Recommender systems: from ratings to factors

MovieLens 1M

LECTURE 21 LECTURE NOTES

Applications of Machine Learning

BSc Mathematics of Artificial Intelligence · Third year

Retrieval, with the query removed

	Retrieval	Recommendation
Input	a query and a corpus	a **user** and a catalogue
Output	a ranking	a ranking
Signal	the query text	what this user did before
Labels	judged by assessors, deliberately	a **by-product of use**

Last lecture a bi-encoder put queries and documents in one space, its ranking was fused with BM25's, and a cross-encoder re-ranked the top of the list. Today the query disappears. The output is the same shape, so Lecture 19's metrics apply unchanged; the *input* is what differs, and the last row is what makes this hard: nobody rated a film in order to help you.

Today

1. **The problem and the data** — MovieLens, a matrix that is 95.5% missing, and the long tail
2. **Baselines** — how much of a rating is the user, and how much the film
3. **Derivation 21** — matrix factorisation, why the SVD of Lecture 6 cannot simply be taken, and ALS and SGD
4. **Neighbourhoods, and implicit feedback** — item–item similarity, feedback with no negatives, and BPR
5. **Design exercise** — an e-learning company's course recommender, in nine steps, with the pitfalls and how to address them

Open Lecture 21 in Colab

The task

6,040 users. 3,706 films. 1,000,209 ratings on a 1–5 scale.

Two questions, and they are not the same question:

- **Rating prediction** — what would this user give this film? Scored by RMSE. This is Lecture 21
- **Top-N recommendation** — which ten films should we show? Scored by the ranking metrics of Lecture 19. This is Lecture 22

> **THEY COME APART, AND THAT IS THE SUBJECT OF THE NEXT LECTURE**
>
> A model that predicts every rating within half a star can still recommend badly, because the films a user would rate highly are not the films they have not seen.

Why MovieLens, and what it is not

- **Explicit ratings**, which most systems do not have — but without them, rating prediction cannot be taught at all
- **Timestamps**, which is why this release and not the smaller one: half of Lecture 22 is the difference between splitting a history at random and splitting it in time
- **1,000,209 ratings** — large enough that the results are not noise, small enough to factorise on a free CPU

> **AND WHAT IT IS NOT**
>
> It is a *research dataset from a system that asked people to rate films*. It is not a log of a production recommender, its users are self-selected, and its sparsity pattern is unlike a real catalogue's. Everything measured here transfers as a **method**, not as a number.

The matrix, and what is not in it

Users × films	22.4 million cells
Ratings present	1,000,209
Density	**4.5%**
Ratings per user, median	96
Ratings per film, median	124

95.5% of the matrix is **missing**, and this is the central fact of the lecture.

✗ Missing is not zero. A cell is empty because the user never saw the film, or saw it and did not rate it, or was never shown it by whatever system was running at the time. Every one of those is a different meaning, and none of them is "disliked".

Not missing at random

Worse: the pattern of what is missing is *informative*. People rate films they chose to watch, and they chose them because they expected to like them.

- A horror film unrated by a user who hates horror is missing **because** they hate horror
- Which is a rating, of a kind — and it is not in the data
- Every model in this lecture trains on observed entries and is evaluated on observed entries, so it never sees this

> **THE SAME SHAPE AS LECTURE 19'S POOLING**
>
> The data records what someone chose to look at, and the choice was not random. In retrieval it made recall a lower bound; here it biases the training set itself.

What “a user” means in the model

p_u is a vector attached to an *account*, and an account is not a person:

- A household shares one account, so p_u averages several tastes into one point — and the average of two tastes is often nobody’s
- A person’s taste changes; p_u is a single vector with no time in it. MovieLens spans years
- The same person on two devices is two users, with two independent vectors estimated from half the data each

WORTH SAYING BECAUSE THE MODEL CANNOT

Nothing in the objective represents any of this. A model whose unit of analysis is wrong will be confidently wrong, and no amount of held-out RMSE reveals it.

What people actually give

Rating	1	2	3	4	5
Share	5.6%	10.8%	26.1%	34.9%	22.6%

Mean 3.58, and heavily skewed upward: 57.5% of ratings are 4 or 5.

People mostly rate things they liked. Which means the rating scale is not being used as a scale — it is being used as “good”, “very good”, and occasionally “I want you to know this was bad”.

