Regression on Ice: Function approximation for the mathematically-inclined glaciologist.

Introduction

The rise of satellite-based remote sensing in the last couple of decades has revolutionized the way we analyze the earth and its dynamics. Some prominent examples of NASA's remote sensing programs include:

- The Gravity Recovery and Climate Experiment (GRACE), which measured variations in the Earth's gravitational field and gave new insight into the changing mass landscape of ice sheets and groundwater supply across the world.
- The Landsat mission (9 iterations, so far), which has provided high-resolution images of earth's landscape for over fifty years.
- The IceSAT mission (2 iterations, so far), which has given high-accuracy laser altimetry measurements, particularly in Greenland and Antarctica.

These programs have given us a wealth of data on the giant ice sheets of Greenland and Antarctica. In particular, they have given us a penetrating insight into the melting of the ice and the subsequent rising of sea levels.

With great volumes of data comes great responsibility. In my opinion, it requires a solid understanding of the mathematical foundations of data science. In this writeup, I would like to exposit on these foundations, in the context of modern research on ice sheet modeling. I focus on the following four topics.

- Problem Formulation and Basic Ideas in Regression
- Learning Linear Hypothesis Classes
- Gaussian Process Regression and Kriging
- On Hilbert Spaces, Smoothness Spaces, and Kernels

Each chapter features both examples from relevant glaciology research and mathematical "curios" which invoke some higher-level themes in the theory of statistics and computation.

Contents

1 Motivation

Modern analysis of the ice sheets often relies on piecing together a series of satellite measurements to create some approximate picture of the surface and its dynamics. Consider, for instance, **the SERAC model**, which used IceSAT's six years of laser altimetry measurements to create a rasterized (i.e. pixel-based) surface model of Greenland [5]. Here's how it works:

- SERAC's spatial regression works by identifying crossover areas traced out by the satellite tracks, fitting a surface polynomial $p(x, y)$ to each of these areas, and then piecing them together for the final image. Note that these polynomials are usually degree 3, but are set to higher degree on an ad hoc basis to cover important, complicated surfaces like the Helheim and Upernavik glaciers.

- This is combined with temporal regression at the crossover points, which is assumed (mostly for simplicity) to be linear. This seems to provide space for significant improvement. We refer the reader to newer work such as the ALPS algorithm, which uses penalized splines to model complex time series of ice elevations more faithfully [6].

- They made volume change estimates, but one should note that this involved a lot more than volume integrals. They had to take into separate models for *glacial isostatic adjustment*—the earth rebounding after the weight of ice is removed—and *firn densification*—the process by which fresh snow compacts into ice. They also had to make error estimates, which they did using a *bootstrapping* technique—creating estimates based on subsets of the data, and then taking the variance of those estimates.

We make these points to illustrate that practical modeling takes a lot of flexibility and attention to detail. Data does not bend to our every assumption. Good models require both domain expertise and a mastery of the mathematical toolbox. As a mathematician-in-training, I can speak much better to the latter.

In this writeup, we abstract away the nitty-gritty issues of scattered measurements, multiple sources of error, and nontrivial ice sheet physics. We are focused on the developing not just the toolbox function approximation but an appreciation for all the mathematical insights along the way. We dive into progressively more theoretical material as the pages go by.

2 Basic Ideas in Regression

What is function approximation? Well, it's important to first decide what kinds of functions you are interested in. If the function maps into a finite set, we call this learning problem *classification*. If it maps to real numbers, we call it *regression*[1]. Since we are usually worried about real-valued inputs in glaciology (such as temperature, flow velocity, and altitude) let's formulate things in terms of regression.

2.1 Problem Formulation

Here is a condensed and somewhat informal formulation of the regression problem.

We want to estimate a real-valued function, say $f : [0,1]^d \to \mathbb{R}$ (perhaps with a function that is "simpler" in some sense). Importantly, instead of having full access to the function, we are given samples $\{(x_i, y_i)\}_{i\in[n]} \subset [0,1]^d \times \mathbb{R}$. We want to use this information find a function $\hat{f}$ which best approximates f.

There are a number of ways to express closeness of approximation. A simple way might be the mean squared error (MSE) between the predicted function and the true function, at the given sample points:

$$L(\hat{f}, f) = \frac{1}{n}\sum_{i=1}^{n}(\hat{f}(x_i) - f(x_i))^2$$

This is a proxy for what we *really* want to know, which is the difference between the functions over not just the given points but the entire domain.

$$L(\hat{f}, f) = \int_{[0,1]^d} \Big(\hat{f}(x) - f(x)\Big)^2 dx = ||f - \hat{f}||_2^2$$

Note that $||\cdot||_p = \Big(\int (\cdot)^p \, dx\Big)^{1/p}$ denotes the p-norm (2-norm is the usual Euclidean distance, 1-norm is Manhattan distance, and so on).

There are two important and not necessarily orthogonal dimensions to this problem which I want to point out: the underlying statistical process (i.e. the sampling procedure) and the assumptions we make on our

[1] If you map into a countably infinite set like $\mathbb{N}$ (i.e. somewhere between finite and the uncountably infinite $\mathbb{R}$), you probably treat the problem more like classification, but I suppose this is a matter of taste.

2.1.1 The sampling distribution

The process by which our samples are drawn is rather important. We often assume for the sake of simplicity that the samples are *independent and identically distributed*, like coin flips or drawing cards with replacement.

If we knew the underlying distribution which determines how our samples are drawn, we could think about loss function in terms of an "expected" squared error, where the expectation is over the randomness in the sampling process $\mathcal{D}$.

$$L(\hat{f}, f) = \mathbb{E}_{(x_i,y_i)\sim\mathcal{D}}[(\hat{f}(x_i) - y_i)^2] = \int_{[0,1]^d\times\mathbb{R}} (\hat{f}(x_i) - y_i)^2 \, d\mu_{\mathcal{D}}$$

While this loss is a good statistical lodestar, it is often impractical to calculate since $\mathcal{D}$ and its corresponding measure $d\mu_{\mathcal{D}}$ is almost always unknown.

