Supervised Learning Algorithms — Part One

Unit 2: Bias–Variance, Linear Regression, Logistic Regression & LDA

Dr. Gopal Chandra Jana

Dept. of CSE, Sharda University
Machine Learning (CSE473) — Unit 2

July 22, 2026

Course Page:
https://www.gcjana.in/courses/shardauniversity/2601/machine-learning/

Outline

Learning Outcomes of Unit 2

After this unit, a student will be able to...

1. **Explain** how supervised learning algorithms work and the factors affecting them.
2. **Derive** the bias–variance decomposition and use it to diagnose models.
3. **Formulate** linear regression and **solve** it by OLS, the normal equation and gradient descent.
4. **Apply** feature scaling, learning-rate selection and convergence tests.
5. **Derive and compare** Ridge and Lasso regularization.
6. **Explain** the logistic model — sigmoid, logit, odds, odds ratio — and fit it by MLE/IRLS.
7. **Derive** Linear Discriminant Analysis and **contrast** it with logistic regression.

Mapping to the Syllabus

A. How supervised learning works; bias–variance and other factors. **B.** Linear regression, OLS, gradient descent, regularization, prediction. **C.** Logistic regression, model fitting, LDA, applications.

Supervised Learning — Formal Setup

The problem

Given a training sample of m i.i.d. pairs drawn from an unknown distribution $\mathcal{D}$,

$$S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\} \sim \mathcal{D}^m,$$

find $h \in \mathcal{H}$ minimising the **expected risk**

$$R(h) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\big[L(y, h(\mathbf{x}))\big].$$

The catch

$\mathcal{D}$ is **unknown**, so we minimise the **empirical risk**

$$\hat{R}(h) = \frac{1}{m}\sum_{i=1}^{m} L\big(y_i, h(\mathbf{x}_i)\big)$$

and hope it generalises — this is **Empirical Risk Minimisation** (ERM).

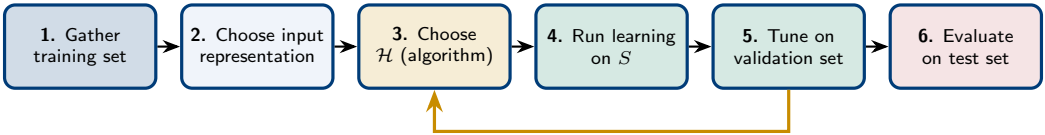
Notation used throughout

$\mathbf{x} \in \mathbb{R}^n$	feature vector
y	target (label)
m	number of examples
n	number of features
$h_{\boldsymbol{\theta}}$	hypothesis with parameters $\boldsymbol{\theta}$
L	loss function
$\mathcal{H}$	hypothesis set

Two output types

$y \in \mathbb{R} \Rightarrow$ **regression** (Part B).
$y \in \{0, 1\} \Rightarrow$ **classification** (Part C).

How a Supervised Learning Algorithm Works — The Steps



Key decisions at each step

- Step 2 fixes the **feature space** — often decisive.
- Step 3 fixes the **inductive bias** of the learner.
- Step 5 selects **hyperparameters**, never on test data.

Five factors that decide success

(i) **bias–variance** trade-off; (ii) **function complexity** vs. data volume; (iii) **dimensionality**; (iv) **noise** in the outputs; (v) the **algorithm** chosen. We take each in turn.

Bias–Variance Trade-off — The Decomposition

Setting

Let $y = f(\mathbf{x}) + \varepsilon$ with $\mathbb{E}[\varepsilon] = 0$, $\mathrm{Var}(\varepsilon) = \sigma^2$. Let $\hat{f}$ be the model fitted on a random training set. For a fixed test point $\mathbf{x}$:

Result

$$\mathbb{E}\big[(y - \hat{f}(\mathbf{x}))^2\big] = \underbrace{\big(\mathbb{E}[\hat{f}(\mathbf{x})] - f(\mathbf{x})\big)^2}_{\text{Bias}^2} + \underbrace{\mathbb{E}\big[(\hat{f}(\mathbf{x}) - \mathbb{E}[\hat{f}(\mathbf{x})])^2\big]}_{\text{Variance}} + \underbrace{\sigma^2}_{\text{irreducible}}$$

Reading it

Bias: error from wrong assumptions (model too rigid).
Variance: sensitivity to the particular training sample.
σ^2: noise — *no* model can beat it.

Consequence

We cannot drive both to zero. Increasing model complexity **lowers bias** but **raises variance**. The best model minimises their *sum*.

Bias–Variance — Proof Sketch and Picture

Derivation (key steps)

Write $\mathbb{E}[\hat{f}] = \bar{f}$. Then

$$\begin{aligned}
\mathbb{E}[(y - \hat{f})^2] &= \mathbb{E}[(f + \varepsilon - \hat{f})^2] \\
&= \mathbb{E}[(f - \hat{f})^2] + \underbrace{2\mathbb{E}[\varepsilon]\mathbb{E}[f - \hat{f}]}_{=0} + \mathbb{E}[\varepsilon^2] \\
&= \mathbb{E}[(f - \bar{f} + \bar{f} - \hat{f})^2] + \sigma^2 \\
&= (f - \bar{f})^2 + \mathbb{E}[(\bar{f} - \hat{f})^2] + \sigma^2
\end{aligned}$$

the cross term vanishing because $\mathbb{E}[\bar{f} - \hat{f}] = 0$.

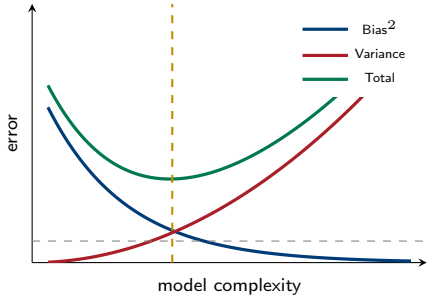


Figure: Total error is minimised where the two curves trade off; the dashed floor is σ^2.

Function Complexity and Amount of Training Data

The trade-off

- **Simple** true function + little noise $\Rightarrow$ an inflexible, high-bias learner with **little data** suffices.
- **Complex** true function (many interactions, non-linearity) $\Rightarrow$ needs a flexible learner *and* **much more data**.

Rule of thumb

Model capacity should **match** both the complexity of the target function and the volume of data available. Flexible model + small data = overfitting.

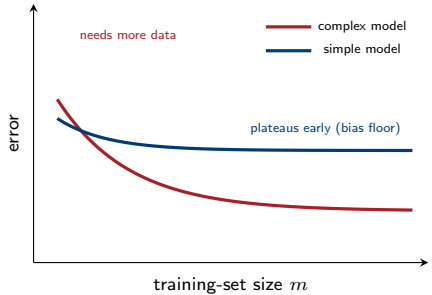


Figure: Learning curves — the complex model wins *only* once enough data is available.

Dimensionality of the Input Space

The curse of dimensionality

As n grows:

- Data becomes **sparse** — to keep the same density, sample size must grow **exponentially**: $m \propto k^n$.
- All pairwise distances become **nearly equal**, so distance-based methods degrade.
- Variance rises; overfitting becomes easy.

Remedies

- **Feature selection** — keep informative features.
- **Dimensionality reduction** — PCA, LDA (Part C), autoencoders.
- **Regularization** — shrink coefficients (Part B).
- Use algorithms robust in high dimensions (linear models, SVM).

A concrete illustration

To cover $[0,1]^n$ with a grid of spacing 0.1 needs 10^n points:

$$n = 1 : 10, \quad n = 3 : 1000, \quad n = 10 : \mathbf{10^{10}}$$

Why linear models survive

A linear model has only $n+1$ parameters — its capacity grows *linearly*, not exponentially, with dimension. This is why they remain strong baselines on wide data.

