Proceedings of the International MultiConference of Engineers and Computer Scientists 2012 Vol I,
IMECS 2012, March 14 - 16, 2012, Hong Kong

Eigen Vector Descent and Line Search for Multilayer Perceptron

Seiya Satoh and Ryohei Nakano

***Abstract*—As learning methods of a multilayer perceptron (MLP), we have the BP algorithm, Newton's method, quasi-Newton method, and so on. However, since the MLP search space is full of crevasse-like forms having a huge condition number, it is unlikely for such usual existing methods to perform efficient search in the space. This paper proposes a new search method which utilizes eigen vector descent and line search, aiming to stably find excellent solutions in such an extraordinary search space. The proposed method is evaluated with promising results through our experiments for MLPs having a sigmoidal or exponential activation function.**

***Index Terms*—multilayer perceptron, polynomial network, singular region, search method, line search**

I. Introduction

The search space of a multilayer perceptron (MLP) is reported to be full of crevasse-like forms such as crevasse-local-minima or a crevasse-gutter having a huge condition number ($10^6 \sim 10^{17}$) [4]. In such an extraordinary space, it will be hard for usual existing search methods to find excellent solutions.

In MLP learning, the BP algorithm is well known as a first-order method, but its learning is usually very slow and will get stuck in crevasse-like forms. Second-order methods, such as Newton's method and quasi-Newton method, can converge much faster into a local minimum or gutter; however, it will not be easy even for them to find a descending route once they get stuck in a gutter [4].

Recently a new search method [3] has been invented, which directly positions hidden units within input space by numerically analyzing the curvature of the output surface.

This paper proposes another new search method which utilizes eigen vector descent and line search, aiming to stably find excellent solutions in such an extraordinary search space full of crevasse-like forms. Our experiments using sigmoidal MLP and polynomial-type MLP showed that the proposed method worked well for artificial and real data.

II. Background

Consider a multilayer perceptron (MLP) having J hidden units and one output unit. The MLP output f for the μ-th data point is calculated as below.

$$f(\boldsymbol{x}^\mu;\boldsymbol{w}) = w_0 + \sum_{j=1}^{J} w_j z_j^\mu, \quad z_j^\mu \equiv g(\boldsymbol{w}_j^T \boldsymbol{x}^\mu) \tag{1}$$

Here $\boldsymbol{w} = \{w_0, w_j, \boldsymbol{w}_j, j = 1, \cdots, J\}$ denotes a weight vector. Given data $\{(\boldsymbol{x}^\mu, y^\mu), \mu = 1, \cdots, N\}$, we want to find the weight vector that minimizes sum-of-squares error E shown below.

$$E = \frac{1}{2}\sum_{\mu=1}^{N}(f^\mu - y^\mu)^2, \quad f^\mu \equiv f(\boldsymbol{x}^\mu;\boldsymbol{w}) \tag{2}$$

A. Existing Search Methods

(1) BP algorithm and Coordinate Descent

The batch BP algorithm is a steepest descent method, and updates weights using the following equations. Here η_t is a learning rate at time t, and $\boldsymbol{g}(\boldsymbol{w})$ denotes a gradient at $\boldsymbol{w}$.

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \boldsymbol{g}_t, \quad \boldsymbol{g}_t \equiv \boldsymbol{g}(\boldsymbol{w}_t), \quad \boldsymbol{g}(\boldsymbol{w}) \equiv \frac{\partial E}{\partial \boldsymbol{w}} \tag{3}$$

The so-called BP algorithm is a stochastic descent method which updates weights each time a data point is given. Both BP algorithms are usually very slow and likely to get stuck at points which are not so good.

Coordinate descent method only changes a single weight with the remaining weights fixed. Since the coordinates of weights are orthogonal, the method selects the descent direction among the orthogonal candidates. There are several ways of selecting a suitable coordinate [2].

(2) Newton's method

The idea behind Newton's method is that the target function is approximated locally by a quadratic function, and the method usually converges much faster than first-order methods stated above. The method updates weights using the following equations.

