Sentiment analysis part 1

Module 2

THE UNIVERSITY of ADELAIDE

Text representation

Sentence Representation

Bag-of-Words (BOW) Model, One-hot encoding

D1 - “I am very happy today”
D2 - “I am not well and not happy today”
D3 - “I wish I could go to play”

Unique list of words: I am feeling very happy today not well wish could go to play

Bag of Words:

	I	am	very	happy	today	not	well	and	wish	could	go	to	play
D1	1	1	1	1	1	0	0	0	0	0	0	0	0
D2	1	0	0	1	1	2	1	1	0	0	0	0	0
D3	2	0	0	0	0	0	0	0	1	1	1	1	1

One Hot

	I	am	very	happy	today	not	well	and	wish	could	go	to	play
D1	1	1	1	1	1	0	0	0	0	0	0	0	0
D2	1	0	0	0	1	1	1	1	0	0	0	0	0
D3	1	0	0	0	0	0	0	0	1	1	1	1	1

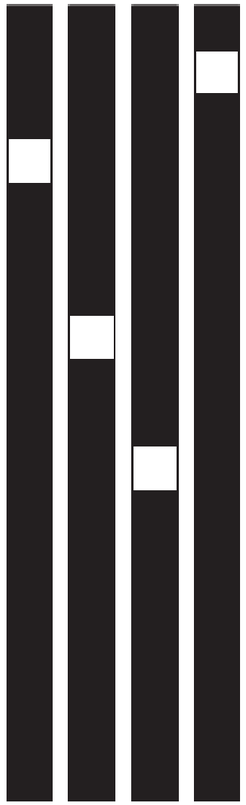
Text Representation: word n-gram

Example: "*Discussing things you care about can be difficult*"

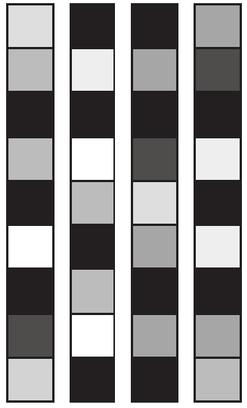
N-gram

- 2-gram (bigram): Discussing things, things you, you care, care about, about can, …
- 3-gram: Discussing things you, things you care,…

One-hot vs Word Embedding



One-hot word vectors:
- Sparse
- High-dimensional
- Hardcoded



Word embeddings:
- Dense
- Lower-dimensional
- Learned from data

Example here:

Source: *Deep Learning with Python* by Francois Chollet

Sub-Word

Word representation cannot handle unseen word or rare word well. Character-level representation is one of the solution to overcome out-of-vocabulary (OOV). However, it may too fine-grained any missing some important information.

Subword is in between word and character. It is not too fine-grained while able to handle unseen word and rare word.

- Byte Pair Encoding (BPE: Jurafsky et al. ch. 2.4.3)
- WordPiece: similar to BPE (Jurafsky, ch. 13.2.1)

THE UNIVERSITY of ADELAIDE

Byte Pair Encoding: BPE

- The algorithm begins with the set of symbols equal to the set of characters. Each word is represented as a sequence of characters plus a special end-of-word symbol.
- At each step of the algorithm, we count the number of symbol pairs, find the most frequent pair (‘A’, ‘B’), and replace it with the new merged symbol (‘AB’).
- We continue to count and merge, creating new longer and longer character strings, until we’ve done k merges; k is a parameter of the algorithm.

Text Pre-processing/normalisation

- Tokenization
- Remove stop words
- Lowercase/uppercase
- (Check misspelling)
- Lemmatization or Stemming

Tokenization

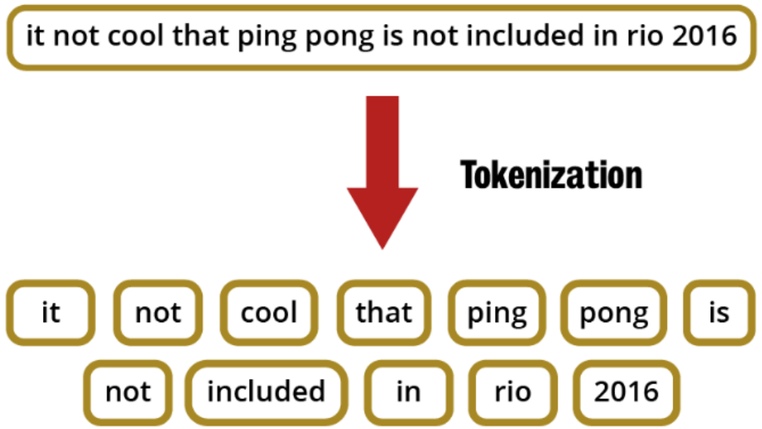


Figure 3.3: Sentence tokenisation 1.
Source: 01-training-tokenizers.ipynb

Problem: which punctuation marks should be kept and which removed?

