An analysis of the impact of subsampling on the neural network error surface

Cody Dennis[a], Andries Engelbrecht[b,c], Beatrice M. Ombuki-Berman[*,a]

[a] *Department of Computer Science, Brock University, St. Catharines, Canada*
[b] *Department of Industrial Engineering, Stellenbosch University, Stellenbosch, South Africa*
[c] *Computer Science Division, Stellenbosch University, Stellenbosch, South Africa*

Abstract

This paper empirically analyses the impact of changes to the set of training examples on the neural network error surface. Specific quantitative characteristics of the error surface related to properties such as ruggedness, modality and structure are measured and visualized as the set of training examples changes. Both the case of random subsampling and active learning from a fixed dataset are examined, producing ten different training scenarios. For each scenario eleven error surface characteristics are calculated for five common benchmark problems. The results demonstrate that the error surface characteristics calculated using only a subsample of the available data commonly do not generalize to that of the full dataset. The observed error surface characteristics are significantly impacted by the particular set of examples used to calculate error. Some error surface characteristics are significantly altered by small changes in the set of examples used to calculate error. The main finding from this study is that when the set of training examples may change during training the training of a neural network is in essence a dynamic optimization problem, suggesting that optimization algorithms developed specifically to solve dynamic optimization problems may be more efficient at training neural networks under such conditions.

Key words: Active Learning, Fitness Landscape Analysis, Exploratory Landscape Analysis, Neural Networks

1. Introduction

Artificial neural networks have been used in practical applications and studied academically for decades [1, 2]. Recently, deep neural networks have even been used to produce systems that meet and exceed human level performance in tasks that traditionally require human input [3]. Despite these successes, the characteristics of neural network error surfaces are not well understood [4]. This is unfortunate, because such characteristics are intimately related to the success or failure of a training algorithm. An understanding of the characteristics of a neural network error surface could be used to identify structures that trap or mislead a training algorithm and to adjust the algorithm appropriately.

Previous works have focused on characterizing properties of the neural network error surface such as modality and structure with respect to features such as the activation functions used, the error function used, or the presence of regularization. Such works have produced a variety of novel realizations and tools for analysis. Some results of existing works include tools for estimating (and analyzing) the modality and size of basins of attraction on an error surface [5, 6]; the realization that saddle points are highly prevalent in neural network error surfaces [7, 8], and evidence that all valleys in the error surface could be a part of the same basin of attraction [9]; evidence that the structure of a valley may be related to the generalization behavior of weight vectors found in that valley [3]; and revelations about the general shape of the error surface [10], and its symmetry about the origin [11]. Other works have analysed how the error surface is influenced by choice of error function [5], approach to regularization [12], and network architecture [13].

However, such works have relied on the implicit assumption that the error surface is static. Often no attention is paid to the particular subset of data used to evaluate the error surface. In such cases it is assumed that all available data is used during error calculations, or that observations made on available data will generalize. In practice such assumptions will not hold. In the case of active learning the training set is continuously updated to identify only those examples that are most informative. Similarly, in online learning new data constantly arrives over time, and useful information must be extracted as it becomes available. A third example is mini-batch training, where a small subset of the available data is randomly chosen for each error calculation. This work considers the case where training is performed with sub-samples from a fixed dataset. The impacts of random subsampling and entropy based uncertainty sampling, a popular algorithm from the active learning literature [14, 15, 16], on error surface characteristics are quantitatively evaluated.

[*] Corresponding author
Email address: bombuki@brocku.ca (Beatrice M. Ombuki-Berman)

The authors of this paper are unaware of any works that analyze the impact of changes in the training set on the properties of the resulting error surface. This work formally analyzes the impact of changing the sample of data over which error is calculated on a collection of specific error surface properties. When the data over which error is calculated changes with time, the same weight vector may have different error values at different times. In addition to a change in the error of specific weight vectors, the error surface's characteristics may change as well. This potential change in the error surface has significant implications for both training the network and generalization performance.

The main contribution of this work is a formal analysis of the impact of changing the set of training examples on specific properties of the error surface. While experiential knowledge of what happens is available, formal analysis of this topic is lacking. Fitness landscape analysis provides a formal approach to study what impact, if any, changes in the set of training example has on the error surface. Some error surface characteristics appear to be relatively insensitive to the examples over which error is calculated, however, this is not always the case. The results of this study suggest that some characteristics of the error surface can vary significantly from even small changes in the subset of training data used. When this occurs, assumptions about the general characteristics of the error surface cannot be made or exploited. It is also demonstrated that minima found for a subset of the available data often do not occupy the same region as minima found for the entire dataset. Similarly, minima found when the entire dataset is processed as a series of smaller batches often do not occupy the same region as minima found when the entire dataset is considered simultaneously. When this is the case, found weight vectors cannot be expected to generalize well to new data. Based on these results, this paper concludes that training on subsamples result in a dynamic optimization problem. This realization opens further opportunities to develop more efficient training algorithms for dynamically changing error surfaces.

The rest of this work is organized as follows: Section 2 provides background material on neural networks, fitness landscape analysis, and neural network training. Section 3 describes the methodology used in this study. Section 4 gives relevant implementation details. Section 5 presents the results of this study. Finally, Section 6 gives concluding remarks and avenues for future work.

2. Background

This section provides background information relevant to this work. Section 2.1 provides a survey of related literature. Section 2.2 discusses neural networks, the evaluation of weight vectors, and training methods. Section 2.3 discusses fitness landscape analysis. Section 2.4 discusses specific properties of the fitness landscape and their mathematical formulations. Finally, Section 2.5 discusses sampling methods for estimating properties of the fitness landscape.

2.1. Related work

Many works have attempted to characterize some properties of the neural network error surface [3, 5, 6, 7, 8, 9, 10, 11, 17, 18], yet the characteristics of these surfaces are still poorly understood [4, 5, 17]. Some reasons for this lack of understanding include high dimensionality, and an over reliance on unrealistic assumptions about the error surface [5]. Such works have examined the relationship between the error surface and features such as the activation functions used, the error function used, modality, regularization, or error surface structure. Studies of the relationship between choice of activation function, and properties of the error surface are common [19, 20]. Multiple researchers have examined the use of piecewise linear activation functions, finding that such functions commonly have spurious local minima in the error surface [20]. Cheridito et al. have identified and classified critical points for shallow neural networks using the ReLU function [19]. Barannikov et al. have recently examined the relationship between the error function, generalization performance, and gradient trajectories [21]. Mehta et al. have recently demonstrated that flat minima in the error surface can be removed via L_2 regularization. However, local minima which are not global minima may be found in the resulting error surface [22]. Interestingly, Vesseron et al. have studied deep neural networks from the prespective of game theory and illustrated relationships between congestion games and properties of the error surface [23]. The relationship between mini batch training, learning rate, and some properties of the error surface has also been examined. Kafka and Wilke primarily focus on the selection of appropriate learning rates, but identify that changing the batch of training examples introduces discontinuities into the error surface, and recognize this as a source of issue [24]. In a similar work, Granziol and Zohren derive expressions for the selection of learning rate based on batch size and describe the relationship between the batch Hessian and the empirical Hessian [25].

Mehta et al. recently performed a study of the error surface of the XOR problem [18], where it was demonstrated empirically that the size of the hidden layer and the value of the regularization coefficient influence the number of local minima in the error surface. Bosman et al. have studied the modality of the neural network error surface in several works [5, 12, 13]. It was observed that choice of error function, use of regularization, and network architecture all significantly influence the number of minima found in the error surface and the structure of these minima. Draxler et al. suggest that all minima in the error surface may actually be a part of the same basin of attraction [9], based on the observation that paths of non-increasing error can be found between various minima. Others have noted this phenomena as well [7]. Liang et al. provide conditions under which all local minima on the training error surface must have zero error [26]. It has also been established that wide and narrow valleys are common in the neural network error surface. It has been observed that the structure of the valley in which a weight vector is found may be related to the generalization behavior of the weight vector [3, 27]. A common issue with empirical studies is the implicit assumption that the error surface will be static and unchanging, or that error calculations made for subsets of the possible data will reflect the general characteristics of the true error surface [5, 9, 12, 13, 18]. Despite the abundance of work examining the neural network error surface, the authors of this paper are unaware of any works that analyze the impact of changes in the training set on the properties of the resulting error surface.

2.2. Neural networks

Neural networks are universal approximators [28], i.e. they can be used to approximate an extensive range of functions, to an arbitrary level of precision, provided that an appropriate structure and set of weights and biases can be found [28].

The function to approximate is represented using a set of input values and desired output values known as examples. The process of finding an appropriate set of weights and biases, or weight vector (w), is known as training the network. Training the network is simply an optimization problem. The goal of this problem is to assign the best possible values to the weights and biases, such that the error function is optimized for a particular set of examples. Gradient descent backpropagation [29] is the most well known training algorithm [30, 31]. This algorithm simply follows the gradients of the error function with respect to the weights and biases to find a local minimum. The resulting weight vector is a potential solution to the neural network training problem. The produced weight vector is then used to process previously unseen examples.

