Introduction to Computational Neuroscience
Practice Session on Machine Learning

Aqeel Labash, Daniel Majoral, Raul Vicente

March 13, 2018

HINT: There is a file called `sample.m` in the code folder. It does not provide sample code, but it summarizes the steps to follow. It works for both exercises. Some helpful functions are in `code/helpers/` folder.

The main purpose of the fields of *neural encoding* and *neural decoding* is to learn the relation between a stimulus and the neural response elicited by this stimulus. When studying *encoding* we want to predict how the brain would respond given a certain stimulus - we estimate the probability $p(brainresponse|stimulus)$. In neural decoding we do the inverse - we read the brain activity using imaging techniques and ask the question ”which stimulus caused this activity?” - we look for $p(stimulus|brainresponse)$.

In a decoding paradigm we present the test subjects with different stimuli while at the same time recording responses on their brains. The task is to create a *model*, which takes any piece of the recorded brain data as an input and *predicts* which stimulus was responsible for producing this piece of data.

In the last practice session we kind of *manually* created one such model: by looking at the firings of 72 neurons we might pretty accurately guess the orientation of the bar on the screen. We would be able to predict which stimulus was shown, given the neural responses. Nevertheless, very often the data can be too massive or too complex to find such direct relation between activity and stimulus by simple observation. That’s where *machine learning* comes to our aid.

The main goal of machine learning can be summarized as providing an automatic way of finding the dependencies between a set of features (neural data) and the corresponding labels (the stimuli).

Before we continue let us go through vocabulary:

- **A dataset** is a structure which contains all the data we have.
- A dataset consist of **instances**, often also referred to as data points.
- In case of classification, instances consist of **features** and a **class label**.
- Features are parameters, which describe our data. For example the average spiking rate of a neuron might be a feature, the power of EEG in a certain frequency band might be a feature, etc. Each instance has its own values for each of the features.
- All feature values of an instance put together form a **feature vector**.
- Feature vector is a representation of the instance in the **feature space**.
- Each instance belongs to a certain **class** - the name/ID number of stimulus that caused the features.
- The goal of a machine learning algorithm is to create a **model**, which can guess the class (**classify**) of the instance given only its feature vector. A good model should be able to

do this also on previously unseen feature vectors (generalize).

- The model is created from examples. Those are instances for which the class is known. A set of such examples is called **training set**, because we train our model on it.
- **Test set** is another set of instances, for which we also know the true class, but we do not share this knowledge with the model. Instead we ask the model to guess the class of each instance.
- We can then see how many instances from the test set model has identified correctly and the rate

$$\frac{\text{Number of correctly classified instances}}{\text{Total number of instances}}$$

is called **accuracy**[1] and it is used to evaluate model's performance.

Exercise 1.1: Where is the rat? (2.5pt)

In this task we will try to predict in which part of space a rat is located based on its neural activity. When collecting the data some electrodes were inserted to the rat's hippocampus where neurons responsible for navigation are located. At the same time the location of the rat was tracked by a camera. I have preprocessed the data so that:

1. **features** are spike counts in different neurons during a 500ms interval
2. **class** corresponds to which of the 16 areas the rat was located during these 500ms
3. your goal is to create a model that predicts the area based on the neural activity.

In the *data* folder you find a file called *rat_data.mat*, which contains the features(spike counts from 71 neurons), the classes (which sector) and the coordinates (not used in this exercise). The same information is also available in plain text format in .txt files.

Matlab and especially Octave is not the perfect tool for machine learning, so if you prefer other languages to solve this exercise, please do so. I especially recommend scikit-learn package in Python. Nevertheless, as the default language of this class is Matlab, I will provide instructions and helpful code in Matlab and not other languages.

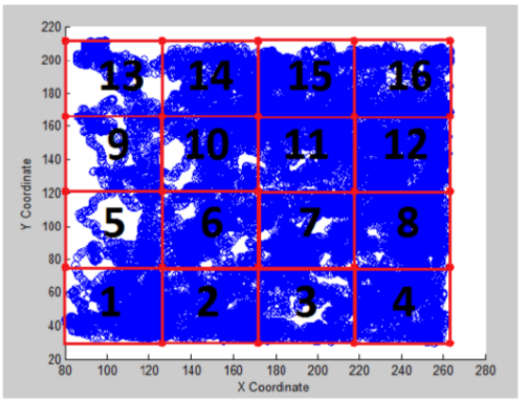

Figure 1: Find in which zone the rat is!

Before the task you might need to install some packages such as (nan, statistics, io ...). To do that in Octave command window, hopefully the following command works:

pkg install -auto -forge *PACKAGE_NAME*

Your task is to:

1. Read in the data and make sure you understand what is what
2. Randomly divide the instances into training and test sets, so that 80 percent of the data is in training set. **HINT:** there is a function in `code/helpers` folder that splits the data given features, labels and proportion

[1] Accuracy is a naive way to measure performance of the model. Read about *precision* and *recall* here `http://en.wikipedia.org/wiki/Precision_and_recall`

3. read how to use the `classify`[2] function, so you would place training and test features and class labels in correct positions. Use "LDA" (linear discriminant analysis) as the type of classifier.
   For people not using Matlab: whatever software you use, you need to perform Linear Discriminant Analysis.
4. Learn a model using the examples in the training set. Predict the locations for instances in test set and compare them with the true locations.
5. How much is the overall accuracy in the test set? How much is the precision for each class separately (return 16 values)?

$$precision\_of\_class\_i = \frac{points\ in\ zone\ i\ correctly\ classified\ as\ i}{total\ number\ of\ points\ classified\ as\ in\ zone\ i}$$

   HINT: class accuracies can be calculated by dividing the diagonal values of confusion matrix with the column sums
6. use the $plot\_confmatrix(groundtruth, predictions)$ from `helpers` folder to draw a **confusion matrix**. What additional information (beyond class accuracies) does this matrix provide?
7. **For people not using Matlab:** you need to produce a confusion matrix, but it does not need to be a nice colourful drawing. It can also be just a readable (!!) printout of the matrix.

