LEC 2: Principal Component Analysis (PCA) – A First Dimensionality Reduction Approach

Guangliang Chen

September 12, 2018

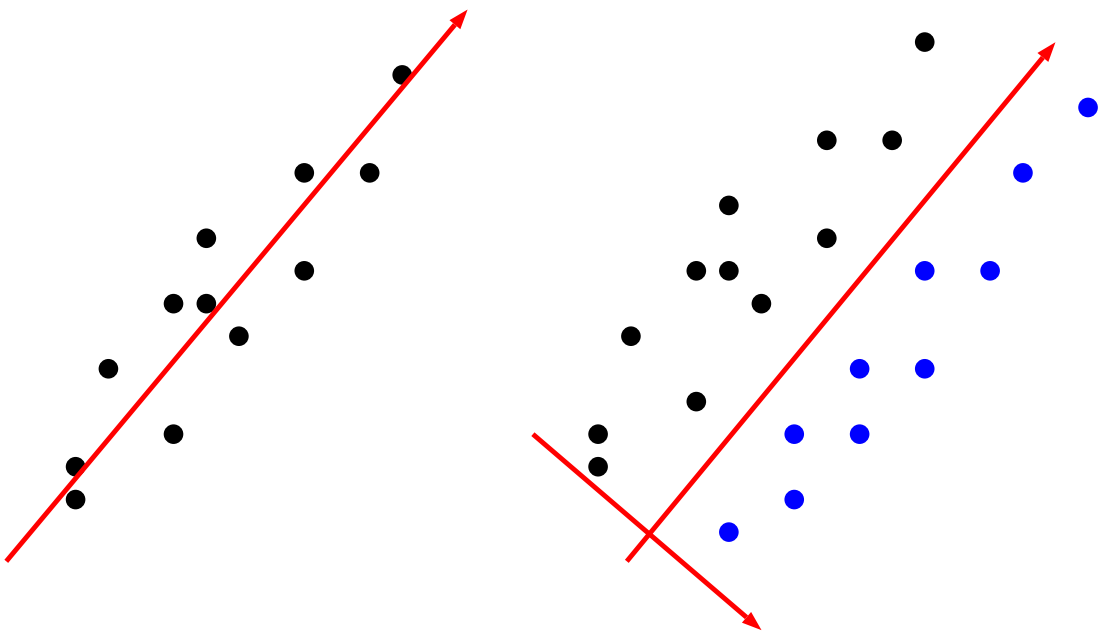
Outline

Motivation

- The MNIST handwritten digits are 784 dimensional - very time consuming to perform simple tasks like kNN search and kNN classification (Recall the $\mathcal{O}(ndk)$ complexity)
- So we need a way to reduce the dimensionality of the data in order to increase speed
- However, if we discard some dimensions, will that degrade the performance?
- The answer can be no (as long as we do it carefully). In fact, it may even improve results in many cases (e.g. owing to noise suppression)

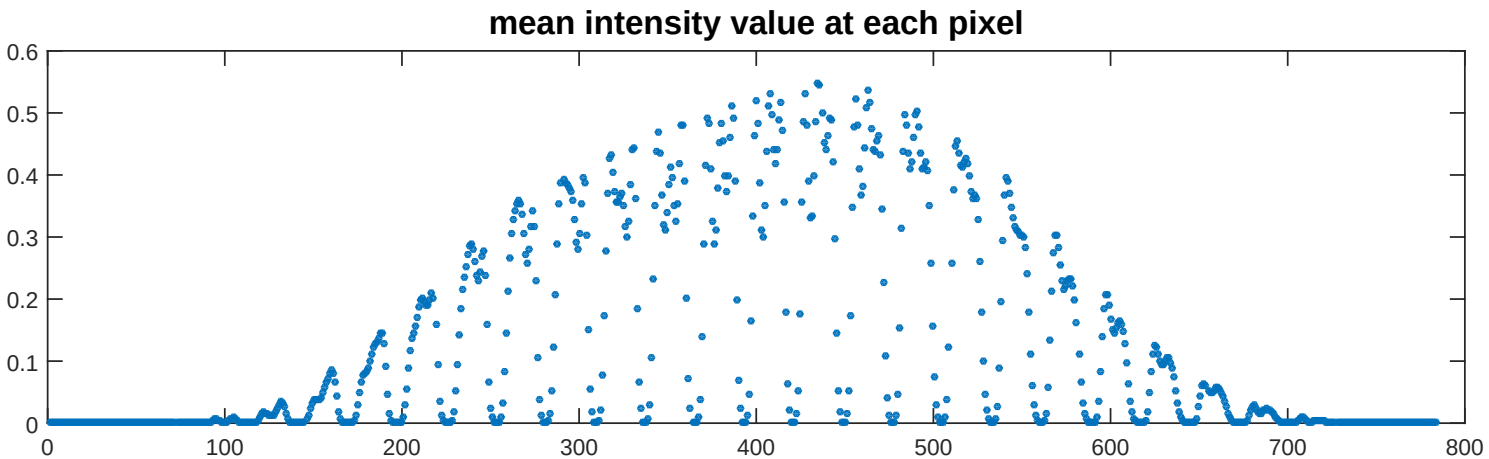
An important observation

"Useful" information of a data set is often contained in only a small number of dimensions.



A real data set

Pixelwise Average Intensity of the MNIST digits:

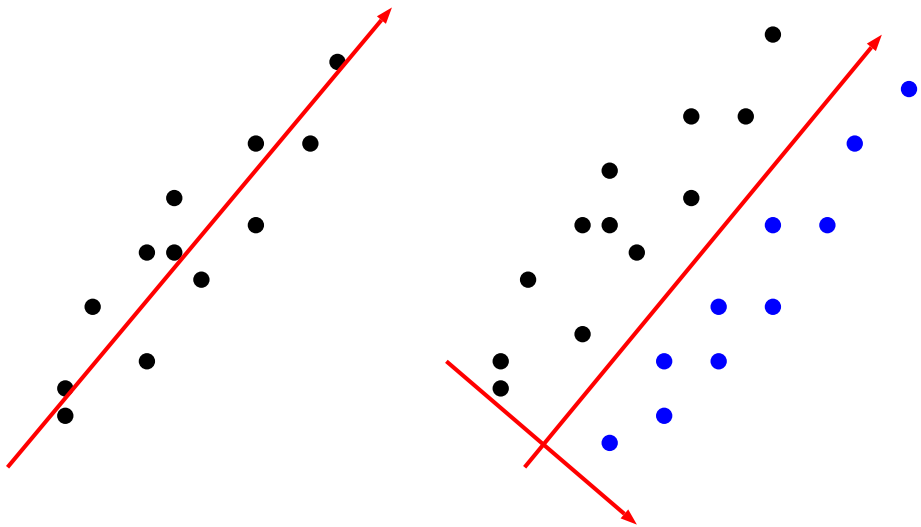


- Boundary pixels tend to be zero;
- The number of degrees of freedom of each digit is much less than 784.

