CS229 - Machine Learning
Linear Algebra & Probability

Tim Reinhart - rtim@stanford.edu

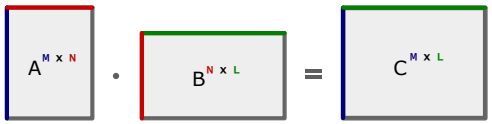
Version: November 3, 2023

Matrices

Matrix Multiplication

Matrices can be multiplied with each other in the following manner:

$$A \cdot B = C \quad \Rightarrow \quad c_{ik} = \sum_{j=1}^{n} a_{ij} \cdot b_{jk}$$



Associative & Distributive Laws:

$$(A \cdot B) \cdot C = A \cdot (B \cdot C)$$
$$(A + B) \cdot C = A \cdot C + B \cdot C$$
$$A \cdot (C + D) = A \cdot C + A \cdot D$$

Warning! The commutative law does not apply! Generally, $A \cdot B \neq B \cdot A$.

Transpose

The transpose of a matrix is obtained by "mirroring" it along its diagonal.

Example: $\begin{pmatrix} a & b \\ c & d \\ e & f \end{pmatrix}^T = \begin{pmatrix} a & c & e \\ b & d & f \end{pmatrix}$

Calculation Rules:

$$(A+B)^T = A^T + B^T \qquad (A^T)^{-1} = (A^{-1})^T$$
$$(A \cdot B)^T = B^T \cdot A^T \qquad rank(A^T) = rank(A)$$
$$(c \cdot A)^T = c \cdot A^T \qquad det(A^T) = det(A)$$
$$(A^T)^T = A \qquad eig(A^T) = eig(A)$$

Inverse

The inverse A^{-1} of A reverses a multiplication with A. When you multiply A with A^{-1}, you get the identity matrix.

Properties:

- Only square matrices can be invertible.
- An invertible matrix is called **regular**, a non-invertible one **singular**.
- The inverse is unique.
- A is invertible if and only if A has full rank.
- A is invertible if and only if A^T is invertible.
- A is symmetric if and only if A^{-1} is symmetric.
- A is a triangular matrix if and only if A^{-1} is a triangular matrix.
- A is invertible if and only if $\det(A) \neq 0$.
- A is invertible if and only if no eigenvalue $\lambda = 0$.
- A and B are invertible implies AB is invertible.

Calculation rules:

$$I^{-1} = I \qquad (A^T)^{-1} = (A^{-1})^T$$
$$(A^{-1})^{-1} = A \qquad rang(A^{-1}) = rang(A)$$
$$(A^k)^{-1} = (A^{-1})^k \qquad det(A^{-1}) = det(A)^{-1}$$
$$(c \cdot A)^{-1} = c^{-1} \cdot A^{-1} \qquad eig(A^{-1} = eig(A)^{-1}$$
$$(A \cdot B)^{-1} = B^{-1} \cdot A^{-1}$$

Matrix Tricks

Probability Rules for Matrices:

- Pull Matrix Multiply out of Variance:

$$Var[Mx] = MVar[x]M^T$$

Eigenvalues and Eigenvectors

Eigenvalues of A: $\det(A - \lambda \cdot I) = 0$

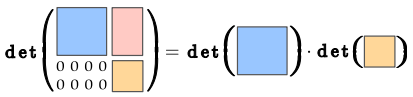
Verify Computation

- Trace(A) $= a_{11} + a_{22} + \cdots + a_{nn} = \sum \lambda_i$
- det(A) = product of λ_i

Eigenvectors: Kernel of the matrix $A - \lambda_i \cdot I$, where λ_i is the eigenvalue corresponding to the eigenvector.

Determinant

Block Sentence for Determinant Computation



Positive (Semi-)Definite Matrices

Definitions

A symmetric matrix $A \in \mathbb{R}^{n \times n}$ is called:

- **Positive Semi-Definite (PSD)** if for any non-zero vector $\mathbf{x} \in \mathbb{R}^n$, we have $\mathbf{x}^T A \mathbf{x} \geqslant 0$.
- **Positive Definite (PD)** if for any non-zero vector $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x}^T A \mathbf{x} > 0$.

