Assignment 4

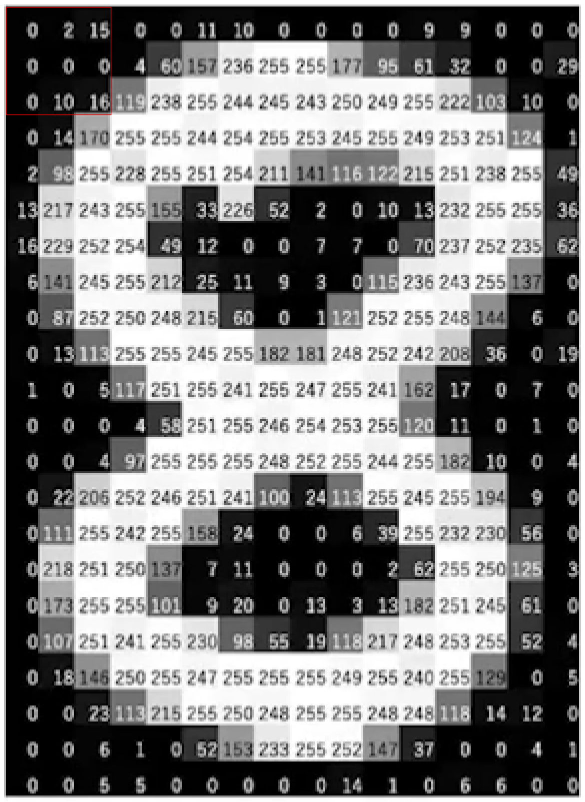
Part I is split into two parts as Part I.B is a short composition that must be submitted separately.

Part I.A (20 points each)

1. A child's game has a spinner with three outcomes: red, blue, and yellow. Suppose that we vary the percentage of the area under the blue spinner, ranging from 1/6 to 5/6, with the other colors sharing whatever probability is left over equally. Compute the entropy for each configuration and create a plot of entropy as a function of the probability that the outcome is blue. If you write a program to do this, show your source code (your program may not call library functions that compute entropy or information). If you do it by hand, show your work. Where is the entropy maximized and why?

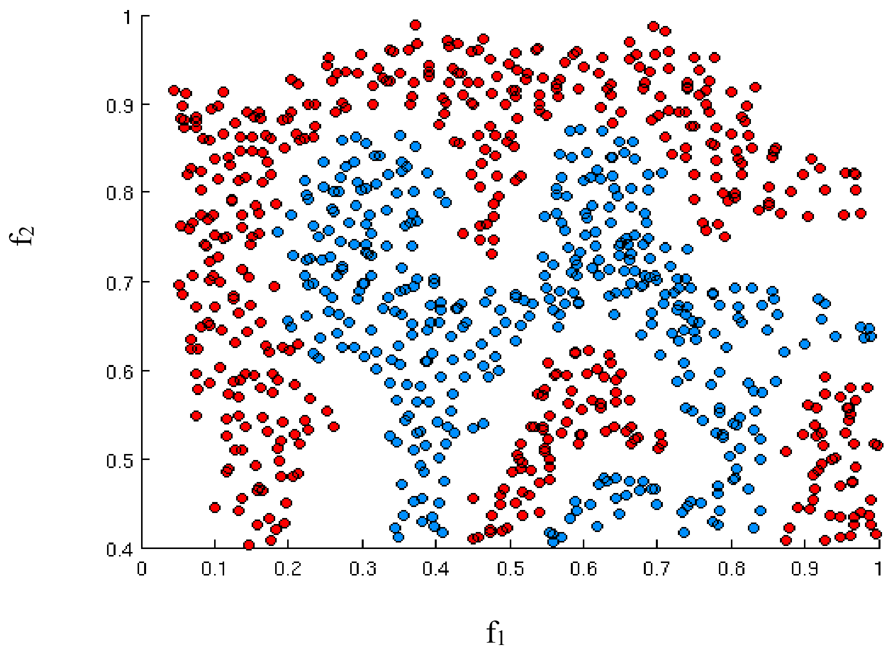
2. A toy machine learning algorithm has 10 examples (0, 1, 2, …, 9). Create a table of train and test splits for a 5-fold cross validation.

3. A 3x3 convolutional kernel has weights: $\begin{bmatrix} 1 & 2 & 2 \\ 0 & 0 & 2 \\ 0 & 0 & 1 \end{bmatrix}$. Assuming a stride of 2 in either direction, compute the top-left entry of the convolutional output and the subsequent convolution to the right and below. Assume the intensity image of the 8 shown here. The target area for the top-left convolution is shown in red.



