Discussion Session Problems 7

3/5/2026

1 K-Means Clustering

Question 1. K-means is not deterministic.

a) True

b) False

(a) True. K-means depends on random initialisation of cluster centroids, so different runs can converge to different local minima and produce different cluster assignments.

Question 2. K-means works:

a) better with data where clusters have similar variance

b) better with data where each cluster has its own variance

c) equally as well with both

Question 3. What do you do with the knowledge that K-means can get stuck in a local minimum?

Run K-means multiple times from different random starting points.

Question 4. It would be helpful to normalise all features in the dataset to have the same variance (unit variance) before running K-means.

a) True

b) False

(a) True. Features with larger scales dominate the Euclidean distance calculation. Normalizing to unit variance ensures every feature contributes equally to the distance metric and avoids the curse of dimensionality being worsened by scale imbalances.

2 K-Nearest Neighbours

Question 5. K-Nearest Neighbours (KNN), a model that assigns a data point to the class of its K nearest neighbours, will often work *better* than softmax classification for highly nonlinear data.

a) True

b) False

(a) True. KNN is a non-parametric method that naturally captures complex, nonlinear decision boundaries without assuming any functional form, whereas softmax regression learns a linear boundary in the input space.

Question 6. A K-Nearest Neighbours model will often work *better* than softmax classification for data with many features.

a) True

b) False

(b) False.

3 Principal Component Analysis (PCA)

Question 7. If we run PCA on a dataset it forms linear combinations of the features, allowing us to run linear regression with weights of just $+1$ and -1.

a) True

b) False

(b) False.

Question 8. What would happen if you run PCA without normalising the dataset?

If features are on very different scales, the feature with the largest variance will dominate the first principal component — PCA will mostly capture scale differences rather than meaningful structure. For example, income in dollars would completely overshadow a binary 0/1 variable. Always standardise features to zero mean and unit variance before running PCA.

Question 9. What does it mean if you get zero eigenvalues in the correlation matrix when running PCA?

A zero eigenvalue indicates a non-trivial null space. Concretely, for each zero eigenvalue, one of the original features is an exact linear combination of the others (i.e. the features are linearly dependent). This means the data lies on a lower-dimensional subspace, and no variance exists along that direction.

Question 10. A principal components analysis was run and the following eigenvalue results were obtained:

$$\lambda = [2.731,\ 2.218,\ 1.442,\ 0.009,\ 0.00183,\ 0.00085]$$

How many components would you retain?

a) 2

b) 3

c) 4

d) 5

(b) 3. The total variance is $2.731 + 2.218 + 1.442 + 0.009 + 0.00183 + 0.00085 \approx 6.403$. The first three components explain $\frac{2.731+2.218+1.442}{6.403} \approx 98.3\%$ of the variance. The remaining three eigenvalues are near zero and add negligible information, so retaining 3 components is the natural choice.

4 Anomaly Detection

Question 11. When do you decide to employ an unsupervised anomaly detection algorithm instead of a supervised learning algorithm?

Use unsupervised anomaly detection when you do **not have enough labelled examples of anomalies** to train a supervised classifier. If fraudulent transactions or equipment failures are extremely rare, a supervised model will struggle to learn what anomalies look like. Anomaly detection instead models what "normal" looks like and flags deviations.

5 Recommender Systems

Question 12. Collaborative filtering is most useful in cases where users' tastes can change and evolve over time.

a) True

b) False

(a) False. Collaborative Filtering works best when user preferences are relatively stable, because it recommends items based on similar users' past behavior.

If users' tastes change frequently over time, past interactions become less reliable, which can reduce the effectiveness of collaborative filtering.

Question 13. Embeddings and Similarity. A collaborative filtering model has been trained *solely* on user star ratings — it has no access to movie metadata such as genre, director, cast, or plot description. After training, two movies are found to have very similar learned embedding vectors $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(k)}$. What is the most accurate interpretation?

a) The two movies were directed by the same person

b) The two movies have been rated by the same users

c) The two movies share similar latent characteristics as inferred from user rating patterns

d) The two movies belong to the same genre as specified in their metadata

(c). In collaborative filtering, embedding vectors are learned entirely from rating patterns — not from hand-crafted metadata like genre or director. Two movies with similar embeddings have attracted similar responses from users, suggesting they share latent characteristics (e.g. tone, pacing, style) that the model has inferred on its own. This is also the basis for finding related items: compute $\|\mathbf{x}^{(i)} - \mathbf{x}^{(k)}\|^2$ (or cosine similarity) across all items and return the top-K most similar.