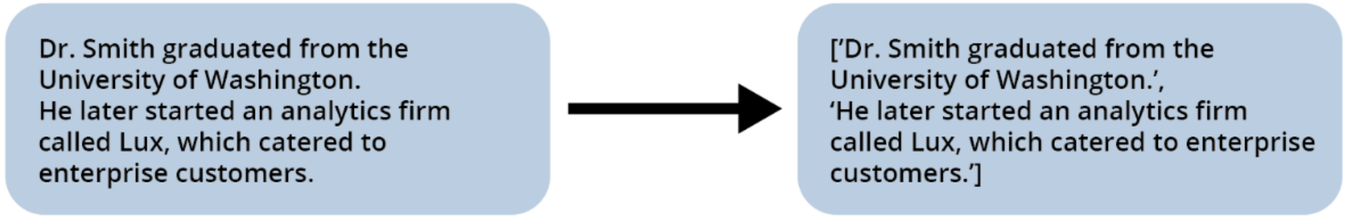
- Ph.D., AT&T?

Should the numericals be removed?

- (01/02/2006), $45.55, …

And URLs, Hashtags (#nlproc), etc.

Bottom line: domain specific treatment



THE UNIVERSITY of ADELAIDE

Stop words

```
>>> from nltk.corpus import stopwords
>>> stopwords.words('english')
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', 'your', 'yours',
'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', 'her', 'hers',
'herself', 'it', 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves',
'what', 'which', 'who', 'whom', 'this', 'that', 'these', 'those', 'am', 'is', 'are',
'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does',
'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until',
'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',
'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down',
'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here',
'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',
'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so',
'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', 'should', 'now']
```

Figure 3.2: NLTK toolbox stop words.

Should stop words be removed?
Some research advise against it.
You can test on a validation set

THE UNIVERSITY of ADELAIDE

Lemmatization & Stemming

- The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form.

Connect
Connections------> Connect
Connected-----> Connect
Connecting-----> Connect
Connection-----> Connect

Lemmatization

Reduce inflections or variant forms to base form

am, are, is → *be*

car, cars, car's, cars' → *car*

the boy's cars are different colors → *the boy car be different color*

Lemmatization: have to find correct dictionary of headword form

Stemming

Morphemes: The small meaningful units that make up words

Stems: The core meaning-bearing units

Affixes: Bits and pieces that adhere to stems

Stemming:

Reduce terms to their stems

Stemming is crude chopping of affixes

- language dependent
- e.g., ***automate(s), automatic, automation*** all reduced to ***automat***.

