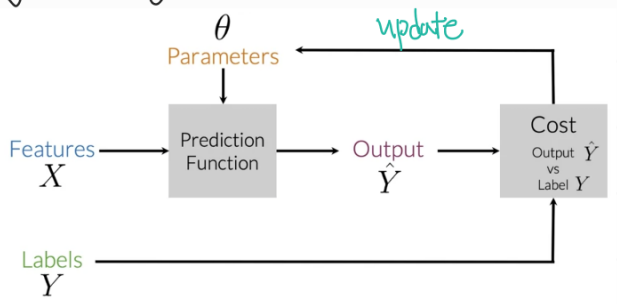
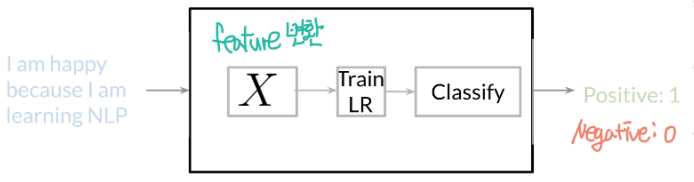
Week 1

Lecture

Logistic Regression





Vocabulary & Feature Extraction

I am happy because I am learning NLP

Index

All zeros!

[1, 1, 1, 1, 1, 1, 1, ..., 0, ..., 0, 0, 0]

1 ⟷ |V| — Vocabulary size

$$[\theta_0, \theta_1, \theta_2, \ldots, \theta_n] \longrightarrow \begin{cases} \text{1. Large training time} \\ \text{2. Large prediction time} \end{cases}$$

$$n = |V|$$

- V가 커질수록 vector는 sparse. 학습시킬 파라메터도 많아짐

Feature Extraction with Frequencies

Given a corpus with positive and negative tweets as follows:

Positive tweets	Negative tweets
I am happy because I am learning NLP I am happy	I am sad, I am not learning NLP I am sad

) 전체 corpus로 Pos/Neg 각각 단어등장횟수 map형성

Vocabulary	PosFreq (1)	NegFreq (0)
I	3	3

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

추출된 input: I am sad, I am not learning NLP

:tionary mapping from
lass to its frequency 단어의 값 더함

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

8

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

I am sad, I am not learning NLP

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

11

NegFreq에서 해당하는 단어의 값더함

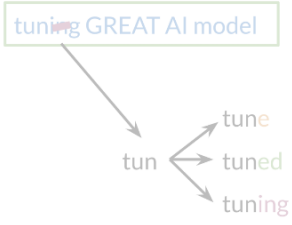
bias

∴ [1, 8, 11]의 3차원 벡터 생성

Preprocessing

1. Eliminate handles and URLs
2. Tokenize the string into words.
3. Remove stop words like "and, is, a, on, etc."
4. Stemming- or convert every word to its stem. Like dancer, dancing, danced, becomes 'danc'. You can use porter stemmer to take care of this.
5. Convert all your words to lower case.

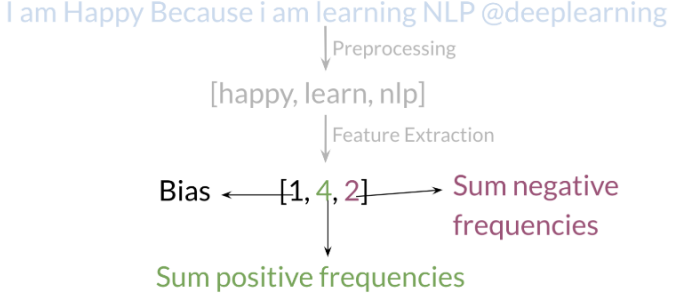
For example the following tweet "@YMourri and @AndrewYNg are tuning a GREAT AI model at https://deeplearning.ai!!!" after preprocessing becomes



Preprocessed tweet:
[tun, great, ai, model]

