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

Dr. Guangliang Chen

February 9, 2016

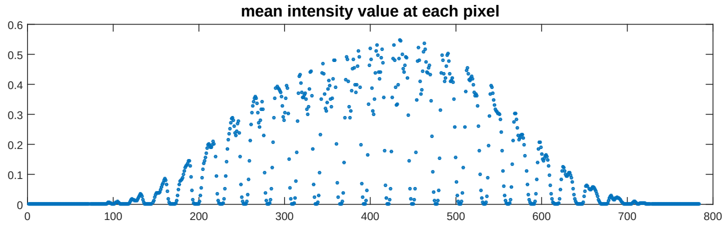
Outline

- Introduction
- Review of linear algebra
- Matrix SVD
- PCA

Motivation

- The digits are 784 dimensional - very time consuming to perform even simple tasks like kNN search
- Need a way to reduce the dimensionality of the data in order to increase speed
- However, if we discard some dimensions, will that degrade performance?
- The answer can be no (as long as we do it carefully). In fact, it may even improve results in some cases.

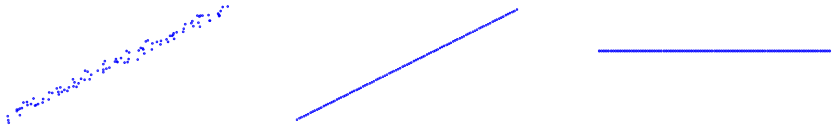
How is this possible?



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

The main idea of PCA

PCA reduces the dimensionality by discarding low-variance directions (under the assumption that useful information is only along high-variance dimensions)



Other dimensionality reduction techniques

PCA reduces the dimension of data by preserving as much variance as possible. Here are some alternatives:

- **Linear Discriminant Analysis (LDA)**: by preserving discriminatory information between the different training classes
- **Multidimensional Scaling (MDS)**: by preserving pairwise distances
- **ISOmap**: by preserving geodesic distances
- **Locally Linear Embedding (LLE)**: by preserving local geometry

Review of matrix algebra

- Special square matrices $\mathbf{A} \in \mathbb{R}^{n\times n}$
  - **Symmetric**: $\mathbf{A}^T = \mathbf{A}$
  - **Diagonal**: $a_{ij} = 0$ whenever $i \neq j$
  - **Orthogonal**: $\mathbf{A}^{-1} = \mathbf{A}^T$ (i.e. $\mathbf{A}\mathbf{A}^T = \mathbf{A}^T\mathbf{A} = \mathbf{I}_n$)
- Eigenvalues and eigenvectors: $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$
  - λ satisfies $\det(\mathbf{A} - \lambda\mathbf{I}) = 0$
  - $\mathbf{v}$ satisfies $(\mathbf{A} - \lambda\mathbf{I})\mathbf{v} = \mathbf{0}$

Eigenvalue decomposition of symmetric matrices

Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be a symmetric matrix. Then there exist an orthogonal matrix $\mathbf{Q} = [\mathbf{q}_1 \dots \mathbf{q}_n]$ and a diagonal matrix $\Lambda = \mathrm{diag}(\lambda_1, \dots, \lambda_n)$, both real & square, such that

$$\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$$

This is also called the *spectral decomposition* of $\mathbf{A}$.

Note that the above equation is equivalent to

$$\mathbf{A}\mathbf{q}_i = \lambda_i \mathbf{q}_i, \quad i = 1, \dots, n$$

Therefore, the λ_i's represent eigenvalues of $\mathbf{A}$ while the $\mathbf{q}_i$'s are the associated eigenvectors.

Principal Component Analysis (PCA)

Remark 1: For convenience the diagonal elements of Λ are often sorted such that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$.

Remark 2: For asymmetric matrices, neither $\underline{\mathbf{Q} \text{ is orthogonal}}$ nor $\underline{\Lambda \text{ is diagonal}}$ needs to be true.

