Complete Mathematical Formulas in Machine Learning

Table of Contents

Foundational Mathematics

Basic Notation

- **Scalar**: Single value, denoted as x, y, a, b
- **Vector**: Column of values, denoted as **x**, **y** (bold)
- **Matrix**: 2D array, denoted as **X**, **W** (bold capital)
- **Tensor**: n-dimensional array

Summation and Product Notation

Summation:

$$\Sigma_{i=1}^{n} x_i = x_1 + x_2 + ... + x_n$$

Product:

$$\prod_{i=1}^{n} x_i = x_1 \times x_2 \times \ldots \times x_n$$

Statistical Measures

Mean (Average)

Arithmetic Mean:

$$\mu = \bar{x} = (1/n) \times \sum_{i=1}^{n} x_i$$

- μ (mu) or $\bar{x}$ represents the mean
- n is the number of observations
- x_i is the i-th observation

Variance

Population Variance:

$$\sigma^2 = (1/n) \times \sum_{i=1}^{n} (x_i - \mu)^2$$

Sample Variance (Bessel's correction):

$$s^2 = (1/(n-1)) \times \sum_{i=1}^{n} (x_i - \bar{x})^2$$

Standard Deviation

$$\sigma = \sqrt{(\sigma^2)} \quad \text{(population)}$$
$$s = \sqrt{(s^2)} \quad \text{(sample)}$$

Covariance

Between two variables X and Y:

$$\mathrm{Cov}(X,Y) = (1/n) \times \sum_{i=1}^{n} (x_i - \mu_x)(y_i - \mu_y)$$

Correlation Coefficient

Pearson Correlation:

$$\rho(X,Y) = \mathrm{Cov}(X,Y) / (\sigma_x \times \sigma_y)$$

- Range: [-1, 1]

- -1: perfect negative correlation
- 0: no linear correlation
- 1: perfect positive correlation

Standardization (Z-score)

$$z = (x - \mu) / \sigma$$

- Transforms data to have $\mu = 0$ and $\sigma = 1$

Linear Algebra in ML

Vector Operations

Dot Product (Inner Product):

$$a \cdot b = \Sigma_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \dots + a_n b_n$$

Vector Norm (Length):

- L1 Norm (Manhattan): $||x||_1 = \Sigma_i |x_i|$
- L2 Norm (Euclidean): $||x||_2 = \sqrt{(\Sigma_i x_i^2)}$
- L∞ Norm (Max): $||x||\infty = \max_i |x_i|$

Matrix Operations

Matrix Multiplication:

$$C = AB \text{ where } C_{ij} = \Sigma_k A_{ik} B_{kj}$$

Matrix Transpose:

$$(A^T)_{ij} = A_{ji}$$

Matrix Inverse:

$$AA^{-1} = A^{-1}A = I$$

where I is the identity matrix

Eigenvalues and Eigenvectors

For matrix A and vector v:

```
Av = λv
```

- λ is the eigenvalue
- v is the eigenvector

Characteristic Equation:

```
det(A - λI) = 0
```

Calculus in ML

Derivatives

Basic Rules:

- Power Rule: $d/dx(x^n) = nx^{n-1}$
- Chain Rule: $d/dx[f(g(x))] = f'(g(x)) \times g'(x)$
- Product Rule: $d/dx[f(x)g(x)] = f'(x)g(x) + f(x)g'(x)$

Partial Derivatives

For function f(x,y):

```
∂f/∂x = partial derivative with respect to x
∂f/∂y = partial derivative with respect to y
```

Gradient

For function $f(x_1, x_2, ..., x_n)$:

$$\nabla f = [\partial f/\partial x_1, \partial f/\partial x_2, ..., \partial f/\partial x_n]^T$$

Gradient Descent Update Rule

$$\theta_{t+1} = \theta_t - \alpha\nabla f(\theta_t)$$

- θ: parameters
- α: learning rate
- ∇f: gradient of loss function

Hessian Matrix

Second-order partial derivatives:

$$H(f)_{ij} = \partial^2 f/(\partial x_i \partial x_j)$$

Probability Theory

Basic Probability

$$P(A) \in [0, 1]$$
$$P(\Omega) = 1 \quad \text{(probability of sample space)}$$
$$P(\emptyset) = 0 \quad \text{(probability of empty set)}$$

Conditional Probability

$$P(A|B) = P(A \cap B) / P(B)$$

- $P(A|B)$: probability of A given B

Bayes' Theorem

$$P(A|B) = [P(B|A) \times P(A)] / P(B)$$

Extended form:

$$P(A|B) = [P(B|A) \times P(A)] / [\Sigma_i P(B|A_i) \times P(A_i)]$$

Probability Distributions

Bernoulli Distribution:

$$P(X = x) = p^x(1-p)^{1-x} \text{ for } x \in \{0,1\}$$

Binomial Distribution:

$$P(X = k) = C(n,k) \times p^k \times (1-p)^{n-k}$$

where $C(n,k) = n!/(k!(n-k)!)$

Normal (Gaussian) Distribution:

$$f(x) = (1/\sqrt{(2\pi\sigma^2)}) \times \exp(-(x-\mu)^2/(2\sigma^2))$$

Multivariate Normal:

```
f(x) = (1/((2π)^(k/2)|Σ|^(1/2))) × exp(-½(x-μ)ᵀΣ⁻¹(x-μ))
```

- Σ: covariance matrix
- |Σ|: determinant of Σ

Expectation and Variance

Expectation (Expected Value):

