# Practical Lecture 8 - PCA and Autoencoders

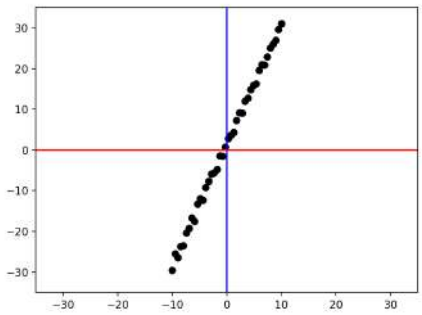
Luis Sa-Couto[1] and Andreas Wichert[2]

INESC-ID, Instituto Superior Tecnico, Universidade de Lisboa
{luis.sa.couto,andreas.wichert}@tecnico.ulisboa.pt

## 1 Principal Component Analysis (PCA)

Lower dimensional problems are generally easier to deal with than higher dimensional ones. Furthermore, it is often the case when data is represented with a needlessly large amount of features. For instance, think about images. A large number of pixels will correspond to a data collection with a large amount of features. However, images really live in a lower dimensional space since they capture a three-dimensional world. PCA is a technique that tries to discover the lower dimensional space where high dimensional points really live.

To exemplify the reasoning behind it, let us look at a group of two-dimensional points.




By looking at the data, we can see that although there are two features, the data seems to really vary accross a one dimensional diagonal line. The idea is to find that line and use it as our coordinates system.

The covariance matrix describes the variations of the data accross all dimensions. Through linear algebra, we can find the coordinate system that fits the data variation by computing the eigenvectors of the covariance matrix.

**Eigenvalues and Eigenvectors** An eigenvector of a matrix is a vector for which the transformation of that vector by that matrix yields the same vector multiplied by a scalar called the eigenvalue. In practice, this means the following:

$$\mathbf{Cu} = \lambda\mathbf{u} \iff (\mathbf{C} - \lambda\mathbf{I})\,\mathbf{u} = \mathbf{0}$$

Many eigenvectors exist with different scales. So, for the condition to met we must have an indetermined system with nontrivial solutions. This means that the determinant of the coefficients must equal zero. In fact, the determinant of the coefficients is called the characteristic polynomial:

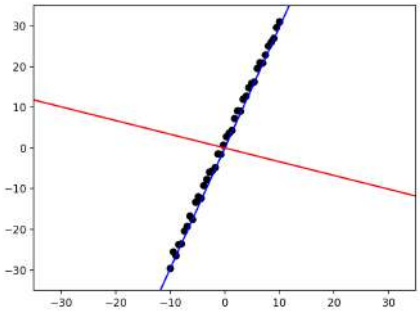
$$|\mathbf{C} - \lambda\mathbf{I}|$$

The eigenvalues $\lambda$ are such that:

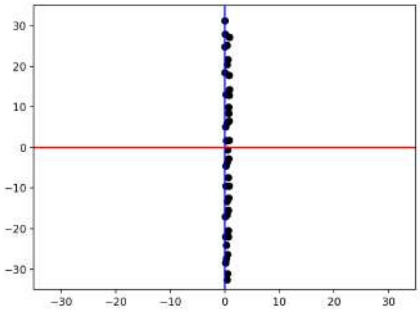
$$|\mathbf{C} - \lambda\mathbf{I}| = 0$$

Finding the list of eigenvalues we can find each corresponding eigenvector by choosing any vector that verifies $(\mathbf{C} - \lambda\mathbf{I})\,\mathbf{u} = \mathbf{0}$. However, it is usual to work with orthonormal eigenvectors, so, typically, one normalizes the chosen vector.

For our example, the eigenvectors define the two directions denoted by blue and red.




**The K-L transform** If we consider a coordinate system defined by the eigenvectors, we see in the figure below that mostly one feature varies while the other remains more or less constant.



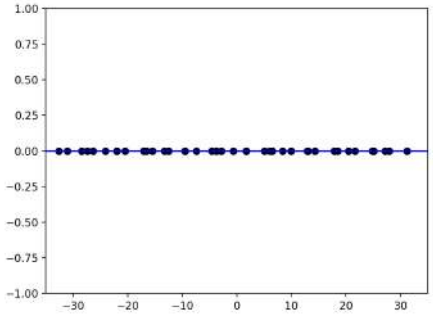
The matrix that rotates the coordinate system from the original one to the eigenvector defined one is called the K-L transform and is defined by placing the eigenvector on the columns:

$$\mathbf{U}_{KL} = \begin{pmatrix} | & | & & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_d \\ | & | & & | \end{pmatrix}$$

So, to map a point $\mathbf{x}$ on the original coordinate system to the new one we do:

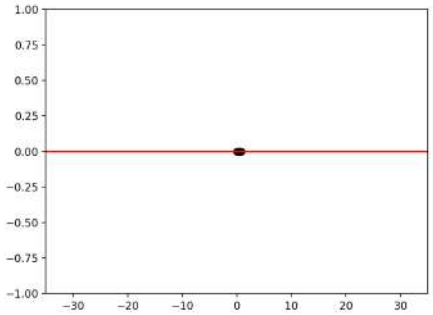
$$\mathbf{x}_{eig} = \mathbf{U}_{KL}^T \mathbf{x}$$

**Mapping to a lower dimensional space** Some dimensions are more relevant than others. The idea behind PCA is to discard the low importance dimensions. In our case, that would mean discard the red one and keep only the blue one:




As we can see, most variations is kept and few information was lost.

If we discard the most significant dimesion, we see that we lose all information since all points collapse:



With that said, the key idea here is to define importance. How to we know which eigendimensions to discard? The eigenvalues store this information. Larger eigenvalues mean more variation. So, one well known criterion, the Kaiser criterion, is to discard all dimensions with eigenvalue $\lambda_i < 1$.

So, to build the PCA transformation matrix, all we need to do is take the K-L matrix and remove the columns that correspond to eigenvectors we want to discard.

**Exercise:** Given the following training data:

$$\mathbf{x}^{(1)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \mathbf{x}^{(2)} = \begin{pmatrix} 4 \\ 0 \end{pmatrix}, \mathbf{x}^{(3)} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \mathbf{x}^{(4)} = \begin{pmatrix} 6 \\ 3 \end{pmatrix}$$

a) Compute the K-L transformation.

---

**Solution:**
First, we compute the mean vector:

$$\mu = \frac{1}{4}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 4 \\ 0 \end{pmatrix} + \begin{pmatrix} 2 \\ 1 \end{pmatrix} + \begin{pmatrix} 6 \\ 3 \end{pmatrix}\right) = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$$

Now, we can compute the covariance matrix. First, we compute the individual contributions from each point:

$$\mathbf{x}^{(1)} \rightarrow \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix} - \begin{pmatrix} 3 \\ 1 \end{pmatrix}\right)\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix} - \begin{pmatrix} 3 \\ 1 \end{pmatrix}\right)^T = \begin{pmatrix} -3 \\ -1 \end{pmatrix}\begin{pmatrix} -3 & -1 \end{pmatrix} = \begin{pmatrix} 9 & 3 \\ 3 & 1 \end{pmatrix}$$

$$\mathbf{x}^{(2)} \rightarrow \left(\begin{pmatrix} 4 \\ 0 \end{pmatrix} - \begin{pmatrix} 3 \\ 1 \end{pmatrix}\right)\left(\begin{pmatrix} 4 \\ 0 \end{pmatrix} - \begin{pmatrix} 3 \\ 1 \end{pmatrix}\right)^T = \begin{pmatrix} 1 \\ -1 \end{pmatrix}\begin{pmatrix} 1 & -1 \end{pmatrix} = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

$$\mathbf{x}^{(3)} \rightarrow \left(\begin{pmatrix} 2 \\ 1 \end{pmatrix} - \begin{pmatrix} 3 \\ 1 \end{pmatrix}\right)\left(\begin{pmatrix} 2 \\ 1 \end{pmatrix} - \begin{pmatrix} 3 \\ 1 \end{pmatrix}\right)^T = \begin{pmatrix} -1 \\ 0 \end{pmatrix}\begin{pmatrix} -1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

$$\mathbf{x}^{(4)} \rightarrow \left(\begin{pmatrix} 6 \\ 3 \end{pmatrix} - \begin{pmatrix} 3 \\ 1 \end{pmatrix}\right)\left(\begin{pmatrix} 6 \\ 3 \end{pmatrix} - \begin{pmatrix} 3 \\ 1 \end{pmatrix}\right)^T = \begin{pmatrix} 3 \\ 2 \end{pmatrix}\begin{pmatrix} 3 & 2 \end{pmatrix} = \begin{pmatrix} 9 & 6 \\ 6 & 4 \end{pmatrix}$$

Now, we can get the covariance matrix:

$$C = \frac{1}{4-1}\left(\begin{pmatrix} 9 & 3 \\ 3 & 1 \end{pmatrix} + \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 9 & 6 \\ 6 & 4 \end{pmatrix}\right) = \frac{1}{3}\begin{pmatrix} 20 & 8 \\ 8 & 6 \end{pmatrix} = \begin{pmatrix} \frac{20}{3} & \frac{8}{3} \\ \frac{8}{3} & 2 \end{pmatrix}$$

