Eliminating Uncertainity in Deep Learning using Bayesian Approach

Shibani P Hegde[1] and Shreya V Shetty[2]

Abstract— **Deep Learning has procured remarkable advances in various domains .Nonetheless,there is an acute requirement for higher level hypothesizing which has resulted in the development of Bayesian probabilistic model- a model that provides uncertainty estimates which indicates the confidence of a given model in its predicted probabilistic distribution.Bayesian Deep learning is a resurgent field that couples deep learning and Bayesian Inference in order to achieve higher order of intelligence and insight for tasks that involves inference and perception. In this paper, we study how it is possible to implement Bayesian Inference in deep learning models by approximating variational inference.**

I. INTRODUCTION

Deep learning (DL) has successfully achieved greater accuracy in various applications by providing the most optimum values for different parameters of any given model.However, these algorithms have not proven to be very successful in providing 'confidence' estimates which is very critical in understanding what a model knows and does not know. In this paper we will explore the various application of the field known as Bayesian deep learning - which provides uncertainty estimates using deep learning framework. Bayesian Deep Learning aids hypothesis testing by providing uncertainty estimates using the posterior distribution. However, this does not provide us the freedom to scale to large models as established methods like the Markov Chain Monte Carlo method converge slowly and require complex hardware . In contrast, variational inference methods provide scalability by using stochastic-gradient methods owing to their simplicity. However, these Variational Inference methods require more computation, memory, and implementation effort as compared to maximum-likelihood estimation. In Section II, we will discuss the use of natural-gradient methods in order to address the issues faced with Variational methods.

The existing SegNet Architecture, a Deep Convolution Encoder-Decoder, performs pixelwise semantic segmentation. However, we need to know the confidence with which we can trust semantic segmentation's output. In Section III, we will discuss about extending the existing SegNet Architecture by adding probabilistic variations to it Bayesian SegNet Architecture. This architecture performs probabilistic pixelwise semantic segmentation and performs well on Scene Understanding Datasets like CamVid Dataset, SUN Dataset and Pascal VOC Dataset compared to other existing architectures.

[1]Shibani P Hedge - pursuing B.Tech in CSE at PES University email id : hegde.shibani8@gmail.com

[2]Shreya V Shetty - pursuing B.Tech in CSE at PES University email id : shreyav4@gmail.com

There exists different types of uncertainties that can be modelled in computer vision. The uncertainties that we deal with include - aleatoric, epistemic and types of aleatoric uncertainity including homoscedic and heteroscedic. In Section IV, we discuss a framework that is derived to perform both classification and regression tasks by combining aleatoric uncertainty together with epistemic uncertainty.

A scene's geometry and semantics need be understood simultaneously to perform scene understanding tasks. In Section V, we will discuss about multi-task learning framework with shared representation which provides better prediction accuracy and learning efficiency compared to independent task learning models. Modelling as muti-task learning frameworks prevents expensive computations and enables systems to run in real-time. Multiple loss functions are combined to simultaneously learn multiple objectives using homoscedastic uncertainty.

Classical induction algorithms used to model a decision tree for a given dataset involves learning the tree structure and estimating the leaf node parameters. Nonetheless , these algorithms allow the learning of just one decision tree which may not necessarily explain the given training dataset as desirable.In Section VI , we will discuss the adaptation of Bayesian inference to address the issues mentioned by introducing a prior over decision tree and the leaf node parameter which is followed by defining a likelihood measure and computing the posterior distribution over decision trees and the node parameters using Bayes Rule.

II. Scalable and Fast Bayesian Deep Learning

Bayesian inference is popularly used to provide uncertainty estimates by approximating over the posterior distribution.However,this approach fails to provide scalability in large models such as Bayesian Neural Networks as conventional methods like the Markov Chain Monte Carlo (M.C.M.C) methods require substantial amount of time to train and large memory. While, variational inference (VI) methods,in comparison, can be scaled to large models by using stochastic-gradient (S.G) methods.Nonetheless, these methods continue to require more computation and memory when compared to maximum-likelihood estimation (M.L.E) . This section aims to showcase the application of natural gradient methods for variational inference which are simpler to implement and require less implementation effort in comparison with gradient-based methods by exploiting the information geometry of posterior approximations.

