On Modeling Error Functions as High Dimensional Landscapes for Fast Learning

***Abstract*—We view the error function of a deep neural network as a high-dimensional landscape. We observe that due to the inherent complexity in its algebraic structure, such an error function may conform to general results of the statistics of large systems. To this end we apply some results from Random Matrix Theory to analyse these functions. We model the error function in terms of a Hamiltonian in N-dimensions and derive some theoretical results about its general behaviour. These results are further used to make better initial guesses of weights so as to improve the time-efficiency when we train deep networks.**

I. Introduction

Machine learning is a discipline in which correlations drawn from samples are used in adaptive algorithms to extract critical and relevant information that helps in classification. The interplay between the formulations of learning models in the primal/dual spaces has a great impact in the theoretical analysis and in the practical implementation of the system. Learning can be supervised, unsupervised or even a combination of the two. In face recognition systems for instance, machine learning is typically supervised since it is trained using a sample set of faces. Whereas, in big data analytics, machine learning is unsupervised since there is no *a priori* knowledge of the features/information associated with the data.

In the terminology of machine learning [**?**] classification is considered an instance of supervised learning, i.e. learning where a training set of correctly identified observations is available (i.e. the training set is labelled). In unsupervised learning, a model is prepared by deducing structures present in the input data (which is either not labelled or no *a priori* labelling is known) and project them as general rules. This could mean identifying a mathematical method/process for data organization that systematically reduces redundancy. The corresponding unsupervised procedure is known as clustering, and involves grouping data into categories based on some measure of inherent similarity or distance. In semi-supervised learning, input data is a mixture of labelled and unlabelled samples. There is a desired prediction problem but the model must learn the structures to organize the data as well as make predictions.

Neural networks (NNs) are one of the major developments in the field of machine learning. The popularity of NNs is due to its substantial learning capacity and adaptability to various application domains. The building blocks of a NN are called neurons that act as processing nodes. Such nodes arranged in layers make the network. The layers are called input, hidden or output layer based on their function and visibility to the programmer. These layers are interconnected by synaptic links that have associated synaptic weights. A pictorial representation of a neuron and a feed forward neural network is shown in Fig. 1(a) and Fig. 1(b) respectively. Training a NN refers to tuning the synaptic weights to implement a given function. The function computed by each neuron is

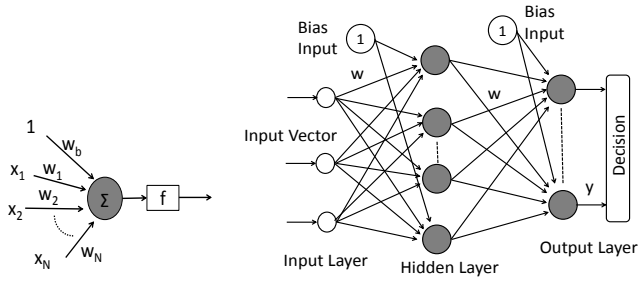
$$y = f\left(w_b + \sum_{i=1}^{N} w_i \times x_i\right) \tag{1}$$

where, N is the dimension of the input sample. x_i and w_i are i^{th} element of the input sample and weight vector respectively. w_b is weight associated with the bias input as shown in Fig. 1(a). f is a differentiable non-linear function. Some of the popular non-linear functions used are $sigmoid$, $tanh$, $ReLU$ etc. During training of this neuron, samples x_{tr} from the training database X_{tr}, each associated with labels y_{tr} are used to train the synaptic weights. This tuning of weights is performed to minimize the error function which is a function of difference between the predicted output and the actual output [**?**] given by

$$E = \frac{1}{N_{samp}} \times \sum_{i=1}^{N_{samp}} \left(y_{tr_i} - f\left(w_b + \sum_{j=1}^{N} w_j \times x_{tr_{i,j}}\right)\right)^2 \tag{2}$$

where $x_{tr_{i,j}}$ is the j^{th} element of i^{th} training sample and N_{samp} is the number of training samples. The error function is a high dimensional landscape, which needs to be explored for its minima.

