
Distributed Backpropagation-Decorrelation Learning

Oliver Obst
CSIRO ICT Centre
Autonomous Systems Lab
PO Box 76, Epping NSW 1710, Australia
oliver.obst@csiro.au

Abstract

In long-term deployments of sensor networks, monitoring the quality of gathered data is a critical issue. Over the time of deployment, exposure to harsh condition may cause sensors to degrade or to fail. If such a degradation remains undetected, the usefulness of a sensor network is greatly reduced. We introduce SODBPDC, a distributed recurrent network architecture, and a method to learn spatio-temporal correlations between different sensors for fault detection in a distributed way. Our approach is evaluated using real sensor network data, and proves to work well with less-than-perfect link qualities and more than 50% of failed sensors.

1 Introduction

Wireless sensor networks (WSN) are increasingly used for environmental monitoring over extended periods of time [1]. To facilitate deployments in remote areas, sensor nodes are typically small, solar-powered devices with limited computational capabilities. Over the duration of the deployment, harsh weather conditions can lead to problems like mis-calibration or build-up of dust on sensors and solar panels, leading to incorrect readings or shorter duty-cycles and thus less data. Our goal is to automatically learn the normal system behaviour and to use this model to detect anomalies.

In our approach, sensor nodes participate in a distributed recurrent neural network, where each of the sensor nodes hosts a few units and communicates only with its local neighbours. Our online learning is a variant of backpropagation-decorrelation (BPDC) learning [2] with intrinsic plasticity [3] (IP). In a similar setting, we have proposed a distributed fault detection [4] based on echo state learning [5], but the offline learning approach is computationally too demanding to be directly executed on sensor nodes. Our new *Spatially Organised and Distributed Backpropagation-Decorrelation* (SODBPDC) architecture and learning algorithm (Section 2) is suited for directly learning on sensor nodes because of a smaller memory footprint than echo state learning during training. In Sect. 3, we present results of an application of SODBPDC to fault detection in sensor network data.

2 Approach

WSN spend a large part of their energy on communication between individual nodes. Routing data between distant nodes involves participation of intermediate nodes (“multi-hop”), and thus further increases energy consumption. To distribute a recurrent neural network over a WSN, we let each node host some units of an entire neural network¹. We restrict connections between units to those hosted on the same node or on nodes in the immediate spatial neighbourhood. This results, on each device, in small local reservoirs with local input units and output units with additional connections between neighbours. From a global perspective, we obtain a spatially organised reservoir, which we train using a distributed version of BPDC learning.

¹We use *unit* for components of a neural network, and *node* for devices in a sensor network.

1. Generate $M_q = L_q + N_q + K_q$ local units (output/hidden/input units)
2. For each neighbour p , create
 - M_q proxy units
 - M_q connect. to neighbour proxy units
3. Generate the local weight matrix \mathbf{W}^q for connections between local units with sparseness s_1 .
4. Add connections from local proxy units to other local units to \mathbf{W}^q with sparseness s_2 .

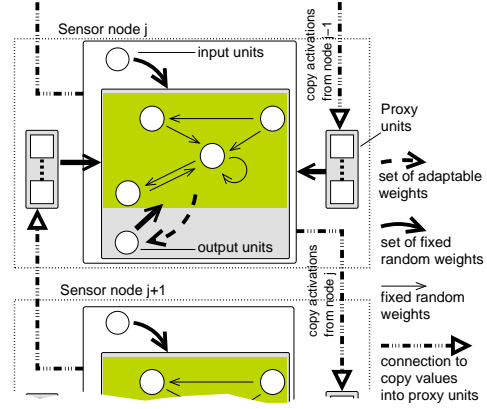


Figure 1: Setup of SODBPDC networks for each sensor node

Each sensor node q hosts the same number M_q of units, namely L_q output units, N_q hidden units, and K_q input units. The whole recurrent network consists of M units, i.e. L output units, N hidden units, and K input units. For first theoretical considerations, it is convenient to represent activations as a global vector \mathbf{x} :

$$\mathbf{x} = \left(\underbrace{x_1, \dots, x_{L_1}}_{\text{outputs units 1}}, \dots, \underbrace{x_{L-L_Q}, \dots, x_L}_{\text{output units } Q}, \right. \\ \left. \underbrace{x_{L+1}, \dots, x_{L+N_1}}_{\text{hidden units 1}}, \dots, \underbrace{x_{M-K_Q}, \dots, x_M}_{\text{input units } Q} \right).$$

