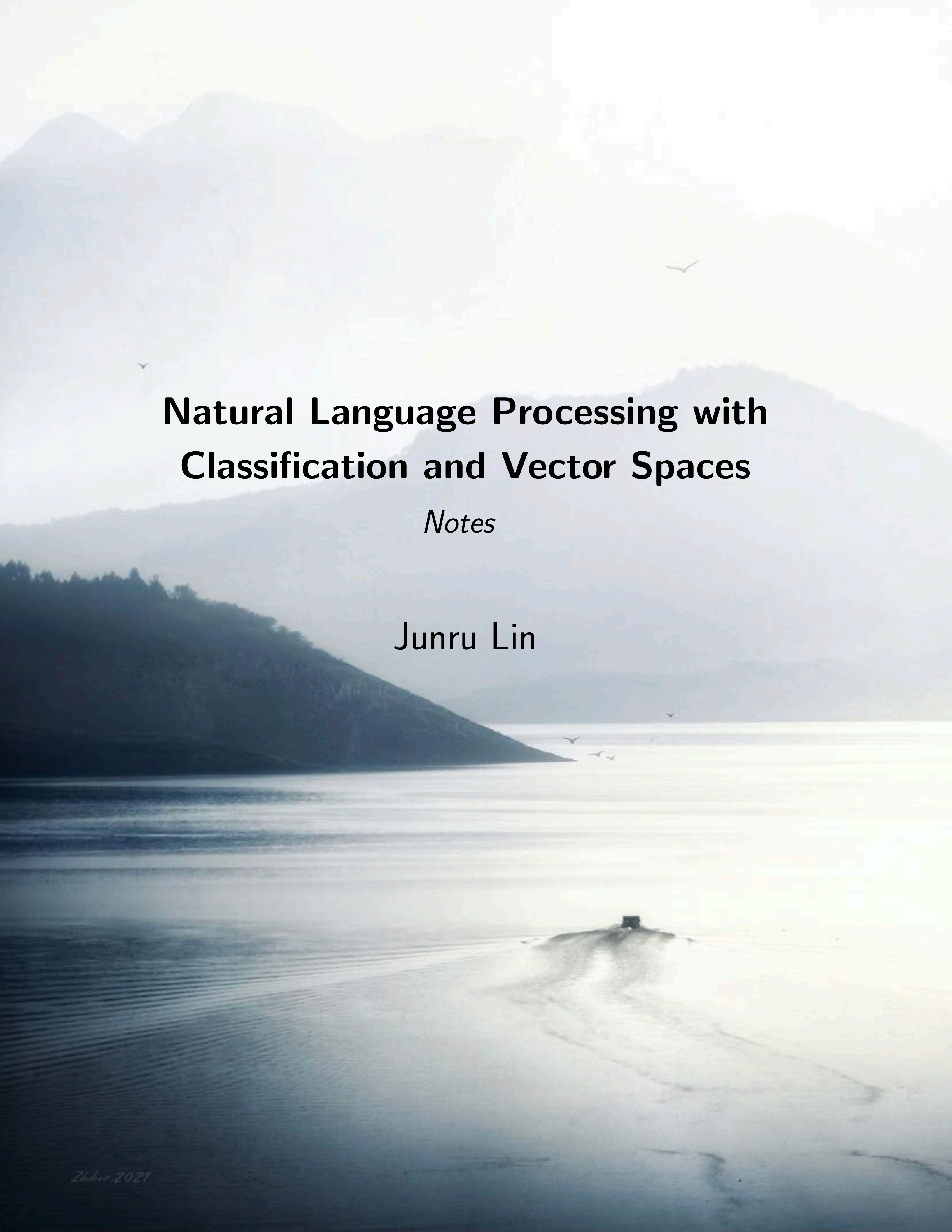


Natural Language Processing with Classification and Vector Spaces

Notes

Junru Lin

Contents

Chapter 1

Logistic Regression

1.1 Learning Objectives

- Sentiment analysis
- Logistic regression
- Data pre-processing
- Calculating word frequencies
- Feature extraction
- Vocabulary creation
- Supervised learning

