TTIC 31230, Fundamentals of Deep Learning

David McAllester, Winter 2019

The Arbitraryness of the Gradient Direction

The Direction from the Loss Hessian

The Direction from the Gradient CoVariance

The Direction from Information Geometry

Arbitraryness of the Gradient Direction

Vector Spaces and Coordinte Systems

A vector space consists of field of scalars (typically the real or complex numbers), a set of "vectors", and the operations of scalar multiplication and vector addition.

A vector space has no "canonical" or "natural" coordinate system.

A coordinate system is defined by a choice of basis vectors b_1, ..., b_N that are linearly independent and span the space.

A basis defines coordinates $x[i]$ for each vector x.

$$x = x[1]b_1 + \cdots + x[N]b_N$$

Orthogonality is Coordinate-Relative

In two dimensions the vectors represented by $(1, 0)$ and $(0, 1)$ need not be orthogonal.

$(1, 0)$ represents b_1 and $(0, 1)$ represents b_2.

We only require that b_1 and b_2 are independent.

Hence inner product is coordinate-relative.

Inner Products in Taylor Expansions are Coordinate-Independent

$$f(\Phi + \Delta\Phi) \approx f(\Phi) + \left[\nabla_\Phi \, f(\Phi)\right](\Delta\Phi)$$

What is a Gradient?

The gradient $\nabla_\Phi f(\Phi)$ is the change in f per change in Φ.

More formally, $\nabla_\Phi f(\Phi)$ is a linear function from $\Delta\Phi$ to Δf.

$$f(\Phi + \Delta\Phi) \approx f(\Phi) + [\nabla_\Phi f(\Phi)](\Delta\Phi)$$

Coordinate-Free Definition of the Gradient

$$f(\Phi + \Delta\Phi) \approx f(\Phi) + [\nabla_\Phi f(\Phi)](\Delta\Phi)$$

$$f(\Phi + \epsilon\Delta\Phi) \approx f(\Phi) + [\nabla_\Phi f(\Phi)](\epsilon\Delta\Phi)$$

$$\frac{f(\Phi + \epsilon\Delta\Phi) - f(\Phi)}{\epsilon} \approx [\nabla_\Phi f(\Phi)](\Delta\Phi)$$

$$[\nabla_\Phi f(\Phi)](\Delta\Phi) \doteq \lim_{\epsilon \to 0} \frac{f(\Phi + \epsilon\Delta\Phi) - f(\Phi)}{\epsilon}$$

No coordinates required.

Dual Vectors

A dual vector is a linear function from vectors to scalars.

The gradient is a dual vector.

Coordinates

We calculate

$$[\nabla_\Phi f(\Phi)]\,\Delta\Phi$$

using coordinates.

$$[\nabla_\Phi f(\Phi)]\,\Delta\Phi \quad = \quad \sum_i \left[\frac{\partial f}{\partial \Phi[i]}\right] \Delta\Phi[i]$$

But this calculation is coordinate-independent.

$$[\nabla_\Phi\ f(\Phi)]\,(\Delta\Phi) \quad \equiv \quad \lim_{\epsilon\to 0}\ \frac{f(\Phi+\epsilon\Delta\Phi)-\ f(\Phi)}{\epsilon}$$

Strange Coordinate Systems

Consider any gradient $\nabla_\Phi f(\Phi)$ at any value of Φ.

For any such situation, and any vector $\Delta\Phi$ with $[\nabla_\Phi f(\Phi)]\,\Delta\Phi > 0$, and any learning rate $\eta > 0$, there exists a coordinate system in which $\eta\Phi.\text{grad}[c] = \Delta\Phi[c]$ and hence

$$\Phi_{t+1} = \Phi_t - \Delta\Phi$$

Note that gradient decent always yields $[\nabla_\Phi f(\Phi)]\,\Delta\Phi > 0$.

