Linear Algebra HW 6

gjd9961

October 2021

1 Problem 7.1

We say that a symmetric matrix $M \in \mathbb{R}^{n\times n}$ is **positive semi-definite** if for all **non-zero** $x \in \mathbb{R}^n$, $x^T Mx \geq 0$. Furthermore, a symmetric matrix $M \in \mathbb{R}^{n\times n}$ is **positive definite** if for all **non-zero** $x \in \mathbb{R}^n$, $x^T Mx > 0$. a) Let $M \in \mathbb{R}^{n\times n}$ be a symmetric matrix. Show that M is positive semi-definite if and only if its eigenvalues are all non-negative.

Since the Matrix M is a symmetric square matrix, it follows that M has an eigen decomposition of such that $M = PDP^T$. where P is the orthonormal basis that consists of the eigenvalues of M and D is a diagonal matrix who's entries correspond to the ordered pair of eigenvalues for the eigenvectors in P. Let the vector $y \in \mathbb{R}^n$ and y be the result of a matrix-vector product between P and x such that $y = Px$.

$$\begin{aligned} x^T Mx &= x^T(P^T DP) \\ x^T(P^T DP) &= (x^T P^T)D(Px) \\ (x^T P^T)D(Px) &= y^T Dy \\ y^T Dy &= \sum_{i=1}^{n} \lambda_i y_i^2 \quad \square \end{aligned} \tag{1}$$

We derived a summation which we obtained by showing that $x^T Mx$ results in the dot product between a vector y and $y\lambda$ (like so: $\langle y, \lambda y\rangle$). Since the y^2 in our summation can never be negative, the only possible way for our sum to be negative is if we have negative eigenvalues. If we only have positive eigenvalues, the result of the summation will always be positive, since our vector x is non zero, thus y must be too, and therefore $x^T Ax > 0$. If our eigenvalues are equal to 0, it is the only case where our transformation $x^T Ax = 0$. Thus, if we have non-negative eigenvalues, $x^T Ax \geq 0$ our matrix is positive semi definite. If we have all positive eigenvalues, then it is positive definite.

b) Consider J_n the $n \times n$ matrix of all ones (all entries equal to 1). Show that J_n is positive semi-definite using (**a**).

We know from homework 6 that if the rows of a matrix A sum to some number μ then μ is an eigenvalue of A.

Proof from HW 6: Let A be our $\mathbb{R}^{n\times n}$ matrix and $\mu \in \mathbb{R}$ such that for any integer i $(1 < i \leq n)$, $\sum_{j=1}^{n} A_{i,j} = \mu$. Let x be a vector in $\mathbb{R}^n$ where all the entries of x are equal to 1, $x_i = 1$ for any integer $0 < i \leq n$. Therefore:

$$\begin{aligned} Ax &= \mu x \\ Ax - \mu x &= 0 \\ (A - Id_n\mu)x &= 0 \quad \square \end{aligned} \tag{2}$$

In the case of our Matrix $J \in \mathbb{R}^{n\times n}$, every row will sum to n, so n will be an eigenvalue with corresponding eigenvalue derived by computing the following: $Ker(J - n \times Id_n)$. Also, since by definition, all of the columns are the same in matrix J, then we know there will be 1 linearly independent column, and $n-1$ lineraly dependent columns. Using Rank-Nullity, we know that if $Rank(J) = 1$ then $dim(Ker(J)) = n - 1$. If $dim(Ker(J)) = n - 1$, then we will have $n - 1$ eigenvectors that correspond to eigenvalue 0 (we could also say we have $eignevalue = 0$ with $multiplicity = n - 1$).

Using $J's$ eigendecomposition, J can be rewritten as $J = PDP^T$ where P is the basis of orthonormal eigenvectors and D is the matrix with eigenvalues on the diagonal and $0's$ elsewhere. In the case of J the matrix D will have only one real value, eigenvalue n, and the rest of the matrix will be filled with 0's, as J has eigenvalues 0 with multiplicity $n - 1$.

$$D = \begin{bmatrix} n & \dots & \dots & 0 \\ \vdots & \ddots & \dots & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \dots & \dots & 0 \end{bmatrix}$$

From part a) we know that if a matrix has eigenvalues ≥ 0, then it is positive semi definite. Since our matrix J has eigenvalues ≥ 0, it is positive semi definite.

c) Let $M \in \mathbb{R}^{n\times n}$ be a symmetric matrix. Show that there exists $\alpha > 0$ such that the matrix $M + \alpha_n$ is positive definite.

