The Singular-Value Decomposition

1 Motivation

The singular-value decomposition (SVD) is a fundamental tool in linear algebra. In this section, we introduce three data-science applications where the SVD plays a crucial role.

1.1 Dimensionality reduction

Consider a set of data each consisting of several features. It is often useful to model such data as a set of vectors in a multidimensional space where each dimension corresponds to a feature. The dataset can then be viewed as a cloud of points or– if we prefer a probabilistic perspective– as a probability density in $\mathbb{R}^p$, where p is the number of features. A natural question is in what directions of $\mathbb{R}^p$ the dataset has more or less variation. In directions where there is little or no variation, the properties of the dataset are essentially constant, so we can safely ignore them. This is very useful when the number of features p is large; the remaining directions form a space of reduced dimensionality, which reduces the computational cost of the analysis. Dimensionality reduction is a crucial preprocessing step in big-data applications. The SVD provides a complete characterization of the variance of a p-dimensional dataset in every direction of $\mathbb{R}^p$, and is therefore very useful for this purpose.

1.2 Linear regression

Regression is a fundamental problem in statistics. The goal is to estimate a quantity of interest called the *response* or the *dependent variable*, from the values of several observed variables known as *covariates*, *features* or *independent variables*. For example, if the response is the price of a house, the covariates can be its size, the number of rooms, the year it was built, etc. A regression model produces an estimate of the house prices as a function of all of these factors. More formally, let the response be denoted by a scalar $y \in \mathbb{R}$ and the features by a vector $\vec{x} \in \mathbb{R}^p$ (p is the number of features). The goal is to construct a function h that maps $\vec{x}$ to y, i.e. such that

$$y \approx h\left(\vec{x}\right), \tag{1}$$

from a set of examples, called a training set. This set contains specific responses and their corresponding features

$$\mathcal{S}_{\text{train}} := \left\{ \left(y^{(1)}, \vec{x}^{\,(1)}\right), \left(y^{(2)}, \vec{x}^{\,(2)}\right), \ldots, \left(y^{(n)}, \vec{x}^{\,(n)}\right) \right\}. \tag{2}$$

An immediate complication is that there exist infinite possible functions mapping the features to the right response with no error in the training set (as long as all the feature vectors are distinct).

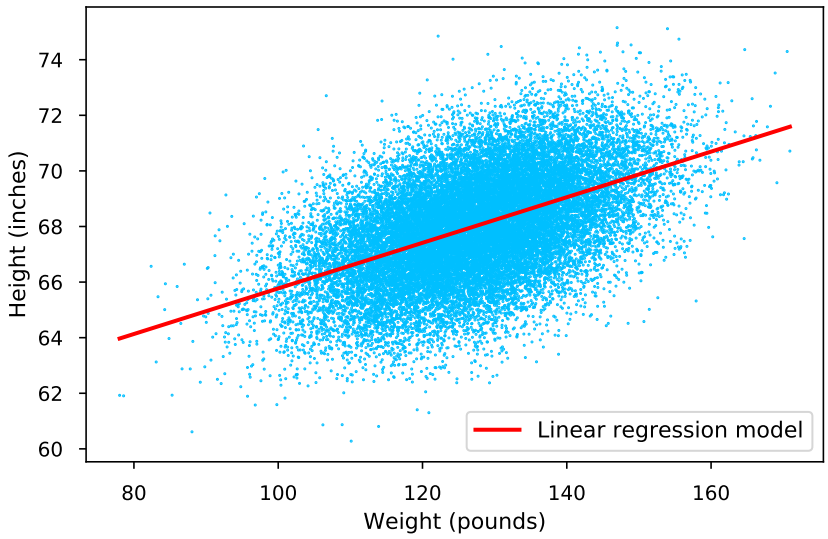


Figure 1: The image shows a linear regression model (red line) of the height of a population of 25,000 people given their weight, superposed on a scatter plot of the actual data (in blue). The data are available here.

It is crucial to constrain h so that it *generalizes* to new examples. A very popular choice is to assume that the regression function is *linear*, so that

$$y \approx \langle \vec{x}, \vec{\beta} \rangle + \beta_0 \tag{3}$$

for a vector of linear coefficients $\vec{\beta} \in \mathbb{R}^p$ and a constant $\beta_0 \in \mathbb{R}$ (strictly speaking the function is affine). Figure 1 shows an example, where the response is the height of a person and the only covariate is their weight. We will study the generalization properties of linear-regression models using the SVD.

1.3 Bilinear model for collaborative filtering

Collaborative filtering is the problem of predicting the interests of users from data. Figure 2 shows an example where the goal is to predict movie ratings. Assume that $y[i, j]$ represents the rating assigned to a movie i by a user j. If we have available a dataset of such ratings, how can we estimate the rating corresponding to a tuple (i, j) that we have not seen before? A reasonable assumption is that some movies are more popular than others, and some users are more generous than others. Let $a[i]$ quantify the popularity of movie i and $b[j]$ the generosity of user j. A large value of $a[i]$ indicates that i is very popular, a large value of $b[j]$ indicates that j is very generous. If we are able to estimate these quantities from data, then a reasonable estimate for the rating given by user j to movie i is given by

$$y[i, j] \approx a[i]b[j]. \tag{4}$$

The model in Eq. (4) is extremely simplistic: different people like different movies! In order to generalize it we can consider r factors that capture the dependence between the ratings and the

Figure 2: A depiction of the collaborative-filtering problem as an incomplete ratings matrix. Each row corresponds to a user and each column corresponds to a movie. The goal is to estimate the missing ratings. The figure is due to Mahdi Soltanolkotabi.

movie/user

$$y[i,j] \approx \sum_{l=1}^{r} a_l[i] b_l[j]. \tag{5}$$

Each user and each movie is now described by r features each, with the following interpretation:

- $a_l[i]$: movie i is positively (> 0), negatively (< 0) or not (≈ 0) associated to factor l.
- $b_l[j]$: user j is positively (> 0), negatively (< 0) or not (≈ 0) associated to factor l.

The model is not linearly dependent on the features. Instead, it is *bilinear*. If the movie features are fixed, the model is linear in the user features; if the user features are fixed, the model is linear in the movie features. As we shall see, the SVD can be used to fit bilinear models.

2 Singular-value decomposition

2.1 Definition

Every real matrix has a singular-value decomposition (SVD).

Theorem 2.1. *Every rank r real matrix $A \in R^{m \times n}$, has a singular-value decomposition (SVD) of the form*

$$A = \begin{bmatrix} \vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_r \end{bmatrix} \begin{bmatrix} s_1 & 0 & \cdots & 0 \\ 0 & s_2 & \cdots & 0 \\ & & \ddots & \\ 0 & 0 & \cdots & s_r \end{bmatrix} \begin{bmatrix} \vec{v}_1^T \\ \vec{v}_2^T \\ \vdots \\ \vec{v}_r^T \end{bmatrix} \tag{6}$$

$$= USV^T, \tag{7}$$

where the singular values $s_1 \geq s_2 \geq \cdots \geq s_r$ are positive real numbers, the left singular vectors $\vec{u}_1$, $\vec{u}_2$, $\ldots \vec{u}_r$ form an orthonormal set, and the right singular vectors $\vec{v}_1$, $\vec{v}_2$, $\ldots \vec{v}_r$ also form an orthonormal set. The SVD is unique if all the singular values are different. If several singular values are the same, the corresponding left singular vectors can be replaced by any orthonormal basis of their span, and the same holds for the right singular vectors.

The SVD of an $m \times n$ matrix with $m \geq n$ can be computed in $\mathcal{O}(mn^2)$. We refer to any graduate linear algebra book for the proof of Theorem 2.1 and for details on how to compute the SVD.

The SVD provides orthonormal bases for the column and row spaces of a matrix.

Lemma 2.2. *The left singular vectors are an orthonormal basis for the column space, whereas the right singular vectors are an orthonormal basis for the row space.*

Proof. We prove the statement for the column space, the proof for the row space is identical. All left singular vectors belong to the column space because $\vec{u}_i = A\left(s_i^{-1}\vec{v}_i\right)$. In addition, every column of A is in their span because $A_{:i} = U\left(SV^T\vec{e}_i\right)$. Since they form an orthonormal set by Theorem 2.1, this completes the proof. □

The SVD presented in Theorem 2.1 can be augmented so that the number of singular values equals $\min(m, n)$. The additional singular values are all equal to zero. Their corresponding left and right singular vectors are orthonormal sets of vectors in the orthogonal complements of the column and row space respectively. If the matrix is tall or square, the additional right singular vectors are a basis of the null space of the matrix.

Corollary 2.3 (Singular-value decomposition)**.** *Every rank r real matrix $A \in R^{m \times n}$, where $m \geq n$, has a singular-value decomposition (SVD) of the form*

$$A := [\underbrace{\vec{u}_1\ \vec{u}_2\ \cdots\ \vec{u}_r}_{\textit{Basis of } \operatorname{range}(A)}\ \vec{u}_{r+1}\ \cdots\ \vec{u}_n] \begin{bmatrix} s_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & s_2 & \cdots & 0 & 0 & \cdots & 0 \\ & & & \cdots & & & \\ 0 & 0 & \cdots & s_r & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ & & & \cdots & & & \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{bmatrix} [\underbrace{\vec{v}_1\ \vec{v}_2\ \cdots\ \vec{v}_r}_{\textit{Basis of } \operatorname{row}(A)}\ \underbrace{\vec{v}_{r+1}\ \cdots\ \vec{v}_n}_{\textit{Basis of } \operatorname{null}(A)}]^T, \tag{8}$$

where the singular values $s_1 \geq s_2 \geq \cdots \geq s_r$ are positive real numbers, the left singular vectors $\vec{u}_1$, $\vec{u}_2$, $\ldots$, $\vec{u}_m$ form an orthonormal set in $\mathbb{R}^m$, and the right singular vectors $\vec{v}_1$, $\vec{v}_2$, $\ldots$, $\vec{v}_m$ form an orthonormal basis for $\mathbb{R}^n$.

If the matrix is fat, we can define a similar augmentation, where the additional left singular vectors form an orthonormal basis of the orthogonal complement of the range.

2.2 Geometric interpretation

The SVD decomposes the action of a matrix $A \in \mathbb{R}^{m\times n}$ on a vector $\vec{x} \in \mathbb{R}^n$ into three simple steps, illustrated in Figure 3:

1. Rotation of $\vec{x}$ to align the component of $\vec{x}$ in the direction of the ith right singular vector $\vec{v}_i$ with the ith axis:
$$V^T\vec{x} = \sum_{i=1}^{n} \langle \vec{v}_i, \vec{x}\rangle \vec{e}_i. \tag{9}$$
2. Scaling of each axis by the corresponding singular value
$$SV^T\vec{x} = \sum_{i=1}^{n} s_i\langle \vec{v}_i, \vec{x}\rangle \vec{e}_i. \tag{10}$$
3. Rotation to align the ith axis with the ith left singular vector
$$USV^T\vec{x} = \sum_{i=1}^{n} s_i\langle \vec{v}_i, \vec{x}\rangle \vec{u}_i. \tag{11}$$

This geometric perspective of the SVD reveals an important property: the maximum scaling produced by a matrix is equal to the maximum singular value. The maximum is achieved when the matrix is applied to any vector in the direction of the right singular vector $\vec{v}_1$. If we restrict our attention to the orthogonal complement of $\vec{v}_1$, then the maximum scaling is the second singular value, due to the orthogonality of the singular vectors. In general, the direction of maximum scaling orthogonal to the first $i-1$ left singular vectors is equal to the ith singular value and occurs in the direction of the ith singular vector.

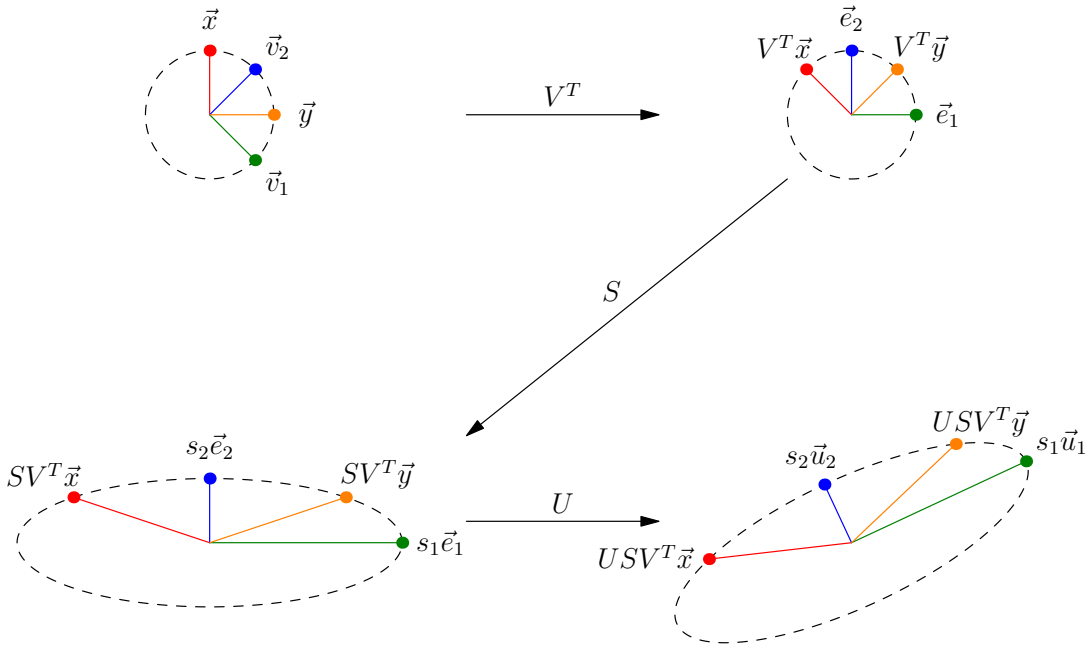
Theorem 2.4. *For any matrix $A \in \mathbb{R}^{m\times n}$, with SVD given by (8), the singular values satisfy*

$$\begin{aligned}
s_1 &= \max_{\{\|\vec{x}\|_2=1 \mid \vec{x}\in\mathbb{R}^n\}} \|A\vec{x}\|_2 && (12)\\
&= \max_{\{\|\vec{y}\|_2=1 \mid \vec{y}\in\mathbb{R}^m\}} \|A^T\vec{y}\|_2, && (13)\\
s_i &= \max_{\{\|\vec{x}\|_2=1 \mid \vec{x}\in\mathbb{R}^n, \vec{x}\perp\vec{v}_1,\ldots,\vec{v}_{i-1}\}} \|A\vec{x}\|_2, && (14)\\
&= \max_{\{\|\vec{y}\|_2=1 \mid \vec{y}\in\mathbb{R}^m, \vec{y}\perp\vec{u}_1,\ldots,\vec{u}_{i-1}\}} \|A^T\vec{y}\|_2, \qquad 2\le i\le \min\{m,n\}, && (15)
\end{aligned}$$

the right singular vectors satisfy

$$\begin{aligned}
\vec{v}_1 &= \operatorname*{arg\,max}_{\{\|\vec{x}\|_2=1 \mid \vec{x}\in\mathbb{R}^n\}} \|A\vec{x}\|_2, && (16)\\
\vec{v}_i &= \operatorname*{arg\,max}_{\{\|\vec{x}\|_2=1 \mid \vec{x}\in\mathbb{R}^n, \vec{x}\perp\vec{v}_1,\ldots,\vec{v}_{i-1}\}} \|A\vec{x}\|_2, \qquad 2\le i\le m, && (17)
\end{aligned}$$

(a) $s_1 = 3$, $s_2 = 1$.



(b) $s_1 = 3$, $s_2 = 0$.

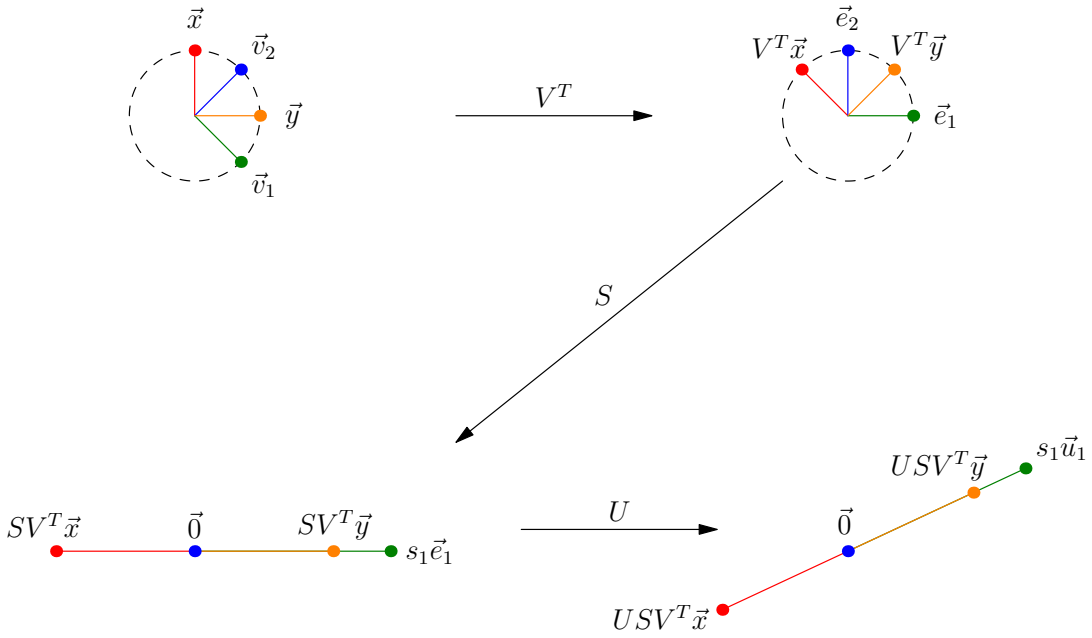


Figure 3: The action of any matrix can be decomposed into three steps: rotation to align the right singular vectors to the axes, scaling by the singular values and a final rotation to align the axes with the left singular vectors. In image (b) the second singular value is zero, so the matrix projects two-dimensional vectors onto a one-dimensional subspace.

and the left singular vectors satisfy

$$\vec{u}_1 = \underset{\{\|\vec{y}\|_2=1 \mid \vec{y}\in\mathbb{R}^m\}}{\arg\max} \|A^T\vec{y}\|_2, \tag{18}$$

$$\vec{u}_i = \underset{\{\|\vec{y}\|_2=1 \mid \vec{y}\in\mathbb{R}^m, \vec{y}\perp\vec{u}_1,\ldots,\vec{u}_{i-1}\}}{\arg\max} \|A^T\vec{y}\|_2, \qquad 2 \leq i \leq n. \tag{19}$$

Proof. Consider a vector $\vec{x} \in \mathbb{R}^n$ with unit ℓ_2 norm that is orthogonal to $\vec{v}_1, \ldots, \vec{v}_{i-1}$, where $1 \leq i \leq n$ (if $i = 1$ then $\vec{x}$ is just an arbitrary vector). We express $\vec{x}$ in terms of the right singular vectors of A and a component that is orthogonal to their span

$$\vec{x} = \sum_{j=i}^{n} \alpha_j \vec{v}_j + \mathcal{P}_{\text{row}(A)^\perp} \vec{x} \tag{20}$$

where $1 = \|\vec{x}\|_2^2 \geq \sum_{j=i}^{n} \alpha_j^2$. By the ordering of the singular values in Theorem 2.1

$$\|A\vec{x}\|_2^2 = \langle \sum_{k=1}^{n} s_k \langle \vec{v}_k, \vec{x}\rangle \vec{u}_k, \sum_{k=1}^{n} s_k \langle \vec{v}_k, \vec{x}\rangle \vec{u}_k \rangle \quad \text{by (11)} \tag{21}$$

$$= \sum_{k=1}^{n} s_k^2 \langle \vec{v}_k, \vec{x}\rangle^2 \quad \text{because } \vec{u}_1, \ldots, \vec{u}_n \text{ are orthonormal} \tag{22}$$

$$= \sum_{k=1}^{n} s_k^2 \langle \vec{v}_k, \sum_{j=i}^{n} \alpha_j \vec{v}_j + \mathcal{P}_{\text{row}(A)^\perp} \vec{x}\rangle^2 \tag{23}$$

$$= \sum_{j=i}^{n} s_j^2 \alpha_j^2 \qquad \text{because } \vec{v}_1, \ldots, \vec{v}_n \text{ are orthonormal} \tag{24}$$

$$\leq s_i^2 \sum_{j=i}^{n} \alpha_j^2 \quad \text{because } s_i \geq s_{i+1} \geq \ldots \geq s_n \tag{25}$$

$$\leq s_i^2 \quad \text{by (20)}. \tag{26}$$

This establishes (12) and (14). To prove (16) and (17) we show that $\vec{v}_i$ achieves the maximum

$$\|A\vec{v}_i\|_2^2 = \sum_{k=1}^{n} s_k^2 \langle \vec{v}_k, \vec{v}_i\rangle^2 \tag{27}$$

$$= s_i^2. \tag{28}$$

The same argument applied to A^T establishes (13), (18), (19) and (15). □

3 Principal component analysis

3.1 Quantifying directional variation

In this section we describe how to quantify the variation of a dataset embedded in a p-dimensional space. We can take two different perspectives:

- **Probabilistic**: The data are samples from a p-dimensional random vector $\vec{\mathbf{x}}$ characterized by its probability distribution.
- **Geometric**: The data are just a just a cloud of points in $\mathbb{R}^p$.

