Universität Hamburg
Department Informatik
Knowledge Technology, WTM

Optimizing Neural Networks

Seminar Paper

Knowledge Processing with Neural Networks

Leena Chennuru Vankadara
Matr.Nr. 6641411
4chennur@informatik.uni-hamburg.de

11.06.2015

Abstract

In this paper we qualitatively and quantitatively analyze the loss surfaces of Neural networks using Random matrix theory and replica theory. Interesting characteristics such as proliferation of saddle points in higher energy layers has been observed through the analysis. Various optimization algorithms are evaluated for their ability to deal with such structures. The problem of overfitting is then analyzed in the perspective of dynamical system theory. It is shown that proper initialization alone can efficiently deal with the problem of achieving depth independent learning rates as well as overfitting.

Contents

1 Introduction

A great deal of research in neural networks have been focused on creating deep architectures. Deep architectures promised greater expressive power than shallow architectures like kernel methods. Kernel methods, for instance, performed incredibly well on many classification problems but they like any other shallow architecture become less reliable when applied on highly varying functions and those that depend on large number of underlying factors. Image analysis, speech recognition, language processing are some areas where the conditional probability of the output given an observation is a highly varying function which depends on a great deal of underlying factors. General geometric intuition deceives us when working in such high dimensional spaces and dealing with this level of abstractness becomes a highly intricate task. This is often referred to as the curse of dimensionality in the literature. However, deep architectures with their great expressive power by means of large number of latent variables are capable of creating such models.

Recent models of neural networks broadly follow a generic framework where the low level layers act as feature extractors and more abstract concepts are learned by the higher levels in the hierarchy by forming conglomerates of lower level features. While such networks deal with the curse of dimensionality in a very elegant way even with a very less number of parameters there is certainly a trade off, as training such deep networks is considered to be very difficult. [13] presents a set of hypothesis to explain the difficulties deep networks face during learning such as saturation of the activation function, bad initialization of weights leading to convergence onto points with high value of the loss function, decrease in the values of gradient when propagated back across layers. Such problems in deep learning are usually identified by obtaining empirical evidence to validate a hypothesis. Several optimization techniques such as using anti symmetric activation functions, techniques to avoid overfitting to create ensembles by model averaging such as dropout, using unsupervised pre-training to initialize the parameters of the network in what is referred to as a good basin have been prevalent in the literature to avoid these problems. However, neural networks are complex intricate dynamical systems which exhibit all the behavior as shown by dynamical systems like long plateaus, sudden transitions, bifurcations, convergence into chaotic regime when trained poorly. In this paper, we think the best way to deal with such level of intricacy is to address the more fundamental issue of lack of a structured theory of learning in these networks. Even simple dynamics of stochastic gradient descent of a small multi-layer feed forward network show highly complex dynamics. Establishing theoretical bounds and providing guarantees by building the theory of learning in neural networks would heavily narrow down the parameter space in which the solutions lie and would also provide a greater intuition to better understand the qualitative nature of objective functions optimized by the neural networks. In this paper certain characteristic features of the surface generated by a commonly used loss function, like the squared loss, which are of interest to us are identified. To be more specific, critical points of such a surface and the curvature

information at the critical points provided by the Hessian is analyzed. This analysis enables a more faithful view of problems that occur in traversing this surface using iterative first or second order methods.

2 Optimization in Neural Networks

In a supervised setting neural networks essentially fit models which map the input data to their corresponding class labels. That is given a set of training points $\{x_i, y_i\}_{i=1,..,n}$ where $x_i \in \mathbb{R}^N$ are the input points derived from an N dimensional input space and $y_i \in \mathbb{R}$ are the corresponding class labels, a function that maps the input data to output points with the least possible error is desired. This function has to be derived in a minimal time period. Also the mapping should be able to map the input points which have not been presented yet to the training model to their appropriate labels. Thus optimization in neural networks can be viewed in the light of three broad objectives. The first objective is to avoid underfitting. Underfitting occurs when the model or the function is not effective to capture the variability in the input data. Underfitting is characterized by large training error. Neural networks are typically trained using first or second order gradient based techniques. As finding exact analytical solutions for loss functions defined on neural networks is highly non trivial, iterative methods are used to achieve function approximations numerically. Hence minimizing the amount of time required to train the network to achieve an acceptable error rate also becomes an important objective. However, the most important objective of a neural network is not a mapping that just minimizes the training error and the training time but to have good generalization abilities. If the model is highly variable thus allowing it to capture noisy properties of the data then the model is said to overfit to the training set as such a model performs incredibly well on the training set but would have very less generalization capability. Overfitting is characterized by a large test error and a small training error. Given a loss function that estimates the strength of a mapping, For obvious reasons, Overfitting is the most non trivial objective of three discussed above.

Given a loss/objective function, the problem of underfitting can be solved by finding the parameters of the loss function which minimizes the value of the loss function. In other words, the problem of underfitting is essentially solved by finding the global minima or a good local minima(characterized by an acceptable loss) of the loss function. For a convex function this is a trivial problem. However typical loss functions are highly non convex and are known to be riddled with a large number of local minima and saddle points. To solve this problem, we first analyze loss surfaces generated by typical loss functions to identify interesting characteristics. Recent analysis of critical points of Gaussian random fields[7],[12],[18] and spin glasses[3] using Replica Theory and Random matrix theory have enabled a good insight into loss surfaces of a multi layered neural network[8][22]. It is to this analysis we first resort to.

3 Proliferation of saddle points

The number of critical points of a Gaussian Random field defined on a N-dimensional Euclidean space($N >> 1$) for a fixed energy and a fixed number of negative eigenvalues has been computed in [7]. More specifically, the distribution of the critical points as a function of the index and energy is computed. Index (denoted here by α) is defined as the number of negative eigenvalues of the Hessian at the critical point. The results in [7] show that for a finite volume, which implies the existence of a global minima(ϵ_{min}) and a global maxima(ϵ_{max}), the number of critical points are exponentially large but for most of the critical points, the energy of a random function in a high dimensional space is a monotonically increasing function of the index. This means in the $\epsilon - \alpha$ plane, the probability of finding a critical point which does not lie on this monotonically increasing curve is extremely low. This indicates that the number of critical points which have index either 0 or 1 is of measure zero. This implies that for a high dimensional random Gaussian function, local minima are very unlikely and saddle points are more likely. [9]. More importantly, the existence of such a curve in which ϵ varies from ϵ_{min} to ϵ_{max} as α ranges from 0 to 1 indicates that all the critical points which are likely to be either local minima or global minima indeed have very similar energy values. This is a really interesting property observed on Gaussian functions in high dimensional spaces.