Let $A \in \mathbb{R}^{n\times n}$, $x \in \mathbb{R}$, and $Ax = \lambda x$, then we know that $\alpha + \lambda$ is an eigenvalue for the matrix $A + Id_n$ with eigenvector x:

$$\begin{aligned} (A + \alpha Id_n)x &= Ax + \alpha Id_n x \\ Ax + \alpha Id_n x &= \lambda x + \alpha x \\ \lambda x + \alpha x &= (\lambda + \alpha)x \\ \textbf{Therefore } (A + \alpha Id_n)x &= (\lambda + \alpha)x \quad \square \end{aligned} \tag{3}$$

If M wasn't already positive definite, we know from part a) that M must have at least one negative eigenvalue, or eigenvalues equal to 0. If we wanted to make the matrix M positive definite, we could take the eigendecomposition of M, $M = PDP^T$, and we could take the absolute value of the minimum eigenvector from D, add a very small number to it, and then add it to the diagonal of our matrix M. Let μ be some very very small number, greater than 0 $\mu > 0$.

$$\begin{aligned} \alpha &= |min(D_{1,1}, \ldots, D_{n,n})| + \mu \\ M_p &= M + (\alpha \times Id_n) \quad \square \end{aligned} \tag{4}$$

We know from the proof above that the eigenvalues of M_p are the displaced eigenvalues of M, where each eigenvalue of M_p is greater than its corresponding eigenvalue in M by α. This means AT THE MINIMUM, the minimum eigenvalue of M_p is some small number greater than 0, since

$$\begin{aligned} \alpha + min(eigenvalues\ of\ M) &> 0 \\ \mu + |min(eigenvalues\ of\ M)| + min(eigenvalues\ of\ M) &> 0 \\ \mu &> 0 \quad \square \end{aligned} \tag{5}$$

Therefore, M_p is the positive definite version of the matrix M. If we expressed M_p as its eigendecomposition $M_p = PDP^T$. What we will find that the entries in D will all be greater than 0, and since M_p has eigenvalues greater than 0 then M_p we know from part a) that it is positive definite.

2 Problem 7.2

Using PCA, we reduce the dimension of a dataset $a_1, \ldots, a_n \in \mathbb{R}^d$ of mean zero, to get a dimensionally reduced dataset $b_1, \ldots, b_n \in \mathbb{R}^k$, for some $1 \leq k \leq d$. We note A the $n \times d$ matrix

$$A = \begin{pmatrix} - & a_1^\top & - \\ & \vdots & \\ - & a_n^\top & - \end{pmatrix}.$$

a) We know that the dataset of $a_1, \ldots, a_n$ has mean 0, so lets use that to our advantage. Let S be the covariance matrix of dataset A, and $S = A^T A$. Let $v = \{v_1, \ldots, v_d\}$ be the orthonormal family of vectors that constitues the basis of the span of the eigenvectors of S. We can do the following:

$$
\begin{aligned}
b_1 + \cdots + b_n = \begin{pmatrix} \langle v_1, a_1 \rangle \\ \dots \\ \langle v_k, a_1 \rangle \end{pmatrix} + \cdots + \begin{pmatrix} \langle v_1, a_n \rangle \\ \dots \\ \langle v_k, a_n \rangle \end{pmatrix} \qquad & \text{by definition of PCA} \\
\begin{pmatrix} \langle v_1, a_1 \rangle \\ \dots \\ \langle v_k, a_1 \rangle \end{pmatrix} + \cdots + \begin{pmatrix} \langle v_1, a_n \rangle \\ \dots \\ \langle v_k, a_n \rangle \end{pmatrix} = \sum_{i=1}^{n} V^T a_i \qquad & \text{rewriting expression} \\
\sum_{i=1}^{n} V^T a_i = V^T \sum_{i=1}^{n} a_i \qquad & \text{via property of matrix scalar operations} \\
V^T \sum_{i=1}^{n} a_i = V^T 0 \qquad & \text{All the means are 0} \\
= 0 \quad \square \qquad & \text{Therefore our means of } b_i\text{'s are 0}
\end{aligned}
\tag{6}
$$

b) Show that for all $i, j \in \{1, \dots, n\}$, we have

$$\|b_i - b_j\| \leq \|a_i - a_j\| .$$

This means that PCA shrinks the distances.

