كلية العلوم

قسم الانظمة الطبية الذكية

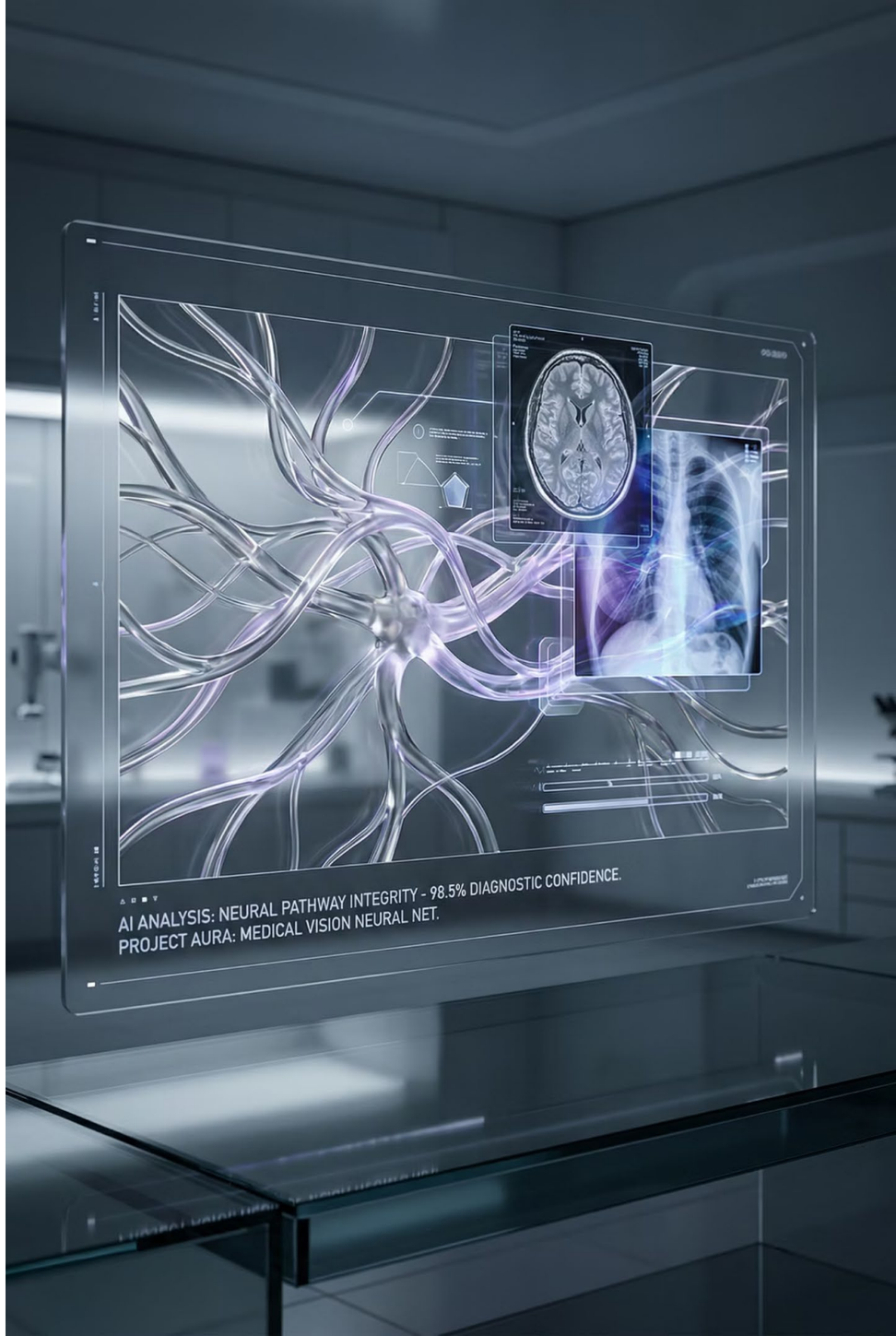
Lec 7 & 8

Feature Extraction & Classification Methods

Subject: Computer Vision

Level: third

Lecturer: Asst. Lecturer Qusai AL-Durrah



From Images to Decisions: The Core Pipeline

In intelligent medical systems, raw pixel data alone cannot drive clinical decisions. We must convert visual information into structured numerical representations that algorithms can reason about.

