Text Classification

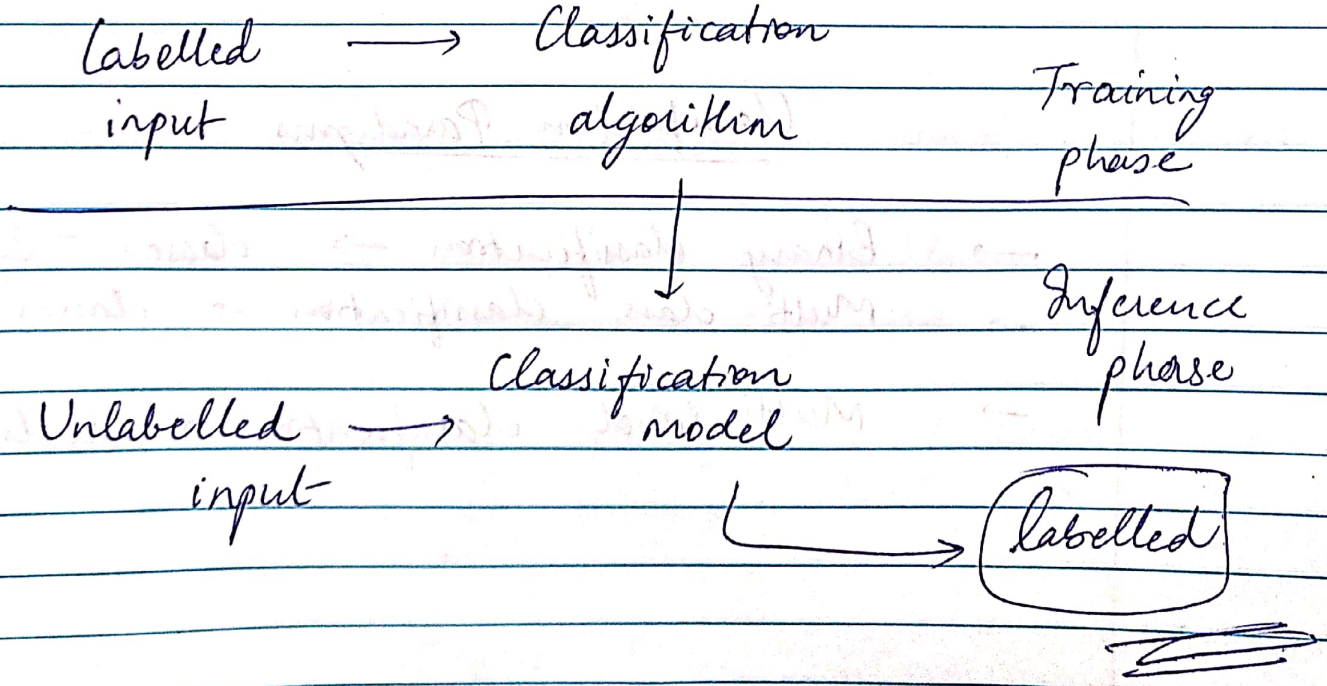
sentence level

- News article → politics, sports, tech
- Spam → yes or no.
- Sentiment analysis → +ve / -ve

Spelling correction → at word level

All above are supervised learning tasks.

Supervised Classification



→ learn from features/properties.

→ Features have importances.

$\{x_1, x_2, ---- x_n\}$

Class labels $y \{y_1, y_2 \dots y_n\}$

Supervised Classification

Labeled data set → Training data

→ Validation data (hold out data)

Training phase

Unlabeled data set → test data

Inference phase

Classification Paradigms

→ Binary classification → classes = 2

→ Multi-class classification → classes > 2

→ Multi label classification → labels ≥ 2

Identifying features from Text

→ Words → most common class of features
→ Stopwords
→ Normalisation → make lower case (vs) leave as is
→ Stemming / lemmatization
→ Characteristics of words : capitalization
→ POS
→ Grammatical structure, sentence parsing
→ Group words of similar meaning, semantics
Ex buy ; purchase
Ex Mr Ms Mrs Dr Prof
Number / digit
dates.
→ Inside the words, word sequences.
Bigrams → White House.
trigrams, n-grams.
→ character sub sequences of words.
Ex -ing, -ion, etc

