Query Based Learning in a Multilayered Perceptron in the Presence of Data Jitter

Seho Oh, Robert J. Marks II and M. A. El-Sharkawi

Department of Electrical Engr., FT-10
University of Washington
Seattle, WA 98195
(206) 543-2150

ABSTRACT

Stochastically perturbed feature data is said to be jittered. Jittered data has a convolutional smoothing effect in the classification (or regression) space. Parametric knowledge of the jitter can be used to perturb the training cost function of the neural network so that more efficient training can be performed. The improvement is more striking when the addended cost function is used in a query based learning procedure.

1 Introduction

In this study, we combine the topics of query based learning in a layered perceptron with the presence of jittered data and demonstrate how parametric knowledge of the jitter can be used to improve learning.

Query based learning applied to neural networks was first proposed in an application to power security assessment [1]. In query based learning, a partially trained neural network is asked "What don't you understand?". The response of the neural network is taken to an *oracle* which, for a price, will clarify the neural network's confusion. Through this interactive process, the neural network can be better trained by not exposing it to training data with which it is already familiar. Methods of querying include interval halving [1, 3] and neural net inversion [2]. In this paper, we use a revised interval halving approach.

Jittered data occurs when the input data to a classification or regression machine are stochastically perturbed. If the input variable is $\vec{x}$ and the desired target value is $t(\vec{x})$, then the effect of jitter on the target value is perturbation of the t surface to $t * P$ where $P(\vec{x})$ is the density of the jitter and $*$ denotes convolution. Knowledge of this alteration can be used to improve the definition of the cost function through an addendment of a first order correction factor. Use of this revised function in query based systems can dramatically improve learning performance. An illustrative exemplar is provided.

2 Training Multilayered Perceptron

Let U be the input space and let $U_t \subset U$ be the training set of the MLP. Define the *energy function* E as

$$E = \sum_{\vec{x} \in U_t} [t(\vec{x}) - NN(\vec{x};\vec{w})]^2 \qquad (1)$$

where $t(\vec{x})$ is the desired target and $NN(\vec{x};\vec{w})$ is the *multilayered perceptron* (MLP) output value with input $\vec{x}$ and weights $\vec{w}$. In the training stage, we find the weight vector $\vec{w}$ thet minimizes E. There are many methods for the training the MLP. The most popular method is *error back propagation* (EBP). This training technique is a gradient descent method. A disadvantage of EBP is that it requires gradient information about the classification or regression surface and can become trapped in local minima. There are other methods, such as random search, which do not required a gradient calculation. Weight updates are generated from a random number generator. The energy corresponding to the previous and new weights are evaluated. The weights with the smaller energy are saved and the process repeated. The covergence rate of this and related methods can be quite good [4]. Unlike EBP, random search can work quite well in the presence of computational inexactness.

3 Inversion Method of Query Based Learning

We propose use of an interval halving method for inversion that is a variation of that previously proposed [1, 3]. We wish to find a vector, $\vec{x}$, that satisfies $NN(\vec{x};\vec{w}) = 0.5$. If $NN(\vec{x}_1;\vec{w}) < 0.5$ and $NN(\vec{x}_2;\vec{w}) > 0.5$, then there is an α on the interval $0 \le \alpha \le 1$ and a corresponding $\vec{x}_3 = \alpha\vec{x}_1 + (1-\alpha)\vec{x}_2$ such that $NN(\vec{x}_3;\vec{w}) = 0.5$. We wish to estimate $\vec{x}_3$. Using the a linear fit, an eatimate of $\vec{x}_3$ is

$$\vec{x}_3 = \frac{f_2 - 0.5}{f_2 - f_1}\vec{x}_1 + \frac{0.5 - f_1}{f_2 - f_1}\vec{x}_2 \qquad (2)$$

where $f_1 = NN(\vec{x}_1;\vec{w})$ and $f_2 = NN(\vec{x}_2;\vec{w})$. Based on this approach, we propose the following iterative method for neural network inversion.

Algorithm

```
(1) Generate two initial input vectors $\vec{x}_1$ and $\vec{x}_2$ such that $f_1 = NN(\vec{x}_1;\vec{w}) < 0.5$ and $f_2 = NN(\vec{x}_2;\vec{w}) > 0.5$.
(2) Compute the intermediate vector $\vec{x}_3$ from (2).
(3) Evaluate $f_3 = NN(\vec{x}_3;\vec{w})$.
(4) If $f_3 > 0.5$ then $\vec{x}_2 = \vec{x}_3$ and $f_2 = f_3$.
    If $f_3 < 0.5$ then $\vec{x}_1 = \vec{x}_3$ and $f_1 = f_3$.
(5) If $f_3$ is sufficiently close to 0.5, then stop, otherwise goto setp (2)
```

This procedure generally converges more quickly than those previously proposed [1, 3].

