Early Stopping as Nonparametric Variational Inference

David Duvenaud* Harvard University

Dougal Maclaurin* Harvard University

Ryan P. Adams Harvard University

Abstract

We show that unconverged stochastic gradient descent can be interpreted as sampling from a nonparametric approximate posterior distribution. This distribution is implicitly defined by the transformation of an initial distribution by a sequence of optimization steps. By tracking the change in entropy of this distribution during optimization, we give a scalable, unbiased estimate of a variational lower bound on the log marginal likelihood. This bound can be used to optimize hyperparameters instead of cross-validation. This Bayesian interpretation of SGD also suggests new overfitting-resistant optimization procedures, and gives a theoretical foundation for early stopping and ensembling. We investigate the properties of this marginal likelihood estimator on neural network models.

1 Introduction

In much of machine learning, the central computational challenge is optimization: we try to minimize some training-set loss with respect to a set of model parameters. If we treat the training loss as a negative log-posterior, this amounts to searching for a maximum *a posteriori* (MAP) solution. Paradoxically, over-zealous optimization can yield worse test-set results than incomplete optimization due to the phenomenon of *over-training*. A popular remedy to over-training is to invoke "early stopping" in which optimization is halted based on the continually monitored performance of the parameters on a separate validation set. However, early stopping is both theoretically unsatisfying and incoherent from a research perspective: how can one rationally design better optimization methods if the goal is to achieve something "powerful but not *too* powerful"? A related trick is to ensemble the results from multiple optimization runs from different starting positions. Similarly, this must rely on imperfect optimization, since otherwise all optimization runs would reach the same optimum.

We propose an interpretation of incomplete optimization in terms of variational Bayesian inference, and provide a simple method for estimating the marginal likelihood of the approximate posterior. Our starting point is a Bayesian posterior distribution for a potentially complicated model, in which there is an empirical loss that can be interpreted as a negative log likelihood and regularizers that have interpretations as priors. One might proceed with MAP inference, and perform an optimization to find the best parameters. The main idea of this paper is that such an optimization procedure, initialized according to some distribution that can be chosen freely, generates a sequence of distributions that are implicitly defined by the action of the optimization update rule on the previous distribution. We can treat these distributions as variational approximations to the true posterior distribution. A single optimization run for N iterations represents a draw from the Nth such distribution in the sequence. Figure 1 shows contours of these approximate distributions on an example posterior.

With this interpretation, the number of optimization iterations can be seen as a variational parameter, one that trades off fitting the data well against maintaining a broad (high entropy) distribution. Early stopping amounts to optimizing the variational lower bound (or an approximation based on a validation set) with respect to this variational parameter. Ensembling different random restarts can be viewed as taking independent samples from the variational posterior.

To establish whether this viewpoint is helpful in practice, we ask: can we efficiently estimate the marginal likelihood implied by unconverted optimization? We tackle this question in section 2. Specifically, for stochastic gradient descent (SGD), we show how to compute an unbiased estimate of a lower bound on the log marginal likelihood of each iteration's implicit variational distribution. We also introduce an 'entropy-friendly' variant of SGD that maintains better-behaved implicit distributions.

We also ask whether model selection based on these marginal likelihood estimates picks models with good test-

*Equal contribution.

Appearing in Proceedings of the 19^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 51.

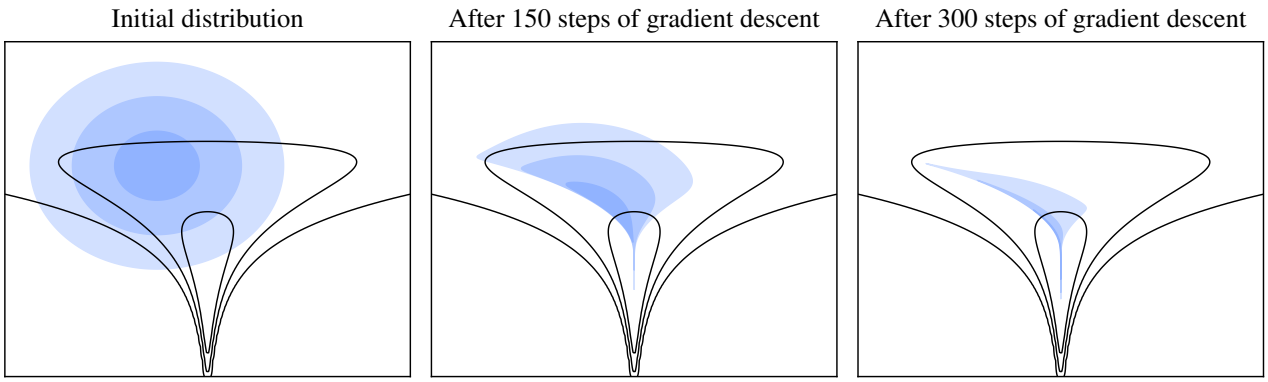

Figure 1: A series of distributions (blue) implicitly defined by gradient descent on an objective (black). These distributions are defined by mapping each point in the initial distribution through a fixed number of iterations of optimization. These distributions have nonparametric shapes, and eventually concentrate around the optima of the objective.

time performance. We give some experimental evidence in both directions in section 5. A related question is how close the variational distributions implied by various optimization rules approximate the true posterior. We briefly address this question in section 6.