To get the intended transformation we need to compute $C$'s eigenvectors. We know that the eigenvalues $\lambda$ of a matrix are the roots of the characteristic polynomial:

$$|\mathbf{C} - \lambda\mathbf{I}| = 0$$

Since we are working with two-dimensional data we know that there will be two eigenvalues. So, our equation becomes:

$$
\begin{aligned}
|\mathbf{C} - \lambda\mathbf{I}| &= \left| \begin{pmatrix} \frac{20}{3} & \frac{8}{3} \\ \frac{8}{3} & 2 \end{pmatrix} - \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix} \right| \\
&= \left| \begin{pmatrix} \frac{20}{3} - \lambda & \frac{8}{3} - 0 \\ \frac{8}{3} - 0 & 2 - \lambda \end{pmatrix} \right| \\
&= \left(\frac{20}{3} - \lambda\right)(2 - \lambda) - \left(\frac{8}{3} - 0\right)\left(\frac{8}{3} - 0\right) \\
&= \frac{40}{3} - \frac{20}{3}\lambda - 2\lambda + \lambda^2 - \frac{64}{9} \\
&= \lambda^2 - \frac{26}{3}\lambda + \frac{56}{9} \\
&= 0
\end{aligned}
$$

Solving the second degree equation, we get:

$$\lambda_1 = 0.79 \vee \lambda_2 = 7.88$$

Having the eigenvalues, we can get the eigenvectors. If we have that $\lambda_1 = 0.79$, then the corresponding eigenvector $u_1 = \begin{pmatrix} u_{11} \\ u_{12} \end{pmatrix}$ will verify the following:

$$\mathbf{C}\mathbf{u}_1 = \lambda_1\mathbf{u}_1 \iff (\mathbf{C} - \lambda_1\mathbf{I})\,\mathbf{u}_1 = 0$$

Which yields:

$$
\begin{gathered}
(\mathbf{C} - \lambda_1\mathbf{I})\,\mathbf{u}_1 = \mathbf{0} \\
\left[\begin{pmatrix} \frac{20}{3} & \frac{8}{3} \\ \frac{8}{3} & 2 \end{pmatrix} - \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_1 \end{pmatrix}\right] \begin{pmatrix} u_{11} \\ u_{12} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\
\begin{pmatrix} \frac{20}{3} - \lambda_1 & \frac{8}{3} \\ \frac{8}{3} & 2 - \lambda_1 \end{pmatrix} \begin{pmatrix} u_{11} \\ u_{12} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\
\begin{pmatrix} \left(\frac{20}{3} - \lambda_1\right) u_{11} + \frac{8}{3} u_{12} \\ \frac{8}{3} u_{11} + (2 - \lambda_1)\, u_{12} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}
\end{gathered}
$$

So, the following condition must be met:

$$u_{12} = -\frac{3}{8}\left(\frac{20}{3} - \lambda_1\right) u_{11}$$

With that, a solution for the system will be:

$$\mathbf{u}_1 = \begin{pmatrix} u_{11} \\ -\frac{3}{8}\left(\frac{20}{3} - \lambda_1\right) u_{11} \end{pmatrix}$$

We can choose for instance $u_{11} = 1$ and get:

$$\mathbf{u}_1 = \begin{pmatrix} 1 \\ -\frac{3}{8}\left(\frac{20}{3} - 0.79\right) \end{pmatrix} = \begin{pmatrix} 1 \\ -2.2 \end{pmatrix}$$

However, it is usual to work with normalized eigenvectors:

$$\mathbf{u}_1 = \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|_2} = \begin{pmatrix} 0.4138 \\ -0.9104 \end{pmatrix}$$

For $\lambda_2 = 7.88$, the corresponding eigenvector $\mathbf{u}_2 = \begin{pmatrix} u_{21} \\ u_{22} \end{pmatrix}$ will verify the following:

$$\mathbf{C}\mathbf{u}_2 = \lambda_2 \mathbf{u}_2 \Longleftrightarrow (\mathbf{C} - \lambda_2 \mathbf{I})\, \mathbf{u}_2 = 0$$

Which yields:

$$(\mathbf{C} - \lambda_2 \mathbf{I})\, \mathbf{u}_2 = \mathbf{0}$$

$$\left[ \begin{pmatrix} \frac{20}{3} & \frac{8}{3} \\ \frac{8}{3} & 2 \end{pmatrix} - \begin{pmatrix} \lambda_2 & 0 \\ 0 & \lambda_2 \end{pmatrix} \right] \begin{pmatrix} u_{21} \\ u_{22} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} \frac{20}{3} - \lambda_2 & \frac{8}{3} \\ \frac{8}{3} & 2 - \lambda_2 \end{pmatrix} \begin{pmatrix} u_{21} \\ u_{22} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} \left(\frac{20}{3} - \lambda_2\right) u_{21} + \frac{8}{3} u_{22} \\ \frac{8}{3} u_{21} + (2 - \lambda_2)\, u_{22} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Again, the following condition must be met:

$$u_{22} = -\frac{3}{8} \left( \frac{20}{3} - \lambda_2 \right) u_{21}$$

So, a solution for the system will be:

$$\mathbf{u}_2 = \begin{pmatrix} u_{21} \\ -\frac{3}{8} \left( \frac{20}{3} - \lambda_1 \right) u_{21} \end{pmatrix}$$