Both perspectives are intertwined. Let us first consider a set of scalar values $\{x_1, x_2, \ldots, x_n\}$. If we interpret them as samples from a random variable $\mathbf{x}$, then we can quantify their average variation using the variance of $\mathbf{x}$, defined as the expected value of its squared deviation from the mean,

$$\mathrm{Var}\left(\mathbf{x}\right) := \mathrm{E}\left(\left(\mathbf{x} - \mathrm{E}\left(\mathbf{x}\right)\right)^2\right). \tag{29}$$

Recall that the square root of the variance is known as the *standard deviation* of the random variable. If we prefer a geometric perspective we can instead consider the average of the squared deviation of each point from their average

$$\mathrm{var}\left(x_1, x_2, \ldots, x_n\right) := \frac{1}{n-1}\sum_{i=1}^{n}\left(x_i - \mathrm{av}\left(x_1, x_2, \ldots, x_n\right)\right)^2, \tag{30}$$

$$\mathrm{av}\left(x_1, x_2, \ldots, x_n\right) := \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{31}$$

This quantity is the *sample variance* of the points, which can be interpreted as an estimate of their variance if they are generated from the same probability distribution. Similarly, the average is the sample mean, and can be interpreted as an estimate of the mean. More formally, let us consider a set of random variables $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ with the same mean μ and variance σ^2. We have

$$\mathrm{E}\left(\mathrm{av}\left(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\right)\right) = \mu, \tag{32}$$

$$\mathrm{E}\left(\mathrm{var}\left(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\right)\right) = \sigma^2, \tag{33}$$

by linearity of expectation. In statistics jargon, the sample mean and variance are unbiased estimators of the true mean and variance. In addition, if the variables are independent, by linearity of expectation,

$$\mathrm{E}\left(\left(\mathrm{av}\left(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\right) - \mu\right)^2\right) = \frac{\sigma^2}{n}. \tag{34}$$

Furthermore, if the fourth moment of the random variables $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ is bounded (i.e. there is some constant c such that $\mathrm{E}\left(\mathbf{x}_i^4\right) \leq c$ for all i), $\mathrm{E}\left(\left(\mathrm{var}\left(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\right) - \sigma^2\right)^2\right)$ also scales as $1/n$. In words, the mean square error incurred by the estimators decreases linearly with the number of data. When the number of data is large, the probabilistic and geometric viewpoints are essentially equivalent, although it is worth emphasizing that the geometric perspective is still relevant if we want to avoid making probabilistic assumptions.

In order to extend the notion of average variation to a multidimensional dataset, we first need to center the data. This can be done by subtracting their average. Then we can consider the projection of the data onto any direction of interest, by computing the inner product of each centered data point with the corresponding unit-norm vector $\vec{v} \in \mathbb{R}^p$. If we take a probabilistic

perspective, where the data are samples of a p-dimensional random vector $\vec{\mathbf{x}}$, the variation in that direction is given by the variance of the projection $\mathrm{Var}\left(\vec{v}^T\vec{\mathbf{x}}\right)$. This variance can be estimated using the sample variance $\mathrm{var}\left(\vec{v}^T\vec{x}_1, \vec{v}^T\vec{x}_2, \ldots, \vec{v}^T\vec{x}_n\right)$, where $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^p$ are the actual data. This has a geometric interpretation in its own right, as the average square deviation of the projected data from their average.

3.2 Covariance matrix

In this section we show that the covariance matrix captures the average variation of a dataset in any direction. The covariance of two random variables $\mathbf{x}$ and $\mathbf{y}$ provides an average of their joint fluctuations around their respective means,

$$\mathrm{Cov}\left(\mathbf{x}, \mathbf{y}\right) := \mathrm{E}\left(\left(\mathbf{x} - \mathrm{E}\left(\mathbf{x}\right)\right)\left(\mathbf{y} - \mathrm{E}\left(\mathbf{y}\right)\right)\right). \tag{35}$$

The covariance matrix of a random vector contains the variance of each component in the diagonal and the pairwise covariances between different components in the off diagonals.

Definition 3.1. *The covariance matrix of a random vector $\vec{\mathbf{x}}$ is defined as*

$$\Sigma_{\vec{\mathbf{x}}} := \begin{bmatrix} \mathrm{Var}\left(\vec{\mathbf{x}}\left[1\right]\right) & \mathrm{Cov}\left(\vec{\mathbf{x}}\left[1\right], \vec{\mathbf{x}}\left[2\right]\right) & \cdots & \mathrm{Cov}\left(\vec{\mathbf{x}}\left[1\right], \vec{\mathbf{x}}\left[p\right]\right) \\ \mathrm{Cov}\left(\vec{\mathbf{x}}\left[2\right], \vec{\mathbf{x}}\left[1\right]\right) & \mathrm{Var}\left(\vec{\mathbf{x}}\left[2\right]\right) & \cdots & \mathrm{Cov}\left(\vec{\mathbf{x}}\left[2\right], \vec{\mathbf{x}}\left[p\right]\right) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{Cov}\left(\vec{\mathbf{x}}\left[n\right], \vec{\mathbf{x}}\left[1\right]\right) & \mathrm{Cov}\left(\vec{\mathbf{x}}\left[n\right], \vec{\mathbf{x}}\left[2\right]\right) & \cdots & \mathrm{Var}\left(\vec{\mathbf{x}}\left[p\right]\right) \end{bmatrix} \tag{36}$$

$$= \mathrm{E}\left(\vec{\mathbf{x}}\vec{\mathbf{x}}^T\right) - \mathrm{E}(\vec{\mathbf{x}})\mathrm{E}(\vec{\mathbf{x}})^T. \tag{37}$$

Note that if all the entries of a vector are uncorrelated, then its covariance matrix is diagonal. Using linearity of expectation, we obtain a simple expression for the covariance matrix of the linear transformation of a random vector.

Theorem 3.2 (Covariance matrix after a linear transformation)**.** *Let $\vec{\mathbf{x}}$ be a random vector of dimension n with covariance matrix Σ. For any matrix $A \in \mathbb{R}^{m\times p}$,*

$$\Sigma_{A\vec{\mathbf{x}}} = A\Sigma_{\vec{\mathbf{x}}}A^T. \tag{38}$$

Proof. By linearity of expectation

$$\Sigma_{A\vec{\mathbf{x}}} = \mathrm{E}\left(\left(A\vec{\mathbf{x}}\right)\left(A\vec{\mathbf{x}}\right)^T\right) - \mathrm{E}\left(A\vec{\mathbf{x}}\right)\mathrm{E}\left(A\vec{\mathbf{x}}\right)^T \tag{39}$$

$$= A\left(\mathrm{E}\left(\vec{\mathbf{x}}\vec{\mathbf{x}}^T\right) - \mathrm{E}(\vec{\mathbf{x}})\mathrm{E}(\vec{\mathbf{x}})^T\right)A^T \tag{40}$$

$$= A\Sigma_{\vec{\mathbf{x}}}A^T. \tag{41}$$

□

An immediate corollary of this result is that we can easily decode the variance of the random vector *in any direction* from the covariance matrix.

Corollary 3.3. *Let $\vec{v}$ be a unit-ℓ_2-norm vector,*

$$\text{Var}\left(\vec{v}^T\vec{\mathbf{x}}\right) = \vec{v}^T\Sigma_{\vec{\mathbf{x}}}\vec{v}. \tag{42}$$

The variance of a random variable cannot be negative by definition (it is the expectation of a nonnegative quantity). The corollary therefore implies that covariance matrices are positive semidefinite, i.e. for any $\Sigma_{\vec{\mathbf{x}}}$ and any vector $\vec{v}$, $\vec{v}^T\Sigma_{\vec{\mathbf{x}}}\vec{v} \geq 0$. Symmetric, positive semidefinite matrices have the same left and right singular vectors[1]. The SVD of the covariance matrix of a random vector $\vec{\mathbf{x}}$ is therefore of the form

$$\Sigma_{\vec{\mathbf{x}}} = USU^T \tag{43}$$

$$= \begin{bmatrix}\vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_n\end{bmatrix} \begin{bmatrix} s_1 & 0 & \cdots & 0 \\ 0 & s_2 & \cdots & 0 \\ & & \cdots & \\ 0 & 0 & \cdots & s_n \end{bmatrix} \begin{bmatrix}\vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_n\end{bmatrix}^T. \tag{44}$$

The SVD of the covariance matrix provides very valuable information about the directional variance of the random vector.

Theorem 3.4. *Let $\vec{\mathbf{x}}$ be a random vector of dimension p with covariance matrix $\Sigma_{\vec{\mathbf{x}}}$. The SVD of $\Sigma_{\vec{\mathbf{x}}}$ given by (44) satisfies*

$$s_1 = \max_{||\vec{v}||_2=1} \text{Var}\left(\vec{v}^T\vec{\mathbf{x}}\right), \tag{45}$$

$$\vec{u}_1 = \arg\max_{||\vec{v}||_2=1} \text{Var}\left(\vec{v}^T\vec{\mathbf{x}}\right), \tag{46}$$

$$s_k = \max_{||\vec{v}||_2=1, \vec{v}\perp\vec{u}_1,\ldots,\vec{u}_{k-1}} \text{Var}\left(\vec{v}^T\vec{\mathbf{x}}\right), \tag{47}$$

$$\vec{u}_k = \arg\max_{||\vec{v}||_2=1, \vec{v}\perp\vec{u}_1,\ldots,\vec{u}_{k-1}} \text{Var}\left(\vec{v}^T\vec{\mathbf{x}}\right), \tag{48}$$

for $2 \leq k \leq p$.

Proof. By Corollary 3.3, the directional variance equals

$$\vec{v}^T\Sigma_{\vec{\mathbf{x}}}\vec{v} = \vec{v}^TUSU^T\vec{v} \tag{49}$$

$$= \left|\left|\sqrt{S}U^T\vec{v}\right|\right|_2^2, \tag{50}$$

where $\sqrt{S}$ is a diagonal matrix that contains the square root of the entries of S. The result then follows from Theorem 2.4 since $\sqrt{S}U^T$ is a valid SVD. □

In words, $\vec{u}_1$ is the *direction of maximum variance*. The second singular vector $\vec{u}_2$ is the direction of maximum variation that is orthogonal to $\vec{u}_1$. In general, the eigenvector $\vec{u}_k$ reveals the direction of maximum variation that is orthogonal to $\vec{u}_1$, $\vec{u}_2$, ..., $\vec{u}_{k-1}$. Finally, $\vec{u}_p$ is the direction of minimum variance. Figure 4 illustrates this with an example, where $p = 2$.

[1] If a matrix is symmetric but not positive semidefinite, some right singular vectors $\vec{v}_j$ may equal $-\vec{u}_j$ instead of $\vec{u}_j$.

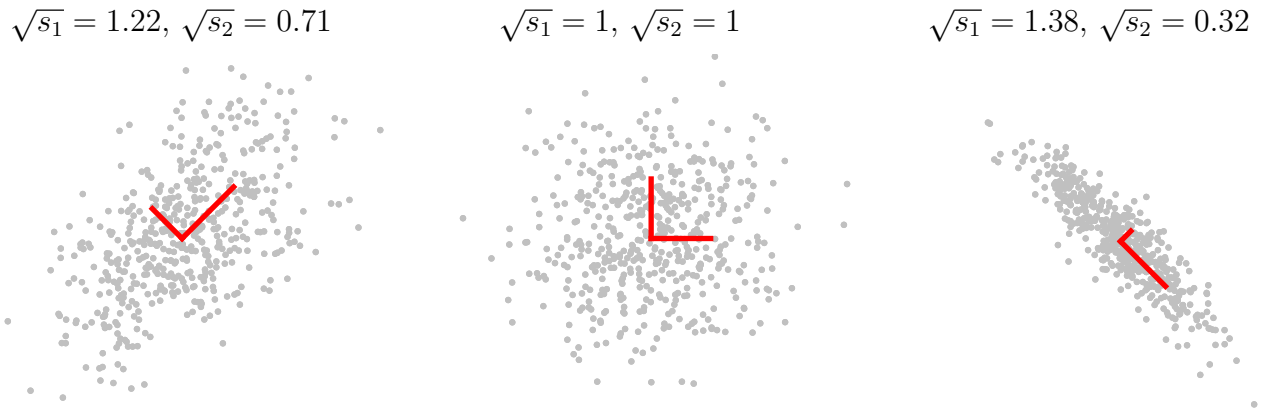


Figure 4: Samples from bivariate Gaussian random vectors with different covariance matrices are shown in gray. The eigenvectors of the covariance matrices are plotted in red. Each is scaled by the square roof of the corresponding singular value s_1 or s_2.

3.3 Sample covariance matrix

A natural estimator of the covariance matrix is the sample covariance matrix.

Definition 3.5 (Sample covariance matrix)**.** *Let $\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}$ be a set of m-dimensional real-valued data vectors, where each dimension corresponds to a different feature. The sample covariance matrix of these vectors is the $p \times p$ matrix*

$$\Sigma\left(\vec{x}_1, \ldots, \vec{x}_n\right) := \frac{1}{n-1} \sum_{i=1}^{n} \left(\vec{x}_i - \text{av}\left(\vec{x}_1, \ldots, \vec{x}_n\right)\right) \left(\vec{x}_i - \text{av}\left(\vec{x}_1, \ldots, \vec{x}_n\right)\right)^T, \tag{51}$$

where the center or average is defined as

$$\text{av}\left(\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\right) := \frac{1}{n} \sum_{i=1}^{n} \vec{x}_i \tag{52}$$

contains the sample mean of each feature. The (i, j) entry of the covariance matrix, where $1 \leq i, j \leq p$, is given by

$$\Sigma\left(\vec{x}_1, \ldots, \vec{x}_n\right)_{ij} = \begin{cases} \text{var}\left(\vec{x}_1\left[i\right], \ldots, \vec{x}_n\left[i\right]\right) & \text{if } i = j, \\ \text{cov}\left(\left(\vec{x}_1\left[i\right], \vec{x}_1\left[j\right]\right), \ldots, \left(\vec{x}_n\left[i\right], \vec{x}_n\left[j\right]\right)\right) & \text{if } i \neq j, \end{cases} \tag{53}$$

where the sample covariance is defined as

$$\text{cov}\left(\left(x_1, y_1\right), \ldots, \left(x_n, y_n\right)\right) := \frac{1}{n-1} \sum_{i=1}^{n} \left(x_i - \text{av}\left(x_1, \ldots, x_n\right)\right) \left(y_i - \text{av}\left(y_1, \ldots, y_n\right)\right). \tag{54}$$

By linearity of expectation, as sketched briefly in Section 3.1 the sample variance converges to the true variance if the data are independent (and the fourth moment is bounded). By the same argument, the sample covariance also converges to the true covariance under similar conditions.

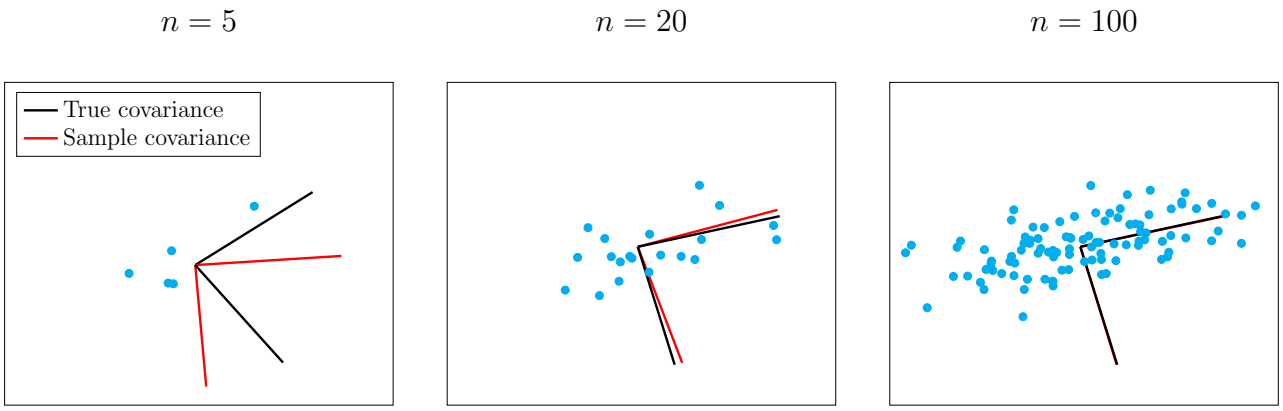


Figure 5: Singular vectors of the sample covariance matrix of n iid samples from a bivariate Gaussian distribution (red) compared to the singular vectors of its covariance matrix (black).