For an abitrary matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, we have the following decomposition

$$\mathbf{A} = \mathbf{P}\mathbf{J}\mathbf{P}^{-1}$$

where $\mathbf{P}$ is invertible and $\mathbf{J}$ is upper triangular. This is called the *Jordan canonical form* of $\mathbf{A}$.

When $\mathbf{J}$ can be made diagonal by selecting $\mathbf{P}$ (not always possible), we say that $\mathbf{A}$ is *diagonalizable*.

Positive (semi)definite matrices

A symmetric matrix $\mathbf{A}$ is said to be positive (semi)definite if $\mathbf{x}^T\mathbf{A}\mathbf{x} > 0$ (≥ 0) for all $\mathbf{x} \neq \mathbf{0}$.

Theorem. A symmetric matrix $\mathbf{A}$ is positive (semi)definite if and only if its eigenvalues are positive (nonnegative).

Proof: This result can be proved by applying the spectral decomposition of $\mathbf{A}$. Left to you as an exercise.

Matlab command for eigenvalue decomposition

D = eig(A) produces a column vector D containing the eigenvalues of a square matrix A.

[V,D] = eig(A) produces a diagonal matrix D of eigenvalues and a full matrix V whose columns are the corresponding eigenvectors so that $A = VDV^{-1}$.

Singular value decomposition (SVD)

Let $\mathbf{X} \in \mathbb{R}^{n\times d}$ be a rectangular matrix. The SVD of $\mathbf{X}$ is defined as

$$\mathbf{X}_{n\times d} = \mathbf{U}_{n\times n}\Sigma_{n\times d}\mathbf{V}_{d\times d}^T$$

where $\mathbf{U} = [\mathbf{u}_1 \ldots \mathbf{u}_n], \mathbf{V} = [\mathbf{v}_1 \ldots \mathbf{v}_d]$ are orthogonal and Σ is “diagonal”:

- $\mathbf{u}_i$'s are called the *left singular vectors* of $\mathbf{X}$;
- $\mathbf{v}_j$'s are called the *right singular vectors* of $\mathbf{X}$;
- The “diagonal” entries of Σ are called the *singular values* of $\mathbf{X}$.

Note. This is often called the *full SVD*, to distinguish from other variations.

Matlab command for 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.

Other versions of matrix SVD

Let $\mathbf{X} \in \mathbb{R}^{n\times d}$. We define

- **Economic SVD**:

$$\mathbf{X}_{n\times d} = \begin{cases} \mathbf{U}_{n\times d}\Sigma_{d\times d}\mathbf{V}_{d\times d}^T, & n > d \text{ (tall matrix)} \\ \mathbf{U}_{n\times n}\Sigma_{n\times n}\mathbf{V}_{d\times n}^T, & n < d \text{ (long matrix)} \end{cases}$$

- **Compact SVD**:

$$\mathbf{X}_{n\times d} = \mathbf{U}_{n\times r}\Sigma_{r\times r}\mathbf{V}_{d\times r}^T, \quad \text{where } r = \operatorname{rank}(\mathbf{X})$$

- **Rank-1 decomposition**:

$$\mathbf{X} = \sum \sigma_i \mathbf{u}_i \mathbf{v}_i^T$$

Matlab command for Economic SVD

- For tall matrices only:

  [U,S,V] = svd(X,0) produces the "economy size" decomposition. If X is m-by-n with m > n, then only the first n columns of U are computed and S is n-by-n. For m <= n, svd(X,0) is equivalent to svd(X).

- For both tall and long matrices:

  [U,S,V] = svd(X,'econ') also produces the "economy size" decomposition. If X is m-by-n with m >= n, then it is equivalent to svd(X,0). For m < n, only the first m columns of V are computed and S is m-by-m.

A brief mathematical derivation of SVD

For any $\mathbf{X} \in \mathbb{R}^{n\times d}$, form $\mathbf{C} = \mathbf{X}^T\mathbf{X} \in \mathbb{R}^{d\times d}$.

Observation: $\mathbf{C}$ is square, symmetric, and positive semidefinite.