Why RMSE, and its quiet assumption

$$\mathrm{RMSE} = \sqrt{\frac{1}{|\Omega_{\text{test}}|} \sum_{(u,i)} (r_{ui} - \hat{r}_{ui})^2}$$

Squared error, so a two-star miss costs four times a one-star miss. Defensible, and worth stating: it says being badly wrong occasionally is worse than being slightly wrong often.

✗ The quiet assumption is bigger. RMSE averages over the observed test ratings — films these users chose to watch and rate. It says nothing about the films they did not, which is precisely the set a recommender operates on.

Lecture 22 replaces it for that reason. Today it is the right metric for the question “what would they rate it”, and the wrong one for “what should we show”.

RMSE on what is really a ranking task

FAILURE CONDITION

RMSE scores how well you predict ratings a user actually gave. A recommender is judged on what it puts in front of a user for items they have not touched, so a model can win on RMSE and still rank the catalogue worse than a popularity list.

The long tail

Top films by rating count	Share of all ratings
top 1%	8.7%
top 5%	28.3%
top 10%	44.4%
top 20%	65.2%
Films with fewer than 20 ratings	17.9%

The top 5% of films take 28.3% of the ratings. Most of the catalogue is barely rated at all.

✗ Two consequences to keep. First, a model can score well by learning popularity and nothing else — Lecture 22 measures exactly that. Second, the films where a recommendation would be *valuable* are the ones with least data.

The tail is where the value is

A recommendation is useful to the extent the user would not have found the item anyway.

- The top 1% of films hold 8.7% of ratings. Recommending those is easy, scores well, and tells the user nothing they did not know
- 17.9% of films have under twenty ratings. These are where a recommender could earn its place
- And they are exactly where every model in this lecture has least to work with

> **THE METRIC DOES NOT KNOW THIS**
>
> RMSE over held-out ratings, and even NDCG over held-out interactions, give full credit for a correct popular recommendation and the same credit for a correct obscure one. **Nothing measured in this lecture rewards usefulness.** Lecture 22 tries.

The baselines, before any model

Every number needs a baseline. Split 90/10 at random, fit on the training half, and predict:

Predictor	RMSE
The global mean, for everyone	1.1185
This user's mean rating	1.0367
This film's mean rating	0.9826
Both, as biases	**0.9109**

A single number for everybody scores 1.1185. Knowing *which film* takes it to 0.9826; knowing *who* and *which film* takes it to 0.9109.

No interaction between the two has been modelled yet. Nothing so far knows that *this* user likes *this kind* of film.

The split, and why it is stated first

- 90/10 on the ratings, at random, one seed, fixed before any model was fitted
- 900,189 ratings to fit, 100,020 to test
- Every number in this lecture uses *that* split — baselines, neighbourhoods, factorisations, the SVD disasters

A RANDOM SPLIT OF RATINGS IS NOT INNOCENT

It puts a user's future ratings in the training set and their past in the test set, which no deployed system ever gets. It is the standard protocol for rating prediction, it is convenient, and it is optimistic. **Lecture 22 measures exactly how optimistic.**

The bias model, written out

$$\hat{r}_{ui} = \mu + b_u + b_i$$

THE GLOBAL MEAN, PLUS THIS USER'S OFFSET, PLUS THIS FILM'S

Fitted by alternating closed forms, each regularised by the count — a user with three ratings should not be given a large offset:

$$b_u = \frac{\sum_{i \in I_u} (r_{ui} - \mu - b_i)}{|I_u| + \lambda}$$

That λ in the denominator is shrinkage — a ridge penalty, λ times a squared norm added to the objective — in a new place: with few observations, pull the estimate toward zero.

What the biases learned

	Lowest	Highest
User offset b_u	-2.2156	+1.4690
Film offset b_i	-1.8545	+1.0546

The harshest user rates nearly 2.2 stars below the mean on everything, and the most generous 1.5 above. That spread is larger than most of what the rest of this lecture will add.

> **WHY THIS MATTERS MORE THAN IT LOOKS**
>
> A recommender that ignores biases spends its capacity re-learning that some users are generous and some films are popular — effects with nothing personal in them. Modelling them **explicitly and cheaply** frees the interesting part of the model for the interesting part of the problem.

Cold start, named now

A user with no ratings has no p_u to solve for; a film with no ratings has no q_i. The factorisation has nothing to say about either, and the failure is total rather than gradual.