As a result, the sample covariance matrix is often an accurate estimate of the covariance matrix, as long as enough samples are available. Figure 5 illustrates this with a numerical example, where the singular vectors of the sample covariance matrix converge to the singular vectors of the covariance matrix as the number of samples increases.

Even if the data are not interpreted as samples from a distribution, the sample covariance matrix has an intuitive geometric interpretation. For any unit-norm vector $\vec{v} \in \mathbb{R}^p$, the sample variance of the data set in the direction of $\vec{v}$ is given by

$$\text{var}\left(\vec{v}^T\vec{x}_1, \ldots, \vec{v}^T\vec{x}_n\right) = \frac{1}{n-1}\sum_{i=1}^{n}\left(\vec{v}^T\vec{x}_i - \text{av}\left(\vec{v}^T\vec{x}_1, \ldots, \vec{v}^T\vec{x}_n\right)\right)^2 \tag{55}$$

$$= \frac{1}{n-1}\sum_{i=1}^{n}\left(\vec{v}^T\left(\vec{x}_i - \text{av}\left(\vec{x}_1, \ldots, \vec{x}_n\right)\right)\right)^2 \tag{56}$$

$$= \vec{v}^T\left(\frac{1}{n-1}\sum_{i=1}^{n}\left(\vec{x}_i - \text{av}\left(\vec{x}_1, \ldots, \vec{x}_n\right)\right)\left(\vec{x}_i - \text{av}\left(\vec{x}_1, \ldots, \vec{x}_n\right)\right)^T\right)\vec{v}$$

$$= \vec{v}^T\Sigma\left(\vec{x}_1, \ldots, \vec{x}_n\right)\vec{v}. \tag{57}$$

Using the sample covariance matrix we can extract the sample variance in every direction of the ambient space.

3.4 Principal component analysis

The previous sections show that computing the SVD of the sample covariance matrix of a dataset reveals the variation of the data in different directions. This is known as principal-component analysis (PCA).

Algorithm 3.6 (Principal component analysis)**.** *Given n data vectors $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n \in \mathbb{R}^d$, compute the SVD of the sample covariance of the data. The singular vectors are called the principal directions of the data. The principal components are the coefficients of the centered data vectors when expressed in the basis of principal directions.*

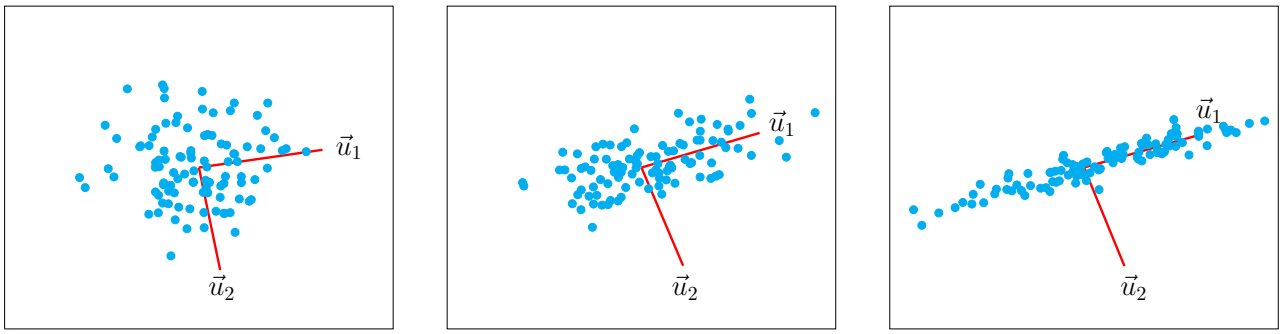


Figure 6: PCA of a dataset with $n = 100$ 2D vectors with different configurations. The two first singular values reflect how much energy is preserved by projecting onto the two first principal directions.

An equivalent procedure to compute the principal directions is to center the data

$$\vec{c}_i = \vec{x}_i - \text{av}\left(\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\right), \qquad 1 \leq i \leq n, \tag{58}$$

and then compute the SVD of the matrix

$$C = \begin{bmatrix} \vec{c}_1 & \vec{c}_2 & \cdots & \vec{c}_n \end{bmatrix}. \tag{59}$$

The result is the same because the sample covariance matrix is equal to $\frac{1}{n-1} CC^T$.

By Eq. (57) and Theorem 2.4 the principal directions reveal the directions of maximum variation of the data.

Corollary 3.7. *Let $\vec{u}_1, \ldots, \vec{u}_k$ be the $k \leq \min\{p, n\}$ first principal directions obtained by applying Algorithm 3.6 to a set of vectors $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^p$. Then the principal directions satisfy*

$$\vec{u}_1 = \underset{\{\|\vec{v}\|_2 = 1 \,|\, \vec{v} \in \mathbb{R}^n\}}{\arg\max} \text{var}\left(\vec{v}^T \vec{x}_1, \ldots, \vec{v}^T \vec{x}_n\right), \tag{60}$$

$$\vec{u}_i = \underset{\{\|\vec{v}\|_2 = 1 \,|\, \vec{v} \in \mathbb{R}^n, \vec{v} \perp \vec{u}_1, \ldots, \vec{u}_{i-1}\}}{\arg\max} \text{var}\left(\vec{v}^T \vec{x}_1, \ldots, \vec{v}^T \vec{x}_n\right), \qquad 2 \leq i \leq k, \tag{61}$$

and the associated singular values satisfy

$$\frac{s_1}{n-1} = \max_{\{\|\vec{v}\|_2 = 1 \,|\, \vec{v} \in \mathbb{R}^n\}} \text{var}\left(\vec{v}^T \vec{x}_1, \ldots, \vec{v}^T \vec{x}_n\right), \tag{62}$$

$$\frac{s_i}{n-1} = \max_{\{\|\vec{v}\|_2 = 1 \,|\, \vec{v} \in \mathbb{R}^n, \vec{v} \perp \vec{u}_1, \ldots, \vec{u}_{i-1}\}} \text{var}\left(\vec{v}^T \vec{x}_1, \ldots, \vec{v}^T \vec{x}_n\right), \qquad 2 \leq i \leq k \tag{63}$$

for any $2 \leq k \leq p$.

In words, $\vec{u}_1$ is the direction of maximum variation, $\vec{u}_2$ is the direction of maximum variation orthogonal to $\vec{u}_1$, and in general $\vec{u}_k$ is the direction of maximum variation that is orthogonal to $\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_{k-1}$. Figure 6 shows the principal directions for several 2D examples.

In the following example, we apply PCA to a set of images.

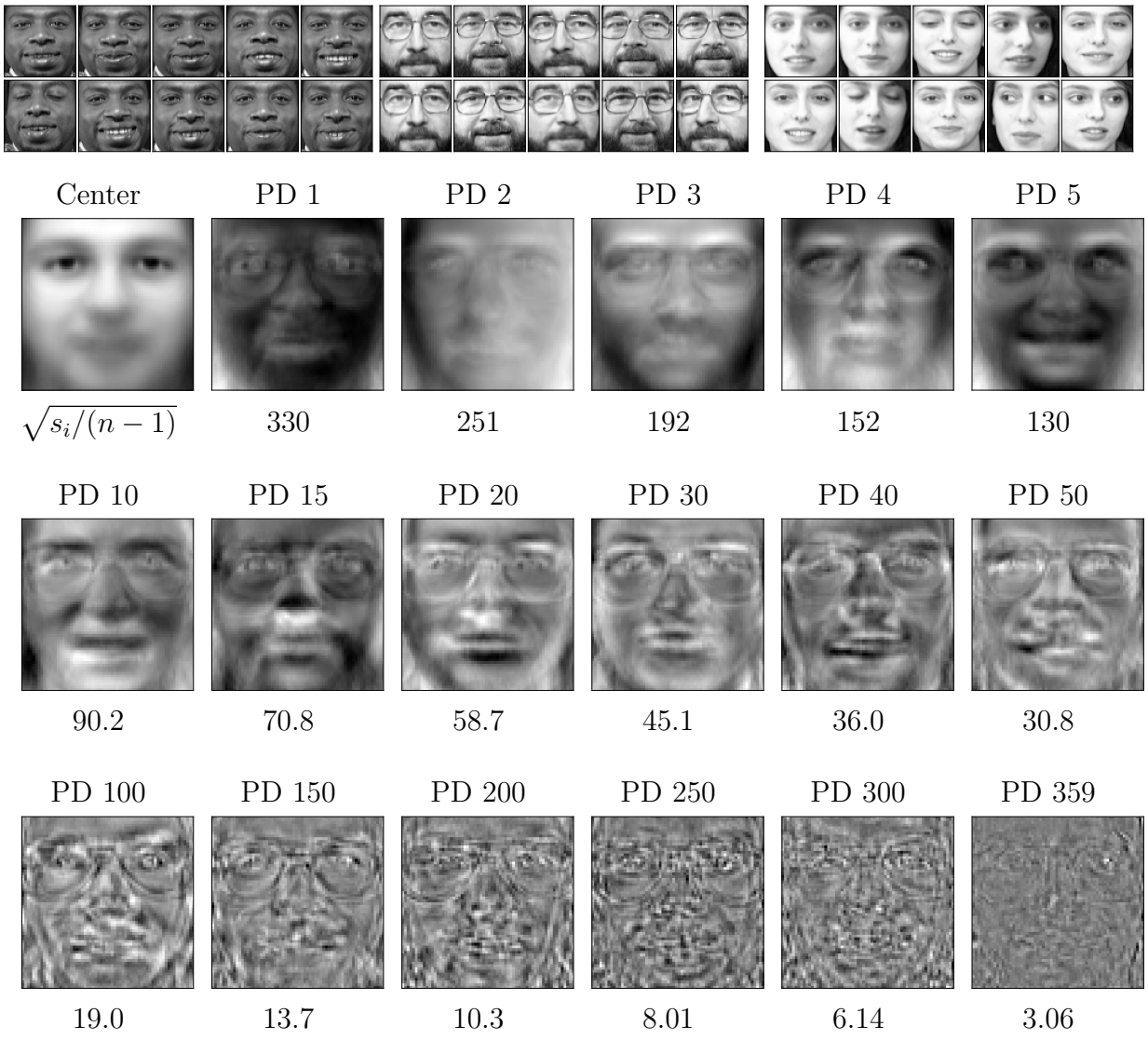


Figure 7: The top row shows the data corresponding to three different individuals in the Olivetti data set. The average and the principal directions (PD) obtained by applying PCA are depicted below. The corresponding singular value is listed below each principal direction.

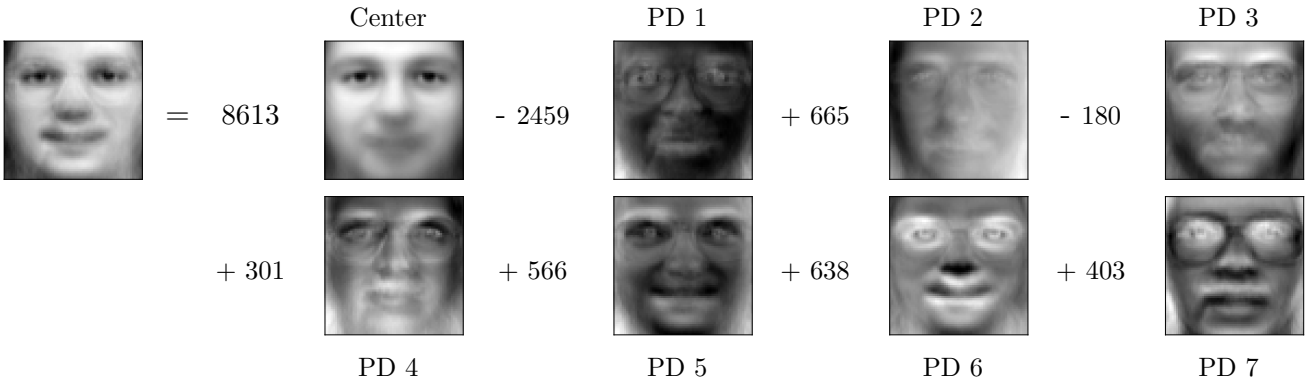


Figure 8: Projection of one of the faces $\vec{x}$ onto the first 7 principal directions and the corresponding decomposition into the 7 first principal components.

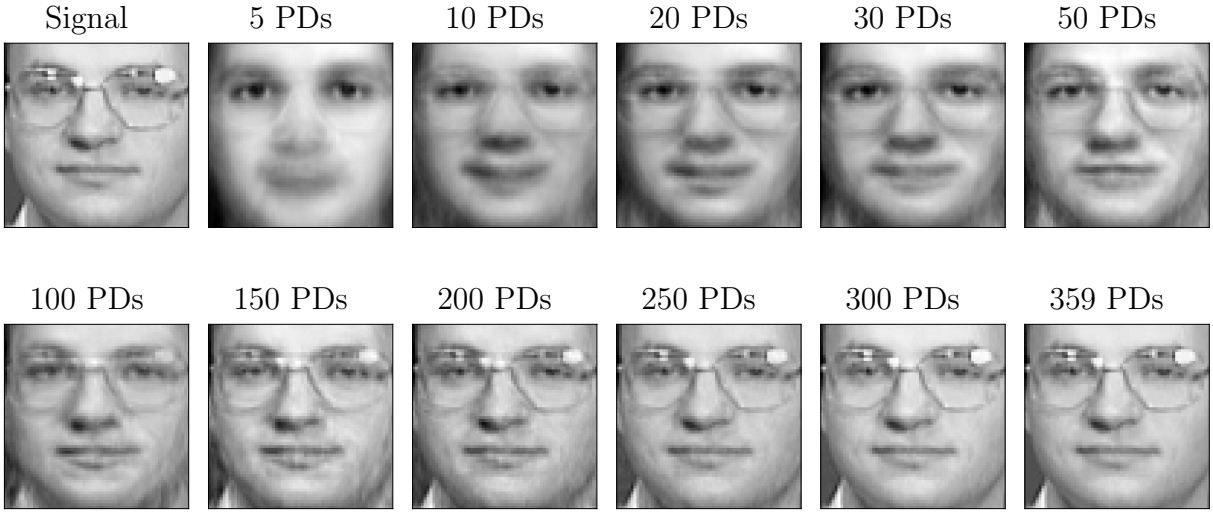


Figure 9: Projection of a face on different numbers of principal directions.

Example 3.8 (PCA of faces)**.** In this example we consider the Olivetti Faces data set[2]. The data set contains 400 64×64 images taken from 40 different subjects (10 per subject). We vectorize each image so that each pixel is interpreted as a different feature. Figure 7 shows the center of the data and several principal directions, together with their associated singular values. The principal directions corresponding to the larger singular values seem to capture low-resolution structure, whereas the ones corresponding to the smallest singular values incorporate more intricate details.

Figure 8 shows the projection of one of the faces onto the first 7 principal directions and the corresponding decomposition into its 7 first principal components. Figure 9 shows the projection of the same face onto increasing numbers of principal directions. As suggested by the visualization of the principal directions in Figure 7, the lower-dimensional projections produce blurry images.

△

[2]Available at http://www.cs.nyu.edu/~roweis/data.html

3.5 Linear dimensionality reduction via PCA

Data containing a large number of features can be costly to analyze. Dimensionality reduction is a useful preprocessing step, which consists of representing the data with a smaller number of variables. For data modeled as vectors in an ambient space $\mathbb{R}^p$ (each dimension corresponds to a feature), this can be achieved by projecting the vectors onto a lower-dimensional space $\mathbb{R}^k$, where $k < m$. If the projection is orthogonal, the new representation can be computed using an orthogonal basis for the lower-dimensional subspace $\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_k$. Each data vector $\vec{x} \in \mathbb{R}^p$ is described using the coefficients of its representation in the basis: $\langle \vec{b}_1, \vec{x} \rangle, \langle \vec{b}_2, \vec{x} \rangle, \ldots, \langle \vec{b}_k, \vec{x} \rangle$.

Given a data set of n vectors $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n \in \mathbb{R}^p$, a crucial question is how to find the k-dimensional subspace that contains the most information. If we are interested in preserving the ℓ_2 norm of the data, the answer is given by the SVD. If the data are centered, the best k-dimensional subspace is spanned by the first k principal directions.

Theorem 3.9 (Optimal subspace for orthogonal projection)**.** *For any matrix*

$$A := \begin{bmatrix} \vec{a}_1 & \vec{a}_2 & \cdots & \vec{a}_n \end{bmatrix} \in \mathbb{R}^{m \times n}, \tag{64}$$

with left singular vectors $u_1, \ldots, u_n$, we have

$$\sum_{i=1}^{n} \| \mathcal{P}_{\text{span}(\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_k)} \, \vec{a}_i \|_2^2 \geq \sum_{i=1}^{n} \| \mathcal{P}_{\mathcal{S}} \, \vec{a}_i \|_2^2, \tag{65}$$

for any subspace $\mathcal{S}$ of dimension $k \leq \min \{m, n\}$.

Proof. Note that

$$\sum_{i=1}^{n} \| \mathcal{P}_{\text{span}(\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_k)} \, \vec{a}_i \|_2^2 = \sum_{i=1}^{n} \sum_{j=1}^{k} \langle \vec{u}_j, \vec{a}_i \rangle^2 \tag{66}$$

$$= \sum_{j=1}^{k} \| A^T \vec{u}_j \|_2^2. \tag{67}$$

We prove the result by induction on k. The base case $k = 1$ follows immediately from (18). To complete the proof we show that if the result is true for $k - 1 \geq 1$ (the induction hypothesis) then it also holds for k. Let $\mathcal{S}$ be an arbitrary subspace of dimension k. The intersection of $\mathcal{S}$ and the orthogonal complement to the span of $\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_{k-1}$ contains a nonzero vector $\vec{b}$ due to the following lemma.

