CS 229 - Machine Learning
Xiangliang Zhang
Computer Science (CS)/Statistics (STAT) Program
Computer, Electrical and Mathematical Sciences & Engineering (CEMSE) Division
King Abdullah University of Science and Technology (KAUST)

Homework IX:
Unsupervised Learning: PCA and SVD

Henrique Aparecido Laureano

Spring Semester 2018

Contents

Question 1: Principal Components Analysis (PCA)

(a) [15pts]

Discuss and show the proof of how PCA maximizes the variance of projected data.

Solution:

PCA can be defined as the orthogonal projection of the data onto a lower dimensional linear space, know as the *principal subspace*, such that the variance of the projected data is maximized.

Consider a dataset of observations $\{x_n\}$ where $n = 1, \ldots, N$ (x_n with dimensionality D).

Denote:

- D the dimensionality;
- M the fixed dimension of the principal subspace;
- $\{\mathbf{u}_i\}, i = 1, \ldots, M$ the basis vectors ($(D \times 1)$ vectors) of the principal subspace.

Consider a unit D-dimensional normalized vector $\mathbf{u}_1$ ($\mathbf{u}_1^\top \mathbf{u}_1 = 1$). Each point $\mathbf{x}_n$ is then projected onto a scalar value $\mathbf{u}_1^\top \mathbf{x}_n$. The mean of the projected data is $\mathbf{u}_1^\top \bar{\mathbf{x}}$ where $\bar{\mathbf{x}}$ is the sample mean given by

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n$$

and the variance of the projected data is given by

$$\frac{1}{N} \sum_{n=1}^{N} \{\mathbf{u}_1^\top \mathbf{x}_n - \mathbf{u}_1^\top \bar{\mathbf{x}}\}^2 = \mathbf{u}_1^\top \mathbf{S} \mathbf{u}_1$$

where $\mathbf{S}$ is the data covariance matrix defined by

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^\top.$$

Now we maximize the projected variance $\mathbf{u}_1^\top \mathbf{S} \mathbf{u}_1$ with respect to $\mathbf{u}_1$. This has to be a constrained maximization to prevent $\|\mathbf{u}_1\| \to \infty$. The appropriate constraint comes from the normalization condition $\mathbf{u}_1^\top \mathbf{u}_1 = 1$. To enforce this constraint, we introduce a Lagrange multiplier that we shall denote by λ_1, and then make an unconstrained maximization of

$$\mathbf{u}_1^\top \mathbf{S} \mathbf{u}_1 + \lambda_1 (1 - \mathbf{u}_1^\top \mathbf{u}_1).$$

Setting the derivative with respect to $\mathbf{u}_1$ equal to zero, we see that this quantity will have a stationary point when

$$\mathbf{S}\mathbf{u}_1 = \lambda_1 \mathbf{u}_1$$

which says that $\mathbf{u}_1$ must be an eigenvector of $\mathbf{S}$. If we left-multiply by $\mathbf{u}_1^\top$ and make use of $\mathbf{u}_1^\top \mathbf{u}_1 = 1$, we see that the variance is given by

$$\mathbf{u}_1^\top \mathbf{S}\mathbf{u}_1 = \lambda_1$$

and so the variance will be a maximum when we set $\mathbf{u}_1$ equal to the eigenvector having the largest eigenvalue λ_1. This eigenvector is known as the first principal component.

We can define additional principal components in an incremental fashion by choosing each new direction to be that which maximizes the projected variance amongst all possible directions orthogonal to those already considered. if we consider the general case of an M-dimensional projection space, the optimal linear projection for which the variance of the projected data is maximized is now defined by the M eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_M$ of the data covariance matrix $\mathbf{S}$ corresponding to the M largest eigenvalues $\lambda_1, \ldots, \lambda_M$.

□

(b) [15pts]

Discuss and show the proof of how PCA minimizes the average projection cost - defined as the mean squared distance between the data points and their projections.

Solution:

PCA can also be defined as the linear projection that minimizes the average projection cost, defined as the mean squared distance between the data points and their projections.