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \boldsymbol{H}_t^{-1}\boldsymbol{g}_t, \boldsymbol{H}_t \equiv \boldsymbol{H}(\boldsymbol{w}_t), \boldsymbol{H}(\boldsymbol{w}) \equiv \frac{\partial^2 E}{\partial \boldsymbol{w}^T \partial \boldsymbol{w}} \tag{4}$$

Here $\boldsymbol{H}(\boldsymbol{w})$ denotes the Hessian matrix at $\boldsymbol{w}$. At a strict local minimum $\boldsymbol{w}^*$ the Hessian matrix $\boldsymbol{H}(\boldsymbol{w}^*)$ is positive definite; however, in a crevasse-like gutter having a huge condition number the positive definiteness and the search direction are in a precarious condition. Moreover, the additional cost of computing the Hessian matrix is required, and Newton's method may head for a local maximum.

(3) quasi-Newton methods and BPQ

Quasi-Newton methods employ an approximation to the inverse Hessian in place of the true required in Newton's method. The approximation can be built up by using a series of gradients obtained through a learning process. Let $\boldsymbol{G}_t$ be an approximation to the inverse Hessian, then the search direction $\boldsymbol{d}_t$ is given as below.

$$\boldsymbol{d}_t = -\boldsymbol{G}_t \boldsymbol{g}_t \tag{5}$$

Once the search direction is determined, a step length how far the search point should be moved is determined by line

This work was supported in part by Grants-in-Aid for Scientific Research (C) 22500212 and Chubu University Grant 22IS27A.

S. Satoh, and R. Nakano are with the Department of Computer Science, Graduate School of Engineering, Chubu University, 1200 Matsumoto-cho, Kasugai 487-8501, Japan email: tp11015-0513@sti.chubu.ac.jp, and nakano@cs.chubu.ac.jp

search. There are a number of techniques for performing line search [2]. BPQ algorithm [5] employs the BFGS update for calculating the inverse Hessian and the second-order approximation for calculating a step length.

B. Properties of MLP Search Space

(1) local minimum, wok-bottom, gutter, and crevasse

Here we review a critical point where the gradient $\partial E/\partial \boldsymbol{w}$ of a target function $E(\boldsymbol{w})$ gets zero. In the context of minimization, a critical point is classified into a local minimum and a saddle. A critical point $\boldsymbol{w}_0$ is classified as a local minimum when any point $\boldsymbol{w}$ in its neighborhood satisfies $E(\boldsymbol{w}_0) \leq E(\boldsymbol{w})$, otherwise is classified as a saddle.

Now we classify a local minimum into a wok-bottom and a gutter [4]. A wok-bottom $\boldsymbol{w}_0$ is a strict local minimum where any point $\boldsymbol{w}$ in its neighborhood satisfies $E(\boldsymbol{w}_0) < E(\boldsymbol{w})$, and a gutter is a set of points connected to each other in the form of a continuous subspace where any two points $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ in a gutter satisfy $E(\boldsymbol{w}_1) = E(\boldsymbol{w}_2)$ or $E(\boldsymbol{w}_1) \approx E(\boldsymbol{w}_2)$.

When a point has a huge condition number (say, more than 10^5), we say the point is in a crevasse. A wok-bottom and a gutter in a crevasse are called a crevasse-wok-bottom and a crevasse-gutter respectively.

(2) singular region

One of the major characteristics of MLP search space is the existence of singular regions [1], [4], [8]. Here MLP(J) denotes an MLP having J hidden units. A singular region is defined as a search subspace where the gradient is equal to zero ($\partial E/\partial \boldsymbol{w} = \boldsymbol{0}$) and the Hessian matrix is not positive definite with at least one eigen value equal to zero. Thus, a singular region is a flat subspace, and points in and around a singular region have a huge condition number.