Proof

Define the basis vectors $b_1, \ldots, b_N$ by

$$b_1 \doteq \frac{\Delta\Phi}{\sqrt{\eta[\nabla_\Phi f(\Phi)]\Delta\Phi}} \qquad [\nabla_\Phi f(\Phi)]\ b_i = 0 \ \text{ for } i > 1$$

In this coordinate system we have

$$\Phi = \Phi[1]b_1 + \ldots + \Phi[N]b_n$$

$$\frac{\partial f(\Phi)}{\partial \Phi[1]} = \frac{\sqrt{[\nabla_\Phi f(\Phi)]\Delta\Phi}}{\sqrt{\eta}} \qquad \frac{\partial f(\Phi)}{\partial \Phi[j]} = 0 \quad \text{for } j > 0$$

$$\sum_i \Phi.\text{grad}[i]\Phi[i]b_i = \frac{\sqrt{[\nabla_\Phi f(\Phi)]\Delta\Phi}}{\sqrt{\eta}}\, b_1 = \frac{\Delta\Phi}{\eta}$$

Manifold SGD

$$\Phi_{t+1} \;=\; \Phi_t - \eta_t H_\Phi^{-1} \hat{g}_t$$

Under what conditions does the loss value converge and any limit point of Φ_t is a stationary point?

The Hessian of Loss Direction (The Newton Direction)

Second Order Methods

Newton's Method

We can make a second order approximation to the loss function

$$f(\Phi + \Delta\Phi) \approx f(\Phi) + (\nabla_\Phi\, f(\Phi))\Delta\Phi + \frac{1}{2}\Delta\Phi^\top H \Delta\Phi$$

where H is the second derivative of f, the Hessian, equal to $\nabla_\Phi \nabla_\Phi\, f(\Phi)$.

Again, no coordinates are needed — we can define the operator ∇_Φ generally indpendent of coordinates.

$$\Delta\Phi_1^\top\; H\; \Delta\Phi_2 = (\nabla_\Phi\,((\nabla_\Phi\, f_t(\Phi)) \cdot \Delta\Phi_1)) \cdot \Delta\Phi_2$$

Newton's Method

We consider the first order expansion of the gradient.

$$\nabla_\Phi f(\Phi)\,|_{\Phi+\Delta\Phi} \approx (\nabla_\Phi f(\Phi)\,|_\Phi) + H\Delta\Phi$$

We approximate Φ^* by setting this gradient approximation to zero.

$$0 = \nabla_\Phi f(\Phi) + H\Delta\Phi$$

$$\Delta\Phi = -H^{-1}\,\nabla_\Phi f(\Phi)$$

This gives Newton's method (without coordinates)

$$\Phi \text{ -= } H^{-1}\,\nabla_\Phi f(\Phi)$$

Newton Updates

It seems safer to take smaller steps. So it is common to use

$$\Phi \;\texttt{-=}\; \eta\, H^{-1}\, \nabla_\Phi\, f(\Phi)$$

for $\eta \in (0, 1)$ where η is naturally dimensionless.

Most second order methods attempt to approximate making updates in the Newton direction.

Quasi-Newton Methods

It is often faster and more effective to approximate the Hessian.

Maintain an approximation $M \approx H^{-1}$.

Repeat:

- Φ **-=** $\eta M \nabla_\Phi f(\Phi)$ (η is often optimized in this step).
- Restimate M.

The restimation of M typically involves a finite difference

$$\left(\nabla_\Phi f(\Phi) \mid_{\Phi_{t+1}}\right) - \left(\nabla_\Phi f(\Phi) \mid_{\Phi_t}\right)$$

As a numerical approximation of $H\Delta\Phi$.

Quasi-Newton Methods

Conjugate Gradient

BFGS

Limited Memory BFGS

Issues with Quasi-Newton Methods

In SGD the gradients are random even when Φ does not change.

We cannot use

$$\left(\nabla_\Phi \, f_{t+1}(\Phi)|_{\Phi_{t+1}}\right) - \left(\nabla_\Phi \, f_t(\Phi) \, |_{\Phi_t}\right)$$

as an estimate of $H\Delta\Phi$.

The Gradient Covariance Direction

The Gradient Covariance Direction

$$\Sigma \doteq E_t \left(\hat{g}_t - g\right)\left(\hat{g}_t - g\right)^\top$$

$$\Phi_{t+1} = \Phi_t - \eta \Sigma^{-1} \nabla_\Phi \text{Loss}(\Phi, x_t, y_y)$$

In the limit $\eta \to 0$, the stationary Langevin distribution of this update is Gibbs!

$$p(\Phi) \propto e^{-\beta \mathcal{L}(\Phi)} \quad \text{with} \quad \beta \propto \frac{1}{\eta}$$

Newton Direction vs. Gradient Covariance Direction

The Newton direction is motivated by moving directly toward the nearest local optimum.