1.2 Supervised ML (Training)

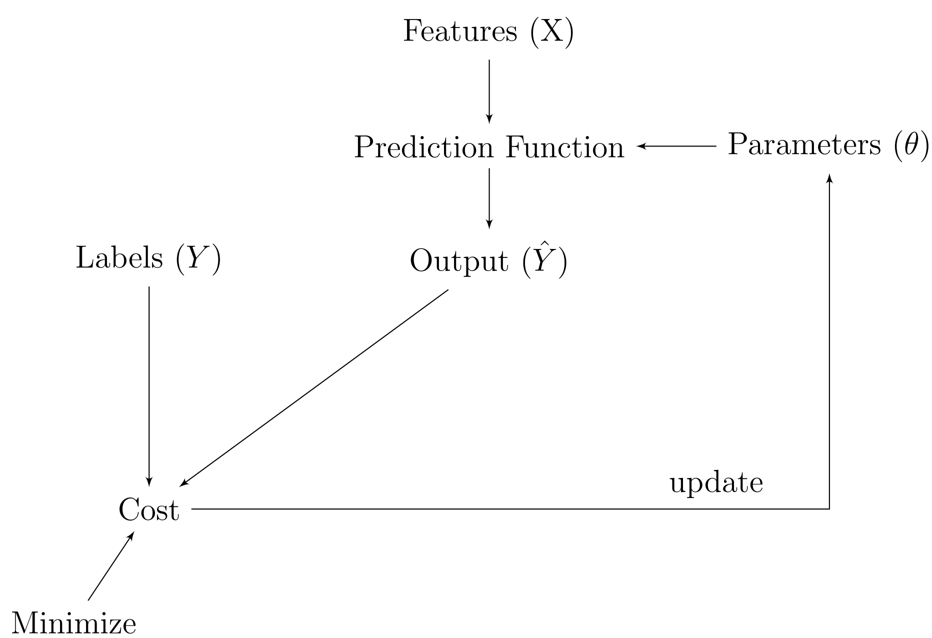


Figure 1.1: Flowchart of Supervised ML (Training)

1.3 Sentiment Analysis

Example
I am happy because I am learning NLP.

Classification Label

- Positive - 1
- Negative - 0

Training
Use Logistic Regression

1.4 Vocabulary

1.4.1 Vocabulary and Frequency

Vocabulary
All unique words in a dataset.

For Each data

An array of 0's and 1's.
$|v|$ = size of vocabulary (number of features)
Logistic Regression: $\theta = \left[\theta_0, \theta_1, \theta_2, \ldots, \theta_n\right]$ $(\theta_0 \rightsquigarrow b$ $(y = ax + b)$, $\quad n = |v|)$

Negative and Positive Frequencies
freqs: dictionary mapping from [word, class] to frequency
example: freqs[('I', 1)] = 2

