COMP 642 — Machine Learning Mar 3, 2022

Lecture 15: K-Means, Image Representation, CNN

Lecturer: Anshumali Shrivastava
Scribe By: Jiaxin Li, Supeng He, Yi Zhang, Ziwei Li

1 K-means Algorithm

K-means algorithm is one of the most widely used clustering technique. It's an iterative algorithm that categorize data into K distinct non-overlapping subgroups (**Clusters**). The algorithm computes centers of the clusters (**Centroid**) and repeats until the optimal centroid is found.The way to assign data points to assign data points to clusters is to minimize the sum of squared distances between data points and centroids. We try to increase distinction of clusters and decrease variation of intra-cluster data points.

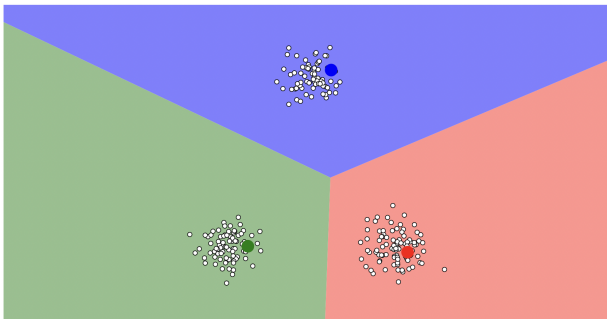

Figure 1: K-means clustering visualization

1.1 Implementation steps

The K-means algorithm works in three steps:

- Step 1: Specify the number of K clusters provided by algorithm
- Step 2: Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement.
- Step 3: Iterating the following steps until there is no change to the centroids, which is the assigning of data points to clusters that do not vary.
  - Firstly compute sum of squared distance between data points and all centroids.
  - Allocate each data point to the cluster that is closest to the others (centroid).
  - Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.

1.2 Issues of K-means

1. Initialization sensitivity: Final formed centroids are significantly affected by the initialization process. A poor initialization can lead to increased numbers of required clustering iterations to reach convergence, a greater overall runtime, and a less-efficient algorithm overall.

    - Example: If two centroids are initialized very close to each other, then a cluster is over split and more than one cluster might end up linked with a single centroid. Some different clusters are classified to the same class. Poor initialization can cause the algorithm to get stuck into an inferior local minimum.
    - Solutions:
        - Employ distinct centroids's initializations
        - Select optimal value for k for specific dataset

2. Very sensitive to outliers due to due to the leverage of the euclidean distance function

3. Difficult to initialize number of clusters K

Figure 2: Example of poor clustering

1.3 Applications of K-means clustering

- Customer segmentation
- Image processing
- Document classification
- Delivery store optimization

2 Machine Learning on Images

2.1 Image Representation

In machine learning, an image is usually represented in an $N * M * 3$ three-dimensional matrix (Figure 3), where N and M are numbers of pixels of length and width, and each of the three layers represents the Red, Green, and Blue (RGB) component of the image plane. In the image RGB matrix, each cell is an integer ranging from 0 to 255 representing the RGB value of the corresponding pixel on the image. This is the most intuitive way to represent an image; however, there are some crucial drawbacks of this representation. For example, the resulting RGB matrices of two images can be very different for two images with only a few pixels difference or two identical but flipped images. In order to solve these problems, a new concept - **Bag of Visual Words** - was introduced.

