Author:
kiking

Chapter 1 Introduction

Knowledge base approach to A.I.

a computer can reasonate automatically about statements in formal languages using logical inference rules

Machine Learning

A.I. systems acquire our knowledge by extracting patterns from raw data

Representation Learning

use M.L. to discover not only the mapping from the output, but also the representation itself.

→ discover a good set of features by itself ← e.g. represent data by Cartesian coordinates vs Polar coordinates

→ e.g. Autoencoder (encoder = input → representation
decoder = representation → output)

Deep Learning

→ solves representation learning by introducing representations expressed in simpler representations

→ depth enables the computer to learn a multi-step computer program

↳ measured either by the depth of the computational graph or the depth of the probabilistic graph

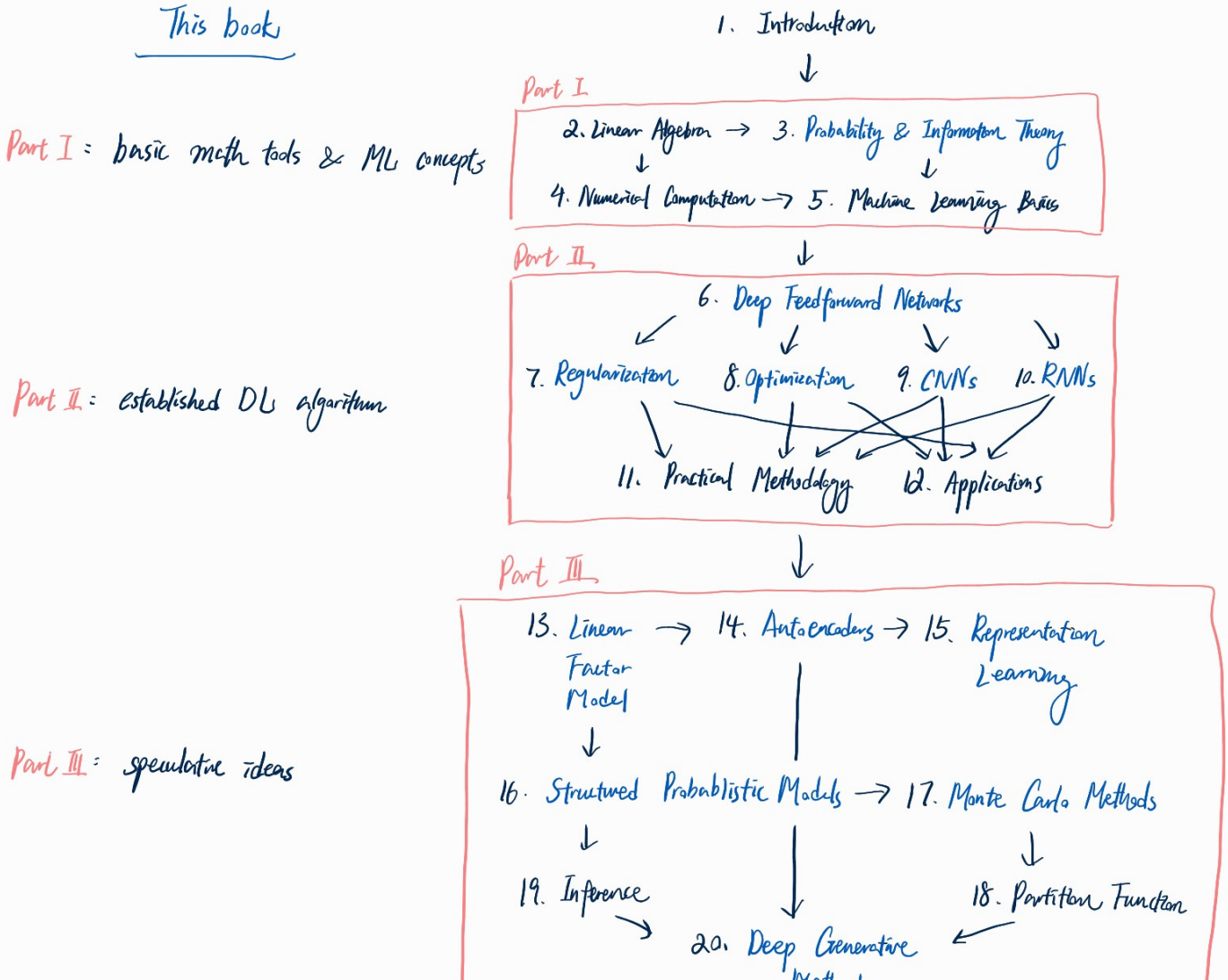
> involve a greater amount of composition of either learned functions or learned concepts than traditional M.L.

This book

Part I : basic math tools & ML concepts

Part II : established DL algorithm

Part III : speculative ideas



Author:
kiking

1.1 Who should read this book?

1.2 Historical Trends in Deep Learning

1.2.1 The many names and changing fortunes of Neural Networks

Cybernetics

Connectionisms / Parallel distributed processing

↳ distributed representation

1.2.2 Increasing Dataset Sizes

1.2.3 Increasing Model Sizes

1.2.4 Increasing Accuracy, Complexity, and Real-World Impact

Chapter 2 Linear Algebra

2.1 Scalars, Vectors, Matrices and Tensors

Tensors: arrays with more than 2 axes

Broadcasting: copying a vector to each row of a matrix

e.g. $C = A + \vec{b} \Rightarrow C_{ij} = A_{ij} + b_j$

2.2 Multiplying Matrices and Vectors

Matrix Product AB vs Element-wise / Hadamard product $A \odot B$

$C_{ij} = \sum_k A_{ik} B_{kj}$ ⟩ dot product of row i of A and column j of B

Dot product between two vectors $\vec{x}, \vec{y}$: $\vec{x}^T \vec{y}$

2.3 Identity and Inverse Matrices

$A\vec{x} = \vec{b} \rightarrow \vec{x} = A^{-1}\vec{b}$ inverse matrix can solve the equation many times with different $\vec{b}$

2.4 Linear Dependence and Span

A^{-1} exists only when exactly one solution $\vec{x}$ exists for each $\vec{b}$

(might have no solution

or infinitely many solutions)

→ Suppose $\vec{x}, \vec{y}$ solution, then $\alpha\vec{x} + (1-\alpha)\vec{y}$ is a solution