Error calculations in supervised neural network training involve comparing the output of the network to the desired output for a set of training examples. The output of the network depends on the network's weight vector. One common choice for classification problems is the cross-entropy error function [32], formulated as

$$CE = -\sum_{l=1}^{L}\sum_{k=1}^{K} t_{k,l} log(o_{k,l}) \tag{1}$$

where K is the number of output nodes, L is the number of examples, $t_{k,l}$ is the expected output of node k for example l, and $o_{k,l}$ is the actual output of node k for example l. The cross-entropy error function requires that the output of the network can be interpreted as probabilities. Typically this is achieved using a softmax output layer, where the output of each node in the output layer is calculated according to

$$o_{k,l} = \frac{e^{net_{k,l}}}{\sum_{i=1}^{K} e^{net_{i,l}}} \tag{2}$$

where $net_{k,l}$ is the input of output node k on example l.

The error for a specific weight vector depends on the specific examples over which it is calculated. If the examples used for error calculations can change with time, then the error for a specific weight vector can change with time as well. When the error of a specific weight vector can change with time, the process of training the network is a dynamic optimization problem [33]. A dynamic optimization problem is a problem in which the quality of solutions is not static and may change with time. Solutions which were previously evaluated, and found to perform well, may perform poorly in subsequent evaluations. Similarly, solutions which underperformed during previous evaluations may be found to perform well during subsequent evaluations. This is in contrast to a standard optimization problem, where the quality of found solutions is static and cannot change between different evaluations. It is important to note that in the context of dynamic optimization changes in the quality of a solution are not the result of noise in the method of evaluation, that is, the quality of a solution also depends on the time at which the solution is evaluated.

As a simple illustration, suppose that for some classification task there is a network that always outputs a positive response. During evaluation, the error of this hypothetical network could be changed arbitrarily by simply varying the proportions of positive and negative training examples over which the error is calculated. In general, if the error of a specific network is calculated over different sets of examples, different error values can be expected. Furthermore, it is possible that the error surface's characteristics, including the location of valleys, will change as the error of specific weight vectors changes. An understanding of these potential changes in the error surface is crucial for selecting a high quality weight vector. A network that performs well on the training set can perform similarly in testing only if the generalization error surface has a similar valley in the same position.

Two situations, among others, where the training examples can change over time, are active learning [34] and mini-batch training. During training one complete pass over the dataset is referred to as an epoch. During mini-batch training random sampling is used to process the entire dataset as a series of smaller batches, with the weight vector updated for each batch. The size of the batches is a parameter set by the user. In active learning, the algorithm attempts to filter redundant data and focuses on only the most informative patterns [34].

2.3. Fitness landscape analysis

Fitness landscape analysis is a concept inspired by genetics research [35] that is commonly applied in the fields of optimization and evolutionary computation [36]. The goal of fitness landscape analysis is to examine and exploit relationships between the spatial distribution of solutions and the solutions' fitness values (i.e. solution quality). For a specific network structure and set of examples, the set of all possible weight vectors, the error of each weight vector, and some measure of the distance between weight vectors forms what is known as a fitness landscape [37] or equivalently an error surface. In this context a specific weight vector is a potential solution to the problem of training the network and the error of that vector is synonymous with its fitness. A specific weight vector can be interpreted as a position and the error value as a height corresponding to that position. Adjacent weight vectors are those that are near to one another according to the distance measure. In this way the set of possible weight vectors can be viewed as a landscape with hills and valleys. Valleys in this landscape correspond to weight vectors which perform well relative to adjacent vectors. The error surface is easily visualized if the weight vector has only two weights. However, neural networks have many more weights and biases in practice. Fortunately, the concept of an error surface can be easily applied, if not visualized, when the number of weights and biases increases.

Fitness landscape analysis provides a formal approach for analyzing an error surface and is an active area of research [5, 38, 39, 40, 41]. It is a popular tool in the field of evolutionary computation and has been recently used to evaluate benchmark problem suites [42], for automated algorithm selection [43], control parameter performance prediction [44], and to gain insights into the search behavior of the particle swarm optimization algorithm [45], among other applications. In fitness landscape analysis numeric measures of an error surface characteristics are formulated in a general manner. The value of such characteristics are estimated based on multiple samples of valid weight vectors. Taken together a collection of error surface characteristic estimates can be used to characterize and describe the sample of weight vectors over which they were calculated. If the samples are representative of the error surface, then characteristics calculated from those samples are representative of the error surface as well [5].

2.4. Measure for error surface characteristics

A variety of different methods exist in the literature which can be used to characterize an error surface, referred to here as error surface characteristics. In general, such methods are related to four high level properties [46]: modality, structure, separability, and searchability. Modality relates to the number and distribution of optima in the error surface. Structure refers to the variability in the error surface and how optima are connected. Separability refers to the correlations and interdependencies among weights and biases. Searchability is a measure of the difficulty of an optimization problem. It can be thought of as the ability of an arbitrary weight vector to be improved [47].

Some of the characteristics used in the literature include: The estimated average magnitude of gradients (Grad-Ave) and the estimated standard deviation of gradients (Grad-Std) [48]. The magnitude of gradients are estimated according to

$$g(i) = \left| \frac{f_i(w(i+1)) - f_i(w(i))}{dist(w(i+1), w(i))} \right| \tag{3}$$

where $w(i)$ gives the i^{th} weight vector produced by the sampling method, f_i gives the error of $w(i)$ calculated for a subset of the available data which is being used for learning at step i, and $dist$ is the distance function.

Dispersion [49] measures the average pairwise distance between high performing weight vectors from a sample. Dispersion is calculated as

$$Disp(b) = \frac{\sum_{i=1}^{b} \sum_{j=i+1}^{b} dist(\hat{w}(i), \hat{w}(j))}{\frac{1}{2}b(b-1)} \tag{4}$$

where b gives the number of best weight vectors to consider, and $\hat{w}(i)$ gives the i^{th} best weight vector. Note that b must be less than the total number of weight vectors in the sample.

Fitness distance correlation (FDC) [50] measures the correlation between the distance and the change in error between weight vectors. FDC is defined for real valued problems as [51]

$$FDC = \frac{\sum_{i=1}^{n} Q(i)R(i)}{\sqrt{\sum_{i=1}^{n} Q(i)^2}\sqrt{\sum_{i=1}^{n} R(i)^2}} \tag{5}$$

where $Q(i) = f_i(w(i)) - f_i(\hat{w}(1))$, $R(i) = dist(w(i), \hat{w}(1)) - \bar{d}$, n is the number of weight vectors in the sample, and $\bar{d}$ gives the average distance between any weight vectors in the sample, and the weight vector from the sample with the best error.

The first entropic measure (FEM) [52] measures the ruggedness or variability in a error surface based on abrupt changes in error between adjacent points. FEM is calculated as

$$H(\epsilon) = -\sum_{p \neq q} P_{pq} log_6 P_{pq} \tag{6}$$

where ϵ is chosen to maximize H, $p, q \in \{\bar{1}, 0, 1\}$, $P_{pq} = \frac{n_{pq}}{n}$, and n_{pq} gives the number of adjacent pairs containing p and q in the sequence $S(\epsilon)$. Finally, $S(\epsilon)$ is a sequence of symbols where each symbol $S_i(\epsilon)$ is determined by

$$S_i(\epsilon) = \begin{cases} \bar{1} & f_i(w(i)) - f_i(w(i-1)) < -\epsilon \\ 0 & |f_i(w(i)) - f_i(w(i-1))| \leq \epsilon \\ 1 & f_i(w(i)) - f_i(w(i-1)) > \epsilon \end{cases} \tag{7}$$

M_1 and M_2 [53] characterize modality and neutrality. M_1 estimates the proportion of the search space that is neutral. A weight vector in the space is considered to occupy a neutral area if its error does not differ greatly from adjacent points. M_2 estimates the size of the largest neutral region, relative to the total size of the search space. M_1 is calculated as

$$M_1 = \frac{n_{neutral}}{n_{3point}} \tag{8}$$

where $n_{neutral}$ gives the number of neutral 3-point structures from the sample, n_{3point} gives the total number of 3-point structures in the sample, and a 3-point structure is defined as three neighboring points $w(i-1)$,$w(i)$, and $w(i+1)$ from the sample. A 3-point structure is defined as neutral if $max(f_i(w(i-1)), f_i(w(i)), f_i(w(i+1))) - min(f_i(w(i-1)), f_i(w(i)), f_i(w(i+1))) \leq \epsilon$ for some ϵ. M_2 is calculated according to

$$M_2 = \frac{|S_{neutral}|}{n_{3point}} \tag{9}$$

where $S_{neutral}$ gives the longest contiguous sequence of neutral 3-point structures in the sample.

The modality and the size of neutral regions within the space can also be estimated with Stag-N and Stag-L [5]. Stag-N and Stag-L estimate the number and the average length of stagnant regions in the search space by first smoothing the sequence of error values associated with the sample, and then comparing the changes in error in the smoothed sample to the standard deviation of the smoothed sample. The complete algorithms for calculating Stag-N and Stag-L can be found in [5]. Such estimates are an important means for characterizing a search space, since stagnant regions represent areas where an optimization algorithm may become trapped [5]. Stag-N and Stag-L may be more robust than M_1 and M_2, because they take into account the spread of error values in a sample when identifying stagnant regions.

