Optimization In Neural Networks using Projection Pursuit Model

Minwoo Kim

Department of Statistics

Seoul National University

Abstract

In supervised learning, the primary goal is to optimize the cost function of the given Neural Network model. Unfortunately, it is never easy to locate the desired global optimum due to high dimensionality and nonlinearity of the cost function. In practice, variants of gradient method algorithm are used for optimization. However, conventional Gradient Descent-based algorithm can never give a reliable solution as they do not use any information on the structure of the given specific cost function. In this paper, we present a new optimization framework that fully exploits information on the functional structure of the given cost function. We first present optimization strategy for Multi-Layer Perceptron based on Projection Pursuit Regression algorithm. The strategy is extended to "layer-wise feed-forward Projection Pursuit Regression algorithm", which is the generalized optimization framework that can be used to optimize general Deep Neural Network in a feed-forward manner.

Keywords: Projection Pursuit Regression, Neural Network, Optimization, Generalized Additive Model, Gradient Descent, Smoothing.

Contents

1 Introduction

In supervised learning, the primary goal is to optimize the cost function of the constructed Neural Network model. In most cases, however, it is never easy to locate the desired global optimum because of high dimensionality and nonlinearity of the given cost function : these complex features inevitably accompany the issues of the existence of local minima and the overwhelming number of saddle points. Especially, as the number of variables grows, so do the number of saddle points dramatically [1]. Although in most cases the issues are circumvented by applying tricks to standard gradient method such as using different initial point or adding specific momentum term to gradient, these modified gradient-based methods innately cannot give a reliable and consistent solution. This becomes clear if we recall No Free Lunch Theorem [2] for any generalized algorithm and remind that gradient descent algorithm does not make use of any specific information on the structure of the given cost function.

In this paper, our overall strategy for optimization problem is to fully exploit the information on the functional structure of the given specific cost function as much as we know, especially, projection structure in most cases. Specifically, once we construct simplified model that approximates the original cost function using projection structure, the next step would be to optimize the new model instead of the original one. This basic optimization strategy is extended to incorporate cases of general Deep Neural Network, by applying generalized Projection Pursuit Regression algorithm for each hidden layer successively and employing feed-forward scheme.

Section 2 introduces motivational example and illustrates the related idea. In section 3, we present optimization strategy for Multi-Layer Perceptron based on Projection Pursuit Regression algorithm. In section 4, the suggested optimization strategy is generalized and extended to deal with general classes of Neural Network models. Finally, section 5 contains a summary on the suggested strategy and discussion.

2 Motivation

Suppose that we want to find the global maximum of the following univariate function f defined by

$$f(x) = -\cos(x \cdot \sin(x)) - \arctan(x) - \sin(x), x \in [0, 15]$$

Once given an explicit formula, one possible choice would be to directly calculate gradient of the given function and obtain explicit expression for gradient. Even if we know the entire formula, however, it is virtually impossible to calculate gradient and seek zero-gradient solutions if the given function has such a complicated form like f. In this situation, our next probable strategy would be to apply gradient descent algorithm. As well illustrated in the Figure 1-(a), however, gradient descent algorithm poorly works if the given function is highly wavy or nonlinear as many Neural Network models are. The main reason is because the resulting solution is likely to get stuck into local maximum as shown in Figure 1-(a). Moreover, gradient descent algorithm is so sensitive to initial point that we might have to be recommended sufficiently good initial point by omniscient God, if we want to get a satisfactory solution from gradient descent algorithm. Otherwise, gradient descent algorithm will be likely to give a poor solution according to different initial point.

What if we could smooth the given function? By smoothing function, we can evaluate "overall local trend" at a certain point x based on the nearby function values. Information

