Module 6 — Capstone

Mathematical Intuition

hbar.university

Lesson 6.1: Schrödinger Equation

Overview

Complete decoding of the Schrödinger equation—the fundamental equation of quantum mechanics.

The Equation

Time-Dependent Schrödinger Equation

`iℏ ∂ψ/∂t = Ĥψ`

Time-Independent Schrödinger Equation

`Ĥψ = Eψ`

Symbol-by-Symbol Breakdown

Time-Dependent Form: `iℏ ∂ψ/∂t = Ĥψ`

Left Side: `iℏ ∂ψ/∂t` **i** (imaginary unit) - **Value**: $\sqrt{}$(-1) - **Purpose**: Makes time evolution unitary (preserves normalization) - **Effect**: Without i, probability wouldn't be conserved - **Physical meaning**: Quantum mechanics is fundamentally complex-valued

$\hbar$ (reduced Planck's constant) - **Value**: 1.054571817… $\times 10^{-34}$ J $\cdot$ s - **Units**: Action (energy $\times$ time) - **Purpose**: Sets quantum scale - **Physical meaning**: Fundamental constant relating energy and frequency

∂ψ/∂t (partial time derivative) - **Operation**: Rate of change of wavefunction with time - ψ: Wavefunction (complex-valued function) - **Interpretation**: "How fast is the quantum state changing?" - **Units**: [ψ]/time

Combined: iℏ ∂ψ/∂t - **Meaning**: "Quantum rate of change" - **Units**: Energy $\times$ [ψ] - **Role**: Describes time evolution

Right Side: `Ĥψ` **Ĥ** (Hamiltonian operator) - **Definition**: Energy operator - **Typical form**: Ĥ = -$\hbar^2$/(2m)∇^2 + V(x) - **Components**: Kinetic energy + potential energy - **Physical meaning**: Total energy of system

ψ (wavefunction) - **Domain**: ψ(x,t) or $|\psi(t)\rangle$ - **Values**: Complex numbers - **Normalization**: $\int|\psi|^2$dx = 1 - **Physical meaning**: Encodes all information about quantum state

Ĥψ (Hamiltonian acting on wavefunction) - **Operation**: Operator multiplication - **Result**: Another wavefunction - **Meaning**: "Energy operator acting on state" - **Units**: Energy $\times$ [ψ]

The Equation as a Whole

$\mathbf{i\hbar\ \partial\psi/\partial t = \hat{H}\psi}$

Plain language: "The rate at which a quantum state changes in time is determined by the energy operator acting on that state."

Deeper meaning: - Time evolution is generated by energy (Hamiltonian) - Unitary evolution (i ensures probability conservation) - Deterministic: given $\psi(0)$, can predict $\psi(t)$ - Linear: superpositions evolve as superpositions

Expanded Form

Position Representation

$i\hbar\ \partial\psi(x,t)/\partial t = [-\hbar^2/(2m)\ \partial^2/\partial x^2 + V(x)]\psi(x,t)$

Breaking down the Hamiltonian:

-$\hbar^2$/(2m) $\partial^2/\partial x^2$ (kinetic energy operator) - $\partial^2/\partial x^2$: Second spatial derivative (curvature) - **-$\hbar^2$/(2m)**: Converts curvature to kinetic energy - **Physical meaning**: Kinetic energy from momentum p = -i$\hbar\partial/\partial$x

V(x) (potential energy) - **Function of position**: Different for each system - **Examples**: V(x) = ½kx^2 (harmonic oscillator), V(x) = -e^2/r (hydrogen atom) - **Physical meaning**: External forces, interactions

Time-Independent Form

Eigenvalue Equation: $\hat{H}\psi = E\psi$

Meaning: "The Hamiltonian acting on ψ just scales it by E"

Interpretation: - ψ is an energy eigenstate (stationary state) - E is the energy eigenvalue - These states don't change shape over time (only phase)

Time evolution of eigenstate:

$\psi(x,t) = \psi(x)e\hat{}(-iEt/\hbar)$

- Only phase changes: e^(-iEt/$\hbar$)
- Probability $|\psi|^2$ is time-independent

Physical Interpretation

What the Equation Tells Us

1. **Determinism**: Given initial state $\psi(0)$, evolution is determined
2. **Linearity**: If ψ_1 and ψ_2 are solutions, so is $\alpha\psi_1 + \beta\psi_2$
3. **Unitarity**: Probability is conserved: d/dt$\int|\psi|^2$dx = 0
4. **Energy generates time**: $\hat{H}$ is the generator of time evolution

What It Doesn't Tell Us

1. **Measurement outcomes**: Born rule ($|\langle\varphi|\psi\rangle|^2$) is separate postulate
2. **Collapse**: Measurement process not described
3. **Why complex numbers**: Fundamental mystery
4. **Relativistic effects**: Need Dirac equation or QFT

Examples

Free Particle (V = 0)

$i\hbar\ \partial\psi/\partial t = -\hbar^2/(2m)\ \partial^2\psi/\partial x^2$

Solution: Plane waves ψ(x,t) = Ae^(i(kx-ωt)) - k: wave vector (momentum p = $\hbar$k) - ω: angular frequency (energy E = $\hbar\omega$) - Dispersion relation: $\omega = \hbar k^2/(2m)$

Harmonic Oscillator (V = ½mω^2x^2)

`iℏ ∂ψ/∂t = [-ℏ²/(2m) ∂²/∂x² + ½mω²x²]ψ`

Energy eigenvalues: $E_n = \hbar\omega(n + ½)$ - Quantized in units of $\hbar\omega$ - Zero-point energy: $E_0 = \hbar\omega/2$ - Equally spaced levels

Hydrogen Atom (V = -e^2/r)

`iℏ ∂ψ/∂t = [-ℏ²/(2m)∇² - e²/r]ψ`

Energy eigenvalues: E_n = -13.6 eV/n^2 - Bound states: n = 1, 2, 3, ... - Ground state: n = 1 (lowest energy) - Explains atomic spectra

Deeper Connections

Relation to Classical Mechanics

Ehrenfest theorem: Quantum expectation values obey classical equations

```
d⟨x⟩/dt = ⟨p⟩/m
d⟨p⟩/dt = -⟨dV/dx⟩
```

Correspondence principle: Quantum $\rightarrow$ classical as $\hbar \rightarrow 0$ or large quantum numbers

Relation to Path Integrals

Feynman formulation: Sum over all paths

`ψ(x_f,t_f) = ∫ψ(x_i,t_i)e^(iS[path]/ℏ)D[path]`

- S: Classical action
- Equivalent to Schrödinger equation

Relation to Heisenberg Picture

Schrödinger picture: States evolve, operators fixed **Heisenberg picture**: Operators evolve, states fixed

`dÂ/dt = (i/ℏ)[Ĥ,Â] + ∂Â/∂t`

- Equivalent formulations
- Different perspectives on time evolution

Practice Problems

1. **Dimensional analysis**: Verify that both sides of $i\hbar\partial\psi/\partial t = \hat{H}\psi$ have the same units.
2. **Probability conservation**: Show that $d/dt\int|\psi|^2dx = 0$ using the Schrödinger equation.
3. **Stationary states**: If ψ(x,t) = φ(x)e^(-iEt/$\hbar$), show that φ(x) satisfies $\hat{H}\varphi = E\varphi$.
4. **Expectation values**: Show that $d\langle x\rangle/dt = \langle p\rangle/m$ using the Schrödinger equation.
5. **Uncertainty principle**: Derive $\Delta x\Delta p \geq \hbar/2$ from $[x,p] = i\hbar$.

Key Takeaways

- Schrödinger equation governs quantum time evolution
- $i\hbar$ ensures unitary evolution (probability conservation)
- Hamiltonian $\hat{H}$ = kinetic + potential energy
- Time-independent form gives energy eigenstates
- Linear equation → superposition principle
- Deterministic evolution, probabilistic measurement

Historical Note

Erwin Schrödinger (1926): Formulated wave mechanics - Inspired by de Broglie's matter waves - Equivalent to Heisenberg's matrix mechanics - Nobel Prize 1933 (shared with Dirac) - "Schrödinger's cat" thought experiment (1935)

Next Steps

Next, we'll decode QAOA—the Quantum Approximate Optimization Algorithm that uses quantum circuits to solve combinatorial problems.

