07: Reality is Hard (Challenges in Optimization)

Chapter Goal: To serve as a "reality check." After learning the clean and elegant tools of calculus, we now confront the fact that the real world (and its data) is messy.

---

1. Real-World Challenges We've Avoided So Far

Our intuition of a "mountain landscape" is great, but in Machine Learning, the reality is much harder for four main reasons:

1. High Dimensionality

- We are no longer working in 2D or 3D. ML models can have thousands, or even millions, of "knobs" (parameters).
- **Consequence:** We can no longer visualize the landscape. We must rely completely on the **"mathematics to be true"** and use our 2D intuition as a conceptual guide.

2. Expensive Functions

- We often don't have an analytical formula for our Loss Function.
- To find the "height" at just a single point, we might have to run a simulation on a supercomputer for days.
- **Consequence:** We cannot "plot the entire landscape". Every function evaluation must be done carefully and efficiently.

3. Non-Smooth Functions

- Real-world landscapes are not always smooth. There can be sharp features, like "cliffs" or "fault lines" (discontinuities).
- **Consequence:** At these sharp points, the concept of a "gradient" becomes undefined or confusing, making navigation difficult.

4. Noisy Data

- Real-world data almost always contains "noise" or measurement errors.
- **Consequence:** Our function landscape becomes "rough" or "pebbly". As we saw in the "Sandpit" lab, this makes the Gradient unstable and unreliable.

---

2. The Key Question: How to Calculate the Gradient Without a Formula?

This is a major problem. All of our methods so far have relied on being able to find the partial derivatives from a formula `f(x,y)`.

If we don't have the formula, how do we find the Gradient?

---

3. The Practical Solution: Back to Basics (Numerical Methods)

- **Core Idea:** Return to the most fundamental definition of a derivative: **Rise / Run**.
- **The Finite Difference Method:** This is a way to **approximate** the Gradient numerically.
  1. Start at the current position `(x, y)`.
  2. **For $\frac{\partial f}{\partial x}$:** Take one small step in the x-direction (by an amount `h`) to the point `(x+h, y)`. Measure the "Rise": $f(x+h, y) - f(x, y)$.

$$\frac{\partial f}{\partial x} \approx \frac{f(x+h, y) - f(x, y)}{h}$$

  3. **For $\frac{\partial f}{\partial y}$:** Take one small step in the y-direction (by an amount `h`) to the point `(x, y+h)`. Measure the "Rise": $f(x, y+h) - f(x, y)$.

$$\frac{\partial f}{\partial y} \approx \frac{f(x, y+h) - f(x, y)}{h}$$

- **Conclusion:** We can approximate the Gradient just by performing a few extra function evaluations around our current point. We don't need the derivative's formula.

---

4. Challenges of the Numerical Method

How Big Should the "Small Step" (`h`) be?

- **If `h` is too large:** The Rise/Run approximation will be very inaccurate (the secant line is too far from the tangent line).
- **If `h` is too small:** The computer might run into numerical precision issues. The difference $f(x+h) - f(x)$ could be so tiny that the computer rounds it to zero, which would cause the division by `h` to become unstable.
- Choosing the "just right" `h` is an art.

How to Deal with "Noise"?

- If the data is noisy, a single Rise/Run measurement can be very misleading.
- **Simple Solution:** Perform the measurement with several different step sizes `h` and then take their average to get a more stable approximation.

5. Final Message

- The real world is "messy". The beautiful, smooth concepts of calculus must be adapted with **approximative numerical methods** to be applied to complex data and functions.
- Understanding these limitations is just as important as understanding the theory itself.

Tentu saja. Mari kita bedah ide **"menghitung Gradien tanpa rumus"** ini dengan sebuah contoh konkret yang bisa kamu bayangkan, yaitu **analogi "Bak Pasir"** kita.

> **Example Scenario**
>
> maksudnyaa gimanaaa yaa aku gak ada bayangan sama sekalii, bisa kasih contoh konkret?

Situasi: Kamu di Dalam Bak Pasir yang Gelap

- **Fungsi `f(x,y)` :** Ini adalah **bentuk dasar bak pasir** yang tidak kamu ketahui. Tidak ada rumus.
- **Komputer:** Ini adalah **kamu** dengan **tongkat pengukur**.
- **Tujuan:** Kamu sedang berada di titik `P = (3, 5)`. Kamu ingin mencari tahu **arah turunan tercepat** dari sini, yaitu `−∇f`. Tapi kamu tidak punya "kompas gradien" otomatis.

Bagaimana cara kamu **membuat kompasmu sendiri** hanya dengan menggunakan tongkat? Inilah **Metode Beda Hingga (Finite Difference Method)**.

