BIOMATH 208 Week 8

Aidan Jan

February 28, 2025

Naive Gradient Descent in Vector Spaces

Given a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, the naive gradient descent algorithm procedes as follows:

1. Choose an initial guess for some point p
2. Compute $f(p)$, and $\mathrm{d}f(p)$ (list of partial derivatives)
3. Update $p \mapsto p - \varepsilon \mathrm{d}f(p)$, $\varepsilon > 0 \in \mathbb{R}$, "small"
4. Repeat 2 and 3, until we converge
    - Stop after some number of repeats
    - if $f(p)$ changes less than threshold, then stop
    - if p changes less than a threshold, then stop
    - etc.
5. maybe, change ε (step size, learning rate, etc.) based on some criteria.

Gradient descent is a minimization method

Gradient descent converges to a stationary point for small enough ε.

Proof

Taylor expand f about p_0.

$$f(p) = f(p_0) + \mathrm{d}f(p_0) \cdot (p - p_0) + O(|p - p_0|^2)$$

- $(p - p_0)$ is our step size. Also, $p - p_0 = -\epsilon \mathrm{d}f(p_0)$
- Notice we are taking the inner product of a covector and a vector.
- f should be smooth, e.g., a derivative in at least one direction must be well defined.
- Essentially, f of the new point p is equal to f of the old point p_0, plus the change when we took a step $\mathrm{d}f(p_0) \cdot (p - p_0)$, plus our loss function $O(|p - p_0|^2)$.

If we substitute $p - p_0 = -\epsilon \mathrm{d}f(p_0)$ for $(p - p_0)$, then we get

$$f(p_0) + (-\varepsilon) \cdot |\mathrm{d}f(p_0)|^2 + O(|p - p_0|^2)$$

$|\mathrm{d}f(p_0)|^2$ is a euclidean norm, and therefore that term is always $\leq f(p_0)$. If it is equal to $f(p_0)$, then we have converged.

Problems with Gradient Descent

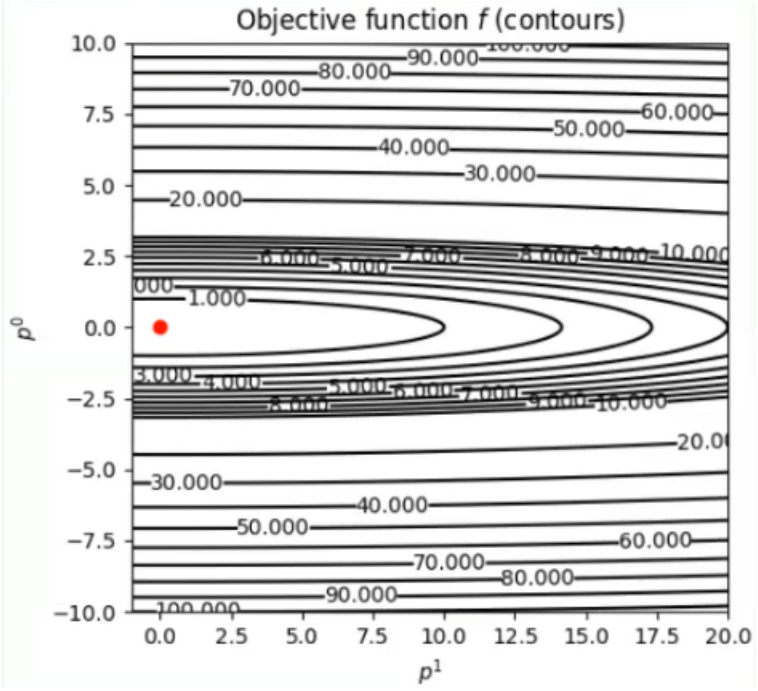
1. The components of $\mathrm{d}f$ may have widely different scales. Choosing "small enough" ε may be dominated by the largest scale, making the algorithm impractically slow.
2. The components of $\mathrm{d}f$ depend on a choice of chart or basis. Some charts may have good performance and others bad. (e.g., preconditioning)
3. Since $\mathrm{d}f$ is a covector, it doesn't make sense to add a scalar multiple of it to a point in our vector space $\mathbb{R}^d$. (you can't add a covector to a vector; they have different units!)
    - if f was unitless, and p has units of length, then $\mathrm{d}f(p)$ has units of "per length".