Noise in the Output Values

Sources of noise

- **Measurement error** in y (faulty sensor, typos).
- **Label noise** — wrong annotations.
- **Missing features**: part of y's variation is simply unexplainable from the available $\mathbf{x}$ (*stochastic* target).

Danger

A flexible learner will happily **fit the noise**. The result: excellent training error, poor test error — overfitting.

Remedies

- **Early stopping** on a validation curve.
- **Detect and remove** outliers before training.
- Prefer **high-bias** models when noise is high.
- Use **robust losses** (absolute/Huber instead of squared).
- **Regularization** and ensembling.

Recall from the decomposition

The noise term σ^2 is **irreducible**: it puts a hard floor under the achievable error. Chasing below it is chasing noise.

Choosing the Algorithm

Algorithm	Bias	Variance	Best suited to
Linear / logistic regression	high	low	linear structure, wide data, interpretability
Regularized linear (Ridge/Lasso)	high	very low	many correlated features
k-NN (small k)	low	high	low dimension, plenty of data
Decision tree (deep)	low	high	interactions, mixed data types
SVM (linear kernel)	high	low	high-dimensional, sparse data
SVM (RBF kernel)	low	medium	non-linear boundaries
Ensembles (RF, boosting)	low	reduced	general-purpose strong baseline
Neural networks	low	high	huge data, complex structure

No Free Lunch theorem

Averaged over *all* possible problems, every algorithm performs equally. Hence there is **no universally best learner** — the choice must match the structure of your data. Always benchmark against a simple baseline.

Other Factors — Heterogeneity, Redundancy, Interactions

Heterogeneity of the data

Features on **different scales/types** (numeric + categorical).

- Distance-based methods (k-NN, SVM, k-means) and gradient descent need **scaling**.
- Categorical variables need **encoding** (one-hot, ordinal).
- **Trees** are naturally immune to monotone rescaling.

Redundancy in the data

Highly **correlated** features (multicollinearity).

- Linear regression becomes **unstable**: $\mathbf{X}^\top \mathbf{X}$ is near-singular, so coefficients swing wildly.
- Fix with **Ridge**, PCA, or by dropping duplicates.

Interactions and non-linearities

The effect of one feature may **depend on another**.

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2$$

- Linear models capture these only if we **add the terms explicitly** (interaction / polynomial features).
- Trees, kernel SVMs and neural nets discover them **automatically**.

Practical implication

If a linear model underfits, first ask: *have I given it the interaction and non-linear terms it cannot invent by itself?*

Quick Check — Round 1 (How Supervised Learning Works)

[Q1]

Write the bias–variance decomposition and name each of its three terms.

[Q2]

Which term is *irreducible*, and what does that mean in practice?

[Q3]

To cover $[0, 1]^n$ with grid spacing 0.1, how many points are needed for $n = 1$, 3 and 6? What phenomenon does this illustrate?

[Q4]

A deep decision tree gives 99% train and 65% test accuracy. Is this high bias or high variance? Give two fixes.

Think for a minute — answers on the next slide.

Answers — Round 1

[A1]

$\mathbb{E}[(y - \hat{f})^2] = \text{Bias}^2 + \text{Variance} + \sigma^2$: **bias** (error from wrong assumptions), **variance** (sensitivity to the training sample), **irreducible noise**.

[A2]

σ^2, the **noise**. No model, however good, can achieve expected error below it — it sets a **hard performance floor**.

[A3]

10^n points: $n = 1 \to$ **10**, $n = 3 \to$ **1000**, $n = 6 \to 10^6$. This is the **curse of dimensionality** — required data grows exponentially with dimension.

[A4]

High variance (overfitting). Fixes: prune / limit depth, more training data, regularization, bagging or random forests, early stopping (any two).

Linear Regression — Model Representation

Simple linear regression ($n = 1$)

$$h_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x$$

θ_0 = intercept, θ_1 = slope.

Multiple linear regression

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n = \sum_{j=0}^{n} \theta_j x_j = \boldsymbol{\theta}^\top \mathbf{x}$$

with the convention $x_0 = 1$ (bias/intercept term).

Matrix form

For m examples, $\mathbf{X} \in \mathbb{R}^{m \times (n+1)}$:

$$\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\theta}$$

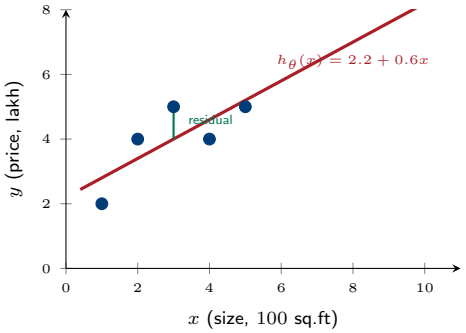


Figure: The fitted line and one residual. This dataset is used throughout Part B.

The Cost Function

Squared-error (least-squares) cost

$$J(\boldsymbol{\theta}) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 = \frac{1}{2m} \|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|^2$$

Why this cost?

- **Convex** in $\boldsymbol{\theta}$ $\Rightarrow$ a unique global minimum, no local traps.
- **Differentiable** $\Rightarrow$ gradient methods apply.
- Equals the **maximum-likelihood** estimate when the noise is Gaussian.

Why the $\frac{1}{2}$?

Purely cosmetic: it cancels the 2 produced when differentiating the square, giving a cleaner gradient. It does not change the minimiser.

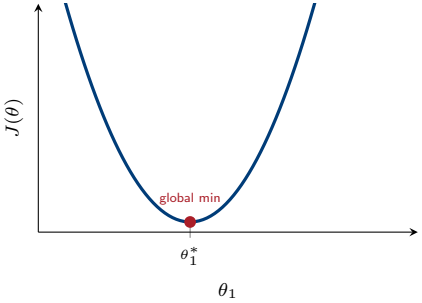


Figure: J is a convex bowl — gradient descent cannot get stuck.

Ordinary Least Squares — Derivation ($n = 1$)

Minimise J by setting the partial derivatives to zero

$$\frac{\partial J}{\partial \theta_0} = 0 \;\Rightarrow\; \sum_i (\theta_0 + \theta_1 x_i - y_i) = 0, \qquad \frac{\partial J}{\partial \theta_1} = 0 \;\Rightarrow\; \sum_i (\theta_0 + \theta_1 x_i - y_i) x_i = 0$$

These are the **normal equations**. Solving them gives:

Closed-form solution

$$\theta_1 = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2} = \frac{S_{xy}}{S_{xx}}$$

$$\theta_0 = \bar{y} - \theta_1 \bar{x}$$

Useful identities

$$\theta_1 = r_{xy} \frac{s_y}{s_x}, \qquad S_{xy} = \sum x_i y_i - n \bar{x} \bar{y}$$

The fitted line always passes through $(\bar{x}, \bar{y})$.

Note

"Least squares" minimises **vertical** distances, not perpendicular ones — it treats x as fixed and y as noisy.

Numerical Example — OLS Step by Step

Data ($m = 5$)

x_i	1	2	3	4	5	$\bar{x} = 3$
y_i	2	4	5	4	5	$\bar{y} = 4$
$(x_i - \bar{x})$	-2	-1	0	1	2	
$(y_i - \bar{y})$	-2	0	1	0	1	
$(x_i - \bar{x})(y_i - \bar{y})$	4	0	0	0	2	$S_{xy} = \mathbf{6}$
$(x_i - \bar{x})^2$	4	1	0	1	4	$S_{xx} = \mathbf{10}$

Solve

$$\theta_1 = \frac{6}{10} = \mathbf{0.6}, \qquad \theta_0 = 4 - 0.6(3) = \mathbf{2.2}$$

$$\boxed{\hat{y} = 2.2 + 0.6x}$$

Predictions and residuals

$\hat{y}_i$	2.8	3.4	4.0	4.6	5.2
e_i	-0.8	0.6	1.0	-0.6	-0.2

Check: $\sum e_i = 0$ ✓

Evaluating the Fit — R^2 and Error Metrics

Sums of squares

$$\text{SST} = \sum_i (y_i - \bar{y})^2, \qquad \text{SSE} = \sum_i (y_i - \hat{y}_i)^2, \qquad R^2 = 1 - \frac{\text{SSE}}{\text{SST}}$$

Continuing the example

$$\begin{aligned}
\text{SSE} &= (-0.8)^2 + 0.6^2 + 1.0^2 + (-0.6)^2 + (-0.2)^2 \\
&= 0.64 + 0.36 + 1.00 + 0.36 + 0.04 = \mathbf{2.4} \\
\text{SST} &= 4 + 0 + 1 + 0 + 1 = \mathbf{6.0} \\
R^2 &= 1 - \tfrac{2.4}{6.0} = \mathbf{0.60}
\end{aligned}$$

So 60% of the variance in y is explained by x.

