Recommendation Systems II and Classification Metrics

COSC 410: Applied Machine Learning
Fall 2025
Prof. Forrest Davis

September 23, 2025

Warm-up

1. Discuss with your neighbor your homecoming highlight
2. When a new user joins a service, will user-based collaborative filtering work well? Justify your answer.

Logistics

- No Class Thursday Sep 25
- Codelet 3 on the website, due Sunday Sep 28
- Midterm Exam next Thursday Oct 2

Learning Objectives

- Give the broad categories of recommender systems
- Describe latent factor approaches to recommender systems
- Understand the role of biases and regularization in latent factor models
- Implement a content-based or collaborative based system

Summary: After recapping approaches to building recommender systems, we discuss the application of latent factor models (in particular, matrix factorization). We then turn to classification metrics.

Recap of Approaches

WE DISCUSSED THREE APPROACHES to building recommendation systems. These belong to two broad categories:[1]

[1] In fact, we can unify the approaches we have been considering under neighborhood based recommendation systems.

1. Feature-Based
   - Content-based
   - Knowledge-based
2. Model-Based
   - User-based collaborative filtering
   - Item-based collaborative filtering
   - Latent Factor (e.g., Matrix Factorization, Autoencoders)

items

users	1	2	3	4	5	6	7	8	9	10	11	12
A	1		3			5			5		4	
B			5	4			4			2	1	3
C	2	4		1	2		3		4	3	5	
D		2	4		5			4			2	
E			4	3	4	2					2	5
F	1		3		3			2			4	

Table 1: Sample data for users rating movies

Weighing Approaches

Question

1. In many settings, item-based collaborative filtering works better than user-based (Amazon proposed item-based because it performed well for them). Why might this be the case?

Ways to Evaluate a Recommender System

First, we need to lay out some basic terminology.

- **X**: A set of Users
- **S**: A set of Items
- Utility function: $\mathbf{f: X \times S \rightarrow R}$

- **R** is space of (ordered) ratings
- e.g., 1-5 stars, number [0, 1], {Good, Bad}

As with all ML problems, we need to partition our data into at least two bins: training data and testing data. There are two broad categories of machine learning evaluation:

1. **Intrinsic Evaluation**: The system is evaluated via a metric that directly relates to its training
2. **Extrinsic Evaluation**: The system is evaluated via a metric that directly relates to its use

With recommendation systems, a common approach for intrinsic evaluation is to evaluate a systems predicted ratings over test data using root-mean squared error:[2]

$$\text{R-MSE} = \sqrt{\frac{1}{K}\sum_{i=0}^{N}\sum_{x=0}^{M}(\hat{r}_i^{(x)} - r_i^{(x)})^2} \tag{1}$$

[2] Where N is the items, M is the users, $r_i^{(x)}$ is the true rating user x gave to item i, $\hat{r}_i^{(x)}$ is the predicted rating user x would give to item i, and K is the total number of user, item pairs.

Common approaches to extrinsic evaluation could try to evaluate things like the overall user engagement with the platform, the diversity of recommendations, etc.

Latent Factor Recommendation Systems

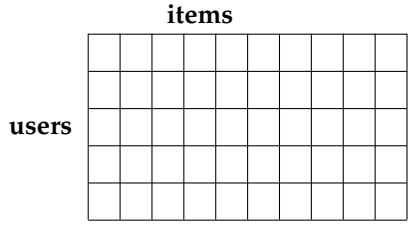
IN REAL SETTINGS, THERE ARE A NUMBER of issues that limit the applicability of the systems we have discussed so far.[3] This includes:

[3] This is the case for at least the simple forms of them we have been discussing.

1. **Sparsity**: In real applications most user-item pairs are missing[4]
2. **Scalability**: Computing the similarity between all items or users can be computationally expensive (reconsider the runtime discussion with KNN modeling)

[4] Despite my best efforts, I have not seen almost every movie or tv show on streaming platforms.

We will consider latent factor recommendation systems (and in particular, matrix factorization based techniques) as a solution to these issues.



Consider a rating matrix $\mathbf{R}^{5,10}$ as in the table above. We want to factor **R** into two matrices, **P** and **Q**. In this factorization, we want to

'compress' the data into some smaller number of features. For example, if we wanted four features, we could build a **P** and **Q**, with sample values, as below:

P

users \ factors				
	1	4	2	3
	5	6	5	2
	2	3	5	5
	1	2	6	1
	7	2	1	1

Q

factors \ items										
	1	8	2	4	2	3	2	1	1	2
	2	7	4	1	2	1	9	1	7	2
	3	5	6	4	8	4	3	5	2	6
	5	1	1	4	4	9	4	1	3	1

Question

Based on the **P** **Q** factorization above, what is the value in **U** for user 3 item 4?

To learn our utility function f (mapping from user-item pairs to ratings), we are searching for a **P** and a **Q** that can reproduce our rating matrix.

Question

Using root-mean squared error, construct an optimization problem that will find **P** and **Q** that represent a good utility function.

Tweaks to the Basic Latent Factor Model

THERE ARE A NUMBER OF TWEAKS that are commonly applied to the basic formulation of matrix factorization as a recommendation system that we gave above. Consider the following questions?

