Complete Machine Learning

Miko-pedia AI

May 4, 2026

Contents

Overview. This complete note surveys the foundations and practice of machine learning at the undergraduate level. It moves from the conceptual framing of learning problems (data, loss, generalization) through mathematical foundations, classical methods, deep learning, generative

models, and reinforcement learning, concluding with practical deployment and ethics. Each section contains definitions, algorithms, key equations, and review guidance. The note is designed as a comprehensive review resource: study it holistically or jump to individual sections for targeted revision.

1 The Questions of ML: Data, Loss, Generalization, and Evaluation

1.1 Core ideas

Machine learning (ML) is the study of algorithms that improve their performance on some task T with experience E, measured by a performance metric P. The central questions are:

1. **Data**: How do we represent, collect, and preprocess examples? Each example is a pair $(\mathbf{x}, y)$ where $\mathbf{x} \in \mathcal{X}$ is the input (feature vector) and $y \in \mathcal{Y}$ is the target. For supervised learning, $\mathcal{Y}$ is a label space (e.g., $\mathbb{R}$ for regression, $\{1, \ldots, K\}$ for classification). For unsupervised learning, y is absent. For reinforcement learning, data arrives as sequential interactions $(\mathbf{s}, a, r, \mathbf{s}')$.
2. **Loss**: A loss function $\ell(\hat{y}, y)$ quantifies the cost of predicting $\hat{y}$ when the true value is y. The expected risk is $R(f) = \mathbb{E}_{(\mathbf{x},y)\sim P}[\ell(f(\mathbf{x}), y)]$, where P is the true data distribution. The empirical risk $\hat{R}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(\mathbf{x}^{(i)}), y^{(i)})$ approximates it from n training samples.
3. **Generalization**: The gap $R(f) - \hat{R}(f)$ is the generalization error. A model that memorizes noise achieves zero empirical risk but high expected risk (overfitting). A model that is too simple may have high bias and underfit.
4. **Evaluation**: We estimate generalization by holding out a test set. Metrics depend on the task: accuracy, precision/recall, F1 for classification; mean squared error (MSE), mean absolute error (MAE), R^2 for regression; AUC-ROC for ranking.

Learning paradigms:

- **Supervised**: learn $f : \mathcal{X} \to \mathcal{Y}$ from labeled pairs $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$.
- **Unsupervised**: discover structure in $\{\mathbf{x}^{(i)}\}_{i=1}^{n}$ (clustering, density estimation, dimensionality reduction).
- **Self-supervised**: construct labels from the data itself (e.g., masked language modeling, contrastive learning).
- **Reinforcement learning**: learn a policy $\pi(\mathbf{a}|\mathbf{s})$ maximizing cumulative reward.

For review, be able to: define the three components of a learning problem (data, model, loss); explain the bias–variance decomposition; distinguish training error from test error; describe cross-validation; and identify whether a problem is supervised, unsupervised, or reinforcement learning.

Section summary ML is about using data to select a hypothesis that minimizes expected loss. The fundamental tension is between fitting the training data (empirical risk minimization) and generalizing to unseen data (controlling model capacity via regularization). All subsequent sections elaborate on specific model families, loss functions, optimization algorithms, and evaluation strategies.

2 Mathematical Foundations: Linear Algebra, Calculus, Probability, and Optimization

2.1 Core ideas

Linear algebra provides the language for representing data and models.

- A vector $\mathbf{x} \in \mathbb{R}^d$ is an ordered d-tuple; the inner product $\mathbf{w}^\top \mathbf{x} = \sum_{j=1}^d w_j x_j$ measures similarity.
- A matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ represents n data points each with d features.
- The transpose $(\mathbf{X}^\top)_{ij} = \mathbf{X}_{ji}$; the identity $\mathbf{I}_d$; the inverse $\mathbf{A}^{-1}$ satisfies $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$.
- Eigendecomposition: $\mathbf{A}\mathbf{v} = \lambda \mathbf{v}$ for square $\mathbf{A}$. Symmetric positive-definite matrices have orthogonal eigenvectors and positive eigenvalues.
- Singular value decomposition (SVD): $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ for any $\mathbf{A} \in \mathbb{R}^{m \times n}$, where $\mathbf{U}, \mathbf{V}$ are orthogonal and $\mathbf{\Sigma}$ is diagonal with singular values $\sigma_1 \geq \cdots \geq \sigma_{\min(m,n)} \geq 0$.
- Norms: L^2 norm $\|\mathbf{x}\|_2 = \sqrt{\sum_j x_j^2}$; L^1 norm $\|\mathbf{x}\|_1 = \sum_j |x_j|$; Frobenius norm $\|\mathbf{A}\|_F = \sqrt{\sum_{ij} A_{ij}^2}$.

Calculus enables optimization.

- The gradient $\nabla_{\mathbf{w}} f(\mathbf{w})$ is the vector of partial derivatives; it points in the direction of steepest ascent.
- The chain rule: if $z = g(f(\mathbf{w}))$, then $\nabla_{\mathbf{w}} z = \frac{dg}{df} \cdot \nabla_{\mathbf{w}} f$.
- Hessian $\mathbf{H}_{ij} = \frac{\partial^2 f}{\partial w_i \partial w_j}$ encodes curvature; positive-definite Hessian $\implies$ local minimum.
- Taylor expansion: $f(\mathbf{w} + \boldsymbol{\delta}) \approx f(\mathbf{w}) + \nabla f(\mathbf{w})^\top \boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^\top \mathbf{H} \boldsymbol{\delta}$.

