PF-LSTM: Belief State Particle Filter for LSTM

Xiao Ma **Peter Karkus** **David Hsu** **Wee Sun Lee**

School of Computing
National University of Singapore

{xiao-ma, karkus, dyhsu, leews}@comp.nus.edu.sg

1 Introduction

Learning generative models of sequential data under uncertainty is a long-standing challenge. There are two general approaches to sequence modeling: the model-based inference approach, *e.g.*, Dynamic Bayesian Networks (DBNs) [Robert, 2014] and the model-free learning approach, *e.g.*, Recurrent Neural Networks (RNNs). DBNs require hand-crafted state representation and a model for inference, but given a task, crafting an effective state representation and learning an accurate model are challenging. In contrast, RNNs can be directly trained from data, but lack a structured representation to effectively capture uncertainty in many real-world applications.

In this paper, we propose a structured LSTM network for learning to model data sequences with high uncertainty. Particle Filter LSTM (PF-LSTM) combines the LSTM and the particle filter: we treat the LSTM hidden state as a random variable, approximated by a set of a *articles*, and update the particles through LSTM gated structures. Each particle conceptually represents a hypothesis for the hidden state. A complex hidden distribution can be approximated by a set of simpler particles. Unlike the LSTM, which only maintains a single deterministic hidden state, PF-LSTM maintains a *hidden belief state*, represented by the set of particles, which capture all the possible configurations of the current state. This belief representation enables PF-LSTM to better handle the data sequences with uncertainty, especially sequences with multi-modal distributions, which are often difficult for LSTMs. To model uncertainty in particle transition and observations, we introduce *stochastic memory update*. More specifically, the PF-LSTM memory is updated by the samples drawn from a distribution from data.

Besides the structured memory for handling sequence uncertainty, as a variant of LSTM, PF-LSTM has another two advantages: (1) since the network parameters are shared by all the particles, we can approximate a hidden state with a set of simpler particles, which leads to a smaller network. (2) unlike the LSTM which has to learn to model and update the complex data distribution itself, PF-LSTM benefits from the well-established particle filter algorithm that in some sense guide the PF-LSTM to filter its hidden distribution from the very beginning.

2 PF-LSTM

In LSTM, the hidden state at time t is defined by a tuple (h_t, c_t), where h_t stands for the hidden state used as the output and c_t stands for the cell state, which is normally regarded as the memory. PF-LSTM replaces the LSTM hidden state by a set of hidden particles, $\{h_t^i, c_t^i, w_t^i\}_{i=1}^k$, where w_t^i is the corresponding weight for particle i. At each time step, we merge the particles in the hidden state space and produce one *estimated mean hidden state* by $\bar{h}_t = \sum_{i=1}^{k} w_t^i h_t^i$. Merging the particles in hidden state space is more reasonable than merging outputs because the particles are defined over the hidden state space and this has demonstrated better performance in our experiments.

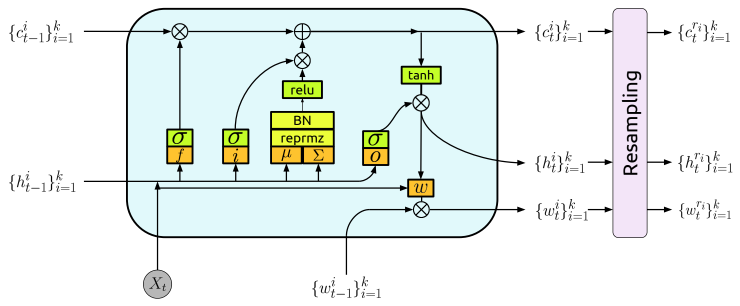

Figure 1: PF-LSTM Network Structure

In particular, to account for the randomness in real-world data, the PF-LSTM implements a stochastic cell update. Different from the deterministic mapping by a fully connected layer in LSTM, PF-LSTM assumes the updates to the memory follow a learned Gaussian distribution. The updates are sampled from the learned distribution then passed through a batch normalization layer followed by a ReLU for faster training [Ravanelli et al., 2018].

Particle weight w_t^i is learned discriminatively by a mapping from current input x_t^i and previous output, h_{t-1}^i. The discriminative setup enables the training to focus only on the features that are helpful for a good prediction, instead of learning a good model for the input data.

To make the particle filter efficient, every time step we sample a new set of particles with indices $\{r_i\}_{i=1}^k$. Normal resampling strategy will block the gradient and disable the end-to-end training. We adopt the same strategy with the Particle Filter Networks [Karkus et al., 2018] and use the *soft resampling*. Instead of sampling from the particle distribution $p_t(k) = w_t^k$, we sample a new set of indices $\{r_t^i\}_{i=1}^k$ from a softened distribution $q_t(k) = \alpha w_t^k + (1-\alpha)\frac{1}{K}$ where α is a hyper-parameter. The new weight are set to be $w_t^{\prime r_t^i} = \frac{w_t^{r_t^i}}{\alpha w_t^{r_t^i} + (1-\alpha)1/K}$.

