Recommender systems: neural, and evaluated honestly

MovieLens 1M, four protocols, one model

LECTURE 22 LECTURE NOTES

Applications of Machine Learning

BSc Mathematics of Artificial Intelligence · Third year

Where we are

Last lecture predicted ratings, and did it well: RMSE 0.8602 from a rank-8 factorisation fitted on the observed ratings alone, against 1.1185 for the global mean — with the bias model and item–item neighbourhoods in between. It ended on implicit feedback, where nobody ever says no.

> **BUT NOBODY DEPLOYS A RATING PREDICTOR**
>
> A system does not fill in a matrix. It picks **ten items** out of a catalogue and puts them in front of a person. That is a *ranking* problem over items the user has *not* interacted with — and RMSE measures neither of those things.

So we change the question, keep the machinery, and discover that the change is not cosmetic.

Today

1. From rating prediction to top-N, and what the metric becomes
2. **Two-tower models** — Lecture 20's bi-encoder with users in place of queries, and what the towers buy: the cold start
3. **Evaluation** — the protocol as an experimental variable, measured on one unchanged model
4. Popularity bias, and what an offline number is worth
5. **Design exercise** — a news app's personalised feed with freshness, in nine steps, with the pitfalls and how to address them

Open Lecture 22 in Colab

No derivation. Part V's mathematics was in Lectures 19 and 21; this one is measurement, which is the other half of the course.

The task, restated

	Lecture 21	Lecture 22
Question	what would they rate it?	what should we show?
Output	a number per pair	an ordered list of ten
Evaluated on	ratings they gave	items they had not yet touched
Metric	RMSE	HR@10, NDCG@10 — Lecture 19's
Data	explicit ratings	implicit interactions

Row three is the one that changes everything. The evaluation set is no longer a sample of what the user did — it is a claim about what they *would* do, on items they never saw.

Precisely why RMSE fails here

Lecture 21's RMSE of 0.8602 was computed on *held-out ratings* — pairs where the user chose to watch the film and chose to rate it.

- A recommender operates on the complement of that set: films the user has **not** chosen
- RMSE weights every prediction equally, and a recommender only shows ten items — being accurate on the 3-star films costs nothing and buys nothing
- And it is a *pointwise* error, whereas showing ten items is a decision about *order*

> **A CONCRETE CASE**
>
> A model that predicts every rating 0.3 stars too low has a poor RMSE and **a perfect ranking**: a constant offset changes no order at all. The two metrics can disagree completely.

The data, binarised

To make MovieLens implicit, call a rating of 4 or 5 an interaction and discard the rest:

Ratings	1,000,209
Positive interactions	575,281
As a fraction of ratings	57.5%
Cells with no interaction	97.4%

✗ Note what the threshold did: a film a user rated 3 is now indistinguishable from one they never saw. We have thrown away the only genuine negatives in the dataset in order to simulate the situation of a real system, which never had them.

HR@10, defined precisely

Precision matters here, because the whole lecture turns on small differences in definition:

1. Hold out exactly **one** interaction per user; call its item g
2. Score the candidate set for that user; remove every item they interacted with in the *training* data
3. Take the top ten. $\mathrm{HR} = 1$ if g is among them, else 0
4. Average over users — macro, one vote each

With one held-out item, HR@10 *is* recall@10, and NDCG@10 is $1/\log_2(\mathrm{rank}+1)$ when hit. Both come straight from Derivation 19 with $|R| = 1$.

✗ Step 2 is where the candidate set is defined, and the candidate set is half of what this lecture measures.

Why already-seen items must be excluded

A small implementation detail with a large effect, and a good example of how a protocol leaks.

- Leave them in and the popularity baseline recommends the blockbusters every heavy user has already watched — scoring nothing, but occupying the ten slots
- Leave them in for the factorisation and it recommends what the user already interacted with, because that is what it was trained to score highly
- Which of the two is hurt more depends on the model, so the choice is not neutral between them

We exclude them everywhere, and say so. It is the only line of the popularity baseline that varies by user, and without it the baseline would not be a baseline.