Example: Minimization with Different Units

Consider a 2D vector space, where the first quantity is measured in centimeters, and the second in millimeters. Then let

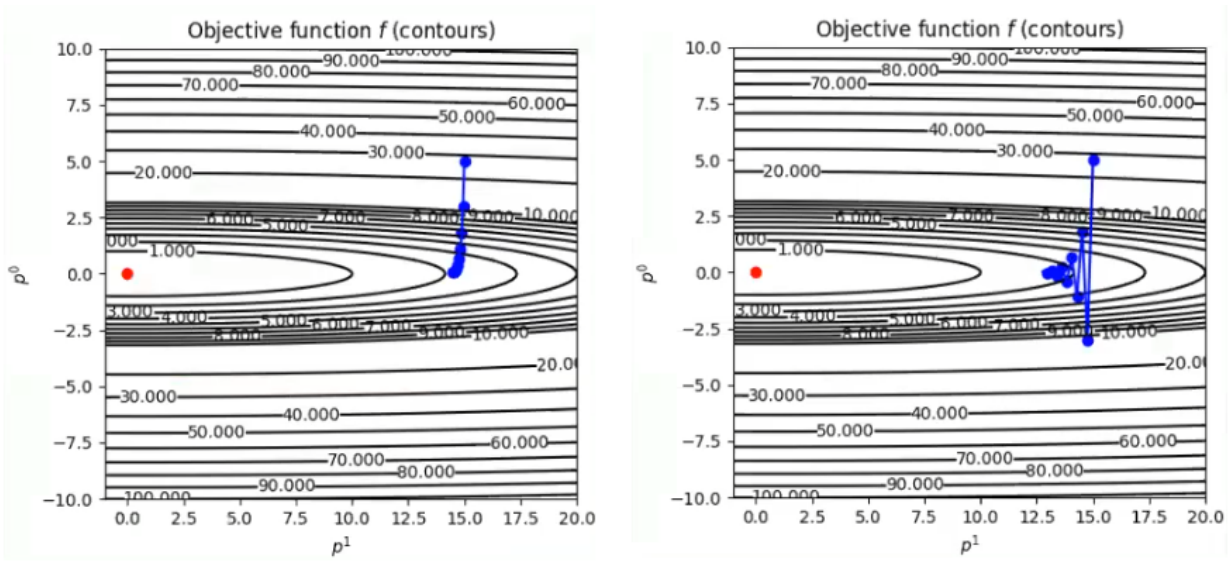
$$f(p) = (p^0)^2 + \frac{2}{10^2}(p^1)^2$$

be the distance from the origin in centimeters. The minimum is $p = 0$.



Example: Sequernce of Points in Gradient Descent

The gradient $\mathrm{d}f = (2p^0, 2/10^2 p^1)$, and a sequence of points for different values of ϵ are shown.



- The highest point on both images are the initial points.
- The $\mathrm{d}f$ is the list of partial derivatives.
- The red dot is the origin.

Because of the gradient, the gradient in the p^1 direction is 100x smaller than the gradient in the p_0 direction. This is why the step goes down very fast, and left very slow.

The right graph uses a much higher step size (ε), and therefore it takes a much larger step per iteration.

Natural Gradient Descent in a Coordinate Chart

We can convert the gradient covector $\mathrm{d}f$ to a vector $\mathrm{d}f^\sharp$. Its direction will be independent of the choice of chart. This algorithm was invented by Shun'ichi Amari in the early 21st century.

1. Choose an initial point p. (On a Lie group we could pick identity)
2. Compute $f(p)$, $\mathrm{d}f(p)$, $g(p)$. $g(p)$ is a metric that acts on two vectors, usually just a symmetric matrix.
3. $x^i \mapsto x^i - \epsilon(g^{-1}(p))^{ij}\mathrm{d}f_j(p)$, for small enough ε.
4. Repeat until convergence
5. Possibly, modify ε based on same heuristic.

One Problem

Note: while the direction (tangent vector), is independent of a choice of chart, the new point is not. (We still have to walk along a curve in that direction, which is not unique.)

