Information Geometry and Natural Gradients

Nathan Ratliff

Nov 19, 2013

Abstract

This document reviews some of the basic concepts behind natural gradients. We start by introducing basic information theoretic concepts such as optimal codes, entropy, and the KL-divergence. We then demonstrate how the choice of metric on perturbations can significantly affect the performance of gradient descent algorithms. Within that discussion, we review the Method of Lagrange Multipliers for equality constrained optimization as well as intuitive interpretations of the action of positive definite matrices and their inverses and how that relates to their role in generalized gradient descent updates. Finally, we review how the second-order Taylor expansion of the KL-divergence between a distribution and a slightly perturbed version of that distribution leads to the notion of the Fisher Information as a natural metric on the manifold of probability distributions.

1 Information Theory and the KL-divergence

Let $p(x)$ be some probability distribution over a discrete set of objects X. If we want to encode those object and transmit them over a wire, then there are trade-offs we need to consider. First, what do we mean when we say we want to "encode" the objects? If I'm on one side of a wire, and I'm able to transmit strings of bits (e.g. 00011101001) across the line, then I can tell the receiver in advance I'm going to, for instance, use the string 101 to represent object $x_1 \in X$, or the string 110 to represent object $x_2 \in X$, etc. If both the sender and receiver agree on a particular convention, then efficient communication is possible. That "convention" for assigning strings to objects, is what we refer to as a *code*. Note that the set X may contain infinitely many objects, so using strings of different lengths is important for both feasibility of the code and, as we'll discuss below, for efficiency.

On the other side, once I've sent a set of bit strings across the wire, if the receiver knows the code, she can then "decode" the string on the other end. For instance, if she receives the string 110, then since we've agreed in advance that 110 represents object x_1, then she can confidently say that the first object to come off the wire is x_1. A question central to Information Theory (Cover & Thomas, 2006) is, "What code is most efficient." I.e. how do we assign bit strings to objects in a way that minimizes the expected length of the transmitted string?

For us to make sense of this question, we need to know something about the distribution of objects $p(x)$ being sent. Formalizing that distribution gives the phrase "the expected length of the transmitted string" an exact meaning:

$$E_p[l_c(x)] = \sum_x p(x) l_c(x),$$

where $l_c(x)$ denotes the length of the bit string assigned to x under code c.

Since some bit strings are shorter than others, intuitively it behooves us to choose a code that reserves the shortest strings for frequently sent objects, and assigns longer strings to infrequently sent objects. It turns out that if we literally do that by ordering the strings by length from shortest to longest and ordering the objects by send-probability from most probable to least probable, then the code that simply matches those up one-to-one (with the most probable object getting the shortest string, the second most probable object getting the second shortest string, etc.) is the optimal code!

Moreover, one can show that the approximate number of bits assigned to object x under this optimal code c^* is

$$l_{c^*}(x) = \log \frac{1}{p(x)} = -\log p(x).$$

The expected number of bits under the optimal code is, therefore,

$$E_p[l_{c^*}(x)] = \underbrace{-\sum_x p(x) \log p(x).}_{H[p]}$$

This quantity, which we denote $H[p]$, is known as the *entropy* of the distribution.

The entropy represents the degree uncertainty in the distribution (in units of bits). If we know exactly with probability 1 that a specific object $\widetilde{x}$ will always be what's sent across the wire, then the optimal code is to send that one object with the shortest bit string. The shortest bit string has no bits at all (except a constant number of boundary bits that we ignore in these theoretical analyses) since we don't have to distinguish between anything on the receiving end. Our mathematical bit-length estimate reflects that intuition since $l_{c^*}(\widetilde{x}) = -\log p(\widetilde{x}) = 0$ when $p(\widetilde{x}) = 1$. An entropy of 0 is the smallest it can be when we consider only discrete distributions.

On the other hand, when we really don't know what the user's going to send over the wire (i.e. $p(x)$ is flatter), then there's nothing we can do but assign some relatively long bit strings to objects that may actually be rather frequent. The expected number of bits for even the optimal code (given by the entropy $H(x)$), in this case, will be a much larger number, significantly away from zero. In general, the more uncertain the distribution, the more bits it takes to optimally encode what we want to transmit. $H(x)$ is therefore a measure of uncertainty in the distribution.