We can choose for instance $u_{21} = 1$ and get:

$$\mathbf{u}_2 = \begin{pmatrix} 1 \\ -\frac{3}{8} \left( \frac{20}{3} - 7.88 \right) \end{pmatrix} = \begin{pmatrix} 1 \\ 0.45 \end{pmatrix}$$

However, it is usual to work with normalized eigenvectors:

$$\mathbf{u}_2 = \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|_2} = \begin{pmatrix} 0.9119 \\ 0.4104 \end{pmatrix}$$

Having the eigenvectors, we can place them on the columns of a matrix to build the K-L tranformation:

$$U_{K-L} = \begin{pmatrix} 0.4138 & 0.9119 \\ -0.9104 & 0.4104 \end{pmatrix}$$

---

b) What is the rotation applied to go from the original coordinate system to the eigenvector coordinate system?

---

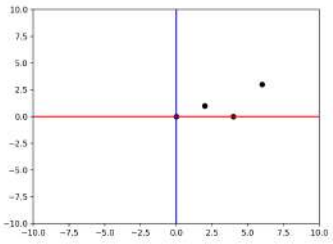
**Solution:**

$$\mathbf{U}\mathbf{e}_1 = \begin{pmatrix} 0.4138 & 0.9119 \\ -0.9104 & 0.4104 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \mathbf{u}_1$$

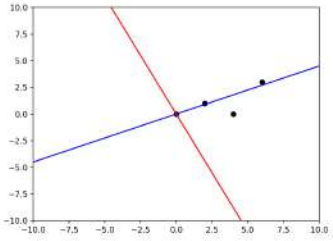
$$\mathbf{u}_1\mathbf{e}_1 = \left\| \begin{pmatrix} 0.4138 \\ -0.9104 \end{pmatrix} \right\|_2 \left\| \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\|_2 \cos\alpha = 0.4138$$

$$\alpha = \arccos(0.4138) = 1.1442rad \simeq 65deg$$

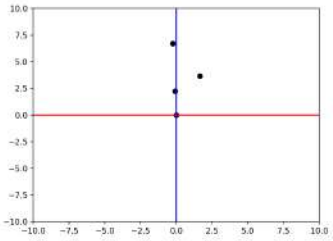
Let us graphically analyze what happened. The original points in the original coordinate system:



The eigenvectors are as follows:



Mapping the points to this new eigenspace through $\mathbf{x}_{eig} = \mathbf{U}^T\mathbf{x}$, we get:



Through this we can see that the 65 degree rotation was negative (i.e. counter-clockwise).

---

c) Which eigenvector is most significant?

---

**Solution:**
The most significant eigenvector is the largest one. In this case $\lambda_2$.

---

d) Can we apply the Kaiser criterion?

---

**Solution:**
Yes, since $\lambda_1 < 1$ we can apply the criterion and discard it.

---

e) Map the points onto the most significant dimension.

---

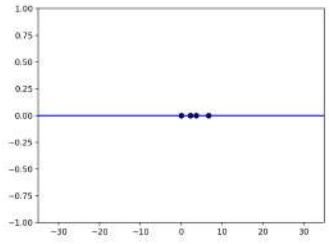
**Solution:**
We can map the points to the eigenspace through $\mathbf{x}_{eig} = \mathbf{U}^T\mathbf{x}$:

$$\begin{aligned}
\mathbf{x}_{eig}^{(1)} &= \begin{pmatrix} 0.4138 & 0.9119 \\ -0.9104 & 0.4104 \end{pmatrix}^T \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\
\mathbf{x}_{eig}^{(2)} &= \begin{pmatrix} 0.4138 & 0.9119 \\ -0.9104 & 0.4104 \end{pmatrix}^T \begin{pmatrix} 4 \\ 0 \end{pmatrix} = \begin{pmatrix} 1.66 \\ 3.65 \end{pmatrix} \\
\mathbf{x}_{eig}^{(3)} &= \begin{pmatrix} 0.4138 & 0.9119 \\ -0.9104 & 0.4104 \end{pmatrix}^T \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} -0.08 \\ 2.23 \end{pmatrix} \\
\mathbf{x}_{eig}^{(4)} &= \begin{pmatrix} 0.4138 & 0.9119 \\ -0.9104 & 0.4104 \end{pmatrix}^T \begin{pmatrix} 6 \\ 3 \end{pmatrix} = \begin{pmatrix} -0.25 \\ 6.70 \end{pmatrix}
\end{aligned}$$

