Personalized Machine Learning

Matrix Factorization

Rodrigo Alves

October 1, 2024

Matrix Factorization

- Matrix factorization is one of the main PML algorithms.
- The targets can be stored in a **target matrix** R of dimensions $m \times n$ with elements from $\mathbb{R}$.
- For recommender systems, an example of a target matrix (rating matrix) with $m = 4$ users and $n = 6$ items is given by:

$$R = \begin{pmatrix} 1 & 4 & 3 & 2 & 2 & 1 \\ 3 & 2 & 3 & 4 & 2 & 1 \\ 1 & 5 & 5 & 5 & 3 & 5 \\ 2 & 1 & 2 & 3 & 3 & 3 \end{pmatrix}.$$

  This means, for example, that user u_3 rated item i_1 with 1 star, item i_2, i_3, i_4, and i_6 with 5 stars, and item i_5 with 3 stars.
- Note that in real-life applications, we never fully observe all the entries.

Idea of Matrix Factorization

- By **matrix factorization**, we usually mean expressing a given matrix R as a product of matrices. For example:

$$R = UV^\top$$

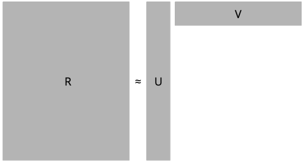
- Matrix factorization methods are a cornerstone of many algorithms and are used to achieve more numerically stable computations.

Intuition of Matrix Factorization

- The very basic idea of the lower dimensional approximation of an input matrix R of dimension $m \times n$ is based on this first-linear-algebra-lesson fact:

Multiplying matrices $U \in \mathbb{R}^{m \times d}$ and $V \in \mathbb{R}^{d \times n}$ will result a **low-rank** matrix of dimension $m \times n$. *The multiplication is true for any positive integer* d *and* R *will have low-rank for any* $d <$ *min*(m, n).

- Optimisation: **Given a rating matrix R, find lower dimensional matrices U and V so that the known elements of R are well approximated by the matrix $\mathbf{UV}^\top$.**

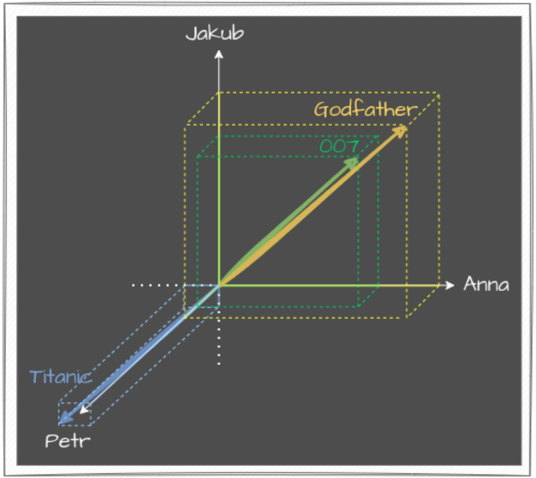


Rank of a Matrix

- The **rank** of a matrix is the number of linearly independent columns it has.
- Alternatively, we can define the **rank** as the number of non-zero singular values of a matrix.
- Let rank(A) denote the rank of a matrix $A \in \mathbb{R}^{m \times n}$. Properties and definitions:
  - rank$(A) =$ rank$(A^\top)$.
  - WLOG, if $m \geq n$, matrix A is considered full rank when rank$(A) = n$. In this case, n is also the maximum possible rank.
  - For matrices where $m = n$, an inverse A^{-1} exists only if A is full rank.
  - A matrix is said to be of low rank (or rank deficient) if it does not have full rank.

