CIS3333: Mathematics of Machine Learning Fall 2025

Lecture: Multivariate Calculus I

Date: October 29th, 2025 *Author: Surbhi Goel*

1 Introduction and Motivation

Recall from the first lecture that the goal of learning is to minimize the empirical risk:

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \hat{R}(f_\theta) = \arg\min_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^{N} \ell(f_\theta(x_i), y_i)$$

where ℓ is the loss function, f_θ is our model with parameters θ, and (x_i, y_i) are the training examples.

How do we actually solve this minimization problem?

Gradient Descent: The Workhorse of Machine Learning. A common method is *gradient descent*, an iterative algorithm that you can think of as trying to find the bottom of a valley by always taking a step downhill. The gradient tells us which direction is steepest, and we take a small step in that direction.

More formally, gradient descent is an iterative algorithm that:

- Starts with an initial guess θ_0 for the parameters
- Repeatedly updates:
$$\theta_{t+1} = \theta_t - \eta \nabla \hat{R}(\theta_t)$$

Here, the step size $\eta > 0$ (also called learning rate) controls how far we move, and the negative gradient $-\nabla \hat{R}(\theta_t)$ points in the direction of steepest descent.

Stochastic gradient descent (SGD). A variant called stochastic gradient descent updates parameters using only a randomly chosen data point (or small mini-batch) at each step:

$$\theta_{t+1} = \theta_t - \eta \nabla \ell(f_{\theta_t}(x_i), y_i)$$

where i is uniformly sampled from $\{1, \ldots, N\}$. This makes each step much cheaper computationally, but introduces randomness—the update is no longer guaranteed to decrease the objective at every step.

This brings us to two fundamental questions:

1. **Why does gradient descent work?** We'll show it eventually converges close to the optimal solution under certain assumptions on the loss function.

2. **How long does it take?** We'll characterize convergence by the number of steps T needed to get within ϵ of the optimal solution. This is known as the *convergence rate*.

For gradient descent on smooth, convex functions, the convergence rate is

$$\hat{R}(\theta_T) - \hat{R}(\theta^*) = O\left(\frac{1}{T}\right)$$

where θ^* is the optimal parameter. To get ϵ error requires $O(1/\epsilon)$ steps.

For stochastic gradient descent, the rate is

$$\hat{R}(\theta_T) - \hat{R}(\theta^*) = O\left(\frac{1}{\sqrt{T}}\right)$$

To get ϵ error requires $O(1/\epsilon^2)$ steps. The penalty for using random gradients is a factor of $\sqrt{T}$.

Why We Need Calculus. To understand why these algorithms work and prove convergence rates, we need calculus tools: gradients, approximations, and Taylor series. The proof for gradient descent convergence combines linear algebra (from module 1), probability (from module 2), and calculus (this module).

2 Univariate Calculus Review

- The derivative of f at x is defined as the limit

$$\frac{\partial f}{\partial x} = \lim_{h \to 0^+} \frac{f(x+h) - f(x)}{h}$$

  This measures the average rate of change of f over a small interval of length h, then takes the limit as $h \to 0^+$. When this limit exists, we say that f is differentiable at x.

- Example: $f(x) = x^2$, then

$$\frac{\partial f}{\partial x} = \lim_{h \to 0^+} \frac{(x+h)^2 - x^2}{h} = \lim_{h \to 0^+} \frac{x^2 + 2xh + h^2 - x^2}{h} = \lim_{h \to 0^+} (2x + h) = 2x$$

2.1 The Derivative as Best Linear Approximation

Beyond just computing slopes, derivatives have a deeper meaning: they provide the *best linear approximation* to a function near a point.