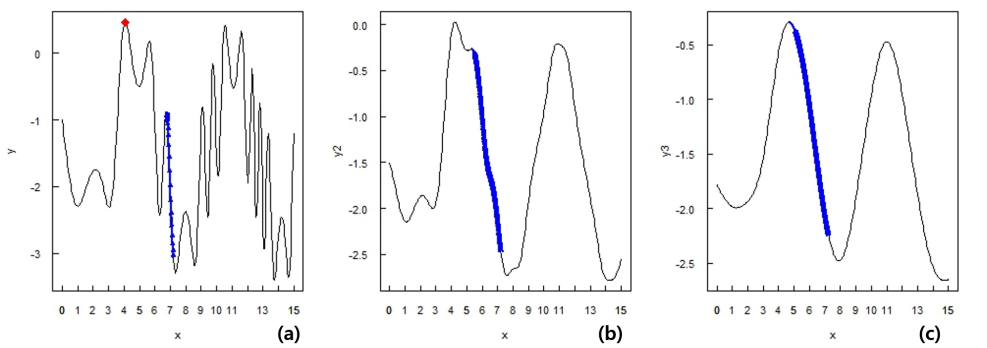

Figure 1: (a) is the original graph with true global maximum marked by red diamond, (b) is the graph smoothed by Gaussian kernels with $= 1$, and (c) is the graph estimated by the model $f(x) = (ax+b)sin(cx)$ from a set of sampled data. In each case, the gradient descent algorithm is equally applied with initial point $= 7.2$ and learning rate $= 0.005$, and their paths are marked by successive blue arrows.

on these locally-averaged values will surely be able to give us better insight in seeking global maximum as they contain local functional structure. That is, if we compute gradient on this smoothed function and obtain a certain direction(gradient) at x, it tells us that "by moving in the indicated direction we could eventually climb up this hill despite of a couple of immediate small downhills in front". Figure 1-(b) illustrates the situation. Figure 1-(b) further suggests that the more smoothed the function is, the better chances are that even gradient descent algorithm outputs better estimate for the global maximum. This is because as the function becomes smoothed, the number of local optima greatly reduces, so that gradient descent technique can have a better chance to reach near the true global maximum. Although the resulting solution is not exactly a true solution to the original function, the smoothed solution is a much better estimate compared to the unsmoothed solution, in that the smoothed solution is located much closer to the true global maximum.

Figure 1-(c) shows an estimated function $\hat{f}$ from a set of sampled data with model $f(x) = (ax + b)\sin(cx)$. If we could have known somehow the functional structure of f, so we guessed a proper model $f(x) = (ax + b)\sin(cx)$ as in the Figure 1-(c), then the estimated function $\hat{f}$ could be an useful model to approximate global maximum.

In this figure, it is worth noticing that function smoothing can be achieved by constructing simplified model that approximates the original function. The idea will be grounds for constructing Projection Pursuit Model for optimization purposes, instead of directly smoothing the given cost function ultimately.

As seen in the above example, if we could somehow simplify the given function by computing smoothed cost function or by appropriately modeling the function, either of which simplifies the original function whilst keeping the information on the functional structure of the given cost function, then it would be much easier to estimate the global optimum even with gradient descent.

3 Smoothed Projection Pursuit Regression

Suppose that we have a Multi-Layer Perceptron (MLP). Since MLP with multiple hidden layers is redundant so not used in practice according to Universal Approximation Theorem (G.Cybenko, 1989) [3], it suffices to assume that the model has a single hidden layer. Assume that we are given observed data $(\mathbf{x}_i, y_i), i = 1, 2 \cdots, N$, where $\mathbf{x}_i = (x_{i1}, \cdots, x_{ip-1}, 1)$ are continuous input data augmented by bias term and y_i are continuous responses. Denote by $w_{ml}^{(1)}$ a weight connecting l-th node in the input layer and m-th node in the hidden layer, and by $w_m^{(2)}$ a weight connecting m-th node in the hidden layer and the output node. Let $\mathbf{w}_l^{(1)} = (w_{l1}^{(1)}, \cdots, w_{lp}^{(1)})^T$, $\mathbf{w}^{(2)} = (w_1^{(2)}, \cdots, w_q^{(2)})^T$, and $\phi : \mathbf{R} \to \mathbf{R}$ be a sigmoid activation function. Our goal is then to find a global minimum of the function

$$
\begin{aligned}
L(\mathbf{w}_l^{(1)}, \mathbf{w}^{(2)}) &:= \frac{1}{N} \sum_{i=1}^{N} \left(y_i - g(x_i; \mathbf{w}_l^{(1)}, \mathbf{w}^{(2)}) \right)^2 \\
&:= \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \sum_{l=1}^{q} w_l^{(2)} \phi \left(\mathbf{w}_l^{(1)^T} x_i \right) \right)^2
\end{aligned} \tag{1}
$$