1.1 Contributions

- We introduce a new interpretation of optimization algorithms as samplers from a variational distribution that adapts to the true posterior, eventually collapsing around its modes.
- We provide a scalable estimator for the entropy of these implicit variational distributions, allowing us to estimate a lower bound on the marginal likelihood of any model whose posterior is twice-differentiable, even on problems with millions of parameters and data points.
- In principle, this marginal likelihood estimator can be used for hyperparameter selection and early stopping without the need for a validation set. We investigate the performance of these estimators empirically on neural network models, and show that they have reasonable properties. However, further refinements are likely to be necessary before this marginal likelihood estimator is more practical than using a validation set.

2 Incomplete optimization as variational inference

Variational inference (Wainwright & Jordan, 2008) aims to approximate an intractable posterior distribution, $p(\theta|\mathbf{x})$, with another more tractable distribution, $q(\theta)$. The usual measure of the quality of the approximation is the Kullback-Leibler (KL) divergence from $q(\theta)$ to $p(\theta, \mathbf{x})$. This measure provides a lower bound on the marginal likelihood of the original model; applying Bayes' rule to the definition of $\mathrm{KL}\left(q(\theta)\|p(\theta|\mathbf{x})\right)$ gives the familiar inequality:

$$\log p(\mathbf{x}) \geq \underbrace{-\mathbb{E}_{q(\theta)}\left[-\log p(\theta, \mathbf{x})\right]}_{\text{Energy } E[q]} \underbrace{-\mathbb{E}_{q(\theta)}\left[\log q(\theta)\right]}_{\text{Entropy } S[q]} := \mathcal{L}[q] \tag{1}$$

Maximizing $\mathcal{L}[q]$, the variational lower bound on the marginal likelihood, with respect to q minimizes $\mathrm{KL}\left(q(\theta)\|p(\theta|\mathbf{x})\right)$, the KL divergence from q to the true posterior, giving the closest approximation available within the variational family. A convenient side effect is that we also get a lower bound on $p(\mathbf{x})$, which can be used for model selection.

To perform variational inference, we require a family of distributions over which to maximize $\mathcal{L}[q]$. Consider a general procedure to minimize the energy $(-\log p(\theta, \mathbf{x}))$ with respect to $\theta \in \mathbb{R}^D$. The parameters θ are initialized according to some distribution $q_0(\theta)$ and updated at each iteration according to a transition operation $T : \mathbb{R}^D \to \mathbb{R}^D$:

$$\theta_0 \sim q_0(\theta)$$
$$\theta_{t+1} = T(\theta_t),$$

Our variational family consists of the sequence of distributions $q_0, q_1, q_2, \ldots$, where $q_t(\theta)$ is the distribution over θ_t generated by the above procedure. These distributions don't have a closed form, but we can exactly sample from q_t by simply running the optimizer for t steps starting from a random initialization.

As shown in (1), $\mathcal{L}$ consists of an energy term and an entropy term. The energy term measures how well q fits the data and the entropy term encourages the probability mass of q to spread out, preventing overfitting. As optimization of θ proceeds from its q_0-distributed starting point, we can examine how $\mathcal{L}$ changes. The negative energy term grows, since the goal of the optimization is to reduce the energy.

The entropy term shrinks because the optimization converges over time. Optimization thus generates a sequence of distributions that range from underfitting to overfitting, and the variational lower bound captures this tradeoff.

We cannot evaluate $\mathcal{L}[q_t]$ exactly, but we can obtain an unbiased estimator. Sampling θ_0 from q_0 and then applying the transition operator t times produces an exact sample θ_t from $q_t(\theta)$, by definition. Since θ_t is an exact sample from $q_t(\theta)$, $\log p(\theta_t, \mathbf{x})$ is an unbiased estimator of the energy term of (1). The entropy term is trickier, since we do not have access to the density $q(\theta)$ directly. However, if we know the entropy of the initial distribution, $S[q_0(\theta)]$, then we can estimate $S[q_t(\theta)]$ by tracking the change in entropy at each iteration, calculated by the change of variables formula.

