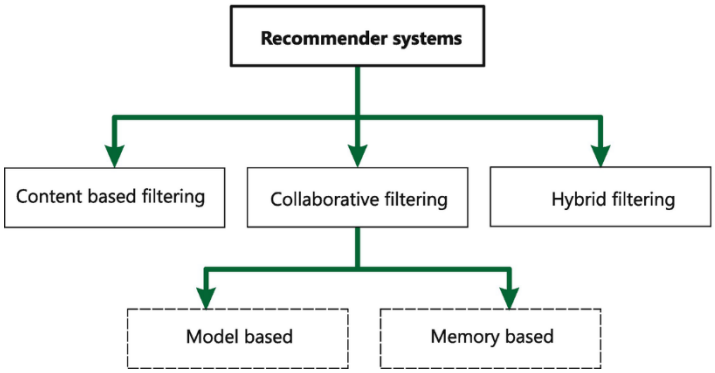
Системи засновани на моделу

Основна подела система за препоруку је приказана на слици 1. Као што се може видети, сарадничко филтрирање се може поделити у две категорије: **системи засновани на моделима и меморији**.

Приступ заснован на меморији користи податке о оценама корисника за израчунавање сличности између корисника или предмета. Типични примери овог приступа су сарадничко филтрирање засновано на суседству (*KNN*) и *top-N* препоруке засноване на производима/ корисницима (*Item-based* и *user-based collaborative filtering*).



Слика 1. Типови система за препоруку

Показало се да системи за препоруку који се базирају на сарадничком филтрирању (овом приликом се мисли на *memory-based* приступе) дају интересантне и смислене резултате, чак и када се ради о великим матрицама. Сарадничко филтрирање је доста критикован приступ да има ограничену скалабилност, пошто се матрица сличности израчунава над великим скуповима података, корисника или производа, и тако троши доста ресурса.

Показало се на примеру Амазона да коришћење сарадничког филтрирања које се заснива на производима, а не на корисницима, доста смањује комплексност читавог система и да се матрица сличности може лако израчунати на једном рачунару. Чак и да то није случај, постоје технологије као што је *Apache Spark* који нам омогућава да конструкцију матрице радимо на више сервера.

Основни проблем сарадничког филтрирања је превелика осетљивост на шум (noisy data) и ретке податке. Отуда се јавља потреба и за другим приступима. Уколико имамо велики скуп података за рад, резултати ће бити заиста добри, али ако то није случај, тада резултати могу изгледати прилично „чудно“ и тешко могу компанији да донесу додатну вредност.

Системи засновани на моделу, за разлику од сарадничког филтрирања које проналази међусобно сличне кориснике или производе, имају за циљ да различитим техникама машинског учења предвиде рејтинг који би корисник потенцијално доделио неком производу. Ови системи се обучавају на скупу података креирајући моделе, како би затим на основу њих предвидели рејтинге које би корисници доделили производима са којима до тада нису били упознати.

Неке од ових метода су јако добре у предвиђању рејтинга који би корисник дао неком производу. Да ли дају добре *top-N* препоруке је нешто што ћемо морати сами да сазнамо. Постоји широк спектар техника које спадају у категорију факторизације матрице. Ови алгоритми су нешто специфичнији у одосу на оно што смо раније видели. Ови алгоритми покушавају да на одређени начин пронађу **шире карактеристике (*features*)** корисника и предмета, као на пример да ли је реч о акционим или романтичним филмовима. Иако математика не зна како да назове ове карактеристике, оне су само описане/представљене матрицама које описују све атрибуте из оригиналних података.

Генерална идеја је да се опишу корисници и филмови као комбинација различитих количина сваке особине. На пример, можда је Боб дефинисан као да је 80% љубитељ акције и 20% љубитељ комедије. Тада бисмо знали да га упаримо са филмовима који су спој око 80% акције и 20% комедије.

