Machine Learning 2018 – Visualization

Kien C Nguyen

December 12, 2018

Contents

Why visualization

- Graphs help us understand data, especially qualitative aspects of data, possibly more quickly and easily
- Graphs are a powerful tool to summarize data
- Graphs help us identify patterns, spot outliers, detect corrupt data
- Graphs help us see the relationships among features, and between features and labels, therefore help us in model selection and feature selection.

Contents

Matplotlib

- Matplotlib is a popular plotting library for Python
- The matplotlib provides a context, one in which one or more plots can be drawn before the image is shown or saved to file. The context can be accessed via functions on pyplot. The context can be imported as follows:

```
import matplotlib import pyplot
```

- There is some convention to import this context and name it *plt*

```
import matplotlib.pyplot as plt
```

Line plots

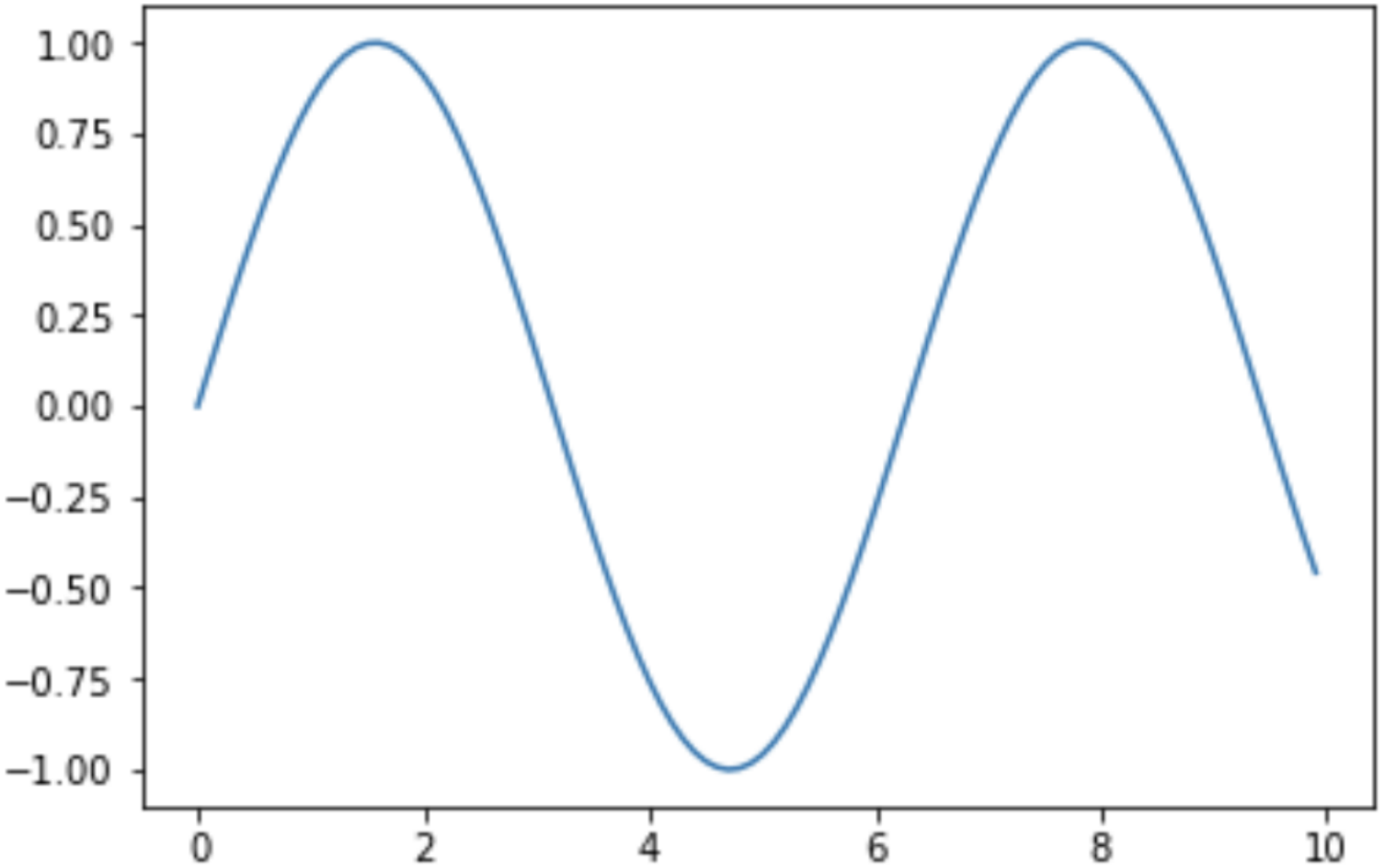
- A line plot is generally used to present observations collected at regular intervals.
- The x-axis represents the regular interval, such as time.
- The y-axis shows the observations, ordered by the x-axis and connected by a line.
- A line plot can be created by calling the plot() function and passing the x-axis data for the regular interval, and y-axis for the observations.

```
# create line plot
pyplot.plot(x, y)
```

