IE643 Deep Learning : Theory and Practice **July-Dec 2025**

Homework 2: For self-practice

1 Questions

Consider the code for perceptron that you have written for 2D data. Now let us extend the code to process data from $\mathbb{R}^d$. All exercises should be attempted with Python. Use of AI tools is encouraged for getting some help with the following problems. However it is just **foolish** (sorry about the harsh term) to just fetch the answers from AI tools, without thinking about the problems yourself.

1. Consider the folder at the following link. This folder contains images corresponding to two classes namely `"o"` and `"ki"`. Your aim is to train your perceptron with this data. Towards this purpose, do the following. Feel free to visualize the data first.
   (a) Write Python code to read an image in `.png` format as an object in $\mathbb{R}^d$. Explain your process of conversion.
   (b) Consider the class label `"o"` as label +1 and class label `"ki"` as label -1. Corresponding to each i-th image, store the object pair (x^i, y^i) where $x^i \in \mathbb{R}^d$ denotes the feature vector of the image and $y^i \in \{+1, -1\}$ denotes class label for i-th image. Use a suitable data structure (e.g. list, dictionary) to store these pairs for all images in the data folder.
   (c) Check the distribution of each class in the data set. Is there a balance in the class distribution?
   (d) Shuffle the list of data items and select the first 80% as train set $\mathcal{S}$ and remaining 20% as test set $\mathcal{T}$. Verify if both the train and test sets have similar class distributions.
   (e) Use the train set $\mathcal{S}$ to train your perceptron. Use a suitable stopping criterion to stop your training.
   (f) After every training epoch (that is, after a single pass through the train set examples), compute the train set accuracy as the percentage of correctly classified examples in the train set using the current weights of perceptron. Similarly compute the test set accuracy after each training epoch. Record the accuracy values in suitable lists.
   (g) After the training is done, plot the train set accuracy vs epochs (accuracy along the vertical axis and number of epochs across horizontal axis). Use appropriate axis labels and discuss your observations of the training set accuracy.
   (h) After the training is done, plot the test set accuracy vs epochs (accuracy along the vertical axis and number of epochs across horizontal axis). Use a different color than the previous plot. Use appropriate axis labels and discuss your observations of the test set accuracy.
   (i) Compare the train set accuracy behavior with that of test set. Discuss your observations.
   (j) From the experiments, discuss if the data set is linearly separable or not. Explain with suitable reasons.
2. Consider the folder at the following link . This folder contains audio files in *.wav* format corresponding to two classes namely `"gac"` and `"flu"`. Your aim is to train your perceptron with this data. Towards this purpose, do the following. Feel free to listen to the audio data first.

(a) Write Python code to read an audio file in `.wav` format as an object in $\mathbb{R}^d$. Explain your process of conversion.

(b) Consider the class label `"gac"` as label +1 and class label `"flu"` as label -1. Corresponding to each i-th audio, store the object pair (x^i, y^i) where $x^i \in \mathbb{R}^d$ denotes the feature vector of the audio and $y^i \in \{+1, -1\}$ denotes class label for i-th audio. Use a suitable data structure (e.g. list, dictionary) to store these pairs for all audio files in the data folder.

(c) Check the distribution of each class in the data set. Is there a balance in the class distribution?

(d) Shuffle the list of data items and select the first 80% as train set $\mathcal{S}$ and remaining 20% as test set $\mathcal{T}$. Verify if both the train and test sets have similar class distributions.

(e) Use the train set $\mathcal{S}$ to train your perceptron. Use a suitable stopping criterion to stop your training.

(f) After every training epoch (that is, after a single pass through the train set examples), compute the train set accuracy as the percentage of correctly classified examples in the train set using the current weights of perceptron. Similarly compute the test set accuracy after each training epoch. Record the accuracy values in suitable lists.

(g) After the training is done, plot the train set accuracy vs epochs (accuracy along the vertical axis and number of epochs across horizontal axis). Use appropriate axis labels and discuss your observations of the training set accuracy.

(h) After the training is done, plot the test set accuracy vs epochs (accuracy along the vertical axis and number of epochs across horizontal axis). Use a different color than the previous plot. Use appropriate axis labels and discuss your observations of the test set accuracy.

(i) Compare the train set accuracy behavior with that of test set. Discuss your observations.

(j) From the experiments, discuss if the data set is linearly separable or not. Explain with suitable reasons.

3. Consider the folder at the following link. This folder contains a single file `reviews.csv`. In this file, you will observe that the first row contains the header information and in every other row, there is a review in textual form and a corresponding label (0 or 1) associated with the review indicating if the review inclines towards a "good" rating (when label is 1) or a "bad" rating (when label is 0). Your aim is to train your perceptron with this data. Towards this purpose, do the following. Feel free to read and understand the review data first.

(a) Write Python code to read each review in text format from `reviews.csv` file as an object in $\mathbb{R}^d$. Explain your process of conversion.

(b) Consider the class label `"0"` as label -1 and class label `"1"` as label +1. Corresponding to each i-th review, store the object pair (x^i, y^i) where $x^i \in \mathbb{R}^d$ denotes the feature vector of the text review and $y^i \in \{+1, -1\}$ denotes class label for i-th review. Use a suitable data structure (e.g. list, dictionary) to store these pairs for all reviews in the file `reviews.csv`.

(c) Check the distribution of each class in the data set. Is there a balance in the class distribution?

(d) Shuffle the list of data items and select the first 80% as train set $\mathcal{S}$ and remaining 20% as test set $\mathcal{T}$. Verify if both the train and test sets have similar class distributions.

(e) Use the train set $\mathcal{S}$ to train your perceptron. Use a suitable stopping criterion to stop your training.

(f) After every training epoch (that is, after a single pass through the train set examples), compute the train set accuracy as the percentage of correctly classified examples in the train set using the current weights of perceptron. Similarly compute the test set accuracy after each training epoch. Record the accuracy values in suitable lists.

(g) After the training is done, plot the train set accuracy vs epochs (accuracy along the vertical axis and number of epochs across horizontal axis). Use appropriate axis labels and discuss your observations of the training set accuracy.

(h) After the training is done, plot the test set accuracy vs epochs (accuracy along the vertical axis and number of epochs across horizontal axis). Use a different color than the previous plot. Use appropriate axis labels and discuss your observations of the test set accuracy.

(i) Compare the train set accuracy behavior with that of test set. Discuss your observations.

(j) From the experiments, discuss if the data set is linearly separable or not. Explain with suitable reasons.