Porter's algorithm in NLTK

Stemming vs Lemmatization

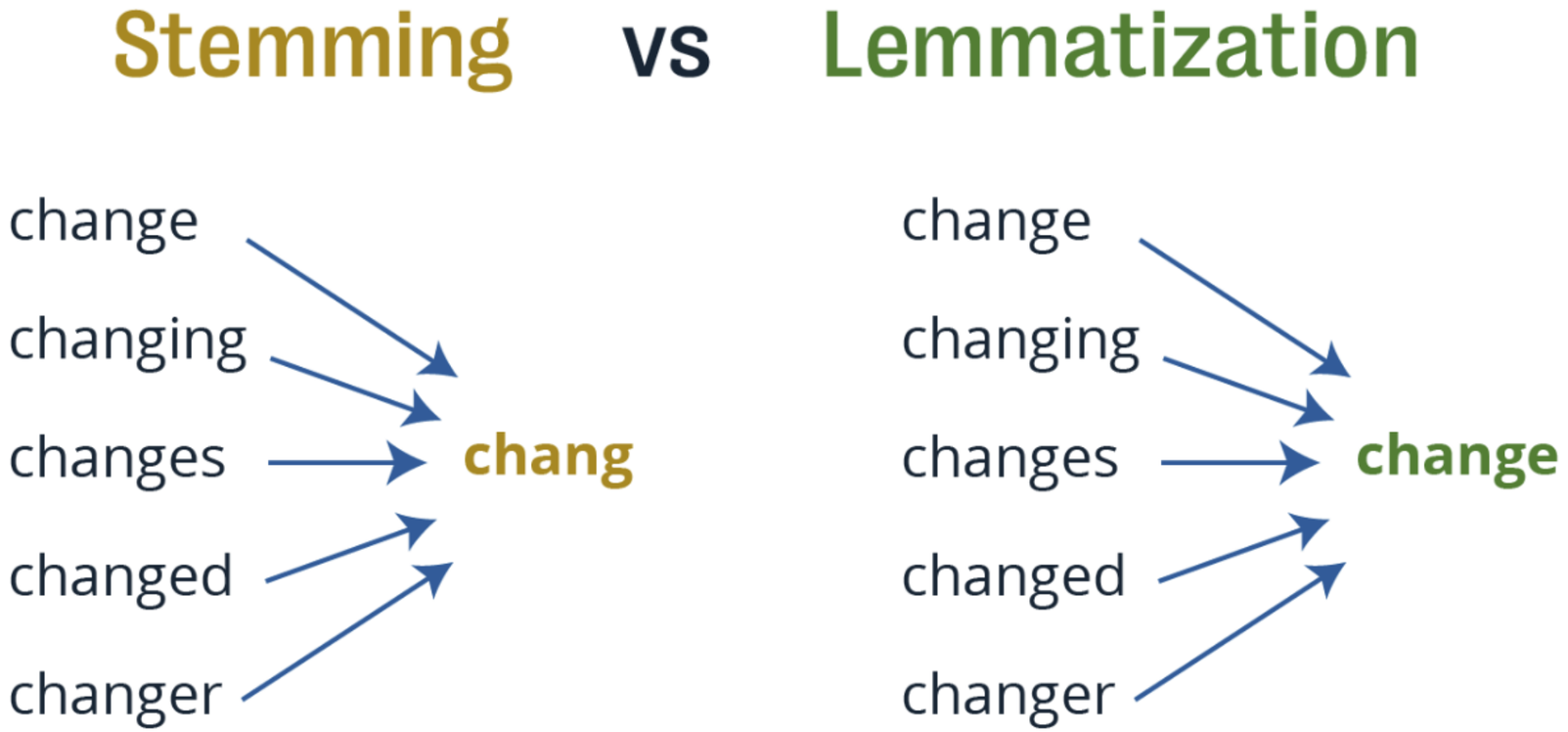


Figure 3.5: Stemming vs. Lemmatisation.
Source: Data Preprocessing

Another example?

Which one is better?
Which one is faster?

THE UNIVERSITY of ADELAIDE

Lemmatization in NLTK

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

# You will have to download the set of stop words the first time
nltk.download('stopwords')
 # Load stop words
stop_words=stopwords.words('english')
example_sent = "This is a sample sentence"
word_tokens = word_tokenize(example_sent)

wnl = WordNetLemmatizer()
wnl.lemmatize("sample")
```

'sample'

Stemming in NLTK

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import *

# You will have to download the set of stop words the first time
nltk.download('stopwords')
 # Load stop words
stop_words=stopwords.words('english')
example_sent = "This is a sample sentence"
word_tokens = word_tokenize(example_sent)

porter = PorterStemmer()
porter.stem("sample")
```

'sampl'

Naïve Bayes vs Multinomial Naïve Bayes

Recap from Online Learning:

probabilities

- Which of the following can be considered a probability distribution? E.g. probability distribution of word ***w*** in 5 documents.
  - A: [11, 22, 3, 4, 8]
  - B: [0.1, 0.5, 0.8, 0.6, 0.7]
  - C: [0.05, 0.2, 0.15, 0.1, 0.5]

Bayes Theorem for Classification

- Bayes theorem can be used to classify instances

$$\mathcal{D} = \{(\boldsymbol{X}_i, y_i) | i = 1,2, \dots, n\}$$

- Data of n instances, m classes *X* is vector of features, y_i is output class: $y_i \in (Y: \{y_1, \dots, y_j, \dots, y_m\})$

$$P(h|X) = \frac{P(X|h) \cdot P(h)}{P(X)} \quad h: y = y_j \text{ outcome to predict}$$

- We want *h* that maximises $P(h|X)$, maximum a posteriori hypothesis (MAP)

$$h_{MAP} = \arg\max_{h \in H} P(h|X) = \arg\max_{h \in H} \frac{P(X|h) \cdot P(h)}{P(X)} = \arg\max_{h \in H} P(X|h) \cdot P(h)$$

Bayes Theorem for Classification cont.

If we know or assume that all hypotheses *h* are equally probable, then we get the maximum likelihood hypothesis (ML)

$$h_{ML} = \arg\max_{h \in H} P(X|h)$$

In order to see how Bayes theorem can be used for classification, let's break down and simplify the formula. Notice that *P(X)* is independent of h, so it was omitted in MAP and ML formulas.