Line plots – Plotting a sine

```
import numpy as np
import matplotlib.pyplot as plt
# consistent interval for x-axis
x = [x*0.1 for x in range(100)]
# function of x for y-axis
y = np.sin(x)
# create line plot
plt.plot(x, y)
# show line plot
plt.show()
```

Line plots – Plotting a sine



Bar Chart

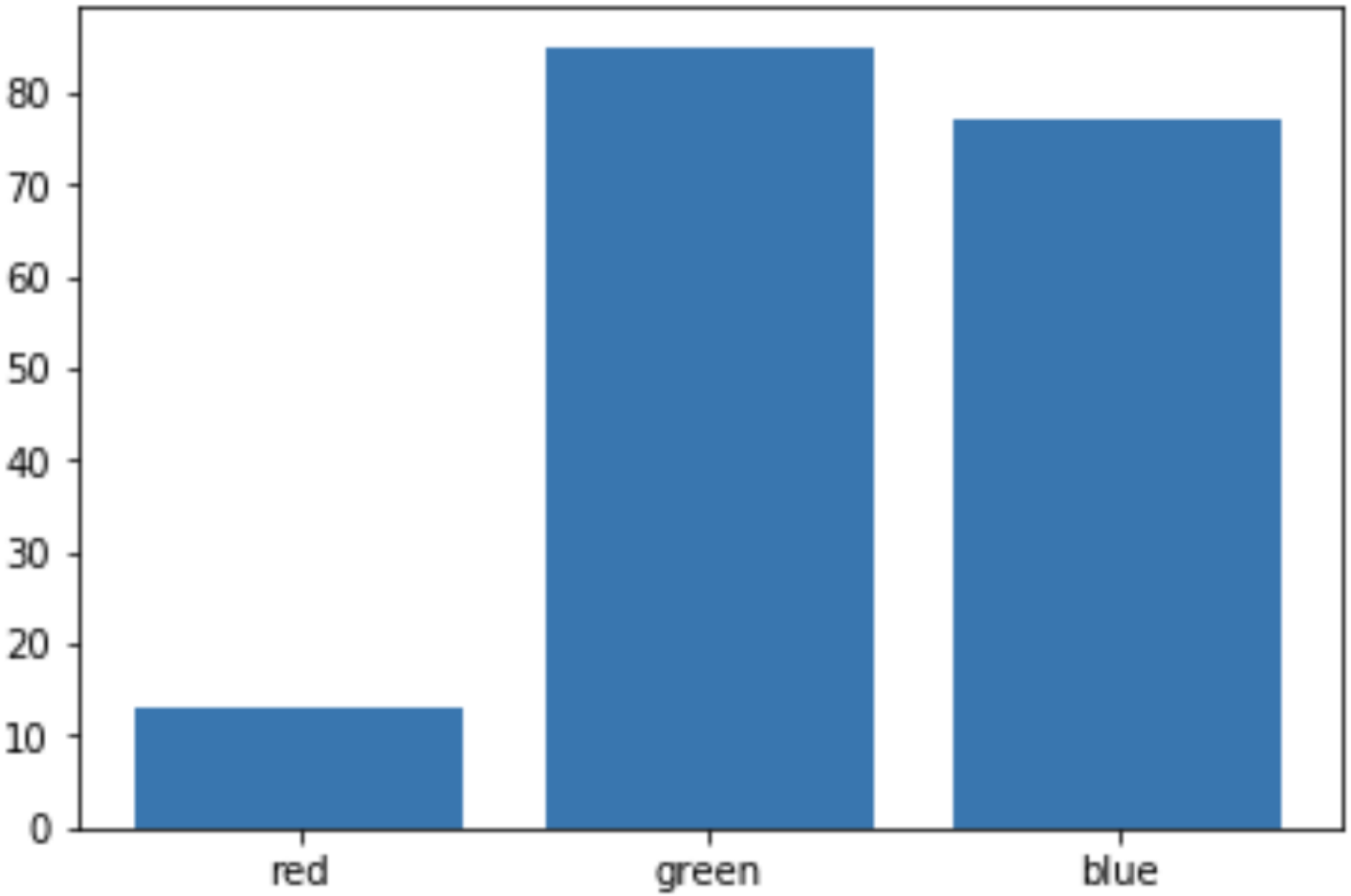
- A bar chart is generally used to present relative quantities for multiple categories.
- The x-axis represents the categories and are spaced evenly.
- The y-axis represents the quantity for each category and is drawn as a bar from the baseline to the appropriate level on the y-axis.
- A bar chart can be created by calling the bar() function and passing the category names for the x-axis and the quantities for the y-axis.
- To create a bar chart

```
# create bar chart
pyplot.bar(x, y)
```

