Task1_Nirnai

May 18, 2017

1 Homework Assignment 1

1.1 Task 1

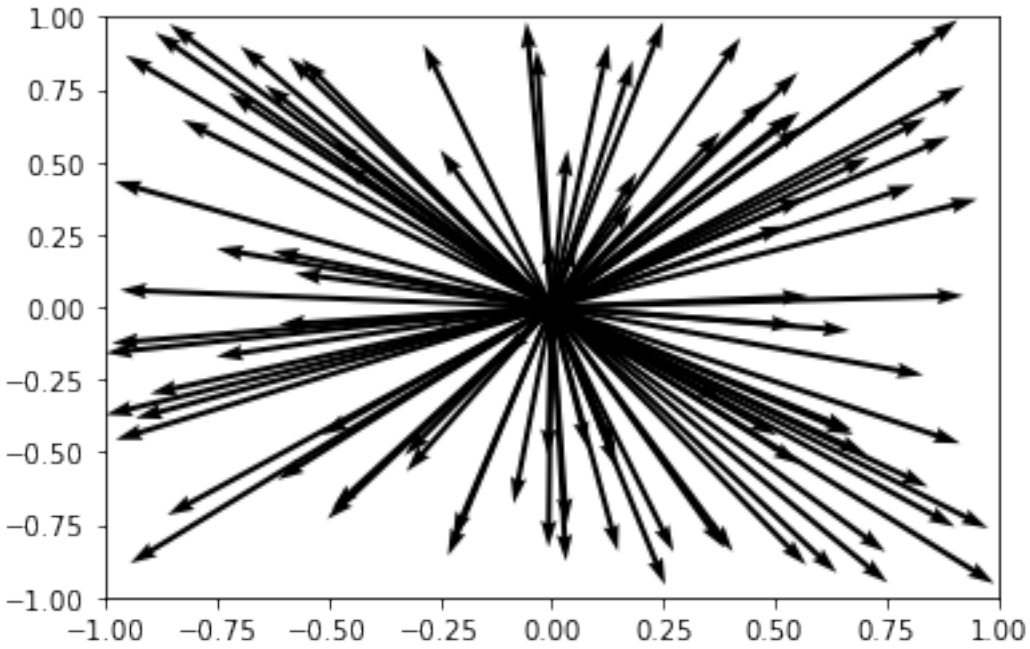
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
```

- **Sample 100 uniformly distributed random vectors from the box $[-1, 1]^d$ for d = 2.**

In [2]:
```
# Matrix of 100 random vectors in 2D
random_vectors = np.random.uniform(-1,1,[2,100])

# Plotting all 100 vectors
X = np.zeros(100)
Y = np.zeros(100)
U = random_vectors[0]
V = random_vectors[1]
plt.figure()
ax = plt.gca()
ax.quiver(X, Y, U, V, angles='xy', scale_units='xy', scale=1)
ax.set_xlim([-1, 1])
ax.set_ylim([-1, 1])
plt.draw()
plt.show()
```



• For each of the 100 vectors determine the minimum angle to all other vectors. Then compute the average of these minimum angles. Note that for two vectors x, y the cosine of the angle between the two vectors is defined as

$$cos(\angle(x,y)) = \frac{< x, y >}{|| \, x \, || \cdot || \, y \, ||}$$

```
In [3]: def angle(v1,v2):

            r'''''''''''
            Calculates the angel between two n-dimensional vectors
            ''''''''''''

            # generate column vector to be able to use norm function
            v1 = v1.reshape(-1,1)
            v2 = v2.reshape(-1,1)

            # calculating the norm of the vectors
            norm_v1 = np.linalg.norm(v1)
            norm_v2 = np.linalg.norm(v2)

            # Formula to calculate angle in rad between to vectors with the norms a
            angle = np.arccos((np.sum(v1*v2))/(norm_v1*norm_v2))
```

