DegreeLive

Bachelor of Commerce (B2001) | Level 3 • Part B Major | Monash University

Introduction to machine learning

Course Code: ETC3250 • Module 4 Notes

Module 4: Tree-Based Ensembles, Neural Networks & Unsupervised Learning

4.1 Decision Trees (CART): Recursive Binary Splitting and Pruning

Classification and Regression Trees (CART; Breiman et al., 1984) partition the p-dimensional predictor space R^p into M disjoint rectangular regions (terminal nodes or leaves) $R_1, R_2, \ldots, R_M$ and assign a simple constant prediction within each region. Because evaluating all possible multi-way partitions is combinatorially intractable, CART proceeds greedily via **recursive binary splitting**: starting from the root node containing all n observations, at each internal node the algorithm chooses a single predictor X_j and split threshold s creating two half-planes $R_1(j,s) = \{X \mid X_j < s\}$ and $R_2(j,s) = \{X \mid X_j \geq s\}$ that minimise impurity or squared error.

Regression Trees

For a numerical response Y, the predicted value in region R_m is the sample mean $\hat{y}_{R_m} = \frac{1}{|R_m|}\sum_{i \in R_m} y_i$. At each split, CART selects (j,s) to minimise the combined Residual Sum of Squares:

$$\mathrm{RSS}_{\text{split}}(j,s) = \sum_{i:x_i \in R_1(j,s)} (y_i - \bar{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \bar{y}_{R_2})^2$$

Where $\bar{y}_{R_1}$ and $\bar{y}_{R_2}$ are the sample means of Y inside the left and right child regions.

Classification Trees and Impurity Measures

For a K-class response, let $\hat{p}_{mk} = \frac{1}{n_m}\sum_{x_i \in R_m} \mathrm{I}(y_i = k)$ denote the proportion of class k observations in node m of size n_m. Observations in node m are predicted as the majority class $\arg\max_k \hat{p}_{mk}$. Because raw misclassification error $E_m = 1 - \max_k \hat{p}_{mk}$ is insensitive to improvements in node purity that fail to change the majority label, CART splits nodes using one of two strictly concave, differentiable impurity criteria:

$$G_m = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) = 1 - \sum_{k=1}^{K} \hat{p}_{mk}^2, \qquad D_m = -\sum_{k=1}^{K} \hat{p}_{mk} \ln(\hat{p}_{mk})$$

Where:

- G_m is the **Gini Index** (measuring expected misclassification rate if labels are randomly drawn according to $\hat{p}_{mk}$).
- D_m is **Shannon Entropy** (or cross-entropy / deviance; with convention $0 \ln 0 = 0$). Both G_m and D_m vanish when a node is completely pure ($\hat{p}_{mk} = 1$ for one class) and peak when all K classes occur in equal proportions $1/K$.

The weighted impurity reduction (Information Gain when using Entropy) of splitting parent node m into left child L (proportion n_L/n_m) and right child R (proportion n_R/n_m) is $\Delta I = I(m) - \frac{n_L}{n_m} I(L) - \frac{n_R}{n_m} I(R)$.

Cost-Complexity Pruning

Unconstrained trees grown until every leaf is pure overfit noise severely (low bias, extreme variance). Stopping early via a greedy minimum-gain rule risks missing valuable splits hidden beneath a temporarily weak first split. Instead, CART grows a large tree T_0 and applies **weakest-link cost-complexity pruning** to select the subtree $T \subset T_0$ minimising:

$$C_\alpha(T) = \sum_{m=1}^{|T|} n_m Q_m(T) + \alpha |T|$$

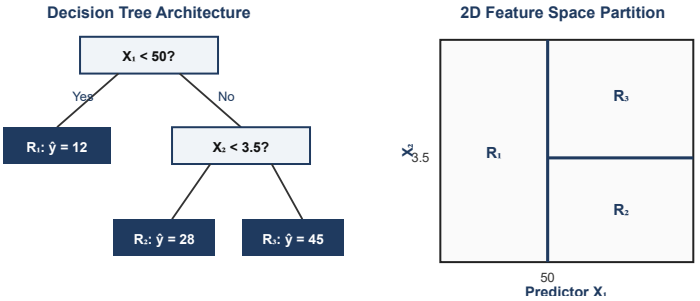
Where:

- $|T|$ is the number of terminal leaves of subtree T.
- $Q_m(T)$ is the node loss (mean squared error or Gini/misclassification rate) in leaf m.
- $\alpha \geq 0$ is the complexity penalty per additional leaf, tuned via k-fold cross-validation.

Degreelive

Categorical Splits and Missing Values in CART

Unlike linear models, decision trees handle unordered categorical predictors (such as Australian postcode zone or industry sector) without arbitrary numeric encoding: a categorical predictor with L levels admits $2^{L-1}-1$ binary subset partitions (for regression or binary classification, sorting levels by average response or class-1 proportion reduces search to $L-1$ ordered thresholds). Furthermore, CART handles missing feature values natively via **surrogate splits**: when the primary split variable X_j is missing for an incoming observation, the tree routes it down the branch predicted by the highest-agreeing alternative predictor recorded during training.