Properties

- All eigenvalues of a PSD matrix are non-negative, and those of a PD matrix are positive.
- A matrix is PSD if and only if it can be written as $B^T B$, where B is any matrix.
- If A is PD (or PSD), then so is A^{-1} (if A is invertible).
- For any matrix A, the matrices $A^T A$ and AA^T are PSD.
- The sum of two PSD matrices is also PSD.

Checking for Positive (Semi-) Definiteness

Determining if a matrix is PSD or PD can be done in several ways:

- **Eigenvalue Criterion:** A symmetric matrix is PSD if and only if all its eigenvalues are non-negative. It is PD if all eigenvalues are positive.
- **Principal Minors:** A symmetric matrix A is PD if all its leading principal minors (determinants of the top-left $k \times k$ submatrix, $1 \leqslant k \leqslant n$) are positive. For PSD, all leading principal minors should be non-negative.
- **Cholesky Decomposition:** A matrix is PD if and only if it has a Cholesky decomposition. For numerical algorithms, attempting a Cholesky decomposition and checking for failure can be an effective way to test for positive definiteness.

Matrix Calculus

Gradient

The gradient of a scalar function $f : \mathbb{R}^n \to \mathbb{R}$ with respect to a vector $\mathbf{x} \in \mathbb{R}^n$ is a vector of partial derivatives:

$$\nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$$

Hessian

The Hessian matrix of a scalar-valued function $f : \mathbb{R}^n \to \mathbb{R}$ is a square matrix of second-order partial derivatives:

$$H(f)(\mathbf{x}) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

Examples

$f(\mathbf{x}) = \mathbf{A}\mathbf{x}$ $\quad \mathbf{A} \in \mathbb{R}^{m \times n}$

Gradient:

$$\nabla_x f = \mathbf{A}^T$$

$f(\mathbf{x}) = \mathbf{x}^T \mathbf{A} \mathbf{x}$ $\quad \mathbf{A} \in \mathbb{R}^{m \times n}$

Gradient:

$$\nabla_x f = (\mathbf{A} + \mathbf{A}^T)\mathbf{x} = 2\mathbf{A}\mathbf{x} \quad \text{if sym.}$$

Hessian:

$$H(f) = \mathbf{A} + \mathbf{A}^T = 2\mathbf{A} \quad \text{if sym.}$$

Linear Regression Loss (ℓ_2 norm)

For the loss function $L(\mathbf{w}) = ||\mathbf{y} - \mathbf{X}\mathbf{w}||^2$:

Gradient:

$$\nabla L = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w})$$

Hessian:

$$H(L) = 2\mathbf{X}^T\mathbf{X}$$

Logistic Regression Loss

- Binary classification with labels $y_i \in \{0, 1\}$
- Predicted probabilities $p_i = \frac{1}{1+e^{-\mathbf{x}_i^T \mathbf{w}}}$
- $L(\mathbf{w}) = -\sum_i [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$

Gradient:

$$\nabla L = \mathbf{X}^T(\mathbf{p} - \mathbf{y})$$

Hessian:

$$H(L) = \mathbf{X}^T \mathbf{S} \mathbf{X}$$

where $\mathbf{S}$ is a diagonal matrix with $S_{ii} = p_i(1 - p_i)$.

Basic Probability

Bayes Theorem

$$P(X{=}x|Y{=}y) = \frac{P(Y{=}y|X{=}x)P(X{=}x)}{P(Y{=}y)}$$

Where:

- $P(X{=}x|Y{=}y)$ is the posterior probability: the probability of event $X{=}x$ given that $Y{=}y$ has occurred.
- $P(Y{=}y|X{=}x)$ is the likelihood: the probability of observing $Y{=}y$ given $X{=}x$.
- $P(X{=}x)$ is the prior probability: the initial belief about $X{=}x$.
- $P(Y{=}y)$ is the marginal probability: the total probability of observing $Y{=}y$ under all possible outcomes of X.

