Week 3 — Practical Session Solution Guide

ROC Curves, Normalization & PCA

Teaching Assistant Reference for Students

Contents

Part 1 — Exam-Style Questions

These multiple-choice questions test understanding of core concepts covered so far.

Question 1. *What is the advantage of gradient descent over random search?*

Solution

Answer: C — Gradient descent computes the *direction of steepest descent*; random search only approximates it.

Why the other options are wrong:

- **A** — Parallel communication is not a property of standard gradient descent.
- **B** — Gradient descent can *also* get stuck in local minima; it does not escape them more easily than random search.
- **D** — Gradient descent is actually *harder* to parallelise than random search because each update depends on the previous one.

Question 2. *Why is accuracy a bad loss function to use in gradient descent?*

Solution

Answer: B — Accuracy produces a gradient that is **zero almost everywhere**.

Explanation: Accuracy counts the fraction of correct predictions. It only changes when the decision boundary crosses a data point — a discrete event. Between any two such events the value is flat, giving a zero gradient. Gradient descent cannot move in any direction because it has no slope information to follow.

Note: option C (class imbalance) is a real problem with accuracy as an *evaluation metric*, but it is not the reason it fails as a *loss function* for gradient descent.

Question 3. *Spam classification with word-count attributes — what are they called?*

Solution

Answer: C — They are called **features** (also known as attributes or input variables). Each email is one *instance*; the three word-counts are its features; the spam/not-spam label is the *class*.

Gradient derivation questions (4–6)

The following derivation is given:

$$\frac{\partial \frac{1}{2}\sum_i (f(x_i)-y_i)^2}{\partial b} \xrightarrow{(1)\to(2)} \frac{1}{2}\sum_i \frac{\partial (x_i w + b - y_i)^2}{\partial b} \xrightarrow{(2)\to(3)} \frac{1}{2}\sum_i \frac{\partial (x_i w + b - y_i)^2}{\partial (x_i w + b - y_i)} \cdot \frac{\partial (x_i w + b - y_i)}{\partial b} \cdots$$

Question 4. *Step (1) → (2): which rule?*

Solution

Answer: D — Sum rule.

We move the derivative *inside* the sum: $\frac{\partial}{\partial b}\sum_i(\cdots) = \sum_i \frac{\partial(\cdots)}{\partial b}$. This is valid because differentiation is a linear operator.

Question 5. *Step (2) → (3): which rule?*

Solution

Answer: A — Chain rule.
We have a composite function: the outer function is u^2 and the inner function is $u = x_i w + b - y_i$. The chain rule says:

$$\frac{\partial u^2}{\partial b} = \frac{\partial u^2}{\partial u} \cdot \frac{\partial u}{\partial b}$$

which is exactly the split shown in step (3).

Question 6. *What is the correct result in line (5)?*

Solution

Answer: C — $\sum_i (x_i w + b - y_i)$.

Step-by-step derivation from line (4):

$$\sum_i (x_i w + b - y_i) \cdot \underbrace{\frac{\partial (x_i w + b - y_i)}{\partial b}}_{=1} = \sum_i (x_i w + b - y_i)$$

The inner partial derivative equals 1 because b appears with coefficient 1 and $x_i w, y_i$ are treated as constants.

Part 2 — ROC Curves and Confusion Matrices

Dataset reference

Point	x_1	x_2	Label
a	1	0	Neg
b	2	0	Pos
c	3	0	Pos
d	1/2	1	Pos
e	2	2	Pos
f	0	3	Neg
g	1	3	Neg
h	2	3	Neg

There are **4 positive** instances (b, c, d, e) and **4 negative** instances (a, f, g, h).

Question 1. *Describe c_{lin} mathematically.*

Key Idea / Hint

The decision boundary is the line $x_1 = x_2$. Ask: for a point *above* this line, is $x_1 > x_2$ or $x_1 < x_2$? Remember the axes: x_1 is horizontal, x_2 is vertical.

Solution

$$c_{\text{lin}}(x_1, x_2) = \begin{cases} \text{Pos} & \text{if } x_1 \geq x_2 \\ \text{Neg} & \text{otherwise} \end{cases}$$

Reasoning: The boundary is all points where $x_1 = x_2$ (a diagonal). Points *below* the diagonal satisfy $x_1 > x_2$ (they are to the right). The problem states points below or on the line are Pos, confirming the rule above.