To emphasize the relevance of higher dimensions we resort to random matrix theory to provide a better intuition. For a random symmetric matrix, with no external constraints, both the negative and the positive values occur with equal probability and hence the distribution is symmetric about the origin. In particular the distribution of eigenvalues of a random symmetric matrix takes the form of Wigner's semicircular distribution[33]. For a random error function [7] also computed the distribution of the eigenvalues of the Hessian at critical points of a given error value. The distribution is dependent on the value of the error and [7] shows that the shape of the eigenvalue distribution remains the same but the mean and hence the mode of the distribution is shifted depending on the error value. For the least error value the distribution is completely shifted to the right and for the highest error value the distribution is completely shifted to the left. This result is consistent with the previous result which says that the the energy of the local minima can not be very distinct from that of the energy of the global minima. One more important thing to observe here would be that for many intermediate error values several eigenvalues are distributed around zero. This implies that there are a lot of degenerate saddle points associated with plateau regions for intermediate error values.

If we think of a function in one dimension, a typical function drawn from a single draw of a Gaussian process is much more likely to have a local minima or a maxima rather than a saddle point. This intuition can not be translated into a higher dimensional space. This difference can be better understood by an argument presented in[22]. For a random function in one dimension, the Hessian is a single valued random variable and under unconstrained conditions the variable is most likely to assume a positive or a negative value with extremely high probability.

The probability of the variable to take a 0 is extremely low and the associated event has an occurrence of measure zero[22]. This implies that saddle points are extremely unlikely to occur and maxima or minima occur very often. However in N dimensions, the probability of all eigenvalues to be positive(or negative) is $O(e^{-N})$. These results indicate two major phenomenon that occur in random error functions of high dimensions. The first is that there is a proliferation of saddle points in higher error levels. The second is that the error value of most of the critical points which are local minima is extremely close to that of that of global minima. This is against what is typically believed to happen in loss surfaces of neural networks. It is typically believed that the loss surface of a neural network is highly non convex and is riddled with a large number of local minima. Dealing with saddle points is a much more trivial problem compared to optimizing in the presence of local minima at higher error values.

It is however possible that the loss surface of a typical neural network is somehow not similar to a typical error function and hence the analysis of a typical error function does not apply here. Hence we look for physical systems which have equivalence to these deep networks. For this analysis we look at physical systems which are usually modeled by Gaussian random fields.

3.1 Physical systems and deep networks

Gaussian random fields(in higher dimensions) or Gaussian processes(in one dimension) are typically used to model systems with a large number of independent variables. According to Central limit theorem, Gaussian random fields spontaneously occur in these systems. String theory is a physical system which has been recently suggested in an anthropic view to have a highly complex landscape riddled with a large number of possible minima each of them possibly indicating a different universe[30]. Gaussian Random fields are suggested as a means to understand this landscape [7]. There are some well established physical systems such as spin glasses and protein folding which have been modeled using Gaussian random fields. [8] establishes a direct equivalence, under certain assumptions, between the loss surface of a fully connected feed-forward deep neural network with a single output and rectified linear units and the Hamiltonian of the H-spin spherical spin glass model. [20] establishes an exact mapping between the Variational renormalization group and deep learning architectures based on Restricted Boltzmann Machines. [2] has proven that the dynamics of the deep network in the long run is governed by the Hamiltonian of the infinite range Ising spin glass models. [8] provides several theoretical results that describe many interesting characteristics of the loss surface of a multi layer neural network based on results worked out in including those discussed above. We resort to this analysis to analyze the surface of a loss function in a quantitative fashion.

3.2 Spin glasses and Neural networks

The Ising spin glass model is a prototypical model for physical systems with large number of interacting individual components. Specifically, this model is used to describe the statistical mechanics of magnetic spins of interacting particles in a lattice of finite volume interacting with each other via an interacting potential. The energy of the spin configuration is referred to as the Hamiltonian of the model. Under the assumptions of variable independence, redundancy in network parameterization and uniformity, [8] shows that the loss function of a neural networks has the form of a Gaussian random field on a sphere which is equivalent to the Hamiltonian of the H-spin spherical spin glass models. [3] has presented several results to explain the complexity of the spherical spin glass models. [8] uses these results to explain the qualitative nature of the loss surface of a neural network. Some of the most important results that are of relevance to us are presented here.

- For a network with large number of parameters and a large number of paths(approaching infinity) through the network, there exists a ground state,$-E_0$ of energy(error value) and it is highly improbable to find any critical point below this level.

- For the same network, there also exists an energy threshold$-E_\infty$ which is referred to as the energy barrier and in the limit of the number of paths through the network approaching infinity any critical point of a fixed index lies below this energy barrier and above the ground state with exceedingly huge probability. Any critical point that lies above this barrier is a high index saddle point with an overwhelming probability.

- Finding a critical value of a fixed index k below the energy level $E_k \in [-E_0, -E_\infty]$ is highly improbable and the sequence $E_k, k = 1, .., N$ is strictly decreasing and in the limit of k tending to infinity converges to the value of the energy threshold $-E_\infty$ indicating a layered structure of critical points in the band of energy values. This is illustrated in the figure 1.

- The ratio between the number of local minima and saddle points is exponentially increasing with the number of paths and hence the probability of finding a saddle point but not a local minima decreases to 0 with the number of paths.

- The amount of time it takes to search for a global minima after reaching the energy barrier is exponentially long.