Bar chart – Example

```
# example of a bar chart
from random import seed
from random import randint
from matplotlib import pyplot
# seed the random number generator
seed(1)
# names for categories
x = ['red', 'green', 'blue']
# quantities for each category
y = [randint(0, 100), randint(0, 100), randint(0, 1
# create bar chart
pyplot.bar(x, y)
# show line plot
pyplot.show()
```

Bar chart – Example



Histogram Plot

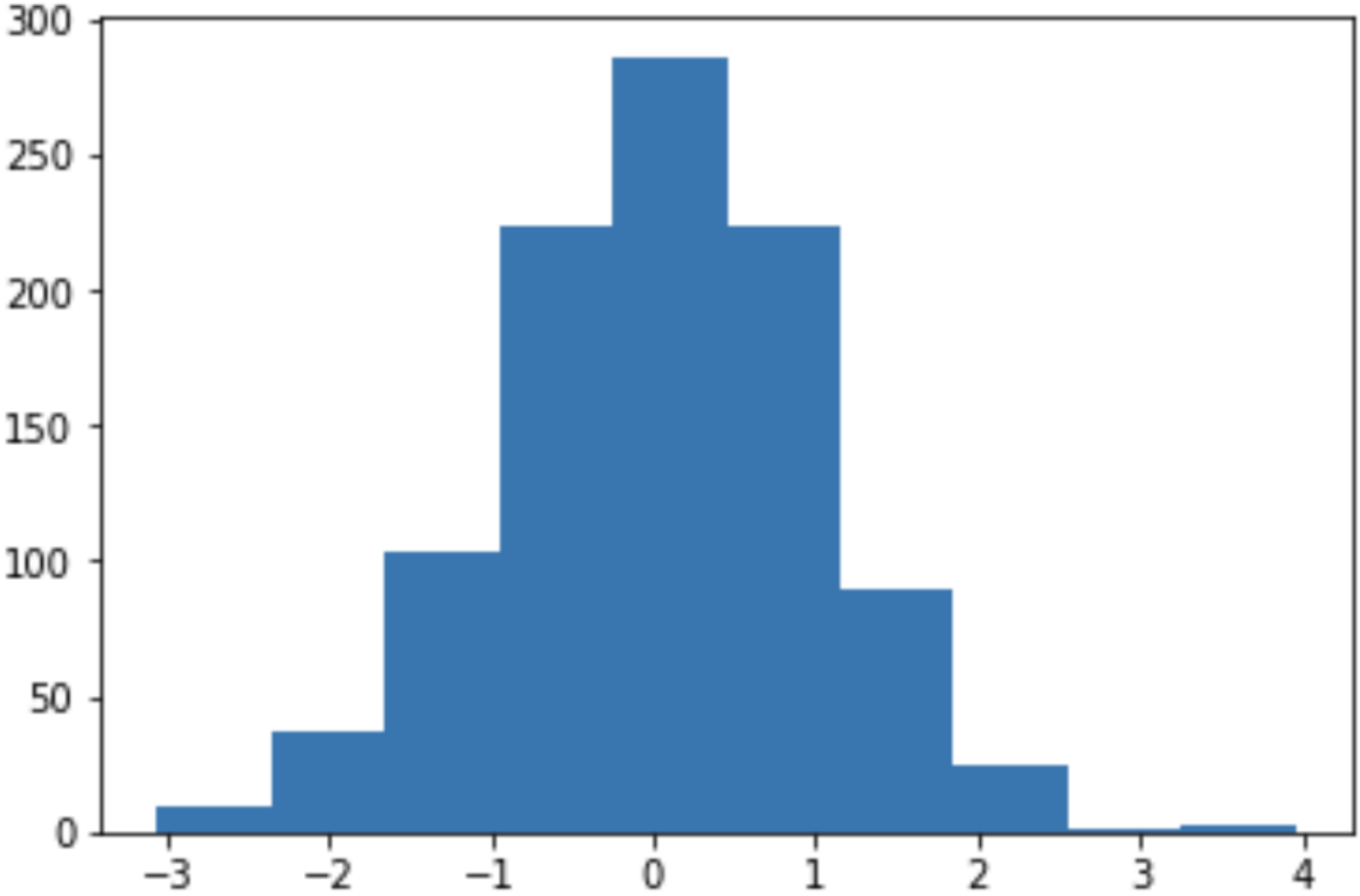
- A histogram plot is generally used to summarize the distribution of a data sample.
- The x-axis represents discrete bins or intervals for the observations.
- For example observations with values between 1 and 10 may be split into five bins, the values [1,2] would be allocated to the first bin, [3,4] would be allocated to the second bin, and so on.
- The y-axis represents the frequency or count of the number of observations in the dataset that belong to each bin.
- To create a Histogram Plot

```
# create histogram plot
pyplot.hist(x)
```

