3E4-3

An Online Self-constructive Locally Updated Normalized Gaussian Network with Localized Splitting

Jana Backhus Ichigaku Takigawa Hideyuki Imai Mineichi Kudo Masanori Sugimoto

Department of Computer Science and Information Technology,
Graduate School of Information Science and Technology, Hokkaido University

In sequential learning schemes, dynamic model adaptation is preferable over static model size selection. In this paper, we apply an incremental model selection approach to a locally updated Normalized Gaussian network (NGnet), and improve it for better robustness against negative interference. Model adaptation is enabled by applying some unit manipulation mechanisms, including a produce, delete and split manipulation, to increase or decrease model complexity. Here, split uses a static threshold for its manipulation, and we suggest a dynamic thresholding approach that selects a threshold according to local information. In our experiments, the NGnet is tested for a function approximation task with balanced and imbalanced sample distributions. We compared local thresholding to static and dynamic global thresholding, and results show that localized thresholding improves performance for both test cases, and especially for imbalanced sample data. Therefore, localized splitting is preferable especially in test cases where negative interference is likely.

1. Introduction

In sequential learning schemes, only one data sample is observable at a time and prior knowledge about the learning environment is limited. These limitations complicate the learning of neural networks in sequential schemes and the problem of negative interference arises. Negative interference refers to the unwanted forgetting of previously learned information in favor of newly arriving data. The grade of negative interference is a consequence of the distributed nature of the representing model. Local models suffer less from it and are therefore preferable in sequential learning schemes.

A Normalized Gaussian Network (NGnet) is a feed-forward three layer neural network, where the hidden layer consists of local linear regression units. These units partition the input space softly, and the output is linearly approximated within each partition. Recently, a local update approach was proposed for the parameter estimation of the NGnet [Celaya 15]. We will call it hereafter locally weighted NGnet (LWNGnet). Through its local characteristics, the LWNGnet is suitable for the application to sequential learning schemes.

Model complexity selection is another problem that arises during network model learning. A complexity can be selected statically in exhaustive trial-and-error studies or dynamically during learning. Only static model selection was considered in [Celaya 15], but dynamic selection is possible by adapting formerly proposed unit manipulation mechanisms for an NGnet with global weighting (GWNGnet) [Sato 00]. The unit manipulation mechanisms include a produce, a delete and a split mechanism. We apply them self-constructively so that the network model is built from scratch during learning, starting with zero units.

The split mechanism employs a threshold parameter for its split decision which is globally set to the same value for the whole input space and is static over the whole learning process. In this paper, we discuss the localization of the split threshold by suggesting a new localized decision process. We test our proposed method for a function approximation task with balanced and imbalanced sample data distributions. Experiments show that the localized decision process is able to improve performance compared with the former static threshold, especially when the data distribution is imbalanced. Additionally, we compare the self-constructive LWNGnet to a self-constructively built version of the GWNGnet. The results show that the LWNGnet is performing better in both test cases.

Contact: Jana Backhus, Department of Computer Science and Information Technology, Graduate School of Information Science and Technology, Hokkaido University, Kita 14 Nishi 9, Kita-ku, Sapporo, 060- 0814, Japan, jana@main.ist.hokudai.ac.jp

2. Normalized Gaussian Network

The Normalized Gaussian Network (NGnet) is a universal function approximator that was first proposed by Moody and Darken in [Moody 89]. Generally, the NGnet approximates a mapping $f : \mathbb{R}^N \to \mathbb{R}^D$ from an N-dimensional input space to a D-dimensional output space, where an input vector x is transformed to an output vector y with

$$y = \sum_{i=1}^{M} N_i(x)\tilde{W}_i\tilde{x}. \quad (1)$$

M is the number of units, $\tilde{x}$ is an $(N+1)$-dimensional input vector with $\tilde{x}' \equiv (x', 1)$, and $\tilde{W}_i$ is a $D\times(N+1)$-dimensional linear regression matrix. Normalized Gaussian functions N_i are used as activation functions, and N_i is the normalized output of the i-th multivariate Gaussian probability density function (pdf). The model softly partitions the input space into local units i.