A single neuron can be trained using standard procedures such as gradient descent [**?**] that updates the weights based on gradient of the error function. When it comes to the training of a multi-layer feed forward NN, the gradient descent involves layer-wise computation of gradients and tuning the weights



(a) A single neuron (b) A three layer feed forward neural network

Fig. 1. A single neuron and a feed forward neural network

accordingly. This method is popularly known as the Back propagation []. To train a multi-layer NN using back propagation, there are two main design parameters to be chosen. First, the learning rate [**?**] which can be visualized as a step size in the search for minima in the error landscape. During training, dynamic update of learning rate has shown to perform better learning in practical examples []. The second, but the most important, design parameter to be chosen is the initial synaptic weights to start the back propagation. The initialized weights have shown to affect the number of iterations required for the convergence along with the classification performance of the trained network [**?**]. Weight initialization methods such Xavier method [1] and NW method [2] are being used in deep learning frameworks like Caffe [**?**] and Matlab Neural Network Toolbox [**?**] for faster and efficient learning.

Deep Neural Networks (DNNs) [**?**] have recently emerged as the area of interest for researchers in the field of machine learning. The strength of DNN lies in the multiple layers of neurons that together are capable of representing a large set of complex functions. Although we see numerous applications of DNN, training DNNs has always been a challenge due to the large number of layers. In addition, these very deep networks have witnessed the problem of vanishing gradients [], that has encouraged the researchers to explore better methods of initializations. A good weight initialization has shown to play a crucial role in achieving better minima with faster learning in DNNs. Therefore, there is a need for better weight initialization methods that will play a major role in training emerging very deep neural networks.

In this paper we explore statistical methods from Random Matrix Theory [**?**](RMT) for large systems, and apply these concepts to explore High Dimensional Landscapes of Error Functions for fast learning. While such methods have been applied to problems in Physics to study complex energy levels of heavy nuclei, financial analytics for stock correlations, communication theory of wireless systems, array signal processing, this is for the first time (to the best of our knowledge) that such a method is being applied to learning systems. The rest of the paper is organized as follows. In section II we give a brief introduction to the RMT and its applicability in learning systems in the context of prior work. We refer to analysis of RMT and results in the literature which are the basis for development of our approach for faster learning. In section III, we describe our approach that applies RMT to the problem of learning in neural networks. Section IV analyses different parameters in RMT to improve the learning ability of the network. In section **??** we report the results to support our theory and we conclude in section V.

II. Some History: Random Matrix Theory and Related Work

Lately Random Matrix Theory (RMT) has been applied effectively in various fields of science and engineering [**?**]. The fact that little knowledge of RMT is sufficient for its application [**?**] has encouraged considerable research work towards exploring applicability of RMT in different application domains.

In 1956 at a conference on Neutron Physics by Time-of-Flight [**?**] held in Gatlinburg, Tennessee, USA Wigner presented his surmise. Wigner claimed that the energy-gaps in a heavy nucleus are akin to the spacings between the eigenvalues of a random matrix. We present some ideas in support of this claim. We proceed as Wigner did in his presentation at the conference. Consider a two cross two random symmetric matrix A defined by

$$A \equiv \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \tag{3}$$

The discriminant of the characteristic equation of A is given by

$$\Delta = \sqrt{(a_{11} - a_{22})^2 + 4a_{12}^2} \tag{4}$$

The discriminant Δ corresponds to the spacing between the two eigenvalues of A. Note that for zero spacing, we would have to have $a_{11} = a_{22}$ and $a_{12} = 0$. This is a rather unlikely event given that A is a random matrix. This result is motivating, because we don't ordinarily want energy levels of physical systems to coincide. The difference in energies is the distance of two points from the origin, with co-ordinates $(a_{11} - a_{22})$ and $2 \cdot a_{12}$. The probability that this distance is S, for small S is proportional to S itself. This is because the radial distance features in the Jacobean of transformation from cartesian to plane-polar co-ordinates. The simplest normalizable distribution that we can conjure up with these constraints is of the form

$$p(\rho, S)dS = \frac{\pi}{2}\rho^2 \exp\Big(-\frac{\pi}{4}\rho^2 S^2\Big) S dS \tag{5}$$

Putting $x = \rho S$ we get

$$p(x)dx = \frac{\pi}{2} x \exp\Big(\frac{\pi}{4}x^2\Big) dx \tag{6}$$

The distribution given by equation 6 is called the Wigner Distribution. The Wigner Surmise has stood the test of time and experiments with increasing degree of accuracy have shown agreement with it. The connection between energies of a heavy nucleus (eigenvalues of a random matrix) and statistics can be made using Random Matrix Theory. We highlight some important results from Random Matrix Theory below.

A. The Semicircle Law

Wigner's Semicircle Law can be stated as follows

Consider an ensemble of $N \times N$ real symmetric matrices with independent identically distributed random variables from a fixed probability distribution $p(x)$ with mean 0, variance 1, and other moments finite. Then as $N \to \infty$

$$\mu_{A,N}(x) = \begin{cases} \frac{2}{\pi}\sqrt{1 - x^2} & \text{if } |x| \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

In other words, the sum of normalized eigenvalues in an interval $[a, b] \subset [-1, 1]$ is found by integrating the semicircle over that interval.