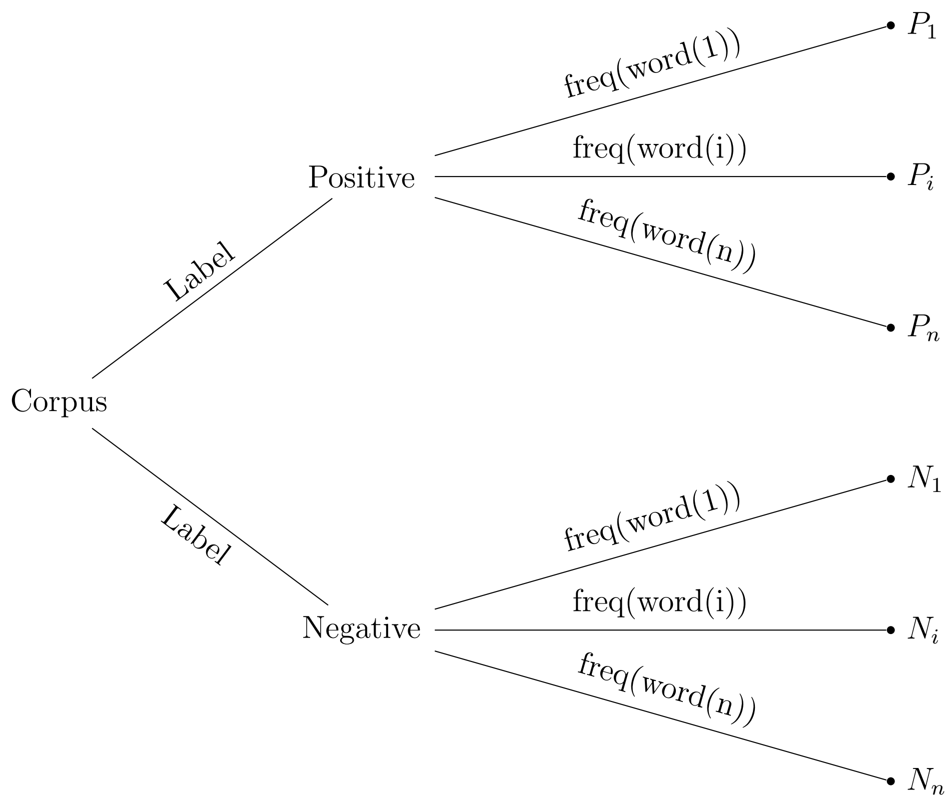


Figure 1.2: Word Frequency Tree

Vocabulary	PosFreq(1)	NegFreq(0)
word(1)	P_1	N_1
$\vdots$	$\vdots$	$\vdots$
word(n)	P_n	N_n

1.4.2 Feature Extraction with Frequencies

$$X_m = \left[1, \sum_m freqs(w, 1), \sum_m freqs(w, 0)\right]$$

- X_m: Feature of data m in the dataset

- 1: Bias
- $\sum_m freqs(w, 1)$: Sum of Pos. Freq
- $\sum_m freqs(w, 0)$: Sum of Neg. Freq

Example
I am sad.
I am not learning NLP.
$\hookrightarrow$ words (w): I, am, sad, not, learning, NLP

1.5 Preprocessing

Stop Words
and, is, a, at, has, for, of, $\cdots$

Punctuation
, . " ! " ' ' . . .

Handles and URLs
@ . . .

Stemming
tune, tuned, tuning $\rightarrow$ tun

lowercasing
Great, GREAT $\rightarrow$ great

1.6 Putting it All Together

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

where m is the sample size.

Example

$$\begin{array}{c} [\,[1, 40, 20], \\ [1, 20, 50], \\ \ldots, \\ [1, 5, 35]\,] \end{array}$$

1.7 Logistic Regression (LR) Overview

Sigmoid Function

$$h(x^{(i)}, \theta) = \frac{1}{1 + e^{-\theta^T x^{(i)}}} \in (0, 1)$$

$$\theta^T x^{(i)} \geq 0 : \; h(x^{(i)}, \theta) \geq 0.5 \to 1$$

$$\theta^T x^{(i)} < 0 : \; h(x^{(i)}, \theta) < 0.5 \to 0$$

1.8 LR: Training

$$\begin{tikzcd}
\text{Initialize parameters: } \theta \arrow[d] \\
\text{Classify/Predict: } h = h(X, \theta) \arrow[d] \\
\text{Get gradient: } \nabla = \frac{1}{m} X^T(h - y) \arrow[d] \\
\text{Update: } \theta = \theta - \alpha \nabla \arrow[d] \\
\text{Get loss: } J(\theta) \arrow[uuu, bend right]
\end{tikzcd}$$

1.9 LR: Testing

- Validation set: X_{val}, Y_{val}
- After training: θ

- pred $= h(X_{val},\ \theta) \geq 0.5$ (sigmoid funvtion)
- Accuracy $= \displaystyle\sum_{i=1}^{m} \frac{pred^{(i)} == y_{val}^{(i)}}{m} = \frac{\sum_{i=1}^{m}(pred^{(i)} == y_{val}^{(i)})}{m}$
- $X_{train} : X_{val} : X_{test} = 8 : 1 : 1$

Example

$$\begin{bmatrix} 0.3 \\ 0.8 \\ \vdots \\ h_m \end{bmatrix} \geq 0.5 = \begin{bmatrix} 0.3 \geq 0.5 \\ 0.7 \geq 0.5 \\ \vdots \\ hm \geq 0.5 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ pred_m \end{bmatrix}$$