The baseline that must be in every table

Recommend the ten most popular items. To everybody. Every time.

- No personalisation. No model. No parameters
- Minus the items this user has already interacted with, which is the only thing in it that varies by user

It sounds like a straw man. It is not: a large reproducibility study of eighteen published neural recommenders found most of them matched or beaten by well-tuned classical baselines, and popularity alone beats a substantial fraction, and by the end of this lecture you will see exactly how that happens.

> **EVERY NUMBER NEEDS A BASELINE, AND HERE MOST OF ALL**
>
> A recommender that does not beat popularity has not been shown to personalise anything.

It is Lecture 20, with the query replaced

$$\text{score}(u, i) = E_{\text{user}}(u) \cdot E_{\text{item}}(i)$$

	Lecture 20	Here
Left tower	the query text	the user — id, history, context
Right tower	the document	the item — id, genre, text
Precomputed	document vectors	item vectors
At request time	encode the query	encode the user

Identical architecture, identical indexability argument, identical approximate-nearest-neighbour problem. **Retrieval and recommendation converge here**, and that convergence is why Part V is one part rather than two.

What the towers can hold

- **Matrix factorisation is the special case** where each tower is a lookup table: $E_{\text{user}}(u) = p_u$, $E_{\text{item}}(i) = q_i$, and there the resemblance ends
- A tower can instead be a network over *features*: the item's genre, its text, its age; the user's recent history, device, time of day
- Which is what buys the cold start: a brand-new item has no q_i to look up, but it does have a genre and a description

THE HONEST SUMMARY OF "NEURAL RECOMMENDERS"

The architecture is a dot product of two learned vectors, exactly as before. What is new is **what the vectors are computed from** — and on datasets with no features beyond ids, there is nothing for the network to compute from, which is a large part of why the reproducibility results came out as they did.

Training: the softmax over the catalogue

$$\mathcal{L} = -\log \frac{\exp(s_{ui})}{\sum_{j \in \text{catalogue}} \exp(s_{uj})}$$

The natural objective: the observed item should win against every other item. It is the cross-entropy of Lecture 17 with the catalogue as the classes.

✗ And it is unaffordable. The denominator runs over 3,706 films here, and over tens of millions in a real catalogue — per interaction, per step.

So in practice the denominator is estimated from a sample of the catalogue rather than summed over all of it.

Cold start, and what the towers buy

	Lookup table (MF)	Feature tower
New item	✗ no vector at all	encode its genre and text
New user	✗ no vector at all	encode context and first actions
Tail item	estimated from very few interactions	shares parameters with similar items

35.5% of films here have fewer than twenty interactions, so row three is more than a third of the catalogue.

This — not accuracy on the head — is the honest case for a neural recommender. And it is invisible to a metric dominated by popular items, which is one more reason the protocol decides what progress looks like.

Two decisions nobody reports

To evaluate a recommender you must decide two things, and papers routinely state neither:

1. **Which interaction is held out?** One chosen at random from the user's history, or their *last* one in time
2. **Against what is it ranked?** The whole catalogue, or the held-out item against a sample of, say, 100 items the user did not interact with

Two binary choices, four protocols. We are going to run the **same model on the same data** under all four.

> **GUESS FIRST**
>
> How much do you think the four numbers can differ? A few percent? A factor of two? Hold the guess until the table.

Decision 1 · Which interaction is held out

	Random (leave one out)	Temporal (leave last)
Held out	one interaction, chosen uniformly	the user's most recent
Training set contains	**✗ the user's future**	only their past
Matches deployment	**✗ no**	**yes ✓**
Prevalence in the literature	very common	less common

Under a random split the model may have been trained on films the user watched *after* the one it is being asked to predict. No deployed system ever has that, and it is a leak of exactly the kind Lecture 3 was about — just harder to see, because no column was duplicated.