. For this problem, one can easily verify that the given problem can be viewed as the following regression problem with parameters $\mathbf{w}_l^{(1)}, f_l \left(= w_l^{(2)} \phi \right)$ to be estimated

$$
\mathrm{E}[Y|X; \mathbf{w}_l^{(1)}, \mathbf{w}^{(2)}] = g\left(X; \mathbf{w}_l^{(1)}, \mathbf{w}^{(2)} \right) = \sum_{l=1}^{q} f_l \left(\mathbf{w}_l^{(1)^T} x_i \right)
$$

, where L_2 loss function is used for error measurement. Note that the model has an exact form of Projection Pursuit Regression (PPR) [4]. The distinguishing feature of PPR model compared to Additive Model (AM) [5] or Alternating Conditional Expectations (ACE) [6] is the existence of "projection pursuit" structure. Because of the projection pursuit structure, however, PPR model requires more than backfitting for fitting algorithm. Specifically, PPR algorithm alternately applies backfitting algorithm and Gauss-Newton method.

With this PPR algorithm, we can obtain the global minimum $\left(\mathbf{w}_l^{(1)}, \mathbf{w}^{(2)} \right)$ by imposing $\hat{f}_l = w_l^{(2)} \phi$ for $l = 1, \cdots, p$ and not permitting any kind of smoothing operation on $\hat{f}_l$ in each backfitting step through iteration. In each iteration in the backfitting step, $w_l^{(2)}$ can be obtained by solving simple OLS problem given $\mathbf{w}_l^{(1)}$ without having to nonparametrically

estimate f_l. The successive update for f_l is done only through parameters $w_l^{(2)}$ so that we restrict the path of convergence of f_l sequences. The convergence of PPR algorithm (Lee K. Jones, 1987) [7] along with the prior knowledge on the true structures of f_l guarantee that true global minimum will be well-approximated by the algorithm.

Meanwhile, consider if we substituted the PPR model by its kernel-smoothed version,

$$\begin{aligned} \mathrm{E}[\tilde{Y}|X;\mathbf{w}_l^{(1)},\mathbf{w}^{(2)},\Sigma] &= \int g\left(\tilde{X};\mathbf{w}_l^{(1)},\mathbf{w}^{(2)}\right) N(\tilde{X};X,\Sigma)d\tilde{X} \\ &= \int \left(\sum_{l=1}^{q} w_l^{(2)}\phi\left(\mathbf{w}_l^{(1)^T}\tilde{X}\right)\right) N(\tilde{X};X,\Sigma)d\tilde{X} \end{aligned} \tag{2}$$

, where $N(\tilde{X};X,\Sigma)$ is multivariate Gaussian kernel with mean X and covariance matrix Σ. Note that the computation of (2) is nearly impossible due to the curse of dimensionality, which requires $O(n^p)$ number of samples to be drawn. To ease the computational burden, we instead use a specific combination of kernels in light of the projection pursuit idea from PPR, which generates the following smoothed Projection Pursuit Regression model

$$\begin{aligned} \mathrm{E}[\tilde{Y}|X;\mathbf{w}_l^{(1)},\mathbf{w}^{(2)},\sigma_l] &= \tilde{g}\left(X;\mathbf{w}_l^{(1)},\mathbf{w}^{(2)},\sigma_l\right) \\ &= \sum_{l=1}^{q}\int w_l^{(2)}\phi\left(\mathbf{w}_l^{(1)^T}\tilde{X}\right) K\left(\frac{w_l^{(1)^T}\left(\tilde{X}-X\right)}{\sigma_l}\right) dz_l \\ &= \sum_{l=1}^{q}\int w_l^{(2)}\phi\left(z_l\right) K\left(\frac{z_l - w_l^{(1)^T}X}{\sigma_l}\right) dz_l \end{aligned} \tag{3}$$

, where $K(\cdot)$ denotes univariate Gaussian kernel and $z_l = w_l^{(1)^T}\tilde{X}$. With this specific choice of kernels, we can approximate the integrations by sampling $z_l = w_l^{(1)^T}\tilde{X}$ directly along each "projected axis", which only requires $O(np)$ number of samples. It is worth noting that the newly obtained regression problem is nothing but a minimization problem with the following smoothed version of minimizing criterion (1)