Quick check with data points:

- $b = (2, 0)$: $x_1 = 2 \geq x_2 = 0 \Rightarrow$ Pos ✓
- $f = (0, 3)$: $x_1 = 0 < x_2 = 3 \Rightarrow$ Neg ✓

Question 2. *Define c_2, whose boundary passes through $(0, 1)$ and $(1, 1.5)$, with Neg above.*

Key Idea / Hint

Fix $b = 1$ in $ax_1 + bx_2 + c = 0$, so the line becomes $x_2 = -ax_1 - c$. Find the slope from the two given points and read off the intercept.

Solution

Step 1 — Find the slope.

$$\text{slope} = \frac{1.5 - 1}{1 - 0} = 0.5$$

So $x_2 = 0.5\,x_1 + 1$, meaning $-a = 0.5$ and $-c = 1$.

Step 2 — Write the classifier. Using $-0.5x_1 + x_2 - 1$:

$$c_2(x_1, x_2) = \begin{cases} \text{Pos} & \text{if } -0.5x_1 + x_2 - 1 \leq 0 \\ \text{Neg} & \text{otherwise} \end{cases}$$

Step 3 — Verify sign convention. Point $(0, 0)$ is clearly below the line (Pos region): $-0.5(0) + 0 - 1 = -1 < 0 \Rightarrow$ Pos. ✓

Question 3. *Label the four leaves A, B, C, D of c_{tree}.*

Key Idea / Hint

Trace each data point through the tree. The root splits on $x_2 > 1.5$: go right (True) or left (False). Then apply the second split. Assign the majority class in each leaf region.

Solution

Tracing the tree:

Path	Region	Points	Majority
$x_2 \leq 1.5$ $x_1 \leq 0.5$	A	d (Pos)	**Pos**
$x_2 \leq 1.5$ $x_1 > 0.5$	B	a (Neg), b (Pos), c (Pos), e (Pos)	**Pos**
$x_2 > 1.5$ $x_1 \leq 1.5$	C	f (Neg), g (Neg)	**Neg**
$x_2 > 1.5$ $x_1 > 1.5$	D	h (Neg), e already counted	**Pos**

Leaf labels: A = **Pos**, B = **Pos**, C = **Neg**, D = **Pos**.

Question 4. *When would you still compute training loss?*

Solution

You compute training loss to **diagnose overfitting**. If training loss is very low but validation loss is high, the model has memorised the training data and does not generalise. This is especially relevant for high-capacity models like decision trees and neural networks.

Question 5. *Give the confusion matrices for c_{lin} and c_{tree}.*

Key Idea / Hint

For each classifier, go through every data point, apply the rule, and tally TP / FP / FN / TN. Remember: **Pos** is the positive class.

Solution

c_{lin} classifications:

- a (Neg): $1 < 0$? No $\Rightarrow$ Neg. **TN**
- b (Pos): $2 > 0 \Rightarrow$ Pos. **TP**
- c (Pos): $3 > 0 \Rightarrow$ Pos. **TP**
- d (Pos): $0.5 < 1 \Rightarrow$ Neg. **FN**
- e (Pos): $2 = 2 \Rightarrow$ Pos (on boundary). **TP**
- f (Neg): $0 < 3 \Rightarrow$ Neg. **TN**
- g (Neg): $1 < 3 \Rightarrow$ Neg. **TN**
- h (Neg): $2 < 3 \Rightarrow$ Neg. **TN** *wait — h is Neg, predicted Neg, so TN*

*Re-check: a is predicted Neg (true Neg = TN), but also h is predicted Neg. The single FP must come from one negative being predicted Pos. Let's re-examine: c_{lin} predicts Pos iff $x_1 \geq x_2$. Point a = (1, 0): $1 \geq 0 \Rightarrow$ **Pos predicted** for a true Neg $\Rightarrow$ FP.*

c_{lin}		Predicted Pos	Predicted Neg
True	Pos	3 (b,c,e)	1 (d)
	Neg	1 (a)	3 (f,g,h)

c_{tree}		Predicted Pos	Predicted Neg
True	Pos	4 (b,c,d,e)	0
	Neg	2 (a,h)	2 (f,g)