Putting it all together

Over all , you start with a given text, you perform preprocessing, then you do feature extraction to convert text into numerical representation as follows:



Your X becomes of dimension $(m, 3)$ as follows.

$$\boldsymbol{X} = \begin{bmatrix} 1 & X_1^{(1)} & X_2^{(1)} \\ 1 & X_1^{(2)} & X_2^{(2)} \\ \vdots & \vdots & \vdots \\ 1 & X_1^{(m)} & X_2^{(m)} \end{bmatrix}$$

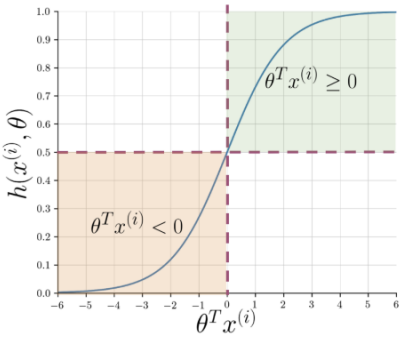
Logistic Regression Overview

Sigmoid function

$$h(x^{(i)}, \theta) = \frac{1}{1 + e^{-\theta^T x^{(i)}}}$$

input: $x^{(i)}$

weights parameter: θ



@YMourri and @AndrewYNg are tuning a GREAT AI model

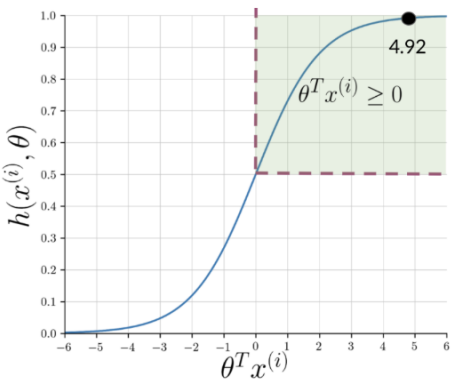
↓

[tun, ai, great, model]

↓

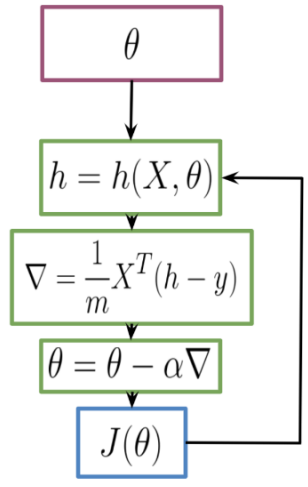
$$x^{(i)} = \begin{bmatrix} 1 \\ 3476 \\ 245 \end{bmatrix} \quad \theta = \begin{bmatrix} 0.00003 \\ 0.00150 \\ -0.00120 \end{bmatrix}$$

prediction

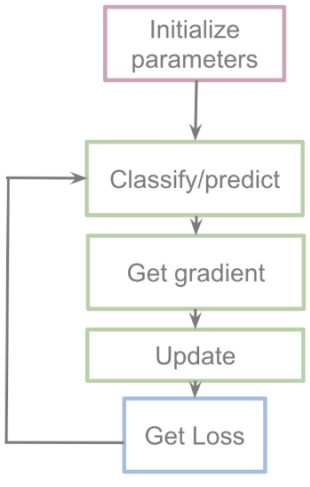


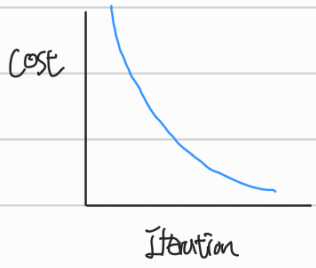
Logistic Regression : Training

To train your logistic regression function, you will do the following:



Until good enough





Logistic Regression : Training

- X_{val} Y_{val} θ

$h(X_{val}, \theta)$

$pred = h(X_{val}, \theta) \geq 0.5$

보통 $X_{train} : X_{val} : X_{test} = 8:1:1$

0.5 ≥ : pos

0.5 < : neg

$$\begin{bmatrix} 0.3 \\ 0.8 \\ 0.5 \\ \vdots \\ h_m \end{bmatrix} \geq 0.5 = \begin{bmatrix} 0.3 \geq 0.5 \\ 0.8 \geq 0.5 \\ 0.5 > 0.5 \\ \vdots \\ pred_m \geq 0.5 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ \vdots \\ pred_m \end{bmatrix}$$

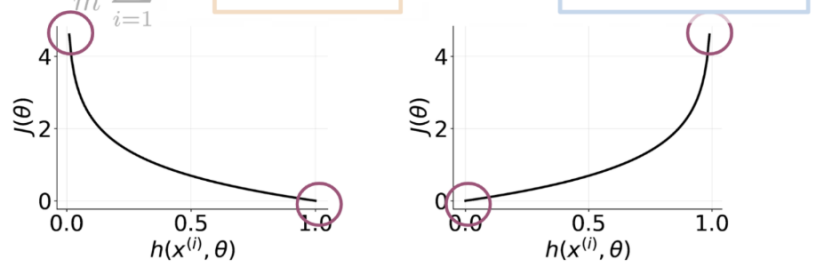
Accuracy →

$$\sum_{i=1}^{m} \frac{(pred^{(i)} == y_{val}^{(i)})}{m}$$

Logistic Regression : Cost Function

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

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



내가 커지도록 θ을 학습하고싶다.

$P(y|x^{(i)},\theta) = h(x^{(i)},\theta)^{y^{(i)}}(1-h(x^{(i)},\theta)^{(1-y^{(i)})}$

From the above, you can see that when $y=1$, you get $h(x^{(i)},\theta)$, and when $y=0$, you get $(1-h(x^{(i)},\theta))$, which makes sense, since the two probabilities equal to 1. In either case, you want to maximize the function $h(x^{(i)},\theta)$ by making it as close to 1 as possible. When $y=0$, you want $(1-h(x^{(i)},\theta))$ to be 0, and therefore $h(x^{(i)},\theta)$ close to 0. When $y=1$, you want $h(x^{(i)},\theta)=1$.

잘못설명한듯 →

entire dataset에 대해

$$L(\theta) = \prod_{i=1}^{m} h(\theta, x^{(i)})^{y^{(i)}}(1-h(\theta,x^{(i)}))^{(1-y^{(i)})}$$

$$\max_{h(x^{(i)},\theta)} \log L(\theta) = \log\prod_{i=1}^{m} h(x^{(i)},\theta)^{y^{(i)}}(1-h(x^{(i)},\theta)^{(1-y^{(i)})}$$

$$= \sum_{i=1}^{m}\log h(x^{(i)},\theta)^{y^{(i)}}(1-h(x^{(i)},\theta)^{(1-y^{(i)})}$$

$$= \sum_{i=1}^{m}\log h(x^{(i)},\theta)^{y^{(i)}} + \log(1-h(x^{(i)},\theta)^{(1-y^{(i)})}$$

$$= \sum_{i=1}^{m} y^{(i)}\log h(x^{(i)},\theta) + (1-y^{(i)})\log(1-h(x^{(i)},\theta))$$

Hence, we now divide by m, because we want to see the average cost.

$$\frac{1}{m}\sum_{i=1}^{m} y^{(i)}\log h(x^{(i)},\theta) + (1-y^{(i)})\log(1-h(x^{(i)},\theta))$$

maximize → minimize

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

A vectorized implementation is:

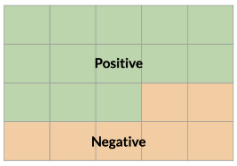
$$h = g(X\theta)$$
$$J(\theta) = \frac{1}{m}\cdot\left(-y^{T}\log(h) - (1-y)^{T}\log(1-h)\right)$$

