Introduction to AI

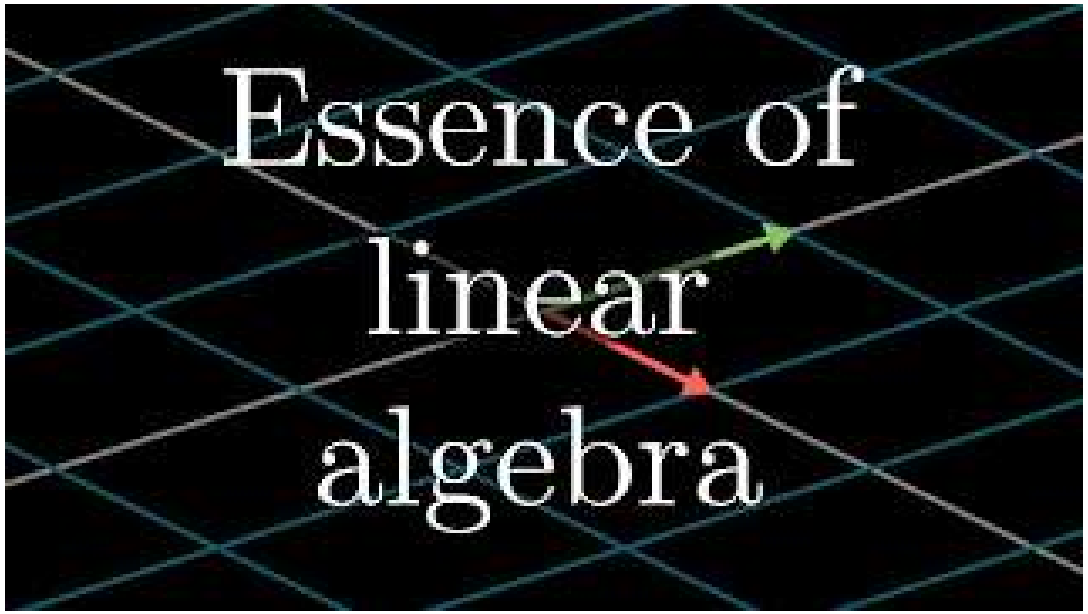
Embedding

Chien-Yu Chen

BIME@NTU

Essence of linear algebra

- https://www.youtube.com/watch?v=fNk_zzaMoSs&list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab&pp=iAQB



Objectives of word embedding

- How learning can be improved by representing words as points in a high-dimensional space, rather than as atomic values
- A representation of words that does not require manual feature engineering, but allows for generalization between related words

One-hot vector

- Encoding the i^{th} word in the dictionary with a 1 bit in the i^{th} input position and a 0 in all the other positions
- Such a representation would not capture the similarity between words

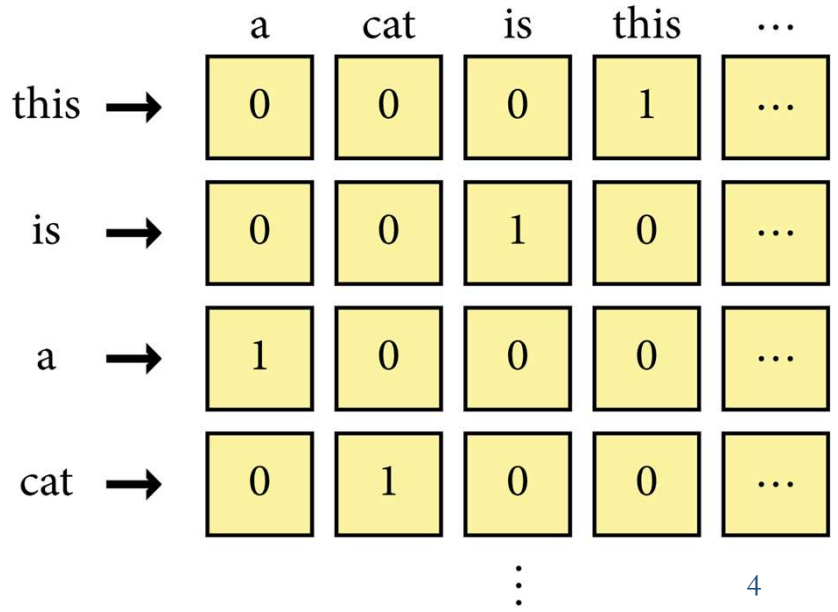


Figure 7 | Automated Prediction of Good Dictionary EXamples (GDEX): A Comprehensive Experiment with Distant Supervision, Machine Learning, and Word Embedding-Based Deep Learning Techniques (hindawi.com)

N-gram count

- "You shall know a word by the company it keeps"
  - representing each word with a vector of n-gram counts of all the phrases that the word appears in
- With a 100,000-word vocabulary, there are 10^{25} 5-grams to keep track of (although vectors in this 10^{25}-dimensional space would be quite sparse—most of the counts would be zero).

Word embedding

- Word embeddings are learned automatically from the data
- The individual dimensions and their numeric values do not have discernible meanings
- The feature space has the property that similar words end up having similar vectors

$$
\begin{aligned}
&\text{“aardvark”} = [-0.7, +0.2, -3.2, \ldots] \\
&\text{“abacus”} = [+0.5, +0.9, -1.3, \ldots] \\
&\ldots \\
&\text{“zyzzyva”} = [-0.1, +0.8, -0.4, \ldots]
\end{aligned}
$$

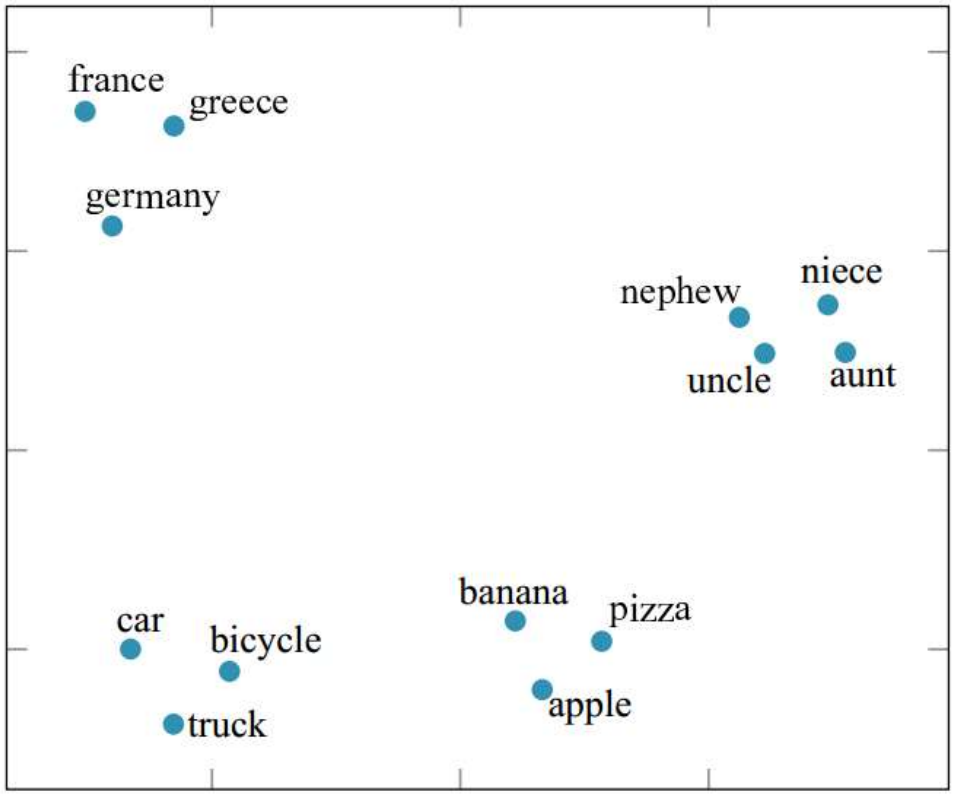


Figure 24.1 Word embedding vectors computed by the GloVe algorithm trained on 6 billion words of text. 100-dimensional word vectors are projected down onto two dimensions in this visualization. Similar words appear near each other.

A	B	C	D = C + (B − A)	Relationship
Athens	Greece	Oslo	Norway	*Capital*
Astana	Kazakhstan	Harare	Zimbabwe	*Capital*
Angola	kwanza	Iran	rial	*Currency*
copper	Cu	gold	Au	*Atomic Symbol*
Microsoft	Windows	Google	Android	*Operating System*
New York	New York Times	Baltimore	Baltimore Sun	*Newspaper*
Berlusconi	Silvio	Obama	Barack	*First name*
Switzerland	Swiss	Cambodia	Cambodian	*Nationality*
Einstein	scientist	Picasso	painter	*Occupation*
brother	sister	grandson	granddaughter	*Family Relation*
Chicago	Illinois	Stockton	California	*State*
possibly	impossibly	ethical	unethical	*Negative*
mouse	mice	dollar	dollars	*Plural*
easy	easiest	lucky	luckiest	*Superlative*
walking	walked	swimming	swam	*Past tense*

Figure 24.2 A word embedding model can sometimes answer the question "**A** is to **B** as **C** is to [what]?" with vector arithmetic: given the word embedding vectors for the words **A**, **B**, and **C**, compute the vector $\mathbf{D} = \mathbf{C} + (\mathbf{B} - \mathbf{A})$ and look up the word that is closest to **D**. (The answers in column **D** were computed automatically by the model. The descriptions in the "Relationship" column were added by hand.) Adapted from ? (?, ?).

Generic pretrained vectors

- The commonly used vector dictionaries include WORD2VEC, GloVe (Global Vectors), and FASTTEXT,
- Unlike generic pretrained embeddings, word embeddings produced for a specific task can be trained on a carefully selected corpus and will tend to emphasize aspects of words that are useful for the task.

DNA and protein languages

JOURNAL ARTICLE

DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome

FREE

Yanrong Ji, Zhihan Zhou, Han Liu ✉, Ramana V Davuluri ✉ Author Notes

Bioinformatics, Volume 37, Issue 15, August 2021, Pages 2112–2120,
https://doi.org/10.1093/bioinformatics/btab083
Published: 04 February 2021 **Article history ▼**

JOURNAL ARTICLE

ProteinBERT: a universal deep-learning model of protein sequence and function

Nadav Brandes ✉, Dan Ofer, Yam Peleg, Nadav Rappoport, Michal Linial Author Notes

Bioinformatics, Volume 38, Issue 8, March 2022, Pages 2102–2110,
https://doi.org/10.1093/bioinformatics/btac020
Published: 10 February 2022 **Article history ▼**

Part-of-speech (POS) tagging

- The word cut can be a present-tense verb (現在式動詞) (transitive (及物) or intransitive (不及物)), a past-tense verb (過去式動詞), an infinitive verb (不定式動詞), a past participle (過去分詞), an adjective (形容詞), or a noun (名詞).

