INF264 - Exercise 8

October 2019

Instructions

- Deadline: 1/11/2019 , 23:59
- Submission place: https://mitt.uib.no/courses/19532/assignments
- Format: Your answers are to be returned in a single pdf report. You can also return scanned pages for your calculations. For results, your answers must include any values and plots that are requested in the Notebook.

1 Principal Component Analysis

The sheer size of data in the modern age is not only a challenge for computer hardware but also a main bottleneck for the performance of many machine learning algorithms. Often, the desired goal is to reduce the dimensions of a d-dimensional dataset by projecting it onto a k-dimensional subspace (where $k < d$) in order to increase the computational efficiency while retaining most of the information.

In PCA, the end goal is to reduce the dimensions of the features in the dataset whislt preserving the information in the features. In order to achieve that, we compute eigenvectors (the principal components) of the covariance matrix of the features in the dataset and collect them in a projection matrix. Each of those eigenvectors is associated with an eigenvalue which can be interpreted as the "length" or "magnitude" of the corresponding eigenvector. If some eigenvalues have a significantly larger magnitude than others, then the reduction of the dataset via PCA onto a smaller dimensional subspace by dropping the *less informative* eigenvectors is reasonable.

Load the "PCA.csv" dataset and answer the following questions:

- Fit sklearn's PCA on the datapoints with 2 components (n_components=2)
- Plot the principal components and explain what are they with respect to the datapoints?

- Fit another PCA on the datapoints but this time with only 1 component and transform the datapoints using this principal component.
- Perform inverse_transformation on the transformed features and plot them with the original features. How do these datapoints relate the first principal component?

Load the Iris dataset and extract features and labels and visualize the features by ploting the histogram of featurs.

- Use StandardScaler from sklearn in order to standardize the dataset features. Plot the histogram once again and explain what has happend to the features?
- The classic approach to PCA is to first centerelize the datapoints by sustracting the mean of each of the dimensions as follows:

  $$\hat{x_i} = x_i - \bar{x} \tag{1}$$

  and then calculate the covariance matrix of features. Afterwards, we perform the eigendecomposition on the covariance matrix S, which is a $d \times d$ matrix where each element represents the covariance between two features. The covariance between two features is calculated as follows:

  $$S_{jk} = \frac{1}{n}\hat{X}^T\hat{X}. \tag{2}$$

  Calculate the Covariance matrix of the features using (2) in comput_cov function.
- Use numpy's linalg.eig to perform eigenvalue decomposition on the covariance matrix and sort the eigen vectors based on their eigen values in Sort_eigens.
- Compute the explained and cumulative explained variance from eigenvalues and plot them. Explain why it's enough to choose the first two principal components and omit the rest based on this plot.
- Complete the PCA procedure by reducing the 4-dimensional feature space of iris dataset to a 2-dimensional feature subspace, by choosing the "top 2" eigenvectors with the highest eigenvalues to construct our 4×2 dimensional eigenvector matrix $\mathbf{W}$.
- Use the 4×2-dimensional projection matrix $\mathbf{W}$ to transform our samples onto the new subspace via the equation:

  $$\mathbf{Z} = \mathbf{X} \times \mathbf{W} \tag{3}$$

  where $\mathbf{Z}$ is a 150×2 matrix of our transformed samples. Plot the transformed datapoints.
- Use sklearn's PCA on iris and "PCA.csv" datasets and compare the results with your own implementation.

2 Autoencoders for Dimensionality Reduction

An autoencoder can be defined as a neural network whose primary purpose is to learn the lower dimensional representaion or the feature space in the dataset. An autoencoder generally consists of two parts an encoder which transforms the input to a hidden representation and a decoder which reconstructs the input from hidden layer.

Fashion-MNIST is a dataset of Zalando's article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. Load the fashion_mnist dataset and complete the following tasks:

- Complete the function *dim_red_ae* for implementation of an autoencoder with one hidden layer of size n_dims_encoded using keras. The layers activation function should be "tanh". Use "adam" optimizer for the loss function of "mse". (5 epochs should be enough)
- train and run the encoder for a hidden layer of two dimensions on the dataset and plot the hidden layer values for the whole dataset.
- Inspect the quality of the representation visually by viewing the reconstructed image for one of the images next to the original image.
- This time train and run the autoencoder for 100 dimensions in the hidden layer and view the reconstructed image for one of the images next to the original image. How did the reconstructed image compare to two dimensional encoding-decoding?
- Split the first 10000 images in the dataset into training and test sets and train a KNN classifier (K=3) on the training set and measure the time it takes for traning and predicting on the test set.
- Train the autoencoder on the training set to reduce the dimensions of the images and perform cross-validation (5 folds) using training set in order to find the best number of dimensions of hidden layer of autoencoder for dimensionality reduction for KNN classifier. (try [2,5,7,10,15,20] number of dimensions)
- Measure the test set accuracy and the time of training and prediction of a KNN classifier using the best number of dimensions you obtain for your autoencoder from cross validation in order to reduce dimensions.