Now, suppose we mistakenly use the wrong distribution $q(x)$ to create the code for a set of objects X that actually will have distribution $p(x)$ when it comes time to transmit it. By that we mean we assign objects that have high probability under q the shortest bit strings rather than those that have high probability under p. Since the ranking of object is increasingly wrong the more p and q differ, the more suboptimal our code is going to be. We can explicitly measure this suboptimality by calculating the expected difference between the number of bits we get from the optimal code and the number of bits we end up with under the suboptimal code. Essentially we ask, "How many bits did we waste by encoding the objects using distribution q rather than the correct distribution p?"

Mathematically, for p, each object x is assigned a string with $\log \frac{1}{p(x)}$ bits, where as for q the object is assigned a string with $\log \frac{1}{q(x)}$ bits. The expected difference between those two at transmission time is

$$E_p\left[\log \frac{1}{q(x)} - \log \frac{1}{p(x)}\right] = \underbrace{\sum_x p(x) \log \frac{p(x)}{q(x)}}_{\mathrm{KL}(p \;\|\; q)}.$$

That last expression is what we typically term the KL-divergence between p and q. It represents the degree of suboptimality in erroneously encoding a set of objects using distribution q when you should have used distribution p. Since we can never do better than encoding the distribution in terms of p, which would assign a string of length $l_{c^*}(x) = \log \frac{1}{p(x)}$ bits to each object x, this "divergence" measure is never negative. And more than that, it's strictly positive for all $q \neq p$ and exactly zero for $q = p$.

The KL-divergence is used throughout Information Theory and Machine Learning (Bishop, 2007), and is the starting point for discussions about Information Geometry as we discuss in Section 3.

2 Gradient descent update rules

This section elaborates more on the derivation of gradient descent update rules under varying metrics. Section 2.1 gives an intuitive and more easily understood derivation, and Section 2.2 grounds the result using the Theory of Lagrange Multipliers. The first section is more important; the second is there mainly for completeness for those who are interested.

2.1 A basic derivation

Suppose we're optimizing a function $f(\theta)$. Then we can derive a generalized gradient descent rule for updating a point θ_t by writing down a constrained optimization problem that says we want to minimize a first-order approximation to our objective around θ_t subject to the constraint that our step $\delta\theta = \theta - \theta_t$, what I'll sometimes call the *perturbation*, shouldn't be too big:

$$\begin{aligned}\theta_{t+1} =& \arg\min_\theta f(\theta_t) + \nabla f(\theta_t)^T(\theta - \theta_t) \\ &\text{s.t.} \frac{1}{2}\|\theta - \theta_t\|_A^2 = \epsilon^2.\end{aligned} \tag{1}$$

The generalized A-weighted norm is defined as $\|x\|_A^2 = x^T A x$; we discuss its properties in relation to how it affects the derived gradient update below. Technically, the constraint should really be that the perturbation $\delta\theta = \theta - \theta_t$ is less than or equal to ϵ (since we just want the perturbation to be small, not necessarily an exact fixed size ϵ), but for our purposes it's sufficient to deal strictly with equality.[1]

We can solve this problem by using Lagrange multipliers. If you've never seen Lagrange multipliers before, suffice it to say that the problem can equivalently be rewritten as an unconstrained problem of the form

$$\theta_{t+1} = \arg\min_\theta f(\theta_t) + \nabla f(\theta_t)^T(\theta - \theta_t) + \frac{\lambda}{2}\|\theta - \theta_t\|_A^2, \tag{2}$$

where there is some relationship between λ and ϵ that we don't really care about because we typically need to tune those anyway for a given problem. Section 2.2 details the Method of Lagrange Multipliers for those who are interested.

Solving Equation 2 by setting its gradient to zero gives

$$\theta_{t+1} = \theta_t - \frac{1}{\lambda}A^{-1}\nabla f(\theta_t). \tag{3}$$

In other words, that optimization entirely characterizes gradient descent!

The weight matrix A that defines the norm on perturbations to θ plays a significant role in defining the search direction. A must be positive definite to define a valid norm, and as such we can think of the operation of such a matrix as a stretching or squishing along particular dimensions of some orthogonal basis in the space.[2] Intuitively, if A, through it's interaction with the norm $\|\delta\theta\|_A^2 = \delta\theta^T A \delta\theta$, penalizes $\delta\theta$ strongly along certain directions and less along

[1] If we have inequality constraints here, the optimal point is either going to be on the surface of that constraint (i.e. it'll hold at equality), or the constraint is irrelevant because the minimizer of the linearization is already in the interior. If it's the latter case, since the approximation is linear, it must be that $\nabla f(\theta_t) = 0$, in which case, we've already arrived at the minimum (or at least a fixed point) and the overall optimization problem is solved (or, in the case of a fixed point, further gradient steps won't help). In all relevant cases, the gradient is never exactly zero, so we can safely assume that the step will be exactly of length ϵ in this derivation.

