MATH4540 Mini Project 1
Vector Bundles of Surfaces

Zichen Wang

May 15, 2022

A central theme of differential geometry is to study geometric objects that are locally Euclidean. As we have seen in this course, every point on a smooth surface can be approximated by its tangent plane consisting of the first order derivatives, while the second order derivatives tell us the Gaussian curvature, that is, how much the surface deviates from the tangent plane. In this sense, the tangential space is intrinsic to the geometric object. Although with the ambient space $\mathbb{R}^3$ we can clearly tell the difference between different tangent planes at different points, as a bug on the surface we could hardly distinguish them without referring to the position we are at. This leads us to the notion of the *tangent bundle*.

Definition 1. *Given a surface S, the* tangent bundle *of the surface $TS = \{(p, v) : p \in S, v \in T_pS\}$*

Instead of directly taking the union of all tangent spaces, we incorporate the position prior to each tangent vector. Intuitively, this can be understood as tying a string of tangent vectors at each point of the surface. If the surface is $\mathbb{R}^2$, then the tangent space at each point is also $\mathbb{R}^2$ (for now imagine embedding the plane into $\mathbb{R}^3$), and the tangent bundle $T\mathbb{R}^2 \cong \mathbb{R}^4$. Similarly, $T\mathbb{R}^3 \cong \mathbb{R}^6$.

In general, it is not favorable to define our tangent vectors based on the ambient space, as our choice of the ambient space can be arbitrary. One natural idea is to generalize how we defined tangent vectors as derivatives of curves on the surface.

Definition 2. *Given smooth surfaces S_1 and S_2, and a smooth map $\phi : S_1 \to S_2$, the* differential *of ϕ is*

$$\begin{aligned} d\phi : & TS_1 \to TS_2 \\ & (p, \alpha'(0)) \to (\phi(p), (\phi \circ \alpha)'(0)) \end{aligned}$$

where α is a curve on S_1 with $\alpha(0) = p$.

Thus one way to understand tangent vectors is to use patches. Let S_1 be a subset of the Euclidean space that the surface S_2 is defined on. Then we can obtain the tangent space of S_2 by calculating the differential of the patch.

Definition 3. *A* section *of a tangent bundle is a map $s : S \to TS$ such that projection of $s(p)$ back to S is p itself for any $p \in S$.*

In other words, the section is a vector field that assigns each point on the surface a tangent vector. It got its name from the intuition that we can "cut" the strings of the tangent bundles to form a section.

The field of geometry studies properties such as length, angles, and curvatures, all of which require a metric. Essentially, the first fundamental form is the metric. Given two tangent vectors at a point, the first fundamental form outputs the inner product of the vectors. As the surface is locally Euclidean, it is natural to think of the tangent space at a point as a vector space. This suits well that the inner product should be bilinear.

Definition 4. *Given a vector space V, the vector space of all real-valued linear functions on V is called the* dual *of V, denoted by V^*.*

Definition 5. *Given a surface S, the* cotangent bundle $T^*S = \{(p, v) : p \in S, v \in T_p^*S\}$

Definition 6. *Given a smooth function $f : V \to \mathbb{R}$, the* differential *of f at p is $df \in V^*$ such that*

$$df_p(v) = \lim_{t\to 0} \frac{f(p + tv) - f(p)}{t} \textit{for any } v$$

Note that this definition can be seen as a special case of definition 2, with $\phi = f$ and S_2 degraded to the real line.

Theorem 1. *If V is a finite dimensional vector space, and $\langle \cdot, \cdot \rangle$ is a non-degenerate inner product, then we have the isomorphism*

$$\begin{aligned} T :& V \to V^* \\ & v \to \langle v, \cdot \rangle \end{aligned}$$

Corollary 1. *The* gradient *of a smooth function f on the surface is the section of the tangent bundle corresponding to the differential of f under the isomorphism specified by the first fundamental form.*

We see that the differential is decided by f whereas the gradient further depends on the metric that we choose. Given the first fundamental form as the metric, we can specify the isomorphism between TS and T^*S. That is, for any curve $\gamma \subseteq S$ which passes p when $t = 0$,

$$\begin{aligned} \langle \nabla_S f, \gamma'(0) \rangle &= \frac{d}{dt} f(\gamma(0)) \\ &= \lim_{t\to 0} \frac{f(\gamma(t)) - f(\gamma(0))}{t} \\ &= df(\gamma'(0)) \end{aligned}$$

Thus, by Theorem 1, at each point on the surface we obtain the gradient as a tangent vector at that point, and together the gradients form a section of the tangent bundle.