The nice thing about this abstraction, however, is that it allows us to formulate an *optimal* regressor, sometimes referred to as the *Bayes optimal regressor.*

$$f^*(x) = \mathbb{E}_{(x_i,y_i)\sim\mathcal{D}}\Big[y_i | x_i = x\Big]$$

Without getting too much in the weeds of the highly non-trivial mathematics of conditional expectation (Cf. Radon-Nikodym Theorem)

It is optimal in the sense that, compared to any other regressor $\hat{f}$, it has a lower *expected* loss.

$$\mathbb{E}_{(x_i,y_i)\sim\mathcal{D}}\Big[(f^*(x_i) - y_i)^2\Big] \leq \mathbb{E}_{(x_i,y_i)\sim\mathcal{D}}\Big[(\hat{f}(x_i) - y_i)^2\Big]$$

Two important remarks, regarding this notion of optimality. Note that, in practice, it is very possible that on some finite set of test points, $\hat{f}$ has a lower mean squared error of f^*. In the limit, however, due to the strong law of large numbers, the mean squared error approaches the actual expectation.

We leave the proof of the optimality of the Bayes regressor for the appendix. Note that there is a corresponding concept in classification known as the Bayes classifier.

2.1.2 Hypothesis classes: the simpler, the better

Even if you have access to the entire function—not just samples—the function approximation problem is interesting, so long as you somehow restrict the set of functions with which you approximate.

The set of functions from which we choose $\hat{f}$ is usually called the **hypothesis class**, often denoted $\mathcal{H}$. We are looking for the *best possible approximation* attainable

by a given hypothesis class, which we write as the following functional optimization problem. Note that $\arg\min_{x \in X} g(x)$ means that we output the x which minimizes the function $g(x)$ (whereas $\min_{x \in X} g(x)$ would simply give us the minimum value attained by g)[2].

$$f^* = \arg\min_{\hat{f} \in \mathcal{H}} L(\hat{f}, y)$$

Intuitively, a hypothesis class is more powerful when it is small but is able to approximate many kinds of functions. Oftentimes, we use knowledge about the application domain to determine a suitable hypothesis class. For instance, if we are modeling some kind of weather pattern, we might be inclined to use some set of functions with inherently periodic behavior, such as sinusoids.

$$\mathcal{H} = \{h \text{ such that } h(x) = \sin(A_1 x_1 + b_1) + ... + \sin(A_n x_n + b_n) + c\}$$

Alternatively, we might suspect that the dependent and independent variables have a simple linear relationship, and so we work with the set of affine functions.

$$\mathcal{H} = \{h \text{ such that } h(x) = A_1 x_1 + ... + A_d x_d + b\}$$

If $d = 2$ (e.g. we are working with surfaces), we can specify the hypothesis class of functions which are degree 2 or below polynomials in the input coordinates:

$$\mathcal{H} = \{h \text{ such that } h(x) = A_1 x_1^2 + A_2 x_2^2 + A_3 x_1 x_2 + b\}$$

We note briefly that this is the hypothesis class used to model patches of Greenland in the aforementioned SERAC model [5].

Notice how in the last two examples, we can describe each of the functions in the hypothesis class in terms of a few real-valued parameters, (e.g $A_1, ..., A_n, b \in \mathbb{R}$). Naturally, we call this *parametric* regression. You can also approach regression using *non-parametric* methods, which tend to fill in the gaps of a function based on local information rather than global structural assumptions. Two remarks on this:

- The line between parametric and non-parametric is not always clear. You might hear of splines (piecewise polynomials) as a type of *semi-parametric* regression, due to their localized nature [4]. It's sort of a matter of taste.

[2]Since we are working with real-valued functions, you might ask: what if $g(x)$ does not attain a minimum? For instance, $\min_{n \in \mathbb{N}} \frac{1}{n}$ does not exist (the infimum, meanwhile, is zero). The simple answer is: we restrict our attention to optimization problems which do not have this issue. So, technically speaking, there are some restrictions to the combinations of objective functions and domains we can pick.

- Non-parametric techniques are generally more flexible and have better convergence properties than parametric techniuqes, because they make fewer assumptions about the intrinsic structure of the data. The tradeoff is that non-parametric methods require more data. There is a sort of conservation of information principle at play.

2.2 Non-parametric Regression

The classic non-parametric regression technique is k**-nearest neighbors**. The idea is simple: given a test point $x_{\text{test}} \in [0,1]^d$, we assign y_{test} to be the average of the k nearest training points. If we are measuring closeness using the Euclidean norm, then, in a mix of math notation and pseudocode, we may write the following.

$$\hat{f}(x_{\text{test}}) = y_{\text{test}} = \frac{1}{k} \text{ sum}\Big(\text{k arg} \min_{x \in X_{\text{train}}} ||x_{\text{test}} - x||_2\Big)$$

Rather than taking a hard cutoff of nearest neighbors, we could use a softer, weighted scheme. In this scheme, every training point makes an impact, but closer points make more of an impact than points which lie further away. For instance, we might model locality in terms of a Gaussian distribution, where the parameter h denotes the variance.

$$\hat{f}(x_{\text{test}}) = \frac{\sum_{x \in X_{\text{train}}} y_i \cdot \exp(-||x_{\text{test}} - x||^2/2h^2)}{\sum_{x \in X_{\text{train}}} \exp(-||x_{\text{test}} - x||^2/2h^2)}$$

This is an example of so-called **kernel regression**, where the kernel refers to the function that mediates our weighted average. The above example uses the Gaussian kernel, but we can easily abstract this to some function K with a locality-controlling parameter h:

$$\hat{f}(x_{\text{test}}) = \frac{\sum_{x \in X_{\text{train}}} y_i \cdot K_h(||x_{\text{test}} - x||)}{\sum_{x \in X_{\text{train}}} K_h(||x_{\text{test}} - x||)}$$

Kernel regressors have nice convergence properties. It makes sense why: the only assumption they impose on the data is continuity (i.e. points that are close together in the input space are mapped close together in the output space).

