Bayesian Deep Learning

Mohammad Emtiyaz Khan
AIP (RIKEN), Tokyo

http://emtiyaz.github.io

emtiyaz.khan@riken.jp

June 06, 2018

What will you learn?

Why is Bayesian inference useful?

1. It allows for computation of uncertainty.
2. It can prevent overfitting in a principled way.

Why is it computationally challenging?

3. It usually requires solving hard integrals.

How does Variational Inference (VI) simplify the computational challenge?

4. Integration is converted to an optimization problem.
5. Optimization gives a lower bound to the integral.

How to perform variational inference?

6. By using stochastic gradient descent.

1 Why be Bayesian?

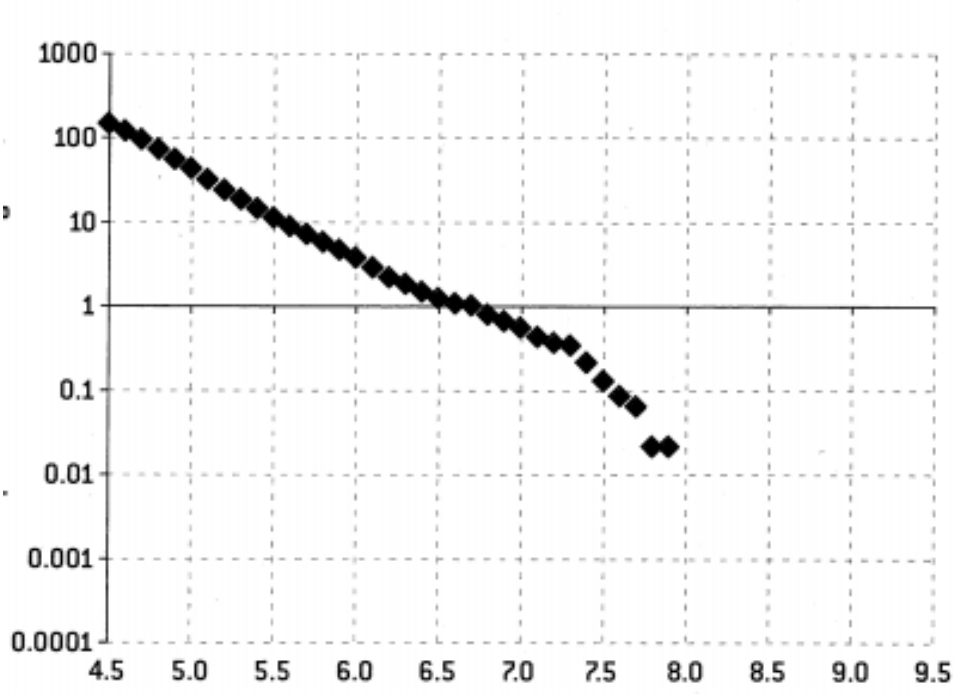
Why be Bayesian

What are the main reasons behind the recent success of Machine-learning and deep-learning methods, e.g., in fields such as computer vision, speech recognition, and recommendation systems?

Existing methods focus on fitting the data well (e.g., using maximum likelihood), but this may be problematic.

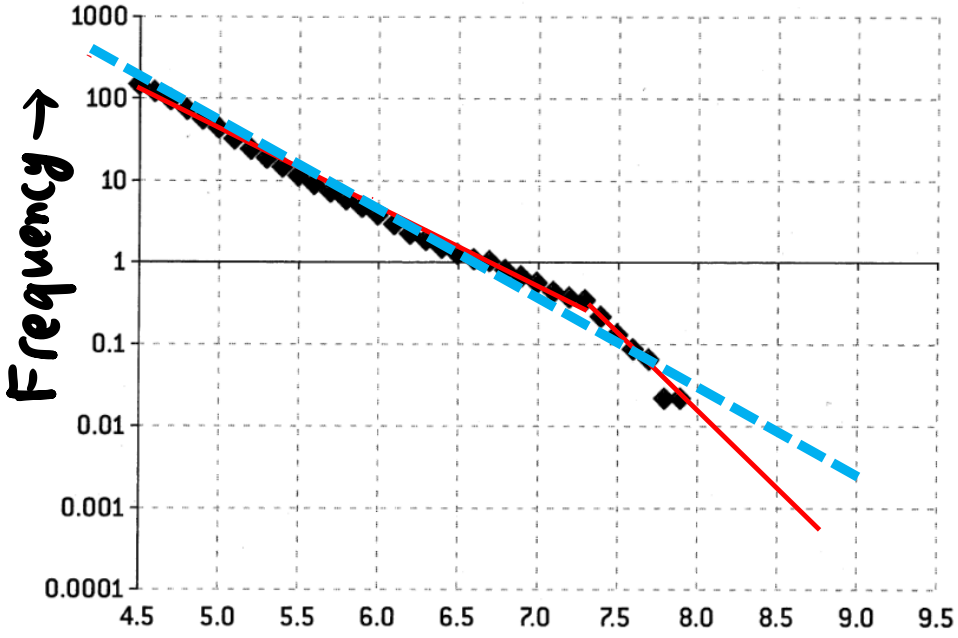
Which is a better fit?