2.1 Parameter Estimation

Sato and Ishii formerly suggested an online Expectation-Maximization (EM) algorithm for parameter estimation [Sato 00]. The EM algorithm interprets the NGnet stochastically so that the network model is defined by a probability distribution $P(x, y, i|\theta)$. Here, $\theta \equiv \{\mu_i, \Sigma_i, \sigma_i^2, \tilde{W}_i | i = 1, ..., M\}$ is the set of model parameters which have to be estimated, where μ_i and Σ_i are the center and covariance matrix of the i-th Gaussian pdf, and $\sigma_i^2(t)$ is an output variance for the i-th unit. The parameters are updated with the online EM-algorithm for every time step t:

$$\mu_i(t) = \langle\!\langle x \rangle\!\rangle_i(t) / \langle\!\langle 1 \rangle\!\rangle_i(t) \tag{2}$$

$$\Sigma_i^{-1}(t) = [\langle\!\langle xx' \rangle\!\rangle_i(t) / \langle\!\langle 1 \rangle\!\rangle_i(t) - \mu_i(t)\mu_i'(t)]^{-1} \tag{3}$$

$$\tilde{W}_i(t) = \langle\!\langle y\tilde{x}' \rangle\!\rangle_i(t) [\langle\!\langle \tilde{x}\tilde{x}' \rangle\!\rangle_i(t)]^{-1} \tag{4}$$

$$\sigma_i^2(t) = \frac{[\langle\!\langle |y|^2 \rangle\!\rangle_i(t) - Tr(\tilde{W}_i(t) \langle\!\langle \tilde{x}y' \rangle\!\rangle_i(t))]}{D \langle\!\langle 1 \rangle\!\rangle_i(t)} \tag{5}$$

These updates include weighted accumulators of the form $\langle\!\langle \cdot \rangle\!\rangle_i$ that are calculated in a step-wise equation:

$$\langle\!\langle f(x,y) \rangle\!\rangle_i(t) = \Lambda(t) \langle\!\langle f(x,y) \rangle\!\rangle_i(t-1) + \Omega(t) f(x(t), y(t)) \tag{6}$$

Here, $\Lambda(t)$ has the effect of a forgetting factor that lets old training results be slowly forgotten, while $\Omega(t)$ is an update factor that decides about how much of the newly received data is learned. For the local update strategy [Celaya 15] that is used by the LWNGnet, the forgetting effect is localized to make learning robuster against negative interference. The local forgetting ensures that only as much old information is forgotten as new information is received for a unit i. The forgetting factor is then set to $\Lambda(t) = \lambda(t)^{P_i(t)}$, and the update factor is $\Omega(t) = \frac{1-\lambda(t)^{P_i(t)}}{1-\lambda(t)}$, with a discount factor $\lambda(t)$ and the i-th unit's posterior probability $P_i(t) = P(i|x(t), y(t), \theta(t-1))$.

The discount factor $\lambda(t)$ has to be chosen so that $\lambda \to 1$ when $t \to \infty$ for fullfilling the Robbins-Monro condition for convergence of stochastic approximations. $\lambda(t)$ plays an important role in discarding the effect of old learning results which were employed to an earlier inaccurate estimator.

2.2 Network Model Selection

Model complexity selection is another arising problem when applying NGnets to learning problems. The complexity plays an important role in the NGnet's learning performance and there are two possibilities to choose it. The first possibility is a static complexity set by the user, but a good decision needs excessive trial-and-error studies. The alternative is a dynamic selection of the network model during learning. In other words, the model complexity is changed dynamically with some manipulation mechanisms that increase or reduce it. Here, some manipulation mechanisms are adapted from [Sato 00] to make the LWNGnet's complexity dynamic. The mechanisms include a produce, delete and split mechanism. They are applied self-constructively, referring to a model build-up from scratch during the learning process. Self-construction possesses the advantage that the initialization problem is mostly avoided as the network model starts its training with zero units.