PCA corresponds to keeping only the most significant dimensions. In this case, we discard the first dimension and get:

$$\begin{aligned}
\mathbf{x}_{PCA}^{(1)} &= \begin{pmatrix} 0 \end{pmatrix} \\
\mathbf{x}_{PCA}^{(2)} &= \begin{pmatrix} 3.65 \end{pmatrix} \\
\mathbf{x}_{PCA}^{(3)} &= \begin{pmatrix} 2.23 \end{pmatrix} \\
\mathbf{x}_{PCA}^{(4)} &= \begin{pmatrix} 6.70 \end{pmatrix}
\end{aligned}$$

Which yields:




---

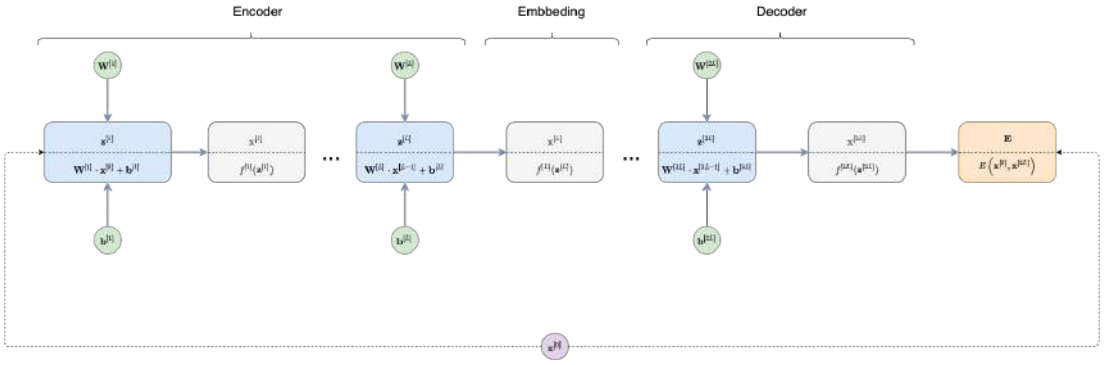
## 2 Autoencoders

Another way to embbed high dimensional data into a lower dimensional space is to use a neural network. To that end, we build a regular neural network that maps an input to an output with the dimension of the lower dimensional target space. Such network is called the encoder.

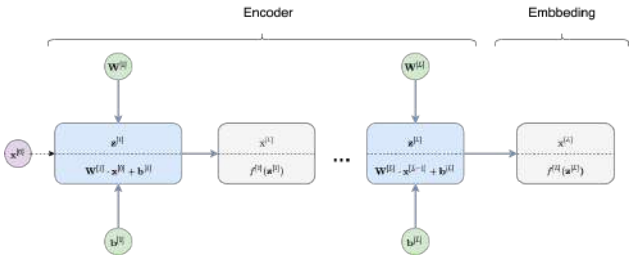
However, we do not know what the target should be. Unlike classification, there is no correct output for each point. Several different good embeddings may be possible.

The way to solve this network is to append a second neural network to the encoder. This new network takes the embbeding as input and outputs a vector of the same dimensionality of the original input point.

If we look at the two networks together as one full system called an autoencoder, we know that for an input vector $\mathbf{x}$ the output should, ideally, be the same vector $\mathbf{x}$. This architecture is presented in the figure below.




Using an autoencoder, we have a way to unsupervisedly learn the weights for both the encoder and the decoder. After doing so, we can separate the two modules and use the encoder as dimensionality reduction machinel. As the image below suggests, all we have to do to map an input to a lower dimensional embbeding is to do forward propagation on the encoder network.




**Exercise:** Consider the following point from a training collection of unsupervised data:

$$\mathbf{x} = \begin{pmatrix} 8 \\ 0 \\ 4 \end{pmatrix}$$

**a)** Build a one hidden layer autoencoder to map this data to two dimensions. Initialize all weights to 1 and all biases to zero. Use linear activation functions. Specify all forward operations, all backward operations and the loss function.