Probability quantifies uncertainty.

- Random variable X with distribution $P(X)$; expectation $\mathbb{E}[X] = \sum_x xP(x)$ (discrete) or $\int xp(x)\,dx$ (continuous).
- Conditional probability $P(Y|X) = P(X,Y)/P(X)$; Bayes' rule: $P(Y|X) = P(X|Y)P(Y)/P(X)$.
- Variance $\mathrm{Var}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2]$; covariance $\mathrm{Cov}(X,Y) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]$.
- Common distributions: Bernoulli(μ), Binomial(n, μ), Gaussian $\mathcal{N}(\mu, \sigma^2)$, Multinomial, Exponential.
- Law of large numbers: sample mean converges to expectation. Central limit theorem: sum of i.i.d. variables is approximately Gaussian.

Optimization finds parameters minimizing a loss.

- Convex functions: $f(\lambda \mathbf{w}_1 + (1-\lambda)\mathbf{w}_2) \leq \lambda f(\mathbf{w}_1) + (1-\lambda) f(\mathbf{w}_2)$. Convex problems have no local minima other than the global minimum.
- Gradient descent: $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla f(\mathbf{w}_t)$ where η is the learning rate.
- Stochastic gradient descent (SGD): uses a minibatch estimate of the gradient for efficiency.
- First-order methods (SGD, momentum, Adam) scale to high dimensions; second-order methods (Newton) use curvature but are costly.

For review, be able to: compute gradients via the chain rule; apply Bayes' rule; derive the normal equations; explain SVD; implement gradient descent; and recognize convex functions.

Section summary ML rests on linear algebra (representing data and models), calculus (optimizing via gradients), probability (modeling noise and uncertainty), and optimization (finding parameters). The normal equations solve linear regression analytically; for most models we use iterative gradient-based methods.

3 Linear Models: Regression, Classification, and Regularization

3.1 Core ideas

Linear regression models the target as $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$. With squared error loss, the empirical risk is:

$$\mathcal{L}(\mathbf{w}, b) = \frac{1}{2n} \sum_{i=1}^{n} (\mathbf{w}^\top \mathbf{x}^{(i)} + b - y^{(i)})^2.$$

The solution (appending 1 to $\mathbf{x}$) is $\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}$, requiring $\mathbf{X}$ to have full column rank. If $d > n$ or features are collinear, $\mathbf{X}^\top \mathbf{X}$ is singular; regularization is needed.

Logistic regression (binary classification) models $P(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b)$ where $\sigma(z) = 1/(1 + e^{-z})$ is the sigmoid function. Loss: binary cross-entropy

$$\mathcal{L}(\mathbf{w}) = -\frac{1}{n}\sum_{i=1}^{n}\big[y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})\big].$$

No closed-form solution; optimize via gradient descent (concave in $\mathbf{w}$ for this loss).

Softmax regression (multi-class) outputs $\hat{y}_k = \exp(\mathbf{w}_k^\top \mathbf{x})/\sum_{j=1}^{K} \exp(\mathbf{w}_j^\top \mathbf{x})$ for $k = 1, \ldots, K$. Loss: categorical cross-entropy

$$\mathcal{L}(\mathbf{W}) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} y_k^{(i)} \log \hat{y}_k^{(i)}.$$

Regularization prevents overfitting by penalizing large weights:

- L^2 (ridge): $\Omega(\mathbf{w}) = \|\mathbf{w}\|_2^2$, solution $\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1}\mathbf{X}^\top \mathbf{y}$. Shrinks weights proportionally.
- L^1 (lasso): $\Omega(\mathbf{w}) = \|\mathbf{w}\|_1$, induces sparsity (some weights become exactly zero). Use subgradient methods.
- Elastic net: $\Omega(\mathbf{w}) = \alpha\|\mathbf{w}\|_1 + (1 - \alpha)\|\mathbf{w}\|_2^2/2$.

Assumptions: linear regression assumes $\mathbb{E}[y|\mathbf{x}]$ is linear, errors are i.i.d. with zero mean and constant variance, and no perfect multicollinearity. Violations lead to biased estimates or inflated variance.

For review, be able to: derive the normal equations; explain the log-odds interpretation of logistic regression; compare L^1 vs L^2 regularization; implement SGD for logistic regression; and diagnose multicollinearity.

Section summary Linear models are the simplest parametric models: regression via least squares, classification via logistic/softmax with cross-entropy. Regularization controls overfitting by penalizing weight magnitudes. These models are convex and well-understood; they serve as baselines and building blocks for deep learning.

4 Evaluation: Cross-Validation, Overfitting, and Bias-Variance Tradeoff

4.1 Core ideas

Overfitting occurs when a model learns noise in the training data, achieving low training error but high test error. It happens with:

- Too many parameters relative to sample size ($d \gg n$).
- Training too long (over-optimization).
- No regularization or too much capacity.