4 Jittered Data

In this section, we establish training when the training data is jittered. If the training data is contaminated by additive noise, the effect of the noise is jitter in the input training space of the MLP. Let $P(\vec{z})$ be the probability density function of the jitter. The probability that $\vec{x}$ comes from $\vec{y}$ is $P(\vec{x}-\vec{y})$. When we have the training data $\vec{x}$, the cost fuction at data $\vec{x}$ is the convolution

$$E(\vec{x}) = \int P(\vec{x}-\vec{y})[t(\vec{x}) - NN(\vec{y};\vec{w})]^2 d\vec{y} \qquad (3)$$

Assume that $P(\vec{z}) = P(-\vec{z})$. We take the first term of the Taylor series of (3), and obtain the addended energy function

$$\begin{aligned} E(\vec{x}) &= [t(\vec{x}) - NN(\vec{x};\vec{w})]^2 \\ &\quad +(\nabla_x NN)^T \underline{C} (\nabla_x NN)^T \end{aligned} \qquad (4)$$

where the covariance matrix $\underline{C}$ is

$$\underline{C} = \int P(\vec{z})\vec{z}\vec{z}^T d\vec{z}.$$

Therefore the total energy for the training stage is

$$E = \sum_{\vec{x}\in\mathcal{U}_t} E(\vec{x})$$

If the jitter is independent with like variance, then $\underline{C} = \sigma^2 \underline{I}$, and the total energy is

$$\begin{aligned} E &= \sum_{\vec{x}\in\mathcal{U}_t} \{[t(\vec{x}) - NN(\vec{x};\vec{w})]^2 \\ &\quad +\sigma^2 \parallel \nabla_x NN \parallel^2\} \end{aligned} \qquad (5)$$

This energy does not readily adapt to the EBP algorithm. We choose use, alternately, of use the random search technique for training.

5 Examples

In this section we provide the example results of simulation of the above theory. We use a binary classification class "1" inside circle and class "0" outside circle. Queries are performed in threes. The boundary point is estimated using the interval halving technique described previously. In addition, conjugate pair data points are generated as described by Hwang *et.al.* [2]. A more efficient gradient evaluation technique, outlined in the Appendix, was used.

The first training is done on 75 randomly generated data points and we take 25 inversion data triplets for the next training phase. The variance of jitter is 0.05. Figure 1 shows classification using 250 randomly generated data points. Figure 2 shows

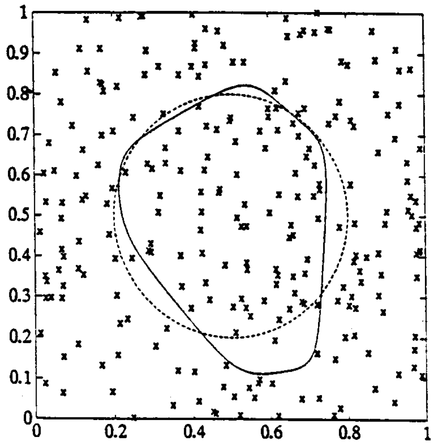

Figure 1 : The result of MLP with 250 randomly generated data. Solid lines denote representation and circular broken line the concept. The mark 'x' indicates training data. The probability of misclassification is 6.35%.

the sequence of the query based learning using the energy function in (1). Figure 3 shows the results of the new cost function in (5). The probability of misclassification using 250 randomly generated data in Figure 1 resulted in an error rate of 6.35%. Use of a conventional query, shown in figure 2(b), decreased this by 31.4% to 4.35% (125 random points were updated 25 query triplets). The cost function in (1) was used. Using the same number of data, as shown in Figure 3, reduced this 57.8% to 4.35% where cost function in (5) is used.

Acknowledgements

This work was supported with grants from the National Science Foundation and the Washington Technology Center.

References

[1] M.A. El-Sharkawi, R.J. Marks II, M.E. Aggoune, D.C. Park, M.J. Damborg and L.E. Atlas, "Dynamic security assessment of power systems using back error propagation artificial neural networks", *Proceedings of the 2nd Annual Symposium on Expert Systems Applications to Power Systems*, pp.366-370, 17-20 July 89, Seattle.