---

**Solution:**

The autoencoder has an input layer with 3 units, an hidden layer with 2 units and an output layer with 3 outputs. So, the initialized parameters are as follows:

$$\mathbf{W}^{[1]} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

$$\mathbf{b}^{[1]} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\mathbf{W}^{[2]} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix}$$

$$\mathbf{b}^{[2]} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

The forward operations are as follows:

$$\mathbf{z}^{[1]} = \mathbf{W}^{[1]}\mathbf{x}^{[0]} + \mathbf{b}^{[1]}$$

$$\mathbf{x}^{[1]} = \mathbf{z}^{[1]}$$

$$\mathbf{z}^{[2]} = \mathbf{W}^{[2]}\mathbf{x}^{[1]} + \mathbf{b}^{[2]}$$

$$\mathbf{x}^{[2]} = \mathbf{z}^{[2]}$$

$$E\left(\mathbf{x}^{[2]}, \mathbf{t}\right) = \frac{1}{2}\left\|\mathbf{x}^{[2]} - \mathbf{t}\right\|_2^2$$

The backward operations are as follows:

$$\frac{\partial E}{\partial \mathbf{x}^{[2]}}\left(\mathbf{t}, \mathbf{x}^{[2]}\right) = \frac{\partial E}{\partial\left(\mathbf{x}^{[2]}-\mathbf{t}\right)^2} \frac{\partial\left(\mathbf{x}^{[2]}-\mathbf{t}\right)^2}{\partial\left(\mathbf{x}^{[2]}-\mathbf{t}\right)} \frac{\partial\left(\mathbf{x}^{[2]}-\mathbf{t}\right)}{\partial \mathbf{x}^{[2]}} \quad = \frac{1}{2}\left[2\left(\mathbf{x}^{[2]}-\mathbf{t}\right)\right] = \mathbf{x}^{[2]}-\mathbf{t}$$

$$\frac{\partial \mathbf{x}^{[2]}}{\partial \mathbf{z}^{[2]}}\left(\mathbf{z}^{[2]}\right) = \mathbf{1}$$

$$\frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{W}^{[2]}}\left(\mathbf{W}^{[2]}, \mathbf{b}^{[2]}, \mathbf{x}^{[1]}\right) = \mathbf{x}^{[1]}$$

$$\frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{b}^{[2]}}\left(\mathbf{W}^{[2]}, \mathbf{b}^{[2]}, \mathbf{x}^{[1]}\right) = \mathbf{1}$$

$$\frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{x}^{[1]}}\left(\mathbf{W}^{[2]}, \mathbf{b}^{[2]}, \mathbf{x}^{[1]}\right) = \mathbf{W}^{[2]}$$

$$\frac{\partial \mathbf{x}^{[1]}}{\partial \mathbf{z}^{[1]}}\left(\mathbf{z}^{[1]}\right) = \mathbf{1}$$

$$\frac{\partial \mathbf{z}^{[1]}}{\partial \mathbf{W}^{[1]}}\left(\mathbf{W}^{[1]}, \mathbf{b}^{[1]}, \mathbf{x}^{[0]}\right) = \mathbf{x}^{[0]}$$

$$\frac{\partial \mathbf{z}^{[1]}}{\partial \mathbf{b}^{[1]}}\left(\mathbf{W}^{[1]}, \mathbf{b}^{[1]}, \mathbf{x}^{[0]}\right) = \mathbf{1}$$

$$\frac{\partial \mathbf{z}^{[1]}}{\partial \mathbf{x}^{[0]}}\left(\mathbf{W}^{[1]}, \mathbf{b}^{[1]}, \mathbf{x}^{[0]}\right) = \mathbf{W}^{[1]}$$

---

**b)** Use a learning rate of 0.001 for an iteration of stochastic gradient descent backpropagation using the half-squared error.

---

**Solution:**

In an autoencoder the target equals the input. So, we have to do the iteration with $\mathbf{x} = \begin{pmatrix} 8 \\ 0 \\ 4 \end{pmatrix}$ and $\mathbf{t} = \begin{pmatrix} 8 \\ 0 \\ 4 \end{pmatrix}$:

$$\mathbf{z}^{[1]} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 8 \\ 0 \\ 4 \end{pmatrix} \quad = \begin{pmatrix} 12 \\ 12 \end{pmatrix}$$

$$\mathbf{x}^{[1]} = \begin{pmatrix} 12 \\ 12 \end{pmatrix} \qquad = \begin{pmatrix} 12 \\ 12 \end{pmatrix}$$

$$\mathbf{z}^{[2]} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 12 \\ 12 \end{pmatrix} \quad = \begin{pmatrix} 24 \\ 24 \\ 24 \end{pmatrix}$$

$$\mathbf{x}^{[2]} = \begin{pmatrix} 24 \\ 24 \\ 24 \end{pmatrix} \qquad = \begin{pmatrix} 24 \\ 24 \\ 24 \end{pmatrix}$$