P(h) (prior)for each hypothesis (each class) can be calculated as $\frac{n_h}{n}$, where n_h is number of instances of training set classified into class y_j, and n is the number of instances in training set (which we use to estimate the probabilities)

From Bayes Theorem to Naïve Bayes

Now let's consider $P(X|h)$. In general, $P(X|h)$ is calculated using chain rule:

$P(x_1 \wedge x_2 \wedge \cdots \wedge x_m) = \prod_{i=1}^{m} P(x_i | \bigcap_{j=1}^{i-1} x_j)$.

In order to calculate this, we would need all possible combinations of X, which we do not have. In Naïve Bayes, we assume that x_i occur independently, so the formula is simplified:

$$P(x_1 \wedge x_2 \wedge \cdots \wedge x_m) = P(x_1) * P(x_2) * \cdots * P(x_m)$$

Therefore:

$$P(D = (x_1, x_2, \cdots, x_m)|h) = P(x_1|h) \cdot \cdots \cdot P(x_m|h) = \prod_{x_i \in D} P(x_i|h)$$

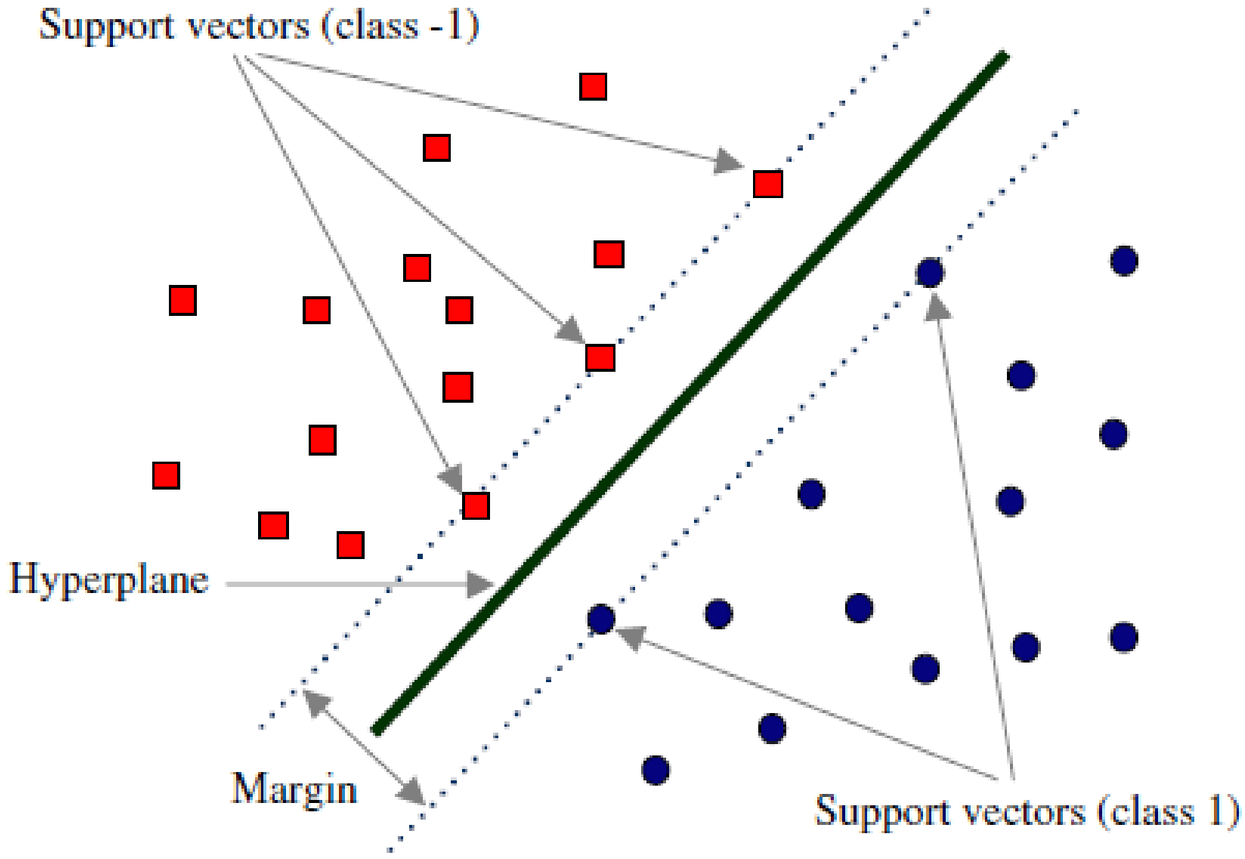
Although the independence assumption is often wrong, the Bayes classifier yields good results.