To compute how the volume shrinks or expands due to an iteration of the optimizer, we require access to the Jacobian of the optimizer's transition operator, $J(\theta)$:

$$S[q_{t+1}] - S[q_t] = \mathbb{E}_{q_t(\theta_t)}\left[\log |J(\theta_t)|\right]. \tag{2}$$

Note that this analysis assumes that the mapping T is bijective. Combining these terms, we have an unbiased estimator of $\mathcal{L}$ at iteration T, based on the sequence of parameters, $\theta_0, \ldots, \theta_T$, from a single training run:

$$\mathcal{L}[q_T] \approx \underbrace{\log p(\theta_T, \mathbf{x})}_{\text{Energy}} + \underbrace{\sum_{t=0}^{T-1} \log |J(\theta_t)| + S[q_0]}_{\text{Entropy}}. \tag{3}$$

3 The entropy of stochastic gradient descent

In this section, we give an unbiased estimate for the change in entropy caused by SGD updates. We'll start with a naïve method, then in section 3.1, we give an approximation that scales linearly with the number of parameters in the model.

Stochastic gradient descent is a popular and effective optimization procedure with the following update rule:

$$\theta_{t+1} = \theta_t - \alpha \nabla L(\theta), \tag{4}$$

where the $L(\theta)$ the objective loss (or an unbiased estimator of it e.g. using minibatches) for example $-\log p(\theta, \mathbf{x})$, and α is a 'step size' hyperparameter. Taking the Jacobian of this update rule gives the following unbiased estimator for the change in entropy at each iteration:

$$S[q_{t+1}] - S[q_t] \approx \log |I - \alpha H_t(\theta_t)| \tag{5}$$

where H_t is the Hessian of $-\log p_t(\theta, \mathbf{x})$ with respect to θ.

Note that the Hessian does not need to be positive definite or even non-singular. If some directions in θ have negative curvature, as on the crest of a hill, it just means that optimization near there spreads out probability mass, increasing the entropy. There are, however, restrictions on α. If $\alpha\lambda_i = 1$, for any i, where λ_i are the eigenvalues of H_t, then the change in entropy will be undefined (infinitely negative). This corresponds to a Newton-like update where multiple points collapse to the optimum in a single step giving a distribution with zero variance in a particular direction. However, gradient descent is unstable anyway if $\alpha\lambda_{\max} > 2$, where $\lambda_{\max}$ is the largest eigenvalue of H_t. So if we choose a sufficiently conservative step size, such that $\alpha\lambda_{\max} < 1$, this situation should not arise. Algorithm 1 combines these steps into an algorithm that tracks the approximate entropy during optimization.

Algorithm 1 stochastic gradient descent with entropy estimate

input: Weight initialization scale σ_0, step size α, twice-differentiable negative log-likelihood $L(\theta, t)$
initialize $\theta_0 \sim \mathcal{N}(0, \sigma_0 \mathbf{I}_D)$
initialize $S_0 = \frac{D}{2}(1 + \log 2\pi) + D \log \sigma_0$
for $t = 1$ **to** T **do**
    $S_t = S_{t-1} + \log |\mathbf{I} - \alpha H_{t-1}|$ ▷ Update entropy
    $\theta_t = \theta_{t-1} - \alpha \nabla L(\theta_t, t)$ ▷ Update parameters
end for
output sample θ_T, entropy estimate S_T

So far, we have treated SGD as a deterministic procedure even though, as the name suggests, the gradient of the loss at each iteration may be replaced by a stochastic version. Our analysis of the entropy is technically valid if we fix the sequence of stochastic gradients to be the same for each optimization run, so that the only randomness comes from the parameter initialization. This is a tendentious argument, similar to arguing that a pseudorandom sequence of numbers has only as much entropy as its seed. However, if we do choose to randomize the gradient estimator differently for each training run (e.g. choosing different minibatches) then the expression for the change in entropy, Equation 5, remains valid as a *lower bound* on the change in entropy and the subsequent calculation of $\mathcal{L}$ remains a true lower bound on the log marginal likelihood.

3.1 Estimating the Jacobian in high dimensions

The expression for the change in entropy given by (5) is impractical for large-scale problems since it requires an $\mathcal{O}\left(D^3\right)$ determinant computation. Fortunately, we can make a good approximation using just two Hessian-vector products, which can usually be performed in $\mathcal{O}\left(D\right)$ time using reverse-mode differentiation (Pearlmutter, 1994).