2.2.1 Produce

The probability distribution $P(x(t), y(t)|\theta(t-1))$ indicates how likely the current model parameters $\theta(t-1)$ can estimate the newly received data sample $(x(t), y(t))$. When the probability is smaller than a certain threshold $T_{Produce}$, a new unit is created accordingly with the produce mechanism in [Sato 00].

2.2.2 Delete

The weighted accumulator $\langle\!\langle 1 \rangle\!\rangle_i(t)$ is an indicator for how much the i-th unit has accounted for the data until the current time step t. In the LWNGnet, $\langle\!\langle 1 \rangle\!\rangle_i(t)$ is a weighted sum which is not normalized and therefore cannot be used directly as a reference. A local unit update counter c_{update} is introduced as a normalizer. It is incremented by one at every time step where the update factor is computationally $\Omega(t) > 0$. A unit is deleted if $\langle\!\langle 1 \rangle\!\rangle_i(t)/c_{update} < T_{Delete}$, where T_{Delete} is a delete threshold.

2.2.3 Split

The output variance $\sigma_i^2(t)$ is representing the accumulated squared error between the i-th unit's predictions and the real outputs. High variance values are related to the unit being in charge of a too large partition of the input space. Splitting is applied to a unit i with the split mechanism in [Sato 00] when $\sigma_i^2(t) > T_{Split}$, with threshold T_{Split}.

3. Localization of Split

The split mechanism in 2.2.3 uses a globally defined static threshold that has to be set by the user before learning. But a threshold should be set in relation to the likely accumulation of error which depends among others on the noisiness of the data. An alternative approach compares the output variance of a unit to the output variances of the other units in the network. This can be done globally, with all units included in the comparison, or locally. The global approach may be problematic for different learning situations in different parts of the input space, as it is e.g. the case for imbalanced sampling distributions.

A local approach is suggested to cope with these learning situations. Local means that only a few units near unit i are included in its threshold evaluation. Then, the locality helps to consider only near neighbors which are more likely located in a similar learning situation. The flow of the localized threshold decision process is described in the following.

1: Number of nearest neighbors: $NoNN = M/Div_{NN}$
2: **for all** units i **do**
3: Find the $NoNN$ nearest neighbors of unit i
4: $\sigma_{NNmax}^2 = \max\{\sigma_j^2, j = 1, 2, \cdots, NoNN\}$
5: Calculate split threshold:
$T_{Split} = \sigma_{NNmax}^2 \cdot Multiplier$
6: **if** $\sigma_i^2 > T_{Split}$ **then**
7: Add unit to splitting candidates V_{Split}
8: **end if**
9: **end for**
10: Split all candidates in V_{Split}

Here, Div_{NN} and $Multiplier$ are two variables that have to be set by the user. Div_{NN} is regulating the number of near-

Table 1: Experiment with Balanced Data

Method	b=50		b=150	
	NMSE	Net. Size	NMSE	Net. Size
GW	0.0268	50.52	0.0197	45.82
LW Static	0.0096	47.76	0.0131	46.52
LW Div_{NN}=1	0.0094	48.46	0.0129	47.08
LW Div_{NN}=6	0.0093	50.44	0.0128	49.18

Table 2: Experiment with Imbalanced Data

Method	b=50		b=150	
	NMSE	Net. Size	NMSE	Net. Size
GW	0.4591	41.42	0.3628	37.34
LW Static	0.0663	66.92	0.0674	67.42
LW Div_{NN}=1	0.0664	72.26	0.0662	71.96
LW Div_{NN}=6	0.0625	71.1	0.0656	69

est neighbors involved in the splitting decision in relation to the overall model complexity M. On the other hand, $Multiplier$ regulates when a unit is considered too error-prone compared with its neighbors and has to be split. The nearest neighbors are found based on their distance from unit i. The $NoNN$ units with the shortest distances are considered as nearest neighbors and the Euclidean distance is used for distance calculation.