The random split, as a leak

Lecture 3 defined a leak: a quantity computed from rows outside the training set enters the training path. With time in the data, a deployed system's training set is everything before the moment it predicts. Check the definition against the random split.

- The model is asked to predict interaction t
- It was trained on interactions $t+1, t+2, \ldots$ by that same user
- A deployed system has none of them, because they have not happened

IT IS A LEAK, AND IT LOOKS LIKE A CONVENTION

No column was duplicated, no target was copied, nothing was fitted before the split. The leak is in the *shape* of the evaluation, which is why it survived so long as a default — and why the effect is measured rather than argued in the next few slides.

A random split over interactions

FAILURE CONDITION

Split a user's interactions at random and the model is trained on what they did afterwards and tested on what they did before. The score improves, the protocol is the reason, and no error is raised because every row is genuinely that user's.

Decision 2 · Against what is it ranked

	Full catalogue	Sampled, 100 negatives
Candidates ranked	3,706	101
Cost	score every item	score 101
Matches deployment	**yes — the system ranks the catalogue ✓**	**✗ no**
Why anyone does it	—	it was expensive, once

Sampling was a computational shortcut from a time when scoring a catalogue was expensive. Ranking one held-out item against 100 random others is a far easier task than ranking it against 3,706 — and the 100 are drawn uniformly, so they are mostly obscure films nobody would recommend.

The experiment

- One model: a rank-32 factorisation of the binarised interaction matrix
- One dataset: MovieLens 1M, 575,281 positive interactions
- One baseline: recommend the most popular items, minus what the user has already seen
- Four protocols: {random, temporal} × {full, sampled 100}

Everything is held fixed except the protocol. Whatever the numbers do, **the model did not change**.

Predict the four numbers

Before the table. You have the pieces:

- 3,706 films in the catalogue, one right answer
- Sampling replaces that catalogue with 101 candidates
- A random split lets the model see the user's later choices

Which of the two decisions do you expect to matter more? And — the question that turns out to be the interesting one — do you expect the *ordering* of the two methods to survive both?

Both effects are large; one of them is much larger. The reason to guess first is to notice, afterwards, that the guess was not obvious — because a protocol that produces a wrong answer does not feel wrong from the inside.

The table

HR@10	Full catalogue	Sampled 100
Temporal · factorisation	✗ **0.0926**	0.6792
Temporal · popularity	0.0404	0.4864
Random · factorisation	0.2132	**0.8102** ✓
Random · popularity	0.0879	0.6186

✗ The same model on the same data scores 0.0926 and 0.8102 — a factor of 8.8. Nothing about the model changed. Only the two sentences describing how it was scored.

Take that apart · the split

HR@10, full catalogue	Temporal	Random	Ratio
Factorisation	0.0926	0.2132	✗ ×2.30
Popularity	0.0404	0.0879	✗ ×2.18

Switching from a temporal to a random split **more than doubles** both numbers, and does it to the personalised model and the unpersonalised baseline alike.

That last part is the tell. If the gain were the model learning something, it would not lift a popularity list by the same factor. It is the *task* getting easier: predicting a randomly chosen item from a history you have already seen most of.

How large are modelling gains, for comparison?

To read the protocol effects, you need a sense of what a *real* improvement looks like in this field.

- A new architecture reporting a 5–10% relative gain on HR@10 is a publishable result
- Our two protocol decisions moved HR@10 by **130%** and **634%** relative, on an unchanged model

✗ The undocumented choices are an order of magnitude larger than the documented contributions. That is the whole reproducibility problem in one comparison, and it is why the reproducibility study found what it found.

Take that apart · the sampling

HR@10, temporal split	Full	Sampled 100	Ratio
Factorisation	0.0926	0.6792	✗ ×7.3
Popularity	0.0404	0.4864	✗ ×12.0

Sampling is the larger effect by far. HR@10 against 100 random negatives asks "is the right item in the top ten of **101**?" — and ten of 101 is already a tenth of the candidate set.