The idea is that since $\alpha\lambda_{\max}$ is small, the Jacobian is just a small perturbation to the identity, and we can approximate

Algorithm 2 linear-time estimate of log-determinant of Jacobian of one iteration of stochastic gradient descent

1: **input:** step size α, current parameter vector θ, twice-differentiable negative log-likelihood $L(\theta)$
2: **initialize** $\mathbf{r}_0 \sim \mathcal{N}(0, \sigma_0 \mathbf{I}_D)$
3: $\mathbf{r}_1 = \mathbf{r}_0 - \alpha \mathbf{r}_0^\mathsf{T} \nabla\nabla L(\theta, t)$
4: $\mathbf{r}_2 = \mathbf{r}_1 - \alpha \mathbf{r}_1^\mathsf{T} \nabla\nabla L(\theta, t)$
5: $\hat{\mathcal{L}} = \mathbf{r}_0^\mathsf{T} \left(-2\mathbf{r}_0 + 3\mathbf{r}_1 - \mathbf{r}_2\right)$
6: **output** $\hat{\mathcal{L}}$, an unbiased estimate of a parabolic lower bound on the change in entropy.

its determinant using traces as follows:

$$\begin{aligned} \log|I - \alpha H| &= \sum_{i=0}^{D} \log\left(1 - \alpha\lambda_i\right) \\ &\geq \sum_{i=0}^{D} \left[-\alpha\lambda_i - (\alpha\lambda_i)^2\right] \qquad (6) \\ &= -\alpha \mathrm{Tr}\left[H\right] - \alpha^2 \mathrm{Tr}\left[HH\right] . \qquad (7) \end{aligned}$$

The bound in (6) is just a second order Taylor expansion of $\log(1-x)$ about $x = 0$ and is valid if $\alpha\lambda_i < 0.68$. As we argue above, the regime in which SGD is stable requires that $\alpha\lambda_{\text{max}} < 1$, so again choosing a conservative learning rate keeps this bound in the correct direction. For sufficiently small learning rates, this bound becomes tight.

The trace of the Hessian can be estimated using inner products of random vectors (Bai et al., 1996):

$$\mathrm{Tr}\left[H\right] = \mathbb{E}\left[\mathbf{r}^T H \mathbf{r}\right], \qquad \mathbf{r} \sim \mathcal{N}(0, I) . \qquad (8)$$

We use this identity to derive algorithm 2. In high dimensions, the exact evaluation of the determinant in step 5 should be replaced with the approximation given by algorithm 2.

Note that the quantity we are estimating (5) is well-conditioned, in contrast to the related problem of computing the log of the determinant of the Hessian itself. This arises, for example, in making the Laplace approximation to the posterior (MacKay, 1992). This is a much harder problem since the Hessian can be arbitrarily ill-conditioned, unlike our small Hessian-based perturbation to the identity.

3.2 Parameter initialization, priors, and objective functions

What initial parameter distribution should we use for SGD? The marginal likelihood estimate given by (3) is valid no matter which initial distribution we choose. We could conceivably optimize this distribution in an outer loop using the marginal likelihood estimate itself.

However, using the prior as the initialization distribution has several advantages. First, it is usually designed to have broad support. Since SGD usually decreases entropy, starting with a high-entropy distribution is a good heuristic.

The second advantage has to do with our choice of objective function. One option is to use the unnormalized log-posterior, but we can use any function we like. A more sensible choice is the negative log-likelihood. Variational distributions only differ from the initial distribution to the extent that the posterior differs from the prior. This difference is just the log-likelihood.

One nice implication of using the log-likelihood as the objective function is that the entropy estimate will be exactly correct for parameters that don't affect the likelihood, since their gradient (and corresponding rows of the Hessian) will always be zero. Because of these favorable properties, we use the prior as the initial distribution and log-likelihood as the objective in our experiments.

4 Entropy-friendly optimization methods

SGD optimizes the training loss, not the variational lower bound. In some sense, if this optimization happens to create a good intermediate distributions, it's only by accident! Why not design a new optimization method that produces good variational lower bounds? In place of SGD, we can use any optimization method for which we can approximate the change in entropy, which in practice means any optimization for which we can compute Jacobian-vector products.

An obvious place to start is with stochastic update rules inspired by Markov chain Monte Carlo (MCMC). Procedures like Hamiltonian Monte Carlo (Neal, 2011) and Langevin dynamics MCMC (Welling & Teh, 2011) look very much like optimization procedures but actually have the posterior as their stationary distribution. This is exactly the approach taken by Salimans et al. (2014). One difficulty with using stochastic updates, however, is that calculating the change in entropy at each iteration requires access to the current distribution over parameters. As an example, consider that convolving a delta function with a Gaussian yields an infinite entropy increase, whereas convolving a broad uniform distribution with a Gaussian yields only a small increase in entropy. Welling & Teh (2011) handle this by learning a highly parameterized "inverse model" which implicitly models the distribution over parameters. The downside of this approach is that the parameters of this model must be learned in an outer loop.