- Discrete: $E[X] = \Sigma_x x \times P(X = x)$
- Continuous: $E[X] = \int x \times f(x)dx$

Variance:

```
Var(X) = E[(X - E[X])²] = E[X²] - (E[X])²
```

Information Theory

Entropy

For discrete variable X:

```
H(X) = -Σᵢ P(xᵢ) × log₂(P(xᵢ))
```

- Measures uncertainty/information content
- Units: bits ($\log_2$) or nats (ln)

Cross-Entropy

Between distributions P and Q:

```
H(P,Q) = -Σᵢ P(xᵢ) × log(Q(xᵢ))
```

Kullback-Leibler (KL) Divergence

```
KL(P||Q) = Σᵢ P(xᵢ) × log(P(xᵢ)/Q(xᵢ))
```

- Measures difference between distributions
- $KL(P||Q) \neq KL(Q||P)$ (not symmetric)

Mutual Information

$$I(X;Y) = \sum_x \sum_y P(x,y) \times \log(P(x,y)/(P(x)P(y)))$$

Loss Functions

Regression Loss Functions

Mean Squared Error (MSE):

$$MSE = (1/n) \times \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Mean Absolute Error (MAE):

$$MAE = (1/n) \times \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

Huber Loss (Robust to outliers):

$$L\_\delta(y,\hat{y}) = \begin{cases} \frac{1}{2}(y-\hat{y})^2 & \text{if } |y-\hat{y}| \le \delta \\ \delta|y-\hat{y}| - \frac{1}{2}\delta^2 & \text{if } |y-\hat{y}| > \delta \end{cases}$$

Classification Loss Functions

Binary Cross-Entropy (Log Loss):

$$BCE = -(1/n) \times \sum_{i=1}^{n} [y_i\log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)]$$

Categorical Cross-Entropy (Multi-class):

$$CCE = -(1/n) \times \sum_{i=1}^{n} \sum_{j=1}^{m} y_{ij}\log(\hat{y}_{ij})$$

- m: number of classes
- y_{ij}: 1 if sample i belongs to class j, 0 otherwise

Hinge Loss (SVM):

$$L = \max(0, 1 - y \times \hat{y})$$

Optimization Algorithms

Gradient Descent Variants

Batch Gradient Descent:

$$\theta = \theta - \alpha \times (1/n) \times \Sigma_{i=1}^{n} \nabla\theta L(x_i, y_i, \theta)$$

Stochastic Gradient Descent (SGD):

$$\theta = \theta - \alpha \times \nabla\theta L(x_i, y_i, \theta)$$

Mini-batch Gradient Descent:

$$\theta = \theta - \alpha \times (1/m) \times \Sigma_{i=1}^{m} \nabla\theta L(x_i, y_i, \theta)$$

Advanced Optimizers

Momentum:

$$v_t = \beta v_{t-1} + \alpha\nabla\theta L$$
$$\theta_t = \theta_{t-1} - v_t$$

RMSprop:

$$s_t = \beta s_{t-1} + (1-\beta)(\nabla\theta L)^2$$
$$\theta_t = \theta_{t-1} - \alpha \times \nabla\theta L/\sqrt{(s_t + \varepsilon)}$$

Adam (Adaptive Moment Estimation):

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla\theta L \qquad \text{(1st moment)}$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla\theta L)^2 \qquad \text{(2nd moment)}$$
$$\hat{m}_t = m_t/(1-\beta_1^{\,t}) \qquad \text{(bias correction)}$$
$$\hat{v}_t = v_t/(1-\beta_2^{\,t}) \qquad \text{(bias correction)}$$
$$\theta_t = \theta_{t-1} - \alpha \times \hat{m}_t/(\sqrt{\hat{v}_t} + \varepsilon)$$

Linear Models Mathematics

Linear Regression

Model:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n = x^T\beta$$

Normal Equation (Closed-form solution):

$$\beta = (X^T X)^{-1} X^T y$$

Cost Function (MSE):

$$J(\beta) = (1/2n) \times \Sigma_{i=1}^{n} (y_i - x_i^T\beta)^2$$

Gradient:

$$\nabla\beta J = -(1/n) \times X^T(y - X\beta)$$

Ridge Regression (L2 Regularization)

Cost Function:

$$J(\beta) = (1/2n) \times \Sigma_{i=1}^{n} (y_i - x_i^T\beta)^2 + \lambda||\beta||_2^2$$

Solution:

$$\beta = (X^T X + \lambda I)^{-1} X^T y$$

Lasso Regression (L1 Regularization)

Cost Function:

$$J(\beta) = (1/2n) \times \Sigma_{i=1}^{n} (y_i - x_i^T\beta)^2 + \lambda||\beta||_1$$

- No closed-form solution
- Solved using coordinate descent or proximal gradient

Logistic Regression

Sigmoid Function:

$$\sigma(z) = 1/(1 + e^{-z})$$

Model:

$$P(y=1|x) = \sigma(x^T\beta) = 1/(1 + e^{-x^T\beta})$$

Log-Likelihood:

$$LL = \Sigma_{i=1}^{n} [y_i \log(\sigma(x_i^T\beta)) + (1-y_i)\log(1-\sigma(x_i^T\beta))]$$