Worked Example 4.1: Gini Impurity Reduction for a Credit Split

A credit scoring node contains $n_m = 20$ borrowers: 8 defaults ($Y=1$) and 12 non-defaults ($Y=0$). A candidate split on Debt-Service Ratio creates a left child L with $n_L = 10$ borrowers (7 defaults, 3 non-defaults) and a right child R with $n_R = 10$ borrowers (1 default, 9 non-defaults). Compute the parent Gini index, child Gini indices, and weighted Gini gain.

1. **Parent Gini Index:** $\hat{p}_{m1} = 8/20 = 0.40, \hat{p}_{m0} = 12/20 = 0.60$.

$$G_m = 1 - (0.40^2 + 0.60^2) = 1 - (0.16 + 0.36) = 0.48$$

2. **Child Gini Indices:**
   - Left child L: $\hat{p}_{L1} = 0.70, \hat{p}_{L0} = 0.30 \implies G_L = 1 - (0.49 + 0.09) = 0.42$.
   - Right child R: $\hat{p}_{R1} = 0.10, \hat{p}_{R0} = 0.90 \implies G_R = 1 - (0.01 + 0.81) = 0.18$.

3. **Weighted Impurity Reduction:**

$$\Delta G = 0.48 - \left(\frac{10}{20}(0.42) + \frac{10}{20}(0.18)\right) = 0.48 - (0.21 + 0.09) = 0.18$$

The split reduces node impurity by 0.18 (a 37.5% relative reduction).

Worked Example 4.1A: Cost-Complexity Pruning Decision

Suppose a regression tree T_1 has $|T_1| = 5$ leaves and total training residual sum of squares $\sum_{m=1}^{5} n_m Q_m(T_1) = 120.0$. Pruning one weak internal branch produces a smaller subtree T_2 with $|T_2| = 3$ leaves and slightly higher training residual sum of squares $\sum_{m=1}^{3} n_m Q_m(T_2) = 138.0$. Determine which subtree minimises the cost-complexity objective $C_\alpha(T) = \text{RSS}(T) + \alpha|T|$ when $\alpha = 6.0$ versus $\alpha = 12.0$:

- **Critical Transition Penalty:** Merging the two leaves increases RSS by $138.0 - 120.0 = 18.0$ while saving $5 - 3 = 2$ leaves, yielding critical threshold $\alpha^* = 18.0/2 = 9.0$.
- **At $\alpha = 6.0 < 9.0$:** $C_6(T_1) = 120 + 6(5) = 150.0 < C_6(T_2) = 138 + 6(3) = 156.0$, so CART retains T_1.
- **At $\alpha = 12.0 > 9.0$:** $C_{12}(T_1) = 120 + 12(5) = 180.0 > C_{12}(T_2) = 138 + 12(3) = 174.0$, so CART prunes the branch and selects the simpler subtree T_2.

4.2 Bagging and Random Forests

Single decision trees suffer from notoriously high sampling variance: removing a handful of observations can trigger a different root split and alter the entire downstream tree structure. **Bootstrap Aggregation (Bagging)** (Breiman, 1996) reduces this variance by growing B deep, unpruned trees $\hat{f}^{*1}(x), \ldots, \hat{f}^{*B}(x)$ on B separate bootstrap samples drawn with replacement from D, and aggregating

DegreeLive

them via averaging (regression) or majority voting (classification):

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)$$

Where B is the number of bootstrap trees (typically 500 or more; increasing B never overfits). Moreover, because each tree leaves roughly 36.8% of observations **out-of-bag (OOB)**, predicting each observation i using only the subset of trees for which i was OOB yields a fast, virtually unbiased **OOB error** estimate closely matching n-fold cross-validation at zero extra fitting cost.

Decorrelating Trees: Random Forests

If one dominant predictor (such as household income) strongly drives the response, nearly every bagged tree places that same predictor at the root split, making the B trees strongly positively correlated. Suppose B identically distributed tree predictions each have variance σ^2 and pairwise correlation $\rho \in [0,1]$; the variance of their average is:

$$\text{Var}\left(\frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)\right) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

Where:

- σ^2 is the prediction variance of a single unpruned bootstrap tree.
- ρ is the pairwise correlation between predictions of two different bootstrap trees.
- B is the number of trees in the ensemble.

Even as $B \to \infty$, the second term vanishes while the first term $\rho\sigma^2$ remains as a floor. **Random Forests** (Breiman, 2001) lower ρ substantially by injecting randomness at every node split: before choosing the best split, only a random subset of $m \leq p$ predictors is considered as split candidates (typically $m \approx \lfloor \sqrt{p} \rfloor$ for classification and $m \approx \lfloor p/3 \rfloor$ for regression). Forcing splits onto secondary predictors decorrelates the trees, pushing $\rho\sigma^2$ sharply downward while preserving low bias.

4.3 Gradient Boosting and XGBoost

Whereas Bagging and Random Forests build deep trees independently in parallel to shrink *variance*, **Boosting** grows shallow trees (often stumps or depth $d \in [1,6]$) *sequentially*, where each new tree corrects the residual mistakes of the current ensemble, progressively eliminating *bias*.