Question 6. *Compute accuracy, precision, recall, TPR, and FPR.*

Key Idea / Hint

Use the definitions below, where the confusion matrix entries are TP, FP, FN, TN and

total $= \text{TP} + \text{FP} + \text{FN} + \text{TN}$:

$$\text{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{total}}, \quad \text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}, \quad \text{Recall} = \text{TPR} = \frac{\text{TP}}{\text{TP}+\text{FN}}, \quad \text{FPR} = \frac{\text{FP}}{\text{FP}+\text{TN}}$$

Solution

From the confusion matrices: for c_{lin}: TP=3, FP=1, FN=1, TN=3. For c_{tree}: TP=4, FP=2, FN=0, TN=2.

Metric	c_{lin}	c_{tree}
Accuracy	$\frac{3+3}{8} = \frac{3}{4}$	$\frac{4+2}{8} = \frac{3}{4}$
Precision	$\frac{3}{3+1} = \frac{3}{4}$	$\frac{4}{4+2} = \frac{2}{3}$
Recall (= TPR)	$\frac{3}{3+1} = \frac{3}{4}$	$\frac{4}{4+0} = 1$
FPR	$\frac{1}{1+3} = \frac{1}{4}$	$\frac{2}{2+2} = \frac{1}{2}$

Interpretation: Both classifiers have the same accuracy (6/8), but they trade off differently. c_{tree} has perfect recall (finds every positive) but a higher false positive rate — it is more aggressive. c_{lin} is more conservative and balanced.

Question 7. *How do we turn c_{lin} into a ranking classifier?*

Key Idea / Hint

The score of a point is its *signed distance* to the decision boundary. Points further into the Pos region get higher scores; points further into the Neg region get lower scores. For a visual approach, draw the boundary and read off relative distances.

Solution

Method: Compute the signed perpendicular distance from each point to the boundary $x_1 - x_2 = 0$. The boundary direction runs diagonally; points above it (higher x_2) are more negative, points below it (higher x_1) are more positive.

Reading off distances from the diagram (most negative to most positive):

$$\underbrace{f}_{\text{most Neg}} \quad g \;\; h \;\; d \;\; e \;\; a \;\; b \quad \underbrace{c}_{\text{most Pos}}$$

Intuition: $f = (0, 3)$ is far above the diagonal $\Rightarrow$ most negative. $c = (3, 0)$ is far below $\Rightarrow$ most positive.

Question 8. *How do we turn c_{tree} into a ranking classifier?*

Key Idea / Hint

Group points by which leaf they fall into. Compute the fraction of positives in each leaf. Sort leaves by that fraction (ascending = most negative first). Points within the same leaf have *equal rank*.

Solution

Leaf class probabilities:

Leaf	Points	P(Pos)	Rank
C ($x_2 > 1.5$, $x_1 \leq 1.5$)	f (Neg), g (Neg)	$0/2 = 0$	most Neg
D ($x_2 > 1.5$, $x_1 > 1.5$)	h (Neg), e (Pos)	$1/2 = 0.5$	middle
B ($x_2 \leq 1.5$, $x_1 > 0.5$)	a (Neg), b,c,e (Pos)	$3/4$	high Pos
A ($x_2 \leq 1.5$, $x_1 \leq 0.5$)	d (Pos)	$1/1 = 1$	most Pos

Ranking (most negative to most positive):

$$(\mathbf{f\ g})\ \ (\mathbf{h\ e})\ \ (\mathbf{a\ b\ c})\ \ (\mathbf{d})$$

Brackets denote equal rank (ties within a leaf).

Question 9. *Coverage matrices, ranking errors, and class imbalance.*

Key Idea / Hint

In a coverage matrix, place **negative** instances on the horizontal axis and **positive** instances on the vertical axis, both ordered from most positive (bottom-left) to most negative (top-right) according to the ranking. A cell is **green** if the positive instance is ranked above the negative one (correct), **red** if ranked below (error), and **yellow** if tied (counts as 0.5 error).

Solution

Part 1 — Coverage matrices.