Histogram Plot – Example

```
# example of a histogram plot
from numpy.random import seed
from numpy.random import randn
from matplotlib import pyplot
# seed the random number generator
seed(1)
# random numbers drawn from a Gaussian distribution
x = randn(1000)
# create histogram plot
pyplot.hist(x)
# show line plot
pyplot.show()
```

Histogram Plot – Example



Box and Whisker Plot

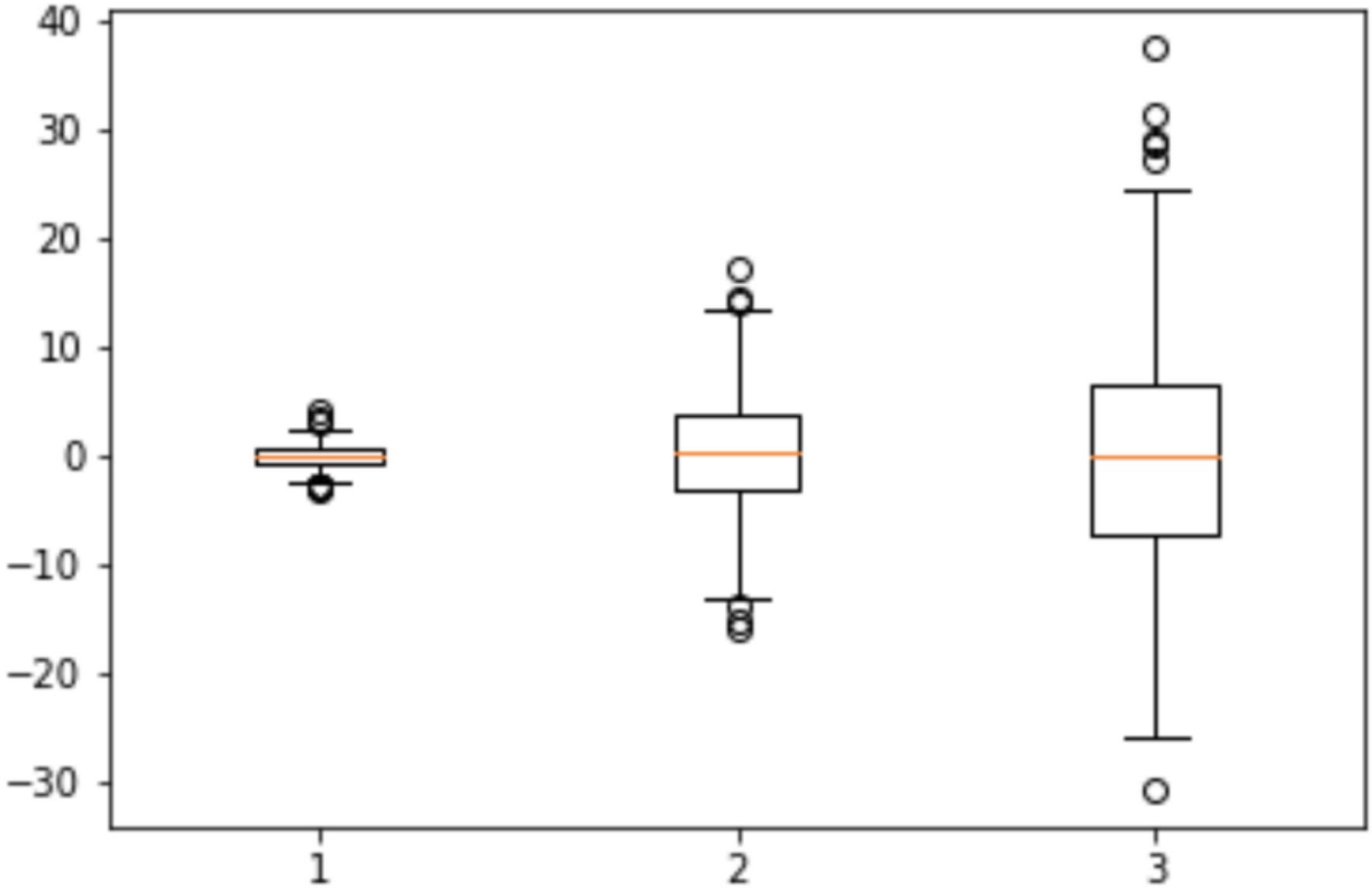
- A box and whisker plot, or boxplot for short, is generally used to summarize the distribution of a data sample.
- The x-axis is used to represent the data sample, where multiple boxplots can be drawn side by side on the x-axis if desired.
- The y-axis represents the observation values.
- A box is drawn to summarize the middle 50
- The median, or 50th percentile, is drawn with a line.
- A value called the interquartile range, or IQR, is calculated as 1.5 * the difference between the 75th and 25th percentiles.
- Lines called whiskers are drawn extending from both ends of the box with the length of the IQR to demonstrate the expected range of sensible values in the distribution.
- Observations outside the whiskers might be outliers and are drawn with small circles.
- To create a Box and Whisker Plot

```
pyplot.boxplot(x)
```