Week 2

Naive Bayes

· Probability and Bayes' Rule

Corpus of tweets



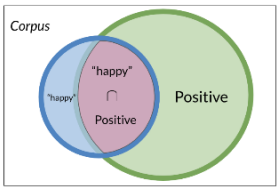
A → Positive tweet

$P(A) = N_{pos} / N = 13 / 20 = 0.65$

P(Negative) = 1 - P(Positive) = 0.35

To calculate a probability of a certain event happening, you take the count of that specific event and you divide by the sum of all events. Furthermore, the sum of all probabilities has to equal 1.

$$P(A \cap B) = P(A, B) = \frac{3}{20} = 0.15$$



· Bayes' Rule

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}$$

· Naive Bayes Introduction - Classifier 만들기

① Corpora에서 word count해서 table 생성

Positive tweets

I am happy because I am learning NLP

I am happy, not sad.

Negative tweets

I am sad, I am not learning NLP

I am sad, not happy

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
N_{class}	13	12

② 확률로 변환

word	Pos	Neg
I	0.24	0.25
am	0.24	0.25
happy	0.15	0.08
because	0.08	0
learning	0.08	0.08
NLP	0.08	0.08
sad	0.08	0.17
not	0.08	0.17

③ Likelihood를 계산해서 classify

Tweet: I am happy today; I am learning.

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

1 > 이면 positive.

$$\frac{0.20}{0.20} * \frac{0.20}{0.20} * \frac{0.14}{0.10} * \frac{0.20}{0.20} * \frac{0.20}{0.20} * \frac{0.10}{0.10}$$

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

일반확률일 경우, 단어가 training에 나타나지 않으면 확률이 0.

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

class ∈ {Positive, Negative}

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

아예 0은 아님

V개의 분모항을 분자에 더해줌

※ of unique words in vocabulary.

Log Likelihood

To do inference, you can compute the following:

$\frac{P(pos)}{P(neg)} \prod_{i=1}^{m} \frac{P(w_i|pos)}{P(w_i|neg)} > 1$ — log를 취한 후에는 이 판단기준

As m gets larger, we can get numerical flow issues, so we introduce the $\log$, which gives you the following equation:

$$\log\left(\frac{P(pos)}{P(neg)} \prod_{i=1}^{n} \frac{P(w_i|pos)}{P(w_i|neg)}\right) \Rightarrow \log \frac{P(pos)}{P(neg)} + \sum_{i=1}^{n} \log \frac{P(w_i|pos)}{P(w_i|neg)}$$

log prior / log likelihood

log likelihood 구할때 λ를 더하면 됨.

doc: I am happy because I am learning.

$$\lambda(w) = log \frac{P(w|pos)}{P(w|neg)}$$

$$\lambda(\text{happy}) = log \frac{0.09}{0.01} \approx 2.2$$

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

doc: I am happy because I am learning.

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

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

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

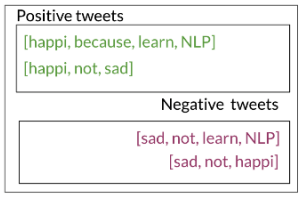
Training naive Bayes

① Get or annotate a dataset with positive and negative tweets.

② Preprocess the tweets: tweet → $[w_1, w_2, w_3 \dots]$

- Lowercase
- Remove punctuation, urls, names
- Remove stop words
- Stemming
- Tokenize sentences

③ Compute freq(w, class)



Step 2: Word count

word	Pos	Neg
happi	2	1
because	1	0
learn	1	1
NLP	1	1
sad	1	2
not	1	2
N_{class}	7	7

freq(w, class)

④ P(w|pos), P(w|neg) 구하기

⑤ λ(w) 구하기 : $\lambda(w) = \log \frac{P(w|pos)}{P(w|neg)}$

⑥ log prior : $\log \frac{P(pos)}{P(neg)}$ 구하기

$= \log \frac{D_{pos}}{D_{neg}}$, D_{pos}: # of positive documents

Testing naive Bayes

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

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

$pred = score > 0$

word	λ
I	-0.01
the	-0.01
happi	0.63
because	0.01
pass	0.5
NLP	0
sad	-0.75
not	-0.75

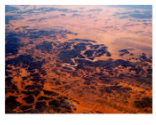
Application of Naive Bayes

- Author identification
- Spam filtering
- Information retrieval
- Word disambiguation

This method is usually used as a simple baseline. It also really fast.