. For instance, there are results which suggest that, as the window size shrinks ($h \to 0$) and the number of random samples goes to infinity ($n \to \infty$), but at a faster rate than the shrinkage of the window ($hn \to \infty$; if the window shrunk faster, then there might be no points in the window to make predictions), then the predictor $\hat{f}$ converges to some sort of optimal regressor. How we quantify this optimality is a

whole other discussion. This is known as a **consistency result**: it establishes that our model does the right thing as we have more data points to work with.

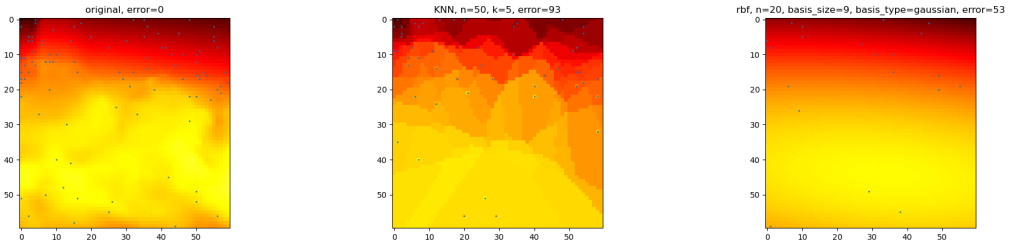

Figure 1: Regression of sea surface temperatures. Left is the original image, middle is k-nearest neighbors, right is kernel regression with Gaussian kernel.

2.3 On the word *kernel*

There are a few fundamentally different and important concepts that the word *kernel* expresses across mathematics.

1. (Algebra) The kernel of a linear map $f : V \to W$ is the set of elements in V which are mapped to the zero element in W. Symbolically, $\ker(f) = \{v \in V : f(v) = 0\}$. The same idea is at play when you describe the kernel of a group homomorphism or a morphism of categories.

2. (Functional Analysis) In a Hilbert space $\mathcal{H}$ (i.e. a vector space of functions with some sense of an inner product), a kernel $k : \mathcal{H} \times \mathcal{H} \mapsto \mathbb{R}$ is simply a symmetric, positive-semidefinite map. Famously, every kernel is "native" to a (often infinite-dimensional) reproducing kernel Hilbert space. See chapter 4 for more details.

3. (Statistics) In the loosest use of the word, a kernel is just a non-negative "window function" which is zero or near-zero outside a small neighborhood. A kernel is sort of like a "measuring device" you can apply to a function to get local information. You usually apply it using an operation called *convolution*.

The last definition is usually the one lurking in the shadows of machine learning terminology such as the aforementioned kernel regression and the so-called "kernel trick." But we will see later how important the function analysis type kernel is. And if you have ever taken linear algebra, you will appreciate the importance of kernels.

2.4 Curio: Voronoi, Delaunay, and Duality

Suppose $d = 2$ (i.e. our inputs are points on a grid) and $k = 1$ (we are only assigning things to the nearest neighbor). Then the nearest neighbor assignments create a tesselation of the plane, where each tile is the neighborhood of a given training point.

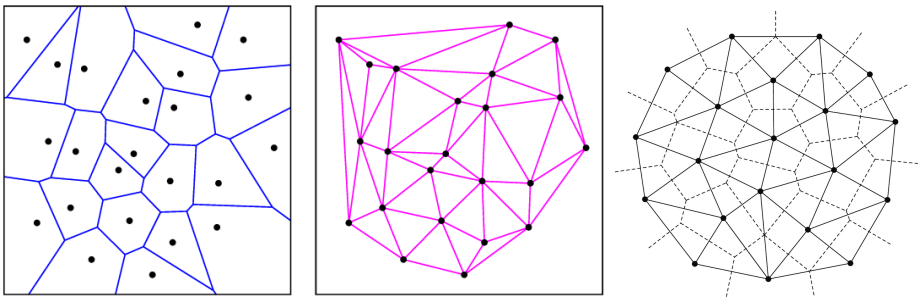

Figure 2: The Voronoi diagram (left) is the tesselation created by 1-nearest neighbors. If we view it as a graph then, the Delaunay triangulation (middle) is its dual. On the right, we show the two superimposed.

If we are using Euclidean distance, then each tile is a convex polytope (does not cave in on itself and has flat edges). If you view the straight-line edges of these tiles as edges of a graph, then (unsurprisingly) the Voronoi diagram is a graph.

Now forget about nearest neighbors for a moment and consider a different problem over our set of test points on the plane: say we want to connect the dots in such a way to form a *triangulation* (i.e. each face of the graph is a triangle). Say we want this triangulation to maximize the smallest angle in this triangulation, i.e. avoid sliver-like triangles. The winner of this optimization problem is the so-called **Delaunay triangulation**.

It turns out that the Delaunay triangulation is the **dual graph** of the Voronoi diagram, in the sense that the faces of the Voronoi graph are the nodes of the Delaunay graph, and corresponding edges cross each other (at ninety degrees in the precise geometric construction). See the illustration above.

It is somewhat remarkable that these different optimizations have this sort of symmetry between them. The term mathematicians use for this kind of behavior is called *duality*. It is an exciting theme that shows up all across mathematics. We note the following examples.

- The dual V^* of a vector space V, which is the set of linear maps from V to its underlying base field.

- The dual of a constrained optimization problem, which, under certain conditions (e.g. Slater's Conditions for convex optimization) is equivalent to the original, primal formulation.
- The duality of open and closed sets in topology; the time and frequency domains in Fourier analysis; intermediate field extensions and subgroups of the corresponding Galois group; kernel and image of group homomorphisms; and so and so forth.

Keep your eyes peeled for duality. It's quite beautiful.

3 Linear Hypothesis Classes

In this chapter, we discuss the standard front-line approach to function approximation: linear regression. After recalling some elementary linear algebra, we explain the standard solution to linear regression. Then we discuss the simple tweaks one makes to upgrade this method in order to run polynomial, spline, and radial basis function regression. Finally, we discuss error estimation techniques and hyperparameter tuning for the more advanced techniques.

