# Embeddings and their usage

# Embeddings

1. Low dimensional representation of sequential data.
2. The data can be reconstructed(or closely approximated) from the embedding itself.
3. Similar sequences (e.g. highly aligned sequences) should have embeddings with low distances.
4. Dissimilar sequences (e.g. less aligned sequences) should have embeddings with high distances.
5. Embeddings are different from compressions since compressions do not necessarily have the above 2 properties.
6. Embeddings used for similarity checking and clustering.

# Principal Component Analysis

1. Create the covariance matrix of the sequence array

2. Find the eigenvalues and the corresponding eigenvectors from the covariance matrix

3. Sort the eigenvectors in the decreasing order of their eigenvalues

4. Dot product the sequence array with the first eigenvector to get the sequence array along the principal component.

5. Repeat step 4 for the first k eigenvectors to obtain the data re-oriented along k principal components.

6. The data along the k principal components act as the k dimensional embedding of the sequence array.

# Covariance matrix

$$\operatorname{corr}(\mathbf{X}) = \begin{bmatrix} 1 & \frac{\mathrm{E}[(X_1-\mu_1)(X_2-\mu_2)]}{\sigma(X_1)\sigma(X_2)} & \cdots & \frac{\mathrm{E}[(X_1-\mu_1)(X_n-\mu_n)]}{\sigma(X_1)\sigma(X_n)} \\ \frac{\mathrm{E}[(X_2-\mu_2)(X_1-\mu_1)]}{\sigma(X_2)\sigma(X_1)} & 1 & \cdots & \frac{\mathrm{E}[(X_2-\mu_2)(X_n-\mu_n)]}{\sigma(X_2)\sigma(X_n)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\mathrm{E}[(X_n-\mu_n)(X_1-\mu_1)]}{\sigma(X_n)\sigma(X_1)} & \frac{\mathrm{E}[(X_n-\mu_n)(X_2-\mu_2)]}{\sigma(X_n)\sigma(X_2)} & \cdots & 1 \end{bmatrix}.$$

# Eigen values and Eigen vectors

If there exists a **square matrix** called ***A***, a **scalar** ***λ***, and a **non-zero vector** **v**, then ***λ*** **is the eigenvalue** and **v is the eigenvector** if the following equation is satisfied:

$$\mathbf{Av} = \lambda\mathbf{v}$$

In other words, if matrix *A* times the vector **v** is equal to the scalar $\lambda$ times the vector **v**, then $\lambda$ is the eigenvalue of **v**, where **v** is the eigenvector.

# PCA example

```
[ ]  1 from sklearn.decomposition import PCA
```

```
[ ]  1 pca = PCA(n_components=64)
```

```
     1 Start coding or generate with AI.
```

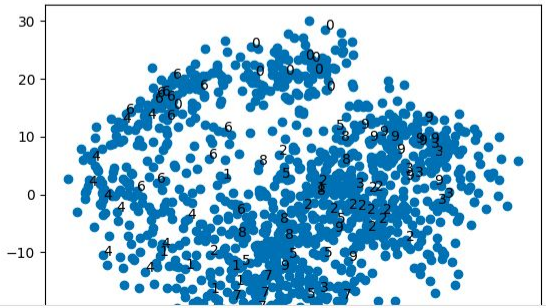
```
[ ]  1 x_new = pca.fit_transform(x_array)
     2 x_new.shape
```

```
(1257, 64)
```

```
[ ]  1 y_train[0]
```

