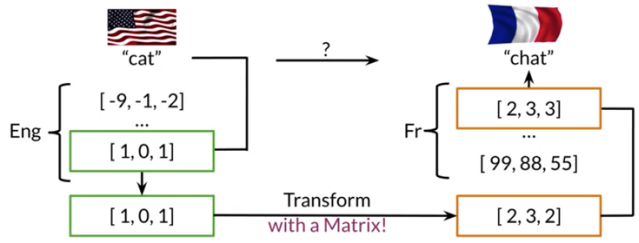
Machine Translation and Document Search

Transforming word vectors

Overview of Translation



Transforming vectors

```
R = np.array([[2,0],
              [0,-2]])

x = np.array([[1,1]])

np.dot(x,R)
```

```
array([[2,-2]])
```

Train only on a subset of English – French vocabulary

Align word vectors

$$\mathbf{XR} \approx \mathbf{Y}$$

["cat" vector]
[... vector]
["zebra" vector]
$\mathbf{X}$

["chat" vecteur]
[... vecteur]
["zébresse" vecteur]
$\mathbf{Y}$

subsets of the full vocabulary

Solving for R

initialize R

in a loop:

$$Loss = \| \mathbf{XR} - \mathbf{Y} \|_F$$

$$g = \frac{d}{dR} Loss$$ gradient

$$R = R - \alpha g$$ update

Frobenius norm

$$\| \mathbf{XR} - \mathbf{Y} \|_F$$

$$\mathbf{A} = \begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix}$$

$$\|\mathbf{A}_F\| = \sqrt{2^2 + 2^2 + 2^2 + 2^2}$$

$$\|\mathbf{A}_F\| = 4$$

$$\|\mathbf{A}\|_F \equiv \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}$$

Frobenius norm

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

A_squared = np.square(A)
A_squared
```

```
array([[4,4],
       [4,4]])
```

```
A_Frobenius = np.sqrt(np.sum(A_squared))
A_Frobenius
```

```
4.0
```

Frobenius norm squared

$$\|\mathbf{XR} - \mathbf{Y}\|_F^2$$

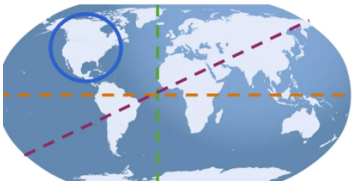
Gradient

$$Loss = \|\mathbf{XR} - \mathbf{Y}\|_F^2$$

$$g = \frac{d}{dR} Loss = \frac{2}{m} \left(\mathbf{X}^T (\mathbf{XR} - \mathbf{Y})\right)$$

K-nearest Neighbors

Nearest neighbors



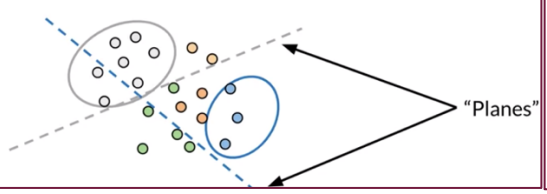
Hash Tabes and Hash Functions

Create a basic hash table

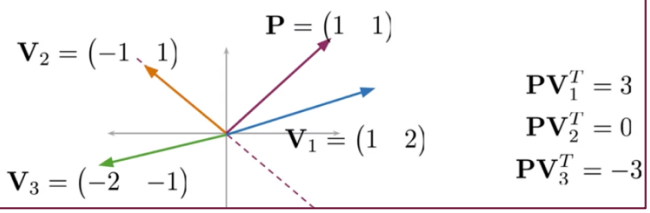
```
def basic_hash_table(value_l,n_buckets):
    def hash_function(value_l,n_buckets):
        return int(value) % n_buckets
    hash_table = {i:[] for i in range(n_buckets)}
    for value in value_l:
        hash_value = hash_function(value,n_buckets)
        hash_table[hash_value].append(value)
    return hash_table
```

Locality Sensitive Hashing

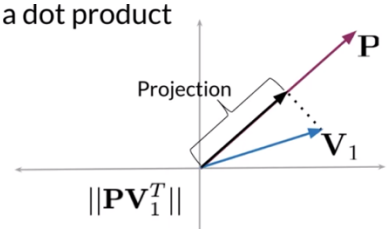
Locality Sensitive Hashing



Which side of the plane?



Visualizing a dot product

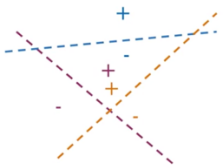


Which side of the plane?

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

Multiple planes

Multiple planes, single hash value?



$\mathbf{P}_1\mathbf{v}^T = 3, sign_1 = +1, h_1 = 1$

$\mathbf{P}_2\mathbf{v}^T = 5, sign_2 = +1, h_2 = 1$

$\mathbf{P}_3\mathbf{v}^T = -2, sign_3 = -1, h_3 = 0$

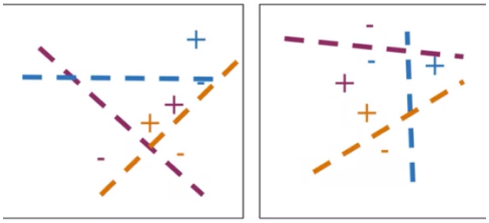
$$hash = 2^0 \times h_1 + 2^1 \times h_2 + 2^2 \times h_3$$
$$= 1 \times 1 + 2 \times 1 + 4 \times 0$$
$$= 3$$

Multiple planes, single hash value!!