The semicircle law provides the requisite connection between the eigenvalues of a random matrix and the moments of an ensemble of random matrices.

B. Tracy Widom Law

C. The Tracy Widom Law

Problems regarding the motion of a particle in a high-dimensional landscape occur throughout physics in various disciplines such as Spin-glass theory, String theory, the theory of Supercooled liquids etc. The dynamics of a system in an N-dimensional potential can be described by

$$\frac{dy_i}{dx} = -\nabla_i V \tag{8}$$

where $V = V(x_1, x_2, ..., x_n)$ is the functional form of the potential of interest. A high-dimensional landscape is characterized by its stationary points. Stationary points are points on the landscape where a particle moving on it is at equilibrium. At stationary points, the gradient of the potential vanishes. The nature of the stationary points is determined by the laplacian of the potential which is given by the eigenvalues of the Hessian matrix of the system. A matrix element of the Hessian matrix is defined as

$$H_{ij} = \nabla^2_{ij} V \tag{9}$$

The probability of finding a local minima is given by

$$P(\lambda_1 < 0, \lambda_2 < 0, ..., \lambda_n < 0) \tag{10}$$

This is equivalent to finding the probability that the maximal eigenvalue $\lambda_{max} < 0$. The study of the maximal eigenvalue of a random matrix is thus of appreciable interest in disciplines that study high-dimensional landscapes. Tracy and Widom [] showed in 1994 that the distribution of the maximal eigenvalue of an ensemble of random matrices is given by

$$P(\lambda_{max} \leq w, N) = \mathcal{F}_\beta(\sqrt{2}N^{-2/3}(w - \sqrt{2})) \tag{11}$$

where $\mathcal{F}_\beta$ is obtained from the solution to the Painlevé II equation. $\beta = 1$ corresponds to the Gaussian Orthogonal Ensemble (G.O.E.). The G.O.E. will be our ensemble of interest in this paper. Setting $\beta = 1$ the expression for $\mathcal{F}_1$(x) is given by

$$\mathcal{F}_1(x) = \exp\left(-\frac{1}{2}\int_x^\infty ((y-x)q^2(y) + q(y))dy\right) \tag{12}$$

where q is given by

$$\frac{d^2y}{dx^2} = 2q^3(y) + yq(y) \tag{13}$$

Here N is the dimension of the random matrix under consideration. The Tracy Widom Law thus provides us with a powerful analytical tool in the study of minima of a large random landscape: the main subject of this paper.

D. High Dimension Landscape

Fyodorov et. al. [3] state that finding total number of stationary points in a spatial domain of random landscape is difficult and no efficient techniques are available to perform the task. However, it is possible to perform such calculation for Gaussian fields $\mathcal{H}(x)$ which have isotropic covariance structure, i.e., covariance only dependent on the Euclidean distances $|x_1 - x_2|$. The estimation of stationary points in a spatial domain is equivalent to evaluating the mean density of eigenvalues of the Gaussian Orthogonal Ensemble (GOE) of real $N \times N$ random matrices which has a well established closed form expression [].

We refer to concepts in [3] to build a base for our theory elaborated in section III. We consider a random Gaussian landscape

$$\mathcal{H} = \frac{\mu}{2}\sum_{i=1}^{N} n_i^2 + V(n_1, n_2, ..., n_N) \tag{14}$$

where μ is a tuning parameter and V is a random mean-zero Gaussian-distributed field with covariance given by

$$\langle V(\mathbf{m}), V(\mathbf{n})\rangle = Nf\left(\frac{1}{2N}|\mathbf{m} - \mathbf{n}|^2\right) \tag{15}$$

where $f(x)$ is a smooth function decaying at infinity. Comparing $\mathcal{H}(x)$ with the Gaussian Orthogonal Ensemble (G.O.E.) and applying the tools [3] of RMT, one can count the average number of minima $\langle \mathcal{N}_m \rangle$ of $\mathcal{H}$. As discussed in [3], $\langle \mathcal{N}_m \rangle$ is given by

$$\langle \mathcal{N}_m \rangle = \left(\frac{\mu_c}{\mu}\right)^N \frac{2^{(N+3)/2}\Gamma(\frac{N+3}{2})}{\sqrt{\pi}(N+1)N^{N/2}} I_N\left(\frac{\mu}{\mu_c}\right) \tag{16}$$

where $I_N(\mu/\mu_c)$ is given by

$$I_N\left(\frac{\mu}{\mu_c}\right) = \int_{-\infty}^{\infty} e^{\left(\frac{s^2}{2} - \frac{N}{2}\left(s\sqrt{\frac{2}{N}} - \frac{\mu}{\mu_c}\right)^2\right)} \frac{d}{ds}\left(P_{N+1}(\lambda_{max} \leq s)\right) ds \tag{17}$$