Naive Bayes Assumptions

① Independence



"It is sunny and hot in the Sahara desert."

"It's always cold and snowy in ___."

In the first image, you can see the word sunny and hot tend to depend on each other and are correlated to a certain extent with the word "desert". Naive Bayes assumes independence throughout. Furthermore, if you were to fill in the sentence on the right, this naive model will assign equal weight to the words "spring, summer, fall, winter".

② Relative frequency

Relative frequencies in corpus

On Twitter, there are usually more positive tweets than negative ones. However, some "clean" datasets you may find are artificially balanced to have to the same amount of positive and negative tweets. Just keep in mind, that in the real world, the data could be much noisier.

Error Analysis

① Removing words

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

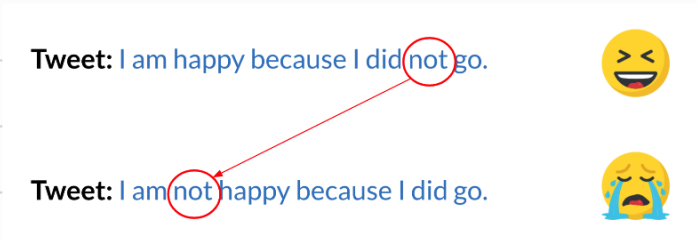
processed_tweet: [good, attitude, close, nice]

② Removing punctuation

Tweet: My beloved grandmother :(

processed_tweet: [belov, grandmoth]

③ Word order



④ Adversarial attacks

Ex. Sarcasm (풍자), Irony (역설), Euphemisms (완곡한)

Week 3

Vector Space Models

• Word by Word Design — 문맥의 단어를 세서 matrix 만들기

I like simple data

I prefer simple raw data

Window size: k=2

	simple	raw	like	I
data	2	1	1	0

n

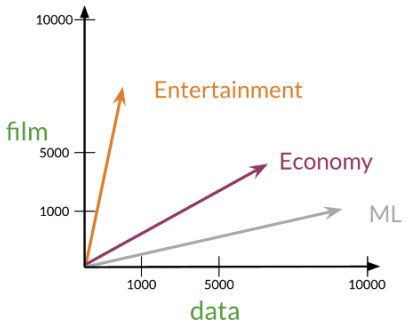
• Word by Document Design

Corpus

Doc

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

※ of times film appeared in Entertainment Doc



	Entertainment	Economy	ML
data	500	6620	9320
film	7000	4000	1000

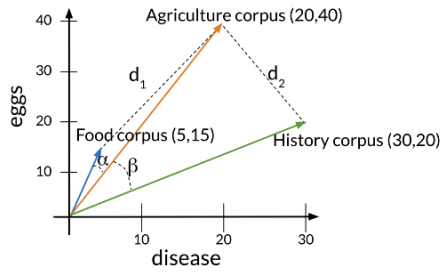
[7500, 7000] : category를 vector로 나타낼 수 있음

