Supervised Learning & KNN

- Training set $S = \{(x_i, y_i)\}_{i=1}^m$ sampled i.i.d. from $\mathcal{D}$; learn $h : \mathcal{X} \to \mathcal{Y}$ that approximates f on unseen x.
- k-NN (classification): for query x' pick k nearest neighbors under a similarity (e.g. Euclidean, Gaussian $e^{-\|x-x'\|^2/2}$, Laplace, cosine) and predict $\hat{y} = \arg\max_y \sum_{i \in \text{knn}(x')} \mathbf{1}(y_i = y)$.
- Weighted k-NN: weight counts by similarity; regression outputs $h(x') = \dfrac{\sum_{i\in\text{knn}(x')} y_i K(x_i, x')}{\sum_{i\in\text{knn}(x')} K(x_i, x')}$.
- Non-parametric, stores all of S; query cost scales with m and d. Small $k \Rightarrow$ low bias/high variance, large $k \Rightarrow$ smoother but biased. Struggles under the curse of dimensionality.

Decision Trees & Hypothesis Search

- **Version Space:** $VS = \{h \in H \mid h(x_i) = y_i \ \forall (x_i, y_i) \in S\}$; list-then-eliminate removes inconsistent hypotheses sequentially.
- **IDT (top-down induction):** if S pure $\Rightarrow$ leaf; if no features $\Rightarrow$ majority leaf; else choose split attribute A, partition $\{S_v\}$, recurse on each branch.
- **Error split score:** $\text{Err}(S) = \min(\#\text{pos}, \#\text{neg})$, $\quad \Delta\text{Err} = \text{Err}(S) - \sum_v \text{Err}(S_v)$.
- Trees memorise training data when unconstrained. Control complexity via depth limits, early stopping (min gain / min samples), or post-pruning.
- Leaves output majority label along the path; structure is easy to interpret and handles categorical or numeric tests.

Bias, Variance, & Overfitting

$$\mathbb{E}[\text{Err}] = \underbrace{\text{Bias}^2}_{\text{model assumptions}} + \underbrace{\text{Variance}}_{\text{sensitivity to } S} + \underbrace{\sigma^2}_{\text{noise}}$$

- Empirical error $\hat{L}_S(h) = \frac{1}{|S|}\sum \mathbf{1}[h(x_i) \neq y_i]$; prediction error $L_{\mathcal{D}}(h) = \mathbb{P}_{(x,y)\sim\mathcal{D}}[h(x) \neq y]$.
- Overfitting: zero train error but large test error (e.g. memorising training points and guessing otherwise).
- Bias $\uparrow$ with simple hypothesis spaces (risk of underfitting); variance $\uparrow$ with overly flexible models. Choose intermediate complexity or add regularization.

Model Assessment

- **k-fold cross-validation:** split S into k folds S_i; for each hyperparameter p, train $h_i = A_p(S \setminus S_i)$, compute $err_{S_i}(h_i)$, average $err_{CV}(A_p) = \frac{1}{k}\sum_i err_{S_i}(h_i)$, pick p^* with smallest error, retrain on full S.
- **Binomial significance test:** under $H_0 : err_P(h) \geq \epsilon$, number of observed errors $K \sim \text{Binomial}(m, \epsilon)$. Compute p-value $P(K \leq k)$ (or $\geq k$). Reject H_0 if $p < \alpha$ (two-sided tests split α across tails).
- **Normal CI:** when $mp(1-p) \geq 5$, $err_P(h) \in \left[err_S(h) \pm 1.96\sqrt{\frac{p(1-p)}{m}}\right]$ with $p \approx err_S(h)$.
- **Hoeffding bound:** with prob. $\geq 1-\delta$, $err_P(h) \in \left[err_S(h) \pm \sqrt{\frac{-0.5\ln(\delta/2)}{m}}\right]$ for losses in $[0,1]$.
- **McNemar's test:** compare h_1, h_2 by wins w (only h_1 correct) and losses l (only h_2 correct); under H_0 wins $\sim \text{Binomial}(w+l, 0.5)$. Reject at 95% if $P(W \leq w)$ or $P(W \geq w)$ is < 0.025.

Linear Classifiers

- Hyperplane: $\{\vec{x} \mid \vec{w}\cdot\vec{x} + b = 0\}$ with normal vector $\vec{w}$; classifier $h_{\vec{w},b}(x) = \text{sign}(\vec{w}\cdot x + b)$.
- Homogeneous form: append bias as extra coordinate $x' = (x, 1)$, $w' = (w, b)$ so $\vec{w}'\cdot\vec{x}' = \vec{w}\cdot\vec{x} + b$.
- Functional margin of (x_i, y_i): $\gamma_i = y_i(\vec{w}\cdot x_i + b)$; geometric margin assumes $\|\vec{w}\| = 1$ so γ_i equals signed distance. Dataset margin $\gamma = \min_i \gamma_i$.
- Larger margins $\Rightarrow$ easier separability and tighter generalization bounds.

Perceptron

- **Batch homogeneous algorithm:** start $\vec{w}^{(0)} = \vec{0}$; while some i has $y_i(\vec{w}^{(t)}\cdot x_i) \leq 0$, update $\vec{w}^{(t+1)} = \vec{w}^{(t)} + y_i x_i$. Shuffle data; stop when no errors or after max epochs.
- **Convergence theorem:** if $\|x_i\| \leq R$ and data separable with margin γ, Perceptron makes $\leq R^2/\gamma^2$ updates (independent of unknown γ). Scaling examples or reordering does not change the bound.