1

Raw Image

Pixel array — thousands of values, no semantic meaning for a classifier

2

Feature Extraction

Computes compact descriptor vector **x** — intensity, edges, texture

3

Classification

A rule maps **x** → predicted class label $\hat{y}$ (e.g., Normal vs. Abnormal)

Key Insight: "Image → Features → Decision" is the universal pipeline in medical computer vision. Every method in this lecture fits into one of these three stages.

Lecture Objectives

By the end of this lecture, you will be able to apply every classification method below using real numeric calculations — not just conceptual understanding.

1

Build Feature Vectors

Construct a feature vector $\mathbf{x = [x_1, x_2, ..., x\_d]}$ from simple image measurements (intensity, edge strength).

2

Apply Classification Rules

Use KNN, Bayes, Linear, and ANN rules with full numeric worked examples using one consistent dataset.

3

Compute Evaluation Metrics

Calculate Accuracy, Precision, and Recall from a confusion matrix — and interpret results in a medical context.

4

Choose Models Wisely

Understand how model choice and threshold tuning affect clinical reliability — balancing false negatives vs. false positives.

Feature Extraction

What Is Feature Extraction?

Feature extraction is the process of transforming raw image data (a 2D or 3D array of pixel intensities) into a compact, fixed-length numerical vector that captures the most diagnostically relevant information from an image or region of interest (ROI).

Rather than feeding thousands of raw pixels into a classifier — which would be computationally prohibitive and statistically unreliable — we compute a small set of **meaningful measurements** (features) that summarize what matters for the classification task.

What Information Do Features Encode?

Intensity Features

Mean, variance, and histogram statistics of pixel brightness — describes overall tissue density.

Edge / Gradient Features

Mean edge strength from Sobel/Canny operators — captures boundary sharpness and structural irregularity.

Texture Features

GLCM entropy, contrast, homogeneity — describes surface pattern variation in tissue.

Shape / Geometry Features

Area, perimeter, circularity of segmented regions — quantifies morphology.

The Feature Vector — Mathematical Representation

Every image or region of interest (ROI) is represented as a single point in *d*-dimensional feature space. This vector is the input to every classifier.

Formal Definition

A feature vector is written as:

$$\mathbf{x} = [x_1, x_2, \ldots, x_d]^T \in \mathrm{R}^d$$

Where each x_d is a scalar measurement extracted from the image. The vector **x** fully represents the image for classification — the original pixels are discarded.

Classification Goal

A classifier is a function *f* that maps:

$$f(\mathbf{x}) = \hat{y}, \quad \hat{y} \in \{\text{Normal}, \text{Abnormal}\}$$

Where $\hat{y}$ is the **predicted class label**.

Intuition: Feature Space

When *d=2*, each case becomes a point on a 2D plane. Cases of the same class cluster together, and the classifier draws a **decision boundary** to separate them.

d = 2

2 features → 2D scatter plot (easy to visualize)

d = 10

10 features → 10D hyperspace (classifier still works)

d = 512

Deep CNN embedding → high-dimensional but powerful

In this lecture we use **d = 2** features only — to keep all math transparent and hand-computable.

Our Consistent Feature Set — Medical ROI

To keep all lecture examples connected, we use exactly **two features** extracted from a medical region of interest. Every classifier in this lecture will use this same feature definition.

Feature x_1 — Mean Intensity

The average pixel gray-level value inside the ROI. Computed as:

$$x_1 = \frac{1}{N} \sum_{i=1}^{N} p_i$$

Where p_i are pixel intensities and N is the total number of pixels in the ROI. **Higher values** → brighter region (e.g., denser or more calcified tissue).

Feature x_2 — Mean Edge Strength

The average gradient magnitude inside the ROI, computed after applying a Sobel or Canny edge operator. Formally:

$$x_2 = \frac{1}{N} \sum_{i=1}^{N} \|\nabla I_i\|$$

Higher values → sharper, more irregular boundaries (e.g., malignant mass borders vs. smooth normal tissue).