Law of Total Probability

A key concept related to Bayes' Theorem is the Law of Total Probability. It is useful for calculating $P(Y{=}y)$, the marginal probability in Bayes' formula, especially when dealing with compound events. The law states:

$$P(Y{=}y) = \sum_i P(Y{=}y|X{=}x_i)P(X{=}x_i)$$

Where $X{=}x_i$ represents all disjoint outcomes that cover the sample space. In the context of Bayes' Theorem, it's used to marginalize over the different possible states of knowledge or evidence.

Bayes' Rule for Multiple Events

In cases involving more than two events, Bayes' Theorem can be generalized as:

$$P(X_1{=}x_1, \ldots, X_n{=}x_n|Y{=}y) = \frac{P(Y{=}y|X_1{=}x_1, \ldots, X_n{=}x_n)\prod_{i=1}^n P(X_n{=}x_n)}{P(Y{=}y)}$$

Bayes' Theorem with Continuous Variables

When dealing with continuous variables, Bayes' Theorem takes the form of probability densities:

$$f_{X|Y}(x|y) = \frac{f_{Y|X}(y|x)f_X(x)}{f_Y(y)}$$

Where $f_{X|Y}(x|y)$ is the conditional density of X given Y, and so on.

Prior and Posterior Probabilities

In Bayesian analysis, the prior probability $P(X{=}x)$ represents our belief about X before observing the evidence Y, while the posterior probability $P(X{=}x|Y{=}y)$ is our updated belief after observing Y. The transformation from the prior to the posterior, via the likelihood and marginal likelihood, is the essence of Bayesian inference.

Expectation Value

$$\mathrm{E}[X] = \sum_i x_i p_i \quad \text{(for discrete var.)} \quad \text{or}$$

$$\mathrm{E}[X] \equiv \int_\Omega X\, d\mathrm{P} = \int_{\mathbb{R}} x f(x)\, dx \quad \text{(for cont. var.)}$$

Properties of Expectation

Linearity The expectation operator is linear:

$$\mathrm{E}[aX + bY] = a\,\mathrm{E}[X] + b\,\mathrm{E}[Y]$$

where a and b are constants, and X and Y are random variables.

Monotonicity If $X \leqslant Y$ (i.e., X is always less than or equal to Y), then:

$$\mathrm{E}[X] \leqslant \mathrm{E}[Y]$$

Law of the Unconscious Statistician This law states that if $Y = g(X)$ for some function g, then:

$$\mathrm{E}[Y] = \mathrm{E}[g(X)] = \sum_x g(x)P(X = x) \quad \text{(discrete case)}$$

or

$$\mathrm{E}[g(X)] = \int_{-\infty}^{+\infty} g(x)f_X(x)\, dx \quad \text{(continuous case)}$$

where $f_X(x)$ is the probability density function of X.

Independence If two random variables X and Y are independent, then:

$$\mathrm{E}[XY] = \mathrm{E}[X] \cdot \mathrm{E}[Y]$$

Conditional Expectation

$$\mathrm{E}(X|Y = y) = \sum_x xP(X = x|Y = y) = \sum_x x\frac{P(X = x, Y = y)}{P(Y = y)}$$

Variance

Variance quantifies the spread or dispersion of a set of data points or the spread of a probability distribution. It is defined as the expected value of the squared deviation from the mean (denoted by μ):

$$Var(X) = \mathrm{E}[(X - \mu)^2] = \mathrm{E}[X^2] - (\mathrm{E}[X])^2$$

Properties of Variance

Non-negativity The variance is always non-negative:

$$Var(X) \geqslant 0$$

Variance of a Constant

$$Var(a) = 0$$

where $a \in \mathbb{R}$ is a constant.

Factor Out Constants

$$Var(aX) = a^2 Var(X)$$

where $a \in \mathbb{R}$ is a constant.