Another approach is to try to develop deterministic update rules that avoid some of the pathologies of update rules like SGD. This could could be a research agenda in itself, but we give one example here of a modification to SGD which can improve the variational lower bound. One problem with SGD in the context of posterior approximation is that SGD can collapse the implicit distribution into low-entropy

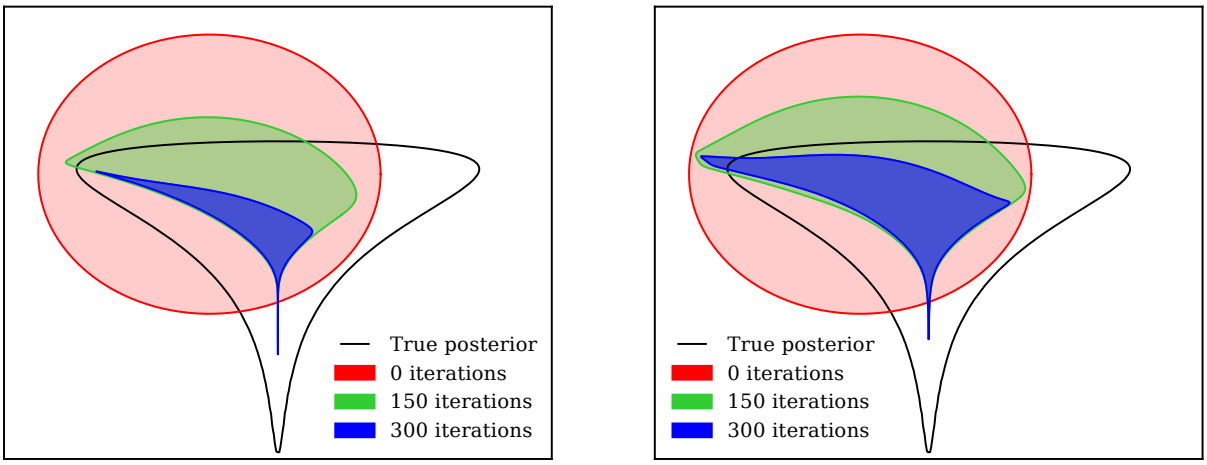

Figure 2: *Left*: The distribution implied by standard gradient descent. *Right*: The distribution implied by the modified, "entropy-friendly", gradient descent algorithm. The entropy-friendly distributions are slower to collapse into low-entropy filaments, causing the marginal likelihood to remain higher.

filaments, shrinking in some directions to be orders of magnitude smaller than the width of the true posterior. A simple trick to prevent this is to apply a nonlinear, parameter-wise warping to the gradient, such that directions of very small gradient do not get optimized all the way to the optimium. For example, the modified gradient (and resulting modified Jacobian) could be

$$g' = g - g_0 \tanh\left(g/g_0\right) \tag{9}$$

$$J' = \left(1 - \cosh^{-2}(g/g_0)\right) J \tag{10}$$

where g_0 is a "gradient threshold" parameter that sets the scale of this shrinkage. The effect is that entropy is not removed from parameters which are close to their optimum. An example showing the effect of this entropy-friendly modification is shown in Figure 2.

5 Experiments

In this section we show that the marginal likelihood estimate can be used to choose when to stop training, to choose model capacity, and to optimize training hyperparameters without the need for a validation set. We are not attempting to motivate SGD variational inference as a superior alternative to other procedures; we simply wish to give a proof of concept that the marginal likelihood estimator has reasonable properties. Further refinements are likely to be necessary before this marginal likelihood estimator is more practical than simply using a validation set.

5.1 Choosing when to stop optimization

As a simple demonstration of the usefulness of our marginal likelihood estimate, we show that it can be used to estimate the optimal number of training iterations before overfitting begins. We performed regression on the Boston housing dataset using a neural network with one hidden layer having 100 hidden units, sigmoidal activation functions, and no regularization. Figure 3 shows that marginal likelihood peaks at a similar place to the peak of held-out log-likelihood, which is where early stopping would occur when using a large validation set.