Here $\mu_c = \sqrt{f''(0)}$. $P_N(\lambda_{max} \leq s)$ is the probability that the maximal eigenvalue of a standardized G.O.E. matrix M is smaller than s. The Tracy-Widom Law [**?**] gives us the following formula as $N \to \infty$.

$$P_N\left(\frac{\lambda_{max} - \sqrt{2}N}{N^{-1/6}/\sqrt{2}} \leq s\right) \sim \mathcal{F}_1(s) \tag{18}$$

where $\mathcal{F}_1$ is a special solution of the Painlevé II equation. Substituting back into equation 17 we get

$$I_N\left(\frac{\mu}{\mu_c}\right) = \int_{-\infty}^{\infty} e^{h_N(t)} dt \tag{19}$$

where $h_N(t)$ is

$$h_N(t) = \frac{s_t^2}{2} - \frac{N}{2}\left(s_t\sqrt{\frac{2}{N}} - \frac{\mu}{\mu_c}\right)^2 + \ln \mathcal{F}_1^{'}(t) \tag{20}$$

where $s_t = \sqrt{2(N+1)} + t\frac{(N+1)^{-1/6}}{\sqrt{2}}$. Thus as originally shown in [3] expressions 16, 19 and 20 help to arrive at the average number of minima for the energy landscape defined by $\mathcal{H}$.

The explicit formula for $P_N(\lambda_{max} \leq s)$ is

$$P_N(\lambda_{max} \leq s) = \frac{Z_N(s)}{Z_N(\infty)} \tag{21}$$

where Z_N is

$$Z_N(s) = \int_{-\infty}^{s} d\lambda_1 \int_{-\infty}^{s} d\lambda_2 \cdots \int_{-\infty}^{s} d\lambda_N \prod_{i<j} |\lambda_i - \lambda_j| \exp\left(-\lambda_i^2/2\right) \tag{22}$$

as given in [3]. Here λ_i is the i^{th} eigenvalue of $\mathcal{H}$ In equation 17 the limits of the integration are over the entire real line. From equations 21 and 22 we see that this manifests as an integral over all the possible minima for each x_i. Consider having finite limits to the integral in equation 17. We see again from equations 21 and 22 that this would mean that we're searching for minima in the range defined by our new limits. We have thus defined an N-dimensional hypercube inside which we choose to search for minima. Introducing finite limits in equation 19 we get

$$I_N\left(\frac{\mu}{\mu_c}\right)\Big|_a^b = \int_a^b e^{h_N(t)} dt \tag{23}$$

with h_N as defined in equation 20. This gives

$$\langle \mathcal{N}_m \rangle_a^b = \frac{\mu}{\mu_c}^N \frac{2^{(N+3)/2}\Gamma(\frac{N+3}{2})}{\sqrt{\pi}(N+1)N^{N/2}} I_N\left(\frac{\mu}{\mu_c}\right)\Big|_a^b \tag{24}$$

Equation 24 provides us a formula by which we can calculate the mean number of energy minima of the Hamiltonian $\mathcal{H}$ within a hypercube defined by the limits of the integral in the R.H.S.

E. Related Work

In [**?**], a Layer-sequential unit-variance (LSUV) initialization method for weight initialization, in connection with standard stochastic gradient descent (SGD) is proposed that leads to state-of-the-art thin and very deep neural nets. Through experimental validation the authors establish that the proposed initialization leads to learning of very deep nets that produces networks with test accuracy better or equal to standard methods and is comparable to complex schemes such as FitNets [**?**] and Highway [**?**].

Authors in [**?**] propose greedy layer-wise unsupervised training strategy for deep multi-layer neural networks that targets initializing weights in a region near a good local minimum, giving rise to internal distributed representations that are high-level abstractions of the input, resulting in better generalization.

