CSE 291C: Introduction

Lecturer: Hao Su

Scribed by Owen Jow on January 08, 2019

1 Course Overview

In this course, we will study the interplay between two broad topics: geometry and machine learning. In particular, we'll take a geometric approach to understanding ML algorithms, and use ML tools to analyze or synthesize geometry (e.g. 3D surfaces).

- As an example, we can view the ML problem of finding useful embeddings from a geometric perspective. In this sense, we're looking for a surface in high-dimensional space s.t. valid data points lie on the surface, and we usually want to avoid having an overly "folded" surface so that we can apply simple algorithms such as nearest neighbors or linear classifiers more effectively.

2 Numerical Tools for Optimization

We will formulate almost every problem as an optimization problem. Then, to solve the optimization problem, we will want to convert it to a more transparent linear algebra problem and solve that. In our studies, there are four main types of problems we will encounter: linear problems, unconstrained optimization problems (harder to solve than linear problems), equality-constrained optimization problems (harder to solve than unconstrained optimization problems), and variational problems.

2.1 Linear Problems

Recall the notion of a linear operator $\mathcal{L}$, which takes in a vector from a vector space and produces a vector in turn. By definition, it obeys the linearity properties:

$$\mathcal{L}[\mathbf{x}+\mathbf{y}] = \mathcal{L}[\mathbf{x}] + \mathcal{L}[\mathbf{y}]$$
$$\mathcal{L}[c\mathbf{x}] = c\mathcal{L}[\mathbf{x}]$$

An example of a linear operator is the first-order derivative operator $\mathcal{L}[f] = \frac{d}{dx}f$. In this case, let the vectors f be infinitely differentiable univariate functions $\in C^\infty(\mathbb{R})$.[1][2][3] Note that

$$\frac{d}{dx}(cf+g) = c\left(\frac{df}{dx}\right) + \frac{dg}{dx}$$

[1] http://mathworld.wolfram.com/C-InfinityFunction.html

[2] Note: with functions, addition and scalar multiplication are both properly defined (as should be the case with vectors). If we add two functions, or multiply a function by a scalar, we get another function of the same form.

[3] We can also easily interpret column vectors as functions (from an index to the value at that index).

The nth order derivative is also a linear operator.

If the vectors are of finite dimensionality, then linear operators can be represented as matrices. Accordingly, we end up with linear equations of the familiar form $\mathbf{Ax} = \mathbf{b}$. Naively we could use Gaussian elimination to solve these equations, but if $\mathbf{A}$ is $n \times n$ this results in a time complexity of $O(n^3)$. Inversion is even worse (unstable and slightly slower than Gaussian elimination).

Fortunately, $\mathbf{A}$ often exhibits some special structure that allows for a much quicker solution strategy. For example, consider the linear equation $\frac{d^2}{dx^2} f = g$ which uses the central difference approximation $f(x+h) - 2f(x) + f(x-h)$ of the second derivative and a discretized 1D space which represents the relevant domain of both functions (f and g).

We want to determine what f should be, evaluated at every discretized location on the 1D axis. We have g and the boundary values. From discretization, the matrix equation looks something like

$$\begin{bmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & -2 & 1 \\ 0 & \cdots & 0 & 1 & -2 \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix} = \begin{bmatrix} g_1 \\ \vdots \\ g_n \end{bmatrix}$$

Note that only the diagonal and adjacent entries have nonzero values. In general, if an $n \times n$ matrix only has nonzero entries in a diagonal band of width k (such as in this case), the solution time complexity should be $O(k^2 n)$. Note the reduced cost from before!

2.1.1 Non-Square Matrices

If $\mathbf{A}$ is tall (overconstrained), we should find the solution which minimizes $\|Ax - b\|^2$ (least-squares). If $\mathbf{A}$ is fat (underconstrained), we should solve for the least-norm solution such that $\mathbf{Ax} = \mathbf{b}$.

2.1.2 Solvers

There are two classes of solvers for linear systems:

- **Direct solvers.** In this case, we explicitly provide $\mathbf{A}$ and $\mathbf{b}$ and use a dense or sparse method to solve. The call would look like $\mathbf{x} = \texttt{solve}(\mathbf{A}, \mathbf{b})$.
- **Iterative solvers.** Here, we apply the matrix repeatedly to solve (e.g. the conjugate gradient method in the case of a positive definite $\mathbf{A}$). Now we just pass the function h (where $h(\mathbf{x}) = \mathbf{Ax}$) to our solver, i.e. $\mathbf{x} = \texttt{solve}(h, \mathbf{b})$.