Natural Minimization with Different Units

It is natural to choose a metric that takes the different units into account, and returns a length in centimeters. For the example before, let's define

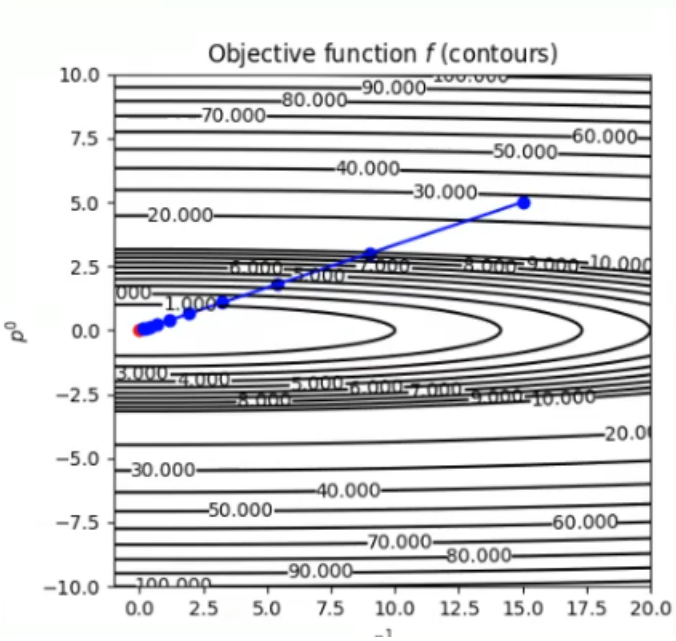
$$g_{ij} = \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{100} \end{pmatrix}_{ij}$$

Therefore,

$$(g^{-1})^{ij} = \begin{pmatrix} 1 & 0 \\ 0 & 10^2 \end{pmatrix}_{ij}$$

The components of our gradient are then

$$(g^{-1})^{ij} \mathrm{d}f(p)_j = \begin{pmatrix} 1 & 0 \\ 0 & 10^2 \end{pmatrix} \begin{pmatrix} 2p^0 \\ \frac{2}{100} p^1 \end{pmatrix} = \begin{pmatrix} 2p^0 \\ 2p^1 \end{pmatrix}$$



- Because we understand the geometry of this scenario better, we can scale the gradient of p_1, so now we walk straight towards the origin, instead of being dominated by the smaller component.
- Our step can be big, but not overstep the valley, since we choise a better direction.
- We are lucky that in this scenario, this is a quadratic optimization problem. If it were more complicated, the results would not be as nice, but it still will help.
- Note, notice that the first matrix, $\begin{pmatrix} 1 & 0 \\ 0 & 10^2 \end{pmatrix}$ is actually the Hessian of f (second derivative). So this is an example of Newton's method. In general, $\mathrm{H}f$ is not positive definite, and in general, g and f are not related.

How to Choose a Metric

- Affine registration is a $d \times (d+1)$ dimensional optimization problem, e.g., 12 dimensions for 3D registration.
- Choosing a metric means choosing $d(d+1)[d(d+1)-1]/2$ different numbers, e.g., 66 numbers for 3D registration.
- AND, we must do this at every point on the manifold.
- Luckily we have developed tools to make this problem simpler.
- For the previous problem, we just used distance squared, but most of the time, the problem isn't this simple.

Pull back metrics

- There is no "natural" way to put a metric on affine transforms T, but there is a natural way to put a metric on the points they act on, the standard Euclidean metric.
- Let Q be a set of points and δQ, δP two tangent vectors to the points at Q, then

$$g_Q^{\mathbb{R}^{d\times N}}(\delta Q, \delta P) = \mathrm{tr}(\delta Q \delta P^T)$$

  - We are multiplying the components and adding them up
  - Q and P are perturbations to points, $d \times N$ matrices for N points
- If X, Y are tangent vectors at T, and Q is a set of points, let $\phi(T) = T \cdot Q$. Then we can define the pullback metric

$$g_T^{\text{affine}}(X, Y) = [\phi^* g_Q^{\mathbb{R}^{d\times N}}](x, y) = g_Q^{\mathbb{R}^{d\times N}}(\phi_* X, \phi_* Y) = g_Q^{\mathbb{R}^{d\times N}}(XQ, YQ) = \mathrm{tr}(XQ(YQ)^T)$$

  - The bracketed term is the pull back. The parenthetical (right side) is the push forward.
  - Recall, push forward is evaluated by multiplying by the Jacobian matrix of ϕ.
  - Jacobian of ϕ is just Q.