Lets define a few things, firstly S is the co-variance matrix where $S = A^T A$ and $v = \{v_1, \dots, v_d\}$ is orthonormal basis for S, and $\{v_i, \dots, v_k\} \subset \{v_1, \dots, v_d\}$ where $k \leq d$. Also, since $a_1, \dots, a_n$ are coordinates in the basis of v then every coordinate a_i where $i \in \{1, 2, \dots, n\}$ can be written as $a_i = v_1\langle v_1, a_i \rangle + \cdots + v_d \langle v_d, a_i \rangle$. Lastly, assume that $|| \cdot ||$ is the L_2 norm. Also, the following:

$$\text{Let: } b_1, \dots, b_n = \begin{pmatrix} \langle v_1, a_1 \rangle \\ \dots \\ \langle v_n, a_1 \rangle \end{pmatrix} + \cdots + \begin{pmatrix} \langle v_1, a_n \rangle \\ \dots \\ \langle v_n, a_n \rangle \end{pmatrix}$$

With that being said, lets consider

$$\|b_i - b_j\| \leq \|a_i - a_j\| .$$

One term at a time. Firstly, with $\|b_i - b_j\|$, the left hand side

$$\|b_i - b_j\| = \begin{pmatrix} \langle v_1, a_i - a_j \rangle \\ \dots \\ \langle v_k, a_i - a_j \rangle \end{pmatrix}$$

And

$$\|b_i - b_j\|^2 = (\sqrt{\sum_{i=1}^{k} \langle v_1, a_i - a_j \rangle^2})^2 = \sum_{l=1}^{k} \langle v_l, a_i - a_j \rangle^2$$

Now for the right hand side, firstly define the following

$$a_i = \langle v_i, a_i \rangle v_i + \cdots + \langle v_d, a_i \rangle v_d \text{ and } a_j = \langle v_i, a_j \rangle v_i + \cdots + \langle v_d, a_j \rangle v_d$$

Now:

$$a_i - a_j = (\langle v_1, a_i \rangle - \langle v_1, a_j \rangle) v_1 + \cdots + (\langle v_d, a_i \rangle - \langle v_d, a_j \rangle) v_d$$

And define $\alpha_1, \ldots, \alpha_d$ in the following way:

$$\textit{Let} \qquad \alpha_i = \langle v_i, a_i - a_j \rangle v_i \text{ and } \alpha_d = \langle v_d, a_i - a_j \rangle v_d$$

Therefore:

$$||a_i - a_j||^2 = \sum_{l=1}^{d} \langle v_l, a_i - a_j \rangle^2$$

Since we know $\{v_i, \ldots, v_k\} \subset \{v_1, \ldots, v_d\}$ where $k \leq d$ that What we have now is:

$$\begin{aligned} \sum_{l=1}^{k} \langle v_l, a_i - a_j \rangle^2 &\leq \sum_{l=1}^{d} \langle v_l, a_i - a_j \rangle^2 \\ ||b_i - b_j||^2 &\leq ||a_i - a_j||^2 \\ ||b_i - b_j|| &\leq ||a_i - a_j|| \quad \square \end{aligned} \tag{7}$$

As $||b_i - b_j||$ and $||a_i - a_j||$ are norms so they must be positive.

c) For $i \in \{1, \ldots, k\}$ we let

$$f^{(i)} = (b_{1,i}, b_{2,i}, \ldots, b_{n,i}) \in \mathbb{R}^n$$

be the vector made of all i^{th} components of the vectors $b_1, \ldots, b_n$. Show that for $i \neq j$, $f^{(i)} \perp f^{(j)}$. This means that the new features computed using PCA are uncorrelated.