```
        return angle * 360/(2*np.pi)



    def min_angles(random_vectors, sample_size):

        r'''''''''''
        Calculates the angle to the closest neighbouring vector for each vector
        and averages over the minimum angles.
        '''''''''''

        # initialize min_angle with the highest possible angle (180°)
        min_angle = 180
        # save all minimum angles in array
        min_angles = np.array([])

        # run through all vectors in the matrix and compute the angle between t
        # If the angle is smaller then min_angles, replace min_angles with this
        for i in np.arange(sample_size):
            v1 = random_vectors[:,i]
            for j in np.arange(sample_size):
                v2 = random_vectors[:,j]
                if(i<j):
                    temp = angle(v1,v2)
                    if(temp < min_angle):
                        min_angle = temp
            # Add the smallest angle to array
            min_angles = np.append(min_angles, min_angle)
            min_angle = 180
        return min_angles

    np.mean(min_angles(random_vectors, 100))
```

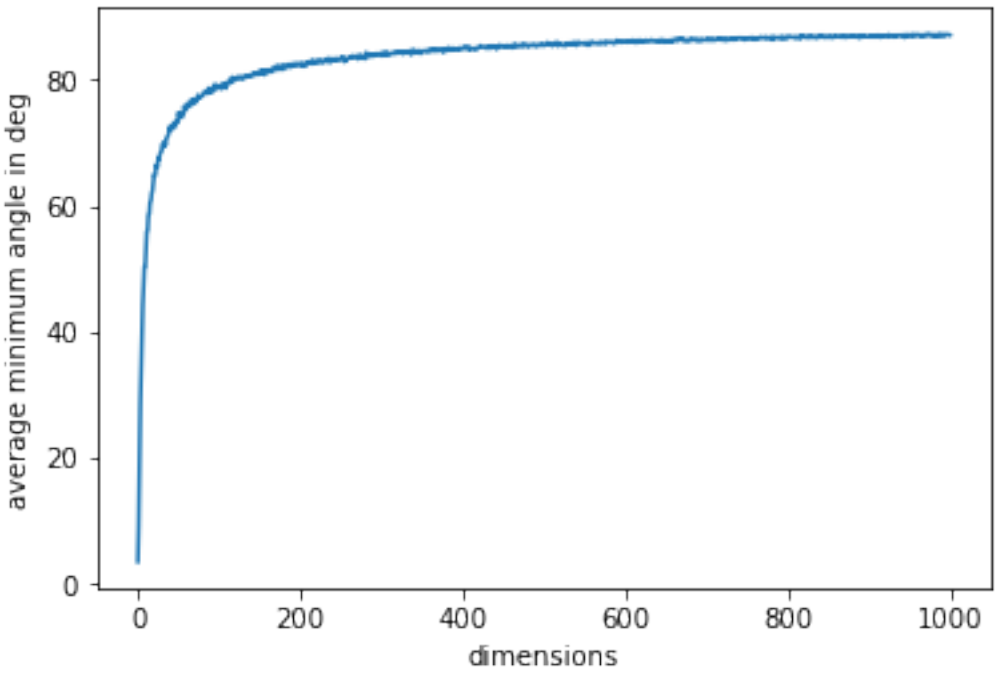
Out[3]: 8.9584494046135514

• Repeat the above for dimensions d = 1, . . . , 1000 and use the results to plot the average minimum angle against the dimension.

```
In [4]: average_min_angles = np.array([])
        # From 1 to 1000 Dimensions
        for i in np.arange(1000)+1:
            dim = i
            random_vectors = np.random.uniform(-1,1,[dim,100])
            average_min_angles = np.append(average_min_angles, np.mean(min_angles(r
```

