

Variational Dropout and the Local Reparameterization Trick

—

2/21/2024

Motivation for the paper

It is useful to regularize the training of neural networks in order to prevent overfitting the training data. Dropout is a popular method that is empirically shown to be effective.

A previous work shows that Gaussian dropout optimizes a lower bound on the marginal likelihood of the data. This paper extends this relationship between dropout and Gaussian inference to improve variational Bayesian inference on model parameters.

Motivation for the paper

Local reparameterization trick:

- Improves efficiency of stochastic gradient-based variational inference (SGVB) with minibatches of data such that the optimization speed matches dropout

Variational dropout:

- Equivalence between Gaussian dropout and SVGB with the local reparameterization trick applied
- By casting dropout as a variational inference problem, we can also learn the dropout rates

Variational Inference (Review)

- Computing the true posterior distribution is intractable.
- Instead, cast inference as an optimization problem where we wish to optimize the parameters ϕ of some distribution q_ϕ such that q_ϕ is an approximation of the true posterior, as measured by the Kullback-Leibler divergence.
- To do this, we maximize the variational lower bound of the marginal likelihood of the data.

Variational Lower Bound (Review)

$$\mathcal{L}(\phi) = -D_{KL}(q_\phi(\mathbf{w})||p(\mathbf{w})) + L_{\mathcal{D}}(\phi)$$

where $L_{\mathcal{D}}(\phi) = \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}} \mathbb{E}_{q_\phi(\mathbf{w})} [\log p(\mathbf{y}|\mathbf{x}, \mathbf{w})]$

← variational lower bound on the marginal likelihood of the data

↑ expected log-likelihood

Stochastic Gradient Variational Bayes



Stochastic Gradient Variational Bayes (SGVB)

SGVB is a method for gradient-based optimization of the variational bound that uses minibatches and requires differentiable p and q distributions.

Parameterize the random parameters (for example, the weights of a neural network) by a differentiable function f :

$$w \sim q_{\phi}(w) \text{ where } w = f(\varepsilon, \phi)$$

And ε is a random noise variable: $\varepsilon \sim p(\varepsilon)$

Stochastic Gradient Variational Bayes (SGVB)


This allows us to write an unbiased differentiable minibatch based Monte Carlo estimator of the expected log-likelihood

$$L_{\mathcal{D}}(\phi) \simeq L_{\mathcal{D}}^{\text{SGVB}}(\phi) = \frac{N}{M} \sum_{i=1}^M \log p(\mathbf{y}^i | \mathbf{x}^i, \mathbf{w} = f(\epsilon, \phi))$$

M : size of the minibatch

ϵ : noise vector

Stochastic Gradient Variational Bayes (SGVB)

$$\mathcal{L}(\phi) = -D_{KL}(q_{\phi}(\mathbf{w})||p(\mathbf{w})) + L_{\mathcal{D}}(\phi)$$


minibatch based
estimator of this term
(previous slide)

Here, we assume that the KL divergence can be computed deterministically. If KL cannot be computed deterministically, we approximate it similarly.

This results in an estimator of the variational lower bound which is differentiable with respect to ϕ . Thus, we can optimize by stochastic gradient ascent.

Variance of the SGVB Estimator

The performance of stochastic gradient ascent depends on the variance of the gradients. Large gradients can delay or prevent convergence.

We prefer estimators with smaller variance.

Variance of the SGVB Estimator

$$\begin{aligned}\text{Var} [L_{\mathcal{D}}^{\text{SGVB}}(\phi)] &= \frac{N^2}{M^2} \left(\sum_{i=1}^M \text{Var} [L_i] + 2 \sum_{i=1}^M \sum_{j=i+1}^M \text{Cov} [L_i, L_j] \right) \\ &= N^2 \left(\frac{1}{M} \text{Var} [L_i] + \frac{M-1}{M} \text{Cov} [L_i, L_j] \right),\end{aligned}$$



Increasing the minibatch size only impacts the contribution of variance.



The contribution of the covariance between data points is not reduced by increasing the minibatch size.

Local Reparameterization Trick

—

Local Reparameterization Trick

There are two motivating factors behind this trick: reducing the variance and making the estimator computationally efficient.

Kigima et al. propose an estimator such that the covariance between the log-likelihood of data points i and j is zero, ensuring that the variance can be scaled using minibatch size.

$$\text{Cov}[L_i, L_j] = 0$$

Rather than sample the noise ϵ directly, only sample $f(\epsilon)$ that influence the SGVB estimator. This uncertainty is independent across examples, translates global uncertainty into local uncertainty, and is easier to sample.

Local Reparameterization Example

Consider one layer of a fully connected neural network with 1000 neurons.