Variance of a Sum For any random variables X, Y:

$$\mathrm{Var}(aX + bY) = a^2\,\mathrm{Var}(X) + b^2\,\mathrm{Var}(Y) + 2ab\,\mathrm{Cov}(X, Y)$$

$$\mathrm{Var}(aX - bY) = a^2\,\mathrm{Var}(X) + b^2\,\mathrm{Var}(Y) - 2ab\,\mathrm{Cov}(X, Y)$$

If X and Y are independent, then $Cov(X, Y) = 0$, and this simplifies to:

$$\mathrm{Var}(aX \pm bY) = a^2\,\mathrm{Var}(X) + b^2\,\mathrm{Var}(Y)$$

Sum of uncorrelated variables

$$\mathrm{Var}\left(\sum_{i=1}^n X_i\right) = \sum_{i=1}^n \mathrm{Var}(X_i)$$

Sum of correlated variables

$$\mathrm{Var}\left(\sum_{i=1}^n X_i\right) = \sum_{i=1}^n \sum_{j=1}^n \mathrm{Cov}\left(X_i, X_j\right) = \sum_{i=1}^n \mathrm{Var}\left(X_i\right) + 2\sum_{1 \leqslant i < j \leqslant n} \mathrm{Cov}\left(X_i, X_j\right)$$

Probability Distributions

Discrete Distributions

Bernoulli Distribution

One Trial, Two outcomes

PMF:

$$P(y;\phi) = \begin{cases} \phi & y = 1 \\ 1 - \phi & y = 0 \end{cases}$$

$$P(y;\phi) = \phi^y(1 - \phi)^{1-y} \quad \text{for } y \in \{0, 1\}$$

Mean and Variance:

$$\mu = p, \quad \sigma^2 = p(1 - p)$$

Binomial Distribution

Distribution of the number of successes in a sequence of n independent experiments, each with two outcomes, and each with its own Boolean-valued outcome: success (with probability p) or failure (with probability $q = 1 - p$) → Multiple Trials, Two outcomes

PMF: The probability of getting exactly k successes in n independent Bernoulli trials (with the same rate p) is given by

$$P(X = k) = \binom{n}{k} p^k (1 - p)^{n-k}$$

Mean and Variance:

$$\mu = np, \quad \sigma^2 = np(1 - p)$$

Multinomial Distribution

Multiple Trials, Multiple outcomes

PMF:

$$P(\mathbf{X} = \mathbf{x}) = \frac{n!}{x_1!x_2! \cdots x_k!} p_1^{x_1} p_2^{x_2} \cdots p_k^{x_k}$$

Mean and Variance: For each category i,

$$\mu_i = np_i, \quad \sigma_i^2 = np_i(1 - p_i)$$

Covariance for categories i and j,

$$\mathrm{Cov}(X_i, X_j) = -np_ip_j \quad (i \neq j)$$

Poisson Distribution

PMF:

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

Mean and Variance:

$$\mu = \sigma^2 = \lambda$$

Geometric Distribution

PMF:

$$P(X = k) = (1 - p)^{k-1}p$$

Mean and Variance:

$$\mu = \frac{1}{p}, \quad \sigma^2 = \frac{1 - p}{p^2}$$

Continuous Distributions

Exponential Distribution

PDF:

$$f(x) = \lambda e^{-\lambda x}, \quad x \geqslant 0$$

Mean and Variance:

$$\mu = \frac{1}{\lambda}, \quad \sigma^2 = \frac{1}{\lambda^2}$$

Uniform Distribution

PDF:

$$f(x) = \frac{1}{b - a} \quad \text{for } a \leqslant x \leqslant b$$

Mean and Variance:

$$\mu = \frac{a+b}{2}, \quad \sigma^2 = \frac{(b-a)^2}{12}$$

Beta Distribution

PDF:

$$f(x;\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\mathrm{B}(\alpha,\beta)}$$

Mean and Variance:

$$\mu = \frac{\alpha}{\alpha+\beta}, \quad \sigma^2 = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$$

Gamma Distribution

PDF:

$$f(x;\alpha,\beta) = \frac{\beta^\alpha x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)} \quad \text{for } x > 0$$

Mean and Variance:

$$\mu = \frac{\alpha}{\beta}, \quad \sigma^2 = \frac{\alpha}{\beta^2}$$

Dirichlet Distribution

PDF:

$$f(\mathbf{x};\boldsymbol{\alpha}) = \frac{1}{\mathrm{B}(\boldsymbol{\alpha})} \prod_{i=1}^{K} x_i^{\alpha_i - 1}$$

Where $\mathrm{B}(\boldsymbol{\alpha})$ is the multivariate beta function.