Natural Gradient Descent for Affine Registration in 1D

Consider aligning a set of points Q to another set of points P, over affine transformations T with T^0 the linear part and T^1 the translation. We seek to minimize

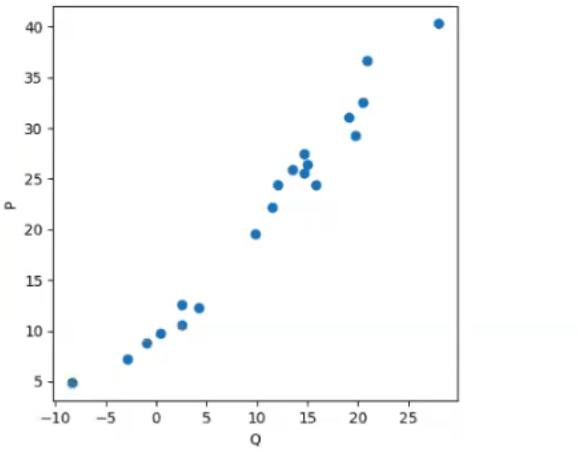
$$f(T) = \|T \cdot Q - P\|_F^2 = \sum_{i=1}^{N} (T^0 Q_i + T^1 - P_i)^2$$

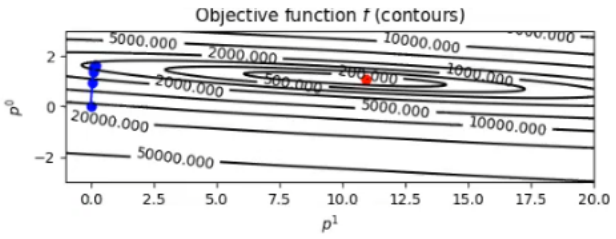
The gradient, $\mathrm{d}f(T)$, is given by

$$\mathrm{d}f_j(T) = \begin{pmatrix} 2\sum_{i=1}^{N} (T^0 Q_i + T^1 - P_i) Q_i \\ 2\sum_{i=1}^{N} (T^0 Q_i + T^1 - P_i) \end{pmatrix}$$

- Now that we know how to evaluate the function and its gradient, we can apply naive gradient descent.

Points Q and P are shown on left, and trajectory is shown on right.





Notice that the points are correlated, so we can approximate one as an affine function of the other.

- Notice that in this graph, the translation seems to be an order of magnitude larger than the transformation.
- Also, notice that the graph of P and Q does not start at the origin. Therefore, we need to find the center of mass of the points and scale accordingly.
- We will converge in T^2 quickly, but T^1 slowly.

Example: The Pullback Metric

For $\phi(T) = T^0 Q + T^1$, the push forward is

$$\phi_*^T(u) = (Q, 1)u$$

- u is a tangent vector

So the pullback metric is

$$\begin{aligned} g_T(u,v) &= (u^0 Q + u'1)(v^0 Q + v'1)^T \\ &= \begin{pmatrix} u^0 & u^1 \end{pmatrix} \begin{pmatrix} QQ^T & Q1^T \\ 1Q^T & 1\,1^t \end{pmatrix} \begin{pmatrix} v^0 \\ v^1 \end{pmatrix} \\ &= \begin{pmatrix} u^0 & u^1 \end{pmatrix} \begin{pmatrix} \sum_{i=1}^N Q_i^2 & \sum_{i=1}^N Q_i \\ \sum_{i=1}^N Q_i & N \end{pmatrix} \begin{pmatrix} v^0 \\ v^1 \end{pmatrix} \end{aligned}$$

Example: The Inverse Metric

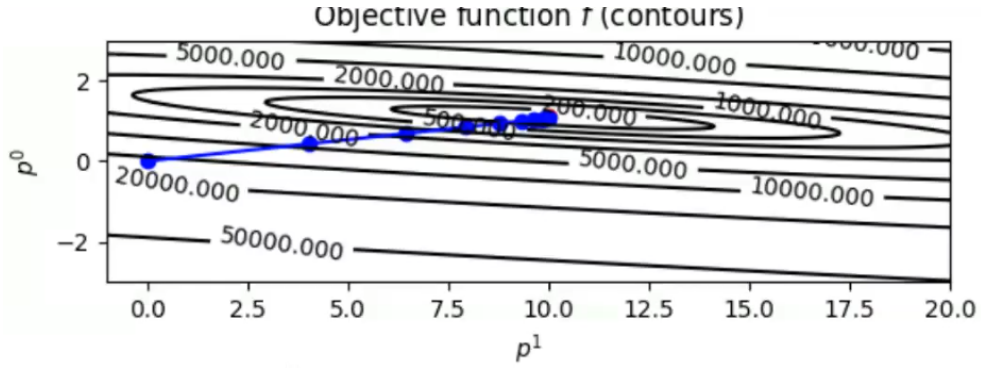
The inverse of the metric is then:

$$g^{-1}(T)_{ij} = \frac{1}{N\sum_{i=1}^N Q_i^2 - \left(\sum_{i=1}^N Q_i\right)^2} \begin{pmatrix} N & -\sum_{i=1}^N Q_i \\ -\sum_{i=1}^N Q_i & \sum_{i=1}^N Q_i^2 \end{pmatrix}$$

Note that if the center of mass is 0, the metric is diagonal.

