09. Linear regression and machine learning

Last time

- Exact enumeration for first-passage times
- Continuous random variables
- Probability density function
- Central Limit Theorem

Goal for today

- What is machine learning?
- Linear regression
- Derivatives and how to calculate them

Understanding data

- Suppose have **data** from some experiment / process
- Examples:
  - Stock price as function of time
  - Jet noise as a function of air flow speed
  - Sales as function of advertising budget
  - Length of spring as function of force applied
  - Variance of random walker as function of time

Characteristics of data

- Data have **inputs**: we specify or measure them
- Data have **outputs**: we are interested in how they change when inputs change
- Data are **noisy**: intrinsic random fluctuations

Understanding data II: Models

- Want to **understand** / characterise structure of data: world of **statistics**
- Also want to **predict** "response" for new input data: world of **machine learning**

- To understand data we will impose some kind of **structure** on it – a **model**
- Model describes relationship that we think data has

Linear regression

- One of simplest models: **linear regression** (bad name)
- Relates quantitative measurements
- Assumes linear (affine) relationship between inputs X and outputs Y:

$$Y = f(X) = aX + b \quad + \epsilon$$

- ϵ describes noise / fluctuations
- Unknown **parameters** a and b

Machine learning: Fitting

- Notation: Data (x_i, y_i)
- Input x_i and corresponding output y_i, for $i = 1, \ldots, N$
- Given data we want to **learn** parameters a and b in model
- **Learning** is just **fitting**!
- What is fitting?

Fitting

- Fitting: Find parameters a and b in model that **best** describe data
- What does **best** mean?
- Need way to decide what is “best” fit
- Need to **minimize** a **cost** / **loss** function L
- How choose loss function?

Least squares

- Common solution: **Least squares**
- Sum of squares of distances of data from line
- This is a function $L(a, b)$ of parameters a and b
- Find values of a and b that minimize $L(a, b)$
- **How**?

Minimization in 1D

- We have a 2D problem. Simplify to 1D
- Think of hill of height $h(x)$ as function of x
- How find minimum of hill?
- "Roll down the hill"
- Take steps and move in direction that **decreases** L
- How do we talk about functions **decreasing**?

Reminder: Derivatives

- **Derivative** = **rate of change**
- Increasing $\iff$ derivative > 0
- Decreasing $\iff$ derivative 0

Derivatives II

- **Derivative** of function $f : \mathbb{R} \to \mathbb{R}$ at point a is slope of **tangent line**
- Notation: $f'(a)$, or $\left.\frac{df}{dx}\right|_a$ (if you must).
- **Tangent line** is straight line that "touches" graph of function at point
- Formal definition of derivative of $f : \mathbb{R} \to \mathbb{R}$ at $a \in \mathbb{R}$:

$$f'(a) := \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

- Intuition: Limit of slopes of "secant lines" ("rise over run")

Why do we care about derivatives?

- They tell us how function looks "locally" (close to a point).
- E.g. used to analyze dynamics near a fixed point.
- Some applications:
  - optimization
  - finding roots (zeros)
  - sensitivity: "how much does output change when input varies"

How to *calculate* derivatives numerically

- Numerically: Cannot take limit $h \to 0$
- So don't! Fix a finite, non-zero h to get **finite difference** approximation:

  $f'(a) \simeq \frac{f(a+h)-f(a)}{h}$
- How good is this approximation? PS4
- Can we do better by calculating derivatives exactly?

A different point of view

- Rewrite definition in more useful way:
- Get rid of that annoying limit! (Or, rather, hide it):

$$\lim_{h\to 0} \left[\frac{f(a+h)-f(a)}{h} - f'(a) \right] = 0$$

- Write as

$$\frac{f(a+h)-f(a)}{h} - f'(a) = o(h)$$

- Define $o(h)$ to mean "any function $g(h)$ that satisfies $g(h)/h \to 0$ when $h \to 0$".
- Then

$$f(a+h) = f(a) + hf'(a) + o(h).$$

- Conversely: If can find A and B with $f(a+h) = A + Bh + o(h)$, then $A = f(a)$ and $B = f'(a)$.
- Use this to calculate derivatives!
- **Intuition**: Tangent line is best affine approximation to f near a.