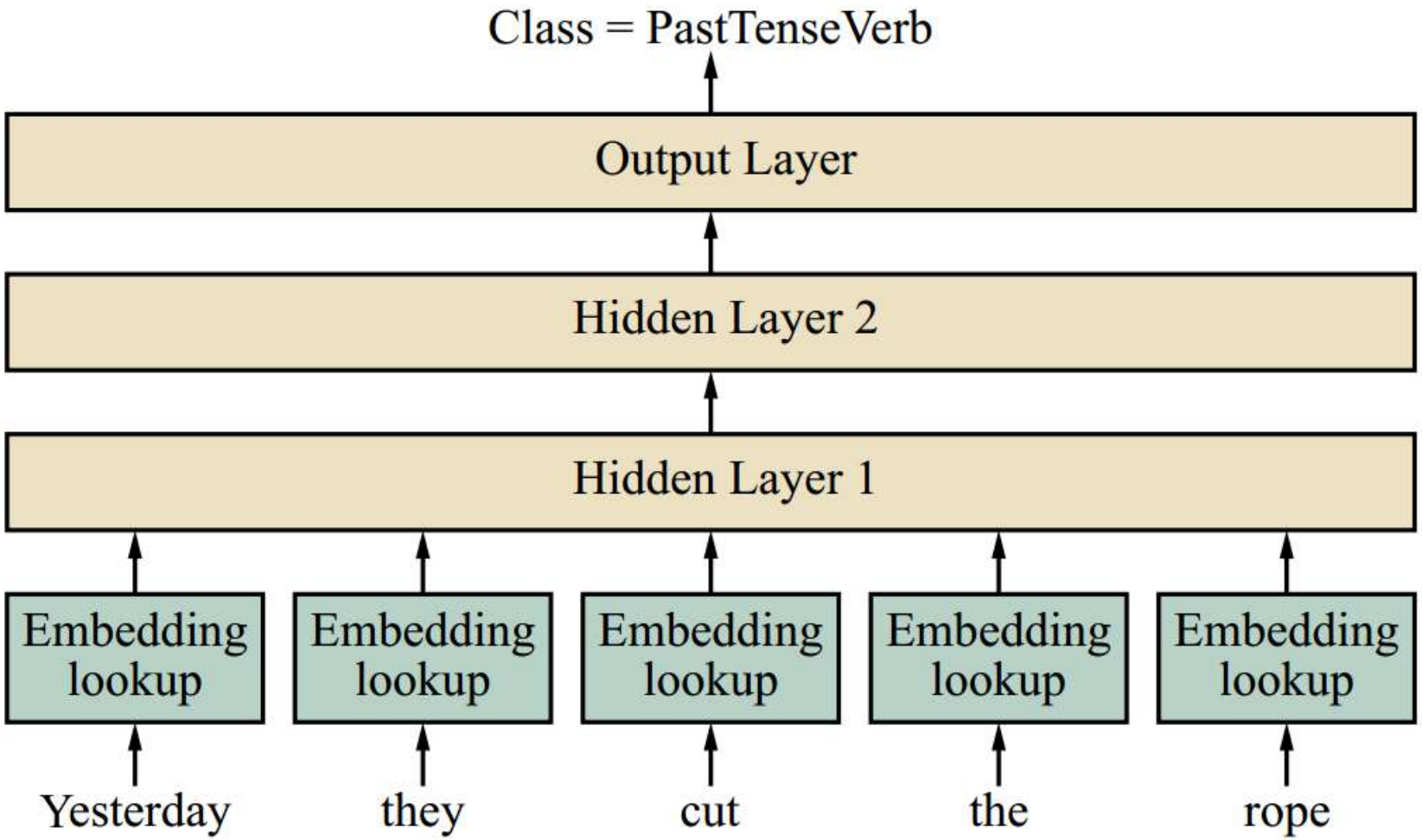


Figure 24.3 Feedforward part-of-speech tagging model. This model takes a 5-word window as input and predicts the tag of the word in the middle—here, *cut*. The model is able to account for word position because each of the 5 input embeddings is multiplied by a different part of the first hidden layer. The parameter values for the word embeddings and for the three layers are all learned simultaneously during training.

POS tagger

1. Choose the width w (an odd number of words) for the prediction window to be used to tag each word. A typical value is $w=5$, meaning that the tag is predicted based on the word plus the two words to the left and the two words to the right. Split every sentence in your corpus into overlapping windows of length w. Each window produces one training example consisting of the w words as input and the POS category of the middle word as output.
2. Create a vocabulary of all of the unique word tokens that occur more than, say, 5 times in the training data. Denote the total number of words in the vocabulary as v.
3. Sort this vocabulary in any arbitrary order (perhaps alphabetically).
4. Choose a value d as the size of each word embedding vector.
5. Create a new v-by-d weight matrix called $\mathbf{E}$. This is the word embedding matrix. Row i of $\mathbf{E}$ is the word embedding of the ith word in the vocabulary. Initialize $\mathbf{E}$ randomly (or from pretrained vectors).
6. Set up a neural network that outputs a part of speech label, as shown in Figure 24.3. The first layer will consist of w copies of the embedding matrix. We might use two additional hidden layers, $\mathbf{z}_1$ and $\mathbf{z}_2$ (with weight matrices $\mathbf{W}_1$ and $\mathbf{W}_2$, respectively), followed by

POS tagger (cont.)

a softmax layer yielding an output probability distribution $\hat{\mathbf{y}}$ over the possible part-of-speech categories for the middle word:

$$\begin{aligned}\mathbf{z}_1 &= \sigma(\mathbf{W}_1\mathbf{x})\\ \mathbf{z}_2 &= \sigma(\mathbf{W}_2\mathbf{z}_1)\\ \hat{\mathbf{y}} &= \text{softmax}(\mathbf{W}_{out}\mathbf{z}_2)\,.\end{aligned}$$

7. To encode a sequence of w words into an input vector, simply look up the embedding for each word and concatenate the embedding vectors. The result is a real-valued input vector $\mathbf{x}$ of length wd. Even though a given word will have the same embedding vector whether it occurs in the first position, the last, or somewhere in between, each embedding will be multiplied by a different part of the first hidden layer; therefore we are implicitly encoding the relative position of each word.
8. Train the weights $\mathbf{E}$ and the other weight matrices $\mathbf{W}_1$, $\mathbf{W}_2$, and $\mathbf{W}_{out}$ using gradient descent. If all goes well, the middle word, *cut*, will be labeled as a past-tense verb, based on the evidence in the window, which includes the temporal past word "yesterday," the third-person subject pronoun "they" immediately before *cut*, and so on.

Softmax function

The standard (unit) softmax function $\sigma : \mathbb{R}^K \mapsto (0,1)^K$ where $K \geq 1$ is defined by the formula

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K.$$

In words, it applies the standard exponential function to each element z_i of the input vector $\mathbf{z}$ and normalizes these values by dividing by the sum of all these exponentials. The normalization ensures that the sum of the components of the output vector $\sigma(\mathbf{z})$ is 1. The term "softmax" derives from the amplifying effects of the exponential on any maxima in the input vector. For example, the standard softmax of $(1, 2, 8)$ is approximately $(0.001, 0.002, 0.997)$, which amounts to assigning almost all of the total unit weight in the result to the position of the vector's maximal element (of 8).

Softmax function - Wikipedia

Optimization – finding global maximum

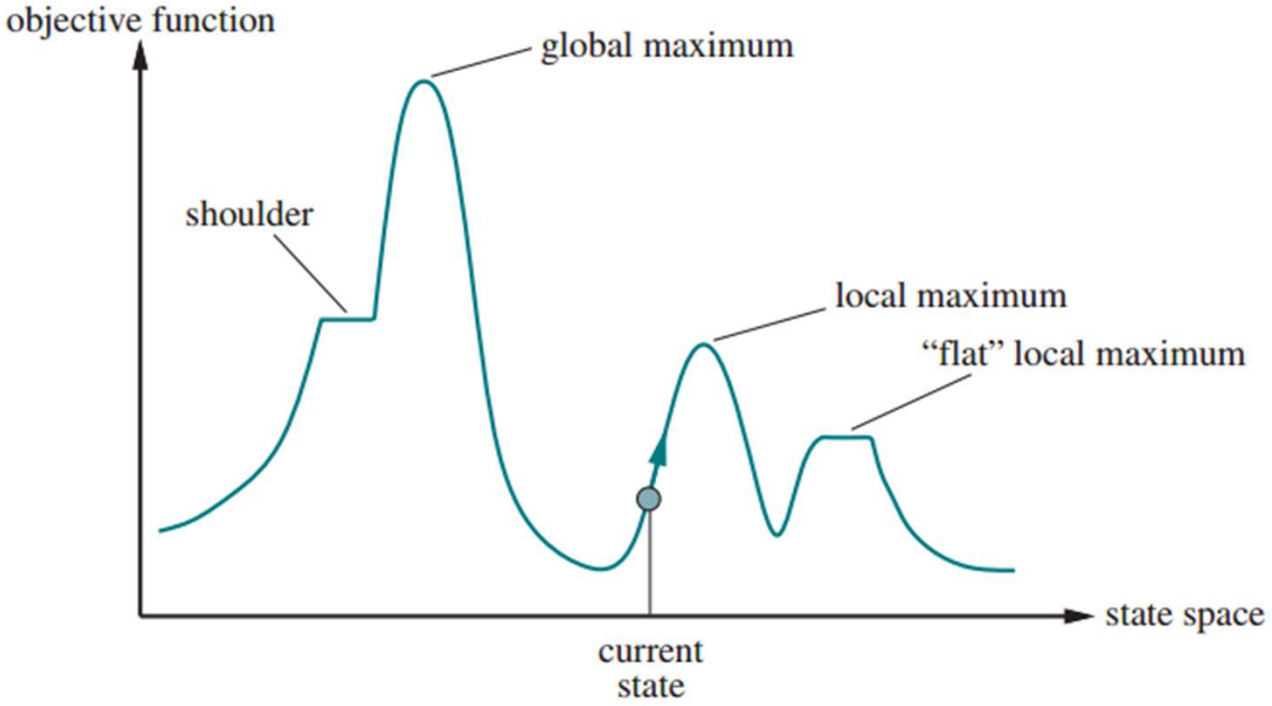


Figure 4.1 A one-dimensional state-space landscape in which elevation corresponds to the objective function. The aim is to find the global maximum.

If elevation corresponds to cost, then the aim is to find the lowest valley—a global minimum—and we call it gradient descent.

Gradient descent (discussed in Chapters 4 and 6) is the industry-leading, convex optimization method for high-dimensional systems. It minimizes residuals by computing the gradient of a given fitting function. The iterative procedure updates the solution by *moving downhill* in the residual space. The Newton–Raphson method is a one-dimensional version of gradient descent. Since it is often applied in high-dimensional settings, it is prone to find only local minima. Critical innovations for big data applications include stochastic gradient descent and the backpropagation algorithm, which makes the optimization amenable to computing the gradient itself.

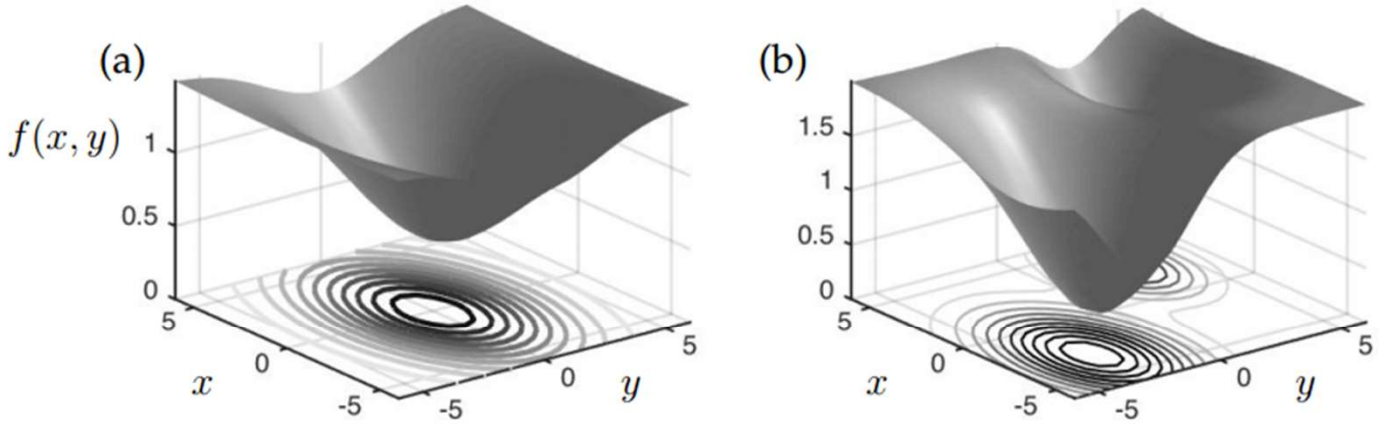
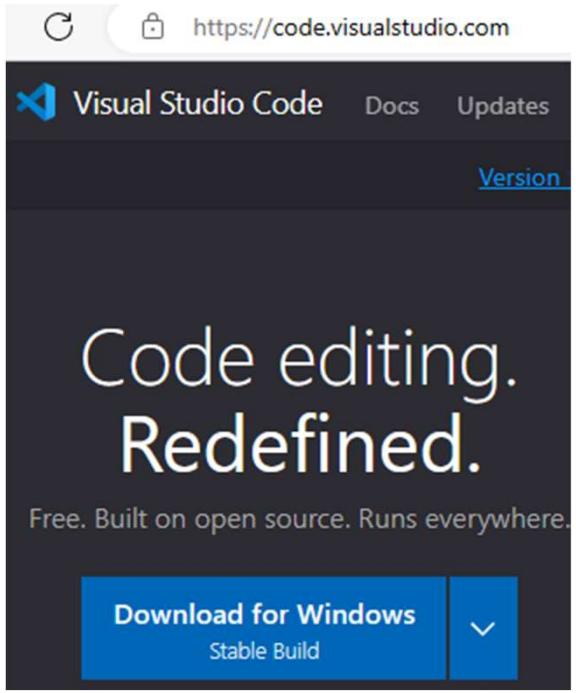