Rank of a Matrix

	007	Godfather	Titanic
Anna	3.75	5.00	-0.50
Jakub	3.50	4.50	-0.75
Petr	0.50	1.00	4.00



SVD

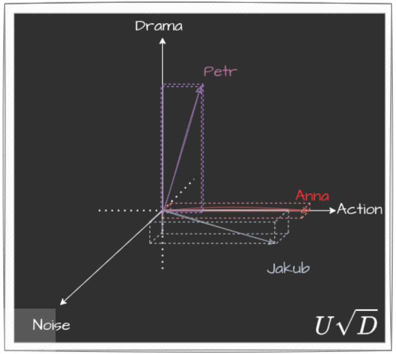
$$
\underbrace{\begin{array}{c|ccc} & \text{007} & \text{Godfather} & \text{Titanic} \\ \hline \text{Anna} & 3.75 & 5.00 & 0.50 \\ \text{Jakub} & 3.50 & 4.50 & 0.75 \\ \text{Petr} & 0.50 & 1.00 & 4.00 \end{array}}_{R}
=
\underbrace{\begin{array}{c|ccc} \text{Anna} & 0.73 & -0.03 & -0.67 \\ \text{Jakub} & 0.67 & -0.10 & 0.73 \\ \text{Petr} & 0.10 & 0.99 & 0.05 \end{array}}_{U}
\times
\underbrace{\begin{array}{ccc} 8.53 & 0.00 & 0.00 \\ 0.00 & 4.08 & 0.00 \\ 0.00 & 0.00 & 0.06 \end{array}}_{D}
\times
\underbrace{\begin{array}{ccc} \text{007} & \text{Godfather} & \text{Titanic} \\ \hline 0.60 & 0.79 & -0.05 \\ -0.01 & 0.07 & 0.99 \\ 0.79 & -0.60 & 0.05 \end{array}}_{V^\top}
$$

$$
\underbrace{\begin{array}{c|ccc} & \text{007} & \text{Godfather} & \text{Titanic} \\ \hline \text{Anna} & 3.75 & 5.00 & 0.50 \\ \text{Jakub} & 3.50 & 4.50 & 0.75 \\ \text{Petr} & 0.50 & 1.00 & 4.00 \end{array}}_{R}
=
\underbrace{\begin{array}{c|ccc} & \text{Action} & \text{Drama} & \text{Noise} \\ \hline \text{Anna} & 2.14 & -0.08 & -0.16 \\ \text{Jakub} & 1.96 & -0.22 & 0.18 \\ \text{Petr} & 0.30 & 2.00 & 0.01 \end{array}}_{U\sqrt{D}}
\times
\underbrace{\begin{array}{c|ccc} & \text{007} & \text{Godfather} & \text{Titanic} \\ \hline \text{Action} & 1.76 & 2.32 & -0.15 \\ \text{Drama} & -0.01 & 0.14 & 2.01 \\ \text{Noise} & 0.19 & -0.15 & 0.01 \end{array}}_{\sqrt{D}V^\top}
$$

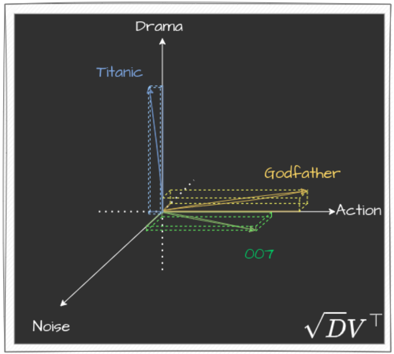
SVD

$$
\underset{R}{\begin{array}{c|ccc} & \text{007} & \text{Godfather} & \text{Titanic} \\ \hline \text{Anna} & 3.75 & 5.00 & -0.50 \\ \text{Jakub} & 3.50 & 4.50 & -0.75 \\ \text{Petr} & 0.50 & 1.00 & 4.00 \end{array}}
=
\underset{U\sqrt{D}}{\begin{array}{c|ccc} & \text{Action} & \text{Drama} & \text{Noise} \\ \hline \text{Anna} & 2.14 & -0.08 & -0.16 \\ \text{Jakub} & 1.96 & -0.22 & 0.18 \\ \text{Petr} & 0.30 & 2.00 & 0.01 \end{array}}
\times
\underset{\sqrt{D}V^\top}{\begin{array}{c|ccc} & \text{007} & \text{Godfather} & \text{Titanic} \\ \hline \text{Action} & 1.76 & 2.32 & -0.15 \\ \text{Drama} & -0.01 & 0.14 & 2.01 \\ \text{Noise} & 0.19 & -0.15 & 0.01 \end{array}}
$$