Underfitting occurs when the model is too simple to capture the underlying pattern.

Bias–variance decomposition (for squared error):

$$\mathbb{E}[(\hat{f}(\mathbf{x}) - y)^2] = \underbrace{(\mathbb{E}[\hat{f}(\mathbf{x})] - f^*(\mathbf{x}))^2}_{\text{Bias}^2} + \underbrace{\mathbb{E}[(\hat{f}(\mathbf{x}) - \mathbb{E}[\hat{f}(\mathbf{x})])^2]}_{\text{Variance}} + \underbrace{\mathbb{E}[(y - f^*(\mathbf{x}))^2]}_{\text{Irreducible error}}.$$

Bias measures how much the model's average prediction deviates from the truth. Variance measures how much predictions fluctuate across different training sets. Simple models have high bias, low variance; complex models have low bias, high variance. The tradeoff guides model selection.

Cross-validation estimates generalization without wasting data:

- k-fold CV: split data into k folds, train on $k-1$, test on the held-out fold, repeat k times, average metrics. Typical $k = 5$ or 10.
- Leave-one-out CV (LOOCV): $k = n$, expensive but nearly unbiased.
- Stratified CV: preserves class proportions in each fold.
- Train/validation/test split: hold out a test set at the very end, use validation set(s) for model selection.

Hyperparameter tuning: model capacity (degree of polynomial, k in k-NN, λ in ridge), learning rate, network architecture. Use validation performance (not test performance) for selection.

Learning curves: plot training and validation error vs. dataset size or training epochs. Gaps indicate overfitting; convergence to high error indicates underfitting.

Diagnostics:
- Training error $\ll$ validation error $\implies$ overfitting (increase regularization, get more data, reduce capacity).
- Both errors high $\implies$ underfitting (increase capacity, reduce regularization, add features).

For review, be able to: derive the bias–variance decomposition; explain why cross-validation is needed; choose k in k-fold CV; interpret learning curves; and design a model selection pipeline.

Section summary Generalization is the central concern of ML. The bias–variance tradeoff explains why we need regularized, capacity-controlled models. Cross-validation provides a reliable estimate of test performance for model selection. Always keep a separate test set for final evaluation.

5 Classical ML: Trees, SVMs, PCA, Clustering, and Ensembles

5.1 Core ideas

Decision trees partition the feature space recursively:
- At each node, select a feature j and threshold t to maximize information gain: $IG = H(\text{parent}) - \sum_v \frac{|D_v|}{|D|} H(D_v)$.
- For classification, H is entropy $-\sum_k p_k \log p_k$ or Gini impurity $1 - \sum_k p_k^2$.
- For regression, H is MSE: $\frac{1}{|D|}\sum_{i \in D}(y^{(i)} - \bar{y}_D)^2$.
- Prune to avoid overfitting: cost-complexity pruning.
- CART, ID3, C4.5 are common algorithms.

Support vector machines (SVMs) find the maximum-margin separating hyperplane.
- Primal: $\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2$ subject to $y^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b) \geq 1$ for all i.
- Dual: $\max_{\boldsymbol{\alpha}} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y^{(i)} y^{(j)} \mathbf{x}^{(i)\top} \mathbf{x}^{(j)}$, with $0 \leq \alpha_i \leq C$.
- Soft-margin (slack variables) handles non-separable data; C controls the penalty on margin violations.
- Kernel trick: replace $\langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \rangle$ with $k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$. Common kernels: polynomial $(\mathbf{x}^\top \mathbf{x}' + c)^d$, RBF $\exp(-\gamma\|\mathbf{x} - \mathbf{x}'\|^2)$.
- SVMs only use support vectors ($\alpha_i > 0$), making them memory-efficient for prediction.

PCA (principal component analysis) finds directions of maximum variance.
- Center the data: $\mathbf{X}_c = \mathbf{X} - \boldsymbol{\mu}$.
- Compute SVD $\mathbf{X}_c = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$; columns of $\mathbf{V}$ are principal components.
- Projection: $\mathbf{Z} = \mathbf{X}_c \mathbf{V}_k$ (first k components).
- Variance explained by k components: $\sum_{j=1}^{k} \sigma_j^2 / \sum_{j=1}^{\min(n,d)} \sigma_j^2$.
- Used for dimensionality reduction, visualization, denoising.

Clustering partitions data into groups:
- k-means: minimize $\sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}[z_i = k] \|\mathbf{x}^{(i)} - \boldsymbol{\mu}_k\|_2^2$. EM-like: assign points to nearest centroid, recompute centroids. Sensitive to initialization (use k-means++).

- DBSCAN: density-based, finds arbitrarily shaped clusters, robust to outliers, no need to specify K.
- Hierarchical (agglomerative): merge closest clusters by linkage (single, complete, average); visualize with dendrogram.

Ensemble methods combine multiple models:

- Bagging (bootstrap aggregating): train models on bootstrap samples, average predictions. Reduces variance without increasing bias. Random forest = bagged decision trees with random feature subsets.
- Boosting (AdaBoost, gradient boosting): sequentially train models to correct previous errors. Gradient boosting machines (GBM, XGBoost, LightGBM) fit new trees to the negative gradient of the loss.
- Stacking: train a meta-learner on predictions of base models.