Authors in ' [**?**] draw a parallel between machine learning problems, such as deep networks, and mean field spherical spin glass model in terms of finding the ground state energy of the mean field system's Hamiltonian ($\mathcal{H}$), though in reality the two systems are not mathematically analogous. They hint at the two systems being special cases of a more general phenomenon which is not yet discovered. Through experiments on teacher-student networks with the MNIST dataset, the authors report that both gradient descent and their proposed stochastic gradient descent methods (based on RMT) are equally efficient in identical number of steps.

As discussed so far, RMT provides statistics about the eigenvalue spectrum of an arbitrary random matrix. Quantum theory treats all physical observables as operators. The energy of any physical system corresponds to the result of the action of the Hamiltonian operator $\mathcal{H}$ on the system. Quantum theory provides the formalism by which the Hamiltonian of a system can be realized as a Hermitian matrix. Thus finding the energy of a system boils down to solving the eigenvalue problem of the Hamiltonian matrix. Since the Hamiltonian is a large complicated infinite dimensional matrix it only makes sense to talk statistically. RMT provides us the necessary mathematical tools to do so.

In this paper, we explore the performance of a multi-layered network by carrying out a Layer sequential supervised initialization of weights by exploring the high dimension error landscape using a stochastic method based on RMT, to be followed by a standard stochastic gradient descent (SGD) method for learning. We believe that our approach/solution is indeed based on the more general phenomena of finding the minima on a high dimension error landscape as conjectured by LeCun [ref LeCun].

III. Exploring high dimensional landscapes for fast learning

In this section we model the error function of NN as a random landscape defined in equation 14. To find the minima of this high dimensional landscape, which are the optimal initial weights for gradient descent, we propose a method based on RMT. We compute weights using our RMT theory which are used to initialize the synaptic weights of NN prior to back propagation. We consider an application of Face Recognition(FR) for analysis and justification of our approach. In FR, the neural network is trained to identify input images as one of the categories/classes in the training database. A detailed description of FR is given in [4]. Initially we consider a toy model of a single layer of neurons with N input nodes and N_{class} output nodes, where N is the dimension of input sample and N_{class} is the number of classes/categories in the training database. For simplicity we do not consider the non linear activation function, f, at the output nodes initially. Each link connecting i^{th} input node to j^{th} output node is associated with a synaptic weight $w_{i,j}$. The output of each output node is given by

$$y_j = \sum_{i=1}^{N} w_{i,j} x_i, \qquad j = 1, 2, ...N_{class} \tag{25}$$

During training of this network, samples from the training database are applied at the input nodes. The synaptic weights

are tuned according to the difference between the network output y and the desired output y^d.

From equation 25, error at the output node j is given by,

$$E_j = \left(y_j^d - \sum_{i=1}^{N} w_{i,j} x_i \right)^2 \tag{26}$$

Suppose we write the desired output y^d in the form

$$y_j^d = \sum_{i=1}^{N} u_{i,j} \tag{27}$$

where $u_{i,j}$ is a real number. Substituting this value of y_j^d from equation 27 back into 26 we get

$$E_j = \left(\sum_{i=1}^{N} u_{i,j} \right)^2 - 2 \left(\sum_{i=1}^{N} u_{i,j} \right) \left(\sum_{i=1}^{N} w_{i,j} x_i \right) + \left(\sum_{i=1}^{N} w_{i,j} x_i \right)^2 \tag{28}$$

Expanding the first term in the R.H.S, we get

$$E_j = \sum_{i=1}^{N} u_{i,j}^2 + \sum_{i=1}^{N} \sum_{\substack{k=1 \\ i \neq k}}^{N} (u_{i,j} u_{k,j}) + \cdots - 2 \left(\sum_{i=1}^{N} u_i \right) \left(\sum_{i=1}^{N} w_{i,j} x_i \right) + \left(\sum_{i=1}^{N} w_{i,j} x_i \right)^2 \tag{29}$$

The training of N_{class} output nodes is independent of each other. As our target is to arrive at a favourable initial synaptic weights for gradient descent, we believe that these weights can be computed by training each output node with samples from the corresponding class.

Consider the following interpretation. Let x^a represent a sample that belongs to the training dataset of our toy network. Let it belong to the a^{th} class of samples and therefore let the sum

$$\sum_{i=1}^{N} w_{i,j} x_i^a = \sum_{i=1}^{N} u_{i,j}^a \tag{30}$$

which by definition is equal to y_j^a. Similarly for every sample $\mathbf{x^k}$ in the training dataset we can associate with it a vector $\mathbf{u^k}$ that maps it to the correct output node.