Mean and Variance: For each component i,

$$\mu_i = \frac{\alpha_i}{\sum_{j=1}^{K} \alpha_j}$$

Exponential Family

A single-parameter exponential family is a set of probability distributions whose probability density function (or probability mass function, for the case of a discrete distribution) can be expressed in the form

$$p(y;\eta) = b(\eta)\, \exp\big[\, \eta^T T(y) - a(\eta)\,\big]$$

- η: natural parameter
- $T(y)$: sufficient statistic
- $a(\eta)$: log partition function

Canonical Response Funtion

$$g(\eta) = \mathrm{E}[T(y);\eta]$$

- For the Gaussian family: identify function
- For the Bernoulli family: logistic function

Gaussian Distributions

Univariate Gaussian

The probability density of a univariate Gaussian distribution is given by:

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

Where μ is the mean and σ^2 is the variance.

Multivariate Gaussian

The probability density of a multivariate Gaussian distribution in d dimensions is:

$$\mathcal{N}(\mathbf{x};\boldsymbol{\mu},\Sigma) = \frac{1}{\sqrt{(2\pi)^k|\Sigma|}} \exp\left(-\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

Where $\boldsymbol{\mu} \in \mathbb{R}^d$ is the mean vector, $\Sigma \in \mathbb{R}^{d\times d}$ is the covariance matrix.

Mean Vector and Covariance Matrix

- The mean vector $\boldsymbol{\mu}$ represents the mean of each dimension. If $\mathbf{x}$ is an n-dimensional random vector, then $\boldsymbol{\mu}$ is given by $\boldsymbol{\mu} = \mathrm{E}[\mathbf{x}]$.
- The covariance matrix Σ is symmetric and positive semi-definite. It represents how each pair of dimensions of the random vector $\mathbf{x}$ co-varies. If $\mathbf{x}$ has dimensions $x_1, x_2, ..., x_n$, then the element Σ_{ij} of the matrix Σ is the covariance between x_i and x_j: $\Sigma_{ij} = \mathrm{Cov}(x_i, x_j)$.
- The determinant of Σ (denoted as $|\Sigma|$) and its inverse Σ^{-1} play a key role in defining the shape and orientation of the multivariate Gaussian distribution in its multi-dimensional space.

Maximum Likelihood Estimation (MLE)

Definition

The likelihood function for a model parameter θ given observations $X = (x_1, x_2, ..., x_n)$ is defined as:

$$\mathcal{L}(\theta|X) = P(X|\theta) = \prod_{i=1}^{n} P(x_i|\theta)$$

where $P(x_i|\theta)$ is the probability of observing x_i given the parameter θ.
For Discriminative Algorithms:

$$\mathcal{L}(\theta) = \prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)};\theta)$$

For Generative Algorithms:

$$\mathcal{L}(\theta) = \prod_{i=1}^{n} p(x^{(i)} \mid y^{(i)};\theta) \cdot p(y^{(i)};\theta)$$

Estimation

The MLE $\hat{\theta}$ is the value of θ that maximizes $\mathcal{L}(\theta|X)$. In practice, it's often more convenient to maximize the log-likelihood because the logarithm is a strictly increasing function and transforms the product into a sum, simplifying the differentiation:

$$\ell(\theta|X) = \log \mathcal{L}(\theta|X) = \sum_{i=1}^{n} \log P(x_i|\theta)$$

Finding the MLE

To find the MLE, we take the derivative of $\ell(\theta|X)$ with respect to θ and set it to zero (for parameter θ where this is possible):

$$\frac{\partial}{\partial\theta}\ell(\theta|X) = 0$$

Solving this equation gives us the MLE $\hat{\theta}$. In cases where the derivative does not exist or the equation is hard to solve, numerical methods such as gradient descent can be used.