3.1 Lines and Linearity

Intuitively, we all know what a line is. The first descriptor that may come to mind is: straight. Lines are straight. They also seem to extend in both directions, as opposed to their one-track counterparts, the rays, or their finite cousins, the line segment.

If you are a bit more descriptive, you might say that a line $f : \mathbb{R} \mapsto \mathbb{R}$ is a function that can be written $y = mx + b$. It turns out, however, that the better word for this kind of function is *affine*. If we are being precise, we reserve the word linear for functions which cross the origin. In other words, we have:

$$\text{(affine) } f(x) = mx + b \qquad \text{(linear) } f(x) = mx$$

If we are generalizing this to higher-dimensional functions $f : \mathbb{R}^d \mapsto \mathbb{R}$, then we have, for $A \in \mathbb{R}^{1\times d}$ and $b \in \mathbb{R}$:

$$\text{(affine) } f(x) = Ax + b \qquad \text{(linear) } f(x) = Ax$$

If the function is $f : \mathbb{R}^d \mapsto \mathbb{R}^n$, then it would be the same except $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$. The pattern should be clear. And for most scientific applications, this conception of the line is sufficient. But from a mathematical perspective, it is more of a description than a real definition.

Here is a more mathematically palatable definition of a line.

Definition: A function $f : \mathbb{R} \mapsto \mathbb{R}$ is a *linear map* (or *line*) if it satisfies:

1. (homogeneity) $f(ca) = cf(a)$ for all $a, c \in \mathbb{R}$.
2. (additivity) $f(a) + f(b) = f(a + b)$ for all $a, b \in \mathbb{R}$

Together, we call these two properties *linearity*.

You might think this is more than enough to qualify such a simple concept. But, if you are familiar with linear algebra, you will know that there is a much richer story at play, where the main characters are *vector spaces.* I will provide two descriptions of vector spaces: one in lay terms and the other in the form of an algebraic definition.

- A vector space V is a set of abstract items (lists or tables of numbers, functions, et cetera). We call these items vectors. I don't know much about them, but I do know this: if I add two vectors, it's still a vector: $v_1 + v_2 \in V$. If I multiply a vector by a real number, it's still a vector: $cv \in V$. There is a zero vector, such that $\vec{0} + v = v$ for all $v \in V$.
- More precisely, a vector space V over a field (of scalars) F has two operations: vector addition $+ : V \times V \mapsto V$ and scalar multiplication $+ : F \times V \mapsto V$. V is an abelian (i.e. commutative) group under vector addition, and scalar multiplication is a left ring action on the group of vectors. (If F were a ring instead of a field, then we call this set a module rather than a vector space).

3.1.1 Vector Spaces of Functions

I make this point of discussing vector spaces because many interesting classes of functions that we want to use for regression form vector spaces over the field of real numbers. Here are the prominent examples $\mathbb{R} \mapsto \mathbb{R}$.

- **Polynomials** are linear combinations of monomials. Polynomials of degree d are parameterized by a vector of coefficients $(a_1, ..., a_d)$.

$$f(x) = \sum_{i=0}^{d} a_i x^i$$

  The space of polynomials of arbitrarily large degree form an infinite-dimensional vector space. If we restrict the dimension, it becomes finite-dimensional.

- **Sinusoids** are linear combinations of shifted and scaled sine functions (or cosines: same difference). We can parameterize them in terms of a set of amplitudes, frequencies, and shifts $\{(a_i, c_i, t_i)\}_{i \in [n]}$.

$$f(x) = \sum_{i=0}^{n} a_i \sin\Big(c_i(x - t_i)\Big)$$

- **Splines** are piecewise polynomials, with "break-points" parameterized by a so-called *knot vector* $(t_1, ...t_m)$. We can write them somewhat perversely as follows, where $\mathbf{1}_{[a,b]}(x)$ is the indicator function of $x \in [a, b]$.

$$f(x) = \sum_{i=0}^{n} \sum_{j=0}^{m} a_{ij} x^i \mathbf{1}_{[t_i, t_{i+1})}(x)$$

The nicer way of writing out a basis for splines is to use the so-called Cox-de Boor recursive formula.

- **Radial basis functions** (RBFs) are linear combinations of some radially symmetric function $\phi(x)$. They are parameterized by function centers $(t_i)_{i\in[n]}$, e.g.

$$f(x) = \sum_{i=0}^{n} a_i \phi_i(x) \qquad \phi_i(x) = \lambda \exp(-||x - t_i||^2)$$

3.1.2 Universal Approximation

The fact that these are vector spaces make them amenable to least squares linear regression. But there is a deeper reason that these function spaces are good candidates for function reconstruction, and it has to do with a concept known as *universal approximation.*

Essentially, any continuous function $f : [a, b] \mapsto \mathbb{R}$ can be approximated, with arbitrarily good precision, by these kinds of functions. The classical result, to this end, is the Weierstrass approximation theorem, stated below.

Theorem 1 *(Weierstrass, 1885) If $f : [a, b] \mapsto \mathbb{R}$ is continuous, then for all $\epsilon > 0$, there exists a polynomial $p(x)$ such that:*

$$||f - p||_\infty = \sup_{x\in[a,b]} |f(x) - p(x)| < \epsilon$$

The Stone-Weierstass theorem famously allows us to generalize this to other spaces of functions, like Fourier series, which are linear combinations of sinusoids.

However, even if a space of functions has universal approximability, it might not be all that good at function approximation in practice. This is because, to meet a given precision level, you may need to use very high degree polynomials or very high-frequency sinusoids. However, in practice, when fitting functions, we cannot work with the entire function space. We specify a a maximum degree for our polynomials and a maxinum frequency of our sinusoids. We bound the number of knots in our splines and the number of centers we use for our radial basis function. If we didn't, then our code would never terminate.

With that humbling reminder, let us descend from the cloud of the continuous world, back into a discussion of more practical, discrete computation.