Example: The natural gradient descent trajectory

Natural gradient descent improves the trajectory dramatically



- We have scale on the y-axis and translation on the x-axis.
- We head straight to the origin this time!
- We can choose a bigger step size and converge quickly in both parameters.

Invariant Metrics

- Above, our metrics were independent of $p \in \mathcal{M}$, but this is not generally possible. With Lie groups, we can define the metric at $p = I$, and use a pullback to define it for any other point.
- For $p, q \in \mathcal{M}$ we can define a map at each p by $\phi_p(q) = p^{-1} \circ q$.
  - If this is a matrix group, ϕ is matrix multiplication, so ϕ_* is just a matrix. (Jacobian is just a matrix.)
- We push forward a vector in $T_p\mathcal{M}$ to $T_I\mathcal{M}$ with $\phi_*(p)$, and define

$$g_p(u, v) = g_I(\phi_*(p)u, \phi_*(p)v)$$

- This is called a "left invariant metric". We could design a "right invariant metric" using $\phi_p(q) = q \circ p^{-1}$.

Review

[FILL]

In general we cannot find analytic solutions. We used the gradient descent algorithm, an iterative algorithm that has... [FILL]

Natural Gradient Descent

1. Start with an initial guess for our parameter, p.
2. Compute $f(p)$, $\mathrm{d}f(p)$, $g(p)$
3. Replace the i-th component of p: $p^i \mapsto p^i - \varepsilon(g(p)^{-1ij})\mathrm{d}f_j(p)$. (Note that g^{ij} means (g_{ij}^{-1}))
4. Same as gradient descent
5. Same as gradient descent

- This is especially helpful when compoennts of p have different units
- Also gives a better search direction allowing a bigger step size ε.

How to choose a metric?

- In a d dimensional manifold, $d(d-1)/2$ functions, and they have to always be positive definite.
- Pull back metrics
  - Find a map $\phi : \mathcal{M} \to$ a nice space where choosing a metric is easy

$$g_p^{\mathcal{M}}(x, y) = g_{\phi(p)}^{\text{nice space}}(\phi_{\phi(p)}X, \phi_{\phi(p)}Y)$$

- We showed how to use this metric for affine transforms acting on points, because it's easy to define a metric on the points themselves.
- Invariant metric, for Lie groups:
  - Define a metric at one point $p = I$, then for any other point p, we just use a pull back metric, pull back with $\phi = p^{-1}$.

Image Registration

(for images as functions, NOT landmark points.)

Consider a piar of grayscale images $I, J : \mathbb{R}^d \to \mathbb{R}$, and a transformation acting via $[T \cdot I](x) = I(T^{-1}x)$. We will minimize the integral square error objective function

$$f(T) = \int (I(*T^{-1}x) - J(x))^2 \mathrm{d}x$$

using gradient based methods.

Note that even though this is a square error cost, it is **not quadratic** in T, (because I is not a linear function of its argument) and therefore cannot be solved analytically

Translation

We will start with a very simple transformation group: the translation.

Consider T as a translation in $\mathbb{R}^d$ with $[T \cdot I](x) = I(x - T)$. (The $\cdot$ is a group action.) The gradient is

$$\mathrm{d}f(T) = -2 \int [IU(x - T) - J] \mathrm{d}I(x - T) \mathrm{d}x$$

- f is integral square error
- $I(x - T) - J$ is the error term.
- $\mathrm{d}I$ is the gradient of the image with respect to space
- $I(x - T)$ is the transformed images, not the original

Proof

Consider a curve $\gamma(t) = T + t\delta T$ for δT an arbitrary translation, and consider the velocity

$$\frac{\mathrm{d}}{\mathrm{d}t} f(T + t\delta t)\bigg|_{t=0} = v_{\gamma,T}(f)$$

Plugging in our definition of f gives:

$$\frac{\mathrm{d}}{\mathrm{d}t} \int (I(x - T - t\delta T) - J(x))^2 \mathrm{d}x \bigg|_{t=0}$$

By the chain rule,

$$\begin{aligned} &= \int 2(I(x - T) - J(x)) \mathrm{d}I(x - T)(-\delta T) \\ &= [-2 \int (I(x - T) - J(x)) \mathrm{d}I(x - T) \mathrm{d}x] \delta T \end{aligned}$$

- T is a vector
- everything before the $\mathrm{d}x$ is the direction, or $\mathrm{d}f(T)$.