Lesson 6.2: QAOA (Quantum Approximate Optimization Algorithm)

Overview

Complete decoding of QAOA—a hybrid quantum-classical algorithm for combinatorial optimization.

The Algorithm

QAOA State Preparation

`|ψ(β,γ)⟩ = U(β_p,γ_p)...U(β_1,γ_1)|+⟩^⊗n`

where each layer is:

`U(β,γ) = e^(-iβĤ_mixer)e^(-iγĤ_cost)`

Cost Function

`C(β,γ) = ⟨ψ(β,γ)|Ĥ_cost|ψ(β,γ)⟩`

Symbol-by-Symbol Breakdown

Initial State: `|+⟩^⊗n`

$|+\rangle$ (plus state) - **Definition**: $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ - **Meaning**: Equal superposition of 0 and 1 - **Preparation**: Apply Hadamard gate to $|0\rangle$

⊗n (tensor product, n times) - **Meaning**: n qubits, each in $|+\rangle$ state - **Result**: $|+\rangle$^⊗n = $|+\rangle_1 \otimes |+\rangle_2 \otimes \ldots \otimes |+\rangle_n$ - **Superposition**: Equal amplitude on all 2^n basis states - **Purpose**: Start with maximum uncertainty (explore all solutions)

Cost Hamiltonian: `Ĥ_cost`

Definition: Encodes optimization problem

`Ĥ_cost = Σ_edges J_ij σ_z^i σ_z^j + Σ_nodes h_i σ_z^i`

For MaxCut problem:

`Ĥ_cost = -½ Σ_(i,j)∈E (1 - σ_z^i σ_z^j)`

Breaking it down:

σ_z^i (Pauli Z on qubit i) - **Eigenvalues**: +1 (for $|0\rangle$), -1 (for $|1\rangle$) - **Meaning**: Measures qubit state - **Role**: Encodes binary variable (0 or 1)

σ_z^i σ_z^j (product of Z operators) - **Eigenvalues**: +1 (same state), -1 (different states) - **Meaning**: Measures agreement between qubits i and j - **MaxCut**: Reward different states (cut edge)

J__ij (coupling strength) - **Values**: Problem-dependent weights - **Meaning**: Importance of edge (i,j) - **Sign**: Negative for MaxCut (reward cuts)

Cost Unitary: `e^(-iγĤ_cost)`

e^(-iγĤ__cost) (exponential of cost Hamiltonian) - γ: Angle parameter (to be optimized) - **Operation**: Unitary transformation - **Effect**: Encodes problem structure into quantum state - **Intuition**: "Rotate toward low-cost states"

For diagonal Ĥ__cost:

`e^(-iγĤ_cost)|z⟩ = e^(-iγC(z))|z⟩`

- **C(z)**: Classical cost of bitstring z
- **Effect**: Phase proportional to cost
- **Low cost → small phase; high cost → large phase**

Mixer Hamiltonian: Ĥ_mixer

Standard mixer:

```
Ĥ_mixer = Σ_i σ_x^i
```

σ_x^i (Pauli X on qubit i) - **Operation**: Bit flip ($|0\rangle \leftrightarrow |1\rangle$) - **Eigenvalues**: ±1 - **Eigenstates**: $|+\rangle$, $|-\rangle$ - **Purpose**: Explores solution space

Why X? - X doesn't commute with Z (cost operator) - Creates superpositions - Enables quantum tunneling between solutions - Classical analog: random walk

Mixer Unitary: e^(-iβĤ_mixer)

e^(-iβĤ__mixer) (exponential of mixer) - β: Angle parameter (to be optimized) - **Operation**: Unitary transformation - **Effect**: Mixes computational basis states - **Intuition**: "Explore nearby solutions"

For Ĥ__mixer = Σ_i σ_x^i:

```
e^(-iβσ_x) = cos(β)I - i sin(β)σ_x
```

- $\beta = \mathbf{0}$: No mixing (identity)
- $\beta = \pi/\mathbf{2}$: Hadamard-like (maximum mixing)
- $\beta = \pi$: Full bit flip

QAOA Layer: U(β,γ) = e^(-iβĤ_mixer)e^(-iγĤ_cost)

Order matters! 1. **First**: Apply cost unitary e^(-iγĤ__cost) - Encodes problem structure - Adds phase based on cost 2. **Second**: Apply mixer unitary e^(-iβĤ__mixer) - Explores solution space - Creates superpositions

Intuition: "Bias toward good solutions, then explore"

Multiple Layers: U(β_p,γ_p)...U(β_1,γ_1)

p layers (circuit depth) - **p = 1**: Single layer (limited expressibility) - **p** → ∞: Can reach any state (universal) - **Trade-off**: More layers → better solutions but more noise

Parameters: 2p total (β_1, γ_1, β_2, γ_2, …, β__p, γ__p) - Each layer has independent parameters - Optimized classically

Final State: |ψ(β,γ)⟩

Full expression:

```
|ψ(β,γ)⟩ = e^(-iβ_pĤ_mixer)e^(-iγ_pĤ_cost)...e^(-iβ_1Ĥ_mixer)e^(-iγ_1Ĥ_cost)|+⟩^⊗n
```

Properties: - Superposition of all 2^n bitstrings - Amplitudes depend on β, γ - Good solutions have higher amplitude (ideally)

Cost Function: C(β,γ) = ⟨ψ(β,γ)|Ĥ_cost|ψ(β,γ)⟩

⟨ψ(β,γ)| (bra) - Dual of |ψ(β,γ)⟩ - Used for inner product

Ĥ__cost (cost Hamiltonian) - Same as before - Observable to measure

⟨ψ(β,γ)|Ĥ__cost|ψ(β,γ)⟩ (expectation value) - **Meaning**: Average cost of state |ψ(β,γ)⟩ - **Measurement**: Sample bitstrings, compute average cost - **Goal**: Minimize C(β,γ) over β, γ

The Complete QAOA Workflow

Step 1: Problem Encoding

- Define cost function C(z) for bitstring z
- Construct Ĥ_cost such that eigenvalues are costs
- Example (MaxCut): Ĥ_cost = -½Σ_(i,j)(1 - σ_z^i σ_z^j)

Step 2: Initialize Parameters

- Choose p (number of layers)
- Initialize β = (β_1, ..., β_p), γ = (γ_1, ..., γ_p)
- Common: random or heuristic initialization

Step 3: Quantum Circuit

- Prepare |+⟩^⊗n
- For each layer k = 1 to p:
  - Apply e^(-iγ_kĤ_cost)
  - Apply e^(-iβ_kĤ_mixer)
- Measure in computational basis

Step 4: Cost Evaluation

- Repeat circuit N_shots times
- Collect bitstrings {z_1, z_2, ..., z_N}
- Estimate: C(β,γ) ≈ (1/N)Σ_i C(z_i)

Step 5: Classical Optimization

- Compute gradient: ∂C/∂β_k, ∂C/∂γ_k (parameter-shift rule)
- Update: $\beta \leftarrow \beta - \alpha \cdot \partial C/\partial\beta$, $\gamma \leftarrow \gamma - \alpha \cdot \partial C/\partial\gamma$
- Repeat until convergence

Step 6: Solution Extraction

- Run final circuit with optimal β, γ
- Sample bitstrings
- Return best bitstring (lowest cost)

Example: MaxCut on Triangle Graph

Problem

- **Graph**: 3 vertices, 3 edges (triangle)
- **Goal**: Maximize number of cut edges
- **Optimal**: 2 edges cut (any partition with 1 vs 2 vertices)

Cost Hamiltonian

```
Ĥ_cost = -½[(1 - σ_z^0 σ_z^1) + (1 - σ_z^1 σ_z^2) + (1 - σ_z^2 σ_z^0)]
       = -3/2 + ½(σ_z^0 σ_z^1 + σ_z^1 σ_z^2 + σ_z^2 σ_z^0)
```

