Variational Dropout*

Jannik Gut

October 2020

1 Inference

1.1 Bayesian Inference

Goal: $p(y|x, w)$
Prior: $p(w)$
Posterior: $p(w|D) = p(w)p(D|w)/p(D)$, where it is very hard to find good values for the second and third probability.

1.2 Variational Inference

We optimize ϕ of $q_\phi(w)$, such that q is a close approximation to $p(w|D)$, measured by the Kullback-Leibler divergence $D_{KL}(q_\phi(w)\|p(w|D)) = \sum_x q_\phi(w) log(\frac{q_\phi(w)}{p(w|D)})$.
In practice this is done by maximizing variational lower bound $L(\phi)$ of the marginal likelihood of the data:
$L(\phi) = -D_{KL}(q_\phi(w)\|p(w)) + L_D(\phi)$
$L_D(\phi) = \sum_{(x,y)\in D} \mathbb{E}_{q_\phi(w)}[log(p(y|x, w))]$ is the expected log-likelihood.
$L(\phi)+D_{KL}(q_\phi(w)\|p(w|D)) = \sum_{(x,y)\in D} log(p(y|x))$ is the (conditional) marginal log likelihood, which by the right-hand side of the equation is constant in ϕ, which means maximizing the bound will minimize $D_{KL}(q_\phi(w)\|p(w|D))$.

1.3 Stochastic Gradient Variational Bayes(SGVB)

Like Variational Inference, but $q_\phi(w) \sim f(\epsilon, \phi)$, where f is a differentiable function and ϵ is a random noise variable. An unbiased differentiable minibatch-based Monte Carlo estimator of the expected log-likelihood is now:
$L_D(\phi) \simeq L_D^{SGVB}(\phi) = \frac{N}{M} \sum_{i=1}^{M} log(p(y^i|x^i, w = f(\epsilon, \phi)))$, where the M and i are a minibatch of D. Also the $\phi-$gradient is unbiased.

*https://papers.nips.cc/paper/5666-variational-dropout-and-the-local-reparameterization-trick

1.4 Variance of the SGVB

In practice the performance of stochastic gradient ascent crucially depends on the variance of the gradients, as the gradients should be small.
For notation: $L_i = log(log(y^i|x^i, w = f(\epsilon^i, \phi)))$ and $L_D^{SGVB}(\phi) = \frac{N}{M}\sum_{i=1}^{M} L_i$.

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

The influence of $Var[L_i]$ is inversely proportional to M, but the influence of the covariances does not decrease with M. In practice, this means the variance can be dominated by the covariances by even moderately large M.

1.5 Local Reparameterization Trick

We can be much faster if the covariances can be set to 0 and the stochastic gradients scale as $\frac{1}{M}$ and as a bonus not sampling ϵ but $f(\epsilon)$, as like that the global uncertainty in the weights gets translated to local uncertainty, which is easier to sample.
Example:
A(M x 1000): Input
W(M x 1000 x 1000): Weights $\sim w_{i,j} = \mu_{i,j} + \sigma_{i,j}\epsilon_{i,j}$ and ϵ is drawn from a normal distribution.
B(M x 1000): Output to non-linearity of neural-net
In this configuration the covariance is 0 if we sample a new W(1000x1000) for each of the M examples.

Instead, sample from B directly:
$q_\phi(b_{m,j}|A) = N(\gamma_{m,j}, \delta_{m,j})$
with $\gamma_{m,j} = \sum_{i=1}^{1000} a_{m,i}\mu_{i,j}$, $\delta_{m,j} = \sum_{i=1}^{1000} a_{m,i}^2\sigma_{i,j}^2$
and then use $b_{m,j} = \gamma_{m,j} + \sqrt{\delta_{m,j}}\zeta_{m,j}$
ζ(M x 1000) being similarly distributed as ϵ and is the only thing that additionally needs drawing instead of all of W(M x 1000 x 1000).
Also the variance is lower as can be seen with the stochastic gradient estimate with respect to $\sigma_{i,j}^2$:

$$\frac{\partial L_D^{SGVB}}{\partial b_{m,j}}\frac{\epsilon_{i,j}a_{m,i}}{2\sigma_{i,j}} \geq \frac{\partial L_D^{SGVB}}{\partial b_{m,j}}\frac{\zeta_{m,j}a_{m,i}^2}{2\sqrt{\delta_{i,j}}}$$

Additionally only one ζ is needed for the gradient instead of all of the ϵ.

2 Variational Dropout

Dropout:
$B = (A \circ \xi)\theta$ with $\xi_{i,j} \sim p(\xi_{i,j})$
p initially was a Bernoulli distribution, but a normal distribution works as well or better.

2.1 Reparameterize dropout

Similarly to the previous reparameterization trick we can reparameterize the dropped-out information B with the Gaussian N(1,α), but this ignores the dependencies in the activation noise:
$q_\theta(b_{m,j}|A) = N(\gamma_{m,j}, \delta_{m,j})$, with $\gamma_{m,j} = \sum_{i=1}^{K} a_{m,i}\theta_{i,j}$ and $\delta_{m,j} = \alpha \sum_{i=1}^{K} a_{m,i}^2 \theta_{i,j}^2$

2.2 Keeping dependencies

Interpreting dropout as a form of correlated weight noise:
$B = (A \circ \xi)\theta$ with $\xi_{i,j} \sim N(1, \alpha) \Longleftrightarrow b_{:m} = a_{:m}W$
with $W_{:i} = s_{:i}\theta_{:i}$ and $q_\phi(s_{:i}) = N(1, \alpha)$

2.3 both together

During dropout training, θ is adapted to maximize the expected log likelihood $\mathbb{E}_{q_\alpha}[L_D(\theta)$, but this is only consistent if $D_{KL}(q_\phi(w)\|p(w))$ does not depend on θ, which is only true if $p(log(|w_{i,j}|)) \sim c$.
Dropout training maximizes, with respect to θ:
$\mathbb{E}_{q_\alpha}[L_D(\theta) - D_{KL}(q_\phi(w)\|p(w))$, where the second term can only be approximated by a function that is in poly(α)

3 other

Dropout can be interpreted as variational inference, that allows α to be adaptive, make alpha small (with constraints), so no local optima.
MNIST is used, since it is a de-facto standard. No hyper-parameters.
The variation is about half/ a quarter as big as previous methods, but the speed-up is quite drastically at around 200x.
The accuracy is similar or even better than the other versions, but they are about as good as no dropout; why even bother?