Denote:

- $\{\mathbf{u}_i\}, i = 1, \ldots, D$ as a complete orthonormal set of D-dimensional basis vectors.

Because this basis is complete, each data point can be represented by a linear combination of the basis vectors

$$\mathbf{x}_n = \sum_{i=1}^{D} \alpha_{ni} \mathbf{u}_i, \quad \text{where} \quad \alpha_{ni} = \mathbf{x}_n^\top \mathbf{u}_i.$$

We can approximate this data point using a representation involving a restricted number $M < D$ of variables corresponding to a projection onto a lower-dimensional subspace. The M-dimensional linear subspace can be represented by the first M of the basis vectors, and so we approximate each data point $\mathbf{x}_n$ by

$$\widetilde{\mathbf{x}}_n = \sum_{i=1}^{M} z_{ni} \mathbf{u}_i + \sum_{i=M+1}^{D} b_i \mathbf{u}_i$$

where the $\{z_{ni}\}$ depend on the particular data point, whereas the $\{b_i\}$ are constants that are the same for all data points. We are free to choose the $\{\mathbf{u}_i\}$, the $\{z_{ni}\}$, and the $\{b_i\}$ so as to minimize the distortion introduced by the reduction in dimensionality. As our distortion measure, we use the squared distance between the original data point $\mathbf{x}_n$ and its approximation $\widetilde{\mathbf{x}}_n$, averaged over the dataset, so the goal is to minimize

$$J = \frac{1}{N}\sum_{n=1}^{N}\|\mathbf{x}_n - \widetilde{\mathbf{x}}_n\|^2 \quad \text{w.r.t.} \quad \{z_{ni}\}, \{b_i\}.$$

Setting derivative w.r.t.

- z_{ni} to 0 and making use of the orthonormality relations $\Rightarrow z_{ni} = \mathbf{x}_n^\top \mathbf{u}_i$ where $i = 1, \ldots, M$;
- b_i to 0 and making use of the orthonormality relations $\Rightarrow b_i = \overline{\mathbf{x}}^\top \mathbf{u}_i$ where $i = M+1, \ldots, D$.

Rewriting the distortion J

$$\begin{aligned}
J &= \frac{1}{N}\sum_{n=1}^{N}\|\sum_{i=1}^{D}(\mathbf{x}_n^\top \mathbf{u}_i)\mathbf{u}_i - \sum_{i=1}^{M}(\mathbf{x}_n^\top \mathbf{u}_i)\mathbf{u}_i - \sum_{i=M+1}^{D}(\overline{\mathbf{x}}^\top \mathbf{u}_i)\mathbf{u}_i\|^2 \\
&= \frac{1}{N}\sum_{n=1}^{N}\|\sum_{i=M+1}^{D}(\mathbf{x}_n^\top \mathbf{u}_i - \overline{\mathbf{x}}^\top \mathbf{u}_i)\mathbf{u}_i\|^2 \\
&= \frac{1}{N}\sum_{n=1}^{N}\sum_{i=M+1}^{D}(\mathbf{x}_n^\top \mathbf{u}_i - \overline{\mathbf{x}}^\top \mathbf{u}_i)^2 \\
&= \frac{1}{N}\sum_{i=M+1}^{D}\sum_{n=1}^{N}\mathbf{u}_i^\top(\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{x}_n^\top - \overline{\mathbf{x}}^\top)\mathbf{u}_i \\
&= \sum_{i=M+1}^{D}\mathbf{u}_i^\top \mathbf{S}\mathbf{u}_i.
\end{aligned}$$

There remains the task of minimizing J with respect to $\{\mathbf{u}_i\}$, which must be a constrained minimization othermise we obtain the vacuous result $\mathbf{u}_i = 0$. The constraints $\mathbf{u}_i^\top \mathbf{u}_i = 1$ arise from the orthonormality conditions and the solution is expressed in terms of the eigenvector expansion of the covariance matrix. Using a Lagrange multiplier λ_i to enforce the constraint, we consider the minimization of

$$\mathbf{u}_i^\top \mathbf{S}\mathbf{u}_i + \lambda_i(1 - \mathbf{u}_i^\top \mathbf{u}_i).$$