Mixer Hamiltonian

```
Ĥ_mixer = σ_x^0 + σ_x^1 + σ_x^2
```

QAOA Circuit (p=1)

1. **Initialize**: $H \otimes H \otimes H|000\rangle = |{+}{+}{+}\rangle$
2. **Cost layer**: e^(-i$\gamma\hat{H}$_cost)
   - Diagonal in Z basis
   - Adds phases based on cost
3. **Mixer layer**: e^(-i$\beta\hat{H}$_mixer)
   - Product of single-qubit rotations
   - e^(-i$\beta\sigma$_x) on each qubit

Optimization

- **Parameters**: β, γ (2 parameters for p=1)
- **Landscape**: C(β,γ) has local minima
- **Optimal**: Found via gradient descent or grid search
- **Result**: $|\psi(\beta,\gamma)\rangle$ has high amplitude on good solutions

Physical Intuition

Adiabatic Connection

- **QAOA ≈ discretized adiabatic evolution**
- **p** $\to \infty$: Approaches adiabatic algorithm
- **Finite p**: Approximate, but faster

Quantum Annealing Analogy

- **Cost layer**: Encodes problem (like problem Hamiltonian)
- **Mixer layer**: Provides quantum fluctuations (like transverse field)
- **Alternation**: Discrete time steps (vs continuous annealing)

Classical Analog

- **Cost layer**: Evaluate objective function
- **Mixer layer**: Random walk / local search
- **Quantum advantage**: Tunneling through barriers

Performance and Limitations

Approximation Ratio

- **Definition**: C_QAOA / C_optimal
- **Depends on**: p (layers), problem instance, optimization quality
- **Typical**: Better than random, worse than best classical (for small p)

Noise Sensitivity

- **Gate errors**: Accumulate over layers
- **Decoherence**: Limits effective p
- **Mitigation**: Error mitigation, noise-aware optimization

Scalability

- **Circuit depth**: O(p) (shallow for fixed p)
- **Parameters**: 2p (grows slowly)
- **Measurements**: O(number of terms in $\hat{H}$_cost)

Practice Problems

1. **MaxCut cost**: For bitstring z = 010 on triangle graph, compute C(z).
2. **Expectation value**: Show that $\langle\psi|\sigma$__zˆi σ__zˆj$|\psi\rangle$ = 2P(same) - 1.
3. **Gradient**: Derive ∂C/$\partial\gamma$ using parameter-shift rule.
4. **Optimal p=1**: For 2-qubit MaxCut, find optimal β, γ analytically.
5. **Mixer design**: Why is $\hat{\mathrm{H}}$_mixer = $\Sigma\sigma$__xˆi a good choice? What if we used $\Sigma\sigma$__zˆi?

Key Takeaways

- QAOA alternates cost and mixer unitaries
- Cost layer encodes problem structure
- Mixer layer explores solution space
- Parameters β, γ optimized classically
- More layers (p) $\rightarrow$ better approximation
- Hybrid quantum-classical algorithm
- Applicable to combinatorial optimization problems

Next Steps

Next, we'll decode VQE—the Variational Quantum Eigensolver for finding ground states of quantum systems.

Lesson 6.3: VQE (Variational Quantum Eigensolver)

Overview

Complete decoding of VQE—the variational quantum eigensolver for finding ground states and energies of quantum systems.

The Core Equations

Energy Expectation

$E(\theta) = \langle\psi(\theta)|\hat{H}|\psi(\theta)\rangle$

Variational Principle

$E(\theta) \geq E_0$ (ground state energy)

Optimization Goal

θ* = argmin_θ E(θ)

Symbol-by-Symbol Breakdown

Parameterized State: $|\psi(\theta)\rangle$

$|\psi(\theta)\rangle$ (trial state) - **Form**: $|\psi(\theta)\rangle = U(\theta)|0\rangle$^⊗n - **U(θ)**: Parameterized quantum circuit (ansatz) - θ: Classical parameters $\theta = (\theta_1, \theta_2, \ldots,$ θ__p) - **Purpose**: Explore Hilbert space to find ground state

Ansatz U(θ) - **Structure**: Product of parameterized gates - **Example**: U(θ) = ∏__k e^(-iθ__k P__k) (Pauli rotations) - **Design**: Hardware-efficient or problem-inspired - **Trade-off**: Expressibility vs depth (noise)

Hamiltonian: $\hat{H}$

$\hat{\mathbf{H}}$ (Hamiltonian operator) - **Meaning**: Energy operator of quantum system - **Hermitian**: $\hat{H}^\dagger = \hat{H}$ (real eigenvalues) - **Eigenvalues**: $\{E_0, E_1, E_2, \ldots\}$ (energy levels) - **Ground state**: $|\psi_0\rangle$ with lowest eigenvalue E_0

Pauli Decomposition:

$\hat{H} = \Sigma_i\ h_i\ P_i$

- $\mathbf{P}_i$: Pauli strings (tensor products of I, X, Y, Z)
- $\mathbf{h}_i$: Real coefficients
- **Example**: $\hat{H} = 0.5 \cdot Z_0Z_1 + 0.3 \cdot X_0 - 0.2 \cdot Y_1$

Energy Expectation: $\langle\psi(\theta)|\hat{H}|\psi(\theta)\rangle$

$\langle\psi(\theta)|$ (bra) - **Definition**: Dual vector (conjugate transpose of $|\psi(\theta)\rangle$) - **Purpose**: Forms inner product

$\hat{\mathbf{H}}|\psi(\theta)\rangle$ (Hamiltonian acting on state) - **Operation**: Linear transformation - **Result**: Another quantum state - **Meaning**: "Energy operator applied to trial state"

$\langle\psi(\theta)|\hat{\mathbf{H}}|\psi(\theta)\rangle$ (expectation value) - **Definition**: $\langle\hat{H}\rangle = \Sigma_i\ h_i\langle P_i\rangle$ - **Meaning**: Average energy if we measured many times - **Real-valued**: Because $\hat{H}$ is Hermitian - **Units**: Energy (Joules, eV, Hartree, etc.)

Measurement of Expectation Value

For each Pauli string $\mathbf{P}_i$:

$\langle P_i\rangle = \langle\psi(\theta)|P_i|\psi(\theta)\rangle$

Measurement process: 1. Prepare state $|\psi(\theta)\rangle$ 2. Rotate to eigenbasis of P_i 3. Measure in computational basis 4. Outcome: ± 1 (eigenvalues of Pauli operators) 5. Repeat N times, estimate: $\langle P_i \rangle \approx (n_+ - n_-)/N$

Total energy:

`E(θ) = Σ_i h_i⟨P_i⟩`

- Sum of weighted expectation values
- Each term measured separately (or grouped if commuting)

Variational Principle: $E(\theta) \geq E_0$

Statement: Any trial state gives upper bound on ground energy

Proof sketch: - Expand: $|\psi(\theta)\rangle = \Sigma_n\ c_n|n\rangle$ (eigenbasis of $\hat{H}$) - Energy: $E(\theta) = \Sigma_n\ |c_n|^2 E_n$ - Since $E_n \geq E_0$ and $\Sigma_n|c_n|^2 = 1$: - $E(\theta) = \Sigma_n|c_n|^2 E_n \geq \Sigma_n|c_n|^2 E_0 = E_0$

Consequence: Minimizing $E(\theta)$ approaches E_0

Equality: $E(\theta) = E_0$ iff $|\psi(\theta)\rangle = |\psi_0\rangle$ (ground state)

Optimization: θ* = argmin_θ E(θ)

Goal: Find parameters that minimize energy

Gradient-based optimization:

`θ_new = θ_old - α·∇E(θ_old)`

- α: Learning rate
- **∇E**: Gradient (vector of partial derivatives)

Gradient computation (parameter-shift rule):

`∂E/∂θ_i = [E(θ + π/2·e_i) - E(θ - π/2·e_i)]/2`

- $\mathbf{e}_i$: Unit vector in direction i
- **Exact**: For Pauli generators
- **Cost**: 2p circuit evaluations (p parameters)