Consider this real data where y-axis is the frequency of an event and x-axis is its magnitude.



This example is taken from Nate Silver's book.

Which model is a better fit? blue or red?

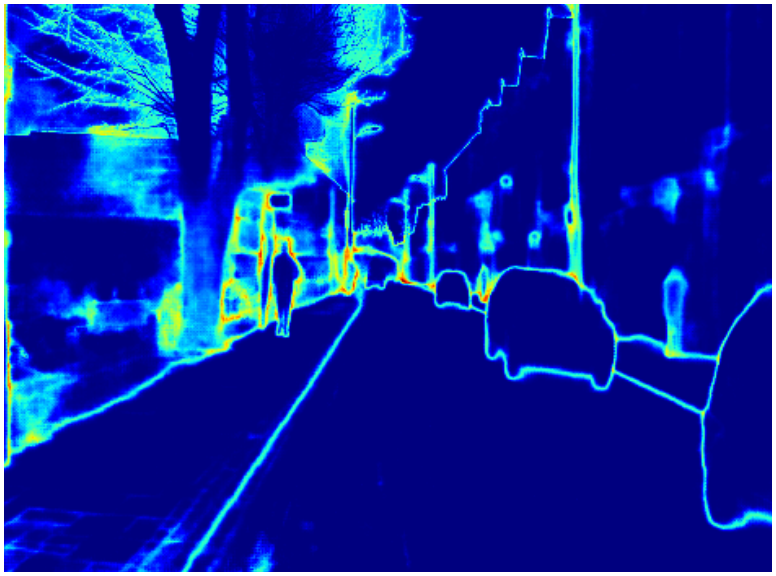


Uncertainty Estimation

In many applications, it is also important to estimate the *uncertainty* of our unknowns and/or predictions. In this lecture, we will learn about a Bayesian way to do so.

Scene

Uncertainty of depth estimates



(taken from Kendall et. al. 2017)

2 Bayesian Model for Regression

Regression

Regression is to relate input variables to the output variable, to either predict outputs for new inputs and/or to understand the effect of the input on the output.

Dataset for regression

In regression, data consists of pairs $(\mathbf{x}_n, y_n)$, where $\mathbf{x}_n$ is a vector of D inputs and y_n is the n'th output. Number of pairs N is the data-size and D is the dimensionality.

Prediction

In prediction, we wish to predict the output for a new input vector, i.e., find a *regression* function that approximates the output "well enough" given inputs.

$$y_n \approx f_\theta(\mathbf{x}_n), \text{ for all } n$$

where $\boldsymbol{\theta}$ is the parameter of the regression model.

Likelihood

Assume y_n to be independent samples from an exponential family distribution, whose mean parameter is equal to $f_\theta(\mathbf{x}_n)$:

$$p(\mathbf{y}|\mathbf{X}) = \prod_{n=1}^{N} p(y_n|f_\theta(\mathbf{x}_n))$$

For example, when p is a Gaussian, the log-likelihood results in mean-square error.

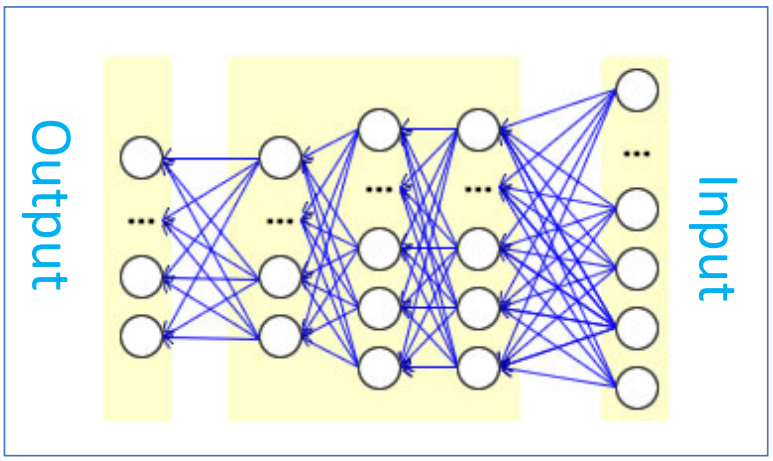
Examples of f_θ

Linear model: $\boldsymbol{\theta}^\top \mathbf{x}_n$

Generalized: $f(\boldsymbol{\theta}^\top \mathbf{x}_n)$

Deep Neural Network:

$$f_1(\boldsymbol{\Theta}_1 f_2(\boldsymbol{\Theta}_2 \dots f_L(\boldsymbol{\Theta}_L \mathbf{x}_n)))$$