A *random* ranker scores 0.0990 under this protocol. Against the full catalogue it scores about 0.0027. The floor moved by a factor of 37, and every method rides up with it.

Why sampling flatters unequally

The 100 negatives are drawn uniformly, and the catalogue is mostly tail: the top 1% of films hold 12.2% of interactions.

- So a uniform sample of 100 contains, in expectation, about 1 of the head films — the only items a user might plausibly have chosen instead
- A **popularity** model is competing against almost no popular items, so it wins easily
- A model whose skill is discriminating *among plausible* items has nothing to discriminate

✗ The distortion therefore depends on what kind of model you have, which is exactly the condition under which a metric can reverse an ordering. A bias that were uniform would at least preserve rankings.

What is stable across protocols

Protocol	Factorisation ÷ popularity
Temporal, full catalogue	**×2.29 ✓**
Random, full catalogue	×2.42
Temporal, sampled 100	×1.40
Random, sampled 100	×1.31

The absolute numbers spanned a factor of 8.8. The *ratio* to the baseline spans far less — from ×1.31 to ×2.42.

HENCE THE PRACTICAL RULE

Report the baseline in the same table, under the same protocol. It does not repair the protocol, and it does make your number readable by someone whose protocol differs — which is everyone.

What the numbers actually license

- Under the honest protocol — temporal, full catalogue — the factorisation scores 0.0926 and popularity 0.0404. **The model is worth 2.3 times the baseline**, and that is the real result
- Every other number in the table is larger and means less
- NDCG@10 tells the same story: 0.0464 against 0.0196

> **THE HONEST HEADLINE**
>
> “A rank-32 factorisation places the user’s next film in the top ten **9.3%** of the time, against **4.0%** for a popularity list, under a temporal split with the full catalogue ranked.” Every clause is load bearing.

The mean hides the users

HR@10 per user is 0 or 1. A mean of 0.0926 therefore means precisely this: the held-out film was in the top ten for **9.3% of users** and not for the rest.

- There is no partial credit and no distribution to inspect — the mean *is* the proportion
- Which makes it honest, and makes its confidence interval computable: it is a proportion over 6,038 users
- NDCG@10 does have a spread, and it is rarely reported

Worth saying because “report the variance” is good advice that means different things for different metrics. For HR the right companion is an interval; for NDCG it is a distribution.

So what should you actually do?

1. **Rank the full catalogue.** It is affordable now; the shortcut is a historical artefact
2. **Split in time**, per user, and say so
3. **Include popularity** and a tuned classical baseline in the same table
4. **Never quote a number from another paper** into your table without re-running it under your protocol
5. **Report coverage and the popularity profile** beside the accuracy metric

All five are cheap. None of them is standard. Every one of them can only make your headline number smaller, which is the whole difficulty.

The feedback loop

A recommender is not an observer of its data. It **produces** it:

1. The system recommends popular items, because they are the safe bet
2. Users interact with what they are shown — they cannot interact with what they are not
3. Those interactions become the next training set
4. Which makes the popular items more popular still

> **NOTHING IN THE OBJECTIVE NOTICES**
>
> Every offline metric in this lecture is computed on interactions produced by whatever system was running at the time. A model that reproduces the old system's behaviour scores well, and a model that would have shown something better scores badly — because the evidence for that something was never collected.

The long tail, and who is served

- The top 1% of films hold 12.2% of interactions, and 35.5% of films have fewer than twenty
- Recommending from the head is *correct* often enough to score well — popularity alone reaches 0.0404 honestly measured
- And it is worth nothing to the user, who had already heard of those films

THE METRIC CANNOT SEE USEFULNESS

HR@10 gives full credit for a correct popular recommendation and the same credit for a correct obscure one. If what you value is discovery, you must **measure something else as well** — coverage, the popularity distribution of what you recommend, the fraction of the catalogue ever shown.