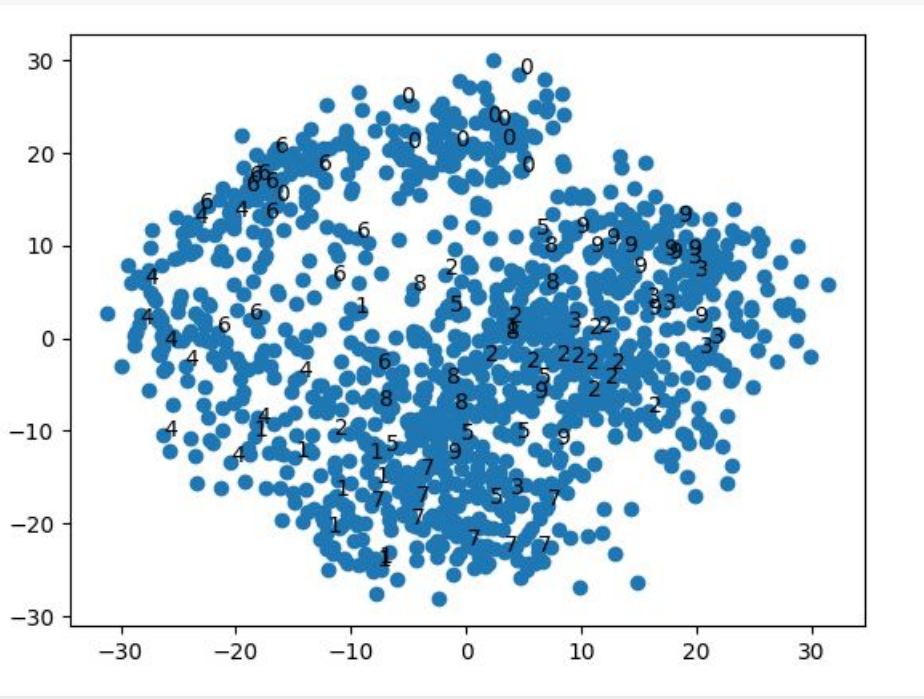
```
2
```

```
[ ]  1 import matplotlib.pyplot as plt
     2 plt.scatter(x_new[:,0],x_new[:,1])
     3 for i in range(100):
     4     plt.annotate(y_train[i], (x_new[i,0], x_new[i,1]))
     5 plt.show()
```

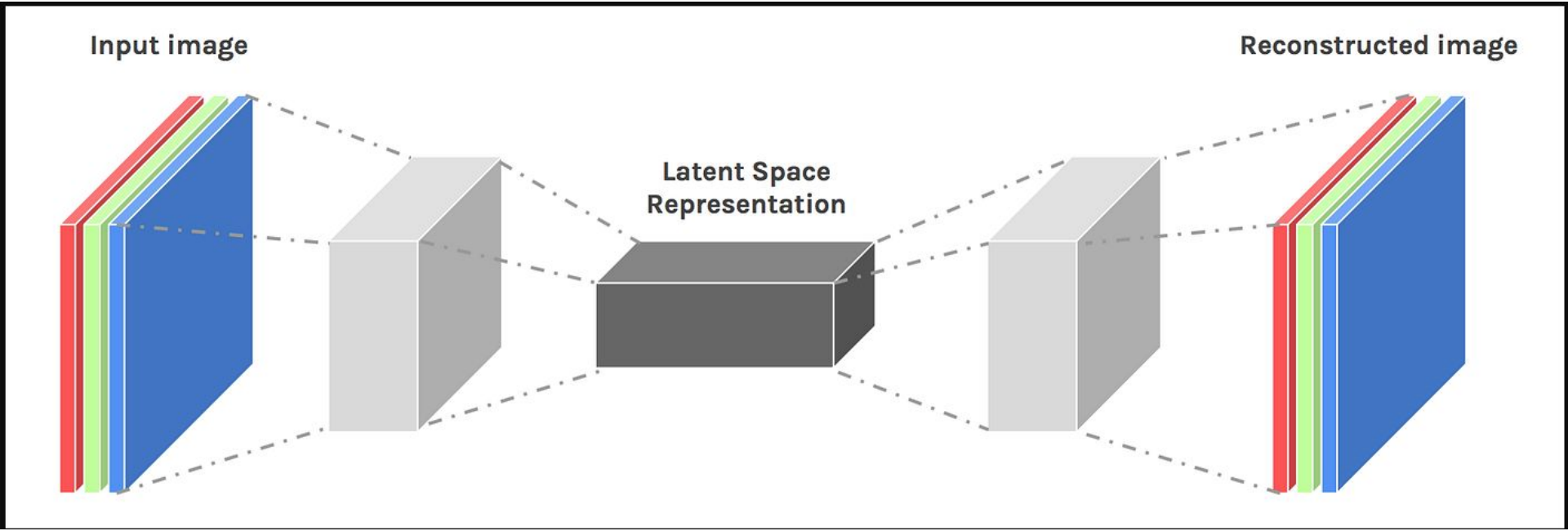



Mar 11 11:58

# Principal components from 64 dimensional MNIST dataset



# Convolutional autoencoder to create sequence embeddings



# Autoencoder based embeddings

1. Input and Output should be the same.
2. 3 parts: encoder - bottleneck - decoder
3. Should have high reconstruction accuracy
4. No skip connections bypassing the bottleneck layer.
5. Bottleneck layer should be lower dimensional than the input layer to obtain a high signal:noise ratio.
6. Other properties of embeddings should be followed.