Dimensionality reduction methods to be covered

– **Principal Component Analysis (PCA)**: preserves variance information (unsupervised)

– **Linear Discriminatory Analysis (LDA)**: preserves only discriminatory information between classes (supervised)



I will also introduce their two dimensional variants for matrix data, i.e., **2DPCA** and **2DLDA**.

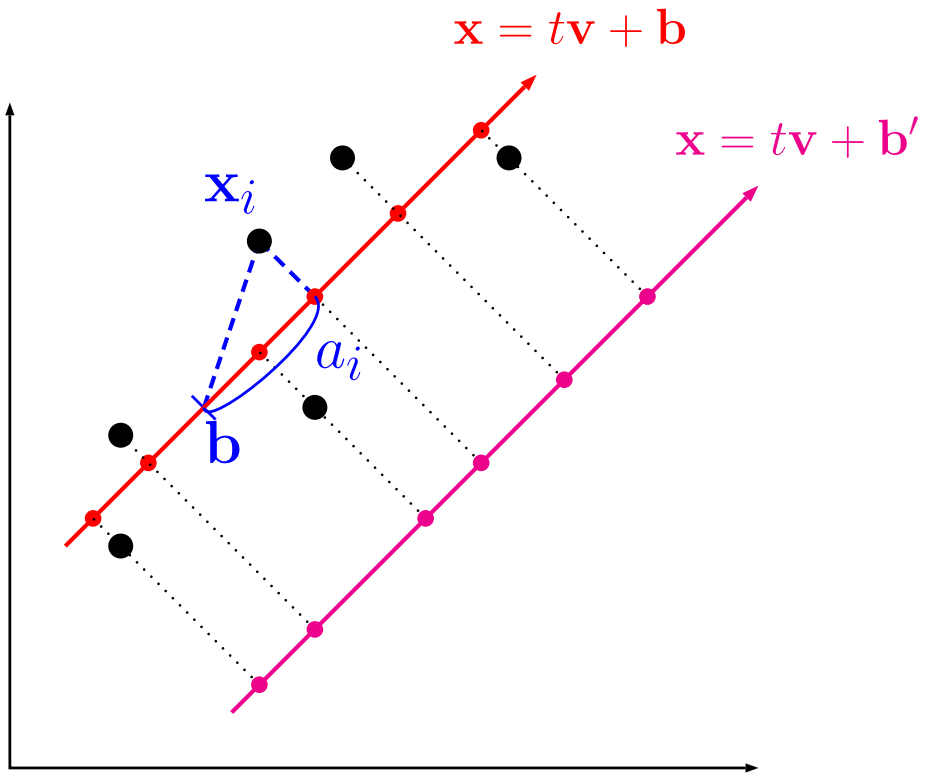
The one-dimensional PCA problem

Given data $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$, find a parametric line $\mathbf{x}(t) = t\mathbf{v} + \mathbf{b}$ defined by certain vectors $\mathbf{v}, \mathbf{b} \in \mathbb{R}^D$ (with $\|\mathbf{v}\| = 1$) such that the 1D projection (coefficients)

$$a_i = \mathbf{v}^T \cdot (\mathbf{x}_i - \mathbf{b}), \quad 1 \le i \le n$$

has the largest possible variance.

Note: For parallel lines, the projections are different, but their variance is the same! $\longleftarrow$ This implies that the solution is not unique.



Mathematical formulation

To make the solution unique, we add a constraint to the problem by requiring that

$$0 = \bar{a} = \frac{1}{n}\sum a_i = \mathbf{v}^T \cdot \frac{1}{n}\sum(\mathbf{x}_i - \mathbf{b}) = \mathbf{v}^T \cdot (\bar{\mathbf{x}} - \mathbf{b})$$

This yields that $\mathbf{b} = \bar{\mathbf{x}}$.

Therefore, with such a fixed choice of $\mathbf{b}$, we have eliminated the free parameter $\mathbf{b}$, so that we only need to find the optimal $\mathbf{v}$.

Since we now have $\bar{a} = 0$, the variance of the projections is simply $\frac{1}{n-1}\sum a_i^2$ and we can correspondingly reformulate the original problem as follows:

$$\max_{\mathbf{v}:\|\mathbf{v}\|=1} \sum a_i^2, \qquad \text{where} \quad a_i = \mathbf{v}^T \cdot (\mathbf{x}_i - \bar{\mathbf{x}}).$$

Let us further rewrite the objective function:

$$\begin{aligned}\sum a_i^2 &= \sum \underbrace{\mathbf{v}^T(\mathbf{x}_i - \bar{\mathbf{x}})}_{a_i} \underbrace{(\mathbf{x}_i - \bar{\mathbf{x}})^T\mathbf{v}}_{a_i} \\ &= \sum \mathbf{v}^T \left((\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T\right)\mathbf{v} \\ &= \mathbf{v}^T \underbrace{\left(\sum (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T\right)}_{:=\mathbf{C}(d\times d \text{ matrix})} \mathbf{v}\end{aligned}$$

Remark. $\mathbf{C}$ is called the **sample covariance matrix**, or the **scatter matrix**, of the data.

Accordingly, we have obtained the following problem

$$\max_{\mathbf{v}:\|\mathbf{v}\|=1} \mathbf{v}^T\mathbf{C}\mathbf{v} \quad \longleftarrow \text{Where did we see this problem?}$$

About the matrix $\mathbf{C}$

Let $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_n^T \end{bmatrix} \in \mathbb{R}^{n\times d}$ and $\widetilde{\mathbf{X}} = \begin{bmatrix} \widetilde{\mathbf{x}}_1^T \\ \vdots \\ \widetilde{\mathbf{x}}_n^T \end{bmatrix} \in \mathbb{R}^{n\times d}$ (where $\widetilde{\mathbf{x}}_i = \mathbf{x}_i - \bar{\mathbf{x}}$) be the original and centered data matrices (rows are data points $\longleftarrow$ MATLAB convention).

Then

$$\mathbf{C} = \sum \widetilde{\mathbf{x}}_i \widetilde{\mathbf{x}}_i^T = [\widetilde{\mathbf{x}}_1 \dots \widetilde{\mathbf{x}}_n] \cdot \begin{bmatrix} \widetilde{\mathbf{x}}_1^T \\ \vdots \\ \widetilde{\mathbf{x}}_n^T \end{bmatrix} = \widetilde{\mathbf{X}}^T \widetilde{\mathbf{X}}.$$

This shows that $\mathbf{C}$ is square, symmetric and positive semidefinite and thus only has nonnegative eigenvalues.

