CS 6350 Machine Learning

Project Final Report

Alok Jadhav (u1265865) Ambuj Arora (u1265867)

10 December 2019

1. **Introduction**

   Unconventional computing is computing by a wide range of new or unusual methods. It is also known as alternative computing. In our project, we explore the idea of 'Physical Reservoir Computing' via 'Crosstalk between conducting wires' as a Transformation function ϕ which can project the lower dimensional data into higher dimensions such that the transformed data will be separable linearly.

2. **What is Reservoir Computing?**

   Reservoir computing is a computational framework suited for temporal/sequential data processing. A reservoir computing system consists of a reservoir for mapping inputs into a high-dimensional space and a readout for pattern analysis from the high-dimensional states in the reservoir. This means that the internal W (weight) matrix of a reservoir kept the same even after the training phase. Thus, the major advantage of reservoir computing compared to other recurrent neural networks is fast learning, resulting in low training costs. Another advantage is that the reservoir without adaptive updating is amenable to hardware implementation using a variety of physical systems, substrates, and devices.

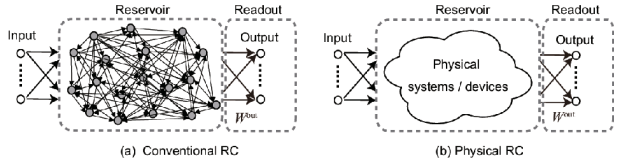

Figure 1: Physical Reservoir: There are 3 layers in above RC architecture. Input, Reservoir, and Readout Layer. Reservoir projects input into higher dimensional space.

3. **Why do we need out data to be projected into higher-dimension?** Cover's Theorem: A complex pattern-classification problem, cast in a high-dimensional space non-linearly, is more likely to be linearly separable than in a low-dimensional space, provided that space is not densely populated. So, according to the cover's theorem, it's more probable to learn a linear classifier when the data is projected in high-dimension space non-linearly. That's why 'Reservoir' plays the biggest role in RC.

4. **Why Learn Linear separable classifier? Why not non-linear?**

   Occam's Razor: The problem-solving principle that states "Entities should not be multiplied without necessity." It is sometimes paraphrased by a statement like "The simplest solution is most likely the right one." The simpler hypothesis are better. That's why we try to learn a 'linear separable' classifier instead of 'non-linear'.

5. **Application of Physical Reservoir in Generation of Waves**

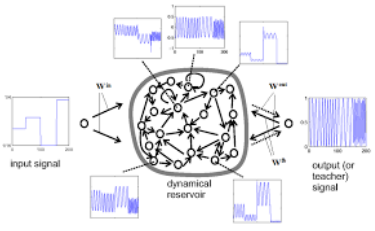

Figure 2: The reservoir as a signal generator. Input is a 'Step signal' and output is a 'Sine Wave' with variable frequency.

6. **Signal transfer in wires: DAC and ADC**

Most of the signals directly encountered in science and engineering are continuous: light intensity that changes with distance; voltage that varies over time; a chemical reaction rate that depends on temperature, etc. Analog-to-Digital Conversion (ADC) and Digital-to-Analog Conversion (DAC) are the processes that allow digital computers to interact with these everyday signals. Digital information is different from its continuous counterpart in two important respects: it is sampled, and it is quantized. Both of these restrict how much information a digital signal can contain. Incoming signals from any device can be both digital or analog. By using ADC and DAC we can convert that signal into 'Analog Signal' such that 'Voltage' will be the major feature that can define any signal.

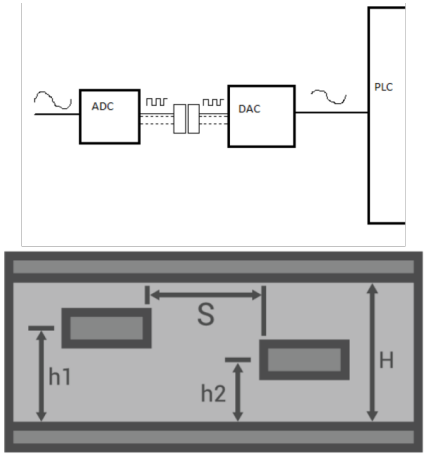

Figure 3: Image 1: DAC converts incoming Digital signals into Analog signals. These signals then can be passed/transmitted through wires. Image 2: Two conducting lines can have crosstalk between them if one of it is carrying signal

Crosstalk is any phenomenon by which a signal transmitted on one circuit or channel of a transmission system creates an undesired effect in another circuit or channel and is usually caused by undesired capacitive, inductive, or conductive coupling from one circuit or channel to another. This phenomenon is undesired in electronics. But we are trying to use this phenomenon as a 'Physical Reservoir' to project our data into higher dimension.