How is a singular region created? It is known that among MLPs the following causes three types of reducibility [7].

a) $w_j = 0$
b) $\boldsymbol{w}_j = (w_{j0}, 0, \cdots, 0)^T$
c) $\boldsymbol{w}_{j_1} = \boldsymbol{w}_{j_2}$

Based on the above reducibility, a singular region in the search space of MLP(J) is generated by applying the following reducible mapping to a local minimum $\widehat{\boldsymbol{w}}_{J-1} = \{\widehat{u}_0, \widehat{u}_j, \widehat{\boldsymbol{u}}_j, j = 2, ..., J\}$ of MLP($J-1$) [1], [4].

$$\widehat{\boldsymbol{\theta}}_{J-1} \xrightarrow{\alpha} \widehat{\boldsymbol{\Theta}}_J^{\alpha}, \quad \widehat{\boldsymbol{\theta}}_{J-1} \xrightarrow{\beta} \widehat{\boldsymbol{\Theta}}_J^{\beta}, \quad \widehat{\boldsymbol{\theta}}_{J-1} \xrightarrow{\gamma} \widehat{\boldsymbol{\Theta}}_J^{\gamma}$$

$$\widehat{\boldsymbol{\Theta}}_J^{\alpha} \equiv \{\boldsymbol{\theta}_J | \; w_0 = \widehat{u}_0, \; w_1 = 0, \; w_j = \widehat{u}_j, \boldsymbol{w}_j = \widehat{\boldsymbol{u}}_j, j{=}2,\cdots,J\} \quad (6)$$

$$\widehat{\boldsymbol{\Theta}}_J^{\beta} \equiv \{\boldsymbol{\theta}_J | \; w_0 + w_1 g(w_{10}) = \widehat{u}_0, \; \boldsymbol{w}_1 {=} [w_{10}, 0, \cdots, 0]^T, \; w_j = \widehat{u}_j, \boldsymbol{w}_j = \widehat{\boldsymbol{u}}_j, j{=}2,\cdots,J\} \quad (7)$$

$$\widehat{\boldsymbol{\Theta}}_J^{\gamma} \equiv \{\boldsymbol{\theta}_J | \; w_0 = \widehat{u}_0, \; w_1 + w_2 = \widehat{u}_2, \; \boldsymbol{w}_1 {=} \boldsymbol{w}_2 {=} \widehat{\boldsymbol{u}}_2, \; w_j = \widehat{u}_j, \boldsymbol{w}_j = \widehat{\boldsymbol{u}}_j, j{=}3,\cdots,J\} \quad (8)$$

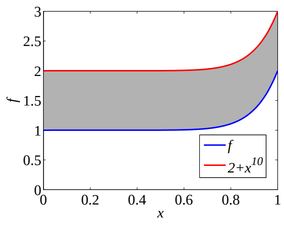
(3) another cause of a huge condition number

As stated above, points in and around a singular region have a huge condition number, which causes stagnation of learning. Moreover, there exists another aspect why MLP search space has points having a huge condition number.

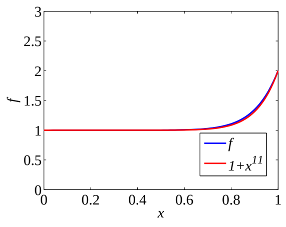
Consider the following simple MLP.

$$f = w_0 + x^{w_{11}} \quad (9)$$

Let the range of x be $(0, \; 1)$, and the current values of weights be as $w_0 = 1$ and $w_{11} = 1$. When the value of w_0 is incremented by one, the MLP output changes widely as shown in Fig. 1 (a). Meanwhile, when the value of w_{11} is similarly incremented by one, the MLP output hardly changes as shown in Fig. 1 (b). As shown in this simple example, the influence over the MLP output differs widely among weights, which may generate points having a huge condition number.



(a) w_0 is changed from 0 to 1



(b) w_{11} is changed from 0 to 1

Fig. 1. How the change of a weight influences the output of MLP.

III. Eigen Vector Descent

In a crevasse-gutter usual existing methods will not proceed along the bottom of a gutter, but will try to go down heading for the opposite steep wall, and finally will get stuck around the bottom of the gutter.

As a robust search method which can proceed along the bottom of a crevasse-gutter, we propose a new method which utilizes a set of eigen vectors to find a desirable search direction even in a crevasse-gutter.

