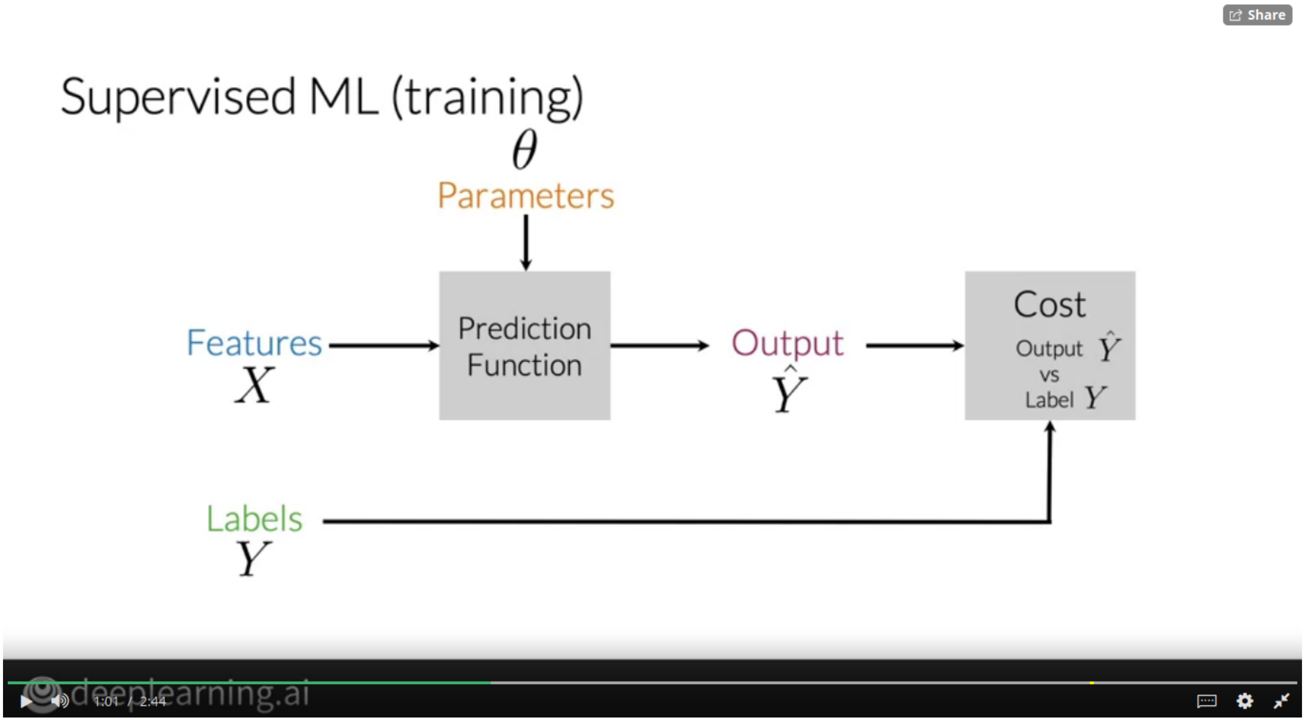
Natural Language Processing with Classification and Vector Spaces

Week1:

Supervised ML & Sentiment Analysis



1. In supervised learning, we have input X and output Y. We try to fit a function f(X) = Y, such that predicted value of function f is close to Y. We change our parameters at each iteration to minimize cost.

Vocabulary & Feature Extraction

Let's consider a Tweet

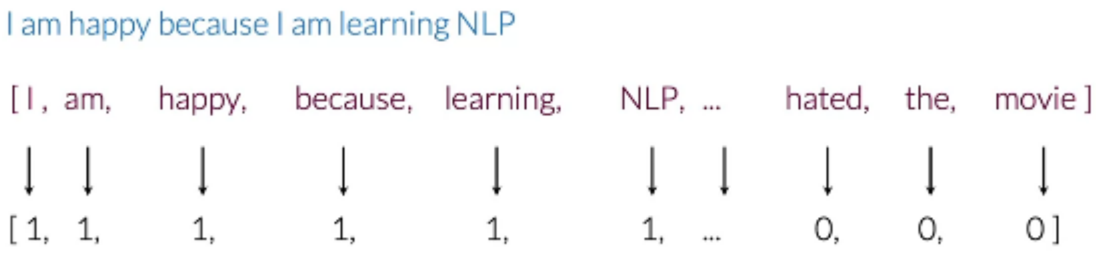
Tweet1: I am happy because I am learning NLP.