3.2 Ordinary Least Squares

If your hypothesis class consists of affine functions, then finding the best least-squares fit from inside this hypothesis class is famously easy. Recall that we are learning a function $[0,1]^d \mapsto \mathbb{R}$. It is affine in its input $x = (x_1, ..., x_d) \in [0,1]^d$ if:

$$h_\omega(x) = \sum_{j=1}^{d} x_j \omega_j$$

We can compute how well it does on the training set $\{(x_i, y_i)\}_{i\in[n]} \subset [0,1]^d \times \mathbb{R}$, by adding up the squared error accumulated by the test function on each (x_i, y_i) pair.

$$L(h_\omega, y) = \sum_{i=1}^{n} \Big(h_\omega(x_i) - y_i\Big)^2 = \sum_{i=1}^{n} \Big(\sum_{j=1}^{d} x_{ij}\omega_j - y_i\Big)^2$$

We can express this in matrix notation as follows, where $X \in \mathbb{R}^{n\times d}$ encodes all of the inputs as row vectors, and $y \in \mathbb{R}^n$ has the dependent variable.

$$L(h_\omega, y) = ||X\omega - y||^2$$

We want to minimize this function with respect to $\omega \in \mathbb{R}^d$, the parameterization of h. We call this the **least-squares problem**.

$$\arg\min_{\omega\in\mathbb{R}^d} ||X\omega - y||^2$$

Without much thinking, we could look for stationary points $\frac{\partial L}{\partial \omega} = 0$ and hope for the best. The outcome of this computation is as follows.

$$\frac{\partial}{\partial \omega}||X\omega - y||^2 = X^T(X\omega - y) = 0$$

$$\implies \boxed{\hat{\omega} = (X^TX)^{-1}X^Ty}$$

It turns out that, if X^TX is indeed invertible[3], then this is the square-loss-minimizing ω. The mathematical argument for this requires a bit of work: you need to show that L_ω is convex in ω. Then we apply a well-known property of convex functions: namely, that all local minima are global minima.

[3]If it is not invertible, then you can use the Moore-Penrose pseudo-inverse.

3.2.1 On Convexity and Optimization

Convexity (like linearity) is an example of remarkably well-behaved mathematics. When optimization problems exhibit convexity, they make our lives a lot easier. Let us introduce convexity from the ground up.

Recall that a *linear combination* of vectors $x_1, ..., x_n$ is of the following form, where $\lambda_1, ..., \lambda_n \in \mathbb{R}$ are the coefficients.

$$\lambda_1 x_1 + \lambda_2 x_2 + ... + \lambda_n x_n$$

Linear combinations of one vector form a line. Linear combinations of two (independent) vectors form a plane. And so on.

There are special types of linear combinations which form more restrictive spaces.

- A *conic combination* is a linear combination such that $\lambda_1, ..., \lambda_n$ are non-negative. As the name suggests, these combinations form a space that looks like a cone.
- A *convex combination* is a linear combination such that the $\lambda_1, ..., \lambda_n$ are non-negative and they add up to one, i.e. $\sum_{i=1}^{n} \lambda_i = 1$. The space created by the convex combination of $x_1, ..., x_n$ is called a *convex hull*.

Now we introduce two concepts that play with the idea of convex combinations.

Definition 1 *A convex function $f : \mathbb{R} \mapsto \mathbb{R}$ is such that, if you draw a line between any two points on the function, that line lies above the function graph. In other words:*

$$\forall x, y \in \mathbb{R} \ , \ \lambda \in (0, 1) \qquad f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda) f(y)$$

The image you should have in mind is a parabola, but be aware that convex functions come in a number of shapes and sizes. Importantly, affine functions are convex (and in their case, the convexity inequality is tight).

Definition 2 *A convex set $S \subset \mathbb{R}$ is a set which is closed under convex combinations. Visually speaking, if you draw a line connecting any two points in the set, the line lies within the set*[4]*. Symbolically:*

$$\forall x, y \in S \ , \ \lambda \in (0, 1) \qquad \lambda x + (1 - \lambda)y \in S$$

[4]So we could have alternatively said that a convex function is such that the space above the function $\{(x, y) : y \geq f(x)\}$ is a convex set.

Each of these concepts are important to convex optimization. Without getting into the weeds of optimization theory, I would like to point out that there are two ingredients to any optimization problem:

- *The objective function* which we are trying to minimize or maximize.
- *The feasible set.* This is the set from which we choose our parameters.

In the case of linear regression, the objective is $f(\omega) = ||X\omega - y||^2$, and the feasible set is the whole space $\mathbb{R}^d$. This is known as an *unconstrained* optimization problem. If we constrained the feasible set to, say $[0,1]^d$, then this is a *constrained* optimization problem.

3.2.2 Maximum-Likelihood Formulation

A tell-tale sign of good mathematics is what I would call *robustness under change in perspective*: there are many ways of looking at the same thing. The least squares problem is a shining example of this.

You may have seen the generative (or Bayesian, if you will) formulation of least squares, which proceeds roughly as follows.

Assume that your points $\{(x_i, y_i)\}_{i \in [n]}$ are noise-perturbed samples of an actual linear function. Importantly, we assume that the noise terms are all independent, identically distributed Gaussians.

$$\forall i \qquad y_i = \omega \cdot x_i + \epsilon_i \qquad \epsilon_i \sim \mathcal{N}(0,1)$$

Then, as the argument goes, it makes sense (computationally and intuitively) to maximize the following quantity, often denoted as the *likelihood*.

$$\arg\max_{\omega} \ \mathbb{P}(y_1, ..., y_n \mid \omega)$$

This a conditional probability: we read the objective function as "the probability of the observations $y_1, ..., y_n$, given some setting of the parameters ω."

The claim is the following: given our assumptions about the noise

$$\arg\max_{\omega} \ \mathbb{P}(y_1, ..., y_n \mid \omega) = \arg\min_{\omega} ||X\omega - y||^2$$

We leave the proof for the appendix. Some interesting questions you might ask, after reading the proof, are: what if we change the noise structure? Is there still an equivalence here?

3.3 Generalizations

There are a number of ways to adapt the technique of linear regression. First of all, we can fit all kinds of functions to data, rather than just straight lines. Secondly, we can apply penalties to enforce certain kinds of structure on our estimate.