SVD

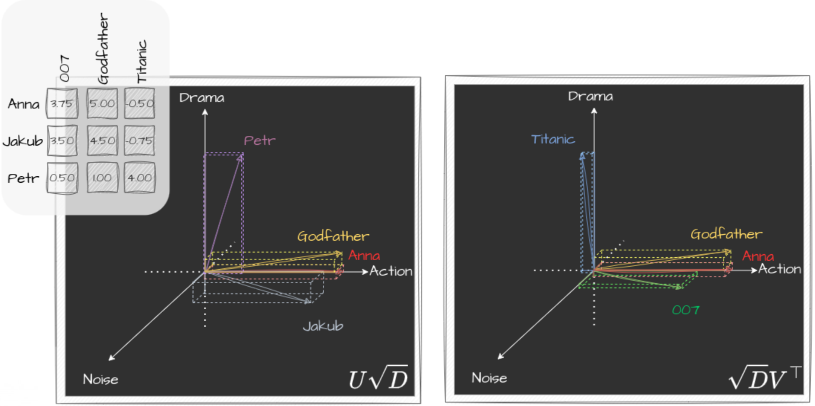


Drama
Petr
Anna
Action
Jakub
Noise
$U\sqrt{D}$



Drama
Titanic
Godfather
Action
007
Noise
$\sqrt{D}V^{\top}$

Prediction



$$\langle a\ ,\ b\rangle = ||a||\,||b||\cos(\theta)$$

Rank Approximation

Facts we have so far in our example:

- Our rating matrix R has full rank.

Does there exist a rank-2 matrix that can approximate R well?

- Note that the genres "Action" and "Drama" explain the phenomenon better than "Noise"!

Eckart–Young–Mirsky Theorem

Let $A \in \mathbb{R}^{m \times n}$ be a matrix with rank r, and let k be a positive integer such that $1 \leq k \leq r$. The best rank-k approximation to A in terms of the Frobenius norm is given by the Singular Value Decomposition (SVD):

$$A_k = \sum_{i=1}^{k} d_i u_i v_i^\top$$

where $d_1 \geq d_2 \geq \ldots \geq d_k > 0$ are the singular values of A, and u_i and v_i are the corresponding left and right singular vectors.

SVD Approximation </>

	Action	Drama	Noise
Anna	2.14	-0.08	-0.16
Jakub	1.96	-0.22	0.18
Petr	0.30	2.00	0.01

$(U\sqrt{D})[:, 1:2]$

$\times$

	007	Godfather	Titanic
Action	1.76	2.32	-0.15
Drama	-0.01	0.14	2.01
Noise	0.19	-0.15	0.01

$(\sqrt{D}V^\top)[1:2, :]$

$=$

	007	Godfather	Titanic
Anna	3.78	4.97	-0.49
Jakub	3.46	4.52	-0.75
Petr	0.49	1.00	3.99

$\hat{R}$

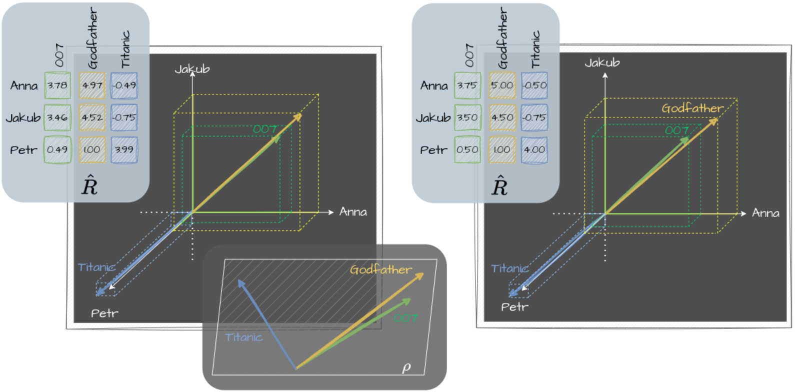
$\approx$

	007	Godfather	Titanic
Anna	3.75	5.00	-0.50
Jakub	3.50	4.50	-0.75
Petr	0.50	1.00	4.00

R

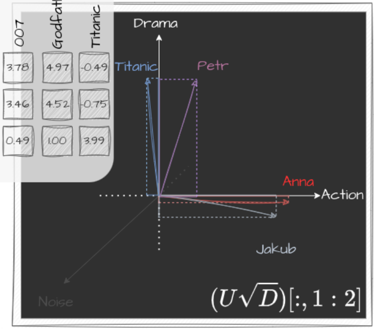
$$(U\sqrt{D})[:, 1:2] \times (\sqrt{D}V^\top)[1:2, :] = \hat{R} \approx R$$

