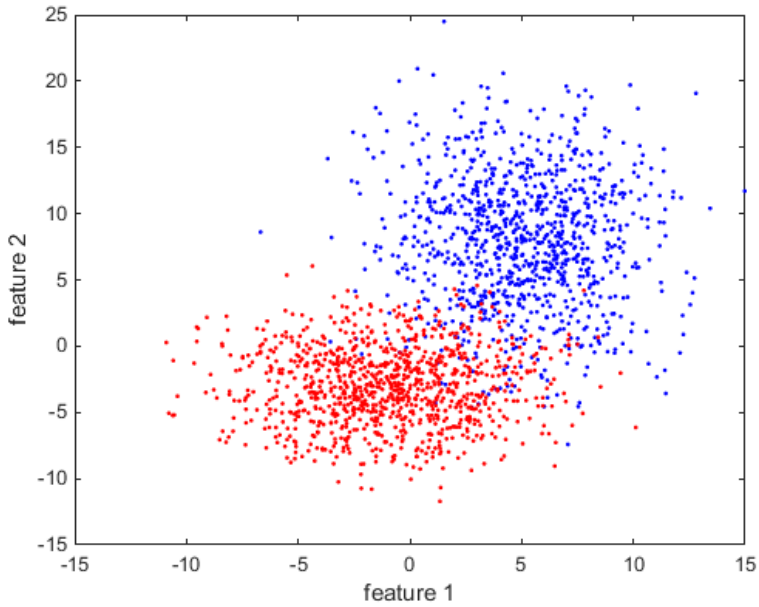
A2

Numerical problems must show work to receive credit. Programs must be well structured and commented.

Part I (20 points each)

1. Show that a classifier that partitions a feature space cannot shatter 4 points in a two-dimensional feature space.
2. A linear regression classifier is used to separate the data below. Would you expect the bias or the variance of the classifier to be higher? Justify your answer.



3. Paolo is a create of habit. He goes to his favorite restaurant every Tuesday for tacos. The restaurant is randomly closed two days per year (not necessarily on a Tuesday). Assuming 365 days per year and a one in seven chance of the closure being on a Tuesday, what is the information (in bits) for the distribution of openings and closings for Paolo's taco Tuesday?

For the next two problems, use the following class to model statistical distributions with scipy:

```
import numpy as np
from scipy.stats import norm, uniform

class Distribution:
    """
    Class for generating samples and estimating their likelihood
```

```
Example:
U = Distribution(scipy.stats.chi2(3))
# Chi-squared 3 degrees of freedom
u = U.draw((2,5))  Draw 10 samples as a 2x5 matrix
p_u = U.likelihood(u)  Estimate the likelihood of each sample
"""

def __init__(self, density):
    """
    :param density:  A scipy.stats density
    Example:  scipy.stats.uniform specifies a uniform density
    over the interval [a, b] as loc=a, scale=b-a
    """
    self.density = density

def draw(self, shape):
    """
    Draw a set of samples specified by tuple shape
    :param shape: sample tensor size
    :return: samples
    """
    values = self.density.rvs(size=shape)
    return values

def likelihood(self, samples):
    """
    Return the likelihood of samples assuming they are drawn
    from this distribution
    :param samples:
    :return: P(samples)
    """
    p = self.density.pdf(samples)
    return p
```

4. Consider a pair of normal distributions, $n(\mu, \sigma^2)$: $N \sim n(0.1, .5)$ and $W \sim n(0,1)$.
    a. Do you expect KL(W || N) or KL(N || W) to be higher? You will not be scored on your intuition. Instead, write a small Python program to sample from these distributions and estimate the KL divergence both ways. Use a minimum of 250,000 samples. You can use the scipy.stats.norm distribution with the Distribution class above to draw (generate) samples and determine their likelihood. Do your results agree with your intuition? Now that you know the result, why or why not? Show the program listing and its output.
    b. Suppose that Y had a uniform distribution, $Y \sim U(-.25, .25)$; explain what you would expect the KL(W || Y) to be?

5. For the same distributions above as in 4.a,
    a. estimate the entropies of N and W.
    b. Compute their cross entropies and show that they are not symmetric.

Part II (100 points)

Before starting this project, you should install Python 3 and tensorflow, I recommend using Anaconda (see instructions if needed) as it will install most of the libraries that you need with the exception of librosa which can be installed with conda install from channel condaforge.