Class	Typical x_1 (Intensity)	Typical x_2 (Edge Strength) — Interpretation
Normal	Low–Moderate (45–60)	Low edges (8–15) — smooth, homogeneous tissue
Abnormal	High (70–90)	High edges (20–35) — irregular, heterogeneous structure

Training Dataset — 4 Labeled Samples

This small training set will be used consistently across **all four classifiers**. Each row is one labeled patient case represented by its 2D feature vector.

Sample ID	True Class	Feature Vector $x = [x_1, x_2]$	Interpretation
A	Normal	[50, 10]	Moderate intensity, very smooth edges
B	Normal	[55, 12]	Slightly brighter, still smooth boundary
C	Abnormal	[80, 25]	High intensity, irregular edges
D	Abnormal	[78, 30]	High intensity, very sharp irregular edges
T (Test)	**Unknown**	**[60, 14]**	Moderately elevated — which class?

Goal: Classify test sample **T = (60, 14)** using KNN, Bayes, Linear, and ANN methods — and compare their predictions at the end of the lecture.

What Is Classification ?

Formal Definition

Classification is the process of assigning an input feature vector $\mathbf{x} \in \mathbb{R}^d$ to one of K predefined discrete class labels, denoted by $\omega_1, \omega_2, \ldots, \omega_K$.

A classifier learns (or is given) a **decision rule** — a mathematical function or boundary that partitions the feature space such that every point in the space maps to exactly one class.

$$\hat{y} = f(\mathbf{x}), \qquad f : \mathbb{R}^d \to \{\omega_1, \omega_2, \ldots, \omega_K\}$$

In binary medical classification, the two class labels are typically defined as

$$\omega_1 = \textit{Normal}, \omega_2 = \textit{Abnormal}.$$

The Four Decision Rule Families

01

Distance-Based (KNN)

Classify based on proximity to known training points in feature space. No parameters to learn.

02

Probability-Based (Bayesian)

Use prior knowledge and likelihood of observed features to compute posterior class probabilities.

03

Geometric Boundary (Linear)

Calculate a hyperplane $g(\mathrm{x}) = \mathrm{w}^T\mathrm{x} + b = 0$ that best separates the classes.

04

Neural Model (ANN Neuron)

Compute weighted sum then apply a nonlinear activation function to produce a soft probabilistic output.

Confusion Matrix — Binary Classification

Before learning classifiers, we define how to **evaluate** them. The **confusion matrix** records the counts of correct and incorrect predictions for each class, giving a complete picture of a classifier's performance.

The Four Outcomes

TP — True Positive

The classifier predicted **Abnormal**, and the true label is **Abnormal**. A correct detection of disease. This is the most clinically valuable outcome.

TN — True Negative

The classifier predicted **Normal**, and the true label is **Normal**. A correct clearance — no false alarm triggered.

FP — False Positive

The classifier predicted **Abnormal**, but the true label is **Normal**. Also called a *false alarm* — leads to unnecessary follow-up or patient anxiety.

FN — False Negative

The classifier predicted **Normal**, but the true label is **Abnormal**. A *missed diagnosis* — potentially **the most dangerous error** in medical AI, as disease goes undetected.

The Confusion Matrix Grid

	Predicted: Abnormal	**Predicted: Normal**
Actual: Abnormal	TP ✅	FN ❌ (Missed!)
Actual: Normal	FP ⚠️ (False alarm)	TN ✅

Medical Priority: In disease screening, minimizing **FN** (missed cases) is critical. It is safer to over-refer (more FP) than to miss a malignancy (FN). This drives threshold design.

Evaluation Metrics — Formulas and Numeric Example

Three core metrics quantify classifier performance. Each formula captures a different aspect of the accuracy/safety tradeoff. Always compute all three — a single number never tells the full story.

Metric Formulas:

1- Accuracy — fraction of all cases classified correctly (both normal and abnormal):

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

2- Precision — of all cases flagged as abnormal, how many truly are? (quality of positive predictions):

$$\text{Precision} = \frac{TP}{TP + FP}$$

3- Recall (Sensitivity) — of all truly abnormal cases, how many did we catch? (completeness of detection):

$$\text{Recall} = \frac{TP}{TP + FN}$$

Numeric Worked Example

Given: **TP = 18, FP = 6, TN = 70, FN = 2**

Metric	Calculation	Result
Accuracy	(18+70) ÷ (18+70+6+2) = 88 ÷ 96	**0.9167 (91.7%)**
Precision	18 ÷ (18+6) = 18 ÷ 24	**0.75 (75%)**
Recall	18 ÷ (18+2) = 18 ÷ 20	**0.90 (90%)**

Interpretation: 91.7% accuracy looks good, but Precision = 75% means 1-in-4 flagged cases are false alarms. Recall = 90% means 2 diseased cases were missed (FN=2). In screening, raise Recall by lowering threshold — at the cost of more FPs.