What to report, beside the metric

- **Catalogue coverage** — what fraction of items are ever in anyone's top ten. A model with a great HR@10 and 2% coverage is a popularity list with extra steps
- **Popularity profile** — the distribution of recommended items' interaction counts, against the catalogue's
- **Per-user spread**, not just the mean — a mean HR of 0.09 can be most users at zero and a few at one
- **The protocol**, in the sentence that states the number

None of these is hard. They are absent from papers because nothing forces them to be present, and because each of them can only make the headline look worse.

And then: none of it is the real test

Every number in this lecture is **offline**: computed on a log, about a system that was never run.

- The only real measurement is an **online experiment** with users, and it measures something different — whether people engaged, returned, cancelled
- Offline and online results correlate weakly, and the literature is candid about that
- An offline metric is a *filter*: cheap enough to reject bad ideas before they reach users, not evidence that a good one works

> **THE HONEST POSITION**
>
> Use offline evaluation to decide what is worth testing. Use an online test to decide what is worth keeping. **Do not use either to make the claim the other one supports.**

The notebook for this lecture

Open Lecture 22 in Colab

- **Runs on free CPU.** The four protocols are four loops over the same fitted model
- The four-cell table reproduced, so you compute the factor rather than take it from a slide
- The popularity baseline built first, because it is four lines and it is the row that makes the rest legible

> **WHAT TO CHANGE**
>
> Raise the sampled negatives from 100 to 1,000 and re-run. Predict the direction first: the sampled column should move towards the full one.

Design exercise · in the manner of an ML system-design interview

The task: a news app’s personalised feed, with freshness

> **THE BRIEF**
>
> A news company’s app opens on a “for you” feed of ten stories, chosen per reader from the articles live right now — an article is live for a day or two, so the catalogue the model ranks is never the one it was trained on. A **ranking over items the reader has not touched**, scored on the ten slots the app shows; the business measures opens, next-day return and subscriptions.

- **Given:** the log of impressions and clicks — which stories each reader was shown, in which slot, and which they opened — plus each article’s headline, section and time
- **Scale:** millions of readers, thousands of live articles at any moment, a catalogue that turns over daily
- **Consumers of the output:** the feed, the push notification, and the newsroom

✓ Nine steps follow, each with the pitfall that catches a first attempt and what to do about it. The model at the centre is today’s: two towers and a dot product, under a protocol stated beside every number.

Step 1 — Frame it: the metric at the cut-off the product shows

- The question is *what should we show*, not *what would they rate it*: an ordered list of ten, evaluated on articles the reader had not touched, by the hit rate and NDCG at ten
- Ten because the feed shows ten; a hit at rank forty is a story nobody saw
- The baseline in every table: the most-clicked live articles, the same for everybody, minus what this reader already opened

> **PITFALL**
>
> Choosing the cut-off for the number. Recall at a hundred is larger and kinder to every model, and it measures a page nobody is shown, so a release approved on it changes nothing a reader sees. Fix the cut-off to the product before the first run, and report the metric with its protocol in the same sentence: which interaction was held out, and against what it was ranked.

Step 2 — The log: what a click is, and what a non-click is not

- The signal is implicit: a click is an interaction, and a story not clicked was never shown, or shown and passed over, or shown below the fold — no negatives, only the absence of positives
- Keep the impressions, with the slot each story occupied: unlike a ratings dataset, this log knows what the reader was offered
- A reader's profile is their recent clicks; last month's are a different reader

PITFALL — POSITION BIAS

The top slot is clicked most because it is the top slot. A model trained on raw clicks learns what the previous ranker put first, and reproduces it — the feed produces its own training data, and the same stories stay pinned. Train on clicks *given the slot*, comparing a story with the others shown in the same position, and keep a small share of impressions whose order was shuffled, so that evidence exists which the old ranker did not choose.