The first term in the R.H.S of equation 29 is quadratic in $\mathbf{u}$. The other terms in the R.H.S are all complicated terms in $\mathbf{x}$ and $\mathbf{u}$. Observe that we can always define a transformation between x_i and $u_{i,j}$ in N-dimensions. This is true because both coordinates represent different ways of looking at the same landscape. x_is are defined over a 'position-field', whereas $\mathbf{u}$s are defined over the 'transform-field'. Thus the error function is the sum of a quadratic term in $\mathbf{u}$ and a complicated N-dimensional function of variables $\mathbf{u}$.

Since we are already in N-dimensions and are dealing with an error function that has complicated terms, consider applying the following trick. We replace the complicated part of the error function,E_j by a random function $V(u_{1,j}, u_{2,j}, \cdots, u_{N,j})$ in N-dimensions where $u_{i,j}$ are mean-zero Gaussian random variables. This assumption allows us to apply the tools of Random Matrix Theory to analyze the error function as we will discuss in section III-A

The above assumption begs the following question: How do we know that are coordinates(images) are indeed mean-zero Gaussian variables? We look to statistics for an answer to this query. Each u_j represents a pixel of a face image in the dataset. Given a large database, by the Central Limit Theorem (C.L.T.) each pixel takes Gaussian values. This answers the Gaussian part, now let's consider the mean-zero part. Each pixel is associated with Gaussian values and a mean. Consider the mean of all the pixels of an image. High resolution images have lots of pixels. Therefore the mean of each pixel must also conform to a Gaussian by the C.L.T. If we redefine the origin of coordinates to this number then provided that the means have low variance, we have succeeded in producing N independent mean-zero Gaussian random variables. Observe that we have introduced a constraint by way of requiring that the means have a low variance. We can ensure this constraint by considering only similar images, i.e., images belonging to same class, because similar images by definition don't 'vary' much from each other. We would like to impress that such a constraint doesn't effect the generality of our results. This is because we choose to train each output node of the neural network independently.

A. Finding Minima on a High Dimensional error Landscape

To find the minima of the error function in equation 29, we look for analogy between the error function and N dimensional GOE in equation 14. By representing terms other than the first term in RHS of equation 14 as a random function $V(x)$, we write it as an N-dimensional Hamiltonian of the form

$$\mathcal{H} = \frac{\mu}{2} \sum_{i=1}^{N} u_i^2 + V(u_1, u_2, ..., u_N) \tag{31}$$

where μ is a tuning parameter and V is a N-dimensional random landscape with covariance given by equation 15. We see that $V(u_1, u_2, ..., u_N)$ is indeed a very complex function which can be approximated to be a mean-zero Gaussian-distributed field. We assume that the $\mathbf{u}$ follows the same distribution as that of $\mathbf{x}$ that helps us in finding the minima of the error landscape. As the training samples are face images which are rotation and translation invariant, we can assume the covariance matrix also to be isotropic. Having confirmed the basic requirements, we can apply the RMT method described in section II.

In our approach, we compute weight vectors connecting each output node independently by using samples of corresponding class. Thus, we compute N_{class} N-dimensional landscape minima to train our toy network. From equation 20, 23 and 24 we conclude that the mean number of energy minima of $\mathcal{H}$ is a function of h_N.

To train an output node, the required solution is the value of $\mathbf{u}$ that will have maximum mean number of minima indicating

the presence of optimum point. Looking at equation 20 we find that h_N is a function of $\frac{\mu}{\mu_c}$, where μ_c is equal to $\sqrt{f''(0)}$, and f() is the covariance function. The last term in equation 20 is logarithm of derivative of solution of Painlevé II equation that is observed to be negligible when compared to the difference of first two terms for large N. Therefore, in equation 20, μ_c behaves as a critical point in every dimension representing the point where we can find majority of minima. Gradient is computed on the covariance matrix two times and square root of the diagonal elements are assigned to μs in respective dimensions. In equations 20, 23 and 24, it is observed that to find average number of minima in a spatial domain, h_N is integrated over the range in each dimension. A randomly selected vector from the hypercube that shows a high density of minima will give us the solution vector $\mathbf{u}$. This implies that to find our initialization weights, it would suffice to find in each dimension $\mathbf{u}$ for which h_N becomes maximum. Fig. 2 shows a three dimensional case where a cube is constructed based on the max of h_N in each dimension. Finally the computed $\mathbf{u}$ is scaled down by the range of input pixels to get the synaptic weights.

In the section IV, we analyse our method in the implementation perspective and select the parameters to achieve better results. In addition, we extend this work to multi-layer weight initialization which has shown promising results. We report results supporting our RMT based weight initialization method.