Properties of MLE

- **Consistency:** As the number of observations n approaches infinity, the MLE converges in probability to the true value of θ.
- **Asymptotic normality:** Under regular conditions, as n increases, the distribution of $\hat{\theta}$ approaches a normal distribution centered around the true θ, with a variance that can be estimated from the data.
- **Invariance:** If $\hat{\theta}$ is the MLE of θ, and g is a function, then the MLE of $g(\theta)$ is $g(\hat{\theta})$.
- **Efficiency:** Under certain conditions, MLE achieves the Cramér-Rao lower bound, making it the most efficient unbiased estimator.

Examples

MLE for Normal Distribution

Consider a dataset $X = \{x_1, x_2, \ldots, x_n\}$ assumed to be drawn from a normal distribution $\mathcal{N}(\mu, \sigma^2)$. The MLE for μ and σ^2 are given by:

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad \hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu})^2$$

MLE for a Binomial Distribution

For a binomial distribution $B(n,p)$ with known n and unknown p, given k successes in n trials, the MLE of p is:

$$\hat{p} = \frac{k}{n}$$

Optimization Methods in Machine Learning

Newton's Method

Given a twice-differentiable function $f(\mathbf{x})$, Newton's method updates the parameter vector $\mathbf{x}$ as follows:

$$\mathbf{x}_{\text{new}} = \mathbf{x} - \left[\nabla^2 f(\mathbf{x})\right]^{-1} \nabla f(\mathbf{x})$$

where $\nabla f(\mathbf{x})$ and $\nabla^2 f(\mathbf{x})$ are the gradient and Hessian matrix of f at $\mathbf{x}$, respectively. Despite its quadratic convergence rate, Newton's method can be computationally expensive due to the need to compute and invert the Hessian matrix.

Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent is a simplification of the traditional gradient descent algorithm. It updates the parameters using the gradient of the loss function with respect to a **single** data point (or a small subset) chosen randomly in each iteration. The update rule is:

$$\mathbf{x}_{\text{new}} = \mathbf{x} - \eta \nabla f_i(\mathbf{x})$$

where η is the learning rate, and $\nabla f_i(\mathbf{x})$ is the gradient computed for the ith data point. While SGD can be noisy compared to full-batch gradient descent, it allows for more frequent updates, which can be beneficial for large datasets.

Mini-Batch Gradient Descent

Mini-Batch Gradient Descent is a variation of SGD where instead of using a single sample at each iteration, a mini-batch of samples is used. This method aims to balance the efficiency of using batches with the reduced variance of the gradient estimate. The update rule for a mini-batch is:

$$\mathbf{x}_{\text{new}} = \mathbf{x} - \eta \nabla f_{\mathbf{B}}(\mathbf{x})$$

where $\mathbf{B}$ represents a mini-batch of samples and $\nabla f_{\mathbf{B}}(\mathbf{x})$ is the gradient of the loss function averaged over all samples in the mini-batch.

Normal Equation for Linear Least Squares

The Normal Equation is a method to analytically solve the linear least squares problem for finding the best-fitting parameters in a linear regression model. Given a design matrix $\mathbf{X}$ and a target vector $\mathbf{y}$, the optimal parameter vector $\mathbf{w}$ is given by:

$$\mathbf{w} = \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{y}$$

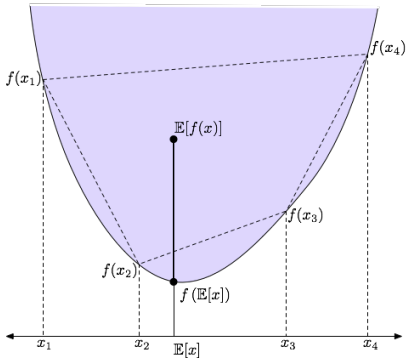
This equation directly computes the parameters that minimize the squared difference between the predictions and the actual values. It's computationally efficient for small to moderate-sized datasets but can be computationally expensive or numerically unstable for very large datasets.