	Бекство из Шошенка	Казабланка	Ратови звезда	Одбегла млада	Титаник
Бранко	4	4	?	?	?
Милош	?	?	5	?	?
Ана	?	?	?	?	5

Наш изазов је да попунимо непознате ћелије неким предвиђеним вредностима. Једна од техника која може да ради са оваквим матрицама назива се **Анализа Главних Компоненти** (Principal Component Analysis, или скраћено PCA). PCA се најчешће помиње у контексту смањења димензије проблема. Од података који описују филмове кроз многе особине (многе димензије), ми желимо да те особине претворимо у скуп мањих димензија који прецизно може да опише филм, на пример кроз жанрове филма.

БЛОГ:

- https://towardsdatascience.com/understanding-pca-fae3e243731d
- https://towardsdatascience.com/principal-component-analysis-pca-from-scratch-in-python-7f3e2a540c51 - пример са **IRIS** скупом података који се користи као state-of-the-art за приказ PCA методе.

Математичко објашњење PCA методе следи на неком од наредних термина. Не планирајте да бежите са часа. 😊

Како би то изгледало у пракси:

Нека је матрица корисности R комплетно попуњена подацима:

	Бекство из Шошенка	Казабланка	Ратови звезда	Одбегла млада	Титаник
Бранко	4	4	5	3	4
Милош	2	4	5	5	5
Ана	5	4	3	5	5

Примени се PCA метода на матрицу R за колоне. Тада добијамо матрицу U (описујемо кориснике, users). На пример:

	PCA1	PCA2
Бранко	0,75	2,25
Милош	0,45	4,55
Ана	0,23	1,77

Примени се PCA метода на **транспоновану** матрицу R. Тада се добија матрица M (описујемо филмове, movies). На пример:

	PCA1	PCA2
Бекство из Шошенка	2,75	2,25
Казабланка	1,45	4,55
Ратови звезда	3,23	1,77
Одбегла млада	0,84	0,16
Титаник	2,33	0,67

Може се показати да су ове две матрице фактори почетне матрице. Другим речима, ако имамо матрице U и M, можемо реконструисати матрицу R.

$$R = U\Sigma M^T$$

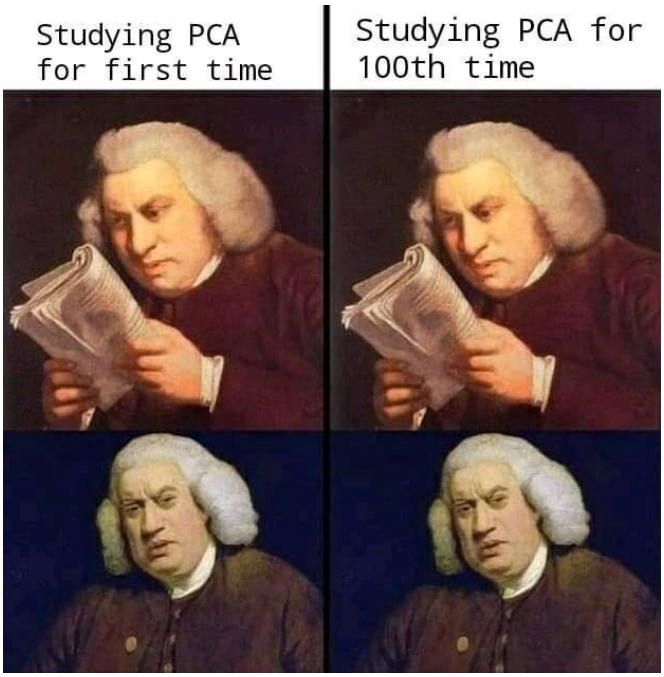
Отуда потиче назив **факторизација матрице**.

Singular Value Decomposition (SVD) је начин рачунања матрица U, Σ, и M^T, одједном и то врло ефикасно. Све што SVD ради је покретање PCA методе за кориснике и предмете одједном враћајући нам матрице које су нам потребне, а то су управо фактори матрице рејтинга R коју желимо.