[2] Mathematically, this is represented explicitly in the diagonalization of the matrix $A = UDU^T$, where U is an orthogonal matrix whose columns consist of the n (mutually orthonormal) Eigenvectors of A, and D is a diagonal matrix whose diagonal entries are the corresponding Eigenvalues. This diagonalization always exists with strictly positive Eigenvalues because the matrix is positive definite. Another way of writing that, which makes the above interpretation explicit is $A = \sum_{i=1}^n \lambda_i e_i e_i^T$, where λ_i and e_i are the ith Eigenvalue and

other directions, the action of A^{-1} on the gradient in Equation 3 does exactly the opposite. It shrinks components along the directions of heavy penalization so that the update has little effect in those direction. Or if A doesn't penalize the perturbation much along a particular dimension, then A^{-1} will either keep those dimensions of the gradient untouched or potentially even amplify those directions to emphasize their effect during the update.

Section 3 below shows that we can use the KL-divergence between a distribution and a perturbed version of that distribution in the above framework by taking its second-order Taylor expansion.

2.2 Full derivation in terms of Lagrange Multipliers

This section briefly reviews how one might solve Equation 1 directly using the Method of Lagrange Multipliers. I won't go into detail about why this procedure works, but at least I'll lay out the procedure itself which is generally applicable in a number of settings where analytic equality constrained optimization is needed.

The procedure for handling equality constraints in an optimization, such the constraint $\frac{1}{2}\|\theta-\theta_t\|_A^2 = \epsilon^2$ in Equation 1, is to place them up into the objective with a particular unknown weight λ to form what's called a Lagrangian. In this case, we get

$$\mathcal{L}(x,\lambda) = f(\theta_t) + \nabla f(\theta_t)^T(\theta-\theta_t) + \lambda\left(\frac{1}{2}\|\theta-\theta_t\|_A^2 - \epsilon^2\right). \tag{4}$$

The Theory of Lagrange Multipliers tells us that the solution to the constrained optimization problem is necessarily described by the conditions

$$\begin{cases} \nabla_x \mathcal{L}(x,\lambda) = 0 \\ \nabla_\lambda \mathcal{L}(x,\lambda) = 0. \end{cases}$$

A full discussion of these "optimality conditions" is beyond the scope of these notes, but a reasonable reference for a first pass is given in the Wikipedia page on Lagrange Multipliers.[3]

In this case, using the Lagrangian in Equation 4 we get the following system of two equations

$$\begin{cases} \nabla f(\theta_t) + \lambda(\theta-\theta_t) = 0 \\ \|\theta-\theta_t\|_A^2 - \epsilon^2 = 0. \end{cases}$$

The second equation simply restates the original constraint, but the two equations in combination allow us to first explicitly solve for λ in terms of ϵ

$$\lambda = \frac{1}{\epsilon}\|\nabla f(\theta_t)\|_A$$

Eigenvector, respectively. Each matrix $e_i e_i^T$ is a projection operator that projects a vector x onto the Eigenvector e_i (its action is $(e_i e_i^T)x = (e_i^T x)e_i$). Scaling that term by λ_i, then stretches the component of x in the direction of e_i by a factor of λ_i. The matrix A therefore stretches or squishes a vector along the orthogonal directions e_i by the factors λ_i.

[3] http://en.wikipedia.org/wiki/Lagrange_multiplier

and then explicitly solve the problem by solving the first equation for θ in terms of λ and then plugging in our expression for λ in terms of ϵ:

$$\begin{aligned}\theta^* &= \theta_t - \frac{1}{\lambda}A^{-1}\nabla f(\theta_t) = \theta_t - \epsilon A^{-1}\left(\frac{\nabla f(\theta_t)}{\|\nabla f(\theta_t)\|_A}\right) \\ &= \theta_t - \epsilon A^{-1}\widehat{\nabla f}(\theta_t).\end{aligned}$$

Relating this back to the above expression in Equation 3, and noting that for a given λ, the condition $\nabla_x \mathcal{L}(x,\lambda) = 0$ completely characterizes the solution to the penalized problem we described in Equation 2, we can say that the constrained problem of Equation 1 and the unconstrained (penalized) problem of Equation 2 are equivalent under the relationship $\lambda = \|\nabla f(\theta_t)\|/\epsilon$. Intuitively, choosing a ball constraint enforces that the length of the step is precisely ϵ, whereas choosing a particular weight λ in Equation 2 defines a step that depends on the length of the gradient $\nabla f(\theta_t)$ (unless, of course, we choose λ to be $\lambda = \frac{1}{\epsilon}\|\nabla f(\theta_t)\|_A$).

