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 - μ) / σ
```

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

Linear Algebra in ML

Vector Operations

Dot Product (Inner Product):

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

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

```
∇f = [∂f/∂x₁, ∂f/∂x₂, ..., ∂f/∂xₙ]ᵀ
```

Gradient Descent Update Rule

```
θₜ₊₁ = θₜ - α∇f(θₜ)
```

- 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:

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

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):

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

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:

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

Soft Margin SVM

Objective:

```
minimize: ½||w||² + C × Σᵢ ξᵢ
subject to: yᵢ(wᵀxᵢ + b) ≥ 1 - ξᵢ
            ξᵢ ≥ 0
```

- ξ_i: slack variables
- C: regularization parameter

Kernel Trick

Kernel Function:

```
K(x, x') = ϕ(x)ᵀϕ(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) = Σᵢ₌₁ⁿ αᵢyᵢK(xᵢ, 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):

```
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:

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

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(θ) = L(θ) + λ₁||θ||₁ + λ₂||θ||₂²
```

Distance Metrics

Euclidean Distance:

```
d(x,y) = √(Σᵢ (xᵢ - yᵢ)²)
```

Manhattan Distance:

```
d(x,y) = Σᵢ |xᵢ - yᵢ|
```

Cosine Similarity:

```
cos(x,y) = (x·y)/(||x||₂ × ||y||₂)
```

Minkowski Distance:

```
d(x,y) = (Σᵢ |xᵢ - yᵢ|ᵖ)^(1/p)
```

Bias-Variance Decomposition

Total Error:

```
E[(y - ŷ)²] = Bias² + Variance + Irreducible Error
```

where:

```
Bias = E[ŷ] - y
Variance = E[(ŷ - E[ŷ])²]
```

Cross-Validation Error

k-Fold CV Error:

```
CV(k) = (1/k) × Σᵢ₌₁ᵏ L(hᵢ, Dᵢ)
```

- 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:

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

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

$$LN(x) = \gamma \times (x-\mu)/\sqrt{(\sigma^2+\varepsilon)} + \beta$$

Recommender Systems

Collaborative Filtering

Matrix Factorization:

$$R \approx PQ^T$$

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

Objective Function:

$$\min \Sigma(i,j)\in K \ (r_{ij} - p_i^T q_j)^2 + \lambda(||p_i||^2 + ||q_j||^2)$$

Cosine Similarity

$$sim(u,v) = (\Sigma_i \ r_{ui} r_{vi}) \ / \ (\sqrt{\Sigma_i \ r_{ui}^2} \times \sqrt{\Sigma_i \ r_{vi}^2})$$

Pearson Correlation

$$sim(u,v) = \Sigma_i(r_{ui}-\bar{r}_u)(r_{vi}-\bar{r}_v) \ / \ \sqrt{(\Sigma_i(r_{ui}-\bar{r}_u)^2 \times \Sigma_i(r_{vi}-\bar{r}_v)^2)}$$

Reinforcement Learning Mathematics

Bellman Equation

For Value Function:

$$V(s) = \max\_a \ [R(s,a) + \gamma\Sigma_{s'} \ P(s'|s,a)V(s')]$$

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:

$$\alpha_t = ½ \log((1-\varepsilon_t)/\varepsilon_t)$$

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:

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

XGBoost Objective

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

where:

$$\Omega(f) = \gamma T + \tfrac{1}{2}\lambda||w||^2$$

- 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

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

Graph Attention

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

Advanced Optimization

Newton's Method

```
θₜ₊₁ = θₜ - H⁻¹∇f(θₜ)
```

- H: Hessian matrix

L-BFGS

Approximates inverse Hessian using limited memory

Conjugate Gradient

```
dₖ₊₁ = -gₖ₊₁ + βₖdₖ
```

where:

```
βₖ = gₖ₊₁ᵀgₖ₊₁ / gₖᵀgₖ (Fletcher-Reeves)
```

Natural Gradient

```
θₜ₊₁ = θₜ - αF⁻¹∇L(θₜ)
```

- 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:

```
xᵢ⁽ᵗ⁺¹⁾ ~ P(xᵢ|x₁⁽ᵗ⁺¹⁾,...,xᵢ₋₁⁽ᵗ⁺¹⁾,xᵢ₊₁⁽ᵗ⁾,...,xₙ⁽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:

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

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

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

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:

$$f(x) = \exp(-½(x-\mu)^T\Sigma^{-1}(x-\mu))$$

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) = (Σᵧ∈N(x) lrd(y)) / (|N(x)| × lrd(x))
```