(4)

$$\tilde{L}(\mathbf{w}_l^{(1)},\mathbf{w}^{(2)};\sigma_l) := \int \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \sum_{l=1}^{q} w_l^{(2)}\phi\left(z_l\right)\right)^2 \prod_{l=1}^{q} K\left(\frac{z_l - w_l^{(1)^T}x_i}{\sigma_l}\right) dz_1\cdots dz_q$$

. Furthermore, this minimizing criterion leads to the solution that can be obtained from PPR algorithm with kernel smoothing operation using the above specific sets of kernels.

Thus, standard PPR algorithm can be used to solve (3), where what "standard" means that this time the algorithm relieves the restriction $\hat{f} = w_l^{(2)}\phi$ and permits Nadaraya-Watson estimators for $\hat{f}_l$ with projection pursuit kernels $K\left(\frac{z_l - {w_l^{(1)}}^T x_i}{\sigma_l}\right)$. The problem is, however, we cannot get explicit values for $w_l^{(2)}$ because this time we nonparametrically estimate $\hat{f}_l$. In order to deal with this issue, we need to take layer-wise approach. To be specific, we first construct PPR model using input layer and output layer to obtain $\mathbf{w}_l^{(1)}$. Then, we next construct second linear model using hidden layer and output layer to obtain $\mathbf{w}^{(2)}$. Details on this layer-wise feed-forward PPR algorithm are discussed in section 4-3 in more general framework.

In kernel smoothing operation, choice of σ_l determines the degree of smoothing in each projected pursuit direction. Fixing $\sigma = \sigma_1 = \cdots = \sigma_q$ would be the simplest and most convenient option. Another strategy would be to allow successive update for σ_l in the backfitting step of PPR. We might be able to seek optimal σ_l based on predetermined model selection criteria, but it would significantly slower each iteration as we have to perform exhaustive grid-search and apply smooth backfitting for each given tick of σ_l grid.

As a way to avoid heavy computation from PPR algorithm in solving (3), we might be able to instead apply gradient descent algorithm on (4). We have seen in section 2 that even gradient descent algorithm can give a better result on a smoothed error surface than unsmoothed one. Since we know the exact form of given smoothed cost function $\tilde{L}$, we can compute gradient explicitly, resulting in the following expression

$$\frac{\partial \tilde{L}}{\partial w_s^{(2)}} = -\int \frac{2}{N}\sum_{i=1}^{N}\phi(z_s)\left(y_i - \sum_{l=1}^{q} w_l^{(2)}\phi\left(z_l\right)\right)\prod_{l=1}^{q} K\left(\frac{z_l - {w_l^{(1)}}^T x_i}{\sigma_l}\right) dz_1 \cdots dz_q$$

$$\frac{\partial \tilde{L}}{\partial w_{st}^{(1)}} = \int \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \sum_{l=1}^{q} w_l^{(2)}\phi\left(z_l\right)\right)^2 x_{it}\left(\frac{z_s - {w_s^{(1)}}^T x_i}{{\sigma_s}^2}\right)\prod_{l=1}^{q} K\left(\frac{z_l - {w_l^{(1)}}^T x_i}{\sigma_l}\right) dz_1 \cdots dz_q$$

. This allows us to directly apply gradient descent algorithm on a smoothed error surface obtained from projection pursuit kernel smoother. Whether we apply PPR algorithm or smoothed gradient descent algorithm, the important fact is that the target cost function is changed to simplified(smoothed) version of the original one and the resulting simplified model keeps functional structure of the original cost function.

4 Extension

In this section, we discuss the extension of the suggested strategy which was only tailored to MLP with squared error loss and linear output function. Extension requires us to generalize output function, loss function, and even architecture of Neural Network.

