In the first part of this assignment, we will consider several general issues in representing features and their impact on classification. In the second part of the assignment, we will experiment with these strategies in the context of realistic data sets. Please make sure your read the lecture notes covering feature representations for this assignment.

Feature Transformations

A code file that is required for this assignment can be found here, and a colab notebook here.

1) Scaling

Consider a linearly separable dataset with two features:

```
data = ([[200, 800, 200, 800],
         [0.2,  0.2,  0.8,  0.8]])
labels = [[-1, -1, 1, 1]]
```

Consider the separator defined by $\theta = (0, 1), \theta_0 = -0.5$.

In order to apply the perceptron mistake bound (see notes), we transform our problem from $\theta^T x + \theta_0 = 0$ to some $\theta'^T x = 0$. We do this by appending θ_0 to θ, and appending 1 to x, as follows:

$$\theta'^T x = \begin{bmatrix} \theta_1 & \theta_2 & \dots & \theta_0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ 1 \end{bmatrix} = 0$$

In this phrasing, our new "θ" is (0, 1, -0.5). For a separator through the origin, recall that the margin of the data set is the minimum of $\gamma =$

$y^{(i)}(\theta^T x^{(i)})/||\theta||$ over all data points $(x^{(i)}, y^{(i)})$.

For the following questions, assume we are working in the transformed (3d) feature space, with perceptron through the origin, and where if the data has bounded magnitude R, then the theoretical upper bound on mistakes made by perceptron is $(\frac{R}{\gamma})^2$, for a separable data set.

You are free to use your perceptron algorithm implemented in the previous homework to answer the following questions. (Some parts require more runs of the perceptron algorithm than one could reasonably perform by hand.)

1A)

What is the margin γ of this data set with respect to that separator (up to 3 decimal places)?

0.26832816

Submit View Answer Ask for Help **100.00%**

You have infinitely many submissions remaining.

1B)

What is the theoretical bound on the number of mistakes perceptron will make on this problem?

8888912

Submit View Answer Ask for Help **100.00%**

You have infinitely many submissions remaining.

1C)

How many mistakes does perceptron through origin have to make in order to find a perfect separator on the data provided above, in the order given? (Try it on your computer, not by hand!) 666696

Submit View Answer Ask for Help **100.00%**

You have infinitely many submissions remaining.

1D)

If we were to multiply both original features of all of the points by $.001$, and considered the separator through origin $\theta = (0, 1, -0.0005)$, what would the margin of the new dataset be? 0.0003

Submit View Answer Ask for Help **100.00%**

You have infinitely many submissions remaining.

1E)

How would the performance of the perceptron (as predicted by the mistake bound) change? More mistakes

Submit View Answer Ask for Help **100.00%**

You have infinitely many submissions remaining.

1F)

If we multiplied just the first original feature (first row of the data) by .001, and used our original separator, what would the new margin be? 0.2683281572999748

Submit View Answer Ask for Help **100.00%**

You have infinitely many submissions remaining.

1G)

What would the mistake bound be in this case? 31.66666

Submit View Answer Ask for Help **100.00%**

You have infinitely many submissions remaining.

1H)

Run the perceptron algorithm on this data; how many mistakes does it make?

11

Submit View Answer Ask for Help **100.00%**

You have infinitely many submissions remaining.

2) Encoding Discrete Values

Some data sets have features that take on discrete values drawn from a set. Examples might be:

- which section of a class a student is in (1, 2, 3, 4)
- manufacturer of a cell phone (Samsung, Xiaomi, Sony, Apple, LG, Nokia)
- which laboratory performed a particular medical test

Sometimes they already have an obvious encoding into integers; other times, they don't but it's easy to make one (e.g., Samsung = 1, Xiaomi = 2, Sony = 3, Apple = 4, LG = 5, Nokia = 6)

2A) Let's consider the case of the cell phones, using the encoding above, and imagine there is some prediction problem, such as predicting whether the phone will last three years, for which we have the data set:

```
data =   [[2, 3,  4,  5]]
labels = [[1, 1, -1, -1]]
```

What value of θ and θ_0 would we get when running perceptron on this data? You are free to use the perceptron implemented in homework 2.

Enter a Python list with two floats, one for θ and one for θ_0. [-2., 7.]

Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

2B) What prediction would we make about other phone types based on this classifier?

Enter a Python list with two labels (1 or -1), the first one for a Samsung phone and the second for a Nokia phone.

[1, -1]

Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

2C)

Are these predictions meaningful given the training data we used? No

Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

2D) It is common to encode a feature which takes on a value from a set of discrete values, not as a single multi-valued feature, but using a *one hot* encoding.