Setting the derivative w.r.t. $\mathbf{u}_i$ to zero we obtain $\mathbf{S}\mathbf{u}_i = \lambda_i \mathbf{u}_i$ so that $\mathbf{u}_i$ is an eigenvector of $\mathbf{S}$ with eigenvalue λ_i. Thus any eigenvector will define a stationary point of the distortion measure. In order to minimize the average squared projection distance, we should choose the principal component subspace to pass through the mean of the data points and to be aligned with the directions of maximum variance. For the case when the eigenvalues are equal, ay choice of principal direction will rise to the same value of J.

As usual the eigenvectors $\{\mathbf{u}_i\}$ are chosen to be orthonormal. The corresponding value of the distortion measure is then given by

$$J = \sum_{i=M+1}^{D} \mathbf{u}_i^\top \mathbf{S}\mathbf{u}_i = \sum_{i=M+1}^{D} \mathbf{u}_i^\top \lambda_i \mathbf{u}_i = \sum_{i=M+1}^{D} \lambda_i$$

which is simply the sum of the eigenvalues of those eigenvectors that are orthogonal to the principal subspace. We therefore obtain the minimum value of J by selecting these eigenvectors to be those having the $D-M$ smallest eigenvalues, and hence the eigenvectors defining the principal subspace are those corresponding to the M largest eigenvalues.

Although we considered $M < D$, if $M = D$ there is no dimensionality reduction but simply a rotation of the coordinate axes to align with principal components.

□

(c) [30pts] Implementation of PCA (Case B)

Take the same wine quality data we used in the SVM homework.

```
# <r code> ============================================================================ #
#                                                                    choosing the red-wine!
path <- "~/Dropbox/KAUST/machine_learning/hw9/"                          # files path
                                                  # df: dataframe. reading the dataset
df <- read.csv(paste0(path, "winequality-red.csv"), header = TRUE, sep = ";")

#     take ‘‘quality’’ as class label, e.g., 1-5 as negative, while 6-10 as positive
                                # defining class label: 1-5: negative, 6-10: positive
df$quality <- as.factor(ifelse(df$quality <= 5, "negative", "positive"))
# </r code> ============================================================================ #
```

Apply PCA at first,

```
# <r code> ============================================================================ #
                          # centering the data without the last column, quality label
df.cent <- as.matrix(df[ , -12])
for (i in 1:11) df.cent[ , i] <- df.cent[ , i] - mean(df.cent[ , i])

                                                       # computing covariance matrix
S <- 1/(nrow(df.cent) - 1) * t(df.cent) %*% df.cent

eigens <- eigen(S)                        # computing the eigenvectors and eigenvalues

pcs <- matrix(NA, nrow = nrow(df.cent), ncol = 11)     # empty matrix for the PC's

              # computing the PC's: multiplying the data matrix by the eigenvectors
for (i in 1:11) pcs[ , i] <- df.cent %*% eigens$vectors[ , i]
```

```r
par(mar = c(4, 4, 0, 0) + .1)                    # plotting eigenvector variances
plot(eigens$values, type = "b", xlab = "Eigenvector", ylab = "Variance")
# </r code> ========================================================================== #
```

