Akademie věd České republiky
Ústav teorie informace a automatizace, v.v.i.

Academy of Sciences of the Czech Republic
Institute of Information Theory and Automation

RESEARCH REPORT

Lukáš Ulrych, Václav Šmídl

DEnFi: Deep Ensemble Filter for Active Learning

No. 2383 8 June 2020

GAČR No. GA18-21409S

ÚTIA AV ČR, P.O.Box 18, 182 08 Prague, Czech Republic
Tel: (+420)266052422, Fax: (+420)286890378, Url: http://www.utia.cas.cz, E-mail: utia@utia.cas.cz

Abstract

Deep Ensembles proved to be a one of the most accurate representation of uncertainty for deep neural networks. Their accuracy is beneficial in the task of active learning where unknown samples are selected for labeling based on the uncertainty of their prediction. Underestimation of the predictive uncertainty leads to poor exploration of the method. The main issue of deep ensembles is their computational cost since multiple complex networks have to be computed in parallel. In this paper, we propose to address this issue by taking advantage of the recursive nature of active learning. Specifically, we propose several methods how to generate initial values of an ensemble based of the previous ensemble. We provide comparison of the proposed strategies with existing methods on benchmark problems from Bayesian optimization and active classification. Practical benefits of the approach is demonstrated on example of learning ID of an IoT device from structured data using deep-set based networks.

1 Introduction

Representation of uncertainty of prediction of a neural network is a long studied topic with many available methods. While exact methods such as Hamiltonian Monte Carlo [10] are available, they are too computationaly expensive to run on deep neural networks. Therefore, more affordable approximations in the form of Dropout MC [6], Stochastic Gradient Descent with Langevine Dynamics [14], and many other alternatives have been proposed. However, recent empirical evidence [1, 11, 5] suggests that most of them suffer from underestimation of uncertainty. One of the most attractive option for uncertainty representation thus remain the deep ensemble approach [8] which uses different networks to approximate different local minima. Diversity of the ensemble is obtained by starting from different initial estimates. This simple approach has a drawback in higher computational cost compared to its simpler alternatives such as dropout. This issue has been addressed by rank-1 approximation

2 Recursive Uncertainty estimation in Active Learning

We are concerned with the task of active learning [2] as an extension of semi-supervised learning. We start with the set of pairs of input data $D_0 = \{x_i, y_i\}_{i=1}^{n_0}$, where x_i is a feature vector and y_i is response variable (continuous in the case of regression and discrete in the case of classification) and the set of feature vectors $U_0 = \{x_j\}_{j=1}^{n}$ for which we can potentially obtain the response variable y_j if we ask a human expert or perform an experiment. Each of such requests comes with a significant cost, therefore we aim to learn the model with as low number of requests as possible. The selection of the point x_1^*, where a new response is requested, is thus formulated as an optimization task. When a new y_1^* is obtained, it is added to the training set $D_1 = D_0 \cup d_1^*, d_1^* = \{x_1^*, y_1^*\}$ and

the feature vector removed from the unlabeled set $U_1 = U_0 \backslash x_1^*$. This procedure is repeated forming a sequence of supervised learning tasks with growing set $D_0 \subset D_1 \subset D_2 \cdots D_n$.

Formally, the task is rather simple. We choose a parametric model (such as deep NN) with parameter θ, evaluate its posterior probability $p(\theta|D_k)$, define acquisition function $a(x)$, typically in the form of an expected value,

$$a_k(x) = \int l(x, \theta) p(\theta|D_k) d\theta, \tag{1}$$

where $l(x, \theta)$ is a chosen loss function. The feature vector for which we request a response is found as the maximum of the acquisition function

$$x_k^* = \arg \max_{x \in U_k} a_k(x).$$

Difficulty with this approach arises in the evaluation of the posterior distribution $p(\theta|D_k)$ and the expectation (1). Despite recent advances in Bayesian neural networks, it is still a complicated task, requiring substantial computational effort. While techniques such as HMC [13], SGLD [14] and many updated versions exist, they are quite expensive to run repeatedly. Moreover, increasing evidence suggests that most of them still underestimate uncertainty and the best choice remains to be the ensemble of models [snoek].

We note that the use of uncertainty in active learning is rather specific. Once the new data point is selected, the data set is modified and the estimation procedure has to be repeated. The existing ensemble techniques are not designed to take advantage of the incremental nature of the Bayes' rule used in active learning:

$$p(\theta|D_k) \propto p(x_k^*, y_k^*|\theta) p(\theta|D_{k-1}). \tag{2}$$

Recursive evaluation of incremental data (2) is known as recursive estimation or Bayesian filtering. While there is a number of available techniques, their systematic use in active deep learning is not explored in detail.