Here, assume you have a feature f which can take on any value from the set $\{1, 2, ..., k\}$. If f takes on value i, then we represent it as a vector of length k of all zeros, except for a +1 at the ith coordinate.

Write a function `one_hot` that takes as input x, a single feature value (between 1 and k), and k, the total possible number of values this feature can take on, and transform it to a numpy column vector of k binary features using a one-hot encoding (remember vectors have zero-based indexing).

For example, `one_hot(3,7)` should return a column vector of length 7 with the entry at index 2 taking value 1 (indices start at 0) and other entries taking value 0.

```python
def one_hot(x, k):
    a = np.zeros((k, 1))
    a[x-1,0] = 1.0
    return a
```

Run Code | Submit | View Answer | Ask for Help 100.00%

You have infinitely many submissions remaining.

2E) What happens if we use one-hot encoding on the data set part 2A) above, and put it into the perceptron? Recall that for a classifier $h(x)$, the prediction is $+1$ if $h(x) > 0$ and -1 otherwise. Further note that the perceptron algorithm makes an update whenever $y^{(i)}(\theta^T x^{(i)} + \theta_0) \leq 0$.

2E i) What is the separator produced by the perceptron algorithm?

Enter a Python list with 7 floats, six for θ and one for θ_0. [0., 2., 1., -2., -1., 0., 0.]

Submit | View Answer | Ask for Help 100.00%

You have infinitely many submissions remaining.

2E ii) What are the predictions for Samsung and Nokia?

Enter a Python list with two labels (1 or -1), the first one for a Samsung phone and the second for a Nokia phone.

[-1, -1]

Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

2E iii) What are the distances for the Samsung and Nokia data points from the separator?

Enter a Python list with two distances, the first one for a Samsung phone and the second for a Nokia phone.

[0., 0.]

Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

2F) Now, what if we have this dataset:

```
data =   [[1, 2, 3, 4, 5, 6]]
labels = [[1, 1, -1, -1, 1, 1]]
```

Is it linearly separable in the original encoding? No

Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

2G) Is it linearly separable in the one-hot encoding? If so, provide the separator found by the perceptron.

Enter a Python list with 7 floats, six for θ and one for θ_0 or `'none'` [1., 1., -2., -2., 1., 1., 0.]

Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

2H) Enter an assignment of data values to labels (with distinct data points) that is not linearly separable using the one-hot encoding, or enter None if no such assignment exists.

Enter a Python list with 6 tuples `(value, label)` or `'none'` 'none'

Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

3) Polynomial Features

One systematic way of generating non-linear transformations of your input features is to consider the polynomials of increasing order. Given a feature vector $x = [x_1, x_2, ..., x_d]^T$, we can map it into a new feature vector that contains all the factors in a polynomial of order d. For example, for $x = [x_1, x_2]^T$ and order 2, we get

$$\phi(x) = [1, x_1, x_2, x_1 x_2, x_1^2, x_2^2]^T$$

and for order 3, we get

$$\phi(x) = [1, x_1, x_2, x_1x_2, x_1^2, x_2^2, x_1^2x_2, x_1x_2^2, x_1^3, x_2^3]^T.$$

In the code file, we have defined `make_polynomial_feature_fun` that, given the order, returns a feature transformation function (analogous to ϕ in the description). You should use it in doing this problem.

3A)

Enter a list of 6 integers indicating the number of polynomial features of degrees [1, 10, 20, 30, 40, 50] for a 2-dimensional feature vector. [3, 66, 231, 496, 861, 1326]

Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

3B) Consider this data-set of four points in two-dimensional space:

```
data = ([[1, 1, 2, 2],
         [1, 2, 1, 2]])
labels = [[-1, 1, 1, -1]]
```

It is standardly called the "exclusive-or" or "xor" problem. These points are not linearly separable, and you could interpret each point as being a pair of truth values, with their label being the XOR of the values.

In the code file, we have defined 4 sample data sets, (1) `super_simple_separable_through_origin`, (2) `super_simple_separable`, (3) `xor`, and (4) `xor_more`. On your own machine, you should run the code we have provided (`test_with_features`) for various orders of polynomial features and enter below the order of the smallest feature that separates the data. Make sure that you have included your implementation of `perceptron` in that file or you can use the implementation we have provided. You may need to adjust the number of iterations that the perceptron runs.

The separators are displayed when the code runs; it's instructive to watch them to see the range of separators that these non-linear

transformations produce. Note that the separators are drawn by evaluating the feature transformations on a grid of points in the feature space and using the separator to classify them. (Note: If you have issues with the graphic not moving forward, try pressing the keys within your terminal.)