Gradient Boosting Machines (GBM)

Given a differentiable loss function $L(y, F(x))$ (such as squared error $\frac{1}{2}(y - F(x))^2$ or binary log-loss), Gradient Boosting constructs an additive expansion $F_M(x) = F_0(x) + \sum_{m=1}^{M}\eta\, h_m(x)$. At boosting iteration m, the algorithm computes **pseudo-residuals** equal to the negative gradient of the loss evaluated at the previous ensemble's predictions $F_{m-1}(x_i)$:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x_i)=F_{m-1}(x_i)}, \qquad i = 1, \ldots, n$$

A shallow tree $h_m(x)$ is fitted to predict r_{im}, and the ensemble updates via $F_m(x) = F_{m-1}(x) + \eta\, h_m(x)$, where $\eta \in (0,1]$ (typically 0.01 to 0.10) is the **learning rate (shrinkage)** preventing any single tree from overfitting. Notice that unlike Random Forests, increasing the number of boosting rounds M excessively *can* overfit noisy datasets, so M must be tuned jointly with η and interaction depth d via cross-validation or early stopping.

Extreme Gradient Boosting (XGBoost)

XGBoost (Chen and Guestrin, 2016) accelerates and regularises gradient boosting via a second-order Taylor expansion of the loss at iteration m combined with explicit penalties on tree complexity:

DegreeLive

$$\mathrm{L}^{(m)} \approx \sum_{i=1}^{n} [g_i h_m(x_i) + \frac{1}{2} h_i h_m(x_i)^2] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

Where:

- $g_i = \partial_{\hat{y}^{(m-1)}} L(y_i, \hat{y}_i^{(m-1)})$ and $h_i = \partial^2_{\hat{y}^{(m-1)}} L(y_i, \hat{y}_i^{(m-1)})$ are the first- and second-order loss gradients (for squared error $\frac{1}{2}(y_i - \hat{y}_i)^2$, $g_i = \hat{y}_i^{(m-1)} - y_i$ and $h_i = 1$).
- T is the number of terminal leaves in tree h_m and w_j is the prediction weight in leaf j.
- $\gamma \geq 0$ penalises adding extra leaves and $\lambda \geq 0$ imposes L_2 shrinkage on leaf weights.

Minimising $\mathrm{L}^{(m)}$ analytically for a fixed tree structure yields the closed-form optimal leaf weight w_j^* and corresponding split gain score:

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}, \qquad \text{Gain} = \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}\right] - \gamma$$

Where I_j indexes observations falling in leaf j, $G_L = \sum_{i \in I_L} g_i$, $H_L = \sum_{i \in I_L} h_i$ (and analogously for right child R). If $\text{Gain} < 0$, XGBoost automatically prunes the split.

Ensemble Property	Bagging	Random Forest	Gradient Boosting / XGBoost
Tree Construction	Independent, parallel bootstrap	Parallel bootstrap + random m features	Sequential additive error correction
Typical Tree Depth	Deep, unpruned	Deep, unpruned	Shallow (depth 1–6)
Primary Error Target	Variance reduction	Variance reduction via decorrelation	Bias reduction (plus regularisation)
Overfitting Risk as Trees Grow	Does not overfit in B	Does not overfit in B	Overfits if M too large without tuning η

Worked Example 4.2: Bagging Variance and XGBoost Leaf Weight

Part A (Bagging vs. Random Forest): Suppose individual unpruned trees have variance $\sigma^2 = 16$ and $B = 100$ trees are averaged. Under Bagging, inter-tree correlation is $\rho = 0.70$; under Random Forest feature subsampling, correlation drops to $\rho = 0.25$. Compare ensemble variances:

- Bagging variance: $0.70(16) + \frac{0.30}{100}(16) = 11.20 + 0.048 = 11.248$.
- Random Forest variance: $0.25(16) + \frac{0.75}{100}(16) = 4.00 + 0.120 = 4.120$ (a 63.4% reduction).

Part B (XGBoost Leaf Weight & Gain): In a regression node with 4 observations under squared error ($h_i = 1$), current first-order gradients are $g = (-3, -1, +2, +2)$ (so $G = 0, H = 4$). A candidate split places observations 1 and 2 in left child L ($G_L = -4, H_L = 2$) and observations 3 and 4 in right child R ($G_R = +4, H_R = 2$), using regularisation $\lambda = 2$ and $\gamma = 1.0$. Compute optimal leaf weights and split gain:

$$w_L^* = -\frac{-4}{2+2} = +1.0, \qquad w_R^* = -\frac{+4}{2+2} = -1.0$$

$$\text{Gain} = \frac{1}{2}\left[\frac{(-4)^2}{2+2} + \frac{4^2}{2+2} - \frac{0^2}{4+2}\right] - 1.0 = \frac{1}{2}[4.0 + 4.0 - 0] - 1.0 = 4.0 - 1.0 = +3.0$$

Because $\text{Gain} = +3.0 > 0$, XGBoost accepts the split.