For review, be able to: compute information gain for a split; derive the SVM dual; explain the kernel trick; perform PCA via SVD; run k-means and choose K; and compare bagging vs. boosting.

Section summary Classical ML offers a rich toolkit: decision trees (interpretable, handle non-linearity), SVMs (maximum-margin with kernels), PCA (unsupervised dimensionality reduction), clustering (discovering groups), and ensembles (combining weak learners). These remain powerful for tabular and moderately-sized data.

6 Statistical Learning: Estimation, Bayes, Causality, and Uncertainty

6.1 Core ideas

Statistical estimation: we observe data $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^n$ i.i.d. from $P(X, Y)$. We estimate parameters $\boldsymbol{\theta}$ of a model $p(y|\mathbf{x}; \boldsymbol{\theta})$.

Maximum likelihood estimation (MLE):

$$\hat{\boldsymbol{\theta}}_{\text{MLE}} = \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log p(y^{(i)}|\mathbf{x}^{(i)}; \boldsymbol{\theta}).$$

MLE is consistent (converges to true $\boldsymbol{\theta}^*$ as $n \to \infty$) and asymptotically efficient (achieves Cramér–Rao lower bound).

Maximum a posteriori (MAP) adds a prior $p(\boldsymbol{\theta})$:

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log p(y^{(i)}|\mathbf{x}^{(i)}; \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}).$$

L^2 regularization on weights corresponds to a Gaussian prior; L^1 corresponds to a Laplace prior.

Bayesian inference computes the full posterior:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} \propto p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}).$$

Prediction: $p(y^*|\mathbf{x}^*, \mathcal{D}) = \int p(y^*|\mathbf{x}^*, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta}$ (posterior predictive). For most models the integral is intractable; approximation via MCMC (sampling) or variational inference (optimization).

Causality goes beyond correlation. Key concepts:

- Structural causal models (SCMs): $X_i = f_i(\mathbf{PA}_i, U_i)$ where $\mathbf{PA}_i$ are parents of X_i and U_i are exogenous noise.

- Interventions $do(X_j = x)$ vs. conditioning $P(Y|X_j = x)$. The do-operator removes incoming edges to X_j.
- Confounders: Z affects both X and Y; conditioning on Z blocks the backdoor path.
- Randomized controlled trials (RCT) are the gold standard for causal inference.
- Methods: propensity score matching, instrumental variables, difference-in-differences.

Uncertainty quantification:

- Aleatoric uncertainty: irreducible noise in data (e.g., measurement error). Cannot be reduced with more data.
- Epistemic uncertainty: uncertainty about model parameters due to limited data. Decreases with more data.
- Calibration: a model is calibrated if among predictions with confidence p, the true fraction correct is p.

For review, be able to: derive MLE for Gaussian and Bernoulli; explain MAP as regularized MLE; describe Bayesian vs. frequentist approaches; define confounding; distinguish aleatoric and epistemic uncertainty.

Section summary Statistical learning provides the probabilistic foundation: MLE for point estimates, Bayesian inference for full uncertainty, and causal reasoning for interventional questions. Understanding estimation properties (bias, consistency, efficiency) is essential for diagnosing models.

7 Probabilistic ML: Generative Models, Latent Variables, and Graphical Models

7.1 Core ideas

Generative vs. discriminative models:

- Discriminative: $p(y|\mathbf{x})$, e.g., logistic regression, neural networks.
- Generative: $p(\mathbf{x}, y) = p(\mathbf{x}|y)p(y)$, e.g., naive Bayes, GMMs.
- Generative models can produce new samples $\mathbf{x} \sim p(\mathbf{x})$ and handle missing data naturally.
- Naive Bayes assumes conditional independence: $p(\mathbf{x}|y) = \prod_{j=1}^{d} p(x_j|y)$. Despite the strong assumption, it works well for text classification.

Gaussian mixture models (GMMs):

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \quad \sum_{k=1}^{K} \pi_k = 1,\ \pi_k \geq 0.$$

Latent variable $z_i \in \{1, \ldots, K\}$ indicates which component generated $\mathbf{x}^{(i)}$. The log-likelihood is non-convex; use expectation–maximization (EM):

1. E-step: compute responsibilities $\gamma_{ik} = P(z_i = k|\mathbf{x}^{(i)}, \boldsymbol{\theta}^{(t)})$.
2. M-step: update $\pi_k = \frac{1}{n}\sum_i \gamma_{ik}$, $\boldsymbol{\mu}_k = \frac{\sum_i \gamma_{ik}\mathbf{x}^{(i)}}{\sum_i \gamma_{ik}}$, $\boldsymbol{\Sigma}_k = \frac{\sum_i \gamma_{ik}(\mathbf{x}^{(i)}-\boldsymbol{\mu}_k)(\mathbf{x}^{(i)}-\boldsymbol{\mu}_k)^\top}{\sum_i \gamma_{ik}}$.

EM monotonically increases the marginal log-likelihood.