Gradient:

$$\nabla\beta LL = X^T(y - \sigma(X\beta))$$

Tree-Based Models Mathematics

Decision Trees

Gini Impurity:

$$Gini = 1 - \Sigma_{j=1}^{c} p_j^2$$

- c: number of classes
- p_j: proportion of samples in class j

Entropy:

$$Entropy = -\Sigma_{j=1}^{c} p_j \log_2(p_j)$$

Information Gain:

$$IG = Entropy(parent) - \Sigma_i (n_i/n) \times Entropy(child_i)$$

- n_i: number of samples in child i
- n: total samples in parent

Variance Reduction (for regression):

$$VR = Var(parent) - \Sigma_i (n_i/n) \times Var(child_i)$$

Random Forest

Prediction (Regression):

$$\hat{y} = (1/B) \times \Sigma_{\beta=1}^{B} T_\beta(x)$$

- B: number of trees
- T_β: prediction from tree b

Prediction (Classification):

$$\hat{y} = \text{mode}\{T_1(x), T_2(x), ..., T^B(x)\}$$

Feature Importance:

$$\text{Importance}_j = (1/B) \times \Sigma_{\beta=1}^{B} \Sigma_{t}\in_{\beta} 1(v(t)=j) \times p(t)\Delta_{it}$$

- v(t): variable used at node t
- p(t): proportion of samples reaching node t
- Δ_{it}: impurity decrease at node t

Gradient Boosting

Additive Model:

$$F(x) = \Sigma_{m=1}^{M} \gamma_m h_m(x)$$

Update Rule:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

Gradient Calculation:

$$r_{im} = -[\partial L(y_i, F(x_i))/\partial F(x_i)]\_\{F=F_{m-1}\}$$

Support Vector Machines Mathematics

Hard Margin SVM

Objective:

$$\text{minimize: } ½||w||^2$$
$$\text{subject to: } y_i(w^T x_i + b) \geq 1 \text{ for all } i$$

Soft Margin SVM

Objective:

$$\text{minimize: } \tfrac{1}{2}||w||^2 + C \times \Sigma_i \xi_i$$
$$\text{subject to: } y_i(w^T x_i + b) \geq 1 - \xi_i$$
$$\xi_i \geq 0$$

- ξ_i: slack variables
- C: regularization parameter

Kernel Trick

Kernel Function:

$$K(x, x') = \phi(x)^T\phi(x')$$

Common Kernels:

- Linear: $K(x, x') = x^T x'$
- Polynomial: $K(x, x') = (\gamma x^T x' + r)^d$
- RBF (Gaussian): $K(x, x') = \exp(-\gamma||x - x'||^2)$
- Sigmoid: $K(x, x') = \tanh(\gamma x^T x' + r)$

Dual Form Prediction:

$$f(x) = \Sigma_{i=1}^n \alpha_i y_i K(x_i, x) + b$$

Neural Network Mathematics

Forward Propagation

Linear Transformation:

$$z^{(l)} = W^{(l)}a^{(l-1)} + b^{(l)}$$

Activation:

$$a^{(l)} = g^{(l)}(z^{(l)})$$

Activation Functions

ReLU (Rectified Linear Unit):

$$\text{ReLU}(z) = \max(0, z)$$

Leaky ReLU:

$$\text{LeakyReLU}(z) = \max(\alpha z, z) \quad \text{where } \alpha \approx 0.01$$

Sigmoid:

$$\sigma(z) = 1/(1 + e^{-z})$$

Tanh:

$$\tanh(z) = (e^{z} - e^{-z})/(e^{z} + e^{-z})$$

Softmax (Multi-class output):

$$\text{softmax}(z_i) = e^{(z_i)} / \Sigma_j \, e^{(z_j)}$$

Backpropagation

Chain Rule Application:

$$\partial L/\partial W^{(l)} = \partial L/\partial z^{(l)} \times \partial z^{(l)}/\partial W^{(l)}$$

Error Propagation:

$$\delta^{(l)} = (W^{(l+1)})^{T}\delta^{(l+1)} \odot g'(z^{(l)})$$

- $\odot$: element-wise multiplication

Weight Update:

$$W^{(l)} = W^{(l)} - \alpha \times \delta^{(l)}(a^{(l-1)})^{T}$$

Batch Normalization

Normalization:

$$\hat{x}_i = (x_i - \mu B)/\sqrt{(\sigma B^2 + \varepsilon)}$$

Scale and Shift:

$$y_i = \gamma \hat{x}_i + \beta$$

- γ, β: learnable parameters

Dropout

Training:

$$a^{(l)} = a^{(l)} \odot m^{(l)} / p$$

- $m^{(l)}$: binary mask (Bernoulli(p))
- p: keep probability

Clustering Mathematics

K-Means

Objective Function:

$$J = \Sigma_{i=1}^{n} \Sigma_{k=1}^{K} r_{ik}||x_i - \mu_k||^2$$

- r_{ik}: 1 if x_i belongs to cluster k, 0 otherwise

Update Rules:

$$\mu_k = (\Sigma_i r_{ik}x_i)/(\Sigma_i r_{ik}) \quad \text{(centroid update)}$$
$$r_{ik} = 1 \text{ if } k = \text{argmin\_j } ||x_i - \mu_j||^2 \quad \text{(assignment)}$$

DBSCAN

Core Point:

$$|N\varepsilon(p)| \geq \text{minPts}$$

- $N\varepsilon(p) = \{q \in D \mid \text{dist}(p,q) \leq \varepsilon\}$

Density-Reachable:

- p is reachable from q if there's a chain of core points

Gaussian Mixture Models

Probability Model:

$$p(x) = \Sigma_{k=1}^{K} \pi_k \times N(x|\mu_k, \Sigma_k)$$

E-step (Expectation):

$$\gamma_{ik} = (\pi_k \times N(x_i|\mu_k, \Sigma_k))/(\Sigma_j \pi_j \times N(x_i|\mu_j, \Sigma_j))$$