To train the PF-LSTM, we optimize the parameters from two aspects: we want the prediction produced by the $\bar{h}_t$ to be good, and we also want to learn a good belief distribution, i.e., give each particle the ability to represent different hypotheses. We define the loss function to be a combination of the *prediction loss* and the *belief loss*, $L = \alpha L_{pred}(\bar{h}) + \beta \sum_{i=1}^{t} w^i L_{pred}(h^i)$, where α and β are two hyper-parameters to balance the two losses.

3 Experiments

In our preliminary experiments, we tested on two tasks from completely different domains: text classification and visual odometry.

In text classification, we treat the text as a time series, use each word as input, and predict a likelihood of the label using the last estimated mean hidden state. Although this task looks simple, it is in fact coupled with high uncertainty. Given the first k words, the $k+1$ words could have multiple candidates. For example, for sentence "I want to", the next words could have almost infinite candidates. Besides, some words could have different meanings, which also introduces the noise. In our experiment, the classification accuracy of PF-LSTM is 9% higher than LSTM, and 2.7% higher than one of the state-of-the-art text classifier, KimCNN [Kim, 2014].

In visual odometry, at each time step, we input two consecutive video frames to the RNN and predict translational and angular velocity of the vehicle. We experimented on the KITTI visual odometry dataset [Geiger et al., 2012] and compared with two baselines: LSTM based approach and Differentiable Particle Filters [Jonschkowski et al., 2018]. Although the velocity normally follows a unimodal distribution and is typically handled by Kalman Filter based approaches [Civera et al., 2010], we found that the structured memory of PF-LSTM still helps to improve the LSTM performance by around 40%.

Detailed experiment results can be found in the appendix.

References

Ana Cardoso-Cachopo. Improving Methods for Single-label Text Categorization. PdD Thesis, Instituto Superior Tecnico, Universidade Tecnica de Lisboa, 2007.

Javier Civera, Oscar G Grasa, Andrew J Davison, and JMM Montiel. 1-point ransac for extended kalman filtering: Application to real-time structure from motion and visual odometry. *Journal of Field Robotics*, 2010.

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

Rico Jonschkowski, Divyam Rastogi, and Oliver Brock. Differentiable particle filters: End-to-end learning with algorithmic priors. *arXiv preprint arXiv:1805.11122*, 2018.

Peter Karkus, David Hsu, and Wee Sun Lee. Particle filter networks with application to visual localization. *arXiv preprint arXiv:1805.08975*, 2018.

Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2005.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

Mirco Ravanelli, Philemon Brakel, Maurizio Omologo, and Yoshua Bengio. Light gated recurrent units for speech recognition. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2018.

Christian Robert. Machine learning, a probabilistic perspective, 2014.

A Appendix: Detailed Experiment Results

Followings are the details results of our preliminary experiments[1].

A.1 Text Classification

We experimented on the R52 dataset [Cardoso-Cachopo, 2007] and MR dataset [Pang and Lee, 2005]. The R52 dataset has 52 labels, while the MR dataset is for sentiment analysis that has only two labels. We use a very simple network: the sentence is first passed through a word-level embedding layer, then directly input to the PF-LSTM. To predict the label, a fully connected layer is used. The network is implemented using PyTorch [Paszke et al., 2017].

By default, we use the PF-LSTM of 20 particles and all the proposed components, including the stochastic input update, ReLU activation with batch normalization, resampling, and auxiliary belief loss training. We then conduct the ablation study over each component and analyze how does the number of particles influence the performance:

- PF-LSTM-P5 / P10 / P30: the PF-LSTM with 5, 10 or 30 particles.
- PF-LSTM-NoBelief: the PF-LSTM without the belief loss.
- PF-LSTM-NoRVReLUBN: the PF-LSTM without the stochastic cell update, the ReLU activation, and batch normalization.
- PF-LSTM-NoResamp: the PF-LSTM without resampling.

[1]The reported results were obtained with a slightly different resampling process. We verified that the results are quantitatively similar, but we did not update the results due to time constraints.

Table 1: Text Classification Accuracy on R52 Dataset

Models	Sentence Lengths		
	32	64	128
LSTM	0.75	0.781	0.811
LSTM1000	0.411	0.437	0.521
StackLSTM	0.76	0.784	0.81
BiLSTM+Att	0.834	0.841	0.845
KimCNN	0.84	0.861	0.874
PF-LSTM-P5	0.832	0.866	0.879
PF-LSTM-P10	0.844	**0.873**	0.887
PF-LSTM-P30	0.829	0.866	0.882
PF-LSTM-NoBelief	0.826	0.865	0.882
PF-LSTM-NoRVReLUBN	0.804	0.817	0.837
PF-LSTM-NoResamp	0.799	0.831	0.872
PF-LSTM-KL	0.805	0.846	0.866
PF-LSTM	**0.853**	0.871	**0.901**

- PF-LSTM-KL: the PF-LSTM with the extra KL-Divergence to regularize the stochastic cell update. This is designed to explain why we do not add any constraint to the stochastic cell update. We wish to show that the extra KL loss will damage the PF-LSTM performance.