Result

Theorem 0.1. *Given a set of points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in $\mathbb{R}^d$ with centroid $\bar{\mathbf{x}} = \frac{1}{n}\sum \mathbf{x}_i$, the optimal direction for projecting the data (in order to have maximum variance) is the largest eigenvector of the covariance matrix $\mathbf{C} = \sum(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T$:*

$$\max_{\mathbf{v}:\|\mathbf{v}\|=1} \mathbf{v}^T\mathbf{C}\mathbf{v} = \lambda_1, \qquad \text{achieved when} \quad \mathbf{v} = \mathbf{v}_1.$$

Remark. It can be similarly proved that

$$\max_{\mathbf{v}:\|\mathbf{v}\|=1, \mathbf{v}_1^T\mathbf{v}=0} \mathbf{v}^T\mathbf{C}\mathbf{v} = \lambda_2, \qquad \text{achieved when} \quad \mathbf{v} = \mathbf{v}_2;$$

$$\max_{\mathbf{v}:\|\mathbf{v}\|=1, \mathbf{v}_1^T\mathbf{v}=0, \mathbf{v}_2^T\mathbf{v}=0} \mathbf{v}^T\mathbf{C}\mathbf{v} = \lambda_3, \qquad \text{achieved when} \quad \mathbf{v} = \mathbf{v}_3.$$

This shows that $\mathbf{v}_2, \mathbf{v}_3$ etc. are the next best orthogonal directions.

It can also be shown that the projections onto different $\mathbf{v}_i$ are **uncorrelated**.

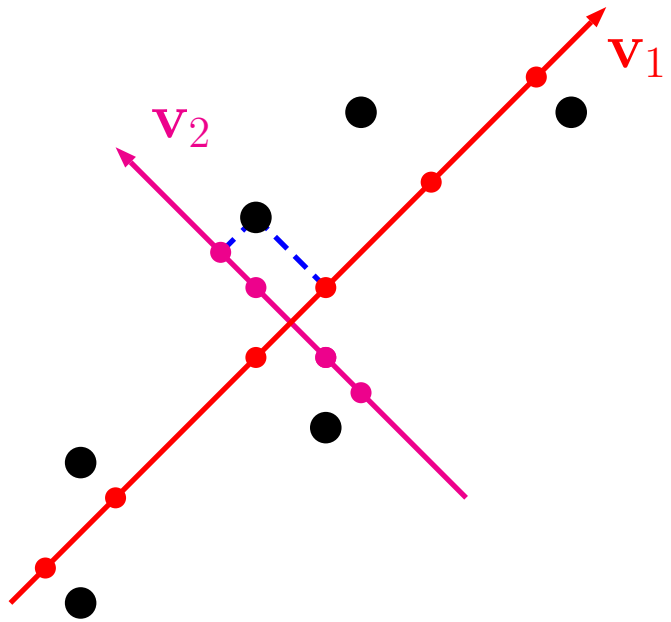
Define for each point $\mathbf{x}_i$,

$$a_i = \mathbf{v}_1^T(\mathbf{x}_i - \bar{\mathbf{x}}),$$
$$b_i = \mathbf{v}_2^T(\mathbf{x}_i - \bar{\mathbf{x}}).$$

Then their covariance is

$$\begin{aligned}\sum a_i b_i &= \sum \mathbf{v}_1^T(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T\mathbf{v}_2 \\ &= \mathbf{v}_1^T\mathbf{C}\mathbf{v}_2 = \mathbf{v}_1^T(\lambda_2\mathbf{v}_2) \\ &= \lambda_2(\mathbf{v}_1^T\mathbf{v}_2) = 0.\end{aligned}$$

This implies that the variances of the different projections add.



Therefore, for any $k \geq 1$, the top k eigenvectors of $\mathbf{C}$ span the k-D projection subspace that preserves the most variance (among all k-D subspaces), which is $\sum_{j=1}^{k} \lambda_j$.

Principal component analysis (PCA)

The previous procedure is called principal component analysis.

– $\mathbf{v}_j$ is called the j**th principal direction**;

– The projection of the data point $\mathbf{x}_i$ onto $\mathbf{v}_j$, i.e., $\mathbf{v}_j^T(\mathbf{x}_i - \bar{\mathbf{x}})$, is called the j**th principal component** of $\mathbf{x}_i$.

In fact, PCA is just a change of coordinate system to use the maximum-variance directions of the data set!

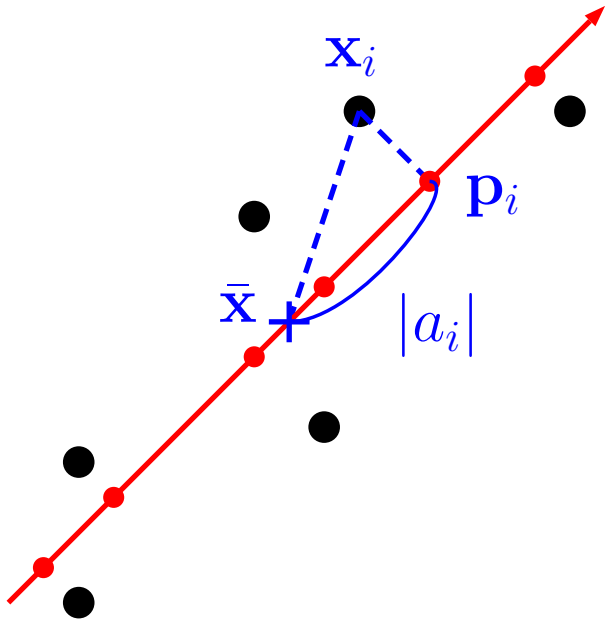
Other interpretations of PCA

PCA reduces the dimensionality of data by maximizing the variance of the projection (for a given dimension k).

It is also known that the PCA subspace

- optimally preserves the pairwise distances between the given data points, and
- minimizes the total orthogonal fitting error

among all k-dimensional subspaces.



$$\underbrace{\sum \|\mathbf{x}_i - \bar{\mathbf{x}}\|^2}_{\text{total scatter}} = \underbrace{\sum a_i^2}_{\text{variance kept}} + \underbrace{\sum \|\mathbf{x}_i - \mathbf{p}_i\|^2}_{\text{ortho. fitting error}}$$

Example 0.1. Perform PCA (by hand) on the following data set (rows are data points):

$$\mathbf{X} = \begin{pmatrix} 1 & -1 \\ -1 & 1 \\ 2 & 2 \\ -2 & -2 \end{pmatrix}.$$

Computing

PCA requires constructing a sample covariance matrix through multiplication of the (centered) data matrix $\widetilde{\mathbf{X}} \in \mathbb{R}^{n \times d}$ and its transpose:

$$\mathbf{C} = \widetilde{\mathbf{X}}^T \widetilde{\mathbf{X}} \in \mathbb{R}^{d \times d}$$

This can be a significant challenge for large data sets in high dimensions:

- The complexity of obtaining $\mathbf{C}$ from matrix multiplication is $\mathcal{O}(nd^2)$.
- It takes $\mathcal{O}(d^3)$ time to find all the eigenvalues and eigenvectors of $\mathbf{C}$.

We show in the next few slides that the eigenvectors of $\mathbf{C}$ can be efficiently computed from the Singular Value Decomposition (SVD) of $\widetilde{\mathbf{X}}$.