M-step (Maximization):

$$\pi_k = (1/n) \times \Sigma_i \gamma_{ik}$$
$$\mu_k = (\Sigma_i \gamma_{ik} x_i)/(\Sigma_i \gamma_{ik})$$
$$\Sigma_k = (\Sigma_i \gamma_{ik}(x_i - \mu_k)(x_i - \mu_k)^T)/(\Sigma_i \gamma_{ik})$$

Dimensionality Reduction Mathematics

Principal Component Analysis (PCA)

Objective: Find projection that maximizes variance

Covariance Matrix:

$$C = (1/n) \times X^T X$$

Eigendecomposition:

$$C = V \Lambda V^T$$

- V: eigenvectors (principal components)
- Λ: eigenvalues (diagonal matrix)

Projection:

$$Z = XV_k$$

- V_k: first k eigenvectors

Reconstruction:

$$\hat{X} = ZV_k^T$$

Explained Variance Ratio:

$$EVR = \lambda_i / \Sigma_j \lambda_j$$

Linear Discriminant Analysis (LDA)

Between-class Scatter:

$$S_B = \Sigma_k n_k(\mu_k - \mu)(\mu_k - \mu)^T$$

Within-class Scatter:

$$S_W = \Sigma_k \Sigma_{i \in C_k} (x_i - \mu_k)(x_i - \mu_k)^T$$

Objective:

$$\text{maximize: } (w^T S_B w)/(w^T S_W w)$$

Solution: Eigenvectors of $S_W^{-1} S_B$

t-SNE

Joint Probability (High-dimensional):

$$p_{ij} = (p_{j|i} + p_{i|j})/(2n)$$

where:

$$p_{j|i} = \exp(-||x_i - x_j||^2/(2\sigma_i^2)) \;/\; \Sigma_{k \neq i} \exp(-||x_i - x_k||^2/(2\sigma_i^2))$$

Joint Probability (Low-dimensional):

$$q_{ij} = (1 + ||y_i - y_j||^2)^{-1} \;/\; \Sigma_{k \neq l} (1 + ||y_k - y_l||^2)^{-1}$$

Cost Function (KL divergence):

$$C = KL(P||Q) = \Sigma_i \Sigma_j p_{ij} \log(p_{ij}/q_{ij})$$

Evaluation Metrics Mathematics

Classification Metrics

Confusion Matrix Elements:

- TP: True Positives
- TN: True Negatives
- FP: False Positives
- FN: False Negatives

Accuracy:

$$\text{Accuracy} = (TP + TN)/(TP + TN + FP + FN)$$

Precision:

$$\text{Precision} = TP/(TP + FP)$$

Recall (Sensitivity, True Positive Rate):

$$\text{Recall} = TP/(TP + FN)$$

Specificity (True Negative Rate):

$$\text{Specificity} = TN/(TN + FP)$$

F1 Score:

$$F1 = 2 \times (\text{Precision} \times \text{Recall})/(\text{Precision} + \text{Recall})$$

F-beta Score:

$$F\beta = (1 + \beta^2) \times (\text{Precision} \times \text{Recall})/((\beta^2 \times \text{Precision}) + \text{Recall})$$

Matthews Correlation Coefficient:

$$MCC = (TP\times TN - FP\times FN)/\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}$$

ROC and AUC

ROC Curve: Plot of TPR vs FPR at various thresholds

AUC (Area Under ROC Curve):

$$AUC = \int_0^1 TPR(FPR)\, dFPR$$

Approximation (Trapezoidal Rule):

$$AUC \approx \Sigma_i \tfrac{1}{2}(TPR_i + TPR_{i+1})(FPR_{i+1} - FPR_i)$$

Regression Metrics

R^2 Score (Coefficient of Determination):

$$R^2 = 1 - (SS\_res/SS\_tot)$$

where:

$$SS\_res = \Sigma_i (y_i - \hat{y}_i)^2 \quad \text{(residual sum of squares)}$$
$$SS\_tot = \Sigma_i (y_i - \bar{y})^2 \quad \text{(total sum of squares)}$$

Adjusted R^2:

$$R^2\_adj = 1 - [(1-R^2)(n-1)/(n-p-1)]$$

- p: number of predictors

Mean Absolute Percentage Error (MAPE):

$$MAPE = (100/n) \times \Sigma_i |y_i - \hat{y}_i|/|y_i|$$

Clustering Metrics

Silhouette Score:

$$s(i) = (b(i) - a(i))/\max(a(i), b(i))$$

where:

- a(i): average distance to points in same cluster
- b(i): minimum average distance to points in other clusters

Davies-Bouldin Index:

$$DB = (1/k) \times \Sigma_{i=1}^{k} \max_{j\neq i} [(\sigma_i + \sigma_j)/d(c_i, c_j)]$$

- σ_i: average distance of points in cluster i to centroid
- $d(c_i, c_j)$: distance between centroids