K-Nearest Neighbors (KNN)

KNN is a *non-parametric, instance-based* classifier. It makes no assumption about the distribution of the data. Instead, it memorizes all training points and classifies a new sample by finding its **K closest neighbors** in feature space, then applying a **majority vote**.

Algorithm — Step by Step

01

Compute Distance

Measure the distance from the test point $\mathbf{x}_T$ to each training point $\mathbf{x}_i$.

02

Sort & Select K Nearest

Rank training points by distance (ascending). Keep the top K with smallest distances.

03

Majority Vote

Count class labels among K neighbors. The class with the most votes becomes $\hat{y}$.

Euclidean Distance — Formal Definition

For two points $\mathbf{x} = [x_1, x_2]$ and $\mathbf{y} = [y_1, y_2]$ in a 2D feature space, the Euclidean distance is:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

This measures the straight-line geometric distance between two feature vectors. In *d* dimensions:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

Key Properties: (1) Distance = 0 means identical features. (2) Larger distance = more different. (3) Features should be normalized so no single feature dominates due to scale.

KNN — Numeric Example: Classify T = (60, 14), K = 3

We compute the Euclidean distance from test point **T = (60, 14)** to each of the 4 training samples, then select the 3 nearest and apply majority vote.

Sample	Coordinates	Distance Calculation (Step-by-Step)	Distance d
A (Normal)	(50, 10)	$\sqrt{[(60-50)^2 + (14-10)^2]} = \sqrt{[100 + 16]} = \sqrt{116}$	**10.77**
B (Normal)	(55, 12)	$\sqrt{[(60-55)^2 + (14-12)^2]} = \sqrt{[25 + 4]} = \sqrt{29}$	**5.39**
C (Abnormal)	(80, 25)	$\sqrt{[(60-80)^2 + (14-25)^2]} = \sqrt{[400 + 121]} = \sqrt{521}$	**22.85**
D (Abnormal)	(78, 30)	$\sqrt{[(60-78)^2 + (14-30)^2]} = \sqrt{[324 + 256]} = \sqrt{580}$	**24.08**

Rank by Distance (Ascending)

1. **B = 5.39** → Normal
2. **A = 10.77** → Normal
3. **C = 22.85** → Abnormal
4. D = 24.08 → Abnormal

K = 3 Majority Vote

K=3 nearest neighbors: **B (Normal), A (Normal), C (Abnormal)**

Vote count: Normal = 2, Abnormal = 1

Majority → **$\hat{y}$ = Normal**

Bayesian Classifier — Detailed Definition

The **Bayesian classifier** is rooted in probability theory. It uses **prior knowledge** about class frequencies and **likelihood** of observing a feature vector given each class to compute a **posterior probability** — and chooses the most probable class.

Bayes' Theorem — Full Form

$$P(\omega \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid \omega)\, P(\omega)}{P(\mathbf{x})}$$

Since $P(\mathbf{x})$ is the same for all classes, the **decision rule** simplifies to:

$$\hat{y} = \arg\max_{\omega} P(\mathbf{x} \mid \omega)\, P(\omega)$$

Three Key Terms

Prior P(ω)

The probability of class ω *before* seeing any features — based on disease prevalence in the population.

Likelihood P(x|ω)

The probability of observing feature vector x *given* the patient belongs to class ω — learned from training data.

Posterior P(ω|x)

The updated probability that the patient is in class ω *after* observing their features x. We choose the class with the highest posterior.

Why Priors Matter in Medicine

If a disease is rare (e.g., prevalence = 1%), then even with a strong positive test, the **posterior probability of disease may still be low** due to the small prior. This is why Bayesian reasoning is essential in medical AI — it integrates epidemiological knowledge.

