CS 6350 Machine Learning

Project Intermediate Report

Alok Jadhav (u1265865) Ambuj Arora (u1265867)

12 Nov 2019

1. Introduction

Finding new methods for computing is a necessity for solving unsolvable problems with the traditional/conventional computing methods. Unconventional computing is computing by a wide range of new or unusual methods. It is also known as alternative computing. There has been many attempts at finding new methods that can replace the traditional ones. Like: Optical Computing, Spintronics, Atomtronics, Quantum Computing, Reservoir Computing and so on. New computational methods like 'Quantum Computing' shows promise in solving encryption and computational problems much faster than the current computers. 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.We know that transformation of lower-D data into higher-D can be computationally very costly. These kind of techniques can be used to replace traditional computing methods to find new and efficient way of computing. In future there might be ASICs(Application oriented integrated circuits) for each costly mathematical operation.

2. What is Reservoir Computing?

Reservoir computing is a computational framework suited for temporal/sequential data processing. It is derived from several neural network models, including echo state networks and liquid state machines. 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. The reservoir is fixed and only the readout is trained with a simple method such as linear regression and classification. 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.

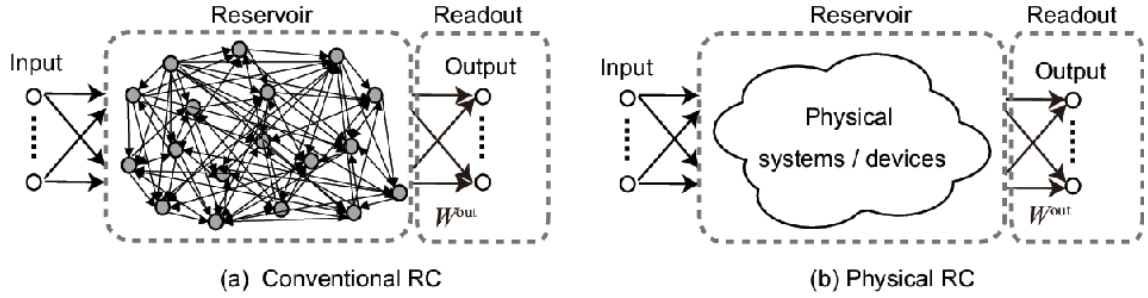

Figure 1: Physical Reservoir: There are 3 layers in above RC architecture. Input, Reservoir, and Readout Layer. Reservoir projects input into higher dimensional space.

In RC, input data are transformed into spatiotemporal patterns in a high-dimensional space by an RNN in the reservoir. Then, a pattern analysis from the spatiotemporal patterns is performed in the readout. The main characteristic of RC is that the input weights (Win) and the weights of the recurrent connections within the reservoir (W) are not trained whereas only the readout weights (W_{out}) are trained with a simple learning algorithm such as linear regression. The role of the reservoir in RC is to nonlinearly transform sequential inputs into a high-dimensional space such that the features of the inputs can be efficiently read out by a simple learning algorithm. Therefore, instead of RNNs, other nonlinear dynamical systems can be used as reservoirs. We can replace the software-based 'Reservoir' with any 'Physical phenomenon' that can project input into high-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. — Cover, T.M., Geometrical and Statistical properties of systems of linear inequalities with applications in pattern recognition, 1965

   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**

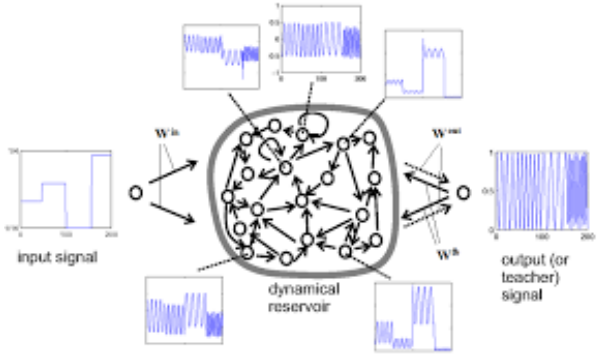

Figure 2: In the figure above the reservoir is shown. Where the reservoir works as a signal generator. Input is a 'Step signal' and output is a 'Sine Wave' with variable frequency.

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.

7. **Cross Talk between wires or strips**

   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. Crosstalk is usually caused by undesired capacitive, inductive, or conductive coupling from one circuit or channel to another. Crosstalk can be induced in circuit-B (child) using the circuit-A (Parent). 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.