2.1.3 Surfaces

If we want to solve differential equations or an optimization problem over a surface (e.g. of a mesh), we will have to discretize the surface (triangulation), decomposing it into many small triangles. Then we can apply a numerical technique such as the finite element method to solve for a function over the surface (determine a function value for all of the vertices).

Generally, the mesh will be sparsely connected and we'll end up with a sparse system.

2.2 General Problems

$$\min_{x \in \mathbb{R}^n} f(x)$$
$$\text{s.t. } g(x) = 0, h(x) \geq 0$$

The general form of an optimization problem involves the minimization of an objective (*energy*) function. To minimize the function, we will typically have to leverage gradient information.

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n} \right)$$

If we have a map (different input/output dimensionality) $f : \mathbb{R}^n \to \mathbb{R}^m$, we'll use the **Jacobian** Df.

$$(Df)_{ij} = \frac{\partial f_i}{\partial x_j}$$

where $j \in [1, ..., n]$ and $i \in [1, ..., m]$. It's just all the partial derivatives arranged in a matrix.

Geometrically, the Jacobian matrix characterizes how a small movement in the domain affects the codomain (it takes a small change in the input to the corresponding small change in the output). The matrix Df evaluated at $\mathbf{p}$ (for $\mathbf{p} \in \mathbb{R}^n$ a point in the domain) can be used to obtain the best linear approximation of f in the neighborhood of $\mathbf{p}$, i.e.

$$f(\mathbf{x}) = f(\mathbf{p}) + [Df(\mathbf{p})](\mathbf{x} - \mathbf{p})$$

If $f : \mathbb{R}^2 \to \mathbb{R}^2$, then $Df : \mathbb{R}^2 \to \mathbb{R}^2$.

There is also the **Hessian** for scalar-valued functions (an analogue of second-order derivatives). Often, we'll convert an optimization problem to a root-finding problem (find $\mathbf{x}$ where the gradient is 0), and the Hessian can be used for analysis here. A point with a zero gradient is called a *critical point*, and there are three manifestations:

- *Local minima*, where the Hessian is positive definite
- *Local maxima*, where the Hessian is negative definite
- *Saddle points*, where the Hessian is indefinite

The Hessian also allows for quadratic (not just linear) approximations.

2.2.1 Common Formulations

- *Least-squares.* We often want to minimize the L_2 norm of the residual vector $\mathbf{Ax} - \mathbf{b}$.
- $f(\mathbf{x}) = \|\mathbf{Ax}\|_2$, $g(\mathbf{x}) = \|\mathbf{x}\|_2 - 1$. The equality constraint makes this problem non-convex, but the solution actually just boils down to the eigenvalue problem $\mathbf{Ax} = \lambda\mathbf{x}$.

2.2.2 Uncommon Formulations

If we have some generic objective, the general approach is to try different solvers, moving from the simpler ones to the more complicated ones.

For the most part, we'll ultimately want to reduce the problem to a linear system.

3 Appendix

3.1 A 3D World

Let's think of objects as the heart of everything 3D – abstract building blocks for a limitless universe, each associated with little facts and figures: attributes, experiences, "knowledge." Here's a chair. That's 3D data. Here's a chair with its shape and its materials and knowledge of the last person to sit on it. That's 3D data. Here's another object, and it is similar in shape to the first chair; it too has legs and a back. It is also a chair. We knew something about the first chair, so we know something about the second one. We can *transfer* information according to relations based in similarity.

3.2 Surface Representation: Points

Just a bunch of (x, y, z) points.

3.3 Surface Representation: Parametric Surfaces

A curve through 3D space can be represented as $s(t) = (x(t), y(t), z(t))$ – a 3D point for every position on a line. A surface in 3D can be represented as $s(u, v) = (x(u, v), y(u, v), z(u, v))$ – a 3D point for every position on a plane.

A Bezier curve is a parametric function $s(t) = \sum_{i=0}^{n} \mathbf{p}_i B_i^n(t)$ which is the linear combination of control points $\mathbf{p}_i$, where the weights $B_i^n(t)$ are parametric basis functions. A Bezier *surface* $s(u, v)$ is a linear combination of control points in two directions, parameterized by variables over a plane.