Input feature matrix \mathbf{A} : $M \times 1000$

Weight matrix \mathbf{W} : 1000×1000

Activations $\mathbf{B} = \mathbf{AW}$: $M \times 1000$

Let's say the posterior approximation on weights is a Gaussian:

$$q_{\phi}(w_{i,j}) = N(\mu_{i,j}, \sigma^2_{i,j}) \forall w_{i,j} \in \mathbf{W}$$

$$w_{i,j} = \mu_{i,j} + \varepsilon_{i,j} \cdot \sigma_{i,j} \text{ where } \varepsilon_{i,j} \sim N(0, 1)$$

Local Reparameterization Example

Drawing the weights for just this fully connected layer requires 1000^2 samples.

Notice that if we use only one weight vector for the entire minibatch:

$$\text{Cov}[L_i, L_j] \neq 0$$

We could draw new weights for every single example in the batch and meet this requirement, but this requires $M * 1000^2$ draws, sacrificing the computational efficiency.

Local Reparameterization Example

The approximate posterior distribution on the weights can be turned into a posterior on the activations, conditioned on the input \mathbf{A} . Thus with this reparameterization we can sample the activations \mathbf{B} directly.

$$q_\phi(w_{i,j}) = N(\mu_{i,j}, \sigma_{i,j}^2) \quad \forall w_{i,j} \in \mathbf{W} \quad \implies \quad q_\phi(b_{m,j} | \mathbf{A}) = N(\gamma_{m,j}, \delta_{m,j}), \text{ with}$$
$$\gamma_{m,j} = \sum_{i=1}^{1000} a_{m,i} \mu_{i,j}, \quad \text{and} \quad \delta_{m,j} = \sum_{i=1}^{1000} a_{m,i}^2 \sigma_{i,j}^2.$$

$$b_{m,j} = \gamma_{m,j} + \sqrt{\delta_{m,j}} \zeta_{m,j}, \text{ with } \zeta_{m,j} \sim N(0, 1)$$

This method only requires $M \times 1000$ draws to obtain all of the activations - computationally efficient!

Local Reparameterization Example

What about the variance?

Consider the gradient estimate with respect to posterior variance, given $M = 1$. The stochastic terms are shown in red. Left: random weights, right: local reparameterization trick.

$$\frac{\partial L_{\mathcal{D}}^{\text{SGVB}}}{\partial \sigma_{i,j}^2} = \frac{\partial L_{\mathcal{D}}^{\text{SGVB}}}{\partial b_{m,j}} \frac{\epsilon_{i,j} a_{m,i}}{2\sigma_{i,j}}$$

$$\frac{\partial L_{\mathcal{D}}^{\text{SGVB}}}{\partial \sigma_{i,j}^2} = \frac{\partial L_{\mathcal{D}}^{\text{SGVB}}}{\partial b_{m,j}} \frac{\zeta_{m,j} a_{m,i}^2}{2\sqrt{\delta_{m,j}}}$$

Estimating the covariance between the gradient and the noise is easier for the second term. In the first, 1000 noise variables influence the gradient and thus the relationship between the two is lost in noise.

Variational Dropout

—

Dropout

Dropout as multiplicative noise on the input, using the same notation as the example.

$$\mathbf{B} = (\mathbf{A} \circ \xi)\theta, \quad \text{with } \xi_{i,j} \sim p(\xi_{i,j})$$

θ : weight matrix

ξ : independent noise vector, drawn from for example a Bernoulli or Gaussian distribution

Variational Dropout: Independent Weight Noise

Gaussian dropout: If ξ is independently drawn from $N(1, \alpha)$, the marginal distribution of the activation is also Gaussian:

$$q_{\phi}(b_{m,j}|\mathbf{A}) = N(\gamma_{m,j}, \delta_{m,j}), \text{ with } \gamma_{m,j} = \sum_{i=1}^K a_{m,i}\theta_{i,j}, \text{ and } \delta_{m,j} = \alpha \sum_{i=1}^K a_{m,i}^2\theta_{i,j}^2.$$

This ignores dependencies between elements of \mathbf{B} .

Variational Dropout: Correlated Weight Noise

Rather than ignoring dependencies between elements of \mathbf{B} , set up the activations such that we draw a stochastic scale variable s that will scale all weights associated with one neuron.

$$\mathbf{B} = (\mathbf{A} \circ \xi)\theta, \quad \xi_{i,j} \sim N(1, \alpha) \iff \mathbf{b}^m = \mathbf{a}^m \mathbf{W}, \text{ with} \\ \mathbf{W} = (\mathbf{w}'_1, \mathbf{w}'_2, \dots, \mathbf{w}'_K)', \text{ and } \mathbf{w}_i = s_i \theta_i, \text{ with } q_\phi(s_i) = N(1, \alpha)$$

Dropout Prior and Posterior

A *dropout posterior* must have a multiplicative noise term determined by α .