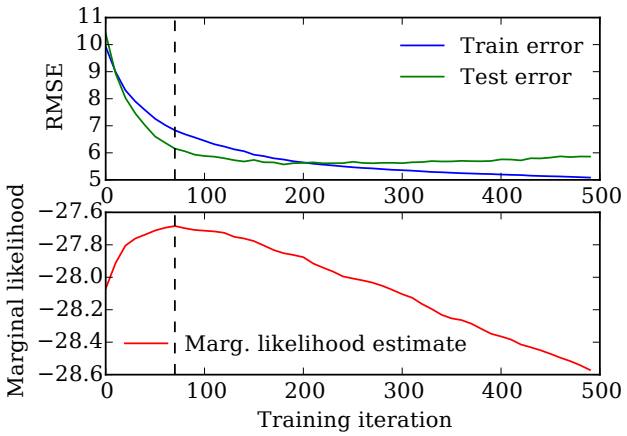

Figure 3: *Top*: Training and test-set error on the Boston housing dataset. *Bottom*: Stochastic gradient descent marginal likelihood estimates. The dashed line indicates the iteration with highest marginal likelihood. The marginal likelihood, estimated online using only the training set, and the test error peak at a similar number of iterations.

5.2 Choosing the number of hidden units

The marginal likelihood estimate is also comparable between training runs, allowing us to use it to select model

hyperparameters, such as the number of hidden units.

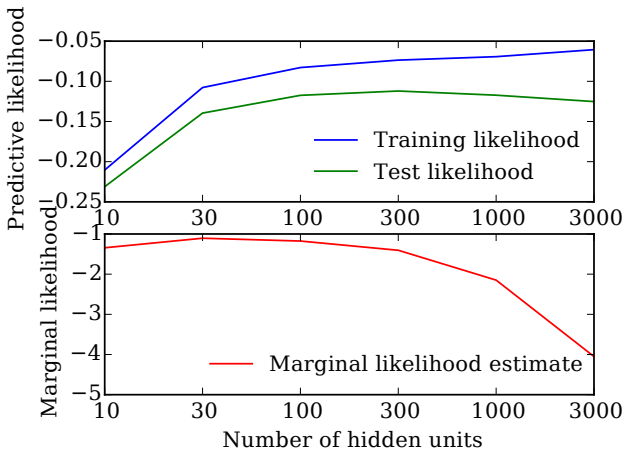

Figure 4: *Top*: Training and test-set likelihood as a function of the number of hidden units in the first layer of a neural network. *Bottom*: Stochastic gradient descent marginal likelihood estimates. In this case, the marginal likelihood over-penalizes high numbers of hidden units.

Figure 4 shows marginal likelihood estimates as a function of the number of hidden units in the hidden layer of a neural network trained on 50,000 MNIST handwritten digits. The largest network trained in this experiment contains 2 million parameters.

The marginal likelihood estimate begins to decrease for more than 30 hidden units, even though the test-set likelihood in maximized at 300 hidden units. We conjecture that this is due to the marginal likelihood estimate penalizing the loss of entropy in parameters whose contribution to the likelihood was initially large, but were made irrelevant later in the optimization.

5.3 Optimizing training hyperparameters

We can also use marginal likelihoods to optimize training parameters such as learning rates, initial distributions, or any other optimization parameters. As an example, Figure 5 shows the marginal likelihood estimate as a function of the gradient threshold in the entropy-friendly SGD algorithm from section 4 trained on 50,000 MNIST handwritten digits.

As the level of thresholding increases, the training and test error get worse due to under-fitting. However, for intermediate thresholds, the lower bound increases. Because it is a lower bound, its increase means that the estimate of the marginal likelihood of the exact model is becoming *more accurate*, even though the performance of the approximate model happens to be getting worse at the same time.

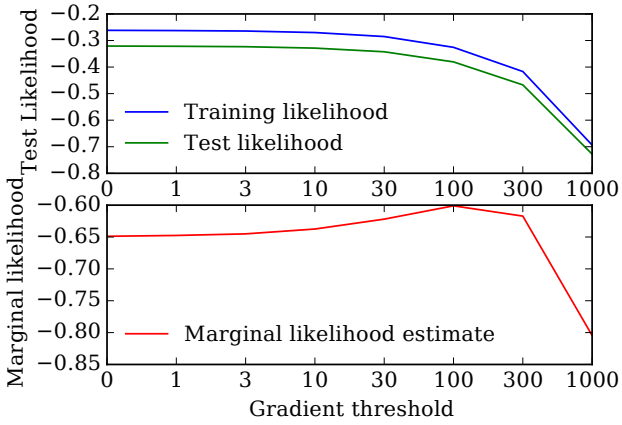

Figure 5: *Top*: Training and test-set likelihood as a function of the gradient threshold. *Bottom*: Marginal likelihood as a function of the gradient threshold. A gradient threshold of zero corresponds to standard SGD. The increased lower bound for non-zero thresholds indicates that the entropy-friendly variant of SGD is producing a better implicit variational distribution.