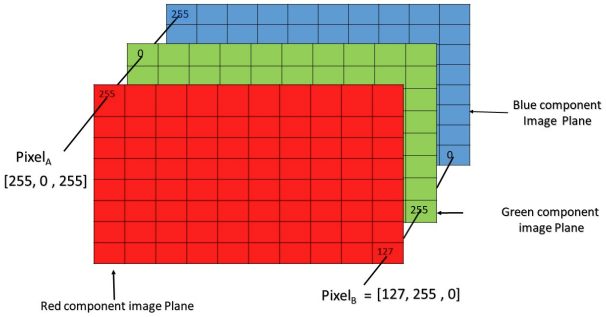

Figure 3: RGB Matrix of Image Representation

2.2 Bag of Visual Words Model

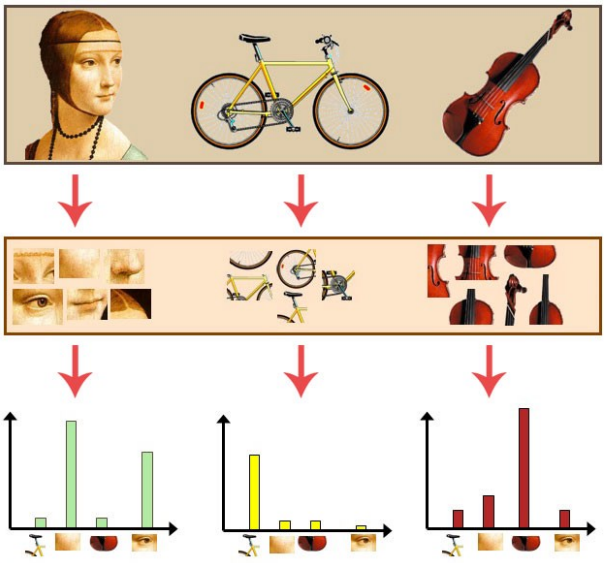

Figure 4: Bag of Visual Words Model

The Bag of Visual Words (**BOVW**) model is an important concept used for the content classification of images. It is similar to the Bag of Words (BOW) model, where a document is represented in a bag of its words and their corresponding frequency. In the BOVW model, an image is represented as a collection of patches, which are smaller pieces of the image, while totally disregarding the order and locality of patches to each other. The initialization of the BOVW model mainly consists of three steps: **feature extraction**, **dictionary construction**, and **vector quantization**. (Figure 4)

2.2.1 Feature Extraction

The first step of building a BOVW model is feature extraction, which involves extracting descriptors, the visual features of the contents, from each input image. Some common ways for detecting features and extracting descriptors of an image include:

- Use gray-scale pixel matrices as Features.
  - Represent the image as a two-dimensional matrix with each cell contains an integer ranging from 0 to 255 representing the grey-scale value of the corresponding pixel of the image.
- Use averaged RGB values of pixels as Features.
  - Represent the image as a two-dimensional matrix with each cell contains an integer ranging from 0 to 255 representing the averaged value of three RBG channels of the corresponding pixel.
- Extract edge features of the images.
  - Extract the shape of the images and represent the shape in a two-dimensional matrix.
- Apply algorithms, such as SIFT and KAZE, to extract salient regions of the input images.

2.2.2 Dictionary Construction

The second step after extracting feature vectors from input images is constructing a dictionary of possible image patches. This step is done by clustering descriptors extracted in the previous step. Common clustering algorithms for dictionary construction include K-Means and DBSCAN. The centroid of each cluster will be the vocabularies of the dictionary.

2.2.3 Vector Quantization

After we already done the feature extraction and dictionary construction, then we can start to build the histogram to quantify those vectors. In this histogram, the x-axis would be the feature vectors we extracted from the first step and the length would be the value we got from the second step by using K-Means and DBSCAN. The figure below is an example that we can get with vector quantization of image:

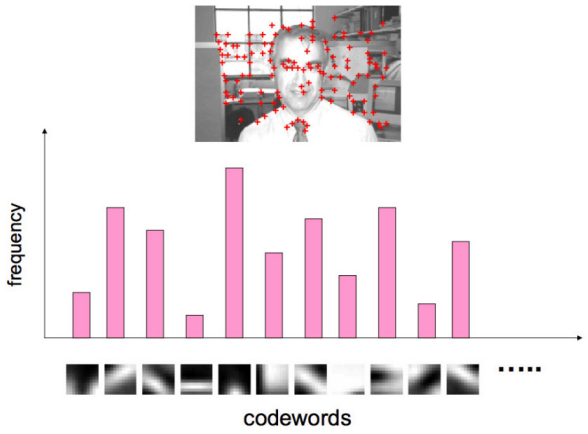

Figure 5: An example of constructing a bag of visual words from an image

After these prepossessing of images, an image can be represented by a single histogram (bag) of discriminative features. Then these data is ready to be fed into other machine learning, computer vision, and CBIR algorithms.

3 Convolutional Neural Network

Convolutional Neural Networks (CNN) are similar to regular Neural Networks, both with learnable weights and biases. Each neuron receives some inputs, performs a dot product, and optionally follows it with a non-linearity (e.g. ReLU). CNN architectures assume that the inputs are images, which allows us to encode certain properties into the architecture. These then preserve the spatial structure, make the forward function more efficient, and reduce the number of parameters in the network. A simple CNN is a sequence of layers. We use three main types of layers to build the architectures: **Convolutional Layer**, **Pooling Layer**, and **Fully-Connected Layer**. We stack these layers to form a full CNN architecture. The following is an example CNN with two convolutional layers, two pooling layers, and a fully connected layer which decides the final classification of the image into one of several categories.