For c_{lin}, ranking (neg → pos): f g h — d e b c (negatives: f,g,h,a; positives: b,c,d,e):

	a	h	g	f
d	R	G	G	G
e	R	G	G	G
b	G	G	G	G
c	G	G	G	G

c_{lin}: 2 red cells = **2 ranking errors**

For c_{tree}, ranking: (f,g), (h,e), (a,b,c), (d):

	a	h	f	g
e	R	Y	G	G
c	Y	G	G	G
b	Y	G	G	G
d	G	G	G	G

c_{tree}: 5 ties $\times$ 0.5 = **2.5 ranking errors**

Part 2 — Counting errors:

- c_{lin}: 2 red cells $\Rightarrow$ **2 errors**.
- c_{tree}: 1 red cells, 3 yellow cells $\Rightarrow 3 \times 0.5 + 1 \times 1 =$ **2.5 errors**.

Part 3 — Class imbalance: The coverage matrix will be *non-square*. More negative examples than positive $\Rightarrow$ wider than tall. The more stretched the matrix, the greater the imbalance.

Question 10. *Converting a coverage matrix to an ROC plot.*

Solution

Conversion: Normalise both axes. If the coverage matrix is $P \times N$ (P positives, N negatives) and a classifier achieves a green corner at cell (i, j) from the bottom-left, the corresponding ROC point is:

$$\text{FPR} = \frac{i}{N}, \qquad \text{TPR} = \frac{j}{P}$$

Relationship between green area and AUC:

- The fraction of *green cells* in the coverage matrix estimates the probability that a randomly chosen positive is ranked above a randomly chosen negative — this is the AUC.
- The ROC AUC is computed from the convex hull of achievable classifiers in ROC space.

Key difference: The green region in a coverage matrix can be *concave* (some bad classifiers rank things backwards), while the ROC curve always takes the *convex hull*. For reasonable classifiers (AUC > 0.5) on large datasets, the difference is negligible.

Part 3 — Probability

Background: Changing Basis

If a point has coordinates $\begin{pmatrix} x \\ y \end{pmatrix}$ in a non-standard basis $\{\mathbf{b}_1, \mathbf{b}_2\}$, then its coordinates in the *standard* basis are:

$$\mathbf{p}_{\text{std}} = x\,\mathbf{b}_1 + y\,\mathbf{b}_2$$

Question 12. *Which of A, B, C have orthogonal columns? Which is orthonormal?*

Key Idea / Hint

Two vectors are **orthogonal** iff their dot product is 0. A vector is **normal** (unit length) iff $\|\mathbf{v}\| = \sqrt{v_1^2 + v_2^2} = 1$.

Solution

Given:

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}, \quad C = \sqrt{\tfrac{1}{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$$

Matrix A: Columns $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ and $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$. Dot product: $0 \cdot 1 + 1 \cdot 1 = 1 \neq 0$. *Not orthogonal.*

Matrix B: Columns $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ and $\begin{pmatrix} -1 \\ 1 \end{pmatrix}$. Dot product: $1 \cdot (-1) + 1 \cdot 1 = 0$. *Orthogonal!* But $\| \begin{pmatrix} 1 \\ 1 \end{pmatrix} \| = \sqrt{2} \neq 1$. *Not normalised.*

Matrix C: Same directions as B, scaled by $\sqrt{1/2}$. Dot product: still 0. *Orthogonal.* Length: $\sqrt{1/2} \cdot \sqrt{1^2 + 1^2} = \sqrt{1/2} \cdot \sqrt{2} = 1$. *Normalised!*

$\Rightarrow$ **C is orthonormal.**

Question 13. *Why is an orthonormal basis useful?*

Solution

For a matrix C representing an orthonormal basis, C is an *orthogonal matrix*, meaning:

$$C^{-1} = C^\top$$

Why this matters: Computing a matrix inverse is computationally expensive ($O(n^3)$). A transpose is free. This means:

- To transform *into* the basis: multiply by $C^\top$.
- To transform *back* to standard: multiply by C.

This is heavily exploited in PCA, where we need to project data into a new basis and reconstruct it efficiently.

Question 14. *Compute the sample covariance of the given data.*

Key Idea / Hint

The sample covariance matrix is $S = \frac{1}{n-1} XX^\top$, where X is the data matrix with the *mean-subtracted* instances as columns.