How to Classify Using Naïve Bayes

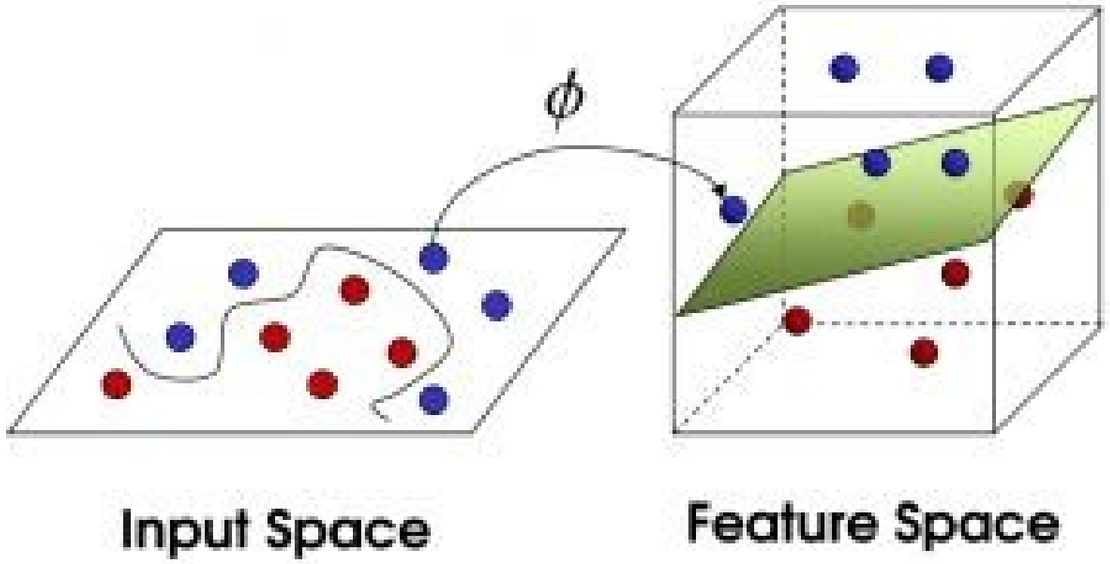
1. **Using training dataset:**
   a) For each class, calculate *P(h)* as $\frac{n_h}{n}$, where *n* is number of instances, n_h is number of instances in class *y* for hypothesis *h*
   b) For each feature in training set, calculate $P(x_j|h)$ as $\frac{n_j}{n_h}$, where n_h is number of instances in class *h*, n_j is number of instances in class *h* with feature *j*
   c) Store the above calculations for later
2. **Given test instance with features** : $\{x_1, \dots, x_j, \dots\}$**:**
   a) Retrieve all *P(h)* and all $P(x_j|h)$ for these features
   b) Calculate $P(X|h) \cdot P(h) = P(h) * P(x_1|h) * \cdots * P(x_j|h) * \cdots$ for all features *x* in the test instance, and all hypotheses *h*, corresponding to all classes. So we will have a vector of *m* values, one for each class
   c) Take arg max of that vector, as the predicted class

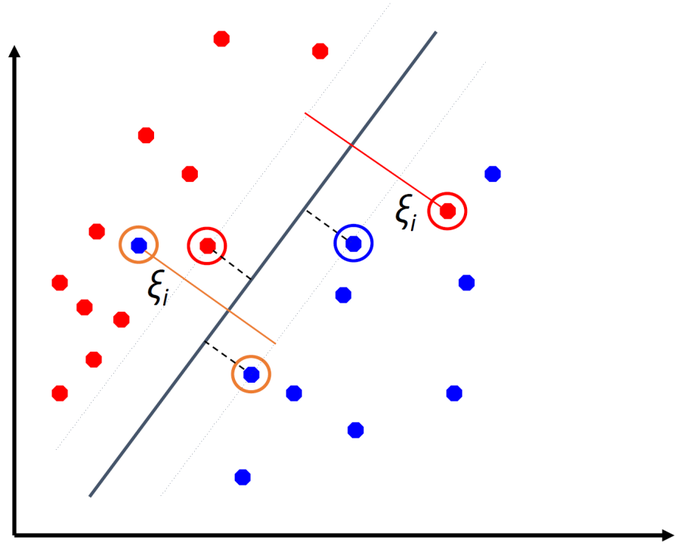
Support Vector Machine (SVM)