Lemma 3.10 (Proof in Section A)**.** *In a vector space of dimension n, the intersection of two subspaces with dimensions d_1 and d_2 such that $d_1 + d_2 > n$ has dimension at least one.*

We choose an orthonormal basis $\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_k$ for $\mathcal{S}$ such that $\vec{b}_k := \vec{b}$ is orthogonal to $\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_{k-1}$

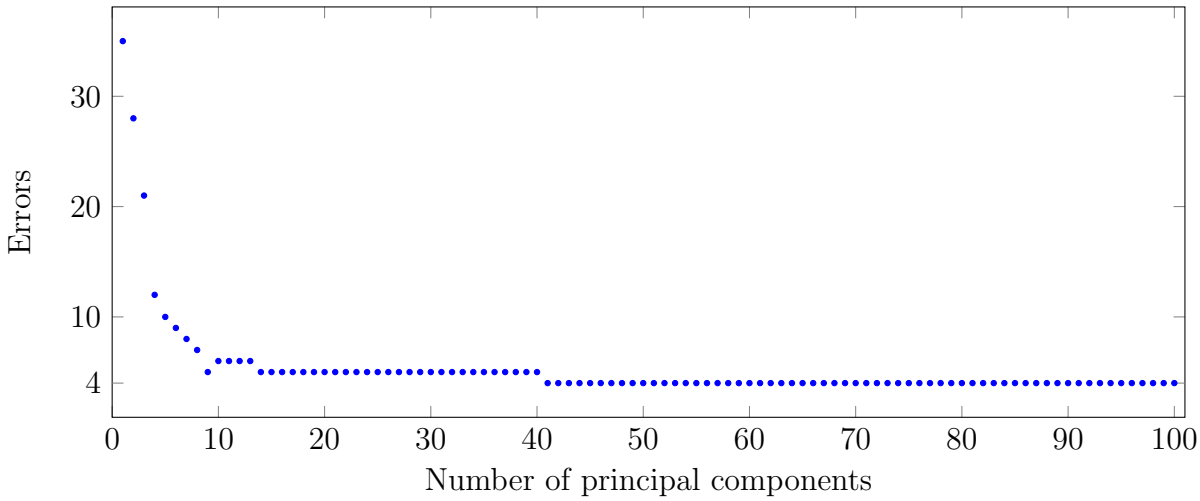


Figure 10: Errors for nearest-neighbor classification combined with PCA-based dimensionality reduction for different dimensions.

(we can construct such a basis by Gram-Schmidt, starting with $\vec{b}$). By the induction hypothesis,

$$\sum_{i=1}^{k-1} \|A^T \vec{u}_i\|_2^2 = \sum_{i=1}^{n} \|\mathcal{P}_{\text{span}(\vec{u}_1,\vec{u}_2,\ldots,\vec{u}_{k-1})}\, \vec{a}_i\|_2^2 \tag{68}$$

$$\geq \sum_{i=1}^{n} \|\mathcal{P}_{\text{span}\left(\vec{b}_1,\vec{b}_2,\ldots,\vec{b}_{k-1}\right)}\, \vec{a}_i\|_2^2 \tag{69}$$

$$= \sum_{i=1}^{k-1} \|A^T \vec{b}_i\|_2^2. \tag{70}$$

By (19)

$$\|A^T \vec{u}_k\|_2^2 \geq \|A^T \vec{b}_k\|_2^2. \tag{71}$$

Combining (70) and (71) we conclude

$$\sum_{i=1}^{n} \|\mathcal{P}_{\text{span}(\vec{u}_1,\vec{u}_2,\ldots,\vec{u}_k)}\, \vec{a}_i\|_2^2 = \sum_{i=1}^{k} \|A^T \vec{u}_i\|_2^2 \tag{72}$$

$$\geq \sum_{i=1}^{k} \|A^T \vec{b}_i\|_2^2 \tag{73}$$

$$= \sum_{i=1}^{n} \|\mathcal{P}_{\mathcal{S}}\, \vec{a}_i\|_2^2. \tag{74}$$

□

Example 3.11 (Nearest neighbors in principal-component space)**.** The nearest-neighbor algorithm is a classical method to perform classification. Assume that we have access to a training

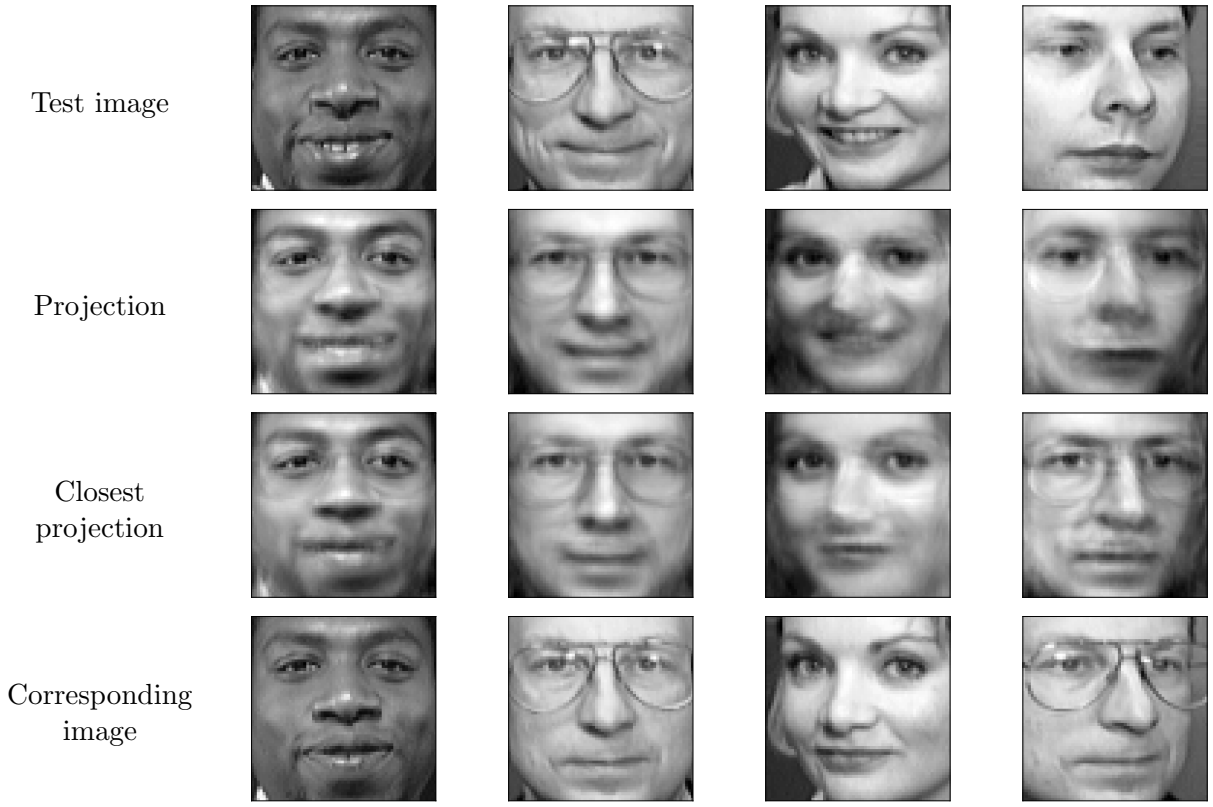


Figure 11: Results of nearest-neighbor classification combined with PCA-based dimensionality reduction of order 41 for four of the people in Example 3.11. The assignments of the first three examples are correct, but the fourth is wrong.

set of n pairs of data encoded as vectors in $\mathbb{R}^p$ along with their corresponding labels: $\{\vec{x}_1, l_1\}, \ldots, \{\vec{x}_n, l_n\}$. To classify a new data point $\vec{y}$ we find the closest element of the training set,

$$i^* := \arg \min_{1 \leq i \leq n} ||\vec{y} - \vec{x}_i||_2 , \tag{75}$$

and assign the corresponding label l_{i^*} to $\vec{y}$. This requires computing n distances in a p-dimensional space. The computational cost is $\mathcal{O}(np)$, so if we need to classify m points the total cost is $\mathcal{O}(mnp)$. To alleviate the cost, we can perform PCA and apply the algorithm in a space of reduced dimensionality k. The computational cost is:

- $\mathcal{O}(np \min\{n, p\})$ to compute the principal directions from the training data.
- knp operations to project the training data onto the first k principal directions.
- kmp operations to project each point in the test set onto the first k principal directions.
- kmn to perform nearest-neighbor classification in the lower-dimensional space.

If we have to classify a large number of points (i.e. $m \gg \max\{p, n\}$) the computational cost is reduced by operating in the lower-dimensional space.

In this example we explore this idea using the faces dataset from Example 3.8. The training set consists of 360 64×64 images taken from 40 different subjects (9 per subject). The test set consists of an image of each subject, which is different from the ones in the training set. We apply the nearest-neighbor algorithm to classify the faces in the test set, modeling each image as a vector in $\mathbb{R}^{4096}$ and using the distance induced by the ℓ_2 norm. The algorithm classifies 36 of the 40 subjects correctly.

Figure 10 shows the accuracy of the algorithm when we compute the distance using k principal components, obtained by applying PCA to the training set, for different values of k. The accuracy increases with the dimension at which the algorithm operates. Interestingly, this is not necessarily always the case because projections may actually be helpful for tasks such as classification (for example, factoring out small shifts and deformations). The same precision as in the ambient dimension (4 errors out of 40 test images) is achieved using just $k = 41$ principal components (in this example $n = 360$ and $p = 4096$). Figure 11 shows some examples of the projected data represented in the original p-dimensional space along with their nearest neighbors in the k-dimensional space. △

Example 3.12 (Dimensionality reduction for visualization)**.** Dimensionality reduction is very useful for visualization. When visualizing data the objective is usually to project it down to 2D or 3D in a way that preserves its structure as much as possible. In this example, we consider a data set where each data point corresponds to a seed with seven features: area, perimeter, compactness, length of kernel, width of kernel, asymmetry coefficient and length of kernel groove. The seeds belong to three different varieties of wheat: Kama, Rosa and Canadian.[3]

Figure 12 shows the data represented by the first two and the last two principal components. In the latter case, there is almost no discernible variation. As predicted by our theoretical analysis

[3]The data can be found at https://archive.ics.uci.edu/ml/datasets/seeds.

Figure 12: Projection of 7-dimensional vectors describing different wheat seeds onto the first two (left) and the last two (right) principal dimensions of the data set. Each color represents a variety of wheat.

of PCA, the structure in the data is much better conserved by the two first principal components, which allow to clearly visualize the difference between the three types of seeds. Note that using the first principal components only ensures that we preserve as much variation as possible; this does not necessarily mean that these are the best low-dimensional features for tasks such as clustering or classification. △

4 Linear regression

4.1 Least squares

As described in Section 1.2, in linear regression we approximate the response as a linear combination of certain predefined features. The linear model is learned from a training set with n examples. The goal is to learn a vector of coefficients $\vec{\beta} \in \mathbb{R}^p$ such that

$$\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \cdots \\ y^{(n)} \end{bmatrix} \approx \begin{bmatrix} \vec{x}^{(1)}[1] & \vec{x}^{(1)}[2] & \cdots & \vec{x}^{(1)}[p] \\ \vec{x}^{(2)}[1] & \vec{x}^{(2)}[2] & \cdots & \vec{x}^{(2)}[p] \\ \cdots & \cdots & \cdots & \cdots \\ \vec{x}^{(n)}[1] & \vec{x}^{(n)}[2] & \cdots & \vec{x}^{(n)}[p] \end{bmatrix} \begin{bmatrix} \vec{\beta}[1] \\ \vec{\beta}[2] \\ \cdots \\ \vec{\beta}[p] \end{bmatrix}. \tag{76}$$

or, more succinctly,

$$\vec{y} \approx X\vec{\beta} \tag{77}$$

where $\vec{y} \in \mathbb{R}^n$ contains all the response values in the training set and X is a $p \times n$ matrix containing the features. For simplicity of exposition, we have omitted the intercept β_0 in Eq.(3); we explain how to incorporate it below.

A reasonable approach to estimate the coefficient vector is to minimize the fitting error of the linear model on the training set. This requires choosing a metric to evaluate the fit. By far, the

most popular metric is the sum of the squares of the fitting error,

$$\sum_{i=1}^{n} \left(y^{(i)} - \langle \vec{x}^{(i)}, \vec{\beta} \rangle \right)^2 = \| \vec{y} - X\vec{\beta} \|_2^2. \tag{78}$$

The least-squares coefficient vector $\vec{\beta}_{\text{LS}}$ is defined as the minimizer of the least-squares fit,

$$\vec{\beta}_{\text{LS}} := \arg\min_{\vec{\beta}} \| \vec{y} - X\vec{\beta} \|_2. \tag{79}$$

If the feature vectors are linearly independent (i.e. X is full rank) and we have at least as many data as coefficients ($n \leq p$), the minimizer is well defined and has a closed-form solution.

Theorem 4.1. *If X is full rank and $n \geq p$, for any $\vec{y} \in \mathbb{R}^n$ we have*

$$\vec{\beta}_{\text{LS}} := \arg\min_{\vec{\beta}} \quad \left|\left| \vec{y} - X\vec{\beta} \right|\right|_2 \tag{80}$$

$$= VS^{-1}U^T\vec{y} \tag{81}$$

$$= \left(X^TX\right)^{-1} X^T\vec{y}, \tag{82}$$

where USV^T is the SVD of X.

Proof. We consider the decomposition of $\vec{y}$ into its orthogonal projection $UU^T\vec{y}$ onto the column space of X $\text{col}(X)$ and its projection $\left(I - UU^T\right)\vec{y}$ onto the orthogonal complement of $\text{col}(X)$. $X\vec{\beta}$ belongs to $\text{col}(X)$ for any $\vec{\beta}$ and is consequently orthogonal to $\left(I - UU^T\right)\vec{y}$ (as is $UU^T\vec{y}$), so that

$$\arg\min_{\vec{\beta}} \left|\left| \vec{y} - X\vec{\beta} \right|\right|_2^2 = \arg\min_{\vec{\beta}} \left|\left| \left(I - UU^T\right)\vec{y} \right|\right|_2^2 + \left|\left| UU^T\vec{y} - X\vec{\beta} \right|\right|_2^2 \tag{83}$$

$$= \arg\min_{\vec{\beta}} \left|\left| UU^T\vec{y} - X\vec{\beta} \right|\right|_2^2 \tag{84}$$

$$= \arg\min_{\vec{\beta}} \left|\left| UU^T\vec{y} - USV^T\vec{\beta} \right|\right|_2^2. \tag{85}$$

Since U has orthonormal columns, for any vector $\vec{v} \in \mathbb{R}^p$ $||U\vec{v}||_2 = ||\vec{v}||_2$, which implies

$$\arg\min_{\vec{\beta}} \left|\left| \vec{y} - X\vec{\beta} \right|\right|_2^2 = \arg\min_{\vec{\beta}} \left|\left| U^T\vec{y} - SV^T\vec{\beta} \right|\right|_2^2 \tag{86}$$

If X is full rank and $n \geq p$, then SV^T is square and full rank. It therefore has a unique inverse, which is equal to VS^{-1}. As a result $VS^{-1}U^T\vec{y} = \left(X^TX\right)^{-1} X^T\vec{y}$ is the unique solution to the optimization problem in Eq. (86) (it is the only vector that achieves a value of zero for the cost function). □

In practice, large-scale least-squares problems are not solved by using the closed-form solution, due to the computational cost of inverting the matrix X^TX, but rather by applying iterative optimization methods such as conjugate gradients [1].

Up to now we have ignored the intercept β_0 in Eq. (3). The reason is that the intercept is just equal to the average of the response values in the training set, as long as the feature vectors are all centered by subtracting their averages. A formal proof of this can be found in Lemma B.1 of the appendix. Centering is a common preprocessing step when fitting a linear-regression model. In addition, the response and the features are usually normalized, by dividing them by their sample standard deviations. This ensures that all the variables have the same order of magnitude, which makes the regression coefficients invariant to changes of units and consequently more interpretable.

Example 4.2 (Linear model for GDP)**.** We consider the problem of building a linear model to predict the gross domestic product (GDP) of a state in the US from its population and unemployment rate. We have available the following data:

	GDP (USD millions)	Population	Unemployment rate (%)
North Dakota	52,089	757,952	2.4
Alabama	204,861	4,863,300	3.8
Mississippi	107,680	2,988,726	5.2
Arkansas	120,689	2,988,248	3.5
Kansas	153,258	2,907,289	3.8
Georgia	525,360	10,310,371	4.5
Iowa	178,766	3,134,693	3.2
West Virginia	73,374	1,831,102	5.1
Kentucky	197,043	4,436,974	5.2
Tennessee	???	6,651,194	3.0

In this example, the GDP is the response, whereas the population and the unemployment rate are the features. Our goal is to fit a linear model to the data so that we can predict the GDP of Tennessee, using a linear model. We begin by centering and normalizing the data. The averages of the response and of the features are

$$\text{av}\left(\vec{y}\right) = 179,236, \qquad \text{av}\left(X\right) = \begin{bmatrix} 3,802,073 & 4.1 \end{bmatrix}. \tag{87}$$

The sample standard deviations are

$$\text{std}\left(\vec{y}\right) = 396{,}701, \qquad \text{std}\left(X\right) = \begin{bmatrix} 7,720,656 & 2.80 \end{bmatrix}, \tag{88}$$

recall that the sample standard deviation is just the square root of the sample variance. For any vector $\vec{a} \in \mathbb{R}^m$,

$$\text{std}\left(\vec{a}\right) := \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}\left(\vec{a}[i] - \text{av}\left(\vec{a}\right)\right)^2}. \tag{89}$$

We subtract the average and divide by the standard deviations so that both the response and the

features are centered and on the same scale,

$$\vec{y} = \begin{bmatrix} -0.321 \\ 0.065 \\ -0.180 \\ -0.148 \\ -0.065 \\ 0.872 \\ -0.001 \\ -0.267 \\ 0.045 \end{bmatrix}, \qquad X = \begin{bmatrix} -0.394 & -0.600 \\ 0.137 & -0.099 \\ -0.105 & 0.401 \\ -0.105 & -0.207 \\ -0.116 & -0.099 \\ 0.843 & 0.151 \\ -0.086 & -0.314 \\ -0.255 & 0.366 \\ 0.082 & 0.401 \end{bmatrix}. \tag{90}$$

The least-squares estimate for the regression coefficients in the linear GDP model is equal to

$$\vec{\beta}_{\text{LS}} = \begin{bmatrix} 1.019 \\ -0.111 \end{bmatrix}. \tag{91}$$

The GDP seems to be proportional to the population and inversely proportional to the unemployment rate. Note that the coefficients are only comparable because everything is normalized (otherwise considering the unemployment as a percentage would change the result!).