A. Gaussian Mean-Field Variational Inference

Maximum-likelihood estimation (M.L.E) is the approach that continues to be popularly used when estimating θ given D by maximizing the log of likelihood: log p(D$|\theta$).Bayesian deep learning aims to achieve the same by applying Bayes' Rule and generating a probability distribution that be continuously sampled in order to estimate the confidence of the model."

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{\int p(D|\theta)p(\theta)d(\theta)}$$

"However,computation of the normalization constant in the denominator continues to be intractable for D.N.Ns.

Variational inference (VI) mitigates the above problem by approximating p(—D) with an another Gaussian distribution q() which is measurable and whose parameters continue to be updated till they are similar to the intractable posterior model -The distribution q() Known as the Gaussian meaneld variational distribution as shown below

$$p(\theta|D) \approx q(\theta) \approx N(\theta|\mu, \sigma^2)$$

The parameters $\mu$and σ^2 can be estimated by the following:

$$maxL(\mu, \sigma^2) := E_q\left[log\frac{p(\theta)}{q(\theta)}\right] + \sum_{i=1}^{N} E_q log\left(\frac{D_i}{\theta}\right)$$

B. Approximate Natural-Gradient VI

Owing to the Riemannian geometry of q(), Natural-gradient VI methods can be applied to scale it's gradient with the inverse of its Fisher information matrix(F.I.M).Building upon the natural gradient method of Khan and Lin(2017) which greatly reduces the computational complexity of the F.I.M -it estimates natural gradients by using the expectation parameters of the exponential-family distribution. The method of Khan and Lin(2017) performs the update as shown below:

$$\mu_{t+1} = \mu_t + \beta_t \sigma_{t+1}^2 o[\nabla_\mu L_t]$$
$$\sigma_{t+1}^{-2} = \sigma_t^{-2} - 2\beta_t[\nabla_{\sigma^2} L_t]$$

This update is known as the Natural Gradient Variational Inference

C. Variational Online Newton(V.O.N)

In order to ensure that the computational complexity of the N.G.V.I update obtained is comparable with that of M.L.E,it is expressed in terms of the M.L.E objective, hence enabling to directly compute the gradients using back-propagation.The following update can be written as

$$\mu_{t+1} = \mu_t - \beta_t(g(\theta_t) + \lambda\mu_t)/(s_{t+1} + \lambda)$$
$$s_{t+1} = (1 - \beta_t)s_t + \beta_t diag[\nabla_{\theta\theta}^2 f(\theta_t)]$$

The update can be scaled to multiple samples and leverage back propagation to perform the gradient and Hessian computation. Since the scaling vector s contains an estimate of the diagonal of the Hessian, this is called the variational online-Newton (V.O.N) method. V.O.N is expected to perform as well as N.G.V.I, with elimination of the computation of the variational objective.

D. Variational Online Gauss-Newton(V.O.G.N)

Due to the computation of the Hessian in V.O.N update,it becomes necessary to avoid negative variances to ensure the method will not break down.Therefore, Generalized Gauss-Newton(G.G.N)approximation is applied to achieve the same.

$$s_{t+1} = (1 - \beta_t)s_t + \beta_t h(\theta_t)]$$

This is known as the variational online Gauss-Newton (V.O.G.N) algorithm. This update will have performance similar to V.O.N update, however, it cannot be easily implemented to the existing deep-learning models

E. Variational RMSprop (Vprop)

Calculation of Hessian update is difficult.Hence,this update approximates the Hessian by gradient Magnitude.Using the GM approximation for the calculation of the Hessian and an additional modication of computing the square root over the scaling factor in the V.O.N update, the V.O.N update is made very similar to RMSprop.

$$\mu_{t+1} = \mu_t - \alpha_t(g(\theta_t) + \lambda\mu_t)/(s_{t+1}^{1/2} + \lambda)$$
$$s_{t+1} = (1 - \beta_t)s_t + \beta_t[g(\theta_t)og(\theta_t)]$$

The Vprop update resembles RMSprop but with certain differences.Therefore, VI can be performed simply by using an RMSprop update with a few simple changes.