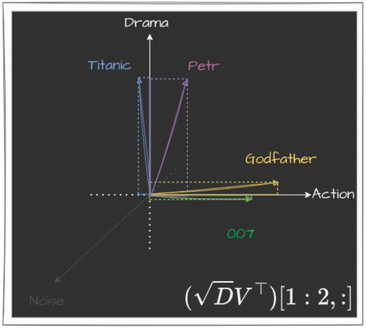
SVD Approximation



SVD Approximation

	007	Godfather	Titanic
Anna	3.78	4.97	-0.49
Jakub	3.46	4.52	-0.75
Petr	0.49	1.00	3.99





$$\langle a\ ,\ b\rangle = ||a||\,||b||\cos(\theta)$$

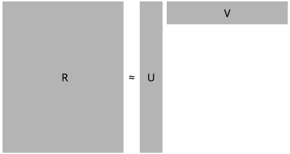
Recommender as a Matrix

- So far, we have examined a fully-known problem that doesn't apply to recommenders.
- For Recommender Systems (RSs), ratings can be stored in a **rating matrix** R of dimension $m \times n$ with elements from $\mathbb{R} \cup \{?\}$.
- An example of a rating matrix for $m = 4$ users and $n = 6$ items could look like this:

$$R = \begin{pmatrix} 1 & ? & ? & 2 & ? & 1 \\ ? & 2 & 3 & ? & 2 & 1 \\ 1 & 5 & 5 & ? & ? & 5 \\ ? & ? & 2 & ? & ? & 3 \end{pmatrix}.$$

  This means, for example, that user u_1 rated items i_1 and i_6 with 1 star, item i_4 with 2 stars, and had no interactions with items i_2, i_3, and i_5.
- **Our goal is to predict the unknown ratings** $r_{u,i} = ?$ using the knowledge of the known ratings $r_{u,i} \neq ?$.

Matrix Factorization for Recommenders



- Let us denote:
  - The i-th row of U as u_i; the number of rows of U equals the number of users $|\mathcal{U}|$.
  - The j-th column of V as v_j; the number of columns of V equals the number of items $|\mathcal{I}|$.
  - Ω the subset of $\mathcal{U} \times \mathcal{I}$ of user-item pairs (i,j) such that $r_{i,j}$ is known, i.e., $r_{i,j} \neq ?$.
- The approximation of $r_{i,j}$ is given by the number $u_i^T v_j$, i.e., by the dot product of the two d-dimensional vectors.
- d is the upper bound to the rank matrix.

Do we need to know all the entries of a matrix R to factorize it, for example $R = UV^\top$?

Optmization Problem

- The **error of approximation** is usually measured by the squared residual:

$$(r_{i,j} - u_i^T v_j)^2.$$

- Hence, the matrices U and V are obtained by solving the optimization task:

$$\text{argmin}_{\mathbf{U},\mathbf{V}} \sum_{(i,j)\in\Omega} (r_{i,j} - u_i^T v_j)^2 + \lambda(\sum_x ||u_x||^2 + \sum_y ||v_y||^2).$$

Sparsity and Prediction

- The matrices U and V are optimized only by considerung the known entries of R that are usually only a minority of entries.
- E.g. in the Netflix prize in 2006 there were $n = 17K$ movies and $m = 500K$ users, meaning that the matrix R had $8500M$ entries. But only $100M$ was given by Netflix!
- Still, the result of the matrix multiplication $UV^\top$ is a matrix having the same dimensions as R **with all entries known**!
- The unknown rating $r_{i,j} =?$ is estimated as $\hat{r}_{i,j} = u_i^T v_j$..

