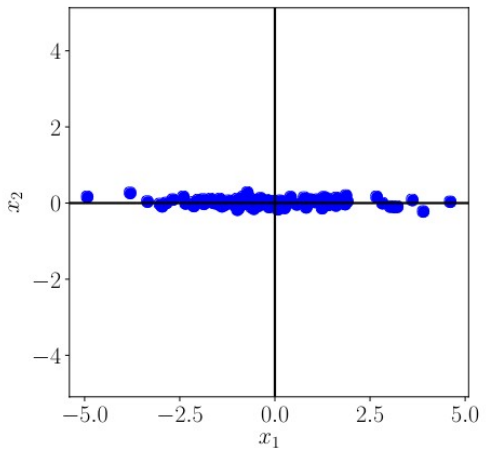
Dimensionality Reduction with Principal Component Analysis:

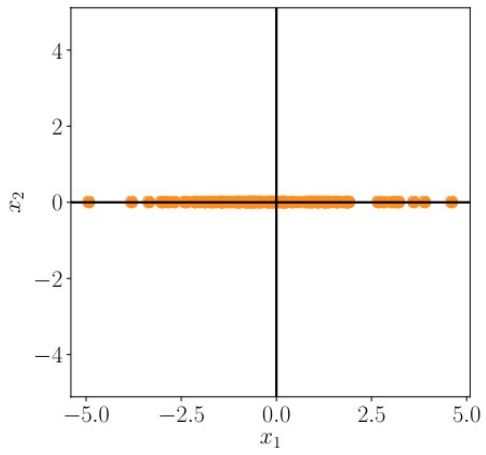
Working directly with high-dimensional data, such as images, comes with some difficulties: It is hard to analyze, interpretation is difficult, visualization is nearly impossible, and (from a practical point of view) storage of the data vectors can be expensive. However, high-dimensional data often has properties that we can exploit. For example, high-dimensional data is often overcomplete, i.e., many dimensions are redundant and can be explained by a combination of other dimensions. Furthermore, dimensions in high-dimensional data are often correlated so that the data possesses an intrinsic lower-dimensional structure. Dimensionality reduction exploits structure and correlation and allows us to work with a more compact representation of the data, ideally without losing information. We can think of dimensionality reduction as a compression technique, similar to jpeg or mp3, which are compression algorithms for images and music.

Principal component analysis (PCA), an algorithm for linear dimensionality reduction. PCA, proposed by Pearson (1901) and Hotelling (1933), has been around for more than 100 years and is still one of the most commonly used techniques for data compression and data visualization. It is also used for the identification of simple patterns, latent factors, and structures of high-dimensional data. In the signal processing community, PCA is also known as the Karhunen-Lo`eve transform.

Dimensionality reduction generally exploits a property of high-dimen-sional data (e.g., images) that it often lies on a low-dimensional subspace. Figure 10.1 gives an illustrative example in two dimensions. Although the data in Figure 1(a) does not quite lie on a line, the data does not vary much in the x2-direction, so that we can express it as if it were on a line – with nearly no loss; see Figure 1(b). To describe the data in Figure 1(b), only the x1-coordinate is required, and the data lies in a one-dimensional subspace of R2.



(a) Dataset with x_1 and x_2 coordinates.



(b) Compressed dataset where only the x_1 coordinate is relevant.

Figure 1
Illustration: dimensionality reduction. (a) The original dataset does not vary much along the x2 direction. (b) The data from (a) can be represented using the x1-coordinate alone with nearly no loss.

We will derive PCA as an algorithm that directly minimizes the average reconstruction error. This perspective allows us to interpret PCA as implementing an optimal linear auto-encoder. We will look at the difference vectors between the original data x_n and their reconstruction $\tilde{x}_n$ and minimize this distance so that x_n and $\tilde{x}_n$ are as close as possible. Figure 1 illustrates this setting.