Langkah 1: Mengukur Kecuraman di Arah Timur-Barat (`∂f/∂x`)

Kamu ingin tahu, "Seberapa miring dasar bak pasir ini jika aku melangkah ke arah Timur (arah sumbu-x)?"

1. **Ukur Ketinggian Awal:**
   - Kamu tusukkan tongkatmu tepat di posisimu, `P = (3, 5)`.
   - Kamu catat kedalamannya: `f(3, 5) = −10 meter`.
2. **Ambil "Langkah Uji" yang Kecil:**
   - Kamu memutuskan untuk mengambil satu "langkah uji" yang sangat kecil **hanya di arah x**. Sebut saja langkahnya `h = 0.01` meter.
   - Kamu pindah ke posisi baru, `P_timur = (3 + 0.01, 5) = (3.01, 5)`.

3. **Ukur Ketinggian Baru:**
    - Di posisi baru ini, kamu tusukkan lagi tongkatmu.
    - Kamu catat kedalaman barunya: `f(3.01, 5) = −10.02 meter`.
4. **Hitung "Rise over Run":**
    - **Rise (Perubahan Ketinggian):** `f_baru − f_lama = −10.02 − (−10) = −0.02` meter.
    - **Run (Jarak Langkah):** `h = 0.01` meter.
    - **Kecuraman Arah-x (`∂f/∂x`):** `Rise / Run = −0.02 / 0.01 = −2`.

Kesimpulan Sementara: Kamu menemukan bahwa kecuraman di arah `x` adalah `−2`. Artinya, jika kamu berjalan ke Timur, tanahnya menurun.

Langkah 2: Mengukur Kecuraman di Arah Utara-Selatan (`∂f/∂y`)

Sekarang kamu kembali ke posisi awalmu `P = (3, 5)` dan mengulangi prosesnya untuk arah `y`.

1. **Ketinggian Awal:** `f(3, 5) = −10 meter` (masih sama).
2. **Ambil "Langkah Uji" yang Kecil:**
    - Kali ini, kamu melangkah sedikit **hanya di arah y**, sebesar `h = 0.01`.
    - Posisi baru: `P_utara = (3, 5 + 0.01) = (3, 5.01)`.
3. **Ukur Ketinggian Baru:**
    - Kamu tusukkan tongkat di `(3, 5.01)`.
    - Kamu catat kedalamannya: `f(3, 5.01) = −9.97 meter`.
4. **Hitung "Rise over Run":**
    - **Rise:** `f_baru − f_lama = −9.97 − (−10) = +0.03` meter.
    - **Run:** `h = 0.01` meter.
    - **Kecuraman Arah-y (`∂f/∂y`):** `Rise / Run = 0.03 / 0.01 = +3`.

Kesimpulan Sementara: Kamu menemukan bahwa kecuraman di arah `y` adalah `+3`. Artinya, jika kamu berjalan ke Utara, tanahnya menanjak.

Langkah 3: Rakit "Kompas" Gradienmu

Sekarang kamu punya semua bahan yang kamu butuhkan:

- `∂f/∂x ≈ −2`
- `∂f/∂y ≈ +3`

Kamu bisa merakit **aproksimasi dari Gradien** di titik `(3, 5)`:
`∇f(3, 5) ≈ [-2, 3]`

"Kompas" Tanjakanmu menunjuk ke arah Barat Laut.

Arah Turunan Tercepatmu (`-∇f`) adalah kebalikannya: `[2, -3]` (Timur Tenggara).

Kesimpulan:
Inilah yang dimaksud dengan "menghitung Gradien secara numerik".

- Kita tidak punya rumus `f(x,y)`.
- Kita hanya punya kemampuan untuk "menusuk tongkat" (mengevaluasi `f` di titik mana pun).
- Dengan melakukan **tiga tusukan** (satu di titik pusat, satu sedikit di kanan, satu sedikit di atas), kita bisa mendapatkan **aproksimasi yang cukup baik** untuk Gradien.
- Ini adalah metode yang sangat kuat, karena ia bekerja bahkan jika kita tidak tahu sama sekali bentuk dari "dasar bak pasir" kita.

> ⚠ **Warning**
>
> lah kalau gitu ngapain donk aku repot repot belajar ginian semuaa?? kalau misalnya ternyata kita cuman meraba raba sajaaa?

Kamu benar-benar sampai pada inti dari "frustrasi" seorang murid matematika saat bertemu dengan dunia nyata. "Aku sudah belajar semua teori indah ini, kenapa pada akhirnya cuma 'kira-kira'?"

Jawaban untuk pertanyaanmu sangat dalam, dan ini adalah pencerahan terakhir yang akan mengikat semuanya.

Kenapa Kita Repot-Repot Belajar Teori?

Kamu tidak "repot-repot". Kamu sedang membangun **"PETA MENTAL"**.