SVM is another “classical” method performing well on text



Support Vector Machine



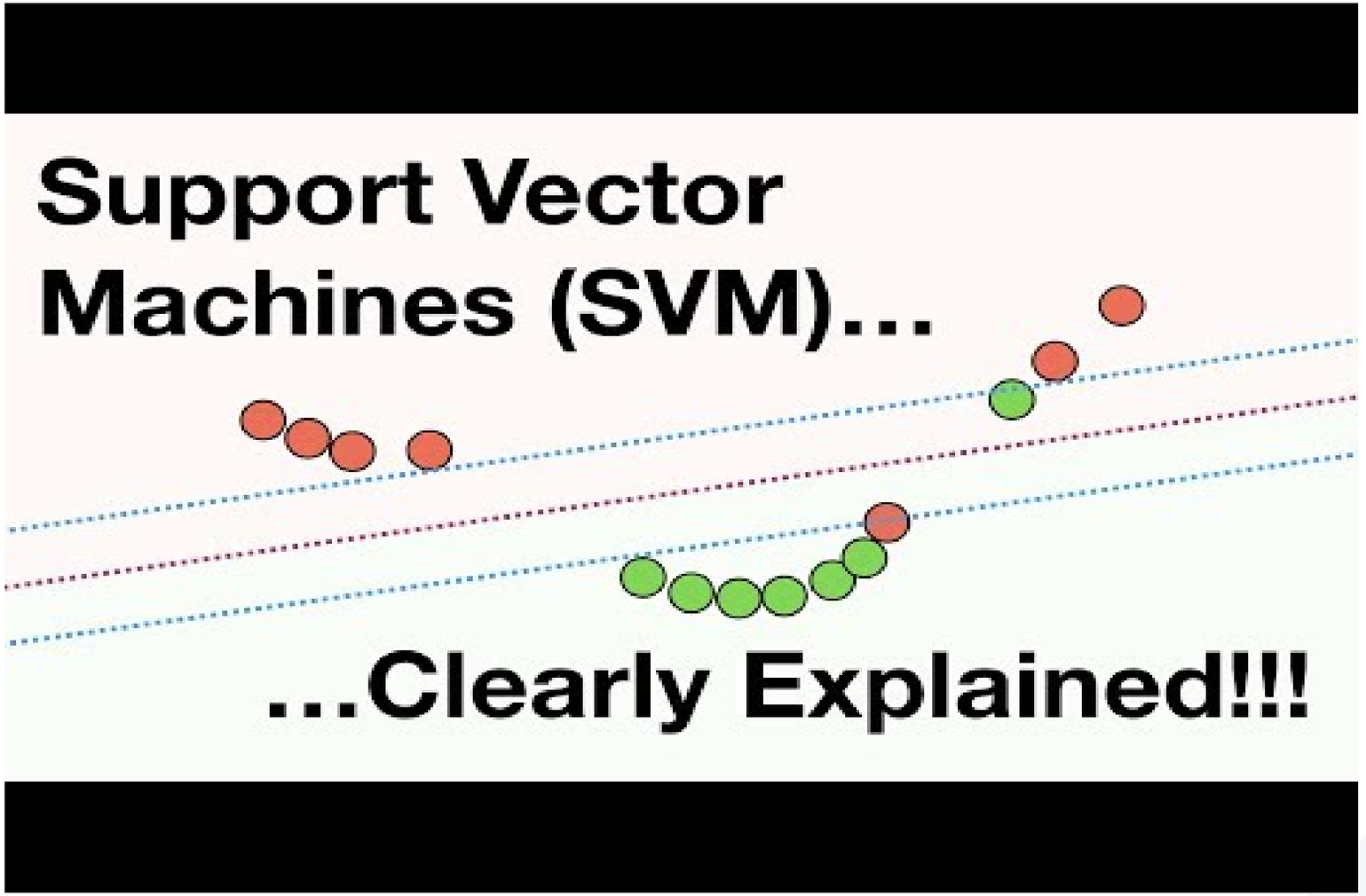


Instances may not be linearly separable. Transform into a higher dimension space but calculate dot product in original space. This is called kernel trick

Sometimes it is better to allow some instances cross be present in the margin space or even cross the boundary
This is called soft-margin SVM

THE UNIVERSITY of ADELAIDE

This video explains soft margin and kernel trick



https://www.youtube.com/watch?v=efR1C6CvhmE

Exercise on probability

A patient takes a lab test, and the result comes back as positive. The test returns a correct positive result (+) in only 98% of the cases in which the disease is actually present, and a correct negative result (-) in only 97% of the cases in which the disease is not present. Furthermore, 0.008 of the entire population have this cancer.