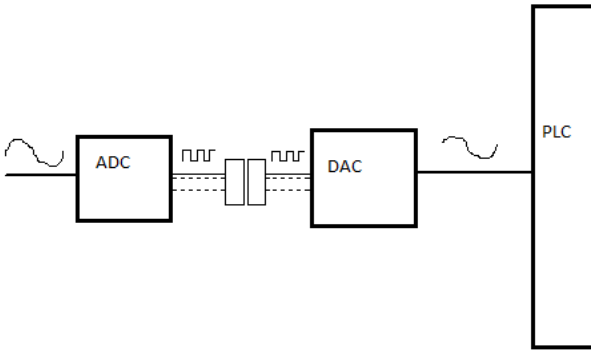

Figure 3: DAC converts incoming Digital signals into Analog signals. These signals then can be passed/transmitted through wires

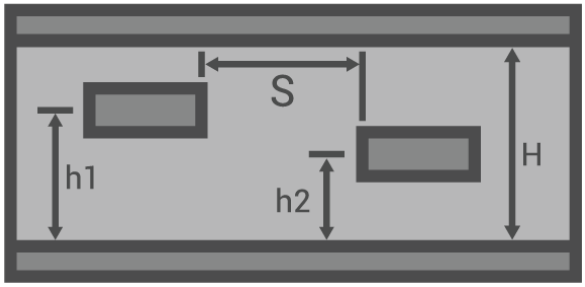

Figure 4: Two conducting lines can have crosstalk between them if one of it is carrying signal

$$T_{RT} = 1.017\sqrt{\varepsilon_{r\cdot 0.475+0.67}} \cdot L \cdot 2$$

Figure 5: $V_{crosstalk}$: Induced voltage on child circuit. V = Input voltage

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

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

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

Figure 6: $V_{crosstalk}$: Induced voltage on child circult. V = Input voltage

$$V_{crosstalk} = V \cdot \frac{1}{1 + \left(\frac{S_{eff}^2}{h_{1eff} \cdot h_{2eff}}\right)} \cdot \frac{T_{RT}}{T_R}$$

Figure 7: $V_{crosstalk}$: Induced voltage on child circuit. V = Input voltage

Equation of Voltage during crosstalk:

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 almost linear relationship between** $V_{crosstalk}$ **and V**

8. **Transformation functions and dataset**

   Transformation function and their application: Feature transformation is simply a function that transforms features from one representation to another.

   When to use Feature Transformation:

   1. Data types are not suitable to be fed into a machine learning algorithm, e.g. text, categories

   2. Data is not separable in the given dimension. Data need to be projected into much higher dimension to be separated.

   3. Feature values may cause problems during the learning process, e.g. data represented in different scales.

   4. We want to reduce the number of features to plot and visualize data, speed up training or improve the accuracy of a specific model.

   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. For different datasets we will try different ϕ (Transformation functions) to see that which one works best for that specific dataset.

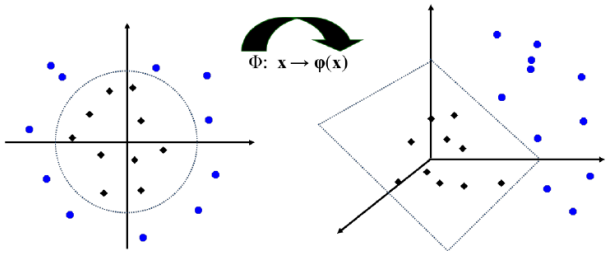

Figure 8: A function ϕ can be used to project low dimensional nonlinear data into higher dimension such that it is linearly separable in that dimension

9. **Methods to prove linear separabiltiy: Convex Hull, Linear Programming, SVM**

   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 several methods:

1. Convex Hull

A set of points is defined to be convex if it contains the line segments connecting each pair of its points. 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. For each choice of coefficients, the resulting convex combination is a point in the convex hull, and the whole convex hull can be formed by choosing coefficients in all possible ways. Expressing this as a single formula, the convex hull is the set:

$$\mathbf{Conv}(S) = \left\{ \sum_{i=1}^{|S|} \alpha_i x_i \;\middle|\; (\forall i : \alpha_i \geq 0) \wedge \sum_{i=1}^{|S|} \alpha_i = 1 \right\}.$$

Figure 9: Equation for Convex Hull

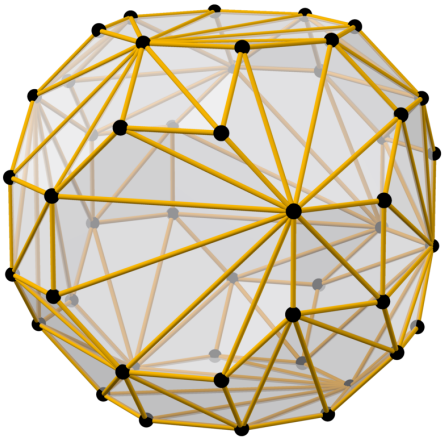

Figure 10: 3-D Convex Hull calculated in SCi-Py

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.