F. VADAM (Variational ADAM)

This paper proposes a natural-momentum method which will enable an Adam-like update.Vadam obtains convergence to variance during training time and the variance computed is found to be similar to that obtained from Adam. With the uncertainty estimates computed, this update can be easily applied to any problem statement with similar distribution. Replacing gradients with natural-gradients is a faster and more robust approach and prevents over fitting. The results of the experiments assert that the new Variational Inference algorithms explained can obtain uncertainty estimates which are comparable to the conventional Variational Inference methods with much less implementation and computational effort and with the simplicity of the algorithms for M.L.E.

III. BAYESIAN SEGNET

Pixel wise semantic segmentation in Computer Vision implies partitioning of image into semantically meaningful coherent pixels. Previous works on segmentation involved Non Deep Learning Approaches using Texton-Boost,TextonForest,Random Forest Based Classifiers and Deep Learning Approaches include core Segmentation Engine likes SegNet,Fully Convolutional Networks,Dilation Network. The existing SegNet Architecture consists of a series of encoders. These encoders consists of a set of convolutional layers, which make use of batch normalization techniques and ReLU non-linearity and max-pool layer. This is followed by a set of decoders with a pixel-wise classifier. The SegNet Architectures differs from other architectures as they upsample the sparse encoding matrix obtained during pooling by using max-pooling indices. This helps in reducing model parameterization and enables retaining the class boundary details of the segmented images. However, the correctness of the semantic segmentation's output is not know. Providing a Bayesian approach to the existing SegNet architecture resolves the problem of modelling uncertainty. The Bayesian SegNet architecture makes use of dropouts which is used as an approximate inference. Finding the posterior distribution over weights, given the input values and labels,

$$p(W|X, Y)$$

is an integral part of this model. This posterior distribution is hard to trace, hence it is necessary to approximate over the distribution of weight, q(W),which can be done by using of variational inference. The distribution q(W) is learnt by minimizing the KL divergence term between this approximating distribution and the full posterior

$$KL(q(W)\|p(W|X, Y))$$

It was observed that the KL divergence term could be minimized by minimizing the cross entropy loss function. Hence the network was trained with stochastic gradient descent. Dropout is used on the model while training it. They are also used during testing since they aid in sampling the posterior distribution over the weights which can further be used to obtain the posterior distribution of softmax class probabilities. The mean of these samples is taken to produce the segmentation output and the uncertainity of each class is given by calculating the variance. The model is trained end to end and a learning rate equal to 0.001 and a weight decay equal to 0.0005 is used. The model is trained until no further convergence is observed.

A. Proabilistic Variants to the Architecture

The encoder consists of a set of convolutional layers followed by a max pool layer and the decoder contains a set of convolutional layers followed by an upsampling layer. On training a Bayesian network with dropouts after every convolutional layer resulted in strong regularization causing the network to learn slowly. Hence some variants to the model are made, which are as follows:

- Dropout is inserted after each Bayesian Encoder unit.
- Dropout is inserted after each Bayesian Decoder unit.
- Dropout is inserted after each Bayesian Encoder and Decoder unit.
- Dropout is inserted after the deepest Bayesian Encoder
- Dropout is inserted after the central four encoder and decoder units.
- Dropout is inserted after the last decoder unit, before the classifier.

Dropping out the half of the encoder and decoder units was obseerved to be a good configuration. Hence, dropout was added to the central six encoders and decoders for the full 26 layer Bayesian SegNet. The lower layers of the convolutional neural network showed better performance with deterministic weights whereas the deeper layers could model the features well with the bayesian weights.