Measures of "similarity:"
Angle
Distance

• Euclidian Distance

n-dim : $d(\vec{v},\vec{w}) = \sqrt{\sum_{i=1}^{n}(v_i - w_i)^2}$

• Cosine Similarity : Intuition



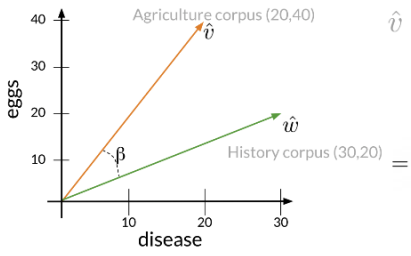
Euclidean distance: $d_2 < d_1$

Angles comparison: $\beta > \alpha$

The cosine of the angle between the vectors

• norm : $\|\vec{v}\| = \sqrt{\sum_{i=1}^{n} |v_i|^2}$

• dot product : $\vec{v} \cdot \vec{w} = \sum_{i=1}^{n} v_i \cdot w_i$

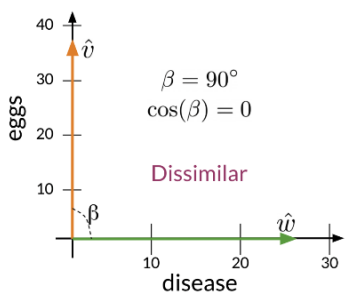


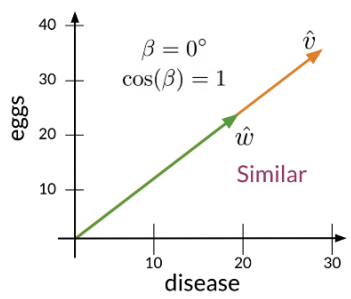
$$\hat{v} \cdot \hat{w} = \|\hat{v}\| \|\hat{w}\| \cos(\beta)$$

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

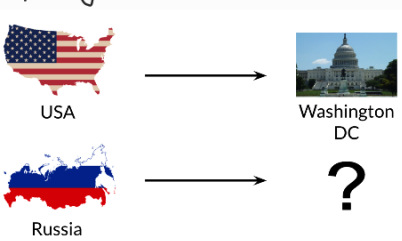
$$= \frac{(20 \times 30) + (40 \times 20)}{\sqrt{20^2 + 40^2} \times \sqrt{30^2 + 20^2}}$$

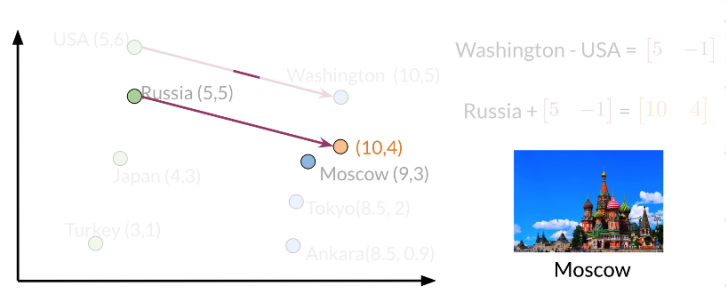
$$= 0.87$$





• Manipulating Words in Vector Spaces.





• Visualization and PCA

- unsupervised learning algorithm.
- reduce dimension of data

Visualization of word vectors

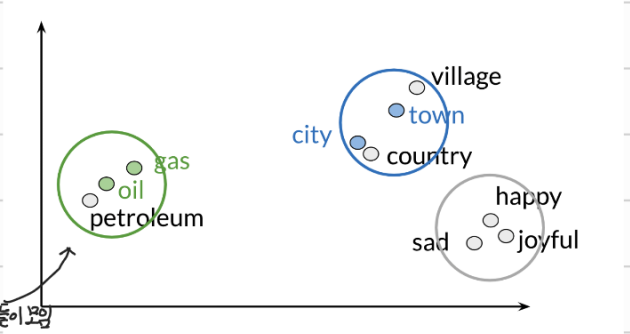
	d > 2		
oil	0.20	...	0.10
gas	2.10	...	3.40
city	9.30	...	52.1
town	6.20	...	34.3

How can you visualize if your representation captures these relationships?