P(cancer) = 0.008, P(not cancer) = 0.992, these are called prior probabilities

P(+ | cancer) = 0.98 P(- | cancer) = 0.02

P(+ | not cancer) = 0.03 P(- | not cancer) = 0.97

Does the patient have cancer or not? To answer this, calculate P1 = P(not cancer | +)and P2 = P(cancer | +), and then calculate likelihood ratio P1/P2. Then select the right answer.

Sentiment Lexicons

Sentiment terminology

- Sentiment
  - How sentiment is related to emotions?
- Aspect-based sentiment analysis
- Stance?
- How sarcasm affects sentiment analysis?

Extending Sentiment Analysis to Affective Computation

- Affective Computation: computational analysis of emotion, sentiment, personality, mood, and attitudes.
- Affects: figure from Jurafsky p. 496

Emotion: Relatively brief episode of response to the evaluation of an external or internal event as being of major significance.
(*angry, sad, joyful, fearful, ashamed, proud, elated, desperate*)

Mood: Diffuse affect state, most pronounced as change in subjective feeling, of low intensity but relatively long duration, often without apparent cause.
(*cheerful, gloomy, irritable, listless, depressed, buoyant*)

Interpersonal stance: Affective stance taken toward another person in a specific interaction, coloring the interpersonal exchange in that situation.
(*distant, cold, warm, supportive, contemptuous, friendly*)

Attitude: Relatively enduring, affectively colored beliefs, preferences, and predispositions towards objects or persons.
(*liking, loving, hating, valuing, desiring*)

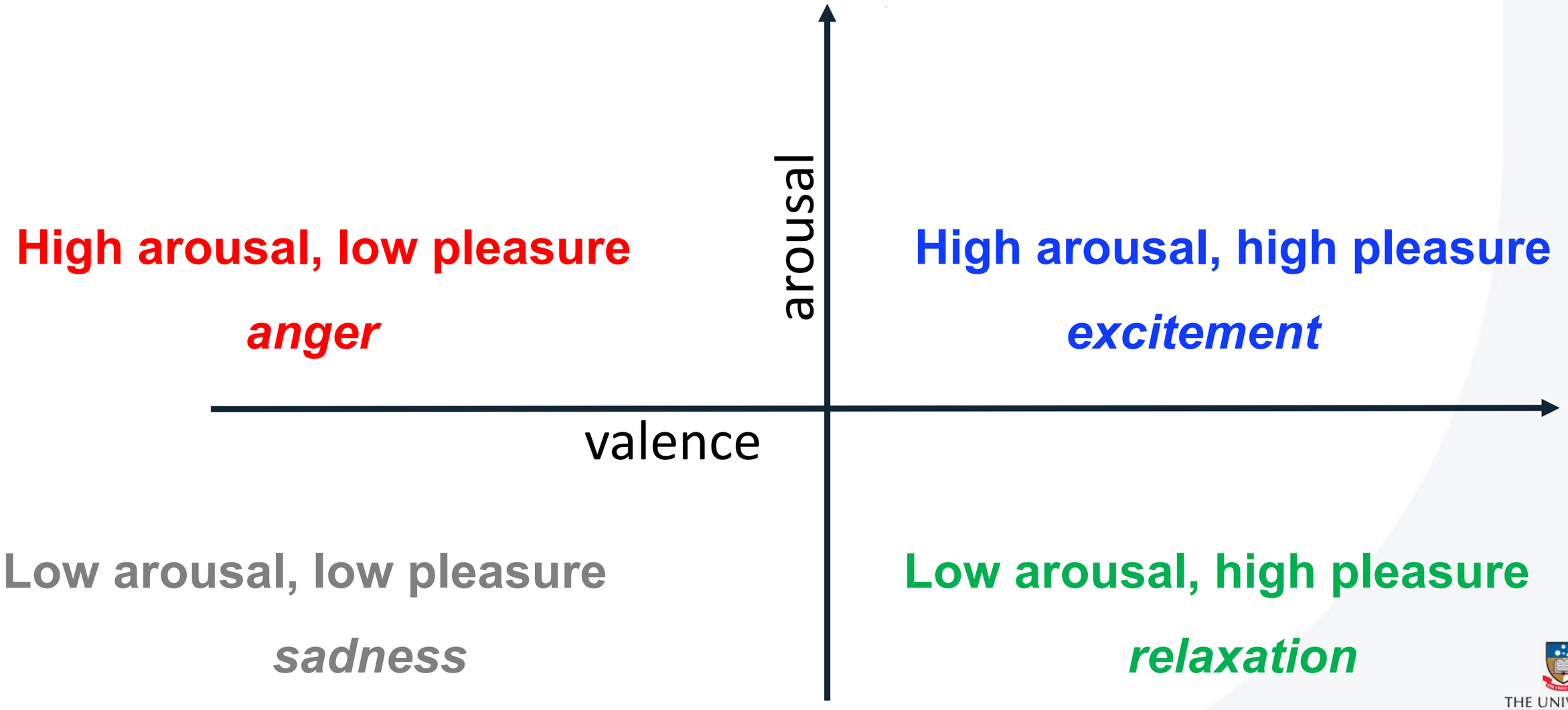
Personality traits: Emotionally laden, stable personality dispositions and behavior tendencies, typical for a person.
(*nervous, anxious, reckless, morose, hostile, jealous*)