Now, we can compute the deltas:

$$
\begin{aligned}
\delta^{[2]} &= \frac{\partial E}{\partial \mathbf{x}^{[2]}} \circ \frac{\partial \mathbf{x}^{[2]}}{\partial \mathbf{z}^{[2]}} \\
&= \left(\mathbf{x}^{[2]} - \mathbf{t}\right) \circ \mathbf{1} \\
&= \left(\begin{pmatrix} 24 \\ 24 \\ 24 \end{pmatrix} - \begin{pmatrix} 8 \\ 0 \\ 4 \end{pmatrix}\right) \circ \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \\
&= \begin{pmatrix} 16 \\ 24 \\ 20 \end{pmatrix}
\end{aligned}
$$

$$
\begin{aligned}
\delta^{[1]} &= \frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{x}^{[1]}}^T \cdot \delta^{[2]} \circ \frac{\partial \mathbf{x}^{[l]}}{\partial \mathbf{z}^{[l]}} \\
&= \left(\mathbf{W}^{[2]}\right)^T \cdot \delta^{[2]} \circ \mathbf{1} \\
&= \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 16 \\ 24 \\ 20 \end{pmatrix} \circ \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\
&= \begin{pmatrix} 60 \\ 60 \end{pmatrix}
\end{aligned}
$$

The gradients become:

$$
\begin{aligned}
\frac{\partial E}{\partial \mathbf{W}^{[1]}} &= \delta^{[1]} \cdot \frac{\partial \mathbf{z}^{[1]}}{\partial \mathbf{W}^{[1]}}^T \\
&= \delta^{[1]} \cdot \left(\mathbf{x}^{[0]}\right)^T \\
&= \begin{pmatrix} 60 \\ 60 \end{pmatrix} \cdot \begin{pmatrix} 8 & 0 & 4 \end{pmatrix} \\
&= \begin{pmatrix} 480 & 0 & 204 \\ 480 & 0 & 240 \end{pmatrix}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial E}{\partial \mathbf{b}^{[1]}} &= \delta^{[1]} \circ \frac{\partial \mathbf{z}^{[1]}}{\partial \mathbf{b}^{[1]}} \\
&= \delta^{[1]} \circ \mathbf{1} \\
&= \begin{pmatrix} 60 \\ 60 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\
&= \begin{pmatrix} 60 \\ 60 \end{pmatrix}
\end{aligned}
$$

$$\begin{aligned}
\frac{\partial E}{\partial \mathbf{W}^{[2]}} &= \delta^{[2]} \cdot \frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{W}^{[2]}}^T \\
&= \delta^{[2]} \cdot \left(\mathbf{x}^{[1]}\right)^T \\
&= \begin{pmatrix} 16 \\ 24 \\ 20 \end{pmatrix} \cdot \begin{pmatrix} 12 & 12 \end{pmatrix} \\
&= \begin{pmatrix} 192 & 192 \\ 288 & 288 \\ 240 & 240 \end{pmatrix}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial E}{\partial \mathbf{b}^{[2]}} &= \delta^{[2]} \circ \frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{b}^{[2]}} \\
&= \delta^{[2]} \circ \mathbf{1} \\
&= \begin{pmatrix} 16 \\ 24 \\ 20 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \\
&= \begin{pmatrix} 16 \\ 24 \\ 20 \end{pmatrix}
\end{aligned}$$

So, we get the updates:

$$\begin{aligned}
\mathbf{W}^{[1]} &= \mathbf{W}^{[1]} - \eta \frac{\partial E}{\partial \mathbf{W}^{[1]}} \\
&= \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} - 0.001 \begin{pmatrix} 480 & 0 & 240 \\ 480 & 0 & 240 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} - \begin{pmatrix} 0.480 & 0 & 0.240 \\ 0.480 & 0 & 0.240 \end{pmatrix} \\
&= \begin{pmatrix} 0.52 & 1 & 0.76 \\ 0.52 & 1 & 0.76 \end{pmatrix}
\end{aligned}$$

$$\begin{aligned}
\mathbf{b}^{[1]} &= \mathbf{b}^{[1]} - \eta \frac{\partial E}{\partial \mathbf{b}^{[1]}} \\
&= \begin{pmatrix} 0 \\ 0 \end{pmatrix} - 0.001 \begin{pmatrix} 60 \\ 60 \end{pmatrix} \\
&= \begin{pmatrix} -0.06 \\ -0.06 \end{pmatrix}
\end{aligned}$$

$$\begin{aligned}
\mathbf{W}^{[2]} &= \mathbf{W}^{[2]} - \eta \frac{\partial E}{\partial \mathbf{W}^{[2]}} \\
&= \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} - 0.001 \begin{pmatrix} 192 & 192 \\ 288 & 288 \\ 240 & 240 \end{pmatrix} \\
&= \begin{pmatrix} 0.808 & 0.808 \\ 0.712 & 0.712 \\ 0.760 & 0.760 \end{pmatrix}
\end{aligned}$$

$$\begin{aligned}\mathbf{b}^{[2]} &= \mathbf{b}^{[2]} - \eta \frac{\partial E}{\partial \mathbf{b}^{[2]}} \\ &= \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} - 0.001 \begin{pmatrix} 16 \\ 24 \\ 20 \end{pmatrix} \\ &= \begin{pmatrix} -0.016 \\ -0.024 \\ -0.020 \end{pmatrix}\end{aligned}$$