$A\vec{x} = \vec{b}$ → Columns of A specify different directions to travel from the origin

number of solutions $\vec{x}$ → determine how many ways to reach $\vec{b}$

each x_i → how far to move in the direction of column i

$A\vec{x} = \sum_i x_i A_{:,i}$ ← linear combination $\sum_i c_i \vec{v}^{(i)}$

Span of $\vec{v}^{(i)}$:

Set of all points obtainable by

decide whether solution $\vec{x}$ exists → test whether $\vec{b}$ is in the span of Columns of A

Author: kiking

Column space / range of A ← this particular span

for $A\vec{x}=\vec{b}$ to have [a solution for any $\vec{b}\in\mathbb{R}^m$] requires [column space of $A=\mathbb{R}^m$]

→ A contains at least one set of [m linearly independent columns] (more in m directions; no redundant)

necessary and sufficient conditions

[for A^{-1} exists], must have [at most one solution]

→ A contains [at most m columns]

→ A is square and non-singular

If linear dependent columns → A is singular

If A is not square or is singular,

can still solve $A\vec{x}=\vec{b}$, but cannot use matrix inversion since A^{-1} does not exist

2.5 Norms

Measure size of a vector

L^p norm: $\|x\|_p = \left(\sum_i |x_i|^p\right)^{\frac{1}{p}}$

Common:

squared L^2 norm: $\vec{x}^T\vec{x}$

↳ easy to work with computationally: e.g. each derivative depends elementwisely instead of the entire vector

↳ problem: increase very slowly near the origin: cannot discriminate between exactly zero vs small but non-zero elements

L^1 norm: $\|\vec{x}\|_1 = \sum_i |x_i|$ ← when this difference between zero vs non-zero elements is important

"L^0 norm": # of non-zero entries

incorrect terminology because this is not a norm,

e.g. scaling by α does not change the number

L^∞ norm: max norm $\|\vec{x}\|_\infty = \max_i |x_i|$

Frobenius norm: $\|A\|_F = \sqrt{\sum_{i,j} A_{i,j}^2}$ measures size of a matrix analogous to the L^2 norm of vector

Dot product $\vec{x}^T\vec{y} = \|\vec{x}\|_2 \|\vec{y}\|_2 \cos\theta$ angle between $\vec{x}$ and $\vec{y}$

2.6 Special Kinds of Matrices and Vectors

Author:
kiking

Diagonal matrix

Symmetric matrix

Unit vector : vector with unit norm $\|x\|_2 = 1$

$\vec{x}$ and $\vec{y}$ are orthogonal to each other if $\vec{x}^T\vec{y} = 0$

↳ if $\vec{x}$ and $\vec{y}$ are unit vectors, they are orthonormal

↳ in $\mathbb{R}^n$ at most n vectors are mutually orthogonal with non-zero norms

Orthogonal matrix $A^TA = AA^T = I \Rightarrow A^{-1} = A^T$ easy to compute inverse matrix

square matrix, rows are mutually orthonormal + columns are mutually orthonormal

2.7 Eigendecomposition

Similar to integers decomposing to prime factors (invariant to the representation "12" in base 10) ~ Decomposing matrices by eigendecomposition — help us understand certain properties of integers / matrices

$A\vec{v} = \lambda\vec{v}$

- Square matrix
- non-zero (right) Eigenvector
- Eigenvalue
- multiplication by A only alters the scale of $\vec{v}$

$\vec{v}^TA = \lambda\vec{v}^T$ left eigenvector

Eigendecomposition

$A = V\,\mathrm{diag}(\lambda)\,V^{-1}$

Suppose A has n lin. indep. eigenvectors, Concatenate to form a matrix (one eigenvector per column) $V = [\vec{v}^{(1)}, \dots, \vec{v}^{(n)}]$

Concatenate the eigenvalues to form $\lambda = [\lambda_1, \dots, \lambda_n]$ and get $\mathrm{diag}[\lambda]$

Every real symmetric matrix can be decomposed into real-valued eigenvectors eigenvalues

$A = Q\Lambda Q^T$

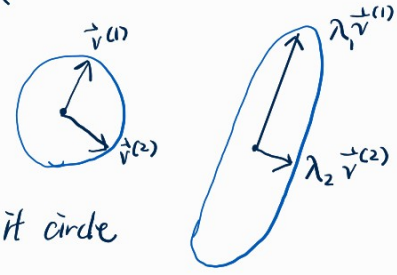
orthogonal

eigenvalue $\Lambda_{i,i}$ associated with eigenvector $Q_{:,i}$

decomposition is unique only if all eigenvalues are unique

Suppose two eigenvectors share the same eigenvalue, then any linear composition gives a new eigenvector with this eigenvalue

e.g. $A \in \mathbb{R}^{2\times 2}$



unit circle

$A\vec{v}$: distort the unit circle → scale the space in direction $\vec{v}^{(i)}$ by λ_i

Author:
kiking

e.g. Optimize quadratic expressions

$$f(\vec{x}) = \vec{x}^T A \vec{x}$$
subject to $\|\vec{x}\|_2 = 1$

Whenever $\vec{x}$ is a eigenvector of A, f takes on the corresponding eigenvalue

max(f) within the constraint region = max eigenvalue (similarly for min(f) = min eigenvalue)

Positive definite → eigenvalues all positive

Positive semidefinite → eigenvalues all positive or zero-valued

guarantee $\forall \vec{x}$, $\vec{x}^T A \vec{x} \geq 0$

2.8 Singular Value Decomposition (SVD)

Singular Value Decomposition another matrix factorization into Singular vectors and Singular values

$$A = UDV^T$$

A: m×n, U: m×m, D: m×n, V^T: n×n

U: orthogonal, left-singular vectors → eigenvectors of AA^T

D: diagonal (may not be square) → square roots of eigenvalues of AA^T or A^TA

V^T: orthogonal, right-singular vectors → eigenvectors of A^TA

Interpret SVD as eigendecomposition of functions of A

- discovers some of the same kind of info from eigendecomposition
- every real matrix has a SVD (more general)
  e.g. not square matrix → no eigendecomposition
- Partially generalize matrix inversion to non-square matrix

2.9 The Moore-Penrose Pseudoinverse

Suppose want to find B s.t. left-inverse of A

$$A\vec{x} = \vec{y} \Rightarrow \vec{x} = B\vec{y}$$

A: m×n

If $m > n$ → may have no solution (m × n)

