# Week 5: An introduction to machine learning

## Commentary on this assignment

*This document is available for comments and questions. Everyone in the class can comment on it, point out errors, ask clarifying questions, and add helpful resources. I will try to update things, fix mistakes, and clarify throughout the week.*

The goal of this part of the assignment is to introduce you to some of the major ideas of machine learning and to give you a chance to explore machine learning with some data. This is not a machine learning course, so we won't dive deeply into the differences between machine learning algorithms. However, I think it is important for you (regardless of your career path) to develop a general understanding of what machine learning is, the typical analysis process, and ways to frame machine learning problems.

Machine learning is a topic where I expect that the class will have a really wide range of background experience. I anticipate that you're all going to come into and out of the next few weeks with different levels of understanding. ***My expectation is that you will commit focused time to build your own understanding, whatever your starting point. You'll be asked to reflect on your learning process at the end of this assignment.***

**If machine learning is new to you:** Welcome to your first taste of machine learning! This assignment will start with some introductory resources (though there are hundreds of alternatives available online if these aren't working for you). I suggest you start early, because some sleep cycles may be helpful in building your understanding.
If you don't get to the 3rd part of the assignment, that's okay. If you do get to this part, I suggest you focus on the train_frmi2d_vt and train_labels.

**If machine learning is pretty familiar to you:** Please feel free to skim or skip through some of this assignment (though it may be valuable to look at the conceptual framing provided in some of these resources. You can also challenge yourself by digging into different ways to select features in the fMRI data.

# The assignment (with embedded resources)

The assignment part is rather long now, since I added in some more detailed guidance about you might start to explore the classification tools. I added in some links here, and I suggest that you read the "What to submit" before jumping into the assignment.

## 1. A general overview of machine learning.

Read this general overview of machine learning. You can just focus on the introduction and supervised learning (though you're welcome to explore unsupervised learning section if you're interested):

- PDF version (available for comments/questions)
- Online version (has some videos embedded)
- Or these other videos may be a helpful starting point if you prefer: https://www.mathworks.com/videos/series/introduction-to-machine-learning.html?s_tid=srchtitle

## 2. An introduction to machine learning with data and tutorial

The steps below are intended to give you some general guidance and some ideas on how to get started. This assignment is meant to give you flexibility and space to explore these tools, so use this guidance to the extent that it is helpful for you.

## Some commentary on this part.

Please don't get hung up on the nuances of every type of classifier. I don't expect you to understand all the classifer types deeply, this is more about realizing that there are lots of options you can plug in here, but that the general problem framing is similar.

The tutorial here uses the Matlab Classification Learner App, which will hopefully help to wrap your head around the concepts without getting too bogged down in the code. If you prefer to engage with this using purely code, here are a few resources that might be helpful (there is not a code version of exactly what he is doing in the video):

- https://www.mathworks.com/products/demos/machine-learning/decision-surface.html
- https://www.mathworks.com/help/stats/classification.html?s_tid=CRUX_lftnav
- https://www.mathworks.com/help/stats/support-vector-machines-for-binary-classification.html
- Or, one of the most useful tools is to work through this with the Classification Learner App, then use the Generate Function button at the top and check out the code that is created.

If you are familiar with machine learning already, use this space to challenge yourself and do some deeper analysis of the data here.

### The data and accompanying functions are available here:

https://drive.google.com/open?id=1ZmT5Vh35hBEA9Vx-pQCbFivsjlu05-T_
Or here (original source, but you'll need to run the downloadSensorData script yourself):
https://www.mathworks.com/matlabcentral/fileexchange/50232-machine-learning-made-easy

The main file is Human_Activity_Learning.m (which has some code to generate the plots).