5.4 Implementation details

To compute Hessian-vector products in our models, we used autograd, a reverse-mode automatic differentiation package for Python capable of arbitrary-order derivatives.

Code for all experiments in this paper is available at `github.com/HIPS/maxwells-daemon`.

6 Limitations

In practice, the marginal likelihood estimate we present might not be useful for several reasons. First, using only a single sample to estimate both the expected likelihood as well as the entropy of an entire distribution will necessarily have high variance under some circumstances. These problems could conceivably be addressed by ensembling, which has an interpretation as taking multiple exact independent samples from the implicit approximate posterior.

Second, as parameters converge, their entropy estimate (and true entropy) will continue to decrease indefinitely, making the marginal likelihood arbitrarily small. However, in practice there is usually a limit to the degree of overfitting possible. This raises the question: when are marginal likelihoods a good guide to predictive accuracy? Presumably the marginal likelihood is more likely to be correlated with predictive performance when the approximate posterior has moderate amounts of entropy. In section 4 we modified SGD to be less prone to produce regions of pathologically low entropy, but a more satisfactory solution is probably possible.

Third, if the model includes a large number of parameters

that do not affect the predictive likelihood, but which are still affected by a regularizer, their convergence will penalize the marginal likelihood estimate even though these parameters do not affect test set performance. This is why in section 3.2 we recommend optimizing only the log-likelihood, and incorporating the regularizer directly into the initialization procedure. More generally however, entropy could be underestimated if a large group of parameters are initially constrained by the data, but are later "turned off" by some other parameters in the model.

Finally, how viable is optimization as an inference method? Standard variational methods find the best approximation in some class, but SGD doesn't even try to produce a good approximate posterior, other than by seeking the modes. Indeed, Figure 1 shows that the distribution implied by SGD collapses to a small portion of the true posterior early on, and mainly continues to shrink as optimization proceeds. However, the point of early stopping is not that the intermediate distributions are particularly good approximations, but simply that they are better than the point masses that occur when optimization has converged.

7 Related work

Estimators for early stopping Stein's unbiased risk estimator (SURE) (Stein, 1981) provides an unbiased estimate of generalization performance under broad conditions, and can be used to construct a stopping rule. Raskutti et al. (2014) derived a SURE estimate for SGD in a regression setting. Interestingly, this estimator depends on the 'shrinkage matrix' $\prod_{t=0}^{T}(\mathbf{I} - \alpha_t H_T)$, which is just the Jacobian of the entire SGD procedure along a particular path. However, this estimator depends on an estimate of the noise variance, and is restricted to the i.i.d. regression setting. It's not clear if this stopping rule could also be used to select other training parameters or model hyperparameters.

Reversible learning Optimization is an intrinsically information-destroying process, since a (good) optimization procedure maps any initial starting point to one or a few final optima. We can quantify this loss of information by asking how many bits must be stored in order to reverse the optimization, as in Maclaurin et al. (2015). We can think of the number of bits needed to exactly reverse the optimization procedure as the average number of bits 'learned' during the optimization.

From this perspective, stopping before optimization converges can be seen as a way to limit the number of bits we try to learn about the parameters from the data. This is a reasonable strategy, since we don't expect to be able to learn more than a finite number of bits from a finite dataset. This is also an example of reducing the hypothesis space to improve generalization.

MCMC for variational inference Our method can be seen as a special case of Salimans et al. (2014), who showed that any set of stochastic dynamics, even those not satisfying detailed balance, can be used to implicitly define a variational distribution. However, to provide a tight variational bound, one needs to estimate the entropy of the resulting implicit distribution. Salimans et al. (2014) do this by defining an inverse model which estimates backwards transition probabilities, and then optimizes this model in an outer loop. In contrast, our dynamics are deterministic, and our estimate of the entropy has a simple fixed form.

Bayesian neural networks Variational inference has been performed in Bayesian neural-network models (Graves, 2011; Hensman & Lawrence, 2014; Hernández-Lobato & Adams, 2015). Kingma & Welling (2014) show how neural networks having unknown weights can be reformulated as neural networks having known weights but stochastic hidden units, and exploit this connection to preform efficient gradient-based inference in Bayesian neural networks.

Black-box stochastic variational inference Kucukelbir et al. (2014) introduce a general scheme for variational inference using only the gradients of the log-likelihood of a model. However, they constrain their variational approximation to be Gaussian, as opposed to our free-form variational distribution.

SGD as an estimator Hardt et al. (2015) give theoretical results showing that the smaller the number of training epochs, the better the generalization performance of models trained using SGD. Toulis et al. (2015) examine the properties of SGD as an estimator, and show that a variant that averages parameter updates has improved statistical efficiency.