- lrd: local reachability density

One-Class SVM

```
min (1/2)||w||² + (1/νn)Σᵢ ξᵢ - ρ
```

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

Mahalanobis Distance

```
D(x) = √((x-μ)ᵀS⁻¹(x-μ))
```

Active Learning

Uncertainty Sampling

Least Confident:

```
x* = argmax_x [1 - P(ŷ|x)]
```

Margin Sampling:

```
x* = argmin_x [P(ŷ₁|x) - P(ŷ₂|x)]
```

Entropy-Based:

```
x* = argmax_x [-Σᵢ P(yᵢ|x)log P(yᵢ|x)]
```

Query by Committee

Vote Entropy:

```
x* = argmax_x [-Σᵢ (V(yᵢ)/C)log(V(yᵢ)/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

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

- 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:

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p_z}[\log(1-D(G(z)))]$$

Wasserstein Distance

$$W(P,Q) = \inf_{\gamma\in\Pi(P,Q)} E_{(x,y)\sim\gamma}[||x-y||]$$

Modern Deep Learning Methods

Diffusion Models

Forward Process (Adding Noise):

$$q(x_t|x_{t-1}) = N(x_t; \sqrt{(1-\beta_t)}x_{t-1}, \beta_t I)$$
$$q(x_t|x_0) = N(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$$

where $\bar{\alpha}_t = \Pi_{s=1}^t \alpha_s$ and $\alpha_t = 1 - \beta_t$

Reverse Process (Denoising):

$$p\theta(x_{t-1}|x_t) = N(x_{t-1}; \mu\theta(x_t,t), \Sigma\theta(x_t,t))$$

Training Objective (Simplified):

$$L = E_{t,x_0,\varepsilon}[||\varepsilon - \varepsilon\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{(1-\bar{\alpha}_t)}\varepsilon, t)||^2]$$

Score Matching:

$$J(\theta) = ½\, E_{x,t}[||s\theta(x,t) - \nabla_x \log p_t(x)||^2]$$

DDPM Sampling:

$$x_{t-1} = 1/\sqrt{\alpha_t}(x_t - (1-\alpha_t)/\sqrt{(1-\bar{\alpha}_t)}\varepsilon\theta(x_t,t)) + \sigma_t z$$

Normalizing Flows

Change of Variables:

$$\log p(x) = \log p(z) - \log|\det(\partial f/\partial x)|$$

Flow Transformation:

$$x = f(z) \text{ where } z \sim p(z)$$

Planar Flow:

$$f(z) = z + uh(w^T z + b)$$

Real NVP Coupling Layer:

$$y_1:d = x_1:d$$
$$yd+1:D = xd+1:D \odot \exp(s(x_1:d)) + t(x_1:d)$$

Glow Objective:

$$\log p(x) = \log p(z) + \Sigma_i \log|\det(J_i)|$$

Contrastive Learning

InfoNCE Loss:

$$L = -\log[\exp(\text{sim}(z_i,z_j)/\tau) / \Sigma_{k=1}^{2N} 1[k\neq i]\exp(\text{sim}(z_i,z_k)/\tau)]$$

- $\text{sim}(u,v) = u^T v/(||u||\cdot||v||)$
- τ: temperature parameter

SimCLR Objective:

$$l(i,j) = -\log[\exp(\text{sim}(z_i,z_j)/\tau) / \Sigma_{k=1}^{2N} 1[k\neq i]\exp(\text{sim}(z_i,z_k)/\tau)]$$

MoCo (Momentum Contrast):

$$\theta_k \leftarrow m\theta_k + (1-m)\theta q$$

- m: momentum coefficient (e.g., 0.999)

CLIP Loss:

$$L = -½\Sigma_i N[\log(\exp(\text{sim}(I_i,T_i)/\tau)/\Sigma_j\exp(\text{sim}(I_i,T_j)/\tau)) + \log(\exp(\text{sim}(I_i,T_i)/\tau)/\Sigma_j\exp(\text{sim}(I_j,T_i)/\tau))]$$

Vision Transformers (ViT)

Patch Embedding:

$$z_0 = [x_{class}; x^1 pE; x^2 pE; ...; x^N pE] + E_{pos}$$

- x^p: flattened patch
- E: embedding projection
- Epos: position embeddings

Patch Size Calculation:

$$N = HW/P^2 \text{ patches}$$

- H,W: image height, width
- P: patch size

Neural ODEs

Continuous Dynamics:

$$dh(t)/dt = f(h(t), t, \theta)$$