Example: P(Cancer) = 0.01 (1% prevalence). Even if a classifier has 95% sensitivity, many positives will still be false positives. The Bayesian framework correctly down-weights predictions when the prior is small.

Bayesian Classifier — Numeric Example for T = (60, 14)

We are given simplified (pre-computed) prior and likelihood values. In practice, likelihoods are estimated from training data using Gaussian models or kernel density estimation.

Given Values

Parameter	Value
P(Normal) — Prior	0.7 (70% of cases are normal)
P(Abnormal) — Prior	0.3 (30% of cases are abnormal)
P(T | Normal) — Likelihood	0.04
P(T | Abnormal) — Likelihood	0.02

Compute Unnormalized Posteriors

Apply the decision rule $\hat{y} = \arg\max_\omega P(\mathbf{x}|\omega) \cdot P(\om$

Class	Score = Likelihood × Prior	Result
Normal	0.04 × 0.7	**0.028**
Abnormal	0.02 × 0.3	**0.006**

Comparison: 0.028 > 0.006

Decision: ŷ = Normal ✅

Insight: Even though the likelihoods differ by only 2×, the **prior P(Normal) = 0.7** amplifies the Normal score, strongly pulling the decision toward Normal. If the prior were P(Abnormal) = 0.7, the result would reverse.

Linear Classifier — Detailed Definition

A **linear classifier** defines a decision boundary as a straight line (in 2D), a plane (in 3D), or a *hyperplane* (in d dimensions). Points on one side of the boundary are classified as one class; points on the other side as the other class.

The Linear Discriminant Function

The **discriminant function** $g(\mathbf{x})$ computes a real-valued score for each input vector:

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = w_1 x_1 + w_2 x_2 + b$$

Where:

- **w = [w_1, w_2, ..., w_d]** is the weight vector — learned from training data. It defines the orientation (direction) of the decision boundary.
- **b** is the bias (intercept) — shifts the boundary away from the origin.
- **g(x) = 0** defines the boundary itself — the set of all points equidistant from both classes.

Decision Rule

$$\hat{y} = \begin{cases} \text{Abnormal} & \text{if } g(\mathbf{x}) \geq 0 \\ \text{Normal} & \text{if } g(\mathbf{x}) < 0 \end{cases}$$

Geometric Interpretation

In 2D feature space (x_1 = intensity, x_2 = edge strength), the boundary line is:

$$w_1 x_1 + w_2 x_2 + b = 0$$

Points with $g(\mathbf{x}) > 0$ lie on the **Abnormal side** of the line. Points with $g(\mathbf{x}) < 0$ lie on the **Normal side**.

How are w and b learned? Methods include: Least Squares (direct solution), Perceptron learning rule (iterative), Support Vector Machine (maximize margin), or Logistic Regression (probabilistic). In this lecture, **w and b are given** — focus is on applying the rule.

Linear Classifier — Numeric Example for T = (60, 14)

We are given a pre-trained weight vector and bias. We apply the linear discriminant function and compare the score to the decision threshold of 0.

Given Parameters

Parameter	Value
Weight vector w	[0.2, 1.0] → $w_1 = 0.2$, $w_2 = 1.0$
Bias b	−25
Test sample T	[$x_1 = 60$, $x_2 = 14$]

Step-by-Step Calculation

Apply: $g(T) = w_1 x_1 + w_2 x_2 + b$

Step	Calculation
$w_1 \times x_1$	0.2 × 60 = **12.0**
$w_2 \times x_2$	1.0 × 14 = **14.0**
Add bias b	12.0 + 14.0 + (−25) = **1.0**
g(T)	**1.0**

Apply decision rule: g(T) = 1.0 ≥ 0

Decision: ŷ = Abnormal ⚠

Note: The score g(T) = +1.0 is barely above the threshold of 0. A small change in w or b could flip this decision. This sensitivity is why threshold tuning and cross-validation are essential in medical deployment.

ANN Neuron — Detailed Definition

An **Artificial Neural Network (ANN) neuron** is the fundamental computational unit of neural networks. It mimics a biological neuron by integrating weighted inputs and producing a nonlinear output. A single neuron is mathematically equivalent to a linear classifier with a nonlinear activation applied.