Kurtosis and skewness [54] both characterize the distribution of error values in a sample. Kurtosis measures how tail heavy the distribution is, relative to a normal distribution. Skewness is a measure of the relative weight of the two tails. Kurtosis is estimated according to

$$Kurtosis = \frac{\frac{1}{n}\sum_{i=1}^{n}(f_i(w(i)) - \bar{f}(w))^4}{(\frac{1}{n}\sum_{i=1}^{n}(f_i(w(i)) - \bar{f}(w))^2)^2} - 3 \tag{10}$$

where $\bar{f}(w)$ gives the average error of all weight vectors from the sample. Skewness is estimated according to

$$Skewness = \frac{\frac{1}{n}\sum_{i=1}^{n}(f_i(w(i)) - \bar{f}(w))^3}{(\frac{1}{n-1}\sum_{i=1}^{n}(f_i(w(i)) - \bar{f}(w))^2)^{\frac{3}{2}}} \tag{11}$$

2.5. Sampling methods

Each of the previously described error surface characteristics attempts to quantify some characteristic of the error surface based on a sample of weight vectors. Thus a method of selecting weight vectors for the sample is required. Different sampling methods can produce sets of weight vectors with different spatial relationships and different distributions of error. As a result, different sampling methods may reveal different characteristics of the error surface [55]. For example, a uniform random sample of weight vectors is unbiased, but is not likely to produce high performing weight vectors. Since this method does not take the quality of weight vectors into account, and successively generated weight vectors are not spatially correlated, measuring stagnant regions or optima on such a sample is not likely to provide a reliable estimate of the optima of the overall error surface. However, examining the stagnant regions or optima of a uniform random sample could provide a measure of the similarity of arbitrary weight vectors, and the variability of the error surface.

Some common sampling strategies include: Uniform random (UR) sampling, which selects weight vectors uniformly at random from within a bounded search space. Progressive random (PR) sampling [56], which begins close to the edge of a bounded search space and generates samples by taking randomly sized steps away from the boundary in each problem dimension. This process is performed repeatedly to traverse the search space. Whenever the opposite boundary is encountered in any problem dimension, the direction of travel is reversed for that dimension. Manhattan progressive random (MPR) sampling [57] is similar to progressive random sampling, except that a fixed step size is used and at each step only a single problem dimension is changed. Progressive gradient (PG) sampling [6] is similar to progressive random sampling except that the direction of travel for each dimension is chosen using the gradients of the error function and the starting point is chosen randomly. Gradient information is used to bias the sampling toward interesting regions of the search space, which may contain high performing weight vectors. Gradients are obtained using the backpropagation algorithm.

3. Methodology

The main purpose of this study is to quantitatively evaluate how changes in the data over which error is calculated may influence the neural network error surface. This section discusses the experimental methodology. Section 3.1 provides some terminology used throughout the rest of the work. Section 3.2 describes the random data sampling performed for this work. Section 3.3 describes the approach to active learning used in this work. Section 3.4 discusses our approach for comparing the locations of minima in different error surfaces.

3.1. Terminology

For the remainder of this paper, an error surface characteristic which is calculated using a fixed proper subset of the available data for a particular problem will be referred to as a partial estimate. An error surface characteristic calculated using all available data for a particular problem is referred to as a fully informed estimate. In addition to partial and fully informed estimates, aggregate estimates were also calculated. An aggregate estimate characterizes the error surface similarly to how it would be seen during training, where the set of training examples may change between error calculations. An aggregate estimate can be calculated for any of the training scenarios previously described. Error calculations for the sample of weight vectors are spread evenly over the various subsets of examples associated with the training scenario. For example, suppose there are ten subsets of data to use for error calculations, and one hundred weight vectors are generated using uniform random sampling, then the first ten weight vectors would be evaluated using the first subset of data, the next 10 using the second subset, and so on.

3.2. Random data sampling

Random data sampling scenarios were simulated using a sliding window over a fixed permutation of each dataset. All combinations of three window sizes and three speeds were tested. The three sizes tested were 50 examples, 100 examples, and 150 examples. The window is moved by removing the k oldest examples, and replacing them with the k next examples. The examples contained in the window at different times represent different subsets of data used. The values of k considered were calculated as $33.\bar{3}\%$, $66.\bar{6}\%$, and 100% of the window size. The size of the window is analogous to the batch size used during training. When k is equal to the window size, then there is no overlap between batches, as in classical mini-batch training. When k is less than the window size, there will be overlap between batches. Intuitively, it may be argued that overlap in data between batches could encourage stability in the error surface. Values of k less than the window size are considered in order to test this intuition. This produces nine random data sampling scenarios. For each scenario, separate simulations were performed using the examples from each of the first five window positions, provided that the size of the dataset permitted this. This results in up to five different subsets of data over which error surface characteristics can be analyzed for each random data sampling scenario for each problem. These combinations of parameters produce scenarios with varying degrees of overlap among the training examples for different window positions.

3.3. Active learning

This subsection describes the use of active learning in this study. Uncertainty sampling, as described in [58], is used to determine the order in which examples should be added to the training set. Conditional entropy, as described in [59], is used to quantify the model's uncertainty. This process begins with 50 randomly chosen training examples. The algorithm is allowed to train for some number of iterations on the current training set, then entropy based uncertainty sampling is used to select the next example to add to the training set. This process is repeated until all examples have been added to the training set. Entropy based uncertainty sampling is a popular method in active learning [14] and is commonly examined in state of the art active learning research [14, 15, 16].

The sequence of training examples for a particular problem will be referred to as the active learning sequence. For each active learning sequence, ten subsets of data were selected for the analysis of error surface characteristics. The first subset consists of the initial 50 random examples. The next subset consists of the 50 random initial examples combined with the next $\frac{1}{9}$th of the active learning sequence. The next subset consists of the 50 random initial examples combined with the next $\frac{2}{9}$ths of the active learning sequence, and so on. This produces a variety of subsets representing different levels of informative and redundant data to be examined for each problem.

3.4. Location of minima

Another important question to examine is how well the minima locations on a partial or aggregate error surface align with the minima locations on the fully informed error surface. This question is significant because a high quality weight vector found using a fixed or shifting subset of the data is only valuable when it performs similarly for all data. This can only be the case if the minima on partial and aggregate error surfaces tend to align with the minima on a fully informed error surface.

To evaluate how well these minima tend to align, high performing weight vectors found while calculating partial estimates are compared to high performing weight vectors found while calculating fully informed estimates. To determine if these two sets of weight vectors tend to occupy a similar region, the distance between each unique pair from the partial estimate is calculated. The distance between each unique pair of high performing weight vectors, with one from the partial estimate and one from the fully informed estimate, is also calculated. If both sets of weight vectors tend to fall within the same region, then there should not be a significant difference between the two sets of differences. The same process is performed for high performing weight vectors found while calculating aggregate estimates.

4. Experimental procedure

This section provides implementation details and control parameters used for the experiments performed in this study. Section 4.1 describes the datasets and the network architecture used for each. Section 4.2 describes the control parameters used for the active learning scenarios and the process with which they were selected. Section 4.3 gives details related to the calculation of error surface characteristics and the sampling algorithms used. Finally, Section 4.4 discusses the methods used to evaluate collected results.

4.1. Datasets and network architecture

The datasets used in this study are well known classification problems commonly studied in the literature with real valued inputs. The architecture used for each problem was taken from the literature. The datasets used are summarized in Table 1. For all problems, softmax was used in the output layer with the cross entropy error function, and the error is averaged over the size of the batch used to calculate it. The sigmoid function was used in the hidden layer.

Table 1: This table describes the datasets and architectures used in this study. Dimensionality refers to the total number of weights and biases.

Dataset	Inputs	Hidden	Outputs	Dimensionality
Parity	10	6 [60]	2	80
Glass [61]	9	9 [46]	6	150
Heart [62]	13	6 [63]	2	98
Wine [64]	13	10 [65]	3	173
Cancer [66]	30	10 [46]	2	332

4.2. Active learning

While performing uncertainty sampling gradient descent backpropagation was used as the training algorithm. The hyperparameters used during this process, such as the number of training iterations before each selection, learning rate, and momentum were chosen to optimize accuracy over the entire dataset using sequential model-based algorithm configuration (SMAC) [67], and are summarized in Table 2. SMAC is a state of the art tool for automatic algorithm configuration and can be used to select appropriate hyperparameters to maximize the performance of an algorithm. The default value for number of iterations provided to SMAC was 25 and the allowable values were 1 to 50. For both learning rate and momentum the default value was 0.001 and the allowable values were 0.00001 to 0.25. For each dataset SMAC had a budget of 1000 runs to select appropriate control parameter values.

Table 2: This table describes the parameters used during active learning.