Let $\boldsymbol{H}$, λ_m, and $\boldsymbol{v}_m$ be the Hessian matrix, its eigen value, and the corresponding eigen vector, respectively.

$$\boldsymbol{H}\boldsymbol{v}_m = \lambda_m \boldsymbol{v}_m \quad (10)$$

Note that eigen vectors are orthogonal to each other if their eigen values are different. Then, we can expect that one of eigen vectors is almost parallel to the bottom of a crevasse-gutter as shown in Fig. 2.

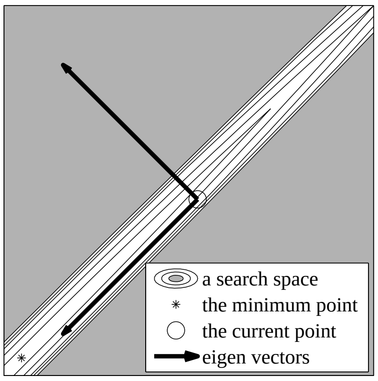


Fig. 2. Eigen vectors in a search space having a huge condition number

Now that the candidates of the search direction are determined, we consider line search which determines an adequate step length how far the search point should be moved. There are many techniques for line search [2], and the second-order Taylor expansion is sometimes used as line search by curve fitting. Here we consider the third-order approximation as well to deal with a negative curvature.

Let $\boldsymbol{v}_{m,t}$ be the m-th eigen vector of the Hessian matrix at time t; then, the target function E at $\boldsymbol{w}_t$ along the direction $\boldsymbol{v}_{m,t}$ is expressed as $\psi_t(\eta) = E(\boldsymbol{w}_t + \eta\ \boldsymbol{v}_{m,t})$. The third-order Taylor expansion is shown as follows.

$$\psi_t(\eta) \approx \psi_t(0) + \psi_t'(0)\eta + \frac{1}{2}\psi_t''(0)\eta^2 + \frac{1}{6}\psi_t'''(0)\eta^3 \qquad (11)$$

The solution η_t which minimizes the above $\psi_t(\eta)$ can be easily obtained by solving the quadratic equation. If we have two solutions positive and negative, we select the positive one. When both solutions are positive, we select the smaller one. Moreover, if we don't have a positive solution, we employ the second-order Taylor expansion.

Now, we describe the proposed search method, which is called *EVD (eigen vector descent)*. EVD repeats the following basic search cycle until convergence. The basic cycle at time t is shown below.

Basic cycle of EVD :
(step 1) Calculate the Hessian matrix and get all the eigen vectors $\{\boldsymbol{v}_{m,t},\ m = 1, \cdots, M\}$.
(step 2) Calculate repeatedly the adequate step length $\eta_{m,t}$ for each eigen vector $\boldsymbol{v}_{m,t}$ by using the second- or third-order Taylor expansion.
(step 3) Update weights using the suitable search direction $\boldsymbol{d}_t^*$ and its step length η_t^*.

$$\boldsymbol{w}_{t+1} \quad \longleftarrow \quad \boldsymbol{w}_t + \eta_t^* \boldsymbol{d}_t^* \qquad (12)$$

Here the pair of $\boldsymbol{d}_t^*$ and η_t^* is determined based on all the eigen vectors and their step lengths obtained above.

We consider three ways of performing step 3. As the first one, all the eigen vectors are considered as candidates and the pair of eigen vector and its step length which minimizes E is selected as $\boldsymbol{d}_t^*$ and η_t^*. This is called *EVD1*.

As the second one, a linear combination of all the eigen vectors is considered as a single candidate. More specifically, $\sum_m \eta_{m,t}\boldsymbol{v}_{m,t}$ gives the search direction and step length. This is called *EVD2*.

As the third one, all the eigen vectors and the linear combination stated above are considered as candidates, and the best one is selected as $\boldsymbol{d}_t^*$ and η_t^*. This is called *EVD3*.