- **New user** — fall back to popularity, or ask for a few ratings up front
- **New item** — fall back to content: genre, cast, description, an embedding of the synopsis
- **New system** — you have neither, which is why every recommender starts as a popularity list

Not examinable in detail, but the reason a real system is always a hybrid of a factorisation and something that works without history.

> **FAILURE CONDITION — A USER WITH NO HISTORY**
>
> Matrix factorisation learns a vector per user from that user's ratings. A user with none has no vector to learn, so the model returns the bias term and nothing else — and an offline metric computed on users who all have histories never contains that case.

Step 1 · The low-rank hypothesis

Assume tastes are simpler than the data: that there are d underlying factors — how much a film is a comedy, how dark, how old — and that a rating is how much the user wants each factor times how much the film has of it.

$$\hat{r}_{ui} = p_u \cdot q_i, \qquad p_u, q_i \in \mathbb{R}^d$$

Equivalently $\hat{R} = PQ^\top$ with $\mathrm{rank} \le d$. The parameter count falls from mn to $d(m+n)$, which for $d = 16$ is about a hundredth of the matrix.

✗ It is a *hypothesis*, not a fact. Its justification is that it predicts held-out ratings better than the alternatives, and we will check that rather than assert it.

Step 2 · What Eckart–Young says

Lecture 6 already answered "the best rank-d approximation", and the derivation turns on what that answer says.

Eckart–Young: *among all matrices of rank at most d, the truncated SVD minimises $\|R - X\|_F$.*

- It is a statement about **one given matrix R**
- The norm sums over every entry, with equal weight
- It says nothing about entries R does not have — the question is not defined

> **SO THE THEOREM IS NOT WRONG; IT IS BEING MISUSED**
>
> Filling the matrix does not extend the theorem to our problem. It **changes our problem** into one the theorem covers, and the two problems have different answers. The next slide prices the difference.

Step 3 · So fill them in?

The obvious repair: put something in the empty cells and take the SVD of the completed matrix. Two obvious choices, both actually tried, both measured on the same held-out ratings as before:

Rank-32 SVD of…	RMSE
the matrix filled with **zeros**	✗ **2.3274**
the matrix filled with the **global mean**	0.9920
predicting the global mean for everyone	1.1185
the bias model	0.9109

✗ Filling with zeros scores 2.3274 — more than twice the error of predicting one number for everybody, from the mathematically optimal rank-32 approximation.

Why zeros are worse than the mean

Rank-32 SVD of the matrix filled with…	RMSE
zeros	✗ 2.3274
the global mean	0.9920

Ratings run from 1 to 5, so a zero is not a low rating — it is **off the scale**. The approximation is dragged toward zero everywhere, and the predictions for observed pairs come out far too low.

Filling with the mean at least puts the fill inside the range, so the damage is bounded: 0.9920, which is merely worse than the bias model (0.9109) rather than catastrophic.

✗ Both are the same error. One is just more visible, and the more visible error is the safer one to make.

Step 4 · Why filling fails

The SVD did exactly what it promised: the best rank-32 approximation to **the matrix it was given**. That matrix is 95.5% zeros, so the best approximation is mostly a good approximation of the zeros.

- The fill value is not data. Optimising against it spends the model on reproducing an assumption
- And the assumption is a strong one: zero means “this user would rate this film 0”, on a scale that starts at 1
- Filling with the mean is less absurd — 0.9920 — and still worse than the bias model, which has $d = 0$

> THE TRANSFERABLE ERROR
>
> **Imputing missing values to make a method applicable makes the method answer a different question.** The same mistake as filling a missing feature with a column mean and then reporting the model as if the data had been complete.

Step 5 · Change the objective, not the data

The problem was never the SVD. It was that we asked for a fit to *every* entry. So ask only for what we can observe:

$$\min_{P,Q} \sum_{(u,i)\in\Omega} \left(r_{ui} - p_u \cdot q_i\right)^2 + \lambda \left(\|p_u\|^2 + \|q_i\|^2\right)$$

THE SUM RUNS OVER OBSERVED ENTRIES ONLY

The unobserved entries are now absent from the objective rather than assigned a value. Nothing is imputed, and nothing is assumed about what a user would have said. λ is a ridge penalty, $\lambda\|p_u\|^2$ added to the objective: a user with three ratings and $d = 16$ has more parameters than observations, and without it gets a perfect fit and no information.