Box and Whisker Plot – Example

```
# example of a box and whisker plot
from numpy.random import seed
from numpy.random import randn
from matplotlib import pyplot
# seed the random number generator
seed(1)
# random numbers drawn from a Gaussian distribution
x = [randn(1000), 5 * randn(1000), 10 * randn(1000)]
# create box and whisker plot
pyplot.boxplot(x)
# show line plot
pyplot.show()
```

Box and Whisker Plot – Example



Scatter Plot

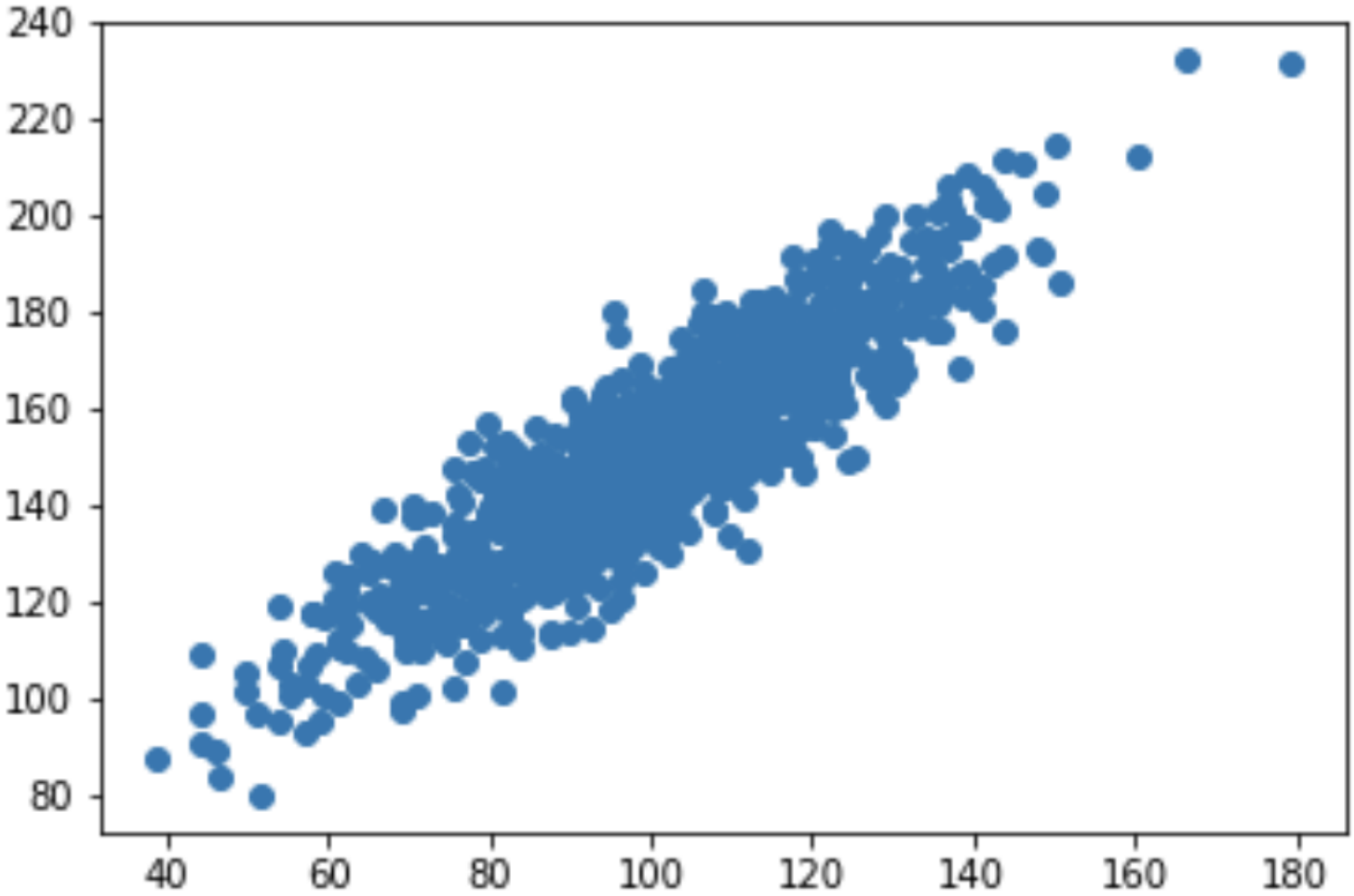
- A scatter plot (or 'scatterplot') is generally used to summarize the relationship between two paired data samples.
- Paired data samples means that two measures were recorded for a given observation, such as the weight and height of a person.
- The x-axis represents observation values for the first sample, and the y-axis represents the observation values for the second sample. Each point on the plot represents a single observation.
- To create a Scatter Plot

```
# create scatter plot
pyplot.scatter(x, y)
```