$$T_{RT} = 1.017\sqrt{S_r^2 - 0.475 + 0.67} \cdot L \cdot 2 \qquad V_{crosstalk} = V \cdot \frac{1}{1 + \left(\frac{S_{eff}^2}{h_{1eff} h_{2eff}}\right)} \cdot \frac{T_{RT}}{T_R}$$

$$S_{eff} = \sqrt{S^2 + (h_2 - h_1)^2}$$

$$h_{1eff} = \frac{h_1 \cdot (H - h_1)}{h_1 + (H - h_1)}$$

$$h_{2eff} = \frac{h_2 \cdot (H - h_2)}{h_2 + (H - h_2)}$$

Figure 4: $V_{crosstalk}$: Induced voltage on child circuit. V = Input voltage. $T_{RT}, S_{eff}, h_{1eff}, h_{2eff}$: physical parameters of crosstalk wires

We can see from the equations that $V_{crosstalk}$ is proportional to 'V' and **in fact it is in the form of** $V_{crosstalk} = V * m + b$ **here 'b' is ground reference voltage for child circuit. And 'm'** $= 1/(1 + (S^2/h_1 * h_2))(T_{RT}/T_R)$**. So, if we keep 'm' and 'b' in control, then we'll be able to get a linear relationship between** $V_{crosstalk}$ **and V**

7. **Ideas Explored from class:**

1. Support Vector Machines: For this project we can use 'Hard SVM' where C=∞.SVMs are also called as large margin classifier. SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.
2. Ensembles: We can use bunch of weak learners to learn a strong classifier. We use this concept during piecewise linear approximation of non-linear function. We can define multiple linear functions to represent or approximate a complex non-linear function.
3. Kernels as feature transformation functions for SVM: It is a mathematical trick to calculate values of transformed features without actually transforming them into higher dimension. Kernel trick uses different non-linear functions ϕ to transform features into higher-D. They are called as 'Kernels'. Hypothesis of SVM= $g(W^T x + b)$. SVM finds a hyperplane that can separate the given dataset with the help of a few 'Support Vectors' such that difference between two classes is maximum.

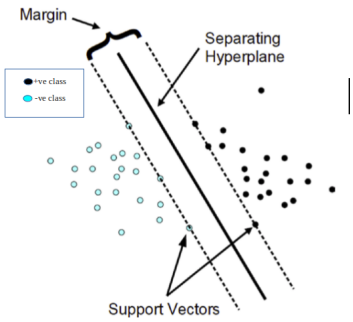

Figure 5: SVM separates two classes as a large margin classifier

4. Feature Transformation function [ϕ]: Feature transformation is simply a function that transforms features from one representation to another. We can use different 'Transformation function' to transform the data into higher dimension hoping that it will be separable in that dimension. (By cover's theorem). There is no universal 'Transformation function' that can project data into separable dimension. We need to try different 'Transformation functions' to see which one works for that particular dataset.

8. **Other methods to prove linear separabiltiy: Convex Hull and Linear Programming**

After the data is transformed by ϕ, we need methods to determine whether the given transformation is linearly separable in the new space or not. For that we have tried twp other methods, apart from SVM:

The convex hull of a given set X may be defined as: The (unique) minimal convex set containing X. The convex hull of a finite point set 'S' is the set of all convex combinations of its points. In a convex combination, each point x_i in 'S' is assigned a weight or coefficient α_i in such a way that the coefficients are all non-negative and sum to one, and these weights are used to compute a weighted average of the points.Expressing this as a single formula, the convex hull is the set:

If convex hulls of two classes intersect with each other then they are not linearly separable in that dimension. So, for classes to be linearly separable their convex hulls should not intersect with each other.
Linear Programming is an optimization technique to find the best outcome of a mathematical model for some given objective function and constraints which are represented as linear relationships. The

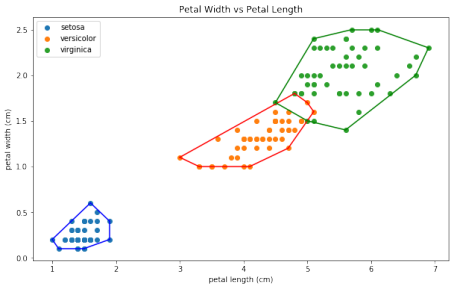

Figure 6: Iris dataset fitted with convex-hull

objective function defines a quantity to be optimized and the goal of linear programming is to either maximize or minimize the objective function.

Let there be small slack variables $s_i \ \forall i \in [1, 2...n]$, then we try to minimize the slacks as the objective function of the LP with the following constraints:

$$\text{Objective function: } min_{w,b,s} \sum_{i=1}^{n} s_i$$
$$\text{Constraints: } s \geq 0,\ y_i(w^T x_i + b) \geq 1 - s_i,\ \forall i \in [1, 2...n]$$

9. **Non-linear Function Approximation using piecewise linear functions:**

   A piecewise linear approximation is one method of constructing a function g(x) that fits a nonlinear objective function f(x) by adding extra binary variables, continuous variables, and constraints to reformulate the original problem. The specific goal is to approximate a single valued function of one variable in terms of a sequence of linear segments. For the function f(x) , defined on the interval [a,b], a piecewise linear approximation will approximate a function g(x) over the same interval. g(x) is to be made up of a sequence of linear segments. Then g(x) is in the form g(x)=m*x + c for every x in [a,b]. So, we can approximate almost any non-linear function with bunch of piecewise linear equations.

