

A fast natural Newton method

Nicolas Le Roux and Andrew Fitzgibbon

Microsoft Research Cambridge

{nicolasl,awf}@microsoft.com

Introduction

Machine learning often looks like optimization: write down the likelihood of some training data under some model and find the model parameters which maximize that likelihood, or which minimize some divergence between the model and the data. In this context, conventional wisdom is that one should find in the optimization literature the state of the art optimizer for one’s problem and use it.

Furthermore, many machine learning objective functions are smooth in the optimization sense, so second-order optimizers are the tools of choice. And indeed, comparing second order methods to first order ones shows significant improvements in learning speed.

However, recent research (Le Roux *et al.*, 2008) has shown that, by paying attention to the special structure of machine learning problems (viewing the gradient obtained as a noisy estimate of the true gradient of the function we are really interested in), one can obtain faster convergence than first order gradient descent methods without using the Hessian. We investigate whether this improvement is due to the similarity of these methods to approximate second-order methods or, if this is not the case, if we can combine these improvements with those obtained when using the information contained in the Hessian.

TONGA: taking uncertainty into account

The argument of Le Roux *et al.* (2008) to account for uncertainty is as follows: with f the generalization error and \hat{f} the training error, we write the gradient of f as

$$g = \frac{df}{d\theta} = \int_x \frac{\partial \mathcal{L}(\theta, x)}{\partial \theta} p(x) dx \quad (1)$$

and the gradient of \hat{f} as

$$\hat{g} = \frac{d\hat{f}}{d\theta} = \frac{1}{n} \sum_i \frac{\partial \mathcal{L}(\theta, x_i)}{\partial \theta}. \quad (2)$$

We may think of g as the “true” gradient of f , and of \hat{g} as an “empirical” gradient of f , which we view as the mean of a set of samples drawn from a distribution with true mean g . If the training samples are iid, and given the “large-scale” assumption of large n , we can use the central-limit theorem, which yields

$$\hat{g} | g \sim \mathcal{N}\left(g, \frac{C}{n}\right) \quad (3)$$

where C is the covariance matrix of the gradients. Relaxing eq. 3 to finite n and defining an isotropic Gaussian prior over g :

$$g \sim \mathcal{N}(0, \sigma^2 I), \quad (4)$$

yields the following posterior distribution over the true gradient:

$$g | \hat{g} \sim \mathcal{N}\left(\left[I + \frac{C}{n\sigma^2}\right]^{-1} \hat{g}, [nC^{-1} + \sigma^{-2}I]^{-1}\right). \quad (5)$$

We see that the direction maximizing the expected gain in generalization error is the mean of this Gaussian, i.e.

$$\Delta\theta \propto \left[I + \frac{\hat{C}}{n\sigma^2}\right]^{-1} \hat{g}, \quad (6)$$

reminiscent of the natural gradient (Amari, 1998).

Le Roux *et al.* (2008) report large speedups on various neural network problems. Intuitively, one can understand why using the covariance may be beneficial. Indeed, it seems wasteful to compute the gradient over many data points and keep only their mean.

Combining Newton method and natural gradient

Building upon this work, we show how to use Hessian information within the natural gradient. We assume a quadratic cost:

$$f(\theta) = \int_x \mathcal{L}(\theta, x)p(x) dx = \int_x \frac{1}{2}(\theta - x)^T H(\theta - x)p(x) dx = \frac{1}{2}(\theta - \theta^*)^T H(\theta - \theta^*) \quad (7)$$

with $\theta^* = \int_x xp(x) dx$. The derivative of this cost is:

$$g(\theta) = \int_x \frac{\partial \mathcal{L}(\theta, x)}{\partial \theta} p(x) dx = H(\theta - \theta^*). \quad (8)$$

We can see that, in the context of a quadratic function, the isotropic prior over g proposed in eq. 4 is erroneous as g is clearly influenced by H . We shall rather consider an isotropic Gaussian prior on the quantity $\theta - \theta^*$ as we do not have any information about the position of θ relative to θ^* . The resulting prior distribution over g is

$$g \sim \mathcal{N}(0, \sigma^2 H^2). \quad (9)$$

As in TONGA, we will use the central-limit theorem to get

$$\hat{g} | g \sim \mathcal{N}\left(g, \frac{C}{n}\right) \quad (10)$$

where C is the true covariance of the gradients. Therefore, the posterior distribution over g is

$$g | \hat{g} \sim \mathcal{N}\left(\left[I + \frac{CH^{-2}}{n\sigma^2}\right] \hat{g}, \left[\frac{H^{-2}}{\sigma^2} + nC^{-1}\right]^{-1}\right) \quad (11)$$