Scatter Plot – Example

```
# example of a scatter plot
from numpy.random import seed
from numpy.random import randn
from matplotlib import pyplot
# seed the random number generator
seed(1)
# first variable
x = 20 * randn(1000) + 100
# second variable
y = x + (10 * randn(1000) + 50)
# create scatter plot
pyplot.scatter(x, y)
# show line plot
pyplot.show()
```

Scatter Plot – Example



Contents

Dimensionality Reduction

- When we work with a dataset with a lot of features, it is difficult to understand or explore the relationships between the features.
- Without a thorough understanding of the data, we might overfit our model or overlook violations of the assumptions of the algorithm, like the independence of features in linear regression.
- This is where dimensionality reduction comes in.
- In machine learning, dimensionality reduction is the process of reducing the number of random variables under consideration by obtaining a set of principal variables.
- By reducing the dimension of your feature space, we have fewer relationships between features to consider which can be explored and visualized easily and also you are less likely to overfit your model.

Dimensionality Reduction

Dimensionality reduction can be achieved in the following ways:

- Feature Elimination: You reduce the feature space by eliminating features. This has a disadvantage though, as you gain no information from those features that you have dropped.
- Feature Selection: You apply some statistical tests in order to rank them according to their importance and then select a subset of features for your work. This again suffers from information loss and is less stable as different test gives different importance score to features. You can check more on this here.
- Feature Extraction: You create new independent features, where each new independent feature is a combination of each of the old independent features. These techniques can further be divided into linear and non-linear dimensionality reduction techniques.

Principal Component Analysis (PCA)

- Principal Component Analysis or PCA is a linear feature extraction technique.
- It performs a linear mapping of the data to a lower-dimensional space in such a way that the variance of the data in the low-dimensional representation is maximized.
- It does so by calculating the eigenvectors from the covariance matrix.
- The eigenvectors that correspond to the largest eigenvalues (the principal components) are used to reconstruct a significant fraction of the variance of the original data.

t-Distributed Stochastic Neighbor Embedding (t-SNE)

- t-Distributed Stochastic Neighbor Embedding (t-SNE) is a non-linear technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets.
- It is extensively applied in image processing, NLP, genomic data and speech processing.
- The algorithms starts by calculating the probability of similarity of points in high-dimensional space and calculating the probability of similarity of points in the corresponding low-dimensional space.
- The similarity of points is calculated as the conditional probability that a point A would choose point B as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian (normal distribution) centered at A.
- It then tries to minimize the difference between these conditional probabilities (or similarities) in higher-dimensional and lower-dimensional space for a perfect representation of data points in lower-dimensional space.

Fashion-MNIST dataset

- The Fashion-MNIST dataset is a 28x28 grayscale image of 70,000 fashion products from 10 categories, with 7,000 images per category.
- The training set has 60,000 images, and the test set has 10,000 images.
- The Fashion-MNIST also consists of 10 labels, which are fashion accessories like sandals, shirt, trousers, etc.