Calinski-Harabasz Index:

$$CH = [tr(B_k)/(k-1)] / [tr(W_k)/(n-k)]$$

- B_k: between-group dispersion matrix

- W_k: within-group dispersion matrix

Additional Important Formulas

Regularization

Elastic Net (L1 + L2):

$$J(\theta) = L(\theta) + \lambda_1||\theta||_1 + \lambda_2||\theta||_2^2$$

Distance Metrics

Euclidean Distance:

$$d(x,y) = \sqrt{(\Sigma_i (x_i - y_i)^2)}$$

Manhattan Distance:

$$d(x,y) = \Sigma_i |x_i - y_i|$$

Cosine Similarity:

$$\cos(x,y) = (x \cdot y)/(||x||_2 \times ||y||_2)$$

Minkowski Distance:

$$d(x,y) = (\Sigma_i |x_i - y_i|^p)^{(1/p)}$$

Bias-Variance Decomposition

Total Error:

$$E[(y - \hat{y})^2] = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

where:

$$\text{Bias} = E[\hat{y}] - y$$
$$\text{Variance} = E[(\hat{y} - E[\hat{y}])^2]$$

Cross-Validation Error

k-Fold CV Error:

$$CV(k) = (1/k) \times \Sigma_{i=1}^{k} L(h_i, D_i)$$

- h_i: model trained on all folds except i
- D_i: validation data from fold i

Time Series Analysis

Autoregressive (AR) Model

$$y_t = c + \Sigma_{i=1}^{p} \phi_i y_{t-i} + \varepsilon_t$$

- φ_i: AR coefficients
- p: order of AR model

Moving Average (MA) Model

$$y_t = \mu + \varepsilon_t + \Sigma_{i=1}^{g} \theta_i \varepsilon_{t-i}$$

- θ_i: MA coefficients
- q: order of MA model

ARIMA Model

$$(1 - \Sigma_{i=1}^{p} \phi_i L^i)(1-L)^d y_t = (1 + \Sigma_{i=1}^{g} \theta_i L^i)\varepsilon_t$$

- L: lag operator
- d: degree of differencing

Exponential Smoothing

Simple: $\alpha y_t + (1-\alpha)\hat{y}_{t-1}$ **Double**: $\alpha y_t + (1-\alpha)(\hat{y}_{t-1} + \hat{b}_{t-1})$ **Triple (Holt-Winters)**: Includes seasonal component

Autocorrelation Function (ACF)

$$\rho_k = Cov(y_t, y_{t-k}) / Var(y_t)$$

Partial Autocorrelation Function (PACF)

Correlation between y_t and y_{t-k} after removing effects of intermediate lags

---

Bayesian Methods

Bayes' Rule (Full Form)

$$P(\theta|D) = P(D|\theta)P(\theta) / P(D)$$

- $P(\theta|D)$: Posterior
- $P(D|\theta)$: Likelihood
- $P(\theta)$: Prior
- $P(D)$: Evidence

Maximum A Posteriori (MAP)

$$\theta\_MAP = \text{argmax}\_\theta\ P(\theta|D) = \text{argmax}\_\theta\ P(D|\theta)P(\theta)$$

Bayesian Linear Regression

Posterior:

$$P(w|D) = N(w|\mu_n, \Sigma_n)$$

where:

$$\Sigma_n = (\Sigma_0^{-1} + \beta X^T X)^{-1}$$
$$\mu_n = \Sigma_n(\Sigma_0^{-1}\mu_0 + \beta X^T y)$$

Variational Inference

ELBO (Evidence Lower Bound):

$$L(q) = Eq[\log P(X,Z)] - Eq[\log q(Z)]$$

Markov Chain Monte Carlo (MCMC)

Metropolis-Hastings Accept Probability:

$$\alpha = \min(1, P(\theta'|D)Q(\theta|\theta') / P(\theta|D)Q(\theta'|\theta))$$

Convolutional Neural Networks

Convolution Operation

$$(f * g)[n] = \Sigma_m f[m] \times g[n-m]$$

2D Convolution (Images)

$$S(i,j) = (I * K)(i,j) = \Sigma_m \Sigma_n I(m,n)K(i-m,j-n)$$

- I: input image
- K: kernel/filter

Pooling

Max Pooling:

$$y = \max(x_1, x_2, \ldots, x_n) \text{ in pooling window}$$

Average Pooling:

$$y = (1/n)\Sigma_i x_i \text{ in pooling window}$$

Output Size Calculation

$$\text{Output Size} = \lfloor (W - F + 2P)/S \rfloor + 1$$

- W: input size
- F: filter size
- P: padding
- S: stride

Number of Parameters

Conv Layer: $(F \times F \times C_{in} + 1) \times C_{out}$ **FC Layer**: $(N_{in} + 1) \times N_{out}$

Recurrent Neural Networks

Vanilla RNN

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$
$$y_t = W_{hy}h_t + b_y$$

LSTM (Long Short-Term Memory)

Forget Gate:

$$f_t = \sigma(Wf \cdot [h_{t-1}, x_t] + bf)$$

Input Gate:

$$i_t = \sigma(Wi \cdot [h_{t-1}, x_t] + bi)$$
$$\tilde{C}_t = \tanh(WC \cdot [h_{t-1}, x_t] + bC)$$

Cell State Update:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Output Gate:

$$o_t = \sigma(Wo \cdot [h_{t-1}, x_t] + bo)$$
$$h_t = o_t * \tanh(C_t)$$

GRU (Gated Recurrent Unit)

Update Gate: $z_t = \sigma(Wz \cdot [h_{t-1}, x_t])$ **Reset Gate**: $r_t = \sigma(Wr \cdot [h_{t-1}, x_t])$ **Hidden State**: $h_t = (1-z_t)h_{t-1} + z_t\tilde{h}_t$