Complete VQE Workflow

Step 1: Problem Setup

- **Define Hamiltonian**: $\hat{H}$ for system of interest
- **Example**: Molecular Hamiltonian for H_2, LiH, etc.
- **Decompose**: $\hat{H} = \Sigma_i h_i\ P_i$ (Pauli strings)

Step 2: Ansatz Design

- **Choose structure**: UCC, hardware-efficient, etc.
- **Determine depth**: Balance expressibility and noise
- **Initialize** θ: Random, classical guess, or heuristic

Step 3: Energy Evaluation

For given θ: 1. Prepare $|\psi(\theta)\rangle$ on quantum computer 2. Measure each $\langle P_i \rangle$ (with shots) 3. Compute $E(\theta) = \Sigma_i h_i \langle P_i \rangle$

Step 4: Gradient Computation

For each parameter θ_i: 1. Evaluate E(θ + $\pi/2 \cdot e_i$) 2. Evaluate E(θ - $\pi/2 \cdot e_i$) 3. Compute $\partial E/\partial\theta_i$ = [E(+) - E(-)]/2

Step 5: Parameter Update

```
θ ← θ - α·∇E(θ)
```

- Classical optimization step
- Various optimizers: GD, Adam, BFGS, etc.

Step 6: Convergence Check

- **Criterion**: |E_new - E_old| < ε
- **Or**: ||∇E|| < ε (gradient vanishes)
- **Or**: Maximum iterations reached

Step 7: Solution Extraction

- **Optimal parameters**: θ*
- **Ground state**: $|\psi(\theta^*)\rangle$
- **Ground energy**: E(θ*) ≈ E_0

Example: H_2 Molecule

Hamiltonian (STO-3G basis, 0.74 Å)

```
Ĥ = -1.0523·I + 0.3979·Z₀ - 0.3979·Z₁ - 0.0112·Z₀Z₁
    + 0.1809·X₀X₁ + 0.1809·Y₀Y₁
```

Terms: - **-1.0523 · I**: Constant energy shift - **0.3979 · Z_0, -0.3979 · Z_1**: One-body terms - **-0.0112 · Z_0Z_1**: Coulomb interaction - **0.1809 · X_0X_1, 0.1809 · Y_0Y_1**: Exchange interaction

Ansatz (UCC Singles and Doubles)

```
U(θ) = e^(-iθ(X₀Y₁ - Y₀X₁)/2)
```

- **Single parameter**: θ (excitation amplitude)
- **Operator**: X_0Y_1 - Y_0X_1 (single excitation)
- **Initial state**: |01⟩ (Hartree-Fock)

Circuit Implementation

1. **Prepare |01⟩**: X gate on qubit 1
2. **Apply U(θ)**:
    - $R_\gamma(\theta)$ on qubit 0
    - CNOT (0 → 1)
    - $R_\gamma(-\theta)$ on qubit 1
    - CNOT (0 → 1)

Energy Landscape

- **E(θ)**: Smooth function of θ
- **Minimum**: θ* ≈ 0.2 radians
- **Ground energy**: E(θ*) ≈ -1.137 Ha
- **Exact**: -1.137 Ha (chemical accuracy achieved!)

Optimization Trace

```
Iteration 0: θ = 0.0,    E = -1.116 Ha
Iteration 1: θ = 0.05,   E = -1.130 Ha
Iteration 2: θ = 0.10,   E = -1.135 Ha
...
Iteration 20: θ = 0.20,  E = -1.137 Ha (converged)
```

Advanced Topics

Excited States

- **Orthogonality constraint**: $\langle\psi(\theta)|\psi_0\rangle = 0$
- **Penalty method**: $\mathrm{E}(\theta) + \lambda|\langle\psi(\theta)|\psi_0\rangle|^2$
- **Subspace search**: Find multiple eigenstates

Adaptive VQE

- **Grow ansatz**: Add layers when needed
- **Prune parameters**: Remove unimportant ones
- **Balance**: Expressibility vs circuit depth

Quantum Natural Gradient

- **Metric**: Fisher information matrix F
- **Update**: $\theta \leftarrow \theta$ - $\alpha \cdot \mathrm{F}^{-1}\nabla\mathrm{E}$
- **Benefit**: Accounts for parameter space geometry
- **Faster convergence** than standard gradient descent

Error Mitigation

- **Zero-noise extrapolation**: Extrapolate to zero noise
- **Probabilistic error cancellation**: Invert noise channel
- **Symmetry verification**: Check physical constraints

Challenges and Solutions

Challenge 1: Barren Plateaus

- **Problem**: Gradients vanish exponentially
- **Solution**: Problem-inspired ansätze, local cost functions

Challenge 2: Local Minima

- **Problem**: Optimizer gets stuck
- **Solution**: Multiple initializations, simulated annealing

Challenge 3: Measurement Noise

- **Problem**: Finite shots → noisy gradients
- **Solution**: Adaptive shot allocation, more shots

Challenge 4: Hardware Noise

- **Problem**: Gate errors, decoherence
- **Solution**: Error mitigation, shorter circuits

Practice Problems

1. **Variational principle**: Prove $E(\theta) \geq E_0$ for any normalized $|\psi(\theta)\rangle$.
2. **Gradient formula**: Derive the parameter-shift rule for $\partial E/\partial\theta$.
3. **H_2 energy**: Compute $E(\theta=0)$ for the H_2 Hamiltonian above with $|\psi(0)\rangle = |01\rangle$.
4. **Measurement**: How many shots needed to estimate $\langle P_i \rangle$ to precision ε?
5. **Ansatz expressibility**: Can a single-parameter ansatz reach all 2-qubit states?

Key Takeaways

- VQE finds ground states via variational principle
- Energy expectation $E(\theta) = \langle\psi(\theta)|\hat{H}|\psi(\theta)\rangle$ is cost function
- Variational principle: $E(\theta) \geq E_0$ (upper bound)
- Parameter-shift rule computes exact gradients
- Hybrid quantum-classical: quantum state prep, classical optimization
- Applications: quantum chemistry, materials science, condensed matter

Historical Note

VQE introduced by Peruzzo et al. (2014) - First demonstrated on photonic system - Calculated H_2 ground state energy - Pioneered variational quantum algorithms - Foundation for NISQ-era quantum computing

Next Steps

Next, we'll decode Maxwell's equations—the fundamental equations of classical electromagnetism.

Lesson 6.4: Maxwell's Equations

Overview

Complete decoding of Maxwell's equations—the four fundamental equations that describe all of classical electromagnetism.

The Four Equations

Differential Form

```
1. ∇·E = ρ/ε₀            (Gauss's Law)
2. ∇·B = 0               (No magnetic monopoles)
3. ∇×E = -∂B/∂t          (Faraday's Law)
4. ∇×B = μ₀J + μ₀ε₀∂E/∂t  (Ampère-Maxwell Law)
```

Integral Form

```
1. ∮ E·dA = Q_enc/ε₀
2. ∮ B·dA = 0
3. ∮ E·dl = -dΦ_B/dt
4. ∮ B·dl = μ₀I_enc + μ₀ε₀dΦ_E/dt
```

Symbol-by-Symbol Breakdown

Common Symbols

E (electric field) - **Vector field**: $E(r,t) = [E_x, E_\gamma, E_u]$ - **Units**: V/m (volts per meter) or N/C (newtons per coulomb) - **Physical meaning**: Force per unit charge - **Source**: Charges (static and moving)

B (magnetic field) - **Vector field**: $B(r,t) = [B_x, B_\gamma, B_u]$ - **Units**: T (tesla) or Wb/m^2 (weber per square meter) - **Physical meaning**: Force on moving charges - **Source**: Moving charges (currents)

ρ (charge density) - **Scalar field**: $\rho(r,t)$ - **Units**: C/m^3 (coulombs per cubic meter) - **Physical meaning**: Charge per unit volume - **Relation to charge**: $Q = \int \rho \, dV$

J (current density) - **Vector field**: $J(r,t) = [J_x, J_\gamma, J_u]$ - **Units**: A/m^2 (amperes per square meter) - **Physical meaning**: Current per unit area - **Relation to current**: $I = \int J \cdot dA$