Dataset	Iterations	Learning Rate	Momentum
Parity	27	6.776246E-4	2.45728484E-4
Glass	49	0.054634534	0.0180009087
Heart	50	0.0084192966	0.003189599
Wine	25	0.001	0.001
Cancer	29	0.0243824902	0.0214997517

4.3. Sampling methods and error surface characteristics

This study has made use of all four sampling methods and eleven error surface characteristics described in Section 2. For dispersion, the 5% best weight vectors of the sample were considered. Partial, aggregate, and fully informed estimates are calculated for each combination of error surface characteristic, sampling method, and problem, for both the active learning scenario and all nine random sampling scenarios. For the PG walk, PR walk, and MPR walk, this is repeated for the four different combinations of walk parameters found in Table 3. For the UR sampling method, experiments are repeated for both initialization ranges only, since step size does not apply. For UR, PR, and MPR, the initialization range also serves as the search space boundary. The initialization ranges and step sizes were chosen based on [5, 6, 46], where they were found to be suitable for these benchmark problems. Different combinations of sampling method, initialization ranges, and step sizes must be considered since different parameter combinations can allow different error surface characteristics to be identified. For each problem the number of times the sampling method is repeated, and the number of weight vectors sampled for each repetition are chosen based on the dimensionality of the problem as recommended in [48, 55, 56]. These parameters are listed in Table 4. All performed experiments have been repeated 30 times using different randomly generated seeds.

Table 3: This table describes the walk parameters used in this study.

Parameter		
Initialization Range	$[-1, 1]$	$[-10, 10]$
Step Size	Micro	Macro
% of Search Space	1%	10%

Table 4: This table describes the sampling parameters used in this study.

Dataset	Samples Taken	Weight vectors Per Sample
Parity	80	800
Glass	150	1500
Heart	98	980
Wine	173	1730
Cancer	332	3320

4.4. Methods of evaluation

The described experiments result in 550 different combinations of training scenario, error surface characteristic, and problem. Each combination is referred to as an error surface characteristic testing scenario. For each of these 550 combinations 14 different sets of partial and aggregate estimates are calculated and averaged over 30 independent runs. The 14 sets of estimates correspond to the different walk types, initialization ranges, and step sizes. The 14 sets of estimates are used in Friedman tests to check for significant differences in an error surface characteristic among the partial, aggregate, and fully informed estimates.

For the tests described in Section 3.4 the Wilcoxon rank sum test is used at a significance level of 0.05 to identify significant differences. The tests described in Section 3.4 are performed for each combination of problem, sampling method, initialization range, and step size, for both the random data sampling and active learning scenarios. For each combination, the high performing weight vectors used are the best weight vectors found by each of the 30 independent runs of the combination. Throughout this paper, when a measure of distance is required, Euclidean distance is used.

5. Empirical results

This section presents the empirical results of this work. Section 5.1 examines the consistency of the methods used for estimating landscape characteristics. Section 5.2 presents a broad overview of the results of this work focusing only on the datasets used and the different error surface characteristics estimated. Section 5.3 summarizes the results obtained for random subsampling. Section 5.4 summarizes the results obtained for active learning. Section 5.5 provides a detailed breakdown of results comparing all partial estimates to their fully informed estimates for both random subsampling and active learning, and for each sampling algorithm. Section 5.6 examines how changes in the training data may influence the location of minima in the error surface.

5.1. Characteristic estimate consistency

Figures 1, 2 and 3 display the value of error surface characteristics calculated at different sampling steps for different sampling algorithms with error calculated over all training examples. Figure 1 displays characteristic estimates obtained for the glass dataset using PG sampling initialized to $[-1, 1]$ with micro steps. Figure 2 displays characteristic estimates for the wine dataset obtained via a UR sample initialized to the range $[-10, 10]$. Figure 3 display characteristics of the parity dataset calculated from a PR sample initialized to $[-10, 10]$ using macro steps. Each run of a sampling algorithm was repeated 30 times, and the value of each error surface characteristic was calculated at 10 points distributed evenly throughout the sample. The shaded regions illustrate the standard deviation. The observed values were normalized to the range $[0, 1]$ to improve the readability of the plots.

In all three plots it can be observed that the error surface characteristic estimates generally produced a smooth curve. Most values changed more quickly early on, and began to stabilize as the sample got larger. Some values such as $M1$ and $M2$ in all figures, Grad-Ave and Grad-Std in Figure 3, or Dispersion in Figure 2 stayed relatively constant throughout the sampling. The estimates obtained from PR, MPR, and UR sampling tended to have higher standard deviation, but as can be seen in Figures 2 and 3, the standard deviation often decreased as more weight vectors were added to the sample. In contrast, PG sampling was quite consistent in the values it produced, with the standard deviation tending to be small for all error surface characteristics. Overall, for all sampling methods the expected value of the error characteristics tended to change smoothly as more weight vectors were added to the sample and did not vary chaotically over time. Many characteristics level out during the latter half of the sample. This provides evidence that the value of each characteristics can be expected to converge with a sufficiently large sample. Perhaps more importantly, this demonstrates that repeated samplings of similar length with the same sampling algorithm can be expected to produce consistent error surface characteristic values.

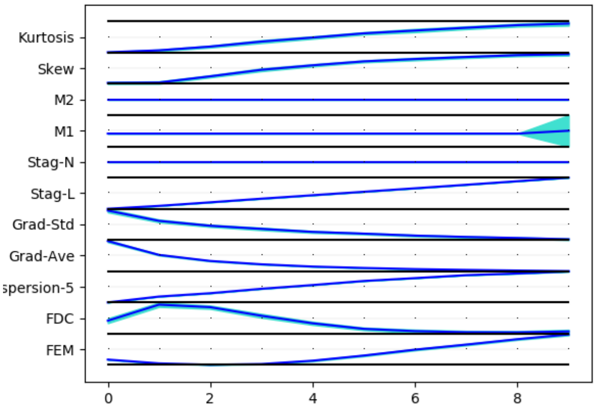

Figure 1: Error surface characteristic values calculated throughout a progressive gradient walk on the glass dataset.

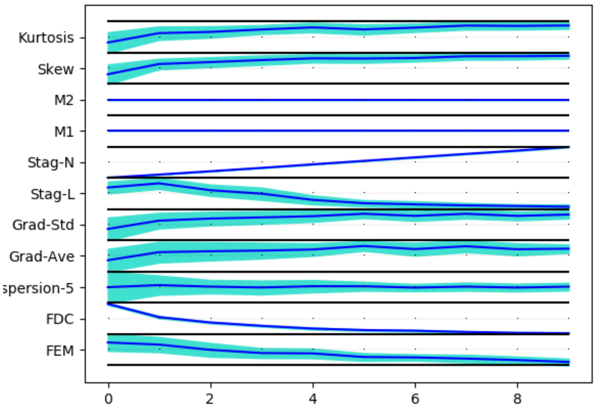

Figure 2: Error surface characteristic values calculated throughout a uniform random sample on the wine dataset.

5.2. Overview of results

The results of this study indicate that the subset of examples over which error is calculated plays a highly significant role in the characteristics of the neural network error surface. Out of 550 testing scenarios considered, 55% revealed a significant change in the error surface between different subsets of training examples (p-value < 0.05). Similar results can be observed for all problems considered, for all sampling methods tested, for both active learning and random data sampling, and the majority of error surface characteristics. These results hold whether comparing aggregate estimates to fully informed estimates or comparing partial estimates to fully informed estimates. This section summarizes these results. Table 5 presents the overall results for each problem and error surface characteristic.

Table 5: This table describes the proportion of Friedman tests which revealed at least one significant (p-value < 0.05) difference in an error surface characteristic between the partial estimates, aggregate estimate, and fully informed estimate for each dataset.

Data Subset	Dispersion-5	FDC	FEM	Grad-Ave	Grad-Std	Kurtosis	Skew	M_1	M_2	Stag-L	Stag-N	Overall
All Data	0.26667	0.63333	0.3	1	1	0.66667	0.63333	0.1	0.1	0.6	0.6	0.53636
Parity	0.3	1	0.8	1	1	1	1	0	0	0.8	0.9	0.70909
Glass	0.5	0.4	0	1	1	0.1	0.1	0	0	0	0	0.28182
Heart	0	0.5	0.1	1	1	0.9	0.8	0.3	0.3	1	0.9	0.61818
Wine	0.5	1	0.3	0.9	1	0.4	0.9	0	0	0.5	0.1	0.50909
Cancer	0.2	0.1	0.7	1	1	1	1	0.3	0	1	0.9	0.65455

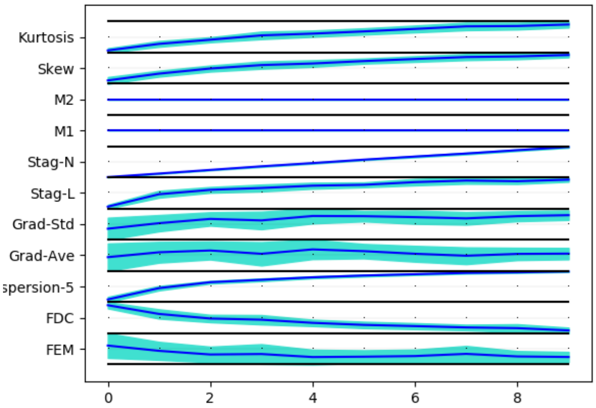

Figure 3: Error surface characteristic values calculated throughout a progressive random walk on the parity dataset.

Table 6: This table describes the proportion of Friedman tests which revealed at least one significant (p-value < 0.05) difference in an error surface characteristic between the partial estimates, aggregate estimate, and fully informed estimate for different combinations of window size and speed.