Enter a Python list of integers indicating the smallest polynomial order for which a separator exists for each of the four datasets in the code file (in order). [1,1,2,3]

Submit View Answer Ask for Help **100.00%**

You have infinitely many submissions remaining.

Experiments

A code and data folder that will be necessary for doing this homework can be found at the top of the page. In the file `code_for_hw3_part2.py`, include your learner code from HW 2. **You will want to modify the evaluation algorithms so that they take a `T` argument to pass to the learners.**

The rest of this assignment will require running the code on your computer; we will be asking only for the results of your runs.

4) Evaluating algorithmic and feature choices for AUTO data

We now want to build a classifier for the auto data, with a focus on the numeric data. In the `code_for_hw3_part2.py`, we have supplied you with the `load_auto_data` function, which can read the relevant .tsv file. It returns a list of dictionaries, one for each data item.

We then specify what feature function to use for each column in the data. The file `hw3_part2_main.py` has an example that constructs the data and label arrays using `raw` features for all the columns.

In the list `features` of `hw3_part2_main.py`, you will find a list of feature name, feature function tuples. There are three options for feature functions: `raw`, `standard` and `one_hot`. `raw` uses the original value; `standard` subtracts out the mean value and divides by the standard deviation; and `one_hot` will one-hot encode the input, as described in the notes.

The function `auto_data_and_labels` processes the dictionaries and return `data, labels`. `data` has dimension $(d, 392)$, where d is the total number of features specified, and `labels` has dimension $(1, 392)$. The data in the file is sorted by class, but it will be shuffled when loaded.

We have included staff implementations of `perceptron` and `average perceptron` in `code_for_hw3_part2.py`. Using the feature arrays and these implementations, you will be able to compute θ and θ_0.

We have also included staff implementations of `eval_classifier` and `xval_learning_alg` (in the same code file). You should use these functions to report accuracies.

4.1) Making choices

We know of two algorithm classes: perceptron and averaged perceptron (which we implemented in HW 1). We have a several parameters that specify the settings for these learning algorithms.

A) Which parameters should we use for the learning algorithm? In the perceptron and averaged perceptron, there is a single parameter, T, the number of iterations.

B) Which features should we use? We have lots of choices here: we can use any subset of the data columns and for each column we have choices of how to compute features.

C) We will use expected accuracy, estimated by 10-fold cross-validation (we have included the definition in the code file), to make these choices of parameters.

- We will try two types of algorithms: perceptron and averaged perceptron.
- We will try 3 values of T: $T = 1$, $T = 10$, $T = 50$.
- We will try 2 feature sets:

1. `[cylinders=raw, displacement=raw, horsepower=raw, weight=raw, acceleration=raw, origin=raw]`

2. `[cylinders=one_hot, displacement=standard, horsepower=standard, weight=standard, acceleration=standard, origin=one_hot]`

Perform 10-fold cross-validation for all combinations of the two algorithms, three T values, and the two choices of feature sets. It will be worthwhile investing in a piece of code to carry out all of the evaluations, in case you need to do this more than once.

In general, you should shuffle the dataset before evaluating, but for this exercise, please use `hw3.xval_learning_alg`, which shuffles the dataset for you, so that your results match ours.

4.1C i)

Enter accuracies (perceptron, averaged perceptron) for T=1, feature set 1:

[0.6526282051282052, 0.8441025641025641]

Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

4.1C ii)

Enter accuracies (perceptron, averaged perceptron) for T=1, feature set 2:

(0.7908333333333333, 0.9004487179487182)

Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

4.1C iii)

Enter accuracies (perceptron, averaged perceptron) for T=10, feature set 1:

(0.7423076923076924, 0.8366025641025641)

Submit View Answer Ask for Help **100.00%**

You have infinitely many submissions remaining.

4.1C iv)

Enter accuracies (perceptron, averaged perceptron) for T=10, feature set 2:

(0.8061538461538461, 0.8979487179487181)

Submit View Answer Ask for Help **100.00%**

You have infinitely many submissions remaining.

4.1C v)

Enter accuracies (perceptron, averaged perceptron) for T=50, feature set 1:

(0.6909615384615384, 0.8366025641025641)

Submit View Answer Ask for Help **100.00%**

You have infinitely many submissions remaining.

4.1C vi)

Enter accuracies (perceptron, averaged perceptron) for T=50, feature set 2:

(0.8060256410256409, 0.9005128205128207)

Submit View Answer Ask for Help 100.00%

You have infinitely many submissions remaining.

Now we have the data we need to make rational choices.

4.1D) Which algorithm class is typically more effective?