Other metrics

$$\text{MSE} = \frac{\text{SSE}}{m} = \frac{2.4}{5} = \mathbf{0.48}$$

$$\text{RMSE} = \sqrt{0.48} \approx \mathbf{0.69}$$

Caution with R^2

R^2 **never decreases** when features are added. Use **adjusted** R^2

$$R^2_{\text{adj}} = 1 - \frac{(1 - R^2)(m - 1)}{m - n - 1}$$

for model comparison.

Multiple Regression — The Normal Equation

Vector derivation

Minimise $J(\boldsymbol{\theta}) = \frac{1}{2m}\|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|^2$. Differentiate:

$$\nabla_{\boldsymbol{\theta}} J = \frac{1}{m}\mathbf{X}^\top(\mathbf{X}\boldsymbol{\theta} - \mathbf{y}) \stackrel{!}{=} 0 \implies \mathbf{X}^\top\mathbf{X}\boldsymbol{\theta} = \mathbf{X}^\top\mathbf{y} \implies \boxed{\boldsymbol{\theta} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}}$$

Same data, matrix form

$$\mathbf{X}^\top\mathbf{X} = \begin{pmatrix} 5 & 15 \\ 15 & 55 \end{pmatrix}, \quad \mathbf{X}^\top\mathbf{y} = \begin{pmatrix} 20 \\ 66 \end{pmatrix}, \quad \det = 50$$

$$\boldsymbol{\theta} = \frac{1}{50}\begin{pmatrix} 55 & -15 \\ -15 & 5 \end{pmatrix}\begin{pmatrix} 20 \\ 66 \end{pmatrix} = \begin{pmatrix} \mathbf{2.2} \\ \mathbf{0.6} \end{pmatrix}$$

Identical to the OLS result. ✓

When it fails

$(\mathbf{X}^\top\mathbf{X})^{-1}$ may not exist if

- features are **linearly dependent** (redundancy), or
- $m < n$ (more features than examples).

Remedies: delete redundant features, or use **Ridge** — adding λI always makes the matrix invertible.

Gradient Descent — The Update Rule

Algorithm

Repeat until convergence, updating *all* θ_j **simultaneously**:

$$\theta_j := \theta_j - \alpha \frac{\partial J}{\partial \theta_j}, \qquad \alpha > 0 \text{ is the } \textbf{learning rate}.$$

Derivative for linear regression

$$\begin{aligned}\frac{\partial J}{\partial \theta_j} &= \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_i \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)}\right)^2 \\ &= \frac{1}{m} \sum_{i=1}^{m} \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)}\right) x_j^{(i)}\end{aligned}$$

Hence

$$\theta_j := \theta_j - \frac{\alpha}{m} \sum_{i=1}^{m} \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)}\right) x_j^{(i)}$$

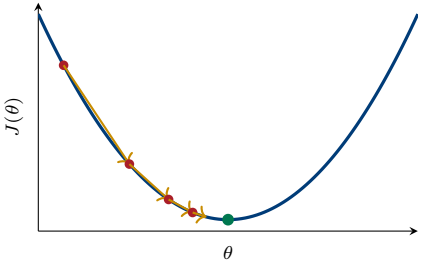


Figure: Each step moves *downhill*; steps shrink automatically as the gradient flattens.

Numerical Example — Gradient Descent Iterations

Same data, $\theta_0 = \theta_1 = 0$, $\alpha = 0.1$, $m = 5$

Initial cost: $J = \dfrac{1}{2(5)}\left(2^2 + 4^2 + 5^2 + 4^2 + 5^2\right) = \dfrac{86}{10} = \mathbf{8.60}$

Iteration 1

Errors $e_i = h(x_i) - y_i = -y_i$:

$$g_0 = \tfrac{1}{m}\sum e_i = \tfrac{-20}{5} = -4.000$$
$$g_1 = \tfrac{1}{m}\sum e_i x_i = \tfrac{-66}{5} = -13.200$$
$$\theta_0 := 0 - 0.1(-4.000) = \mathbf{0.4000}$$
$$\theta_1 := 0 - 0.1(-13.200) = \mathbf{1.3200}$$

New cost $J = \mathbf{0.8232}$.

Progress

Iter	θ_0	θ_1	J
0	0.0000	0.0000	8.6000
1	0.4000	1.3200	0.8232
2	0.3640	1.0680	0.5523
3	0.4072	1.1040	0.5334
⋮	⋮	⋮	⋮
∞	**2.2000**	**0.6000**	0.2400

Gradient descent converges to *exactly* the OLS solution.

The Learning Rate α

Choosing α

- α **too small** $\Rightarrow$ correct but painfully slow.
- α **too large** $\Rightarrow$ overshoots; J may increase or diverge.
- Practical sweep:
  $\ldots, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, \ldots$
  (roughly $\times 3$ each step).

Diagnosis

Plot $J(\boldsymbol{\theta})$ against iteration number.
If J ever **increases** $\Rightarrow$ reduce α.
For sufficiently small α, J *must* decrease every iteration (convex cost).

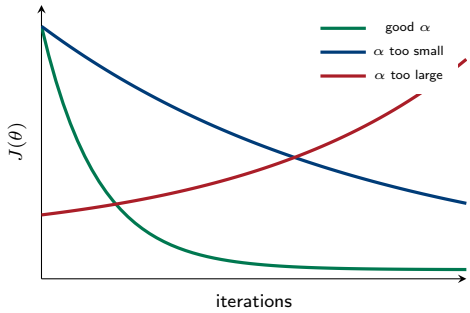


Figure: The learning curve immediately reveals a bad learning rate.

Automatic Convergence Test

The test

Declare convergence when the decrease in J falls below a small threshold ϵ in one iteration:

$$\left|J(\boldsymbol{\theta}^{(t)}) - J(\boldsymbol{\theta}^{(t+1)})\right| < \epsilon, \qquad \epsilon = 10^{-3} \text{ (typical)}$$

Alternatives: stop when $\|\nabla J\| < \epsilon$, or after a maximum iteration count.

Why plotting is still better

Choosing ϵ well is difficult — too large stops early, too small wastes time. Andrew Ng's advice: **look at the curve**. The shape tells you far more than a single threshold.

Applying it to our run

Iter	J	ΔJ
1	0.8232	—
2	0.5523	0.2709
3	0.5334	0.0189
4	0.5300	0.0034
5	0.5288	0.0012
6	0.5281	**0.0007** $< \epsilon$

With $\epsilon = 10^{-3}$ we would stop at iteration **6**.

Feature Scaling — Why It Matters

The problem

If $x_1 \in [0, 2000]$ (sq.ft) and $x_2 \in [1, 5]$ (bedrooms), the cost contours become **highly elongated ellipses**. Gradient descent then **zig-zags** and converges very slowly.

After scaling

Contours become nearly **circular** and gradient descent heads almost straight for the minimum — often an order of magnitude fewer iterations.

Note

Scaling is **not** needed for the normal equation or for decision trees — only for gradient-based and distance-based methods.