Step 3 — Split in time, per reader, and say so

- Hold out each reader's *last* click; train on everything before it. The training set is the reader's past, which is all a deployed system ever has
- Candidates at test time are the articles live at that moment, nothing published later
- Protocol choices are made on a validation slice of readers; the test readers are read once

> **PITFALL**
>
> Holding out one click at random. The model is then trained on what the reader opened *after* the click it is asked to predict — a leak in the shape of the evaluation, with no column duplicated and no error raised. On the course's own data a random split more than doubled the hit rate of the model and of the popularity list alike — the tell that the task got easier, not the model, and the gain promised to the product team never reaches the app.

Step 4 — Two towers, and why the item tower must be built from features

- $\text{score}(u, i) = E_{\text{user}}(u) \cdot E_{\text{item}}(i)$: the reader encoded at request time, every live article encoded as it is published and stored
- The item tower reads the headline, section, source and age; the user tower reads the recent clicks, the time of day and the device
- Matrix factorisation is the special case where each tower is a lookup table on an id — useless here, because every article is new

PITFALL

An id embedding per article. A story published an hour ago has no row in the table, so the model has no vector for what it is supposed to recommend, and the story the newsroom just filed cannot reach a feed: the cold start is not an edge case in news, it is the whole catalogue. The feature tower is what buys freshness, and it has to be measured on articles the model never saw in training, which the time split guarantees.

Step 5 — Training: the softmax over the catalogue, and the negatives it samples

- The natural objective says the clicked story should win against every other live story — the cross-entropy with the catalogue as the classes — and it is unaffordable per click, per step
- So the denominator is estimated from a sample of the catalogue. The choice of sample is a modelling decision, and it should be reported
- Sample from what was shown and passed over, and from the live catalogue weighted by exposure, not uniformly

PITFALL — WHAT A SAMPLED NEGATIVE ASSUMES

It assumes the reader would not have clicked that story. Sampled uniformly from the catalogue, the negatives are obscure stories nobody would have been shown, so the model learns to beat the tail and nothing about choosing among the plausible; and a story the reader never saw may be the one they would have opened. Being unclicked is not evidence of disinterest: say what the sample assumes, and measure what it costs.

Step 6 — Evaluate honestly: full catalogue, time split, popularity in the same row

Decision	Honest	Flattering
Which click is held out	the reader's last	one at random
Against what it is ranked	every live article	a hundred sampled ones
What sits beside the number	popularity, same protocol	a number quoted from a paper
How it is read	a proportion of readers, with its interval	a mean, alone

PITFALL

Ranking the held-out click against a hundred random articles. The floor rises by an order of magnitude, and *unequally*: the sample holds almost no popular stories, so popularity wins easily. A sampled metric can reverse the order of two systems and ship the worse one. Rank the full live catalogue, and beat popularity, or nothing has been personalised.

Step 7 — Diversity and the filter bubble, as quantities

- **Coverage:** the share of live articles that ever reach a feed. A fine hit rate with tiny coverage is the front page with extra steps
- **Popularity profile:** the click counts of what is recommended, against the catalogue's
- **Per reader:** how many sections the ten slots come from, against the sections that reader has ever opened — the bubble measured as a number the newsroom can track

> **PITFALL**
>
> The hit rate gives full credit for a correct popular story and the same credit for a correct discovery, and a feed that narrows to one section scores well on the clicks it caused, until the subscriber cancels because the app "only shows sport". If breadth is what the product values it must be measured, because nothing in the objective will notice. Report coverage and the per-reader spread beside the headline, and let the policy stage enforce a floor on sections.

Step 8 — In production: the offline number filters, the A/B test measures

- Every offline number comes from a log produced by whatever ranker was running: a model that reproduces it scores well, and one that would have shown something better scores badly, because the evidence was never collected
- Use offline evaluation to decide what is worth testing; roll out to a share of readers behind an A/B test — opens, time spent, whether they return the next day — to decide what is worth keeping
- Retrain daily, since the catalogue has turned over; on call, watch coverage and the share of feeds served by the popularity fallback

> **PITFALL — THE OFFLINE–ONLINE GAP**
>
> Shipping on an offline gain. Offline and online results correlate weakly. The feed also changes the readers it is measured on: it recommends popular stories, readers click what they are shown, those clicks train the next model, and the popular grow more popular. Read the A/B test on what readers do afterwards, and watch coverage week on week for the loop closing.