Affine Group Transformations

The gradient covector, as a $(d+1) \times (d+1)$ matrix, with bottom row all zeros, is given by

$$\mathrm{d}f(T) = -2 \int [I(T^{-1}x) - J(x)] \mathrm{d}[I(T^{-1}x)](T^{-1}x)^T \mathrm{d}x$$

This can be written in a matrix as:

$$\begin{pmatrix} L & T \\ 0 & 1 \end{pmatrix}$$

where

- L is linear
- T is translation
- bottom row is fixed (homogeneous coordinates)
- $[I(T^{-1}x) - J(x)]$ is the error term, transformed I minus J.
- $\mathrm{d}[I(T^{-1}x)]$ is the gradient of the transformed I.

To prove this equation, we first need to define some other things:

Part 1: Derivative of inverse matrix

$$\frac{\mathrm{d}}{\mathrm{d}t}(T + t\delta T)^{-1} = -T^{-1}\delta T T^{-1}$$

where

- δT is direction
- t is time
- This is very well defined if t is small, and δT has zeros on its bottom row.

The derivative of a matrix with respect to another matrix is multiplying the inverse matrix on both sides of the direction. In this case, it is the $T^{-1}\delta T T^{-1}$ section. We can show this because

$$I = (T + t\delta T)(T + t\delta T)^{-1}$$

Now, we take the derivative of both sides:

$$0 = \frac{\mathrm{d}}{\mathrm{d}t} I = \frac{\mathrm{d}}{\mathrm{d}t}(T + t\delta T)(T + t\delta T)^{-1}$$

By product rule,

$$= \delta T(T + t\delta T)^{-1} + (T + t\delta T)\frac{\mathrm{d}}{\mathrm{d}t}(T + t\delta T)^{-1}\Big|_{t=0}$$

When we evaluate at $t = 0$, a lot of things disappear

$$\begin{aligned} &= \delta T T + T\frac{\mathrm{d}}{\mathrm{d}t}(T + t\ deltaT)^{-1}\Big|_{t=0} \\ -\delta T T^{-1} &= T\frac{\mathrm{d}}{\mathrm{d}t}(T + t\ deltaT)^{-1}\Big|_{t=0} \\ -T^{-1}\delta T T^{-1} &= \frac{\mathrm{d}}{\mathrm{d}t}(T + t\delta T)^{-1}\Big|_{t=0} \end{aligned}$$

If big T were a scalar, this is equivalent to the "quotient rule" for taking derivatives

Part 2: Derivative of image with affine

$$\mathrm{d}I(T^{-1}x)T^{-1} = \mathrm{d}[I(T^{-1}x)]$$

- Left side: gradient of image, transformed
- Right side: gradient of the transformed image

Proof:

Start with the right side and apply the chain rule.

$$\begin{aligned}\mathrm{d}[I(T^{-1}x)] &= \mathrm{d}I\Big|_{T^{-1}x} \cdot \frac{\mathrm{d}}{\mathrm{d}x}T^{-1}x \\ &= \mathrm{d}I\Big|_{T^{-1}x} \cdot T^{-1} \\ &= \mathrm{d}I[T^{-1}x]T^{-1}\end{aligned}$$

Note: $\mathrm{d}I \doteq [\mathrm{d}I] \neq [\mathrm{d}][I]$

Part 3: The gradient

Find the directional derivative

$$\frac{\mathrm{d}}{\mathrm{d}t}\int [I((T+t\delta T)x) - J(x)]^2 \mathrm{d}x\Big|_{t=0}$$

$$\begin{aligned}&= \int 2[I((T+t\delta T)_x) - J(x)] \cdot \mathrm{d}I((T+t\delta T)^{-1}x) \cdot (-T^{-1}\delta T T^{-1})x\mathrm{d}x \\ &= \int 2[I(T^{-1}x) - J(x)]\mathrm{d}I(T^{-1}x)T^{-1}\delta T T^{-1}x\mathrm{d}x \\ &= \int -2[I(T^{-1}x) - J(x)]\mathrm{d}[I(T^{-1}x)]\delta T T^{-1}x\mathrm{d}x\end{aligned}$$

At this point, we would like to isolate δT, but it doesn't factor nicely because these are matrices. Therefore, we will work with the trace of the matrix.