4.1 Autoencoder

Autoencoder is a special form of MLP with the same number of input and output neurons, which is often used for dimensionality reduction of input data. For Autoencoder, the cost function is given by

$$L(\mathbf{w}_l^{(1)}, \mathbf{w_m}^{(2)}) = \sum_{i=1}^{N} \left\| \mathbf{y}_i - \mathbf{h}(x_i; \mathbf{w}_l^{(1)}, \mathbf{w}_m^{(2)}) \right\|^2$$

, where $\mathbf{y}_i = (y_{i1}, \cdots, y_{iK})$, $\mathbf{h}(x_i; \mathbf{w}_l^{(1)}, \mathbf{w}_m^{(2)}) = (\phi(\sum_{l=1}^{q} w_{ml}^{(2)} \phi(\mathbf{w}_l^{(1)^T} x_i)))_{m=1}^{K}$, $\mathbf{w}_m^{(2)} = (w_{m1}^{(2)}, \cdots, w_{mq}^{(2)}), m = 1, \cdots, K$ and $w_{ml}^{(2)}$ denotes weight between m-th node in the output layer and l-th node in the hidden layer. Compared to MLP, additional nonlinear function in the output layer complicates the resulting cost function. As a result, we cannot separate the given function into an additive form of functions of projection pursuit vector (allowing up to multiplication between these functions). This implies our projection pursuit smoother cannot be applied virtually, because it requires $O(n^p)$ number of samples to be drawn to approximate integration in smoother. However, as sigmoid function ϕ is invertible, we can approximate Euclidean distance between y_{im} and $\phi(\sum_{l=1}^{q} w_{ml}^{(2)} \phi(\mathbf{w}_l^{(1)^T} x_i))$ by

$$\begin{aligned}
\left| y_{im} - \phi(\sum_{l=1}^{q} w_{ml}^{(2)} \phi(\mathbf{w}_l^{(1)^T} x_i)) \right| &\approx \left| \left(\frac{y_{im}}{\phi^{-1}(y_{im})} \right) \left(\phi^{-1}(y_{im}) - \sum_{l=1}^{q} w_{ml}^{(2)} \phi(\mathbf{w}_l^{(1)^T} x_i) \right) \right| \\
&= \left| \left(\frac{y_{im}}{\log \frac{y_{im}}{1-y_{im}}} \right) \left(\log \frac{y_{im}}{1-y_{im}} - \sum_{l=1}^{q} w_{ml}^{(2)} \phi(\mathbf{w}_l^{(1)^T} x_i) \right) \right|
\end{aligned}$$

. With this approximation, we can approximate integration using $O(np)$ number of samples when projection pursuit kernel smoother is applied to the given cost function. Then, we can use smoothed gradient descent algorithm by directly calculating gradient and obtaining explicit form of gradient first.

4.2 Multi-layer Perceptron Classifier

In classification problem, the most popular model would be a MLP with Softmax output function, which allows nominal type of response. In this problem, the objective is to optimize probability-like cost function. In K-class classification problem, the cost function is given by

$$L(\mathbf{w}_l^{(1)}, \mathbf{w}_m^{(2)}) = -\sum_{i=1}^{N}\sum_{m=1}^{K} \mathbf{1}\{y_i = m\} \log \frac{\exp\left(\sum_{l=1}^{q} w_{ml}^{(2)} \phi(\mathbf{w}_l^{(1)^T} x_i)\right)}{\sum_{j=1}^{K} \exp\left(\sum_{l=1}^{q} w_{jl}^{(2)} \phi(\mathbf{w}_l^{(1)^T} x_i)\right)}$$

, where y_i takes discrete values. As in autoencoder, non-separability of the denominator part within log function renders projection pursuit smoother useless. At this point, a noticeable fact is that $\mathbf{w}_m^{(2)} = (w_{m1}^{(2)}, \cdots, w_{mq}^{(2)})$ can be arbitrarily scaled down, as multiplying the same factor in the nominator and denominator gives the same function value. Also, the range of sigmoid function is $[-1, 1]$. With these two facts in mind, we approximate

$$\log\left(\sum_{j=1}^{K} \exp\left(\sum_{l=1}^{q} w_{jl}^{(2)} \phi(\mathbf{w}_l^{(1)^T} x_i)\right)\right) \approx \log K + \frac{1}{K}\sum_{j=1}^{K}\sum_{l=1}^{q} w_{jl}^{(2)} \phi\left(\mathbf{w}_l^{(1)^T} x_i\right)$$

, using Talyor's expansion near origin. Now we can apply projection pursuit smoother since the entire function is expressible in an additive form of functions of projection pursuit vector. Applying smoothed gradient descent algorithm would be the next step.