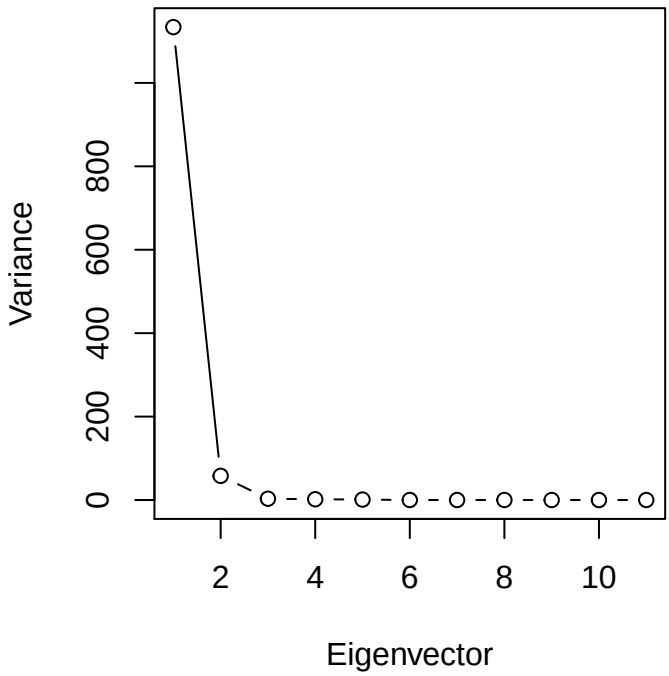

Figure 1: Eigenvalues distribution. Variance explained by each eigenvector.

We see in Figure 1 that the first principal component is responsable for almost all the variance explanation.

and then learn SVM from the new representation. Show whether PCA is helpful on improving your classification accuracy (choosing any one of the kernels is ok, and parameters can be set according to the search result in SVM homework). How the result will be different if different numbers of PCs are selected for the new representation?

```r
# <r code> ========================================================================== #
library(e1071)                                  # loading library for the svm fit
# obs.    by default the method svm uses a classification machine algorithm,
#         C-classification, and scale the variables to zero mean and unit variance
svm.original <- svm(quality ~ ., df             # using all 11 available variables
                    , kernel = "linear"                          # linear kernel
                    , cross = 5                                      # 5-fold CV
                    , cost = .5             # chosen cost of constraints violation
                    , gamma = .001)                              # chosen \gamma
aucs <- numeric(12)                                    # vector to keep the AUC's
library(pROC)                                  # loading library for the AUC's
                                                                 # computing AUC
aucs[1] <- auc(roc(as.numeric( df$quality ), as.numeric( svm.original$fitted )))
```

```r
#                                                    doing the same for the PC's
for (i in 1:11) {
  model <- svm(df$quality ~ pcs[ , 1:i]
               , kernel = "linear", cross = 5, cost = .5, gamma = .001)
  aucs[i + 1] <- auc(roc(as.numeric( df$quality ), as.numeric( model$fitted )))
}
# </r code> ========================================================================== #
```

Table 1: AUC obtained with the original data and with different numbers of PC's.

Variable	**AUC**
Original data	0.7509267
1(st) principal component	0.5952344
2 principal components	0.5964747
3 principal components	0.6029593
4 principal components	0.6041289
5 principal components	0.7239483
6 principal components	0.7356088
7 principal components	0.7368657
8 principal components	0.7356348
9 principal components	0.7478895
10 principal components	0.7515988
11 (all) principal components	0.7509267

With the original data, eleven variables, we have an AUC of 0.7509. Using only the first principal component (PC), responsable for almost all the variance, the AUC is 0.5952.

Considering more PC's the AUC increase, but not so much. However, when we consider 5 PC we see a bigger increase in the AUC, reaching a value closer to the value obtained with all the data. With all the PC's the AUC is exactly the same that with the original data.

Conclusion. Here, with five PC's we're already able to reach a result, AUC, similar to the obtained using all the data.

□

Question 2: Singular Value Decomposition (SVD)

(a) [10pts]

Discuss

- **How PCA and SVD are related to each other (with proof).**

Solution:

By SVD we can write X as $U\Sigma V^\top$ and in consequence we can do

$$X^\top X = (U\Sigma V^\top)^\top (U\Sigma V^\top) = V\Sigma^\top U^\top U\Sigma V^\top = V(\Sigma^\top\Sigma)V^\top,$$

which means that $X^\top X$ and $\Sigma^\top\Sigma$ are similar. Similar matrices have the same eigenvalues, so the eigenvalues λ_i of the covariance matrix $S = (n-1)^{-1}X^\top X$ are related to the singular values σ_i of the matrix X via

$$\sigma_i^2 = (n-1)\lambda_i, \quad i = 1, \ldots, r, \qquad \text{where} \quad r = \operatorname{rank}(X).$$