If $m < n$ → may have multiple solutions (m × n)

Moore-Penrose Pseudoinverse

$$A^+ = \lim_{\alpha \to 0} (A^TA + \alpha I)^{-1} A^T$$

practical computation $= VD^+U^T$ where U, V from singular value decomposition

D^+ = take reciprocal of D's non zero elements, then take transpose

When $m < n$, $\vec{x} = A^+\vec{y}$ provides one of many solutions with minimal Euclidean norm $\|\vec{x}\|_2$

When $m > n$, $\vec{x} = A^+\vec{y}$ is the solution for $\arg\min_{\vec{x}} \|A\vec{x} - y\|_2$
closest possible solution to $\vec{y}$

2.10 The Trace Operator

Author:
kiking

$Tr(A) = \sum_i A_{i,i}$ the sum of all diagonal entries of A

e.g. Frobenius norm: $\|A\|_F = \sqrt{\sum_{i,j} A_{i,j}^2} = \sqrt{Tr(AA^T)}$

Invariant to cyclic permutation

e.g. $Tr(ABC) = Tr(BCA) = Tr(CAB)$

2.11 Determinant

$\det(A)$ = product of eigenvalues of matrix

$|\det(A)|$ = measures how much multiplication by the matrix expands or contracts space (absolute value)

↳ $|\det A| = 0$: space contracted completely along at least one dim (lose all volume)

↳ $|\det A| = 1$: the transformation preserves volume

2.12 Example: Principal Components Analysis (PCA)

A simple machine learning algorithm

Suppose: m points $\{\vec{x}^{(1)}, \dots, \vec{x}^{(m)}\}$ in $\mathbb{R}^n$

Want: apply lossy compression (store points with less memory but lose some precision)
↘ lose as little precision as possible

Want: find some encoding function to produce code vector $\vec{c}^{(i)} \in \mathbb{R}^l$ where $l < n$
$f(\vec{x}) = \vec{c}$
and a decoding function that produces the reconstructed input given its code
$g(\vec{c}) = g(f(\vec{x})) \approx \vec{x}$

Simple PCA: matrix multiplication for decoding:

let $g(\vec{c}) = D\vec{c}$ where $D \in \mathbb{R}^{n \times l}$

PCA: constrains the columns to be orthogonal to each other ‖ for unique solution columns of D have unit norm
(may not be orthogonal matrix since may not be square)

ⓐ Step ①: generate optimal code point $\vec{c}^*$ for each $\vec{x}$

In PCA: L^2 norm → $c^* = \underset{\vec{c}}{\operatorname{argmin}} \|\vec{x} - g(\vec{c})\|_2$
or squared L^2 norm (same) → $\underset{\vec{c}}{\operatorname{argmin}} \|\vec{x} - g(\vec{c})\|_2^2 = \underset{\vec{c}}{\operatorname{argmin}} \|\vec{x} - D\vec{c}\|_2^2$
⋮

ⓑ solving: $\vec{c}^* = D^T\vec{x}$ ∴ $f(\vec{x}) = D^T\vec{x}$

Step ②: PCA reconstruction for the decoding step

$g(\vec{c}) = DD^T\vec{x}$

Step ③: Choose encoding matrix D

$D^* = \underset{D}{\operatorname{argmin}} \sqrt{\sum_{i,j} (x_j^{(i)} - g(f(\vec{x}^{(i)}))_j)^2}$ Frobenius norm to compute matrix error (similar idea)

subject to $D^TD = I_l$

Author:
kiking

$\vec{d}^* = \arg\min_{\vec{d}} \sum_i \| \vec{x}^{(i)} - \vec{d}\vec{d}^T\vec{x}^{(i)} \|_2^2$ subject to $\|\vec{d}\|_2 = 1$

$\vec{d}^* = \arg\min_{\vec{d}} \|X - X\vec{d}\vec{d}^T\|_F^2$ subject to $\vec{d}^T\vec{d} = 1$ where $X_{i,:} = \vec{x}^{(i)T}$

$\vdots$

$\vec{d}^* = \arg\max_{\vec{d}} Tr(\vec{d}^T X^T X \vec{d})$ subject to $\vec{d}^T\vec{d} = 1$

Solution: $\vec{d}^*$ = eigenvector of X^TX with the largest eigenvalue

recovers only the first principal component

Chapter 3 Probability and Information Theory

Probability theory: means of quantifying uncertainty + axioms for deriving new uncertainty statements

A.I. application: ① laws of probability → how AI systems should respond
② theoretically analyze the behavior of proposed AI systems

Information theory: quantify the amount of uncertainty in a probability distribution

3.1 Why Probability?

3 possible sources of uncertainty: ① Inherent stochasticity of the system
② Incomplete observability
e.g. Monty Hall problem (3 doors win a prize)
③ Incomplete modelling

Frequentist probability (treat as if same) vs Bayesian probability

Frequentist probability: if an experiment happen many times, with probability p an event happen

Bayesian probability: degree of belief: 1 absolute yes, 0 absolute no; qualitative levels of uncertainty

3.2 Random Variables

random variable → variable that can take on different values randomly

e.g. x random variable, x_1 x_2 the possible values

→ just a description of the states that are possible

→ must be coupled with a probability distribution that specifies how likely each of these states are

→ discrete / continuous

3.3 Probability Distributions

probability distribution → describe how likely a random variable / set of random variables is to take on each of its possible states

3.3.1 Discrete variables and Probability Mass Functions

over discrete variables → described with PMF (probability mass functions)

→ $P(x) \neq P(y)$ usually (different PMF)

Author:
kiking

to distinguish which PMF, write $P(x = x)$ or $x \sim P(x)$

over many variables → joint probability distribution e.g. $P(x = x, y = y)$ or $P(x, y)$

PMF : $\forall x \in X,\ 0 \le P(x) \le 1,\ \sum_{x \in X} P(x) = 1$ (normalized)

3.3.2 Continuous variables and Probability Dense Functions

over Continuous variables → described with PDF (probability dense functions)

→ the probability of landing inside an infinitesimal region with volume δx is given by $p(x)\delta x$