DegreeLive

4.4 Explainable Artificial Intelligence (XAI) for Black-Box Ensembles

Tree ensembles and neural networks deliver state-of-the-art predictive accuracy on complex tabular data, yet lack the transparent slope coefficients of linear regression or GAMs. Within Australian financial services (governed by APRA prudential expectations) and regulated credit scoring, practitioners rely on **Explainable AI (XAI)** tools to audit model logic globally and locally:

- **Permutation Variable Importance:** Measures how much out-of-sample (or OOB) prediction error increases when the values of predictor X_j are randomly shuffled across observations, breaking its relationship with Y while preserving marginal distributions. Unlike impurity-based importance, permutation importance is unbiased against high-cardinality continuous features.
- **Partial Dependence Plots (PDP) & Individual Conditional Expectation (ICE):** A PDP estimates the average marginal effect of predictor subset X_S over the joint empirical distribution of remaining predictors X_C:

  $$\hat{f}_S(x_S) = \frac{1}{n}\sum_{i=1}^{n} \hat{f}(x_S, x_{iC})$$

  Plotting individual curves $\hat{f}(x_S, x_{iC})$ per observation produces **ICE curves**, revealing heterogeneous interactions masked by the PDP average.
- **SHAP (SHapley Additive exPlanations):** Grounded in cooperative game theory, Shapley values decompose an individual prediction $\hat{f}(x_i)$ exactly into baseline average $\phi_0 = \mathrm{E}[\hat{f}(X)]$ plus additive feature contributions $\sum_{j=1}^{p} \phi_j(x_i) = \hat{f}(x_i) - \phi_0$, uniquely satisfying local accuracy, missingness, and consistency axioms:

  $$\phi_j(x_i) = \sum_{S \subseteq \mathrm{F} \setminus \{j\}} \frac{|S|!\,(p - |S| - 1)!}{p!} \left[v(S \cup \{j\}) - v(S)\right]$$

  Where F is the set of all p features and $v(S) = \mathrm{E}[\hat{f}(X) \mid X_S = x_{iS}]$ is the expected prediction conditional on knowing only feature subset S.

Worked Example 4.2A: Exact Two-Feature Shapley Decomposition

Suppose a credit-scoring Random Forest predicts default probability $\hat{f}(x)$ using two features: Debt-to-Income (X_1) and Arrears History (X_2). For a specific applicant x^*, the unconditional baseline average is $v(\emptyset) = \phi_0 = 0.10$, the marginal expectations when only one feature is fixed to x^* are $v(\{1\}) = 0.18$ and $v(\{2\}) = 0.24$, and the full model prediction is $v(\{1,2\}) = \hat{f}(x^*) = 0.40$. Compute the exact SHAP values ϕ_1 and ϕ_2:

1. **Shapley Contribution of X_1 ($p = 2$):** With two subsets $S = \emptyset$ (weight $0!1!/2! = 0.5$) and $S = \{2\}$ (weight $1!0!/2! = 0.5$):

   $$\phi_1 = 0.5\,[v(\{1\}) - v(\emptyset)] + 0.5\,[v(\{1,2\}) - v(\{2\})] = 0.5(0.18 - 0.10) + 0.5(0.40 - 0.24) = 0.04 + 0.08 = +0.12$$

2. **Shapley Contribution of X_2:**

   $$\phi_2 = 0.5\,[v(\{2\}) - v(\emptyset)] + 0.5\,[v(\{1,2\}) - v(\{1\})] = 0.5(0.24 - 0.10) + 0.5(0.40 - 0.18) = 0.07 + 0.11 = +0.18$$

3. **Local Accuracy Check:** $\phi_0 + \phi_1 + \phi_2 = 0.10 + 0.12 + 0.18 = 0.40 = \hat{f}(x^*)$, cleanly attributing 40% of the excess risk above baseline to Debt-to-Income and 60% to Arrears History.

4.5 Multi-Layer Perceptrons (MLPs) and Backpropagation

Artificial Neural Networks (ANNs) construct flexible nonlinear mappings by composing layers of linear affine transformations followed by element-wise nonlinear **activation functions**. In a standard feedforward **Multi-Layer Perceptron (MLP)** with input $x \in \mathrm{R}^p$ and H hidden units in layer 1:

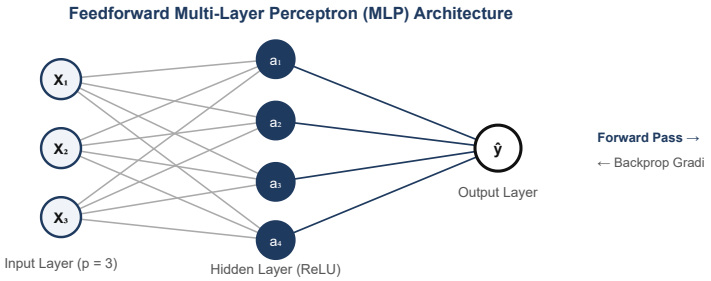
DegreeLive

$$z^{(1)} = W^{(1)}x + b^{(1)}, \qquad a^{(1)} = \phi(z^{(1)}), \qquad \hat{y} = g(W^{(2)}a^{(1)} + b^{(2)})$$

Where:

- $W^{(1)} \in \mathrm{R}^{H \times p}$ and $b^{(1)} \in \mathrm{R}^{H}$ are the hidden-layer weight matrix and bias vector.
- $\phi(\cdot)$ is a nonlinear hidden activation function (without nonlinearity, stacking linear layers collapses to a single linear matrix).
- $g(\cdot)$ is the output activation (identity for regression, sigmoid for binary classification, softmax for K-class probabilities).