Assume an (ordered) orthonormal basis $B = (b_1, \ldots, b_D)$ of R^D , we know that for a basis $(b_1, \ldots, b_D)$ of R^D any $x \in R^D$ can be written as a linear combination of the basis vectors of R^D , i.e.,

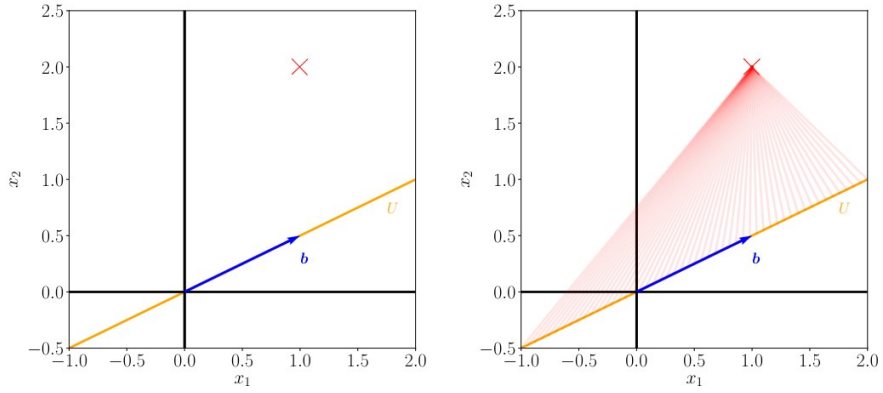
$$\boldsymbol{x} = \sum_{d=1}^{D} \zeta_d \boldsymbol{b}_d = \sum_{m=1}^{M} \zeta_m \boldsymbol{b}_m + \sum_{j=M+1}^{D} \zeta_j \boldsymbol{b}_j \tag{10.26}$$

for suitable coordinates $\zeta_d \in R$. We are interested in finding vectors $\tilde{x} \in R^D$, which live in lower dimensional subspace $U \subseteq R^D$, $\dim(U) = M$, so that

$$\tilde{\boldsymbol{x}} = \sum_{m=1}^{M} z_m \boldsymbol{b}_m \in U \subseteq \mathbb{R}^D \tag{10.27}$$

is as similar to x as possible. Note that at this point we need to assume that the coordinates z_m of $\tilde{x}$ and ζ_m of x are not identical.

we use exactly this kind of representation of $\tilde{x}$ to find optimal coordinates z and basis vectors b1, . . . , bM such that $\tilde{x}$ is as similar to the original data point x as possible, i.e., we aim to minimize the (Euclidean) distance $\|x - \tilde{x}\|$. Figure 2 illustrates this setting.



(a) Setting.

(b) Differences $\boldsymbol{x} - \tilde{\boldsymbol{x}}_i$ for 50 different $\tilde{\boldsymbol{x}}_i$ are shown by the red lines.

Figure 2 Simplified projection setting. (a) A vector $x \in R^2$ (red cross) shall be projected onto a one-dimensional subspace $U \subseteq R^2$ spanned by b. (b) shows the difference vectors between x and some candidates $\tilde{x}$.

Without loss of generality, we assume that the dataset $X = \{x_1, \ldots, x_N\}$, $x_n \in R^D$, is centered at 0, i.e., $E[X] = 0$. We are interested in finding the best linear projection of X onto a lowerdimensional subspace U of R^D with $\dim(U) = M$ and orthonormal basis vectors $b_1, \ldots, b_M$. We will call this subspace U the principal subspace. The projections of the data points are denoted by

$$\tilde{\boldsymbol{x}}_n := \sum_{m=1}^{M} z_{mn}\boldsymbol{b}_m = \boldsymbol{B}\boldsymbol{z}_n \in \mathbb{R}^D\,, \tag{10.28}$$

where $z_n := [z_{1n}, \ldots, z_{Mn}]^\top \in \mathbb{R}^M$ is the coordinate vector of $\tilde{x}_n$ with respect to the basis $(b_1, \ldots, b_M)$. More specifically, we are interested in having the $\tilde{x}_n$ as similar to x_n as possible.