Pick one: Averaged Perceptron

Submit View Answer Ask for Help 100.00%

You have infinitely many submissions remaining.

4.1E) For the better algorithm, which combination of T and feature would you use? Consider expected accuracy as of primary importance, take into account running time for near ties in accuracy.

Enter a tuple of two integers (T, feature_set): (1, 2)

Submit View Answer Ask for Help 100.00%

You have infinitely many submissions remaining.

4.2) Analysis

4.2 A) For the best algorithm type, best T and best feature set, construct your best classifier (θ, θ_0) using all the data. Based on the values of the coefficients, which feature has the most impact on the output predictions?

Choose the name of one feature: weight

Submit | View Answer | Ask for Help | 100.00%

You have infinitely many submissions remaining.

4.2 B) (Optional) Is there any set of two features you can use to attain comparable results as your best accuracy? What are they?

5) Evaluating algorithmic and feature choices for review data

In the code file (and the colab notebook), we have supplied you with the `load_review_data` function, that can be used to read a .tsv file and return the labels and texts. We have also supplied you with the `bag_of_words` function, which takes the raw data and returns a dictionary of unigram words. The resulting dictionary is an input to `extract_bow_feature_vectors` which computes a feature matrix of ones and zeros that can be used as the input for the classification algorithms. The file `hw3_part2_main.py` has code for constructing the data and label arrays. Using these arrays and our implementation of the learning algorithms, you will be able to compute θ and θ_0. You will need to add your (or the one written by staff) implementation of perceptron and averaged perceptron.

5.1) Making choices

We have two algorithm classes: perceptron and averaged perceptron. We have a couple of choices of parameters to make to completely specify the learning algorithms.

5.1A) Which parameters should we use for the learning algorithm? In the perceptron and averaged perceptron, there is a single parameter, T, the number of iterations.

5.1B) Which features should we use? We could do variations of bag-of-words, for example, simply indicating whether a word is present or, alternatively, using a count of how many times it is present. We can also remove commonly used words with little information; the code distribution includes a file of those words: `stopwords.txt`. You're welcome to explore these on your own; we'll use only a binary indicator for all the words.

5.1C) Perform 10-fold cross-validation for all combinations of the two algorithms and three T values (1, 10, 50). Record the accuracies for each combination (there should be 6 values total).

Note: These tests may take a couple of minutes to run; don't expect instant response, but it shouldn't run for 10 minutes.

Now we have the data we need to make rational choices.

5.1D) Which algorithm class is typically more effective?

Pick one: Averaged Perceptron

Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

5.1E) For the better algorithm, which value of T would you use? Consider expected accuracy as of primary importance, take into account running time for near ties in accuracy.

Enter a value of T: 10

Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

5.1F) For the better algorithm and best value of T, what is your accuracy?

Enter a number between 0 and 1: 0.8237

Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

5.2) Analysis

For the best algorithm and best T, construct your best classifier (θ, θ_0) using all the data.

Note: We have included a function called `reverse_dict` in `code_for_hw3_part2.py` that you may find convenient. You are not required to use this function.

5.2A) What are the 10 most positive words in the dictionary, that is, the words that contribute most to a positive prediction?

Enter a Python list of ten strings: ['great', 'individually', 'bright', 'yummy', 'skeptical', 'perfect', 'easily

Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

5.2B) What are the 10 most negative words in the dictionary, that is, the words that contribute most to a negative prediction.

Enter a Python list of ten strings: ['worst', 'awful', 'unfortunately', 'horrible', 'stuck', 'changed', 'disap

Submit | View Answer | Ask for Help | 100.00%

You have infinitely many submissions remaining.

5.2C) (Optional) You might find it amusing to find the most positive and most negative reviews. That is, ones with the most positive and negative signed distance to the hyperplane.

6) Evaluating features for MNIST data

This problem explores how well the perceptron algorithm works to classify images of handwritten digits, from the well-known ("MNIST") dataset, buiding on your thoughts from lab about extracting features from images. This exercise will highlight how important feature extraction is, before linear classification is done, using algorithms such as the perceptron.

Dataset setup

Often, it may be easier to work with a vector whose spatial orientation is preserved. In previous parts, we have represented features as one long feature vector. For images, however, we often represent a m by n image as a `(m,n)` array, rather than a `(mn,1)` array (as the previous parts have done).

In the code file, we have supplied you with the `load_mnist_data` function, which will read from the provided image files and populate a dictionary, with image and label vectors for each numerical digit from 0 to 9. These images are already shaped as `(m,n)` arrays.

6.1) Feature extraction

In the real world, there may be complicated ways to extract meaningful features from images, but in this section we will explore several simple methods.