Exercise 1.2: A few questions to keep in mind (1.5pt)
(answer in more than 1 phrase)
Q$_1$: Why do you need to separate training and test datasets, that is why can't you evaluate your model on the training set? What does it mean if you accuracies on training set and test set are very different?

Q$_2$: If our model would just predict class labels completely randomly, what would be the average prediction accuracy?

Q$_3$: If our model would always predict the class label that is the most common label in our dataset, what would the average accuracy be?

[2]http://se.mathworks.com/help/stats/classify.html

Exercise 2: Which picture was shown? (2pt)

In this exercise we will work with fMRI dataset[3] by Haxby et al.[4]. As you may recall, fMRI measures the blood oxygen level in the brain with high spatial precision. The data was recorded while test subject was presented with images from 9 categories: (1) house, (2) scrambled, (3) cat, (4) shoe, (5), bottle, (6) scissors, (7) chair, (8) face, (9) something else. You can see them (except for the "something else" category) on the Figure 2. The data we have is already preprocessed, so instead of ≈25000 voxels in the whole brain we only use 577 voxels from relevant brain areas. In machine learning terminology this means that each instance has 577 features and belongs to one of the 9 classes. You will find the feature data in `data/voxels.mat` and class information `data/labels.mat`. First one is 1492×577 matrix (1492 instances 577 features each) and the second one is a vector of length 1492 (each instance has a class).

The question we want to answer is: *Is it possible to decode from the fMRI signal the image the test subject was looking at?*. Your task will once again be to use Matlab `classify`[5] function to build a predictive model.

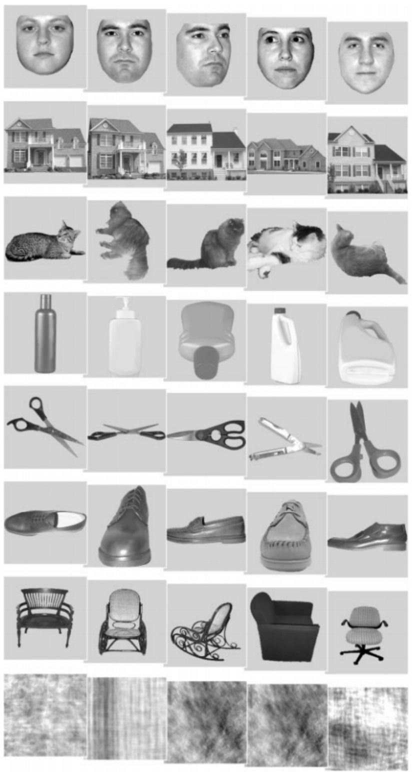
Figure 2: Examples of stimuli.

Perform the following steps and report the results:

1. Load the data. (Look at it. Always look at the data)
2. Split data into training and test set. We want to have enough instances of each class in the test set to calculate meaningful statistics, so this time take only 70 % instances for training set).
3. Classify using "LDA" as function type like before
4. Use training set to train a model. With the model predict the classes of the test set.
5. For each class calculate the precision on test set. Which class of images was the easiest to predict based on fMRI data? Which one was the hardest?
6. **For people not using Matlab:** Train a LDA (Linear Discriminant Analysis) model on this data, using 80/20 train/test split. Answer the questions.

———————————— End of obligatory exercises ————————————

Exercise 3*: Precision, Recall, F1-score (Bonus 1pt)
Sometimes accuracy can fail us if we are dealing with unbalanced datasets. If we classify images of cats and dogs and in out test set we have 90 % of cats, it is possible to achieve accuracy of 0.9 by simply always answering "cat". One possible countermeasure is to look not at the accuracy of the model, but at its *precision* and *recall*.

[3] `https://openfmri.org/dataset/ds000105`
[4] `http://www.cs.pomona.edu/~syang/arch/visual_decoding/haxby_2001pdf.pdf`
[5] http://se.mathworks.com/help/stats/classify.html

Imagine you have 2 classes: "cat" and "dog". Precision is calculated for each class separately and shows how many of the instances the model has identified as cats are really cats. For example if out of 10 instances classified as "cats" two turn out to be "dogs" we say that precision is 0.8.

Recall is also calculated separately for each class. It shows how many of all cats present in the test set your model correctly identified as such. For example if there were 100 cats and 100 dogs and our model correctly classified only 78 cats (other 22 it guessed as dogs) we say that its recall is 0.78.

F1 score is a convenient metric to write precision and recall as one number:

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Your task is to calculate precision, recall and f1-score for each of 9 classes.

Please submit a `pdf` report with answers to the questions and comments about your solutions. Include figures, explanations. Include the code within the pdf or as a separate file (zip them together). Upload it on the practice session page on the course website. Please mark how long it took to complete this set of exercises.