Transformer Architecture

Scaled Dot-Product Attention

$$\text{Attention}(Q,K,V) = \text{softmax}(QK^T/\sqrt{d_k})V$$

- Q: Query matrix
- K: Key matrix
- V: Value matrix
- d_k: dimension of keys

Multi-Head Attention

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,...,\text{head}_h)W^O$$

where:

$$\text{head}_i = \text{Attention}(QW_i^{\psi}, KW_i^K, VW_i^V)$$

Positional Encoding

```
PE(pos,2i) = sin(pos/10000^(2i/d_model))
PE(pos,2i+1) = cos(pos/10000^(2i/d_model))
```

Layer Normalization

```
LN(x) = γ × (x-μ)/√(σ²+ε) + β
```

Recommender Systems

Collaborative Filtering

Matrix Factorization:

```
R ≈ PQᵀ
```

- R: user-item rating matrix
- P: user feature matrix
- Q: item feature matrix

Objective Function:

```
min Σ(i,j)∈K (rᵢⱼ - pᵢᵀqⱼ)² + λ(||pᵢ||² + ||qⱼ||²)
```

Cosine Similarity

```
sim(u,v) = (Σᵢ rᵤᵢrᵥᵢ) / (√Σᵢ rᵤᵢ² × √Σᵢ rᵥᵢ²)
```

Pearson Correlation

```
sim(u,v) = Σᵢ(rᵤᵢ-r̄ᵤ)(rᵥᵢ-r̄ᵥ) / √(Σᵢ(rᵤᵢ-r̄ᵤ)² × Σᵢ(rᵥᵢ-r̄ᵥ)²)
```

Reinforcement Learning Mathematics

Bellman Equation

For Value Function:

```
V(s) = max_a [R(s,a) + γΣₛ' P(s'|s,a)V(s')]
```

For Q-Function:

$$Q(s,a) = R(s,a) + \gamma\Sigma_{s'} P(s'|s,a)\max_{a'} Q(s',a')$$

Policy Gradient Theorem

$$\nabla\theta J(\theta) = E[\nabla\theta \log \pi\theta(a|s) \times Q\pi(s,a)]$$

Temporal Difference Learning

TD(0):

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$$

TD(λ):

$$V(s) \leftarrow V(s) + \alpha\delta_t e_t(s)$$

where eligibility trace:

$$e_t(s) = \gamma\lambda e_{t-1}(s) + 1(s=s_t)$$

Actor-Critic

Critic Update: Update value function V(s) **Actor Update**: $\nabla\theta J = E[\nabla\theta \log \pi\theta(a|s) \times A(s,a)]$ where advantage $A(s,a) = Q(s,a) - V(s)$

Statistical Hypothesis Testing

Z-Test

$$z = (\bar{x} - \mu) / (\sigma/\sqrt{n})$$

T-Test

One Sample:

$$t = (\bar{x} - \mu) / (s/\sqrt{n})$$

Two Sample (Equal Variance):

$$t = (\bar{x}_1 - \bar{x}_2) / (s_p\sqrt{(1/n_1 + 1/n_2)})$$

where pooled variance:

$$s_p^2 = ((n_1-1)s_1^2 + (n_2-1)s_2^2) / (n_1+n_2-2)$$

Chi-Square Test

$$\chi^2 = \Sigma_i (O_i - E_i)^2 / E_i$$

- O_i: observed frequency
- E_i: expected frequency

ANOVA F-Statistic

$$F = MSB / MSW = (SSB/(k-1)) / (SSW/(n-k))$$

- MSB: mean square between groups
- MSW: mean square within groups

p-value

$$\text{p-value} = P(|\text{Test Statistic}| \geq |\text{Observed Value}| \mid H_0)$$

Ensemble Methods Mathematics

Bagging

Bootstrap Sampling: Sample n instances with replacement **Aggregation**: $\hat{y} = (1/B)\Sigma_\beta f_\beta(x)$

AdaBoost

Sample Weight Update:

$$w_i^{(t+1)} = w_i^{(t)} \times \exp(-\alpha_t y_i h_t(x_i))$$

Classifier Weight:

$$\alpha_t = \frac{1}{2} \log((1-\varepsilon_t)/\varepsilon_t)$$

Final Prediction:

$$H(x) = \text{sign}(\Sigma_t \alpha_t h_t(x))$$

Gradient Boosting Mathematics

Pseudo-Residuals:

$$r_{im} = -[\partial L(y_i,f(x_i))/\partial f(x_i)]_{f=f_{m-1}}$$

Line Search:

$$\gamma_m = \text{argmin}_\gamma \Sigma_i \ L(y_i, f_{m-1}(x_i) + \gamma h_m(x_i))$$

XGBoost Objective

$$L = \Sigma_i \ l(y_i,\hat{y}_i) + \Sigma_k \ \Omega(f_k)$$

where:

$$\Omega(f) = \gamma T + ½\lambda||w||^2$$

- T: number of leaves
- w: leaf weights

Graph Neural Networks

Graph Convolution

$$H^{(l+1)} = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)})$$

- $\tilde{A}$ = A + I (adjacency matrix with self-loops)
- $\tilde{D}$: degree matrix of $\tilde{A}$

Message Passing

$$h_i^{(k+1)} = \sigma(W_{self}h_i^{(k)} + \Sigma_j \in N(i) \ W_{neigh}h_j^{(k)})$$