These results are in consistent with the results on random error functions derived in [7] and [12]. Furthermore they also suggest that there is a layered structure of critical points of fixed indices ranging from energy level $-E_0$ to $-E_\infty$. The analysis shows that there is a band of energy values above which an optimization algorithm can find a saddle point with a probability approaching 1 with size of the

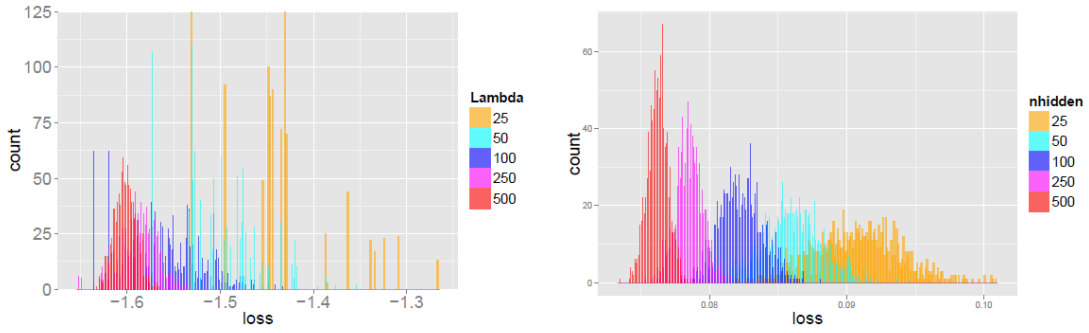

Figure 1: Distributions of scaled losses for the spin glasses(left) and neural network(right) experiments [8]

network and hence any algorithm which can effectively deal with negative curvatures should be able to reach the energy barrier. This barrier is often referred to as the floor [14]. Hence the first obstruction for an optimization algorithm based on gradient based techniques traversing on the loss surface would occur at this energy barrier which is riddled with exponentially large number of local minima. One of the results proved for a random error function according to which the band of the energy values $[-E_0, -E_\infty]$ is very small is not presented as a result in [8]. Also, since the assumption of variable independence is pretty strong and since we know that it is not true since the neural network has huge number of dependencies between the variables we resort to empirical results for further evidence.

Experimental results [27] performed on the Hamiltonian of the spherical spin glass model with mean field assumption has demonstrated the existence of a floor below which a large number of local minima are present. Figure 2 shows the cumulative number of critical points of index k = 0 and index k = 1 in logarithmic scale. $\Theta(u)$ denotes the cumulative number of critical points that lie below the energy value Nu where N is the dimension of the system. The vertical line shows the floor value where the values of $\Theta_0(u)$ and $\Theta_1(u)$ agree. The global minimum of the Hamiltonian is bound by $-NE_0$. The experiments show that the energy of the ground state and floor is calculated to be $-N1.657$ and $-N1.633$ [27]. As it can be seen the floor and the ground state have exceedingly similar energy levels as the band of energy values between the ground state corresponding to the global minima and the floor is very small.

The following figures that show the histograms of normalized energy values also show striking difference between low and high dimensional systems. The figure demonstrates that in low dimensions the band of energy values between the floor and the ground state is very huge in contrast to high dimensional systems.

These results have demonstrated a very interesting phenomenon. For small sized networks local minima pose a difficult optimization challenge. The problem of non convex optimization is still an open question. However as the network size increases, the number of local minima are concentrated around the floor whose energy value is not very dissimilar from that of the energy of the global minima.

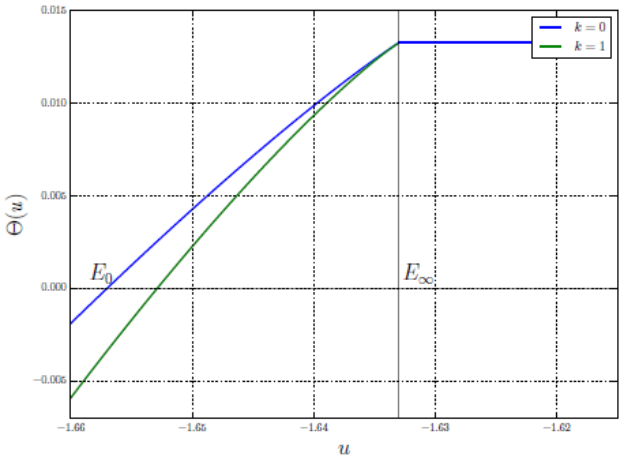

Figure 2: Plot of $\Theta_0(u)$ (upper curve for local minima) and $\Theta_1(u)$ (lower curve for saddle points of index 1)

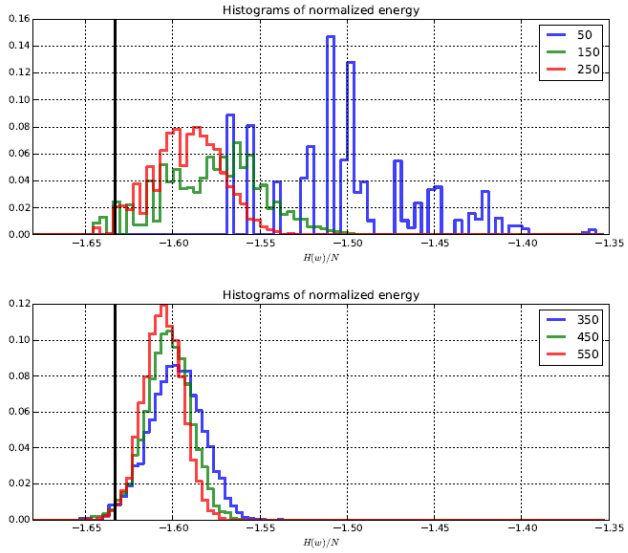

Figure 3: Histograms of normalized energy varied with the number of dimensions indicated by different colors. The black lines indicate the value of the floor determined theoretically as N approaches infinity

Under the guarantee that the floor of the loss surface is very close to that of the ground state, optimization algorithms only have to deal with saddle points which is a much more easier problem to solve.

We first review some first and second order algorithms which have been prevalent in literature and have been typically used to train neural networks. We analyze the ability of these algorithms to deal with saddle points and the $O(time)$ it takes to reach the floor. To recall, the problem of reaching a local minima of a low

energy value is that of avoiding the problem of underfitting.