Figure 25.1 The Scherer typology of affective states (Scherer, 2000).

Sentiment Lexicons

- Lexicon: list of words with some meaning or class. Usually hand-built.
- Sentiment, or affective lexicons: list of words with affective meaning or (aka: connotation).
- 8 basic emotions in four opposing pairs:
  - joy–sadness
  - anger–fear
  - trust–disgust
  - anticipation–surprise
- Two more terms:
  - valence: the pleasantness of the stimulus
  - arousal: the intensity of emotion provoked by the stimulus

Valence/Arousal Dimensions



Jurafsky https://web.stanford.edu/~jurafsky/slp3/slides/20_Affect_Jan_18_2021.pptx

THE UNIVERSITY of ADELAIDE

Popular Sentiment Lexicons

- MPQA (Multi-Perspective Question Answering), e.g.

type=weaksubj len=1 word1=**able** pos1=adj stemmed1=n priorpolarity=positive
type=weaksubj len=1 word1=**abnormal** pos1=adj stemmed1=n priorpolarity=negative
type=strongsubj len=1 word1=**absurd** pos1=adj stemmed1=n priorpolarity=negative

  - How to use it?

    Calculate a score for each word you find in the text: combine type with priorpolarity and if the word is stemmed or not.

Popular Sentiment Lexicons

- NRC (National Research Council Canada), e.g.

```
abandon   anticipation      0
abandon   disgust     0
abandon   fear  1
abandon   joy   0
abandon   negative    1
abandon   positive    0
abandon   sadness     1
abandon   surprise    0
abandon   trust 0
```

  - How to use it?

Calculate a score for each word you find in the text taking into account how many positive or negative emotions it triggers. Or just use positive/negative label.

Popular Sentiment Lexicons

- VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media, and works well on texts from other domains.

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
# analyse with VADER
analyser = SentimentIntensityAnalyzer()
for text in text_data:
    score = analyser.polarity_scores(text)
    if score['compound'] >= 0.05:
        print(text+": "+"VADER positive")
    elif score['compound'] <= -0.05:
        print(text+": "+"VADER negative")
    else:
        print(text+": "+"VADER neutral")
```

Popular Sentiment Lexicons

- LIWC (Linguistic Inquiry and Word Count)
  - 73 lexicons, over 2300 words
  - Commercial product
  - Operator manual can be found here http://downloads.liwc.net.s3.amazonaws.com/LIWC2015_OperatorManual.pdf

Positive Emotion	Negative Emotion	Insight	Inhibition	Family	Negate
appreciat*	anger*	aware*	avoid*	brother*	aren't
comfort*	bore*	believe	careful*	cousin*	cannot
great	cry	decid*	hesitat*	daughter*	didn't
happy	despair*	feel	limit*	family	neither
interest	fail*	figur*	oppos*	father*	never
joy*	fear	know	prevent*	grandf*	no
perfect*	griev*	knew	reluctan*	grandm*	nobod*
please*	hate*	means	safe*	husband	none
safe*	panic*	notice*	stop	mom	nor
terrific	suffers	recogni*	stubborn*	mother	nothing
value	terrify	sense	wait	niece*	nowhere
wow*	violent*	think	wary	wife	without

Figure 25.6 Samples from 5 of the 73 lexical categories in LIWC (Pennebaker et al., 2007). The * means the previous letters are a word prefix and all words with that prefix are included in the category.

How Sentiment Lexicons are Created

- Human annotation for each affective word
- Machine annotation:
  - Large corpus of texts, each with polarity (positive/negative)
  - For each word frequently occurring in texts, assign a score based on counts of how many times the word occurs in pos/neg texts.

THE UNIVERSITY
of ADELAIDE