Tweet2: I Hated the movie

To represent these tweets in vector form we need to follow the following steps:

1. List all the unique words from all the available tweets.

2. Assign value =1 if that word appears in dictionary else 0.



A lot of Zeros! That's a sparse Representation

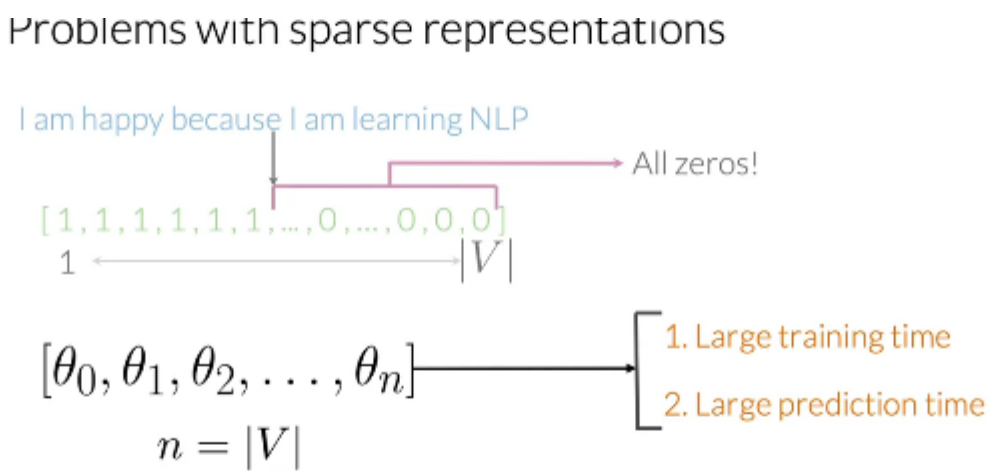
Problem With Sparse Representation

1. Most of the values are zeros if my tweet is small.

2. Logistic Regression will require a V number of parameters to train for each word in the vocabulary.

3. It will take more training time since vector size is very big

4. Prediction will also be slower.



Negative and Positive Frequencies

Corpus: a collection of written texts

Consider having a corpus of tweets as given below:

Corpus

I am happy because I am learning NLP

I am happy

I am sad, I am not learning NLP

I am sad

To count the number of positive and negative frequencies, we will make a table as given below:

Positive and negative counts

Positive tweets

I am happy because I am learning NLP

I am happy

Vocabulary	PosFreq (1)
I	3
am	3
happy	2
because	1
learning	1
NLP	1
sad	0
not	0

Similarily for negative class, we can count the frequencies.

Word frequency in classes

Vocabulary	PosFreq (1)	NegFreq (0)
I	3	3
am	3	3
happy	2	0
because	1	0
learning	1	1
NLP	1	1
sad	0	1
not	0	1

freqs: dictionary mapping from (word, class) to frequency

Feature Extraction with Frequencies

$$\underbrace{X_m}_{\text{Features of tweet m}} = [\underbrace{1}_{\text{Bias}}, \underbrace{\sum_w freqs(w,1)}_{\text{Sum Pos. Frequencies}}, \underbrace{\sum_w freqs(w,0)}_{\text{Sum Neg. Frequencies}}]$$