ε_0 (permittivity of free space) - **Value**: 8.854×10^{-12} F/m (farads per meter) - **Physical meaning**: How much electric field is produced by charges - **Role**: Couples charges to electric field

μ_0 (permeability of free space) - **Value**: $4\pi \times 10^{-7}$ H/m (henries per meter) - **Physical meaning**: How much magnetic field is produced by currents - **Role**: Couples currents to magnetic field

c (speed of light) - **Value**: $c = 1/\sqrt{(\mu_0\varepsilon_0)} \approx 3 \times 10^8$ m/s - **Emerges from**: Maxwell's equations - **Significance**: Electromagnetic waves travel at c

Equation 1: Gauss's Law

Differential Form: $\nabla \cdot E = \rho/\varepsilon_0$

$\nabla \cdot$ **E** (divergence of E) - **Definition**: $\nabla \cdot E = \partial E_x/\partial x + \partial E_\gamma/\partial y + \partial E_u/\partial z$ - **Meaning**: "Outflow" of electric field from a point - **Positive**: Field lines emanate (source) - **Negative**: Field lines converge (sink)

ρ/ε_0 (charge density scaled) - **Meaning**: Charge density creates divergence - **Positive** ρ: Electric field points outward - **Negative** ρ: Electric field points inward

Physical interpretation: - "Electric field lines begin and end on charges" - Positive charges are sources (field lines out) - Negative charges are sinks (field lines in) - No charges → no divergence (field lines continuous)

Integral Form: $\oint$ `E·dA = Q_enc/`ε_0

$\oint$ **E · dA** (flux through closed surface) - **E · dA**: Electric field dot surface element - $\oint$: Integral over closed surface - **Meaning**: Total electric field passing through surface

Q_enc (enclosed charge) - **Definition**: Q_enc = $\int$_V ρ dV - **Meaning**: Total charge inside surface

Physical interpretation: - "Total flux through surface equals enclosed charge" - Gauss's law for calculating E from symmetric charge distributions - Divergence theorem connects differential and integral forms

Equation 2: No Magnetic Monopoles

Differential Form: ∇`·B = 0`

∇ **· B** (divergence of B) - **Always zero**: No magnetic charges - **Meaning**: Magnetic field lines never begin or end - **Consequence**: Magnetic field lines form closed loops

Physical interpretation: - "No magnetic monopoles exist" - Magnetic field lines always form closed loops - Every north pole has a south pole - Fundamental asymmetry between E and B

Integral Form: $\oint$ `B·dA = 0`

$\oint$ **B · dA** (magnetic flux through closed surface) - **Always zero**: What goes in must come out - **Meaning**: No net magnetic flux through any closed surface

Physical interpretation: - "Magnetic field lines cannot begin or end" - If field lines enter surface, they must exit - No isolated magnetic charges

Equation 3: Faraday's Law

Differential Form: $\nabla\times$`E = -`∂`B/`∂`t`

$\nabla\times$**E** (curl of E) - **Definition**: $\nabla\times$E = [$\partial E_u/\partial$y - $\partial E_\gamma/\partial$z, $\partial E_x/\partial$z - $\partial E_u/\partial$x, $\partial E_\gamma/\partial$x - $\partial E_x/\partial$y] - **Meaning**: "Circulation" or "rotation" of electric field - **Non-zero**: Electric field has vortex structure

-∂B/∂t (negative time derivative of B) - **Meaning**: Changing magnetic field - **Negative sign**: Lenz's law (opposes change)

Physical interpretation: - "Changing magnetic field creates circulating electric field" - Basis of electromagnetic induction - Generators, transformers, inductors

Integral Form: $\oint$ `E·dl = -d`Φ`_B/dt`

$\oint$ **E · dl** (line integral of E around closed loop) - **Meaning**: Electromotive force (EMF) around loop - **Units**: Volts

Φ_B (magnetic flux) - **Definition**: Φ_B = $\int$B · dA - **Units**: Wb (webers) - **Meaning**: Total magnetic field through loop

-dΦ_B/dt (rate of change of flux) - **Meaning**: How fast magnetic flux is changing - **Negative sign**: Induced EMF opposes change (Lenz's law)

Physical interpretation: - "Changing magnetic flux induces EMF" - Faraday's law of induction - Moving magnet through coil $\rightarrow$ current

Equation 4: Ampère-Maxwell Law

Differential Form: $\nabla\times$`B = `μ_0`J + `$\mu_0\varepsilon_0\partial$`E/`$\partial$`t`

$\nabla\times$**B** (curl of B) - **Meaning**: Circulation of magnetic field - **Sources**: Currents and changing electric fields

μ_0**J** (current term) - **Ampère's law**: Currents create circulating magnetic field - **Right-hand rule**: Curl fingers with current, thumb points along B

$\mu_0\varepsilon_0\partial$**E/**$\partial$**t** (displacement current) - **Maxwell's addition**: Changing electric field acts like current - **Crucial**: Enables electromagnetic waves - **Symmetry**: Completes parallel with Faraday's law

Physical interpretation: - "Magnetic field circulates around currents and changing electric fields" - Currents create magnetic fields (Ampère) - Changing E creates magnetic fields (Maxwell's insight) - Displacement current enables wave propagation

Integral Form: $\oint$ `B·dl = `μ_0`I_enc + `$\mu_0\varepsilon_0$`dΦ_E/dt`

$\oint$ **B · dl** (line integral of B around closed loop) - **Meaning**: Circulation of magnetic field - **Related to**: Current enclosed by loop

μ_0**I_enc** (enclosed current) - **Definition**: I_enc = $\int$J · dA - **Meaning**: Total current through surface bounded by loop

$\mu_0\varepsilon_0$**dΦ_E/dt** (displacement current term) - **Φ_E**: Electric flux through surface - **dΦ_E/dt**: Rate of change of electric flux - **Acts like current**: Even without physical charges moving

Physical interpretation: - "Magnetic field circulates around currents" - Ampère's law for calculating B from currents - Displacement current: changing E field equivalent to current

Electromagnetic Waves

Wave Equation from Maxwell

Taking curl of Faraday's law and using Ampère-Maxwell:

$$\nabla^2 E - \mu_0\varepsilon_0\partial^2 E/\partial t^2 = 0$$
$$\nabla^2 B - \mu_0\varepsilon_0\partial^2 B/\partial t^2 = 0$$

Wave speed:

$$c = 1/\sqrt{(\mu_0\varepsilon_0)} \approx 3 \times 10^8 \text{ m/s}$$

Physical meaning: - E and B satisfy wave equation - Waves propagate at speed c (light!) - E and B oscillate perpendicular to each other and to propagation direction

Plane Wave Solution

$$E = E_0 \sin(k\cdot r - \omega t)$$
$$B = B_0 \sin(k\cdot r - \omega t)$$

Properties: - **E** $\perp$ **B**: Electric and magnetic fields perpendicular - **E** $\perp$ **k, B** $\perp$ **k**: Both perpendicular to propagation direction - **|E| = c|B|**: Magnitudes related by c - ω **= ck**: Dispersion relation

Energy and Momentum

Energy Density

$$u = ½(\varepsilon_0 E^2 + B^2/\mu_0)$$

- Electric energy: $½\varepsilon_0 E^2$
- Magnetic energy: $½B^2/\mu_0$

Poynting Vector (Energy Flux)

$$S = (1/\mu_0)E\times B$$

- **Meaning**: Energy flow per unit area

- **Direction**: $E \times B$ (perpendicular to both)
- **Units**: W/m^2 (watts per square meter)

Momentum Density

$g = \varepsilon_0 E \times B = S/c^2$

- Electromagnetic fields carry momentum
- Radiation pressure

Conservation Laws

Charge Conservation

$\partial\rho/\partial t + \nabla \cdot J = 0$

- **Meaning**: Charge is conserved
- **Derived from**: Maxwell's equations (take divergence of Ampère-Maxwell)

Energy Conservation (Poynting's Theorem)