Let S be the covariance matrix of dataset A, and $S = A^T A$. Let $v = \{v_1, \ldots, v_d\}$ be the orthonormal family of vectors that constitues the basis of the span of the eigenvectors of S with associated eigenvalues $\lambda_1, \ldots \lambda_n$.

$$\text{Let: } b_1, \ldots, b_n = \begin{pmatrix} \langle v_1, a_1 \rangle \\ \ldots \\ \langle v_k, a_1 \rangle \end{pmatrix} + \cdots + \begin{pmatrix} \langle v_1, a_n \rangle \\ \ldots \\ \langle v_k, a_n \rangle \end{pmatrix}$$

$$\textbf{Then: } f^i = \begin{pmatrix} \langle v_i, a_1 \rangle \\ \ldots \\ \langle v_i, a_n \rangle \end{pmatrix} = A v_i \text{ and } f^j = \begin{pmatrix} \langle v_j, a_1 \rangle \\ \ldots \\ \langle v_j, a_n \rangle \end{pmatrix} = A v_j$$

$$
\begin{aligned}
\langle Av_i, Av_j \rangle = 0 \quad & \text{Show this property} \\
(Av_i)^T (Av_j) = 0 \quad & \text{By definition of dot product} \\
v_i^T A^T A v_j = 0 \quad & \text{By definition of dot product} \\
v_i^T S v_j = 0 \quad & \text{By definition of the covariance matrix} \\
v_i^T \lambda_j v_j = 0 \quad & \text{By definition of matrix-eigenvector product} \\
\lambda_j \times \langle v_i, v_j \rangle = 0 \quad & \text{By definition of dot products} \\
\lambda_j \times 0 = 0 \quad & 0 \text{ if } i \neq j \text{ } 1 \text{ if } i = j
\end{aligned}
\tag{8}
$$

$$0 = 0 \quad \square$$

3 Problem 7.3

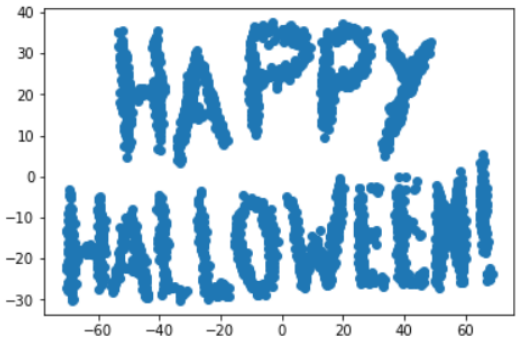
You have been given a mysterious dataset that may contain important informations! This dataset is a collection of $n = 6344$ points of dimension $d = 1000$. Investigate the structure of this dataset using PCA/plots... , and find out if the dataset contains any information.

The `zip` file `mysterious_data.zip` contains a text file containing the 6344×1000 data matrix. The *Jupyter notebook* `mysterious_data.ipynb` contains a function to read the text file.

You are not allowed to use any builtin PCA function: you have to do the all process by yourself (centering the data, computing the covariance matrix...). Of course, for computing eigenvalues/eigenvectors you will need to use the numpy library. The numpy function `numpy.linalg.eigh` is great to compute eigenvalues and eigenvectors of a symmetric matrix.

It is intended that you code in Python and use the provided Jupyter Notebook. Please only submit a pdf version of your notebook (right-click → 'print' → 'Save as pdf').

```
In [11]: x = output[:,0]
         y = output[:,1]*-1
         plt.scatter(x,y)
         plt.show()
```



In [14]:
```
%matplotlib inline
import matplotlib
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
import pandas as pd
```

In [15]:
```
# Load the data matrix
A = np.loadtxt('mysterious_data.txt')
n,d = A.shape
print(f'The matrix A contains {n} points in dimension {d}')
```

```
The matrix A contains 3000 points in dimension 1000
```

Each row of A corresponds to a datapoint.

In [16]:
```
#Define PCA Algorithm
def pca(data,k=0,var=0):

    n,d = data.shape
    data_copy = data
    #Center Data at 0, Each Column must have mean 0
    #Loop through every column
    for i in range(d):
        data_copy[:,i] = data_copy[:,i] -  np.mean(data_copy[:,i])


    #Compute Covariance Matrix
    Cov_matrix = data_copy.T @ data_copy

    #Calculate eigendecomp for Cov_Matrix
    vals, vectors = np.linalg.eigh(Cov_matrix)

    #Sort the eigenvalues descending order
    vals = vals[::-1]
    vectors = vectors[:,::-1]
    total_var = vals.sum()

    #If we wanted the least dimensions for a certain amount of variance explained

    #See if Var was passed to function and its valid
    if var > 0 and var <=1:
        tracker = 0
        eig_vals_of_interest, eig_vectors_of_interest = [], []
        for i in range(len(vals)):
            if tracker < var:
                tracker += vals[i] / total_var
                eig_vals_of_interest.append(vals[i])
        eig_vectors_of_interest = vectors[:,:len(eig_vals_of_interest)]

    #Check if we just wanted k dimensions
    elif k > 0 and k <= d:

        #If we wanted k dimensions, set the appropriate amount
        eig_vals_of_interest = vals[:k]
        eig_vectors_of_interest = vectors[:,:k]

    else:

        #if no other arguments were passed, return the eigen decomp
        return vals, vectors

    #Make sure data types are compatible
    eig_vectors_of_interest = np.array(eig_vectors_of_interest)
    eig_vals_of_interest = np.array(eig_vals_of_interest)
    percent_of_variance = eig_vals_of_interest.sum() / total_var
    #Compute the Principle Component Matrix
    new_data = data_copy@eig_vectors_of_interest
```