B. Weight Averaging vs Monte Carlo Dropout Sampling

Using dropouts in neural network is equivalent to a form of approximate Bayesian inference. Performing these dropouts at test time as a way to sample from the posterior distribution is called Monte Carlo dropout. Weight averaging technique aims to eliminate dropouts at test time, instead it scales the weights proportionally to the dropout percentage. Monte Carlo Dropout Sampling performed better than weight averaging as weight averaging produced poor segmentation resulting in reduced global accuracy and there is no measure of model uncertainty.

C. Experiments

Bayesian SegNet performs well on scene segmentation datasets. Here are the results on various datasets :

- CamVid Dataset : This is a road scene understanding dataset. It contains 367 training images and 233 testing images of day and dusk scenes. There are 11 classes to be classified. All the images are resized to 360x480 pixels. Bayesian SegNet performs better since it has a reasonably high mean IoU score and very high overall class average obtains the highest overall class average. It also sets a new benchmark on 7 out of the 11 classes.
- Scene Understanding(SUN) : This dataset consists is indoor scenes dataset. It contains 5285 training and 5050 testing images. The images are of different resolutions and object have frequent partial occlusions hence the dataset is challenging. The images are resized 224x224 pixels. Bayesian SegNet performs well compared to all previous benchmarks with an IoU of 30.7%.
- Pascal VOC : This dataset consists of 12031 training images and 1456 testing images. It consists of 20 salient object classes. Images are resized to 224x224 pixels. The IoU so obtained is 60.5% which is better than non-bayesian approach.

The observation made indicates that the model uncertainty is generally very high when the model predicts an incorrect label. This is mostly due to ambiguity surrounding the definition of defining where these labels transition, visually difficult to identify objects that are occluded and visually ambiguous to the model.

IV. Uncertainties associated with Bayesian Deep Learning for Computer Vision

Deep Learning is used in Computer Vision to achieve state of art results. These methods do not represent uncertainty in regression settings and classification settings. To capture uncertainty Bayesian approach can be used. There are two main types of uncertainty, Aleatoric and Epistemic. Aleoatoric uncertainty captures noise inherent in the observation. This noise can arise due to sensor noise or motion noise. This uncertainty cannot be reduced even with more data. It is modeled by placing a distribution over the output of the model. Aleatoric uncertainty is further classified into homoscedastic uncertainty and heteroscedastic uncertainty. Homoscedastic uncertainty remains the same with different inputs whereas Heteroscedastic uncertainty varies with inputs to the model. To capture aleatoric uncertainty in regression, we would have to tune the observation noise parameter σ.

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2\sigma(x_i)^2}||y_i - f(x_i)||^2 + \frac{1}{2}log\sigma(x_i)^2$$

In the above equation, variational inference is not performed over the weights but it is instead performed over Maximum a posteriori inference. This approach does not capture epistemic model uncertainty, as epistemic uncertainty is not a property of the data but of the data.

Epistemic uncertainty is caused because of uncertainty involved in model parameters and this uncertainty can be reduced with enough data. It is modeled by placing prior distribution over a models weights. Bayesian inference is used to compute the posterior over the weights $p(W|X,Y)$. Denoting the random output of the BNN as $f^w(x)$ the model likelihood is defined as $p(y|f^w(x))$. For regression, likelihood is defined as a Gaussian with mean given by the models output and a noise scalar σ :

$$p(y|f^w(x)) = N(f^w(x), \sigma^2)$$

For classification, the models output is squashed using a softmax function. The resulting probability vector is :

$$p(y|f^w(x)) = Softmax(f^w(x))$$

However, its difficult to evaluate the posterior. Hence it is approximated. MC Dropout is used since it as like an approximate variational inference. The minimisation objective is given by :

$$L(\theta, p) = \frac{-1}{N}\sum_{i=1}^{N} logp(y_i|f^{w_i}(x_i)) + \frac{1-p}{2N}||\theta||^2$$