3.3.1 Linear Hypothesis Classes

We can use least squares regression to fit not just lines but polynomials, splines, and sinusoids to all kinds of data. How we do this requires a clever transformation of the input matrix X. We build up the intuition for this carefully.

Recall that, if we are learning a function $\mathbb{R} \mapsto \mathbb{R}$, then linear regression looks particularly simple. We are finding the coefficients ω which allow us to express y as a linear combination of sampled points $x_1, ..., x_m \in \mathbb{R}$.

$$X\omega - y = \begin{pmatrix} x_1 & \dots & x_m \end{pmatrix} \begin{pmatrix} \omega_1 \\ \vdots \\ \omega_m \end{pmatrix} - \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

If we are learning a function $\mathbb{R}^d \mapsto \mathbb{R}$ (this is sometimes called ordinary *multi-linear regression*), then we are learning to express y as a linear combination of sampled *vectors* $x_1, ..., x_m \in \mathbb{R}^d$.

$$X\omega - y = \begin{pmatrix} | & & | \\ x_1 & \dots & x_m \\ | & & | \end{pmatrix} \begin{pmatrix} \omega_1 \\ \vdots \\ \omega_m \end{pmatrix} - \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

Learning a to express a function $\mathbb{R} \mapsto \mathbb{R}$ in terms of, say, polynomials up to degree m, looks very similar to multi-linear regression. Say we have three test points $x_0, x_1, x_2 \in \mathbb{R}$. Since our basis functions are effectively $\{1, x, x^2, ..., x^m\}$, each column of the matrix is an application of the basis function to the vector (x_0, x_1, x_2).

$$X\omega - y = \begin{pmatrix} 1 & x_0 & x_0^2 & \dots & x_0^m \\ 1 & x_1 & x_1^2 & \dots & x_1^m \\ 1 & x_2 & x_2^2 & \dots & x_2^m \end{pmatrix} \begin{pmatrix} \omega_1 \\ \vdots \\ \omega_m \end{pmatrix} - \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

More generally, we can fix any finite basis $\{\phi_1(x), ..., \phi_m(x)\}$ and run least squares on the following quantity.

$$\begin{pmatrix} | & & | \\ \phi_1(x) & \dots & \phi_m(x) \\ | & & | \end{pmatrix} \begin{pmatrix} \omega_1 \\ \vdots \\ \omega_m \end{pmatrix} - \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

Whether those ϕ's are splines, polynomials, sinusoids, or some mix theoreof, is effectively a design choice. The basis functions you choose also can be multi-dimensional. Once should just note that the size of the feature matrix X grows faster when you do fancy basis functions in high dimensions.

3.3.2 Regularization

As we mentioned in the first chapter on regression, it is important that our approximations are as simple as possible. This is not only intuitive but backed by certain theoretical results (Cf. Ockham's Razor theorem, in computational learning theory).

The process of enforcing simplicity in our regression estimates is often referred to as *regularization*. In the context of linear regression, there are two canonical types of regularization, referred to as ridge and lasso regression.

We identify these types based on th

$$\textbf{Ridge (L2)} \qquad \arg\min_{\omega} ||X\omega - y||^2 + \lambda||\omega||_2^2$$

$$\textbf{Lasso (L1)} \qquad \arg\min_{\omega} ||X\omega - y||^2 + \lambda||\omega||_1$$

Spline basis. Penalty. Generalized cross-validation.

$$\arg\min_{\Theta} \left(||X\omega - y||^2 + \Theta^T P \Theta\right) \qquad P = \lambda \Delta_q^T \Delta_q$$

[2]

You can imagine more complicated

3.4 Curio: Free Knot Linear Splines and Neural Networks

As we have seen, spline regression with fixed knots is easily solved using least-squares regression. However, this masks a highly non-trivial aspect of spline fitting. If you try to optimize the *placement of the knots*, you find yourself mired in a rather complicated, non-convex optimization procedure, with many potentially poor local minima, captured by David Jupp calls the "lethargy theorem" [3].

[1]

Definition 3 *A neural network $N(W, L, d, d')$ of width W, length L, input dimension d, and output dimension d' is a function $\mathbb{R}^d \mapsto \mathbb{R}^{d'}$ defined via the following recurrence relation, where x_0 is the input, x_{L+1} is the output.*

$$x_k = (W_k x_{k-1} + b_{k-1})_+ \qquad 1 \leq k \leq L+1$$

This definition merits some explanation:

- Note that this is only well-defined for W_k and b_k of certain dimensions, which are constant the first as last layer.
- We could define neural nets more broadly to have varying

Let $Y^{W,L}(d, d')$ denote the set of neural networks with these specificiations. Let Σ_T denote the set of continuous piecewise linear functions (AKA free knot linear splines) with D breakpoints (i.e. D discontinuities in the derivative).

(DeVore 2019) showed that, for $L = 1$ single hidden layer ReLU nets, we have the following *strict* inclusions.

$$\Sigma_{W-1} \subset Y^{W,1} \subset \Sigma_W$$

In general, we have that the neural network functions we are working with are continuous piecewise linear functions. Furthermore, we can upper bound how many "pieces" we partition the input space.

Theorem 3.3 (DeVore 2020): Every $f \in Y^{W,L}(d, d')$ is a continuous piecewise linear function subordinate to a polytope partition with at most 3^{WL} cells.

The proof uses the idea of activation patterns. The polytope partition is derived from zones where the $m = WL$ hidden layer neurons are either positive, negative, or zero. For $\nu \in -1, 0, 1^m$ and pre-activations of each neuron $z(x) = (z_k(x), k \in [n])$:

$$\Sigma_\nu = \{x \in \mathbb{R}^d : \text{sgn}(z_k(x)) = \nu\}$$

Since there are at most 3^n possible ν with Σ_ν non-zero measure, this is our upper bound on the size of the partition. We can trivially reduce this to 2^n and have almost-everywhere subordination to a polytope partition (since the contribution of the zero activations is measure zero).

Problem (DeVore 2020): Characterize the continuous piecewise linear functions that belong (and don't belong) to $Y^{W,L}$. Perhaps begin with considering only those piecewise linear functions which have breakpoints at the dyadic rationals.