oil & gas


town & city

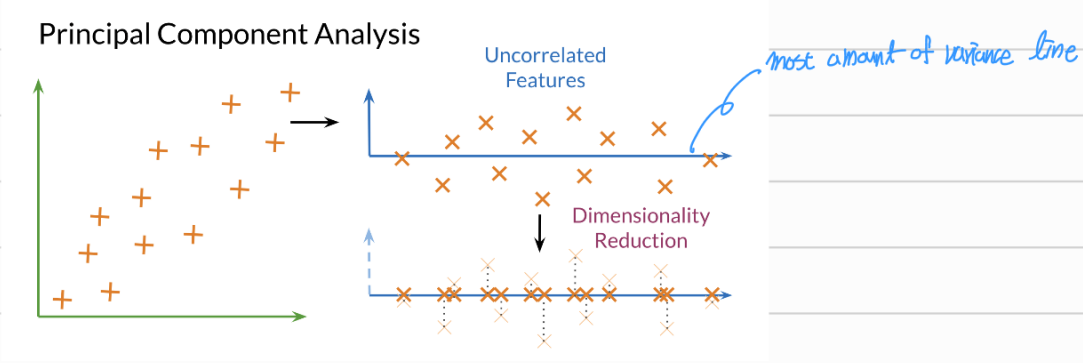
d=2



POS: part of speech. 품사 (세팔) 태깅.

• 많은 training 알고리즘들이 주변단어를 식별하며 학습하기 때문

- PCA algorithm

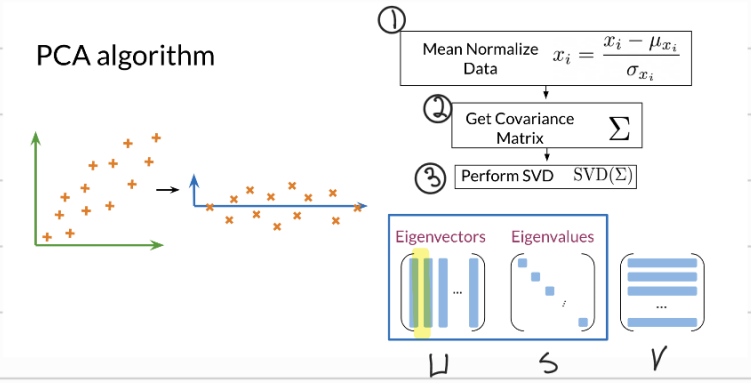


Eigenvector: the resulting vector. uncorrelated features

Eigenvalue: the amount of information retained by each new feature

variance in the eigenvector

- Each eigenvalue has corresponding eigenvector. Eigenvalue tells you how much variance there is in the eigenvector.



∴ use the first n columns of vector U, to get new data by multiplying

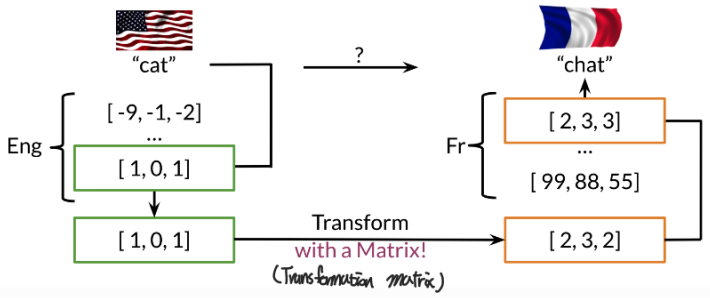
$X U[:, 0:n]$

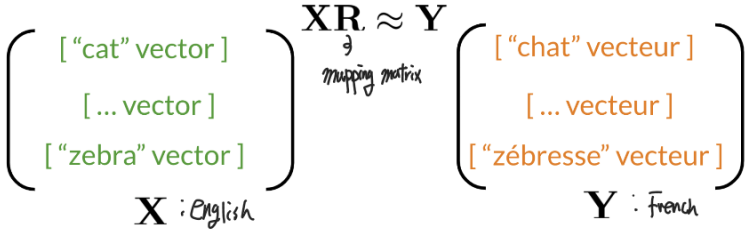
$m \times p \quad p \times n \Rightarrow m \times n$