```
In [7]: fig = plt.figure()
        plt.plot(average_min_angles)
```

```
plt.xlabel('dimensions')
plt.ylabel('average minimum angle in deg')
plt.show()
```



• Give an interpretation of the result. What conclusions can you draw for 2 randomly sampled vectors in a d-dimensional space?

Answer: In average the vectors space out more with growing dimensions, but suddenly converge towards 90 degrees after already a few dimensions added. This shows one aspect of the curse of dimensionality, because in higher dimensions all neighbouring vectors are 90 degrees away. E.g. this would make k-neirest algorithm impossible.

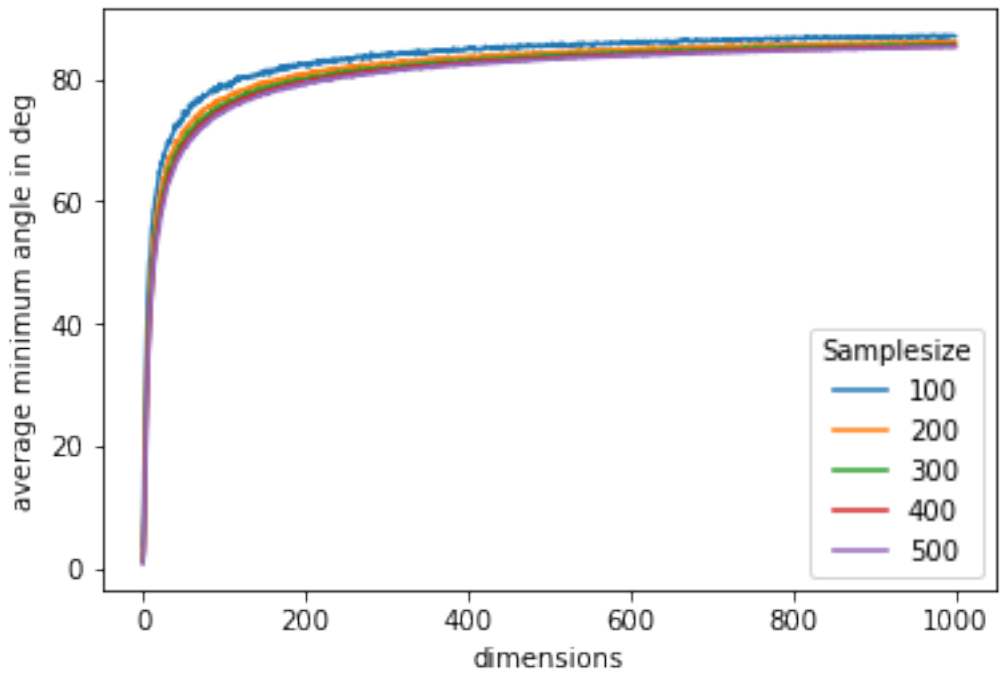
• Does the result change if the sample size increases?

```
In [11]: average_min_angles_200 = np.array([])
         average_min_angles_300 = np.array([])
         average_min_angles_400 = np.array([])
         average_min_angles_500 = np.array([])
         for i in np.arange(1000)+1:
             dim = i
             random_vectors_200 = np.random.uniform(-1,1,[dim,200])
             random_vectors_300 = np.random.uniform(-1,1,[dim,300])
             random_vectors_400 = np.random.uniform(-1,1,[dim,400])
             random_vectors_500 = np.random.uniform(-1,1,[dim,500])
```

```
            average_min_angles_200 = np.append(average_min_angles_200, np.mean(mir
            average_min_angles_300 = np.append(average_min_angles_300, np.mean(mir
            average_min_angles_400 = np.append(average_min_angles_400, np.mean(mir
            average_min_angles_500 = np.append(average_min_angles_500, np.mean(mir
            print i,
```

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
```

```
In [12]: plt.plot(average_min_angles, label='100')
         plt.plot(average_min_angles_200, label='200')
         plt.plot(average_min_angles_300, label='300')
         plt.plot(average_min_angles_400, label='400')
         plt.plot(average_min_angles_500, label='500')
         plt.legend(title='Samplesize')
         plt.xlabel('dimensions')
         plt.ylabel('average minimum angle in deg')
         plt.show()
```



Answer: The result with 200 samples is almost identical to the on with 100 samples. The Flattens out a little bit, but in no relation to the increase in the sample size