

# DP bayes by backprop: algorithm and analysis

May 14, 2020

This section describes a differentially private bayes by backprop(BBB) algorithm(modified from [Weight Uncertainty in Neural Networks](#)), the moment accountant approach (keeping track of privacy loss), and hyperparameter tuning(left during implementation).

## 1 sensitivity Analysis

The loss function of bayes by back prop is

$$f(\mathbf{w}, \theta) = \log q(\mathbf{w}|\theta) - \log P(\mathbf{w}) - \sum_i \log P(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w})$$

Therefore, the BP gradient to  $\mathbf{w}$  is:

$$\frac{\partial f}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}} [\log q(\mathbf{w}|\theta) - \log P(\mathbf{w})] - \sum_i \frac{\partial}{\partial \mathbf{w}} \log P(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w})$$

For neighboring database

$$\Delta_2 \frac{\partial f}{\partial \mathbf{w}} = \Delta_2 \sum_i \frac{\partial}{\partial \mathbf{w}} \log P(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w}) \leq 2 \max_{x_i, y_i} \frac{\partial}{\partial \mathbf{w}} \log P(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w})$$

Since we experiment on MNIST classification,

$$\Delta_2 \frac{\partial f}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}} (y_i \cdot p_i)$$

where  $p_i$  is a vector of the same size as  $y_i$ , and its elements sum to 1, and  $p_i \propto \exp(W_2(\sigma(W_1 x_i)))$ , where  $W_1, W_2$  are weight matrix in the first and second layer.

If we consider the bound of the gradient norm to be  $C$ , we can get the following DP-BBB algorithm.

## 2 Differentially Private BBB Algorithm(Gaussian posterior)

---

**Algorithm 1:** Differentially Private BBB(Outline)

---

**Input:** Examples  $\{x_1, \dots, x_n\}$ , Loss function  
 $f(\mathbf{w}, \theta) = \log q(\mathbf{w}|\theta) - \log P(\mathbf{w}) - \sum_i \log P(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w})$ , Prior distribution  $P(\mathbf{w})$ . Parameters: learning rate  $\eta_t$ , noise scale  $\sigma$ , gradient norm bound C  
initialize  $\mu^0, \theta^0$  randomly for  $t=0$   
construct train-loader: split training data to  $B$  different batches and index each batch using the tuple  $(batch_b, x_{train}, y_{train})$

**for**  $t \in [T]$  **do**

- $(\mu_0, \theta_0) = (\mu^t, \theta^t)$
- for**  $(batch_b, x_{train}, y_{train}) \in train - loader$  **do**

  - Sample  $\epsilon \sim \mathcal{N}(0, I)$
  - Calculate  $\mathbf{w}_b = \mu + \log(1 + \exp(\rho)) \odot \epsilon$
  - Compute the point-wise BP gradient of likelihood function to  $\mathbf{w}_b$ : For each  $i \in batch$ , compute  $\mathbf{g}_b(x_i) \leftarrow \nabla_{\mathbf{w}_b} - \log P(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w}_b)$
  - Clip gradient:  $\bar{g}_b(x_i) \leftarrow \bar{g}_b(x_i) / \max(1, |\bar{g}_b(x_i)|_2 / C)$
  - Add noise:  $\bar{g}_b(batch) = \sum_i \bar{g}_b(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})$  Calculate the noisy BP gradient of  $f(\mathbf{w}, \theta)$  to  $\mathbf{w}_b$ :  $\bar{g}_b = \bar{g}_b(batch) - \nabla_{\mathbf{w}_b} \log P(w_b)$
  - Let  $\theta_b = (\mu_b, \rho_b)$
  - Calculate the BP gradient of  $f(\mathbf{w}, \theta)$  to  $\theta_b = (\mu_b, \rho_b)$ :
$$m_b = \nabla_{\mu_b} P(w_b|\mu_b, \rho_b), r_b = \nabla_{\rho_b} P(w_b|\mu_b, \rho_b)$$
  - Calculate the noisy overall BP gradient of  $f(\mathbf{w}_b(\epsilon_b, \theta_b), \theta_b)$  to  $\theta_b = (\mu_b, \rho_b)$ :
$$\Delta_{\mu_b} = \bar{g}_b + m_b, \Delta_{\rho_b} = \bar{g}_b \frac{\epsilon}{1 + \exp(-\rho)} + r_b$$
  - Update the variational parameters
$$\mu_{b+1} \leftarrow \mu_b - \eta_b \Delta_{\mu_b}, \rho_{b+1} \leftarrow \rho_b - \eta_b \Delta_{\rho_b}$$

- end**
- $(\mu_{t+1}, \theta_{t+1}) = (\mu_B, \theta_B)$
- end**

**Output:**  $\theta_T = (\mu_T, \rho_T)$  and compute the overall privacy cost  $(\epsilon, \delta)$  using a privacy accounting method.

---

### 3 Analysis of the Privacy Bound

For each epoch, there is only 1 batch which will result in potentially different update of parameters. Therefore we only need to consider privacy cost of each epoch, and then do composition.

One can see the sensitivity of  $f(\mathbf{w}(\epsilon, \theta), \theta)$  to  $\theta = (\mu, \rho)$  (after gradient clipping etc.) is smaller than  $C$ . Therefore by choosing  $\sigma$  to be  $\sqrt{2\log\frac{1.25}{\delta}}/\epsilon$ , then by property of gaussian mechanism,

In one of the batch in the epoch,  $\frac{\partial f}{\partial \mathbf{w}}$  will be  $(\epsilon, \delta)$ -differentially private.

By post-processing property of differential privacy, the whole one epoch is  $(\epsilon, \delta)$ -differentially private.

#### 3.1 Standard Composition Analysis

By advanced composition theorem, the whole algorithm achieves  $(O(\sqrt{T}\epsilon\sqrt{1/\delta}), T\delta)$ -differential privacy. (**T is not the number of iterations, but the number of epochs.**)

#### 3.2 Moment Accountant Analysis

Since we are adding Gaussian noise, by exactly the same process in [Deep Learning with Differential Privacy](#), we can get a stronger bound than strong composition theorem, the whole algorithm achieves  $(O(\sqrt{T}\epsilon), \delta)$ -differential privacy, where  $B$  is the number of batches in one epoch.

**The DP bound for BBB algorithm is the same as that of the noisy SGD algorithm.**

Take moment accountant analysis as an example, we assume: the gradient sensitivity for one datapoint is the same; calculating the fully-batched gradient is  $(\epsilon, \delta)$ -differentially private;  $N$  is the size of the database is the same; the batch size  $B$  are the same; we run roughly the same number of epochs  $T$ .

Then in DP-SGD, one iteration would be  $(O(\sqrt{B/N}\epsilon), \sqrt{B/N}\delta)$ , the algorithm would be  $(O(\sqrt{TN/B}\sqrt{B/N}\epsilon), \delta)$ -differentially private, i.e.  $(O(\sqrt{T}\epsilon), \delta)$ -differentially private.

And the DP-BBB is  $(O(\sqrt{T}\epsilon), \delta)$ -differentially private. So these two bounds are the same.