Step 9 — Ship it: the pipeline, and the pitfalls in one table

Impressions and clicks with their slot → last click held out per reader, candidates the live articles → feature towers for reader and article → sampled softmax, negatives from what was shown → hit rate at ten over the full live catalogue, popularity in the same row → coverage and per-reader spread → A/B test on a share of readers → retrain daily.

Pitfall	Address it by
A cut-off chosen for the number	the metric at the ten slots the feed shows
Position bias in the clicks	clicks given the slot; a shuffled share of impressions
A click held out at random	the last click per reader; candidates live at that moment
An id embedding for a new article	an item tower over headline, section and age
Uniform negatives	negatives from what was shown; the assumption stated
A hundred sampled candidates	the full live catalogue; popularity in the same table
The feed narrowing unseen	coverage, popularity profile, sections per reader
Shipping on an offline gain	an A/B test read on what readers do afterwards

Summary

1. A recommender ranks a catalogue for a user, so it is scored by Lecture 19's metrics on items the user had not touched — not by RMSE
2. The two-tower model is Lecture 20's bi-encoder with users in place of queries; matrix factorisation is its lookup-table special case, and feature towers are what buy the cold start
3. The **protocol** moved HR@10 from 0.0926 to 0.8102 on one unchanged model; the ratio to a popularity baseline under the same protocol is far steadier, so it goes in the same table
4. Popularity bias is a feedback loop no offline metric can see; offline numbers filter ideas, and only an online test measures the thing you care about
5. Designed a news app's personalised feed in nine steps — the metric at the slots shown, clicks read given their slot, the last click held out per reader, an item tower from features, the full catalogue ranked with popularity beside it, the bubble measured, an A/B test before anything is kept — each named for its pitfall

FAILURE CONDITION — THE PROTOCOL, NOT THE MODEL

One unchanged model reports HR@10 of 0.0926 or 0.8102 depending only on which interaction was held out and what it was ranked against. A recommender's headline is a property of the evaluation protocol at least as much as of the model, and the protocol is the half that goes unreported.

The examination

Written — two hours, closed book

- Technical exercises, closed choices, short open answers
- A · the derivations (40%)
- B · choosing a method for a stated situation (35%)
- C · reading results (25%)

Oral — optional, seven to ten minutes

- Three questions, on any topic from the course
- No notes, no computer
- You decide after seeing your written mark

The written is marked out of 30, pass at 18, capped at 27 on its own; the oral moves it up by at most three marks or down by any amount, below 18 included, and 28, 29 and 30 exist only through it. Everything in Chapters 1–10 and 12–15 and in the lecture notes is examinable; nothing marked *engineering* or *for context* is. The 110 exercises that end these decks are published with their solutions.

NOT PRESENTED — FOR AFTERWARDS

Exercises Lecture 22

five, in the style of the written paper

Exercises — Lecture 22

1 A recommender is scored by RMSE on held-out ratings. State why that is the wrong metric, using the set the model actually operates on. [5]

2 The same model scores HR@10 of 0.0926 and 0.8102 on the same data. Name the two decisions that differ, and which moves the number more. [5]

3 Explain why a sampled metric is not merely noisier than a full-catalogue one, and give the consequence for comparing two published systems. [5]

Solutions overleaf — this is the last lecture.

Exercises — Lecture 22 (continued)

4 A new film arrives with a genre and a description but no interactions. State what a matrix factorisation can do with it, what a two-tower model with a feature tower can do, and why. [4]

5 A recommender is trained on interactions produced by an earlier recommender. Name the effect, and say why no offline metric computed on that log can detect it. [4]

Solutions overleaf — this is the last lecture.