Tangent line and little-o notation. If f is differentiable at x_0, we can write

$$f(x_0 + h) = f(x_0) + f'(x_0)h + o(h)$$

where $o(h)$ (read "little-o of h") denotes terms that vanish faster than h as $h \to 0$. Formally:

$$\lim_{h \to 0} \frac{o(h)}{h} = 0$$

- The term $f'(x_0)h$ is a *linear function* of h: doubling h doubles this term.
- The tangent line $y = f(x_0) + f'(x_0)(x - x_0)$ is the graph of this linear approximation.
- The error $o(h)$ is negligible compared to h for small perturbations.
- Example: For $f(x) = x^2$ at $x_0 = 1$, we have

$$\begin{aligned} f(1+h) &= (1+h)^2 = 1 + 2h + h^2 \\ &= f(1) + f'(1)h + h^2 \end{aligned}$$

  The linear part is $f'(1)h = 2h$, and the error $o(h) = h^2$ vanishes faster than h.

Why "best"? Among all possible linear functions $L(h) = ch$ that could approximate $f(x+h) - f(x)$, the choice $c = f'(x)$ gives the unique linear function where the error is $o(h)$. Any other choice would give an error that's at least $O(h)$ (proportional to h, not vanishing faster). See stackexchange for a proof.

Geometric intuition. If you zoom in sufficiently close to the point $(x_0, f(x_0))$ on a smooth curve, the curve becomes indistinguishable from its tangent line. The derivative $f'(x_0)$ is the slope of this tangent line. This is why calculus is sometimes called "the study of things that are locally linear."

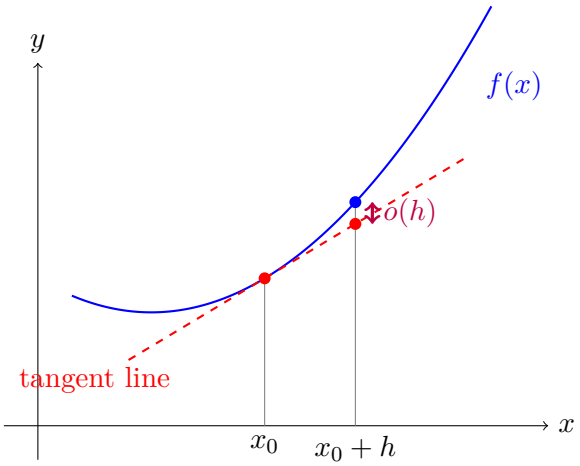

Figure 1: The tangent line provides the best linear approximation to f near x_0. The approximation error $o(h)$ vanishes faster than h as $h \to 0$.

2.2 Differentiation Rules

Now that we understand what derivatives represent conceptually, let's review the mechanical rules for computing them.

Useful properties of little-o notation. Before proving rules, note that little-o notation is robust under basic operations:

1. If c is a constant, then $c \cdot o(h) = o(h)$ (multiplying by a constant preserves the order)
2. If $f(h)$ and $g(h)$ are both $o(h)$, then $f(h) + g(h)$ is also $o(h)$ (sum of negligible terms is negligible)
3. If $|\ell| \leq C|h|$ for some constant C, then $o(\ell) = o(h)$ (proportional arguments have same order)

We'll use these facts to manipulate error terms in the proofs below.

Summary of differentiation rules. Let f' denote the derivative of f. Then:

1. Product rule: $(fg)' = f'g + fg'$
2. Sum rule: $(f + g)' = f' + g'$
3. Chain rule: $(g(f))' = g'(f)f'$
4. Quotient rule: $\left(\frac{f}{g}\right)' = \frac{f'g - fg'}{g^2}$

These rules allow us to compute derivatives of complicated functions by breaking them into simpler pieces.

Chain Rule via Composition of Linear Approximations. The chain rule is particularly insightful - it shows that the derivative of a composition is the *composition of the linear approximations*. This will be key for understanding backpropagation!

Suppose we want to differentiate $h(x) = g(f(x))$. Write the linear approximations:

$$f(x_0 + k) = f(x_0) + f'(x_0)k + o(k)$$
$$g(y_0 + \ell) = g(y_0) + g'(y_0)\ell + o(\ell)$$

Now compose them. Let $y_0 = f(x_0)$ and note that the change in f is:

$$\ell = f(x_0 + k) - f(x_0) = f'(x_0)k + o(k)$$

This tells us that ℓ is proportional to k for small k (specifically, $|\ell| \leq C|k|$ for some constant C).