## Some suggested steps for exploration

1. Watch this video through the first example (up to 24 minutes):
   https://www.mathworks.com/videos/machine-learning-with-matlab-100694.html
   You may want to do this in conjunction with the next step.
2. **Explore the data with the 5 activity conditions.**
   a. Follow along with the video, going through each of the steps to load the data, plot the data, train a model, export a model, and apply it to your test data. The code in Human_Activity_Learning.m should give you all of the essential pieces. If you're struggling to follow, I did add a little extra commentary in SamsGuideToThisVideo.mlx.

    b. Once you can reproduce the analysis done in the video, it's time to extend this to explore some different models and parameters.
        i. Choose at least 5 different models to train on this data set. You can tweak parameters using the Advanced button on the top ribbon. Play around with these different models and see how they affect your training accuracy and your confusion matrix. This part is just about getting a sense of how these tools work, so don't worry too much about the specifics of the models right now.
        ii. Go to the confusion matrix for one of your models.
            1. Which two activities are most likely to be confused with one another?
            2. Which ones are very easy to distinguish? Does this make sense?

3. **Explore the data with only 2 activity conditions.** The original activity classification video looks at all 6 classes at once. However, it may be a bit easier to build an intuition about these things if we just focus on the case of 2 classes (which you determined above).
    a. Create a new table that only includes the data for these two classes. You can do this by going back to the Command Window in the regular matlab interface and using some variation of this code:
    ```
    twoClassHumanActivityData =
    humanActivityData(humanActivityData.activity=="Standing" |
    humanActivityData.activity=="Walking", :);
    ```
    This tells Matlab to create a new table based on the old table, but only take the rows in which the activity column is Standing or Walking.
    b. Go back to the Classification Learner and start a New Session from the workspace. Continue using 20% holdout. When you load the data, you should now only see two classes (activity categories).
    c. **Explore different features**
        i. Look at the scatter plots of the data. You can change the x and y axes that are plotted using the Predictors dropdown on the right.
            1. Flip through different predictor combinations and notice how some of them show the blue and red dots overlapping, while some do a better job separating the two categories.
        ii. Choose two Predictors (often called features in machine learning) to include in your model… we're going to start with a simple version.
            1. Start with two Predictors that do a poor job of separating the Standing and Walking category (this means that the dots are overlapping a lot in the plot).
            2. Click on the Feature Selection option in the top left. Unselect everything except for these two Predictors/Features.
            3. Choose a Linear Discriminant model and use the default settings. Hit the Train button to train the model.

4. Look at the Scatter Plot for the Model Predictions. Flip back and forth between this and the Scatter Plot for the Data (there is a toggle in the top left). Note what the data say the classes should be versus the model (the dots should be the correct values and the x's are the ones that model got wrong). Note, the model predictions are based on the 20% that were "held out" when we loaded the data.
   The Linear Discriminant model is basically just drawing a line in this 2-d space and saying everything on one side is Standing and everything on the other is Walking (you should be able to imagine where this line is based on the model predictions). This version of your model probably sucks (~50% accuracy and lots of x's on the plot).
5. Look at the confusion matrix to understand the type of errors your model is making with these features.
6. Now, go back to the Scatter plot for the Data. Choose two predictors that naturally separate the two activity classes. Make a note of the names of these two predictors. (If you're struggling to find two, check out the mean vs standard deviation for the acc). Choose these two features to be included in your model by using the Feature Selection tool (unclick the old features). Then repeat steps 2-5. Compare this to the model with the first pair of features.
7. Play around with these data by exploring different pairs of features or trying out some different models.

**d. Explore different model parameters**

i. For this next part, I strongly recommend using a new table that just includes data from the Walking and ClimbingStairs classes. These two classes are more difficult to tell apart, which will make the analysis more interesting. Create this new table and load this into the Classification Learner.

ii. Explore the Scatter Plots of the data, noticing that the differences between these two classes are not as visually obvious as Walking vs Standing. Look at the features that worked well in the Walking vs Standing… there is probably a lot more overlap now.

iii. Run the Linear Discriminant model again. The accuracy is probably notably lower now (you can experiment with a subset of the Predictors/Features if you want).

iv. Choose one of the model types and read a little about how they work. I think KNN and Linear SVM may be the easiest places to start.

v. Start with the default parameters for your model and train on the 2 classes.