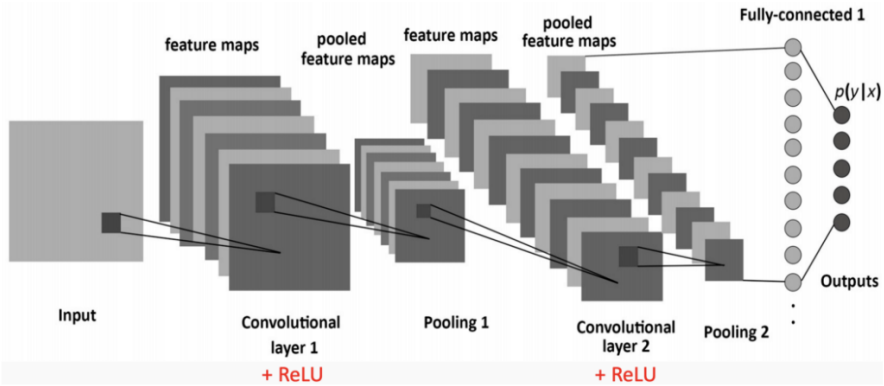

Figure 6: CNN Architecture Example

3.1 Convolutional Layer

Basic action is to apply filters to extract features, i.e. slide over the image spatially and compute dot products (if we consider bias, one bias per filter), generate feature maps, then apply activation function (e.g. ReLU) on every value of feature maps. A filter could be related to anything, for pictures of humans, one filter could be associated with seeing noses, and our nose filter would give us an indication of how strongly a nose seems to appear in our image, and how many times and in what locations they occur. Notice, filters always extend the full depth of the input volume.

- **Stride**: we start with the convolution window at the upper-left corner of the input, and then slide it over all locations both down and to the right. we may default to sliding one element at a time, but we can also move our window more than one element at a time, skipping the intermediate locations.
- **Padding**: one issue when applying convolutional layers is that we tend to lose pixels on the perimeter of our image. Then a straight solution is that we add extra pixels of filler around the boundary of our input image, thus increasing the effective size of the image.
- **Computational Summary**: Let's assume input is $W_1 \times H_1 \times C$, then we need 4 hyperparameters to determine the output size: number of filters K, filter size F, stride S, zero padding P. With the value set, we produce an output of $W_2 \times H_2 \times K$, where $W_2 = \frac{W_1 - F + 2P}{S} + 1$, $H_2 = \frac{H_1 - F + 2P}{S} + 1$

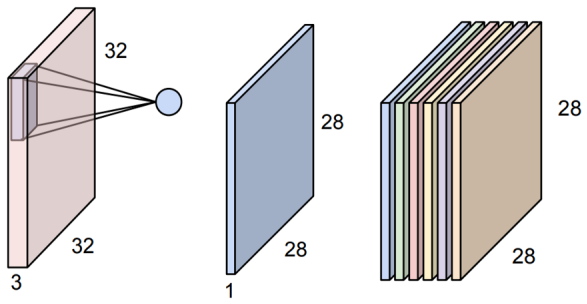

Figure 7: Illustration of Conv Layer with F=5, S=1, P=0, K=6

3.2 Pooling Layer

Pooling layers are similar to convolutional layers, but they perform a specific function such as max pooling, which takes the maximum value in a certain filter region, or average pooling, which takes the average value in a filter region. These are typically used to reduce the dimensionality of the network with negligible loss, speed up the computation, and prevent overfitting.

- **Computational Summary**: Let's assume input is $W_1 \times H_1 \times C$, then we need 2 hyperparameters to determine the output size: spatial extent F, stride S. With the value set, we produce an output of $W_2 \times H_2 \times C$, where $W_2 = \frac{W_1 - F}{S} + 1$, $H_2 = \frac{H_1 - F}{S} + 1$

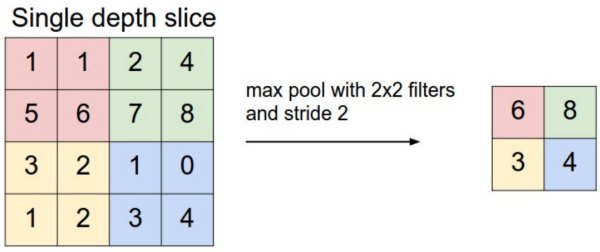

Figure 8: Illustration of Max Pooling with F=2, S=2

3.3 Fully-Connected Layer

The fully-connected layer is placed before the output and is used to flatten the results. This layer contains neurons that connect to the entire input volume, as in regular Neural Networks.

4 Reference

[1] K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks

[2] Understanding K-means Clustering in Machine Learning

[3] Toward Increased K-means Clustering Efficiency with the Naive Sharding Centroid Initialization Method

[4] Bag of Visual Words in a Nutshell

[5] The Bag of (Visual) Words Model

[6] 3 Beginner-Friendly Techniques to Extract Features from Image Data using Python

[7] Convolutional Neural Networks

[8] Simple Introduction to Convolutional Neural Networks

[9] CS231n Convolutional Neural Networks for Visual Recognition