	50.00	100.0	150.0
1.00	0.618	0.509	0.473
0.67	0.636	0.618	0.436
0.33	0.600	0.564	0.455

5.3. Random subsampling summary

In the case of random sampling, it can be shown that the batch size used, and the rate at which examples change, is closely related to variability in the error surface. Table 6 gives the proportion of Friedman tests that revealed a significant difference for each of the window size and rate combinations tested. From all three diagonals of Table 6, from the upper left to lower right, it can be seen that as the window size increases and window speed decreases, the proportion of tests that reveal at least one significant difference also decreases. It is worth noting that the proportion of significant differences is well above the expected false positive rate in all cases. That is, although increasing the window size and amount of overlap did reduce the number of significant differences detected in error surface characteristics, changes in the dataset consistently produced changes in the error surface.

Figures 4, 5 and 6 display the difference between the partial estimate of an error surface characteristic and the fully informed estimate in blue compared against the standard deviation of the fully informed estimate in red for random sampling scenarios. The shaded blue region indicates the standard deviation of the partial estimate. All three figures contain examples where partial estimates do not accurately reflect the fully informed estimates, such results were common. Figure 6 displays partial estimates from random sampling with a batch size of 50 and a window speed of 0.33, meaning that there is a very high degree of overlap in the data used in adjacent partial estimates. From Figure 6 it can be seen that error surface characteristic estimates obtained from PG sampling tended to have low standard deviation. Despite the high degree of overlap in training data the partial estimates often varied dramatically. This suggests that estimates obtained from PG sampling are highly sensitive to the specific training examples used to calculate error. This is significant since PG sampling follows the gradients to areas of low error, similar to a training algorithm, and most closely reflects the error surface as it would be experience by a training algorithm. Similar results can be observed in Figure 4 which were obtained via PR sampling.

MPR, PR, and UR sampling tended to produce higher standard deviation in error surface characteristic estimates, some examples of this can be seen in Figures 4 and 5, leading to less significant variation between partial estimates and the fully informed estimate. However, clear differences between partial estimates, and between partial estimates and the fully informed estimate still exist. Some examples from Figures 5 and 4 include Grad-Ave, Grad-Std, FEM, FDC, Dispersion, Stag-N and Stag-L. For the sake of clarity it is worth noting that all partial estimates for a specific random data sampling scenario are calculated using the same number of training examples, so the amount of training data considered cannot play a role in any differences between estimates.

However, not all tested error surface characteristics responded to the different random sampling scenarios in the same way. Tables 7, 8, and 9 give the proportion of Friedman tests that revealed a significant difference for the different random sampling scenarios for Stag-L, FDC, and FEM. These three characteristics most clearly illustrate

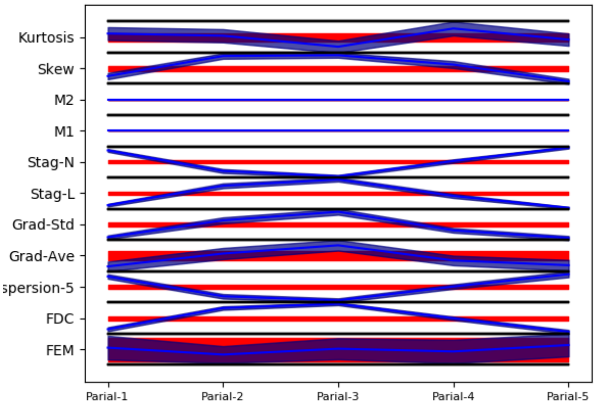

Figure 4: The value of each partial estimate compared to the fully informed estimated for random data sampling with a batch size of 150 and window speed of 0.67 on the cancer dataset. Estimates were performed using PR sampling in the range $[-1, 1]$ with micro steps.

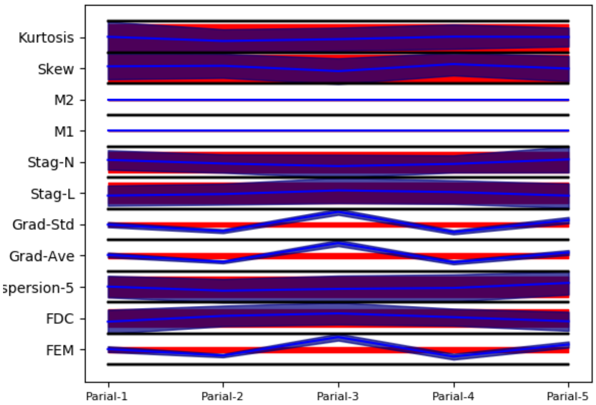

Figure 5: The value of each partial estimate compared to the fully informed estimated for random data sampling with a batch size of 50 and window speed of 0.33 on the glass dataset. Estimates were performed using MPR sampling in the range $[-1, 1]$ with macro steps.

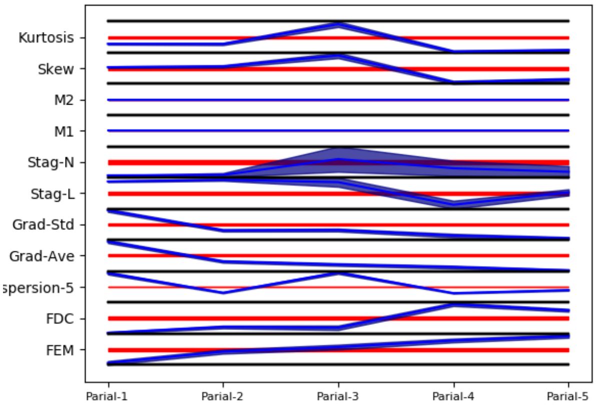

Figure 6: The value of each partial estimate compared to the fully informed estimated for random data sampling with a batch size of 50 and window speed of 0.33 on the glass dataset. Estimates were performed using PG sampling in the range $[-1, 1]$ with macro steps.

Table 7: This table describes the proportion of Friedman tests which revealed at least one significant (p-value < 0.05) difference in Stag-L between the partial estimates, aggregate estimate, and fully informed estimate for different combinations of window size and speed.

	50.00	100.0	150.0
1.00	0.800	0.400	0.600
0.67	0.800	0.800	0.400
0.33	0.800	0.600	0.600

the overall trend, with the proportion of significant differences decreasing as training batch size and the overlap between training batches increases. For all three of these error surface characteristics, batch size had a more consistent effect on the proportion of significant differences found than the amount of overlap. A larger batch size tended to reduce the total number of significant differences found. This indicates that when the batch size is too small, there can be significant changes in ruggedness, searchability, and modality.

For several error surface characteristics, the proportion of significant Friedman tests was relatively unaffected by changes in the random sampling scenario. This is illustrated by Tables 10, 11, 12, and 13, which give the proportion of Friedman tests that revealed significant differences for Kurtosis, Skew, Grad-Ave and Grad-Std. Grad-Ave and Grad-Std were notable because for both of these

Table 8: This table describes the proportion of Friedman tests which revealed at least one significant (p-value < 0.05) difference in FDC between the partial estimates, aggregate estimate, and fully informed estimate for different combinations of window size and speed.

	50.00	100.0	150.0
1.00	0.800	0.400	0.400
0.67	0.800	0.600	0.400
0.337	0.800	0.600	0.400

Table 9: This table describes the proportion of Friedman tests which revealed at least one significant (p-value < 0.05) difference in FEM between the partial estimates, aggregate estimate, and fully informed estimate for different combinations of window size and speed.

	50.00	100.0	150.0
1.00	0.400	0.200	0.000
0.67	0.800	0.400	0.200
0.33	0.600	0.400	0.400

Table 10: This table describes the proportion of Friedman tests which revealed at least one significant (p-value < 0.05) difference in Kurtosis between the partial estimates, aggregate estimate, and fully informed estimate for different combinations of window size and speed.

	50.00	100.0	150.0
1.00	0.600	0.600	0.600
0.67	0.800	0.800	0.600
0.33	0.600	0.600	0.600

characteristics, almost all tests revealed at least one significant difference between the partial, aggregate, and fully informed estimates. These results suggest that both the magnitude and standard deviation of error gradients are highly dependent on the batch of training examples. Neither increasing the batch size, nor the amount of shared data between batches prevented significant differences is these two characteristics from occurring.

5.4. Active learning summary

Similar results were obtained when examining the active learning scenario. When all Friedman tests are considered, 64% of tests indicate at least one significant difference among the partial, aggregate, and fully informed estimates. This percentage is again well above the expected false positive rate. Subsequent critical difference plots indicate that no significant differences between partial estimates and the fully informed estimate are detected for FEM and M2. For active learning, increasing the size of the training set often encouraged similarity in error surface characteristics. However, for Grad-Ave, Grad-Std, Stag-L, and Stag-N a significant difference was detected between the fully informed estimate and at least one of the five largest partial estimates. Since the five largest partial estimates are performed over at least half of the available data, this indicates that Grad-Ave, Grad-Std, Stag-L, and Stag-N are highly sensitive to the set of examples over which they are calculated.

Table 11: This table describes the proportion of Friedman tests which revealed at least one significant (p-value < 0.05) difference in Skew between the partial estimates, aggregate estimate, and fully informed estimate for different combinations of window size and speed.