4. Experiments

For the experiments, we apply the LWNGnet to the Schaal or cross function approximation task which is commonly used to test learning performance (e.g. [Celaya 15], [Sato 00]). The function has the input dimension $N = 2$ and is defined by:

$$g(x_1, x_2) = \max\{e^{-10x_1^2}, e^{-50x_2^2}, 1.25e^{-5(x_1^2+x_2^2)}\}. \quad (7)$$

Additionally, the function output g adds a normally distributed random noise $\epsilon(t) \sim N(0, 0.01)$ so that $y(t) = g(x_1(t), x_2(t)) + \epsilon(t)$ is obtained as the noisy data sample output. The function approximation task is tested with balanced and imbalanced sampling of 10.000 training data samples. For the balanced test case, the sample data is independent and identically distributed (i.i.d) over the whole input space $(-1 \leq x_1, x_2 \leq 1)$. The imbalanced test case uses non-identically distributed sample data. 95% of the data samples are extracted from a sub-region of the input domain with $(0 \leq x_1, x_2 \leq 0.25)$, while the remaining 5% of the data are i.i.d in $(-1 \leq x_1, x_2 \leq 1)$.

Four different methods are compared in our experiments. The first method is GW, the conventional online NGnet with globally weighted forgetting [Sato 00]. It is applied self-constructively and uses the static split threshold. The second to forth method uses the LWNGnet, where $LW Static$ uses the static split threshold and the other two use the split decision process described in the 3. Chapter. For $LW\ Div_{NN}=1$, the number of nearest neighbors is equal to the whole input space. In other words, this method applies the decision process globally, while $LW\ Div_{NN}=6$ uses only some nearest neighbors for its locally oriented decision. Thresholds were chosen so as to have approximately similar behavior in all methods for better comparison. The discount factor is updated with $\lambda(t) = 1 - \frac{0.99}{0.01t+b}$ dependent on time step t and a variable b. We will cover two different values of b ($b = 50$, $b = 150$) in our experiments. The learning performance is evaluated with the Normalized Mean Square Error (NMSE) for all methods.

Experimental results for the balanced case are presented in Table 1. The results show similar performance for all LW methods, while they all perform better than GW. A different picture is drawn for the second experiment with imbalanced data. The experimental results are presented in Table 2. Again, the LW methods outperform GW. This time, there is an even bigger difference in performance and also in model complexity which is related to the different learning approaches with globally and locally weighted forgetting. GW forgets old information based on a global time-based forgetting factor, making it difficult to achieve good performance when sample data is imbalanced. This was also discussed for static model complexity in [Celaya 15]. Among the LW methods, the local threshold method $LW\ Div_{NN}=6$ shows the best performance. The performance gap is bigger than for the balanced data and emphasizes the advantage of the local method. Overall, the local split decision performs best out of the compared methods.

5. Conclusion

In this paper, we suggested a localized threshold decision for a split unit manipulation mechanism. It is applied together with other manipulations for dynamic model selection of the LWNGnet. The proposed approach is tested for a function approximation task with two different cases where the sample data distribution is balanced and imbalanced. Our experimental results show that the localized threshold decision helped to improve learning performance, especially for the imbalanced data case. The LWNGnet was also compared with an older version of the NGnet and showed greater stability in learning performance. Possible future works include the application of the proposed approach to real world systems and the automatisation of the unit manipulation threshold parameter selection.

References

[Celaya 15] Celaya, E., Agostini, A.: On-line EM with Weight-Based Forgetting. Neural Comput. 27, no.5, 1142–1157 (2015)

[Moody 89] Moody, J., Darken, C. J.: Fast learning in networks of locally-tuned processing units. Neural Comput. 1, no.2, 281–294 (1989)

[Sato 00] Sato, M., Ishii, S.: On-line EM algorithm for the normalized Gaussian network. Neural Comput. 12, no.2, 407–432 (2000)