Question 14. Content-Based Filtering — Worked Prediction. A platform has three movies with known feature vectors capturing [action, romance] levels on a 0–1 scale. User A has learned weights $\mathbf{w}^{(A)} = [0.9,\ 0.1]$ and bias $b^{(A)} = 0.5$.

(a) Using the prediction formula $\hat{y}^{(i,j)} = (\mathbf{w}^{(j)})^\top \mathbf{x}^{(i)} + b^{(j)}$, compute the predicted rating for each movie below. Which movie does the model recommend to User A, and does it make intuitive sense?

Movie	$\mathbf{x}^{(i)}$	$\hat{y}^{(A,i)}$
Movie 1	[0.8, 0.1]	_____
Movie 2	[0.2, 0.9]	_____
Movie 3	[0.7, 0.3]	_____

(b) A brand new movie is added with no ratings yet. Can content-based filtering still recommend it? Can collaborative filtering? Explain the difference in one or two sentences — this is the **cold-start problem**.

(a) Applying $\hat{y} = \mathbf{w}^\top \mathbf{x} + b$:

$$\begin{aligned}
\text{Movie 1:} \quad & 0.9(0.8) + 0.1(0.1) + 0.5 = 0.72 + 0.01 + 0.5 = \mathbf{1.23} \\
\text{Movie 2:} \quad & 0.9(0.2) + 0.1(0.9) + 0.5 = 0.18 + 0.09 + 0.5 = \mathbf{0.77} \\
\text{Movie 3:} \quad & 0.9(0.7) + 0.1(0.3) + 0.5 = 0.63 + 0.03 + 0.5 = \mathbf{1.16}
\end{aligned}$$

Movie 1 is recommended. This makes intuitive sense: User A heavily weights action ($w_1 = 0.9$) and Movie 1 has the highest action score.

(b) Content-based filtering *can* still recommend the new movie because it only needs the item's feature vector $\mathbf{x}^{(i)}$ — no ratings required. Collaborative filtering *cannot*, because it learns $\mathbf{x}^{(i)}$ from observed ratings; a movie with zero ratings has no signal. This is the cold-start problem for new items.

Question 15. Mean Normalization. The table below shows ratings (1–5 stars). "?" means not yet rated. User C has no ratings at all.

	User A	User B	User C	μ_i
Movie 1	4	5	?	_____
Movie 2	2	?	?	_____
Movie 3	5	4	?	_____

(a) Compute the per-item mean $\mu_i = \frac{\sum_j r(i,j)\, y^{(i,j)}}{\sum_j r(i,j)}$ for each movie. Then compute the normalized rating $\tilde{y}^{(i,j)} = y^{(i,j)} - \mu_i$ for every observed entry.

(b) Without mean normalization, what rating would the model predict for User C on Movie 1 (whose $\mathbf{w}$ and b are initialized to zero)? With mean normalization the final prediction is $\hat{y}^{(i,j)} = (\mathbf{w}^{(j)})^\top \mathbf{x}^{(i)} + b^{(j)} + \mu_i$. What does this predict for User C, and why is it more sensible?

(a)

$$\mu_{\text{Movie 1}} = \tfrac{4+5}{2} = 4.5, \quad \mu_{\text{Movie 2}} = \tfrac{2}{1} = 2.0, \quad \mu_{\text{Movie 3}} = \tfrac{5+4}{2} = 4.5$$

Normalized ratings $\tilde{y} = y - \mu_i$:

	User A	User B
Movie 1	$4 - 4.5 = -0.5$	$5 - 4.5 = +0.5$
Movie 2	$2 - 2.0 = \ \ 0.0$	—
Movie 3	$5 - 4.5 = +0.5$	$4 - 4.5 = -0.5$

(b) Since User C has never rated anything, $r(i, j) = 0$ for every item i. This means User C never appears in the cost function sum, so the gradients with respect to $\mathbf{w}^{(C)}$ and $b^{(C)}$ are always zero throughout training — the parameters receive no update signal and stay at their initialized values: $\mathbf{w}^{(C)} = \mathbf{0}$, $b^{(C)} = 0$.

Without mean normalization: $\hat{y} = \mathbf{0}^\top \mathbf{x} + 0 = 0$, predicting a rating of 0 — outside the 1–5 scale and completely uninformative. With mean normalization: $\hat{y} = \mathbf{0}^\top \mathbf{x} + 0 + \mu_{\text{Movie 1}} = 4.5$. For a user we know nothing about, predicting the average rating for that movie is the most reasonable default.