	50.00	100.0	150.0
1.00	0.800	0.800	0.800
0.67	0.800	0.800	0.800
0.33	0.600	0.800	0.600

Table 12: This table describes the proportion of Friedman tests which revealed at least one significant (p-value < 0.05) difference in Grad-Ave between the partial estimates, aggregate estimate, and fully informed estimate for different combinations of window size and speed.

	50.00	100.0	150.0
1.00	1.000	1.000	1.000
0.67	1.000	1.000	0.800
0.33	1.000	1.000	1.000

Table 13: This table describes the proportion of Friedman tests which revealed at least one significant (p-value < 0.05) difference in Grad-Std between the partial estimates, aggregate estimate, and fully informed estimate for different combinations of window size and speed.

	50.00	100.0	150.0
1.00	1.000	1.000	1.000
0.67	1.000	1.000	1.000
0.33	1.000	1.000	1.000

Figure 7 displays the difference between the partial estimate of an error surface characteristic and the fully informed estimate in blue compared against the standard deviation of the fully informed estimate in red for an active learning scenario on the wine dataset. The shaded blue region indicates the standard deviation of the partial estimate. Similarly to Figures 4, 5 and 6 examples are present where partial estimates do not accurately reflect the fully informed estimates. The most dramatic differences are in Grad-Ave and Grad-Std. The estimates of these characteristics were very consistent, did not reflect the fully informed estimate and can be observed to change from one partial estimate to the next. For the other characteristics the standard deviation of the estimates in Figure 7 are relatively high as was common for UR sampling.

Figure 8 presents a critical difference plot illustrating significant differences in Skew between partial estimates, and the fully informed estimated. From Figure 8 it can be seen that Skew tended to increase as training examples were added, with the Skew of partial estimates 10, 9, 1, 6, and the fully informed estimate being significantly higher than the Skew of partial estimate 2. This suggests that the quality of weight vectors tended to improve as the size of the training set increased when active learning was used. Grad-Ave and Grad-Std exhibited similar results, with an increase in training data tending to produce smaller and more consistent gradients.

5.5. Detailed breakdown of error surface characteristic comparisons

It has been established that for both random sampling and active learning some significant difference exists between the partial, aggregate, and fully informed estimates in most cases. Pairwise testing was performed between the fully informed estimates and the corresponding partial and aggregate estimates to identify where differences were found. The results of these tests are summarized in Table 14.

Table 14: This table describes the proportion of Wilcox Rank Sum tests which revealed a significant (p-value < 0.05) difference in an error surface characteristic between the partial and aggregate estimates, and fully informed estimate for different subsets of the data. Wilcox Rank Sum testing was also performed to compare the standard deviation of the error surface characteristic between partial and aggregate estimates, and fully informed estimate. The proportion of Wilcox Rank Sum tests which revealed a significant difference in standard deviation is listed in parenthesizes.

Data Subset	Dispersion-5	FDC	FEM	Grad-Ave	Grad-Std	Kurtosis	Skew	M_1	M_2	Stag-L	Stag-N	Overall
All Partial	0.55974 (0.40455)	0.58182 (0.43474)	0.60747 (0.417532)	0.70390 (0.47792)	0.73052 (0.57338)	0.58409 (0.50844)	0.63474 (0.48539)	0.19773 (0.18766)	0.08766 (0.08506)	0.51981 (0.42305)	0.37273 (0.24350)	0.50729 (0.38557)
PG Partial	0.90795 (0.82614)	0.94091 (0.81477)	0.92159 (0.83864)	0.92841 (0.75568)	0.76023 (0.65114)	0.84773 (0.82955)	0.88182 (0.80909)	0.25114 (0.25114)	0.19432 (0.19318)	0.90682 (0.78409)	0.37500 (0.35455)	0.71963 (0.646177)
PR Partial	.55227 (0.42386)	0.54886 (0.37500)	0.41250 (0.18182)	0.59205 (0.19659)	0.69886 (0.47500)	0.45682 (0.34659)	0.58409 (0.32159)	0.00455 (0.00455)	0.00000 (0.00000)	0.49091 (0.32727)	0.52955 (0.34773)	0.44277 (0.27272)
UR Partial	0.65909 (0.04545)	0.56591 (0.39318)	0.78409 (0.19091)	0.94318 (0.55455)	0.92500 (0.63636)	0.82955 (0.65909)	0.84091 (0.57955)	0.00000 (0.00000)	0.00000 (0.00000)	0.19773 (0.08864)	0.24773 (0.01136)	0.54483 (0.28719)
MPR Partial	0.16932 (0.14318)	0.26364 (0.13523)	0.40000 (0.34545)	0.47159 (0.44318)	0.63523 (0.56250)	0.32500 (0.27386)	0.33523 (0.27841)	0.43636 (0.40114)	0.11250 (0.10455)	0.32273 (0.32500)	0.27614 (0.14432)	0.34070 (0.28698)
Init-1 Partial	0.67597 (0.49675)	0.71039 (0.51039)	0.57468 (0.43117)	0.70000 (0.45714)	0.70325 (0.52922)	0.62208 (0.52338)	0.68571 (0.53961)	0.03766 (0.03766)	0.02532 (0.02532)	0.55390 (0.44091)	0.38896 (0.25195)	0.51617 (0.38577)
Init-10 Partial	0.44351 (0.31234)	0.45325 (0.35909)	0.64026 (0.40390)	0.70779 (0.49870)	0.75779 (0.61753)	0.54610 (0.49351)	0.58377 (0.43117)	.35779 (0.33766)	0.15000 (0.14481)	0.48571 (0.40519)	0.35649 (0.23506)	0.49841 (0.38536)
Micro Step Partial	0.47879 (0.42727)	0.51364 (0.35152)	0.48106 (0.42576)	0.60227 (0.45303)	0.58485 (0.55076)	0.48258 (0.43258)	0.52045 (0.41212)	0.24848 (0.23636)	0.13333 (0.13106)	0.48712 (0.44091)	0.21136 (0.13182)	0.43127 (0.36302)
Macro Step Partial	0.60758 (0.50152)	0.65530 (0.53182)	0.67500 (0.48485)	0.72576 (0.47727)	0.81136 (0.57500)	0.60379 (0.53409)	0.68030 (0.52727)	0.21288 (0.20152)	0.07121 (0.06742)	0.65985 (0.51667)	0.57576 (0.43258)	0.57080 (0.44091)
All Aggregate	0.72714 (0.55286)	0.67571 (0.55286)	0.68714 (0.49143)	.90714 (0.62857)	0.95714 (0.82000)	0.81571 (0.66857)	0.76143 (0.67571)	0.25714 (0.24286)	0.12000 (0.11429)	0.73143 (0.56143)	0.72143 (0.57000)	0.66922 (0.53442)
PG Aggregate	0.97500 (0.87000)	0.97000 (0.86000)	0.96000 (0.87000)	0.95000 (0.79000)	0.96000 (0.83000)	0.95500 (0.86000)	0.98000 (0.86000)	0.35000 (0.34500)	0.24500 (0.24000)	0.98500 (0.86500)	0.86500 (0.82500)	0.83591 (0.74682)
PR Aggregate	0.76000 (0.62500)	0.42500 (0.35500)	0.60500 (0.44000)	0.80000 (0.30000)	0.90500 (0.73000)	0.73500 (0.51500)	0.69000 (0.49500)	0.02500 (0.02500)	0.00000 (0.00000)	0.58000 (0.32500)	0.64500 (0.52000)	0.56091 (0.39364)
UR Aggregate	0.65000 (0.05000)	0.50000 (0.23000)	0.72000 (0.05000)	0.99000 (0.49000)	0.97000 (0.62000)	0.86000 (0.70000)	0.82000 (0.66000)	0.00000 (0.00000)	0.00000 (0.00000)	0.50000 (0.20000)	0.60000 (0.13000)	0.60091 (0.28455)
MPR Aggregate	0.48500 (0.41500)	0.72000 (0.60500)	0.48000 (0.38500)	0.93000 (0.86500)	1.00000 (1.00000)	0.73500 (0.61500)	0.58500 (0.68000)	0.52500 (0.48000)	0.17500 (0.16000)	0.74500 (0.67500)	0.71500 (0.58500)	0.64500 (0.58773)
Init-1 Aggregate	0.78571 (0.60571)	0.76000 (0.60857)	0.70571 (0.52571)	0.91143 (0.67143)	0.96286 (0.84571)	0.83714 (0.67429)	0.74571 (0.68286)	0.09714 (0.09714)	0.05143 (0.05143)	0.80000 (0.61714)	0.76571 (0.59429)	0.67481 (0.54312)
Init-10 Aggregate	0.66857 (0.50000)	0.59143 (0.49714)	0.66857 (0.45714)	0.90286 (0.58571)	0.95143 (0.79429)	0.79429 (0.66286)	0.77714 (0.66857)	0.41714 (0.38857)	0.18857 (0.17714)	0.66286 (0.50571)	0.67714 (0.54571)	0.66364 (0.52571)
Micro Step Aggregate	0.78333 (0.68333)	0.83000 (0.73667)	0.64667 (0.61667)	0.91667 (0.71667)	0.98667 (0.93000)	0.91667 (0.73333)	0.76333 (0.78000)	0.32000 (0.30000)	0.16667 (0.16000)	0.82000 (0.69000)	0.77000 (0.76667)	0.72000 (0.64667)
Macro Step Aggregate	0.69667 (0.59000)	0.58000 (0.47667)	0.71667 (0.51333)	0.87000 (0.58667)	0.92333 (0.77667)	0.70000 (0.59333)	0.74000 (0.57667)	0.28000 (0.26667)	0.11333 (0.10667)	0.72000 (0.55333)	0.71333 (0.52000)	0.64121 (0.50545)