Probabilistic graphical models represent conditional independence:

- Bayesian networks (DAG): $p(\mathbf{x}) = \prod_{j=1}^{d} p(x_j|\mathbf{pa}(x_j))$. Factorization according to graph structure.
- Markov random fields (undirected): $p(\mathbf{x}) = \frac{1}{Z}\prod_{C\in\mathcal{C}} \psi_C(\mathbf{x}_C)$ where Z is the partition function.
- Hidden Markov models (HMMs): latent states z_t evolve as a Markov chain $p(z_t|z_{t-1})$, observations x_t depend on z_t. Forward-backward algorithm computes $p(z_t|\mathbf{x}_{1:T})$; Viterbi finds the most likely sequence.

Latent variable models (LVMs) assume the observed data is generated from unobserved latent variables:

- Factor analysis, probabilistic PCA ($\mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\epsilon}$, $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$).
- Topic models (LDA): documents are mixtures of topics, topics are distributions over words.

Variational inference: approximate intractable posterior $p(\mathbf{z}|\mathbf{x})$ with a simpler $q(\mathbf{z})$ by minimizing KL divergence:

$$\mathrm{KL}(q\|p) = \mathbb{E}_q[\log q(\mathbf{z}) - \log p(\mathbf{z}|\mathbf{x})] = -\mathrm{ELBO} + \log p(\mathbf{x}).$$

Maximizing the evidence lower bound (ELBO) $\mathcal{L} = \mathbb{E}_q[\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z})]$ is equivalent to minimizing KL.

For review, be able to: derive EM for GMMs; explain the difference between generative and discriminative models; factorize a joint distribution from a DAG; derive the ELBO; and implement the forward-backward algorithm.

Section summary Probabilistic ML models the full data distribution. GMMs cluster with soft assignments via EM. Graphical models encode structure and enable efficient inference. Variational inference approximates intractable posteriors, forming the foundation for modern deep generative models.

8 Fundamentals of Deep Learning: MLPs, Backpropagation, and Optimization

8.1 Core ideas

Multilayer perceptrons (MLPs) stack affine transformations with elementwise nonlinearities:

$$\mathbf{h}_1 = \sigma_1(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1), \quad \mathbf{h}_2 = \sigma_2(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2), \quad \hat{\mathbf{y}} = \mathbf{W}_L\mathbf{h}_{L-1} + \mathbf{b}_L.$$

Activation functions introduce nonlinearity:

- ReLU: $\sigma(z) = \max(0, z)$. Cheap gradient, mitigates vanishing gradient.
- Sigmoid: $\sigma(z) = 1/(1+e^{-z})$, output in $(0, 1)$. Saturated regions cause vanishing gradients.
- Tanh: $\tanh(z) = (e^z - e^{-z})/(e^z + e^{-z})$, output in $(-1, 1)$.
- Leaky ReLU: $\max(\alpha z, z)$ for small $\alpha > 0$.
- Swish/GELU: $z \cdot \sigma(z)$, smooth approximation used in transformers.

Backpropagation computes gradients via the chain rule:

1. Forward pass: compute hidden activations $\mathbf{h}_\ell = f_\ell(\mathbf{h}_{\ell-1})$ for $\ell = 1, \ldots, L$.
2. Compute loss $\mathcal{L}$.
3. Backward pass: $\frac{\partial \mathcal{L}}{\partial \mathbf{h}_\ell} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}_{\ell+1}} \frac{\partial \mathbf{h}_{\ell+1}}{\partial \mathbf{h}_\ell}$.
4. $\frac{\partial \mathcal{L}}{\partial \mathbf{W}_\ell} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}_\ell} \frac{\partial \mathbf{h}_\ell}{\partial \mathbf{W}_\ell}$.

Automatic differentiation (autograd) implements this for arbitrary computation graphs.

Optimization algorithms for deep learning:

- SGD: $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \tilde{\nabla}_t$ (unbiased gradient estimate using minibatches).
- Momentum: $\mathbf{v}_{t+1} = \beta \mathbf{v}_t + \tilde{\nabla}_t$, $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{v}_{t+1}$. Accumulates past gradients to dampen oscillations.
- Adam: adaptive learning rates per parameter using running averages of gradient and squared gradient:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \quad v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \quad \theta_{t+1} = \theta_t - \eta \frac{m_t}{\sqrt{v_t} + \epsilon}.$$

- Learning rate scheduling: step decay, cosine annealing, warmup.

Numerical considerations:

- Vanishing/exploding gradients: caused by repeated multiplication in deep nets. Solutions: proper initialization (Xavier/Glorot, He), batch normalization, residual connections, gradient clipping.
- Batch normalization: normalize activations per batch: $\hat{x} = (x - \mu_B)/\sqrt{\sigma_B^2 + \epsilon}$, then scale and shift: $y = \gamma\hat{x} + \beta$. Allows higher learning rates and reduces sensitivity to initialization.
- Dropout: randomly zero out neurons during training with probability p. Prevents co-adaptation. Equivalent to training an ensemble of sub-networks.

Universal approximation theorem: a feedforward network with one hidden layer and a nonlinear activation can approximate any continuous function on a compact domain arbitrarily well, given enough hidden units.

For review, be able to: derive backpropagation for a 2-layer MLP; explain why nonlinearities are necessary; implement SGD with momentum; diagnose vanishing gradients; and describe batch normalization.