Graph Attention

$$\alpha_{ij} = \text{softmax}_j(\text{LeakyReLU}(a^T[Wh_i||Wh_j]))$$

Advanced Optimization

Newton's Method

$$\theta_{t+1} = \theta_t - H^{-1}\nabla f(\theta_t)$$

- H: Hessian matrix

L-BFGS

Approximates inverse Hessian using limited memory

Conjugate Gradient

$$d_{k+1} = -g_{k+1} + \beta_k d_k$$

where:

$$\beta_k = g_{k+1}^T g_{k+1} / g_k^T g_k \text{ (Fletcher-Reeves)}$$

Natural Gradient

$$\theta_{t+1} = \theta_t - \alpha F^{-1}\nabla L(\theta_t)$$

- F: Fisher Information Matrix

Sampling Methods

Rejection Sampling

Accept x with probability:

```
p(accept) = f(x) / (M×g(x))
```

Importance Sampling

```
E_p[f(x)] = E_q[f(x)p(x)/q(x)]
```

Gibbs Sampling

Sample each variable conditioned on others:

$$x_i^{(t+1)} \sim P(x_i | x_1^{(t+1)},\ldots,x_{i-1}^{(t+1)},x_{i+1}^{(t)},\ldots,x_n^{(t)})$$

Online Learning

Stochastic Gradient Descent

$$\theta_{t+1} = \theta_t - \eta_t \nabla l(x_t, y_t, \theta_t)$$

Online Gradient Descent Regret Bound

$$\text{Regret} \leq ||\theta^*||^2/(2\eta) + \eta \times \Sigma_t ||g_t||^2/2$$

Exponential Weighted Average

$$w_{t+1,i} = w_{t,i} \times \exp(-\eta l_{t,i}) / Z_t$$

Semi-Supervised Learning

Self-Training Loss

$$L = \Sigma_l L(x_l, y_l) + \lambda \Sigma_u L(x_u, \hat{y}_u)$$

Graph-Based SSL

Label Propagation:

$$f_{t+1} = \alpha W f_t + (1-\alpha) y$$

Co-Training

Train two classifiers on different views:

$$h_1 : X_1 \to Y$$
$$h_2 : X_2 \to Y$$

Probabilistic Graphical Models

Hidden Markov Models (HMM)

Forward Algorithm:

$$\alpha_t(j) = [\Sigma_i \alpha_{t-1}(i) a_{ij}] \times b_j(o_t)$$

- a_{ij}: transition probability
- $b_j(o_t)$: emission probability

Backward Algorithm:

$$\beta_t(i) = \Sigma_j\ a_{ij}b_j(o_{t+1})\beta_{t+1}(j)$$

Viterbi Algorithm:

$$\delta_t(j) = \max\_i\ [\delta_{t-1}(i) \times a_{ij}] \times b_j(o_t)$$

Baum-Welch Update:

$$a_{ij} = \Sigma_t\ \xi_t(i,j)\ /\ \Sigma_t\ \gamma_t(i)$$

Conditional Random Fields (CRF)

$$P(y|x) = (1/Z(x)) \times \exp(\Sigma_t\ \Sigma_k\ \lambda_k f_k(y_{t-1},y_t,x,t))$$

- $Z(x)$: partition function
- f_k: feature functions
- λ_k: weights

Belief Propagation

Message Update:

$$m_{i\to j}(x_j) = \Sigma_{x_i}\ \psi_i(x_i)\psi_{ij}(x_i,x_j)\ \Pi_{k\in N(i)\backslash j}\ m_{k\to i}(x_i)$$

More Probability Distributions

Poisson Distribution

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

- Mean = Variance = λ

Exponential Distribution

$$f(x) = \lambda e^{-\lambda x} \text{ for } x \geq 0$$

- Mean = $1/\lambda$
- Variance = $1/\lambda^2$

Gamma Distribution

$$f(x) = (\beta^\alpha/\Gamma(\alpha)) \times x^{\alpha-1}e^{-\beta x}$$

Beta Distribution

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

where $B(\alpha,\beta) = \Gamma(\alpha)\Gamma(\beta)/\Gamma(\alpha+\beta)$

Dirichlet Distribution

$$f(x_1,...,x_k) = (1/B(\alpha)) \times \Pi_i \; x_i^{\alpha_i-1}$$

Student's t-Distribution

$$f(x) = \Gamma((\nu+1)/2) / (\sqrt{(\nu\pi)}\Gamma(\nu/2)) \times (1 + x^2/\nu)^{-(\nu+1)/2}$$

Laplace Distribution

$$f(x) = (1/2b) \times \exp(-|x-\mu|/b)$$

Anomaly Detection

Gaussian Anomaly Detection

Anomaly Score:

$$f(x) = \exp(-\tfrac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))$$

Anomaly if $f(x) < \varepsilon$

Isolation Forest

Path Length:

$$h(x) = E[\text{path length to isolate } x]$$

Anomaly Score:

$$s(x,n) = 2^{-E(h(x))/c(n)}$$

Local Outlier Factor (LOF)

$$LOF(x) = (\Sigma_{y}\in N(x)\ lrd(y)) / (|N(x)| \times lrd(x))$$

- lrd: local reachability density

One-Class SVM

$$\min\ (1/2)||w||^2 + (1/\nu n)\Sigma_i\ \xi_i - \rho$$

subject to: $w\cdot\varphi(x_i) \geq \rho - \xi_i$

Mahalanobis Distance

$$D(x) = \sqrt{((x-\mu)^T S^{-1}(x-\mu))}$$

Active Learning

Uncertainty Sampling

Least Confident:

$$x^* = argmax\_x\ [1 - P(\hat{y}|x)]$$

Margin Sampling:

$$x^* = argmin\_x\ [P(\hat{y}_1|x) - P(\hat{y}_2|x)]$$

Entropy-Based:

$$x^* = argmax\_x\ [-\Sigma_i\ P(y_i|x)\log P(y_i|x)]$$

Query by Committee

Vote Entropy:

$$x^* = argmax\_x\ [-\Sigma_i\ (V(y_i)/C)\log(V(y_i)/C)]$$

- $V(y_i)$: votes for class i
- C: committee size

Expected Model Change

$$x^* = \text{argmax}_x \, ||\nabla_\theta L(\theta;x,y)||$$

Multi-Task Learning

Hard Parameter Sharing Loss

$$L = \Sigma_{t=1}^{T} \alpha_t L_t(f_t(x; \theta_{shared}, \theta_t), y^{(t)})$$