In this assignment, you will be working with the King Speaker Verification corpus (Higgins and Vermilyea, 1995). This database is licensed to SDSU by the Linguistic Data Consortium and may not be redistributed or used outside the context of this class. The King corpus consists of 51 male speakers recorded by The International Telephone and Telegraph Corporation in San Diego, CA and Nutley, NJ. We will be using this corpus for a speaker identification task. Most of the speakers have 10 recording sessions, and the provided King class will find all of the audio files for you, determine the speaker number, and provide methods to access the files by speaker as well as convert speaker identifiers to class numbers. The King class discards the few speakers who do not have enough examples. There are two versions of this corpus, a high fidelity one sampled at 16 kHz and referred to as the wide-band corpus, and the speech that has been transmitted through a public telephone network and sampled at 8 kHz; the narrow-band corpus. For the purposes of this experiment, you will only be working with the 25 Nutley, NJ speakers on the wide-band corpus, there is an option during object construction to restrict the speakers to the Nutley speakers. Pay attention to how this class functions, you may be expected to write corpus manipulation classes for future assignments.

During training, you will use spectra (frequency energy information) derived from frames of speech (10 ms advance, 20 ms length) from the first 5 recordings. The provided function get_features will create a linear spectrogram. (Setting the feature argument to "Mel" will create a perceptually weighted spectrogram, we will discuss that later in the semester and you do not need to worry about that now.) Function get_features will identify areas of speech activity and noise by using a Gaussian mixture model speech activity detector. The mean noise of each frequency of the spectrogram is computed from the noise frames and subtracted from all frames. Noise frames are then discarded and we return the speech-only spectrogram as a feature matrix along with a label vector. Each entry of the label vector will contain the speaker class number (as the same speaker produced all of the spectral frames from any given file). It is recommended to look at some of the spectrograms and/or listen to speech. Spectrograms may be displayed with get_features' debug=True argument.

You will need to concatenate (see np.concatenate) the features and labels from all of the recordings together and create an example matrix and a one-hot set of labels. You will need to train a variety of networks to determine an appropriate architecture to recognize these. A sample architecture has been provided to get you started. It will not provide wonderful performance. A critical component is the batch normalization layer which we will discuss when we cover regularization. Note that this corpus does not have enough

data to support training a deep network, and you should keep your network relatively shallow. Split your training data (see scipy) with a 90/10 split to use 90% of it for training and 10% for validation. The validation data can be provided to keras's fit function and can show you how your model is performing during training. Do not expect wonderful accuracy at this stage. In general, it is very difficult to identify a speaker from 20 ms of speech and we rely on longer utterances to do identification.

During test, you will need to load in all of the features associated with each file of a specific speaker. Use the model to perform prediction on all of the examples of each file. This will result in a frames X speakers probability matrix for each of the 5 tests per speaker. Assume that frames are independent and take the product of the probabilities. Do so in the log domain (multiplication becomes addition) or your computation will underflow. You can then make a decision as to the identity of the speaker. If you were to pick randomly from the 25 Nutley speakers, you would have an error rate of 96%. While it is possible to do very well with this corpus, without too much effort you should be able to achieve an error rate of about 20%.

For each speaker that you classify, populate an entry in a confusion matrix. Confusion matrices are square matrices where each row represents the actual category of an example to be classified. Each column represents a classification decision. You can create an empty square matrix with numpy: np.zeros((n,n)) where n is the desired dimension. As examples are classified, the matrix cell corresponding to the actual and predicted class are incremented. Correct classifications lie along the diagonal and misclassifications are shown on the off diagonal. You can visualize the confusion matrix with scikit learn's ConfusionMatrixDisplay. Note that you don't need to use their matrix creation routines, just pass your confusion matrix to it, e.g.:

```
conf_disp = ConfusionMatrixDisplay(confusion, display_labels=speakers)
conf_disp.plot()
```

Create a table that shows each of the model architectures that you tried, the number of epochs you trained (you probably do not need to vary this), and the **error rate** (not the accuracy). Save labeled images of the confusion matrices for each of these. You are not required to execute a large number of experiments for this part, you will be asked to write a laboratory report where you explore different architectures more in a subsequent assignment.

If you are using PyCharm, you may wish to undock your plots as the plots in sciview tend to be small. When using Matplotlib, I like to make my plots interactive, the driver skeleton code has a line that turns this on, but you will need to set a breakpoint as the plots close when the program exits.

What to turn in:

- Parts I and II should be submitted separately to Canvas.

- Part I documents will be accepted as Word or PDF.
- Part II
    - Be sure to comment your programs, comments account for 10% of the program points.
    - If you are pair programming, make sure to add both programmers to the submission (the submission program has facilities for doing this)
    - Program files are to be uploaded individually or as a zip file. Be sure to preserve directory structure. Include a text log of your program running, showing your error rate. Include a screen shot of your confusion matrix. Top-level files should be at the top level of your zip file, not within a subdirectory.
    - All submission must have an affidavit. See submitting work for details and guidelines on commenting.
    - Figures can be submitted as standard graphic files (e.g. png, jpg) separately or in a Word document or pdf. When submitting zip archives, just include them.

References

Higgins, A. and Vermilyea, D. (1995). King Speaker Verification, (ed. I. T. a. T. Corporation). 3600 Market Street, Suite 810, Philadelphia, PA 19104-2653: Linguistic Data Consortium.