✗ And in doing so we have left the world where the SVD applies. This objective has no closed form, no orthogonality, no unique solution — it is not an eigenproblem any more.

Step 6 · Hold one side fixed

The objective is not convex in P and Q together. But fix Q, and every term involving p_u is a quadratic in p_u alone:

$$\min_{p_u} \sum_{i \in I_u} (r_{ui} - p_u \cdot q_i)^2 + \lambda \|p_u\|^2$$

That is a **ridge regression**, with the films this user rated as the rows of the design matrix and their ratings as the target. And it is *separate* for each user: no user's solution depends on another's.

So the whole P step is m independent small regressions, each over only that user's own ratings.

Step 7 · Alternate

By symmetry, fixing P makes each q_i a ridge regression over the users who rated film i. So:

1. Fix Q; solve for every p_u
2. Fix P; solve for every q_i
3. Repeat

Each half-step **solves its subproblem exactly**, so neither can increase the objective. The objective is bounded below by zero, so the sequence converges.

Converges to a *local* minimum: monotone descent is not global optimality, and a different initialisation gives a different answer. That is the price of Step 5, and it was paid knowingly.

Step 8 · Or descend, one rating at a time

The alternative, and the one that scales past memory. For each observed rating, take the error and step both vectors:

$$e_{ui} = r_{ui} - \hat{r}_{ui}$$

$$p_u \mathrel{+}= \eta \left(e_{ui} q_i - \lambda p_u \right)$$

$$q_i \mathrel{+}= \eta \left(e_{ui} p_u - \lambda q_i \right)$$

- **ALS** parallelises across users and each step is exact — but needs a $d \times d$ solve per user
- **SGD** touches one rating at a time, so it suits implicit data and streams — but needs a learning rate

Same objective, different route. At $d = 32$: ALS reaches 0.8733, SGD 0.8615.

Step 9 · Put the biases back

The factorisation models the *interaction*. The parts that are not interactions — a generous user, a popular film — should not be spent on it:

$$\hat{r}_{ui} = \mu + b_u + b_i + p_u \cdot q_i$$

Fit the biases first, then factorise what they leave behind.

Bias model alone ($d = 0$)	0.9109
Factorisation, $d = 8$, with biases	0.8602
Factorisation, $d = 8$, no biases	0.8590

The biases alone close 80.4% of the distance from the global mean to the full model. Most of a rating is not personal.

Step 10 · An honest wrinkle

At $d = 8$, the model *without* explicit biases scores 0.8590 — slightly **better** than the one with them (0.8602).

The textbook claim is that biases help. Here they do not, and the reason is visible in the algebra: a factor dimension can hold a constant, so a rank-8 factorisation can *represent* the biases itself, using two of its eight dimensions to do it.

> **WHAT IS STILL TRUE**
>
> Biases are worth having because they are **cheap, stable and interpretable**, they help most at low rank, and they let the factors be about interaction. Not because they always lower RMSE — and the difference here, 0.0012, is small enough that reporting it as a finding either way would be overreaching.

Step 11 · How many factors?

d	1	2	4	8	16	32	64
Validation RMSE	0.8888	0.8811	0.8680	0.8591	0.8623	0.8798	0.9056

Scored on 10% of the *training* ratings, held out. Best at $d = 8$ (0.8591), rising again by $d = 64$ (0.9056). Refitted at $d = 8$ on all the training ratings, the test set is read once: 0.8602.

A U-curve, which is Lecture 5 exactly: more capacity fits the training ratings better and the held-out ratings worse. The rank is a capacity parameter, and it is chosen the way every capacity parameter in this course is chosen — on data the model did not see, and not on the test set it is reported on.

Even $d = 1$ scores 0.8888 on validation, already better than the bias model there (0.9090). One dimension of taste is worth something.

The other family: neighbourhoods

Before factorisation there was a simpler idea, and it is still deployed: predict from **similar things**.

- **User-based** — find users who rated as you did, and average their ratings for the film in question
- **Item-based** — find films rated as this film was, and average *this user's* ratings of them

Item-based is what gets deployed, for a reason that is operational rather than statistical: there are usually far fewer items than users, item–item similarities change slowly, and they can be computed offline and cached.