Important Theorems

Jensen's Inequality

Let X be an integrable real-valued random variable and ϕ a convex function. Then:

$$\varphi\left(\mathrm{E}[X]\right) \leqslant \mathrm{E}\left[\varphi(X)\right]$$



Concavity

A real-valued function f on an interval (or, more generally, a convex set in vector space) is said to be concave if, for any x and y in the interval and for any $\alpha \in [0,1]$:

$$f((1-\alpha)x + \alpha y) \geqslant (1-\alpha)f(x) + \alpha f(y)$$

Decision Trees

Feature Split selection

Given a subset of data M (node in a tree).

1. For each feature $h_i(x)$:
   (a) Split data of M according to feature $h_i(x)$
   (b) Compute misclassification error of split:
   $$\frac{\#\text{ mistakes}}{\#\text{ datapoints}}$$
2. Choose feature $h^\star$ with lowest error

Threshold Split Selection

1. Sort the values of a feature $h_j(x)$. Let $\{v_1, \ldots, v_n\}$ denote sorted values.
2. For $i = 1 \ldots n-1$:
   (a) Consider split $t_i = \frac{v_i + v_{[i+1]}}{2}$
   (b) Compute error for split $h_j(x) \geqslant t_i$
3. Choose $t^\star$ with lowest error

Threshold split caveat: same feature can be used multiple times!

Greedy Decision Tree Learning

1. Start with empty tree
2. Select a feature to split on (see above)
3. For each split of tree:
   (a) If done, make predictions → Stopping Cond.
   (b) Otherwise, go to step 2 and recurse on this split

Stopping Conditions

- All data agrees on y
- Already split on all features
- NOT: Error does not decrease!

Tree Pruning Algorithm

Repeat for every split in Tree:

1. Compute total cost $C(T)$ of split using nr. of leaves $L(T)$:
   $$C(T) = \text{Error}(T) + \lambda L(T)$$
2. Undo the split to get T_{small} and calculate cost of smaller tree
3. Prune split if total cost is lower

Boosting

Classification $y = \pm 1$ from input x. Ensemble model consists of classifiers $f_1(x), \ldots, f_T(x)$ and coefficients $\hat{w}_1, \ldots, \hat{w}_T$. Prediciton:

$$\hat{y} = sign\left(\sum \hat{w}_t f_t(x)\right) \qquad (1)$$

Boosting: focus learning on 'hard' points, using weighted data. Each x_i, y_i weighted by α_i. Loss function becomes:

$$\sum_{i=1}^{n} \alpha_i \ell(x_i, y_i; \theta)$$

Gradient of loss will be multiplied by α_i, in a decision tree a data point counts α_i times.

AdaBoost Algorithm

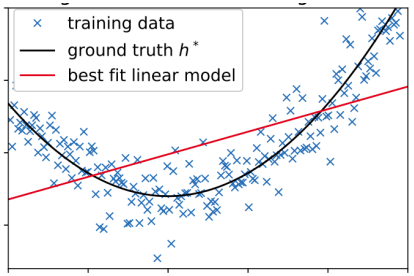
1. Start with $\alpha_i = \frac{1}{n}$ for all data points
2. For $t = 1, \ldots, T$:
   (a) Learn $f_t(x)$ with data weights α_i
   (b) Compute coefficients $\hat{w}_t$ using:
   $$\hat{w}_t = \frac{1}{2}\ln\left(\frac{1 - \text{weightederr}(f_t)}{\text{weightederr}(f_t)}\right)$$
   $$\text{weightederr}(f_t) = \sum_{i=1}^{n} \alpha_i 1\{f_t(x_i) \neq y_i\}$$
   (c) Recompute weights α_i using:
   $$\alpha_i := \begin{cases} \alpha_i \exp(-\hat{w}_t) & \text{if } f_t(x_i) = y_i \\ \alpha_i \exp(\hat{w}_t) & \text{if } f_t(x_i) \neq y_i \end{cases}$$
   which decreases the weight for data points where the classifier got it right and increases the weights otherwise.
   (d) Normalize weights
   $$\alpha_i := \frac{\alpha_i}{\sum_j^n \alpha_j}$$

Training error of Ensemble will go to zero (under some conditions).