Support Vector Machines

- Hard-margin SVM: $\min_{\vec{w},b} \dfrac{1}{2}\|\vec{w}\|^2$ s.t. $y_i(\vec{w}\cdot x_i + b) \geq 1$. Maximizes geometric margin $1/\|\vec{w}\|$ while keeping zero training error; support vectors satisfy equality.
- Soft-margin SVM introduces slack $\xi_i \geq 0$: $\min_{\vec{w},b,\xi} \dfrac{1}{2}\|\vec{w}\|^2 + C\sum_i \xi_i$ with constraints $y_i(\vec{w}\cdot x_i + b) \geq 1 - \xi_i$. $\xi_i = 0$ correct with margin, $0 < \xi_i < 1$ inside margin, $\xi_i \geq 1$ misclassified. C trades margin size vs. training errors; only points with margin ≤ 1 influence $\vec{w}$.

$$\min_{\vec{w},b,\vec{\xi}} \frac{1}{2}\vec{w}\cdot\vec{w} + C\sum_{i=1}^n \xi_i$$

$$\text{s.t.} \quad y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad \forall i$$

- **Slack variable** ξ_i measures how much (x_i, y_i) fails to achieve the margin.
  - **Idea:** Slack variables ξ_i capture training error.
  - For any training example $(\vec{x}_i, y_i)$:
    - $\xi_i \geq 1 \iff y_i(\vec{w}\cdot\vec{x}_i + b) \leq 0$ (error)
    - $0 < \xi_i < 1 \iff 0 < y_i(\vec{w}\cdot\vec{x}_i + b) < 1$ (correct, but within margin)
    - $\xi_i = 0 \iff y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1$ (correct)
  - The sum $\sum_i \xi_i$ is an upper bound on the number of training errors.

  - C controls the trade-off between margin size and training error.
- Examples with margin less than or equal to 1 are support vectors.

Duality & Kernels

- **Perceptron dual:** represent $\vec{w} = \sum_j \alpha_j y_j x_j$; update $\alpha_i \leftarrow \alpha_i + 1$ when $y_i \sum_j \alpha_j y_j (x_j \cdot x_i) \le 0$. Prediction $\operatorname{sign}\left(\sum_j \alpha_j y_j (x_j \cdot x)\right)$.
- **SVM dual:** maximize $D(\alpha) = \displaystyle\sum_i \alpha_i - \frac{1}{2}\sum_{ij} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j)$ subject to $\sum_i y_i \alpha_i = 0$, $0 \le \alpha_i \le C$; optimal $\vec{w} = \sum_i \alpha_i y_i x_i$ and $P^* = D^*$.
- **Kernel trick:** replace dot products by $K(x,z) = \Phi(x) \cdot \Phi(z)$ to work in high-dimensional feature spaces without explicit Φ. Common kernels: linear $x \cdot z$, polynomial $(x \cdot z + 1)^k$, RBF $e^{-\gamma \|x-z\|^2}$, sigmoid $\tanh(\gamma x \cdot z + c)$.
- Valid kernel $\Rightarrow$ Gram matrix $G_{ij} = K(x_i, x_j)$ is symmetric positive semi-definite; closure: non-negative sums, products, $f(x)f(z)$, compositions with feature maps, and $x^\top K z$ for PSD K.
- Leave-one-out for SVM: $err_{loo} \le \#SV/m$ and for homogeneous hard-margin also $\le R^2/(\gamma^2 m)$.

Regularized Linear Models

- **ERM:** choose hypothesis space $\mathcal{H}$ and minimize empirical loss; avoid overfitting via regularization or validation.
- **Bayes decision rule (0/1 loss):** $h_{\text{Bayes}}(x) = \arg\max_y P(Y = y \mid X = x)$ with minimum risk $\mathbb{E}_x[1 - \max_y P(Y = y \mid x)]$.
- **Logistic regression:** $P(y_i \mid x_i, w) = \sigma(y_i w \cdot x_i)$, $\sigma(z) = 1/(1+e^{-z})$; $\sigma(-z) = 1 - \sigma(z)$. MLE minimizes $\sum_i \ln(1 + e^{-y_i w \cdot x_i})$. Separable data drive $\|w\| \to \infty$.
- **ℓ_2-regularized logistic (MAP with $w \sim \mathcal{N}(0, \sigma^2 I)$):** minimize $\frac{\lambda}{2}\|w\|_2^2 + \sum_i \ln(1 + e^{-y_i w \cdot x_i})$ with $\lambda = 1/\sigma^2$.
- **Linear regression:** assume $Y_i \mid x_i, w \sim \mathcal{N}(w \cdot x_i, \eta^2)$; MLE $\Leftrightarrow$ minimize $\sum_i (w \cdot x_i - y_i)^2$. Prediction is $w \cdot x$.
- **Ridge regression (Gaussian prior):** minimize $\frac{1}{2}\|w\|_2^2 + C\sum_i (w \cdot x_i - y_i)^2$ with $C = \sigma^2/(2\eta^2)$.
- **Unified template:** $\min_w R(w) + C\sum_i L(w \cdot x_i, y_i)$. Examples: SVM ($R = \frac{1}{2}\|w\|^2$, $L = \max(0, 1 - y\hat{y})$), Perceptron ($R = 0$, $L = \max(0, -y\hat{y})$), Logistic Regression ($R = \frac{1}{2}\|w\|^2$, $L = \ln(1 + e^{-y\hat{y}})$), Linear Regression ($R = 0$, $L = (y - \hat{y})^2$), Ridge ($R = \frac{1}{2}\|w\|^2$, $L = (y - \hat{y})^2$), Lasso ($R = \lambda \sum_j |w_j|$, $L = (y - \hat{y})^2$).