Item–item similarity, defined

$$s_{ij} = \frac{\tilde{\boldsymbol{r}}_i \cdot \tilde{\boldsymbol{r}}_j}{\|\tilde{\boldsymbol{r}}_i\| \, \|\tilde{\boldsymbol{r}}_j\|}$$

COSINE BETWEEN TWO FILMS' RATING COLUMNS

The tilde matters: the columns are **bias-adjusted** first, $\tilde{r}_{ui} = r_{ui} - \mu - b_u - b_i$. Without that, every popular film is similar to every other popular film, because both columns are full of fours.

✗ Which is the same trap as an unnormalised dot product in Lecture 20, and the same trap as term frequency without idf in Lecture 19: subtract what is true of everything before measuring what is specific.

The prediction, and the measurement

$$\hat{r}_{ui} = \mu + b_u + b_i + \frac{\sum_{j \in N_k(i)} s_{ij}\, \tilde{r}_{uj}}{\sum_{j \in N_k(i)} s_{ij}}$$

Method	RMSE
item–item, $k = 10$	0.9376
item–item, $k = 20$	0.9171
item–item, $k = 50$	0.8867
factorisation, $d = 8$	**0.8602**

The best neighbourhood model (0.8867) does not reach the factorisation, and still beats the bias model (0.9109).

Why factorisation wins, and where it does not

	Neighbourhood	Factorisation
Uses	direct co-rating evidence	a global low-rank structure
Sparse items	**✗ few co-ratings, poor estimate**	borrows strength from everything
Explainable	**“because you liked X” ✓**	**✗ a dot product of latent vectors**
New rating	update a cached similarity	refit, or fold in approximately

Row three is why neighbourhood methods survive: a recommendation a user can be *told the reason for* is worth points a metric does not award.

And in practice the two are combined, exactly as BM25 and dense retrieval were combined last lecture, and for the same reason.

Explicit ratings are the easy case

Everything so far assumed a number the user typed. Most real systems have none.

Explicit	Implicit
a rating, deliberately given	a click, a play, a purchase, a dwell time
scarce — most users rate nothing	abundant — every interaction is one
a scale, with a middle	presence or absence
a stated preference	a revealed one, plus the interface

MovieLens is explicit, which is why it can be used to teach rating prediction at all. It is *not* representative: your first recommender will almost certainly have no ratings in it.

Implicit feedback has no negatives

A user watched film A. What does that say about film B, which they did not watch?

- They disliked it — a true negative
- They have not seen it yet — possibly the best recommendation available
- They were never shown it — and that was the old system's choice, not theirs

> **WHICH IS THE WHOLE DIFFICULTY**
>
> In explicit data the missing entries are excluded from the objective, and we saw exactly what happens when they are not. In implicit data the **only** thing that exists is the positives, so an objective over observed entries alone is trivially satisfied by predicting "yes" for everything.

BPR: rank the pair, not the item

$$\max \sum_{(u,i,j)} \log \sigma(\hat{r}_{ui} - \hat{r}_{uj}) - \lambda \|\Theta\|^2$$

I OBSERVED FOR U, J SAMPLED FROM THE UNOBSERVED

- Only the *difference* is modelled, so no absolute score is ever claimed — and none was ever observed
- j is **sampled**, not enumerated, which is what makes it affordable
- The logistic of a score difference is the pairwise logistic loss of Lecture 4, and the sampling is the negative sampling of Lecture 20

✗ And it inherits Lecture 20's trap exactly: a sampled "negative" may be an item the user would love. Being unobserved is not evidence of dislike.

What a deployed recommender actually is

1 Candidates

catalogue → hundreds. Factorisation, popularity, co-visitation, recency — several sources, unioned.

2 Rank

hundreds → tens. A richer model with features the first stage cannot afford.

3 Policy

diversity, freshness, business rules, exploration.

The same three stages as Lecture 19's search pipeline, for the same reason: the accurate model does not scale to the catalogue. What we built today is stage 1.

Stage 3 has no equation in it and decides more of what a user sees than stages 1 and 2 together.

The distance travelled, in one number

	RMSE	Distance closed
Global mean	1.1185	0%
Bias model	0.9109	80.4%
Item–item, $k = 50$	0.8867	89.7%
Factorisation, $d = 8$	**0.8602**	**100%**

Two arithmetic means and a shrinkage parameter get you most of the way. The derivation, the alternating solves and eight latent dimensions get the rest.