0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

4. Given the two-feature data set below where two classes are distinguished by color,



would a single neuron neural net be more likely to have high bias or high variance? Justify your answer.

5. In the slides, we updated the weight for w_1 in a toy neural network (slides 15-19). Showing your work, use gradient descent to update the weight w_2 with a learning rate $\varepsilon=.01$ for the same network and show how the loss is reduced.

Part I.B

This short writing assignment (40 points) is to provide you with an opportunity to practice written communication for different audiences, something that people in industry and academia are frequently expected to do. In it, you will write three short single-spaced paragraphs. In each of them, you will explain a decision tree learner to a different audience. The paragraphs are limited to one-half page each (standard 8.5 x 11" page size and margins with 12 point type). Do not turn in more than one and a half pages, as no credit will be given if you exceed the allotted space. Part of good communication is learning to be concise.

1. The first paragraph is to be directed to a professional colleague who knows very little about computing in general and AI in particular. Describe to her what a decision tree learner is and what it is intended to do. Make sure to use the language she can understand or define the required terms for her.
2. The second paragraph is directed to a colleague who is a physician. He knows a bit more about computing than your first colleague. Explain to him what a

decision tree is and how it can be used in the context of medicine. Go into as much detail as you can in the allotted space and use relevant terms from science and computer science.

3. The third paragraph is directed to a colleague who is a computer scientist and has a Ph.D. in AI. Explain to her the issues that you're having implementing your decision tree system. Get into the technical details of a made-up scenario about your system.

Part II – Programming

1. (20 points) We indicated in class that modern neural network libraries build computational graphs and perform derivatives automatically. To provide you with greater intuition as to how automated differentiation works, you will write two functions related to a simple type of function, the family of polynomials.

   Remember that polynomial functions are of the form:
   $$f(x) = k_i x^i + k_{i-1} x^{i-1} + \cdots + k_1 x^1 + k_0 x^0$$
   A common way to represent the coefficients of a polynomial in a computer is as a list where the last coefficient is the coefficient of x^0 and each preceding one is the coefficient of the next power:
   $$[k_i, k_{i-1}, \dots, k_1, k_0]$$

   Complete the code skeletons in module polynomial.py (polyval and derivative) in the provided code package.

2. Dolphin identification (140 points)

The code package contains support functions for developing a neural network using Tensorflow. Instructions for setting up Tensorflow are included in the code package. Data (29 MB) are provided to distinguish acoustically between two species of dolphins (Figure 1): Pacific white-sided dolphins and Risso's dolphins.

Figure 1 – Pacific white-sided dolphin (Lagenorhynchus obliquidens, left) and Risso's dolphin (Grampus griseus, right). Photos: Marine Biaoacoustics Research Collaborative

Dolphins produce echolocation clicks for several purposes including navigation and finding food. The clicks are highly influenced by the animals' biology; the melon (forehead) acts as an acoustic lens to focus sound energy. We showed that these two species could be distinguished by analyzing their echolocation clicks (Soldevilla *et al.*, 2008). The data provided to you contain representations of echolocation clicks recorded at seven locations off the coast of Southern California. The data in this assignment are from a study that showed how instrumentation and recording location could influence performance (Roch *et al.*, 2015). You will be replicating a small subset of the experiments from this study where the features have already been extracted from the raw acoustic data. These features are called cepstral features, if you are curious about them, they are explained in Roch *et al.* (2011).

The focusing mechanism of the melon makes recordings of echolocation clicks highly sensitive to the angle between the animal's longitudinal axis and the location of the recording device (Figure 2). As a consequence, while we train models to predict species identity from training data that consists of individual clicks, during classification we usually need to make decisions based on a group of clicks in order to have accurate classification.

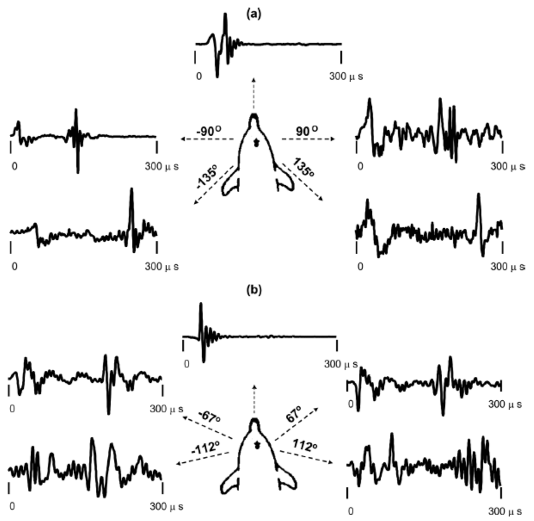


Figure 2 – Echolocation clicks measured at different angles (Fig 2, Au et al., 2012)

In this assignment, you will create a neural network to classify 100 clicks at a time to species. The network will predict the probability of each dolphin species one click at a time, and you must fuse the predictions together to make classification decisions based on a statistic of the predictions as described on the next page.