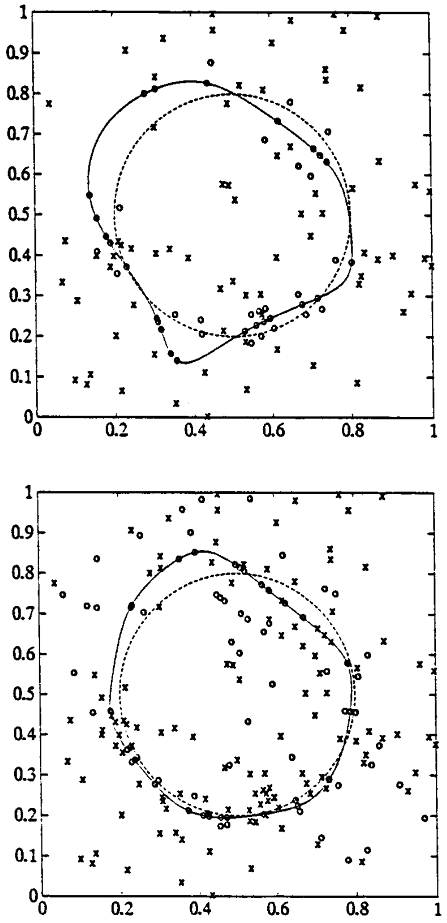

Figure 2 : The results of query based MLP with energy function in (1). The upper figure shows initial training with 75 random points. The lower figure shows training using 25 additional query generated data triplets. The mark 'o' marks the triplets. The probabilities of misclassification of upper and lower figures are 6.74% and 4.35% respectively.

[2] J.N. Hwang, J.J. Choi, S. Oh and R.J. Marks II, "Query based learning applied to partially trained multilayer perceptrons", *IEEE Transactions on Neural Networks*, Vol. 2, pp.131-136, (1991).

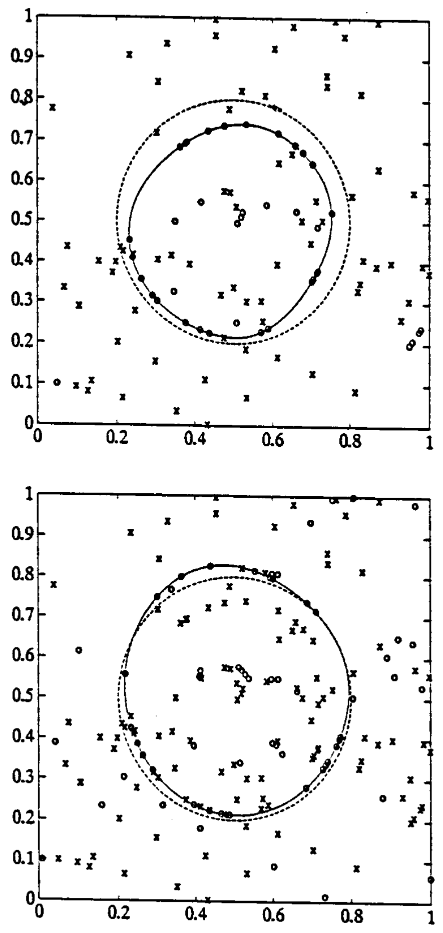

Figure 3 : The results of query based MLP with the addended energy function in (5). The upper figure shows initial training with 75 randomly generated data points. The lower figure shows training using 25 additional query generated data triplets. The probabilities of misclassification of upper and lower figures are 7.51% and 2.58% respectively.

[3] E.M. Baum, "Neural nct algorithms that learn in polynomial time from examples and queries", *IEEE Transactions on Neural Networks*, Vol. 2, pp.5-19, (1991).

[4] Jai J. Choi, "Heuristics on efficient learning in back-propagation neural networks" PhD Dissertation, Department of Electrical Engineering, University of Washington, Seattle (1990).

APPENDIX

Recurrent Formula of Gradient Evaluation

The multilayered perceptron is described by

$$u_i(l+1) = \sum_{j=0}^{N_l} w_{ij}(l+1)a_j(l)$$

$$a_i(l+1) = f(u_i(l+1))$$

where $a_j(l)$ and $u_j(l)$ denote the activation of the input of the jth neuron in the lth layer, $w_{ij}(l)$ denotes the weight between the ith neuron in the lth layer and the jth neuron in the $(l-1)$th layer and f is the nonlinear activation function (sigmoid). The jth component of kth output is

$$\begin{aligned}\rho_{kj}(0) &= \frac{\partial a_k(L)}{\partial a_j(0)} \\ &= \sum_{i=1}^{N_1} \frac{\partial a_k(L)}{\partial a_i(1)} \frac{\partial a_i(1)}{\partial a_j(0)}\end{aligned}$$

where $l = L$ corresponds to the output layer of the neural network. Define

$$\delta_i(l) = \frac{\partial a_k(L)}{\partial a_i(l)}$$

The recursive relation for $\delta_i(l)$ is the same as EBP algorithm. Therefore the recursive relation is

$$\rho_{kj}(0) = \sum_{i=1}^{N_1} \delta_i(1) f'(u_i(1)) w_{ij}(1)$$