Current Implementation and Experimental Results:

Following are the steps and decisions we took while doing this project:

(a) All the text below refers to the repository: https://github.com/ar-ambuj23/ml-project2019

(b) As an initial step, to understand the working of Linear Programming using Python, we implemented a small demo problem for minimization using LP using PULP library. But later, we found that the *linprog* module in SCiPy serves the same purpose and is also faster and stable. [0_*pulp_tutorial.ipynb*]

(c) For our study on linear separability for a given dataset, we had planned on studying the dataset on the lines of Convex Hull, Linear Programming and SVM.

(d) For a dataset which is not separable by a linear hyperplane, we will try different transformation functions from a library of pre-defined functions and transform the data using these one-by-one.

(e) Then, a transformation function which makes the data separable will be approximated by piecewise defined linear functions.

(f) Currently, we have The Banknote Authentication Dataset and The Iris Dataset from The UCI ML Repository for our analysis. [*datsetsfolder*]

(g) We have also defined a library of transformation functions which is used to transform the data into a different space. [*func_library.ipynb*]

(h) We started our analysis on the Banknote dataset by making convex hulls around the data points, testing for linear separability using LP and also trying to train a Perceptron and SVM(Linear and RBF kernels). We found that the convex hulls intersect, the LP reports the data to be linearly non-separable and also no perfect classifier was given by Perceptron and the SVM. Thus, the data is linearly not separable in its original space. [1_*LP_banknote_dataset.ipynb*]

(i) As an complementary example, we did everything mentioned above on the Iris dataset, in which one class is linearly separable from the others. All the three methods gave expected results for linear separability. [2_*LP_iris_dataset.ipynb*]

(j) Next, we tried a sample transformation function on the Banknote dataset and tested for linear separability. The function we used was $x^2 + x^3$. The dataset was still not linearly separable. [1*a_LP_banknote_dataset_testing_x*2*_x*3*.ipynb*]

(k) Then, we tried all the transformation functions we have in our library on the Banknote dataset. But still we could not find a transfromation function which could make the data linearly separable in the new space. [1*b_LP_banknote_dataset_all_transformations.ipynb*]

(l) We will continue this analysis on the Banknote dataset to find the transformation function which works well.

(m) Our next part of the study was to test for linear separability when the data is transfromed from a lower dimensional space to a higher dimensional space.

(n) For this we made a demo dataset, which we have saw in the class, and performed all the three methods on non-transformed data and then on the transformed data to verify our hypothesis. The dataset was not linearly separable in the 1D space but it became linearly separable in 2D space. [3_*feature_transformation_in_higher_d.ipynb*]

Plan ahead:

(a) We'll use different datasets and try out different transformation functions on them.

(b) Convex Hull, Linear Programming and SVM then will be used to see whether transformed data is linearly separable or not.

(c) Then we'll approximate that transformation function ϕ using 'Crosstalk reservoir' which contains Piece-wise linear functions.

(d) After approximation of ϕ we'll use 'Crosstalk Reservoir' to project data into higher-D and compare the results with the other ϕ (Transformation Functions)

References:

(a) Gouhei Tanakaa,b,1 , Toshiyuki Yamanec , Jean Benoit Herouxc : Recent Advances in Physical Reservoir Computing: A Review

(b) H. Burgsteiner. Training networks of biological realistic spiking neurons for real-time robot control. In Proc. of EANN, pages 129–136, Lille, France, 2005.

(c) H. Jaeger. The "echo state" approach to analyzing and training recurrent neural networks. Technical report, German National Research Center for Information Technology, 2001.

(d) H. Jaeger. Short term memory in echo state networks. Technical report, German National Research Center for Information Technology, 2001.

(e) W. Maass, T. Natschlager, and M. H. Fading memory and kernel properties of generic cortical microcircuit models. Journal of Physiology, 98(4-6):315–330, 2004

(f) Crosstalk calculator: https://www.eeweb.com/tools/microstrip-crosstalk

(g) P. Joshi and W. Maass. Movement generation and control with generic neural microcircuits. In Proc. of BIO-AUDIT, pages 16–31, 2004.

(h) H. Burgsteiner. On learning with recurrent spiking neural networks and their applications to robot control with real-world devices. Ph.D. thesis, Graz University of Technology, 2005

(i) W. Maass, R. A. Legenstein, and H. Markram. A new approach towards vision suggested by biologically realistic neural microcircuit models. In Proc. of the 2nd Workshop on Biologically Motivated Computer Vision, 2002.

(j) M. Oubbati, P. Levi, M. Schanz, and T. Buchheim. Velocity control of an omnidirectional RoboCup player with recurrent neural networks. In Proc. of the RoboCup Symposium, pages 691–701, 2005.

(k) Zhao, Xi Niu, Lingfeng Shi, Yong. (2013). A Simple Regularized Multiple Criteria Linear Programs for Binary Classification. Procedia Computer Science. 18. 58-61. 10.1109/WI-IAT.2013.150.

(l) UC Berkley's lecture on Linear Classification: https://people.eecs.berkeley.edu/r̃ussell/classes/cs194/f11/lectures/CS194%20Fall%202011%20Lecture%2005.pdf