3 KL-divergence between perturbed distributions

Let $p(x;\theta)$ be some family of probability distributions over x parameterized by a vector of real numbers θ. We're interested in knowing how much the distribution changes when we perturb the parameter vector from a fixed θ_t to some new value $\theta_t + \delta\theta$. As a measure of change in probability distribution, we can use the KL-divergence measure we introduced in Section 1. Specifically, we want to measure $\mathrm{KL}(\ p(x;\theta_t)\ \|\ p(x;\theta_t+\delta\theta)\ )$, but we want to write it in a form amenable to the gradient-based update formulation presented in Section 2. We can do this by taking it's second-order Taylor expansion around θ_t. During the derivation, we'll find that a lot of terms in the expansion disappear leaving us with a very simple expression that's perfect for our purposes.

Looking first at the full KL-divergence, we see that the term we want to expand using a second-order Taylor approximation is $\log p(x;\theta_t+\delta\theta)$:

$$\begin{aligned}&\mathrm{KL}(\ p(x;\theta_t)\ \|\ p(x;\theta_t+\delta\theta)\ ) \\ &\quad= \int p(x;\theta_t)\log\left(\frac{p(x;\theta_t)}{p(x;\theta_t+\delta\theta)}\right)dx \\ &\quad= \int p(x;\theta_t)\log p(x;\theta_t)dx - \int p(x;\theta_t)\log p(x;\theta_t+\delta\theta)dx.\end{aligned}$$

The second-order Taylor series expansion generically is

$$f(\theta) \approx f(\theta_t) + \nabla f(\theta_t)^T\delta\theta + \frac{1}{2}\delta\theta^T\left(\ \nabla^2 f(\theta_t)\ \right)\delta\theta,$$

where $\theta = \theta_t + \delta\theta$, or equivalently $\delta\theta = \theta - \theta_t$. Applying that expansion to the pertinent term in the KL-divergence expression, we get

$$\log p(x;\theta_t+\delta\theta) \approx \log p(x;\theta_t) + \left(\frac{\nabla p(x;\theta_t)}{p(x;\theta_t)}\right)^T\delta\theta + \frac{1}{2}\delta\theta^T\left(\ \nabla^2\log p(x;\theta_t)\ \right)\delta\theta.$$

Plugging this second-order Taylor expansion back into the above expression for the KL-divergence gives

$$
\begin{aligned}
&\mathrm{KL}(\ p(x;\theta_t) \ \|\ p(x;\theta_t+\delta\theta)\ ) \qquad (5)\\
&\quad\approx \int p(x;\theta_t)\log p(x;\theta_t)dx\\
&\qquad - \int p(x;\theta_t)\left(\log p(x;\theta_t) + \left(\frac{\nabla p(x;\theta_t)}{p(x;\theta_t)}\right)^T \delta\theta + \frac{1}{2}\delta\theta^T\left(\ \nabla^2 \log p(x;\theta_t)\ \right)\delta\theta\right)dx\\
&\quad= \underbrace{\int p(x;\theta_t)\log\frac{p(x;\theta_t)}{p(x;\theta_t)}dx}_{=0} \ - \ \underbrace{\left(\int \nabla p(x;\theta_t)dx\right)^T\delta\theta}_{=0}\\
&\qquad\qquad\qquad - \ \frac{1}{2}\delta\theta^T\left(\int p(x;\theta_t)\nabla^2\log p(x;\theta_t)\right)\delta\theta.
\end{aligned}
$$

As indicated, the first term, since it's the KL-divergence between a distribution and itself, is zero, and the second term, because of the relation

$$\int \nabla p(x;\theta_t)dx = \nabla\int p(x;\theta_t)dx = \nabla 1 = 0$$

is also zero.