Solution

Step 1 — Compute the mean.

$$\bar{x}_1 = \frac{0+2+0+2}{4} = 1, \qquad \bar{x}_2 = \frac{2+0+2+0}{4} = 1 \quad \Rightarrow \quad \mathbf{m} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Step 2 — Subtract the mean. Arranging mean-centred data as columns:

$$X = \begin{pmatrix} -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix}$$

Step 3 — Compute $XX^\top$.

$$XX^\top = \begin{pmatrix} -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix} \begin{pmatrix} -1 & 1 \\ 1 & -1 \\ -1 & 1 \\ 1 & -1 \end{pmatrix} = \begin{pmatrix} 4 & -4 \\ -4 & 4 \end{pmatrix}$$

Step 4 — Scale.

$$S = \frac{1}{n-1} XX^\top = \frac{1}{3} \begin{pmatrix} 4 & -4 \\ -4 & 4 \end{pmatrix} = \begin{pmatrix} 4/3 & -4/3 \\ -4/3 & 4/3 \end{pmatrix}$$

The strong negative off-diagonal tells us x_1 and x_2 are **negatively correlated** — when one is high, the other tends to be low (as visible in the data: entries are always $(0, 2)$ or $(2, 0)$).

Question 15. *Sample from $\mathcal{N}(\mu, \Sigma)$ using the affine transformation.*

Key Idea / Hint

Given $y = Ax + t$ with $x \sim \mathcal{N}(0, I)$, the resulting $y \sim \mathcal{N}(\mu, \Sigma)$ where $\mu = t$ and $\Sigma = AA^\top$. Pair consecutive numbers as 2D vectors for x, then apply the transformation.

Solution

Given: $A = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$, $t = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. Available numbers: 0.5, 1.1, 0.1, 0.9, -0.2, 1.3.

Step 1 — Form three standard normal samples.

$$\mathbf{x}^{(1)} = \begin{pmatrix} 0.5 \\ 1.1 \end{pmatrix}, \quad \mathbf{x}^{(2)} = \begin{pmatrix} 0.1 \\ 0.9 \end{pmatrix}, \quad \mathbf{x}^{(3)} = \begin{pmatrix} -0.2 \\ 1.3 \end{pmatrix}$$

Step 2 — Apply $y = Ax + t$.

$$\mathbf{y}^{(1)} = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} 0.5 \\ 1.1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0.5 \\ 2.2 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1.5 \\ 3.2 \end{pmatrix}$$

$$\mathbf{y}^{(2)} = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} 0.1 \\ 0.9 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0.1 \\ 1.8 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1.1 \\ 2.8 \end{pmatrix}$$

$$\mathbf{y}^{(3)} = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} -0.2 \\ 1.3 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} -0.2 \\ 2.6 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0.8 \\ 3.6 \end{pmatrix}$$

Step 3 — Identify the distribution.

$$\mu = t = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad \Sigma = AA^\top = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix}$$

So $y \sim \mathcal{N}\left(\begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix} \right)$: the y_1 component has variance 1 (unchanged) and y_2 has variance 4 (standard deviation 2), which is reflected in the samples above.

Bonus: Sample covariance of the transformed data

Solution

The three y samples are $\{(1.5, 3.2), (1.1, 2.8), (0.8, 3.6)\}$.
Sample mean:

$$\bar{y}_1 = \frac{1.5 + 1.1 + 0.8}{3} \approx 1.133, \qquad \bar{y}_2 = \frac{3.2 + 2.8 + 3.6}{3} \approx 3.2$$

Mean-centred matrix (columns = instances):

$$Y \approx \begin{pmatrix} 0.367 & -0.033 & -0.333 \\ 0 & -0.4 & 0.4 \end{pmatrix}$$

Sample covariance:

$$S_y = \frac{1}{2} Y Y^\top \approx \begin{pmatrix} 0.123 & 0.06 \\ 0.06 & 0.16 \end{pmatrix}$$

True covariance: $\Sigma = AA^\top = \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix}$.
The sample estimate is far from the true value because $n = 3$ is tiny. As $n \to \infty$, $S_y \to \Sigma$ by the Law of Large Numbers.