Ако вас интересује математичка позадина и веза између SVD и PCA:

https://stats.stackexchange.com/questions/134282/relationship-between-svd-and-pca-how-to-use-svd-to-perform-pca

https://en.wikipedia.org/wiki/Singular_value_decomposition#Calculating_the_SVD



Слика 2. Моји студенти док спремају испит.

1.1. SVD (singular-value decomposition)

Ако говоримо SVD препорукама, предвиђање рејтинга, тј. попуњавање матрице корисности, заснива се на чињеници да се ова матрица може посматрати као производ две матрице са мањим бројем димензија. Наиме, за велики броја производа релативно мали број његових обележја (особина) утиче на одлуку корисника о томе како ће га оценити. Најпознатија техника смањења димензионалности јесте *SVD* (*singular-value decomposition*), тј. сингуларна декомпозиција вредности. Као што је раније речено, углавном релативно мали број особина неког производа утиче на начин на који људи на њега реагују. Конкретније, ако је реч о филму, корисници обраћају пажњу на жанр, омиљене глумце, режисере и продуценте, док им нпр. подаци о камерману и шефу расвете нису толико битни.

Али, када се каже да радимо SVD препоруке, то заправо није класичан SVD, јер не можемо да направимо прави SVD са подацима који недостају. Тачније, можемо ако ретку матрицу попунимо неким случајним вредностима, али се онда губи смисао система за препоруку. **Simon Funk** је развио SVD-инспирисан алгоритам који је направљен за Netflix Prize такмичење, и то је није чист SVD.

SVDFunk идеја: Претпоставимо да имамо бар неке познате оцене за било који дати ред и колону у U и M^T. Ово можемо третирати као проблем минимизације, где покушавамо да пронађемо вредности тих комплетних редова и колона које најбоље смањују грешке у познатим рејтингима у R.

1.2. SVDFunk FROM SCRATCH – верзија без регуларизације

Нека је R матрица корисности са m редова и n колона (m – број корисника, n-број производа). Циљ је пронаћи матрицу U са m редова и d колона и матрицу M са n редова и d колона, такве да су вредности у матрици $U \cdot M$ приближне постојећим вредностима у матрици R. Уколико је описана апроксимација задовољавајућа, тада се може тврдити да су корисници и производи успешно описани помоћу d димензија. Приликом овог поступка није могуће именовати особине (*латентне факторе*) којима су описани корисник или производ. Корисник је представљен вектором дужине d, где свака координата представља „ниво важности“ одговарајуће особине за тог корисника. Координате векторске репрезентације производа се могу посматрати као степен „присутности“ особине у тој јединици производа. На пример, за неки филм се условно може рећи да је 0.8 авантура, 0.1 комедија, 0.5 драма итд. Слична аналогија важи и за корисника.

На почетку поступка је потребно обавити стандардне процедуре у машинском учењу (*preprocessing*). Од сваког рејтинга у матрици корисности потребно је одузети просечну вредност оцена које тај корисник оставља. Овај корак се користи због различитог начина оцењивања корисника (због културолошких разлика, различитих типова личности итд). Овај корак може и да се прескочи, јер је показано да се не добијају бољи резултати када је број оцена по кориснику велики (сетимо се приче о ретким матрицама и просека оцена ако је корисник оценио нпр. један или два производа).

Након тога, вредности у тако добијеној матрици, треба умањити за вредност просечне оцене одговарајућег производа. Овај корак се односи на колоне и познат је под називом центрирање колона (column-centered). Наравно, постоје и други начини нормализације матрице корисности, али њих нећемо наводити, јер се поступци не разликују превише. ОВАЈ КОРАК ЈЕ БИТАН И НЕ МОЖЕ СЕ ПРЕСКОЧИТИ! – Ово је већ имплементирано у библиотеци.