Since the whole quantity is scalar, we can take the trace:

$$= -2\int (I(T^{-1}x) - J(x))\mathrm{tr}\left(\mathrm{d}[I(T^{-1}x)]\delta T T^{-1}x\right)\mathrm{d}x$$

With the trace, we have the cyclic permutation property, so we can rearrange the terms.

$$\begin{aligned}&= \cdots \mathrm{tr}\left(\delta T T^{-1}x\mathrm{d}[I(T^{-1}x)]\right) \\ &= \cdots \mathrm{tr}\left(\mathrm{d}(I(T^{-1}x))^T(T^{-1}x)^T\delta T^T\right) \\ &= \det(\mathrm{d}(I(T^{-1}x))(T^{-1}x)^T, \delta T) \\ &[FILL] \\ &= \mathrm{tr}\left(-2\int (I(T^{-1}x) - J(x))\mathrm{d}(I(T^{-1}x))^T x^T (T^{-1})^T \mathrm{d}x \cdot \delta T^T\right)\end{aligned}$$

Notice that at this point, we have factored the equation into a vector times a covector, and therefore the group action on the derivative. This completes the proof.

Metrics for Image Registration

One simple metric is to choose every voxel as a point, and use our pull back metric for point sets. (Use what we already do for point sets. [FILL])

Automatic Differentiation

Automatic differentiation works by defining a computation graph with data as edges, and functions as nodes.

For every function $(f(x))$, we define push forward $(f_*(x)[X])$, and a pull back $(f^*(x)[\chi])$. The former tells us how a perturbation of inputs affects outputs. The latter tells us how gradients should be pulled back (chain rule) for optimization.

- The push forward represents the forward pass (calculating the gradients)
- The pull back is how the nodes should be modified through backpropagation on the gradient to adjust parameters.
- In some applications, the push forward isn't needed sometime.

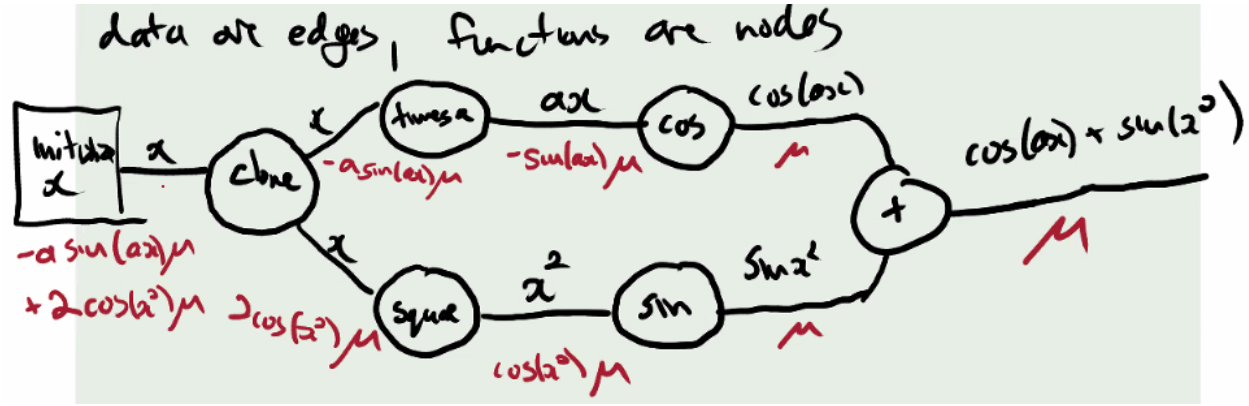
Example: (Computation graph)

Build a computation graph for the function:

$$f(x) = \cos(ax) + \sin(x^2)$$

Forward propagation in black, backpropagation in red.

- When you go backwards, you apply the derivative of each function.



- pull back, multiply by transpose of Jacobian, often can be done without computing or storing Jacobian matrix.