Section summary Deep learning stacks differentiable nonlinear layers. Backpropagation via the chain rule enables gradient computation. SGD and its variants (momentum, Adam) optimize non-convex objectives. Careful initialization, normalization, and regularization are essential for training deep networks.

9 Modern Deep Learning: CNNs, RNNs, Attention, and Transformers

9.1 Core ideas

Convolutional neural networks (CNNs) exploit spatial structure:

- 2D convolution: $(I * K)(i, j) = \sum_m \sum_n I(i + m, j + n)K(m, n)$. Shares weights across spatial locations (weight tying).
- Properties: translation equivariance (shifting input shifts output equally), local connectivity (receptive field).
- Pooling (max, average) provides local translation invariance and downsampling.
- Common architectures: LeNet-5 (conv+pool+FC), AlexNet (ReLU, dropout, data augmentation), VGG (deep stack of 3x3 conv), ResNet (skip connections via $y = F(x) + x$), Inception/GoogLeNet (parallel filters of different sizes).
- Depthwise separable convolutions (used in MobileNet): spatial conv + pointwise conv, fewer parameters.

Recurrent neural networks (RNNs) model sequential data:

$$\mathbf{h}_t = \sigma(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{hx}\mathbf{x}_t + \mathbf{b}_h), \quad \hat{\mathbf{y}}_t = \mathbf{W}_{hy}\mathbf{h}_t + \mathbf{b}_y.$$

Backpropagation through time (BPTT) unrolls the computation graph. RNNs suffer from vanishing/exploding gradients over long sequences.

Gated RNNs address long-range dependencies:

- LSTM: cell state $\mathbf{c}_t$ with forget $\mathbf{f}_t$, input $\mathbf{i}_t$, output $\mathbf{o}_t$ gates:

$$\mathbf{f}_t = \sigma(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f), \quad \mathbf{i}_t = \sigma(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i), \quad \tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c),$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t, \quad \mathbf{o}_t = \sigma(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o), \quad \mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t).$$

- GRU: simplified with reset and update gates.

Attention mechanism allows the model to focus on relevant parts of the input:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}.$$

Q (queries), **K** (keys), **V** (values); scaling by $1/\sqrt{d_k}$ prevents softmax saturation.

Transformer (Vaswani et al., 2017): purely attention-based architecture.

- Multi-head attention: h parallel attention heads, each on projected Q/K/V, concatenated and re-projected.
- Positional encoding (sinusoidal or learned) injects sequence order.
- Layer normalization and residual connections after each sub-layer.
- Encoder: self-attention + feedforward. Decoder: masked self-attention + cross-attention + feedforward.
- Masks: padding mask (ignore padding tokens), causal/autoregressive mask (prevent attending to future tokens).
- $O(n^2)$ complexity in sequence length n motivates efficient variants (Longformer, Linformer, Flash attention).

Large-scale pretraining: BERT (bidirectional encoder, masked language modeling + next sentence prediction), GPT (autoregressive decoder, causal language modeling), T5 (encoder-decoder, text-to-text framework). Fine-tune on downstream tasks. Scaling laws: performance improves predictably with model size, data size, and compute.

For review, be able to: explain the receptive field of a CNN; derive gradient flow in an RNN; implement attention; draw the transformer architecture; and describe masked language modeling pretraining.

Section summary Modern deep learning is built on CNNs (spatial data), RNNs/LSTMs (sequential data), and transformers (general sequence modeling via attention). Pretraining large models on massive data followed by fine-tuning dominates NLP and increasingly computer vision.

10 Generative AI: VAEs, GANs, Diffusion, and Self-Supervised Learning

10.1 Core ideas

Variational autoencoders (VAEs) combine deep learning with variational inference:

- Encoder $q_\phi(\mathbf{z}|\mathbf{x})$ maps data to a latent distribution (typically Gaussian $\mathcal{N}(\boldsymbol{\mu}_\phi(\mathbf{x}), \boldsymbol{\sigma}^2_\phi(\mathbf{x})\mathbf{I})$).
- Decoder $p_\theta(\mathbf{x}|\mathbf{z})$ reconstructs data from latent code.
- ELBO objective:

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})).$$

  First term is reconstruction, second is KL regularization toward prior $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$.
- Reparameterization trick: sample $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$, then $\mathbf{z} = \boldsymbol{\mu}_\phi(\mathbf{x}) + \boldsymbol{\sigma}_\phi(\mathbf{x}) \odot \boldsymbol{\epsilon}$ enables backpropagation through sampling.
- Conditional VAE: condition on y to generate specific classes.

Generative adversarial networks (GANs):

- Generator G_θ maps noise $\mathbf{z} \sim p(\mathbf{z})$ to fake data $\tilde{\mathbf{x}}$.
- Discriminator D_ϕ distinguishes real $\mathbf{x}$ from fake $\tilde{\mathbf{x}}$.
- Minimax game:

$$\min_\theta \max_\phi \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D_\phi(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D_\phi(G_\theta(\mathbf{z})))].$$

- Training is challenging: mode collapse, non-convergence, vanishing gradients.
- Improvements: DCGAN (convolutional architecture), Wasserstein GAN (WGAN) with gradient penalty, conditional GANs, StyleGAN.