Week 4

Machine Translation and Document Search

- Transforming word vectors





subsets of the full vocabulary

Steps required to learn R:

- Initialize R
- For loop

Frobenius norm

$$Loss = \|XR - Y\|_F^2$$

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

$$R = R - \alpha * g$$

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

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

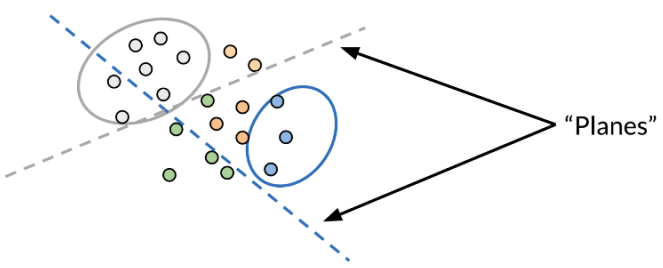
$$\|\mathbf{A}_F\| = \sqrt{2^2 + 2^2 + 2^2 + 2^2}$$
$$\|\mathbf{A}_F\| = 4$$

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

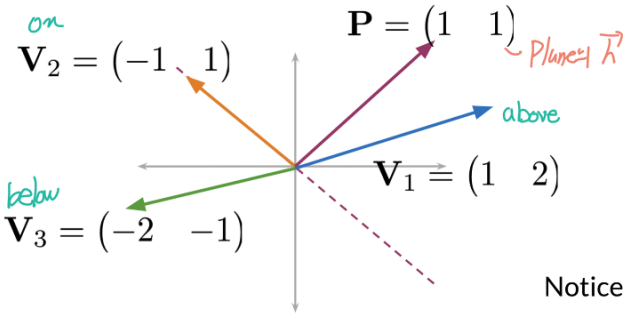
In summary you are making use of the following:

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

• Locality Sensitive Hashing



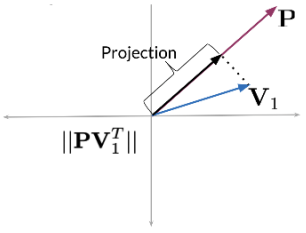
- Hash similar inputs into the same bucket with high probability
- Clustering the points by deciding whether they are above or below the plane



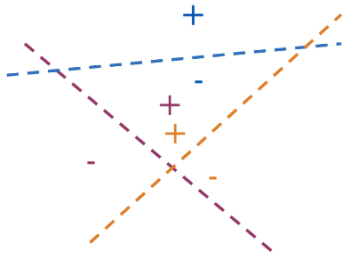
Projection + 내적

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

Notice the signs?



• Multiple Planes



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

Positive: 1

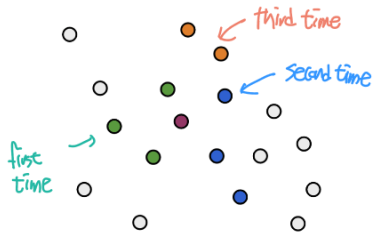
Negative: 0

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

hash value: $\sum_i^H 2^i \times h_i$

- At each time you are asking the plane to which side will you find the point (i.e. 1 or 0) until you find your point bounded by the surrounding planes.

• Approximate nearest neighbors



Approximate nearest
(friendly) neighbors

Tradeoff : accuracy ↔ efficiency

• hash bucket에 있는 점들을 계산하듯 - Random plane을 사용해서 hash 값 계산

• Searching Document

- Document에 등장하는 word vector를 모두 더해서 Doc vector 생성 가능

〈Course 요약〉

- Transform vector
- "K nearest neighbors"
- Hash tables
- Divide vector space into regions
- Locality sensitive hashing
- Approximated nearest neighbors

→ Machine translation

→ Document search

Practice your skills in the assignment!