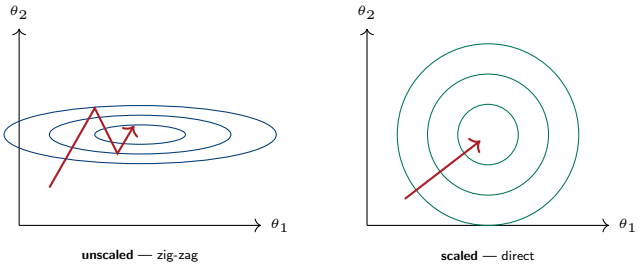


Figure: Feature scaling reshapes the cost contours.

Normalization, Mean Normalization, Standardization

Three formulas

Min–max: $x' = \dfrac{x - \min}{\max - \min} \in [0, 1]$

Mean normalization: $x' = \dfrac{x - \mu}{\max - \min}$

Standardization (z-score): $x' = \dfrac{x - \mu}{\sigma}$

Target

Get every feature into roughly $-1 \le x' \le 1$ (Ng's rule of thumb: anything from about $[-3, 3]$ to $[-\frac{1}{3}, \frac{1}{3}]$ is acceptable).

Worked example

Sizes (sq.ft): $2104,\ 1416,\ 1534,\ 852,\ 1940$.
$\min = 852$, $\max = 2104$, range $= 1252$,
$\mu = 1569.2$, $\sigma = 438.77$.

Scale $x = \mathbf{1416}$:

$$\text{min–max} = \frac{1416 - 852}{1252} = \mathbf{0.451}$$

$$\text{mean norm} = \frac{1416 - 1569.2}{1252} = \mathbf{-0.122}$$

$$\text{z-score} = \frac{1416 - 1569.2}{438.77} = \mathbf{-0.349}$$

Golden rule

Compute $\mu, \sigma, \min, \max$ on the **training set only**, then apply the *same* transform to validation and test data.

Gradient Descent vs. the Normal Equation

Aspect	Gradient descent	Normal equation
Learning rate α	needed (must be tuned)	not needed
Iterations	many	none (one shot)
Feature scaling	**required**	not required
Complexity	$O(k\,n^2 m)$ for k iterations	$O(n^3)$ to invert $\mathbf{X}^\top\mathbf{X}$
Large n	works well ($n > 10^4$)	very slow
Non-invertible $\mathbf{X}^\top\mathbf{X}$	unaffected	fails (needs pseudo-inverse)
Other models	generalises to logistic, NNs	specific to least squares

Practical guidance

Use the **normal equation** when n is small (say $n \lesssim 10^4$) and **gradient descent** otherwise. Variants: **batch** (all m per step), **stochastic** (one example per step), **mini-batch** (typically 32–512) — the last is the standard in practice.

Overfitting in Regression

Polynomial fits

- Degree 1: **underfits** — high bias.
- Degree 2–3: **good fit**.
- Degree 9 (with $m = 10$): passes through *every* point — **overfits**, huge variance.

Symptom

Training error $\to 0$ while test error **explodes**; the fitted coefficients become enormous (e.g. 10^5), oscillating to thread every point.

Cure

Fewer features, more data, or — keeping all features — **regularization**, which penalises large coefficients.

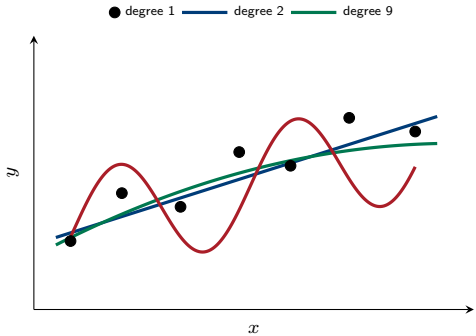


Figure: The degree-9 curve wiggles through every point — it has memorised the noise.

Regularization — The Idea

Penalise large coefficients

Add a penalty term to the cost so that the optimiser must **trade fit against coefficient size**:

$$J(\boldsymbol{\theta}) = \underbrace{\frac{1}{2m}\sum_{i=1}^{m}\left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)}\right)^2}_{\text{fit the data}} + \underbrace{\lambda\,\mathcal{P}(\boldsymbol{\theta})}_{\text{keep } \boldsymbol{\theta} \text{ small}}$$

The two classic penalties

$$\text{Ridge } (L_2): \ \mathcal{P} = \sum_{j=1}^{n}\theta_j^2$$

$$\text{Lasso } (L_1): \ \mathcal{P} = \sum_{j=1}^{n}|\theta_j|$$

Note j starts at **1**: the intercept θ_0 is *not* penalised.

Role of λ

- $\lambda = 0$: ordinary least squares (may overfit).
- λ **small**: mild shrinkage — usually best.
- $\lambda \to \infty$: all $\theta_j \to 0$, model $\to$ the constant $\bar{y}$ (**underfits**).

λ is a **hyperparameter** — chosen by cross-validation.

Ridge Regression (L_2)

Cost and closed-form solution

$$J(\boldsymbol{\theta}) = \frac{1}{2m}\|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|^2 + \frac{\lambda}{2m}\sum_{j\geq 1}\theta_j^2 \quad \Longrightarrow \quad \boxed{\boldsymbol{\theta} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}')^{-1}\mathbf{X}^\top\mathbf{y}}$$

where $\mathbf{I}'$ is the identity with a 0 in the $(0,0)$ entry (intercept excluded).

Two great properties

- Adding $\lambda\mathbf{I}'$ makes the matrix **always invertible** — it solves the multicollinearity problem of the normal equation.
- It **shrinks** coefficients smoothly toward zero, *never exactly* to zero.

Gradient-descent form

$$\theta_j := \theta_j\left(1 - \alpha\tfrac{\lambda}{m}\right) - \frac{\alpha}{m}\sum_i (h - y^{(i)})x_j^{(i)}$$

The factor $(1 - \alpha\lambda/m) < 1$ shrinks θ_j every step — hence "**weight decay**".

Numeric: shrinkage in action

Fit $y = \theta x$ (no intercept) on $x = (1,2,3)$, $y = (2,4,6)$:

$$\theta = \frac{\sum x_i y_i}{\sum x_i^2 + \lambda} = \frac{28}{14 + \lambda}$$

λ	0	1	5	10
θ	**2.000**	1.867	1.474	**1.167**

As λ grows, θ is pulled toward 0 — **bias up, variance down**.

Lasso Regression (L_1) and the Geometry

Lasso cost

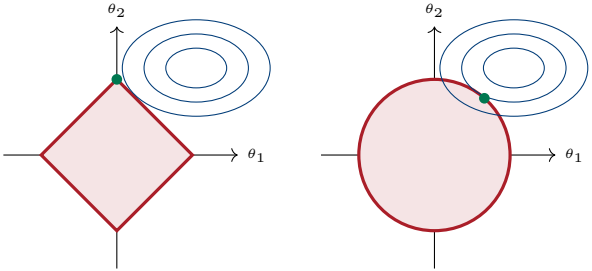
$$J(\boldsymbol{\theta}) = \frac{1}{2m}\|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|^2 + \lambda \sum_{j \geq 1} |\theta_j|$$

- **No closed form** — $|\cdot|$ is not differentiable at 0; solved by coordinate descent or LARS.
- Drives some coefficients **exactly to zero** $\Rightarrow$ automatic **feature selection**.

Elastic Net — the compromise

$$\lambda\Big(\alpha \sum |\theta_j| + \tfrac{1-\alpha}{2} \sum \theta_j^2\Big)$$

Keeps Lasso's sparsity while handling correlated groups like Ridge.



Lasso (L_1): hits a *corner* **Ridge** (L_2): no corner

Figure: The L_1 ball has corners *on the axes*, so the optimum often lands where $\theta_1 = 0$ — this is why Lasso produces sparsity.