4 Gradient based optimization techniques

Gradient based techniques use first order gradient information to generate an iterative process which converges to the local minima in a finite time. Gradient Descent, Gradient descent with momentum, Nestrov's accelerated gradient, Method of conjugate gradients without preconditioning, Natural gradient descent[1] are a few of them.

4.1 Gradient Descent

This algorithm is one of the simplest, intuitive and the basis for most first order algorithms used in unconstrained optimization. The update of the parameter vector Θ at a step k is given by equation 1

$$\Theta_{k+1} = \Theta_k - \alpha_k \nabla f_k \tag{1}$$

α_k is the learning rate which determines the step size of the update in the direction opposite to that of the gradient. As we have seen in the analysis on proliferation of saddle points in section 3, local minima are not major hurdles but saddle points are. The gradient of the loss function becomes zero both at local minima as well as saddle points. Saddle points are unstable fixed points so theoretically speaking, gradient descent is extremely unlikely to land up in one of these points. Hence the main issue of gradient descent is not the direction but the step size or the learning rate (α_k). If it is chosen to be a constant then it either slows down too much around plateau regions associated with saddle points which leads to convergence of the algorithm to an apparent local minima or oscillates too much in regions of high curvature which leads to an in-existent convergence.

Line search or line minimization is an algorithm which would solve for this problem. A line search picks a search or descent direction and then searches for minima along the line. The ideal choice of the step would be the value of α that is a global minimizer the function ϕ as given in equation 2

$$\phi(\alpha) = f(\Theta_k + \alpha p_k) \tag{2}$$

This is however very expensive to compute at every gradient direction and there is always a trade off between between the computational complexity and the step size that can be chosen by a means of approximation to the solution of this objective function.

4.2 Momentum

Momentum, also known as classical momentum[25] is a very successful optimization technique which retains information of the persistent directions which cause

a reduction in the value of objective function and provides momentum in those direction by iteratively adding speed in those directions. Formally its given by:

$$\nu_{k+1} = \mu\nu_k - \varepsilon\nabla f(\Theta_k) \tag{3}$$

$$\Theta_{k+1} = \Theta_k + \nu_{k+1} \tag{4}$$

where ε is the learning rate, $\mu \in [0, 1]$ is the momentum coefficient and $\nabla f(\Theta_k)$ is the gradient of the objective function at Θ_k.

Momentum, by definition accelerates the velocity vector in persistent directions. Since in plateau regions, the gradient vector points in the same direction over several iterations, momentum accelerates it in those directions. This is a really simple and effective technique to effectively deal with structures that occur due to saddle points. However similar to the case of gradient descent, choosing the momentum coefficient appropriately is a non trivial task.

4.3 Nestrov's Accelerated Gradient

This algorithm is proved to be a more stable version of Momentum[31]. It is proven to have better convergence rate than gradient descent. For smooth convex functions, NAG has a convergence rate of $O(1/T^2)$. The gradient descent has a convergence of $O(1/T)$. NAG can be rewritten as[31]:

$$\nu_{k+1} = \mu\nu_k - \varepsilon\nabla f(\Theta_k + \mu\nu_k) \tag{5}$$

$$\Theta_{k+1} = \Theta_k + \nu_{k+1} \tag{6}$$

NAG changes the velocity vector quicker and in a more stable manner by immediately correcting for the undesirable updates. Also the path traversed by NAG contains less oscillations compared to Momentum and hence tuning the momentum coefficient to suit NAG is an easier task compared to Momentum and the behavior of the algorithm around saddle points is qualitatively similar to that of Momentum which has been discussed above. [31] shows that the performance gap that occurs in using these first order momentum based techniques compared to second order methods such as Hessian Free optimization[19] is due to poorly chosen momentum constants.

5 Hessian based optimization techniques

The Hessian computed at the critical point contains information about the local curvature of the function. Second order optimization methods exploit this information to either accelerate or slow down according the curvature information. Newton's method, Hessian Free optimization[19], Krylov's subspace descent[32] are some of them to begin with.

5.1 Newton's method

Newton's method essentially approximates the function at the critical point with a quadratic function obtained by taking the taylor's expansion upto second degree and minimizes that function to obtain the value of the next iteration. Formally its given by:

$$\Theta_{k+1} = \Theta_k - \eta_k H_k^{-1} \nabla f(\Theta_k) \tag{7}$$

Newton's method essentially rescales the gradients in the eigenvector directions by their corresponding eigenvalues. This solves the problem of slowing down in the plateau regions around saddle points but when there are negative curvatures, this rescaling would infact make the algorithm traverse towards unstable critical points and thus making them attractors of the dynamical system[9]. Also since the computation of the Hessian is computationally very expensive, several other methods approximate this matrix to find the best condition number of the Hessian.

5.2 Method of conjugate gradients(CG)

Conjugate gradient descent is a technique in which search directions are not the directions of steepest descent anymore but the directions(ν_i) are chosen such that all the steps are A orthogonal(${\nu_i}^T A \nu_i = 0$ for $i \neq j$)to each other(where A is the matrix whose quadratic form is being minimized. The first direction is chosen to be the residual of the gradient descent. The step size is chosen exactly the same way as Newton's method by re-scaling the search directions by the corresponding curvatures($\frac{-\nabla f^T d_i}{{d_i}^T A d_i}$) at the critical points in the search directions.

The idea of conjugate gradient descent is very powerful since it allows for exact convergence in exactly n steps by choosing the next directions such that the reduction in the objective obtained in the previous steps is not undone. The step size is also exactly calculated such that the algorithm never has to optimize in that search direction again.However numerical errors do not allow the convergence to happen in n steps. The algorithm optimizes iteratively converging the minimum error as time t $\rightarrow \infty$. Hence the algorithm reaches an acceptable error value in finite time.

Proper preconditioning is very essential to enable CG to deal with negative curvatures and saddle points as its computationally expensive to compute the Hessian and dealing with negative curvatures becomes difficult as saddle points become attractors of CG.