6.1A) You may notice that some numbers (like 8) take up more horizontal space than others (like 1). We can compute a feature based on the average value in each row. Write a Python function that takes in a `(m,n)` array and returns a `(m,1)` array, where element `i` is the average value in row `i`.

```
                lambda a: [np.average(a)],
                axis=1,
                arr=xi
            )
            for xi
            in x
        ])
    elif ndims == 2:
        return np.apply_along_axis(
            lambda a: [np.average(a)],
            axis=1,
            arr=x
        )

```

Run Code | Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

6.1B) We can also compute a feature based on the average value in each column. Write a Python function that takes in a `(m,n)` array and returns a `(n,1)` array, where element `j` is the average value in column `j`.

```
        return np.apply_along_axis(
            lambda a: [np.average(a)],
            axis=0,
            arr=arr
        ).T
    if ndims == 3:
        return np.array([
            f(xi)
            for xi
            in x
        ])
    elif ndims == 2:
        return f(x)

```

Run Code | Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

6.1C) Finally, you may notice that some features are more "top heavy" while others are more "bottom heavy." Write a function that takes in a `(m,n)` array and returns a `(2,1)` array, where the first element is the average value in the top half of the image, and the second element is the average value in the bottom half of the image.

You may use the `cv` function from homework 1.

```
    @return (2,n_samples) array where the first entry of each column is the average of the
    top half of the image = rows 0 to floor(m/2) [exclusive]
    and the second entry is the average of the bottom half of the image
    = rows floor(m/2) [inclusive] to m
    """
    ndims = len(x.shape)
    def f(arr):
        n, m = x.shape
        return cv([np.average(x[0:n//2,:]), np.average(x[n//2:,:])])
    if ndims == 3:
        return np.array([
            f(xi)
            for xi
```

Run Code | Submit | View Answer | Ask for Help 100.00%

You have infinitely many submissions remaining.

Important: In `hw3_part2_main.py`, you may need to **modify** these functions to accept a data matrix of size `(n, 28, 28)`, or you may use these functions in other ways (loops are allowed).

6.2) Feature evaluation

We can use these features to distinguish between numbers.

Important: For this section, we will be using `T=50` with the base perceptron algorithm to train our classifier and 10-fold cross validation to evaluate our classifier. A function called `get_classification_accuracy` has already been implemented for you in `code_for_hw3_part2` to compute the accuracy, given your selected data and labels.

You *must* use our implementation of cross validation to report accuracies (you may call `hw3.xval_learning_alg`).

6.2A) First we will find baseline accuracies using the raw 0-1 features.

Convert each image into a `(28*28, 1)` vector for input into the perceptron algorithm. **Hint:** `np.reshape` may be helpful here.

Run the perceptron on four tasks: 0 vs. 1, 2 vs. 4, 6 vs. 8, and 9 vs. 0.

Enter a list of accuracies [0 vs. 1, 2 vs. 4, 6 vs. 8, 9 vs. 0]: (0.975, 0.8641666666666665, 0.9479166666666667, 0.647083333333

Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

Now let's evaluate the features we extracted.

6.2B) Using the extracted features from above, run the perceptron algorithm on the set of 0 vs. 1 images.

Enter a list of accuracies [row, column, top/bottom]: [0.48125, 0.6375, 0.48125]

Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

6.2C) Using the extracted features from above, run the perceptron algorithm on the set of 2 vs. 4 images.

Enter a list of accuracies [row, column, top/bottom]: [0.7754166666666668, 0.49749999999999994, 0.497499999999999

Submit View Answer Ask for Help 100.00%

You have infinitely many submissions remaining.

6.2D) Using the extracted features from above, run the perceptron algorithm on the set of 6 vs. 8 images.

Enter a list of accuracies [row, column, top/bottom]: [0.92125, 0.52125, 0.5650000000000001]

Submit View Answer Ask for Help 100.00%

You have infinitely many submissions remaining.

6.2E) Using the extracted features from above, run the perceptron algorithm on the set of 9 vs. 0 images.

Enter a list of accuracies [row, column, top/bottom]: [0.49749999999999994, 0.5041666666666667, 0.497499999999999

Submit View Answer Ask for Help 100.00%

You have infinitely many submissions remaining.

6.2F) (Optional) What does it mean if a binary classification accuracy is below 0.5, if your dataset is balanced (same number from each class)? Are these datasets balanced?

6.2G) (Optional) Feel free to classify other images from each other. Which combinations perform the best, and which perform the worst? Do these make sense? Other than row and column average, are there any other features you could think of that would preserve some spatial information?