1.10 LR: Costing Function

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

Want $\min_{\theta} J(\theta)$

Chapter 2

Probability and Bay's Rule

2.1 Probability

$A \rightarrow$ ***Positive tweet***

$$P(A) = P(Positive) = N_{pos}/N$$
$$P(Negative) = 1 - P(Positive)$$

$B \rightarrow$ ***tweet contains "happy"***

$$P(B) = P(happy) = N_{happy}/N$$

Probability of the intersection

$$P(A \cap B) = P(A, B)$$

2.2 Bayes' Rule

Conditional Probability
Probability of A, given B happened. $\rightarrow P(A|B)$

Bayes' Rule

$$\begin{cases} P(A|B) = \dfrac{P(A \cap B)}{P(B)} \\ P(B|A) = \dfrac{P(A \cap B)}{P(A)} \end{cases}$$
$$\Rightarrow P(A|B) = P(B|A) \cdot \frac{P(A)}{P(B)}$$

2.3 Naive Bayes Introduction

Assumption
$X_1, X_2, \ldots, X_m$ are independent

By Bayes' Rule

$$\begin{aligned} P(Positive|X) &= P(X|Positive) \cdot P(Positive)/P(X) \\ &= P(X_1|Positive) \cdots P(X_m|Positive) \cdot P(Positive)/P(X) \\ &= \frac{P(Positive)}{P(X)} \cdot \prod_{i=1}^{m} P(X_i|Positive) \\ P(Negative|X) &= \frac{P(Negative)}{P(X)} \cdot \prod_{i=1}^{m} P(X_i|Negative) \end{aligned}$$

Assumption
$P(Positive) = P(Negative)$

$$\begin{aligned} \Rightarrow \frac{P(Positive|X)}{P(Negative|X)} &= \frac{\prod_{i=1}^{m} P(X_i|Positive)}{\prod_{i=1}^{m} P(X_i|Negative)} \\ &= \prod_{i=1}^{m} \frac{P(X_i|Positive)}{P(X_i|Negative)} \end{aligned}$$

Note

- In a positive words table, we will have the probability for each word, which is $P(X_i|Positive)$.
- Same for $P(X_i|Negative)$.

2.4 Laplacian Smoothing

To avoid $P(W_i|class) = 0$

- N_{class} = frequency of all words in class
- V = number of unique words in vocabulary
- class $\in$ { Positive, Negative }
- $P(W_i|class) = \dfrac{freq(w_i|class)}{N_{class}}$
  $\downarrow$

- $P(W_i|class) = \dfrac{freq(w_i|class) + 1}{N_{class} + V}$

$$freq(w_i|class) + 1 \rightsquigarrow \text{ make it} \neq 0$$

$$N_{class} + V \rightsquigarrow \text{ make it add up to 1}$$

2.5 Log Likelihood

Ratio of probability

$$ratio(w_i) = \frac{P(w_i|Pos)}{P(w_i|Neg)} \approx \frac{ferq(w_i, 1) + 1}{freq(w_i, 0) + 1}$$

$$0(Negative) \longleftarrow 1(Neutral) \longrightarrow \infty(positive)$$

Naive Bayes, without assumption $P(Pos) = P(Neg)$

$$\frac{P(Pos|w)}{P(Neg|w)} = \frac{P(Pos)}{P(Neg)} \cdot \prod_{i=1}^{m} \frac{P(w_i|Pos)}{P(w_i|Neg)}$$

w: set of m words.