PDF : $\forall x \in X,\ p(x) \ge 0$, not necessary $p(x) \le 1$, $\int p(x)\,dx = 1$

the prob of x lies in some set S is given by the integral of $p(x)$ over that set

e.g. for x lies in $[a,b]$, the prob is $\int_{[a,b]} p(x)\,dx$

e.g. Uniform Distribution on an interval of real numbers

$u(x; a, b)$ for $[a,b]$ with $b > a$

↳ $u(x; a, b) = 0$ for all $x \notin [a,b]$

↳ $u(x; a, b) = \frac{1}{b-a}$ for all $x \in [a,b]$

} non-negative, integrates to 1

$x \sim U(a,b)$

3.4 Marginal Probability

To know the probability distribution over just a subset of variables

e.g. $\forall x \in X,\ P(x = x) = \sum_y P(x = x, y = y)$

or $p(x) = \int p(x, y)\,dy$

(imagine writing a grid to enumerate all possibles $P(x, y)$, then write in the margin for the sum of the row $P(x)$)

3.5 Conditional Probability

$$P(y = y \mid x = x) = \frac{P(y = y, x = x)}{P(x = x)}$$

y conditions on x happens

← only defined when $\neq 0$ because cannot compute if event does not happen

do not confuse with Casual Modelling / Intervention Query

consequences of making an action

3.6 The Chain Rule of Conditional Probabilities

Chain Rule / Product Rule of Probability

$$P(x^{(1)}, \ldots, x^{(n)}) = P(x^{(1)}) \cdot \prod_{i=2}^{n} P(x^{(i)} \mid x^{(1)}, \ldots, x^{(i-1)})$$

3.7 Independence & Conditional Independence

Two random variables x and y are independent if their probability distribution:

Author:
kiking

$\forall x \in \mathbb{X}, y \in \mathbb{Y}$ $p(\mathbf{x}=x, \mathbf{y}=y) = p(\mathbf{x}=x)\, p(\mathbf{y}=y)$ $\mathbf{x} \perp \mathbf{y}$

— if — are conditionally independent

if $\forall x \in \mathbb{X}, y \in \mathbb{Y}, z \in \mathbb{Z}$ $p(\mathbf{x}=x, \mathbf{y}=y \mid \mathbf{z}=z) = p(\mathbf{x}=x \mid \mathbf{z}=z)\, p(\mathbf{y}=y \mid \mathbf{z}=z)$ $\mathbf{x} \perp \mathbf{y} \mid \mathbf{z}$

3.8 Expectation, Variance & Covariance

Expected value of some function $f(x)$ w.r.t. $P(x)$

$$\mathbb{E}_{x \sim P}[f(x)] = \begin{cases} \sum_x P(x) f(x) & \text{discrete} \\ \int p(x) f(x)\, dx & \text{continuous} \end{cases}$$

$= \mathbb{E}_x[f(x)] = \mathbb{E}[f(x)]$

Expectation is linear e.g. $\mathbb{E}_x[\alpha f(x) + \beta g(x)] = \dots$

$\mathbb{E}[\cdot]$: averages over the values of all the random variables inside the brackets

Variance

measures how much the function values of a random variable x vary as diff values x are sampled from its probability distribution

$$\mathrm{Var}(f(x)) = \mathbb{E}\left[(f(x) - \mathbb{E}[f(x)])^2\right]$$

expected difference between $f(x)$ and expected $f(x)$

$\sqrt{\mathrm{Var}(f(x))}$: Standard Deviation

Covariance

measures how much two values are linearly related & the scale

$$\mathrm{Cov}(f(x), g(y)) = \mathbb{E}\left[(f(x) - \mathbb{E}[f(x)])(g(y) - \mathbb{E}[g(y)])\right]$$

expected (difference between $f(x)$ and expected $f(x)$) × (difference between $g(y)$ and expected $g(y)$)

→ Correlation normalizes the contribution of each variable

→ Zero Covariance → no linear dependence for the two variables

Independence → above & no non-linear relationships - stronger requirement

e.g.

continuous $x \in [-1, 1]$, $f(x) = U(x; [-1, 1]) = \frac{1}{2}$

discrete $S = \{-1, 1\}$, $g(s=-1) = g(s=1) = \frac{1}{2}$

let $y = sx$ be a random variable

$\mathrm{Cov}(x, y) = 0$ but x completely determines y

Covariance Matrix of a random vector $x \in \mathbb{R}^n$

$$\mathrm{Cov}(\vec{x})_{ij} = \mathrm{Cov}(\vec{x}_i, \vec{x}_j)$$ diag $\mathrm{cov}(\vec{x}_i, \vec{x}_i) = \mathrm{Var}(\vec{x}_i)$

3.9 Common Probability Distributions

3.9.1 Bernoulli Distribution

→ over a single binary random variable

Author:
kiking

→ controlled by $\phi \in [0,1]$

Choose ϕ to be
Logistic sigmoid
$\sigma(x) = \frac{1}{1+\exp(-x)}$

$P(x=1) = \phi$
$P(x=0) = 1-\phi$
$P(x=x) = \phi^x (1-\phi)^{1-x}$

$\mathbb{E}[x] = \phi$
$Var(x) = \phi(1-\phi)$

3.9.2 Multinoulli Distribution (Categorical Distribution)

→ over a single discrete variable with k different states (finite)

↳ special case of multinomial distribution
($n=1$ for how many times each of the k categories is visited when n samples are drawn from a multinoulli distribution)

Predict probabilities with
Softmax function
ith: $\frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)}$

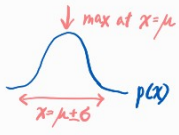
→ parametrized by $\vec{p} \in [0,1]^{k-1}$ — p_i gives the prob of the ith state

the prob of final state $= 1 - \vec{1}^T \vec{p}$ — must constraint $\vec{1}^T \vec{p} \leq 1$

3.9.3 Gaussian Distribution (Normal Distribution)

→ commonly use over real numbers

$$\mathcal{N}(x; \mu, \sigma^2) = \sqrt{\frac{1}{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$



where $\mu \in \mathbb{R}$, $\sigma \in (0,\infty)$ $\mathbb{E}(x) = \mu$ $Var(x) = \sigma^2$

Choose μ, β to be
Softplus function
$\zeta(x) = \log(1+\exp(x))$

→ To control precision = $\beta \in (0,\infty)$

$$\mathcal{N}(x; \mu, \tfrac{1}{\beta}) = \sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{1}{2}\beta(x-\mu)^2\right)$$

※ Central Limit Theorem
the sum of many independent variables ~ normally distributed