5.3 Hessian Free optimization(HF)

Hessian free optimization[19] is a very successful optimization algorithm that guarantees faster convergence. The idea of HF is two fold. The approximation to the Hessian is done by computing Hd using the method of finite differences and partial calls to the conjugate gradient descent [24] are made. The trick here is that computing $H\nu$ is much easier problem than H and CG does not require explicit computation of the Hessian matrix. In order to compute Hd the Guass Newton

approximation to the Hessian is used as it is guaranteed to be positive semi-definite by construction. Due to this saddle point structures can be dealt with efficiently using Hessian Free optimization.

5.4 Krylov's subspace descent

This algorithm is very similar to that of Hessian Free optimization the only difference being instead of computing Hd at every iteration, a basis of Krylov's subspace is generated spanned by $\nu_i, \nu_i H_i, ..., \nu_i {H_i}^k - 1$ for a fixed value of k. The objective is numerically optimized in this subspace using a subset of the training samples. Use of only a subset of training data to optimize over the subspace might lead to convergence to an uninteresting optima. The efficiency of this algorithm to deal with saddle points is not very clear.

5.5 Algorithm to deal with saddle point structures

As seen in the the second order optimization algorithms discussed above, all of them solve the slowness problem by re-scaling the search directions by the curvature information. This method does not address the problem of negative curvatures. The way of dealing with this problem is to use an approximation to the damped Hessian making the resulting matrix positive semi definite. However, when there are huge discrepancies between the strength of the eigenvalues, re-scaling them with damped eigenvalues will cause very slow convergence in some directions while accelerating in some others leading to poor convergence. These methods are referred to as trust region approaches in literature[1].

Other approaches used in literature are that of ignoring the negative curvature regions which can not deal with saddle points either. [9] presents an algorithm which can effectively deal with saddle point structures.

The main idea in [9] is that instead of re-scaling the directions by their eigenvalues, the algorithm rescales them using the absolute values of the eigenvalues. This can efficiently deal with saddle point structures as saddle points become unstable fixed points of the system and hence the algorithm moves away from them. However, although the idea sounds straight forward it is not clear if the optimization problem that is being solved is still the same problem.

To solve this, [9] defines what is referred to as a generalized trust region approach. It is defined by allowing two changes to the trust region approach. Rather than optimizing the second order Taylor series expansion around the critical point, a first order one is considered and optimized. The second change is that instead of constraining the norm of the step $\triangle\theta$ the distance between θ and $\theta + \triangle\theta$ is constrained.

$$\triangle\theta = \underset{\triangle\theta}{\operatorname{argmin}}\, \tau_k f, \theta, \triangle\theta k \in \{1, 2\}\, S.T. d(\theta, \theta + \triangle\theta) \leq \triangle \tag{8}$$

The validity of the algorithm is shown in [9] in the following manner.

As a first order approximation is minimized, the minima is obtained at ∞ and since a trust region approach is used constraining the step size allowed for each step through a distance measure, the algorithm always takes a step to reach the boundary of the trust region. To identify the validity of first order approximation to the function as a function of $\nabla\theta$, we look at the difference between the first and the second order approximation and limit the effect of the second order terms. This gives

$$d(\theta, \theta + \triangle\theta) = |f(\theta) + \nabla f \triangle\theta + \frac{1}{2}\triangle\theta^T H \triangle\theta - f(\theta) - \nabla f \triangle\theta| = \frac{1}{2}|\triangle\theta^T H \triangle\theta| \leq \triangle \tag{9}$$

Here $\triangle$ is a small positive real number denoting the relaxation between the first and the second order approximations. ∇f is the partial derivative of f with respect to θ

However, solving for 9 is not trivial and [9] solves the following problem using the following lemma.

Lemma 1. *Let A be a non singular square matrix in $\mathbb{R}^n \times \mathbb{R}^n$ and $x \in \mathbb{R}^n$ be some vector. Then it holds that $|x^T Ax| \leq x^T|A|x$, where $|A|$ the matrix obtained by taking the absolute value of each eigenvalue of A*

Using this lemma, the upper bound for the distance is $\triangle\theta$ is approximated and the the step obtained in this form is given by

$$\triangle\theta = -\nabla f |H|^{-}1 \tag{10}$$

Hence the algorithm as seen above still solves the same optimization problem by replacing the rescaling with the absolute values of the eigenvalues and is very effective in dealing with saddle point structures. Moreover [10] proves that the absolute value of the Hessian is the only positive definite ideal preconditioner and that the absolute value provides a better estimation to the curvature than the Hessian itself.

In all the algorithms seen above, Momentum, NAG and the algorithm developed in [9] are designed to deal with saddle point structures. As [31] shows NAG is a more stable version than Momentum and its much easier to choose a momentum coefficient for NAG than Momentum. The algorithm discussed above has no parameters to optimize and given that choosing an appropriate momentum constant is not a very straightforward problem for reasons identified in [31], makes it attractive to be applied to deep neural networks.

6 Overfitting and training time

So far we have seen that underfitting is a problem that can be well tackled with an algorithm that effectively deals with saddle point structures in deep networks. However, the analysis of the loss surface shows that the floor contains exponentially large number of local minima. However all the local minima do not essentially have the same generalization abilities. To understand this better we turn towards

the perspective of dynamical system theory. In the analysis, the importance of initializations and depth independent training times is also presented. The effect of low curvature regions that occur due to saddle point structures or saturating non linearities[5] can be effectively dealt with an appropriate algorithm as discussed in the previous sections. However training time, specially in deep networks is also affected by the exploding [23], [21] and the vanishing gradient problems[5] specially in recurrent neural networks and very deep neural networks. In this paper, we provide evidence[29] that the right kind of initialization would lead to depth independent training times even in recurrent neural networks.

6.1 Dynamics of Neural Networks

Neural networks, including the feedforward networks have always demonstrated highly non linear dynamics. These dynamics include sudden bifurcations characterized by long plateaus followed by sudden transitions. [29] develops an exact analytical theory of learning in deep feed-forward linear networks and demonstrates that even feedforward linear networks with just one hidden layer exhibits highly non linear dynamics. In this paper we resort to these exact solutions[29] to get a better understanding of these dynamics in the perspective of overfitting. We also analyze the highly efficient greedy layerwise unsupervised pre-training in the light of its ability to obtain better generalization abilities.