$\partial u/\partial t + \nabla \cdot S = -J \cdot E$

- **u**: Energy density
- **S**: Poynting vector (energy flux)
- **-J · E**: Work done by field on charges

Applications

Static Fields ($\partial/\partial t = 0$)

Electrostatics: - $\nabla \cdot E = \rho/\varepsilon_0$ (Gauss) - $\nabla \times E = 0$ (no curl → $E = -\nabla\varphi$)

Magnetostatics: - $\nabla \cdot B = 0$ (no monopoles) - $\nabla \times B = \mu_0 J$ (Ampère)

Electromagnetic Induction

- Generators: Rotating coil in magnetic field
- Transformers: Changing current in primary → induced EMF in secondary
- Inductors: Changing current → self-induced EMF

Electromagnetic Waves

- Radio waves, microwaves, infrared, visible light, UV, X-rays, gamma rays
- All described by Maxwell's equations
- Different frequencies, same physics

Practice Problems

1. **Gauss's law**: Calculate E field of point charge Q using integral form.
2. **Faraday's law**: A circular loop of radius r is in a uniform B field increasing at rate dB/dt. Find induced EMF.
3. **Ampère's law**: Calculate B field inside a long straight wire carrying current I.
4. **Wave equation**: Derive $\nabla^2 E = \mu_0\varepsilon_0 \partial^2 E/\partial t^2$ from Maxwell's equations.
5. **Poynting vector**: For a plane wave with $E = E_0 \sin(kz-\omega t)x$, find S.

Key Takeaways

- Four equations describe all classical electromagnetism
- Gauss: Charges create electric field (divergence)
- No monopoles: Magnetic field lines form closed loops
- Faraday: Changing B creates circulating E (induction)
- Ampère-Maxwell: Currents and changing E create circulating B
- Together: Predict electromagnetic waves at speed c
- Unified electricity, magnetism, and light

Historical Note

James Clerk Maxwell (1865): Published unified theory - Combined known laws (Gauss, Faraday, Ampère) - Added displacement current term ($\mu_0\varepsilon_0\partial$E/$\partial$t) - Predicted electromagnetic waves - **Heinrich Hertz** (1887): Experimentally confirmed waves - Foundation of modern physics and technology

Next Steps

Next, we'll decode logistic regression—the fundamental equation of binary classification in machine learning.

Lesson 6.5: Logistic Regression

Overview

Complete decoding of logistic regression—the fundamental model for binary classification in machine learning.

The Core Equations

Model (Prediction)

```
ŷ = σ(w^T x + b) = 1/(1 + e^(-(w^T x + b)))
```

Loss Function (Binary Cross-Entropy)

```
L(w,b) = -1/N Σᵢ[yᵢ log(ŷᵢ) + (1-yᵢ)log(1-ŷᵢ)]
```

Gradient Descent Update

```
w ← w - α ∂L/∂w
b ← b - α ∂L/∂b
```

Symbol-by-Symbol Breakdown

Input and Output

x (feature vector) - **Form**: x = [x_1, x_2, . . . , x_d]^T - **Dimension**: d features - **Example**: [age, income, credit_score] - **Meaning**: Input data describing an instance

y (true label) - **Values**: 0 or 1 (binary classification) - **Example**: 0 = not spam, 1 = spam - **Meaning**: Ground truth class

ŷ (predicted probability) - **Range**: [0, 1] - **Meaning**: P(y=1|x) (probability of class 1) - **Interpretation**: Confidence in positive class

Model Parameters

w (weight vector) - **Form**: w = [w_1, w_2, . . . , w_d]^T - **Meaning**: Importance of each feature - **Learned**: Optimized during training - **Interpretation**: $w_i > 0 \rightarrow$ feature i increases probability of class 1

b (bias term) - **Scalar**: Single number - **Meaning**: Baseline log-odds (intercept) - **Learned**: Optimized during training - **Interpretation**: Shifts decision boundary

Linear Combination: `z = w^T x + b`

w^T x (dot product) - **Expansion**: w^T x = w_1x_1 + w_2x_2 + . . . + w_dx_d - **Meaning**: Weighted sum of features - **Units**: Depends on features and weights

z (logit or log-odds) - **Range**: ($-\infty$, $+\infty$) - **Meaning**: Log of odds ratio - **Interpretation**: - z > 0 → more likely class 1 - z < 0 → more likely class 0 - z = 0 → equal probability

Sigmoid Function: `σ(z) = 1/(1 + e^(-z))`

σ (sigmoid activation) - **Domain**: $z \in (-\infty, +\infty)$ - **Range**: $\sigma(z) \in (0, 1)$ - **Properties**: - $\sigma(0) = 0.5$ - $\sigma(z) \rightarrow 1$ as $z \rightarrow +\infty$ - $\sigma(z) \rightarrow 0$ as $z \rightarrow -\infty$ - $\sigma(-z) = 1 - \sigma(z)$

Why sigmoid? - Squashes real line to [0,1] (valid probability) - Smooth and differentiable - Derivative: $\sigma'(z) = \sigma(z)(1 - \sigma(z))$ - Probabilistic interpretation (logistic distribution)

Alternative forms:

σ(z) = e^z/(1 + e^z)
σ(z) = 1/(1 + e^(-z))

Prediction: ŷ = σ(w^T x + b)

Complete model:

ŷ = P(y=1|x;w,b) = 1/(1 + e^(-(w^T x + b)))

Decision rule: - If $\hat{y} > 0.5 \rightarrow$ predict class 1 - If $\hat{y} < 0.5 \rightarrow$ predict class 0 - Threshold can be adjusted based on application

Interpretation: - Linear decision boundary in feature space - Boundary: w^T x + b = 0 - Probabilistic output (not just hard classification)

Loss Function

Binary Cross-Entropy: L = -1/N Σ_i[y_i log($\hat{y}_i$) + (1-y_i)log(1-$\hat{y}_i$)]

Breaking it down:

For single example:

ℓ(ŷ,y) = -[y log(ŷ) + (1-y)log(1-ŷ)]

If y = 1 (true class is 1):

ℓ = -log(ŷ)

- Penalizes low predicted probability
- $\hat{y} \rightarrow 1$: loss $\rightarrow 0$ (good)
- $\hat{y} \rightarrow 0$: loss $\rightarrow \infty$ (bad)

If y = 0 (true class is 0):

ℓ = -log(1-ŷ)

- Penalizes high predicted probability
- $\hat{y} \rightarrow 0$: loss $\rightarrow 0$ (good)
- $\hat{y} \rightarrow 1$: loss $\rightarrow \infty$ (bad)

Average over dataset:

L = 1/N Σ_i ℓ($\hat{y}_i$, y_i)

- N: number of training examples
- Sum over all examples
- Average loss (mean)

Why Cross-Entropy?

Maximum likelihood: - Equivalent to maximizing likelihood of data - Assumes Bernoulli distribution for y - Convex loss function (single global minimum)

Gradient properties: - Smooth and differentiable - Gradients don't vanish (unlike squared error) - Penalizes confident wrong predictions heavily

Connection to KL divergence:

L = D_KL(P_true || P_model) + H(P_true)

- Minimizing cross-entropy = minimizing KL divergence
- H(P_true) is constant (entropy of true distribution)

Gradients

Gradient of Loss w.r.t. Weights

Chain rule:

$\partial L/\partial w = \partial L/\partial \hat{y} \cdot \partial \hat{y}/\partial z \cdot \partial z/\partial w$

Step 1: $\partial L/\partial \hat{y}$

$\partial L/\partial \hat{y} = -y/\hat{y} + (1-y)/(1-\hat{y})$

Step 2: $\partial \hat{y}/\partial z$ (sigmoid derivative)

$\partial \hat{y}/\partial z = \sigma(z)(1 - \sigma(z)) = \hat{y}(1-\hat{y})$

Step 3: $\partial z/\partial w$

$\partial z/\partial w = x$

Combining (beautiful cancellation):

$\partial L/\partial w = 1/N\ \Sigma_i(\hat{y}_i - y_i)x_i$

Interpretation: - Gradient = (prediction error) × (input) - Large error → large gradient - Correct prediction ($\hat{y} = y$) → zero gradient