✗ That ratio is not an argument against the model. It is an argument for always fitting the baseline first, so you know which part of your result is the idea and which part is arithmetic.

The notebook for this lecture

Open Lecture 21 in Colab

- **Runs on free CPU.** ALS is a $d \times d$ solve per user, and there are only 6,040 of them
- The SVD-with-zeros disaster reproduced, so you see the number rather than take my word for it
- ALS implemented from the half-step of Step 6, and checked against a per-rating SGD written from Step 8

> **WHAT TO CHANGE**
>
> Set the regularisation to zero and re-run the rank sweep. Watch where the U-curve moves, and which rank stops being the best one.

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

The task: an e-learning company's course recommender

> **THE BRIEF**
>
> An e-learning company sells a subscription to a catalogue of video courses, and wants a row "for you" on every learner's home screen. Behind it the system must estimate, for each (learner, course) pair, the **rating the learner would give the course** on the app's five-star scale, from what they and everyone else have rated so far. The business measures enrolments from the row, completions, and renewals.

- **Given:** the ratings learners left on courses they finished, with timestamps; enrolments and completions; each course's title, topic and description
- **Scale:** hundreds of thousands of learners, thousands of courses, a matrix almost entirely missing
- **Consumers of the output:** the "for you" row and the team that commissions new courses

✓ 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: biases plus a low-rank factorisation, fitted on the observed entries only.

Step 1 — Frame it: which question, and which number answers it

- Two questions, not one: *what would this learner rate this course*, scored by RMSE on held-out ratings; and *which ten courses go in the row*, a ranking over courses the learner has not touched, scored by ranking metrics. This exercise builds the first
- RMSE: a two-star miss costs four times a one-star miss — being badly wrong occasionally is worse than slightly wrong often; defensible, and worth stating
- The first baseline is one number for everybody: the global mean

> **PITFALL**
>
> Reporting RMSE to the product team as if it measured the row. RMSE averages over ratings the learner chose to give, on courses they chose to take; the row is chosen from the courses they did not. A model within half a star on every held-out rating can still fill the row worse than the most popular courses, and the release is judged on enrolments. Keep the two questions apart, and measure the row on untouched courses with the ranking metrics.

Step 2 — The matrix, and what is not in it

- Learners by courses, a cell holding a rating where one was given. Most subscribers rate nothing, a rater rates a handful of the courses they finished, and the matrix is a few per cent full at best
- An empty cell means never enrolled, or enrolled and never rated, or never shown it by the old home screen — three meanings, none of them “disliked”
- What exists in abundance is implicit: enrolments, lessons watched, completions. A click is not a rating, and it has no negative

> **PITFALL — MISSING IS NOT ZERO, AND NOT AT RANDOM**
>
> A learner who dislikes statistics never enrols in the statistics course, so its cell is empty *because* they dislike it — a rating of a kind that is not in the data. Every model here trains and is scored on observed entries, so none of them sees this. Write the three meanings into the data description, and never let a method assign them a value.

Step 3 — Split first, and carve the validation ratings from the training half

- Hold out a share of the ratings for the test, at random, one seed, fixed before any model exists; every number in the review uses that split
- From the *training* ratings, hold out a further share as validation ratings. Every choice — rank, regularisation, shrinkage — is made there; the test ratings are read once
- The timestamps allow a split in time, which is what the shipped system faces; say which split a number came from

> **PITFALL**
>
> A random split of ratings is the standard protocol and it is optimistic: it puts a learner's later ratings in the training set and their earlier ones in the test set, which no deployed system ever gets. Acceptable for rating prediction if it is stated; not acceptable as a promise to the business about what the row will achieve.

Step 4 — Baselines: how much of a rating is the learner, and how much the course

- $\hat{r}_{ui} = \mu + b_u + b_i$: the global mean, this learner's offset, this course's offset, fitted by alternating closed forms, each regularised by the count
- The λ in the denominator is shrinkage: a learner with three ratings is pulled towards zero offset, a course rated by thousands is barely moved
- Expect the biases alone to close most of the distance from the global mean to the final model. Most of a rating is not personal

PITFALL

Skipping the baseline, and crediting the factorisation with what two arithmetic means and a shrinkage parameter bought. Or fitting it unshrunk: a new course with one five-star rating gets the largest offset in the catalogue, home screens fill with courses rated once, and the commissioning team reads it as demand. Fit the bias model first, shrunk, and report every later model as what it adds.