The only prior that can be used is the scale invariant log-uniform prior, as this ensures the KL divergence does not depend on the parameters θ that we are attempting to maximize.

$$p(\log(|w_{i,j}|)) \propto c$$

The authors also claim that using this prior regularizes the number of significant digits stores for the weights, which would help prevent the network from overfitting to noise in the data.

Dropout's Variational Objective

Thus the dropout variational objective maximizes the following bound with respect to θ :

$$\mathbb{E}_{q_\alpha} [L_{\mathcal{D}}(\theta)] - D_{KL}(q_\alpha(\mathbf{w}) || p(\mathbf{w}))$$

And the authors prove in an appendix that with a factorized Gaussian approximate posterior, while the KL divergence cannot be analytically evaluated, it can be approximated by

$$-D_{KL}[q_\phi(w_i)|p(w_i)] \approx \text{constant} + 0.5 \log(\alpha) + c_1\alpha + c_2\alpha^2 + c_3\alpha^3$$

Adaptive Regularization by Optimizing Dropout Rate

Dropout rate is typically treated as a tunable hyperparameter.

Now, given that we have a variational objective, the model can learn α by optimizing the objective with respect to α .

Experiments



Variance

	top layer 10 epochs	top layer 100 epochs	bottom layer 10 epochs	bottom layer 100 epochs
stochastic gradient estimator				
local reparameterization (ours)	7.8×10^3	1.2×10^3	1.9×10^2	1.1×10^2
weight sample per data point (slow)	1.4×10^4	2.6×10^3	4.3×10^2	2.5×10^2
weight sample per minibatch (standard)	4.9×10^4	4.3×10^3	8.5×10^2	3.3×10^2
no dropout noise (minimal var.)	2.8×10^3	5.9×10^1	1.3×10^2	9.0×10^0

Table 1: Average empirical variance of minibatch stochastic gradient estimates (1000 examples) for a fully connected neural network, regularized by variational dropout with independent weight noise.

Results shown for variational dropout with noise on the weights. The local reparameterization trick shows the lowest variance in the estimator among all methods that use dropout.

Speed

Regular SGVB estimator: 1635 seconds/epoch

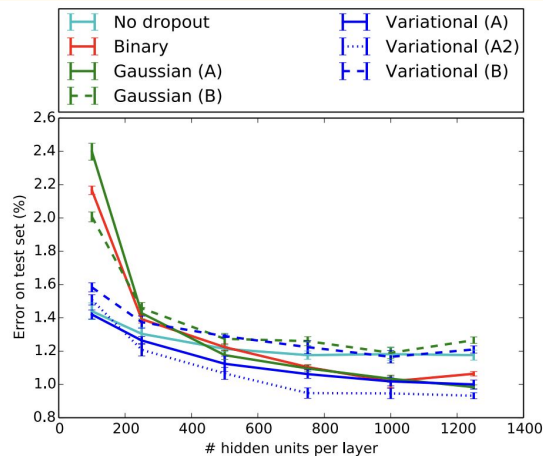
Local reparameterization estimator: 7.4 seconds/epoch

Experiments

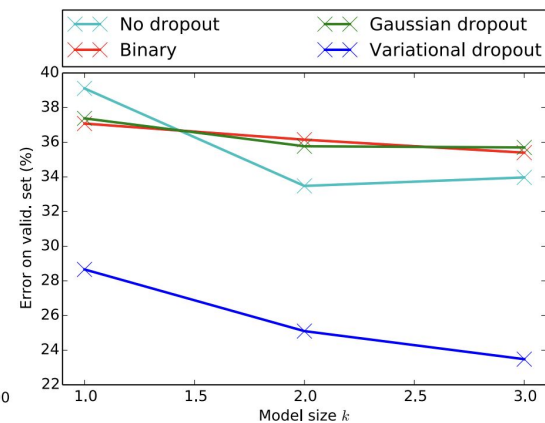
A: correlated weight noise

B: independent weight noise

A2: type A with the
KL-divergence downscaled



(a) Classification error on the MNIST dataset



(b) Classification error on the CIFAR-10 dataset

Figure 1: Best viewed in color. **(a)** Comparison of various dropout methods, when applied to fully-connected neural networks for classification on the MNIST dataset. Shown is the classification error of networks with 3 hidden layers, averaged over 5 runs. The variational versions of Gaussian dropout perform equal or better than their non-adaptive counterparts; the difference is especially large with smaller models, where regular dropout often results in severe underfitting. **(b)** Comparison of dropout methods when applied to convolutional net a trained on the CIFAR-10 dataset, for different settings of network size k . The network has two convolutional layers with each $32k$ and $64k$ feature maps, respectively, each with stride 2 and followed by a softplus nonlinearity. This is followed by two fully connected layers with each $128k$ hidden units.