SVD of general matrices

Theorem: For any matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$, there exist two orthogonal matrices $\mathbf{U} \in \mathbb{R}^{n\times n}, \mathbf{V} \in \mathbb{R}^{d\times d}$ and a nonnegative "diagonal" matrix $\mathbf{\Sigma} \in \mathbb{R}^{n\times d}$ such that

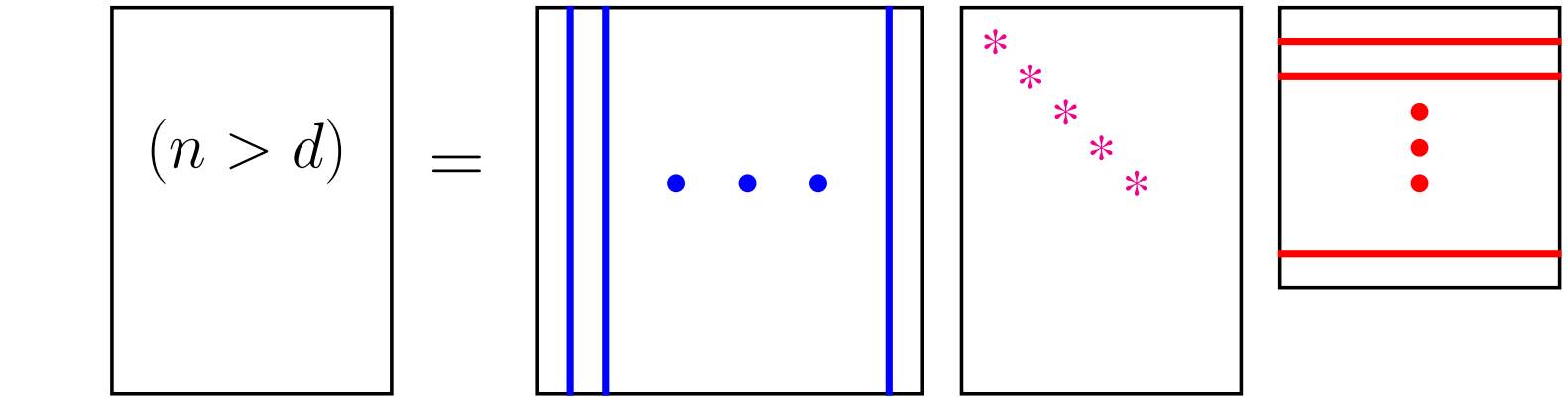
$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T.$$

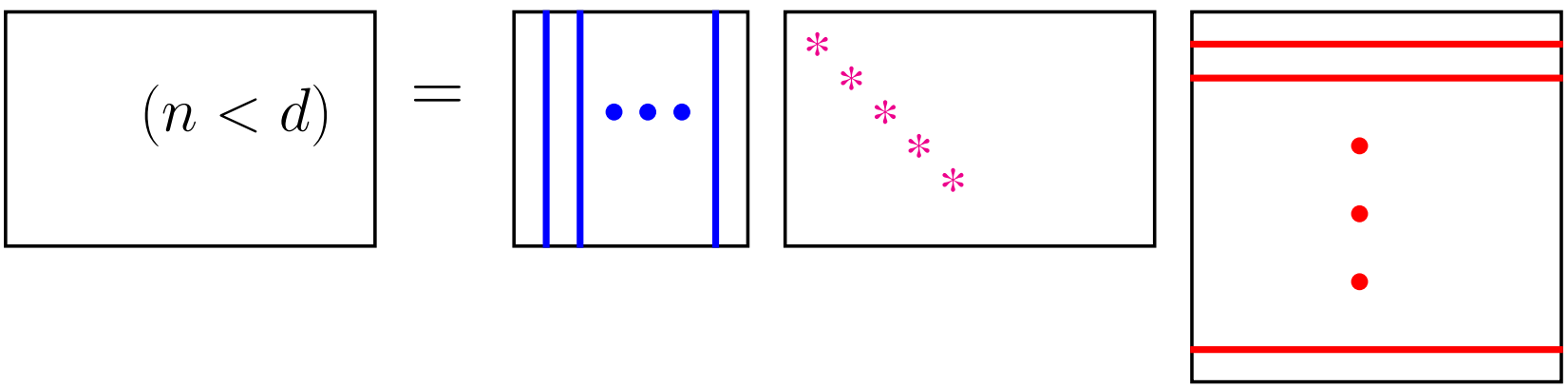
Definition 0.1. This is called the *Singular Value Decomposition (SVD)* of $\mathbf{X}$:

- The columns of $\mathbf{U}$ are called the left singular vectors of $\mathbf{X}$.
- The diagonals of $\mathbf{\Sigma}$ are called the singular values of $\mathbf{X}$ (often sorted in decreasing order).
- The columns of $\mathbf{V}$ are called the right singular vectors of $\mathbf{X}$.

Remark. *Compare with symmetric matrices:* $\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T \in \mathbb{R}^{n\times n}$.

Principal Component Analysis (PCA)





Connections to symmetric matrix decomposition

From the SVD of $\mathbf{X}$ we obtain that

$$\mathbf{X}\mathbf{X}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \cdot \mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T = \mathbf{U}\left(\mathbf{\Sigma}\mathbf{\Sigma}^T\right)\mathbf{U}^T;$$
$$\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T \cdot \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{V}\left(\mathbf{\Sigma}^T\mathbf{\Sigma}\right)\mathbf{V}^T.$$

This shows that

- $\mathbf{U}$ is the eigenvectors matrix of $\mathbf{X}\mathbf{X}^T$;
- $\mathbf{V}$ is the eigenvectors matrix of $\mathbf{X}^T\mathbf{X}$;
- The nonzero eigenvalues of $\mathbf{X}\mathbf{X}^T, \mathbf{X}^T\mathbf{X}$ (which must be the same) equal the squared singular values of $\mathbf{X}$.

Reasoning for proving the SVD theorem

Given any matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, the SVD can be thought of as solving a matrix equation for three unknowns (each from a class of matrices):

$$\mathbf{X} = \underbrace{\mathbf{U}}_{\text{orthogonal}} \cdot \underbrace{\mathbf{\Sigma}}_{\text{diagonal}} \cdot \underbrace{\mathbf{V}^T}_{\text{orthogonal}} .$$

Suppose such solutions exist.

- From previous slide:

$$\mathbf{X}^T\mathbf{X} = \mathbf{V}\left(\mathbf{\Sigma}^T\mathbf{\Sigma}\right)\mathbf{V}^T$$

  This tells us how to find $\mathbf{V}$ and $\mathbf{\Sigma}$ (eigenvectors and square roots of eigenvalues of $\mathbf{X}^T\mathbf{X}$).

- After we have found $\mathbf{V}$ and $\mathbf{\Sigma}$, rewrite the matrix equation as

$$\mathbf{XV} = \mathbf{U\Sigma},$$

or in columns,

$$\mathbf{X}[\mathbf{v}_1 \dots \mathbf{v}_d] = [\mathbf{u}_1 \dots \mathbf{u}_r\, \mathbf{u}_{r+1} \dots \mathbf{u}_n] \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \\ & & \end{bmatrix}.$$

By comparing columns, we obtain

$$\mathbf{X}\mathbf{v}_i = \sigma_i \mathbf{u}_i, \quad 1 \le i \le r$$

This tells us how to find the first r columns of $\mathbf{U}$: $\mathbf{u}_i = \frac{1}{\sigma_i}\mathbf{X}\mathbf{v}_i$.