Here N is the number of data points, p is the dropout probability, w_i-approximated weights and θ are parameters to be optimized. The uncertainity associated with prediction in classification can be stated as :

$$p(y=c|x,X,Y) \approx \frac{1}{T}\sum_{t=1}^{T} Softmax(f^{w_t}(x_t))$$

Here T is the number of samples, w_t is the approximated weights. For regression settings, epistemic uncertainty is captured by the predictive variance and this is given by :

$$Var(y) \approx \sigma^2 + \frac{1}{T}\sum_{t=1}^{T} f^{w_t}(x_t)E(y)^T E(y)$$

” The predictions in this epistemic model can be done by approximating the predictive mean:

$$E(y) \approx \frac{1}{T}\sum_{t=1}^{T} f^{w_t}(x)$$

σ^2 in the first term indicates the noise inherent in the data. The second term indicates the predictive variance which quantifies how much the model is uncertain about its predication.

A. Combining Aleatoric Uncertainty and Epistemic Uncertainity

In Computer Vision, it is most effective to model aleatoric uncertainity in comparison to epistemic uncertainty as more data cannot reduce this uncertainity. Aleatoric uncertainty is costly to compute since absence of data which can be realised due to epistemic uncertainty, cannot be identified with aleatoric uncertainty alone. Existing approaches to Bayesian deep learning captures either epistemic uncertainty alone or aleatoric uncertainty alone. Hence, a unified Bayesian deep learning framework which learns aleatoric uncertainty composed together with epistemic uncertainty approximations. As before model weights are drawn from the approximate posterior to obtain model output which is composed of predictive mean,$\hat{y}$, and predictive variance,$\hat{\sigma}^2$. Given its a Gaussian likelihood , the minimisation objective is as follows :

$$L(\theta) = \frac{1}{D}\sum_{i}\frac{1}{2}\hat{\sigma_i}^{-2}||y_i - \hat{y_i}||^2 + \frac{1}{2}log\hat{\sigma_i}^2$$

Here, D is the number of output pixels y_i corresponding to input image x, indexed by i and $\hat{\sigma_i}^2$ is the prediction variance. In order to make it numerically stable, $\hat{\sigma_i}^2$ is replaced by $\hat{s_i}$ as the loss avoids a potential division by zero.

$$L(\theta) = \frac{1}{D}\sum_{i}\frac{1}{2}exp(-s_i)||y_i - \hat{y_i}||^2 + \frac{1}{2}s_i$$

This represents an intelligent regression sysytem as the networks predicts uncertainty by controlling $exp(s_i)$. The network tries to adapt the residuals weighting, also the network to learn to attenuate the effect from vague labels. Inputs have lesser effect on the loss function, if the model predicts high uncertainity for that input.

B. Experiments

The experiments were performed for both semantic segmentation and depth regression. For semantic segmentation datasets like CamVid and NYU v2 were used. On CamVid dataset the mean IoU score of 67.5% was observed. Thus modelling both aleatoric and epistemic uncertainty improved over the baseline result. NYUv2 is a challenging indoor segmentation. Bayesian architecture was used as baseline model. The baseline performace was improved to 37.3%. For pixel wise depth regression Make3D and NYUv2 Depth Datasets were used. Aleatoric uncertainty captured those aspects of the task that are very difficult, example higher uncertainity for large depths, reflective surfaces and occlusion boundaries. Epistemic uncertainty captured difficulties due to lack of data. Thus, modelling both aleatoric and epistemic uncertainty improves the model performance.

V. Modelling Uncertainty for Multi-Task Learning

A given scene understanding problem is a multi-task learning problem as it is necessary to understand both the semantics and the geometry of the scene at the same time. Previous approaches to multi-task learning had weights that were uniform or manually tuned. Finding the optimal weights through manual tuning is hard and the performance of the system is highly dependent on the right weights given for each task's loss.