3 Ensemble Filter

The basic idea of Ensemble methods is to approximate the posterior distribution $p(\theta|D_k)$ by an ensemble, i.e. a set of samples:

$$p(\theta|D_k) \approx p_\delta(\theta|D_k) = \frac{1}{N} \sum_{i=1}^{N} \delta(\theta - \theta_k^{(i)}), \tag{3}$$

where N is the number of ensemble members. Various techniques differ in a way how to generate the samples, ranging from different initial conditions [8], to Monte Carlo estimates such as [14] or [9]. To our best knowledge methods combing deep ensembles with importance sampling have not been investigate despite a number of advanced importance sampling methods [3].

Under the importance sampling procedure the ensemble weights are not uniform

$$p(\theta|D_k) \approx p_\delta(\theta|D_k) = \sum_{i=1}^{N} w_k^{(i)} \delta(\theta - \theta_k^{(i)}),\ \sum_i w_k^{(i)} = 1. \tag{4}$$

This is a consequence of not sampling $p(\theta|D_k)$ but an (almost) arbitrary proposal density $q(\theta|D_k)$ and compensating for mismatch between p and q by weighting

$$\tilde{w}_k^{(i)} \propto \frac{p(D_k|\theta_k^{(i)})p(\theta = \theta_k^{(i)})}{q(\theta_k^{(i)}|D_k)},\ \ w_k = \frac{\tilde{w}_k^{(i)}}{\sum_i \tilde{w}_k^{(i)}}, \tag{5}$$

where the second equality is a normalization required for (4) being normalized.

Quality of approximation of the posterior greatly depends on proximity of the proposal function $q()$ to the true posterior. Since it is problematic to define on general problems, importance sampling is less popular than Monte Carlo counterparts. The most obvious exception is the recursive estimation, where availability of posterior distribution from the previous step, $p(\theta|D_{k-1})$ in our case, provides a lead for design of efficient proposals as demonstrated by success of sequential Monte Carlo methods for Bayesian filtering [4]. While our problem differs from conventional SMC problem since θ does not change with the data (i.e. it is a parameter not a state [7]), we will use ideas from the filtering literature. Therefore, we call the proposed method Deep Ensemble Filter (DEnFi).

3.1 Deep Ensemble Filter (DEnFi)

The core idea of the proposed filter is a novel way how to design a proposal function. The traditional "bootstrap" filter introduce a kernel around the latest ensemble members i.e. $q(\theta|D_k) = \sum_i \mathcal{N}(\theta_{k-1}^{(i)}, \sigma_k I)$. However, this is known to require too many samples. An alternative is to adapt each ensemble member to the new data point $\hat{\theta}_k^{(i)} = \arg\max p(D_k|\theta)$ starting from $\theta_{k-1}^{(i)}$ and to use proposal

$$q(\theta|D_k) = \sum_i \mathcal{N}(\hat{\theta}_k^{(i)}, \sigma_k I), \tag{6}$$

this is reminiscent of the auxiliary particle filter [Pitt]. However, due to special features of the probability landscape of deep models [Fort], we conjecture, that it is unable to escape from the local minima. Therefore, we propose a two stage approach:

1. Sample initial point of, $\tilde{\theta}_k^{(i)} \sim \mathcal{N}(\theta_{k-1}^{(i)}, Q_k^{(i)})$,
2. Estimate centroids of the proposal components

   $$\hat{\theta}_k^{(i)} = \arg\max p(D_k|\theta),$$

   starting from $\tilde{\theta}_k$.

3. Sample from the proposal $q(\theta|D_k) = \sum_i \mathcal{N}(\hat{\theta}_k^{(i)}, R_k^{(i)})$

4. Compute importance weights (5) and resample.

Consider distribution $p(\theta|D_{k-1})$ to be in the form (3) with ensemble members $\Theta_{k-1} = \{\theta_{k-1}^{(i)}\}_{i=1}^N$. The resampling operation is required to prevent degeneration of the weights to zero. The resampling operation generates ensemble with equal weights such that it matches the cumulative density of the weighted ensemble as close as possible []. In essence it duplicates samples with high weight and removes those with low weight. The number of samples in step three can differ from the number of ensemble members. In that case, the resapling operation is generating only the N ensemble members.

3.1.1 Covariance structure

The most simple proposal is constant variance $Q = qI$ and $R = rI$ where q and r are hyperparameter of the method.