   1. Look at the Scatter plot for the Model Predictions.
   2. Check out the Confusion Matrix

3. Check out the ROC curve. The ROC curve, provides another measure of how good a model is, called the AUC for area under the curve. This AUC measure is more informative but less intuitive than accuracy. If you have had some exposure to machine learning before, this might be a great time to learn more about ROC curves. If this is your first time, feel free to just stick with the accuracy and the confusion matrix.

vi. Train a few different versions of the same “type of model”, but using different parameters. You can tweak the parameters using the “Advanced” option in the top ribbon. You may want to revisit your reading about whatever classifier you chose to make sure you understand what this parameter is doing.

vii. Investigate how the accuracy changes as you alter a particular parameter. People sometimes make nice plots of how accuracy changes with a parameter such as the number of neighbors in KNN. This could be a nice thing to include in your summary.

e. Continue to explore the effects of different models and data to build some intuition about how this works in general.

4. After you have explored the training data and selected a model or two that you like, then save and apply the models to the test set (this is shown in the video and example code). Note: if you build your model on only 2 classes, then your classifier will be designed to only distinguish between two classes. In this case, you should trim down the test data to only have two classes (or you can build a model for 5 classes).

## 3. Classifying faces versus houses from fMRI data

It will probably be helpful to complete the Learn about fMRI assignment first.

Like the example where researchers asked a person to imagine playing tennis or not, we also expect to see different brain responses depending on what a person is looking at. In this data set, a person saw images of faces or houses at different times.

Here, the goal is to determine how well we can decode whether a person is looking at a house or a face, based on the fMRI data (values of the voxels).

### About the dataset (linked here)

This is the fMRI data from one person (subject 2 in this study: Haxby, J.V., Gobbini, M.I., Furey, M.L., Ishai, A., Schouten, J.L., Pietrini, P. (2001). Distributed and overlapping representations of faces and objects in ventral temporal cortex. Science, 293(5539):2425-30).

There are 162 examples in the training set (these correspond to times where a person saw a house or a face). The labels tell you whether they were looking at a face or a house.

Think of the voxel values as the potential features. In the train/test_fmri3d data, the voxels are in a 3 dimensional array (that is 40 x 64 x 64), plus a 4th dimension that is the 162 examples. However, for the classifier, you need to reshape the data to a 2 dimensional array. This reshaped data can be found in train/test_fmri2d_brain mask. These variables have the dimensions of examples x voxels. (If you do the math, the number of voxels here is too small… this is because I have removed all the voxels that aren't actually in the head, since they probably don't have any important information).

Even when only looking at the voxels in the head, we would still have 33913 voxels and 162 examples. This may lead to overfitting our classifier. If you're an advanced machine learning person, you might want to explore ways to reduce the number of features.

Based on what researchers know about the brain, it will be helpful to focus on the ventral temporal lobe (where we think lots of visual processing happens). I have "masked" voxels to only include the voxels in the ventral temporal cortex.  For this part of the analysis, the **data that I suggest that everyone starts with is the train_fmri2d_vt and test_fmri2d_vt with their train_labels and test_labels.** In order to use this with the Classification Learner tool, you'll need to put the data and labels together in a table. This code should help:

```
train_table2d = array2table(train_fmri2d_vt); %
train_table2d.labels = train_labels;
```

## Important tips (feel free to add some here or ask questions)

- If you are using the Classification Learner, only use a table that has a reduced number of features already, otherwise, it will take forever to load the data. (My computer took several minutes to load a table that was 162 x 39913).
  - Essentially, if you are using the Classification Learning, you probably only want to use the train_fmri2d_vt data (or another subset of the data).

## What to submit

- Please choose one of the things that you did (e.g. classify the activity data using an SVM) and write a short summary with at least 1 figure to help you reflect on your process. This should just be a regular pdf with text and image(s), not a full notebook. The purpose of this is to help you step back and think big picture about what you did.
- Please also share a more detailed reflection on your learning process (you will also do your self-reflection, but this offers more space to reflect on what you did, what you understand, and what you're confused about). This should be at least one real paragraph.
- You are also welcome to submit your code if you like to use Canvas as a repository for your work.