Multi-Task Gaussian Process

$$f \sim GP(0, K \otimes K_t)$$

- K: covariance between inputs
- K_t: covariance between tasks

Task Uncertainty Weighting

$$L = \Sigma_t (1/2\sigma_t^2)L_t + \log \sigma_t$$

Meta-Learning

Model-Agnostic Meta-Learning (MAML)

Inner Loop:

$$\theta'_i = \theta - \alpha\nabla_\theta L_i(f\theta)$$

Outer Loop:

$$\theta = \theta - \beta\nabla_\theta \Sigma_i L_i(f\theta'_i)$$

Prototypical Networks

Prototype:

$$c_k = (1/|S_k|) \times \Sigma_{(x_i,y_i)\in S_k} f\phi(x_i)$$

Prediction:

$$P(y=k|x) = \exp(-d(f\phi(x),c_k)) / \Sigma_{k'} \exp(-d(f\phi(x),c_{k'}))$$

Reptile Algorithm

$$\theta = \theta + \varepsilon(\tilde{\theta} - \theta)$$

where $\tilde{\theta}$ is obtained after k steps of SGD

Federated Learning

FedAvg Algorithm

Local Update:

$$w_k^{t+1} = w_k^t - \eta\nabla F_k(w_k^t)$$

Global Aggregation:

$$w^{t+1} = \Sigma_{k=1}^K (n_k/n)w_k^{t+1}$$

Differential Privacy in FL

$$\tilde{w} = w + N(0, \sigma^2S^2I)$$

- S: sensitivity
- σ: noise scale

Fairness Metrics

Demographic Parity

$$P(\hat{Y}=1|A=0) = P(\hat{Y}=1|A=1)$$

Equalized Odds

$$P(\hat{Y}=1|A=0,Y=y) = P(\hat{Y}=1|A=1,Y=y) \text{ for } y\in\{0,1\}$$

Equal Opportunity

$$P(\hat{Y}=1|A=0,Y=1) = P(\hat{Y}=1|A=1,Y=1)$$

Disparate Impact

$$DI = P(\hat{Y}=1|A=0) / P(\hat{Y}=1|A=1)$$

Fair if DI ≥ 0.8

Individual Fairness

$$d(f(x_1), f(x_2)) \leq Ld(x_1, x_2)$$

Causal Inference

Average Treatment Effect (ATE)

$$\tau = E[Y(1) - Y(0)]$$

Propensity Score

$$e(x) = P(T=1|X=x)$$

Inverse Probability Weighting

$$\hat{\tau}_{IPW} = (1/n)\Sigma_i [T_iY_i/e(X_i) - (1-T_i)Y_i/(1-e(X_i))]$$

Doubly Robust Estimator

$$\hat{\tau}_{DR} = (1/n)\Sigma_i [\mu_1(X_i) - \mu_0(X_i) + T_i(Y_i-\mu_1(X_i))/e(X_i) - (1-T_i)(Y_i-\mu_0(X_i))/(1-e(X_i))]$$

Instrumental Variables

$$\hat{\beta}_{IV} = Cov(Y,Z) / Cov(X,Z)$$

Additional Important Concepts

Rademacher Complexity

$$\hat{R}_n(F) = E_\sigma[\sup_{f\in F} (1/n)\Sigma_i \sigma_i f(x_i)]$$

- σ_i: Rademacher random variables

VC Dimension

For hypothesis class H:

- VCD(H) = largest set size that can be shattered

PAC Learning Bound

```
P(|R(h) - R̂(h)| > ε) ≤ 2exp(-2nε²)
```

Margin Theory (SVM)

Generalization Bound:

```
R(f) ≤ R̂_γ(f) + O(√(d/γ²n))
```

Spectral Clustering

Graph Laplacian:

```
L = D - W
```

Normalized Laplacian:

```
L_norm = I - D⁻¹/²WD⁻¹/²
```

Gaussian Processes

Prior:

```
f(x) ~ GP(m(x), k(x,x'))
```

Posterior:

```
f*|X,y,X* ~ N(μ*, Σ*)
```

where:

```
μ* = K*ᵀ(K+σ²I)⁻¹y
Σ* = K** - K*ᵀ(K+σ²I)⁻¹K*
```

Variational Autoencoders (VAE)

ELBO:

```
L = Eq(z|x)[log p(x|z)] - KL(q(z|x)||p(z))
```

Generative Adversarial Networks (GAN)

Objective:

```
min_G max_D V(D,G) = Ex~pdata[log D(x)] + Ez~pz[log(1-D(G(z)))]
```

Wasserstein Distance

```
W(P,Q) = inf_{γ∈Π(P,Q)} E(x,y)~γ[||x-y||]
```