4.3 General Feed-Forward Deep Neural Network

In general, feed-forward Deep Neural Network model is so complicated that it is extremely difficult to represent the model by a single explicit formula. Typical optimization strategy is to use variants of gradient descent method. There are so many different types of gradient descent-based algorithms developed in different context, which includes the use of different initialization method, different learning method and/or different types of momentum. However, these methods barely use information on the true structure of error surface, so poor results are expected. One way to utilize the so-far discussed PPR method for general Deep Neural Network would be to apply "layer-wise feed-forward PPR algorithm". Although we cannot represent the whole function structure by a single simple explicit formula, we

instead model only a portion of function structure. Suppose $\mathbf{w}_j^{(l)} = (w_{j1}^{(l)}, \cdots, w_{jp}^{(l)})^T$ & $w_{ji}^{(l)}$ denotes a weight between i-th node in l-th layer and j-th node in $(l+1)$-th layer. For the case $l = 1$, $w_{ji}^{(l)}$ represents a weight between input layer and first hidden layer. Let g denote the output function. At step l, we give a structure only to a portion of entire function, by modeling

$$g\left(\mathrm{E}[Y|X]\right) = \sum_{j=1}^{q} f_j\left(\mathbf{w}_j^{(l)^T} X\right) + \alpha$$

where $X = (x_1, \cdots, x_p)$ is a vector of node values in l-th layer. In this way, we keep "projection structure" through linear combinations between weights and input values, and let the remaining part of function unknown which is nonparametrically represented by a set of functions f_j. The presented "generalized PPR" model can be solved by embedding PPR algorithm into local scoring procedure, as follows:

Algorithm 1 Generalized PPR algorithm

procedure GPPR($\mathbf{x}_i, y_i$)
  $\alpha^{(0)} = g\left(\sum_{i=1}^{n} y_i/n\right)$, $f_1^{(0)} = \cdots = f_q^{(0)} = 0$ ▷ Initialize
  while r <IterationNumber **do**
    $\eta_i^{(r)} = \alpha^{(r)} + \sum_{j=1}^{q} f_j^{(r)}\left(\mathbf{w}_j^{(r)^T}\mathbf{x}_i\right)$, $\mu_i^{(r)} = g^{-1}\left(\eta_i^{(r)}\right)$
    $z_i^{(r)} = \eta_i^{(r)} + \left(y_i - \mu_i^{(r)}\right)\left(\frac{\partial\eta}{\partial\mu}|_{\mu=\mu_i^{(r)}}\right)$ ▷ Define working response
    $v_i^{(r)} = \left(\frac{\partial\eta}{\partial\mu}|_{\mu=\mu_i^{(r)}}\right)^2\left(V(\mu_i^{(r)})\right)^{-1}$ ▷ Construct weight
    procedure PPR($\mathbf{x}_i, y_i, v_i^{(r)}$) ▷ Fit PPR with weight $v_i^{(r)}$
      return $f_j^{(r+1)}$, $\mathbf{w}_j^{(r+1)}, \alpha^{(r+1)}$
    end procedure
    if $f_j^{(r)}$ converged **then break**
    end if
  end whilereturn f_j, $\mathbf{w}_j, \alpha$
end procedure

Here, $\eta(X) = \mathrm{E}[Y|X]$, $\mu = g(\eta)$, and $V(\mu) = Var(Y|\mu)$. At step l, the goal is to estimate $\mathbf{w}_j^{(l)}$ using this generalized PPR algorithm, and we forward input values in l-th layer to next layer by computing $\mathbf{w}_j^{(l)^T} X$ followed by known operation in $(l+1)$-th layer such as activation function or convolution operator. Then we proceed to the next step $(l+1)$ and repeat the same procedure until we reach the output layer. At the final step, we can

construct simple GLM model $g\left(\mathrm{E}[Y|X]\right) = \mathbf{w}^T X$, as there is no uncertain part in the function structure.