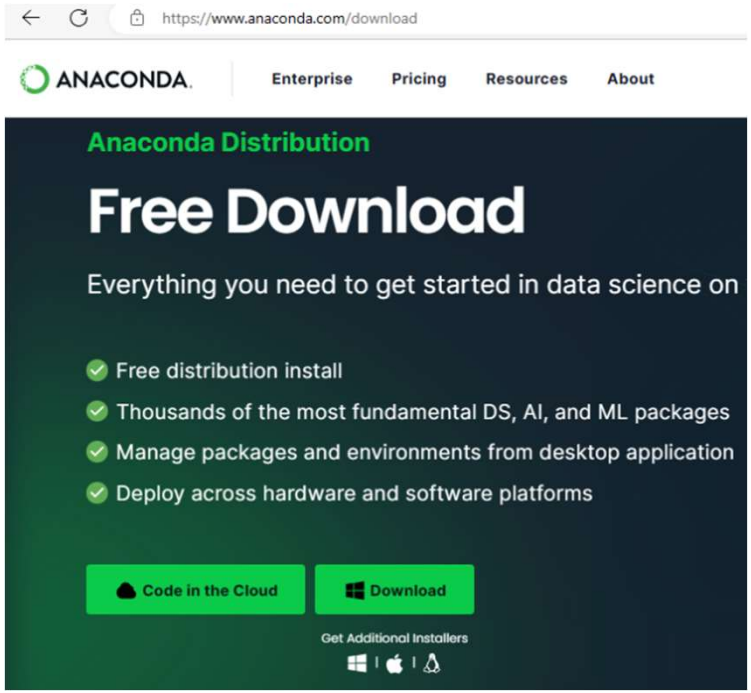
Figure 4.3: Two objective function landscapes representing (a) a convex function and (b) a non-convex function. Convex functions have many guarantees of convergence, while non-convex functions have a variety of pitfalls that can limit the success of gradient descent. For non-convex functions, local minima and an inability to compute gradient directions (derivatives that are near zero) make it challenging for optimization.

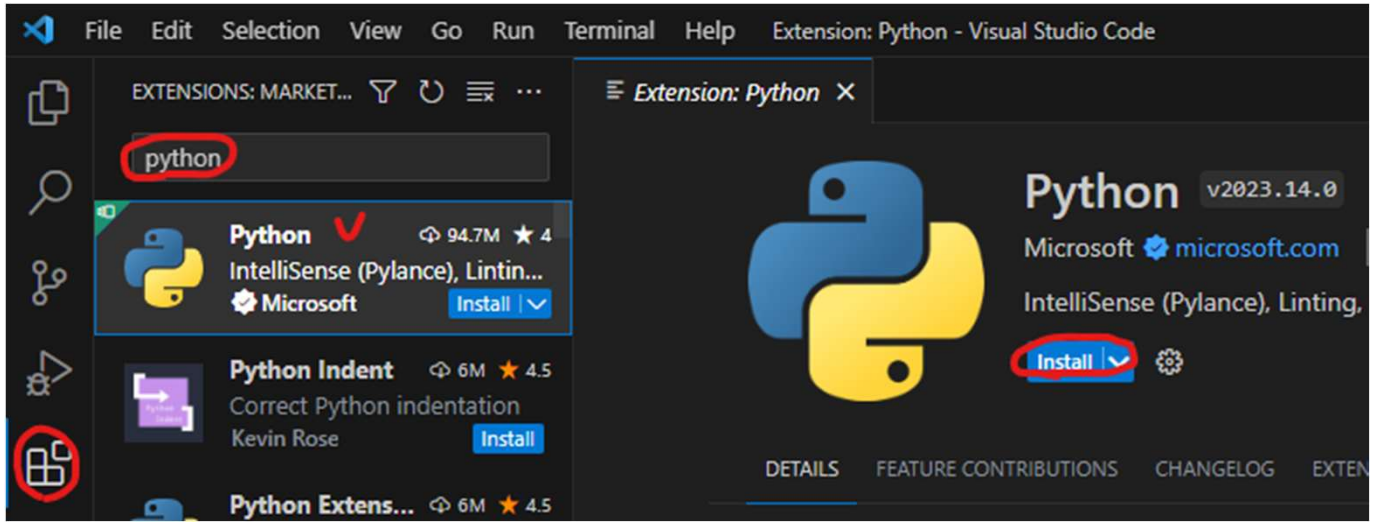
Learning Python

NumPy documentation — NumPy v1.25 Manual

Matplotlib documentation — Matplotlib 3.7.2 documentation







Install Jupyter (Notebook), too.

Singular value decomposition (SVD)

Definition of the SVD

Generally, we are interested in analyzing a large data set $\mathbf{X} \in \mathbb{C}^{n \times m}$:

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_m \\ | & | & & | \end{bmatrix}. \tag{1.1}$$

The columns $\mathbf{x}_k \in \mathbb{C}^n$ may be measurements from simulations or experiments.

For many systems n >> m, resulting in a tall-skinny matrix, as opposed to a short-fat matrix when n << m.

The SVD is a unique matrix decomposition that exists for every complex-valued matrix $\mathbf{X} \in \mathbb{C}^{n \times m}$:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*, \tag{1.2}$$

where $\mathbf{U} \in \mathbb{C}^{n \times n}$ and $\mathbf{V} \in \mathbb{C}^{m \times m}$ are *unitary* matrices[1] with orthonormal columns, and $\mathbf{\Sigma} \in \mathbb{R}^{n \times m}$ is a matrix with real, non-negative entries on the diagonal and zeros off the diagonal. Here * denotes the complex conjugate transpose.[2] As we will discover throughout this chapter, the condition that $\mathbf{U}$ and $\mathbf{V}$ are unitary is used extensively.

When $n \geq m$, the matrix $\mathbf{\Sigma}$ has at most m non-zero elements on the diagonal, and may be written as

$$\mathbf{\Sigma} = \begin{bmatrix} \hat{\mathbf{\Sigma}} \\ \mathbf{0} \end{bmatrix}.$$

Therefore, it is possible to *exactly* represent $\mathbf{X}$ using the *economy* SVD:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* = \begin{bmatrix} \hat{\mathbf{U}} & \hat{\mathbf{U}}^{\perp} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{\Sigma}} \\ \mathbf{0} \end{bmatrix} \mathbf{V}^* = \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\mathbf{V}^*. \tag{1.3}$$

[1] A square matrix $\mathbf{U}$ is unitary if $\mathbf{U}\mathbf{U}^* = \mathbf{U}^*\mathbf{U} = \mathbf{I}$.

[2] For real-valued matrices, this is the same as the regular transpose $\mathbf{X}^* = \mathbf{X}^T$.

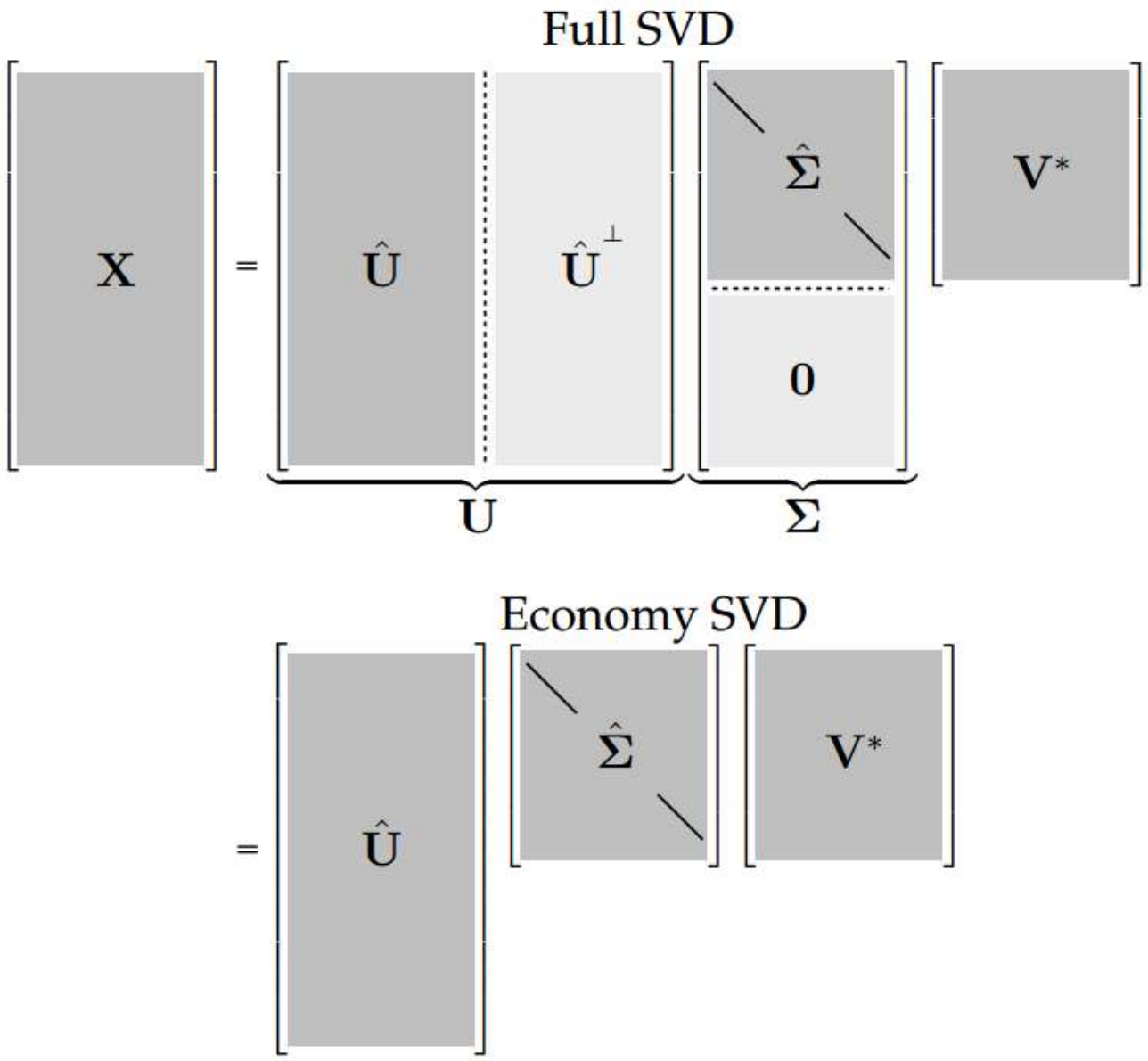


Figure 1.1: Schematic of matrices in the full and economy SVD.