※ Encodes the max. amount of uncertainty
out of all possible prob. distributions with the same variance

→ Multivariate Normal Distribution (generalize to $\mathbb{R}^n$)

$$\mathcal{N}(\vec{x}; \vec{\mu}, \Sigma) = \sqrt{\frac{1}{(2\pi)^n \det(\Sigma)}} \exp\left(-\frac{1}{2}(\vec{x}-\vec{\mu})^T \Sigma^{-1} (\vec{x}-\vec{\mu})\right)$$

where Σ a positive definite symmetric matrix $\mathbb{E}(\vec{x}) = \vec{\mu}$
$Cov(\vec{x}) = \Sigma$

→ Simpler: fix Σ diagonal control variance along axis-aligned direction

Set $Cov(\vec{x}) = \alpha I$ (isotropic Gaussian distribution) — same variance along each direction

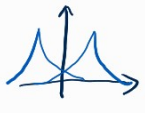
3.9.4 Exponential and Laplace Distributions

Exponential distribution

$$p(x; \lambda) = \lambda \mathbf{1}_{x \geq 0} \exp(-\lambda x)$$

Laplace distribution

$$Laplace(x; \mu, \gamma) = \frac{1}{2\gamma} \exp\left(-\frac{|x-\mu|}{\gamma}\right)$$



3.9.5

Author:
kiking

3.9.5 The Dirac Distribution & Empirical Distribution

→ specify all the mass in a probability distribution clusters around a single point

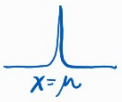
→ a PDF with the Dirac delta function $\delta(x)$:

$$p(x) = \delta(x - \mu)$$

limit point of a series of functions that put less & less density on all points other than zero

→ zero-valued everywhere except $x - \mu = 0$

→ integrate to 1



infinitely narrow & high peak at $x = \mu$

a generalized function

Empirical Distribution

over continuous variables: use Dirac delta distribution

$$\hat{p}(\vec{x}) = \frac{1}{m} \sum_{i=1}^{m} \delta(\vec{x} - \vec{x}^{(i)})$$

put probability mass $\frac{1}{m}$ on each of the m points $\vec{x}^{(1)}, \ldots, \vec{x}^{(m)}$ forming the given collection of samples

over discrete variables:

a multinoulli distribution whose probability with each value equal to the empirical frequency of that in the training set

★ the probability density that maximizes the likelihood of the training data

3.9.6 Mixtures of Distributions

one way of combining probability distributions

Mixture Distribution

Generate the sample x:

$$P(x) = \sum_i P(c = i) P(x \mid c = i)$$

where $P(c)$: the multinoulli distribution over component identities

E.g. the empirical distribution over real-valued variables = mixture distribution with one Dirac component for each sample

Latent variable

a random variable that we cannot observe directly

e.g. component identity variable c

Gaussian mixture model

$p(\vec{x} \mid c = i)$ are Gaussian distributions — each has $\mu^{(i)}$ and $\Sigma^{(i)}$

→ Some mixtures can have more constraints

e.g. shared covariance $\Sigma^{(i)} = \Sigma \ \forall i$

→ the parameters of the Gaussian mixture specify

the prior probability $\alpha_i = P(c = i)$ for each component i — model's beliefs before observed x

the posterior probability $P(c \mid \vec{x})$ after the observation of x

→ a Universal Approximator

Author:
kiking

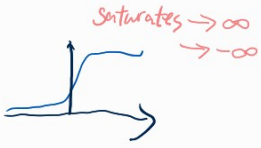
can approximate any smooth density within specified error
with enough components

3.10 Useful Properties of Common Functions

Logistic sigmoid

$$\sigma(x) = \frac{1}{1+\exp(-x)}$$

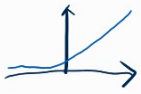
Range is $(0,1)$
✓ produce ϕ in Bernoulli distribution



Softplus function

$$\zeta(x) = \log(1+\exp(x))$$

Range is $(0,\infty)$
✓ produce β or δ of Gaussian distribution
✓ softed version of $x^+ = \max(0,x)$



$$\sigma(x) = \frac{\exp(x)}{1+\exp(x)}$$

$$\frac{d}{dx}\sigma(x) = \sigma(x)\cdot(1-\sigma(x))$$

$$1-\sigma(x) = \sigma(-x)$$

$$\forall x \in (0,1),\ \sigma^{-1}(x) = \log\left(\frac{x}{1-x}\right)$$ logit

$$\log\sigma(x) = -\zeta(-x)$$

$$\frac{d}{dx}\zeta(x) = \sigma(x)$$

$$\zeta(x) = \int_{-\infty}^{x}\sigma(y)\,dy$$

e.g. $x^+ + x^- = x$

$$\zeta(x) - \zeta(-x) = x$$

$$\forall x > 0,\ \zeta^{-1}(x) = \log(\exp(x)-1)$$

3.11 Bayes' Rule

Bayes' Rule $$P(x|y) = \frac{P(x)P(y|x)}{P(y)}$$

3.12 Technical Details of Continuous Variables

Measure Theory: provides a characterization of the set of sets that can compute probability without paradoxes

e.g. by infinite precision of real numbers / fractal-shaped sets / transformation of rational numbers

e.g. The Banach-Tarski theorem 

provides a rigorous way of describing a set of points that is negligibly small i.e. measure zero
occupies no volume e.g. line in $\mathbb{R}^2$

Almost Everywhere: Some important theory holds for discrete values but hold "almost everywhere" only for continuous values

Handle continuous random variables that are deterministic functions of one another

$$\vec{y} = g(\vec{x}) \xrightarrow{\text{assume invertible } g} p_y(\vec{y}) = p_x(g^{-1}(\vec{y}))$$

$$p_x(x) = p_y(g(x))\left|\frac{\partial g(x)}{\partial x}\right| \quad \Big| \quad p_x(\vec{x}) = p_y(g(\vec{x}))\left|\det\left(\frac{\partial g(\vec{x})}{\partial x}\right)\right|$$

det of Jacobian matrix $J_{ij} = \frac{\partial x_i}{\partial y_j}$

3.13 Information Theory

Quantify how much info is present in a signal

originally: send msg from discrete alphabets over noisy channel

Author:
kiking

communication via radio transmission

Information theory: design optimal code
↳ calculate expected length of msg sampled from different probability distributions with diff encoding schemes

in ML: to continuous variables
↳ characterize probability distributions
quantify similarity between probability distributions