The gradient covariance direction is motivated by improving exploration of SGD viewed as an MCMC process.

Both directions are difficult (infeasible?) to compute. But see the discussion of Levenberg-Marquart below.

The Information Geometry Direction (Amari's Natural Gradient)

KL Divergence as a Geometric Distance

We consider the case where loss is determined by a probability distribution. For example $-\ln P_\Phi(y)$.

We consider a ball around Φ determined by the KL divergence between P_Φ and $P_{\Phi'}$

$$B(\Phi, \epsilon) = \{\Phi' \mid KL(P_\Phi, P_{\Phi'}) \leq \epsilon\}$$

Note that this ball is defined by the set of distributions of the form P_Φ independent of the choice of the coordinates for Φ.

Proximal Gradient Descent

Proximal gradient descent is defined by a distance measure.

$$\begin{aligned}\Phi_{t+1} &= \operatorname*{argmin}_{\Phi' \in B(\Phi,\, \epsilon)} E_{(x,y)\sim \text{Train}} \; -\ln P_{\Phi'}(y|x) \\ &= \operatorname*{argmin}_{\Phi' \in B(\Phi,\, \epsilon)} \mathcal{L}(\Phi')\end{aligned}$$

Here ϵ plays the role of the learning rate η for **total** gradient descent (stochastic proximal descent does not work).

This update is independent of the choice of coordinates for Φ.

This is Amari's "natural gradient".

Distances and Metrics

We now consider an arbitrary smooth distance function d with $d(\Phi, \Phi) = 0$ and $d(\Phi, \Phi') \geq 0$.

Assuming smoothness, $d(\Phi, \Phi + \Delta\Phi)$ is locally quadratic in $\Delta\Phi$.

$$d(\Phi, \Phi + \Delta\Phi) \approx \frac{1}{2} \Delta\Phi^\top H \Delta\Phi$$

$$H \doteq \nabla_{\Delta\Phi} \nabla_{\Delta\Phi} \; d(\Phi, \Phi + \Delta\Phi)|_{\Delta\Phi=0}$$

For proximal gradient descent we then have

$$\Delta\Phi^* \approx \operatorname{argmin} \mathcal{L}(\Phi + \Delta\Phi) \text{ subject to } \Delta\Phi^\top H \Delta\Phi \leq 2\epsilon$$

Again $\mathcal{L}(\Phi)$ is the total gradient.

Distances and Metrics

$$\Delta\Phi^* \approx \text{argmin}\, \hat{g}^\top \Delta\Phi \text{ subject to } \Delta\Phi^\top H \Delta\Phi \leq 2\epsilon$$

Setting the gradient of the objective to be proprtional to the gradient of the constraint (KTT conditions) we get

$$H\Delta\Phi = -\lambda g$$

$$\Delta\Phi = -\lambda H^{-1} g$$

Distances and Metrics

$$\Delta\Phi = -\eta H^{-1} g$$

$$(\eta H^{-1} g)^\top H (\eta H^{-1} g) = 2\epsilon$$

$$\eta = \sqrt{\frac{2\epsilon}{g^\top H^{-1} g}}$$

g is the total gradient.

Conversion to SGD

$$\Delta\Phi = -\eta H^{-1} g$$

Can now do this stochastically with

$$\Delta\Phi_t = -\eta H^{-1} \hat{g}_t$$

Hence information geometry provides a third "natural" gradient direction in addition to the Hessian of the loss and the covariance of gradient vectors.

Information Geometry and Gradient Descent

For KL divergence H is diagonal with

$$\Delta P^\top H \Delta P = \sum_y \frac{\Delta P(y)^2}{P(y)}$$

Although KL is not symmetric, H happens to be the same for $KL(P, P + \Delta P)$ and $KL(P + \Delta P, P)$.

Hessian-Vector Products

$$Hv = \nabla_\Phi \left((\nabla_\Phi \, f^t(\Phi))^\top v \right)$$

This is supported in PyTorch — in PyTorch Φ.grad is a variable while Φ.grad.data is a tensor.

Hessian-Vector Products

For backpropagation to be efficient it is important that the value of the graph is a scalar (like a loss). But note that for v fixed we have that

$$(\nabla_\Phi \, f^t(\Phi))^\top v$$

is a scalar and hence its gradient with respect to Φ, which is Hv, can be computed efficiently.