The similarity measure we use in the following is the squared distance (Euclidean norm) $\|x - \tilde{x}\|^2$ between x and $\tilde{x}$. We therefore define our objective as minimizing the average squared Euclidean distance (reconstruction error)

$$J_M := \frac{1}{N}\sum_{n=1}^{N} \|\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n\|^2\,, \tag{10.29}$$

where we make it explicit that the dimension of the subspace onto which we project the data is M . In order to find this optimal linear projection, we need to find the orthonormal basis of the principal subspace and the coordinates $z_n \in \mathbb{R}^M$ of the projections with respect to this basis. To find the coordinates zn and the ONB of the principal subspace, we follow a two-step approach. First, we optimize the coordinates zn for a given ONB (b1, . . . , bM); second, we find the optimal ONB.

Finding Optimal Coordinates:

Consider Figure 2(b), where the principal subspace is spanned by a single vector b. Geometrically speaking, finding the optimal coordinates z corresponds to finding the representation of the linear projection $\tilde{x}$ with respect to b that minimizes the distance between $\tilde{x} - x$. From Figure 2(b), it is clear that this will be the orthogonal projection, and in the following we will show exactly this.

We assume an ONB (b1, . . . , bM) of $U \subseteq RD$. To find the optimal coordinates zm with respect to this basis, we require the partial derivatives

$$\frac{\partial J_M}{\partial z_{in}} = \frac{\partial J_M}{\partial \tilde{\boldsymbol{x}}_n}\frac{\partial \tilde{\boldsymbol{x}}_n}{\partial z_{in}}\,, \tag{10.30a}$$

$$\frac{\partial J_M}{\partial \tilde{\boldsymbol{x}}_n} = -\frac{2}{N}(\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n)^\top \in \mathbb{R}^{1\times D}\,, \tag{10.30b}$$

$$\frac{\partial \tilde{\boldsymbol{x}}_n}{\partial z_{in}} \overset{(10.28)}{=} \frac{\partial}{\partial z_{in}}\left(\sum_{m=1}^{M} z_{mn}\boldsymbol{b}_m\right) = \boldsymbol{b}_i \tag{10.30c}$$

for $i = 1, \ldots, M$, such that we obtain

$$\frac{\partial J_M}{\partial z_{in}} \overset{(10.30b)\,(10.30c)}{=} -\frac{2}{N}(\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n)^\top \boldsymbol{b}_i \overset{(10.28)}{=} -\frac{2}{N}\left(\boldsymbol{x}_n - \sum_{m=1}^{M} z_{mn}\boldsymbol{b}_m\right)^\top \boldsymbol{b}_i \tag{10.31a}$$

$$\overset{\text{ONB}}{=} -\frac{2}{N}(\boldsymbol{x}_n^\top \boldsymbol{b}_i - z_{in}\boldsymbol{b}_i^\top \boldsymbol{b}_i) = -\frac{2}{N}(\boldsymbol{x}_n^\top \boldsymbol{b}_i - z_{in})\,. \tag{10.31b}$$

since $b^T_i.b_i = 1$. Setting this partial derivative to 0 yields immediately the optimal coordinates

$$z_{in} = \boldsymbol{x}_n^\top \boldsymbol{b}_i = \boldsymbol{b}_i^\top \boldsymbol{x}_n \tag{10.32}$$

for i = 1, . . . , M and n = 1, . . . , N . This means that the optimal coordinates z_{in} of the projection $\tilde{x}_n$ are the coordinates of the orthogonal projection (see Section 3.8) of the original data point x_n onto the onedimensional subspace that is spanned by b_i. Consequently:

- The optimal linear projection $\tilde{x}_n$ of x_n is an orthogonal projection.
- The coordinates of $\tilde{x}_n$ with respect to the basis $(b_1, . . . , b_M)$ are the coordinates of the orthogonal projection of x_n onto the principal subspace.
- An orthogonal projection is the best linear mapping given the objective (10.29).
- The coordinates ζ_m of x in (10.26) and the coordinates z_m of $\tilde{x}$ in (10.27) must be identical for m= 1, . . . , M since $U^\perp = \text{span}[b_{M+1}, . . . , b_D]$ is the orthogonal complement (see Section 3.6) of $U = \text{span}[b_1, . . . , b_M]$.