IV. Analysis and Results

In section III-A it is clear that μ_c is the critical parameter which is responsible for variations in h_N for a fixed N. A three dimensional plot in fig. 3 shows variation of h_N with different values of $\mathbf{u}$ and $\frac{\mu}{\mu_c}$. We get components of $\mathbf{u}$ in different dimensions where h_N becomes maximum based on the $\frac{\mu}{\mu_c}$ ratio.

A. Pre-processing the training data

We pre-process the samples by reducing the dimension and also mean centring them. We define a GOE of input obtained by performing Principle Component Analysis (PCA) [5] on the test samples of faces. The eigenspace defined for the faces not only serves to reduce the dimension of the inputs, but also helps defining an error landscape over a GOE of principal components/axes. It is also possible to carry out Feature Extraction (PCA) following the Random Matrix Theory approach as reported in [PCA using RMT] for dimension reduction. Resultant feature vectors are mean centred and divided by their standard deviation to make it mean-zero and unit variance.

B. Selection of μ

Our aim is to get maximum variations in the value of computed $\mathbf{u}$ so that it covers the entire range of $\mathbf{u}$ for effective weight computation. Fig. 4 shows a plot of indices of $\mathbf{u}$ for maximum h_N with varying $\frac{\mu}{\mu_c}$. We find that the plot saturates beyond a point after which we will have the same computed values of $\mathbf{u}$. To address this problem we select the tuning parameter μ such that the ratio $\frac{\mu}{\mu_c}$ falls in the required range.

C. Multi-layer weight computation

Until now we have established our theory for our toy model of network which is a single layer of neurons. When it comes to training a network for a large face database with complex features, single layer of neurons are not sufficient to achieve the required classification performance. To make our theory applicable to these conditions, we extend our theory to multi-layer networks which requires computation of more than one set of weights. In multi-layer neural networks, weights of each neurons in hidden layers try to identify a pattern in the input samples that is going to be used collectively for classification

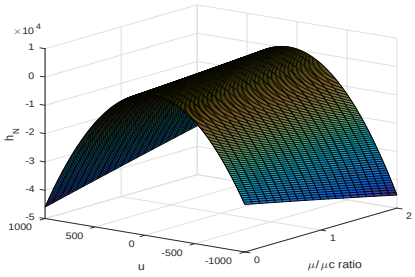


Fig. 3. A 3D plot showing variation of h_N with $\frac{\mu}{\mu_c}$ ratio and $\mathbf{u}$ for N equal to 256

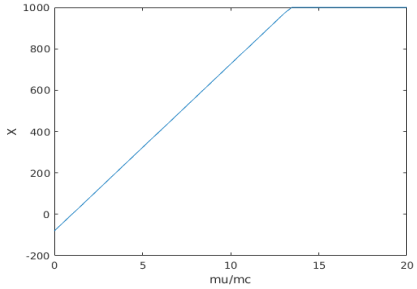


Fig. 4. saturation of $\mathbf{u}$ for high $\frac{\mu}{\mu_c}$

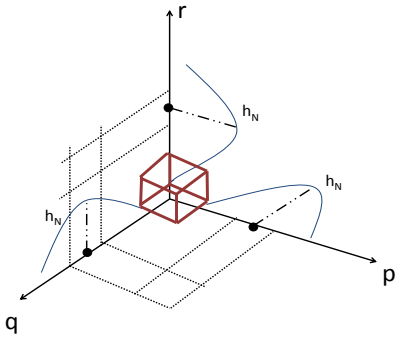


Fig. 2. Construction of a cube based on maximum point of h_N in a three dimensional case.

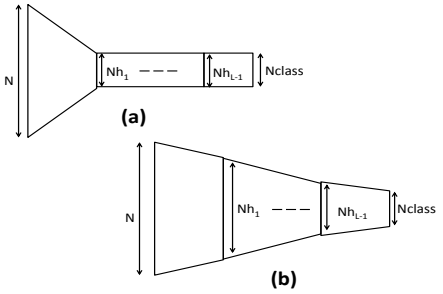
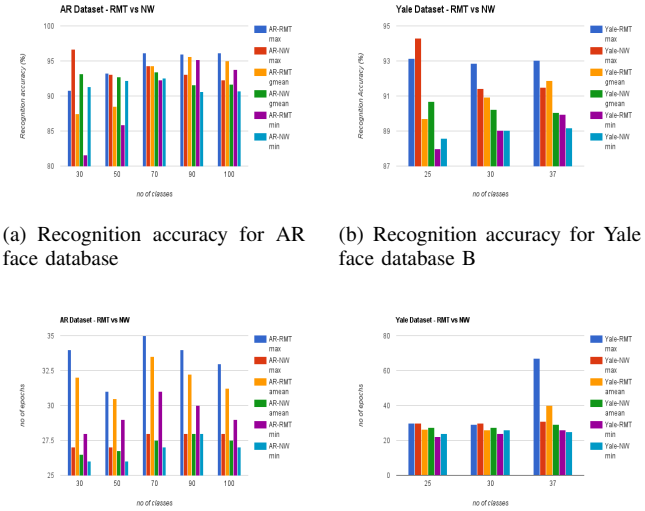