Activation Function	Formula $\phi(z)$	Range	Primary Role & Gradient Behaviour
Sigmoid	$\frac{1}{1+e^{-z}}$	$(0, 1)$	Binary output probability; saturates in hidden layers (vanishing gradients)
Hyperbolic Tangent (tanh)	$\frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$	$(-1, 1)$	Zero-centred hidden activation; still saturates at large $\lvert z \rvert$
ReLU (Rectified Linear Unit)	$\max(0, z)$	$[0, \infty)$	Default hidden activation; constant derivative 1 for $z > 0$, avoids vanishing gradient



Training via Backpropagation, SGD, and Regularisation

Neural networks possess millions of parameters and non-convex loss landscapes $\mathrm{L}(\theta)$. Training proceeds iteratively via **Stochastic Gradient Descent (SGD)** (or adaptive optimisers such as **Adam**) on mini-batches of size B: $\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_{\theta} \mathrm{L}_{\text{batch}}(\theta^{(t)})$. The exact parameter gradients are computed backward from the output loss to the earliest layers via the multivariate chain rule—**backpropagation**:

$$\frac{\partial \mathrm{L}}{\partial W_{kj}^{(1)}} = \frac{\partial \mathrm{L}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial a_k^{(1)}} \cdot \phi'(z_k^{(1)}) \cdot x_j$$

Where $\phi'(z_k^{(1)})$ is the derivative of the hidden activation evaluated at pre-activation $z_k^{(1)}$. To prevent overfitting in over-parametrised networks, practitioners enforce **weight decay** (an L_2 penalty on weights), **Dropout** (randomly zeroing each hidden unit with probability p_{drop} during each training step so units cannot co-adapt), and **early stopping** monitored on a validation split.

Universal Approximation, Adaptive Optimisers, and Vanishing Gradients

The **Universal Approximation Theorem** guarantees that even a single hidden-layer MLP with sufficiently many nonlinear neurons can approximate any continuous function on a compact domain to arbitrary precision. In practice, deep architectures (multiple narrower layers) learn hierarchical feature representations far more parameter-efficiently than a single astronomically wide layer. However, stacking many sigmoid or tanh layers multiplies derivatives bounded below 1 (at most 0.25 for sigmoid), causing gradients in early layers to decay exponentially toward zero—the **vanishing gradient problem**. Modern networks mitigate this via ReLU activations, Xavier/He variance-preserving weight initialisation, **Batch Normalisation** (standardising layer pre-activations across each mini-batch), and adaptive momentum optimisers such as **Adam**, which maintains exponentially decaying averages of both the first moment m_t (momentum) and second raw moment v_t (per-parameter learning rate scaling) of mini-batch gradients:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \qquad v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \qquad \theta_t = \theta_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Where $g_t = \nabla_\theta \mathrm{L}_{\text{batch}}(\theta_{t-1})$ is the stochastic gradient at step t, $\hat{m}_t = m_t/(1-\beta_1^t)$ and $\hat{v}_t = v_t/(1-\beta_2^t)$ are bias-corrected moments (typically $\beta_1 = 0.9, \beta_2 = 0.999$), η is the base step size, and $\epsilon \approx 10^{-8}$ ensures numerical stability.

Worked Example 4.3: One Forward and Backpropagation Step

Consider a minimal regression MLP with 1 input $x = 2.0$, target $y = 5.0$, 1 ReLU hidden unit (initial weight $w_1 = 1.5$, bias $b_1 = 0$), linear output weight $w_2 = 2.0$ ($b_2 = 0$), loss $L = \frac{1}{2}(y - \hat{y})^2$, and learning rate $\eta = 0.05$.

1. **Forward Pass:** $z_1 = w_1 x + b_1 = 1.5(2.0) = 3.0$. Since $z_1 > 0$, ReLU yields $a_1 = \max(0, 3.0) = 3.0$. Output prediction: $\hat{y} = w_2 a_1 = 2.0(3.0) = 6.0$. Loss: $L = \frac{1}{2}(5.0 - 6.0)^2 = 0.50$.
2. **Backpropagation Gradients:** Error derivative at output: $\frac{\partial L}{\partial \hat{y}} = \hat{y} - y = 6.0 - 5.0 = +1.0$.
   - Output weight gradient: $\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \hat{y}} \cdot a_1 = 1.0(3.0) = 3.0$.
   - Hidden pre-activation gradient (ReLU derivative $\phi'(3.0) = 1$): $\frac{\partial L}{\partial z_1} = 1.0 \cdot w_2 \cdot 1 = 2.0$.
   - Input weight gradient: $\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial z_1} \cdot x = 2.0(2.0) = 4.0$.
3. **Gradient Descent Update ($\eta = 0.05$):**

$$w_2^{\text{new}} = 2.0 - 0.05(3.0) = 1.85, \qquad w_1^{\text{new}} = 1.5 - 0.05(4.0) = 1.30$$

Verifying the updated prediction: $\hat{y}^{\text{new}} = 1.85 \times (1.30 \times 2.0) = 4.81$, reducing loss sharply from 0.50 to $\frac{1}{2}(5.0 - 4.81)^2 = 0.01805$.