Ridge vs. Lasso — Summary

Aspect	Ridge (L_2)	Lasso (L_1)
Penalty	$\sum \theta_j^2$	$\sum \lvert\theta_j\rvert$
Solution	closed form	iterative (no closed form)
Coefficients	shrunk, never exactly 0	some become exactly 0
Feature selection	no	**yes** (automatic)
Correlated features	shares weight among them	tends to pick *one* arbitrarily
Best when	many features, all somewhat useful	few features truly matter (sparse truth)

Choosing λ in practice

Run k-fold cross-validation over a grid such as $\lambda \in \{0.01, 0.1, 1, 10, 100\}$ and pick the value minimising validation error. Always **scale features first** — the penalty is not scale-invariant.

Quick Check — Round 2 (Linear Regression)

[Q1]

For data $x = (1, 2, 3, 4, 5)$, $y = (2, 4, 5, 4, 5)$ we found $S_{xy} = 6$, $S_{xx} = 10$. Compute θ_1 and θ_0.

[Q2]

Write the normal equation and state two situations in which it cannot be used directly.

[Q3]

During gradient descent, J *increases* each iteration. What is wrong, and how do you fix it?

[Q4]

With $\sum x_i y_i = 28$ and $\sum x_i^2 = 14$, compute the ridge estimate $\theta = \frac{\sum x_i y_i}{\sum x_i^2 + \lambda}$ for $\lambda = 0$ and $\lambda = 6$. Which penalty would instead set θ exactly to zero for large enough λ?

Answers — Round 2

[A1]

$\theta_1 = \dfrac{S_{xy}}{S_{xx}} = \dfrac{6}{10} = \mathbf{0.6}$; $\theta_0 = \bar{y} - \theta_1 \bar{x} = 4 - 0.6(3) = \mathbf{2.2}$. Model: $\hat{y} = 2.2 + 0.6x$.

[A2]

$\boldsymbol{\theta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$. It fails when $\mathbf{X}^\top \mathbf{X}$ is **singular**: (i) **redundant/linearly dependent features**, (ii) $m < n$ (more features than examples). Also impractical when n is very large ($O(n^3)$).

[A3]

The **learning rate α is too large** — the steps overshoot the minimum. Fix: reduce α (try $\times \frac{1}{3}$ each time) and re-plot J vs. iterations. Also check that features are scaled.

[A4]

$\lambda = 0:\ \theta = \frac{28}{14} = \mathbf{2.0}$; $\lambda = 6:\ \theta = \frac{28}{20} = \mathbf{1.4}$. Ridge only shrinks; the **Lasso (L_1)** penalty can drive a coefficient **exactly to zero**.

Why Not Linear Regression for Classification?

Three problems

1. Predictions are **unbounded**: $h_{\theta}(\mathbf{x})$ can exceed 1 or fall below 0 — meaningless as a probability.
2. The fit is **sensitive to outliers**: one far-right positive example rotates the line and moves the threshold.
3. The squared-error cost becomes **non-convex** when composed with a thresholding function.

The fix

Pass the linear score through a **squashing function** that maps $\mathbb{R} \to (0, 1)$: the **logistic (sigmoid)** function.

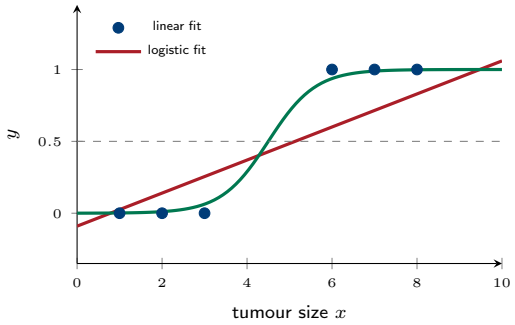


Figure: The linear fit leaves the $[0, 1]$ range; the logistic curve stays inside it.

The Logistic Function — Definition and Properties

Definition

$$\sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^{z}}{1+e^{z}}, \qquad z = \boldsymbol{\theta}^{\top}\mathbf{x}$$

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \sigma(\boldsymbol{\theta}^{\top}\mathbf{x}) = P(y = 1 \mid \mathbf{x}; \boldsymbol{\theta})$$

Properties

- Range $(0, 1)$; $\sigma(0) = 0.5$.
- **Symmetry**: $\sigma(-z) = 1 - \sigma(z)$.
- **Derivative**: $\sigma'(z) = \sigma(z)\big(1 - \sigma(z)\big)$ — remarkably simple, and central to the gradient.
- Monotone increasing, S-shaped, asymptotes at 0 and 1.

Values to memorise

z	-2	-1	0	1	2
$\sigma(z)$	0.119	0.269	0.500	0.731	0.881

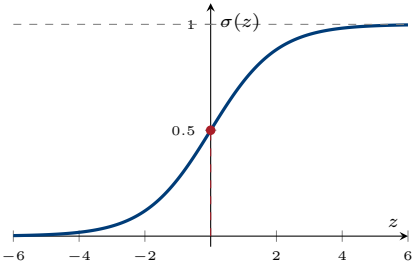


Figure: The sigmoid. Predict $y = 1$ when $\sigma(z) \geq 0.5$, i.e. when $z = \boldsymbol{\theta}^{\top}\mathbf{x} \geq 0$.

Odds, Log-Odds and the Logit

Definitions

$$\text{odds} = \frac{p}{1-p}, \qquad \text{logit}(p) = \ln\Big(\frac{p}{1-p}\Big)$$

The logit is the **inverse** of the logistic function: $\sigma^{-1}(p) = \text{logit}(p)$.

Derivation of the inverse

Start from $p = \dfrac{1}{1+e^{-z}}$:

$$1 + e^{-z} = \tfrac{1}{p} \;\Rightarrow\; e^{-z} = \tfrac{1-p}{p}$$

$$-z = \ln \tfrac{1-p}{p} \;\Rightarrow\; \boxed{z = \ln \tfrac{p}{1-p}}$$

So the model is **linear in the log-odds**:

$$\ln \frac{p}{1-p} = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n$$

Numeric table

p	odds $= \frac{p}{1-p}$	logit $= \ln(\text{odds})$
0.20	0.25	-1.386
0.50	1.00	0.000
0.75	3.00	1.099
0.80	4.00	1.386
0.90	9.00	2.197

Read it as

$p = 0.8$ means odds of **4:1** — "four times as likely to happen as not". Probability is bounded; **log-odds are unbounded**, which is exactly what a linear model needs.

Latent Variable Interpretation

The model behind the model

Suppose there is an unobserved (**latent**) continuous variable

$$y^* = \boldsymbol{\theta}^\top \mathbf{x} + \varepsilon$$

and we only observe the binary outcome

$$y = \begin{cases} 1 & \text{if } y^* > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Why this view is useful

- Gives an **economic/behavioural** reading: y^* is a "propensity" or utility, and the decision flips when it crosses a threshold.
- Explains the **choice of link function** rather than treating the sigmoid as arbitrary.
- Connects logistic regression to **discrete-choice models** in econometrics.

Where the sigmoid comes from

If ε follows the **standard logistic distribution**, then

$$P(y = 1) = P(\varepsilon > -\boldsymbol{\theta}^\top \mathbf{x}) = \sigma(\boldsymbol{\theta}^\top \mathbf{x}).$$

If instead $\varepsilon \sim \mathcal{N}(0, 1)$, the same argument gives the **probit** model.

Example

y^* = a student's (unobserved) "readiness"; we only see *pass/fail*.