8 Future work and extensions

Optimization with momentum One obvious extension would be to design an entropy estimator of momentum-based optimizers such as stochastic gradient descent with momentum, or refinements such as Adam (Kingma & Ba, 2014). However, it is difficult to track the entropy change during the updates to the momentum variables.

Gradient-based hyperparameter optimization Optimizing marginal likelihood rather than training loss lets us choose both training and regularization parameters without using a validation set. However, optimizing more than a few hyperparameters is difficult without gradients. Following Domke (2012) and Maclaurin et al. (2015), we could compute exact gradients of the variational lower bound with respect to all variational parameters using reverse-mode differentiation through SGD. Chaining gradi-

ents through SGD would allow one to set all hyperparameters using gradient-based optimization without the need for a validation set.

Stochastic dynamics One possible method to deal with over-zealous reduction in entropy by SGD would be to add noise to the dynamics. In the case of Gaussian noise, we would recover Langevin dynamics (Neal, 2011). However, estimating the entropy is more difficult in this case. Welling & Teh (2011) introduced stochastic gradient Langevin dynamics for doing inference with minibatches, but do not track the entropy of the implicit distribution.

More generally, we are free to design optimization algorithms that do a better job of producing samples from the true posterior, as long as we can track their entropy. The gradient-thresholding method proposed in this paper is a simple first example of a refinement to SGD that maintains a tractable entropy estimate while improving the quality of the intermediate distributions.

9 Conclusion

Most regularization methods have an interpretation as approximate inference in some Bayesian model. We give such an interpretation for early stopping and ensembling: as sampling from a nonparametric approximation to the intractable posterior.

This interpretation leads naturally to a variational lower bound on the marginal likelihood. We also gave an unbiased estimate of this lower bound by approximately tracking the entropy loss at each step of optimization. Our estimator is compatible with using data minibatches, and scales linearly with the number of parameters, making it suitable for large-scale problems. This inexpensive calculation turns standard gradient descent into an inference algorithm. Our method can be used to choose model and training hyperparameters even when a validation set is not available.

References

Bai, Zhaojun, Fahey, Gark, and Golub, Gene. Some large-scale matrix computation problems. *Journal of Computational and Applied Mathematics*, 74(1):71–89, 1996.

Domke, Justin. Generic methods for optimization-based modeling. In *International Conference on Artificial Intelligence and Statistics*, pp. 318–326, 2012.

Graves, Alex. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pp. 2348–2356, 2011.

Hardt, Moritz, Recht, Benjamin, and Singer, Yoram. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*, 2015.

Hensman, James and Lawrence, Neil D. Nested variational compression in deep Gaussian processes. *arXiv preprint arXiv:1412.1370*, 2014.

Hernández-Lobato, José Miguel and Adams, Ryan P. Probabilistic backpropagation for scalable learning of bayesian neural networks. *Arxiv preprint arXiv:1502.05336*, 2015.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, Diederik and Welling, Max. Efficient gradient-based inference through transformations between bayes nets and neural nets. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1782–1790, 2014.

Kucukelbir, Alp, Ranganath, Rajesh, Gelman, Andrew, and Blei, David. Fully automatic variational inference of differentiable probability models. In *NIPS Workshop on Probabilistic Programming*, 2014.

MacKay, David JC. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3): 448–472, 1992.

Maclaurin, Dougal, Duvenaud, David, and Adams, Ryan P. Gradient-based hyperparameter optimization through reversible learning. *Arxiv preprint arXiv:1502.03492*, 2015.

Neal, Radford M. MCMC using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2, 2011.

Pearlmutter, Barak A. Fast exact multiplication by the Hessian. *Neural computation*, 6(1):147–160, 1994.

Raskutti, Garvesh, Wainwright, Martin J., and Yu, Bin. Early stopping and non-parametric regression: an optimal data-dependent stopping rule. *The Journal of Machine Learning Research*, 15(1):335–366, 2014.

Salimans, Tim, Kingma, Diederik P., and Welling, Max. Markov chain Monte Carlo and variational inference: Bridging the gap. *arXiv preprint arXiv:1410.6460*, 2014.

Stein, Charles M. Estimation of the mean of a multivariate normal distribution. *The Annals of Statistics*, 9(6):1135–1151, 1981.

Toulis, Panos, Tran, Dustin, and Airoldi, Edoardo M. Stability and optimality in stochastic gradient descent. *arXiv preprint arXiv:1505.02417*, 2015.

Wainwright, Martin J and Jordan, Michael I. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

Welling, Max and Teh, Yee Whye. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 681–688, 2011.