Example

- Consider our toy example matrix from above:

$$R = \begin{pmatrix} 1 & ? & ? & 2 & ? & 1 \\ ? & 2 & 3 & ? & 2 & 1 \\ 1 & 5 & 5 & ? & ? & 5 \\ ? & ? & 2 & ? & ? & 3 \end{pmatrix}.$$

- Assume that we chose the hyperparameter $d = 2$, i.e., we look for approximation matrices U and V with dimensions 4×2 and 2×6, respectively.
- Let us pretend that the matrices resulting from the optimization are

$$U = \begin{pmatrix} 0.3 & 0.7 \\ 0.3 & 0.5 \\ 0.2 & 0.4 \\ 0.2 & 0.1 \end{pmatrix} \quad \text{and} \quad V^\top = \begin{pmatrix} 1 & 10 & 11 & 10 & 4 & 20 \\ 1 & -1 & -2 & -1 & 1 & -4 \end{pmatrix}.$$

Example

- The resulting approximation is

$$\mathbf{UV}^\top = \begin{pmatrix} 0.3 & 0.7 \\ 0.3 & 0.5 \\ 0.2 & 0.4 \\ 0.2 & 0.1 \end{pmatrix} \begin{pmatrix} 1 & 10 & 11 & 10 & 4 & 20 \\ 1 & -1 & -2 & -1 & 1 & -4 \end{pmatrix} =$$

$$= \begin{pmatrix} 1 & \color{red}{2.3} & \color{red}{1.9} & 2.3 & \color{red}{1.9} & 3.2 \\ \color{red}{0.8} & 2.5 & 2.3 & \color{red}{2.5} & 1.7 & 4 \\ 0.6 & 1.6 & 1.4 & \color{red}{1.6} & \color{red}{1.2} & 2.4 \\ \color{red}{0.3} & \color{red}{1.9} & 2 & \color{red}{1.9} & \color{red}{0.9} & 3.6 \end{pmatrix},$$

where the red numbers are the desired predictions!

- E.g. the 3rd user predicted rating of the 4th item is $\hat{r}_{3,4} = 1.6$.

Supervised Learning Task

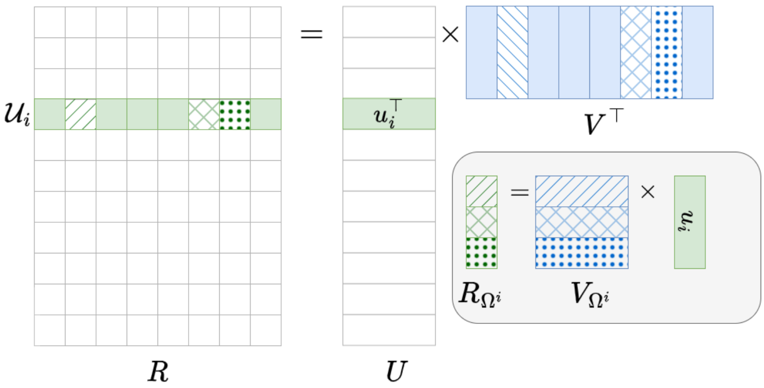
- The learning parameters: $U \in \mathbb{R}^{m \times d}$ and $V \in \mathbb{R}^{n \times d}$
- The hyperparameters:
  - the regularization constant $\lambda > 0$,
  - the matrix dimension d, which is a positive integer (significantly smaller than $\min\{m, n\}$).
- These hyperparameters can be tuned in the usual way via crossvalidation
- Therefore we would like to learn U and V, given d and λ by

$$\operatorname{argmin}_{\mathbf{U},\mathbf{V}} \sum_{(i,j)\in\Omega} (r_{i,j} - u_i^T v_j)^2 + \lambda(\sum_x ||u_x||^2 + \sum_y ||v_y||^2).$$

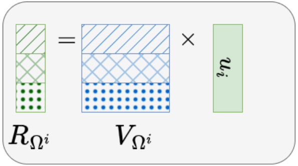
Alternating Least Squares (ALS)

- The idea of ALS is to fix alternately the matrix U and V. The non-fixed matrix is then considered learning variable and a subject to minimization.
- With one of the matrices fixed, the optimization problem becomes convex and very similar to the linear regression problem.
- Let's try ti understand how the mechanism works

Alternating Least Squares (ALS)



Alternating Least Squares (ALS)



- Then we have the following optimization problem

$$\min_{u_i} ||R_{\Omega^i} - V_{\Omega^i} u_i||^2 + \lambda ||u_i||^2$$

- Convex problem with closed-form

$$\hat{u}_i = (V_{\Omega^i}^\top V_{\Omega^i} + \lambda I)^{-1} V_{\Omega^i}^\top R_{\Omega^i}$$

Alternating least squares (ALS)

Randomly initialize U and V

- **WHILE** does not converge
  - $\forall i \in \mathcal{U}$, $\min_{u_i} ||R_{\Omega^i} - V_{\Omega^i} u_i||^2 + \lambda ||u_i||^2$
  - $\forall j \in \mathcal{I}$, $\min_{v_j} ||R_{\Omega^j} - U_{\Omega^j} v_j||^2 + \lambda ||v_j||^2$

MF for Implicit Feedback

- In real-world applications, we often observe more implicit feedback than explicit feedback.
- In fact, explicit feedback is sometimes considered implicit.
- Suppose user i watched 35% of movie A and 85% of movie B.

