Common Patterns Across Models

1. **Modeling:** Decide whether to model $p(y \mid x)$ (discriminative) or $p(x, y)$ (generative).

2. **Loss Functions:** Examples include squared error, cross-entropy, hinge loss, etc.

3. **Optimization:** Solve in closed form (e.g. linear regression) or use iterative methods such as gradient descent/backpropagation.

Discriminative Models: Directly model

$$p(y \mid x; w).$$

For binary logistic regression:

$$p(y = 1 \mid x; w) = \sigma(w^T x), \quad \sigma(z) = \frac{1}{1 + e^{-z}}.$$

Generative Models: Model $p(x, y)$ then apply Bayes' rule to compute $p(y \mid x)$.

MLE & MAP:

$$w_{\text{MLE}} = \arg\max_w \log\, p((x, y) \mid w),$$

$$w_{\text{MAP}} = \arg\max_w \Big\{ \log\, p((x, y) \mid w) + \log\, p(w) \Big\},$$

with a Gaussian prior $w \sim \mathcal{N}(0, \sigma_0^2 I)$ yielding ridge regression:

$$w_{\text{MAP}} = \arg\min_w \sum_{i=1}^{N} \Big(y^{(i)} - w^T x^{(i)} \Big)^2 + \lambda \|w\|_2^2, \quad \lambda = \frac{\sigma^2}{\sigma_0^2}.$$

Optimal w & Likelihood Functions

Linear Regression:

$$w^* = (X^T X)^{-1} X^T y, \quad p((x, y) \mid w) = \prod_{i=1}^{N} \mathcal{N}\big(y^{(i)} \mid w^T x^{(i)}, \sigma^2\big)$$

Binary Logistic Regression:

$$p(y \mid x; w) = \sigma(w^T x)^y \,[1 - \sigma(w^T x)]^{1-y},$$

with log-likelihood

$$\ell(w) = \sum_{i=1}^{N} \Big[y^{(i)} \log \sigma(w^T x^{(i)}) + (1 - y^{(i)}) \log(1 - \sigma(w^T x^{(i)})) \Big].$$

Multiclass (Softmax):

Dataset Likelihood:

$$\mathcal{L}(\{w_\ell\}) = \prod_{i=1}^{N} \prod_{k=1}^{K} \left(\frac{\exp\big(w_k^T x^{(i)}\big)}{\sum_{\ell=1}^{K} \exp\big(w_\ell^T x^{(i)}\big)} \right)^{\mathbb{I}\{y^{(i)} = k\}}.$$

Taking the logarithm gives the log-likelihood:

$$\ell(\{w_\ell\}) = \sum_{i=1}^{N} \left[w_{y^{(i)}}^T x^{(i)} - \log \left(\sum_{\ell=1}^{K} \exp\big(w_\ell^T x^{(i)}\big) \right) \right].$$

Ridge Regression:

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

Loss Functions

- **0/1 Loss:** $L(y, \hat{y}) = \mathbb{I}(y \neq \hat{y})$.
- **Hinge Loss:** $L(y, f(x)) = \max(0,\, 1 - y\, f(x))$.
- **L1 Loss:** $L(y, \hat{y}) = |y - \hat{y}|$.
- **L2 Loss:** $L(y, \hat{y}) = (y - \hat{y})^2$.
- **Binary Cross-Entropy:** $L(y, \hat{y}) = -\Big[y \log \hat{y} + (1 - y) \log(1 - \hat{y}) \Big]$.
- **Softmax Loss:** $L = -\log \frac{\exp(w_k^T x)}{\sum_\ell \exp(w_\ell^T x)}$.

Gradient Descent

Gradient Descent:

- Iteratively update parameters by moving opposite to the gradient:

$$w \leftarrow w - \eta \nabla L(w),$$

  where η is the learning rate.
- A proper choice of η is crucial: if too high, updates overshoot minima; if too low, convergence is slow.

Stochastic Gradient Descent (SGD):

- Approximates the full gradient using a single (or a mini-batch of) training example(s):

$$w \leftarrow w - \eta \nabla L^{(i)}(w),$$

  where $L^{(i)}(w)$ is the loss for the ith example.
- Benefits: faster iterations and potential to escape shallow local minima.
- Trade-off: introduces variance in updates, often requiring a decaying learning rate schedule.

Matrix Rules

Algebra:

$$(AB)^T = B^T A^T, \quad (A^{-1})^T = (A^T)^{-1}.$$

For column vectors a, b: $a^T b$ is scalar. If X is $n \times d$ and w is $d \times 1$, then Xw is $n \times 1$.

Derivatives:

$$\frac{\partial}{\partial w}(w^T A w) = (A + A^T) w \quad (\text{or } 2Aw \text{ if } A \text{ is symmetric}),$$

$$\frac{\partial}{\partial w} \frac{1}{2} \|y - Xw\|_2^2 = -X^T (y - Xw).$$

Norms:

$$\|w\|_2 = \sqrt{w^T w}, \quad \|w\|_1 = \sum_i |w_i|.$$

Lagrangian Method

Steps:

1. Form the Lagrangian: $\mathcal{L}(w, \lambda) = \text{Objective}(w) + \lambda\,(\text{Constraint}(w))$.
2. Differentiate with respect to w and λ.
3. Set derivatives to zero and solve.

Example (SVM):

$$\mathcal{L}(w, b, \lambda) = \frac{1}{2} \|w\|_2^2 - \sum_{i=1}^{N} \lambda_i \Big[y^{(i)} (w^T x^{(i)} + b) - 1 \Big].$$

Terminology & Notation

Terminology:

Posterior: $p(w \mid (x, y))$ after observing data.