Gradient of Loss w.r.t. Bias

Similarly:

$\partial L/\partial b = 1/N\ \Sigma_i(\hat{y}_i - y_i)$

Interpretation: - Average prediction error - Shifts decision boundary

Gradient Descent Update

Update Rules

Weights:

$w \leftarrow w - \alpha\ \partial L/\partial w = w - \alpha/N\ \Sigma_i(\hat{y}_i - y_i)x_i$

Bias:

$b \leftarrow b - \alpha\ \partial L/\partial b = b - \alpha/N\ \Sigma_i(\hat{y}_i - y_i)$

α (learning rate) - **Hyperparameter**: Controls step size - **Too large**: Oscillation, divergence - **Too small**: Slow convergence - **Typical values**: 0.001 to 0.1

Variants

Batch Gradient Descent: - Use all N examples per update - Exact gradient - Slow for large datasets

Stochastic Gradient Descent (SGD): - Use single example per update - Noisy gradient - Fast, online learning

Mini-batch Gradient Descent: - Use batch of B examples (e.g., B=32) - Balance between batch and SGD - Most common in practice

Complete Training Algorithm

Step 1: Initialize

```
w ← random small values (e.g., N(0, 0.01))
b ← 0
```

Step 2: Forward Pass

For each example i:

$$z_i = w^T x_i + b$$
$$\hat{y}_i = \sigma(z_i)$$

Step 3: Compute Loss

$$L = -1/N \, \Sigma_i[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)]$$

Step 4: Backward Pass (Compute Gradients)

$$\partial L/\partial w = 1/N \, \Sigma_i(\hat{y}_i - y_i)x_i$$
$$\partial L/\partial b = 1/N \, \Sigma_i(\hat{y}_i - y_i)$$

Step 5: Update Parameters

$$w \leftarrow w - \alpha \, \partial L/\partial w$$
$$b \leftarrow b - \alpha \, \partial L/\partial b$$

Step 6: Repeat

- Until convergence (loss stops decreasing)
- Or maximum iterations reached

Example: Spam Classification

Problem Setup

- **Features**: x = [word_count_"free", word_count_"money", email_length]
- **Label**: y = 1 (spam), y = 0 (not spam)
- **Goal**: Predict P(spam|email)

Training Data (simplified)

x_1 = [5, 3, 100], y_1 = 1 (spam)
x_2 = [0, 0, 200], y_2 = 0 (not spam)
x_3 = [2, 1, 150], y_3 = 1 (spam)

After Training

```
w = [0.8, 0.6, -0.002]  (learned weights)
b = -1.5                 (learned bias)
```

Interpretation: - "free" increases spam probability ($w_1 = 0.8 > 0$) - "money" increases spam probability ($w_2 = 0.6 > 0$) - Longer emails slightly decrease spam probability ($w_3 = -0.002 < 0$) - Baseline bias: b = -1.5 (slightly biased toward not spam)

Prediction on New Email

x_new = [4, 2, 120]
$z = w^T x_{new} + b = 0.8(4) + 0.6(2) - 0.002(120) - 1.5 = 2.46$
$\hat{y} = \sigma(2.46) = 1/(1 + e^{-2.46}) \approx 0.92$

Prediction: 92% probability of spam → classify as spam

Regularization

L2 Regularization (Ridge)

`L_reg = L + λ||w||²`

- Penalizes large weights
- Prevents overfitting
- λ: regularization strength

L1 Regularization (Lasso)

`L_reg = L + λ||w||₁`

- Encourages sparsity (some $w_i = 0$)
- Feature selection
- λ: regularization strength

Elastic Net

`L_reg = L + λ₁||w||₁ + λ₂||w||²`

- Combines L1 and L2
- Balance between sparsity and smoothness

Multiclass Extension: Softmax Regression

Model

`P(y=k|x) = e^(w_k^T x + b_k) / Σⱼ e^(w_j^T x + b_j)`

Loss (Cross-Entropy)

`L = -1/N Σᵢ Σₖ y_ik log(ŷ_ik)`

- y_ik: one-hot encoded (1 if example i is class k, else 0)
- ŷ_ik: predicted probability of class k

Practice Problems

1. **Sigmoid derivative**: Prove that $\sigma'(z) = \sigma(z)(1 - \sigma(z))$.
2. **Decision boundary**: For w = [1, 2], b = -3, find the equation of the decision boundary.
3. **Gradient**: Compute $\partial L/\partial w$ for a single example with x = [1, 2], y = 1, ŷ = 0.7.
4. **Convergence**: Why does cross-entropy loss lead to better convergence than squared error for classification?
5. **Regularization**: How does L2 regularization affect the gradient update?

Key Takeaways

- Logistic regression: Linear model + sigmoid activation
- Predicts probabilities: $P(y=1|x) = \sigma(w^T x + b)$
- Cross-entropy loss: Penalizes wrong predictions
- Gradient descent: Iteratively updates w and b
- Convex optimization: Single global minimum
- Foundation for neural networks (single-layer network)
- Interpretable: Weights show feature importance

Historical Note

Logistic function introduced by Pierre François Verhulst (1838) - Originally for population growth models - **David Cox** (1958): Applied to regression (logistic regression) - Foundation of generalized linear models (GLMs) - Still widely used despite deep learning advances

Next Steps

Next, we'll decode the Fourier transform—the fundamental tool for signal processing and frequency analysis.

Lesson 6.6: Fourier Transform

Overview

Complete decoding of the Fourier transform—the fundamental tool that decomposes signals into frequency components.

The Core Equations

Fourier Transform (Continuous)

```
F(ω) = ∫_{-∞}^{∞} f(t)e^{-iωt} dt
```

Inverse Fourier Transform

```
f(t) = 1/(2π) ∫_{-∞}^{∞} F(ω)e^{iωt} dω
```

Discrete Fourier Transform (DFT)

```
F_k = Σ_{n=0}^{N-1} f_n e^{-2πikn/N}
```

Symbol-by-Symbol Breakdown

Time and Frequency Domains

f(t) (time-domain signal) - **Domain**: $t \in \mathbb{R}$ (time) - **Values**: Real or complex - **Meaning**: Signal as function of time - **Examples**: Audio waveform, voltage over time, stock price

F(ω) (frequency-domain representation) - **Domain**: $\omega \in \mathbb{R}$ (angular frequency) - **Values**: Complex numbers - **Meaning**: Amplitude and phase at each frequency - **Interpretation**: "How much of frequency ω is in f(t)?"

ω (angular frequency) - **Units**: rad/s (radians per second) - **Relation to frequency**: $\omega = 2\pi$f (f in Hz) - **Meaning**: Rate of oscillation

The Transform: `F(ω) = ∫_{-∞}^{∞} f(t)e^{-iωt} dt`

∫_{-∞}^{∞} … dt (integral over all time) - **Meaning**: Sum up contributions from all time points - **Accumulation**: Weighted sum of signal

f(t) (signal at time t) - **Multiplied by**: Complex exponential - **Contribution**: Each time point contributes to each frequency

e^{-iωt} (complex exponential) - **Euler's formula**: e^{-iωt} = cos(ωt) - i sin(ωt) - **Meaning**: Basis function (pure frequency ω) - **Real part**: cos(ωt) (cosine wave) - **Imaginary part**: -sin(ωt) (sine wave)

Why complex exponentials? - Encode both amplitude and phase - Orthogonal basis functions - Simplify convolution (multiplication in frequency domain) - Eigenfunctions of LTI systems

Complete Interpretation

F(ω) = ∫ f(t)e^{-iωt} dt

Plain language: "The Fourier transform at frequency ω is the overlap between the signal f(t) and a pure oscillation at frequency ω."