One final remark is that when we discuss tangent vectors, we said that it is conducive to think of surface tangent vectors corresponding to tangent vectors in patches. The problem is that this still does not avoid the arbitrary choice of our patches. A solution to this is to instead think of not curves in patches, but functions along curves. Thus we have tangent vectors as real-valued functions from the set of all smooth functions to $\mathbb{R}$.

$$\alpha'(0)f = \left.\frac{d(f \circ \alpha)}{dt}\right|_{t=0}$$

Optimization using Gradient Descent

Zichen Wang

May 15, 2022

Gradient descent has been one of the most commonly used technique by computer scientists in all sorts of optimization problems. Other than the already abundant blogs on this amazing hack, I would like to present you some more advanced usage of gradient descent that you might not have seen before.

1 ML101 Intro to Gradient Descent

Below is the pseudo-code of the simplest gradient decent algorithm. It take only a few lines of codes!

Algorithm 1 Gradient Descend

```
starting at $x_0$
for $t = 1, ..., T$ do
    $x_t \leftarrow x_{t-1} - \eta \nabla f_x$
end for
return $x = \arg\min\{f(x_0), ..., f(x_T)\}$
```

Given a function f, our goal is to find a point that minimizes f. To do this, we follow a greedy approach: repeatedly take small steps that "descend" us to a position with smaller function value. A few things to notice here

- At each step, we follow the direction of the *gradient* at our current position. As I showed in my last paper, the negative gradient gives the direction that the function value decreases the "fastest".
- The *step size* of each each small step is reflected in the parameter η. If we take a large step every time, then we might reach the destination faster, but we are also much more susceptible to overreach the destination and get a bad approximation.
- T is the total number of small steps we will take. It directly decides how fast (or slow...) our algorithm will be. Again, if we take too few steps, then the algorithm might end before we reach our destination. But couldn't we just stop whenever we repeatedly see the approximately same number? Sadly, this seemingly convergence might result from that we are following a path that descends very slowly, and we could not tell if we have reached the optimal position yet. The good news, however, is that if we assume good enough properties of the function (convexity, Lipschitz, etc.), then we can calculate a theoretical upper bound of T, and this upper bound does not depend on the dimension of the input.

Thus, although the algorithm is extremely simple, **the secret of a good performance of the gradient descent actually lies in the clever choice of these parameters**. In practice, gradient descend often serve as a symbol of progress for the training of machine learning models: f would be a *loss function* that measures how poorly the model fits the training data, and the input of f are different parameter settings for the model. If we want our model to have better performances, it is natural to add a tons of parameters to reflect all aspects of the training data. This causes the dimension of the input of f to be very large.

2 Choosing the Gradient

As you might have notices, I placed a quotation mark when I said that the gradient gives the optimal descending direction. The formal definition of the gradient requires an isomorphism, often in the form of an inner product, between the tangent bundle and the cotangent bundle. Normally we would use the 2-norm, so the gradient gives the local optimal direction. However, if we have some prior knowledge of the distribution of training data, then we can modify the inner product to accelerate our training. For example, if we are doing gradient descend on $f(x) = 4x^2 + y^2$, whose level curves would be a family of ellipses, then the negative gradient would direct not to the origin (also the minimum point), but the major axis of the ellipse. Thus we would follow a zig-zag path along the major axis (since it is unlikely to stop exactly on the axis), repeatedly overshooting in one direction and then correcting it in the next iteration. If, instead, we take the gradient with respect to the inner product $\langle u, v \rangle = 2u_1v_1 + u_2v_2$ (suppose the major axis points to the y direction), we give more weight to changes in the x direction. This forces the x component to stabilize before the gradient descent even ends, so we would now descend along the y axis.

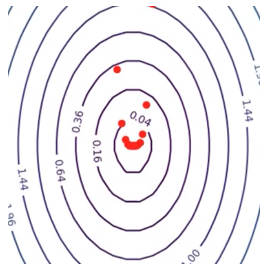

Figure 1: 2-norm

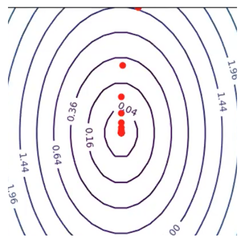

Figure 2: non-standard norm

3 The Dog Chasing Its Tail

A common dilemma in many computer science problems is that we want our model to tell us something about the input data, but at the same time we want to have prior knowledge of the input data so that our model will have much better performance. At certain points this all sounds self-contradictory, just like the classic question of whether the chicken came first or the egg came first. In the case of the gradient descent, we see this in the choice of parameters and also the use of non-standard inner products. Perhaps one day we will have a better way to accommodate this, but for now, the dog keeps on chasing its tail.