Fashion-MNIST dataset

Each training and test example is assigned to one of the following labels:

1. T-shirt/top
2. Trouser
3. Pullover
4. Dress
5. Coat
6. Sandal
7. Shirt
8. Sneaker
9. Bag
10. Ankle boot

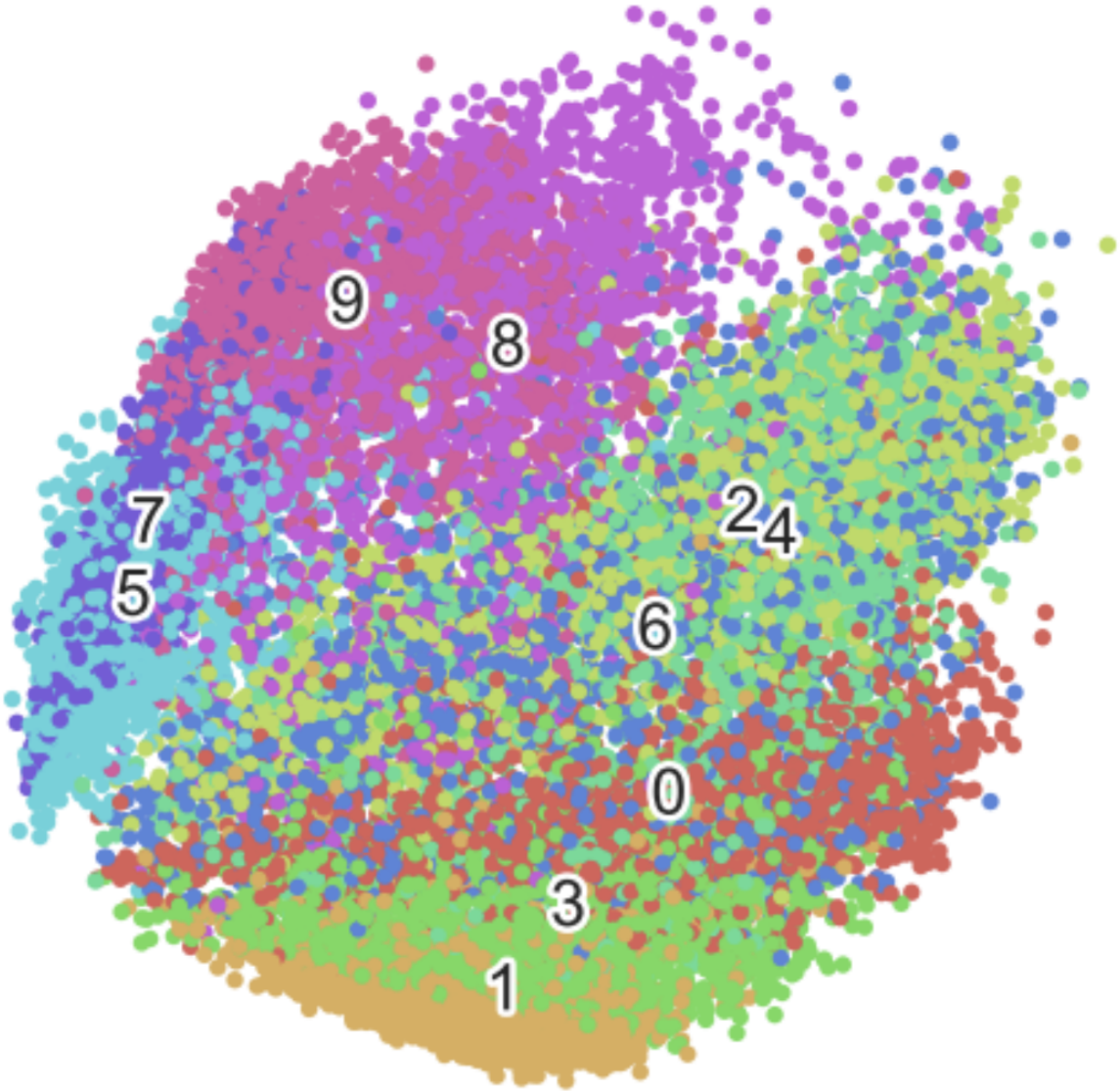
Running PCA on Fashion-MNIST dataset

```
from sklearn.decomposition import PCA
time_start = time.time()

pca = PCA(n_components=4)
pca_result = pca.fit_transform(x_subset)

print 'PCA done! Time elapsed: {} seconds'.\
        format(time.time()-time_start)
```