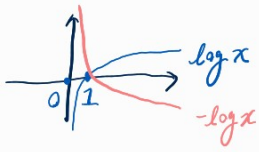
Basic Intuition:

learning that an unlikely event has occurred is more informative than a likely event has occurred

- quantify s.t. an event guarantee to happen = no information content
- less likely event → higher information content
- independent events → additive information content

Self-information

for an event $x = x$: $I(x) = -\log P(x)$



Units of nats: one nat = $I(x)=1 \Leftrightarrow P(x)=\frac{1}{e}$
one bit / shannon $\Leftrightarrow P(x)=\frac{1}{2}$

for continuous variable,
zero information → an event with unit density (instead of must occur)

deals with single outcome only

Shannon entropy

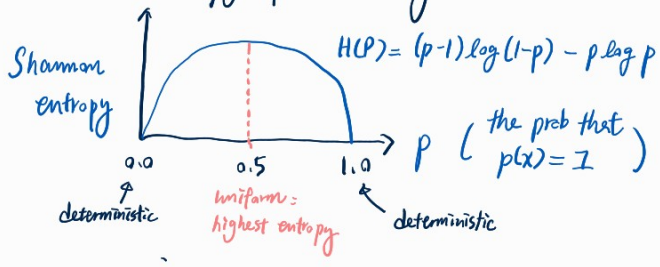
the amount of uncertainty in the entire probability distribution

$$H(x) = \mathbb{E}_{x\sim P}[I(x)]$$
$$= -\mathbb{E}_{x\sim P}[\log P(x)] = H(P)$$

i.e. the expected amount of information in an event drawn from that distribution

⇔ the lower-bound on # bits on average to encode symbols drawn from that distribution

e.g. Shannon entropy of a binary random variable



When x is continuous → Differential Entropy

Kullback-Leiber (KL) divergence

Same random variable x, different probability distributions → measure difference

Author:
kiking

$$D_{KL}(P\|Q) = \mathbb{E}_{x\sim p}\left[\log \frac{P(x)}{Q(x)}\right] = \mathbb{E}_{x\sim p}\left[\log P(x) - \log Q(x)\right]$$

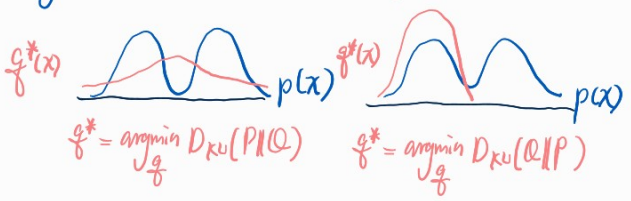
In ML:

the extra amount of information needed (in nats/bits) to send a message containing symbols drawn from P, when we use a code that was designed to minimize the length of messages drawn from Q

$D_{KL}(P\|Q) = 0 \rightarrow$ same distribution (discrete variables) or equal almost everywhere (continuous variables)

non-negative

asymmetric: $D_{KL}(P\|Q) \neq D_{KL}(Q\|P)$ not a distance measure!



$q^* = \arg\min_q D_{KL}(P\|Q)$ $\quad$ $q^* = \arg\min_q D_{KL}(Q\|P)$

select depends on application

left: find q s.t. high $p(x) \rightarrow$ high $q(x)$

right: find q s.t. low $p(x) \rightarrow$ low $q(x)$

Cross-Entropy

$$H(P,Q) = H(P) + D_{KL}(P\|Q) = -\mathbb{E}_{x\sim p}[\log Q(x)]$$

& similar to KL Divergence without the term on the left

minimize cross-entropy w.r.t. Q = minimize the KL Divergence

$0 \log 0$

in information theory, treat as $\lim_{x\to 0} x \log x = 0$

3.14 Structured Probabilistic Models

Motivation: don't use one single function to represent a probability distribution, split it to multiple ones by factorization over fewer variables

e.g. $p(a,b,c) = p(a)\cdot p(b|a)\cdot p(c|b)$ → reduce cost of representing a distribution

Structured Probabilistic Model / Graphical Model

represent the factorization with a graph (in context of graph theory)

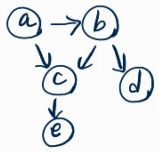
Graph G: node → random variable

edge → the probability distribution is able to represent direct interaction between the two random variables

only a description of prob distribution, not exclusive

- Directed Model (to each other)

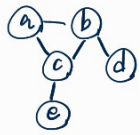
$$p(\vec{x}) = \prod_i p(x_i \mid \text{parent}_G(x_i))$$



$p(a,b,c,d,e) = p(a)\cdot p(b|a)\cdot p(c|a,b)\cdot p(d|b)\cdot p(e|c)$

- Undirected Model

$$p(\vec{x}) = \frac{1}{Z}\prod_i \phi^{(i)}(C^{(i)})$$



$p(a,b,c,d,e) = \frac{1}{Z}\phi^{(1)}(a,b,c)\cdot\phi^{(2)}(b,d)\cdot\phi^{(3)}(c,e)$

Clique $C^{(i)}$: set of nodes connecting to each other

associated with a factor $\phi^{(i)}(C^{(i)})$ factor: just functions ≠ probability distribution

$\frac{1}{Z}$: normalizing → non-negative

Author:
kiking

Chapter 4 Numerical Computation

Algorithms that solve math problems by methods that update estimates of the solution via an iterative process

4.1 Overflow and Underflow

Real numbers → approximation error → rounding error

Underflow : e.g. division by zero → not-a-number error

Overflow : ∞ or $-\infty$ → not-a-number error

Softmax function

predict probabilities associated with a multinoulli distribution

$$\text{softmax}(\vec{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)}$$

prone to underflow / overflow

e.g. suppose all $x_i = c \Rightarrow \text{softmax} = \frac{\exp(c)}{\sum_i \exp(c)} = \frac{1}{n}$ analytically

but when c is very negative / positive

→ $\exp(c)$ underflow → $\text{softmax} = \frac{0}{0}$

/ overflow → $\text{softmax} = \frac{\infty}{\infty}$ > undefined

e.g. suppose get $\text{softmax} = 0 \Rightarrow \log(\text{softmax}) = \log(0) \rightarrow$ undefined