- Furthermore, can we characterize the distribution of the breaktpoints in some way? Even just first moments?

4 Gaussian Processes

One of the biggest problems in regression is finding out how best to express the error of our estimates. If we model our predictions as a family of inter-related random variables, we tend to have a much easier time quantifying uncertainty. This is the basic motivation of Gaussian process regression.

We recall some basic ideas in probability theory.

- A *random variable* is a function. In particular, it is a measurable function (generalization of a continuous function, in some sense) associating subsets of some set Σ (the "state space") to their probabilities in $\mathbb{R}$. A simple example might be a Rademacher random variable $f : \mathcal{P}(\{-1, 1\}) \mapsto \mathbb{R}$, with $f(\{-1\}) = f(\{1\}) = 1/2$ and $f(\{-1, 1\}) = 1$.
- The Gaussian is a continuous random variable (i.e. its state space looks like subsets of the real number line, an uncountable set). Thus, we use integration (rather than summation) in order to describe how it measures these sets. For instance, in order to find the probability that a Gaussian random variables lies in the set $[-3, 3]$, we write

$$\int_{-3}^{3} p_{\mu,\sigma}(x)dx = \int_{-3}^{3} \Big[\frac{1}{\sqrt{2\pi\sigma^2}} \exp(-|x-\mu|^2/2\sigma^2)\Big]dx$$

  where $p_{\mu,\sigma}(x)$ is the is Gaussian's *probability density function*. μ is the mean. σ^2 is the variance. The Gaussian is special for a number of reasons. Most prominently, it is the limiting distribution in the Central Limit Theorem.
- The multivariate Gaussian has a similar-looking probability density function $p_{\mu,\Sigma} : \mathbb{R}^d \mapsto \mathbb{R}$.

$$p_{\mu,\Sigma}(\vec{x}) = \frac{1}{(2\pi)^d} \det(\Sigma)^{1/2} \exp((x-\mu)^T \Sigma (x-\mu))$$

  where $\mu \in \mathbb{R}^d$ is the mean and $\Sigma \in \mathbb{R}^{d\times d}$ is the covariance matrix.

In this section, we first discuss Kriging, a prominent example of Gaussian process regression in geostatistics. Then we dive into the more general formulation of things.

4.1 Kriging

Kriging interpolation was developed in the context of a search for gold. Since then it has become the standard in geostatistical surface estimation.

The mathematical formulation of it all should not distract from the simple initial goal: similar to kernel regression, as described in Chapter 1, we want to express the surface as a linear combination of the given measurements. For instance, if $(z_i)_{i\in[n]}$ are the measured points and $x \in \mathbb{R}^d$ is the coordinate, then:

$$Z(x) = \sum_{i=1}^{n} \alpha_i(x) z_i$$

Importantly, we treat the output of Z as a Gaussian random variable, and $\{Z(x)\}_{x\in[0,1]^d}$ as a family of inter-related (NOT independent) Gaussian random variables. This set is sometimes called a **Gaussian process**.

In the same vein as seeking out the smallest possible hypothesis class, we make two assumptions in order to restrict the kinds of functions that α_i can be. Let $\mu \in \mathbb{R}$ and $\gamma : \mathbb{R}^+ \mapsto \mathbb{R}^+$ be a monotone, continuous function we call the variogram.

- (O1) In expectation, there is no x-dependent trend in the surface. $\mathbb{E}(f) = \mu$.
- (O2) The variance is radially symmetric. $\text{Var}(f(\vec{x} + \vec{h}) - f(\vec{x})) = \gamma(||\vec{h}||)$

If we know what the variogram is, then there is a reasonable condition we can enforce on the α_i: namely, to minimize the variance in the estimate Z.

For any point x_0 on the surface, the variance is given as follows.

$$\begin{aligned}
\mathbb{E}[Z^*(x_0) - Z(x_0)]^2 &= \mathbb{E}\Big[\sum_{i=1}^{n} \alpha_i (Z(x_i) - Z(x_0))\Big]^2 \\
&= \mathbb{E}\Big[\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j (Z(x_i) - Z(x_j))^2 - \sum_{i=1}^{n} \alpha_i (Z(x_i) - Z(x_0))^2\Big] \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j \mathbb{E}[(Z(x_i) - Z(x_j)]^2 - \sum_{i=1}^{n} \alpha_i \mathbb{E}[Z(x_i) - Z(x_0)]^2\Big] \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j \gamma(||h_{ij}||) - \sum_{i=1}^{n} \alpha_i \gamma(||h_{i0}||)
\end{aligned}$$

We can formulate this as a constrained optimization problem.

$$\min_{\alpha} \sigma_{s0}^2 \qquad \text{s.t.} \qquad \sum_{i=1}^{n} \alpha_i = 1$$

Applying the first fundamental insight of Lagrange optimization, we notice that the above problem can be rewritten as a s. In the literature this is called the *primal problem*, hence the p.

$$p^* = \min_{\alpha} \max_{\lambda} \left(\sigma_{s0}^2 + \lambda(1 - \sum_{i=1}^{n} \alpha_i) \right)$$

The second step is to ask the question: what happens if I switch the min and max? We call this the dual optimization problem.

$$d^* = \max_{\lambda} \min_{\alpha} \left(\sigma_{s0}^2 + 2\lambda(1 - \sum_{i=1}^{n} \alpha_i) \right)$$

We can use the Lagrange multiplier method and solve the dual problem. Since this is a convex optimization problem with affine constraints (Slater's conditions), we can assert that the solution to the dual problem is equal to that of the original (primal) problem.

$$L(\alpha, \lambda) = \sigma_{s0}^2 + 2\lambda(\sum_{i=1}^{n} \alpha_i - 1)$$

$$\frac{\partial L}{\alpha_i} = \alpha_i \sum_{j=1}^{n} \alpha_j \gamma(||h_{ij}||) - \alpha_i \gamma(||h_{i0}||) + 2\lambda\alpha_i = 0$$

$$\begin{cases} \sum_{j=1}^{n} \alpha_j \gamma(||h_{ij}||) + 2\lambda\alpha_i = \alpha_i \gamma(||h_{i0}||) \\ \sum_i \alpha_i = 1 \end{cases}$$

4.2 Curio (Neural Tangent Kernel)

5 Function Spaces

This chapter is intended as an introduction to some of the fundamental ideas and intuitions of functional analysis. Although some of the topics here are quite abstract, this is meant to be accessible and insightful to the *practitioner*. Mathematics is supposed to help us better organize our thoughts. A deeper understanding of what functions are and how they act gives us better ability to build new, exciting models. The curio at the end of this chapter is an example.