Maximum A Posteriori (MAP)

By using a regularizer $\log p(\boldsymbol{\theta})$, we can estimate $\boldsymbol{\theta}$ by maximizing the following loss denoted by $\mathcal{L}_{MAP}(\boldsymbol{\theta}) :=$

$$\sum_{n=1}^{N} \log p(y_n | f_\theta(\mathbf{x}_n)) + \log p(\boldsymbol{\theta})$$

Example: an L_2 regularizer. The $\boldsymbol{\theta}_{MAP}$ obtained by maximizing $\mathcal{L}_{MAP}$ is known as the maximum-a-posteriori estimate.

Stochastic Gradient Descent (SGD)

When N is large, choose a random pair $(\mathbf{x}_n, y_n)$ in the training set and approximate the gradient:

$$\frac{\partial \mathcal{L}_{MAP}}{\partial \boldsymbol{\theta}} \approx \frac{\partial}{\partial \boldsymbol{\theta}} \left[N \log p(y_n | f_\theta(\mathbf{x}_n)) + \log p(\boldsymbol{\theta}) \right]$$

Using the above *stochastic* gradient, take a step:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \rho_t \widehat{\frac{\partial \mathcal{L}_{MAP}}{\partial \boldsymbol{\theta}}}$$

where t is the iteration, and ρ_t is a learning rate.

The Joint Distribution

How can we obtain a distribution over $\boldsymbol{\theta}$? We can make use of the Bayes' rule.

We can view $p(\boldsymbol{\theta})$ as a *prior* distribution to define the following joint distribution:

$$\begin{aligned}
&\log p(\mathbf{y}, \boldsymbol{\theta}|\mathbf{X}) \\
&:= \sum_{n=1}^{N} \log p(y_n|f_\theta(\mathbf{x}_n)) + \log p(\boldsymbol{\theta}) + \text{cnst} \\
&= \log \left[\prod_{n=1}^{N} p(y_n|f_\theta(\mathbf{x}_n)) \right] p(\boldsymbol{\theta})
\end{aligned}$$

For example: L_2 loss corresponds to a Gaussian prior distribution.