IV. Experimental Evaluation

The proposed methods are evaluated for MLPs having a sigmoidal or exponential activation function using two artificial data sets and one real data set. The forward calculation of a sigmoidal MLP goes as below.

$$f = w_0 + \sum_{j=1}^{J} w_j z_j, \quad z_j = \sigma(\boldsymbol{w}_j^T \boldsymbol{x}) \qquad (13)$$

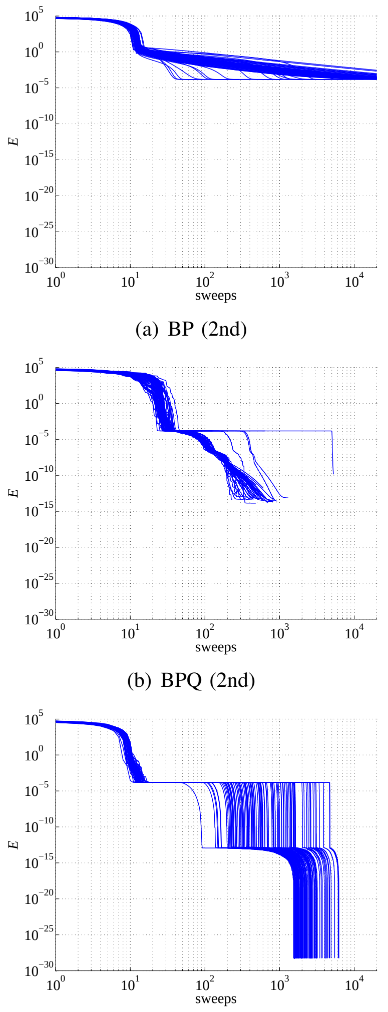
The forward calculation of a polynomial-type MLP goes as shown below. This type of MLP can represent multivariate polynomials [6]. Note that there is no bias unit in the input layer of this model.

$$z_j = \prod_{k=1}^{K} (x_k)^{w_{jk}} = \exp\left(\sum_{k=1}^{K} w_{jk} \ln x_k\right), \qquad (14)$$

$$f = w_0 + \sum_{j=1}^{J} w_j z_j \qquad (15)$$

The proposed methods have six types: EVD1, EVD2, and EVD3 are combined with two kinds of line search using the second- or third-order Taylor expansion. EVD1 (2nd), for example, indicates EVD1 combined with line search using the second-order Taylor expansion. As the existing methods, BP, CD (coordinate descent), and BPQ are combined with two kinds of line search. Thus, BP (2nd), for example, follows the same notation as above.

The basic cycle is repeated until one of the following is satisfied: the step length gets smaller than 10^{-30}, the absolute value of any gradient element gets smaller than 10^{-30}, or the number of sweeps exceeds $20,000$. For each data set, learning was performed 100 times changing initial weights. During a learning process we monitor eigen values and the condition number.



(a) BP (2nd)

(b) BPQ (2nd)

(c) EVD3 (2nd)

Fig. 3. Learning Process of Sigmoidal MLP for Artificial Data 1

A. *Sigmoidal MLP for Artificial Data 1*

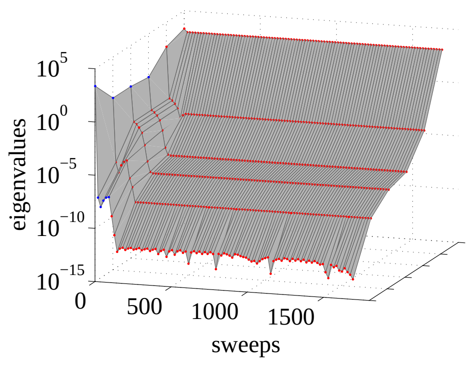
Artificial data 1 was generated for a sigmoidal MLP. The value of x_k was randomly selected from the range $(0, 1)$, and teacher signal y was calculated using the following weights; 500 data points were generated ($N = 500$).

$$w_0 = 5, \quad w_1 = 10, \quad \boldsymbol{w}_1 = \begin{pmatrix} 1 \\ 22 \\ 23 \\ 24 \end{pmatrix} \tag{16}$$

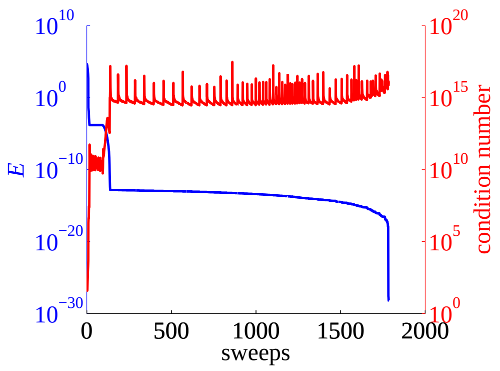
Weights are randomly initialized from the range $(-1, +1)$, and the final weights are classified as correct if any difference between the corresponding weights of the original and the final is less than 10^{-3}.