Fig. 5. Multi-layer networks with (a) equal and (b)unequal number of neurons in layers

in subsequent layers. During the back propagation, the weights tune themselves in identifying a pattern in the input samples. This behaviour of hidden nodes help us in extending our theory to multi-networks. Here, similar to our toy model, we do not use bias inputs in all the layers.

In multi-layer networks, application of our method is not straight forward. Our method relies on class-wise computation of weights, that needs equal number of nodes in each hidden layer and the output layer as shown in Fig. 5. Let N and Nh_i are the number of nodes in the input and i^{th} hidden node respectively. Initially we apply our method on first layer of neurons and compute a weight matrix of dimension $N \times Nh_1$. This is analogous to training each first layer hidden node with samples from respective class. This may look like an unorthodox way of training, but we believe that this will help to achieve better weight initialization. Once the weight matrix for the first layer is computed, we compute output of the first hidden layer by multiplying with the input sample set $N_{samp} \times N$ and a sigmoid operation. Although our theory does not include a sigmoid unit at the neuron outputs, we have observed experimentally that the computed weights performed well with sigmoid units included in the network. The computed outputs of dimension $N_{samp} \times Nh_1$ from first hidden layer act as inputs for the second layer with associated labels. Similarly we compute synaptic weights for rest of the hidden layers and the output layer. Previously in our toy model of NN, synaptic weights of a neuron were computed considering only samples of respective class. However, our method when extended to a multi-layer NN, brings this indirect dependency which will improve the classification performance. e.g., first node in the second layer gets inputs from all the nodes in the previous layer and thus, is dependent on the samples used under each those nodes in computing the weights. The related results and analysis are reported in section **??**.

The multi-layer network described above has a limitation that it can only have a number of nodes in each hidden and output layer equal to N_{class}. In practical networks this is not a valid scenario, as the number of nodes in each layer differs based on the training database and available resources. In addition, for inputs with very large dimensions and small number of classes, the network cannot represent the classification function efficiently. We address this issue by using our



(a) Recognition accuracy for AR face database

(b) Recognition accuracy for Yale face database B

(c) Number of epochs for convergence for AR face database

(d) Number of epochs for convergence for Yale face database B

Fig. 6. Scalability of REFRESH with N_{SURE}; N_{reg}=16, N_{pc} = 32

RMT method with a clustering algorithm introduced at each layer. We use k-means clustering which is an unsupervised clustering algorithm for our experiments. The samples are clustered by the algorithm based on their similarity and the target number of clusters. To apply our method on this set-up, we consider each cluster as a single class. We believe that it is a valid assumption as the samples in a cluster are similar to each other . In addition, the hidden similar patterns in samples from different classes are explicitly brought together under a neuron. Fig. 5(b) shows a NN with varying number of nodes in each layer.

In section **??** we report experimental results of our RMT based method of weight initialization. We also compare our method with other initialization methods being used in neural network frameworks.

********** Till here.

V. Conclusion

References

[1] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS10). Society for Artificial Intelligence and Statistics*, 2010.

[2] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing," in *Initial Values of the Adaptive Weights, International Joint Conference of Neural Networks*, 1990, pp. 21–26.

[3] Y. V. Fyodorov and C. Nadal, "Critical behavior of the number of minima of a random landscape at the glass transition point and the tracy-widom distribution," *Phys. Rev. Lett.*, vol. 109, p. 167203, Oct 2012. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevLett.109.167203

[4] G. Mahale, H. Mahale, S. Nandy, and N. Ranjani, "Refresh: Redefine for face recognition using sure homogeneous cores," *IEEE Transaction on Parallel and Distributed Systems*, 2016.

[5] M. Turk and A. Pentland, "Eigenfaces for recognition," *J. Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, jan 1991. [Online]. Available: http://dx.doi.org/10.1162/jocn.1991.3.1.71