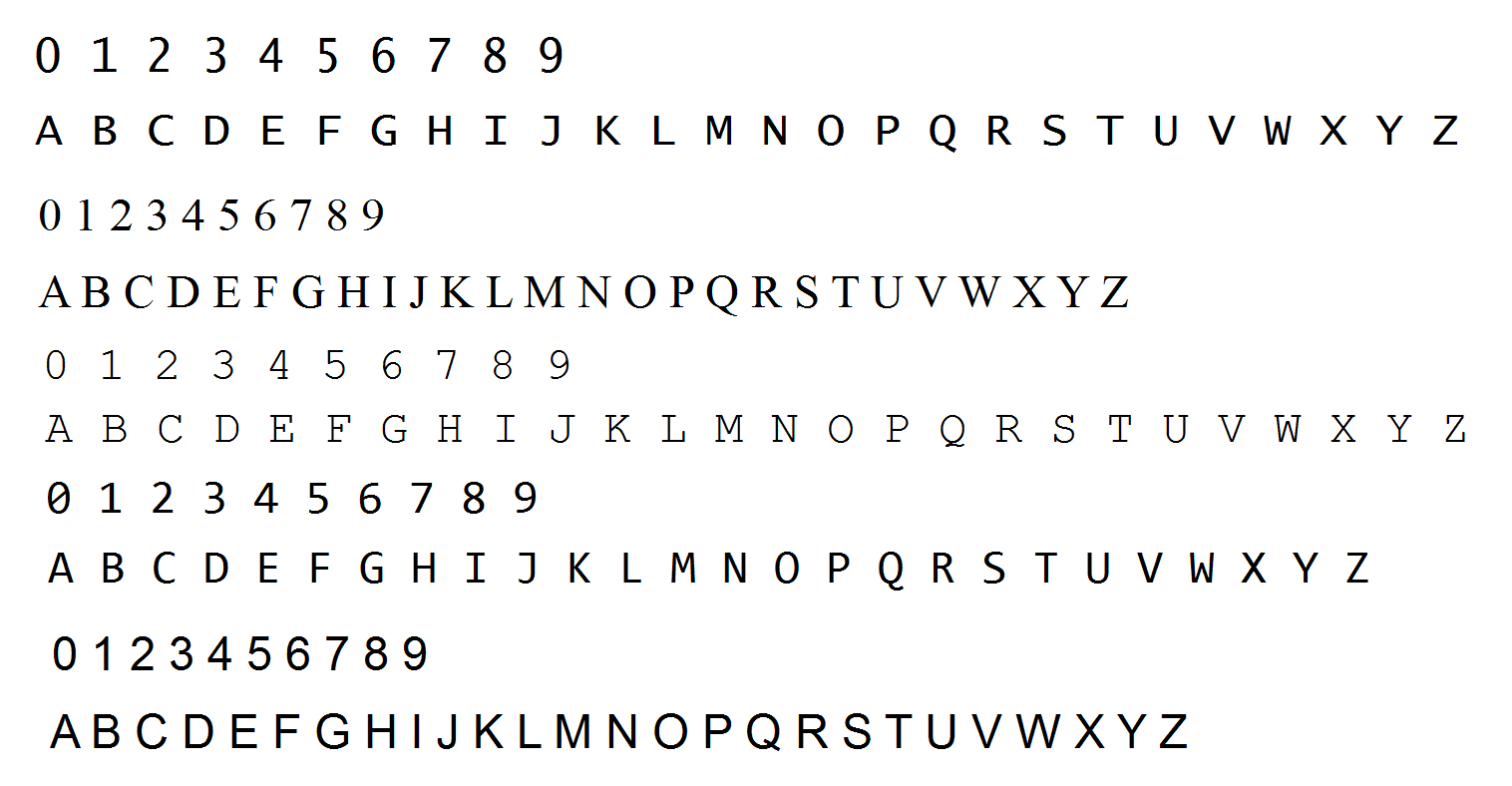
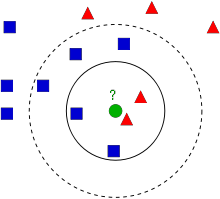
Text recognition

\*Image for training set:



\*KNN is one of the simplest of classification algorithms available for supervised learning. The idea is to search for closest match of the test data in feature space.

In the image, we should find what type is the green dot, is it blue square or red triangle, for classification we will use KNN algorithm which nearest neighbors type and assigns to this point. By default, we classify it with k=3, it means that we are classifying nearest three points.

KNN compares entire image under test to the entire image on the training set. Training set will produce 2 parallel data structures:1) a set of numbers indicating which “group” or classification each corresponding image is in, 2) a set of images.

Here below shown training set:

npaClassifications = npaClassifications.reshape((npaClassifications.size, 1))

kNearest = cv2.ml.KNearest\_create()

kNearest.train(npaFlattenedImages, cv2.ml.ROW\_SAMPLE, npaClassifications)

retval, results, neigh\_resp, dists = kNearest.findNearest(resized, k = 3)

Here, resized is point which we will classify and return value “results”. We will classify it with a help of feature matrix vector.

