between human and nonhuman images. Perform a t-test on the classification accuracies. In your report, you should provide:

- **The average decoding accuracy**
- **Your p-value in the answer document.**
- **An assessment whether this result provides confidence in the usefulness of the learned pattern**
- **A screenshot of the relevant part of the code**
- **Also mention that you completed Bonus assignment A**

Bonus Question option B: SVM Generalization

Everyone's brain is different and processes information differently. Some types of information are processed similarly and give rise to similar patterns of activity whereas other processing happens are very differently between individuals. As you have seen in exercise two, SVMs can learn to differentiate between such patterns of neural activity. This means they can also be used to test if two categories are similarly different in two separate datasets, for example in fMRI recordings of two separate participants. If an SVM that is trained on one dataset can perform well on another dataset it is said to generalize well. In this bonus exercise we want to compare the neural responses to animate and inanimate objects between participants.

1. **Test if the multivariate patterns (animate versus inanimate) in one subject (NeuralResponses_S1) are similar to another subject (NeuralResponses_S2) using SVM decoding. Report the decoding accuracy.**
2. **You might be surprised by the results of the previous analysis. Take a closer look at the individual predictions of your SVM. Try to swap your training and test dataset and check the predictions again. Try to find a**

way to increase the performance of your classification and report your decoding accuracy (Hint: Self Check (3) might help here).

When you complete this bonus assignment, hand in (in your answer sheet):

- **The decoding accuracy**
- **A description in words of how you solved the issue mentioned at (2) above**
- **The result of various tests that you did under (2)**
- **An explanation of why this changes things**
- **A screenshot of the relevant part of your code**
- **Also mention that you completed Bonus assignment B**

Bonus Question option C: RDM Generalization

Test if the neural signals in response to animate images are similar between subject 1 (NeuralResponses_S1) and subject 2 (NeuralResponses_S2) using Representational Dissimilarity Analysis. Make sure to use the correct statistical test and calculate the average dissimilarity value as well as the p and t value.

When you complete this bonus assignment, hand in (in your answer sheet):

- **Average dissimilarity values**
- **P -value**
- **t-value**
- **An assessment/judgement whether the neural signals are similar between subject 1 and subject 2.**
- **An explanation/motivation for that assessment/judgement**
- **A screenshot of the relevant part of your code**
- **Also mention that you completed Bonus assignment C**