This leads to the proof of the SVD theorem on next slide.

A brief mathematical proof

Let $\mathbf{C} = \mathbf{X}^T\mathbf{X} \in \mathbb{R}^{d\times d}$. Then $\mathbf{C}$ is square, symmetric, and positive semidefinite.

Therefore, by the Spectral Theorem, $\mathbf{C} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$ for an orthogonal $\mathbf{V} \in \mathbb{R}^{d\times d}$ and diagonal $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_d)$ with $\lambda_1 \geq \cdots \geq \lambda_r > 0 = \lambda_{r+1} = \cdots = \lambda_d$ (where $r = \mathrm{rank}(\mathbf{X}) \leq d$).

Define $\sigma_i = \sqrt{\lambda_i}$ and $\mathbf{u}_i = \frac{1}{\sigma_i}\mathbf{X}\mathbf{v}_i$ for $1 \leq i \leq r$. Then $\mathbf{u}_1, \ldots, \mathbf{u}_r$ are orthonormal vectors (verify this).

Choose $\mathbf{u}_{r+1}, \ldots, \mathbf{u}_n \in \mathbb{R}^n$ such that $\mathbf{U} = [\mathbf{u}_1 \ldots \mathbf{u}_r \mathbf{u}_{r+1} \ldots \mathbf{u}_n]$ is an orthogonal matrix. Let $\mathbf{\Sigma} \in \mathbb{R}^{n\times d}$ with the only nonzero entries being $\mathbf{\Sigma}_{ii} = \sigma_i, 1 \leq i \leq r$.

It can then be verified that $\mathbf{X}\mathbf{V} = \mathbf{U}\mathbf{\Sigma}$, or equivalently, $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$.

Example 0.2. Compute the SVD of

$$\mathbf{X} = \begin{pmatrix} 1 & -1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Answer:

$$\mathbf{X} = \begin{pmatrix} \frac{2}{\sqrt{6}} & 0 & \frac{1}{\sqrt{3}} \\ -\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{3}} \end{pmatrix} \cdot \begin{pmatrix} \sqrt{3} & \\ & 1 \end{pmatrix} \cdot \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}^T$$

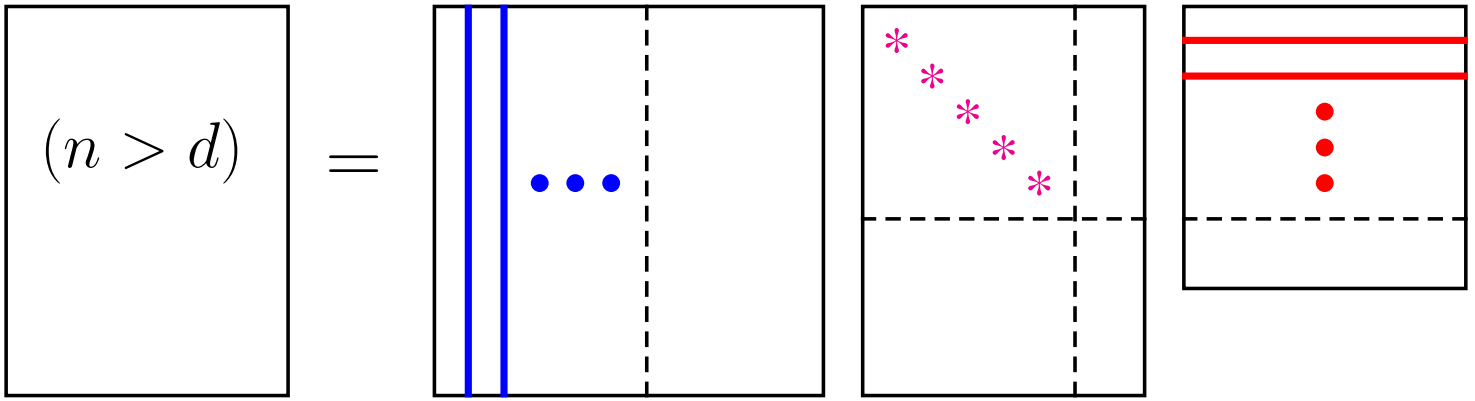
Different versions of SVD

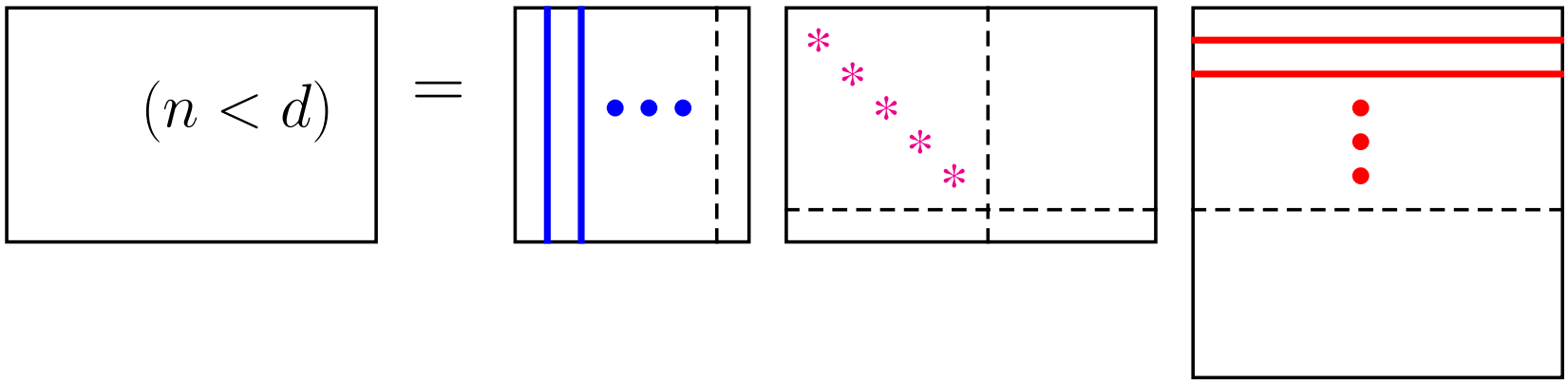
- **Full SVD**: $\mathbf{X}_{n\times d} = \mathbf{U}_{n\times n}\mathbf{\Sigma}_{n\times d}\mathbf{V}^T_{d\times d}$
- **Compact SVD**: Suppose $\text{rank}(\mathbf{X}) = r$. Define