Complex-Step Differentiation

Consider a function $f : \mathbb{R} \to \mathbb{R}$ defined by a computer program.

For C code on a CPU we can run a program on complex numbers simply by changing the data type of x.

James Lyness and Cleve Moler, Numerical Differentiation of Analytic Functions SIAM J. of Numerical Analysis, 1967.

Complex-Step Differentiation

Consider $f(x+i\epsilon)$ at real input x and consider the first order Taylor expansion.

$$f(x+i\epsilon) = f(x) + i(df/dx)\epsilon$$

Note that $f(x)$ and df/dx must both be real. Therefore

$$\mathrm{Im}(f(x+i\epsilon)) = \epsilon(df/dx)$$

$$\frac{df}{dx} = \frac{\mathrm{Im}(f(x+i\epsilon))}{\epsilon}$$

Complex-Step Differentiation

$$\frac{df}{dx} = \frac{\mathrm{Im}(f(x+i\epsilon))}{\epsilon}$$

This is vastly better than

$$\frac{df}{dx} \approx \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

The point is that in complex arithmetic the real and imaginary parts have independent floating point representations.

In 64 bit floating point arithmetic ϵ can be taken to be 2^{-50}.

For $\epsilon = 2^{-50}$, division by ϵ simply changes the exponent of the floating point representation leaving the mantissa unchanged.

First Order Polynomial Arithmetic

Numerically, complex-step differentiation is equivalent to first order polynomial arithmetic.

$$(a + b\epsilon)(a' + b'\epsilon) = (a + a') + (ab' + a'b)\epsilon$$

Differentiation based on first order polynomial arithmetic is exact.

Equivalence to Polynomial Arithmetic

$$(a + ib\epsilon)(a' + ib'\epsilon) = (a + a' - bb'\epsilon^2) + i(ab' + a'b)\epsilon$$

$$\epsilon = 2^{-50}$$

Here the ϵ^2 term is below the precision of $a + a'$.

Numerically, complex-step arithmetic and first order polynomial arithmetic are the same.

Hessian-Vector Products

We are interested in computing $H_t v$ for $v = (\eta \odot \hat{g})$.

$$H_t v = \frac{\mathrm{Im}(\nabla_\Phi f(\Phi)|_{\Phi + i\epsilon v})}{\epsilon}$$

$$\epsilon = 2^{-50}$$

Structure From Motion

See the Videos
https://www.youtube.com/watch?v=i7ierVkXYa8
http://www.mada.org.il/brain/Shape/shape.html

The Levenberg-Marquart Algorithm (Bundle Adjustment)

$$\text{Loss}_\Phi(x, y) = \sum_i \frac{1}{2} ||f_\Phi(x_i) - y_i||^2$$

$$f_{\Phi+\Delta\Phi}(x_i) \approx f_\Phi(x_i) + J_i \Delta\Phi$$

$$J_i = \nabla_\Phi \, f_\Phi(x_i)$$

$$\hat{g}_i = (f_\Phi(x_i) - y_i) J_i \quad = r_i J_i$$

$$r_i = f_\Phi(x_i) - y_i$$

The Levenberg-Marquart Algorithm

$$\text{Loss}(\Phi + \Delta\Phi) \approx \sum_i \frac{1}{2}||r_i + J_i\Delta\Phi||^2$$

Minimizing this squared error over the choice of $\Delta\Phi$ is a least squares regression problem.

The Levenberg-Marquart Algorithm

$$\text{Loss}(\Phi + \Delta\Phi) \approx \sum_i \frac{1}{2}||r_i + J_i\Delta\Phi||^2$$

$$\begin{aligned} 0 &= \sum_i (r_i + J_i\Delta\Phi)J_i \\ 0 &= (\sum_i r_i J_i) + \sum_i J_i^\top J_i\Delta\Phi \\ (\sum_i J_i^\top J_i)\Delta\Phi &= -\sum_i \hat{g}_i \\ \Delta\Phi &= -(\sum_i J_i^\top J_i)^{-1}(\sum_i \hat{g}_i) \end{aligned}$$

Levenberg-Marquart for Information Geometry

$$\mathcal{L}'_t(\Phi + \Delta\Phi) = \mathcal{L}_t(\Phi) + \lambda \mathrm{KL}(P_\Phi, P_{\Phi+\Delta\Phi})$$

We consider only the second order structure in the KL divergence.

This will control the step size.

END