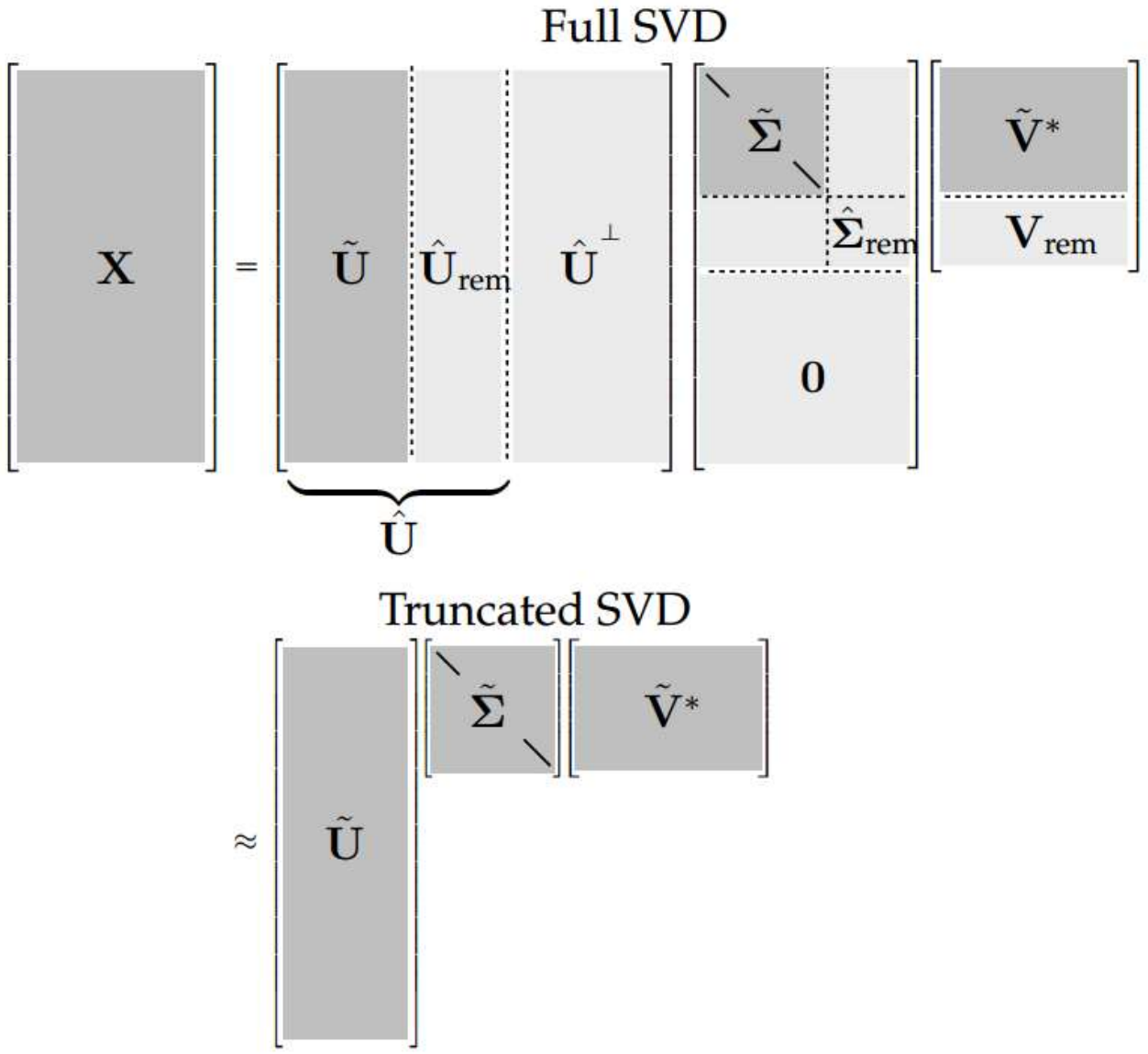


Figure 1.2: Schematic of truncated SVD. The subscript '$_{rem}$' denotes the remainder of $\hat{U}$, $\hat{\Sigma}$, or V after truncation.

Matrix approximation

Because $\mathbf{\Sigma}$ is diagonal, it is possible to express the matrix $\mathbf{X} = \mathbf{U\Sigma V}^*$ as a sum of rank-one matrices:

$$\mathbf{X} = \sum_{k=1}^{m} \sigma_k \mathbf{u}_k \mathbf{v}_k^* = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^* + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^* + \cdots + \sigma_m \mathbf{u}_m \mathbf{v}_m^*, \tag{1.4}$$

where σ_k is the kth diagonal entry of $\mathbf{\Sigma}$, and $\mathbf{u}_k$ and $\mathbf{v}_k$ are the kth columns of $\mathbf{U}$ and $\mathbf{V}$, respectively. This is known as the *dyadic* summation. The singular values σ_k are arranged in decreasing order, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m \geq 0$, so each subsequent rank-one matrix $\sigma_k \mathbf{u}_k \mathbf{v}_k^*$ is less important than the previous matrix in capturing the information in $\mathbf{X}$. For many systems, the singular values σ_k decrease rapidly, and it is possible to obtain a good approximation of $\mathbf{X}$ by truncating at some rank r:

$$\mathbf{X} \approx \tilde{\mathbf{X}} = \sum_{k=1}^{r} \sigma_k \mathbf{u}_k \mathbf{v}_k^* = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^* + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^* + \cdots + \sigma_r \mathbf{u}_r \mathbf{v}_r^*. \tag{1.5}$$

The optimal rank-r approximation

Theorem 1.1 (Eckart–Young [228]) *The optimal rank-r approximation to* $\mathbf{X}$*, in a least-squares sense, is given by the rank-r SVD truncation* $\tilde{\mathbf{X}}$*:*

$$\underset{\tilde{\mathbf{X}},\ \text{s.t. rank}(\tilde{\mathbf{X}})=r}{\operatorname{argmin}} \quad \|\mathbf{X}-\tilde{\mathbf{X}}\|_F = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^*. \tag{1.6}$$

Again, $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}$ denote the first r leading columns of $\mathbf{U}$ and $\mathbf{V}$, and $\tilde{\mathbf{\Sigma}}$ contains the leading $r \times r$ sub-block of $\mathbf{\Sigma}$. The Frobenius norm above is defined as $\|\mathbf{X}\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}|X_{ij}|^2}$, which is equivalent to the 2-norm of the vectorized matrix $\mathbf{X}(:)$.

Example: Image Compression

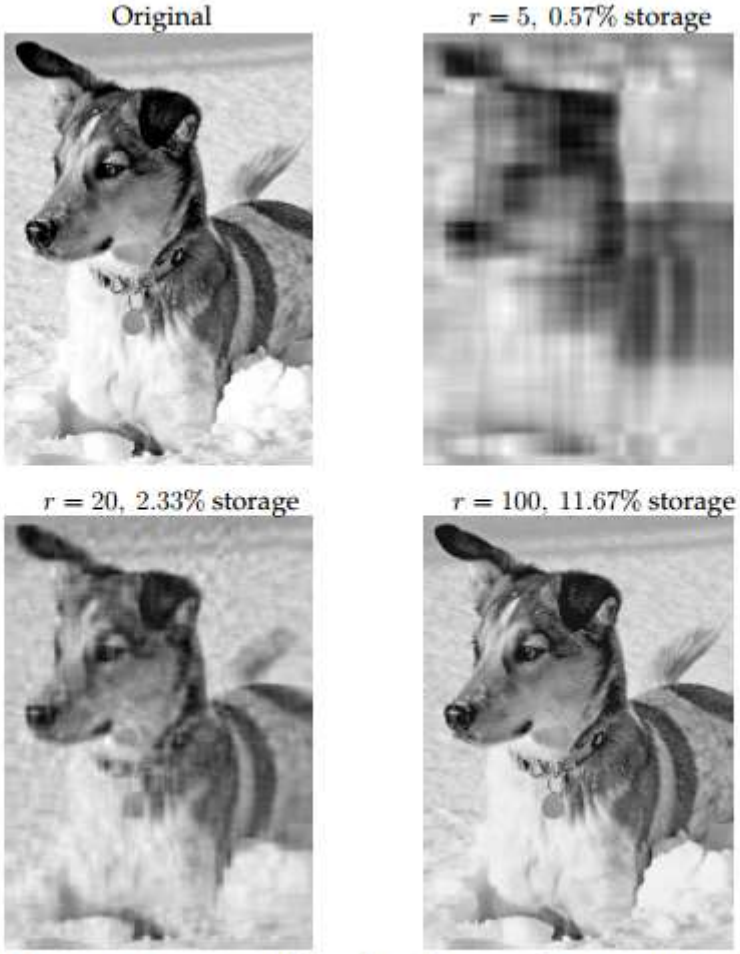


Figure 1.3: Image compression of Mordecai the snow dog, truncating the SVD at various ranks r. Original image resolution is 2000×1500.

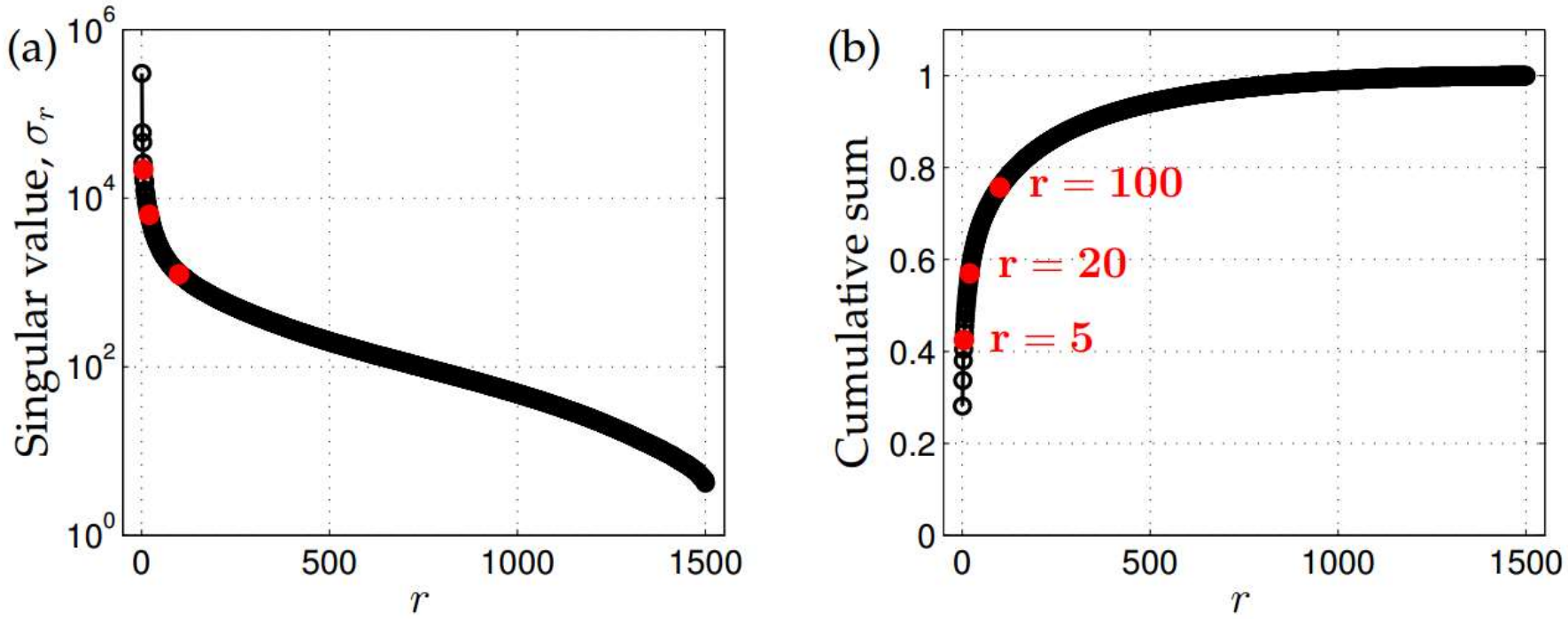


Figure 1.4: (a) Singular values σ_r and (b) cumulative sum $\sum_{k=1}^{r} \sigma_k$ of the first r singular values.