Solution :

evaluate $\text{softmax}(\vec{z})$ where $\boxed{\vec{z} = \vec{x} - \max_i x_i}$; implement $\log(\text{softmax})$ in a numerically stable way

4.2 Poor Conditioning

Conditioning : how rapidly a function changes w.r.t. small changes in its inputs perturbed slightly

Condition number of $A \in \mathbb{R}^{n \times n}$ for $f(\vec{x}) = A^{-1}(\vec{x})$

$\max_{i,j} \left| \frac{\lambda_i}{\lambda_j} \right|$ ratio of max eigenvalue to min eigenvalue

* intrinsic property

large → sensitive to errors in inputs

4.3 Gradient-based Optimization

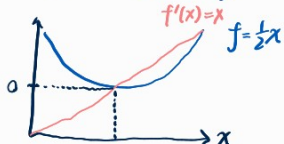
Objective function : the function we wish to maximize / minimize

Cost / Loss / Error function : when minimizing

* Optimization in continuous space

Gradient Descent : minimize $f(x)$ by moving x in small steps with the opposite sign of the derivative

(vs hill climbing)



$$\vec{x}' = \vec{x} - \varepsilon \nabla_{\vec{x}} f(\vec{x})$$

steepest descent

Author:
kiking

(in Discrete space) <0: x↓↓ >0: x↑↑

Critical point = local minimum/maximum, saddle point

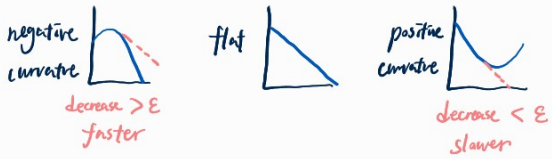
Line search = evaluate several ε (learning rate)

4.3.1 Beyond the Gradient: Jacobian & Hessian Matrices

Jacobian matrix: $J_{i,j} = \frac{\partial}{\partial x_j} f(\vec{x})_i$ first derivative

Hessian matrix: $H(f)(\vec{x})_{i,j} = \frac{\partial^2}{\partial x_i \partial x_j} f(\vec{x})$ second derivative

↳ measures curvature



↳ symmetric (and real) → eigendecomposition

Taylor Series approximation

Second order, with $\vec{x} = \vec{x}^{(0)} - \varepsilon \cdot \vec{g}$ g: gradient

$$f(\vec{x}) \approx \underbrace{f(\vec{x}^{(0)})}_{\text{original value}} \underbrace{- \varepsilon \vec{g}^T \vec{g}}_{\text{expected improvement from slope of the function}} + \underbrace{\frac{1}{2}\varepsilon^2 \vec{g}^T H g}_{\text{correction we must apply to account for the curvature of the function}}$$

Optimizing ε gives

$$\varepsilon^* = \frac{\vec{g}^T \vec{g}}{\vec{g}^T H \vec{g}}$$ ↙ only when this is positive

worst case when $\vec{g}$ = eigenvector of H with the max eigenvalue,

$$\varepsilon^* = \frac{1}{\lambda^{max}}$$

* Scale of the learning rate → determined by eigenvalues of Hessian when function can be approximated well by quadratic function (second-order)

Second Derivative test

$f'(x) = 0$ and

$f''(x) > 0$ → local minimum
$f''(x) < 0$ → local maximum
$f''(x) = 0$ → inconclusive

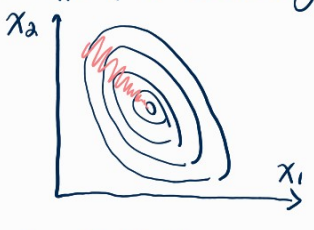
Multiple dimensions: examine eigenvalues of Hessian
- positive definite (all eigenvalues > 0)
- negative definite (all eigenvalues < 0)
- saddle point (+ − eigenvalues)
- inconclusive (same-sign eigenvalues with same zeros)

Poor-conditioned Hessian

Hessian with a poor condition number
→ one derivative changes more rapidly than another
→ difficult to choose a good step size

(the ratio of λ_{max} to λ_{min}; direction ↓ direction ↓)

too large → overshooting at one direction
too small → converges slowly



e.g. condition number 5
→ direction of most curvature is 5-times more
→ waste time descending in the steepest feature

Newton's Method

Author:
kiking

Newton's Method

resolve above by using info from the Hessian to guide the search:

$$\vec{x}^* = \vec{x}^{(0)} - H(f)(\vec{x}^{(0)})^{-1} \nabla_{\vec{x}} f(\vec{x}^{(0)})$$

f: positive definite quadratic function

directly jumps to the minimum of the function

converge quicker than gradient descent ✡ but not good for saddle points

First-order optimization (e.g. Gradient descent)

Second-order optimization (e.g. Newton's Method, also uses Hessian)

Obtain some guarantees by restricting to

Lipschitz continuous functions — a function with rate of change bounded by a Lipschitz constant L

$\forall \vec{x}, \forall \vec{y}$

$|f(\vec{x}) - f(\vec{y})| \leq L \|\vec{x} - \vec{y}\|_2$

useful to quantify assumption that ✡ only a weak constraint

a small change in the input → a small change in the output

✡ applicable for many DL algo

Convex Optimization

Stronger guarantees by stronger restrictions

Convex functions → Hessian is positive semidefinite everywhere

Well-behaved

→ lack saddle points

→ local minima = global minima

✡ difficult to apply to most DL algo

4.4 Constrained Optimization

Constrainted Optimization

Only maximize/minimize $f(x)$ for values of $\vec{x}$ in some set S

Feasible points: $\vec{x}$ that lie within the set S

Often: find small solution

Norm constraints: e.g. $\|\vec{x}\| \leq 1$

Simple approach

modify gradient descent: make step, project back into S

line search: search only step size ε that yields feasible points / project back into S

Sophisticated approach

↓ design a unconstrainted optimization problem, convert the solution into that of the original constrained problem

Karush-Kuhn-Tucker (KKT)

introduce the generalized Lagrangian (function)

① Describe S ...