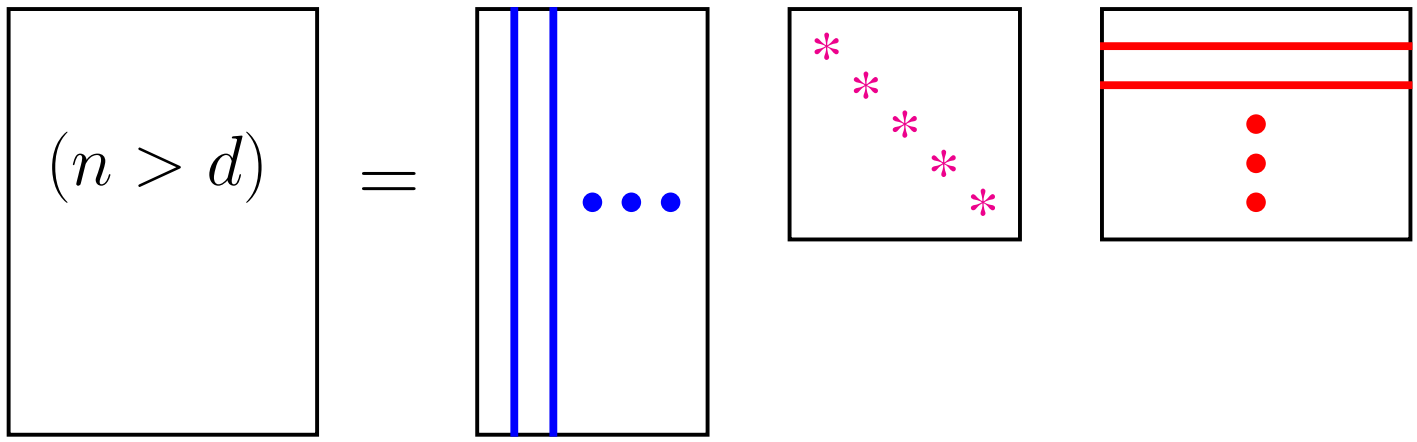
$$\mathbf{U}_r = [\mathbf{u}_1, \ldots, \mathbf{u}_r] \in \mathbb{R}^{n\times r}$$
$$\mathbf{V}_r = [\mathbf{v}_1, \ldots, \mathbf{v}_r] \in \mathbb{R}^{d\times r}$$
$$\mathbf{\Sigma}_r = \text{diag}(\sigma_1, \ldots, \sigma_r) \in \mathbb{R}^{r\times r}$$

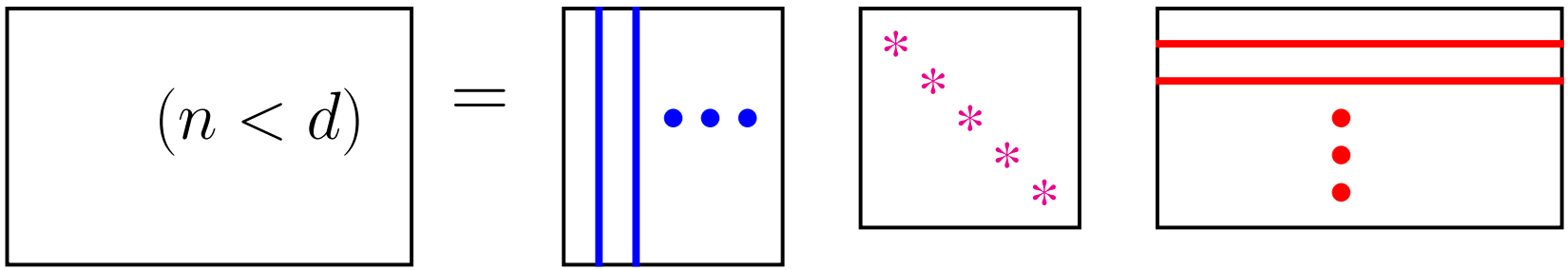
We then have

$$\mathbf{X} = \mathbf{U}_r\mathbf{\Sigma}_r\mathbf{V}_r^T.$$


$(n > d)$
=


$(n < d)$
=


$(n > d)$
=


$(n < d)$
=

- **Rank-1 decomposition**:

$$\mathbf{X} = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T.$$

This has the interpretation that $\mathbf{X}$ is a weighted sum of rank-one matrices, as for a square, symmetric matrix:

$$\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T = \sum \lambda_i \mathbf{q}_i \mathbf{q}_i^T.$$

In sum, $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$ where both $\mathbf{U}, \mathbf{V}$ have orthonormal columns and $\mathbf{\Sigma}$ is diagonal.

Lastly, for any version, the SVD of a matrix is not unique.

Matlab commands for computing matrix SVD

svd – Singular Value Decomposition.

[U,S,V] = svd(X) produces a diagonal matrix S, of the same dimension as X and with nonnegative diagonal elements in decreasing order, and orthogonal matrices U and V so that X = U*S*V^T.

s = svd(X) returns a vector containing the singular values.

svds – Find a few singular values and vectors.

S = svds(A,K) computes the K largest singular values of A.

[U,S,V] = svds(A,K) computes the singular vectors as well. If A is M-by-N and K singular values are computed, then U is M-by-K with orthonormal columns, S is K-by-K diagonal, and V is N-by-K with orthonormal columns.

PCA through SVD

Recall that the principal directions of a data set are given by the top eigenvectors of the covariance matrix

$$\mathbf{C} = \widetilde{\mathbf{X}}^T \widetilde{\mathbf{X}} \in \mathbb{R}^{d \times d}.$$

We have shown that they are also the right singular vectors of $\widetilde{\mathbf{X}}$:

$$\widetilde{\mathbf{X}}^T \widetilde{\mathbf{X}} = \mathbf{V} \mathbf{\Sigma}^T \mathbf{U}^T \cdot \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T = \mathbf{V} \underbrace{\left(\mathbf{\Sigma}^T \mathbf{\Sigma} \right)}_{\mathbf{\Lambda}} \mathbf{V}^T$$

Thus, one may just use the SVD of $\widetilde{\mathbf{X}}$ to compute the principal directions (and components), which is more efficient.

Principal Component Analysis (PCA)

The *principal components* of the data can be computed from the SVD as follows: For any principal direction $\mathbf{v}_j$, the projection is

$$\widetilde{\mathbf{X}}\mathbf{v}_j = \sigma_j \mathbf{u}_j$$

with optimal variance $\lambda_j = \sigma_j^2$.

Collectively, for the top k principal directions, the principal components are

$$\begin{aligned}\mathbf{Y} &= [\widetilde{\mathbf{X}}\mathbf{v}_1 \dots \widetilde{\mathbf{X}}\mathbf{v}_k] = \widetilde{\mathbf{X}}[\mathbf{v}_1 \dots \mathbf{v}_k] \\ &= [\sigma_1 \mathbf{u}_1 \dots \sigma_k \mathbf{u}_k] = [\mathbf{u}_1 \dots \mathbf{u}_k] \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix}.\end{aligned}$$

The total variance preserved by them is $\sum_{1 \le j \le k} \sigma_j^2$.

An SVD-based algorithm for PCA

Input: Data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ and integer k (with $0 < k < d$)

Output: Top k principal directions $\mathbf{v}_1, \ldots, \mathbf{v}_k$ and corresponding principal components $\mathbf{Y} \in \mathbb{R}^{n \times k}$.