Diffusion models (score-based generative models):

- Forward process: gradually add Gaussian noise over T steps: $q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$.
- Reverse process: learn to denoise: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$.
- Training objective: predict the noise $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ added at step t:

$$\mathcal{L} = \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\|^2].$$

- Sampling: start from $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$, iteratively denoise.
- DDPM (Ho et al., 2020) achieved high-quality image generation. Latent diffusion models (Stable Diffusion) operate in VAE latent space.
- Connection to score matching: $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) \approx -\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)/\sqrt{1-\bar{\alpha}_t}$.

Self-supervised learning (SSL) learns representations without human labels:

- Contrastive: pull positive pairs together, push negative pairs apart. SimCLR: augmentations (crop, color jitter, blur), InfoNCE loss:

$$\mathcal{L} = -\log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k\neq i}\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)}.$$

- Non-contrastive: BYOL, SimSiam predict one view's representation from another without negatives.
- Masked autoencoders: mask patches of image/text, predict the missing content (BERT, MAE).
- SSL is a key ingredient in large foundation models.

For review, be able to: derive the VAE ELBO; explain the reparameterization trick; describe the GAN minimax game; outline the diffusion forward/reverse process; and define contrastive learning.

Section summary Generative AI creates new data. VAEs learn smooth latent spaces via variational inference. GANs pit generator against discriminator in a minimax game. Diffusion models iteratively denoise and currently produce the highest-quality images. Self-supervised learning extracts useful representations from unlabeled data, enabling large-scale pretraining.

11 Reinforcement Learning: MDPs, TD Learning, Q-Learning, and Policy Gradients

11.1 Core ideas

Markov decision process (MDP) formalizes sequential decision-making:

- State space $\mathcal{S}$, action space $\mathcal{A}$.
- Transition dynamics $P(\mathbf{s}'|\mathbf{s}, a)$: probability of next state given current state and action.
- Reward function $R(\mathbf{s}, a, \mathbf{s}')$: immediate reward.
- Discount factor $\gamma \in [0, 1]$: $G_t = \sum_{k=0}^{\infty}\gamma^k R_{t+k+1}$ (return).
- Policy $\pi(a|\mathbf{s})$: distribution over actions given state.
- State-value function: $V^\pi(\mathbf{s}) = \mathbb{E}_\pi[G_t|\mathbf{S}_t = \mathbf{s}]$.
- Action-value function: $Q^\pi(\mathbf{s}, a) = \mathbb{E}_\pi[G_t|\mathbf{S}_t = \mathbf{s}, A_t = a]$.
- Bellman equations:

$$V^\pi(\mathbf{s}) = \sum_a \pi(a|\mathbf{s})\sum_{\mathbf{s}'} P(\mathbf{s}'|\mathbf{s}, a)[R(\mathbf{s}, a, \mathbf{s}') + \gamma V^\pi(\mathbf{s}')].$$

$$Q^\pi(\mathbf{s}, a) = \sum_{\mathbf{s}'} P(\mathbf{s}'|\mathbf{s}, a)[R(\mathbf{s}, a, \mathbf{s}') + \gamma\sum_{a'}\pi(a'|\mathbf{s}')Q^\pi(\mathbf{s}', a')].$$

- Optimal policy π^* satisfies $V^*(\mathbf{s}) = \max_a Q^*(\mathbf{s}, a)$.

- Bellman optimality equation: $Q^*(\mathbf{s}, a) = \mathbb{E}_{\mathbf{s}'}[R(\mathbf{s}, a, \mathbf{s}') + \gamma \max_{a'} Q^*(\mathbf{s}', a')]$.

Dynamic programming solves MDPs when dynamics are known:

- Policy iteration: evaluate V^π (solve linear system), improve $\pi'(a|\mathbf{s}) = \mathbb{1}[a = \arg\max_a Q^\pi(\mathbf{s}, a)]$.
- Value iteration: $V_{k+1}(\mathbf{s}) = \max_a \sum_{\mathbf{s}'} P(\mathbf{s}'|\mathbf{s}, a)[R(\mathbf{s}, a, \mathbf{s}') + \gamma V_k(\mathbf{s}')]$.

Model-free RL: learn from experience without knowing P or R.

Monte Carlo methods: sample full episodes, average returns: $V(\mathbf{s}) \leftarrow V(\mathbf{s}) + \alpha(G_t - V(\mathbf{s}))$.

Temporal difference (TD) learning: bootstrap from current estimate:

$$V(\mathbf{s}_t) \leftarrow V(\mathbf{s}_t) + \alpha(\underbrace{r_t + \gamma V(\mathbf{s}_{t+1})}_{\text{TD target}} - V(\mathbf{s}_t)).$$

TD uses fewer samples than MC and works with incomplete episodes. TD(0) is the simplest form; TD(λ) averages over n-step returns.

Q-learning (off-policy TD control):

$$Q(\mathbf{s}_t, a_t) \leftarrow Q(\mathbf{s}_t, a_t) + \alpha \left(r_t + \gamma \max_{a'} Q(\mathbf{s}_{t+1}, a') - Q(\mathbf{s}_t, a_t) \right).$$

Uses max over next actions regardless of behavior policy (off-policy). Deep Q-Network (DQN) approximates Q with a neural network; uses experience replay (store transitions in buffer, sample random minibatches) and target network (stable targets) to stabilize training.