A principled approach of combining multiple loss functions simultaneously to learn multiple objectives,both classification and regression, using homoscedastic uncertainty is proposed. Homoscedastic uncertainty is interpreted as task-dependent weighting. Multi-task learning involves 3 tasks - semantic segmentation, instance segmentation and pixel-wise metric depth. Combining all these multi-tasks into a single model allows the system to run in real time and reduces computations. It also ensures that the model agrees between the separate task outputs. Using a shared representation with multi-task learning improves performance on various metrics, making the models more effective. The naive approach to combining multi objective losses would be to simply perform a weighed linear sum of the losses for each individual tasks: $L_{total} = \sum_i w_i L_i$. However, the issues with this is that the model performance is extremely sensitive to weight selection and tuning these weight hyper-parameters is expensive. Probailistic modelling provides a better approach to learn optimal weights.

A. *Multi-task Likelihood*

Multi-task loss function is derived based on maximising the Gaussian likelihood with homoscedastic uncertainty. With $f^w(x)$ as the neural networks output, x being input and w being the weights,$y_1 ... y_k$ being the model outputs(such as semantic segmentation, depth regression, etc), the multi-task likelihood can be defined as follows :

$$p(y_1, y_2, ...y_k|f^w(x)) = p(y_1|f^w(x))...p(y_k|f^w(x))$$

In maximum likelihood inference, the log likelihood of the model is maximized. The equation for regression is :

$$logp(y|f^w(x))\alpha - \frac{1}{2\sigma^2}||y - f^w(x)||^2 - log\sigma$$

Let us now assume that the models output is composed of two vectors y_1 and y_2 , each following a Gaussian distribution:

$$p(y_1, y_2|f^w(x)) = p(y_1|f^w(x)).p(y_2|f^w(x))$$

"

$$= N(y_1; f^w(x), {\sigma_1}^2).N(y_2; f^w(x), {\sigma_2}^2)$$

This leads to the minimisation objective, $L(W, \sigma_1, \sigma_2)$:

$$= logp(y_1, y_2|f^w(x))$$

$$\alpha\frac{1}{2\sigma_1^2}||y_1 f^w(x)||^2 + \frac{1}{2\sigma_2^2}||y_2 f^w(x)||^2 + log\sigma_1\sigma_2$$

$$= \frac{1}{2\sigma_1^2}L_1(W) + \frac{1}{2\sigma_2^2}L_2(W) + log\sigma_1\sigma_2$$

$L_1(W)$ and $L_2(W)$ indicate the losses for y_1 and y_2 respectively. With the increase in the noise parameter σ the weight associated with the corresponding loss function decreases and with the decrease in the noise parameter σ the weight associated with the corresponding loss function increases. The last term in the equation prevents the network from ignoring the data with the increase in noise. For classification, the scaled version of likelihood is as follows (Boltz-mann distribution) :

$$p(y|f^w(x), \sigma) = Softmax(\frac{1}{\sigma^2}f^w(x))$$

The log-likelihood for this output can then be written as:

$$logp(y = c|f^w(x), \sigma) = \frac{1}{\sigma^2}{f_{\hat{c}}}^w(x) - log\sum_{\hat{c}} exp(\frac{1}{\sigma^2}{f_{\hat{c}}}^w(x))$$

with ${f_{\hat{c}}}^w(x)$ the c element of the vector $f_w(x)$. Joint loss,$L(W, \sigma_1, \sigma_2)$, for two discrete outputs y_1 and y_2 can be stated as :

$$= -logp(y_1, y_2 = c|f^w(x))$$

$$= -logN(y_1; f^w(x), {\sigma_1}^2).Softmax(y_2 = c; f^w(x), {\sigma_1}^2)$$

$$\approx \frac{1}{2\sigma_1^2}L_1(W) + \frac{1}{\sigma_2^2}L_2(W) + log\sigma_1 + log\sigma_2$$

This loss so obtained is smoothly differentiable and it will not converge to zero unlike the naive approach of lianear summation over losses. The model is trained to predict $log\sigma^2$ A SegNet Architecture is used and at the encoders a shared respresentation is produced. Dialated convolutions are appliesd at the encoder to sub-sample the feature map by a factor of 8. Each decoder consists of a 3*3 convolutional layer with output feature size 256, followed by a 1*1 layer regressing the tasks output. There are 3 decoders, each to produce semantic segmentations output,instance segmentations output and depth regressions output. The experiments performed showed improvement in performance when training with multi-task losses, over both single-task models and weighted losses. This shows that our loss function can automatically learn a better performing weighting between the tasks than the baselines.