4.6 Unsupervised Learning: Principal Component Analysis (PCA)

Unlike supervised learning, **unsupervised learning** observes only the predictor matrix $X \in \mathrm{R}^{n \times p}$ with no target label Y, aiming to discover low-dimensional latent structure or natural groupings. **Principal Component Analysis (PCA)** constructs a small sequence of uncorrelated linear combinations of the original p standardised features that capture maximum variance.

Assuming each column of X has been centred to mean zero (and standardised to unit variance whenever variables carry different units, such as ASX market capitalisation in billions versus dividend yield in percent), the first principal component loading vector $\phi_1 = (\phi_{11}, \ldots, \phi_{p1})^\top$ solves:

$$\max_{\phi_1 \in \mathrm{R}^p} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{j1} x_{ij} \right)^2 \right\} = \max_{\|\phi_1\|_2 = 1} \phi_1^\top S \phi_1$$

Where:

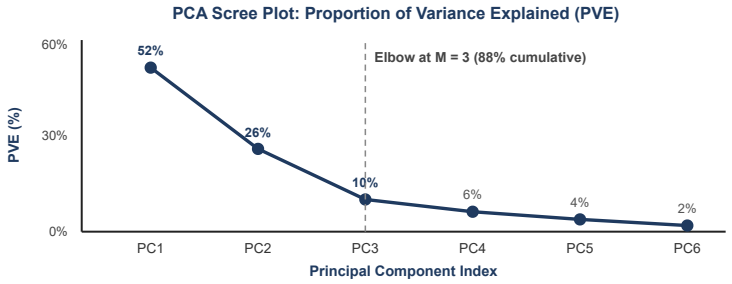
- $S = \frac{1}{n} X^\top X$ (or $\frac{1}{n-1} X^\top X$) is the sample covariance (or correlation) matrix of standardised features.
- $\|\phi_1\|_2 = 1$ is the unit-length normalisation constraint ensuring the variance cannot grow artificially by scaling loadings.
- $z_{i1} = \sum_{j=1}^{p} \phi_{j1} x_{ij}$ are the first principal component **scores**.

By the spectral theorem, the optimal loading vector ϕ_1 is the eigenvector corresponding to the largest eigenvalue λ_1 of S (equivalently the first right singular vector of X via Singular Value Decomposition $X = UDV^\top$), and the variance of Z_1 equals λ_1. Subsequent components $\phi_2, \ldots, \phi_p$ maximise remaining variance subject to orthogonality $\phi_k^\top \phi_l = 0$ ($k \neq l$). The **Proportion of Variance Explained (PVE)** by component m is:

$$\mathrm{PVE}_m = \frac{\lambda_m}{\sum_{j=1}^{p} \lambda_j} = \frac{\lambda_m}{\mathrm{tr}(S)}$$

Analysts plot cumulative PVE or individual eigenvalues on a **scree plot** to locate an "elbow" choosing $M \ll p$ components, and visualise both observations and variable loading arrows simultaneously on a **biplot**. Crucially, PCA is *not* scale-invariant: if predictors are left unstandardised, variables measured in larger units (such as revenue in AUD versus interest rates in percent) carry artificially huge variance and dominate the first principal component. Unless all predictors share identical units and comparable dispersion, PCA must be performed on standardised variables (eigendecomposition of the correlation matrix R).

Degree.live



Worked Example 4.4: Exact 2D PCA Eigendecomposition and PVE

Consider two standardised financial indicators across n ASX companies with sample correlation matrix:

$$R = \begin{pmatrix} 1.0 & 0.8 \\ 0.8 & 1.0 \end{pmatrix}$$

Find the eigenvalues, unit loading eigenvectors, and PVE of each principal component.

1. **Characteristic Equation** $\det(R - \lambda I_2) = 0$**:**

$$(1-\lambda)^2 - 0.8^2 = 0 \implies 1 - \lambda = \pm 0.8 \implies \lambda_1 = 1.8, \quad \lambda_2 = 0.2$$

   Check trace: $\lambda_1 + \lambda_2 = 1.8 + 0.2 = 2.0 = \text{tr}(R)$.

2. **Proportion of Variance Explained:**

$$\text{PVE}_1 = \frac{1.8}{2.0} = 0.90 \ (90\%), \qquad \text{PVE}_2 = \frac{0.2}{2.0} = 0.10 \ (10\%)$$

3. **Unit Loading Vectors:** For $\lambda_1 = 1.8$, solving $(R - 1.8I)\phi_1 = 0$ gives $-0.8\phi_{11} + 0.8\phi_{21} = 0 \implies \phi_{11} = \phi_{21}$. Normalising to unit length yields $\phi_1 = (1/\sqrt{2},\ 1/\sqrt{2})^\top \approx (0.707, 0.707)^\top$ (overall size factor). For $\lambda_2 = 0.2$, $\phi_2 = (1/\sqrt{2},\ -1/\sqrt{2})^\top$ (contrast factor).