Log Likelihood

$$\log\left(\frac{P(Pos|w)}{P(Neg|w)}\right) = \underbrace{\log \frac{P(Pos)}{P(Neg)}}_{\text{log prior}} + \underbrace{\sum_{i=1}^{m} \log\left(\frac{P(w_i|Pos)}{P(w_i|Neg)}\right)}_{\text{log likelihood}}$$

Definition of $\lambda(w)$

$$\lambda(w) = \log \frac{P(pos)}{P(neg)} \quad \text{(stored in a table for reference)}$$

$$\log \prod_{i=1}^{m} ratio(w_i) = \sum_{i=1}^{m} \lambda(w_i)$$

$$-\infty(Negative) \longleftarrow 0(Neutral) \longrightarrow \infty(positive)$$

2.6 Training Naive Bayes

Steps (Example of tweets)

1. Get or annotate a dataset with positive and negative tweets
2. Preprocess the tweets
3. Compute $freq(w, class)$
4. Get $P(w|pos)$, $P(w|neg)$
5. Get $\lambda(w)$
6. Compute $logprior = \log \frac{P(pos)}{P(neg)}$

2.7 Testing Naive Bayes

- $X_{val}, Y_{val}, \lambda, log\ prior$
- $score = predict(X_{val}, \lambda, log\ prior)$
- $pred = (score > 0)$
- $Accuracy = \frac{1}{m} \sum_{i=1}^{m} (pred_i == Y_{val\ i}$

2.8 Application of Naive Bayes

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

2.9 Naive Bayes Assumptions

- Independence
- Relative frequency in corpus

2.10 Error Analysis

- Removing punctuation
- Removing words
- Words order
- Adversarial attacks

Chapter 3

Vector Space Models

3.1 Vector Space Models

- Know a word by the company it keeps

Application

- Information Extraction
- Machine Translation
- Chatbots

3.2 Word by Word and Word by Doc

Word by Word Design
Number of times they occur together within a certain distance

Word by Document Design
Number of times a word occurs within a certain category

Vector Spaces
Similarity between words/documents

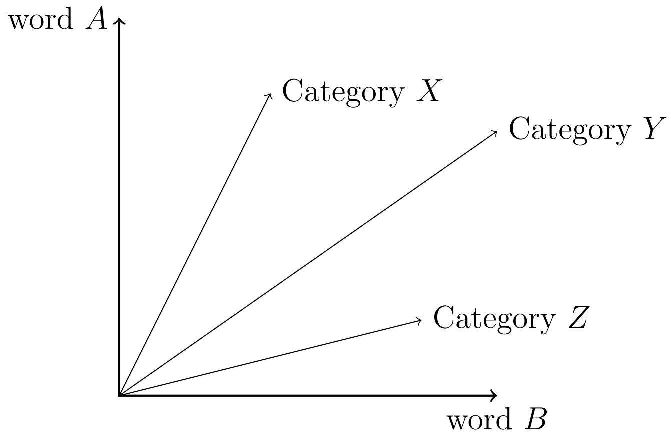


Figure 3.1: Vector Space

3.3 Euclidean Distance and Cosine Similarity

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

3.4 Cosine Similarirty

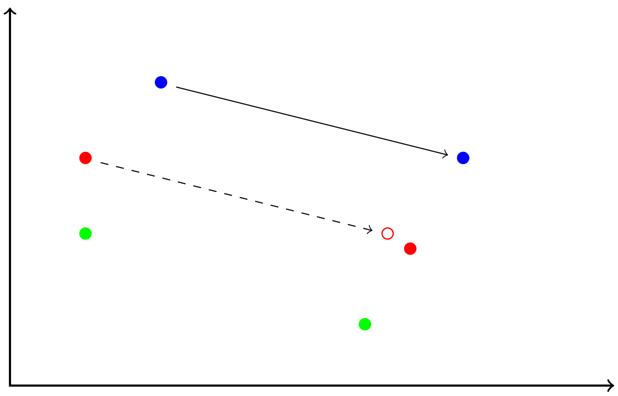
- Use cosine similarity when corpora are different sizes

$$\cos\beta = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}||\vec{w}|}$$

- $\beta = 90° \rightarrow \cos\beta = 0 \rightarrow$ dissimilar
- $\beta = 0° \rightarrow \cos\beta = 1 \rightarrow$ similar

3.5 Manipulating Words in Vector Space

Use Known Relationships to Make Prediction



3.6 Visualization and PCA

- Original Space $\rightarrow$ Uncorrelated features $\rightarrow$ Dimension reduction

3.7 PCA Algorithm

Eigenvector
Uncorrelated features for data gives the direction of uncorrelated features