Echolocation clicks have been extracted and processed with a noise compensation routine (cepstral means subtraction) as described in Roch *et al.* (2015). Files are organized into a directory containing the Risso's dolphins clicks (Gg, the first letter of each word in the Latin species name *Grampus griseus*), and another one for the Pacific white-sided dolphin clicks (Lo, *Lagenorhynchus obliquidens*). The Pacific white-sided dolphin directory is further subdivided, but that is not relevant to this assignment.

There are a number of functions that are provided to you that will be helpful in this effort. Module lib.file_utilities contains two functions that you will need to call:

- Given a directory, get_files will recursively explore the directory for files with a specific extension which can be overridden. It also supports a stop_after keyword which lets you specify a maximum number of files to return. During the development cycle of your software, this feature is worth using as it will enable you to load small amounts of data quickly. You are unlikely to produce a good model when processing a small number of files, but it is nearly always a good idea to get everything working first with a small data set before training on something that will take a little while to complete.
- Given a list of filenames, parse_files returns a list of RecordingInfo tuples. These are named tuples that can be accessed with an attribute name instead of an index. This function extracts information about when and where the data were recorded, the start time of the data, the species that were recorded, and a numpy 2D tensor (matrix) of features where each row represents an echolocation click. See function documentation for further details on attribute names and interpretation.

Data should be split into training and test data separately for each species. The criterion for splitting is that all data from a given day must be entirely in the training data or entirely in the test data. Module lib.partition contains function split_by_day that takes a list of RecordingInfo tuples. It returns a dictionary keyed by the day of recording. All values associated with a key started recording on the same day. We do this to make splitting or training and test data more appropriate. If we randomly assigned echolocation clicks to training or test, we might have echolocation clicks from the same animal at roughly the same time in both our training and test data. This would make for a better performing system, but would not be representative of how we would expect a system to perform in the field. Consequently, when we select data for training and testing, we split the keys of the dictionary into training and testing rather than the features themselves. All of the data associated with each key is thus placed entirely in the training data or the test data.

These keys for these dictionaries should be used to split the data into training days and test days. Scikit learn has a function train_test_split that can be used to split the keys into training keys and test keys. Note that the dictionary keys must be cast to a list before they can be used by train_test_split. By default, this uses a 70% training, 30% test split. Gather all of the list elements associated with the date keys for each species and split them into training and testing lists. Note that as the number of echolocation clicks per day is not controlled, you are unlikely to have a balanced number of training examples for each category. You have three choices:

1. For the purpose of this assignment, let it remain unbalanced. This is not a great choice, but it should be okay. By splitting the training and testing sets for each species, you at least ensured that there were a significant number of clicks for each species even if they were not properly balanced.
2. Randomly select echolocation clicks to drop from the larger set such that the number of clicks are roughly equal.
3. Use a weighted loss function to adjust the loss penalty based on the number of examples. More detail on this may be found in the description of what you must do to have a score of Good at the end of the assignment.

The next step is to prepare data to train your model. You will need to construct example and label tensors for all the days in the training data. The list items you previously constructed will have attributes for features and label that contain the echolocation click features and the Gg/Lo label code. Numpy's array manipulation routines are excellent for this, and you might want to consider looking at concatenate, hstack, or vstack for this purpose.

Train a feed-forward model with these data. You will need to specify the neural network model and then call fit to train it. The model should have an output layer that uses a softmax activation function, producing one output for Pacific white-sided dolphins and a second for Risso's dolphins. With the splitting criterion of any given day not being split across the train/test boundary, this task should be relatively easy for a neural network and your choice of network is not critical to having good performance. You must use some form of regularization in your network, such as an L2 regularizer. As these data are relatively easy to classify, and the data set is relatively large, you should not need more than a few epochs of training. On an Intel I7-11700K without using GPU acceleration and the suggested architecture, training 3 epochs of the full data set takes about 20 minutes with the full training set.