Process: 1. Take signal f(t) 2. Multiply by e^{-iωt} (oscillation at frequency ω) 3. Integrate over all time 4. Result: Complex number F(ω)

F(ω) as complex number:

`F(ω) = |F(ω)|e^{iφ(ω)}`

- **$|F(\omega)|$**: Magnitude (amplitude at frequency ω)
- **$\varphi(\omega)$**: Phase (time shift of that frequency component)

Inverse Transform: `f(t) = 1/(2π) ∫ F(ω)e^{iωt} dω`

Meaning: "Reconstruct signal by summing up all frequency components"

Process: 1. Take frequency component F(ω) 2. Multiply by e^{iωt} (oscillation at frequency ω) 3. Integrate over all frequencies 4. Scale by 1/(2π) 5. Result: Original signal f(t)

Interpretation: Every signal is a superposition of pure frequencies

1/(2π) (normalization factor) - Ensures f(t) = $\mathcal{F}$^{-1}[$\mathcal{F}$[f(t)]] - Convention: Can be split between forward and inverse - Alternative: 1/$\sqrt{}$(2π) on both

Properties of Fourier Transform

Linearity

`𝓕[af(t) + bg(t)] = aF(ω) + bG(ω)`

- Transform of sum = sum of transforms
- Scaling preserved

Time Shift

$\mathcal{F}$`[f(t - t`$_0$`)] = e^{-iωt`$_0$`}F(ω)`

- Shifting in time → phase shift in frequency
- Magnitude unchanged: |F(ω)| same

Frequency Shift

$\mathcal{F}$`[e^{iω`$_0$`t}f(t)] = F(ω - ω`$_0$`)`

- Modulation in time → shift in frequency
- Basis of AM/FM radio

Scaling

`𝓕[f(at)] = 1/|a| F(ω/a)`

- Compress time → stretch frequency
- Stretch time → compress frequency
- Uncertainty principle manifestation

Convolution Theorem

`𝓕[f * g] = F(ω)·G(ω)`

- Convolution in time → multiplication in frequency
- Crucial for signal processing
- Filtering = multiplication in frequency domain

Parseval's Theorem (Energy Conservation)

$\int |f(t)|^2 \, dt = 1/(2\pi) \int |F(\omega)|^2 \, d\omega$

- Total energy same in time and frequency domains
- Energy preserved under transform

Common Transform Pairs

Delta Function

$f(t) = \delta(t) \quad \longleftrightarrow \quad F(\omega) = 1$

- Impulse at t=0 → all frequencies equally
- Flat spectrum

Constant

$f(t) = 1 \quad \longleftrightarrow \quad F(\omega) = 2\pi\delta(\omega)$

- DC signal → spike at ω=0
- No oscillation → zero frequency

Cosine

$f(t) = \cos(\omega_0 t) \quad \longleftrightarrow \quad F(\omega) = \pi[\delta(\omega-\omega_0) + \delta(\omega+\omega_0)]$

- Pure frequency → two spikes ($\pm\omega_0$)
- Positive and negative frequencies

Gaussian

$f(t) = e^{-t^2/2\sigma^2} \quad \longleftrightarrow \quad F(\omega) = \sigma\sqrt{(2\pi)}e^{-\sigma^2\omega^2/2}$

- Gaussian → Gaussian
- Width inversely related: narrow in time → wide in frequency

Rectangular Pulse

$f(t) = \mathrm{rect}(t/T) \quad \longleftrightarrow \quad F(\omega) = T\cdot\mathrm{sinc}(\omega T/2)$

- Box function → sinc function
- Sharp edges → high frequencies

Discrete Fourier Transform (DFT)

Definition: F_k = Σ_{n=0}^{N-1} f_n e^{-2πikn/N}

f__n (discrete time samples) - **Index**: n = 0, 1, 2, . . . , N-1 - **Meaning**: Signal sampled at N points - **Spacing**: Δt (sampling interval)

F__k (discrete frequency bins) - **Index**: k = 0, 1, 2, . . . , N-1 - **Meaning**: Frequency components - **Frequency**: ω__k = 2πk/(NΔt)

e^{-2πikn/N} (DFT basis) - **Discrete version** of e^{-iωt} - **Periodic**: e^{-2πi(k+N)n/N} = e^{-2πikn/N} - **Orthogonal**: Different k values are orthogonal

Fast Fourier Transform (FFT)

Naive DFT: O(N^2) operations **FFT algorithm**: O(N log N) operations - **Cooley-Tukey algorithm** (1965) - Exploits symmetry and periodicity - Revolutionized signal processing

Applications

Signal Processing

Audio compression (MP3, AAC): 1. Transform audio to frequency domain 2. Remove inaudible frequencies 3. Quantize remaining frequencies 4. Inverse transform

Noise filtering: 1. Transform noisy signal 2. Identify noise frequencies 3. Attenuate or remove 4. Inverse transform

Image Processing

2D Fourier Transform:

```
F(u,v) = ∬ f(x,y)e^{-i(ux+vy)} dx dy
```

- Decomposes image into spatial frequencies
- JPEG compression uses DCT (related to Fourier)

Quantum Mechanics

Position ↔ Momentum:

```
ψ(p) = 1/√(2πℏ) ∫ ψ(x)e^{-ipx/ℏ} dx
```

- Fourier transform relates position and momentum representations
- Uncertainty principle: $\Delta x \Delta p \geq \hbar/2$

Differential Equations

Solving PDEs: - Transform equation to frequency domain - Differential operators → algebraic - Solve algebraically - Inverse transform

Example (heat equation):

```
∂u/∂t = α∇²u  →  ∂Û/∂t = -αω²Û
```

- Laplacian → multiplication by $-\omega^2$

Spectrograms and Time-Frequency Analysis

Short-Time Fourier Transform (STFT)

```
STFT(t,ω) = ∫ f(τ)w(τ-t)e^{-iωτ} dτ
```

- **w(τ-t)**: Window function centered at t
- **Meaning**: Fourier transform of windowed signal
- **Result**: Time-frequency representation
- **Trade-off**: Time resolution vs frequency resolution

Wavelet Transform

```
W(a,b) = ∫ f(t)ψ*((t-b)/a) dt
```

- ψ: Mother wavelet (localized in time and frequency)
- **a**: Scale (frequency)
- **b**: Translation (time)
- **Advantage**: Better time-frequency localization

Uncertainty Principle

Time-Frequency Uncertainty

$\Delta t \cdot \Delta\omega \geq 1/2$

- Cannot have perfect localization in both time and frequency
- Narrow in time → wide in frequency
- Narrow in frequency → wide in time
- Fundamental limit (not just measurement)

Example: - Short pulse (narrow Δt) → broad spectrum (large Δω) - Pure tone (narrow Δω) → must last long time (large Δt)

Practice Problems

1. **Cosine transform**: Compute $\mathcal{F}[\cos(\omega_0 t)]$ using Euler's formula.
2. **Gaussian**: Verify that Gaussian is its own Fourier transform (up to scaling).
3. **Convolution**: If f(t) = e^{-t^2} and g(t) = e^{-t^2}, find $\mathcal{F}$[f * g].
4. **Sampling**: What is the Nyquist frequency for sampling rate 1000 Hz?
5. **DFT**: Compute 4-point DFT of [1, 0, 0, 0] by hand.

Key Takeaways

- Fourier transform decomposes signals into frequency components
- Complex exponentials e^{iωt} are basis functions
- F(ω) is complex: magnitude = amplitude, phase = time shift
- Convolution in time = multiplication in frequency
- FFT makes computation practical (O(N log N))
- Fundamental tool in signal processing, quantum mechanics, PDEs
- Time-frequency uncertainty: cannot localize perfectly in both

Historical Note

Jean-Baptiste Joseph Fourier (1822): "Théorie analytique de la chaleur" - Studied heat equation - Claimed any function can be represented as sum of sines and cosines - Controversial at the time - **Dirichlet** (1829): Proved convergence conditions - **Cooley & Tukey** (1965): FFT algorithm - Foundation of modern signal processing

Course Complete!

Congratulations! You've completed the Foundations of Mathematical Intuition course. You can now: - Understand fundamental operations and their meanings - Read equations across multiple mathematical domains - Navigate different number systems and spaces - Apply intuition to VQE and quantum optimization - Decode complex equations from physics, ML, and quantum computing

You've achieved true mathematical literacy. Keep practicing, and remember: every equation tells a story.