Since the function is locally quadratic, we wish to move in the direction $H^{-1}g$. This direction follows a Gaussian distribution with mean

$$\left[I + \frac{H^{-1}CH^{-1}}{n\sigma^2}\right]^{-1} H^{-1}\hat{g} \quad (12)$$

which is the direction of maximum expected gain. Since eq. 12 appears complicated, we shall explain it. Let us first replace the true covariance matrix C by its empirical version \hat{C} . If we define the Newton directions $d_i = H^{-1}g_i$, since \hat{C} is the covariance matrix of the gradients g_i , $H^{-1}\hat{C}H^{-1} = \hat{D}$ is the covariance matrix of the d_i 's. Similarly, we have $H^{-1}\hat{g} = \hat{d}$ the average of the Newton directions. We can therefore rewrite

$$\left[I + \frac{H^{-1}CH^{-1}}{n\sigma^2}\right]^{-1} H^{-1}\hat{g} = \left[I + \frac{\hat{D}}{n\sigma^2}\right]^{-1} \hat{d}, \quad (13)$$

which is exactly eq. 6, but on the Newton directions. This is great news as it means that one may choose his favorite second-order gradient descent method to compute the (approximate) Newton directions, and then his favorite natural gradient algorithm to apply to these Newton directions, to yield an algorithm naturally combining the advantages of both methods.

Experiments

We performed experiments on datasets from the Pascal Large Scale Challenge, using SGD-QN (Bordes *et al.*, 2009) as our approximate Newton method and an exponentially moving diagonal covariance matrix for our natural gradient algorithm. We coin our algorithm **Natural-Newton**.

The results on these three datasets are typical of the ones we obtained in general:

- either using the covariance matrix brought a gain in convergence speed, whether applied to a first or second-order method;
- either using the covariance proved only useful when applied to the first-order method;
- either using the covariance did not provide any gain on the convergence speed, but yielded increased stability, whether applied to a first or second-order method.

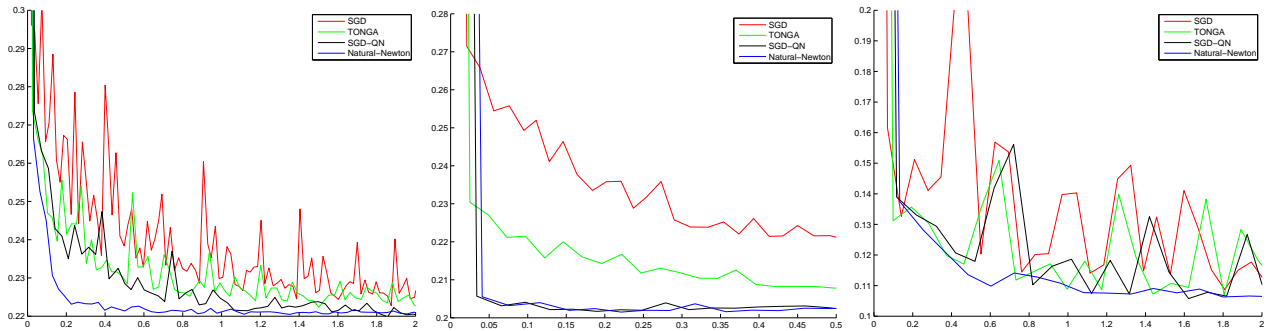


Figure 1: Test error vs. time on the Alpha dataset

Figure 2: Test error vs. time on the Gamma dataset

Figure 3: Test error vs. time on the Epsilon dataset

Conclusion

A lot of effort has been put into designing efficient online optimization algorithms, with great results. Most of these algorithms rely on some approximation to the Hessian or to the covariance matrix of the gradients. While the latter is commonly believed to be an approximation of the former, they encode very different kinds of information. Based on this, we proposed a way of combining information contained in the Hessian and in the covariance matrix of the gradients.

Experiments showed that, on most datasets, our method offered either faster convergence or increased robustness compared to the original algorithm. Furthermore, our algorithm never performed worse than the Newton algorithm it was built upon.

Moreover, our algorithm is able to use any existing second-order algorithm as base method. Therefore, while we used SGD-QN for our experiments, one may pick any algorithm best suited for a given task.

We hope to have shown two things. Firstly, the covariance matrix of the gradients is usefully viewed, not as an approximation to the Hessian, but as a source of additional information about the problem, for typical “machine learning” objective functions. Secondly, it is possible with little extra effort to use this information in addition to that provided by the Hessian matrix, yielding in some cases faster or more robust convergence.

References

- Amari, S. (1998). Natural gradient works efficiently in learning. *Neural Computation*, **10**(2), 251–276.
- Bordes, A., Bottou, L., and Gallinari, P. (2009). SGD-QN: Careful quasi-newton stochastic gradient descent. *Journal of Machine Learning Research*, **10**, 1737–1754.
- Le Roux, N., Manzagol, P.-A., and Bengio, Y. (2008). Topmoumoute online natural gradient algorithm. In *Advances in Neural Information Processing Systems 20*, pages 849–856. MIT Press, Cambridge, MA.