Feature extraction

Vocabulary	PosFreq (1)
I	3
am	3
happy	2
because	1
learning	1
NLP	1
sad	0
not	0

I am sad, I am not learning NLP

$$X_m = [1, \underbrace{\sum_w freqs(w,1)}_{\boxed{8}}, \sum_w freqs(w,0)]$$

$$X_m = [1, \sum_w freqs(w,1), \sum_w freqs(w,0)]$$

$$\downarrow$$

$$X_m = [1, 8, 11]$$

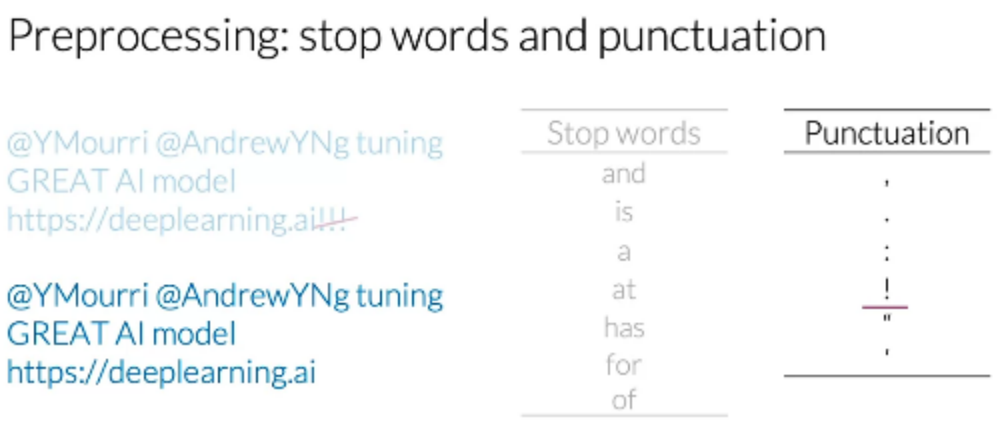
Preprocessing

Preprocess Tweet:

@Ymourri @AndrewNg are tuning a GREAT ai Model at https://deeplearning.ai

1. We need to remove stop words and punctuation mark which does not contribute any meaning in the task of sentiment analysis

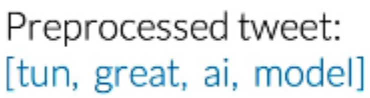
After removing StopWords and punctuation from the tweet.



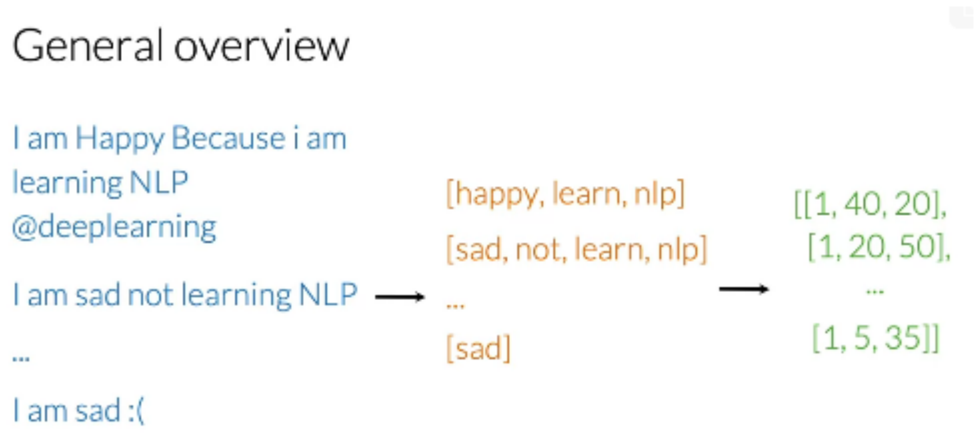
tweets having handles and URLs also does not contribute anything to Sentiment Analysis. We will remove them too.

2. We need to perform Stemming(Transforming any word to its base term). SO the word **tune, tuned or tuning** have the same base word, so after stemming it will become **tun.**

3. lower case all the words. GREAT, Great, great will reduced to great. since it does not change the sentiment of the sentence.



Putting It All Together(Stemming, tokenizing, Removing Stop Words, Punctuation etc)



For each tweet, we will use the sum of +ve and -ve frequency to represent it in vector form.

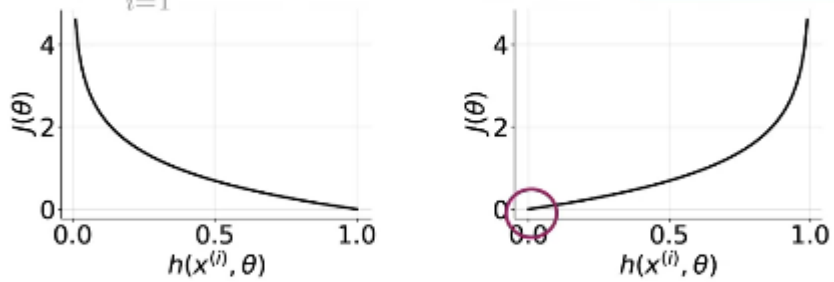
At the end, you will have X matrix with m rows and 3 columns. as shown below.