Two-Stage Computation

Stage 1 — Linear Aggregation (Pre-activation):

$$z = \mathbf{w}^T \mathbf{x} + b = w_1 x_1 + w_2 x_2 + b$$

This is identical to the linear discriminant. z is the raw score — it can range from $-\infty$ to $+\infty$.

Stage 2 — Nonlinear Activation (Sigmoid):

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

The sigmoid function **squashes** any real number z into the range $(0, 1)$, making it interpretable as a **probability**.

Sigmoid Function — Properties

$\sigma(z) \to 0$

When $z \to -\infty$: output ≈ 0. Strong evidence for Normal class.

$\sigma(0) = 0.5$

When $z = 0$: output = exactly 0.5. Maximum uncertainty — decision boundary point.

$\sigma(z) \to 1$

When $z \to +\infty$: output ≈ 1. Strong evidence for Abnormal class.

Decision Rule

$$\hat{y} = \begin{cases} \text{Abnormal} & \text{if } \sigma(z) \geq 0.5 \\ \text{Normal} & \text{if } \sigma(z) < 0.5 \end{cases}$$

Note: $\sigma(z) \geq 0.5$ is exactly equivalent to $z \geq 0$. The threshold can be adjusted (e.g., 0.7) to favor Precision or Recall.

ANN Neuron — Numeric Example for T = (60, 14)

We reuse the same z = 1.0 computed in the linear classifier slide — the pre-activation score is identical. We now apply the sigmoid to convert it to a probability.

Stage 1: Compute z (Pre-activation)

Using same weights w = [0.2, 1.0], b = −25, T = (60, 14):

$$z = 0.2(60) + 1.0(14) - 25 = 12 + 14 - 25 = 1.0$$

Stage 2: Apply Sigmoid Activation

Compute *sigma(z)* for *z = 1.0*:

$$\sigma(1) = \frac{1}{1 + e^{-1}} = \frac{1}{1 + 0.3679} = \frac{1}{1.3679} \approx 0.731$$

Step	Value	Note
e^{-1}	0.3679	Euler's number e ≈ 2.718
$1 + e^{-1}$	1.3679	Denominator
σ(1)	**0.731**	73.1% probability of Abnormal

Decision: σ(1) = 0.731 ≥ 0.5 → ŷ = Abnormal ⚠. The neuron outputs **73.1% confidence** that T is abnormal. This soft output is more informative than a binary prediction alone — a clinician can use the 0.731 value to gauge certainty and adjust follow-up protocols.

Comparing All Four Classifiers — Results for T = (60, 14)

Each method applies a different mathematical framework to the same test point T = (60, 14). The disagreement is expected — it illustrates why model selection and validation matter in medical AI.

Method	Prediction ŷ	Key Computation	Key Assumption / Requirement
KNN (K=3)	Normal ✅	3 nearest: B=5.39, A=10.77, C=22.85 → 2 Normal votes	Relies on dense, representative training data near T
Bayes (Given Likelihoods)	Normal ✅	Score Normal=0.028 > Score Abnormal=0.006	Priors must reflect true population prevalence
Linear Rule	Abnormal ⚠️	$g(T) = 0.2(60)+1.0(14)-25 = +1.0 \geq 0$	Assumes a linear boundary separates classes globally
ANN Neuron (Sigmoid)	Abnormal ⚠️	$\sigma(1.0) = 0.731 \geq 0.5$	Same linear boundary as above + soft probability output

Clinical Takeaway: The 2-vs-2 split shows that T = (60, 14) sits in an ambiguous boundary region. In clinical deployment, such a case would trigger human expert review. No single model should make autonomous decisions in high-stakes zones.

Choosing a Classifier in Medical Context

There is no universally "best" classifier. The choice depends on the clinical task, data characteristics, and the relative cost of different errors. These four criteria guide responsible medical AI deployment.

Maximize Recall When Missing Disease Is Dangerous

In screening tasks (e.g., cancer detection, diabetic retinopathy), a missed positive (FN) can delay treatment and harm the patient. Prioritize **Recall = TP / (TP + FN)**. Accept more false alarms (FP) to ensure near-zero FN. Tune the decision threshold downward (below 0.5 for sigmoid).