- How are missing values (i.e., items a user didn't rate) treated?
- How do we handle user, item, and population biases?
- How do we handle overfitting?

$$\underset{P,Q,b}{\operatorname{argmin}} \sqrt{\frac{1}{|K|} \sum_{(u,i)\in\mathcal{K}} ((\mathbf{p}^{(u)} \cdot \mathbf{q}_i + b_u^{(u)} + b_i^{(i)} + b) - r_i^{(u)})^2 + \lambda(\|\mathbf{P}\|^2 + \|\mathbf{Q}\|^2 + \|\mathbf{b}_u\|^2 + \|\mathbf{b}_i\|^2 + b^2)} \tag{2}$$

We can learn $\mathbf{P}, \mathbf{Q}, \mathbf{b}$ using gradient descent!

Refreshing Classification Tasks

WE ARE TRANSITIONING BACK TO MODELS FOR CLASSIFICATION. Recall there are a variety of tasks that can be framed as classification problems:

- Identify pets in images
- Determine the likelihood someone has a disease
- Predict the word you will type next
- ...

We will focus on model-based approaches to classification. In particular, those that assume a linear relationship between the classification label and the input features. We will start with **binary classification**, a model whose output space (i.e., the set of labels) is constrained to one feature which is either present or absent (or true or false). For example, determining whether an image contains a cat or not. In particular, we will focus on **logistic regression**, which is a linear model that is used in binary classification problems.

Classification Metrics

IN A TURN FROM OUR USUAL APPROACH, we will begin with the standard ways we evaluate a classification system. Then we will turn to the model formulation.

Accuracy, the number of correct predictions divided by the total number of predictions, is something we are all reasonably familiar with. It is actually a horrible metric in real world settings.

Question

What is the accuracy of **Model A** and **Model B** in Table 2?

Model A	**Model B**	True Labels
Cat	Not Cat	Cat
Cat	Not Cat	Not Cat
Not Cat	Not Cat	Not Cat
Not Cat	Not Cat	Not Cat
Not Cat	Not Cat	Not Cat
Not Cat	Not Cat	Not Cat
Not Cat	Not Cat	Not Cat
Not Cat	Not Cat	Not Cat
Not Cat	Not Cat	Not Cat
Not Cat	Not Cat	Not Cat

Table 2: Predictions over ten samples for two models (**Model A** and **Model B**) trained for binary classification compared to the true labels.

Matrices Meant To Be Confusing

WE CAN BETTER REPRESENT THE PREDICTIONS from a classification model than a simple table. One approach is called **confusion matrices**. The confusion matrix for a classifier organizing results by true label and the predicted label, breaking it down by particular labels. For example, Model B in Table 2 can be represented with the confusion matrix shown in Figure 1.

	Predicted Label	
True Label	**Cat**	**Not Cat**
Cat	0	1
Not Cat	0	9

Figure 1: Confusion matrix of **Model B** from Table 2.

Recalling Accuracy: We Can Be More Precise

Practice Problems

For the following question, consider the classifier represented by the confusion matrix in Figure 2.

1. How reliable are the model's cat predictions? How reliable are the model's dog predictions?

2. What proportion of cats was identified correctly? What proportion of dogs was identified correctly?

	Predicted Label	
True Label	**Cat**	**Not Cat**
Cat	490	10
Not Cat	48	2

Figure 2: Confusion matrix of a binary classification model.

Answering the reliability question (1 above) is called **precision**. Precision measures how accurate our model's predictions are. Answering the identification question (2 above) is called **recall**. Recall measures the proportion of true labels that were correctly identified by our model.

We can give a equation for precision and recall by using the terms in Figure 3.

$$\text{precision} = \frac{TP}{TP + FP} \qquad \text{recall} = \frac{TP}{TP + FN}$$

TP	**True Positive** (i.e., cases where you predict the class C and it is from C)
FP	**False Positive** (i.e., cases where you predict the class C and it is not from C)
TN	**True Negative** (i.e., cases where you predict not class C and it is not from C)
FN	**False Negative** (i.e., cases where you predict not class C and it is from C)

Figure 3: Key terms in evaluating predictions from a classification model.

Question

Label the below confusion matrices with the meaning of each cell (e.g., which cell represents TPs). Notice that the left confusion matrix is for Cat and the right for Not Cat.

CAT	**Predicted Label**	
True Label	**Cat**	**Not Cat**
Cat		
Not Cat		

NOT CAT	**Predicted Label**	
True Label	**Cat**	**Not Cat**
Cat		
Not Cat		

But I Want One Number

We can combine precision and recall together using an **f-score**. A common f-score is F_1 (pronounced 'f one'), where β is set to 1:

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

There are two approaches to aggregating metrics over labels:

- **Micro** Precision/Recall/F_1: Aggregate the true positives, false negatives, etc. over all classes (e.g., you combine the true positives for Cat and Not Cat to give one true positive number). Then calculate the relevant metric.
- **Macro** Precision/Recall/F_1: Calculate separate metrics for each class and then average the metrics (e.g., calculate the recall for Cat and the recall for Not Cat then take the average of the two).

Practice Problems

1. Calculate the macro F_1 for the following classifier:

	Predicted Label	
True Label	**Cat**	**Not Cat**
Cat	45	5
Not Cat	15	35

2. When do we want high precision? Give concrete scenarios

3. When do we want high recall? Give concrete scenarios

No Free Lunch

There is empirically a trade-off between precision and recall. Consider Figure 4 below.

Figure 4: Sample dataset with cats and dogs

Before Next Class

- Read and pre-class quiz
- Review and make progress on Codelet 3