A more complex proposal is to make diagonal covariance structure $Q = \mathrm{diag}(\boldsymbol{g})$, $R = \mathrm{diag}(\boldsymbol{r})$, where each element is a function the proposal centroids, such as relative variance

$$\boldsymbol{g}_j^{(i)} = q \log(\hat{\theta}_j^{(i)}) \qquad\qquad \boldsymbol{r}_j^{(i)} = r \log(\hat{\theta}_j^{(i)}) \tag{7}$$

in this case, the importance sampling $q()$ can not be neglected.

4 Experiments

We show the behavior of DEnFi on two artificial problems. First one is a Bayesian optimization task of minimization of function $f(x) = \frac{\cos(4.3x)}{|x|+1} + \frac{x}{20}$ on bounded interval $(-4, 4)$. The other one is the active classification task on $(0, 1) \times (0, 1)$ square with two classes separated by the diagonal with added noise on squares at both ends of the diagonal. More detailed initial configuration of both problems is shown in Figure 1.

For the first problem we performed 100 independent runs each one consisting of 50 iterations of Bayesian optimization, i.e 50 requests of the function value. The specifics of the setup are given in Section B. Statistical comparison of the proposed method with SGLD [14] and deep ensembles (DE) [12] in terms of absolute distance from the true minimum is provided in Figure 2. The proposed method provides most reliable discovery of the minima. In Section B we provide execution time comparison.

For the second problem we performed 100 independent runs of active classification each one consisting of 200 requests. Since DE was originally designed for optimization purposes, we can compare only with the SGLD method. The estimated classifiers were compared in terms of AUC of the resulting classifications as a function of the number of requests in Figure 3. Note much lower

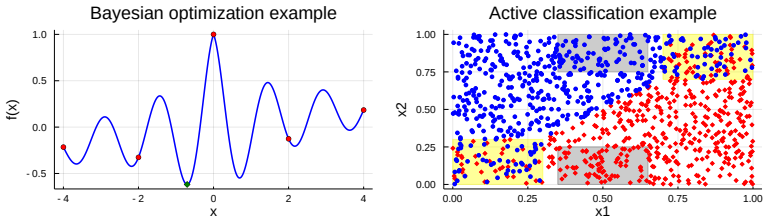

Figure 1: Experimental data. **Left**: Bayesian optimization. Red circles denote D_0, solid blue line denotes set U_0, .i.e. the ground truth function with minimum denoted by the green diamond. **Right**: Active classification. The grayed areas contain data from the initial set D_0 with known labels. Data points outside of the grayed areas belong to U_0 set and the labels have to be requested. The yellowed areas contain data with noise.

random noise of the proposed DEnFi method. This is due to the use of non-inflated ensemble for prediction. The contours of the predicted classes are given in Section B.

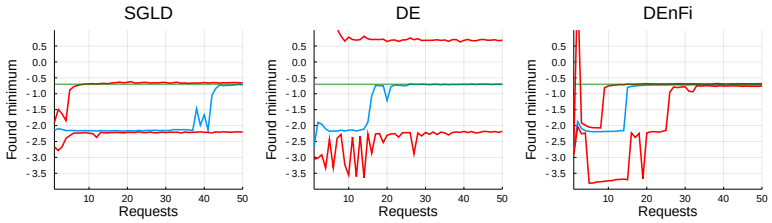

Figure 2: Comparison of DEnFi with SGLD and DE for Bayesian Optimization. The green line shows the true minimum, the blue line denotes the median across 100 runs and the red lines are 10% and 90% quantiles.

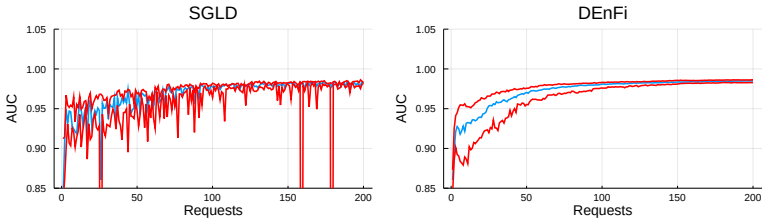

Figure 3: Comparison of DEnFi with SGLD for Active Classification. The blue line denotes the median across 100 runs and the red lines are 10% and 90% quantiles.