```
def hash_multiple_plane(P_l,v):

    hash_value = 0

    for i, P in enumerate(P_l):
        sign = side_of_plane(P,v)
        hash_i = 1 if sign >=0 else 0
        hash_value += 2**i * hash_i

    return hash_value
```

Approximate nearest neighbors

Random planes



Make one set of random planes

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

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

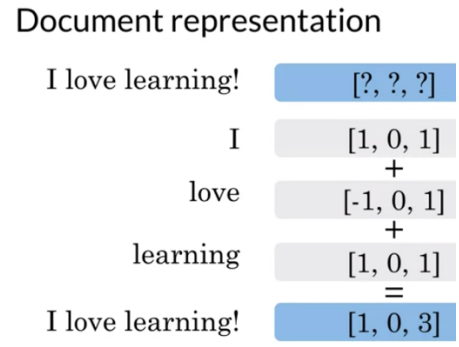
```
array([[ 1.76405235  0.40015721]
       [ 0.97873798  2.2408932 ]
       [ 1.86755799 -0.97727788]])
```

```
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.])
```

Searching documents




```
word_embedding = {"I": np.array([1,0,1]),
                  "love": np.array([-1,0,1]),
                  "learning": np.array([1,0,1])}
words_in_document = ['I', 'love', 'learning']
document_embedding = np.array([0,0,0])
for word in words_in_document:
    document_embedding += word_embedding.get(word,0)
print(document_embedding)
array([1 0 3])
```

The data

The full dataset for English embeddings is about 3.64 gigabytes, and the French embeddings are about 629 megabytes. To prevent the Coursera workspace from crashing, we've extracted a subset of the embeddings for the words that you'll use in this assignment.

If you want to run this on your local computer and use the full dataset, you can download the

- English embeddings from Google code archive word2vec look for GoogleNews-vectors-negative300.bin.gz
  - You'll need to unzip the file first.
- and the French embeddings from cross_lingual_text_classification.
  - in the terminal, type (in one line) `curl -o ./wiki.multi.fr.vec https://dl.fbaipublicfiles.com/arrival/vectors/wiki.multi.fr.vec`

Then copy-paste the code below and run it.

```
# Use this code to download and process the full dataset on your local comput
er

from gensim.models import KeyedVectors



en_embeddings = KeyedVectors.load_word2vec_format('./GoogleNews-vectors-negat
ive300.bin', binary = True)
```

```
fr_embeddings = KeyedVectors.load_word2vec_format('./wiki.multi.fr.vec')



# loading the english to french dictionaries

en_fr_train = get_dict('en-fr.train.txt')

print('The length of the english to french training dictionary is', len(en_fr
_train))

en_fr_test = get_dict('en-fr.test.txt')

print('The length of the english to french test dictionary is', len(en_fr_tra
in))



english_set = set(en_embeddings.vocab)

french_set = set(fr_embeddings.vocab)

en_embeddings_subset = {}

fr_embeddings_subset = {}

french_words = set(en_fr_train.values())



for en_word in en_fr_train.keys():

    fr_word = en_fr_train[en_word]

    if fr_word in french_set and en_word in english_set:

        en_embeddings_subset[en_word] = en_embeddings[en_word]

        fr_embeddings_subset[fr_word] = fr_embeddings[fr_word]




for en_word in en_fr_test.keys():

    fr_word = en_fr_test[en_word]

    if fr_word in french_set and en_word in english_set:

        en_embeddings_subset[en_word] = en_embeddings[en_word]

        fr_embeddings_subset[fr_word] = fr_embeddings[fr_word]
```

```
pickle.dump( en_embeddings_subset, open( "en_embeddings.p", "wb" ) )

pickle.dump( fr_embeddings_subset, open( "fr_embeddings.p", "wb" ) )
```