To obtain a linear model for the GDP we need to rescale according to the standard deviations (88) and recenter using the averages (87), i.e.

$$y_{\text{est}} := \text{av}\,(\vec{y}) + \text{std}\,(\vec{y})\,\langle \vec{x}_{\text{norm}}, \vec{\beta}_{\text{LS}} \rangle \tag{92}$$

where $\vec{x}_{\text{norm}}$ is centered using $\text{av}\,(X)$ and normalized using $\text{std}\,(X)$. This yields the following values for each state and, finally, our prediction for Tennessee:

	GDP	Estimate
North Dakota	52,089	46,241
Alabama	204,861	239,165
Mississippi	107,680	119,005
Arkansas	120,689	145,712
Kansas	153,258	136,756
Georgia	525,360	513,343
Iowa	178,766	158,097
West Virginia	73,374	59,969
Kentucky	197,043	194,829
Tennessee	328,770	345,352

△

Example 4.3 (Temperature prediction via linear regression)**.** We consider a dataset of hourly temperatures measured at weather stations all over the United States[4]. Our goal is to design a

[4]The data are available at http://www1.ncdc.noaa.gov/pub/data/uscrn/products

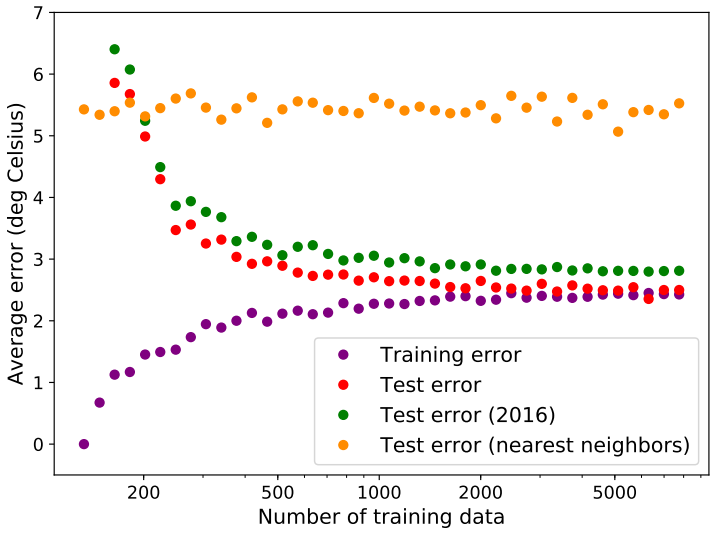


Figure 13: Performance of the least-squares estimator on the temperature data described in Example 4.3. The graph shows the RMSE achieved by the model on the training and test sets, and on the 2016 data, for different number of training data and compares it to the RMSE of a nearest-neighbor approach.

model that can be used to estimate the temperature in Yosemite Valley from the temperatures of 133 other stations, in case the sensor in Yosemite fails. We perform estimation by fitting a linear model where the response is the temperature in Yosemite and the features are the rest of the temperatures ($p = 133$). We use 10^3 measurements from 2015 as a test set, and train a linear model using a variable number of training data also from 2015 but disjoint from the test data. In addition, we test the linear model on data from 2016. Figure 13 shows the results. With enough data, the linear model achieves an error of roughly 2.5°C on the test data, and 2.8°C on the 2016 data. The linear model outperforms nearest-neighbor estimation, which uses the station that best predicts the temperature in Yosemite in the training set. △

4.2 Geometric interpretation

The least-squares estimator can be derived from a purely geometric viewpoint. The goal of the linear-regression estimate is to approximate the response vector $\vec{y}$ by a linear combination of the corresponding features. In other words, we want to find the vector in the column space of the feature matrix X that is closest to $\vec{y}$. By definition, that vector is the orthogonal projection of $\vec{y}$ onto $\text{col}(X)$. This is exactly equal to the least-squares fit $X\vec{\beta}_{\text{LS}}$ as established by the following corollary. Figure 14 illustrates this geometric perspective.

Lemma 4.4. *If X is full rank and $n \geq p$, the least-squares approximation of any vector $\vec{y} \in \mathbb{R}^n$ obtained by solving problem (79)*

$$\vec{y}_{\text{LS}} = X\vec{\beta}_{\text{LS}} \tag{93}$$

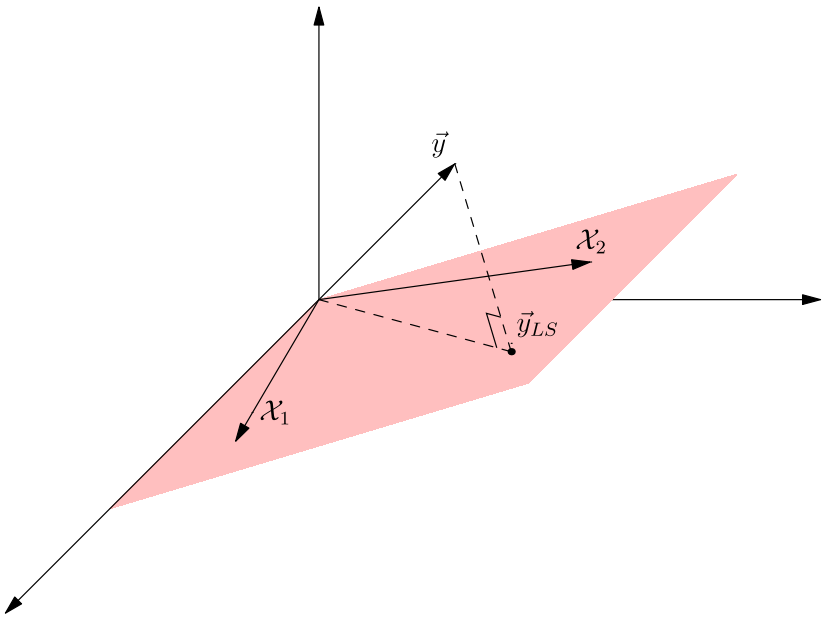


Figure 14: Illustration of Lemma 4.4. The least-squares solution is the orthogonal projection of the data onto the subspace spanned by the columns of X, denoted by X_1 and X_2.

is equal to the orthogonal projection of $\vec{y}$ onto the column space of X.

Proof. Let USV^T be the SVD of X. By Theorem 4.1

$$X\vec{\beta}_{\text{LS}} = XVS^{-1}U^T\vec{y} \tag{94}$$
$$= USV^TVS^{-1}U^T\vec{y} \tag{95}$$
$$= UU^T\vec{y}. \tag{96}$$

Since the columns of U form an orthonormal basis for the column space of X the proof is complete. □

4.3 Probabilistic interpretation

The least squares estimator can also be derived from a probabilistic perspective. Let $\mathbf{y}$ be a scalar random variable with zero mean and $\vec{\mathbf{x}}$ a p-dimensional random vector. Our goal is to estimate $\mathbf{y}$ as a linear combination of the entries of $\vec{\mathbf{x}}$. A reasonable metric to quantify the accuracy of the linear estimate is the mean square error (MSE), defined as

$$\text{MSE}(\vec{\beta}) := \text{E}((\mathbf{y} - \vec{\mathbf{x}}^T\vec{\beta})^2) = \text{E}\left(\mathbf{y}^2\right) - 2\text{E}\left(\mathbf{y}\vec{\mathbf{x}}\right)^T\vec{\beta} + \vec{\beta}^T\text{E}\left(\vec{\mathbf{x}}\vec{\mathbf{x}}^T\right)\vec{\beta} \tag{97}$$
$$= \text{Var}\left(\mathbf{y}\right) - 2\Sigma_{\mathbf{y}\vec{\mathbf{x}}}^T\vec{\beta} + \vec{\beta}^T\Sigma_{\vec{\mathbf{x}}}\vec{\beta}, \tag{98}$$

where $\Sigma_{\mathbf{y}\vec{\mathbf{x}}}$ is the cross-covariance vector that contains the covariance between $\mathbf{y}$ and each entry of $\mathbf{x}$, and $\Sigma_{\vec{\mathbf{x}}}$ is the covariance matrix of $\mathbf{x}$. If $\Sigma_{\vec{\mathbf{x}}}$ is full rank the MSE is a strictly convex cost

function, which can be minimized by setting the gradient

$$\nabla \operatorname{MSE}(\vec{\beta}) = 2\Sigma_{\vec{\mathbf{x}}}\vec{\beta} - 2\Sigma_{\mathbf{y}\vec{\mathbf{x}}} \tag{99}$$

to zero. This yields the minimum MSE estimate of the linear coefficients,

$$\vec{\beta}_{\text{MMSE}} := \Sigma_{\vec{\mathbf{x}}}^{-1}\Sigma_{\mathbf{y}\vec{\mathbf{x}}}. \tag{100}$$

Now, for a linear-regression problem with response vector $\vec{y}$ and feature matrix X, we can interpret the rows of X as samples of $\vec{\mathbf{x}}$ and the entries of $\vec{y}$ as samples from $\mathbf{y}$. If the number of data n are large then the covariance matrix is well approximated by the sample covariance matrix,

$$\Sigma_{\vec{\mathbf{x}}} \approx \frac{1}{n}X^T X, \tag{101}$$

and the cross-covariance is well approximated by the sample cross-covariance, which contains the sample covariance between each feature and the response,

$$\Sigma_{\mathbf{y}\vec{\mathbf{x}}} \approx \frac{1}{n}X\vec{y}. \tag{102}$$

Using the sample moment matrices, we can approximate the coefficient vector by

$$\vec{\beta}_{\text{MMSE}} \approx \left(X^T X\right)^{-1} X\vec{y}. \tag{103}$$

This is exactly the least-square estimator.

4.4 Training error of the least-squares estimator

A friend of yours tells you:

I found a cool way to predict the daily temperature in New York: It's just a linear combination of the temperature in every other state. I fit the model on data from the last month and a half and it's almost perfect!

You check the training error of their model and it is indeed surprisingly low. What is going on here?

To analyze the training error of the least-squares estimator, let us assume that the data are generated by a linear model, perturbed by an additive term that accounts for model inaccuracy and noisy fluctuations. The model is parametrized by a vector of *true* linear coefficients $\vec{\beta}_{\text{true}} \in \mathbb{R}^p$. The training data $\vec{y}_{\text{train}} \in \mathbb{R}^n$ are given by

$$\vec{y}_{\text{train}} := X_{\text{train}}\vec{\beta}_{\text{true}} + \vec{z}_{\text{train}}, \tag{104}$$

where $X_{\text{train}} \in \mathbb{R}^{n\times p}$ contains the corresponding n p-dimensional feature vectors and $\vec{z}_{\text{train}} \in \mathbb{R}^n$ is the noise. The regression model is learned from this training set by minimizing the least-squares fit

$$\vec{\beta}_{\text{LS}} := \arg\min_{\vec{\beta}} \ \|\vec{y}_{\text{train}} - X_{\text{train}}\vec{\beta}\|_2. \tag{105}$$

To quantify the performance of the estimator, we assume that the noise follows a Gaussian iid distribution with zero mean and variance σ^2. Under this assumption, the least-squares estimator is also the maximum likelihood estimator.

Lemma 4.5. *If the training data are interpreted as a realization of the random vector*

$$\vec{\mathbf{y}}_{\text{train}} := X_{\text{train}} \vec{\beta}_{\text{true}} + \vec{\mathbf{z}}_{\text{train}}, \tag{106}$$

where the noise is an iid Gaussian random vector with mean zero, the maximum-likelihood (ML) estimate of the coefficients equals the least-squares estimate (105).

Proof. For ease of notation, we set $X := X_{\text{train}}$ and $\vec{y} := \vec{y}_{\text{train}}$. The likelihood is the probability density function of $\vec{\mathbf{y}}_{\text{train}}$ evaluated at the observed data $\vec{y}$ and interpreted as a function of the coefficient vector $\vec{\beta}$,

$$\mathcal{L}_{\vec{y}}(\vec{\beta}) = \frac{1}{\sqrt{(2\pi\sigma^2)^n}} \exp\left(-\frac{1}{2\sigma^2} \left|\left| \vec{y} - X\vec{\beta} \right|\right|_2^2 \right). \tag{107}$$

To find the ML estimate, we maximize the log likelihood

$$\vec{\beta}_{\text{ML}} = \arg\max_{\vec{\beta}} \mathcal{L}_{\vec{y}}(\vec{\beta}) \tag{108}$$

$$= \arg\max_{\vec{\beta}} \log \mathcal{L}_{\vec{y}}(\vec{\beta}) \tag{109}$$

$$= \arg\min_{\vec{\beta}} \left|\left| \vec{y} - X\vec{\beta} \right|\right|_2^2. \tag{110}$$

□

Theorem 4.1 makes it possible to obtain a closed-form expression for the error in the estimate of the linear coefficients.

Theorem 4.6. *If the training data follow the linear model* (104) *and* X_{train} *is full rank, the error of the least-squares estimate equals*

$$\vec{\beta}_{\text{LS}} - \vec{\beta}_{\text{true}} = \left(X_{\text{train}}^T X_{\text{train}}\right)^{-1} X_{\text{train}}^T \vec{z}_{\text{train}}. \tag{111}$$

Proof. By Theorem 4.1

$$\vec{\beta}_{\text{LS}} - \vec{\beta}_{\text{true}} = \left(X_{\text{train}}^T X_{\text{train}}\right)^{-1} X_{\text{train}}^T \vec{y}_{\text{train}} - \vec{\beta}_{\text{true}} \tag{112}$$

$$= \vec{\beta}_{\text{true}} + \left(X_{\text{train}}^T X_{\text{train}}\right)^{-1} X_{\text{train}}^T \vec{z}_{\text{train}} - \vec{\beta}_{\text{true}} \tag{113}$$

$$= \left(X_{\text{train}}^T X_{\text{train}}\right)^{-1} X_{\text{train}}^T \vec{z}_{\text{train}}. \tag{114}$$

□

The training error has an intuitive geometric interpretation.

Theorem 4.7. *If the training data follow the additive model* (104) *and* X_{train} *is full rank, the training error is the projection of the noise onto the orthogonal complement of the column space of* X_{train}.

Proof. By Lemma 4.4

$$\vec{y}_{\text{train}} - \vec{y}_{\text{LS}} = \vec{y}_{\text{train}} - \mathcal{P}_{\text{col}(X_{\text{train}})}\, \vec{y}_{\text{train}} \tag{115}$$

$$= X_{\text{train}}\vec{\beta}_{\text{true}} + \vec{z}_{\text{train}} - \mathcal{P}_{\text{col}(X_{\text{train}})}\, (X_{\text{train}}\vec{\beta}_{\text{true}} + \vec{z}_{\text{train}}) \tag{116}$$

$$= X_{\text{train}}\vec{\beta}_{\text{true}} + \vec{z}_{\text{train}} - X_{\text{train}}\vec{\beta}_{\text{true}} - \mathcal{P}_{\text{col}(X_{\text{train}})}\, \vec{z}_{\text{train}} \tag{117}$$

$$= \mathcal{P}_{\text{col}(X_{\text{train}})^{\perp}}\, \vec{z}_{\text{train}}. \tag{118}$$

□

To leverage this result under the probabilistic assumption on the noise, we invoke the following theorem, which we will prove later on in the course.

Theorem 4.8. *Let $\mathcal{S}$ be a k-dimensional subspace of $\mathbb{R}^n$ and $\vec{\mathbf{z}} \in \mathbb{R}^n$ a vector of iid Gaussian noise with variance σ^2. For any $\epsilon \in (0, 1)$*

$$\sigma\sqrt{k\,(1-\epsilon)} \leq ||\mathcal{P}_{\mathcal{S}}\, \vec{\mathbf{z}}||_2 \leq \sigma\sqrt{k\,(1+\epsilon)} \tag{119}$$

with probability at least $1 - 2\exp\left(-k\epsilon^2/8\right)$.

In words, the ℓ_2 norm of the projection of an n-dimensional iid Gaussian vector with variance σ^2 onto a k-dimensional subspace concentrates around $\sigma\sqrt{k}$.

Corollary 4.9. *If the training data follow the additive model (104) and the noise is iid Gaussian with variance σ^2, then the error of the least-squares estimator,*

$$\textit{Training RMSE} := \sqrt{\frac{||\vec{y}_{\text{train}} - \vec{y}_{\text{LS}}||_2^2}{n}}, \tag{120}$$

concentrates with high probability around $\sigma\sqrt{1 - p/n}$.

Proof. The column space of X_{train} has dimension p, so its orthogonal complement has dimension $n - p$. The result then follows from Theorem 4.8. □

Figure 15 shows that this analysis accurately characterizes the training error for an example with real data.

We can now explain your friend's results. When $p \approx n$, error of the least-squares estimator is indeed very low. The bad news is that this does not imply that the true vector of coefficients has been recovered. In fact, the error achieved by $\vec{\beta}_{\text{true}}$ is significantly larger! Indeed, setting $k := n$ in Theorem 4.8 we have

$$\text{Ideal training RMSE} := \sqrt{\frac{\left|\left|\vec{y}_{\text{train}} - X_{\text{train}}\vec{\beta}_{\text{true}}\right|\right|_2^2}{n}} \tag{121}$$

$$= \sqrt{\frac{||\vec{z}_{\text{train}}||_2^2}{n}} \approx \sigma. \tag{122}$$

If an estimator achieves an error of less than σ it must be *overfitting* the training noise, which will result in a higher generalization error on held-out data. This is the case for the least-squares estimator when the number of data is small with respect to the number of features. As n grows, the training error increases, converging to σ when n is sufficiently large with respect to p. This does not necessarily imply good generalization, but it is a good sign.

4.5 Test error of the least-squares estimator

The test error of an estimator quantifies its performance on held-out data, which have not been used to fit the model. In the case of the additive model introduced in the previous section, a test example is given by

$$y_{\text{test}} := \langle \vec{x}_{\text{test}}, \vec{\beta}_{\text{true}} \rangle + z_{\text{test}}. \tag{123}$$

The vector of coefficients is the same as in the training set, but the feature vector $\vec{x}_{\text{test}} \in \mathbb{R}^p$ and the noise $z_{\text{test}} \in \mathbb{R}$ are different. The estimate produced by our linear-regression model equals

$$y_{\text{LS}} := \langle \vec{x}_{\text{test}}, \vec{\beta}_{\text{LS}} \rangle, \tag{124}$$

where $\vec{\beta}_{\text{LS}}$ is the least-squares coefficient vector obtained from the training data in Eq. (105).

In order to quantify the test error *on average*, we consider the mean square error (MSE) when the test features are realizations of a zero-mean random vector $\vec{\mathbf{x}}_{\text{test}}$ with the same distribution as the training data[5]. In addition, we assume iid Gaussian noise with the same variance σ^2 as the training noise. If $\vec{\mathbf{x}}_{\text{test}}$ is generated from the same distribution as the features in the training set, the sample covariance matrix of the training data will approximate the covariance matrix of $\vec{\mathbf{x}}_{\text{test}}$ for large enough n. The following theorem provides an estimate of the MSE in such a regime.