\*Draw contours of the image

Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition.

import numpy as np

import cv2 as cv

im = [cv.imread](https://docs.opencv.org/3.4.0/d4/da8/group__imgcodecs.html#ga288b8b3da0892bd651fce07b3bbd3a56)('test.jpg')

imgray = [cv.cvtColor](https://docs.opencv.org/3.4.0/d7/d1b/group__imgproc__misc.html#ga397ae87e1288a81d2363b61574eb8cab)(im, cv.COLOR\_BGR2GRAY)

ret, thresh = [cv.threshold](https://docs.opencv.org/3.4.0/d7/d1b/group__imgproc__misc.html#gae8a4a146d1ca78c626a53577199e9c57)(imgray, 127, 255, 0)

im2, contours, hierarchy = [cv.findContours](https://docs.opencv.org/3.4.0/d3/dc0/group__imgproc__shape.html#ga95f5b48d01abc7c2e0732db24689837b)(thresh, cv.RETR\_TREE, cv.CHAIN\_APPROX\_SIMPLE)

[cv.drawContours](https://docs.opencv.org/3.4.0/d6/d6e/group__imgproc__draw.html#ga746c0625f1781f1ffc9056259103edbc)(img, contours, -1, (0,255,0), 3)

There are two types of bounding rectangles to the contours, but we will use straight bounding rectangle.

x,y,w,h = [cv.boundingRect](https://docs.opencv.org/3.4.0/d3/dc0/group__imgproc__shape.html#gacb413ddce8e48ff3ca61ed7cf626a366)(cnt)

[cv.rectangle](https://docs.opencv.org/3.4.0/d6/d6e/group__imgproc__draw.html#ga346ac30b5c74e9b5137576c9ee9e0e8c)(img,(x,y),(x+w,y+h),(0,255,0),2)

Here, (x,y) be the top-left coordinate of the rectangle and (w,h) be its width and height.

\*Image processing

We know that images have variety of colors, and so it is hard to us detect text from image which is colorful, so we should make it gray scale or the green scale.

For color conversion, we use the function cv.cvtColor(input\_image, flag) where flag determines the type of conversion.

For the gray scale conversion we use the flags [cv.COLOR\_BGR2GRAY](https://docs.opencv.org/3.4.0/d7/d1b/group__imgproc__misc.html#gga4e0972be5de079fed4e3a10e24ef5ef0a353a4b8db9040165db4dacb5bcefb6ea).

Scaling is just resizing of the image. OpenCV comes with a function [cv.resize()](https://docs.opencv.org/3.4.0/da/d54/group__imgproc__transform.html" \l "ga47a974309e9102f5f08231edc7e7529d" \o "Resizes an image. ) for this purpose. The size of the image can be specified manually, or you can specify the scaling factor. Different interpolation methods are used. Preferable interpolation methods are [cv.INTER\_AREA](https://docs.opencv.org/3.4.0/da/d54/group__imgproc__transform.html" \l "gga5bb5a1fea74ea38e1a5445ca803ff121acf959dca2480cc694ca016b81b442ceb) for shrinking and [cv.INTER\_CUBIC](https://docs.opencv.org/3.4.0/da/d54/group__imgproc__transform.html" \l "gga5bb5a1fea74ea38e1a5445ca803ff121a55e404e7fa9684af79fe9827f36a5dc1)(slow) & [cv.INTER\_LINEAR](https://docs.opencv.org/3.4.0/da/d54/group__imgproc__transform.html" \l "gga5bb5a1fea74ea38e1a5445ca803ff121ac97d8e4880d8b5d509e96825c7522deb) for zooming. By default, interpolation method used is [cv.INTER\_LINEAR](https://docs.opencv.org/3.4.0/da/d54/group__imgproc__transform.html" \l "gga5bb5a1fea74ea38e1a5445ca803ff121ac97d8e4880d8b5d509e96825c7522deb) for all resizing purposes. You can resize an input image either of following methods:

import numpy as np

import cv2 as cv

img = [cv.imread](https://docs.opencv.org/3.4.0/d4/da8/group__imgcodecs.html#ga288b8b3da0892bd651fce07b3bbd3a56)('words.jpg')

res = [cv.resize](https://docs.opencv.org/3.4.0/da/d54/group__imgproc__transform.html#ga47a974309e9102f5f08231edc7e7529d)(img,None,fx=2, fy=2, interpolation = cv.INTER\_CUBIC)

height, width = img.shape[:2]

res = [cv.resize](https://docs.opencv.org/3.4.0/da/d54/group__imgproc__transform.html#ga47a974309e9102f5f08231edc7e7529d)(img,(2\*width, 2\*height), interpolation = cv.INTER\_CUBIC)

By the KNN classifier we can find letters and digits in an image.