PCA on Fashion-MNIST dataset – Result

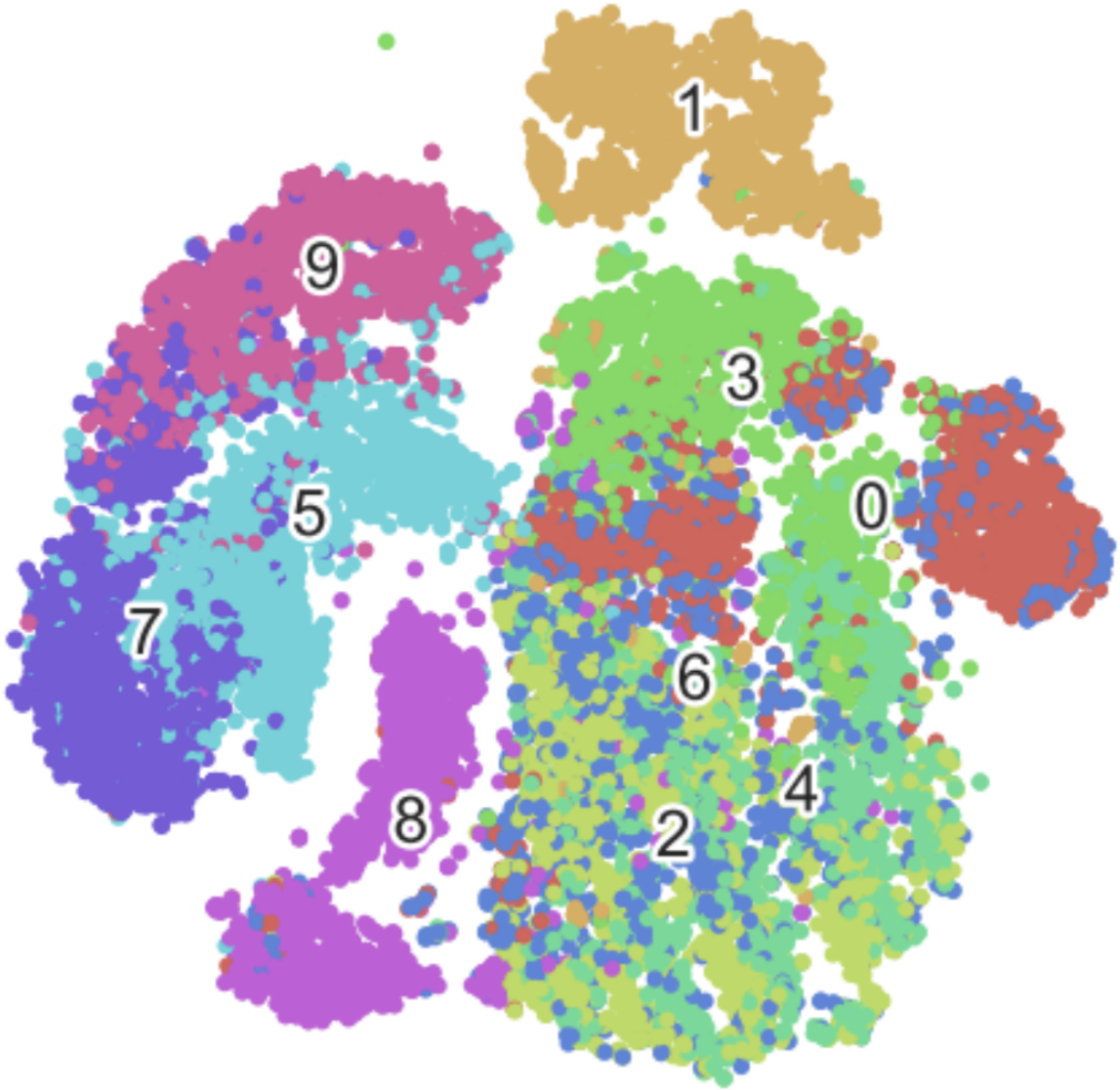


Running t-SNE on Fashion-MNIST dataset

```
from sklearn.manifold import TSNE
import time
time_start = time.time()

fashion_tsne = TSNE(random_state=RS).\
        fit_transform(x_subset)
print 't-SNE done! Time elapsed: {} seconds'.\
        format(time.time()-time_start)
```

t-SNE on Fashion-MNIST dataset – Result



PCA vs. t-SNE

Although both PCA and t-SNE have their own advantages and disadvantages, some key differences between PCA and t-SNE can be noted as follows:

- t-SNE is computationally expensive and can take several hours on million-sample datasets where PCA will finish in seconds or minutes.
- PCA it is a mathematical technique, but t-SNE is a probabilistic one.
- Linear dimensionality reduction algorithms, like PCA, concentrate on placing dissimilar data points far apart in a lower dimension representation.
- But in order to represent high dimension data on low dimension, non-linear manifold, it is essential that similar data points must be represented close together, which is something t-SNE does, not PCA.
- Since PCA is a linear algorithm, it will not be able to interpret the complex polynomial relationship between features while t-SNE is made to capture exactly that.

Contents

References

[1] J. Brownlee, "A Gentle Introduction to Data Visualization Methods in Python", https://machinelearningmastery.com/data-visualization-methods-in-python/
[2] M. Pathak, "Introduction to t-SNE", https://www.datacamp.com/community/tutorials/introduction-t-sne