Theorem 4.10. *Let the training and test data follow the linear model in Eq. (104) and Eq. (123), and the training and test noise be iid Gaussian with variance σ^2. If the training and test noise, and the feature vector are all independent, then the error of the least-squares estimator in Eq. (124) satisfies*

$$\textit{Test RMSE} := \sqrt{\text{E}\left((\boldsymbol{y}_{\text{test}} - \boldsymbol{y}_{\text{LS}})^2 \right)} \approx \sigma \sqrt{1 + \frac{p}{n}}, \tag{125}$$

as long as

$$\text{E}\left(\vec{\mathbf{x}}_{\text{test}} \vec{\mathbf{x}}_{\text{test}}^T \right) \approx \frac{1}{n} X_{\text{train}}^T X_{\text{train}}. \tag{126}$$

Proof. By Eq. (111) the test error equals

$$\boldsymbol{y}_{\text{test}} - \boldsymbol{y}_{\text{LS}} = \langle \vec{\mathbf{x}}_{\text{test}}, \vec{\beta}_{\text{true}} - \vec{\boldsymbol{\beta}}_{\text{LS}} \rangle + \boldsymbol{z}_{\text{test}} \tag{127}$$

$$= -\langle \vec{\mathbf{x}}_{\text{test}}, \left(X_{\text{train}}^T X_{\text{train}} \right)^{-1} X_{\text{train}}^T \vec{\mathbf{z}}_{\text{train}} \rangle + \boldsymbol{z}_{\text{test}}. \tag{128}$$

[5] We assume that the distribution has zero mean to simplify the exposition. For sufficiently large training sets, the estimate of the mean from the training set will be sufficient to center the training and test features accurately.

Since $\vec{\mathbf{x}}_{\text{test}}$ and $\vec{\mathbf{z}}_{\text{train}}$ are independent and zero mean, and $\boldsymbol{z}_{\text{test}}$ is zero mean, we can decompose the MSE into

$$\mathrm{E}\left((\boldsymbol{y}_{\text{test}} - \boldsymbol{y}_{\text{LS}})^2\right) = \mathrm{E}\left(\langle \vec{\mathbf{x}}_{\text{test}}, \left(X_{\text{train}}^T X_{\text{train}}\right)^{-1} X_{\text{train}}^T \vec{\mathbf{z}}_{\text{train}} \rangle^2\right) + \mathrm{E}\left(\boldsymbol{z}_{\text{test}}^2\right). \tag{129}$$

To alleviate notation, let $X^\dagger := \left(X_{\text{train}}^T X_{\text{train}}\right)^{-1} X_{\text{train}}^T$ denote the pseudoinverse of X_{train}. Rearranging the first term and applying iterated expectation combined with the assumption that $\vec{\mathbf{x}}_{\text{test}}$ and $\vec{\mathbf{z}}_{\text{train}}$ are independent, we obtain

$$\begin{aligned}
\mathrm{E}\left((\boldsymbol{y}_{\text{test}} - \boldsymbol{y}_{\text{LS}})^2\right) &= \mathrm{E}\left(\vec{\mathbf{x}}_{\text{test}}^T X^\dagger \vec{\mathbf{z}}_{\text{train}} \vec{\mathbf{z}}_{\text{train}}^T (X^\dagger)^T \vec{\mathbf{x}}_{\text{test}}\right) + \mathrm{E}\left(\boldsymbol{z}_{\text{test}}^2\right) && (130)\\
&= \mathrm{E}\left(\mathrm{E}\left(\vec{\mathbf{x}}_{\text{test}}^T X^\dagger \vec{\mathbf{z}}_{\text{train}} \vec{\mathbf{z}}_{\text{train}}^T (X^\dagger)^T \vec{\mathbf{x}}_{\text{test}} \,|\, \vec{\mathbf{x}}_{\text{test}}\right)\right) + \sigma^2 && (131)\\
&= \mathrm{E}\left(\vec{\mathbf{x}}_{\text{test}}^T X^\dagger \mathrm{E}\left(\vec{\mathbf{z}}_{\text{train}} \vec{\mathbf{z}}_{\text{train}}^T\right) (X^\dagger)^T \vec{\mathbf{x}}_{\text{test}}\right) + \sigma^2 && (132)\\
&= \sigma^2 \,\mathrm{E}\left(\vec{\mathbf{x}}_{\text{test}}^T X^\dagger (X^\dagger)^T \vec{\mathbf{x}}_{\text{test}}\right) + \sigma^2, && (133)
\end{aligned}$$

because $\vec{\mathbf{z}}_{\text{train}}$ is iid Gaussian noise, and its covariance matrix therefore equals the identity scaled by σ^2. Expanding the pseudoinverse,

$$\begin{aligned}
X^\dagger (X^\dagger)^T &= \left(X_{\text{train}}^T X_{\text{train}}\right)^{-1} X_{\text{train}}^T X_{\text{train}} \left(X_{\text{train}}^T X_{\text{train}}\right)^{-1} && (134)\\
&= \left(X_{\text{train}}^T X_{\text{train}}\right)^{-1}. && (135)
\end{aligned}$$

We now incorporate the trace operator to obtain

$$\begin{aligned}
\mathrm{E}\left(\vec{\mathbf{x}}_{\text{test}}^T \left(X_{\text{train}}^T X_{\text{train}}\right)^{-1} \vec{\mathbf{x}}_{\text{test}}\right) &= \mathrm{E}\left(\mathrm{tr}\left(\vec{\mathbf{x}}_{\text{test}}^T \left(X_{\text{train}}^T X_{\text{train}}\right)^{-1} \vec{\mathbf{x}}_{\text{test}}\right)\right) && (136)\\
&= \mathrm{E}\left(\mathrm{tr}\left(\left(X_{\text{train}}^T X_{\text{train}}\right)^{-1} \vec{\mathbf{x}}_{\text{test}} \vec{\mathbf{x}}_{\text{test}}^T\right)\right) && (137)\\
&= \mathrm{tr}\left(\left(X_{\text{train}}^T X_{\text{train}}\right)^{-1} \mathrm{E}\left(\vec{\mathbf{x}}_{\text{test}} \vec{\mathbf{x}}_{\text{test}}^T\right)\right) && (138)\\
&\approx \frac{1}{n} \mathrm{tr}\left(I\right) = \frac{p}{n}, && (139)
\end{aligned}$$

where the approximate equality follows from Eq. (126). □

Figure 15 shows that the approximation is accurate on a real data set, as long as n is large enough. However, for small n it can significantly underestimate the error.

4.6 Noise amplification

In this section we examine the least-squares estimator using the SVD to reason about the test error when the sample covariance matrix of the training data is not an accurate estimate of the true covariance matrix. The coefficient error incurred by the least-squares estimator can be expressed in terms of the SVD of the training matrix $X_{\text{train}} := USV^T$,

$$\begin{aligned}
\vec{\beta}_{\text{LS}} - \vec{\beta}_{\text{true}} &= \left(X_{\text{train}}^T X_{\text{train}}\right)^{-1} X_{\text{train}}^T \vec{z}_{\text{train}} && (140)\\
&= V_{\text{train}} S_{\text{train}}^{-1} U_{\text{train}}^T \vec{z}_{\text{train}} && (141)\\
&= \sum_{i=1}^{p} \frac{\langle \vec{u}_i, \vec{z}_{\text{train}} \rangle}{s_i} \vec{v}_i. && (142)
\end{aligned}$$

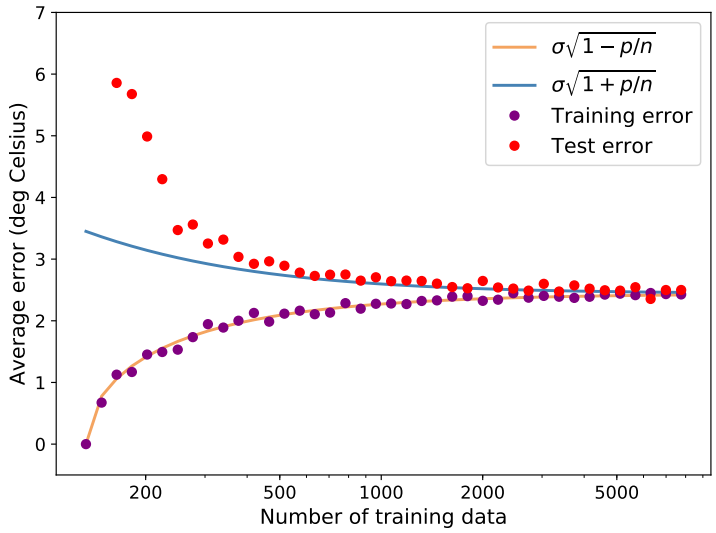


Figure 15: Comparison of the approximate bounds for the training and test error of the least-squares estimator with the actual errors on the temperature data described in Example 4.3.

This yields a decomposition of the test error into components weighted by the inverse of the singular values

$$y_{\text{test}} - y_{\text{LS}} = \langle \vec{x}_{\text{test}}, \vec{\beta}_{\text{true}} - \vec{\beta}_{\text{LS}} \rangle + z_{\text{test}} \tag{143}$$

$$= z_{\text{test}} - \sum_{i=1}^{p} \frac{\langle \vec{x}_{\text{test}}, \vec{v}_i \rangle \langle \vec{u}_i, \vec{z}_{\text{train}} \rangle}{s_i}. \tag{144}$$

The singular values of the training matrix can be very small when some of the features are correlated, as illustrated by Figure 16. This amplifies the contribution of the corresponding term in Eq. (144). When the sample covariance matrix is a good estimate for the covariance matrix of the test data, this is not a problem. Under the probabilistic assumptions described in the previous section,

$$\text{E}\left(\langle \vec{\mathbf{x}}_{\text{test}}, \vec{v}_i \rangle^2 \right) = \text{E}\left(\vec{v}_i^T \vec{\mathbf{x}}_{\text{test}} \vec{\mathbf{x}}_{\text{test}}^T \vec{v}_i \right) \tag{145}$$

$$= \vec{v}_i^T \text{E}\left(\vec{\mathbf{x}}_{\text{test}} \vec{\mathbf{x}}_{\text{test}}^T \right) \vec{v}_i \tag{146}$$

$$\approx \frac{1}{n} \vec{v}_i^T X_{\text{train}}^T X_{\text{train}} \vec{v}_i \tag{147}$$

$$= \frac{1}{n} \vec{v}_i^T V S^2 V^T \vec{v}_i \tag{148}$$

$$= \frac{s_i^2}{n}. \tag{149}$$

The standard deviation of the term $\langle \vec{\mathbf{x}}_{\text{test}}, \vec{v}_i \rangle$ in the numerator is approximately $s_i/\sqrt{n}$, which cancels out the effect of the small singular values. Geometrically, singular vectors corresponding to very small singular values are almost orthogonal to the test data.

When the sample covariance matrix is not an accurate estimate of the true covariance matrix, the alignment of $\vec{v}_i$ and $\vec{\mathbf{x}}_{\text{test}}$ is no longer proportional to s_i. To illustrate the consequences, let us

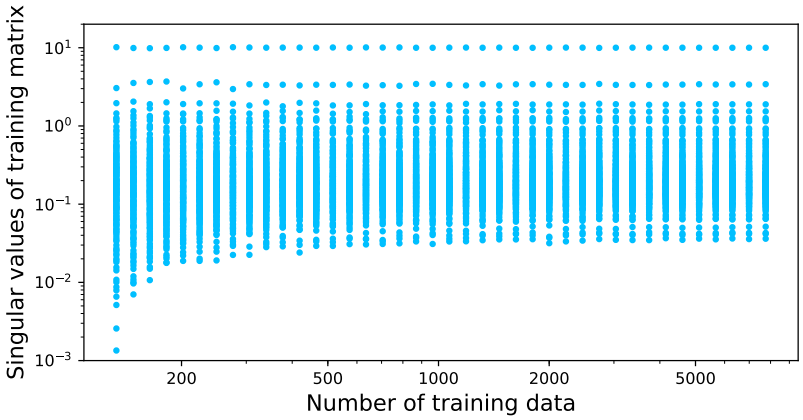


Figure 16: Singular values of the training matrix in Example 4.3 for different numbers of training data.

assume that $\vec{v}_i$ is oriented in a random direction with respect to $\vec{\mathbf{x}}_{\text{test}}$. In that case, $\mathrm{E}\left(\langle \vec{\mathbf{x}}_{\text{test}}, \vec{v}_i \rangle^2\right)$ approximately equals $||\vec{\mathbf{x}}_{\text{test}}||_2 / \sqrt{p} \approx 1$ (assuming that the standard deviation of each entry of $\vec{\mathbf{x}}_{\text{test}}$ equals one). As a result, the components in (144) are scaled proportionally to $1/s_i$, which can produce a dramatic amplification of the error is some of the singular values are very small. The effect is apparent in Figure 13 for small values of n. In the next section, we describe a technique designed to alleviate this issue.

4.7 Ridge regression

As we saw in the previous section, the least-squares estimator can suffer from significant noise amplification when the number of training data are small. This results in coefficients with very large amplitudes, which overfit the noise in the training set, as illustrated by the left image in Figure 17. A popular approach to avoid this problem is to add an extra term to the least-squares cost function, which penalizes the norm of the coefficient vector . The goal is to promote solutions that yield a good fit to the data using linear coefficients that are not too large. Modifying cost functions to favor structured solutions is called *regularization*. Least-squares regression combined with ℓ_2-norm regularization is called ridge regression in statistics and Tikhonov regularization in the inverse-problems literature.

Definition 4.11 (Ridge regression / Tikhonov regularization)**.** *For any $X \in \mathbb{R}^{n \times p}$ and $\vec{y} \in \mathbb{R}^p$ the ridge-regression estimator is the minimizer of the optimization problem*

$$\vec{\beta}_{\text{RR}} := \arg\min_{\vec{\beta}} \|\vec{y} - X\vec{\beta}\|_2^2 + \lambda \|\vec{\beta}\|_2^2, \tag{150}$$

where $\lambda > 0$ is a fixed regularization parameter.

As in the case of least-squares regression, the ridge-regression estimator has a closed form solution.

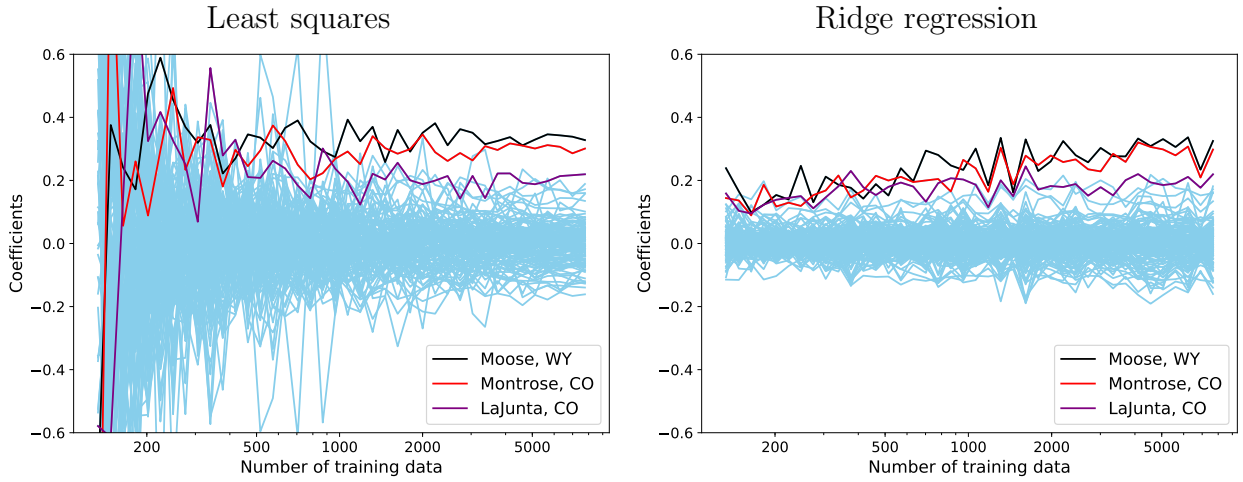


Figure 17: Coefficients of the least-squares (left) and ridge-regression (right) estimators computed from the data described in Example 4.3 for different values of training data. All coefficients are depicted in light blue except the three that have the largest magnitudes for large n, which correspond to the stations of Moose in Wyoming, and Montrose and La Junta in Colorado.

Theorem 4.12 (Ridge-regression estimate)**.** *For any $X \in R^{n \times p}$ and $\vec{y} \in \mathbb{R}^n$ we have*

$$\vec{\beta}_{\mathrm{RR}} := \left(X^T X + \lambda I\right)^{-1} X^T \vec{y}. \tag{151}$$

Proof. The cost function can be reformulated to equal a modified least-squares problem

$$\vec{\beta}_{\mathrm{RR}} = \arg\min_{\vec{\beta}} \left|\left| \begin{bmatrix} \vec{y} \\ 0 \end{bmatrix} - \begin{bmatrix} X \\ \sqrt{\lambda} I \end{bmatrix} \vec{\beta} \right|\right|_2^2. \tag{152}$$

By Theorem 4.1 the solution equals

$$\vec{\beta}_{\mathrm{RR}} := \left(\begin{bmatrix} X \\ \sqrt{\lambda} I \end{bmatrix}^T \begin{bmatrix} X \\ \sqrt{\lambda} I \end{bmatrix} \right)^{-1} \begin{bmatrix} X \\ \sqrt{\lambda} I \end{bmatrix}^T \begin{bmatrix} \vec{y} \\ 0 \end{bmatrix} \tag{153}$$

$$= \left(X^T X + \lambda I\right)^{-1} X^T \vec{y}. \tag{154}$$

□

Notice that when $\lambda \to 0$, $\vec{\beta}_{\mathrm{RR}}$ converges to the least-squares estimator. When $\lambda \to \infty$, $\vec{\beta}_{\mathrm{RR}}$ converges to zero.