Infinitesimals

- Simplify by thinking of "infinitesimal" perturbation ϵ
- With $\epsilon^2 = 0$
- So $f(a+\epsilon) = f(a) + \epsilon f'(a)$
- Expand $f(a+\epsilon)$; coefficient of ϵ is derivative

Sum rule for derivatives

- Sum of two functions:

$$(f+g)(x) := f(x) + g(x)$$

- Its derivative:

$$\begin{gathered}[f+g](a+\epsilon) = f(a+\epsilon) + g(a+\epsilon) \\ [f(a) + \epsilon f'(a)] + [g(a) + \epsilon g'(a)] \\ [f(a) + g(a)] + [f'(a) + g'(a)]\epsilon.\end{gathered}$$

- Hence $(f+g)'(a)$ = (coefficient of ϵ) = $f'(a) + g'(a)$.

Product rule for derivatives

- Product of two functions:

$$(f \cdot g)(x) := f(x) \cdot g(x)$$

  (Here $\cdot$ is normal scalar multiplication)

- Its derivative:

$$\begin{gathered}[f \cdot g](a+\epsilon) = f(a+\epsilon) \cdot g(a+\epsilon) \\ [f(a) + \epsilon f'(a)] \cdot [g(a) + \epsilon g'(a)] \\ [f(a) \cdot g(a)] + [f(a)g'(a) + g(a)f'(a)]\epsilon.\end{gathered}$$

- Hence $(f \cdot g)'(a)$ = (coefficient of ϵ) = $f(a)g'(a) + g(a)f'(a)$.

Derivatives by executing rules: Algorithmic differentiation

- For more complicated function, execute each rule in turn
- E.g. $h(x) = 3x^2 + 2x$ is $h(x) = +(3 * (x * x), 2 * x)$
- Differentiating by hand feels pointless – we are *executing an algorithm*
- Computers are good at that! **Algorithmic / automatic differentiation**
- How *encode* rules to find $f'(a)$ on computer?
- What information do we need for each function?

Information we need

- Fix point a where taking derivatives
- For each function f, need exactly *two* pieces of information:
- Value $f(a)$ and derivative $f'(a)$.
- So can represent function using just those two pieces of information
- How represent in Julia?

Representation in Julia

- Need to group together 2 pieces of information
- Could use tuple or vector etc.
- But want to implement novel **behaviour**, i.e. rules for $+$ and $\times$.
- So instead should *define a new type*
- Commonly called “dual number”

Dual number type

- Make an immutable dual number type:

```julia
struct Dual
    value::Float64
    deriv::Float64
end
```

- Recall: this is template for box holding two variables, `value` and `derivative`.
- `Dual(a, b)` corresponds directly to $a + \epsilon b$

Implementing arithmetic

- To implement arithmetic, import relevant functions:

```julia
import Base: +, *
```

- Add methods acting on objects of type `Dual`: `julia    +(f::Dual, g::Dual) = Dual(f.value + g.value,                    f.deriv + g.deriv)`
- Here we have defined the sum of two functions to have the correct value and derivative

Differentiation

- Suppose have Julia function like `f(x) = x^2 + 2x`
- How differentiate f at $a = 3$?
- $f(a + \epsilon) = f(a) + \epsilon f'(a)$
- So pass in $a + \epsilon$ to f, i.e. `Dual(a, 1)`.
- [Represents identity function $x \mapsto x$ at $x = a$, with derivative 1].
- **Exercise**: Write function `differentiate` taking function `f` and value `a` that calculates $f'(a)$.

Review

- Motivation: Linear regression – fitting straight line
- Optimization wants derivatives
- Calculate derivatives using automatic differentiation