The results of the experiments on R52 dataset are given in Table 1. We observe that the PF-LSTM, with only a simple structure, outperforms all the baselines with all sentence lengths, including the state-of-the-art KimCNN.

In particular, we can have the following observations:

- PF-LSTM outperforms the LSTM in text classification problem. It is reasonable because the PF-LSTM maintains a belief state over the sentence. For LSTM, it theoretically has the ability to approximate any given distribution but fails in this example. It also explains why the stochastic representation of the hidden state is useful for handling the sequence with uncertainty
- The LSTM1000, which has the same memory size with PF-LSTM but much larger parameter space, works worst among all the baselines. It is possibly because the parameter space becomes too large to train with the limited data. In this case, using the PF-LSTM structured memory would be a better choice.
- StackLSTM and BiLSTM with attention all share some ideas with PF-LSTM: the hierarchical structure of memory and attention mechanism. However, these methods have a similar problem with LSTM: they still suffer from the long sequence length. Given more information (longer text sequence), these models, however, are not able to aggregate it effectively. This is because the uncertainty accumulates if we use a deterministic representation of the states, so it is hard for them to gain much improvement from the longer sequence.
- Comparing the PF-LSTMs with the different number of particles, we observe that more particles indeed increase the performance, but not always. When we increase the number of particles from 5 to 20, the performance keeps increasing. This is reasonable because the more particles, the more possible to get a good approximation of a complex belief distribution. However, using 30 particles does not seem to have further benefit. One possible reason is with more particles, it becomes harder for the PF-LSTM to learn a good set of weights.
- Comparing the PF-LSTM with the PF-LSTM-NoBelief, we could see that the belief loss could increase the PF-LSTM's performance. It is because the belief loss forces the model to learn a better particle distribution, instead of simply optimize for the objective.
- The PF-LSTM-NoResamp performs the worst among all the PF-LSTM based models when the sentence length is 32. This implies the resampling is indeed necessary for an effective particle filtering, even if the particles are in the hidden state space.

Table 2: Text Classification Accuracy on MR Dataset

KimCNN (Reported)	PF-LSTM			
	p=5	p=10	p=20	p=30
0.761	0.739	**0.783**	0.772	0.752

- The PF-LSTM-KL works worse than many PF-LSTM based models. The extra constraint on the hidden distribution turns out to make the particles hard to diverge from each other, so the particles cannot be utilized efficiently.
- The PF-LSTM works even much better than the KimCNN in this example. The KimCNN is a well-established architecture but due to the nature of the convolutional neural networks, it still not robust enough to noise and is still restricted to local dependencies.

The results of the experiment on the MR dataset are given in Table 2. We can see the PF-LSTM still outperforms the KimCNN on another challenging dataset. What's interesting is that actually, the PF-LSTM with 10 particles works the best. Similar to the result on the R52 dataset, we observe that the more particles do not necessarily improve the performance, which again supports our analysis on the R52 dataset, that because each particle itself is already a hidden distribution which could capture some distributions, a large number of the particles, do not necessarily give benefits.

A.2 Visual Odometry

In this experiment our main objective is to compare the LSTM and PF-LSTM. The same CNN structure is used for both approaches to extract visual features. The outputs of the network are two values: translational velocity and angular velocity. More specifically, we assume the initial state of the car is known, then use the learned velocity to update the car's (x, y) coordinates, heading direction and velocities.

We follow the experiment setting of the Differentiable Particle Filters (DPFs)[Jonschkowski et al., 2018], which embeds a fully differentiable particle filter algorithm into the neural network, specialized for state estimation. The state representation in DPF is manually chosen, as well as the transitions. In contrast, the PF-LSTM learns a hidden state representation from data, and thus it relies on significantly less prior knowledge.

Table 3: Visual Odometry Experiment Result for 100m

Models	Angular Loss (°/m)	Translational Loss (m/m)
LSTM	0.229	0.313
PF-LSTM	0.141	0.170
DPF (Reported)	0.049	0.147

We randomly segment the visual sequence into subsequences of length 7 (the frame rate is approximately 1 frame per meter) and augment the dataset by horizontally flipping the image and the corresponding coordinates. The trained network is then tested on subsequences of length 100, where we reset the hidden state of RNN every 7 frames, since the network is being trained with subsequences of length 7, and longer sequence input will lead to worse performance.

The result of the visual odometry experiment is given in Table 3. Compared with LSTM, PF-LSTM significantly improves the performance even on this uni-modal problem, since even if the problem is uni-modal, the motion of the vehicle is still coupled with noise, which is difficult for the LSTM.

However, when compared to DPF, PF-LSTM performs worse. DPF builds on the assumption that the motion of the vehicle follows a Gaussian distribution. This assumption is very suitable and it is a strong prior knowledge which greatly simplifies the learning. On the contrary, the PF-LSTM has less prior knowledge, so it is understandable that PF-LSTM performs worse. We believe that the real benefit of PF-LSTM is in domains where finding a good model and state representation is difficult.