5 Summary

In this paper we presented Projection Pursuit Regression Model-based optimization strategy to solve the Neural Network optimization problem. Up to now, a lot of optimization methods have been developed and studied. However, the usual gradient descent based methods can easily lead to poor solution due to a number of local optima and saddle points. This is because successive update of gradient descent can easily be stucked into one of those unwanted critical points. For standard MLP, applying PPR algorithm is one good way to obtain well-approximated global minimum estimate, as PPR algorithm itself contains smoothing operation within the algorithm. Or, by explicitly smoothing the cost function using Projection Pursuit Kernel smoother and applying gradient descent on this smoothed error surface, we can still obtain a good estimate. For general Deep Neural Network, we cannot represent the whole model by a single simple explicit formula. In this situation, a successive application of generalized PPR algorithm with layer-wise training technique would be one good strategy to train the given Neural Network model.

References

1 Dauphin, Y., Pascanu, R., Gulcehre, C., Kyunghyun, C., Ganguli, S., and Bengio, Y. **Identifying and attacking the saddle point problem in high-dimensional non-convex optimization.** *Advances in Neural Information Processing Systems,* pages 2933–2941, 2014.

2 David H. Wolpert and William G. Macready. **No Free Lunch Theorems for Optimization** *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, **1**, No. 1, pages 67-82, 1997.

3 CYBENKO, G. **Approximation by superposition of sigmoidal functions** *Math. Control, Sign & Systems*, **2**, No. 4, pages 303-314, 1989.

4 Friedman, J.H. and Stuetzle, W. **Projection Pursuit Regression** *Journal of the American Statistical Association*, **76**, No. 376, pages 817–823, 1981.

5 Hastie, T. and Tibshirani, R. **Generalized Additive Models** *Statistical Science*, **1**, No. 3, pages 297-318, 1986.

6 Breiman, L. and Friedman, J.H. **Estimating optimal transformations for multiple regression and correlation** *Journal of the American Statistical Association*, **80**, No. 391, pages 580-598, 1985.

7 Lee K. Jones. **On a Conjecture of Huber Concerning the Convergence of Projection Pursuit Regression** *The Annals of Statistics*, **15**, No. 2, pages 880-882, 1987.

8 Nadaraya, E. A. **On Estimating Regression** *Theory of Probability and its Applications*, **9**, No. 1, pages 141-142, 1964.

9 Gallagher, M. R. **Multi-layer Perceptron Error Surfaces: Visualization, Structure and Modelling** *University of Queensland*, 1999.

10 Weise, T. Global Optimization Algorithms : Theory and Application, 2nd Edition.

11 Leordeanu, M., Hebert, M. **Smoothing-based Optimization** *Proceedings of Computer Vision and Pattern Recognition(CPVR)*

12 Roosen, C. and Hastie, T., **Logistic Response Projection Pursuit**

Optimization In Neural Networks using Projection Pursuit Model

김민우
서울대학교
자연과학대학 통계학과

초록

지도학습에서 가장 주된 목표는 주어진 인공신경망모형의 비용함수를 최적화하는 것이다. 하지만, 많은 경우에 있어서 인공신경망모형의 비용함수는 그 고차원적인 특성과 비선형성으로 인해 원하는 전역최적값을 찾기 쉽지 않다. 이러한 최적화문제에 있어서 다양한 종류의 경사하강법 기반 알고리즘들이 사용되곤하지만, 일반적인 경사하강법 기반 알고리즘들로는 신뢰성있는 해를 얻기 어려운데, 이는 이러한 알고리즘들이 특정문제에 주어진 비용함수의 구조에 관한 정보를 전혀 이용하지 않기 때문이다. 본 논문에서는 이러한 문제점을 인지하여 경사하강법과 달리 주어진 비용함수의 함수적 구조에 관한 정보를 가능한 충분히 이용할 수 있는 새로운 최적화 전략을 제시하고자 한다. 이를위해 우선 사영추적회귀 알고리즘을 이용한 다층퍼셉트론 최적화 전략을 먼저 제시하고, 이 방법을 확장하여 일반적인 심화신경망의 최적화문제에 까지 적용할 수 있는 일반화된 사영추적회귀 기반의 최적화전략을 최종적으로 제시하고자 한다.

주요어: 사영추적회귀, 인공신경망, 최적화, 일반화가법모형, 경사하강법, 평활법.