The Odds Ratio and Interpreting Coefficients

Key result

Increasing x_j by **one unit** multiplies the odds by e^{θ_j}:

$$\text{OR}_j = \frac{\text{odds}(x_j + 1)}{\text{odds}(x_j)} = \frac{e^{\theta_0 + \theta_j(x_j+1) + \ldots}}{e^{\theta_0 + \theta_j x_j + \ldots}} = \boxed{e^{\theta_j}}$$

Interpretation guide

- $\theta_j > 0 \Rightarrow \text{OR} > 1$: raises the odds.
- $\theta_j = 0 \Rightarrow \text{OR} = 1$: no effect.
- $\theta_j < 0 \Rightarrow \text{OR} < 1$: lowers the odds.
- For a c-unit change, $\text{OR} = e^{c\theta_j}$.

2×2 numeric example

	Success	Failure
Treatment	30	10
Control	20	40

$\text{odds(treat)} = \frac{30}{10} = 3$, $\text{odds(control)} = \frac{20}{40} = 0.5$

$$\text{OR} = \frac{3}{0.5} = \mathbf{6.0} \quad \left(= \frac{ad}{bc} = \frac{30 \times 40}{10 \times 20} \right)$$

Treatment multiplies the odds of success by **6**.

Numerical Example — Making Predictions

Fitted model

A logistic model predicts whether a student passes, from $x =$ marks scored:

$$\ln \frac{p}{1-p} = -4 + 0.05x \quad \Longrightarrow \quad p = \sigma(-4 + 0.05x)$$

Compute for several x

x	$z = -4 + 0.05x$	$p = \sigma(z)$	odds
40	-2.00	0.1192	0.135
60	-1.00	0.2689	0.368
80	**0.00**	**0.5000**	**1.000**
100	$+1.00$	0.7311	2.718

Odds-ratio reading

$$e^{\theta_1} = e^{0.05} = \mathbf{1.051}$$

Each **extra mark** multiplies the odds of passing by 1.051 (a 5.1% increase).

$$e^{10\theta_1} = e^{0.5} = \mathbf{1.649}$$

Ten extra marks raise the odds by about **65%**.

Decision boundary

$p \geq 0.5 \iff z \geq 0 \iff -4 + 0.05x \geq 0 \iff x \geq \mathbf{80}.$

Careful

The effect on the *odds* is constant, but the effect on the *probability* is not — it is largest near $p = 0.5$ and small in the tails.

Multiple Explanatory Variables and the Decision Boundary

General model

$$p(\mathbf{x}) = \sigma\big(\theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n\big)$$

The **decision boundary** is the set where $p = 0.5$, i.e.

$$\boldsymbol{\theta}^\top \mathbf{x} = 0$$

— a **hyperplane**: a line in 2-D, a plane in 3-D.

Non-linear boundaries

Add polynomial features. With $z = \theta_0 + \theta_1 x_1^2 + \theta_2 x_2^2$ and $\boldsymbol{\theta} = (-1, 1, 1)$ the boundary $x_1^2 + x_2^2 = 1$ is a **circle**.

Logistic regression is linear *in the parameters*, not necessarily in the original features.

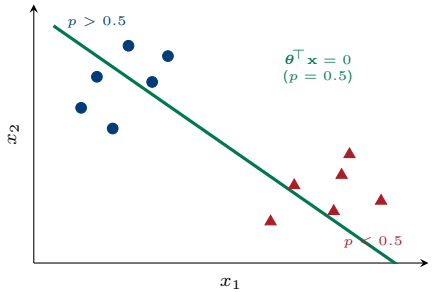


Figure: The boundary is where the linear score is zero.

The Cost Function — Log Loss

Why not squared error?

Substituting the sigmoid into $\frac{1}{2m}\sum(h-y)^2$ gives a **non-convex** function of $\boldsymbol{\theta}$ — gradient descent could stall in a local minimum.

Log loss (cross-entropy)

$$\text{Cost}(h, y) = \begin{cases} -\log(h) & y = 1 \\ -\log(1-h) & y = 0 \end{cases}$$

Combined into one expression:

$$J(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log h^{(i)} + (1-y^{(i)})\log(1-h^{(i)})\right]$$

This J is **convex**, so there is a unique optimum.

The intuition

For $y = 1$: cost $\to 0$ as $h \to 1$, and cost $\to \infty$ as $h \to 0$.
Being confidently wrong is punished without limit.

h (with $y = 1$)	0.5	0.1	0.01
$-\log h$	0.693	2.303	**4.605**

Numeric

$y = (1, 0, 1, 1)$, $h = (0.9, 0.2, 0.7, 0.6)$: losses $0.105,\ 0.223,\ 0.357,\ 0.511$;

$$J = \frac{1.196}{4} = \mathbf{0.299}$$

Model Fitting — Maximum Likelihood and the Gradient

Likelihood

Treating each $y^{(i)} \sim \text{Bernoulli}(h^{(i)})$:

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{m} \left(h^{(i)}\right)^{y^{(i)}} \left(1 - h^{(i)}\right)^{1-y^{(i)}}, \quad \ell(\boldsymbol{\theta}) = \log L(\boldsymbol{\theta}) = -m\, J(\boldsymbol{\theta})$$

Maximising $\ell \equiv$ minimising the log loss.

The gradient — a pleasant surprise

Using $\sigma' = \sigma(1 - \sigma)$, the terms collapse to

$$\frac{\partial J}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)}\right) x_j^{(i)}$$

Identical in form to linear regression — only h differs (sigmoid vs. linear).

No closed form

Setting the gradient to zero gives **transcendental** equations with no algebraic solution. We must iterate:

- Gradient descent / SGD
- **Newton–Raphson** ⇒ **IRLS**
- Quasi-Newton: BFGS, L-BFGS

Note

Add $\frac{\lambda}{2m} \sum_{j \geq 1} \theta_j^2$ to J for regularized logistic regression — essential when features are many or classes are separable.

Iteratively Reweighted Least Squares (IRLS)

Newton–Raphson applied to logistic regression

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \mathbf{H}^{-1}\nabla J, \qquad \mathbf{H} = \mathbf{X}^\top \mathbf{W}\mathbf{X}$$

where $\mathbf{W} = \operatorname{diag}\left(h^{(i)}(1-h^{(i)})\right)$ holds the Bernoulli variances.

The IRLS form

Rearranging gives a **weighted least-squares** problem at each step:

$$\boldsymbol{\theta}^{(t+1)} = (\mathbf{X}^\top \mathbf{W}\mathbf{X})^{-1}\mathbf{X}^\top \mathbf{W}\,\mathbf{z}$$

with the *adjusted response* (working variable)

$$\mathbf{z} = \mathbf{X}\boldsymbol{\theta}^{(t)} + \mathbf{W}^{-1}(\mathbf{y} - \mathbf{h}).$$

Hence the name: we repeatedly solve a re-weighted least-squares fit.

Properties

- **Quadratic convergence** near the optimum — typically 5–10 iterations.
- Uses second-order (curvature) information, so **no learning rate** is needed.
- Cost $O(n^3)$ per iteration from inverting $\mathbf{H}$ $\Rightarrow$ poor for very large n.

Failure mode

With **perfectly separable** data the MLE diverges ($|\boldsymbol{\theta}| \to \infty$) and $\mathbf{W} \to 0$. Fix by adding regularization.

Sample Size — The "Rule of Ten"

The guideline

A logistic regression needs at least **10 events per explanatory variable** (EPV — "events per variable"):

$$m_{\min} \approx \frac{10\,n}{p_{\min}}$$

where $p_{\min}$ is the proportion of the *rarer* outcome and n the number of predictors.

Worked example

$n = 5$ predictors, and only 20% of cases are positive:

$$m_{\min} = \frac{10 \times 5}{0.20} = \textbf{250} \text{ observations}$$

(giving ≈ 50 events for 5 variables).

If violated

- Coefficient estimates become **biased** and unstable.
- Standard errors inflate; confidence intervals are unreliable.
- Risk of **complete separation** rises.

Modern view

The rule of ten is a **rough** guide; later work suggests 5–9 EPV can suffice in some settings, while noisy or highly correlated predictors may need *more*. When data is scarce, prefer **penalised** logistic regression (Ridge/Firth).