Figure 3 shows how training error E decreased through learning of BP (2nd), BPQ (2nd), and EVD3 (2nd). We can clearly see BP was trapped in the first gutter, and BPQ somehow escaped it but got stuck in the second gutter, while EVD3 (2nd) escaped even the second, reaching the true minimum.

All six types of proposed methods reached the correct weights for all 100 runs. Any existing method, however, didn't reach the correct weights at all.



(a) Eigen Values



(b) Condition Numbers

Fig. 4. How Eigen Values and the Condition Number Changed through EVD3 (2nd) Learning of Sigmoidal MLP for Artificial Data 1

Figure 4 shows how eigen values and the condition number changed during learning of EVD3 (2nd). The largest eigen value kept being almost constant through learning, while the smallest one decreased rapidly in an early stage, which reflected the growth of the condition number. That is, the condition number increased from 10^4 to 10^{16} in an early stage of learning, and then kept vibrating around 10^{16}. Note also that there is a remarkable tendency that a rapid increase of the condition number occurs simultaneously with a rapid decrease of training error E. Since the condition number of the final point is around 10^{16}, the final point is a crevasse-wok-bottom.

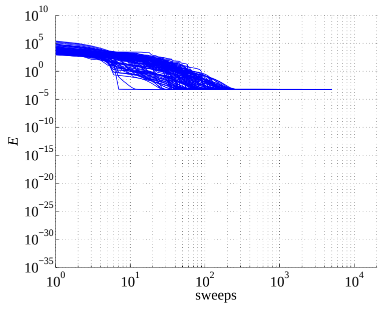
B. *Polynomial-type MLP for Artificial Data 2*

Here we consider the following polynomial.

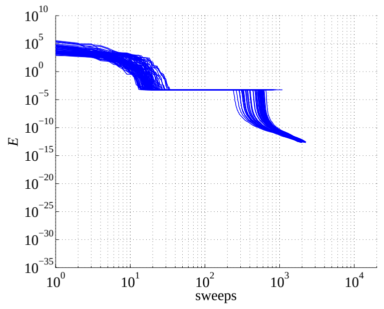
$$y \quad = \quad 5 + 10x_1^{22}x_2^{23}x_3^{24} \tag{17}$$

Artificial data 2 was generated for a polynomial-type MLP. The value of x_k was randomly selected from the range $(0, 1)$, and teacher signal y was calculated using the above equation; 200 data points were generated ($N = 200$). The weight initialization and the correctness judgement were performed in the same way as artificial data 1.

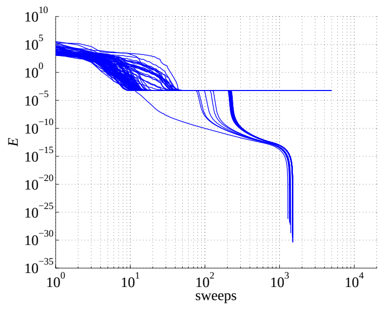
Figure 5 shows how error E decreased through learning. Much the same tendency as in artificial data 1 was observed; that is, BP was trapped in the first gutter, BPQ was trapped in the second gutter, while EVD3 escaped these gutters, reaching the correct weights in most runs.