We can think of the coordinates as a representation of the projected vector in a new coordinate system defined by $(b_1, . . . , b_M)$. Note that although $\tilde{x} \in R^D$, we only need M coordinates $z_1, . . . , z_M$ to represent this vector; the other D − M coordinates with respect to the basis vectors $(b_{M+1}, . . . , b_D)$ are always 0.

Finding the Basis of the Principal Subspace:

To determine the basis vectors $b_1, . . . , b_M$ of the principal subspace, we rephrase the loss function (10.29) using the results we have so far. This will make it easier to find the basis vectors. To reformulate the loss function, we exploit our results from before and obtain

$$\tilde{\boldsymbol{x}}_n = \sum_{m=1}^{M} z_{mn}\boldsymbol{b}_m \overset{(10.32)}{=} \sum_{m=1}^{M} (\boldsymbol{x}_n^\top \boldsymbol{b}_m)\boldsymbol{b}_m\,. \tag{10.35}$$

We now exploit the symmetry of the dot product, which yields

$$\tilde{\boldsymbol{x}}_n = \left(\sum_{m=1}^{M} \boldsymbol{b}_m \boldsymbol{b}_m^\top\right) \boldsymbol{x}_n\,. \tag{10.36}$$

Since we can generally write the original data point x_n as a linear combination of all basis vectors, it holds that

$$\boldsymbol{x}_n = \sum_{d=1}^{D} z_{dn}\boldsymbol{b}_d \overset{(10.32)}{=} \sum_{d=1}^{D} (\boldsymbol{x}_n^\top \boldsymbol{b}_d)\boldsymbol{b}_d = \left(\sum_{d=1}^{D} \boldsymbol{b}_d \boldsymbol{b}_d^\top\right) \boldsymbol{x}_n \tag{10.37a}$$

$$= \left(\sum_{m=1}^{M} \boldsymbol{b}_m \boldsymbol{b}_m^\top\right) \boldsymbol{x}_n + \left(\sum_{j=M+1}^{D} \boldsymbol{b}_j \boldsymbol{b}_j^\top\right) \boldsymbol{x}_n\,, \tag{10.37b}$$

where we split the sum with D terms into a sum over M and a sum over D − M terms. With this result, we find that the displacement vector $x_n - \tilde{x}_n$, i.e., the difference vector between the original data point and its projection, is

$$\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n = \left(\sum_{j=M+1}^{D} \boldsymbol{b}_j \boldsymbol{b}_j^\top \right) \boldsymbol{x}_n \tag{10.38a}$$

$$= \sum_{j=M+1}^{D} (\boldsymbol{x}_n^\top \boldsymbol{b}_j) \boldsymbol{b}_j \, . \tag{10.38b}$$

This means the difference is exactly the projection of the data point onto the orthogonal complement of the principal subspace: We identify the matrix $\Sigma^{D}_{j=M+1} b_j \, b^{T}_j$ in (10.38a) as the projection matrix that performs this projection. Hence the displacement vector $x_n - \tilde{x}_n$ lies in the subspace that is orthogonal to the principal subspace as illustrated in Figure 3.

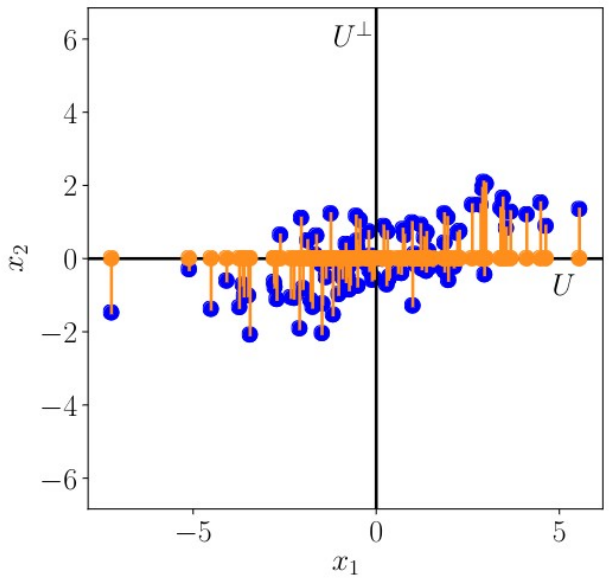


Figure 3
Orthogonal projection and displacement vectors. When projecting data points x_n (blue) onto subspace U1, we obtain $\tilde{x}_n$ (orange). The displacement vector $\tilde{x}_n - x_n$ lies completely in the orthogonal complement U2 of U1.