VI. Decision Trees with probabilistic perspective

Classical decision tree learning algorithms used to train a decision tree for a given dataset involves learning the tree structure and estimating the leaf node parameters.However, these algorithms allow the learning of just one decision tree while there may exist many decision trees that can explain the given model .The classical tree induction procedure is used to train a decision tree with its leaf node parameters until there is an unique data point at each leaf node but the learning is stopped by specifying a hyper-parameter that decides the threshold beyond which the nodes are not split further. However,this can lead to over- tting of the training data, resulting in overcondent estimates of any unseen data. Moreover, the leaf node parameters are computed using only the data points available at the given leaf node which amounts to poor generalization.

A. Bayesian Approach

The Bayesian inference aims to mitigate the issues mentioned by introducing a prior over decision tree and the leaf node parameters , by dening a likelihood measure that indicates the 'goodness' of the learnt decision tree and ,finally,computing the posterior distribution of the decision trees and the node parameters using Bayes' Rule as shown below.

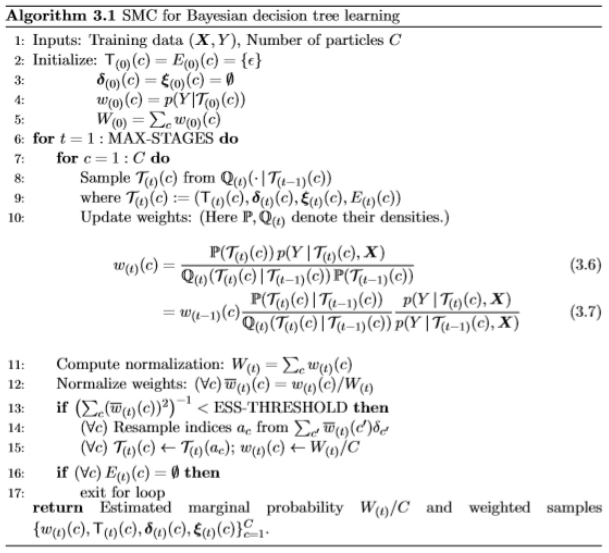
$$p(T,\theta|Y,X) = \frac{1}{Z(Y,X)} p(Y|T,\theta,X)p(\theta|T)p(T|X)$$

$$Z(Y,X) = \sum \int_{\theta} p(Y|T,\theta,X)p(\theta|T)p(T|X)d\theta$$

The potential drawback in the Bayesian approach is the difficulty faced in the computation of the summation term over T as there are exponentially many trees. Alternatively, the posterior is computed by approximating over a subset of trees with more weightage provided to informative trees.

B. SMC for Bayesian decision trees

This section introduces the application of Sequential Monte Carlo (S.M.C) to Bayesian Decision Trees for estimating variance in Bayesian decision trees.Moreover,This method prevents over-tting owing to the resampling and the combination of the prior term with the likelihood to limit the growth of the trees,