(a) BP (2nd)



(b) BPQ (2nd)



(c) EVD3 (3rd)

Fig. 5. Learning Process of Polynomial-type MLP for Artificial Data 2

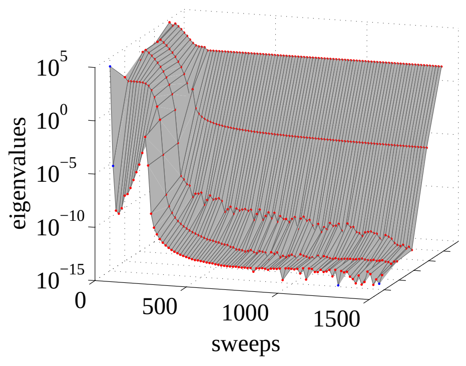
Table I shows the number of success runs for each method. Here success means reaching the correct weights. Since this is a hard problem, any existing method could not reach the correct weights at all for all 100 runs. Among EVD1, EVD2, and EVD3, EVD3 performed best as a whole. As for line search, the third-order Taylor expansion worked better than the second-order for this data.

TABLE I
THE NUMBER OF SUCCESSES OF POLYNOMIAL-TYPE MLP LEARNING FOR ARTIFICIAL DATA 2 (OUT OF 100 RUNS)

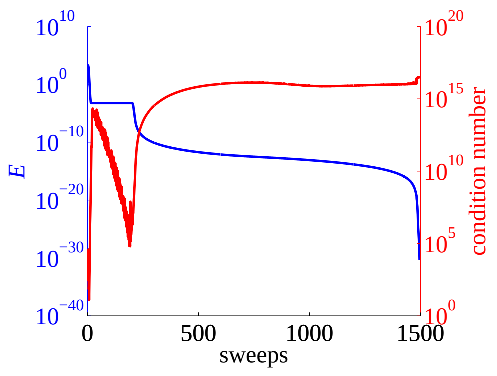
Method	EVD1		EVD2		EVD3		BPQ	
Taylor Exp.	2nd	3rd	2nd	3rd	2nd	3rd	2nd	3rd
Success Count	37	49	0	80	57	69	0	0

CD		BP	
2nd	3rd	2nd	3rd
0	0	0	0

Figure 6 shows how eigen values and the condition number changed during learning of EVD3 (3rd). The largest eigen value kept being almost constant after an early stage, while the smallest one went rapidly down, up and down in an early stage, and then kept being almost constant. After an early stage, the condition number kept being huge around 10^{16}. Around the 200th sweep a rapid increase of the condition number occurred simultaneously with a rapid decrease of training error E. The final point is a crevasse-wok-bottom.



(a) Eigen Values



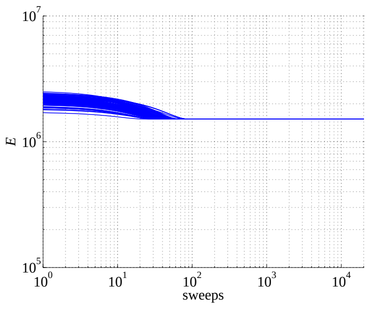
(b) Condition Numbers

Fig. 6. How Eigen Values and the Condition Number changed through EVD3 (3rd) Learning of Sigmoidal MLP for Artificial Data 2

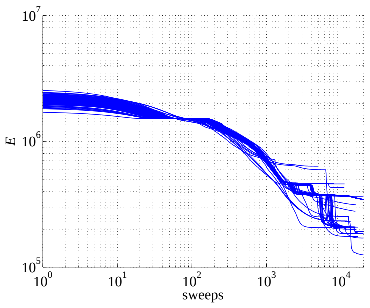
C. Sigmoidal MLP for Real Data

As real data, ball bearings data (Journal of Statistics Education) ($N = 200$) was used. The objective is to estimate fatigue(L50) using load(P), the number of balls (Z), and diameter (D). The weight initialization was performed in the same way as artificial data 1. In this experiment a weak weight decay regularizer (coefficient 10^{-5}) was employed.

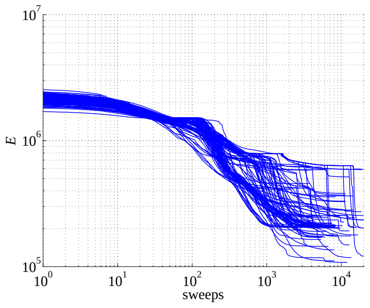
Figure 7 shows how training error E decreased through learning of BP (2nd), BPQ (2nd), and EVD3 (3rd). BP was trapped in a relatively large error gutter and could not move any further, and BPQ performed relatively well with a few good-quality solutions, while EVD3 (3rd) worked better than BPQ as a whole, finding many better solutions than BPQ could reach.