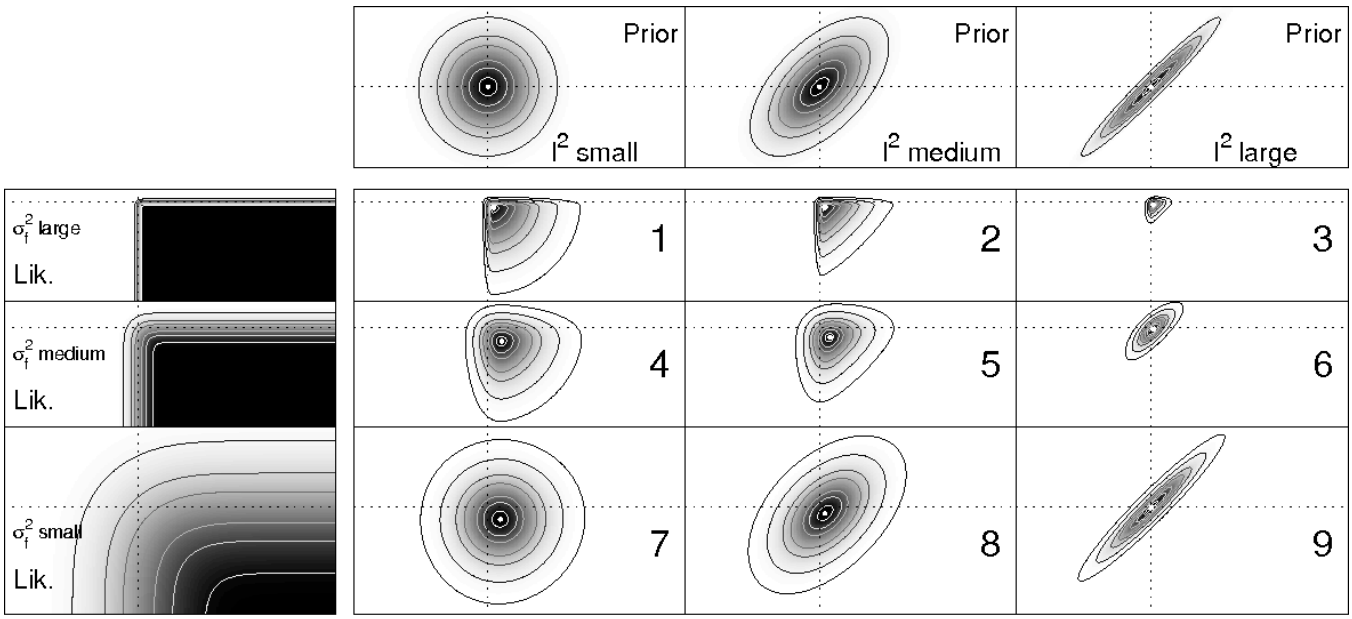
The Posterior Distribution

The posterior distribution is defined using the Bayes' rule:

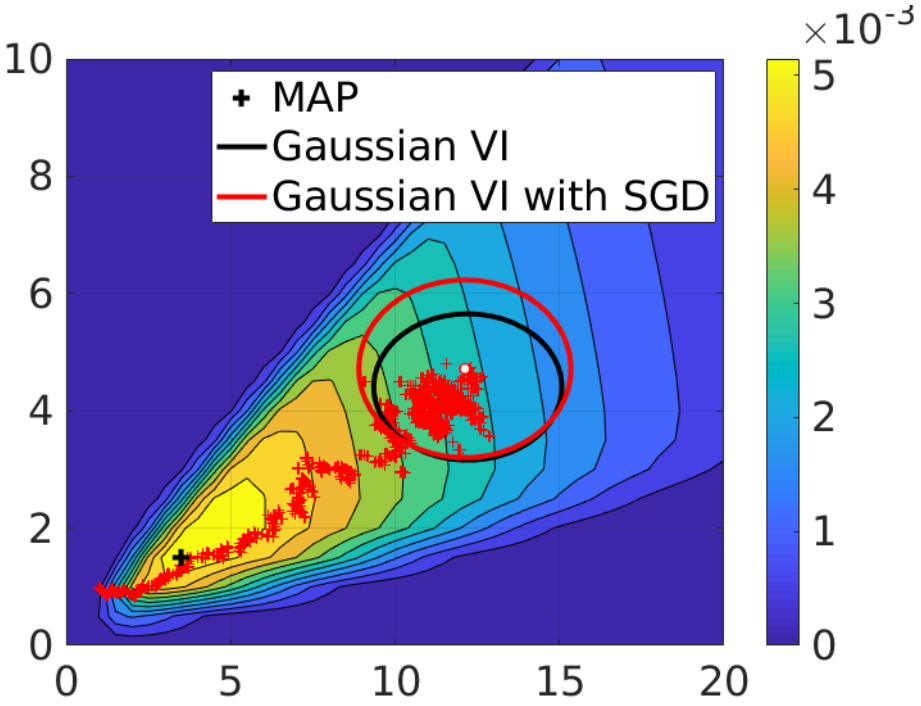
$$p(\boldsymbol{\theta}|\mathbf{y},\mathbf{X}) = \frac{p(\mathbf{y},\boldsymbol{\theta}|\mathbf{X})}{p(\mathbf{y}|\mathbf{X})}$$
$$= \frac{\left[\prod_{n=1}^{N} p(y_n|f_\theta(\mathbf{x}_n))\right] p(\boldsymbol{\theta})}{\int \left[\prod_{n=1}^{N} p(y_n|f_\theta(\mathbf{x}_n))\right] p(\boldsymbol{\theta}) d\boldsymbol{\theta}}$$

The integral over all possible $\boldsymbol{\theta}$ defines the normalizing constant of the posterior distribution.

Without it, we cannot know the true *spread* of the distribution, which gives us a notion of uncertainty or confidence.



Taken from [Nickisch and Rasmussen, 2008]



Difficult Integral

The normalizing constant of the posterior distribution is difficult to compute in general:

$$p(\mathbf{y}|\mathbf{X}) = \int \left[\prod_{n=1}^{N} p(y_n|f_\theta(\mathbf{x}_n))\right] p(\boldsymbol{\theta})d\boldsymbol{\theta}$$

There are (at least) three reasons behind it:

1) Too many factors (large N)
2) Too many dimensions (large D)
3) Nonconjugacy (e.g., non Gaussian likelihood with a Gaussian prior)

3 Variational Inference

Integration to Optimization.