6.2 Learning dynamics of Gradient Descent

The dynamics in this section has been derived in [29] and is essential for further analysis of overfitting and is hence briefly presented in this section. The learning dynamics of a feedforward linear network with one hidden layer is considered as shown in the figure 4. Let N_i be the number of units in layer i. Then the output of the network denoted by y is given by $y = W^{32}W^{21}x$. The loss function used here is the squared loss and the loss is given by $\sum_{i=1}^{P} \|y^\mu - W^{32}W^{21}x^\mu\|^2$ where (x^μ, y^μ) , $\mu = 1, 2..P$ are P training examples. Gradient descent on this loss function is results in the following set of equations[29].

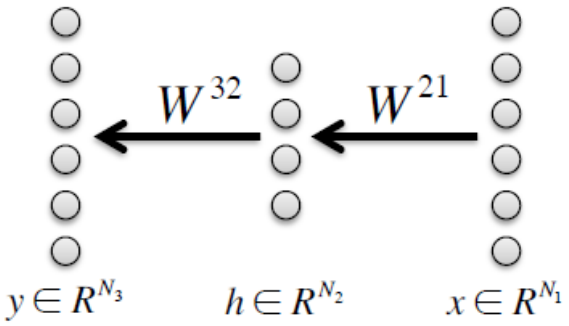

Figure 4: Three layer network

$$\nabla W^{21} = \lambda \sum_{i=1}^{P} W^{32^T}(y^\mu x^{\mu T} - W^{32}W^{21}x^\mu x^{\mu T}), \nabla W^{32} = \lambda \sum_{i=1}^{P}(y^\mu x^{\mu T} - W^{32}W^{21}x^\mu x^{\mu T})W^{21^T} \tag{11}$$

λ here is a small learning rate. For a small learning rate, continuous time limit results in the following set of coupled differential equations.

$$\tau \frac{d}{dt}W^{21} = W^{32^T}(\Sigma^{31} - W^{32}W^{21}\Sigma^{11}), \tau \frac{d}{dt}W^{32} = (\Sigma^{31} - W^{32}W^{21}\Sigma^{11})W^{21^T} \tag{12}$$

Where $\Sigma^{11} = \lambda \sum_{i=1}^{P} x^\mu x^{\mu T}$ and $\Sigma^{31} = \lambda \sum_{i=1}^{P} y^\mu x^{\mu T}$ are the input correlation matrix and the input output correlation matrix respectively.$\tau = \frac{1}{\lambda}$.

The goal is to learn the dynamics of learning as a function of the statistical structure of the input(Σ^{11}as well as the input-output mapΣ^{31}. To make the analysis easier, input correlation is set to I and hence only orthogonal inputs are used.

The singular value decomposition(SVD) of Σ^{31} is considered which provides crucial information about the statistical structure of the statistics of input-output map[28]. SVD of Σ^{31} is given by:

$$\Sigma^{31} = U^{33}S^{31}V^{11^T} = \sum_{\alpha=1}^{N_1} s_\alpha u_\alpha {\nu_\alpha}^T \tag{13}$$

V^{11} is an $N_1 \times N_1$ orthogonal matrix and has ν_α as columns which indicate the independent modes of variation in the input data. U^{33} is an $N_3 \times N_3$ orthogonal matrix and has ν_α as columns which indicate independent modes of variation in the output. S^{31} is an $N_3 \times N_1$ diagonal matrix and s_α are the singular values ordered by the value as $\alpha = 1, ..., N_1$.

A change of variables on synaptic space, $W^{21} = \overline{W}^{21}V^{11^T}$, $W^{32} = U^{33}\overline{W}^{32}$ simplifies the equation 12 to

$$\tau \frac{d}{dt}\overline{W}^{21} = \overline{W}^{32^T}(S^{31} - \overline{W}^{32}\overline{W}^{21}), \tau \frac{d}{dt}\overline{W}^{32} = (S^{31} - \overline{W}^{32}\overline{W}^{21})\overline{W}^{21^T}. \tag{14}$$

Let a^α be the α^{th} column of $\overline{W}^{21}$, and let $b^{\alpha T}$ be the α^{th} row of $\overline{W}^{32}$. Change of variables will transform equation 14 to:

$$\tau \frac{d}{dt}a^\alpha = (s^\alpha - a^\alpha \cdot b^\alpha)b^\alpha - \sum_{\gamma \neq \alpha} b^\gamma (a^\alpha \cdot b^\gamma), \tau \frac{d}{dt}b^\alpha = (s^\alpha - a^\alpha \cdot b^\alpha)b^\alpha - \sum_{\gamma \neq \alpha} a^\gamma (b^\alpha \cdot a^\gamma). \tag{15}$$

The dynamics here shows very interesting competitive and cooperative properties between the weights going from an independent input mode the hidden layer and the weights going from the hidden layer to the independent variations in the output. As it can be seen in equation 15, the first term in both equations show

that the modes in both layers compete with each other to reflect the singular value, s^α, by both pointing in the same direction in the weight space as well as driving each other to higher magnitudes. The second term shows that the weights in different connectivity modes compete with each other thus forcing different modes to become orthogonal. This leads the network to a decoupled regime.

The dynamics of 15 has been worked out in [4]. The results show that the fixed points of this dynamical system are only saddle points and it has no non global, local minima.

The trajectories of the dynamics of these equations is obtained by taking a special class of initial conditions on the weights in both the connectivity modes in the form of a^α and $b^\alpha \propto r^\alpha$. Here r^α is a fixed set of N_2 vectors which form an orthonormal basis for connectivity modes in both layers. It can be clearly seen that under this class of initial conditions, a^α and $b^\alpha \propto r^\alpha$ for the rest of the time course of learning. As the different connectivity modes do not interact with each other they evolve independently of each other.