THE FIVE SET AT THE END OF LECTURE 21

Solutions
Lecture 21

not part of this lecture

Solutions — Lecture 21

1. State why the truncated SVD cannot be applied directly to a ratings matrix, naming the condition of…

✓ The Frobenius norm sums over every entry, and 95.5% of them have no value.

- The theorem is about one fully specified matrix
- Filling the gaps changes the problem into a different one with a different answer

2. A rank-32 SVD of the zero-filled matrix scores worse than predicting one number for everybody. Explain how…

✓ It is optimal for the matrix it was given, and that matrix is mostly zeros.

- Ratings run from 1 to 5, so a zero is off the scale rather than low
- The approximation spends its capacity reproducing an assumption nobody made deliberately

Solutions — Lecture 21 (2)

3. Fix Q and write the subproblem ALS solves for one user's p_u. Identify it as a familiar estimator.

✓ $\min_{p_u} \sum_{i \in I_u} (r_{ui} - p_u \cdot q_i)^2 + \lambda \|p_u\|^2$ — a ridge regression.

- The films this user rated are the rows of the design matrix and their ratings the target, so the missing entries never enter it
- No user's solution depends on another's, so the whole P step is one small regression per user

4. Validation RMSE is lowest at $d = 8$ and rises again by $d = 64$. State what this says about the rank, and on…

✓ The rank is a capacity parameter: more factors fit the training ratings better and held-out ratings worse. Choose it on ratings carved from the training half, and read the test set once.

- It is the U-curve of every capacity dial, with the rank as the dial
- A rank chosen on the test ratings makes the reported number optimistic

Solutions — Lecture 21 (3)

5. Implicit feedback has no negatives. State why an objective over observed entries alone fails, and how BPR's…

✓ It is satisfied by predicting yes everywhere. BPR asks only that an observed item outrank a sampled unobserved one, so it models differences and never claims an absolute score.

- Only positives exist, so there is nothing to push down
- A sampled 'negative' may be an item the user would love: unobserved is not evidence of dislike

ANSWERED HERE, BECAUSE THERE IS NO LECTURE 25

Solutions Lecture 22

not part of this lecture

Solutions — Lecture 22

1. A recommender is scored by RMSE on held-out ratings. State why that is the wrong metric, using the set the…

✓ RMSE is computed on films the user chose to watch and rate; a recommender operates on the complement.

- It weights every prediction equally when only ten items are shown
- A constant offset ruins RMSE and changes no ranking at all

2. The same model scores HR@10 of 0.0926 and 0.8102 on the same data. Name the two decisions that differ, and…

✓ Random against temporal split, and sampled-100 against the full catalogue. Sampling moves it far more.

- Ten of 101 candidates is already a tenth of the set
- A random ranker goes from 0.0027 to 0.0990 under the same change

Solutions — Lecture 22 (2)

3. Explain why a sampled metric is not merely noisier than a full-catalogue one, and give the consequence for…

✓ The 100 negatives are drawn uniformly from a mostly-tail catalogue, so the distortion depends on the model. Orderings can reverse.

- A popularity model competes against almost no popular items
- A uniform bias would at least preserve rankings; this one does not

4. A new film arrives with a genre and a description but no interactions. State what a matrix factorisation can…

✓ The factorisation has no vector for it at all; a feature tower encodes its genre and text, so it has a vector at once.

- Matrix factorisation is the two-tower model whose towers are lookup tables, and there is no row to look up for an item never interacted with
- The same argument covers the tail: a feature tower shares parameters with similar items, where a lookup row is estimated from very few interactions

Solutions — Lecture 22 (3)

5. A recommender is trained on interactions produced by an earlier recommender. Name the effect, and say why no…

✓ The feedback loop. The log records only what the earlier system chose to show, so it rewards reproducing that system.

- Users cannot interact with what they were not shown
- A model that would have shown something better scores badly, because the evidence for it was never collected; only an online test can see it

THE LAST LECTURE

End of the course

The examination is described on the course page, and on the slide before the exercises.