Step 5 — The model: factorise the observed entries, not a filled matrix

- $\hat{r}_{ui} = \mu + b_u + b_i + p_u \cdot q_i$: a few latent factors — how practical a course is, how mathematical — and a rating is how much the learner wants each times how much the course has of it
- The objective sums the squared error over the *observed* cells only, with a ridge penalty on both vectors; nothing is imputed
- Fit by ALS — a ridge regression per learner, then per course, alternated — or by SGD one rating at a time when the matrix outgrows memory

> **PITFALL**
>
> Filling the empty cells and taking the SVD, because the SVD is the best low-rank approximation. It is — of the matrix it is given, which is almost entirely the fill value. Zeros are off a scale that starts at one and score worse than predicting the global mean for everybody; the mean is less absurd and still loses to the bias model. Change the objective, not the data.

Step 6 — How many factors: a capacity parameter, chosen on the validation ratings

- Sweep the rank from one upwards on the training ratings and score each on the validation ratings carved out earlier: a U-curve, falling as the factors capture taste and rising again as they fit the noise
- Take the bottom of the U, refit at that rank on all the training ratings, and read the test ratings once
- The regularisation moves the curve: set it to zero and a learner with three ratings and a dozen factors gets a perfect fit and no information

> **PITFALL**
>
> Choosing the rank on the test ratings. The number in the review is then the minimum of several noisy estimates, biased downwards, and the gain evaporates on live learners. The rank is a capacity parameter, chosen the way every capacity parameter in this course is chosen: on data the model did not see, not on the set it is reported on.

Step 7 — Cold start: new courses every month, new subscribers every day

- A learner with no ratings has no p_u to solve for; a course with no ratings has no q_i. The factorisation returns the bias term and nothing else, and the failure is total rather than gradual
- **New subscriber:** fall back to popularity, or ask for a few ratings at sign-up. **New course:** fall back to content — topic, description, syllabus
- So the shipped system is a hybrid of the factorisation and something that works without history, and the switch between them is a rule written down

> **PITFALL**
>
> The offline RMSE never contains the case. It is computed on learners and courses that all have ratings, so a model that fails totally on every newcomer scores the same as one that does not — and newcomers are the subscribers still deciding whether to stay. Hold out some learners and courses entirely, and report the fallback on them beside the headline number.

Step 8 — In production: the release, and what a predicted rating is

- Release behind an A/B test on enrolments from the row and completions, against the popular row; refit nightly; on call, watch coverage and the newcomers on the fallback
- A predicted rating is what this learner would give *if they took the course*, learned from courses people chose: not a probability they will enrol, and not usefulness
- For subscribers who never rate, the signal is implicit, and an objective over observed entries is satisfied by predicting "yes" everywhere: rank pairs, an enrolled course above a sampled unenrolled one

PITFALL — POPULARITY, AND A SAMPLED "NO"

A row of the ten most popular courses scores respectably on every metric here and tells the learner nothing new, which renewals report months later; and a sampled negative may be the course the learner would have loved: unenrolled is not evidence of dislike. Report coverage beside the error, and read the implicit objective as a ranking, never as a prediction that anyone said no.

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

Ratings log → the three meanings of an empty cell written down → test held out, validation carved from the training half → biases with shrinkage → factorisation of the observed entries → rank at the bottom of the validation U → fallbacks for newcomers → test read once, coverage beside it → A/B test on enrolments, nightly refit.

Pitfall	Address it by
RMSE read as the quality of the row	rating prediction and the ranking kept apart
An empty cell treated as a rating	naming its three meanings; never assigning a value
Choices made on the test ratings	validation ratings carved from the training half
Unshrunk offsets; the baseline skipped	shrinkage by the count; the bias model first
The SVD of a filled matrix	an objective over the observed entries only
The rank chosen on the test	the U-curve on validation; refit, then read once
Newcomers invisible to the metric	fallbacks, measured on held-out learners and courses
A popular row: unenrolled read as “no”	coverage beside the error; pairwise ranking for implicit feedback

