Naive Bayes and text classification

Gustave Cortal

Summary

Naive Bayes classifier

Text classification: definition

Input

a document d
a fixed set of classes $C = c_1, c_2, ..., c_J$

Output

a predicted class $c \in C$

Tasks

Spam detection
Author profiling
Sentiment analysis
...

Supervised machine learning

Input

documents $d_1, d_2, ..., d_m$
a fixed set of classes $C = c_1, c_2, ..., c_J$
a training set of m hand-labeled documents $(d_1, c_1), ..., (d_m, c_m)$

Output

a learned classifier $\gamma : d \rightarrow c$

Methods

Naïve Bayes
Logistic Regression
Support-Vector Machines
k-Nearest Neighbors
...

Naive Bayes classifier

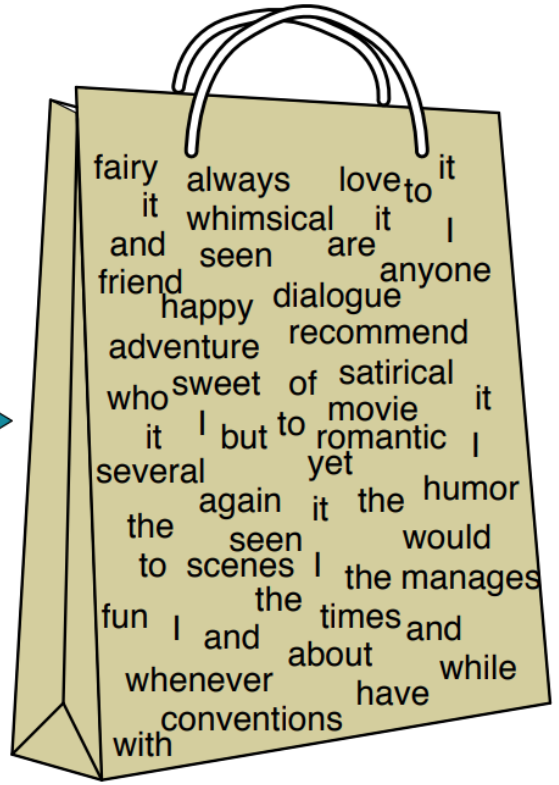
A simple classification method based on **Bayes' rule** and a *Bag-of-Words representation* of the text

Token	*Doc 1*	*Doc 2*	*Doc 3*	*Doc 4*
`cat`	1	0	2	1
`dog`	0	3	1	0
`fish`	0	1	0	1
`mouse`	2	0	0	0
`cheese`	0	0	1	3

Table: Example of a Bag-of-Words representation

Bag-of-Words representation

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
…	…

Bayes' rule applied to documents

For a document d and a class c:

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

$P(d|c)$ is the **likelihood**
$P(c)$ is the **prior**
We drop the denominator $P(d)$

The classifier selects the most likely class:

$$\hat{c} = \text{argmax}_{c \in C} P(c|d)$$

$\rightarrow$ Naive Bayes is a **generative** model!

Assumptions

$P(d|c)$ is the **likelihood**
Document d is represented as n features $x_1, x_2, ..., x_n$

We rely on the **Bag-of-Words assumption** and the **conditional independence**: word positions do not matter and probabilities $P(x_i|c)$ are independent given c

Hence, we have:

$$P(d|c) = P(x_1, x_2, ..., x_n|c) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c)$$

$$\begin{aligned}
\hat{c} &= \text{argmax}_{c \in C} P(c|d) \\
&= \text{argmax}_{c \in C} P(d|c)P(c) \\
&= \text{argmax}_{c \in C} P(x_1, x_2, ..., x_n|c)P(c) \\
&= \text{argmax}_{c \in C} P(c) \prod_{x \in X} P(x|c)
\end{aligned}$$

Learning the Naive Bayes classifier

We use the **Maximum Likelihood Estimate**, relying on the counts of words in the training set

$$\hat{P}(c_j) = \frac{N_{c_j}}{N_{\text{total}}}$$

$$\hat{P}(w_i|c_j) = \frac{\text{count}(w_i, c_j)}{\sum_{w \in V} \text{count}(w, c_j)}$$

Handling probabilities

Multiplying probabilities can cause **underflow**. To address this, we use the **logarithmic space**: $\log(ab) = \log(a) + \log(b)$

$$\hat{c} = \text{argmax}_{c \in C} P(c) \prod_{x \in X} P(x|c)$$

$$\hat{c} = \text{argmax}_{c \in C} \log P(c) + \sum_{x \in X} \log P(x|c)$$

Unknown words

How do you handle **unknown words** that appear in the test set but not in the training set? We can remove them from the test set or apply Laplace smoothing:

$$\hat{P}(w_i|c_j) = \frac{\text{count}(w_i, c_j) + 1}{\sum_{w \in V}(\text{count}(w, c_j) + 1)}$$
$$= \frac{\text{count}(w_i, c_j) + 1}{\sum_{w \in V} \text{count}(w, c_j) + |V|}$$

Stop words

Stop words are very frequent words such as `the` and `a`

They are often removed from both training and test sets to reduce noise and hopefully improve model performance

You can use a predefined list or make your stopword list by filtering the most frequent words of your dataset

Example

Set	*Class*	*Documents*
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

Calculate priors and likelihoods to determine whether the test set is positive or negative

Evaluation metrics

Evaluation metrics

Confusion matrix:

	System positive	*System negative*
Gold positive	True Positives	False Negatives
Gold negative	False Positives	True Negatives

Precision measures how many predicted positives are correct:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Recall measures how many gold positives were correctly identified:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