By construction as a sum of rank-one matrices $b_m b_m^T$ see that BB^T is symmetric and has rank M . Therefore, the average squared reconstruction error can also be written as

$$\frac{1}{N} \sum_{n=1}^{N} \|\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n\|^2 = \frac{1}{N} \sum_{n=1}^{N} \left\| \boldsymbol{x}_n - \boldsymbol{B}\boldsymbol{B}^\top \boldsymbol{x}_n \right\|^2 \tag{10.40a}$$

$$= \frac{1}{N} \sum_{n=1}^{N} \left\| (\boldsymbol{I} - \boldsymbol{B}\boldsymbol{B}^\top) \boldsymbol{x}_n \right\|^2 \, . \tag{10.40b}$$

Finding orthonormal basis vectors $b_1, \ldots, b_M$, which minimize the difference between the original data x_n and their projections $\tilde{x}_n$, is equivalent to finding the best rank-M approximation BB^T of the identity matrix I.
Now we have all the tools to reformulate the loss function (10.29).

$$J_M = \frac{1}{N}\sum_{n=1}^{N}\|\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n\|^2 \overset{(10.38b)}{=} \frac{1}{N}\sum_{n=1}^{N}\left\|\sum_{j=M+1}^{D}(\boldsymbol{b}_j^\top \boldsymbol{x}_n)\boldsymbol{b}_j\right\|^2 . \tag{10.41}$$

We now explicitly compute the squared norm and exploit the fact that the b_j form an ONB, which yields

$$J_M = \frac{1}{N}\sum_{n=1}^{N}\sum_{j=M+1}^{D}(\boldsymbol{b}_j^\top \boldsymbol{x}_n)^2 = \frac{1}{N}\sum_{n=1}^{N}\sum_{j=M+1}^{D}\boldsymbol{b}_j^\top \boldsymbol{x}_n \boldsymbol{b}_j^\top \boldsymbol{x}_n \tag{10.42a}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\sum_{j=M+1}^{D}\boldsymbol{b}_j^\top \boldsymbol{x}_n \boldsymbol{x}_n^\top \boldsymbol{b}_j , \tag{10.42b}$$

where we exploited the symmetry of the dot product in the last step to write $b_j^T x_n = x_n^T b_j$. We now swap the sums and obtain

$$J_M = \sum_{j=M+1}^{D}\boldsymbol{b}_j^\top \underbrace{\left(\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}_n\boldsymbol{x}_n^\top\right)}_{=:\boldsymbol{S}}\boldsymbol{b}_j = \sum_{j=M+1}^{D}\boldsymbol{b}_j^\top \boldsymbol{S}\boldsymbol{b}_j \tag{10.43a}$$

$$= \sum_{j=M+1}^{D}\mathrm{tr}(\boldsymbol{b}_j^\top \boldsymbol{S}\boldsymbol{b}_j) = \sum_{j=M+1}^{D}\mathrm{tr}(\boldsymbol{S}\boldsymbol{b}_j\boldsymbol{b}_j^\top) = \mathrm{tr}\Big(\underbrace{\Big(\sum_{j=M+1}^{D}\boldsymbol{b}_j\boldsymbol{b}_j^\top\Big)}_{\text{projection matrix}}\boldsymbol{S}\Big) , \tag{10.43b}$$

where we exploited the property that the trace operator tr()is linear and invariant to cyclic permutations of its arguments. Since we assumed that our dataset is centered, i.e., E[X] = 0, we identify S as the data covariance matrix. Since the projection matrix in (10.43b) is constructed as a sum of rank-one matrices $b_j b_j^T$ it itself is of rank $D - M$.