Algorithm 3.1 SMC for Bayesian decision tree learning

```
1: Inputs: Training data $(\boldsymbol{X}, Y)$, Number of particles $C$
2: Initialize: $\mathsf{T}_{(0)}(c) = E_{(0)}(c) = \{\epsilon\}$
3:             $\boldsymbol{\delta}_{(0)}(c) = \boldsymbol{\xi}_{(0)}(c) = \emptyset$
4:             $w_{(0)}(c) = p(Y|\mathcal{T}_{(0)}(c))$
5:             $W_{(0)} = \sum_c w_{(0)}(c)$
6: for $t = 1$ : MAX-STAGES do
7:     for $c = 1 : C$ do
8:         Sample $\mathcal{T}_{(t)}(c)$ from $\mathbb{Q}_{(t)}(\cdot \mid \mathcal{T}_{(t-1)}(c))$
9:         where $\mathcal{T}_{(t)}(c) := (\mathsf{T}_{(t)}(c), \boldsymbol{\delta}_{(t)}(c), \boldsymbol{\xi}_{(t)}(c), E_{(t)}(c))$
10:        Update weights: (Here $\mathbb{P}, \mathbb{Q}_{(t)}$ denote their densities.)

           $w_{(t)}(c) = \frac{\mathbb{P}(\mathcal{T}_{(t)}(c))\, p(Y \mid \mathcal{T}_{(t)}(c), \boldsymbol{X})}{\mathbb{Q}_{(t)}(\mathcal{T}_{(t)}(c) \mid \mathcal{T}_{(t-1)}(c))\, \mathbb{P}(\mathcal{T}_{(t-1)}(c))}$   (3.6)
           $= w_{(t-1)}(c) \frac{\mathbb{P}(\mathcal{T}_{(t)}(c) \mid \mathcal{T}_{(t-1)}(c))}{\mathbb{Q}_{(t)}(\mathcal{T}_{(t)}(c) \mid \mathcal{T}_{(t-1)}(c))} \frac{p(Y \mid \mathcal{T}_{(t)}(c), \boldsymbol{X})}{p(Y \mid \mathcal{T}_{(t-1)}(c), \boldsymbol{X})}$   (3.7)

11:    Compute normalization: $W_{(t)} = \sum_c w_{(t)}(c)$
12:    Normalize weights: $(\forall c)\, \overline{w}_{(t)}(c) = w_{(t)}(c)/W_{(t)}$
13:    if $\left(\sum_c (\overline{w}_{(t)}(c))^2\right)^{-1}$ < ESS-THRESHOLD then
14:        $(\forall c)$ Resample indices $a_c$ from $\sum_{c'} \overline{w}_{(t)}(c')\delta_{c'}$
15:        $(\forall c)$ $\mathcal{T}_{(t)}(c) \leftarrow \mathcal{T}_{(t)}(a_c)$; $w_{(t)}(c) \leftarrow W_{(t)}/C$
16:    if $(\forall c)\, E_{(t)}(c) = \emptyset$ then
17:        exit for loop
   return Estimated marginal probability $W_{(t)}/C$ and weighted samples
   $\{w_{(t)}(c), \mathsf{T}_{(t)}(c), \boldsymbol{\delta}_{(t)}(c), \boldsymbol{\xi}_{(t)}(c)\}_{c=1}^{C}$.
```

Fig. 1. Plate diagram for the LDA model.

CONCLUSION

The paper discussed in detail the new Variational Inference algorithms which provide an advantage of simplicity and less implementation effort comparable with that of M.L.E .The proposed algorithms can be implemented within existing codebases with minimal changes. We have presented the Bayesian SegNet Architecture which provides a probabilistic pixel wise semantic segmentation framework which outputs a measure of model uncertainty for each class. We presented a bayesian approach to deep learning which models both aleatoric and epistemic uncertainty. We also present a bayesian deep learning framework for multi-task learning with shared representation which correctly weights the loss terms for each task.We have,finally, presented the application of Bayesian inference algorithms to learn decision trees, based on the sequential Monte Carlo framework which are easier to implement than their M.C.M.C counterparts.This literature survey shows , in depth, the emolument of application of Bayes' Rule in various upcoming fields of machine learning.

REFERENCES

[1] What uncertainties do we need in Bayesian deep Learning for Computer Vision Oct 2017 : Alex Kendall,Yarin Gal,2017

[2] Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics:Alex Kendall ,Roberto Cipolla, Yarin Gal, 2018

[3] Fast and Scalable Bayesian Deep Learning by Weight-Perturbation in Adam arXiv August 2018

[4] Decision Trees and Forests A Probabilistic Perspective balaji-phd-thesis 2016

[5] Bayesian SegNet Model Uncertainty in Deep Convolutional Encoder-Decoder architectures for Scene Understanding arXic 2016