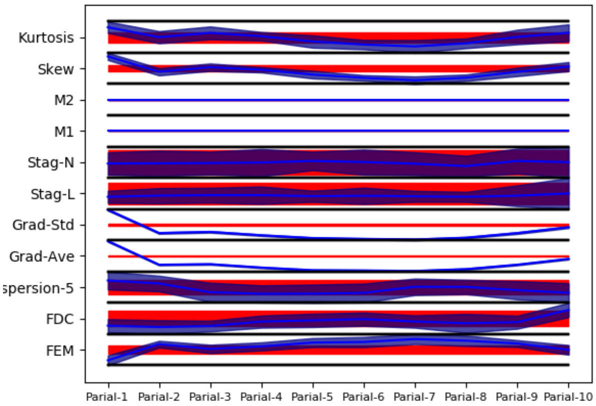

Figure 7: The value of each partial estimate compared to the fully informed estimated for active learning on the wine dataset. Estimates were performed using UR sampling in the range $[-1, 1]$.

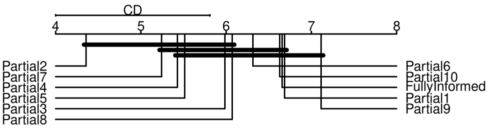

Figure 8: This figure presents a critical difference plot comparing partial and fully informed estimates of Skew when active learning is used. Estimates are listed from lowest to highest, left to right. A black bar connecting estimates indicates no significant difference detected.

From Table 14, it can be seen that most partial estimates, and the majority of aggregate estimates are significantly different from the fully informed estimates. Furthermore, for the majority of cases, the rate at which a significant difference is detected is well above the expected false positive rate for a p-value < 0.05. This is true for the error surface characteristic's value, and the standard deviation of the error surface characteristic. This suggests that a subset of the available data rarely provides a reliable estimate of the error surface characteristics of the entire dataset. The PG walk in particular revealed a high proportion of significant differences for both partial and aggregate estimates. This may be related to the unbounded nature of the walk. Furthermore, since the PG walk actively follows the gradients of the error surface. It is biased towards regions containing strong weight vectors. For this reason, the PG walk is likely to evaluate regions which a (gradient based) training algorithm may traverse.

5.6. Location of minima

Finally, testing for a difference in the location of minima between partial error surfaces and fully informed error surfaces yielded a significant difference in 48% of tests. For aggregate error surfaces, 51% of tests found a significant difference. That is, in approximately half of all tests, the distance between a pair of high performing weight vectors found on a partial (or aggregate) error surface is significantly different from the distance between a high performing weight vector from the partial (or aggregate) error surface and a high performing weight vector from the fully informed error surface. This suggests that minima on a partial or aggregate error surface may not be a reliable predictor of minima on the fully informed error surface. Similar statistics are obtained for each separate type of walk, each initialization range, each step size, and both active learning and random data sampling.

Of particular interest is the progressive gradient walk, which is unbounded and actively attempts to find high performing weight vectors. Since uniform random sampling, the progressive random walk, and the Manhattan progressive random walk are bounded and do not attempt to find minima, it may be misleading to investigate the distance between high performing weight vectors found by these sampling methods. The best weight vectors found by these sampling methods during a run may not accurately reflect the location of a minimum since no effort was made to reduce the error of generated weight vectors. The progressive gradient walk avoids this issue and provides a reasonable approximation of the behavior of a search algorithm.

When only progressive gradient walks are considered, then testing for a difference in the location of minima between partial error surfaces and fully informed error surfaces yielded a significant difference in 82% of tests. For aggregate error surfaces, a significant difference is detected in 81.5% of tests. In the performed experiments, when actively searching for high performing weight vectors the locations of minima on partial and aggregate error surfaces did not reliably predict the location of minima on the fully informed error surface.

6. Conclusions

The main purpose of this study was to quantitatively evaluate how changes in the data over which error is calculated may influence the neural network error surface. Extensive experimentation using five common benchmark problems and ten different training scenarios was performed. For each combination of training scenario and problem, partial, aggregate, and fully informed estimates were generated for eleven different error surface characteristics using fourteen different sampling configurations. This experimentation enabled an empirical evaluation of how various changes in the set of examples over which error is calculated influence some error surface characteristics.

For most error surface characteristics the majority of performed tests indicated a significant difference between the partial estimates and fully informed estimate, as well as between the aggregate estimates and fully informed estimates. Characteristics such as Kurtosis, Skew, Grad-Ave, Grad-Std, Stag-L, and Stag-N were particularly sensitive to the data used to estimate them, even with relatively large batch sizes and high degrees of overlap. This suggests that certain aspects of structure, modality, and searchability are highly sensitive to the examples used to calculate error. As a result, it may not be possible to reliably estimate such characteristics using only a subset of the possible data. It was also common for the spread of error surface characteristics to change significantly, suggesting that changing the examples over which an error surface characteristic is calculated can affect both the value and distribution of the measure. The results suggest that the error surface can become rather chaotic as the data over which error is calculated changes.

Evidence was also presented indicating that high performing weight vectors found by a sampling procedure using a fixed subset of the data do not occupy the same region as high performing weight vectors found by a sampling procedure operating on all available data. This was particularly evident when only progressive gradient walks, which actively seek out high performing weight vectors, were considered. This suggests that, for at least some problems, different subsets of data will produce minima at different locations, and that the minima of a random subset of the data does not reliably predict the location of true minima. This observation held for aggregate estimates as well, which use all available data by evaluating different weight vectors using different subsets of examples. It is concluded from the findings that training on changing subsamples of the available data is a dynamic optimization problem, opening avenues for future research.

Future work should investigate the use of dynamic optimization algorithms for neural network training. Existing dynamic global optimization algorithms could be modified to exploit the gradient information available during neural network training. Such algorithms may offer improved performance relative to gradient descent backpropagation, given that changes in error surface characteristics, including the location of minima, have been observed. Future work should also use fitness landscape analysis to examine online learning approaches, additional active learning approaches, and the impact of these on the error surface of deep neural networks.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] C. M. Bishop, et al., Neural networks for pattern recognition, Oxford university press, 1995.

[2] G. Dreyfus, Neural networks: methodology and applications, Springer Science & Business Media, 2005.

[3] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, P. T. P. Tang, On large-batch training for deep learning: Generalization gap and sharp minima, in: International Conference on Learning Representations, 2017, pp. 1–16.

[4] A. Choromanska, Y. LeCun, G. B. Arous, Open problem: The landscape of the loss surfaces of multilayer networks, in: Conference on Learning Theory, 2015, pp. 1756–1760.

[5] A. Bosman, A. Engelbrecht, M. Helbig, Visualising basins of attraction for the cross-entropy and the squared error neural network loss functions, Neurocomputing (2020) 113–136.

[6] A. Bosman, A. Engelbrecht, M. Helbig, Progressive gradient walk for neural network fitness landscape analysis, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2018, pp. 1473–1480.

[7] L. Sagun, U. Evci, V. U. Guney, Y. Dauphin, L. Bottou, Empirical analysis of the hessian of over-parametrized neural networks, in: International Conference on Learning Representations, 2018, pp. 1–15.

[8] Y. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, Y. Bengio, Identifying and attacking the saddle point problem in high-dimensional non-convex optimization, in: Advances in neural information processing systems, 2014, pp. 2933–2941.

[9] F. Draxler, K. Veschgini, M. Salmhofer, F. Hamprecht, Essentially no barriers in neural network energy landscape, in: Proceedings of the 35th International Conference on Machine Learning, Vol. 80, 2018, pp. 1309–1318.

[10] J. Denker, D. Schwartz, B. Wittner, S. Solla, R. Howard, L. Jackel, J. Hopfield, Large automatic learning, rule extraction, and generalization, Complex systems 1 (5) (1987) 877–922.

[11] A. Chen, H. Lu, R. Hecht-Nielsen, On the geometry of feedforward neural network error surfaces, Neural computation 5 (6) (1993) 910–927.

[12] A. Bosman, A. Engelbrecht, M. Helbig, Fitness landscape analysis of weight-elimination neural networks, Neural Processing Letters 48 (1) (2018) 353–373.

[13] A. Bosman, A. Engelbrecht, M. Helbig, Loss surface modality of feed-forward neural network architectures, in: International Joint Conference on Neural Networks, 2020, pp. 1–8.

[14] V.-L. Nguyen, S. Destercke, E. Hüllermeier, Epistemic uncertainty sampling, in: International Conference on Discovery Science, Springer, 2019, pp. 72–86.

[15] L. Sun, X. Zhang, Y. Qian, J. Xu, S. Zhang, Feature selection using neighborhood entropy-based uncertainty measures for gene expression data classification, Information Sciences 502 (2019) 18–41.

[16] C. Mayer, R. Timofte, Adversarial sampling for active learning, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2020, pp. 3071–3079.