Evaluating Goodness of Fit

Likelihood-based measures

- **Deviance**: $D = -2\ell(\boldsymbol{\theta})$; compare nested models via $\Delta D \sim \chi^2$ with Δdf.
- **Likelihood-ratio test**: $\text{LR} = -2(\ell_{\text{null}} - \ell_{\text{model}})$.
- **Pseudo-R^2** (McFadden): $R^2_{\text{McF}} = 1 - \dfrac{\ell_{\text{model}}}{\ell_{\text{null}}}$; values 0.2–0.4 already indicate a good fit.
- **AIC** $= -2\ell + 2k$, **BIC** $= -2\ell + k \ln m$.

Calibration and discrimination

- **Hosmer–Lemeshow test**: group cases into (usually 10) deciles of predicted risk and compare observed vs. expected counts by χ^2. A **large p-value** indicates good fit.
- **ROC / AUC**: ability to *discriminate* classes; 0.5 = chance, > 0.8 = good.
- **Confusion matrix**: accuracy, precision, recall, F1.

Remember

A model can **discriminate** well but be poorly **calibrated**. Check both.

Limitations of Logistic Regression

Limitations

- Assumes **linearity in the log-odds** — curved effects must be added by hand.
- Cannot learn **interactions** automatically.
- Sensitive to **multicollinearity** (unstable coefficients).
- Requires a **reasonably large** sample (rule of ten).
- **Complete separation** $\Rightarrow$ MLE fails to converge.
- Sensitive to **outliers** and to strong **class imbalance**.
- Assumes **independent** observations (not for repeated measures).

Strengths, for balance

- **Highly interpretable** — coefficients are log-odds ratios.
- Outputs **calibrated probabilities**, not just labels.
- Fast to train; a strong **baseline**.
- Extends naturally: multinomial (softmax), ordinal, regularized versions.

When to move on

If diagnostics show strong non-linearity or interactions, try trees, ensembles or kernel methods — or engineer the missing terms explicitly.

Linear Discriminant Analysis — The Idea

Two complementary views

1. **Classification**: model each class density $p(\mathbf{x} \mid y = k)$ as Gaussian, then apply Bayes' rule.
2. **Dimensionality reduction** (Fisher): find the direction $\mathbf{w}$ that *best separates* the classes.

Bayes' rule

$$P(y = k \mid \mathbf{x}) = \frac{\pi_k \, f_k(\mathbf{x})}{\sum_l \pi_l f_l(\mathbf{x})}$$

with prior π_k and class density f_k. Assign $\mathbf{x}$ to the class with the largest posterior.

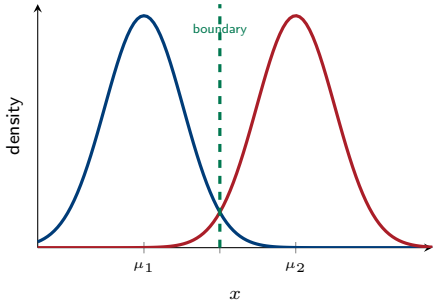


Figure: Two Gaussians with *equal variance* — the optimal boundary is midway between the means.

LDA — Assumptions and Discriminant Function

Assumptions

1. Each class is **multivariate normal**: $\mathbf{x} \mid y = k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$.
2. **Homoscedasticity** — a *common* covariance $\boldsymbol{\Sigma}$ for all classes.
3. Features are **not perfectly collinear**.
4. Observations are independent.

Deriving the discriminant

Taking logs of $\pi_k f_k(\mathbf{x})$ and dropping terms common to all classes, the quadratic term $\mathbf{x}^\top \boldsymbol{\Sigma}^{-1}\mathbf{x}$ **cancels** (shared $\boldsymbol{\Sigma}$), leaving the **linear** discriminant

$$\boxed{\delta_k(\mathbf{x}) = \mathbf{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k - \tfrac{1}{2}\boldsymbol{\mu}_k^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \ln \pi_k}$$

Rule: assign $\mathbf{x}$ to $\arg\max_k \delta_k(\mathbf{x})$.

If assumption 2 fails

Allowing $\boldsymbol{\Sigma}_k$ to differ per class gives **QDA**, whose boundary is **quadratic** rather than linear.

Two-class boundary

Setting $\delta_1 = \delta_2$ gives a hyperplane with normal

$$\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1).$$

LDA for Two Classes — Decision Rule

The rule

Assign $\mathbf{x}$ to class 2 iff

$$\mathbf{x}^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) > \tfrac{1}{2}(\boldsymbol{\mu}_2 + \boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) + \ln \frac{\pi_1}{\pi_2}$$

With **equal priors** ($\pi_1 = \pi_2$) the threshold is simply the **midpoint** of the projected means.

Estimating the parameters

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{m_k} \sum_{i:y_i=k} \mathbf{x}_i, \qquad \hat{\pi}_k = \frac{m_k}{m}$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{m-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^\top$$

— the **pooled** within-class covariance.

Fisher's criterion (the other view)

Choose the projection direction $\mathbf{w}$ maximising

$$J(\mathbf{w}) = \frac{\mathbf{w}^\top \mathbf{S}_B \mathbf{w}}{\mathbf{w}^\top \mathbf{S}_W \mathbf{w}} \quad \Rightarrow \quad \mathbf{w} \propto \mathbf{S}_W^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)$$

with $\mathbf{S}_B$ = between-class and $\mathbf{S}_W$ = within-class scatter.

Same answer as the Bayes derivation — maximise between-class separation relative to within-class spread.

Numerical Example — LDA in One Dimension

Data

Class 1: $\{2, 3, 4, 5\}$ Class 2: $\{7, 8, 9, 10\}$ (equal priors)

Step 1 — means and pooled variance

$$\hat{\mu}_1 = \frac{2+3+4+5}{4} = 3.5, \qquad \hat{\mu}_2 = \frac{7+8+9+10}{4} = 8.5$$

Each class has sample variance $s^2 = \frac{(1.5)^2+(0.5)^2+(0.5)^2+(1.5)^2}{3} = \frac{5}{3}$, so

$$s_p^2 = \frac{3(\frac{5}{3}) + 3(\frac{5}{3})}{4+4-2} = 1.6667$$

Step 2 — the direction

$$w = \frac{\hat{\mu}_2 - \hat{\mu}_1}{s_p^2} = \frac{5}{1.6667} = 3.0$$

Step 3 — threshold and classification

Equal priors $\Rightarrow$ threshold at the midpoint

$$x^* = \frac{\hat{\mu}_1 + \hat{\mu}_2}{2} = \frac{3.5 + 8.5}{2} = 6.0$$

x	δ_1	δ_2	Class
5.5	-1.200	-2.700	1
6.0	-1.875	-1.875	tie
6.5	-2.700	-1.200	2

using $\delta_k = -\frac{(x-\hat{\mu}_k)^2}{2s_p^2}$ (equal priors).

At $x = 6$ the two discriminants tie exactly — the boundary.

Eigenvalues, Discriminant Dimensions and Effect Size

The eigen-problem

Fisher's criterion is maximised by solving

$$\mathbf{S}_W^{-1}\mathbf{S}_B\,\mathbf{w} = \lambda\,\mathbf{w}$$

- The eigenvector with the largest **eigenvalue** λ is the best discriminant direction.
- With K classes there are at most $K-1$ useful discriminants (rank of $\mathbf{S}_B$).
- So two classes give exactly **one** direction.

Interpreting the eigenvalues

$$\text{proportion of separation}_i = \frac{\lambda_i}{\sum_j \lambda_j}$$

$$\text{canonical correlation} = \sqrt{\frac{\lambda}{1+\lambda}}$$

Effect size — Mahalanobis distance

$$D^2 = (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)$$

For our 1-D example: $D^2 = \frac{(8.5-3.5)^2}{1.6667} = \mathbf{15.0}$, so $D = 3.87$ — the classes are **very well separated** (roughly 3.9 pooled SDs apart).