The regularization parameter λ governs the trade-off between the term that promotes a good model fit on the training set and the term that controls the magnitudes of the coefficients. Ideally we would like to set the value of λ that achieves the best test error. However, we do not have access to the test set when training the regression model (and even if we did, one should never use test data for anything else other than evaluating test error!). We cannot use the training data to determine λ, since $\lambda = 0$ obviously achieves the minimum error on the training data. Instead, we use *validation* data, separate from the training and test data, to evaluate the error of the model for

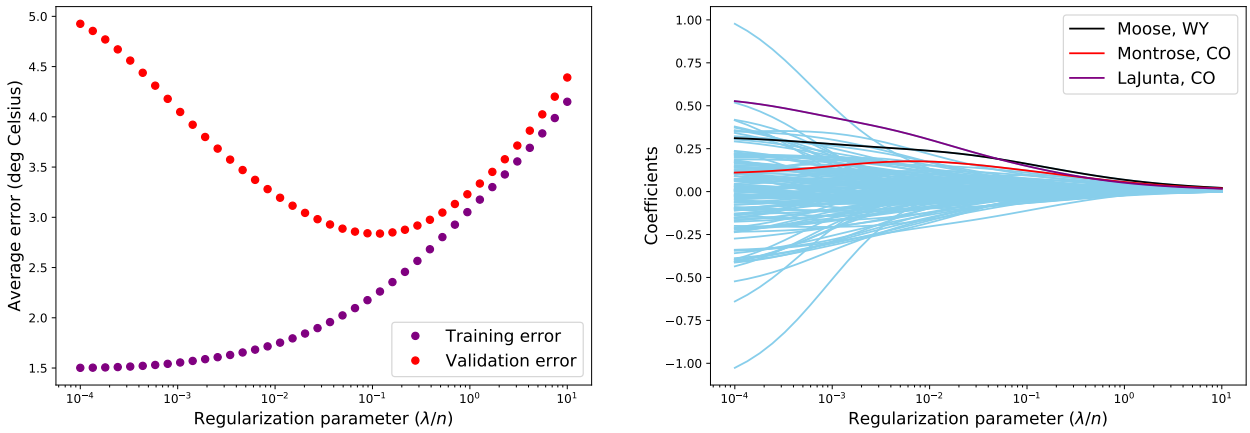


Figure 18: The left graph shows the training and validation errors of the ridge-regression estimator applied to the data described in Example 4.3 for different values of the regularization parameter λ. The number of training data is fixed to $n = 202$ training data. The right figure shows the values of the model coefficients for the different λ values. All coefficients are depicted in light blue except the three that have the largest magnitudes for large n, which correspond to the stations of Moose in Wyoming, and Montrose and La Junta in Colorado.

different values of λ and select the best value. This approach for setting model hyper parameters is commonly known as cross validation.

As shown in Figure 18, in the regime where the least-squares estimator overfits the training data, the ridge-regression estimator typically also overfits for small values of λ, which is reflected in a high validation error. Increasing λ improves the validation error, up until a point where the error increases again, because the least-squares term loses too much weight with respect to the regularization term. Figure 18 also shows the coefficients of the model applied to the data described in Example 4.3 for different values of λ. When λ is small, many coefficients are large, which makes it possible to overfit the training noise through cancellations. For larger λ their magnitudes decreases, eventually becoming too small to produce an accurate fit.

Figure 19 shows that ridge regression outperforms least-squares regression on the dataset of Example 4.3 for small values of n, and has essentially the same performance for larger values, when the least-squares estimator does not overfit the training data (this is expected as the estimators are equivalent for small λ values). The figure also shows that λ values selected by cross validation are larger for small values of n, where regularization is more useful.

In order to analyze the ridge-regression estimator, let us consider data generated by a linear model as in Eq. (104). The following lemma provides an expression for the coefficients obtained by ridge regression estimator in that case. The estimator can be decomposed into a term that depends on the signal and a term that depends on the noise.

Lemma 4.13 (Ridge-regression estimator). *If* $\vec{y}_{\text{train}} := X_{\text{train}}\vec{\beta}_{\text{true}} + \vec{z}_{\text{train}}$, *where* $X_{\text{train}} \in \mathbb{R}^{n\times p}$,

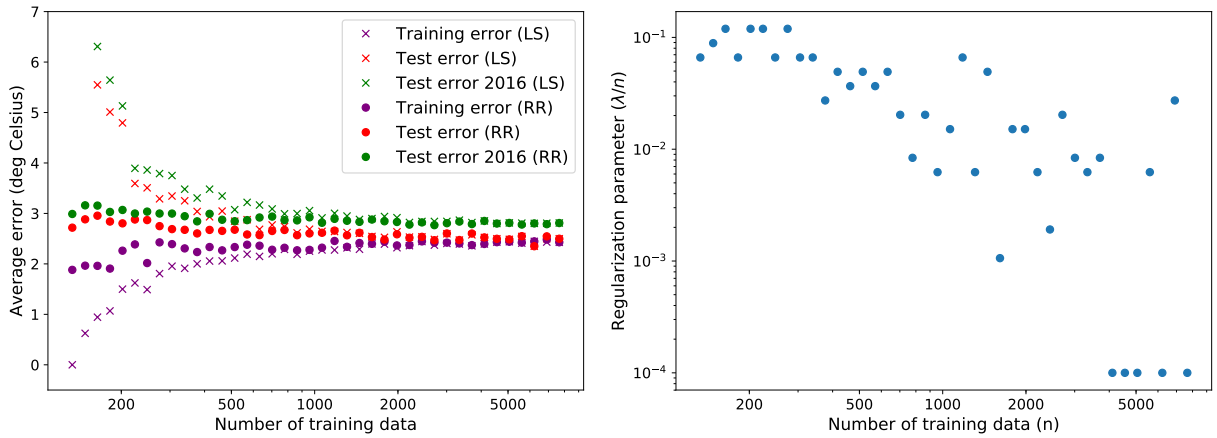


Figure 19: Performance of the ridge-regression estimator on the temperature data described in Example 4.3. The left image shows the RMSE achieved by the model on the training and test sets, and on the 2016 data, for different number of training data and compares it to the RMSE of least-squares regression. The right graph shows the values of λ selected from a validation dataset of size 100 for each number of training data.

$\vec{z}_{\text{train}} \in \mathbb{R}^n$ and $\vec{\beta}_{\text{true}} \in \mathbb{R}^p$, then the solution of Problem (150) is equal to

$$\vec{\beta}_{\text{RR}} = V \begin{bmatrix} \frac{s_1^2}{s_1^2+\lambda} & 0 & \cdots & 0 \\ 0 & \frac{s_2^2}{s_2^2+\lambda} & \cdots & 0 \\ & & \cdots & \\ 0 & 0 & \cdots & \frac{s_p^2}{s_p^2+\lambda} \end{bmatrix} V^T \vec{\beta}_{\text{true}} + V \begin{bmatrix} \frac{s_1}{s_1^2+\lambda} & 0 & \cdots & 0 \\ 0 & \frac{s_2}{s_2^2+\lambda} & \cdots & 0 \\ & & \cdots & \\ 0 & 0 & \cdots & \frac{s_p}{s_p^2+\lambda} \end{bmatrix} U^T \vec{z}_{\text{train}}, \tag{155}$$

where USV^T is the SVD of X_{train} and $s_1, \ldots, s_p$ are the singular values.

Proof. By Theorem 4.1 the solution equals

$$\begin{aligned} \vec{\beta}_{\text{RR}} &= \left(X_{\text{train}}^T X_{\text{train}} + \lambda I\right)^{-1} X_{\text{train}}^T \left(X_{\text{train}} \vec{\beta}_{\text{true}} + \vec{z}_{\text{train}}\right) && (156) \\ &= \left(V S^2 V^T + \lambda V V^T\right)^{-1} \left(V S^2 V^T \vec{\beta}_{\text{true}} + V S U^T \vec{z}_{\text{train}}\right) && (157) \\ &= V \left(S^2 + \lambda I\right)^{-1} V^T \left(V S^2 V^T \vec{\beta}_{\text{true}} + V S U^T \vec{z}_{\text{train}}\right) && (158) \\ &= V \left(S^2 + \lambda I\right)^{-1} S^2 V^T \vec{\beta}_{\text{true}} + V \left(S^2 + \lambda I\right)^{-1} S U^T \vec{z}_{\text{train}}, && (159) \end{aligned}$$

because V is an orthogonal matrix. □

The lemma allows us to derive an expression for the coefficient error:

$$\vec{\beta}_{\text{RR}} - \vec{\beta}_{\text{true}} = -\lambda V \left(S^2 + \lambda I\right)^{-1} V^T \vec{\beta}_{\text{true}} + V \left(S^2 + \lambda I\right)^{-1} S U^T \vec{z}_{\text{train}}. \tag{160}$$

In contrast to the least-squares estimator, the ridge-regression estimator does not produce a perfect estimate when the noise equals zero, unless λ equals zero. The resulting test error is given by

$$y_{\text{test}} - y_{\text{RR}} = \langle \vec{x}_{\text{test}}, \vec{\beta}_{\text{true}} - \vec{\beta}_{\text{RR}} \rangle + z_{\text{test}} \tag{161}$$

$$= z_{\text{test}} + \sum_{i=1}^{p} \frac{\langle \vec{x}_{\text{test}}, \vec{v}_i \rangle \left(\lambda \left\langle \vec{\beta}_{\text{true}}, \vec{v}_i \right\rangle - s_i \langle \vec{u}_i, \vec{z}_{\text{train}} \rangle \right)}{s_i^2 + \lambda}. \tag{162}$$

This expression reveals why ridge regression is an effective remedy against noise amplification. Recall that noise amplification is driven by the small singular values of the training matrix, when the corresponding singular vectors are not an accurate estimate of the singular vector of the covariance matrix of the test data. These small singular values are neutralized by the presence of λ in the denominator of Eq. (162). Consequently, the noise-amplification factor is reduced from $1/s_i^2$ to λ. For the data in Example 4.3, the effect is dramatic: $1/s_i^2$ can be of order 10^4 (the smallest singular values in Figure 16 have magnitude around 10^{-2}), whereas $1/\lambda$ is around 10 for small n (see Figure 19).

5 Low-rank matrix estimation

In this section we consider the problem of estimating a low-rank matrix from data. We begin by studying the connection between the rank of a matrix and its SVD.

5.1 Connection between the rank and the SVD

A direct corollary of Lemma 2.2 is that the rank of a matrix is equal to the number of nonzero singular values. If a matrix only has r nonzero singular values, then its columns are all linear combinations of the corresponding r left singular vectors. Moreover, the SVD also reveals when a matrix is *approximately* rank r. To make this precise, we consider an alternative interpretation of the SVD as a decomposition into rank-1 matrices. For any matrix $A \in \mathbb{R}^{m \times n}$, where $n \leq m$, we can write its SVD in the following form

$$A = \sum_{i=1}^{n} s_i \vec{u}_i \vec{v}_i^T, \tag{163}$$

where the ith term equals the outer product of the left singular vector $\vec{u}_i$ and the right singular vector $\vec{v}_i$ scaled by the corresponding singular value s_i.

Let us consider the vector space of $\mathbb{R}^{m \times n}$ matrices endowed with the standard inner product

$$\langle A, B \rangle := \operatorname{tr}\left(A^T B\right), \quad A, B \in \mathbb{R}^{m \times n}, \tag{164}$$

and the corresponding norm

$$||A||_{\text{F}} := \sqrt{\operatorname{tr}\left(A^T A\right)} = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij}^2}, \tag{165}$$

called the Frobenius norm, which equals the ℓ_2 norm of the vectorized matrix. The rank-1 matrices $\vec{u}_i \vec{v}_i^T$ can be interpreted as orthonormal vectors. Indeed, for all $1 \leq i, j \leq n$

$$\left|\left|\vec{u}_i \vec{v}_i^T\right|\right|_{\mathrm{F}}^2 = \mathrm{tr}\left(\vec{v}_i \vec{u}_i^T \vec{u}_i \vec{v}_i^T\right) \tag{166}$$

$$= \vec{v}_i^T \vec{v}_i \vec{u}_i^T \vec{u}_i = 1, \tag{167}$$

$$\left\langle \vec{u}_i \vec{v}_i^T, \vec{u}_i \vec{v}_i^T \right\rangle = \mathrm{tr}\left(\vec{v}_i \vec{u}_i^T \vec{u}_j \vec{v}_j^T\right) \tag{168}$$

$$= \vec{v}_i^T \vec{v}_j \vec{u}_j^T \vec{u}_i = 0, \text{ if } i \neq j, \tag{169}$$

because the left and right singular vectors are orthonormal sets.

The Frobenius norm of a matrix equals the ℓ_2 norm of its singular values.

Lemma 5.1. *For any matrix $A \in \mathbb{R}^{m \times n}$, with singular values $s_1, \ldots, s_{\min\{m,n\}}$*

$$||A||_{\mathrm{F}} = \sqrt{\sum_{i=1}^{\min\{m,n\}} s_i^2}. \tag{170}$$

Proof. Assume $n \leq m$ (for $m > n$ the same argument applies to the transpose of A),

$$||A||_{\mathrm{F}}^2 = \sum_{i=1}^{n} \left|\left|s_i \vec{u}_i \vec{v}_i^T\right|\right|_{\mathrm{F}}^2, \quad \text{by (163) and Pythagoras's Theorem,} \tag{171}$$

$$= \sum_{i=1}^{n} s_i^2, \quad \text{by homogeneity of norms.} \tag{172}$$

□

We can therefore view the SVD as a decomposition of the matrix in n orthogonal rank-1 matrices. The energy of each component is exactly equal to the corresponding singular value. This provides a way to quantify when a matrix is approximately rank r: its last $n - r$ singular values are small with respect to the first r. If that is the case, most of its energy is concentrated in the r first components of the rank-1 decomposition. The distance (induced by the Frobenius norm) between the matrix and the sum of those components, interpreted as a rank-r approximation, is exactly equal to the ℓ_2 norm of its last $n - r$ singular values,

$$\left|\left|A - \sum_{i=1}^{r} s_i \vec{u}_i \vec{v}_i^T\right|\right|_{\mathrm{F}}^2 = \left|\left|\sum_{i=r+1}^{n} s_i \vec{u}_i \vec{v}_i^T\right|\right|_{\mathrm{F}}^2 \tag{173}$$

$$= \sum_{i=r+1}^{n} s_i^2. \tag{174}$$

In the following section we establish that this is the best possible rank-r approximation to the matrix in Frobenius norm.

5.2 Optimal low-rank matrix estimation

The bilinear model with r factors

$$y[i,j] \approx \sum_{l=1}^{r} a_l[i] b_l[j] \tag{175}$$

introduced in Section 1.3 can be expressed in matrix form:

$$Y \approx \begin{bmatrix} a_1 & a_2 & \cdots & a_r \end{bmatrix} \begin{bmatrix} b_1 & b_2 & \cdots & b_r \end{bmatrix}^T = AB^T, \tag{176}$$

where $Y \in \mathbb{R}^{m \times n}$, $A \in \mathbb{R}^{m \times r}$ and $B \in \mathbb{R}^{r \times m}$. In the collaborative filtering example, m would be the number of products, e.g. movies, and n the number of users. The bilinear model states that the matrix of data Y is approximately rank r.

Let us assume that all entries of Y are observed, although this is usually not the case in collaborative filtering (we will discuss how to deal with missing data later on). Fitting the bilinear model amounts to finding a rank-r matrix that is as close as possible to Y, i.e. solving the optimization problem

$$\min_{A \in \mathbb{R}^{m \times r}, B \in \mathbb{R}^{r \times n}} ||Y - AB||_{\mathrm{F}} \qquad \text{subject to} \quad ||A_{:,1}||_2 = 1, \ldots, ||A_{:,r}||_2 = 1, \tag{177}$$

where without loss of generality we constrain the columns of the left factor A, denoted by $A_{:,i}$ $1 \leq i \leq r$ to have unit norm. The following theorem establishes that truncating the SVD of the observed matrix solves the optimization problem *for any value of* r. For any matrix M, we denote by $M_{1:i,1:j}$ the $i \times j$ submatrix formed by taking the entries that are both in the first i rows and the first j columns. Similarly, we denote by $M_{:,i:j}$ the matrix formed by columns i to j.

Theorem 5.2 (Best rank-r approximation)**.** *Let USV^T be the SVD of a matrix $A \in \mathbb{R}^{m \times n}$. The truncated SVD $U_{:,1:r} S_{1:r,1:r} V_{:,1:r}^T$ is the best rank-r approximation of A in the sense that*

$$U_{:,1:r} S_{1:r,1:r} V_{:,1:r}^T = \underset{\left\{\tilde{A} \mid \operatorname{rank}\left(\tilde{A}\right)=k\right\}}{\arg\min} \left|\left|A - \widetilde{A}\right|\right|_{\mathrm{F}}. \tag{178}$$

Proof. Let $\widetilde{A}$ be an arbitrary matrix in $\mathbb{R}^{m \times n}$ with $\operatorname{rank}(\widetilde{A}) = r$, and let $\widetilde{U} \in \mathbb{R}^{m \times r}$ be a matrix with orthonormal columns such that $\operatorname{col}(\widetilde{U}) = \operatorname{col}(\widetilde{A})$. By Theorem 3.9,

$$\left|\left|U_{:,1:r} U_{:,1:r}^T A\right|\right|_{\mathrm{F}}^2 = \sum_{i=1}^{n} \|\mathcal{P}_{\operatorname{col}(U_{:,1:r})} A_{:,i}\|_2^2 \tag{179}$$

$$\geq \sum_{i=1}^{n} \|\mathcal{P}_{\operatorname{col}(\widetilde{U})} A_{:,i}\|_2^2 \tag{180}$$

$$= \left|\left|\widetilde{U}\widetilde{U}^T A\right|\right|_{\mathrm{F}}^2. \tag{181}$$

The proof relies on the following simple Pythagorean lemma.

Lemma 5.3 (Proof in Section C)**.** *If the column spaces of any pair of matrices $A, B \in \mathbb{R}^{m \times n}$ are orthogonal*

$$||A + B||_{\mathrm{F}}^2 = ||A||_{\mathrm{F}}^2 + ||B||_{\mathrm{F}}^2. \tag{182}$$

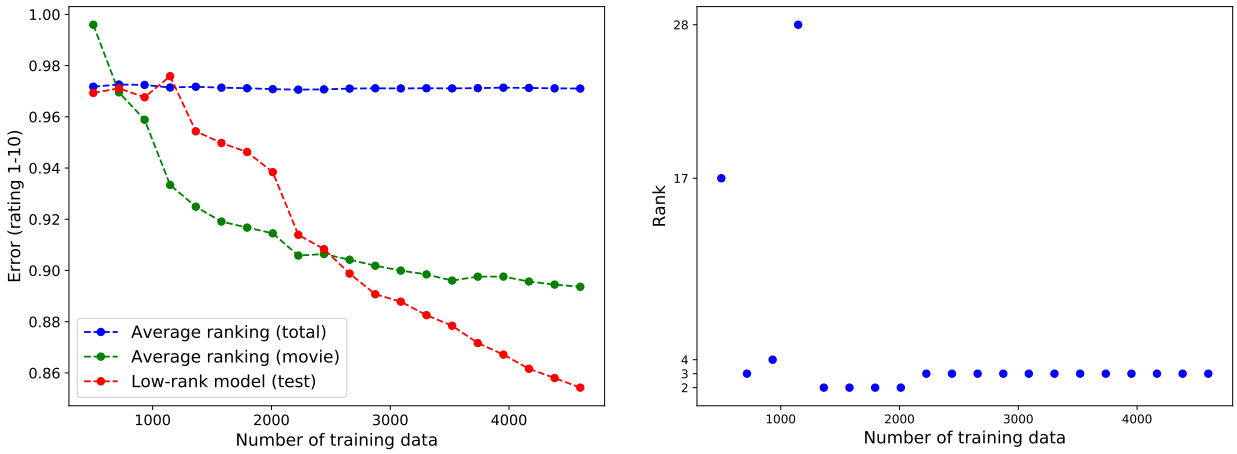


Figure 20: On the left, test error for the experiment described in Example 5.4. On the right, rank selected using the validation data for different amounts of training data.