Principal Component Regression (PCR) vs. Partial Least Squares (PLS)

When PCA scores $Z_1, \ldots, Z_M$ (with $M \ll p$) replace the raw p predictors in an OLS regression, the resulting two-stage procedure is **Principal Component Regression (PCR)**. PCR eliminates multicollinearity and mitigates high-dimensional overfitting whenever the directions in X carrying the highest variance align with the directions predictive of Y. However, because PCA is completely unsupervised, it ignores Y when constructing components; if the true signal resides along a low-variance minor component, PCR can discard critical predictive information. **Partial Least Squares (PLS)** resolves this limitation by computing orthogonal linear combinations $Z_m = \sum_{j=1}^{p} \phi_{jm} X_j$ iteratively in a *supervised* manner, weighting each predictor in proportion to its sample covariance with the response Y (after residualising against prior components).

4.7 Clustering Methods: K-Means, Hierarchical, and Model-Based Clustering

Clustering algorithms partition n unlabelled observations into homogeneous subgroups (clusters) such that observations within a cluster are highly similar while observations across clusters are dissimilar.

K-Means Clustering

Given a pre-specified number of clusters K, K**-means** partitions observations into disjoint sets $C_1, \ldots, C_K$ to minimise total **Within-Cluster Sum of Squares (WCSS)**:

DegreeLive

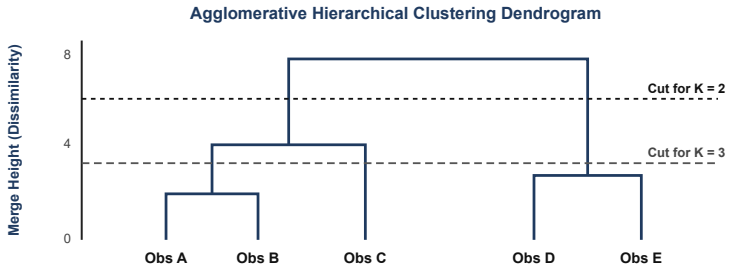
$$\min_{C_1,\ldots,C_K} \left\{ \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|_2^2 \right\}, \qquad \mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

Where $\mu_k \in \mathrm{R}^p$ is the centroid vector of cluster k. Lloyd's iterative algorithm alternates two steps until cluster assignments stop changing: (1) assign each observation x_i to the nearest centroid $\arg\min_k \|x_i - \mu_k\|_2^2$, and (2) recompute each centroid μ_k as the mean of points currently assigned to C_k. Because WCSS decreases monotonically at each step yet the objective is non-convex, practitioners run multiple random initialisations (or K**-Means++**, spreading initial centres apart) and choose K using the elbow plot or **Silhouette Score** $s(i) = \frac{b(i)-a(i)}{\max(a(i),b(i))} \in [-1,1]$, where $a(i)$ is mean intra-cluster distance and $b(i)$ is mean distance to the nearest neighbouring cluster.

Hierarchical Clustering

Agglomerative hierarchical clustering avoids committing to a fixed K upfront, building a tree-like **dendrogram** bottom-up: starting with n singleton clusters, at each step it merges the two closest clusters according to a chosen **linkage criterion** until all observations unite in a single cluster.

Linkage Criterion	Inter-Cluster Dissimilarity Definition	Geometric Behaviour
Complete Linkage	Maximum pairwise distance between clusters A and B	Compact, roughly spherical clusters; balanced dendrogram
Single Linkage	Minimum pairwise distance between clusters A and B	Prone to elongated "chaining" along noisy bridges
Average Linkage	Mean of all $\|A\| \cdot \|B\|$ pairwise distances	Balanced compromise between single and complete
Ward's Method	Increase in total WCSS produced by merging A and B	Directly mimics K-means objective; compact equal-variance clusters



Model-Based Clustering (Gaussian Mixture Models)

Hard spherical partitions from K-means fail when clusters have elongated, overlapping elliptical shapes or varying sizes. **Model-based clustering** treats the data as arising from a probabilistic **Gaussian Mixture Model (GMM)** with K components:

$$p(x) = \sum_{k=1}^{K} \pi_k \, \mathrm{N}_p(x \mid \mu_k, \Sigma_k)$$

Where $\pi_k \geq 0$ ($\sum_{k=1}^{K} \pi_k = 1$) are mixing proportions, μ_k are component means, and Σ_k are component covariance matrices. Parameters are estimated via the **Expectation–Maximisation (EM)** algorithm, which computes soft posterior cluster responsibilities $\gamma_{ik} = P(Z_i = k \mid x_i)$ in the E-step and updates weighted means, covariances, and mixing weights in the M-step. Model selection across both K and covariance structures (spherical, diagonal, or full ellipsoidal) is guided objectively by the Bayesian Information Criterion (BIC).

Worked Example 4.5: One Iteration of K-Means Clustering

Four Australian retail suburbs are described by standardised median income and population density: $A = (1,1)$, $B = (2,1)$, $C = (5,4)$, $D = (6,5)$. Run one complete K-means iteration with $K = 2$ starting from initial centroids $\mu_1^{(0)} = (1,2)$ and $\mu_2^{(0)} = (5,2)$.