Equation (10.43a) implies that we can formulate the average squared reconstruction error equivalently as the covariance matrix of the data,projected onto the orthogonal complement of the principal subspace. Minimizing the average squared reconstruction error is therefore equivalent tominimizing the variance of the data when projected onto the subspace we ignore, i.e., the orthogonal complement of the principal subspace. Equivalently, we maximize the variance of the projection that we retain in the principal subspace.

The average squared reconstruction error, when projecting onto the M - dimensional principal subspace, is

$$J_M = \sum_{j=M+1}^{D} \lambda_j \,, \qquad (10.44)$$

where λ_j are the eigenvalues of the data covariance matrix. Therefore, to minimize (10.44) we need to select the smallest D − M eigenvalues, which then implies that their corresponding eigenvectors are the basis of the orthogonal complement of the principal subspace. Consequently, this means that the basis of the principal subspace comprises the eigenvectors $b_1, \ldots, b_M$ that are associated with the largest M eigenvalues of the data covariance matrix.

We can also think of PCA as a linear auto-encoder as illustrated in Figure 4. An auto-encoder encodes the data $x_n \in R^D$ to a code $z_n \in R^M$ and decodes it to a $\tilde{x}_n$ similar to x_n. The mapping from the data to the code is called the encoder, and the mapping from the code back to the original data space is called the decoder. If we consider linear mappings where the code is given by $z_n = B^T x_n \in R^M$ and we are interested in minimizing the average squared error between the data x_n and its reconstruction $\tilde{x}_n = B z_n$, n = 1, . . . , N

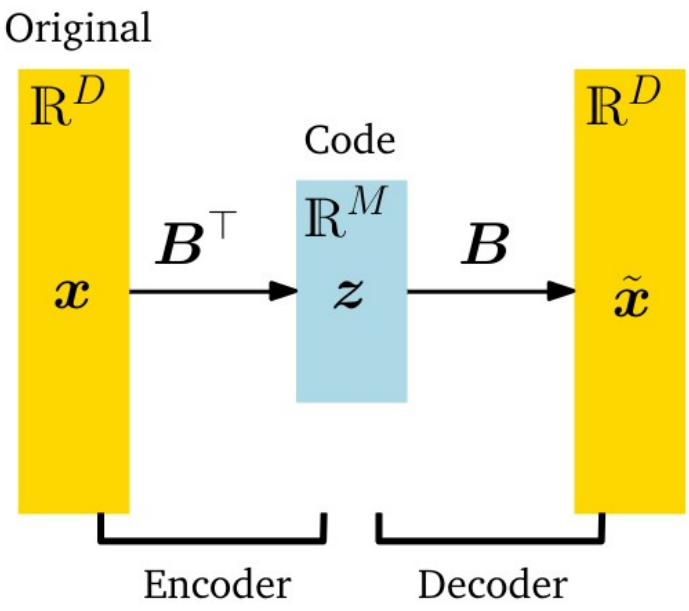


Figure 4
PCA can be viewed as a linear auto-encoder. It encodes the high-dimensional data x into a lower-dimensional representation (code) z ∈ RM and decodes z using a decoder. The decoded vector $\tilde{x}$ is the orthogonal projection of the original data x onto the M -dimensional principal subspace.

Recommender Systems:

Predictive modeling problems that involve the recommendation of products are called recommender systems, a sub-field of machine learning. Examples include the recommendation of books based on previous purchases and purchases by customers like you on Amazon, and the recommendation of movies and TV shows to watch based on your viewing history and viewing history of subscribers like you on Netflix. The development of recommender systems is primarily concerned with linear algebra methods. A simple example is in the calculation of the similarity between sparse customer behavior vectors using distance measures such as Euclidean distance or dot products. Matrix factorization methods like the singular-value decomposition are used widely in recommender systems to distill item and user data to their essence for querying and searching and comparison.

Let D be an n × d ratings matrix representing the ratings of n users for d items. The $(i, j)^{th}$ entry in the matrix D is denoted by x_{ij} , and it represents the rating of user i for item j. The key distinguishing point of a recommendation application is that the vast majority of ratings are missing. This is because users do not specify the ratings of the vast majority of the items in collaborative filtering applications. An example of a ratings matrix with missing entries is shown in Figure 5. In other words, the value of x_{ij} is observed (known) for only a small subset of the entries. The goal of the recommendation problem is to predict the missing ratings from the known ones.