Bayangkan ada dua orang penjelajah di "bak pasir" yang gelap, sama-sama hanya punya tongkat.

Penjelajah A (Hanya Tahu Cara "Meraba"):

- Dia tahu resep "Finite Difference": "Tusuk di sini, tusuk sedikit di kanan, bagi selisihnya."

- Dia mendapatkan Gradien aproksimasi `[-2, 3]`.
- Dia tahu dia harus berjalan ke arah `[2, -3]`.
- Dia melakukannya berulang-ulang dan akhirnya (mungkin) menemukan lembah.
- **Dia berhasil menyelesaikan tugasnya.** Dia adalah seorang **teknisi yang baik**.

Penjelajah B (Kamu - yang Paham Teori):

- Dia juga tahu resep "Finite Difference" dan mendapatkan Gradien `[-2, 3]`.
- **TAPI**, di dalam kepalanya, dia punya **GAMBARAN BESAR**.
- Saat dia melihat hasil `[-2, 3]`, dia tidak hanya melihat angka. Dia berpikir:
  - "Oke, Gradienku `[-2, 3]`. Ini adalah **vektor**. Arah turunanku adalah `[2, -3]`."
  - "Langkahku berikutnya akan ke arah itu. Tapi aku harus hati-hati memilih **learning rate**, karena kalau terlalu besar, aku bisa 'melompati' lembahnya."
  - "Hmm, setelah beberapa langkah, langkahku jadi zig-zag. Ah, aku pasti sedang berada di **'ngarai sempit'**. Gradien di sini tidak efisien. Mungkin aku harus coba algoritma dengan **Momentum**."
  - "Setelah 1000 langkah, algoritmaku berhenti. Gradiennya sudah `[0.001, -0.002]`, sangat kecil. Tapi, apakah ini **Global Minimum** atau hanya **Local Minimum**? Aku harus coba **restart** dari titik lain untuk memastikannya."
  - "Atau, mungkin aku bisa menghitung **Matriks Hessian** secara numerik di sini untuk mengecek 'kelengkungan'-nya. Jika determinannya negatif, ini pasti **saddle point**, bukan solusi."

Jadi, Apa Gunanya Belajar Ini Semua?

"Meraba-raba" (metode numerik) adalah apa yang **komputer lakukan**.
Belajar teori adalah apa yang **KAMU lakukan** untuk **MENGATUR, MENGAWASI, dan MEMPERBAIKI** cara komputer "meraba-raba".

Guna Belajar Teori:

1. **Untuk Membangun Algoritmanya:** Tanpa konsep **Gradien** dari kalkulus, kita bahkan tidak akan pernah tahu bahwa "meraba" ke arah $-\nabla f$ adalah strategi yang cerdas. Kita akan selamanya terjebak di "Random Search". Teori memberimu **STRATEGI** untuk "meraba".
2. **Untuk Mendiagnosis Masalah:** Saat algoritma Gradient Descent-mu gagal, seorang teknisi hanya bisa berkata, "Programnya tidak konvergen." Tapi **KAMU** bisa berkata, "Programnya tidak konvergen, **KEMUNGKINAN BESAR** karena kita terjebak di *local minimum*, atau karena *learning rate*-nya terlalu besar, atau karena lanskapnya *ill-conditioned* (seperti ngarai sempit)."
3. **Untuk Memilih Alat yang Tepat:** Ada puluhan algoritma optimisasi selain Gradient Descent (Adam, RMSProp, L-BFGS). Semuanya adalah variasi yang mencoba

memperbaiki kelemahan Gradient Descent. Tanpa memahami **kelemahan** itu (local minima, ngarai sempit, dll.), kamu tidak akan pernah tahu kapan harus menggunakan alat yang mana.

4. **Untuk Menciptakan Alat Baru:** Inilah tujuan akhirmu. Suatu saat nanti, kamu mungkin akan menghadapi masalah baru di mana tidak ada algoritma yang ada yang bekerja dengan baik. Karena kamu paham **prinsip-prinsip pertamanya** (apa itu turunan, apa itu gradien), kamu punya kesempatan untuk merancang **strategi "meraba" yang baru dan lebih baik**, yang disesuaikan untuk masalahmu.

Kesimpulan:
Kita tidak "repot-repot" belajar semua ini hanya untuk berakhir dengan "meraba-raba".
Kita belajar semua ini agar kita bisa **"meraba-raba" dengan cerdas, efisien, dan terarah**.

Komputer adalah prajurit yang buta dan hanya mengikuti perintah. Kamu adalah **jenderalnya**. Kamu harus paham "peta pertempuran" (teori) untuk bisa memberikan perintah yang benar kepada prajuritmu.

Tags: #mml-specialization #multivariate-calculus #optimization #numerical-methods #finite-difference