To fully relate SVD and PCA we describe the correspondence between principal components and singular vectors. For the right singular vectors we take

$$\widehat{V}^\top = \begin{pmatrix} & v_1^\top & \\ & \vdots & \\ & v_r^\top & \end{pmatrix}$$

where v_i are the principal components of X. For the left singular vectors we take

$$u_i = \frac{1}{\sqrt{(n-1)\lambda_i}} X v_i.$$

Since $X = U\Sigma V^\top$,

$$X = \sum_{i=1}^{r} \sigma_i u_i v_j^\top.$$

We can prove this by a small example to simplify the notation. Let $v_i = (v_{1i}, v_{2i})^\top$ for $i = 1, 2$ and $u_i = (u_{1i}, u_{2i}, u_{3i})$ for $i = 1, 2, 3$. Then,

$$U\Sigma V^\top = \begin{pmatrix} u_1 & u_2 \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} \begin{pmatrix} & v_1^\top & \\ & v_2^\top & \end{pmatrix} = \begin{pmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \\ u_{31} & u_{32} \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} \begin{pmatrix} v_{11} & v_{21} \\ v_{12} & v_{22} \end{pmatrix}.$$

Now splitting up Σ

$$U\Sigma V^\top = U \begin{pmatrix} \sigma_1 & 0 \\ 0 & 0 \end{pmatrix} V^\top + U \begin{pmatrix} 0 & 0 \\ 0 & \sigma_2 \end{pmatrix} V^\top$$

and tackle the terms individually. One way to rewrite the first term is

$$U \begin{pmatrix} \sigma_1 & 0 \\ 0 & 0 \end{pmatrix} V^\top = \begin{pmatrix} u_{11}\sigma_1 & 0 \\ u_{21}\sigma_1 & 0 \\ u_{31}\sigma_1 & 0 \end{pmatrix} \begin{pmatrix} v_{11} & v_{21} \\ v_{12} & v_{22} \end{pmatrix} = \sigma_1 u_1 v_1^\top,$$

where the last step follows because the entries highlighted in red do not affect the result of the matrix multiplication. A similar calculation shows that the second term is $\sigma_2 u_2 v_2^\top$, which proves the claim for the given example. In general some of the singular values could be 0, which makes the sum go only up to $r = \text{rank}(X)$.

Applying this fact to X we have

$$U\Sigma V^\top = \sum_{i=1}^{r} \sigma_i u_i v_i = \sum_{i=1}^{r} \sqrt{(n-1)\lambda_i} \frac{1}{\sqrt{(n-1)\lambda_i}} X v_i v_i^\top = X \sum_{i=1}^{r} v_i v_i^\top = X,$$

where the last step follows from $I = V^\top V = \sum_{i=1}^{r} v_i v_i^\top$. Therefore, we established the connection between PCA and SVD.

□

- **What's the difference between PCA and SVD when both of them are used for reduce the dimensionality.**

  Solution:

  Dimensionality reduction by PCA is given by the representation of the matrix of points by a small number of its eigenvectors, in this form we can approximate the data in a way that minimizes the root-mean-square error for the given number of columns in the representing matrix.

  Using SVD for dimensionality reduction is different. In a complete SVD for a matrix, U and V are typically as large as the original. To use fewer columns for U and V we delete the columns corresponding to the smallest singular values from U, V and Σ. This choice minimizes the error in reconstructing the original matrix from the modified U, V and Σ.

  □

(b) [30pts] Implementation of SVD (Case B)

Take the same wine quality data we used in the SVM homework. Apply SVD at first,

```
# <r code> ============================================================================ #
#                                                                    df.cent = X: U S V^{\top}
                                                  # S: square root of eigenvalues of X^{\top} X
S <- diag( sqrt(eigen(t(df.cent) %*% df.cent)$values) )
                                                                  # U: eigenvectors of X X^{\top}
```

```r
U <- eigen(df.cent %*% t(df.cent))$vectors[ , 1:ncol(df.cent)]
                                                # V: eigenvectors of X^{\top} X
V <- eigen(t(df.cent) %*% df.cent)$vectors

par(mar = c(4, 4, 0, 0) + .1)                       # plotting eigenvector variances
plot(diag(S), type = "b", xlab = "Eigenvector", ylab = "Variance")
# </r code> ============================================================================ #
```