2. Linear Programming

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 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 two sets

$$H = \{H^1, \cdots, H^h\} \subseteq R^d \text{ and } M = \{M^1, \cdots, M^m\} \subseteq R^d$$

The sets are said to be linearly separable for a weight vector w and bias b if $\exists$ w $\in R^n, b \in R$

$$H \subseteq \{x \in R^n : w^T x + b > 0\} \text{ and } M \subseteq \{x \in R^n : w^T x + b \leq 0\}$$

In Linear Programming, H and M are linearly separable if the optimal value of the Linear Program (LP) is 0

By the above definition, for a binary classification data x_i, y_i, with $x \in R^n$ and $y \in \{0, 1\}$, the data is said to be linearly separable if and only if

$$y_i(w^T x_i + b) \geq 0, \forall i \in [1, 2...n]$$

Assuming strict separability, we can always rescale (w , b) and work with

$$y_i(w^T x_i + b) \geq 1, \forall i \in [1, 2...n]$$

Separability constraints are homogeneous, so without loss of generality we can work with the above inequation. If solving this inequation is infeasible, we can formulate the above as a Linear Programming problem in variables w,b and s.

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:

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

3. SVM (Support Vector Machine)

SVM can be used for both classification and regression. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall. 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.

Kernel Trick: 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= $g(W^T x + b)$

Class Labels: y = +, −

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. Computing the (soft-margin) SVM classifier amounts to minimizing an expression of the form:

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i(w \cdot x_i - b)\right)\right] + \lambda\|w\|^2. \qquad (2)$$

Figure 11: SVM soft margin classifier

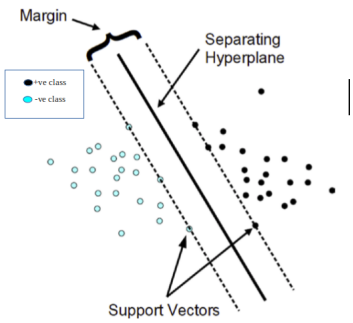

Figure 12: SVM separates two classes as a large margin classifier

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

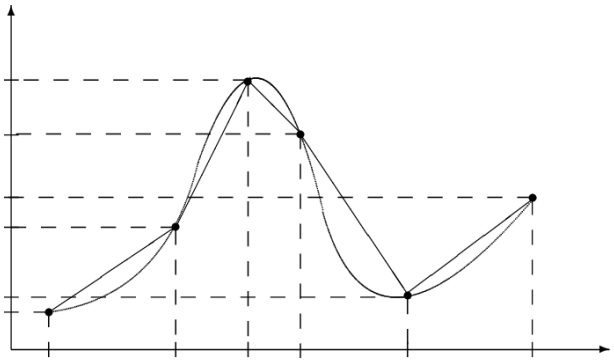

Figure 13: A function can be represented/approximated by piecewise linear functions

A piecewise approximation plays many important roles in many area of mathematics and engineering. 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.

11. **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:

1. $V_{crosstalk1} = m_1 * V_1 + b_1$

2. $V_{crosstalk2} = m_2 * V_2 + b_2$
3. $V_{crosstalk3} = m_3 * V_3 + b_3$

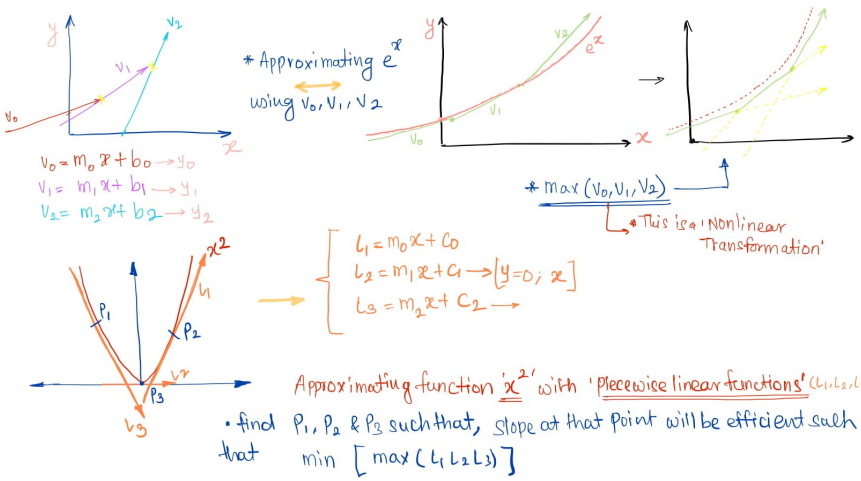

Figure 14: Example of e^x and x^2 approximated by piece wise linear functions

12. **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. [$1a\_LP\_banknote\_dataset\_testing\_x2\_x3.ipynb$]

(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. [$1b\_LP\_banknote\_dataset\_all\_transformations.ipynb$]

(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. [$3\_feature\_transformation\_in\_higher\_d.ipynb$]

13. **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)

14. **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