Policy gradient methods directly optimize the policy π_θ:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|\mathbf{s}) \cdot G_t].$$

REINFORCE: Monte Carlo policy gradient. Actor-critic: combine policy gradient (actor) with value function (critic). A2C/A3C use advantage $A(\mathbf{s}, a) = Q(\mathbf{s}, a) - V(\mathbf{s})$ to reduce variance. PPO clips policy updates for stability.

Exploration vs. exploitation: ϵ-greedy (with probability ϵ take random action), Boltzmann (softmax over Q-values), upper confidence bound (UCB).

For review, be able to: define an MDP; derive Q-learning; explain why TD is more efficient than MC; describe the policy gradient theorem; and implement ϵ-greedy exploration.

Section summary RL solves sequential decision problems. Value-based methods (Q-learning) learn the optimal action-value function. Policy gradient methods directly optimize the policy. Deep RL combines these with neural networks (DQN, A2C, PPO). The exploration–exploitation tradeoff is central to all RL algorithms.

12 Practice and Ethics: MLOps, Fairness, Explainability, and Safety

12.1 Core ideas

MLOps (ML operations) manages the ML lifecycle:

- Data pipeline: collection, cleaning, versioning (DVC, Dolt), feature store, data quality monitoring.
- Experiment tracking: hyperparameters, metrics, artifacts (MLflow, Weights & Biases, TensorBoard).
- Model registry: versioning, staging (dev/staging/production), rollback.
- CI/CD: automated testing (data validation, model evaluation), deployment.
- Serving: batch inference, real-time inference (REST/gRPC), edge deployment.
- Monitoring: data drift (distribution of inputs changes), concept drift ($P(y|\mathbf{x})$ changes), model degradation.

- Retraining strategies: scheduled (e.g., weekly), triggered by drift, continuous.

Fairness in ML:

- Sources of bias: historical bias, representation bias (under-sampled groups), measurement bias, aggregation bias.
- Group fairness notions:
  - Demographic parity: $P(\hat{y} = 1|A = a) = P(\hat{y} = 1|A = b)$ for protected attributes A.
  - Equalized odds: $P(\hat{y} = 1|y = 1, A = a) = P(\hat{y} = 1|y = 1, A = b)$ (equal TPR) and same for FPR.
  - Equal opportunity: equal TPR only.

  - Impossibility results: in general, demographic parity and equalized odds cannot both be satisfied unless the model is perfect or base rates are equal. - Mitigation approaches: pre-processing (reweigh training data), in-processing (fairness constraints in loss), post-processing (adjust thresholds per group). - Fairness audits: disaggregated evaluation across demographic groups.

Explainability and interpretability:

- Intrinsic interpretability: linear models, decision trees, attention weights.
- Post-hoc methods:
  - Feature attribution: SHAP (Shapley values from cooperative game theory, additive feature attribution), LIME (local surrogate model), integrated gradients.
  - Saliency maps: gradient of output w.r.t. input pixels.
  - Gradient×input, DeepLIFT, Guided backpropagation.
  - Counterfactual explanations: smallest change to input that changes the prediction.

  - Partial dependence plots (PDPs): show average prediction as a function of one feature. - Permutation feature importance: drop in performance when shuffling a feature. - Tradeoff: simpler explanations may not capture complex model behavior.

AI safety and robustness:

- Adversarial examples: small imperceptible perturbations $x + \delta$ that flip predictions. Fast Gradient Sign Method (FGSM): $\delta = \epsilon \cdot \text{sign}(\nabla_x \mathcal{L})$. Projected Gradient Descent (PGD): iterative FGSM with projection.
- Adversarial training: train on adversarially perturbed inputs. Tradeoff between standard and robust accuracy.
- Distributional shift: domain adaptation, domain generalization, out-of-distribution detection.
- Alignment: ensuring AI systems pursue intended goals. Reward hacking, specification gaming, value learning.
- Robustness certification: guaranteed bounds on prediction under ℓ_p perturbations.

Privacy:

- Differential privacy (DP): adding calibrated noise to training or outputs ensures that removing any single example does not significantly change the result. ϵ-DP: $\frac{P(\mathcal{M}(D)\in S)}{P(\mathcal{M}(D')\in S)} \leq e^{\epsilon}$ for neighboring datasets D, D'.
- Federated learning: train across decentralized data without sharing raw data. Secure aggregation using cryptographic protocols.
- Model inversion, membership inference: attacks that extract training data.

For review, be able to: describe the MLOps lifecycle; define demographic parity and equalized odds; compute SHAP values for a simple model; generate adversarial examples via FGSM; and explain differential privacy.

Section summary Practice and ethics are essential for responsible ML deployment. MLOps ensures reliable systems through versioning, monitoring, and automated pipelines. Fairness requires careful auditing and mitigation of bias. Explainability builds trust and detects failure modes. Safety addresses robustness, alignment, and privacy. These considerations are becoming

as important as model accuracy.