Steps:

1. Center data: $\widetilde{\mathbf{X}} = [\mathbf{x}_1 \ldots \mathbf{x}_n]^T - [\mathbf{m} \ldots \mathbf{m}]^T$ where $\mathbf{m} = \frac{1}{n} \sum \mathbf{x}_i$
2. Perform SVD: $\widetilde{\mathbf{X}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$
3. Return: $\mathbf{Y} = \widetilde{\mathbf{X}} \cdot \mathbf{V}(:, 1:k) = \mathbf{U}(:, 1:k) \cdot \mathbf{\Sigma}(1:k, 1:k)$

MATLAB implementation of PCA

MATLAB built-in: **[V, US] = pca(X);** % *Rows of X are observations*

Alternatively, you may want to code it yourself:

```
n = size(X,1);
center = mean(X,1);
Xtilde = X - repmat(center, n, 1);
[U,S,V] = svds(Xtilde, k); % k is the reduced dimension
Y = Xtilde*V;
```

Note: The first three lines can be combined into one line

```
Xtilde = X - repmat(mean(X,1), size(X,1), 1);
```

Out-of-sample extension for PCA

Suppose we have carried out PCA on a given data set (e.g., training data):

- $\widetilde{\mathbf{X}} = [\mathbf{x}_1 \dots \mathbf{x}_n]^T - [\mathbf{m} \dots \mathbf{m}]^T$ where $\mathbf{m} = \frac{1}{n}\sum \mathbf{x}_i$
- $\widetilde{\mathbf{X}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$

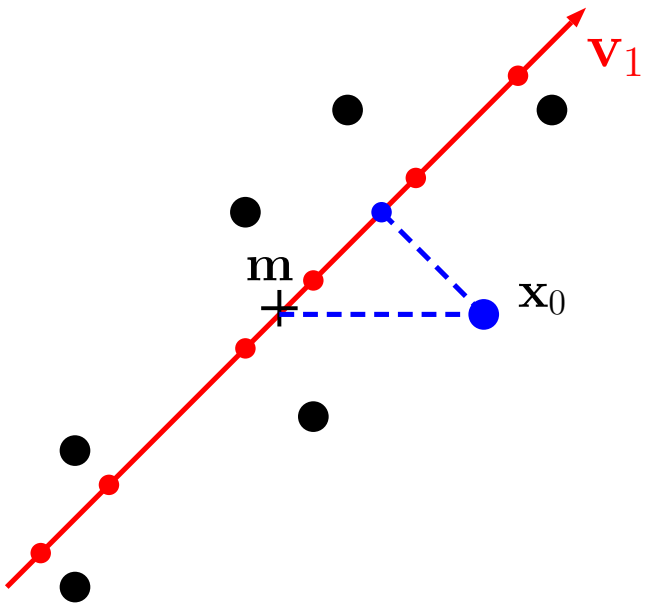
Now there is a new point $\mathbf{x}_0$ (e.g., a test point). How can we extend PCA to $\mathbf{x}_0$?

Two options:

- Add the new point to the data set and re-run PCA (maybe more accurate, but time-consuming)

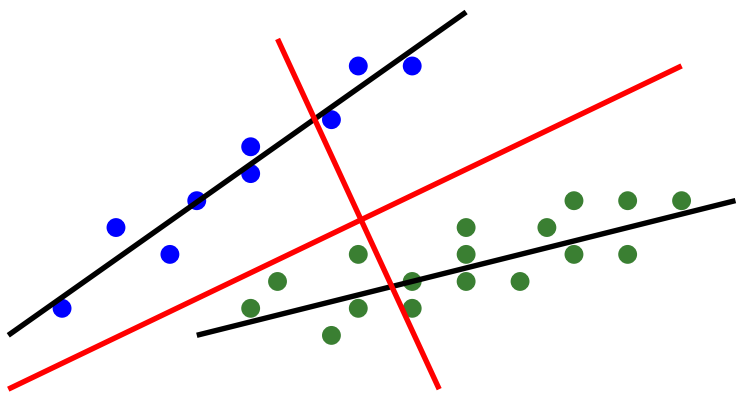
- Just use the PCA subspace that has already been obtained to project the new point directly:

$$\mathbf{v}_i^T \cdot (\mathbf{x}_0 - \mathbf{m}), \quad i = 1, 2, \ldots$$



PCA for labeled data

In the supervised setting when data points have labels which divide them into different groups (i.e., classes), one can perform PCA on the full data set but without using the labels to project the different classes onto the same PCA plane.



We call this procedure **Global PCA**, which requires a higher dimension than each of the **classwise PCAs**.

PCA for classification

Note that in the classification setting there are two data sets: $\mathbf{X}_{\text{train}}$ and $\mathbf{X}_{\text{test}}$.

Perform PCA on the entire training set (without the labels) and then project the test data using the training PCA plane:

$$\mathbf{Y}_{\text{test}} = \left(\mathbf{X}_{\text{test}} - [\mathbf{m}_{\text{train}} \dots \mathbf{m}_{\text{train}}]^T\right) \cdot \mathbf{V}_{\text{train}}$$

Finally, select a classifier to work in the reduced space:

- PCA + kNN
- PCA + nearest local centroid
- PCA + other classifiers

How to set the parameter k in principle?

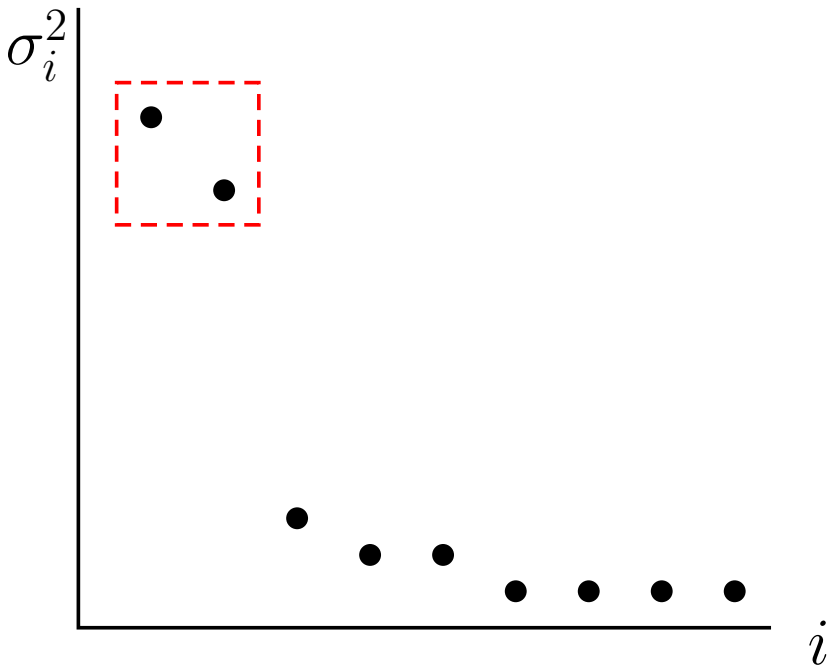
Generally, there are two ways to choose the reduced dimension s:

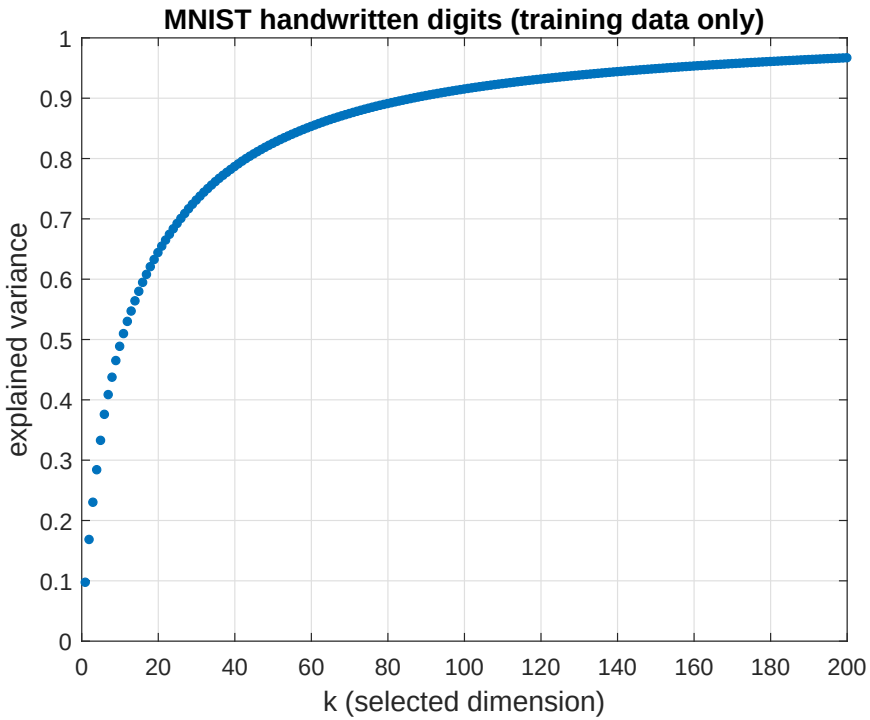
- Set $k = \#$"dominant" singular values
- Choose k such that the top k principal directions explain a certain fraction of the variance of the data:

$$\underbrace{\sum_{i=1}^{k} \sigma_i^2}_{\text{explained variance}} \Big/ \underbrace{\sum_{i=1}^{r} \sigma_i^2}_{\text{total variance}} > p.$$

Typically, $p = .95$, or $.99$ (more conservative), or $.90$ (more aggressive).

Principal Component Analysis (PCA)





Some further comments

PCA is an *unsupervised* method, and the 95% criterion is only a conservative choice which discards only the directions with small amounts of variance.

In the context of classification it is possible to get much lower than this threshold while maintaining or even improving the classification accuracy.[1]

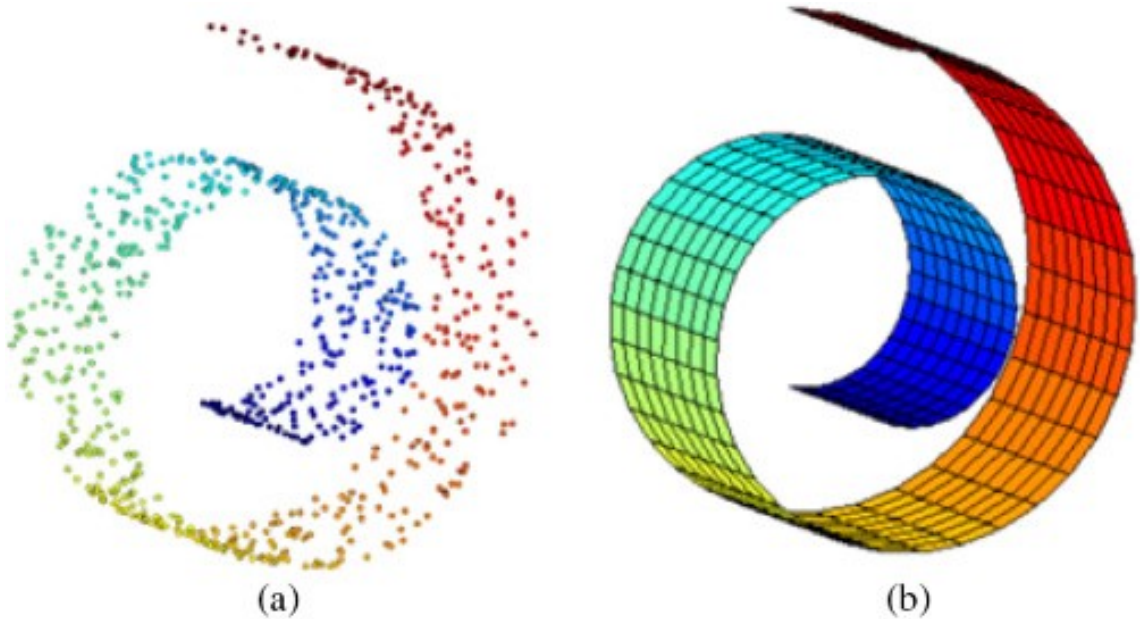
The reason is that higher variance is not always useful for classification.

In practice, one may want to use cross validation to select the optimal projection dimension.

[1] `http://www.math.sjsu.edu/~gchen/Math285S16/HW2-Terry.pdf`

Principal Component Analysis (PCA)

Lastly, PCA is a linear projection method, which means that for nonlinear data, PCA will need to use a dimension higher than the manifold dimension (in order to preserve most of the variance).



(Picture taken from `https://medium.com/@snk.nitin`)

HW2 (due Wednesday, October 3)

Please type your work for questions 1 and 2 below.

1. Given a data set of three points $(1, 2, 0), (2, 1, 0), (0, 0, 0)$ in $\mathbb{R}^3$, find the first two principal directions and corresponding principal components of the data. How much variance is explained by each principal component?

2. Prove the claim in the middle of the proof of the SVD theorem (slide 22) that $\mathbf{u}_1, \ldots, \mathbf{u}_r$ are orthonormal vectors. Hint: consider the dot product $\mathbf{u}_i^T \mathbf{u}_j$ for $i = j$ and $i \neq j$ separately.

3. Download the MNIST data set from the link given in the course syllabus and then perform PCA on (a) only the digits 0 (b) only the digits 1 and (c) the two digit classes 0,1 together to project the data onto a 2-dimensional PCA plane. Plot the projected data in each case (for the combined set of digits 0 and 1, display data points in the two classes with different symbols and colors to make them distinguishable from each other).

4. For the USPS data set, how many PCA dimensions are required to preserve 95% of the total variance (for the training data)? 50% variance?

   Next, perform kNN classification for each $k = 1, 2, \ldots, 10$ with such two choices of PCA dimension and also no projection (i.e., all 256 dimensions). Compare the test error rates for the three versions of data by plotting them together in one figure (all against the parameter k). Which choice of PCA dimension seems to work the best? Record also the running time of the

kNN classifier for each k on each version of data. How much faster is PCA 50% + kNN than kNN alone (i.e., no projection)?

5. Repeat Question 4 with the nearest local centroid classifier instead.

6. For the new data set you found in Question 5 of HW1, explore it again by adding PCA (with a dimension parameter) as an extra option to consider. How much improvement can you make regarding test accuracy and time?