Variational Distribution

Approximate the posterior:

$$p(\boldsymbol{\theta}|\mathbf{y}, \mathbf{X}) \approx q_\lambda(\boldsymbol{\theta})$$

q_λ is called the *variational distribution*, e.g., a Gaussian distribution with $\boldsymbol{\lambda} := \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$

Variational Lower Bound

Given q_λ, we can obtain a lower bound to the integral:

$$p(\mathbf{y}|\mathbf{X}) \geq \mathcal{L}_{VI}(\boldsymbol{\lambda}) :=$$
$$\sum_{i=1}^{N} \mathbb{E}_{q_\lambda}[\log p(y_n|f_\theta(\mathbf{x}_n))] + \mathbb{E}_{q_\lambda}\left[\log \frac{p(\boldsymbol{\theta})}{q_\lambda(\boldsymbol{\theta})}\right],$$

a.k.a. *variational objective*, but it is very similar to the MAP objective:

$$\mathcal{L}_{MAP}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \log p(y_n|f_\theta(\mathbf{x}_n)) + \log p(\boldsymbol{\theta})$$

So we can just use SGD!

Stochastic Gradients I

How do we compute a stochastic gradient of $\mathcal{L}_{VI}$? The *reparameterization trick* is one method to do so [Kingma and Welling, 2013]. Let $q_\lambda := \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2))$, then

$$\boldsymbol{\theta}(\boldsymbol{\lambda}) = \boldsymbol{\mu} + \boldsymbol{\sigma} \circ \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}|0, \mathbf{I})$$

is a sample from q_λ. We have written $\boldsymbol{\theta}$ as a function of $\boldsymbol{\lambda} := \{\boldsymbol{\mu}, \boldsymbol{\sigma}\}$.

Here is a stochastic gradient with one Monte-Carlo sample $\boldsymbol{\theta}(\boldsymbol{\lambda})$:

$$\frac{\partial \mathcal{L}_{VI}(\boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}} \approx \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\lambda}} \frac{\partial \mathcal{L}_{MAP}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} - \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\lambda}} \frac{\partial \log q_\lambda(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} - \frac{\partial \log q_\lambda(\boldsymbol{\theta})}{\partial \boldsymbol{\lambda}}$$

This can be done whenever q_λ is *reparameterizable*.

Stochastic Gradients II

REINFORCE is another approach to computing a stochastic gradient of $\mathcal{L}_{VI}(\boldsymbol{\lambda})$ [Williams, 1992]. It uses the log-derivative trick:

$$\frac{\partial q_\lambda}{\partial \boldsymbol{\lambda}} = q_\lambda \frac{\partial \log q_\lambda}{\partial \boldsymbol{\lambda}}$$

to express the gradient of the expected MAP objective as

$$\frac{\partial}{\partial \boldsymbol{\lambda}} \mathbb{E}_{q_\lambda}[\mathcal{L}_{MAP}(\boldsymbol{\theta})] = \mathbb{E}_{q_\lambda}\left[\frac{\partial \log q_\lambda(\boldsymbol{\theta})}{\partial \boldsymbol{\lambda}} \mathcal{L}_{MAP}(\boldsymbol{\theta})\right]$$

The REINFORCE gradient estimator with one Monte-Carlo sample $\boldsymbol{\theta}(\boldsymbol{\lambda})$ is given by:

$$\frac{\partial \mathcal{L}_{VI}(\boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}} \approx \frac{\partial \log q_\lambda(\boldsymbol{\theta})}{\partial \boldsymbol{\lambda}} \mathcal{L}_{MAP}(\boldsymbol{\theta}) - \frac{\partial \log q_\lambda(\boldsymbol{\theta})}{\partial \boldsymbol{\lambda}} (1 + \log q_\lambda(\boldsymbol{\theta}))$$

REINFORCE is more widely applicable compared to the reparametrization trick, but has higher variance.

Homework

1. Derive the expression for REINFORCE.
2. VI for Linear regression
   (a) Derive the variational lower bound for a linear-regression problem. Assume a Gaussian prior on $\boldsymbol{\theta}$.
   (b) What kind of posterior approximation is appropriate in this case?
   (c) When will the lower bound be tight?
   (d) What is the memory and computational complexity of SGD?

References

[Kingma and Welling, 2013] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*.

[Nickisch and Rasmussen, 2008] Nickisch, H. and Rasmussen, C. (2008). Approximations for binary Gaussian process classification. *Journal of Machine Learning Research*, 9(10).

[Williams, 1992] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Extra space

Extra space