This class of conditions becomes an invariant manifold in the weight space. By considering the scalar projections $a = a^\alpha \cdot r^\alpha$, and $b = b^\alpha \cdot r^\alpha$ and $s = s^\alpha$ the dynamics of a,b is given by:

$$\tau \frac{d}{dt} a = b(s - ab), \tau \frac{d}{dt} b = a(s - ab) \tag{16}$$

This dynamics is derived from gradient descent of the following functions

$$E(a, b) = \frac{1}{2\tau}(s - ab)^2 \tag{17}$$

Deriving this form was essential to understand the train of thought needed to understand the essence of terms in the discussion. It can be seen clearly in set of coupled differential equations that scaled symmetries exist in the system meaning that $E(a, b)$ satisfies symmetries under one scaling transformation $a \to \frac{a}{\lambda}$, and $b \to \frac{b}{\lambda}$. Noether's theorem implies the existence of a conserved quantity[29], $a^2 - b^2$. Hence the dynamical trajectories follow hyperbolas of constant $a^2 - b^2$ until it approaches the fixed point manifold which is a hyperbola $ab = s$. A phase potrait of the dynamics is shown in the figure 5

The dynamics of deep networks has also been worked out in [29] which show similar scaled symmetries in the system. As discussed later in the section, such scaled symmetries caused by random initializations lead to a proliferation of saddle points as discussed in section 3. The time scales of these trajectories show that for decoupled initialization conditions, the amount of time that is required to reach the fixed point is only a finite time larger in deep networks than a shallow network. The dynamics also show that as the network learns the input output map, the modes with higher singular values are learned first. Recall that the singular values indicate the strength of the connection between the independent variation in the input and the independent variation in the output. The results obtained here show that with right kind initializations, the amount of training time required to train a network is independent of the depth of the network up to scale.

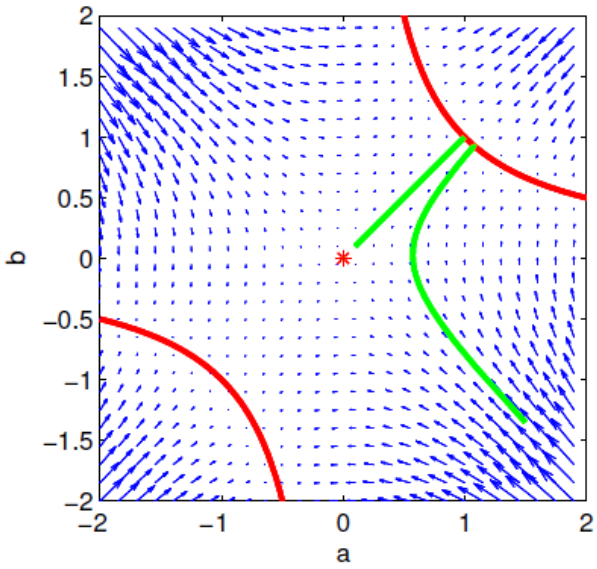

Figure 5: The red hyperbola indicates the invariant manifold hyperbolic manifold which is the fixed point of the system. The dynamics follow hyperbolic trajectories(green) to reach this invariant manifold.

6.3 Unsupervised pre-training

In the previous section the significance of a good initialization and its effect on the amount of training time has been identified. Existence of unstable fixed points or unstable limit cycles in the phase space of a system indicates the existence of several possible bifurcation points. This has been proven to happen in simulations of neural networks. To demonstrate the effect of proper initialization, a simple example is shown below. Consider a deep symmetric double well potential of two wave functions whose phase space portrait is shown in the figure 6. Starting from any point to the right of the unstable fixed point would lead to a convergence to the local minima on the right and vice versa. Hence if the local minima are qualitatively different then the initialization of the parameters defines whether or not the trajectories reach the local minima of higher quality. By extrapolating this to higher dimensions in which there is a proliferation of saddle points, the effect of a good initialization can not be emphasized enough.

Unsupervised pre training and greedy layer wise pre training[6] ave proven to be extremely successful in training deep networks. Empirical analysis on the same has proven that unsupervised pre training somehow initialized the weights in a good basin of attraction[11]. [29] shows that unsupervised pre-training also leads to good optimization in terms of depth independent convergence rates by initializing the weights in a near orthogonal regime and with order of strength between connectivity modes at $O(1)$. Apart from unsupervised pre- training, carefully scaled random initializations called normalized initialization which preserves the norm of the gradient vector back propagated through the network have shown good convergence

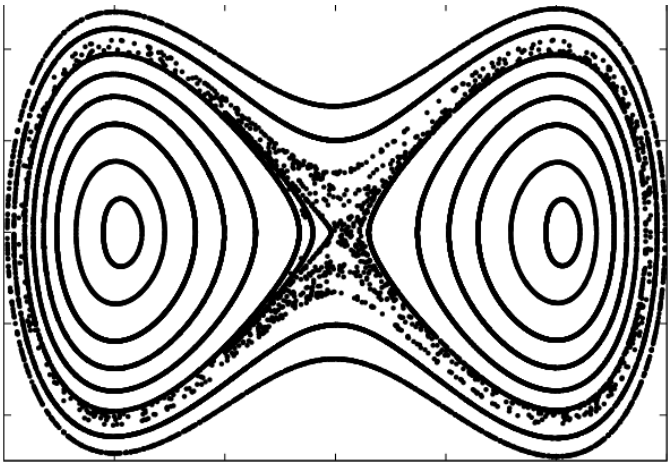

Figure 6: Phase space portrait of the double well potential

rates[5]. [29] hypothesizes that the property that is required for an initialization to achieve depth independent learning time is not that of mere norm preservation but that of dynamical isometry. This means that the Jacobian should have as many singular values as possible around $O(1)$. Normalized initialization has been shown to not have this property as norm preservation through scaling could lead to an anisotropic projection onto eigenvectors in different directions leading to much slower steps in those directions translating into slower convergence rates. [29] shows that random orthogonal initializations achieve perfect dynamical isometry while unsupervised pre-training achieves it approximately leading to depth independent learning rates. For deep non-linear networks operating just beyond the edge of orthogonal chaos is a good regime for the learning dynamics [29].