Balance Precision to Manage False Alarm Burden

Excessive false positives (FP) cause unnecessary biopsies, patient anxiety, and system cost. Once Recall meets clinical requirements, optimize **Precision = TP / (TP + FP)**. The F1-score (harmonic mean of Precision and Recall) provides a single balanced measure.

Validate on Unseen, Diverse Data

A classifier trained on one hospital's scanner may fail on a different device or population. Always validate on held-out data from different sources. Use **k-fold cross-validation** and report performance on external test sets, not training sets.

Use Threshold Tuning for FN vs. FP Tradeoff

For sigmoid-output classifiers: lowering the threshold (e.g., 0.3 instead of 0.5) reduces FN at the cost of more FP. Plot the **ROC curve** (TPR vs. FPR at all thresholds) and select the operating point that meets the clinical requirement before deployment.

Homework 6 — Mathematical Problems

Solve all four parts. Show every calculation step. No coding required. Answers without shown work will not receive credit.

DUE: NEXT LECTURE

ALL PARTS MANDATORY

Homework — Part A: Evaluation Metrics

Given Confusion Matrix Values

Parameter	Value
True Positives (TP)	25
False Positives (FP)	10
True Negatives (TN)	60
False Negatives (FN)	5

Questions — Show All Steps

1 Compute Accuracy

Use the formula: Accuracy = (TP + TN) ÷ (TP + TN + FP + FN). Substitute all four values and simplify to a decimal and percentage.

2 Compute Precision

Use: Precision = TP ÷ (TP + FP). Interpret: out of all flagged abnormal cases, what fraction are truly abnormal?

3 Compute Recall

Use: Recall = TP ÷ (TP + FN). Interpret: out of all truly abnormal cases, what fraction did the system catch?

Homework — Part B: KNN Distance Calculation

Use the same training set from lecture. Classify a new test point $\mathbf{T_2 = (70, 20)}$ using KNN with K = 3.

Training Set (Same as Lecture)

ID	Class	Feature Vector
A	Normal	[50, 10]
B	Normal	[55, 12]
C	Abnormal	[80, 25]
D	Abnormal	[78, 30]

Test point: $\mathbf{T_2 = (70, 20)}$

Questions — Show All Steps

1 Compute All Four Distances

For each training sample (A, B, C, D), compute: $d(T_2, \text{sample}) = \sqrt{[(70 - x_1)^2 + (20 - x_2)^2]}$. Show the expansion of squares before taking the square root.

2 K = 3 Classification

Rank all four samples by distance. Identify the 3 nearest neighbors. Apply majority vote and report the predicted class $\hat{y}$ for T_2.

Homework — Part C: Bayes Decision + Part D: Linear Rule & Sigmoid

Part C — Bayesian Classifier for $T_2 = (70, 20)$

Given:

- P(Normal) = 0.6
- P(Abnormal) = 0.4
- $P(T_2 \mid \text{Normal}) = 0.01$
- $P(T_2 \mid \text{Abnormal}) = 0.03$

---

1 Compute Both Scores

Score(Normal) = $P(T_2|\text{Normal}) \times P(\text{Normal})$. Score(Abnormal) = $P(T_2|\text{Abnormal}) \times P(\text{Abnormal})$. Show multiplication.

2 Choose Predicted Class

Compare the two scores. Report ŷ. Explain in one sentence why the prior P(Abnormal) = 0.4 changes the result compared to the lecture example where P(Abnormal) = 0.3.

Part D — Linear Rule + Sigmoid for $T_2 = (70, 20)$

Given: $w = [0.2, 1.0]$, $b = -25$, $T_2 = (70, 20)$

---

1 Compute $g(T_2)$

Apply: $g(T_2) = w_1(70) + w_2(20) + b$. Show each multiplication and the final sum.

2 Compute $\sigma(g(T_2))$

Apply: $\sigma(z) = 1 \div (1 + e^{-z})$ using $z = g(T_2)$. Use $e^{-z} \approx$ value from calculator or table. Show denominator and division.

3 Classify with Threshold 0.5

If $\sigma(z) \geq 0.5 \rightarrow$ Abnormal. If $\sigma(z) < 0.5 \rightarrow$ Normal. Report the class and note: does this agree with the lecture result for T = (60, 14)?

Thank you

Google Classroom