LDA vs. Logistic Regression

Aspect	LDA	Logistic regression
Approach	**Generative** — models $p(\mathbf{x} \mid y)$	**Discriminative** — models $p(y \mid \mathbf{x})$ directly
Assumptions	Gaussian classes, common $\boldsymbol{\Sigma}$	none on $p(\mathbf{x})$; only linear log-odds
Estimation	closed form (means, pooled cov.)	iterative MLE / IRLS
Small samples	more **stable**	can be unstable
Separable data	works fine	MLE **diverges**
Non-Gaussian / outliers	degrades	more **robust**
Multi-class	natural	needs softmax extension
Extra use	**dimensionality reduction**	none

Practical verdict

Both give **linear boundaries** and often perform similarly. Prefer **LDA** when its Gaussian assumptions roughly hold, classes are well separated, or m is small; prefer **logistic regression** when they clearly do not, or when you need probabilities with fewer assumptions.

Practical Use and Applications

Bankruptcy prediction

Altman's Z-score (1968) is a discriminant function of financial ratios:

$$Z = 1.2X_1 + 1.4X_2 + 3.3X_3 + 0.6X_4 + 1.0X_5$$

(X_1 working capital/assets, X_2 retained earnings/assets, X_3 EBIT/assets, X_4 equity/liabilities, X_5 sales/assets).

$Z > 2.99$ safe, $1.81 - 2.99$ grey, $Z < 1.81$ distress.

Face recognition

Fisherfaces = PCA (to reduce dimension) followed by LDA (to maximise class separation). Outperforms plain *Eigenfaces* under varying lighting and expression.

Marketing

Classifying customers as likely buyers vs. non-buyers from demographic and behavioural variables; churn prediction; segment discrimination and profiling.

Biomedical studies

Disease vs. healthy classification from clinical measurements; severity staging; gene-expression classification (with regularized/sparse LDA when $n \gg m$).

Common thread

All are problems where classes are reasonably **Gaussian-like**, the sample is modest, and an **interpretable linear score** is valued.

Quick Check — Round 3 (Logistic Regression & LDA)

[Q1]

If $p = 0.75$, compute the odds and the logit.

[Q2]

A logistic model has $\theta_j = 0.7$ for a binary feature. Interpret $e^{0.7}$. ($e^{0.7} \approx 2.01$.)

[Q3]

Why is squared error not used as the cost for logistic regression?

[Q4]

For LDA with $\hat{\mu}_1 = 4$, $\hat{\mu}_2 = 10$, $s_p^2 = 2$ and equal priors: give the decision threshold and classify $x = 8$.

Answers — Round 3

[A1]

odds $= \dfrac{0.75}{0.25} = \mathbf{3.0}$ (i.e. $3{:}1$); logit $= \ln 3 = \mathbf{1.099}$.

[A2]

Having the feature (vs. not) **doubles the odds** of the positive outcome (OR ≈ 2.01), **holding all other variables fixed**. Note it doubles the *odds*, not the probability.

[A3]

Because composing the sigmoid with squared error yields a **non-convex** cost with local minima. **Log loss (cross-entropy)** is convex and is also the negative log-likelihood of the Bernoulli model.

[A4]

Threshold $= \dfrac{4+10}{2} = \mathbf{7}$. Since $x = 8 > 7$, assign to **class 2**. (Check: $\delta_1 = -\frac{16}{4} = -4$, $\delta_2 = -\frac{4}{4} = -1$, and $-1 > -4$.)

Summary of Unit 2

What we covered

1. **Supervised learning** — ERM, the six steps, and the five factors governing success.
2. **Bias–variance** — full decomposition; complexity, data volume, dimensionality, noise.
3. **Linear regression** — representation, squared-error cost, OLS derivation and the normal equation.
4. **Gradient descent** — update rule, learning rate, convergence test, feature scaling; GD vs. normal equation.
5. **Regularization** — Ridge (L_2, closed form, weight decay) and Lasso (L_1, sparsity); Elastic Net.
6. **Logistic regression** — sigmoid, logit, odds, odds ratio, latent-variable view, log loss, MLE and IRLS, rule of ten, goodness of fit.
7. **LDA** — Gaussian assumptions, discriminant functions, Fisher's criterion, eigenvalues, effect size, and applications.

One dataset, three routes — OLS, the normal equation and gradient descent — all gave $\hat{y} = 2.2 + 0.6x$.

Practice Questions (Exam Oriented)

Short answer (2–5 marks)

1. State and explain the bias–variance decomposition.
2. Derive $\theta_1 = S_{xy}/S_{xx}$ for simple linear regression.
3. Why must features be scaled for gradient descent but not for the normal equation?
4. Define odds, odds ratio and logit; state the relation between them.
5. Why is the sigmoid derivative $\sigma' = \sigma(1-\sigma)$ convenient?
6. List the assumptions of LDA.

Long answer (8–10 marks) — include numerics

7. For given (x_i, y_i), compute θ_0, θ_1, the residuals, SSE, SST and R^2.
8. Derive the normal equation from $J(\boldsymbol{\theta}) = \frac{1}{2m}\|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|^2$ and solve a 2×2 case.
9. Perform two iterations of gradient descent from $\boldsymbol{\theta} = 0$ with a stated α.
10. Compare Ridge and Lasso: cost functions, solutions, geometry and effect on coefficients.
11. Derive the logit as the inverse of the sigmoid; interpret e^{θ_j} with a numeric example.
12. Derive the LDA discriminant function and classify a given point numerically.

Think & Explore (Take-Home)

Numeric drill

Using $x = (1, 2, 3, 4, 5)$, $y = (3, 4, 6, 7, 9)$:

- Compute θ_0, θ_1 by OLS.
- Verify with the normal equation.
- Run two gradient-descent steps with $\alpha = 0.05$ from $\boldsymbol{\theta} = 0$.
- Compute R^2 and comment on the fit.

Coding (self-learning)

With `scikit-learn`: fit `LinearRegression`, `Ridge`, `Lasso` on a dataset; plot coefficient paths against λ. Then fit `LogisticRegression` and `LinearDiscriminantAnalysis` and compare their decision boundaries.

Reading

- Hastie, Tibshirani & Friedman, *ESL* — Ch. 3 (linear regression), Ch. 4 (LDA, logistic).
- James et al., *ISLR* — Ch. 3–4 (very accessible).
- Bishop, *PRML* — Ch. 3–4.
- Ng, *CS229* notes — linear & logistic regression.

Reminder

Unit 3 continues supervised learning (trees, SVM, naive Bayes, k-NN, ensembles). Post doubts on the course page.

References

Textbooks

1. T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning*, 2nd ed., Springer. **[Main reference]**
2. G. James, D. Witten, T. Hastie, R. Tibshirani, *An Introduction to Statistical Learning*, Springer.
3. C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer.
4. T. M. Mitchell, *Machine Learning*, McGraw Hill.
5. M. Mohri, A. Rostamizadeh, A. Talwalkar, *Foundations of Machine Learning*, 2nd ed., MIT Press.
6. D. W. Hosmer, S. Lemeshow, *Applied Logistic Regression*, Wiley.

Classic papers

7. R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, 1936.
8. E. I. Altman, "Financial ratios, discriminant analysis and the prediction of corporate bankruptcy," *J. Finance*, 1968.
9. R. Tibshirani, "Regression shrinkage and selection via the Lasso," *JRSS-B*, 1996.

Thank You

Any Questions?

"All models are wrong, but some are useful."
— George E. P. Box

Next: **Unit 3** — Supervised Learning Algorithms, Part Two.

Post your doubts on the course page →
https://www.gcjana.in/courses/shardauniversity/2601/machine-learning/#Post-doubts