[17] H. Shen, Towards a mathematical understanding of the difficulty in learning with feedforward neural networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 811–820.

[18] D. Mehta, X. Zhao, E. A. Bernal, D. J. Wales, Loss surface of xor artificial neural networks, Physical Review E 97 (2018) 1–12.

[19] P. Cheridito, A. Jentzen, F. Rossmannek, Landscape analysis for shallow relu neural networks: complete classification of critical points for affine target functions, arXiv preprint arXiv:2103.10922.

[20] B. Liu, Spurious local minima are common for deep neural networks with piecewise linear activations, arXiv preprint arXiv:2102.13233.

[21] S. Barannikov, G. Sotnikov, I. Trofimov, A. Korotin, E. Burnaev, Topological obstructions in neural networks learning, arXiv preprint arXiv:2012.15834.

[22] D. Mehta, T. Chen, T. Tang, J. Hauenstein, The loss surface of deep linear networks viewed through the algebraic geometry lens, IEEE Transactions on Pattern Analysis and Machine Intelligence.

[23] N. Vesseron, I. Redko, C. Laclau, Deep neural networks are congestion games: From loss landscape to wardrop equilibrium and beyond, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2021, pp. 1765–1773.

[24] D. Kafka, D. N. Wilke, Resolving learning rates adaptively by locating stochastic non-negative associated gradient projection points using line searches, Journal of Global Optimization 79 (1) (2021) 111–152.

[25] D. Granziol, S. Zohren, S. Roberts, Learning rates as a function of batch size: A random matrix theory approach to neural network training, arXiv preprint arXiv:2006.09092.

[26] S. Liang, R. Sun, Y. Li, R. Srikant, Understanding the loss surface of neural networks for binary classification, in: International Conference on Machine Learning, PMLR, 2018, pp. 2835–2843.

[27] P. Chaudhari, A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, C. Borgs, J. Chayes, L. Sagun, R. Zecchina, Entropy-sgd: Biasing gradient descent into wide valleys, Journal of Statistical Mechanics: Theory and Experiment 2019 (12) (2019) 1–24.

[28] K. Hornik, M. Stinchcombe, H. White, et al., Multilayer feedforward networks are universal approximators., Neural networks 2 (5) (1989) 359–366.

[29] P. J. Werbos, Beyond regression: New tools for prediction and analysis in the behavioural sciences, Ph.D. thesis, Harvard University (1974).

[30] X. Xiao, Z. Wang, Q. Li, S. Xia, Y. Jiang, Back-propagation neural network on markov chains from system call sequences: a new approach for detecting android malware with system call sequences, IET Information Security 11 (1) (2017) 8–15.

[31] M. A. Hazza, E. Adesta, Investigation of the effect of cutting speed on the surface roughness parameters in cnc end milling using artificial neural network, in: IOP Conference Series: Materials Science and Engineering, Vol. 53, IOP Publishing, 2013, pp. 1–12.

[32] P. Golik, P. Doetsch, H. Ney, Cross-entropy vs. squared error training: a theoretical and experimental comparison., in: Interspeech, Vol. 13, 2013, pp. 1756–1760.

[33] A. S. Rakitianskaia, A. P. Engelbrecht, Training feedforward neural networks with dynamic particle swarm optimisation, Swarm Intelligence 6 (3) (2012) 233–270.

[34] M. Hasenjäger, H. Ritter, Active learning in neural networks, in: New learning paradigms in soft computing, Springer, 2002, pp. 137–169.

[35] S. Wright, The roles of mutation, inbreeding, crossbreeding, and selection in evolution, in: International Congress of Genetics, 1932, pp. 356–366.

[36] T. Jones, et al., Evolutionary algorithms, fitness landscapes and search, Ph.D. thesis, Citeseer (1995).

[37] P. Stadler, Fitness landscapes, in: Biological evolution and statistical physics, Springer, 2002, pp. 183–204.

[38] P. Kerschke, H. Trautmann, Comprehensive feature-based landscape analysis of continuous and constrained optimization problems using the r-package flacco, in: Applications in Statistical Computing, Springer, 2019, pp. 93–123.

[39] B. Derbel, A. Liefooghe, S. Verel, H. Aguirre, K. Tanaka, New features for continuous exploratory landscape analysis based on the soo tree, in: Proceedings of the 15th ACM/SIGEVO Conference on Foundations of Genetic Algorithms, 2019, pp. 72–86.

[40] A. Janković, C. Doerr, Adaptive landscape analysis, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2019, pp. 2032–2035.

[41] Y. Sun, M. Kirley, S. K. Halgamuge, Quantifying variable interactions in continuous optimization problems, IEEE Transactions on Evolutionary Computation 21 (2) (2016) 249–264.

[42] R. W. Garden, A. P. Engelbrecht, Analysis and classification of optimisation benchmark functions and benchmark suites, in: 2014 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2014, pp. 1641–1649.

[43] P. Kerschke, H. H. Hoos, F. Neumann, H. Trautmann, Automated algorithm selection: Survey and perspectives, Evolutionary computation 27 (1) (2019) 3–45.

[44] K. R. Harrison, B. M. Ombuki-Berman, A. P. Engelbrecht, A parameter-free particle swarm optimization algorithm using performance classifiers, Information Sciences 503 (2019) 381–400.

[45] P. R. Bosman, The influence of fitness landscape characteristics on the search behaviour of particle swarm optimisers, Ph.D. thesis, University of Pretoria (2019).

[46] A. S. Bosman, Fitness landscape analysis of feed-forward neural networks, Ph.D. thesis, University of Pretoria (2019).

[47] K. Malan, A. Engelbrecht, Characterising the searchability of continuous optimisation problems for PSO, Swarm Intelligence 8 (4) (2014) 275–302.

[48] K. Malan, A. Engelbrecht, Ruggedness, funnels and gradients in fitness landscapes and the effect on PSO performance, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, 2013, pp. 963–970.

[49] M. Lunacek, D. Whitley, The dispersion metric and the cma evolution strategy, in: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, 2006, pp. 477–484.

[50] T. Jones, S. Forrest, Fitness distance correlation as a measure of problem difficulty for genetic algorithms, in: Proceedings of the 6th International Conference on Genetic Algorithms, 1995, pp. 184–192.

[51] K. M. Malan, A. P. Engelbrecht, Characterising the searchability of continuous optimisation problems for PSO, Swarm Intelligence 8 (4) (2014) 275–302.

[52] K. M. Malan, A. P. Engelbrecht, Quantifying ruggedness of continuous landscapes using entropy, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, 2009, pp. 1440–1447.

[53] W. A. van Aardt, A. S. Bosman, K. M. Malan, Characterising neutrality in neural network error landscapes, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, 2017, pp. 1374–1381.

[54] O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, G. Rudolph, Exploratory landscape analysis, in: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, 2011, pp. 829–836.

[55] R. Lang, A. Engelbrecht, On the robustness of random walks for fitness landscape analysis, in: Proceedings of the IEEE Symposium Series on Computational Intelligence, IEEE, 2019, pp. 1898–1906.

[56] K. M. Malan, A. P. Engelbrecht, A progressive random walk algorithm for sampling continuous fitness landscapes, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, 2014, pp. 2507–2514.

[57] K. Malan, A. Engelbrecht, Steep gradients as a predictor of PSO failure, in: Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation, 2013, pp. 9–10.

[58] D. D. Lewis, W. A. Gale, A sequential algorithm for training text classifiers, in: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, Springer, 1994, pp. 3–12.

[59] M. Sharma, M. Bilgic, Evidence-based uncertainty sampling for active learning, Data Mining and Knowledge Discovery 31 (1) (2017) 164–202.

[60] D. Liu, T.-S. Chang, Y. Zhang, A new learning algorithm for feedforward neural networks, in: Proceeding of the IEEE International Symposium on Intelligent Control, IEEE, 2001, pp. 39–44.

[61] D. Dua, C. Graff, Uci machine learning repository., (accessed 13 Jan 2021) (2017).
URL `http://archive.ics.uci.edu/ml`

[62] A. Janosi, W. Steinbrunn, M. Pfisterer, R. Detrano, Heart disease dataset., (accessed 13 Jan 2021) (1989).
URL `https://archive.ics.uci.edu/ml/datasets/heart+disease`

[63] A. H. Chen, S.-Y. Huang, P.-S. Hong, C.-H. Cheng, E.-J. Lin, Hdps: Heart disease prediction system, in: Proceedings of the IEEE Conference on Computing in Cardiology, IEEE, 2011, pp. 557–560.

[64] M. Forina, et al., Wine data set., (accessed 13 Jan 2021) (1991).
URL `https://archive.ics.uci.edu/ml/datasets/Wine`

[65] A. B. van Wyk, A. P. Engelbrecht, Overfitting by PSO trained feedforward neural networks, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, 2010, pp. 1–8.

[66] W. Wolberg, Breast cancer wisconsin (original) data set., (accessed 13 Jan 2021) (1990).
URL `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29`

[67] F. Hutter, H. H. Hoos, K. Leyton-Brown, Sequential model-based optimization for general algorithm configuration, in: C. A. Coello (Ed.), Learning and Intelligent Optimization, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 507–523.