Likewise, synaptic connection weights between units can be represented in a global $M \times M$ matrix $\mathbf{W} = (w_{ij})$. Activations are updated in a distributed way as in the non-distributed version of BPDC [2]

$$\mathbf{x}(k+1) = \mathbf{W} \mathbf{f}(\mathbf{x}(k)), \text{ with } f_i(x) = \frac{1}{1 + \exp(-a_i x_i - b_i)}$$

for each time step k , so that each node computes all local $x_i(k+1)$.

2.1 Proxy units

In a practical implementation, both \mathbf{W} and \mathbf{x} are distributed over multiple sensor nodes. Moreover, there are connections in \mathbf{W} between units on different devices, which require to have a specified physical location. We store the incoming connections from units hosted on neighbour sensor nodes on the local node. Units with outgoing connections to units on other devices just forward their activations with no changes to the neighbour device. Additional proxy units on the neighbour act as a place holder for remote units and take activations from connected units (see Fig. 1). From proxy units, there are only local connections to the reservoir or to output units. Proxy units also eliminate the need for all sensor nodes being tightly synchronised, as long as they all use the same interval to process data — typical update frequencies are very low, e.g. once every minute or once every 15 minutes. Newly computed activations are forwarded to connected proxy units where they can be used by the neighbour device. After their values have been used, proxy units are reset to 0 in order to avoid using old values in case of a sensor network link failure.

2.2 SODBPDC online learning

Each sensor node is responsible for updating its local output weights. Let \mathbf{x}^q denote the vector of activations in \mathbf{x} which can be accessed locally on node q either directly or by reading out a proxy unit. Let the set O contain indices j of output units, $\hat{O} \subset O$ the local output units, and $g : O \rightarrow \hat{O}$ a mapping from global to local unit indices. The SODBPDC learning rule is executed on each local

sensor node and updates the global matrix \mathbf{W} :

$$\Delta w_{ij}(k+1) = \frac{\eta f(x_{g(j)}^q(k))}{\sum_s f(x_{g(s)}^q(k))^2 + \epsilon} \gamma_i(k+1),$$

where $\gamma_i(k+1) =$

$$\sum_{s \in \tilde{O}} (w_{is} f'(x_{g(s)}^q(k))) e_{g(s)}(k) - e_{g(i)}(k+1).$$

2.3 Training for distributed fault detection

In our application to detect sensor faults, the task is to predict local sensor readings based on information from other nodes. We expect a reading and its prediction to be approximately equal when the sensor works normally. Faults are detected when the difference between the two exceeds a specified threshold. During the initial training period, we assume there are no sensor faults, so that the training output for each sensor is exactly the same as the input time series.

We aim to learn a model of all sensors simultaneously, but independent of the local input. To this end, we create an identical copy of the local recurrent network on each node. The original instance, the primary network, is connected to the local networks on neighbour nodes as described above, and receives normal input from its local sensors. The global network joining all local primary instances with activations \mathbf{x}^q has an activation \mathbf{x} . In \mathbf{x} , all input units are sensor readings at all times. The second instance, the shadow network, has only incoming connections from primary networks on neighbour nodes, but does not forward its local activations $\hat{\mathbf{x}}^q$ to any other node. Local input units of the shadow network are fed with random noise. This results in an individual global activation \mathbf{x}^q for each node q . In each \mathbf{x}^q , there is no correlation between local input and the local training signal. The primary network is responsible to feed its activations to neighbour nodes. Both the SODBPDC rule and IP are applied to the shadow network. After application of learning rules, a consolidation step copies output weights and local IP learning parameters from the shadow network back to the primary network (Fig. 2).