1. **Assignment Step (Squared Euclidean Distances):**
   - Point $A(1,1)$: $d^2(A,\mu_1) = 0^2 + (-1)^2 = 1, d^2(A,\mu_2) = 16 + 1 = 17 \implies C_1$.

- Point $B(2,1)$: $d^2(B,\mu_1) = 1+1 = 2, d^2(B,\mu_2) = 9+1 = 10 \implies C_1$.
- Point $C(5,4)$: $d^2(C,\mu_1) = 16+4 = 20, d^2(C,\mu_2) = 0+4 = 4 \implies C_2$.
- Point $D(6,5)$: $d^2(D,\mu_1) = 25+9 = 34, d^2(D,\mu_2) = 1+9 = 10 \implies C_2$.

2. **Centroid Update Step:**

$$\mu_1^{(1)} = (\frac{1+2}{2},\ \frac{1+1}{2}) = (1.5,\ 1.0), \qquad \mu_2^{(1)} = (\frac{5+6}{2},\ \frac{4+5}{2}) = (5.5,\ 4.5)$$

Evaluating total WCSS at the updated centroids gives $[0.25 + 0.25] + [0.25 + 0.25] = 1.00$, and re-checking distances confirms convergence with zero further reassignments.

Worked Example 4.6: Silhouette Score and GMM Posterior Responsibility

Part A (Silhouette Validation): Using the converged 2-cluster solution from Example 4.5 ($C_1 = \{A(1,1), B(2,1)\}$ and $C_2 = \{C(5,4), D(6,5)\}$), compute the Silhouette Score $s(B)$ for suburb $B(2,1)$:

- Intra-cluster distance $a(B)$ to cluster peer $A(1,1)$: $a(B) = \sqrt{(2-1)^2 + (1-1)^2} = 1.00$.
- Mean distance $b(B)$ to nearest neighbouring cluster C_2: $d(B,C) = \sqrt{(5-2)^2 + (4-1)^2} = \sqrt{18} \approx 4.243$ and $d(B,D) = \sqrt{(6-2)^2 + (5-1)^2} = \sqrt{32} \approx 5.657$, giving $b(B) = (4.243 + 5.657)/2 = 4.950$.
- Silhouette Score:

$$s(B) = \frac{4.950 - 1.00}{\max(1.00, 4.950)} = \frac{3.950}{4.950} \approx +0.798$$

A value near $+1$ confirms that suburb B is well matched to cluster C_1 and cleanly separated from C_2.

Part B (GMM E-Step Soft Responsibility): Suppose a univariate two-component Gaussian Mixture Model has equal mixing weights $\pi_1 = \pi_2 = 0.5$ and current densities at observation x_i of $\mathrm{N}(x_i \mid \mu_1, \sigma_1^2) = 0.30$ and $\mathrm{N}(x_i \mid \mu_2, \sigma_2^2) = 0.10$. The E-step computes posterior cluster probability:

$$\gamma_{i1} = \frac{\pi_1 \mathrm{N}(x_i \mid \mu_1, \sigma_1^2)}{\pi_1 \mathrm{N}(x_i \mid \mu_1, \sigma_1^2) + \pi_2 \mathrm{N}(x_i \mid \mu_2, \sigma_2^2)} = \frac{0.5(0.30)}{0.5(0.30) + 0.5(0.10)} = \frac{0.15}{0.20} = 0.75, \qquad \gamma_{i2} = 0.25$$

Unlike K-means' hard $(1,0)$ assignment, the GMM assigns 75% weight to component 1 and 25% weight to component 2 when updating component means and variances in the M-step.

4.8 Synthesis: Method Selection Across Supervised and Unsupervised Tasks

In practical Australian business and economic analytics, choosing an appropriate algorithm requires matching data geometry, sample size, scaling sensitivity, and regulatory explainability requirements:

Algorithm	Learning Paradigm	Feature Scaling Needed?	Handles Missing / Nonlinearities	Key Hyperparameters (R Ecosystem)
CART Decision Tree	Supervised	No (monotone-invariant)	Surrogate splits; axis-aligned steps	Complexity α (`cp`), tree depth (`rpart`)
Random Forest	Supervised Ensemble	No	Strong interaction capture; OOB error	`mtry` (m), `trees` (B), `min_n` (`ranger`)
XGBoost	Supervised Boosting	No	Sparsity-aware split routing; high accuracy	Learning rate η, depth, rounds M, λ, γ (`xgboost`)
MLP Neural Network	Supervised Deep	Yes (mandatory)	Smooth high-order interactions	Hidden units, layers, dropout, weight decay, epochs (`keras/torch`)

DegreeLive

Algorithm	Learning Paradigm	Feature Scaling Needed?	Handles Missing / Nonlinearities	Key Hyperparameters (R Ecosystem)
PCA	Unsupervised Reduction	Yes (unless identical units)	Linear orthogonal projections only	Retained components M via scree / cumulative PVE (`prcomp`)
K-Means / GMM	Unsupervised Clustering	Yes (distance-based)	Spherical (K-Means) vs. elliptical (GMM)	Clusters K, covariance structure via BIC / Silhouette (`kmeans`, `mclust`)

DegreeLive