Forward Pass:

$$h(t_1) = h(t_0) + \int_{t_0}^{t_1} f(h(t), t, \theta)dt$$

Adjoint Sensitivity Method:

$$da(t)/dt = -a(t)^T \partial f/\partial h$$
$$dL/d\theta = -\int_{t_1}^{t_0} a(t)^T \partial f/\partial \theta \, dt$$

Advanced Reinforcement Learning

Multi-Armed Bandits

Upper Confidence Bound (UCB):

$$A_t = \text{argmax}_a[\hat{Q}_t(a) + c\sqrt{(\ln t/N_t(a))}]$$

- $N_t(a)$: times action a selected
- c: exploration constant

Thompson Sampling:

$$\theta_a \sim \text{Beta}(\alpha_a, \beta_a)$$
$$A_t = \text{argmax}_a\ \theta_a$$

ε-Greedy:

$$A_t = \begin{cases} \text{argmax}_a\ \hat{Q}_t(a) & \text{with prob } 1-\varepsilon \\ \text{random action} & \text{with prob } \varepsilon \end{cases}$$

Exp3 (Exponential-weight for Exploration):

$$p_t(a) = (1-\gamma)w_t(a)/W_t + \gamma/K$$

Soft Actor-Critic (SAC)

Soft Value Function:

$$V(s) = E_a{\sim}\pi[Q(s,a) - \alpha \log \pi(a|s)]$$

Soft Q-Function:

$$Q(s,a) = r + \gamma E_s'[V(s')]$$

Policy Objective:

$$J(\pi) = E_s{\sim}D,\varepsilon{\sim}N[\alpha \log \pi(f(\varepsilon;s)|s) - Q(s,f(\varepsilon;s))]$$

Entropy Temperature Update:

$$\alpha \leftarrow \text{argmin}_a\ E_a{\sim}\pi[-\alpha \log \pi(a|s) - \alpha\bar{H}]$$

Trust Region Policy Optimization (TRPO)

Surrogate Objective:

$$L(\theta) = E_{s,a}[\pi\theta(a|s)/\pi\theta_{old}(a|s) \times A^\pi(s,a)]$$

KL Constraint:

$$E_s \sim \rho\pi[KL(\pi\theta_{old}(\cdot|s)||\pi\theta(\cdot|s))] \leq \delta$$

Natural Policy Gradient:

$$\theta_{k+1} = \theta_k + \sqrt{(2\delta/g^T F g)} \times F^{-1}g$$

- F: Fisher information matrix
- g: policy gradient

Proximal Policy Optimization (PPO)

Clipped Objective:

$$L(\theta) = E[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)A_t)]$$

where $r_t(\theta) = \pi\theta(a_t|s_t)/\pi\theta_{old}(a_t|s_t)$

Specialized Machine Learning Areas

Survival Analysis

Hazard Function:

$$h(t) = \lim(\Delta t \to 0)\ P(t \leq T < t+\Delta t \mid T \geq t)/\Delta t$$

Survival Function:

$$S(t) = P(T > t) = \exp(-\int_0^t h(u)du)$$

Cox Proportional Hazards:

$$h(t|x) = h_0(t)\exp(\beta^T x)$$

Partial Likelihood:

$$L(\beta) = \Pi_{i=1}^n [\exp(\beta^T x_i) / \Sigma_j \in R(t_i) \exp(\beta^T x_j)]^{\wedge}\delta_i$$

Kaplan-Meier Estimator:

$$\hat{S}(t) = \Pi_i : t_i \leq t\ [(n_i - d_i)/n_i]$$

Learning to Rank

RankNet Loss:

$$L = -\Sigma_{ij}\ \bar{P}_{ij}\log P_{ij} + (1-\bar{P}_{ij})\log(1-P_{ij})$$

where $P_{ij} = 1/(1 + \exp(-(s_i - s_j)))$

LambdaRank Gradient:

$$\lambda_{ij} = -\sigma/(1 + \exp(\sigma(s_i - s_j))) \times |\Delta NDCG_{ij}|$$

ListNet Loss:

$$L = -\Sigma_i\ P(\pi_i|y)\log P(\pi_i|s)$$

NDCG (Normalized Discounted Cumulative Gain):

$$NDCG@k = DCG@k / IDCG@k$$
$$DCG@k = \Sigma_{i=1}^{k}\ (2^{rel_i} - 1)/\log_2(i + 1)$$

Calibration Methods

Platt Scaling:

$$P(y=1|f(x)) = 1/(1 + \exp(Af(x) + B))$$

Temperature Scaling:

$$\hat{q}_i = \mathrm{softmax}(z_i/T)$$

Isotonic Regression: Fit monotonic function $m: \mathbb{R} \to [0,1]$

$$\min\ \Sigma_i\ w_i(y_i - m(f_i))^2 \text{ s.t. } m \text{ monotonic}$$