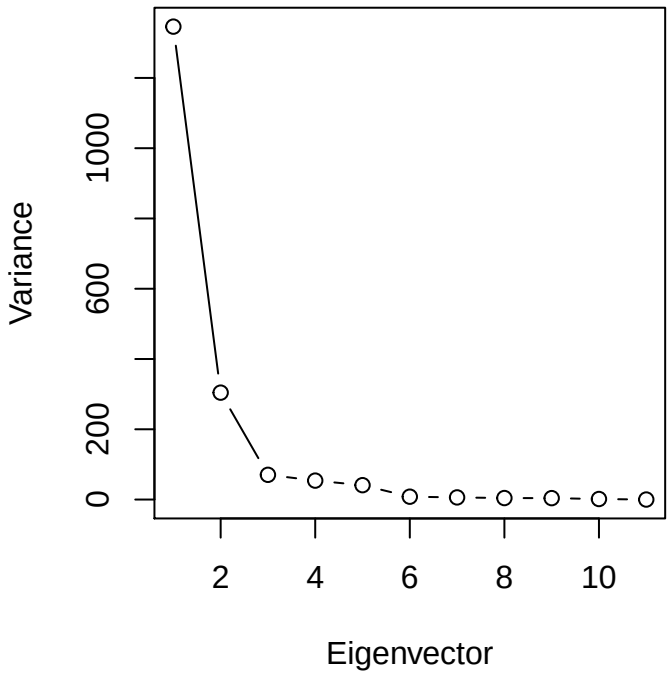

Figure 2: Eigenvalues distribution. Variance explained by each eigenvector.

We see in Figure 2 that the first principal component is responsable for almost all the variance explanation.

and then learn SVM from the reduced dimension space. Show whether SVD is helpful on improving your classification accuracy (choosing any one of the kernels is ok, and parameters can be set according to the search result in SVM homework).

```r
# <r code> ============================================================================ #
                                        # AUC for the model with the original data
aucs[1] <- auc(roc(as.numeric( df$quality ), as.numeric( svm.original$fitted )))

#                       computing the AUC for different numbers of singular vectors
for (i in 1:11) {
  s1 <- as.matrix(S[1:i, 1:i])
  u1 <- as.matrix(U[ , 1:i]) ; v1 <- as.matrix(V[ , 1:i])

  m.svd <- u1 %*% s1 %*% t(v1)                          # computing the SVD matrix
```

```
  model <- svm(df$quality ~ m.svd                                    # fitting the model
               , kernel = "linear", cross = 5, cost = .5, gamma = .001)
  aucs[i + 1] <- auc(roc(as.numeric( df$quality ), as.numeric( model$fitted )))
}
# </r code> ============================================================================ #
```

Table 2: AUC obtained with the original data and with different numbers of singular vectors.

Variable	**AUC**
Original data	0.7509267
1 singular vector	0.5952344
2 singular vectors	0.5964747
3 singular vectors	0.6029593
4 singular vectors	0.6041289
5 singular vectors	0.7239483
6 singular vectors	0.7356088
7 singular vectors	0.7368657
8 singular vectors	0.7380612
9 singular vectors	0.7485616
10 singular vectors	0.7495826
11 (all) singular vectors	0.7515988

With the original data, eleven variables, we have an AUC of 0.7509. Using only the first singular vector, responsable for almost all the variance, the AUC is 0.5965.

Considering more singular vectors the AUC increase, but not so much. However, when we consider 5 singular vectors we see a bigger increase in the AUC, reaching a value closer to the value obtained with all the data. With all the singular vectors the AUC is exactly the same that with the original data.

Conclusion. Here, with five singular vectors we're already able to reach a result, AUC, similar to the obtained using all the data.

The results, conclusions, for the PCA and SVD methodologies for this dataset are basically equal. With five PC's (in the PCA approach) or singular vectors (in the SVD approach) we're already able to reach a result similar to the obtained using all the data (eleven variables).

■