Summary

1. Recommendation is retrieval with the query replaced by a history, and the labels are a by-product of use rather than a measurement
2. 95.5% of the matrix is missing, and missing is not zero — filling it and taking the SVD scores 2.3274, worse than predicting one number for everybody
3. Restrict the objective to observed entries and you leave the SVD behind: no closed form, no uniqueness — and a model that works
4. ALS is a ridge regression per user; SGD is one update per rating. Same objective, different route, comparable answers
5. Most of a rating is not personal. The biases alone reach 0.9109 against the full model's 0.8602
6. Implicit feedback has no negatives: an unobserved item is not a disliked one, which is why BPR ranks pairs instead of predicting values
7. Designed an e-learning company's course recommender in nine steps — rating prediction kept apart from the row, the three meanings of an empty cell written down, validation ratings carved from the training half, biases shrunk by the count, a factorisation of the observed entries rather than a filled SVD, the rank at the bottom of the validation U, fallbacks for newcomers measured on their own, and an A/B test on enrolments with coverage beside the error — each step named for the pitfall it avoids

NOT PRESENTED — FOR AFTERWARDS

Exercises Lecture 21

five, in the style of the written paper

Exercises — Lecture 21

1 State why the truncated SVD cannot be applied directly to a ratings matrix, naming the condition of Eckart–Young that fails. [5]

2 A rank-32 SVD of the zero-filled matrix scores worse than predicting one number for everybody. Explain how the optimal approximation can lose to a constant. [5]

3 Fix Q and write the subproblem ALS solves for one user's p_u. Identify it as a familiar estimator. [5]

Solutions on Lecture 22's deck.

Exercises — Lecture 21 (continued)

4 Validation RMSE is lowest at $d = 8$ and rises again by $d = 64$. State what this says about the rank, and on which ratings it must be chosen. [4]

5 Implicit feedback has no negatives. State why an objective over observed entries alone fails, and how BPR's objective avoids the problem. [5]

Solutions on Lecture 22’s deck.

THE FIVE SET AT THE END OF LECTURE 20

Solutions Lecture 20

not part of this lecture

Solutions — Lecture 20

1. Explain why a bi-encoder can be indexed and a cross-encoder cannot, and say what the two differences have in…

✓ $E(d)$ does not depend on the query, so it can be precomputed; the cross-encoder reads the pair jointly. The same fact causes the accuracy difference.

- Whether the query is present when the document is read decides both
- Nothing can be stored for a function that does not exist until the query arrives

2. A first stage has recall@100 of 0.885. State the highest recall a perfect re-ranker over those 100 candidates…

✓ 0.885. Improving the first stage raises the ceiling; improving the second never does.

- A re-ranker reorders a fixed candidate set and cannot retrieve
- 11.5% of relevant documents are simply absent from the candidates

Solutions — Lecture 20 (2)

3. BM25 scores 0.6611 NDCG@10 on SciFact and a zero-shot bi-encoder 0.6451, yet fusing their rankings by…

✓ 68 of 339 relevant abstracts are found in the top ten by exactly one of the two: they fail on different documents. Ranks are comparable and the scores are not.

- Two systems with almost the same mean can have different error sets, and fusion is how you keep both
- BM25 returns an unbounded sum and the bi-encoder a cosine in $[-1, 1]$, so RRF needs nothing but the two orderings, with no training and no tuning

4. Hard negatives are mined by calling the unjudged documents in a first stage's top results negative. On…

✓ The 9.14 are unjudged, not judged irrelevant, and on a corpus this sparsely judged some of them are relevant. Trained on as negatives, they push the model away from correct answers.

- SciFact judges 339 of 1.55 million claim–abstract pairs, 1.13 relevant per claim
- It is the commonest reason a fine-tuned retriever underperforms the paper it copied

Solutions — Lecture 20 (3)

5. Zero shot on SciFact, a bi-encoder loses to BM25 on NDCG@10 (0.6451 against 0.6611) and wins on recall@100…

✓ NDCG@10 judges the order a user sees; recall@100 judges a first stage, by what it makes possible. As a first stage, the bi-encoder.

- A first stage is judged on what it hands the next stage, not on what the user sees
- Worse at ordering and better at finding are not a contradiction: they are answers to two different questions

LECTURE 22 · LECTURE NOTES

Recommender systems: neural, and evaluated honestly

From ratings to rankings — HR@10, and the popularity baseline every table needs
Two-tower models — Lecture 20’s bi-encoder, with users in place of queries
Then evaluation: which interaction is held out, against what it is ranked, and popularity bias

Part V · Lecture notes · examinable