Hide menu

Lecture: Machine Translation

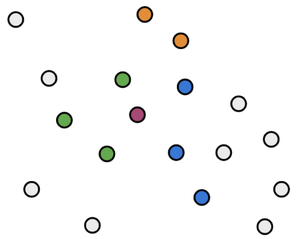
- **Video:** Week Introduction 46 sec
- **Video:** Overview 1 min
- **Video:** Transforming word vectors 7 min
- **Reading:** Transforming word vectors 10 min
- **Lab:** Rotation matrices in R2 1h
- **Video:** K-nearest neighbors 3 min
- **Reading:** K-nearest neighbors 10 min
- **Video:** Hash tables and hash functions 3 min
- **Reading:** Hash tables and hash functions 10 min
- **Video:** Locality sensitive hashing 5 min
- **Reading:** Locality sensitive hashing 10 min
- **Video:** Multiple Planes 3 min
- **Reading:** Multiple Planes 10 min
- **Lab:** Hash tables 1h
- **Video:** Approximate nearest neighbors

Week 4 > Approximate nearest neighbors

Previous Next

Approximate nearest neighbors

Approximate nearest neighbors does not give you the full nearest neighbors but gives you an approximation of the nearest neighbors. It usually trades off accuracy for efficiency. Look at the following plot:



Approximate nearest (friendly) neighbors

You are trying to find the nearest neighbor for the red vector (point). The first time, the plane gave you green points. You then ran it a second time, but this time you got the blue points. The third time you got the orange points to be the neighbors. So you can see as you do it more times, you are likely to get all the neighbors. Here is the code for one set of random planes. Make sure you understand what is going on.

```
num_dimensions = 2 #300 in assignment
num_planes = 3 #10 in assignment

random_planes_matrix = np.random.normal(
                       size=(num_planes,
                             num_dimensions))
```

```
array([[ 1.76405235  0.40015721]
       [ 0.97873798  2.2408932 ]
       [ 1.86755799 -0.97727788]])
```

```
v = np.array([[2,2]])
```

```
def side_of_plane_matrix(P,v):
    dotproduct = np.dot(P,v.T)
    sign_of_dot_product = np.sign(dotproduct)
    return sign_of_dot_product

num_planes_matrix = side_of_plane_matrix(
                    random_planes_matrix,v)
```

```
array([[1.]
       [1.]
       [1.])
```

See notebook for calculating the hash value!