Нека је матрица корисности R, без губитка општости, представљена на следећи начин:

$$\begin{bmatrix} r_{11} & & & r_{14} & r_{15} & & \\ r_{21} & r_{22} & r_{23} & & & & \\ & & & r_{34} & r_{35} & r_{36} & \\ & r_{42} & & & & & r_{47} \end{bmatrix}$$

Наведена матрица приказује интеракцију између 4 корисника и 7 производа. Ознака r_{ij} представља оцену коју је корисник i доделио производу j $(i = \overline{1 \dots 4},\ j = \overline{1 \dots 7})$. Подразумева се да су у празна поља матрице уписане нуле. Нека $\mu_i^{(u)}$ означава просечну вредност рејтинга корисника i, а $\mu_j^{(p)}$ просек оцена производа j $(i = \overline{1 \dots 4},\ j = \overline{1 \dots 7})$.

Након првог корака нормализације (није неопходан), матрица корисности се може представити на следећи начин:

$$\begin{bmatrix} r_{11} - \mu_1^{(u)} & & & r_{14} - \mu_1^{(u)} & r_{15} - \mu_1^{(u)} & & \\ r_{21} - \mu_2^{(u)} & r_{22} - \mu_2^{(u)} & r_{23} - \mu_2^{(u)} & & & & \\ & & & r_{34} - \mu_3^{(u)} & r_{35} - \mu_3^{(u)} & r_{36} - \mu_3^{(u)} & \\ & r_{42} - \mu_4^{(u)} & & & & & r_{47} - \mu_4^{(u)} \end{bmatrix}$$

Ради једноставнијег записа нека x_{ij} означава разлику $r_{ij} - \mu_i^{(u)}$ где $i = \overline{1 \dots 4}$, $j = \overline{1 \dots 7}$. Спроводећи даље поступак нормализације долази се до финалног облика матрице R:

$$\begin{bmatrix} x_{11} - \mu_1^{(p)} & & & x_{14} - \mu_4^{(p)} & x_{15} - \mu_5^{(p)} & & \\ x_{21} - \mu_1^{(p)} & x_{22} - \mu_2^{(p)} & x_{23} - \mu_3^{(p)} & & & & \\ & & & x_{34} - \mu_4^{(p)} & x_{35} - \mu_5^{(p)} & x_{36} - \mu_6^{(p)} & \\ & x_{42} - \mu_2^{(p)} & & & & & x_{47} - \mu_7^{(p)} \end{bmatrix}$$

У следећем кораку се обавља иницијализација матрица U и M. Матрице се попуњавају насумично одабраним вредностима[1].

Корен просека квадрата грешака (*RMSE*) је једна од метрика којом се прати остварени ниво апроксимације матрице $U \cdot M$ у односу на матрицу R.

Затим се спроводи минимизација по *RMSE*. Као и у осталим областима машинског учења, могу се користити различити алгоритми. Најчешће се употребљава градијентно спуштање (**gradient descent**). На више начина је могуће одабрати елементе матрица U и M у односу на које се обавља минимизација. Најједноставније решење је спроводити оптимизацију ред по ред. Услов за прекид поступка може бити да *RMSE* постане нула, што и није толико честа ситуација. Далеко реалнији услов јесте критеријум прекида у коме се *RMSE* не смањује значајно[2] у односу на остварену вредност у претходним итерацијама.

Након обучавања модела предикција оцене, коју би корисник i доделио производу j, у ознаци $\hat{r}_{ij}$, се рачуна помоћу следеће формуле:

$$\hat{r}_{ij} = \mu_i^{(u)} + \mu_j^{(p)} + u_i \cdot {v_j}^T \tag{9}$$

где је u_i i-ти ред у матрици U (вектор који представља корисника i), а v_j j-ти ред матрице M (вектор којим је представљен производ j). Због свега наведеног *SVD* спада у алгоритме засноване на факторизацији матрице.

[1] Најчешће се користи нормална расподела. Углавном су то вредности између 0 и 1.

[2] Износ умањења RMSE је мањи од унапред одређене вредности

Додатни ресурси:

Surprise:
https://surprise.readthedocs.io/en/stable/matrix_factorization.html#surprise.prediction_algorithms.matrix_factorization.SVD

Blog: https://engineering.fb.com/2015/06/02/core-data/recommending-items-to-more-than-a-billion-people/

ДОМАЋИ: Избор хиперпараметара (hyperparameters tuning).