Eigenvalue
The amount of information retained by each feature

Chapter 4

Machine Translation

4.1 Transforming Word Vectors

Transformation (using a matrix)

$$\begin{gathered} XR \approx Y \\ \searrow \;\; \swarrow \\ \text{subsets of the full vocabulary} \end{gathered}$$

Solving for R

- Initialize R
- In a loop,

$$\begin{aligned} &Loss = ||XR - Y||_F \\ &g = \frac{d}{dR} Loss \quad \text{(gradient)} \\ &R = R - \alpha g \quad \text{(update)} \\ &\qquad\quad \downarrow \\ &\qquad \text{learning rate} \end{aligned}$$

For a matrix A,

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

4.2 K-nearest Neighbors

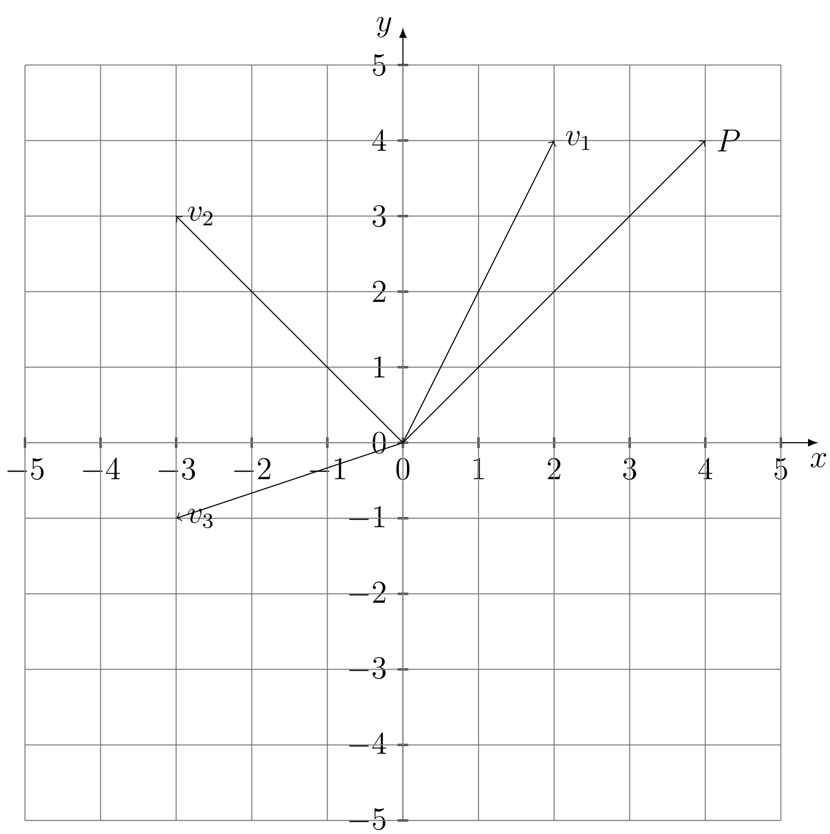
Translation
word $\xrightarrow{R}$ [transformed] $\xrightarrow[\textit{words}]{\text{find similar}}$ word′

4.3 Hash Tables and Hash Functions

Hash Function
A function that takes data of arbitrary sizes and maps it to a fixed value. The values returned are known as hash values or even hashes.

4.4 Locality Sensitive Hashing

To hash similar inputs into the same buckets with high probability



- $Pv_1^T > 0$
- $Pv_2^T = 0$

- $Pv_3^T < 0$

4.5 Multiple Planes

Use multiple planes to get a single hash value

Example

- $\vec{v}$, planes $\vec{P_1}, \vec{P_2}, \vec{P_3}$
- $hash = 2^0 \cdot h_1 + 2^1 \cdot h_2 + 2^2 \cdot h_3$
- $h_i = \begin{cases} 1 & if\ \vec{P_i} \cdot \vec{v} > 0 \\ 0 & else \end{cases}$

4.6 Approximate Nearest neighbors

- Does not give the full nearest neighbors
- Trade off accuracy for efficiency