References

[1] William H Beluch, Tim Genewein, Andreas Nürnberger, and Jan M Köhler. The power of ensembles for active learning in image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9368–9377, 2018. 1

[2] David A Cohn, Zoubin Ghahramani, and Michael I Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145, 1996. 2

[3] JEAN-MARIE CORNUET, JEAN-MICHEL MARIN, Antonietta Mira, and Christian P Robert. Adaptive multiple importance sampling. *Scandinavian Journal of Statistics*, 39(4):798–812, 2012. 3

[4] Arnaud Doucet, Nando de Freitas, and Neil Gordon. *Sequential Monte Carlo Methods in Practice*. Springer, 2013. 3

[5] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019. 1

[6] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016. 1

[7] Nicholas Kantas, Arnaud Doucet, Sumeetpal Sindhu Singh, and Jan Marian Maciejowski. An overview of sequential monte carlo methods for parameter estimation in general state-space models. *IFAC Proceedings Volumes*, 42(10):774–785, 2009. 3

[8] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413, 2017. 1, 3

[9] Stephan Mandt, Matthew D Hoffman, and David M Blei. Stochastic gradient descent as approximate bayesian inference. *The Journal of Machine Learning Research*, 18(1):4873–4907, 2017. 3

[10] Radford M Neal. *Probabilistic inference using Markov chain Monte Carlo methods*. Department of Computer Science, University of Toronto Toronto, ON, Canada, 1993. 1

[11] Jasper Snoek, Yaniv Ovadia, Emily Fertig, Balaji Lakshminarayanan, Sebastian Nowozin, D Sculley, Joshua Dillon, Jie Ren, and Zachary Nado. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, pages 13969–13980, 2019. 1

[12] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*, pages 2171–2180, 2015. 4

[13] Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust bayesian neural networks. In *Advances in Neural Information Processing Systems*, pages 4134–4142, 2016. 2

[14] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011. 1, 2, 3, 4

A Discussion

It may seem that optimization with different initialization is redundant and it will ultimately converge to the same result. However, after inflation, the resulting ensemble members represent various local minima in the solution. This is illustrated in Figure 4 left. This is possible if the network has excessive capacity to model the data well and the redundant parameters are used to explore the space between the known data points.

To demonstrate the importance of the inflation step we again performed 100 independent runs of DEnFi with covariance matrix $\Lambda = \lambda \boldsymbol{I}$, varying the value of λ. The results of the Bayesian optimization after 50 requests are displayed in Figure 4 right. Note that without inflation step. i.e. $\lambda = 0$, the optimization is stuck at local optimum without exploring the space. With increasing value of λ DEnFi finds the correct minimum with higher probability. For $\lambda > 0.3$ the performance slightly deteriorates but never reaching that without inflation.

The choice of the number of iterations I_1 and I_2 is also important. The number of iterations I_2 for training the inflation ensemble is relatively small, because even after the prediction step the loss decreases relatively fast. On the other hand, the original ensemble requires more training iterations $I_1 > I_2$.

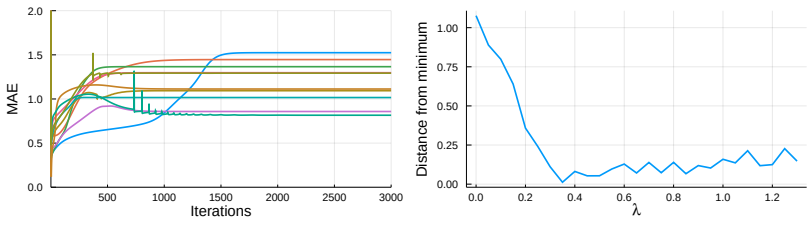

Figure 4: Analysis of Bayesian optimization. **Left**: convergence of mean absolute distance to the true function independently for 10 ensemble members after inflation. **Right**: sensitivity of the distance to minimum after 50 requests as a function of the hyper-parameter λ of the inflation covariance matrix.

Full description of DEnFi is provided in Algorithm 1

Algorithm 1 Update algorithm of Deep Ensemble Filter (DEnFi)

- Initialize with: input data $D_0 = \{x_i, y_i\}_{i=1}^{n_0}$, number of calls N_C, covariance matrix Λ, structure and initial parameters of the ensemble members, number of training iterations for both steps I_1 and I_2, iteration index $k = 0$.
- While $k < K$
  1. Train each member of the ensemble independently on D_k for I_1 epochs, yielding $\theta_k^{(i)}$.
  2. Evaluate predictive probability using θ_k.
  3. Perform inflation of the ensemble (**??**).
  4. Train each member of the inflated ensemble independently on D_k for I_2 epochs.
  5. Find $x_{k+1}^* = \arg\max_{x \in U_k} a_k(x)$ and request the value of y_{k+1}^*.
  6. Expand training set: $D_{k+1} = D_k \cup \{x_{k+1}^*, y_{k+1}^*\}$.
  7. Increase k.