---

**c)** Encode the point with the resulting autoencoder. That is, use it to map $\mathbf{x}$ to the two-dimensional manifold discovered in **b)**.

---

**Solution:**
All we need to do is to perform forward propagation until the hidden layer:

$$\begin{aligned}\mathbf{z}^{[1]} &= \begin{pmatrix} 0.52 & 1 & 0.76 \\ 0.52 & 1 & 0.76 \end{pmatrix} \begin{pmatrix} 8 \\ 0 \\ 4 \end{pmatrix} + \begin{pmatrix} -0.06 \\ -0.06 \end{pmatrix} &&= \begin{pmatrix} 7.14 \\ 7.14 \end{pmatrix} \\ \mathbf{x}^{[1]} &= \begin{pmatrix} 7.14 \\ 7.14 \end{pmatrix} &&= \begin{pmatrix} 7.14 \\ 7.14 \end{pmatrix}\end{aligned}$$

---

**d)** Decode the encoding from **c)** and measure the squared error.

---

**Solution:**
To decode, all we need to do is to perform forward propagation from the hidden layer until the output layer:

$$\begin{aligned}\mathbf{z}^{[2]} &= \begin{pmatrix} 0.808 & 0.808 \\ 0.712 & 0.712 \\ 0.760 & 0.760 \end{pmatrix} \begin{pmatrix} 7.14 \\ 7.14 \end{pmatrix} + \begin{pmatrix} -0.016 \\ -0.024 \\ -0.020 \end{pmatrix} &&= \begin{pmatrix} 11.52224 \\ 10.14336 \\ 10.83280 \end{pmatrix} \\ \mathbf{x}^{[2]} &= \begin{pmatrix} 11.52224 \\ 10.14336 \\ 10.83280 \end{pmatrix}\end{aligned}$$

The error is simply the loss function applied to the original point and the decoded version:

$$E\left(\mathbf{x}^{[2]}, \mathbf{t}\right) = \frac{1}{2} \left\| \begin{pmatrix} 11.52224 \\ 10.14336 \\ 10.83280 \end{pmatrix} - \begin{pmatrix} 8 \\ 0 \\ 4 \end{pmatrix} \right\|_2^2 = 80.99$$

As we can see, one iteration was not nearly enough. In practice, we would run SGD on several points for many epochs.

---

## 3 Thinking Questions

a) Name possible applications of an autoencoder

---

**Solution:**

• Dimensionality reduction:

• Generating Sparse Codes

• Pretraining: An autoencoder can be used to pretrain another neural network, e.g. a classifier. For instance, we could take the trained encoder, add a few dense layers for prediction and re-train the resulting model end-to-end with a prediction loss such as the binary cross entropy.

• Outlier detection: The reconstruction error is often large for inputs which are very different from the usual patterns observed in the training data. This can be used for outlier/anomaly detection by simply using the reconstruction error as an outlier score.

• Generative model: The learned decoder can be used as a generative model. Similarly as a GAN, it can be used for generating additional synthetic training samples, which is particularly useful in a setting where only few labeled samples are available. The Variational Autoencoder (VAE) is especially suited for this, since the latent dimensions are independently Gaussian distributed with unit variance, so we can just sample from these distributions and decode the resulting latent vector.

---

b) Consider a linear autoencoder without regularization or denoising. What happens if the number of dimensions of the latent space is set larger than the dimensionality of the data space.

---

**Solution:**

In this case, the autoencoder will simply learn the identity function and achieve zero reconstruction error. Since the latent space has a sufficient number of dimensions, the autoencoder can just copy the input into the latent space and back. This is a general problem if the autoencoder is given too much capacity. By introducing a bottleneck, the autoencoder is forced to learn compact and expressive representations. One can as well introduce l2 or l1 regularization to constraint the capacity of the Autoencoder.

---

c) When is an autoencoder equivalent to a PCA? What are the main differences?

---

**Solution:**

Linear units where the number of the hidden units corresponds to the number of the principal components (the eigenvectors with large eigenvalues) In PCA

the eigenvalues indicate which dimensions are important. In a linear autoencoder the number of hidden units has to be determined by experiments