$$\begin{bmatrix} 1 & X_1^{(1)} & X_2^{(1)} \\ 1 & X_1^{(2)} & X_2^{(2)} \\ \vdots & \vdots & \vdots \\ 1 & X_1^{(m)} & X_2^{(m)} \end{bmatrix} \longleftrightarrow \begin{array}{l} [[1, 40, 20] \\ [1, 20, 50], \\ \dots \\ [1, 5, 35]] \end{array}$$

Logistic Regression Cost Function

Cost function for logistic regression

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)} \log h(x^{(i)}, \theta) + (1 - y^{(i)})\boxed{\log(1 - h(x^{(i)}, \theta))}\right]$$



When label is 0 and output is 0, then Cost = 0, when label is 1 and output is 1 then cost = 0, else cost is +ve inf.

Week2 Naive Bayes

Summary

- Conditional probabilities ⟶ Bayes' Rule
- $P(X|Y) = P(Y|X) \times \frac{P(X)}{P(Y)}$

$P(w_i \mid class)$

word	Pos	Neg
I	3	3
am	3	3
happy	2	1
because	1	0
learning	1	1
NLP	1	1
sad	1	2
not	1	2
Nclass	**13**	**12**

$$p(I|Pos) = \frac{3}{13}$$

word	Pos	Neg
I	0.24	

Words having the same probability in both class don't add anything to sentiment.

Naïve Bayes

Tweet: I am happy today; I am learning.

$$\prod_{=1}^{m} \frac{P(w_i|pos)}{P(w_i|neg)} = \frac{0.14}{0.10} = 1.4 \ \mathbf{> 1}$$

$$\cancel{\frac{.20}{.20}} * \cancel{\frac{0.20}{0.20}} * \boxed{\frac{0.14}{0.10}} * \cancel{\frac{0.20}{0.20}} * \cancel{\frac{0.20}{0.20}} * \cancel{\frac{0.10}{0.10}}$$

word	Pos	Neg
I	0.20	0.20
am	0.20	0.20
happy	0.14	0.10
because	0.10	0.05
learning	0.10	0.10
NLP	0.10	0.10
sad	0.10	0.15
not	0.10	0.15

Laplacian Smoothing:

little transformation avoids the probability being zero.

Laplacian Smoothing

$$P(\mathrm{w_i}|\mathrm{class}) = \frac{\mathrm{freq}(\mathrm{w_i}, \mathrm{class})}{\mathrm{N_{class}}}$$

class $\in$ {Positive, Negative}

$$P(\mathrm{w_i}|\mathrm{class}) = \frac{\mathrm{freq}(\mathrm{w_i}, \mathrm{class}) + 1}{\mathrm{N_{class}} + \mathrm{V_{class}}}$$

N_{class} = frequency of all words in class

V_{class} = number of unique words in class

Log Likelihood

Log Likelihood

$$\frac{P(pos)}{P(neg)} \prod_{i=1}^{m} \frac{P(w_i|pos)}{P(w_i|neg)}$$

- Products bring risk of underflow
- $log(a * b) = log(a) + log(b)$
- $log(\frac{P(pos)}{P(neg)} \prod_{i=1}^{n} \frac{P(w_i|pos)}{P(w_i|neg)}) \Rightarrow log\frac{P(pos)}{P(neg)} + \sum_{i=1}^{n} log\frac{P(w_i|pos)}{P(w_i|neg)}$

log prior + log likelihood

Log Likelihood

doc: I am happy because I am learning.

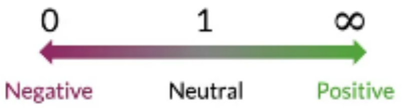
$$\sum_{i=1}^{m} log \frac{P(w_i|pos)}{P(w_i|neg)} = \sum_{i=1}^{m} \lambda(w_i)$$

word	Pos	Neg	λ
I	0.05	0.05	0
am	0.04	0.04	0
happy	0.09	0.01	2.2
because	0.01	0.01	0
learning	0.03	0.01	1.1
NLP	0.02	0.02	0
sad	0.01	0.09	-2.2
not	0.02	0.03	-0.4