Eigenvectors and eigenvalues

Linear System of Equations

$$\mathbf{A}\mathbf{x} = \mathbf{b}.$$

The matrix $\mathbf{A} \in \mathbb{R}^{p \times n}$ and vector $\mathbf{b} \in \mathbb{R}^{p}$ are generally known, and the vector $\mathbf{x} \in \mathbb{R}^{n}$ is unknown.

Eigenvalue Equation

$$\mathbf{A}\mathbf{T} = \mathbf{T}\boldsymbol{\Lambda}.$$

The columns $\boldsymbol{\xi}_k$ of the matrix $\mathbf{T}$ are the eigenvectors of $\mathbf{A} \in \mathbb{C}^{n \times n}$ corresponding to the eigenvalue λ_k: $\mathbf{A}\boldsymbol{\xi}_k = \lambda_k \boldsymbol{\xi}_k$. The matrix $\boldsymbol{\Lambda}$ is a diagonal matrix containing these eigenvalues, in the simple case with n distinct eigenvalues.

Interpretation as Dominant Correlations

The SVD is closely related to an eigenvalue problem involving the correlation matrices $\mathbf{X}\mathbf{X}^*$ and $\mathbf{X}^*\mathbf{X}$, shown in Fig. 1.5 for a specific image, and in Figs. 1.6 and 1.7 for generic matrices. If we plug (1.3) into the row-wise correlation matrix $\mathbf{X}\mathbf{X}^*$ and the column-wise correlation matrix $\mathbf{X}^*\mathbf{X}$, we find

$$\mathbf{X}\mathbf{X}^* = \mathbf{U}\begin{bmatrix}\hat{\mathbf{\Sigma}}\\ \mathbf{0}\end{bmatrix}\mathbf{V}^*\mathbf{V}\begin{bmatrix}\hat{\mathbf{\Sigma}} & \mathbf{0}\end{bmatrix}\mathbf{U}^* = \mathbf{U}\begin{bmatrix}\hat{\mathbf{\Sigma}}^2 & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\mathbf{U}^*, \tag{1.12a}$$

$$\mathbf{X}^*\mathbf{X} = \mathbf{V}\begin{bmatrix}\hat{\mathbf{\Sigma}} & \mathbf{0}\end{bmatrix}\mathbf{U}^*\mathbf{U}\begin{bmatrix}\hat{\mathbf{\Sigma}}\\ \mathbf{0}\end{bmatrix}\mathbf{V}^* = \mathbf{V}\hat{\mathbf{\Sigma}}^2\mathbf{V}^*. \tag{1.12b}$$

Recalling that $\mathbf{U}$ and $\mathbf{V}$ are unitary, $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{V}$ are solutions to the following eigenvalue problems:

$$\mathbf{X}\mathbf{X}^*\mathbf{U} = \mathbf{U}\begin{bmatrix}\hat{\mathbf{\Sigma}}^2 & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}, \tag{1.13a}$$

$$\mathbf{X}^*\mathbf{X}\mathbf{V} = \mathbf{V}\hat{\mathbf{\Sigma}}^2. \tag{1.13b}$$

This provides an intuitive interpretation of the SVD, where the columns of $\mathbf{U}$ are eigenvectors of the correlation matrix $\mathbf{X}\mathbf{X}^*$, and the columns of $\mathbf{V}$ are eigenvectors of $\mathbf{X}^*\mathbf{X}$. We choose to arrange the singular values in descending order by magnitude, and thus the columns of $\mathbf{U}$ are hierarchically ordered by how much correlation they capture in the columns of $\mathbf{X}$; similarly $\mathbf{V}$ captures correlation in the rows of $\mathbf{X}$.

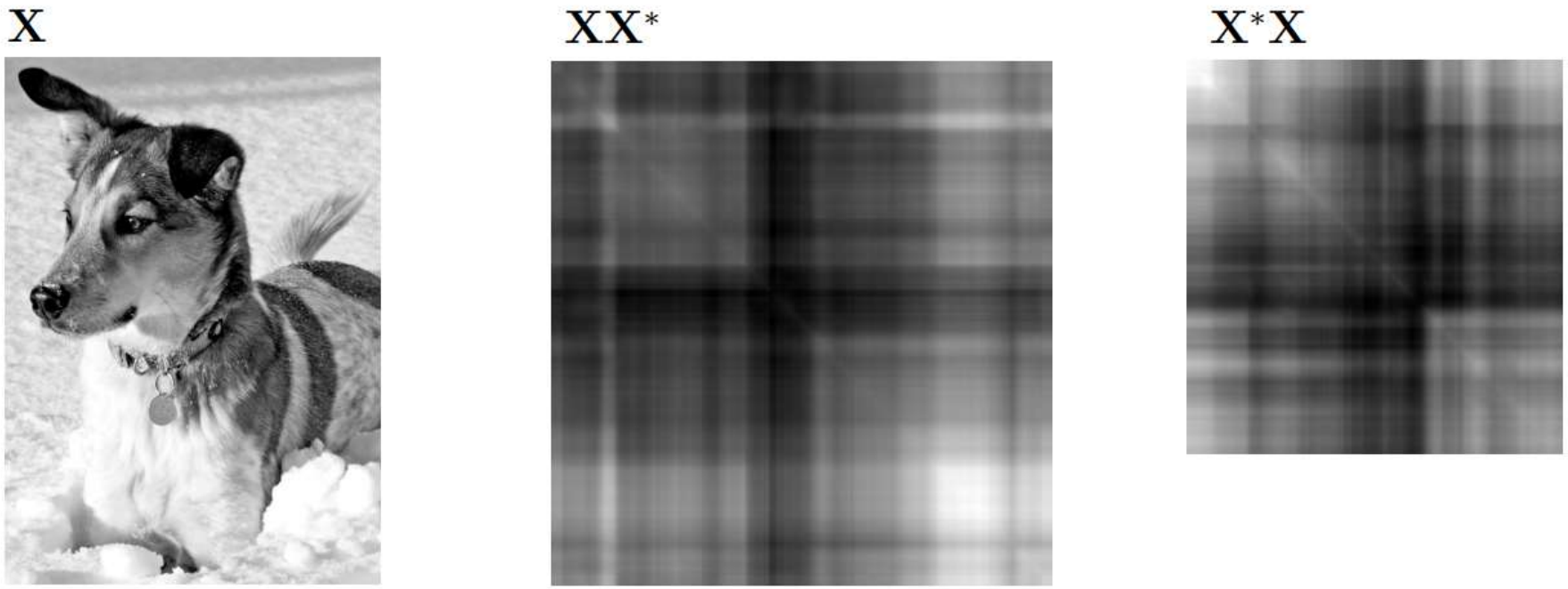


Figure 1.5: Correlation matrices $\mathbf{X}\mathbf{X}^*$ and $\mathbf{X}^*\mathbf{X}$ for a matrix $\mathbf{X}$ obtained from an image of a dog. Note that both correlation matrices are symmetric.

Example of using SVD for embedding

- For eigenfaces, the features are the PCA modes generated by the SVD. Thus each PCA mode is high-dimensional, but the only quantity of importance in feature space is the weight of that particular mode in representing a given face.
- If one performs an r-rank truncation, then any face needs only r features to represent it in feature space. This ultimately gives a low-rank embedding of the data in an interpretable set of r features that can be leveraged for diagnostics, prediction, reconstruction, and/or control.

Principal component analysis (PCA)

- PCA pre-processes the data by mean subtraction and setting the variance to unity before performing the SVD.
- The geometry of the resulting coordinate system is determined by principal components (PCs) that are uncorrelated (orthogonal) to each other, but have maximal correlation with the measurements.

Computation

We now compute the average row $\bar{\mathbf{x}}$ (i.e., the mean of all rows), and subtract it from $\mathbf{X}$. The mean $\bar{\mathbf{x}}$ is given by

$$\bar{\mathbf{x}}_j = \frac{1}{n}\sum_{i=1}^{n} \mathbf{X}_{ij}, \tag{1.36}$$

and the mean matrix is

$$\bar{\mathbf{X}} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \bar{\mathbf{x}}. \tag{1.37}$$

Subtracting $\bar{\mathbf{X}}$ from $\mathbf{X}$ results in the mean-subtracted data $\mathbf{B}$:

$$\mathbf{B} = \mathbf{X} - \bar{\mathbf{X}}. \tag{1.38}$$

The covariance matrix of $\mathbf{B}$ is given by

$$\mathbf{C} = \frac{1}{n-1}\mathbf{B}^*\mathbf{B}. \tag{1.39}$$

Note that the covariance is normalized by $n-1$ instead of n, even though there are n sample points. This is known as Bessel's correction, which compensates

for the fact that the sample variance is biased because it does not capture the variance of the sample mean $\bar{\mathbf{X}}$ about the true mean. The covariance matrix $\mathbf{C}$ is symmetric and positive semi-definite, having non-negative real eigenvalues. Each entry C_{ij} quantifies the correlation of the i and j features across all experiments.

The principal components are the eigenvectors of $\mathbf{C}$, and they define a change of coordinates in which the covariance matrix is diagonal:

$$\mathbf{CV} = \mathbf{VD} \quad \Longrightarrow \quad \mathbf{C} = \mathbf{VDV}^* \quad \Longrightarrow \quad \mathbf{D} = \mathbf{V}^*\mathbf{CV}. \tag{1.40}$$

The columns of the eigenvector matrix $\mathbf{V}$ are the principal components, and the elements of the diagonal matrix $\mathbf{D}$ are the variances of the data along these directions. This transformation is guaranteed to exist, since $\mathbf{C}$ is Hermitian and the columns of $\mathbf{V}$ are orthonormal. In these principal component coordinates, all features are linearly uncorrelated with each other.

The matrix of principal components $\mathbf{V}$ is also the matrix of right singular vectors of $\mathbf{B}$. Substituting $\mathbf{B} = \mathbf{U\Sigma V}^*$ into (1.39) and comparing with (1.40) yields

$$\mathbf{C} = \frac{1}{n-1}\mathbf{B}^*\mathbf{B} = \frac{1}{n-1}\mathbf{V\Sigma}^2\mathbf{V}^* \quad \Longrightarrow \quad \mathbf{D} = \frac{1}{n-1}\mathbf{\Sigma}^2. \tag{1.41}$$

The variance of the data in these coordinates, given by the diagonal elements λ_k of $\mathbf{D}$, is related to the singular values as

$$\lambda_k = \frac{\sigma_k^2}{n-1}. \tag{1.42}$$

Thus, the SVD provides a numerically robust approach for computing the principal components. An approximation $\tilde{\mathbf{B}}$ obtained by keeping only the first r principal components will have a missing variance related to the squared Frobenius norm error in (1.7).

Two examples

Noisy Gaussian Data

Ovarian Cancer Data

Eigenfaces

- PCA (i.e., SVD on mean-subtracted data)
- Dataset

Figure 1.16: (left) A single image for each person in the Yale database, and (right) all images for a specific person. Left panel generated by Code 1.6.