Bias-Variance Tradeoff

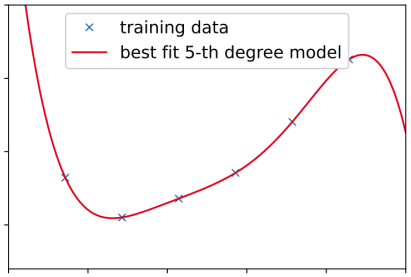
Bias

The bias of a model is defined to be the test error even if we were to fit it to a very (say, infinitely) large training dataset. Thus, if a model suffers from large bias, it **underfits** (i.e., fails to capture structure exhibited by).

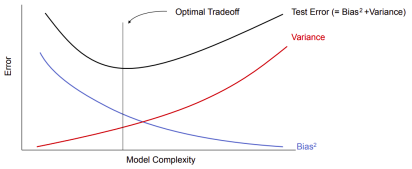


Variance

When fitting a complex model, there is a large risk that we are fitting patterns in the data that happened to be present in our small, finite training set, but that do not reflect the wider pattern of the relationship between x and y.
"Spurious" patterns in the training set are (mostly) due to the observation noise, and fitting these spurious patters results in a model with large test error.



Tradeoff



- More Training samples: Bias ?, Variance ↓
- Less Training samples: Bias ?, Variance ↑*
- Smaller set of features: Bias ↑, Variance ↓
- Larger Set of features: Bias ↓, Variance ↑
- Try email header features: Bias ↓, Variance ↑
- Replace the value of your last feature by a random number: Bias ↑**, Variance ↑
- Stopping Optimization early: Bias ↑, Variance ↓

⋆:*A specific classifier (with a fixed model complexity) will be more likely to overfit to noise in the training data when there is less training data, and is therefore more likely to overfit.*
⋆⋆: *Bias should increase as you are replacing information that could be useful for your prediction with a meaningless feature.*

Kernels

Kernelizing:

$$\text{step 1: } c^{[i](t+1)} = \arg\min_j \|x^{(t)} - \mu_j\|^2$$

$$\text{step 2: } \mu_j^{(t+1)} = \frac{\sum_{i=1}^m 1\{c^{(t+1)} = j\} x^{(i)}}{\sum_{i=1}^m 1\{c^{(t+1)} = j\}}$$

Answer:
Given the formula for $c^{(t+1)}$, we want to express the distance in the feature space:

$$c^{[i](t+1)} = \arg\min_j \|\phi(x^{(i)}) - \mu_j\|^2$$

(def. of distance in feature space)

Expanding the squared norm, we get:

$$= \arg\min_j \langle\phi(x^{(i)}), \phi(x^{(i)})\rangle + \langle\mu_j, \mu_j\rangle - 2\langle\phi(x^{(i)}), \mu_j\rangle$$

Using the definition of the mean μ_j and the given update rules, we can expand:

$$= \arg\min_j \langle \phi(x^{(i)}), \phi(x^{(i)}) \rangle$$
$$- 2\phi(x^{(i)})^T \left(\frac{1}{|S_j|} \sum_{l \in S_j} \phi(x^{(l)}) \right)$$
$$\text{(where } S_j = \{l : c^{(l)} = j\}\text{)}$$

Now, to simplify calculations in the feature space, we use a kernel function, $K(u, v) = \langle \phi(u), \phi(v) \rangle$.

$$= \arg\min_j K(x^{(i)}, x^{(i)})$$
$$- 2 \left(\frac{1}{|S_j|} \sum_{l \in S_j} K(x^{(t)}, x^{(l)}) \right)$$
$$+ \frac{1}{|S_j|^2} \sum_{l,p \in S_j} K(x^{(l)}, x^{(p)})$$

This provides a kernelized version of the update formula for $c^{(t+1)}$, where operations are defined in terms of the kernel function K, which implicitly computes inner products in the infinite-dimensional feature space.