10. **Non-linear Transformation Function Approximation using piecewise linear functions:**

    We can find a ϕ such that it separated two classes in higher dimension. Now we can approximate the ϕ using 'Crosstalk on wire' as a 'Physical Reservoir'. We can find different values of V_1, V_2, V_3 and so one, such that they can be used as linear approximators for the given non-linear transformation function. We know that $V_{crosstalk} = m * V + b$. Where 'm' and 'b' can be controlled by physical variation in the system. So, we can use multiple wires $V_1, V_2, ..$ such that combination of these systems can be used to approximate desired ϕ: $V_{crosstalk1} = m_1 * V_1 + b_1$, $V_{crosstalk2} = m_2 * V_2 + b_2$, $V_{crosstalk3} = m_3 * V_3 + b_3$ and so on.

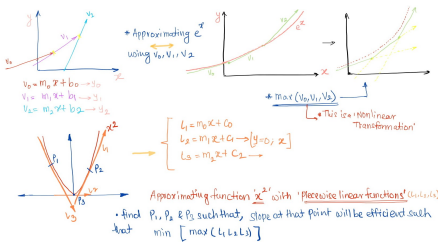

Figure 7: Example of e^x and x^2 approximated by piece wise linear functions

11. **Current Implementation and Experimental Results:**

1. We have used a non-linear 2-D data that can't be separated by a linear classifier. When a SVM or Convex-hull is used to separate these data points, we can see that the data is not separable in original dimension.

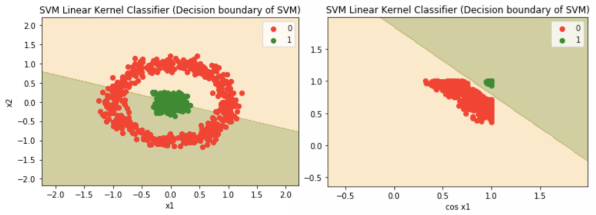

Figure 8: Non-linear dataset can't be learned by a 'Hard SVM' in the original dimension(left) but can be learned in the cosx dimensional space(right)

2. We tried out different kernels to project our data into higher dimension such that we can learn that data linearly in the higher dimension. In consecutive experiments we found '$\cos(x)$' as a good feature-transformation function. Projected data with '$\cos(x)$' looks linearly separable in the given space. We can learn a linear decision boundary in this space that can separate datapoints into 2 separate classes.

3. Wires and Crosstalk: Now, we want this process of mapping of features to be very very fast. We want our feature mapping or transformation function as a piece of hardware that can deal with analog signals and project our data into higher dimensions. We approximate "$\phi = \cos(x)$" with ensembles of linear functions. Each of these linear function can be represented by 'Crosstalk Voltage of Hardware Wire'. **Effectively we are approximating our** $software_\phi$ **with** $Hardware_\phi$

4. We need to define maximum and minimum limit of the transformation function. After that we approximate the part of transformation function using Ensembles of linear functions.

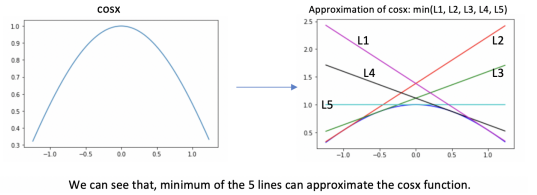

Figure 9: "$\phi = \cos(x)$" can be approximated using 'Ensemble' of linear decision functions

5. Now, these linear functions [L1, L2, L3, L4, L5] can be represented or calculated by 'Crosstalk Voltage on wires' by using crosstalk equations.

$V_{crosstalk/output} = V_{drivercircuits} * (PhysicalConstants) + V_{biasvoltage}$. We can use 5 wires to represent feature transformation functions like COS(x). Each wire has capacity to represent each linear function. Thus, in this way, we can implement hardware feature transformation functions which can drastically increase processing power of machine learning architectures.

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 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.

(d) We did our analysis on The Banknote Authentication Dataset, The Iris Dataset, Sklearn Circles dataset and one custom made dataset. [$datsetsfolder$]

(e) We have also defined a library of transformation functions which is used to transform the data into a different space. [$func\_library.ipynb$]

(f) 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_$LP_banknote_dataset.ipynb$]

(g) 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_$LP_iris_dataset.ipynb$]

(h) 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. [1a_$LP_banknote_dataset_testing_x2_x3.ipynb$]

(i) 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. [1b_$LP_banknote_dataset_all_transformations.ipynb$]

(j) 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.

(k) 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_$feature_transformation_in_higher_d.ipynb$]

(l) Next, we took the Circles dataset from Sklearn and did everything we did for Banknote and the Iris dataset. Fortunately, we found that this non-linear dataset becomes separable in cosx space. [5_$feature_mapping_in_another_space.ipynb$]

(m) Next, we approximated the cosx function as a piecewise function using five linear classifiers. [5_$feature_mapping_in_another_space.ipynb$]

(n) This project can be continued by implementing our hypothesis by using hardware and coming up with a piecewise function of various linear functions.

12. **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.