5.1 Functions as vectors

One of the fundamental conceptual leaps in higher mathematics is the idea that *functions are vectors*. You can imagine this in two different ways.

- (Lists) In practical settings, we often think of vectors as lists of numbers, say $v = (v_1, ..., v_n) \in \mathbb{R}^n$. In some sense, this means that I am associating each of the numbers 1 through n with a distinct real number. So there is some function $\tilde{v} : \{1, ..., n\} \mapsto \mathbb{R}$ corresponding to v. This makes the step towards generalization a little easier: why limit ourselves to discrete domains? Why not $\tilde{v} : \mathbb{N} \mapsto \mathbb{R}$ (i.e. sequences) or $\tilde{v} : \mathbb{R} \mapsto \mathbb{R}$ (functions)?

- (Algebra) Say we are looking at the set of functions $\mathbb{R} \mapsto \mathbb{R}$, which we indeed may write as $\mathbb{R}^{\mathbb{R}}$. This is a vector space because its elements satisfy the classic linearity property, under point-wise addition.

  $$(cf + bg)(x) = c \cdot f(x) + b \cdot g(x) \qquad \forall f, g \in \mathbb{R}^{\mathbb{R}}, b, c \in \mathbb{R}$$

  The more algebraic way of saying this is that, if f, g are functions, then linear combinations $cf + bg$ are also functions. It seems almost too obvious to record, but it is really quite profound.

Now, the space of all functions on the real line $\mathbb{R}^{\mathbb{R}}$ is unimaginably huge and incredibly ugly. Almost none of these functions are continuous, and you can find all kinds of examples of functions which are, say, continuous everywhere but differentiable nowhere (in other words, very, very rough).

Exercise: Find a function $\mathbb{R} \mapsto \mathbb{R}$ which is discontinuous everywhere except at one point. (**Hint**: Leverage the density of the rationals in the reals.)

Badly behaved functions are harder to approximate. It is easier when they have intrinsic structure like continuity or differentiability. Consider *Taylor's approximation theorem*, for instance, which tells us how to approximate a function on a given neighborhood, in terms of linear combinations of its derivatives.

5.2 Topology, and why it matters

- Topological Space
- Metric Space
- Normed Space
- Inner Product Space

5.3 Hilbert Spaces

A Hilbert space $\mathcal{H}$ is vector space with some extra structure: in particular, it is equipped with an inner product, which in turn defined a

An inner product space $\mathcal{H}$ is a vector space equipped with a map $\langle \cdot, \cdot \rangle : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ which is symmetric, positive definite, and linear in both arguments. Inner products spaces are automatically metric spaces, since one can define $d(x, y) = \sqrt{\langle x - y, x - y \rangle}$. A metric space $(\mathcal{H}, d)$ is complete if every Cauchy sequence converges inside the space. The rationals are famously not complete.

: for instance, vectors in $\mathbb{R}^n$ with $\langle x, y \rangle = x^T y$ or the space of square integrable functions $L_2 = \{f : ||f||_2 < \infty\}$ with $\langle f, g \rangle = \int f(x)g(x)dx$.

$$L_p = \{f : ||f||_p < \infty\}$$

$$K(x, y) = \langle \phi(x), \phi(y) \rangle$$

Theorem (Equivalence of Kernel and Embedding): $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is a kernel if and only if there exists a Hilbert space $\mathcal{H}$ and a map $\Phi : \mathcal{X} \mapsto \mathcal{H}$ such that

Reproducing Property: for all $x \in X$ there exists a uniwue $K_x \in H$ with the reproducing property:

$$f(x) = L_x(f) = \langle f m K_x \rangle_H$$

Example: multi-scale basis functions and applications in digital elevation modeling. (Patra Shekhar)

5.3.1 Curio (Multi-scale RKHS)

6 Appendix

6.1 MLE Formulation of Least Squares

$$\begin{aligned}
\arg\max_{\omega} \ \mathbb{P}(y_1, ..., y_n \mid \omega) &= \arg\max_{\omega} \ \prod_{i=1}^{n} \mathbb{P}(y_i \mid \omega) \\
&= \arg\max_{\omega} \ \log\Big(\prod_{i=1}^{n} \mathbb{P}(y_i \mid \omega)\Big) \\
&= \arg\max_{\omega} \ \sum_{i=1}^{n} \log\Big(\mathbb{P}(y_i \mid \omega)\Big)
\end{aligned}$$

References

[1] Ronald DeVore, Boris Hanin, and Guergana Petrova. "Neural network approximation". In: *Acta Numerica* 30 (2021), pp. 327–444.

[2] Paul HC Eilers and Brian D Marx. "Flexible smoothing with B-splines and penalties". In: *Statistical science* 11.2 (1996), pp. 89–121.

[3] David LB Jupp. "Approximation to data by splines with free knots". In: *SIAM Journal on Numerical Analysis* 15.2 (1978), pp. 328–343.

[4] David Ruppert, Matt P Wand, and Raymond J Carroll. *Semiparametric regression*. 12. Cambridge university press, 2003.

[5] Tony Schenk and Beata Csatho. "A new methodology for detecting ice sheet surface elevation changes from laser altimetry data". In: *IEEE Transactions on Geoscience and Remote Sensing* 50.9 (2012), pp. 3302–3316.

[6] Prashant Shekhar et al. "Alps: a unified framework for modeling time series of land ice changes". In: *IEEE Transactions on Geoscience and Remote Sensing* 59.8 (2020), pp. 6466–6481.