Author:
kiking

$$S = \left\{ \vec{x} \;\middle|\; \begin{array}{l} \forall i,\ g^{(i)}(\vec{x}) = 0 \\ \forall j,\ h^{(j)}(\vec{x}) \le 0 \end{array} \right\} \begin{array}{l} \leftarrow \text{Equality constraints} \\ \leftarrow \text{Inequality constraints} \end{array}$$

also an "active constraint"

② Introduce KKT multipliers

new variables λ_i and α_j for each constraint

(Lagrange multipliers: does not allow inequality constraints)

③ Define the generalized Lagrangian

$$L(\vec{x}, \vec{\lambda}, \vec{\alpha}) = f(\vec{x}) + \sum_i \lambda_i g^{(i)}(\vec{x}) + \sum_j \alpha_j h^{(j)}(\vec{x})$$

④ Unconstrained optimization

$$\min_{\vec{x}} \max_{\vec{\lambda}} \max_{\vec{\alpha} \ge 0} L(\vec{x}, \vec{\lambda}, \vec{\alpha}) \quad \leftrightarrow \quad \min_{\vec{x} \in S} f(\vec{x})$$

when satisfied: equal $f(\vec{x})$
if violate: equal ∞

\# Solution at convergence

remains a stationary point whether or not the inactive constraints are included

i.e. the inequality has no influence on the solution

and represented by zeroing out the KKT multipliers

(but inactive constraint can exclude some solutions)

K.K.T conditions

necessary (but not sufficient) for optimal points

→ ∇L is zero

→ All constraints on $\vec{x}$ and KKT multipliers are satisfied

→ The inequality constraints exhibit Complementary Slackness

$$\vec{\alpha} \odot h(\vec{x}) = \vec{0}$$

4.5 Example: Linear Least Square

$$\min f(\vec{x}) = \frac{1}{2} \| A\vec{x} - b \|_2^2$$

Gradient-based optimization:

① Obtain gradient:

$$\nabla_{\vec{x}} f(\vec{x}) = A^T A \vec{x} - A^T \vec{b}$$

② Gradient Descent:

While $\| \nabla_{\vec{x}} f(\vec{x}) \| > \xi$ do

$\vec{x} \leftarrow \vec{x} - \varepsilon \cdot \| \nabla_{\vec{x}} f(\vec{x}) \|$

end while

or Newton's method:

since the function is quadratic, the approximation by Newton's method is exact → one step reaches min.

Suppose now subject to $\vec{x}^T \vec{x} \le 1$:

① Introduce the Lagrangian

$$L(\vec{x}, \lambda) = f(\vec{x}) + \lambda(\vec{x}^T \vec{x} - 1)$$

② Solve

$$\min_{\vec{x}} \max_{\lambda \ge 0} L(\vec{x}, \lambda)$$

Author:
kiking

(3a) Solve by Moore-Penrose Pseudoinverse $\vec{x} = A^+ \vec{b}$
→ smallest-norm solution, if the point is feasible; otherwise

(3b) Solve by
differentiating the Lagrange w.r.t. $\vec{x}$ → $\vec{x} = (A^TA + 2\lambda I)^{-1} A^T \vec{b}$
gradient ascent on λ until $\vec{x}$ has the correct norm & derivative on $\lambda = 0$
$\frac{\partial}{\partial \lambda} L(\vec{x}, \lambda) = \vec{x}^T\vec{x} - 1$

Chapter 5 Machine Learning Basics

5.1 Learning Algorithms

Mitchell (1997): learn from experience E
w.r.t. some class of tasks T
and performance measure P

5.1.1 The Task T

Classification with missing inputs
eg learn a joint distribution for the 2^n classification functions with n (possible missing) inputs
Structured output
eg sentence → grammatical tree
Density / PMF estimation
learn the PDF/PMF on the space where the examples are drawn

5.1.2 The Performance Measure P

5.1.3 The Experence E

Supervised vs Unsupervised learning → blurred definition
e.g. learning a joint probability distribution (unsupervised) by factorization (multiple supervised)
Re-inforcement learning
interact with the environment with feedback loop
(not only a fixed dataset)

5.1.4 Example: Linear Regression

$\hat{y} = \vec{w}^T\vec{x} + b$ b: bias (the "bias" towards b when input is absent)
affine function: prediction plot is a line not necessary passing through origin

$MSE_{test} = \frac{1}{m} \sum_i (\hat{y}^{(test)} - y^{(test)})^2$ mean squared error

5.2 Capacity, Overfitting and Underfitting

Data-generating distribution: assumption that the underlying (shared) train, test distributions
i.i.d. assumptions → are identically distributed and samples are independent

Underfitting: Struggle to reduce training error
Overfitting: Struggle to reduce test error

Author:
kiking

Alter Capacity by choosing the hypothesis space

e.g. increase cap by generalize linear regression to include polynomial functions (on top of linear functions) as its input

$\hat{y} = b + wx \longrightarrow \hat{y} = b + w_1 x + \underbrace{w_2 x^2}_{\text{quadratic function of inputs}}$

⇒ change # input features and add new parameters

still linear function of the parameters

Representational Capacity

the family of functions the model can choose from when varying the parameters

Effective Capacity

less than representational capacity often due to imperfection of the algo

Occam's razor

improve generalization of a model by choosing the "simplest" one

Vapnik-Chervonenkis dimension

VC dimension measures the capacity of a binary classifier

✡ deciding the bounds of difference between training vs test error

rarely applied to DL algo

Nonparametric models

can have arbitrary high capacity

can design by making their complexity a function of the training set size

e.g. Nearest Neighbor Regression

Bayes error

the error incurred by an oracle making predictions from the true distribution

happen when some noise in the distribution

5.2.1 The No Free Lunch Theorem

(Wolpert, 1996) Averaged over all possible data-generating distributions, no ML algorithm is universally better than any other.

→ Goal of ML research

understand what kind of distributions are relevant to the real world, and what kind of ML algorithms perform well on them

5.2.2 Regularization

another way to control a model's capacity by expressing preferences for one function over another (through adding regularizer / penalty)

no-free-lunch thm → must design for the particular task

e.g. $J(\vec{w}) = MSE_{train} + \underbrace{\lambda \vec{w}^T \vec{w}}_{\text{preference for smaller squared } L^2 \text{ norm}}$

5.3 Hyperparameters and Validation Sets

✡ In practise, when the same test set is repeated evaluated for

Author:
kiking

different algorithms over many years (esp for research community)
we end up having over-optimistic evaluations with the test set
⇒ Benchmark became stale solution = community move on to newer dataset

5.3.1 Cross-Validation

On trial i, the i-th subset of the data → test set
Test error = est. by taking the average test error across k trials