A good starting point might be a four-layer feed forward network with 100 nodes each and an L2 regularizer with a value of 0.01 (you can probably improve this by increasing the network capacity). During training, the classification results on the training data will be shown on individual clicks. While you should expect to see the accuracy increase as training continues, remember that clicks are highly variable and you will not achieve stellar results on individual echolocation clicks.

Predicting species from more than one echolocation clicks yields much better results. During test, you will process each feature set that is in the test data. After you have your per click predictions, group the probabilities into groups of 100 predications and compute their joint likelihood (under the assumption that they are independent from one another). In theory, this is accomplished by multiplication, but with likelihoods this can lead to numeric underflow. Take the log of the probabilities and sum them instead. The maximum log likelihood indicates the class. If the last group in a file has less than 100 clicks, discard these predictions.

As you predict, keep track of correct and incorrect predictions. Create a confusion matrix, a matrix where the rows represent the actual classes (Gg, Lo) and the columns represent what these were actually classified as. If Gg is encoded as 0 and Lo as 1, misclassifying a group of 100 Gg clicks as Lo would add one to confusion[0][1]. If they were classified correctly, we would add one to confusion[0][0]. While computing a confusion matrix is relatively straightforward, you are welcome to use the implementation in scikit learn, which also has a library function for graphical display. Output your confusion matrix either graphically (submit the file) or to the console. Be sure that your rows and columns are labeled. If your system is performing well, most of the counts should be on the diagonals. In addition, you must compute and print your overall error rate (1 – accuracy).

Your error rate will depend significantly on the random split of training and test data, but on average it should be achievable to have an error rate under 3%. Neural networks are non-deterministic due to their random starting points. Small differences in data handling and shuffling of training data also contribute to making exactly reproducible results difficult.

As a consequence, there is no automated grader for functionality in this assignment. Points for functionality will be assigned as follows:

- Valiant Effort/Right Track – You made a good faith effort but were unable to produce a network that classified appropriately and did not produce the appropriate metrics. If you were a good portion of the way to having a successful program, right track will be awarded.
- Mostly Right – You must produce a network that properly partitions and uses the data and produces an error rate under 25% and produce a confusion a matrix.
- Good – The above must hold, your error rate must be under 10%, and your confusion matrix must be a plot that is well labeled with rows and column functionality clearly indicated, and use a weighted loss function. You can compensate to some extent for class imbalance by providing a dictionary whose keys are class numbers 0:N-1 and whose values are a weight which will scale the loss present for each class. The dictionary is passed to the fit function with the class_weight named parameter. If our training data had a 10 examples of class 0, 5 of class 1, and 5 of class 2, we would use weights={0:1, 1:2, 2:2} which would double the loss of the classes with fewer examples.
- Excellent – All of the above and write a new function that partitions data by site (location) instead of day. Show the error rates for the original experiment plus the new one.

What to submit:

- Part I
  - Submit your individual work for I.A and I.B to Canvas. Note that I.B must be submitted **separately**.
- Part II
  - Submit module derivative.py.
  - Submit driver.py. This must be the entry point to your neural network program. Be sure to use appropriate abstraction, do not put your entire program in one function.

    You may either leave all of the code that you write in driver.py or use modules. If you introduce other modules, leave them in the main directory and be sure to submit them.
  - Submit the output of your program in file output.txt. Upload the file, not a screenshot.
  - Submit the confusion matrix image if you created a graphical confusion matrix. Use filename confusion.png (or other appropriate file extension such as .jpg).

References

Au, W. W. L., Branstetter, B., Moore, P. W., and Finneran, J. J. (2012). "Dolphin biosonar signals measured at extreme off-axis angles: Insights to sound propagation in the head," J Acoust Soc Am 132(2). 1199-1206.

Roch, M. A., Klinck, H., Baumann-Pickering, S., Mellinger, D. K., Qui, S., Soldevilla, M. S., and Hildebrand, J. A. (2011). "Classification of echolocation clicks from odontocetes in the Southern California Bight," J Acoust Soc Am 129(1). 467-475.

Roch, M. A., Stinner-Sloan, J., Baumann-Pickering, S., and Wiggins, S. M. (2015). "Compensating for the effects of site and equipment variation on delphinid species identification from their echolocation clicks," J Acoust Soc Am 137(1). 22-29.

Soldevilla, M. S., Henderson, E. E., Campbell, G. S., Wiggins, S. M., Hildebrand, J. A., and Roch, M. A. (2008). "Classification of Risso's and Pacific white-sided dolphins using spectral properties of echolocation clicks," J Acoust Soc Am 124(1). 609-624.