Substitute into the approximation for g:

$$\begin{aligned} h(x_0 + k) &= g(f(x_0 + k)) = g(y_0 + \ell) \\ &= g(y_0) + g'(y_0)\ell + o(\ell) \\ &= g(f(x_0)) + g'(f(x_0)) \cdot [f'(x_0)k + o(k)] + o(\ell) \\ &= h(x_0) + g'(f(x_0))f'(x_0)k + \underbrace{g'(f(x_0)) \cdot o(k) + o(\ell)}_{o(k)} \end{aligned}$$

The last step uses two facts: (1) multiplying $o(k)$ by the constant $g'(f(x_0))$ keeps it $o(k)$, and (2) since ℓ is proportional to k, we have $o(\ell) = o(k)$. Therefore $h'(x_0) = g'(f(x_0))f'(x_0)$, which is the chain rule!

The key insight: the derivative of g at y_0 acts as a linear map on the change ℓ, and that change itself is (approximately) the linear map $f'(x_0)$ acting on k. Composition of linear maps means multiplying their derivatives!

3 Multivariate Calculus: Gradients as Best Linear Approximations

Machine learning problems are typically multivariate (think pixels or words in a sentence). The "best linear approximation" perspective extends naturally: gradients are linear maps from $\mathbb{R}^n \to \mathbb{R}$.

3.1 Partial Derivatives and the Gradient

For a function $f : \mathbb{R}^n \to \mathbb{R}$ with $x = (x_1, \ldots, x_n)$, the partial derivatives are:

$$\frac{\partial f}{\partial x_i} = \lim_{h \to 0} \frac{f(x_1, \ldots, x_{i-1}, x_i + h, x_{i+1}, \ldots, x_n) - f(x)}{h}$$

for $i = 1, \ldots, n$. We collect these into the *gradient* (a column vector):

$$\nabla_x f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \in \mathbb{R}^n$$

Example: For $f(x_1, x_2) = x_1^2 x_2 + x_1 x_2^3$:

$$\frac{\partial f}{\partial x_1} = 2x_1 x_2 + x_2^3, \quad \frac{\partial f}{\partial x_2} = x_1^2 + 3x_1 x_2^2$$

3.2 Gradient as Best Linear Approximation

We've defined the gradient mechanically as $\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$. But what does it represent conceptually?

A function $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable at x_0 if there exists a linear map $L : \mathbb{R}^n \to \mathbb{R}$ such that

$$f(x_0 + h) = f(x_0) + L(h) + o(\|h\|)$$

where $o(\|h\|)$ denotes terms that vanish faster than $\|h\|$ as $h \to 0$. The gradient $\nabla f(x_0)^\top$ is a row vector representing this linear map. It's linear because dot products are linear.

Connection between definitions: When all partial derivatives exist and are continuous at x_0, the function is differentiable and the linear map is $L(h) = \nabla f(x_0)^\top h$. In other words:

- *Mechanically:* compute partial derivatives along each coordinate axis, stack into gradient
- *Conceptually:* the gradient defines the unique linear map that best approximates f
- *Why they agree:* continuous partial derivatives ensure the approximation works for all directions, not just coordinate axes
- *Important caveat:* Partial derivatives can exist at isolated points without the function being differentiable. Continuous partial derivatives are sufficient (but not always necessary) for differentiability.

Steepest ascent. The gradient points in the direction of steepest ascent. Using the linear approximation, a small step in direction d ($\|d\| = 1$) changes f by:

$$f(x_0 + \epsilon d) - f(x_0) \approx \epsilon \nabla f(x_0)^\top d = \epsilon \|\nabla f(x_0)\| \cos(\theta)$$

where θ is the angle between ∇f and d (dot product formula). This is maximized when d is parallel to ∇f. Thus gradient descent takes steps $-\nabla f$ for steepest descent, maximizing the decrease in f.

Tangent hyperplane. The gradient defines the tangent hyperplane at x_0:

$$z = f(x_0) + \nabla f(x_0)^\top (x - x_0)$$

This generalizes the tangent line from 1D. The hyperplane locally approximates the surface of f.