Expected Calibration Error (ECE):

$$ECE = \Sigma_{m=1}^{M}\ |B_m|/n\ |acc(B_m) - conf(B_m)|$$

Domain Adaptation

Domain Adversarial Loss:

$$L = L_\gamma(\theta f, \theta y) - \lambda Ld(\theta f, \theta d)$$

- L_γ: task loss
- Ld: domain classifier loss

Maximum Mean Discrepancy (MMD):

$$MMD^2(X,Y) = ||\mathbb{E}[\phi(x)] - \mathbb{E}[\phi(y)]||^2_{\mathcal{H}}$$

CORAL Loss:

$$L\_CORAL = (1/4d^2)||Cs - Ct||^2F$$

- Cs, Ct: source/target covariance matrices

Wasserstein Distance (for DA):

$$W(Ps,Pt) = \inf_{\gamma\in\Pi} \mathbb{E}(xs,xt)\sim\gamma[||xs - xt||]$$

Optimal Transport

Kantorovich Problem:

$$W(\mu,v) = \min_{\pi\in\Pi(\mu,v)} \int c(x,y)d\pi(x,y)$$

Sinkhorn Algorithm:

$$u^{i+1} = a \oslash (Kv^i)$$
$$v^{i+1} = b \oslash (K^T u^{i+1})$$

where K = exp(-C/ε)

Wasserstein Barycenter:

$$\min_{\mu} \Sigma_k \lambda_k W_2^2(\mu, \mu_k)$$

Gromov-Wasserstein Distance:

$$GW(C_1,C_2) = \min_{\pi} \Sigma_{ijkl} |C_1(i,k) - C_2(j,l)|^2\pi_{ij}\pi_{kl}$$

Advanced Theoretical Concepts

Conformal Prediction

Prediction Set:

$$C(x) = \{y : s(x,y) \geq \hat{q}\}$$

where $\hat{q}$ is (1-α) quantile of scores

Coverage Guarantee:

$$P(Y \in C(X)) \geq 1 - \alpha$$

Conformity Score:

$$s(x,y) = -|y - f(x)| \quad \text{(regression)}$$
$$s(x,y) = f(x)[y] \quad \text{(classification)}$$

Advanced Differential Privacy

Gaussian Mechanism:

$$M(x) = f(x) + N(0, \sigma^2 S^2 f)$$

where $Sf = \max_{x,x'} ||f(x) - f(x')||_2$

Exponential Mechanism:

$$P(M(x) = r) \propto \exp(\varepsilon u(x,r)/(2\Delta u))$$

Rényi Differential Privacy:

$$D\alpha(P||Q) = (1/(\alpha-1))\log \mathbb{E}q[(p(X)/q(X))^\alpha]$$

Moments Accountant:

$$\alpha M(\lambda) = \max\_aux \log \mathbb{E}[\exp(\lambda \cdot priv\_loss(aux))]$$

Advanced Kernel Methods

String Kernel:

$$K(s,t) = \Sigma_u \Sigma_{i:u=s[i]} \Sigma_{j:u=t[j]} \lambda^{(|i|+|j|)}$$

Graph Kernel (Random Walk):

$$K(G_1,G_2) = \Sigma_{ij} [I - \lambda W\times]^{-1}{}_{ij}$$

Fisher Kernel:

$$K(x,x') = \nabla\theta \log p(x|\theta)^T F^{-1} \nabla\theta \log p(x'|\theta)$$

Polynomial Kernel with Offset:

$$K(x,x') = (\alpha x^T x' + c)^\wedge d$$

Capsule Networks

Squashing Function:

$$v_j = ||s_j||^2/(1 + ||s_j||^2) \times s_j/||s_j||$$

Routing Algorithm:

$$c_{ij} = \exp(b_{ij}) / \Sigma_k \exp(b_{ik})$$
$$s_j = \Sigma_i \, c_{ij}\hat{u}_j|_i$$

Agreement Update:

$$b_{ij} \leftarrow b_{ij} + \hat{u}_j|_i \cdot v_j$$

Self-Supervised Learning

Masked Autoencoders (MAE)

Reconstruction Loss:

$$L = (1/|M|) \, \Sigma_{i \in M} \, ||x_i - \hat{x}_i||^2$$

- M: set of masked patches

Masking Strategy: Random masking with ratio r (typically 75%)

BERT Objectives

Masked Language Model (MLM):

$$L\_MLM = -\mathbb{E}[\Sigma_{i} \in M \log P(x_i | x_{(-M)})]$$

Next Sentence Prediction (NSP):

$$L\_NSP = -\mathbb{E}[\log P(y | [CLS], A, B)]$$

BERT Total Loss:

$$L = L\_MLM + L\_NSP$$

GPT Objectives

Causal Language Modeling:

$$L = -\Sigma_t \log P(x_t | x_1, ..., x_{t-1})$$

Momentum Contrast (MoCo)

Queue Update:

```
enqueue(queue, kq)
dequeue(queue)
```

Momentum Update:

$$\theta_k \leftarrow m\theta_k + (1-m)\theta q$$

InfoNCE Loss with Queue:

$$L = -\log[\exp(q \cdot k_+/\tau) / (\exp(q \cdot k_+/\tau) + \Sigma_{k-} \exp(q \cdot k_-/\tau))]$$

BYOL (Bootstrap Your Own Latent)

Loss (No Negatives):

$$L = ||\bar{q} - z'||^2_2$$

where:

- $\bar{q} = q\theta(z\theta)$ (prediction)
- $z' = z\xi$ (target)

Target Network Update:

$$\xi \leftarrow \tau\xi + (1-\tau)\theta$$

SwAV (Swapping Assignments between Views)

Clustering Assignment:

$$Q = \text{Sinkhorn}(Z/\varepsilon)$$

Swapped Prediction Loss:

$$L = -\Sigma_i[q_i^{(1)}\log p_i^{(2)} + q_i^{(2)}\log p_i^{(1)}]$$

Additional Modern Architectures

EfficientNet Scaling

Compound Scaling:

$$\begin{aligned}&\text{depth: } d = \alpha^\phi\\&\text{width: } w = \beta^\phi\\&\text{resolution: } r = \gamma^\phi\end{aligned}$$

where $\alpha\cdot\beta^2\cdot\gamma^2 \approx 2$

Squeeze-and-Excitation

Squeeze:

$$z_c = (1/HW)\ \Sigma_i\Sigma_j\ u_c(i,j)$$

Excitation:

$$s = \sigma(W_2\delta(W_1 z))$$

Scale:

$$\tilde{x}_c = s_c \cdot u_c$$

Feature Pyramid Networks

Top-down Pathway:

$$P_i = \mathrm{Conv}(P_{i+1}\uparrow + C_i)$$

Focal Loss

$$FL(p_t) = -\alpha_t(1-p_t)^\gamma \log(p_t)$$

where:

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{otherwise} \end{cases}$$

IoU Loss

Standard IoU:

$$IoU = |A \cap B| \;/\; |A \cup B|$$

GIoU (Generalized):

$$GIoU = IoU - |C \setminus (A \cup B)| \;/\; |C|$$

where C is smallest enclosing box

DIoU (Distance):

$$DIoU = IoU - \rho^2(b, b^{gt}) \;/\; c^2$$

CIoU (Complete):

$$CIoU = IoU - (\rho^2(b, b^{gt})/c^2 + \alpha v)$$

where v measures aspect ratio consistency