Unsupervised pre-training achieves more than depth independent learning times. It has been shown to act as a good regularizer to avoid overfitting. Several other techniques[15],[16],[17],[26] have been known to avoid overfitting. However unsupervised pre-training has an unique place among these regularizers[11]. It purely initializes the weights in an appropriate regime which in turn acts as a good regularizer. [29] shows that unsupervised pre training only works if the independent modes of variation in the input is nearly the same as the independent modes of variation in the input - output map. This is interesting because several local minima at the floor have the same energy level (training error) but the test error differs from the parameters. No free lunch theorem[34] clearly states this phenomenon. In this paper, we suggest that unsupervised pre training acts as the appropriate inductive bias placing the initialization in the the basin of attraction corresponding to a good quality local minima which leads to a better generalization error.

7 Conclusion

In this paper we discussed optimization in deep neural networks in terms of avoiding underfitting, improving the convergence rate and avoiding overfitting. The

first important point of discussion is the proliferation of saddle points for random functions in high dimensions. This has been proven by establishing an equivalence between deep neural networks and the Hamiltonian of the spherical spin glass model. Various optimization algorithms typically used in neural networks have been evaluated in the presence of saddle points. A proposed algorithm which theoretically is able to solve the problem of saddle points is discussed. From this analysis, it can be concluded that momentum based first order techniques with appropriately chosen momentum coefficients and rescaling the search directions by the inverse of the absolute values of the curvature are the two methods found to deal with the problem of saddle points effectively. Hence these algorithms solve the problem of underfitting in deep neural networks as the floor of the loss surface is very close to the ground state and for all practical purposes reaching the floor would minimize the training error to an acceptable accuracy. The problem of training time caused due to pathological curvatures[19] in the loss surfaces is effectively solved by the same algorithms used to minimize the training error. The problem of reducing the training time which occurs due to the exploding or the vanishing gradients problem can be solved with initializations which exhibit dynamical isometry and for deep non linear networks, it can be solved by operating just beyond the edge of orthogonal chaos[29]. In this paper we postulate that the most difficult of the optimization problems in deep networks is that of overfitting or poor generalization caused due to lack of inductive bias. As shown in [29] scaled symmetries caused due to random initializations cause several unstable fixed points in the learning dynamics. Poor inductive bias is the main reason for poor generalization. Unsupervised pretraining is a form of regularizer which initializes the weights by creating a dynamical isometry and at the same time, adding inductive bias to the dynamics by appropriately initializing the weights to fall in the basin of attraction of a good quality local minima.

References

[1] Shun-ichi Amari. *Differential-Geometrical Methods in Statistics*, volume 28 of *Lecture Notes in Statistics*. Springer New York, 1985.

[2] Daniel Amit, Hanoch Gutfreund, and H. Sompolinsky. Spin-glass models of neural networks, 1985.

[3] Antonio Auffinger. Random matrices, complexity of spin glasses and heavy tailed processes. (May):1–26, 2011.

[4] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53–58, 1989.

[5] Yoshua Bengio. Learning long-term dependencies with gradient descent is difficult.

[6] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19(1):153, 2007.

[7] Alan J. Bray and David S. Dean. Statistics of critical points of gaussian fields on large-dimensional spaces. *Physical Review Letters*, 98(15):1–4, 2007.

[8] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. 38, 2015.

[9] Yann Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. pages 1–14, 2014.

[10] Yann N Dauphin, Harm De Vries, Junyoung Chung, and Yoshua Bengio. Rmsprop and equilibrated adaptive learning rates for non-convex optimization. 2014.

[11] D Erhan, Y Bengio, and a Courville. Why does unsupervised pre-training help deep learning? *of Machine Learning* , 9(2007):201–208, 2010.

[12] Yan V. Fyodorov and Ian Williams. Replica symmetry breaking condition exposed by random matrix calculation of landscape complexity. *Journal of Statistical Physics*, 129(5-6):1081–1116, 2007.

[13] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 9:249–256, 2010.

[14] Ian J. Goodfellow, Oriol Vinyals, and Andrew M. Saxe. Qualitatively characterizing neural network optimization problems. *To appear: ICLR-2015*, pages 1–11, 2015.

[15] Geoffrey Hinton. Dropout : A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15:1929–1958, 2014.

[16] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv: 1207.0580*, pages 1–18, 2012.

[17] Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton Google. A simple way to initialize recurrent networks of rectified linear units. pages 1–9, 2015.

[18] Satya N. Majumdar, Céline Nadal, Antonello Scardicchio, and Pierpaolo Vivo. How many eigenvalues of a gaussian random matrix are positive? *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 83(4):1–18, 2011.

[19] James Martens. Deep learning via hessian-free optimization. *27th International Conference on Machine Learning*, 951:735–742, 2010.

[20] Pankaj Mehta and David J. Schwab. An exact mapping between the variational renormalization group and deep learning. *arXiv*, 2014.

[21] Razvan Pascanu and Yoshua Bengio. Learning to deal with long-term dependencies. (3):0–1, 1994.

[22] Razvan Pascanu, Yann N. Dauphin, Surya Ganguli, and Yoshua Bengio. On the saddle point problem for non-convex optimization. *arXiv preprint*, page 11, 2014.

[23] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. 2012.

[24] Barak A. Pearlmutter. Fast exact multiplication by the hessian, 1994.

[25] B.T. Polyak. Some methods of speeding up the convergence of iteration methods, 1964.

[26] Jason Tyler Rolfe and Yan LeCun. Discriminative recurrent sparse auto-encoders. *CoRR*, page 15, 2013.

[27] Levent Sagun and Yann Lecun. Xplorations on high dimensional landscapes 2 . (2013):1–11, 2015.

[28] a.M. Saxe, James L McClelland, and Surya Ganguli. Learning hierarchical category structure in deep neural networks. *Proceedings of the 35th annual meeting of the Cognitive Science Society*, pages 1271–1276, 2013.

[29] Andrew M Saxe, N E Jan, and James L Mcclelland. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. pages 1–21.

[30] Leonard Susskind. The anthropic landscape of string theory. page 21, 2003.

[31] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. *Jmlr W&Cp*, 28(2010):11391147, 2013.

[32] Oriol Vinyals and Daniel Povey. Krylov subspace descent for deep learning. *arXiv preprint arXiv:1111.4259*, (2):1–11, 2011.

[33] E P Wigner. On the distribution of the roots of certain symmetric matrices. *The Annals of Mathematics*, 67(2):325–327, 1958.

[34] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 4 1997.