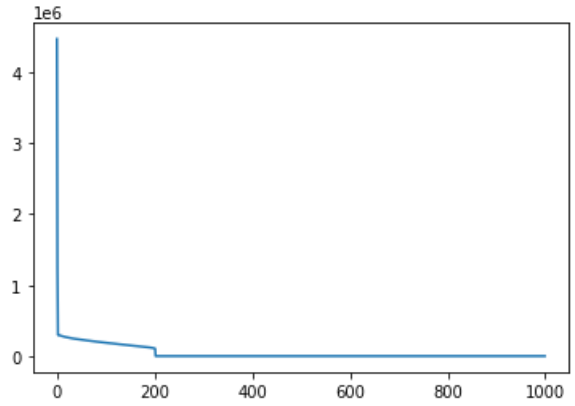
```
    #Return completed pca, eigenvalues, and eigenvectors
    return new data, eig vals of interest,eig vectors of interest, percent of variance
```

What does the distribution of eigenvalues of the covariance matrix look like?

In [17]:
```
vals, vectors = pca(A)
index = [i for i in range(len(vals))]
df = pd.DataFrame({'vals':vals})
sb.lineplot(index,vals)
print("It looks like the first few eigenvalues explain the majority of the variance")
print("It also appears that after 200 eigenvalues, the values are basically 0")
```

```
It looks like the first few eigenvalues explain the majority of the variance
It also appears that after 200 eigenvalues, the values are basically 0

C:\Users\Giulio\Anaconda3\lib\site-packages\seaborn\_decorators.py:43: FutureWarning: Pass t
he following variables as keyword args: x, y. From version 0.12, the only valid positional a
rgument will be `data`, and passing other arguments without an explicit keyword will result
in an error or misinterpretation.
  FutureWarning
```



How many principle components would we need to capture 50% of the variance?

In [18]:
```
#Call function, and shape
output, values, vectors,var_percent = pca(A,var=.5)
print("We would need", output.shape[1], "Principle Components")
```

```
We would need 67 Principle Components
```

How many principle components would we need to capture 80% of the variance?

In [19]:
```
#Call function, and shape
output, values, vectors,var_percent = pca(A,var=.8)
print("We would need", output.shape[1], "Principle Components")
n,d = output.shape
print(f'The matrix A contains {n} points in dimension {d}')
```

```
We would need 138 Principle Components
The matrix A contains 3000 points in dimension 138
```

How many principle components would we need to capture all of the variance?

In [20]:
```
#Call function, and shape
output, values, vectors,var_percent = pca(A,var=1)
print("We would need", output.shape[1], "Principle Components")
n,d = output.shape
print(f'The matrix A contains {n} points in dimension {d}')
```

```
We would need 1000 Principle Components
The matrix A contains 3000 points in dimension 1000
```

What does the 202 Principle Components 100% of the variance dataset look like?

In [21]:
```
df = pd.DataFrame(output)
df
```

Out[21]:

	0	1	2	3	4	5	6	7	8	9	...
0	52.286909	9.834370	0.291642	-9.550960	-5.360088	0.845779	-8.844195	3.416257	-3.595199	17.273449	...
1	-19.311017	27.148588	16.740921	-3.105850	-13.432818	-5.870330	-0.775702	14.039010	-9.678481	20.248258	...
2	-9.624371	-22.207301	20.915428	2.860974	-12.175551	-2.437017	5.323750	-7.498591	1.266475	4.515910	...
3	52.129243	20.519438	14.572979	8.574215	5.743858	-1.218111	2.632488	-3.157580	-16.452796	15.562120	...
4	-68.571024	29.082733	5.485627	13.705943	1.410991	12.654287	2.235758	5.276971	-8.876717	1.297797	...
...	...	...	...	...	...	...	...	...	...	...	...
2995	34.642930	-33.888549	-5.493122	-4.180623	-1.623344	-12.501368	11.759640	-0.110850	-0.296381	0.059233	...
2996	-28.119669	-25.601034	-4.838316	0.617135	3.923323	1.424779	-4.446596	9.295150	-10.511800	-2.395244	...
2997	68.100775	24.906250	2.048599	-7.729703	3.528660	15.116484	13.054016	1.872415	1.670500	1.116577	...
2998	46.352552	-29.236058	10.495395	-5.126592	5.137411	10.984298	-4.400079	-11.196389	-6.420155	-12.158058	...
2999	-19.709776	-12.184919	6.041056	10.553933	13.507182	-7.264965	5.221503	-6.863069	-0.178527	3.854670	...

3000 rows × 1000 columns

How much of the variance does 25 Principle Components Capture?

In [22]:
```
#Call function, and shape
output, values, vectors, var_percent = pca(A,k=25)
print("We would explain", np.round(var_percent*100,decimals=4), "% of the variance with 25 Prin
n,d = output.shape
print(f'The matrix A contains {n} points in dimension {d}')
```

```
We would explain 27.4701 % of the variance with 25 Principle Components
The matrix A contains 3000 points in dimension 25
```

How much of the variance does 50 Principle Components Capture?

In [23]:
```
#Call function, and shape
output, values, vectors, var_percent = pca(A,k=50)
print("We would explain", np.round(var_percent*100,decimals=4), "% of the variance with 50 Prin
n,d = output.shape
print(f'The matrix A contains {n} points in dimension {d}')
```

```
We would explain 41.3561 % of the variance with 50 Principle Components
The matrix A contains 3000 points in dimension 50
```

How much of the variance does 75 Principle Components Capture?

In [24]:
```
#Call function, and shape
output, values, vectors, var_percent = pca(A,k=75)
print("We would explain", np.round(var_percent*100,decimals=3), "% of the variance with 75 Prin
n,d = output.shape
print(f'The matrix A contains {n} points in dimension {d}')
```

```
We would explain 53.88 % of the variance with 75 Principle Components
The matrix A contains 3000 points in dimension 75
```

How much of the variance does 100 Principle Components Capture?

In [25]:
```
#Call function, and shape
output, values, vectors, var_percent = pca(A,k=100)
print("We would explain", np.round(var_percent*100,decimals=3), "% of the variance with 100 Pri
```

```
n,d = output.shape
print(f'The matrix A contains {n} points in dimension {d}')
```

```
We would explain 65.16 % of the variance with 100 Principle Components
The matrix A contains 3000 points in dimension 100
```

In [ ]:

In [ ]:

In [ ]:

4 Problem 7.4

Let $A \in \mathbb{R}^{n \times n}$ be a symmetric positive semi-definite matrix. Prove that there exists $B \in \mathbb{R}^{n \times n}$ positive semi-definite such that $A = B^2$.

Since we know that A is positive semi definite, we know that it has eigenvalues of at least 0. Therefore, we can take the square root of the eigenvalues. We also know that A is square symmetric, and therefore Since A is square symmetric, we know $A^k = PD^kP^T$, where D is the diagonal matrix of A's eigenvalues, and P, and P^T are the orthonormal basis of A's eigenvectors.

$$\text{Let } D = \begin{pmatrix} \lambda_1 & \dots & 0_{1,n} \\ 0 & \ddots & 0 \\ 0_{n,1} & \dots & \lambda_n \end{pmatrix} \text{ and therefore let } D^{\frac{1}{2}} = \begin{pmatrix} \sqrt{\lambda_1} & \dots & 0_{1,n} \\ 0 & \ddots & 0 \\ 0_{n,1} & \dots & \sqrt{\lambda_n} \end{pmatrix}$$

And therefore, let $B = PD^{\frac{1}{2}}P^T$.

$$\begin{aligned} A &= PDP^T \\ PDP^T &= PD^{\frac{1}{2}}P^TPD^{\frac{1}{2}}P^T \\ PDP^T &= BB \\ A &= B^2 \quad \square \end{aligned} \tag{9}$$

Since A is positive semi definite, B must also be positive semi definite, as the eigenvalues of A are greater or equal to 0, and the eigenvalues of B are equal to the square root of the eigenvalues of A, which of course will be positive.