	GLADIATOR	GODFATHER	BEN-HUR	GOODFELLAS	SCARFACE	SPARTACUS
TOM	1			5		2
JIM		5			4	
JOE	5	3		1		
ANN			3			4
JILL				3	5	
SUE	5		4			

Figure 5
A ratings matrix with missing ratings

Let S represent the set of indices of all the observed ratings. Therefore, we have:

$$S = \{(i, j) : xij \text{ is observed}\} \quad (1)$$

As in the case of traditional matrix factorization, we would like to factorize the incomplete ratings matrix D with the use of only the entries in S. In the terminology of recommender systems, the n × k matrix U is referred to as the user factor matrix, and the d × k matrix V is referred to as the item factor matrix. Regularization is particularly important in the case of the collaborative filtering application because of the paucity of observed data. Therefore, an additional term $\lambda/2$ ($\|U\|^2_F + \|V\|^2_F$) is added to the objective function.

Once the user and item factor matrices have been learned, the entire ratings matrix can be reconstructed as U V T . In practice, we only need to reconstruct the (i, j)th entry of matrix D as follows:

$$\hat{x}_{ij} = \sum_{s=1}^{k} u_{is} \cdot v_{js} \tag{2}$$

Note the "hat" symbol (i.e., circumflex) on the rating on the left-hand side to indicate that it is a predicted value rather than an observed value. The error e_{ij} of the prediction is $e_{ij} = x_{ij} - \hat{x}_{ij}$ for ratings that are observed.

One can then formulate the objective function in terms of the observed entries in D as follows:

$$\text{Minimize } J = \frac{1}{2} \sum_{(i,j) \in S} \left(x_{ij} - \sum_{s=1}^{k} u_{is} \cdot v_{js} \right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n} \sum_{s=1}^{k} u_{is}^2 + \frac{\lambda}{2} \sum_{j=1}^{d} \sum_{s=1}^{k} v_{js}^2 \tag{3}$$

we can compute the partial derivative of the objective function with respect to the various parameters as follows:

$$\frac{\partial J}{\partial u_{iq}} = \sum_{j:(i,j) \in S} (e_{ij})(-v_{jq}) + \lambda u_{iq} \quad \forall i \in \{1 \ldots n\}, q \in \{1 \ldots k\}$$

$$\frac{\partial J}{\partial v_{jq}} = \sum_{i:(i,j) \in S} (e_{ij})(-u_{iq}) + \lambda v_{jq} \quad \forall j \in \{1 \ldots d\}, q \in \{1 \ldots k\}$$

One can also define these errors in matrix calculus notation. Let E be an n × d error matrix, which is defined to be eij for each observed entry (i, j) $\in$ S and 0 for each missing entry in the ratings matrix. Note that (unlike vanilla SVD), the error matrix E is already sparse because the vast majority of entries are not specified.

$$\frac{\partial J}{\partial U} = -EV + \lambda U$$

$$\frac{\partial J}{\partial V} = -E^T U + \lambda V$$

Note that the form of the derivative is exactly identical to traditional SVD except for the regularization term and the difference in how the error matrix is defined (to account for missing ratings). Then, the gradient-descent updates for the matrices U and V are as follows:

$$U \Leftarrow U - \alpha \frac{\partial J}{\partial U} = U(1 - \alpha\lambda) + \alpha EV$$
$$V \Leftarrow V - \alpha \frac{\partial J}{\partial V} = V(1 - \alpha\lambda) + \alpha E^T U$$

Here, $\alpha > 0$ is the learning rate. The matrix E can be explicitly materialized as a sparse error matrix, and the above updates can be achieved using only sparse matrix multiplications. Although this approach is referred to as singular value decomposition in the literature on recommender systems (because of the relationship of unconstrained matrix factorization with the SVD optimization model), one will typically not obtain orthogonal columns of U and V with this approach.