3 Experimental results

Our training data are time series from a sensor network deployment in Belmont, near Brisbane, Australia with 32 sensor nodes. Because the data were collected by forwarding to a central node, it contained ‘‘holes’’ as a result of duty cycling. We resampled smaller gaps by interpolation, and left the larger and network-wide gaps in the data. Our application is to monitor the condition of solar panels by measuring the solar voltage on each device. In all our experiments, the SODBPDC network consisted of 32 output units, 160 hidden units, and 32 input units (i.e. 1/5/1 units on each node).

1. Create a shadow network without any outgoing connections to neighbours.
2. Until trained:
 - feed sensor data into the primary network.
 - feed white noise into the shadow network.
 - update local activations \mathbf{x}^q and $\hat{\mathbf{x}}^q$.
 - update weights in the shadow network.
 - apply IP learning to the shadow network.
 - reset all proxy units to 0.
 - copy weights and IP parameters from shadow network back to primary network.
3. Delete the shadow network.

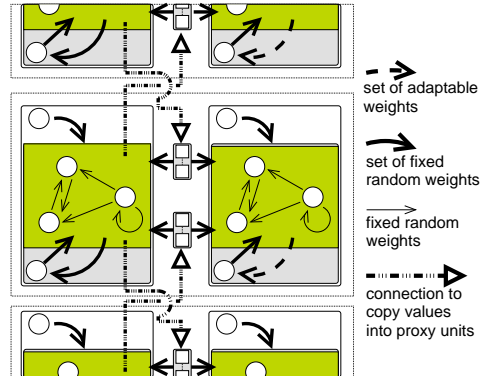


Figure 2: Training of SODBPDC networks for distributed fault detection

Example 1 — comparison to a centralised approach using BPDC and IP learning. We compared the prediction errors of our approach against an approach using a centralised reservoir of the same size. For both approaches, we simulated link qualities from 10% to 100%. In the distributed approach, this represents the probability of communication between any two nodes, and in a centralised setting the communication probability from any node to central reservoir. The centralised reservoir was used to predict the solar voltage of one node, taking the solar voltage of the remaining 31 nodes as an input. SODBPDC learns slower in the beginning, very likely an effect of the additional random noise signal. For high link qualities, SODBPDC performs equally or better than BPDC, but BPDC manages to maintain lower NRMSE even for poor link qualities.

Example 2 — robustness against multiple failures. In this experiment we randomly selected an increasing number of sensors to fail, and let faulty sensors return 25% of the original value. We were interested in the prediction error for the remaining healthy nodes, when (a) the faulty sensors continued to feed data into the network, and (b) the input from faulty sensors was replaced with (recursive) predictions. We trained all nodes using the first 43000 values of the time series. Using recursive predictions manages to keep the error very low for up to 29 (out of 32) failed sensors.

4 Conclusion

In this paper, we presented SODBPDC, a novel distributed recurrent neural network architecture based on a reservoir computing approach. We introduced an online learning approach for SODBPDC closely related to training BPDC. We then presented an approach to train SODBPDC for fault detection in WSN, where predictions of sensor values are made based on information from neighbour nodes. In our examples, we have shown that the approach helps to solve a real-world problem, and its robustness against multiple sensor faults. Experiments not presented in this paper also showed that SODBPDC learns quicker than our previous approach based on spatially organised, distributed echo state learning [4], which, on the other hand, is more robust to poor link qualities.

References

- [1] Peter Corke, Philip Valencia, Pavan Sikka, Tim Wark, and Les Overs. Long-duration solar-powered wireless sensor networks. In *EmNets '07: Proceedings of the 4th workshop on Embedded networked sensors*, pages 33–37, New York, NY, USA, 2007. ACM.
- [2] Jochen J. Steil. Backpropagation-decorrelation: Recurrent learning with $O(N)$ complexity. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 1, pages 843–848, 2004.
- [3] Jochen J. Steil. Online reservoir adaptation by intrinsic plasticity for backpropagation-decorrelation and echo state learning. *Neural Networks*, 20(3):353–364, April 2007.
- [4] Oliver Obst. Poster abstract: Distributed fault detection using a recurrent neural network. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN 2009)*, pages 373–374. IEEE Computer Society, April 2009.
- [5] Herbert Jaeger. The “echo state” approach to analysing and training recurrent neural networks. GMD Report 148, GMD – German National Research Institute for Computer Science, 2001.