B Experiment details

B.1 Setup for the Bayesian optimization task

We started with five initial points at $x = \{-4, -2, 0, 2, 4\}$. Both ensembles consisted of 20 members. We used NNs with three layers with 16, 8 and 2 neurons respectively connected with swish functions. Same structure was used for SGLD and DE. The inflation parameter λ was set as 0.2. ADAM was used as the optimization algorithm to minimize the MSE loss function. Training epochs were set as $I_2 = 100$, $I_1 = 200$.

B.2 Execution time comparison

The nature of DEnFi allows for very simple implementation of parallel training. The same, however, cannot be said about SGLD nor DE. To alleviate this disadvantage, we ran all 100 runs, each consisting of 50 requests, in parallel instead. The execution time of DEnFi was aproximately 65 minutes, of SGLD 119 minutes and of DE 148 minutes.

B.3 Setup for the Active Classification task

We provided the initial labels for data in squares $(0.35, 0.65) \times (0, 0.3)$ and $(0.35, 0.65) \times (0.7, 1)$. Both ensembles consisted of 20 members. The ensemble

NNs for DEnFi and SGLD had four layers with 16, 10, 6 and 2 neurons respectively connected with swish functions. The inflation parameter λ was set as 0.2. As an optimization algorithm we used ADAM with the crossentropy loss function. Training epochs were set as $I_2 = 250$, $I_1 = 500$.

B.4 Classifier after 200 requests

The results of the predicted classes of each point in the considered domain after 50–200 requests are displayed in Figure 5. Note that DEnFi provides better estimates of class uncertainty in the areas of mixed classes (bottom-left and top-right) than SGLD.

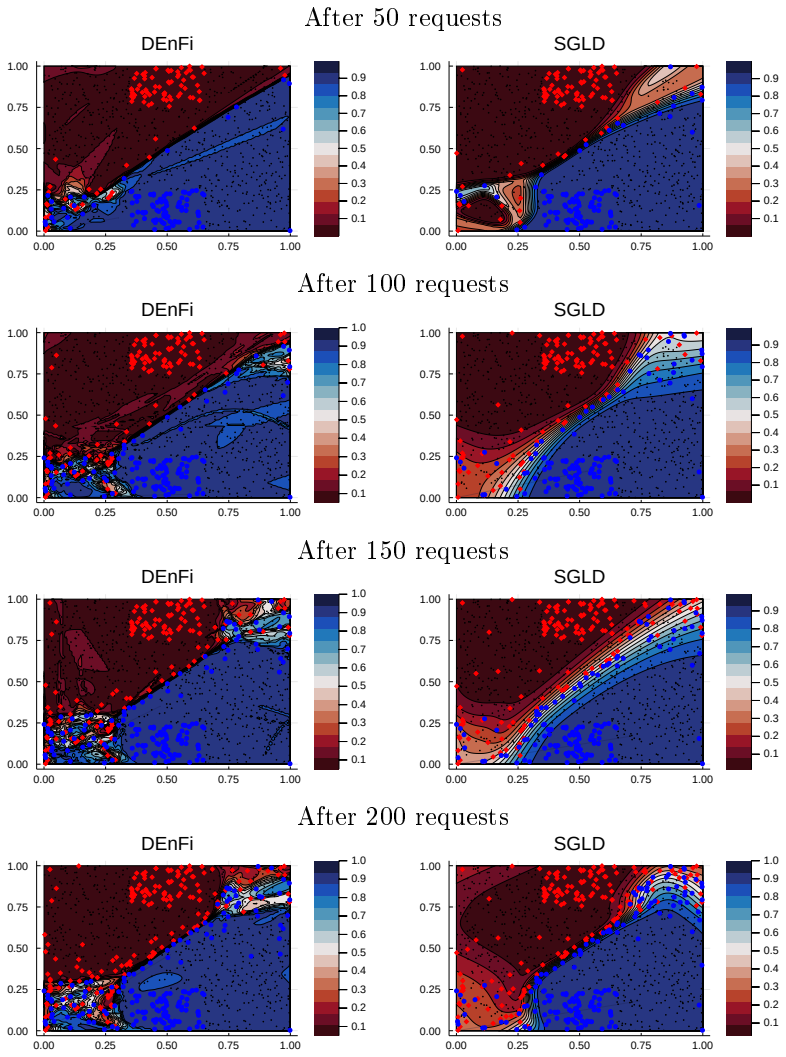

Figure 5: Comparison of DEnFi with SGLD for Active classification in terms of predictive probability of the class after 50, 100, 150, 200 requests from top to bottom respectively.