(a) BP (2nd)



(b) BPQ (2nd)



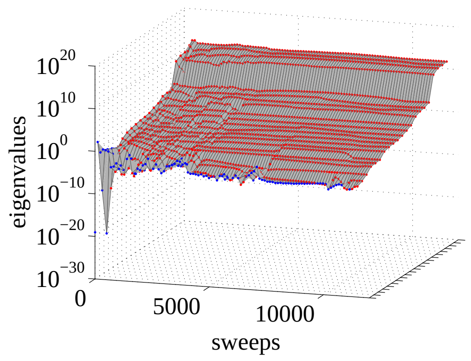
(c) EVD3 (3rd)

Fig. 7. Learning Process of Sigmoidal MLP for Real Data

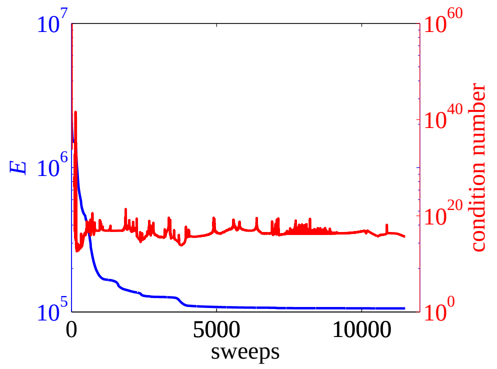
Figure 8 shows how eigen values and the condition number changed during learning of EVD3 (3rd). In this experiment, the largest and the smallest eigen values kept being almost constant after a very early stage. The condition number stayed high between 10^{15} and 10^{20} throughout the learning except the beginning.

V. CONCLUSION

This paper proposed a family of new search methods called EVD, which utilizes eigen vector descent and line search, aiming to stably find excellent solutions in a search space full of crevasse-like forms. Our experiments using sigmoidal MLPs and polynomial-type MLPs showed that the proposed methods worked well, moving through a crevasse-like form for two artificial data sets and one real data set. In the future



(a) Eigen Values



(b) Condition Numbers

Fig. 8. How Eigen Values and the Condition Number changed through EVD3 (3rd) Learning of Sigmoidal MLP for Real Data

we plan to reduce the load for computing eigen vectors, and try to apply our methods to wider variety of data sets to enlarge the applicability.

References

[1] K. Fukumizu and S. Amari, “Local minima and plateaus in hierarchical structure of multilayer perceptrons,” *Neural Networks*, vol. 13, no. 3, pp. 317–327, 2000.

[2] D. G. Luenberger, *Linear and Nonlinear Programming*, Addison-Wesley Pub.Co., 1973.

[3] R. C.J. Minnett, A. T. Smith, W. C. Lennon Jr. and R. Hecht-Nielsen, “Neural network tomography: network replication from output surface geometry,” *Neural Networks*, vol.24, no. 5, pp. 484–492, 2011.

[4] R. Nakano, S. Satoh and T. Ohwaki, “Learning method utilizing singular region of multilayer perceptron, ” *Proc. of the 3rd International Conference on Neural Computation Theory and Applications (NCTA ’11)*, pp. 106–111, 2011.

[5] K. Saito and R. Nakano, “Partial BFGS update and efficient step-length calculation for three-layer neural networks,” *Neural Computation*, vol.9, no. 1, pp. 239–257, 1997.

[6] K. Saito and R. Nakano, “Law discovery using neural networks,” *Proc. of 15th International Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 1078–1083, 1997.

[7] H. J. Sussmann, “Uniqueness of the weights for minimal feedforward nets with a given input-output map,” *Neural Networks*, vol.5, no. 4, pp. 589–593, 1992.

[8] S. Watanabe, *Algebraic geometry and statistical learning theory*, Cambridge University Press, 2009.