Naive Bayes Classifier

Classifying text search queries
→ ~~Politics~~, CSE, Zoology, Entertainment

query = "python"

→ Python → snake → zoology

→ python → programming → CSE

→ python → Monty Python → Entertainment

Most common class → python → zoology.

query = "python download" → CSE.

probabilistic model → likelihood of class.

Bayes' rule

$$\text{posterior prob} = \frac{\text{prior prob} \times \text{likelihood}}{\text{evidence}}$$

$$P(y|x) = \frac{P(y) \times P(x|y)}{P(x)}$$

Naive Bayes

$$P(y=CS|\text{python}) = \frac{P(y=CS) \times P(\text{python}|y=CS)}{P(\text{python})}$$

$$pr(y|x) = \frac{p(y) \times pr(x|y)}{p(x)}.$$

$$y^* = \text{argmax } pr(y|x)$$

Assumption → given the class label, the features are independent of each other.

$$y^* = \text{argmax } P_r(y|x) = \text{argmax } P(y) \times \prod_{i=1}^{n} P(x_i|y)$$

Parameters → prior probabilities → $pr(y)$.
→ likelihood → $pr(x_i|y)$ for all features.

603

$|y| = 3$ $\quad x \rightarrow n = 100.$ binary class

→ $p(y) = 3$ for every y.

→ $p_r(x_i|y) = 100 + 100 + 100. = 300.$ for class 1.

$pr(x_i|y) = 100 + 100 + 100 = 300$ for class 0

Prior probability calculation.

→ count of number of instances in each class

$pr(y) = n/N$.

Likelihood calculation.

$p(x_i|y)$ for x_i and y_i in Y.

count x_i appears in instances labeled as y

p instances of class y.

x_i appears k times in class y.

$p(x_i|y) = k/p$.

Naive Bayes → Smoothing

→ What if $pr(x_i|y) = 0$?

If. x_i never occurs in y.

⇒ posterior prob will become 0.

(Π product).

⇒ Smooth parameters.

Laplace smoothing or additive smoothing

add count + 1 → dummy count

$$p(x_i|y) = \frac{k+1}{p+n}$$

↳ number of features.

Naive Bayes is a probabilistic model

↓

features are independent of each other, given the class label.

Ex White House

These 2 features are not independent

⇒ Naive

Naive Bayes → baseline strong for text classification.

Naive Bayes Variations

- Multinomial Naive Bayes
  - ↳ follows a multinomial distribution
  - ↳ Each feature value is a count

TFIDF, TF.

- Bernoulli multinomial distribution
  - ↳ follows multivariate Bernoulli distribution
  - ↳ feature is binary

Text → Multinomial NB

Support Vector Machines

Classifier = function on input data

$f(\begin{matrix}- & - & - \\ - & - & -\end{matrix}) \longrightarrow$ class 1 / 2 / 3

Decision Boundaries

Classification function is dependent on decision boundaries.

Data overfitting → fits well on train data not on test.