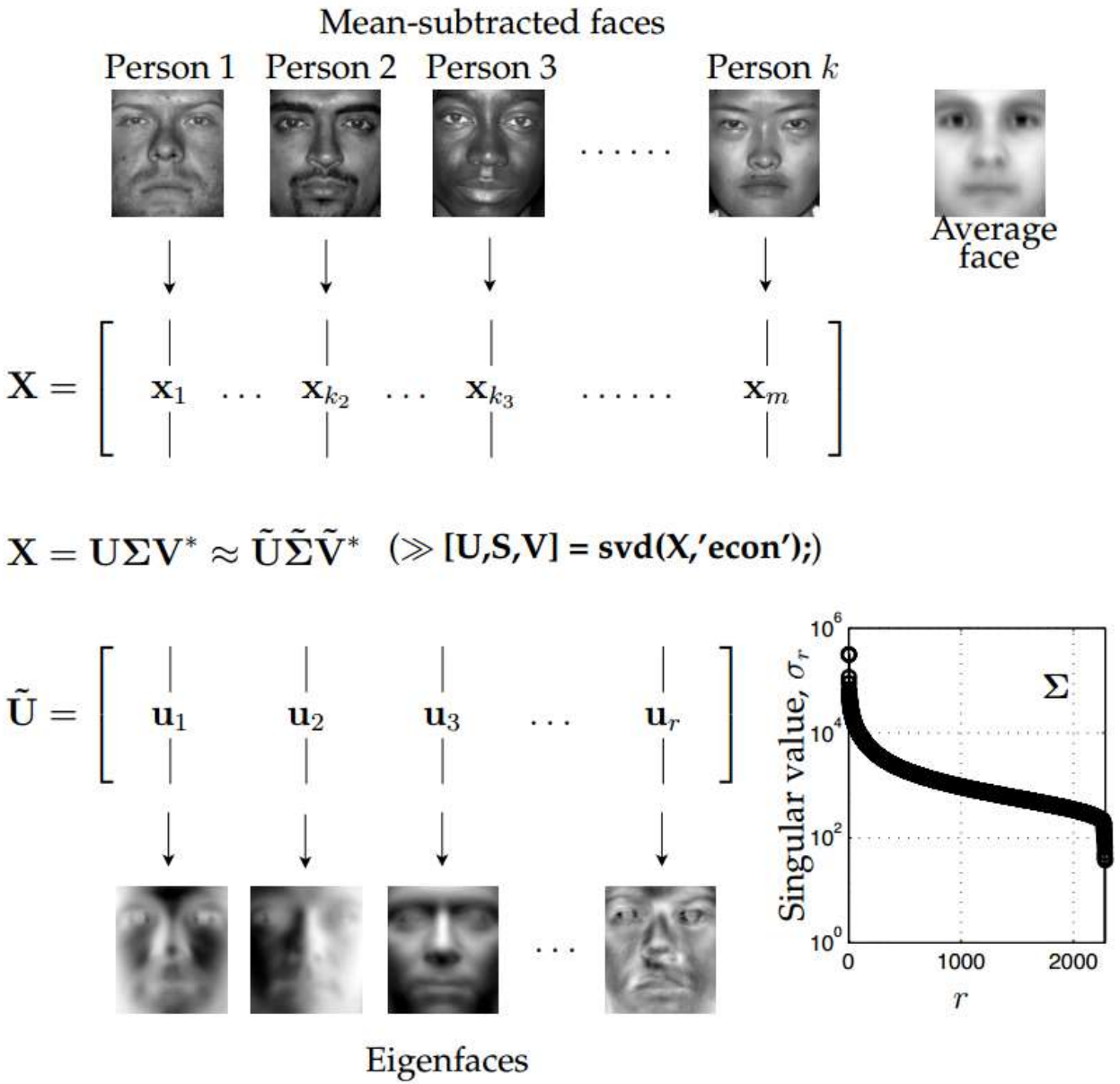


Figure 1.17: Schematic procedure to obtain eigenfaces from library of faces $\mathbf{X}$ after subtracting off average face $\bar{\mathbf{X}}$.

Figure 1.18: Approximate representation of test image using eigenfaces basis of various order r. Test image is not in training set.

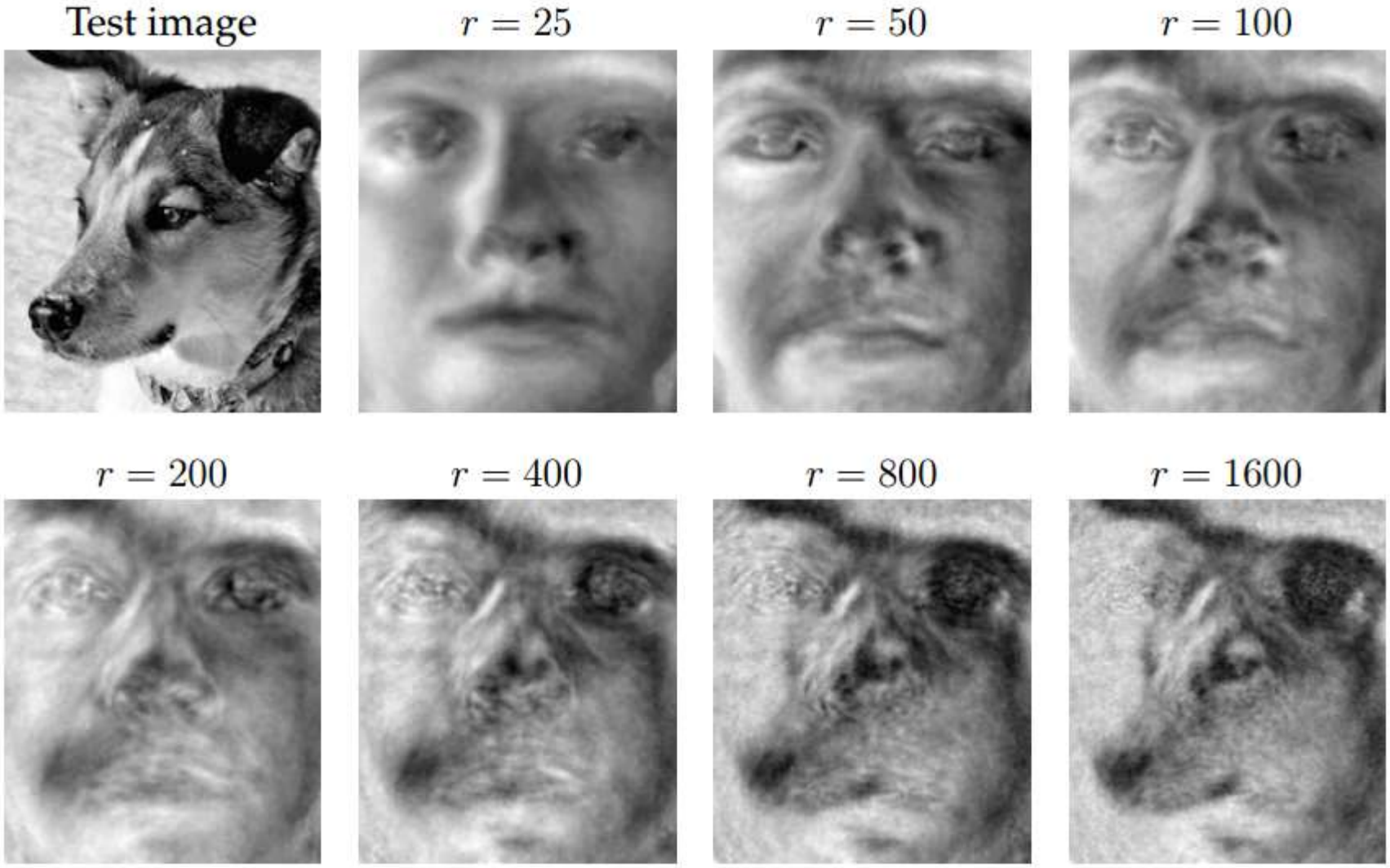


Figure 1.19: Approximate representation of an image of a dog using eigenfaces.

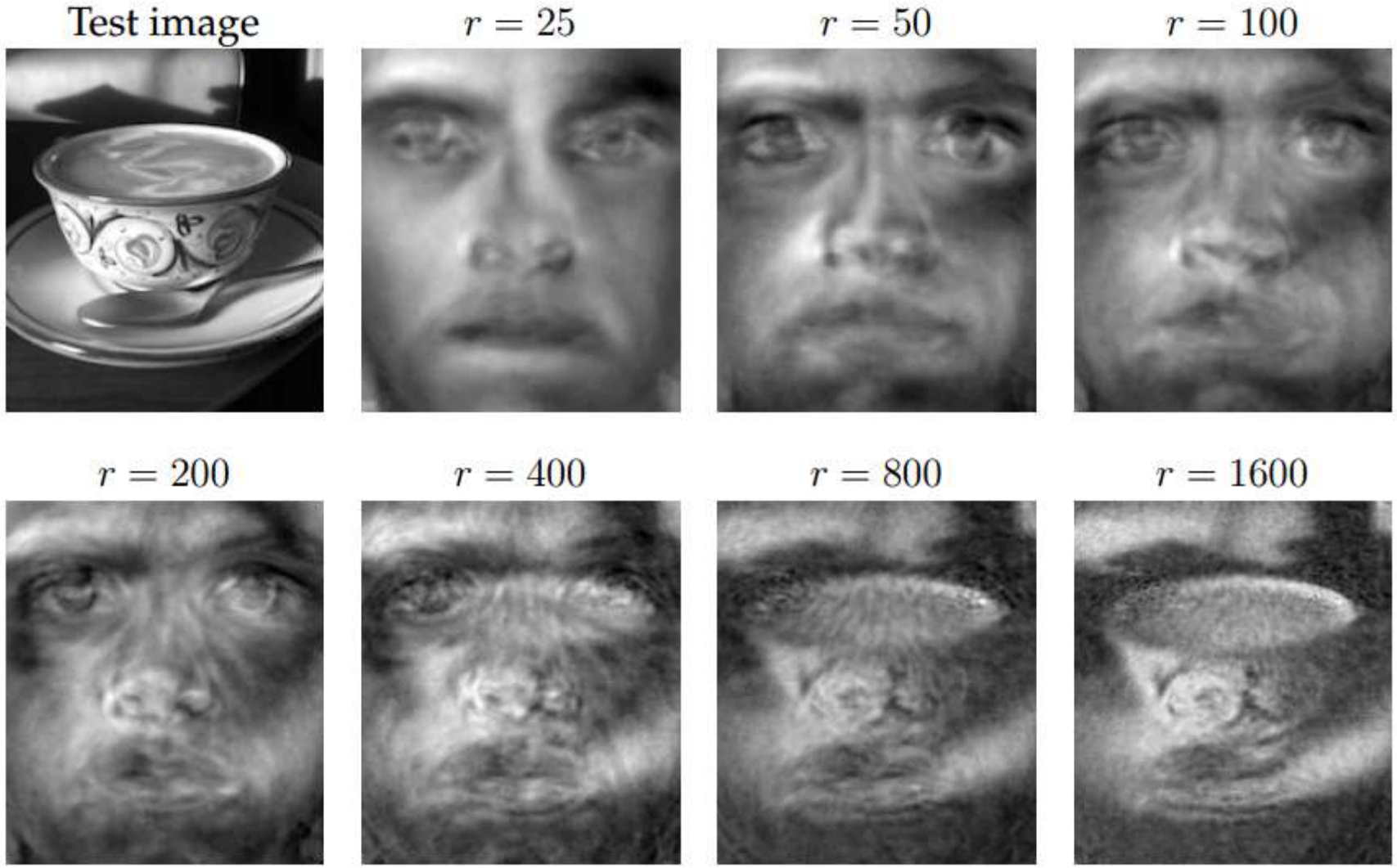


Figure 1.20: Approximate representation of a cappuccino using eigenfaces.

