Supervised Learning Organization

Bias Trick: Augment the design matrix X with a column of 1's so that the bias w_0 is incorporated into w.

kNN:

- **Basic:** Compute (e.g. Euclidean) distances from a new point to all training examples; classify by majority vote or average for regression.
- **Kernelized:** Weight neighbors with a kernel (e.g. Gaussian) to smooth the influence of distant points.

Common Patterns Across Models

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

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

Optimization: Solve in closed form (e.g. linear regression) or use 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:

$$\mathcal{L}(\{v_\ell\}) = \prod_{i=1}^{N} \frac{\exp\big(v_{y^{(i)}}^T \phi(x^{(i)})\big)}{\sum_{\ell=1}^{K} \exp\big(v_\ell^T \phi(x^{(i)})\big)}.$$

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:** $L = -\log \frac{\exp(v_k^T \phi(x))}{\sum_\ell \exp(v_\ell^T \phi(x))}$.

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

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: $p(x \mid y)$.

Bias–Variance

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

Neural Nets:

- Activation functions: ReLU, Sigmoid, Softmax, etc.
- Use backpropagation (chain rule) to update parameters.
- Always include bias via the augmented input.

SVMs:

- **Hard-Margin SVM:**

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

- **Kernels:** Linear, Polynomial, Gaussian (RBF):

$$K(x, x') = \exp(-\gamma \|x - x'\|_2^2).$$

- Regularization parameter C balances margin width and classification errors.

Neural Networks

Architecture:
• Feedforward NN with one hidden layer:

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

• For deep networks, multiple hidden layers allow feature reuse.

Activation Functions:
Sigmoid, ReLU, tanh, etc. introduce non-linearity enabling universal approximation.

Backpropagation:
Compute gradients via the chain rule:

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

• Forward pass computes outputs; backward pass updates parameters.

Key Points:
• Neural networks learn adaptive basis functions.
• Overparameterization can improve gradient descent.

Support Vector Machines (SVMs)

Hard Margin SVM:

$$\min_{w,w_0} \frac{1}{2}\|w\|_2^2 \quad \text{s.t. } y^{(i)}(w^T x^{(i)} + w_0) \geq 1.$$

Soft Margin SVM:

$$\min_{w,w_0} \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{N}\xi_i \quad \text{s.t. } y^{(i)}(w^T x^{(i)} + w_0) \geq 1 - \xi_i, \quad \xi_i \geq 0.$$

Dual Formulation & Kernel Trick:
Express the solution as:

$$w = \sum_{i=1}^{N} \alpha_i y^{(i)} x^{(i)},$$

with $\sum_i \alpha_i y^{(i)} = 0$.

Replace inner products with kernels:

$$K(x,z) = \phi(x)^T \phi(z),$$

enabling nonlinear decision boundaries.

Regularization

Ridge Regression:

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

Lasso Regression:
Minimize

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