Posterior Predictive:

$$p(y^* \mid x^*, (x, y)) = \int p(y^* \mid x^*, w)\, p(w \mid (x, y))\, dw.$$

Marginal Likelihood:

$$p((x, y)) = \int p((x, y) \mid w)\, p(w)\, dw.$$

Class-Conditional: $p(x \mid y)$.

Notation:

- N: Number of training examples.
- K: Number of classes.
- $x^{(i)}$: ith input data point.
- $y^{(i)}$: Label corresponding to $x^{(i)}$.
- X: Design matrix whose rows are $x^{(i)}$.
- w, w_ℓ: Weight vector(s); w_ℓ denotes the weight for class ℓ in multiclass models.
- w_0 or b: Bias term.
- η: Learning rate in gradient descent.
- λ: Regularization parameter (ridge: ℓ_2, lasso: ℓ_1).
- C: SVM regularization parameter trading off margin and classification errors.
- ξ_i: Slack variable for the ith example in soft-margin SVM.
- $\phi(x)$: Feature mapping or basis function.
- $\sigma(z)$: Sigmoid function, $\frac{1}{1+e^{-z}}$.
- $\mathbb{I}\{\cdot\}$: Indicator function.
- L: Generic loss function.
- ℓ: Often denotes log-likelihood.

Bias–Variance & Regularization

Bias–Variance Tradeoff: High bias → underfitting; high variance → overfitting. Regularization (e.g. ridge, lasso) can reduce variance.

Ridge Regression:

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

Lasso Regression:
Minimize

$$\sum_{i=1}^{N} \left(y^{(i)} - w^T x^{(i)} \right)^2 + \lambda \|w\|_1.$$

Neural Networks

Architecture:
• Feedforward NN with one hidden layer:

$$h = \sigma(W_1 x + b_1), \quad f = W_2 h + b_2.$$

• Deep networks with multiple hidden layers enable hierarchical feature learning.

Activation Functions:
Introduce non-linearity (e.g. Sigmoid, ReLU, tanh) to enable universal approximation.

Backpropagation:
Compute gradients via the chain rule:

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial f} \frac{\partial f}{\partial W}.$$

Performs a forward pass to compute outputs and a backward pass to update parameters.

Additional Points:

- **Model Selection:** Techniques such as cross-validation and regularization are key for avoiding overfitting.
- **Loss Functions:** Choice depends on the task—least squares for regression; softmax loss for classification.
- **Task Adaptation:** Neural networks can be tailored for both regression and classification tasks.
- **Bias Inclusion:** Always incorporate the bias term via the augmented input (see Supervised Learning Organization).

Support Vector Machines (SVMs)

Maximum Margin Classifier:
Finds the hyperplane that maximizes the distance (margin) between classes.

Hard Margin SVM:

$$\min_{w,b} \frac{1}{2} \|w\|_2^2 \quad \text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1.$$

Soft Margin SVM:

$$\min_{w,b} \frac{1}{2} \|w\|_2^2 + C \sum_{i=1}^{N} \xi_i \quad \text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \quad \xi_i \geq 0.$$

Additional Points:

- **Hinge Loss:** Penalizes points within the margin, defined as $L(y, f(x)) = \max(0, 1 - y\, f(x))$.
- **Regularization (C):** A higher C emphasizes minimizing classification errors, potentially at the cost of a smaller margin.
- **Kernel Trick:** Replaces inner products $x^T z$ with $K(x, z) = \phi(x)^T \phi(z)$ to handle nonlinearly separable data. Common kernels include linear, polynomial, and RBF.
- **Example Kernel:** For RBF, $K(x, x') = \exp(-\gamma \|x - x'\|_2^2)$.
- **Support Vectors:** The data points that lie closest to the decision boundary; they determine the position of the hyperplane.
- **Decision Boundaries:** SVMs often yield sharper boundaries compared to logistic regression.

Naive Bayes & Bayesian Linear Regression

Naive Bayes:

• **Modeling:** A generative model that estimates $p(x, y)$ by assuming feature independence given the class:

$$p(x \mid y) = \prod_j p(x_j \mid y).$$

• **Classification:** Compute the posterior via Bayes' rule:

$$p(y \mid x) \propto p(y)\, p(x \mid y).$$

• **Characteristics:** Simple, fast, and effective in high-dimensional settings, though it relies on the independence assumption.

Working with Generative Models for Classification:

• Estimate class priors $p(y)$ and class-conditional likelihoods $p(x \mid y)$ from the data.
• Apply Bayes' rule to obtain $p(y \mid x)$ and classify by choosing the class with maximum posterior probability.

Discriminative vs. Generative Models:

• **Discriminative models** (e.g., Logistic Regression) directly model $p(y \mid x)$ focusing on the decision boundary.
• **Generative models** (e.g., Naive Bayes) model the joint distribution $p(x, y)$ and derive $p(y \mid x)$ using Bayes' rule.
• Discriminative models often achieve higher asymptotic accuracy, while generative models can perform better with limited data or when model assumptions hold.

Bayesian Linear Regression:

• **Concept:** Treats weights w as random variables with a prior (commonly Gaussian).
• **Posterior:** Update beliefs with:

$$p(w \mid D) \propto p(D \mid w)\, p(w).$$

• **Prediction:** Integrate over the posterior to obtain:

$$p(y^* \mid x^*, D) = \int p(y^* \mid x^*, w)\, p(w \mid D)\, dw.$$