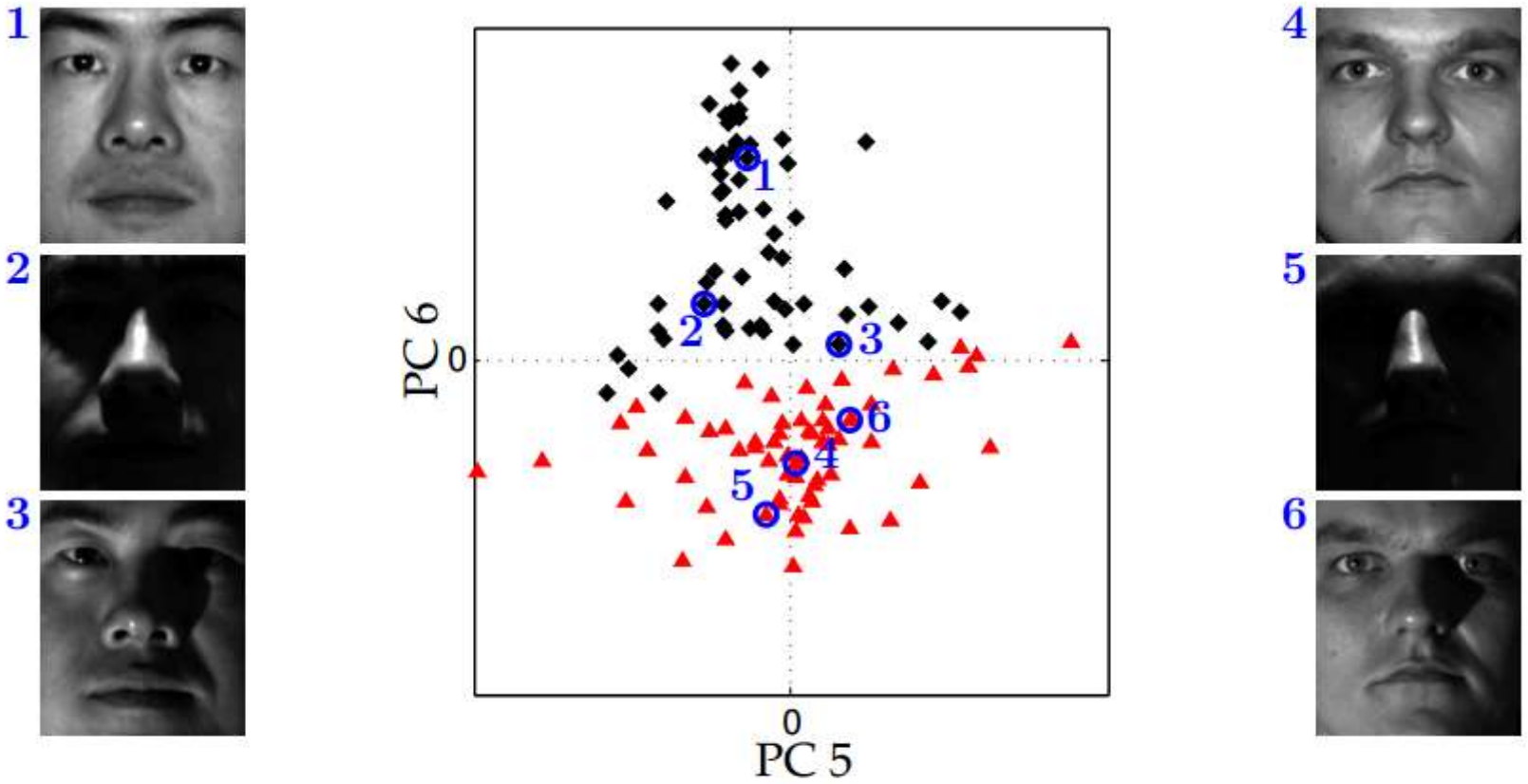


Figure 1.21: Projection of all images from two individuals onto the 5th and 6th PCA modes. Projected images of the first individual are indicated with black diamonds, and projected images of the second individual are indicated with red triangles. Three examples from each individual are circled in blue, and the corresponding image is shown.

26.4.3 Supervised and unsupervised learning in robot perception

Machine learning plays an important role in robot perception. This is particularly the case when the best internal representation is not known. One common approach is to map high-dimensional sensor streams into lower-dimensional spaces using unsupervised machine learning methods (see Chapter 19). Such an approach is called **low-dimensional embedding**. Machine learning makes it possible to learn sensor and motion models from data, while simultaneously discovering a suitable internal representation.

More on embedding

- Machine learning is based upon optimization techniques for data. The goal is to find both a low-rank subspace for optimally embedding the data, for clustering and classification as well as regression methods of different data types.
- Machine learning thus provides a principled set of mathematical methods for extracting meaningful features from data, i.e., data mining, as well as binning the data into distinct and meaningful patterns that can be exploited for decision making.

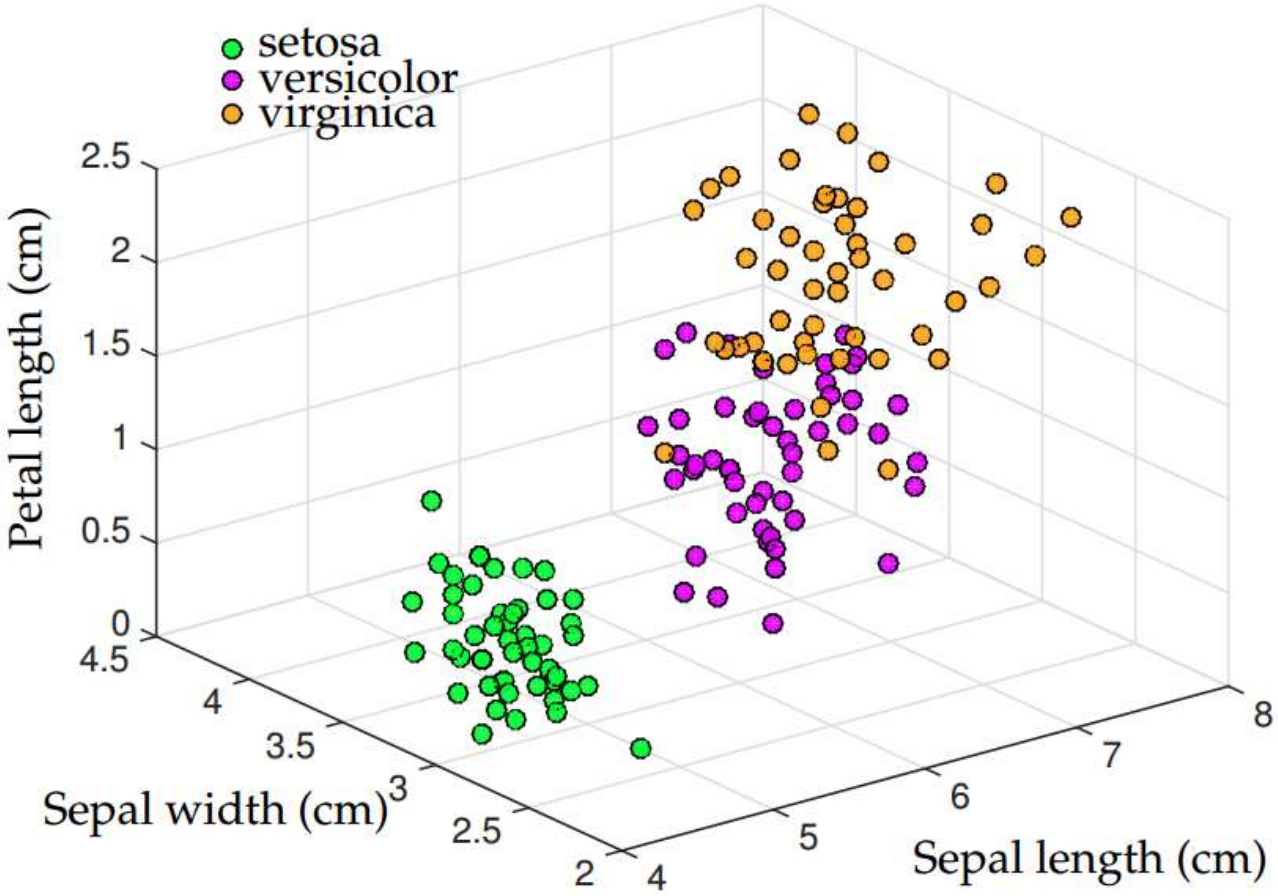


Figure 5.1: Fisher iris data set with 150 measurements over three varieties, including 50 measurements each of *Iris setosa*, *I. versicolor*, and *I. virginica*. Each flower includes a measurement of sepal length, sepal width, petal length, and petal width. The first three of these are illustrated here, showing that these simple biological features are sufficient to show that the data has distinct, quantifiable differences between the species.

Figure 5.2: Example images of dogs (left) and cats (right). Our goal is to construct a feature space where automated classification of these images can be efficiently computed.

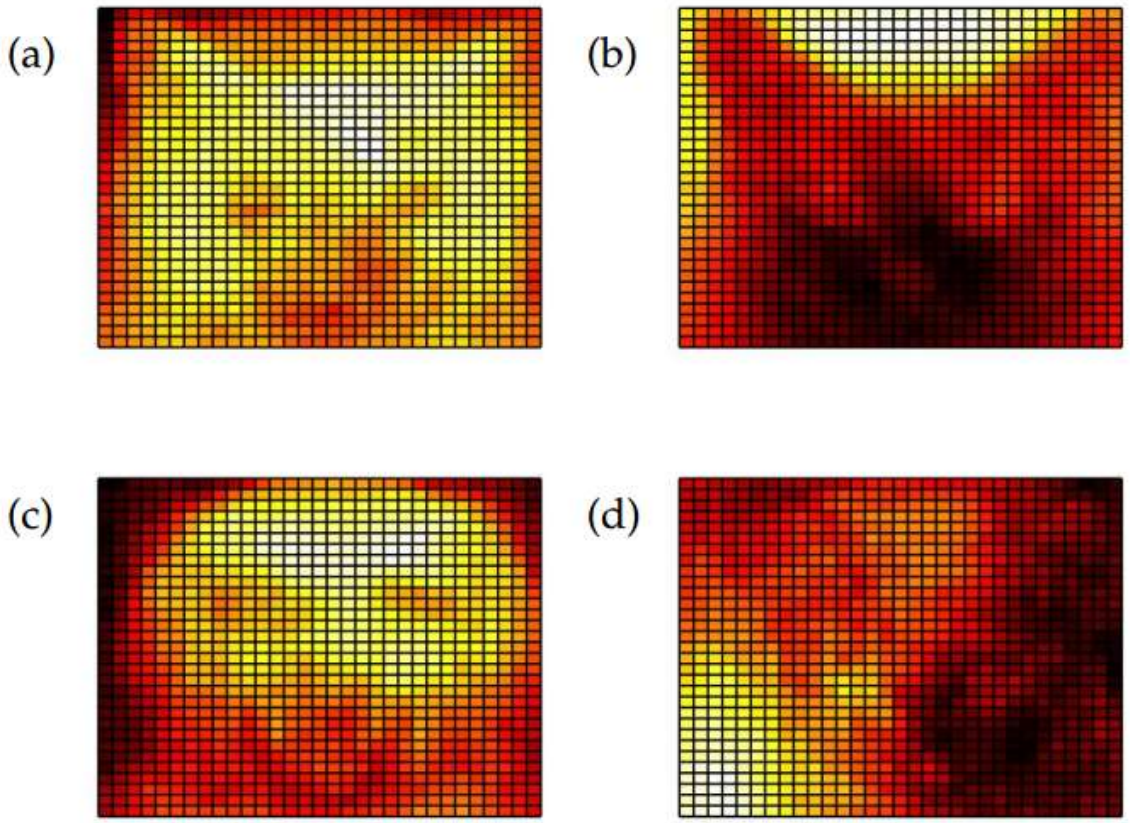


Figure 5.3: First four features (a)–(d) generated from the SVD of the 160 images of dogs and cats, i.e., these are the first four columns of the **U** matrix of the SVD. Typical cat and dog images are shown in Fig. 5.2. Note that the first two modes (a) and (b) show that the triangular ears are important features when images are correlated. This is certainly a distinguishing feature for cats, while dogs tend to lack this feature. Thus, in feature space, cats generally add these two dominant modes to promote this feature, while dogs tend to subtract these features to remove the triangular ears from their representation.