The column space of $A - \widetilde{U}\widetilde{U}^T A$ is orthogonal to the column space of $\widetilde{A}$ and $\widetilde{U}$, so by Lemma 5.3

$$\left|\left|A - \widetilde{A}\right|\right|_{\mathrm{F}}^2 = \left|\left|A - \widetilde{U}\widetilde{U}^T A\right|\right|_{\mathrm{F}}^2 + \left|\left|\widetilde{A} - \widetilde{U}\widetilde{U}^T A\right|\right|_{\mathrm{F}}^2 \tag{183}$$

$$\geq \left|\left|(I - \widetilde{U}\widetilde{U}^T)A\right|\right|_{\mathrm{F}}^2 \tag{184}$$

$$= ||A||_{\mathrm{F}}^2 - \left|\left|\widetilde{U}\widetilde{U}^T A\right|\right|_{\mathrm{F}}^2 \qquad \text{also by Lemma 5.3} \tag{185}$$

$$\geq ||A||_{\mathrm{F}}^2 - \left|\left|U_{:,1:r}U_{:,1:r}^T A\right|\right|_{\mathrm{F}}^2 \qquad \text{by (181)} \tag{186}$$

$$= \left|\left|A - U_{:,1:r}U_{:,1:r}^T A\right|\right|_{\mathrm{F}}^2 \qquad \text{again by Lemma 5.3.} \tag{187}$$

□

A direct corollary of this theorem for fitting bilinear models is that the problem in Eq. (178) can be solved by computing the SVD and setting $A := U_{:,1:r}$ and $B := S_{1:r,1:r}V_{:,1:r}^T$. It is important to emphasize that these factors are not unique. For any orthogonal matrix $\widetilde{U} \in \mathbb{R}^{r\times r}$, the factors $\widetilde{A} := A\widetilde{U}$ and $\widetilde{B} := \widetilde{U}^T B$ yield exactly the same low-rank matrix.

Example 5.4 (Collaborative filtering)**.** The Movielens dataset[6] contains ratings given by a group of users to popular movies. The ratings are integers between 1 and 10. For this example we select the 100 users and movies with most ratings to form the matrix Y in Eq. (176). From the 10^4 possible ratings, 6,031 are observed. The goal is to perform collaborative filtering: we observe a subset of ratings and try to predict the missing ones. In order to have a ground truth to compare to, we build a test set with 10^3 ratings. We separate the remaining observed ratings into a training set of size n_{train} and a validation set of size $n_{\text{val}} := \max\{n_{\text{train}}, 400\}$.

In order to perform collaborative filtering, we use a bilinear model of the form

$$Y \approx AB + \mu, \tag{188}$$

[6]The data are available at https://grouplens.org/datasets/movielens.

where $A \in \mathbb{R}^{100\times r}$ and $B \in \mathbb{R}^{r\times 100}$ are rank-r factors, and $\mu \in \mathbb{R}$ is a constant that represents the average rating over the whole dataset. To fit the model we first compute μ as the average rating over the training data. We then subtract μ from all the observed ratings, and set the remaining entries of Y to zero. Finally, we compute the best rank-r approximation to the centered matrix by computing its SVD USV^T and setting $A := U_{:,1:r}$ and $B := S_{1:r,1:r}V^T_{:,1:r}$. The estimate for the missing ratings can be read from the corresponding entries in the low-rank matrix (after adding back μ). The rank r is a hyper parameter, which is chosen to minimize the error on the validation data.

We compare the performance of the low-rank model with two simple baseline estimators:

- *Average rating*: The average of all ratings in the training set.
- *Average movie rating*: The average rating for that particular movie in the dataset (the average user rating yielded much worse results).

Figure 20 shows the test error for different values of the training data n_{train}, as well as the corresponding value of r selected in the low-rank model. For small amounts of training data, the average movie rating is a better estimator of the missing ratings. As the number of training data grows, the low-rank estimator becomes the best estimator. In that regime, the selected r is equal to 3, indicating that the data are well approximated by a rank-3 model.

The left image of Figure (21) shows the training and validation errors when $n_{\text{train}} := 4,600$ for different values of r. The number of parameters of the model depends directly on r (depending on how we constrain the factors, it is roughly $r(m+n)$). Large values of r make it possible to fit the training data very well, but do not capture the structure of the test data: the model *overfits*. The optimal value of r in terms of generalization to the validation set is 3. The left image of Figure 21 shows the singular values of the matrix Y containing the 4,600 training data. The matrix is not exactly low rank, but the first few singular values do contain a large fraction of its energy.

Finally, Figure 22 shows the entries of the factor A corresponding to the 20 movies with more ratings, for the rank-3 model trained on 4,600 data. As mentioned above, one can scramble the factors with any orthogonal matrix, so a priori there's no reason for them to be readily interpretable. However, because the rank is so low, the entries are interpretable to some extent. In particular the first and third singular vectors cluster the movies into roughly three groups, depending on the sign of the entries of the vectors. This is visualized on the right of the figure. The red cluster seems to contain *highbrow* movies. The blue cluster contains three Star Wars movies. The green cluster contains some children movies. A couple of movies in the green cluster are close to being assigned to the blue cluster, so their assignment is probably due to noise. △

References

[1] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain, 1994.

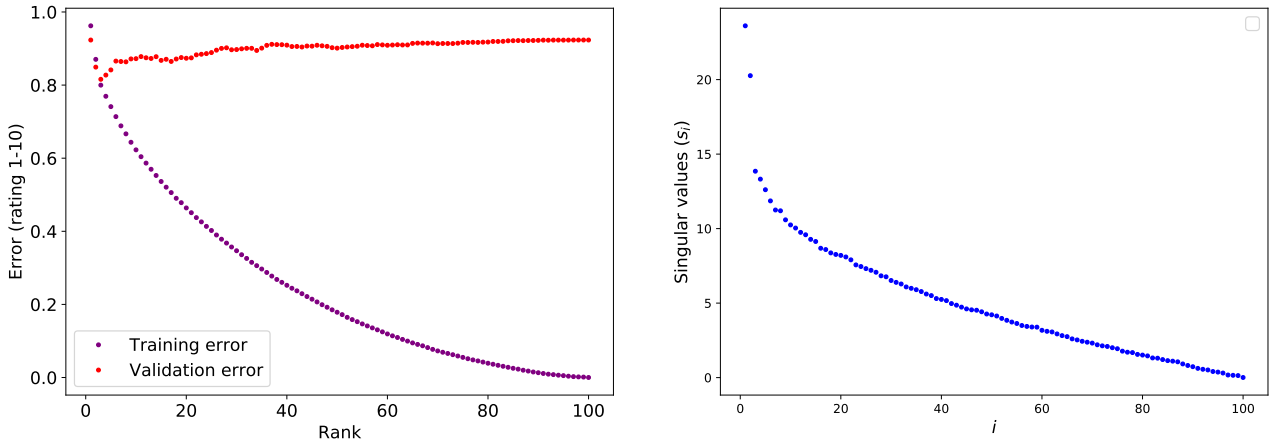


Figure 21: On the left, training and validation error for the experiment described in Example 5.4 when $n_{\text{train}} := 4,600$ for different values of the rank r. On the right, singular values of the matrix Y containing the 4,600 training data. The SVD is computed after centering the matrix by subtracting the average rating from each observed entry.

u_1	u_2	u_3	
-0.03	-0.15	0.06	Forrest Gump
0.14	-0.15	-0.10	Pulp Fiction
0.05	-0.15	-0.11	The Shawshank Redemption
0.04	-0.16	-0.12	Silence of the Lambs
0.09	-0.22	0.06	Star Wars Ep. IV
-0.12	-0.14	0.03	Jurassic Park
0.10	-0.14	0.10	The Matrix
-0.07	-0.10	0.03	Toy Story
0.04	-0.13	-0.12	Schindler's List
-0.01	-0.03	0.08	Terminator 2
0.08	-0.2	0.22	Star Wars Ep. V
0.00	-0.12	-0.10	Braveheart
0.02	-0.12	0.00	Back to the Future
0.07	-0.09	-0.27	Fargo
0.04	-0.23	0.12	Raiders of the Lost Ark
0.06	-0.13	-0.12	American Beauty
-0.21	0.07	0.08	Independence Day
0.06	-0.16	0.19	Star Wars Ep. VI
-0.09	-0.08	0.01	Aladdin
-0.04	-0.08	-0.02	The Fugitive

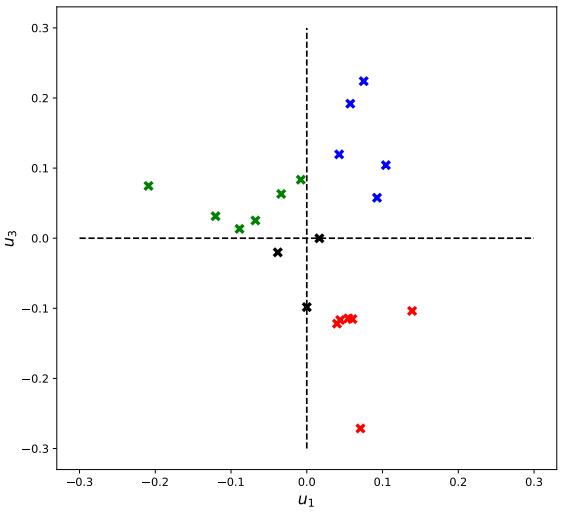


Figure 22: Entries of the factor A corresponding to the 20 movies with more ratings in the rank-3 model of Example 5.4. The sign of the first and third singular vectors are used to cluster the movies, as illustrated by the graph on the right.

Appendix

A Proof of Lemma 3.10

Let $\vec{a}_1, \vec{a}_2, \ldots, \vec{a}_{d_1}$ be a basis for the first subspace and $\vec{b}_1$, $\vec{b}_2$, ..., $\vec{b}_{d_2}$ a basis for the second. Because the dimension of the vector space is n, the set of vectors $\vec{a}_1$, ..., $\vec{a}_{d_1}$,$\vec{b}_1$, ..., $\vec{b}_{d_2}$ are not linearly independent. There must exist scalars α_1, ..., α_{d_1}, β_1, ..., β_{d_2}, which are not all equal to zero, such that

$$\sum_{i=1}^{d_1} \alpha_i \vec{a}_i + \sum_{j=1}^{d_2} \beta_i \vec{b}_i = 0. \tag{189}$$

The vector

$$\vec{x} := \sum_{i=1}^{d_1} \alpha_i \vec{a}_i = -\sum_{j=1}^{d_2} \beta_i \vec{b}_i \tag{190}$$

cannot equal zero because both $\vec{a}_1$, ..., $\vec{a}_{d_1}$ and $\vec{b}_1$, ..., $\vec{b}_{d_2}$ are bases by assumption. $\vec{x}$ belongs to the intersection of the two subspaces, which completes the proof.

B Equivalence of affine and linear regression model

The following lemma shows that centering the data before computing the least-squares fit is exactly equivalent to fitting an affine model with the same cost function.

Lemma B.1. *For any matrix $X \in \mathbb{R}^{n \times m}$ and any vector $\vec{y}$, let*

$$\left\{\beta_{\text{LS},0}, \vec{\beta}_{\text{LS}}\right\} := \arg\min_{\beta_0, \vec{\beta}} \left|\left|\vec{y} - X\vec{\beta} - \beta_0 \vec{1}\right|\right|_2^2 \tag{191}$$

be the coefficients corresponding to an affine fit, where $\vec{1}$ is a vector containing n ones, and let

$$\vec{\beta}_{\text{LS}}^{\,\text{cent}} := \arg\min_{\vec{\beta}} \left|\left|\vec{y}^{\,\text{cent}} - X^{\,\text{cent}}\vec{\beta}\right|\right|_2^2 \tag{192}$$

be the coefficients of a linear fit after centering both X and $\vec{y}$ using their respective averages (in the case of X, the column-wise average). Then,

$$X\vec{\beta}_{\text{LS}} + \beta_{\text{LS},0} = X^{\,\text{cent}}\vec{\beta}_{\text{LS}}^{\,\text{cent}} + \text{av}\,(y)\,. \tag{193}$$

Proof. To ease notation let $\widetilde{X} := X^{\,\text{cent}}$ and $\widetilde{x} := X^T\vec{1}$. Note that

$$\vec{y}^{\,\text{cent}} = \vec{y} - \frac{1}{n}\vec{1}\,\vec{1}^{\,T}\vec{y}, \tag{194}$$

$$\widetilde{X} = X - \frac{1}{n}\vec{1}\,\widetilde{x}^{\,T}. \tag{195}$$

By Theorem 4.1

$$\begin{bmatrix} \vec{\beta}_{\mathrm{LS}} \\ \beta_{\mathrm{LS},0} \end{bmatrix} = \left(\begin{bmatrix} X & \vec{1} \end{bmatrix}^T \begin{bmatrix} X & \vec{1} \end{bmatrix} \right)^{-1} \begin{bmatrix} X & \vec{1} \end{bmatrix}^T \vec{y} \tag{196}$$

$$= \begin{bmatrix} X^T X & \widetilde{x} \\ \widetilde{x}^T & n \end{bmatrix}^{-1} \begin{bmatrix} X^T \vec{y} \\ \vec{1}^T \vec{y} \end{bmatrix}. \tag{197}$$

We now apply the following fact, which can be checked by multiplying the two matrices and verifying that the product is the identity: For any matrices $A \in \mathbb{R}^{m\times}$, if

$$B = A - \frac{1}{n} \widetilde{x} \widetilde{x}^T \tag{198}$$

be invertible, then

$$\begin{bmatrix} A & \widetilde{x} \\ \widetilde{x}^T & n \end{bmatrix}^{-1} = \begin{bmatrix} B^{-1} & -\frac{1}{n} B^{-1} \widetilde{x} \\ -\frac{1}{n} \widetilde{x}^T B^{-1} & \frac{1}{n} + \frac{1}{n^2} \widetilde{x}^T B^{-1} \widetilde{x} \end{bmatrix}. \tag{199}$$

Setting $A := X^T X$, we have

$$B = X^T X - \frac{1}{n} \widetilde{x} \widetilde{x}^T \tag{200}$$

$$= \left(X - \frac{1}{n} \vec{1} \widetilde{x}^T \right)^T \left(X - \frac{1}{n} \vec{1} \widetilde{x}^T \right) \tag{201}$$

$$= \widetilde{X}^T \widetilde{X}. \tag{202}$$

As a result,

$$\begin{bmatrix} \vec{\beta}_{\mathrm{LS}} \\ \beta_{\mathrm{LS},0} \end{bmatrix} = \begin{bmatrix} \left(\widetilde{X}^T \widetilde{X} \right)^{-1} & -\frac{1}{n} \left(\widetilde{X}^T \widetilde{X} \right)^{-1} \widetilde{x} \\ -\frac{1}{n} \widetilde{x}^T \left(\widetilde{X}^T \widetilde{X} \right)^{-1} & \frac{1}{n} + \frac{1}{n^2} \widetilde{x}^T \left(\widetilde{X}^T \widetilde{X} \right)^{-1} \widetilde{x} \end{bmatrix} \begin{bmatrix} X^T \vec{y} \\ \vec{1}^T \vec{y} \end{bmatrix} \tag{203}$$

$$= \begin{bmatrix} \left(\widetilde{X}^T \widetilde{X} \right)^{-1} X^T \left(\vec{y} - \frac{1}{n} \vec{1} \vec{1}^T \vec{y} \right) \\ -\frac{1}{n} \widetilde{x}^T \left(\widetilde{X}^T \widetilde{X} \right)^{-1} X^T \left(\vec{y} - \frac{1}{n} \vec{1} \vec{1}^T \vec{y} \right) + \frac{\vec{1}^T \vec{y}}{n} \end{bmatrix}, \tag{204}$$

which implies

$$X \vec{\beta}_{\mathrm{LS}} + \beta_{\mathrm{LS},0} \vec{1} = X \left(\widetilde{X}^T \widetilde{X} \right)^{-1} X^T \vec{y}^{\,\mathrm{cent}} - \frac{1}{n} \vec{1} \widetilde{x}^T \left(\widetilde{X}^T \widetilde{X} \right)^{-1} X^T \vec{y}^{\,\mathrm{cent}} + \mathrm{av}\left(\vec{y}\right) \vec{1} \tag{205}$$

$$= \widetilde{X} \left(\widetilde{X}^T \widetilde{X} \right)^{-1} X^T \vec{y}^{\,\mathrm{cent}} + \mathrm{av}\left(\vec{y}\right) \vec{1} \tag{206}$$

$$= \widetilde{X} \left(\widetilde{X}^T \widetilde{X} \right)^{-1} \widetilde{X}^T \vec{y}^{\,\mathrm{cent}} + \mathrm{av}\left(\vec{y}\right) \vec{1}, \tag{207}$$

where the last equality follows from

$$\widetilde{X}^T \vec{y}^{\,\text{cent}} = \left(X - \frac{1}{n}\vec{1}\,\vec{1}^{\,T}X\right)^T \left(\vec{y} - \frac{1}{n}\vec{1}\vec{1}^{\,T}\vec{y}\right) \tag{208}$$

$$= X^T\vec{y} - \frac{1}{n}X^T\vec{1}\vec{1}^{\,T}\vec{y} - \frac{1}{n}X^T\vec{1}\vec{1}^{\,T}\vec{y} + \frac{1}{n^2}X^T\vec{1}\vec{1}^{\,T}\vec{1}\vec{1}^{\,T}\vec{y} \tag{209}$$

$$= X^T\vec{y} - \frac{1}{n}X^T\vec{1}\vec{1}^{\,T}\vec{y} \tag{210}$$

$$= X^T\vec{y}^{\,\text{cent}}. \tag{211}$$

Since $\vec{\beta}_{\text{LS}}^{\,\text{cent}} = \left(\widetilde{X}^T\widetilde{X}\right)^{-1}\widetilde{X}^T\vec{y}^{\,\text{cent}}$ the proof is complete. □

C Proof of Lemma 5.3

If the column spaces of any pair of matrices $A, B \in \mathbb{R}^{m\times n}$ are orthogonal then

$$\langle A, B\rangle = 0, \tag{212}$$

since can write the inner product as a sum of products between the columns of A and B, which are all zero:

$$\langle A, B\rangle := \text{tr}\left(A^T B\right) \tag{213}$$

$$= \sum_{i=1}^{n} \langle A_{:,i}, B_{:,i}\rangle \tag{214}$$

$$= 0. \tag{215}$$

The lemma then follows from Pythagoras's theorem.