Linear boundaries ✓ → straight line
- → easy to find
- → easy to evaluate
- → more generalizable

simple models ↗ [Occam's razor]

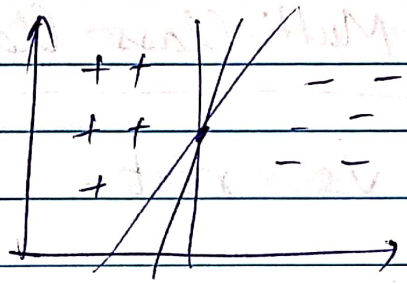
Finding a linear Boundary

→ Perceptron

→ LDA

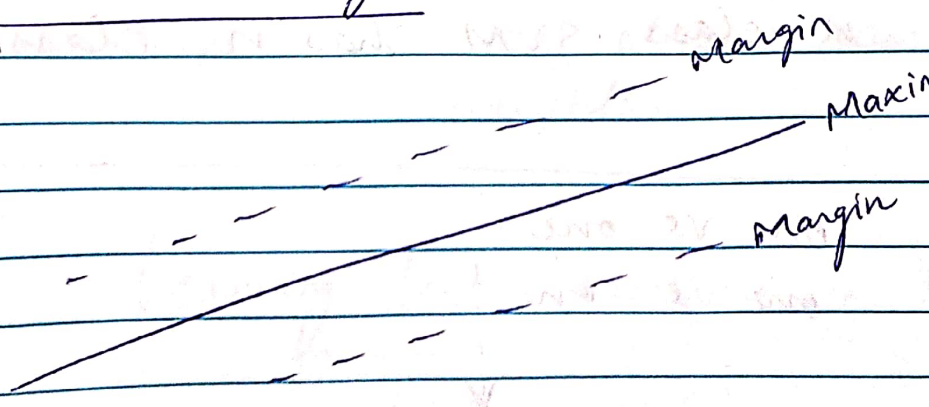
→ Linear least squares.

problem if linearly separable, infinite number of lines



Which boundary is reasonable?

Maximum Margin



SVM are maximum margin classifiers.

SVMs are linear classifiers that find a hyperplane

$y_i = f(x_i) = w \cdot x_i + b$

- y_i: train output
- x_i: train data
- w: weight vector
- $\cdot$: dot product

Binary

$y_i = f(x_i) \geq 0 \rightarrow 1$

$y_i = f(x_i) < 0 \rightarrow -1$

SVM : Multi Class Classification

✓ one vs rest. — binary

one vs rest — binary

⋮

so on.

n-class SVM has n classifiers.

---

one vs. one

one vs. one

⋮

n-class SVM has ${}^nC_2 \rightarrow \frac{n(n-1)}{2}$

Parameter C → Regularization.

How much importance should we give to individual data points.

C ↑ regularization ↓

fit training data as well as possible, every data point is important.

C ↓ regularization ↑

more tolerant to errors on training data points.

linear kernels work best for text data.

Polynomial, radial, -----

one vs rest better than one vs one

class_weight → diff classes can have different weights.

Learning Text Classifiers in Python

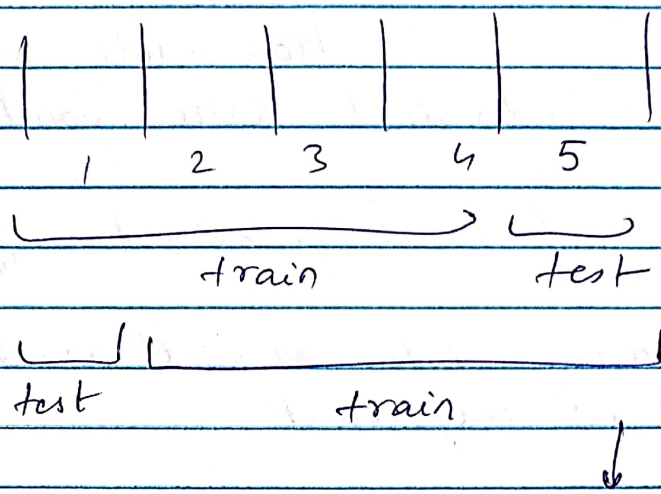
→ sklearn MultinomialNB()

→ sklearn SVC

Model Selection

→ sklearn train_test_split

model selection sklearn Cross validation



so on.

NLTK

- Naive Bayes Classifier
- SKlearn classifier

↳ calls sklearn algorithms using NLTK.