t-distributed stochastic neighbor embedding (t-SNE)

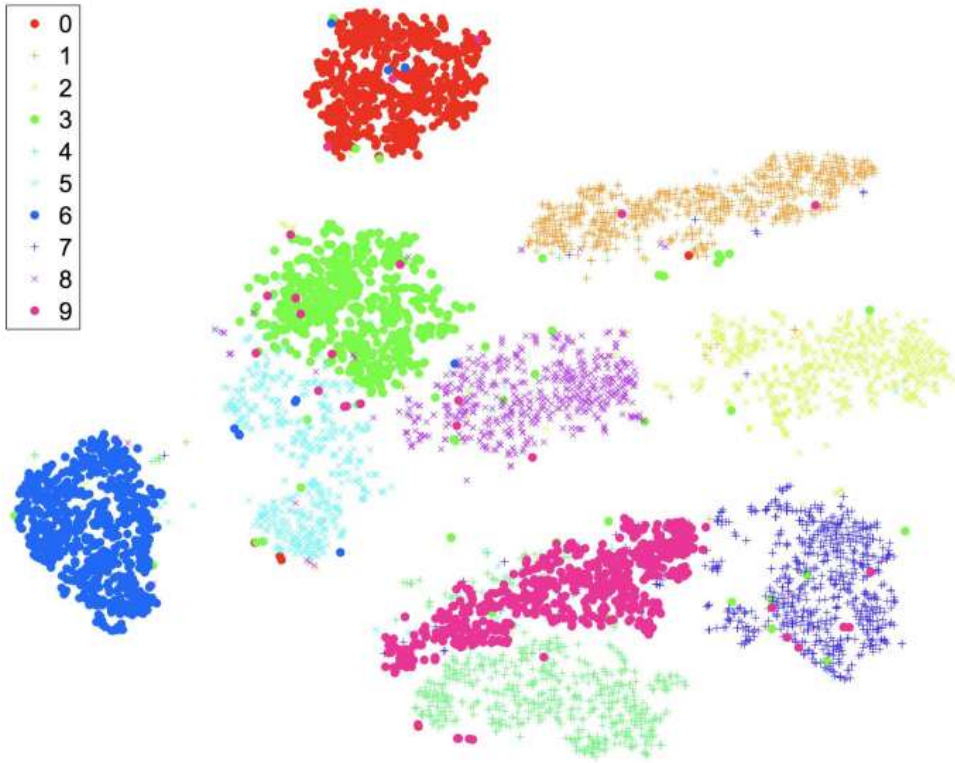


Figure 19.27 A two-dimensional t-SNE map of the MNIST data set, a collection of 60,000 images of handwritten digits, each 28 × 28 pixels and thus 784 dimensions. You can clearly see clusters for the ten digits, with a few confusions in each cluster; for example the top cluster is for the digit 0, but within the bounds of the cluster are a few data points representing the digits 3 and 6. The t-SNE algorithm finds a representation that accentuates the differences between clusters.

Autoencoders

- Autoencoder neural networks are a flexible and advantageous structure for exploiting low-dimensional features in high-dimensional data.
- They are featured here since many scientific and engineering applications leverage lowdimensional coordinate systems for building parsimonious models characterizing a physical process.
- The autoencoder generalizes the linear subspace embedding of SVD/PCA to a nonlinear manifold embedding, often of a lower dimension.

Specifically, the autoencoder maps the original high-dimensional input vectors $\mathbf{x}_j \in \mathbb{R}^n$ to a low-dimensional latent variable $\mathbf{z}_j \in \mathbb{R}^r$ and then back to the high-dimensional space $\tilde{\mathbf{x}}$, which is technically the output $\mathbf{y}$. The goal of the autoencoder is to map the output back to itself, i.e., $\|\tilde{\mathbf{x}} - \mathbf{x}\|_2 \approx 0$. Typically $r \ll n$ for autoencoding and mathematically

$$\mathbf{Z} = \phi(\mathbf{X}), \tag{6.37}$$

where $\mathbf{Z}$ is the latent space data and $\mathbf{X}$ is the input high-dimensional data. Note that the columns of $\mathbf{Z}$ are $\mathbf{z}_j$ and the columns of $\mathbf{X}$ are $\mathbf{x}_j$. Decoding is represented as

$$\tilde{\mathbf{X}} = \psi(\mathbf{Z}), \tag{6.38}$$

where the neural network weights are optimized so that the output $\tilde{\mathbf{X}}$ is as close as possible to the input,

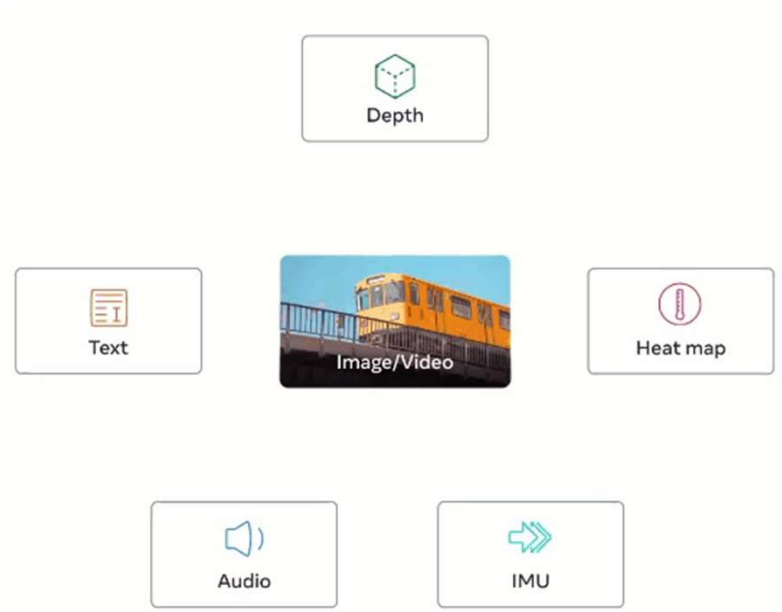
$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}} \|\mathbf{X} - \tilde{\mathbf{X}}\|_2^2 = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \|\mathbf{X} - \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{X})\|_2^2, \tag{6.39}$$

where $\boldsymbol{\theta}$ are the weights of the autoencoder network $\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}) = \psi(\phi(\mathbf{x}))$. The dimension of the latent space r is often determined by hyperparameter tuning. Thus r is made as small as possible until the autoencoder performance starts to fail. This is often informative, as it can discover the intrinsic dimensionality of the data.

HW1

https://hackmd.io/@CiqLOooyRwWmK--mMkfetA/H1rrtnthh

IMAGEBIND



- Multimodal
- Cross-modal retrieval
- Semantics
- Audio-to-Image generation
- Joint embedding
- Six different modalities: images, text, audio, depth, thermal, and IMU data

ImageBind by Meta AI (metademolab.com)

Joint embedding space

- All combinations of paired data are not necessary to train such a joint embedding, and only image-paired data is sufficient to bind the modalities together

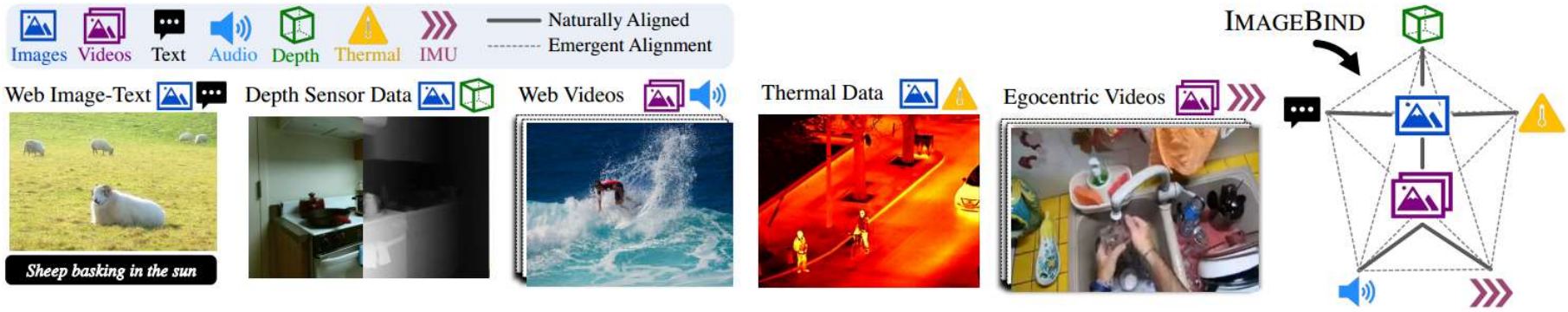


Figure 2. IMAGEBIND overview. Different modalities occur naturally aligned in different data sources, for instance images+text and video+audio in web data, depth or thermal information with images, IMU data in videos captured with egocentric cameras, *etc*. IMAGEBIND links all these modalities in a common embedding space, enabling new emergent alignments and capabilities.

[2305.05665] ImageBind: One Embedding Space To Bind Them All (arxiv.org)

Zero-shot learning

- Zero-shot learning (ZSL) is a problem setup in deep learning where, at test time, a learner observes samples from classes which were not observed during training, and needs to predict the class that they belong to.
- Zero-shot methods generally work by associating observed and non-observed classes through some form of auxiliary information, which encodes observable distinguishing properties of objects.
  - For example, given a set of images of animals to be classified, along with auxiliary textual descriptions of what animals look like, an artificial intelligence model which has been trained to recognize horses, but has never been given a zebra, can still recognize a zebra when it also knows that zebras look like striped horses.
- This problem is widely studied in computer vision, natural language processing, and machine perception.

[2305.05665] ImageBind: One Embedding Space To Bind Them All (arxiv.org)

Binding modalities with images

IMAGEBIND uses pairs of modalities $(\mathcal{I}, \mathcal{M})$, where $\mathcal{I}$ represents images and $\mathcal{M}$ is another modality, to learn a single joint embedding. We use large-scale web datasets with (image, text) pairings that span a wide range of semantic concepts. Additionally, we use the natural, self-supervised pairing of other modalities – audio, depth, thermal, and Intertial Measurement Unit (IMU) – with images.

Consider the pair of modalities $(\mathcal{I}, \mathcal{M})$ with aligned observations. Given an image $\mathbf{I}_i$ and its corresponding observation in the other modality $\mathbf{M}_i$, we encode them into normalized embeddings: $\mathbf{q}_i = f(\mathbf{I}_i)$ and $\mathbf{k}_i = g(\mathbf{M}_i)$ where f, g are deep networks. The embeddings and the encoders

[2305.05665] ImageBind: One Embedding Space To Bind Them All (arxiv.org)

Binding modalities with images (cont.)

are optimized using an InfoNCE [54] loss:

$$L_{\mathcal{I},\mathcal{M}} = -\log \frac{\exp(\mathbf{q}_i^\intercal \mathbf{k}_i/\tau)}{\exp(\mathbf{q}_i^\intercal \mathbf{k}_i/\tau) + \sum_{j\neq i} \exp(\mathbf{q}_i^\intercal \mathbf{k}_j/\tau)}, \quad (1)$$

where τ is a scalar temperature that controls the smoothness of the softmax distribution and j denotes unrelated observations, also called 'negatives'. We follow [76] and consider every example $j \neq i$ in the mini-batch to be a negative. The loss makes the embeddings $\mathbf{q}_i$ and $\mathbf{k}_i$ closer in the joint embedding space, and thus aligns $\mathcal{I}$ and $\mathcal{M}$. In practice, we use a symmetric loss $L_{\mathcal{I},\mathcal{M}} + L_{\mathcal{M},\mathcal{I}}$.

[2305.05665] ImageBind: One Embedding Space To Bind Them All (arxiv.org)