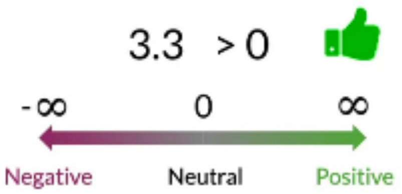
log likelihood = 0 + 0 + 2.2 + 0 + 0 + 0 + 1.1 = **3.3**

Log Likelihood

$$\prod_{i=1}^{m} \frac{P(w_i|pos)}{P(w_i|neg)} > 1$$



$$\sum_{i=1}^{m} log \frac{P(w_i|pos)}{P(w_i|neg)} > 0$$



Summary

0. Get or annotate a dataset with positive and negative tweets
1. Preprocess the tweets: process_tweet(tweet) → [w_1, w_2, w_3, ...]
2. Compute freq(w, class)
3. Get P(w | pos), P(w | neg)
4. Get λ(w)
5. Compute logprior = log(P(pos) / P(neg))

Testing Naive Bayes

Predict using Naïve Bayes

- log-likelihood dictionary $\lambda(w) = log\frac{P(w|pos)}{P(w|neg)}$
- $logprior = log\frac{D_{pos}}{D_{neg}} = 0$
- Tweet: [I, pass, the, NLP, interview]

$$score = -0.01 + 0.5 - 0.01 + 0 + logprior = 0.48$$

$$pred = score > 0$$

word	λ
$\boxed{\text{I}}$	$\boxed{-0.01}$
$\boxed{\text{the}}$	$\boxed{-0.01}$
happi	0.63
because	0.01
$\boxed{\text{pass}}$	$\boxed{0.5}$
$\boxed{\text{NLP}}$	$\boxed{0}$
sad	-0.75
not	-0.75

Naïve Bayes Applications

- Sentiment analysis
- Author identification
- Information retrieval
- Word disambiguation
- Simple, fast and robust!

Summary

- X_{val} $Y_{val} \longrightarrow$ Performance on unseen data
- Predict using λ and $logprior$ for each new tweet
- Accuracy $\longrightarrow \frac{1}{m}\sum_{i=1}^{m}(pred_i == Y_{val_i})$
- What about words that do not appear in λ(w)?

Summary

- Independence: Not true in NLP
- Relative frequency of classes affect the model

Error Analysis

Processing as a Source of Errors: Removing Words

Tweet: This is not good, because your attitude is not even close to being nice.

processed_tweet: [good, attitude, close, nice]

Processing as a Source of Errors: Word Order

Tweet: I am happy because I did not go.

Tweet: I am not happy because I did go.

Adversarial attacks

Sarcasm, Irony and Euphemisms

Tweet: This is a ridiculously powerful movie. The plot was gripping and I cried right through until the ending!

processed_tweet: [ridicul, power, movi, plot, grip, cry, end]

Week3 (Word Embedding)

Vector Space Models:

Why learn vector space models?

Where are you heading?	What is your age?
Where are you from?	How old are you?
Different meaning	Same Meaning

Fundamental concept

"You shall know a word by the company it keeps" Firth, 1957

(Firth, J. R. 1957:11)

1. Vector Space model helps to derive the dependencies between words

2. Vector space models are used in information extraction to answer the question in the style of who, what, where, how etc., chatbot programming and machine translation.

Cooccurance Matrix of words:

1. The Number of times they occur together in a corous within a certain distance k.

Word by Word Design

Number of times they ***occur together within a certain distance k***

I like simple data

I prefer simple raw data

k=2

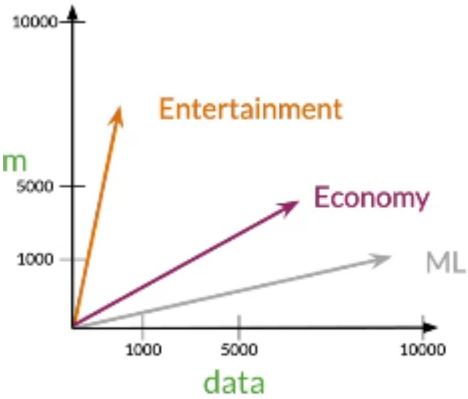
	simple	raw	like	I
data	2	1	1	0

Word by Document Design

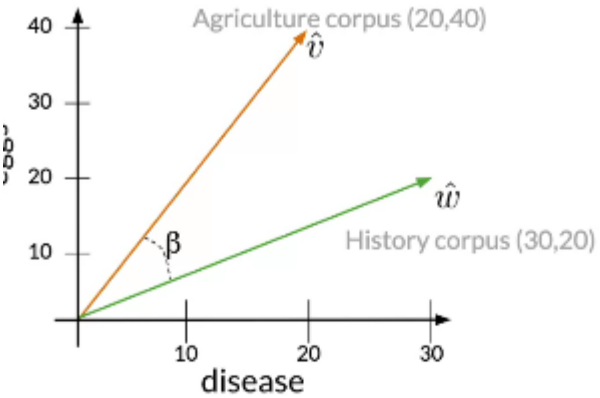
Number of times a word *occurs within a certain category*

	Entertainment	Economy	Machine Learning
data	500	6620	9320
film	7000	4000	1000

Vector Space



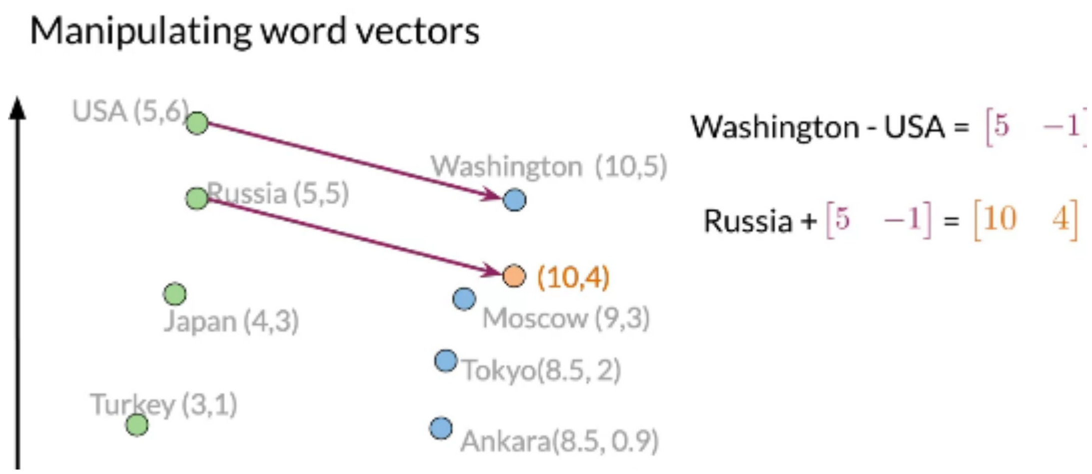
Cosine Similarity



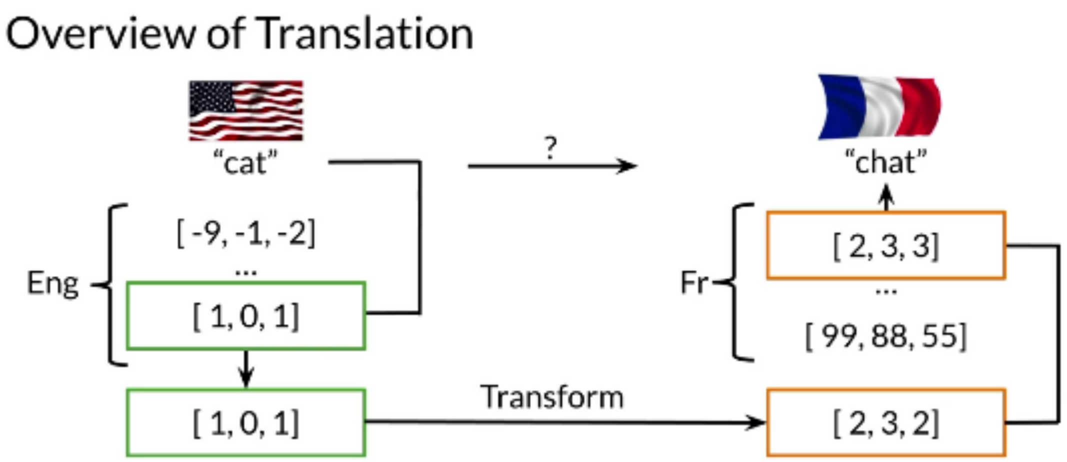
$$\hat{v} \cdot \hat{w} = \|\hat{v}\| \|\hat{w}\| \cos(\beta)$$