Does this mean that the user likes A more than B? If so, does it mean that the user likes A more than twice as much as B?

- The method we learned above is more appropriate for explicit feedback. **Why?**

Modelling Implicit Feedback

- Let's understand a more appropriate method
- Assume the binary interaction matrix P:

$$P = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

- That is, if user-i interact with item-j, than $P_{ij} = 1$, otherwise $P_{ij} = 0$.
- Now let C be a matrix of confidence regarding the interaction:

$$C = \begin{pmatrix} 0.85 & 0 & 0 & 0.34 & 0 & 0.98 \\ 0 & 0.37 & 0.10 & 0 & 0.63 & 0.01 \\ 0.45 & 0.42 & 0.43 & 0 & 0 & 0.23 \\ 0 & 0 & 0.26 & 0 & 0 & 0.88 \end{pmatrix}.$$

Collaborative Filtering for Implicit Feedback

- Then we propose the following optimisation problem:

$$\text{min}_{U,V} \sum_{i,j} C_{ij}(P_{ij} - u_i^\top v_j)^2 + \lambda ||u_i||^2 + \lambda ||v_j||^2$$

- Two main differences from previous MF method:
  - We need to account for the varying confidence levels
  - Optimization should account for all possible j, j pairs, rather than only those corresponding to observed data.
- We can use gradient descent to solve it.
- And ALS? By fixing V, can we find u_i?

Closed form

- Assume V being fix and let's find u_i.
- Then we need to minimize the following loss

$$\mathcal{L}_i = \text{min}_{u_i} \sum_j C_{ij}(P_{ij} - u_i^\top v_j)^2 + \lambda ||u_i||^2$$

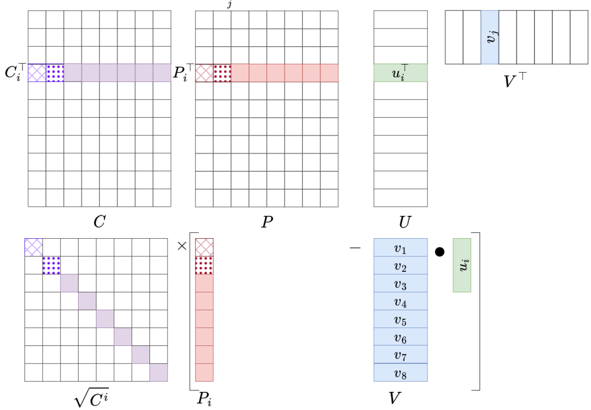
That is the same of:

$$\mathcal{L}_i = \text{min}_{u_i} \sum_j (\sqrt{C_{ij}}(P_{ij} - u_i^\top v_j))^2 + \lambda ||u_i||^2$$

Exercise: Find the closed form.

Alternating Least Squares (ALS)

$$\mathcal{L}_i = \min_{u_i} \sum_j (\sqrt{C_{ij}}(P_{ij} - u_i^\top v_j))^2 + \lambda ||u_i||^2$$



Closed form

- Therefore is the same of solving:

$$\mathcal{L}_i = ||\sqrt{C^i}P_i - \sqrt{C^i}Vu_i||^2 + \lambda + ||u_i||^2$$

- Taking the derivative

$$\nabla u_i = -2(\sqrt{C^i}V)^\top(\sqrt{C^i}P_i - \sqrt{C^i}Vu_i) + 2\lambda u_i$$

- Remind if D is diagonal $D = \sqrt{D} \times \sqrt{D}$ is trivial and $D = D^\top$
- Therefore, with just some algebraic derivations

$$u_i = (V^\top C^i V + \lambda I)^{-1} V^\top C^i P_i$$

Obrigado :) - Faculty of Information Technology