Probably the most challenging part of the derivation, depending on your fluency in vector calculus, is simply calculating the Hessian of $\log p(x;\theta_t)$. It's sometimes easiest to write out complicated vector calculus computations out in terms of their individual partial derivatives first and then only later recognize how the resulting expression can be more compactly written in matrix form without directly resorting to vector calculus formulas that can be easy to mess up. We'll use that technique here to derive the Hessian:

$$
\begin{aligned}
&\frac{\partial^2}{\partial\theta_t^{(i)}\partial\theta_t^{(j)}}\left[\ \log p(x;\theta_t)\ \right]\\
&\qquad= \frac{\partial}{\partial\theta_t^{(i)}}\left(\frac{\frac{\partial}{\partial\theta_t^{(j)}}p(x;\theta_t)}{p(x;\theta_t)}\right)\\
&\qquad= \frac{p(x;\theta_t)\frac{\partial^2}{\partial\theta_t^{(i)}\partial\theta_t^{(j)}}p(x;\theta_t) - \frac{\partial}{\partial\theta_t^{(i)}}p(x;\theta_t)\frac{\partial}{\partial\theta_t^{(j)}}p(x;\theta_t)}{p(x;\theta_t)^2}\\
&\qquad= \frac{1}{p(x;\theta_t)}\frac{\partial^2}{\partial\theta_t^{(i)}\partial\theta_t^{(j)}}p(x;\theta_t) - \left(\frac{\frac{\partial}{\partial\theta_t^{(i)}}p(x;\theta_t)}{p(x;\theta_t)}\right)\left(\frac{\frac{\partial}{\partial\theta_t^{(j)}}p(x;\theta_t)}{p(x;\theta_t)}\right).
\end{aligned}
$$

The first term is an element of the Hessian $\nabla^2 p(x;\theta_t)$ weighted by a factor $\frac{1}{p(x;\theta_t)}$, and the second term is an element of the outer product between $\nabla\log p(x;\theta_t)$ and itself. So in matrix form, this becomes

$$\nabla^2\log p(x;\theta_t) = \frac{1}{p(x;\theta_t)}\nabla^2 p(x;\theta_t) - \nabla\log p(x;\theta_t)\nabla\log p(x;\theta_t)^T.$$

Finally, plugging this expression for the Hessian back into our latest KL-divergence expression in Equation 5 we get

$$\begin{aligned}
&\mathrm{KL}(\ p(x;\theta_t)\ \|\ p(x;\theta_t+\delta\theta)\ ) \\
&\quad\approx -\frac{1}{2}\delta\theta^T\left(\int p(x;\theta_t)\nabla^2\log p(x;\theta_t)dx\right)\delta\theta \\
&\quad= -\frac{1}{2}\delta\theta^T\underbrace{\left(\int \nabla^2 p(x;\theta_t)dx\right)}_{=0}\delta\theta \\
&\qquad\quad+\frac{1}{2}\delta\theta^T\underbrace{\left(\int p(x;\theta_t)\left[\ \nabla\log p(x;\theta_t)\nabla\log p(x;\theta_t)^T\right]dx\right)}_{G(\theta_t)}\delta\theta.
\end{aligned}$$

As indicated, the first term in this expression is zero for the same reason as before:

$$\int \nabla^2 p(x;\theta_t)dx = \nabla^2\int p(x;\theta_t)dx = \nabla^2 1 = 0.$$

The central matrix here $G(\theta_t)$ is known as the Fisher Information matrix and can has been thoroughly studied within the field of Information Geometry (Amari & Nagaoka, 2000) as the natural Riemannian structure on a manifold of probability distributions. As such it defines a natural norm on perturbations to probability distributions, which was our original motivation for examining the second-order Taylor expansion of the KL-divergence in the first place.

Given this analysis, setting $A = G(\theta_t)$ in the above generalized gradient update expression of Equation 3 gives an update rule that explicitly measures the size of perturbations to the parameters based on the extent to which they change the probability distribution as measured by KL-divergence. In full, the natural gradient update for optimizing a function $f(\theta)$ is

$$\theta_{t+1} = \theta_t - \eta_t G(\theta_t)^{-1}\nabla f(\theta_t).$$

Here we use η_t in place of $\frac{1}{\lambda}$ to emphasize that the step size parameter may change at each iteration.

References

Amari, S. and Nagaoka, H. *Methods of Information Geometry*, volume 191 of *Translations of Mathematical monographs.* Oxford University Press, 2000.

Bishop, Christopher M. *Pattern Recognition and Machine Learning.* Springer, 2007.

Cover, Thomas M. and Thomas, Joy A. *Elements of information theory.* Wiley-Interscience, 2nd edition edition, 2006.