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

Define $\sigma_i = \sqrt{\lambda_i}$ for all i and $\mathbf{u}_i = \frac{1}{\sigma_i}\mathbf{X}\mathbf{v}_i$ for $1 \leq i \leq r$.

Claim: $\mathbf{u}_1, \ldots, \mathbf{u}_r$ are orthonormal vectors.

Let $\mathbf{U} = [\mathbf{u}_1 \ldots \mathbf{u}_r \mathbf{u}_{r+1} \ldots \mathbf{u}_d]$ such that it is orthogonal and $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_d)$. Then from $\mathbf{X}\mathbf{v}_i = \sigma_i\mathbf{u}_i, \forall i$ we obtain $\mathbf{X}\mathbf{V} = \mathbf{U}\boldsymbol{\Sigma}$, or equivalently, $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$.

Connection with eigenvalue decomposition

Let $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}$ be the full SVD of $\mathbf{X} \in \mathbb{R}^{n\times d}$. Then

- $\mathbf{U}$ consists of the eigenvectors of $\mathbf{X}\mathbf{X}^T \in \mathbb{R}^{n\times n}$ (gram matrix);
- $\mathbf{V}$ consists of the eigenvectors of $\mathbf{X}^T\mathbf{X} \in \mathbb{R}^{d\times d}$ (convariance matrix, if the rows of $\mathbf{X}$ have a mean of zero);
- Σ consists of the square roots of the eigenvalues of either matrix (and zero).

Principal Component Analysis (PCA) Algorithm

Input: data set $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_n]^T \in \mathbb{R}^{n \times d}$ and integer s (with $0 < s < d$)

Output: compressed data $\mathbf{Y} \in \mathbb{R}^{n \times s}$

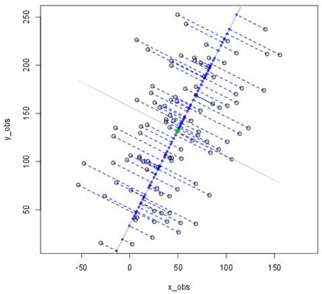
Steps:

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

Terminology. We call the columns of $\mathbf{V}$ the *principal directions* of the data and the columns of $\mathbf{Y}$ its top s *principal components*.

Geometric meaning

The principal components $\mathbf{Y}$ represent the *coefficients* of the projection of $\mathbf{X}$ onto the subspace through the point $\mathbf{m} = \frac{1}{n}\sum \mathbf{x}_i$ and spanned by the basis vectors $\mathbf{v}_1, \ldots, \mathbf{v}_s$.