Riemannian Manifolds and Geodesics

- A Riemannian manifold is one with a metric defined at every point
- Geodesics is the shortest path between two points

Motivation

- Here we consider the implications of putting an inner product on a manifold.
- This will allow us to measure lengths and angles, and define straight lines.
- These operations will allow us to compute a distance between any pair of points, which can serve as an input to many machine learning algorithms.
- These operations will extend the definitions of a lot of familiar data processing techniques: filtering, averaging, regression

Riemannian Manifolds

- A Riemannian manifold is a smooth manifold with a (0, 2) tensor field that describes an inner product at every point
- This is usually denoted by the symbol g_p for a metric tensor at the point p. It is a nonlinear map $T_p\mathcal{M} \times T_p\mathcal{M} \to \mathbb{R}$.

FILL

The length of a curve

Given a curve $\gamma : [0, 1] \to \mathcal{M}$, its length is given by

$$L(\gamma) = \int_0^1 \sqrt{g_{\gamma(t)}(v_{\gamma,\gamma(t)}, v_{\gamma,\gamma(t)})}\mathrm{d}t$$

This corresponds to the familiar definition "distance equals speed times time", but note the difference between speed and velocity.

Velocity is more fundamental, whereas speed requires us to add additional structural to our manifolds.

Distances and Geodesics

We define the distance between two points on a manifold as the length of the shortest curve that connects them.

$$d(p, q) = \min_{\gamma : [0,1]\to\mathcal{M}\gamma(0)=p,\gamma(1)=q} L(\gamma)$$

These length minimizing curves are called geodesics.

"Geodesic" has two definitions.

- the shortest path between two points
- a stationary point in the above optimization problem

Most of the time, they coincide, but consider the following:

The red path meets the second definition, but not the first.

Action Integrals

Because there are an infinite number of parameterizations of the same curve γ, we choose to work with a constant speed geodesics. These are minimizers of the action integral

$$A(\gamma) = \int_0^1 g_{\gamma(t)}(v_{\gamma,\gamma(t)}, v_{\gamma,\gamma(t)})\mathrm{d}t$$

In a coordinate chart x, with $x(\gamma(t)) \doteq q(t)$ and $\dot{\gamma}_{(x)}(t) \doteq \dot{q}(t)$, and $g_{x^{-1}(q(t))}\left(\frac{\partial}{\partial x^i}, \frac{\partial}{\partial x^j}\right) \doteq g_{ij}(q(t))$, we have:

$$A(q) = \int_0^1 g_{ij}(q(t))\dot{q}^i(t)\dot{q}^j(t)\mathrm{d}t$$

Constant Speed Geodesics

With fixed endpoints minimizers of A are constant speed, and are also minimizers of L.

Proof:

Let $f(t) = \sqrt{g_{\gamma(t)}} \dots$ [FILL] [FILL]

Now, consider another function $h(t)$ (arbitrary), and consider the L_2 inner product $\int_0^1 f(t)h(t)\mathrm{d}t$. By Cauchy-Schwartz, we have

$$\left(\int_0^1 f(t)h(t)\mathrm{d}t\right)^2 \leq \int |f|^2(t)\mathrm{d}t \int |h|^2(t)\mathrm{d}t$$

Choosing $h(t) = 1$ gives

$$\left(\int_0^1 f(t)\mathrm{d}t\right)^2 \leq \int_0^1 |f|^2(t)\mathrm{d}t \cdot 1$$

Notice that this is the action integral. This implies that $L^2(\gamma) \leq A(\gamma)$. If γ has a constant speed c, then the left and right side are equal to c^2, and the inequality becomes an equality (f, h are colinear).

So A achieves the lower bound of L over reparameterizations.

- L does not change when we reparameterize γ.
- A obtains its smallest value when it is constant speed and when A is using a [FILL]
- Suppose for the purpose of contradiction that γ is a minimizer of A but not a minimizer of L. Then there is another curve α with constant speed reparameterization $\tilde{\alpha}$ such that

$$L(\alpha) < L(\gamma)$$

- But the left side is $\sqrt{A(\tilde{\alpha})}$ and the right side is $\sqrt{A(\gamma)}$, meaning γ is not a minimizer of A. We therefore reject our assumption.

The constant speed geodesic equation

In a given coordinate chart, with components of a metric tensor field g written as g_{ij}, and its inverse written g^{ij}, constant speed geodesics are determined by

$$\ddot{q^i} + \frac{1}{2}g^{ij}(-\partial_j g_{kl})[FILL]$$