F1-score is the harmonic mean of precision and recall:

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Confusion matrix: example

Class 1: Urgent

	true urgent	true not
system urgent	8	11
system not	8	340

$$\text{precision} = \frac{8}{8+11} = .42$$

Class 2: Normal

	true normal	true not
system normal	60	55
system not	40	212

$$\text{precision} = \frac{60}{60+55} = .52$$

Class 3: Spam

	true spam	true not
system spam	200	33
system not	51	83

$$\text{precision} = \frac{200}{200+33} = .86$$

Pooled

	true yes	true no
system yes	268	99
system no	99	635

$$\text{microaverage precision} = \frac{268}{268+99} = \mathbf{.73}$$

$$\text{macroaverage precision} = \frac{.42+.52+.86}{3} = \mathbf{.60}$$

Micro and macro average for multi-class problems

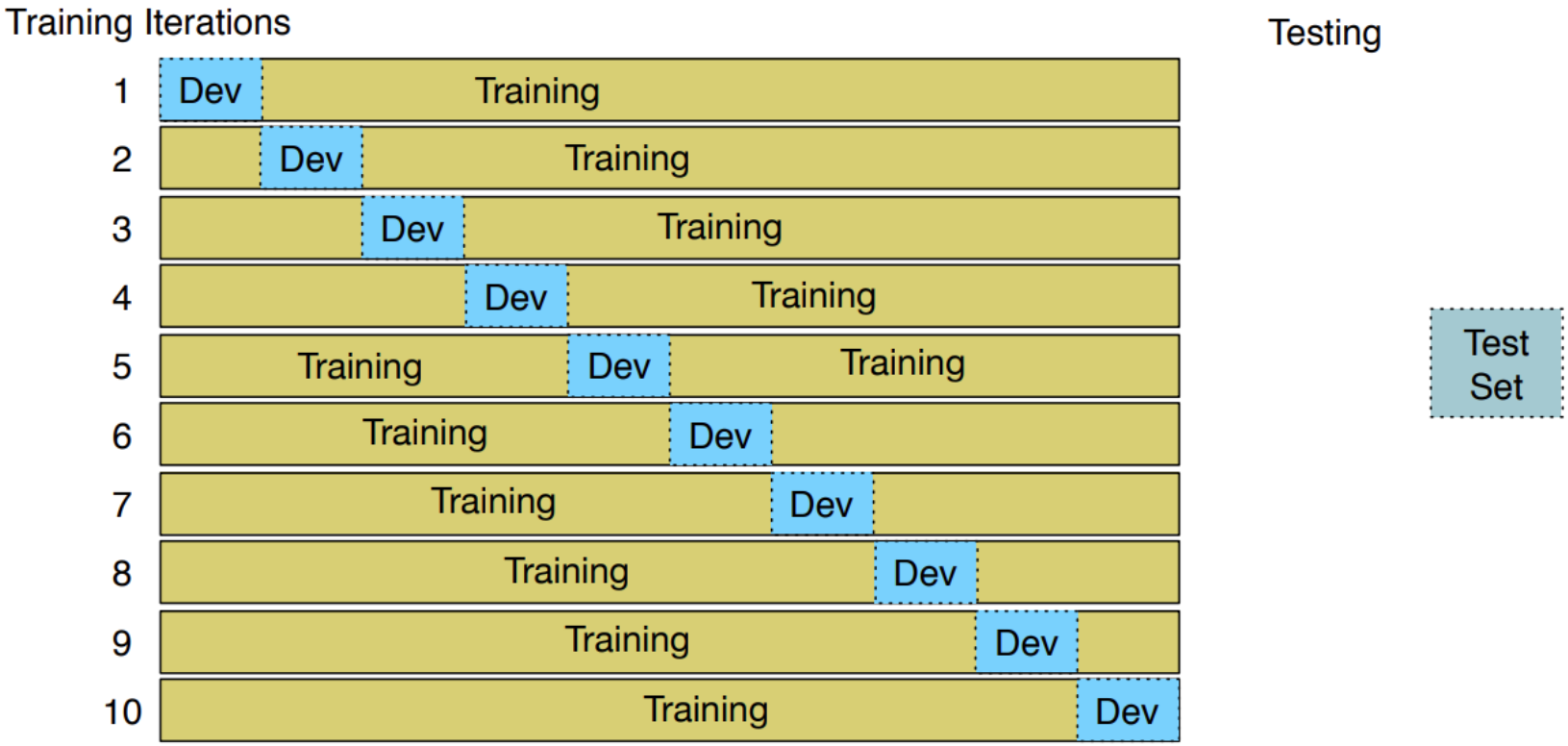
Micro average

Aggregate TP, TN, FP, FN across all classes, then calculate metrics. This gives equal weight to each classification, the results are dominated by the most frequent classes

Macro average

Calculate metrics for each class independently, then take the average. This gives equal weight to each class. You should use macro-average when all classes are equally important

10-fold cross-validation



Why using cross-validation

Each data point is used for both training and validation $\rightarrow$ efficient when the dataset is small

Having multiple performance estimates (one per fold) allows quantification of the variance in model performance $\rightarrow$ if the scores differ greatly across folds, you know the performance is sensitive to specific training sets or might depend heavily on a few samples

Trying different model configurations and comparing their average scores helps you find the configuration that generalizes best rather than one that is simply tuned to a single train/test split

Exercices

Exercices

- Continue implementing n-gram language models
- Implement Naive Bayes classifier from scratch and using scikit-learn
- Implement evaluation metrics (precision, recall, F1-score) and cross-validation from scratch and using scikit-learn
- Form groups for the project and discuss possible datasets