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] = svd(Xtilde, 'econ');
Y = Xtilde*V(:,1:k); % k is the reduced dimension
```

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

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

Some properties of PCA

Note that the *principal coordinates*

$$\mathbf{Y} = [\widetilde{\mathbf{X}}\mathbf{v}_1 \ldots \widetilde{\mathbf{X}}\mathbf{v}_s] = [\sigma_1\mathbf{u}_1 \ldots \sigma_s\mathbf{u}_s]$$

- The projection of $\widetilde{\mathbf{X}}$ onto the ith *principal direction* $\mathbf{v}_i$ is $\widetilde{\mathbf{X}}\mathbf{v}_i = \sigma_i\mathbf{u}_i$.
- The variance of the ith projection $\widetilde{\mathbf{X}}\mathbf{v}_i$ is σ_i^2.
  Proof. First, $\mathbf{1}^T\widetilde{\mathbf{X}}\mathbf{v}_i = \mathbf{0}^T\mathbf{v}_i = 0$. This shows that $\widetilde{\mathbf{X}}\mathbf{v}_i$ has mean zero. Second, $\|\sigma_i\mathbf{u}_i\|_2^2 = \sigma_i^2$. Accordingly, the projection $\widetilde{\mathbf{X}}\mathbf{v}_i = \sigma_i\mathbf{u}_i$ has a variance of σ_i^2.
- Therefore, the right singluar vectors $\mathbf{v}_1, \ldots, \mathbf{v}_s$ represent the directions of decreasing variance ($\sigma_1^2 \geq \cdots \geq \sigma_s^2$).

Is there an even better direction than v_1?

The answer is **no**; that is, $\pm\mathbf{v}_1$ contains the maximum possible variance among all single directions.

Mathematically, it is the solution of the following problem:

$$\max_{\mathbf{v}\in\mathbb{R}^d:\, \|\mathbf{v}\|_2=1} \quad \|\widetilde{\mathbf{X}}\mathbf{v}\|_2^2$$

Proof. Consider the full SVD of $\widetilde{\mathbf{X}} = \mathbf{U}\Sigma\mathbf{V}^T$. For any unit vector $\mathbf{v}\in\mathbb{R}^d$, write $\mathbf{v} = \mathbf{V}\alpha$ for some unit vector $\alpha\in\mathbb{R}^d$. Then $\mathbf{X}\mathbf{v} = \mathbf{X}\mathbf{V}\alpha = \mathbf{U}\Sigma\alpha$. Accordingly, $\|\mathbf{X}\mathbf{v}\|_2^2 = \|\mathbf{U}\Sigma\alpha\|_2^2 = \|\Sigma\alpha\|_2^2 = \sum\sigma_i^2\alpha_i^2 \le \sigma_1^2$, where the equality holds when $\alpha = \pm\mathbf{e}_1$ and correspondingly, $\mathbf{v} = \pm\mathbf{V}\mathbf{e}_1 = \pm\mathbf{v}_1$.

Remark: The maximum value is σ_1^2, which is consistent with before.

How to set the parameter s in principle?

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

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

$$\underbrace{\sum_{i=1}^{s} \sigma_i^2}_{\text{explained variance}} \quad > \quad p \ \cdot \ \underbrace{\sum \sigma_i^2}_{\text{total variance}}$$

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

More interpretations of PCA

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

It is also known that PCA for a given dimension s

- preserves the most of the pairwise distances between the data points
- minimizes the total orthogonal fitting error by a s-dimensional subspace

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 and then project the test data using the training basis:

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

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

- PCA + kNN
- PCA + local $k$means
- PCA + other classifiers

A final remark about the projection dimension s

PCA is an *unsupervised* method, and the 95% (or 90%) criterion is only a conservative choice in order to avoid losing any useful direction.

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

The reason is that variance is not necessarily useful for classification.

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

HW2a (due in about two weeks)

1. Apply the plain kNN classifier with 6-fold cross validation to the projected digits with $s = 154$ (corresponding to 95% variance) and 50 (corresponding to 82.5% variance) separately to select the best k from the range 1:10 (for each s). Plot the curves of validation error versus k and compare them with that of no projection (i.e. HW1-Q1). For the three choices of s (50, 154, and 784), what are the respective best k? Overall, which (s, k) pair gives the smallest validation error?

2. For each of the three values of s above (with corresponding optimal k), apply the plain kNN classifier to the test data and display their errors using a bar plot. Interpret the results. Is what you got in this question consistent with that in Question 1?

Principal Component Analysis (PCA)

3. Apply the local $k$means classifier, with each $k = 1, \ldots, 10$, to the test set of handwrittend digits after they are projected into $\mathbb{R}^{50}$. How does your result compare with that of no projection (i.e. HW1-Q5)? What about speed?

Note: HW2b will be available after we finish our next topic.

Midterm project 1: Instance-based classification

Task: Concisely describe the classifiers we have learned thus far and summarize their corresponding results in a poster to be displayed in the classroom. You are also encouraged to try new ideas/options (e.g., normal weights for weighted kNN, other metrics such as cosine) and include your findings in the poster.

Who can participate: One to two students from this class, subject to instructor's approval.

When to finish: In 2 to 3 weeks.

How it will be graded: Based on clarity, completeness, correctness, originality.

You are welcome to consult with the instructor for ideas to try in this project.