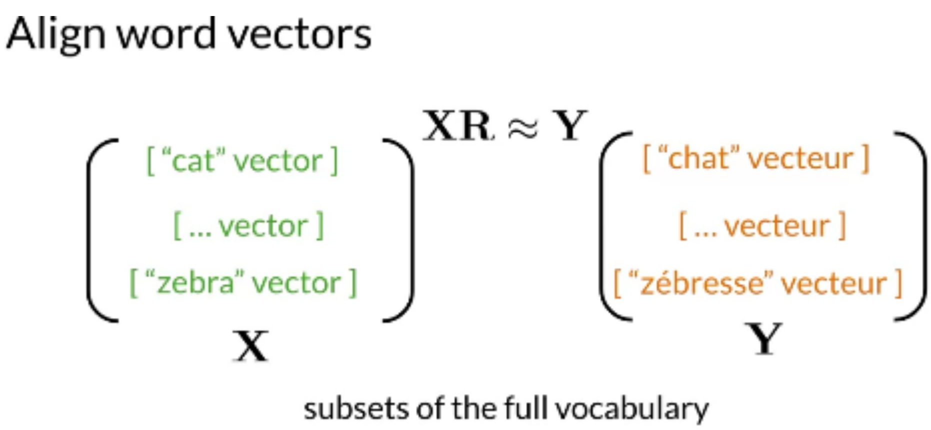
$$\boxed{\cos(\beta) = \frac{\hat{v} \cdot \hat{w}}{\|\hat{v}\| \|\hat{w}\|}}$$

$$= \frac{(20 \times 30) + (40 \times 20)}{\sqrt{20^2 + 40^2} \times \sqrt{30^2 + 20^2}}$$



Week4: Machine Translation





initialize R

in a loop:

$$Loss = \parallel \mathbf{XR} - \mathbf{Y} \parallel_F$$

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

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

Frobenius norm

$$\parallel \mathbf{XR} - \mathbf{Y} \parallel_F$$

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

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

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

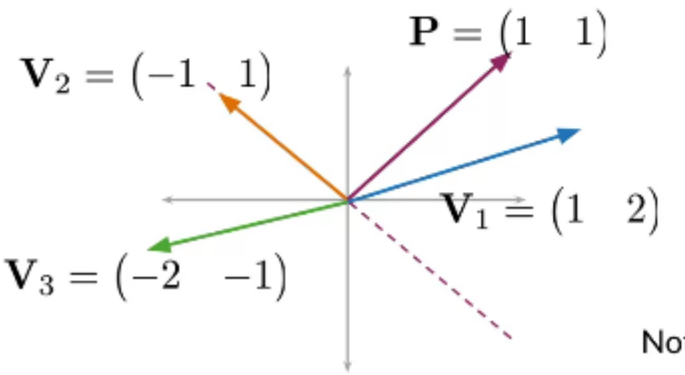
$$\|\mathbf{A}\|_F \equiv \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}$$

Gradient

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

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

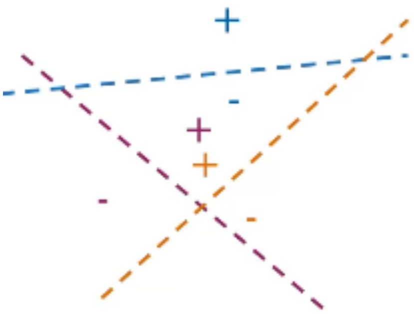
Which side of the plane?



$$\mathbf{P}\mathbf{V}_1^T = 3$$
$$\mathbf{P}\mathbf{V}_2^T = 0$$
$$\mathbf{P}\mathbf{V}_3^T = -3$$

Notice the signs?

Multiple planes, single hash value?



$$\boxed{\mathbf{P}_1\mathbf{v}^T = 3, sign_1 = +1, h_1 = 1}$$
$$\boxed{\mathbf{P}_2\mathbf{v}^T = 5, sign_2 = +1, h_2 = 1}$$
$$\boxed{\mathbf{P}_3\mathbf{v}^T = -2, sign_3 = -1, h_3 = 0}$$

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

Multiple planes, single hash value!!

```python
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
```