UiO : University of Oslo

# Classification for Information Retrieval

Pierre Lison
University of Oslo, Dep. of Informatics

*INF3800: Søketeknologi*
April 9, 2014



## Outline of the lecture

- The classification task
- Naive Bayes
- Feature selection
- Evaluation of classifiers
- Conclusion

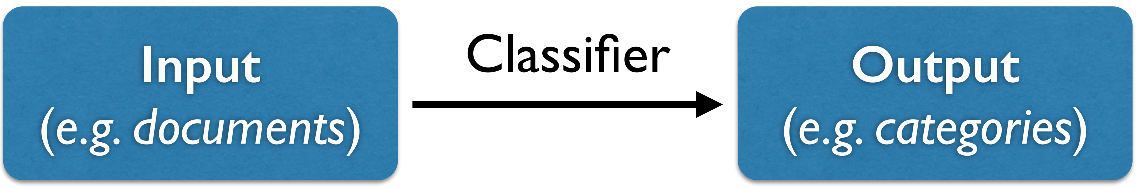
# Outline of the lecture

- **The classification task**
- Naive Bayes
- Feature selection
- Evaluation of classifiers
- Conclusion





# What is classification?




- **Classification** is the process of analyzing a particular *input* and assigning it to (one or more) category
- The set of possible categories is *discrete* and *finite*
  - If the output is continuous, the task is a *regression* problem
  - Classification may be *one-of* (exactly one category allowed per input) or *any-of* (multiple categories allowed)

# What is classification?

**Input** (*e.g. documents*) → Classifier → **Output** (*e.g. categories*)

- **Classification** is one of the core areas in *machine learning*
  - Numerous applications, in computer vision, speech & language processing, bioinformatics, data mining etc.
- Within **information retrieval,** classification is used in various subtasks of the search pipeline
  - Preprocessing, content filtering, sorting, ranking, etc.





# Examples of classification

| Tasks | Input | Output |
|---|---|---|
| **Spam filtering** | an email | Spam or not spam |
| **Sentiment detection** | A product review | Positive or negative |
| **Topic classification** | A document | A set of topics for the document |
| **Language identification** | A document | The language(s) used in the document |
| **Truecasing** | A word | true/false (should the word be capitalized or not) |
| **Retrieval of standing queries** | A document | true/false (does the document match the query) |

# Classification approaches

| | Manual classification | Rule-based classification (based on hand-crafted rules) | Statistical classification (based on training data) |
|---|---|---|---|
| + | High-quality | Can encode complex decision strategies | Robust, adaptive, scalable |
| - | Slow, expensive | Need domain expertise | Need training data! |

Focus in this course





# Formalisation

- Space $\mathbb{X}$ of possible inputs
- Set $\mathbb{C} = \{c_1, c_2, \ldots c_J\}$ of possible classes or categories
- Classifier $\gamma$ maps inputs to categories:

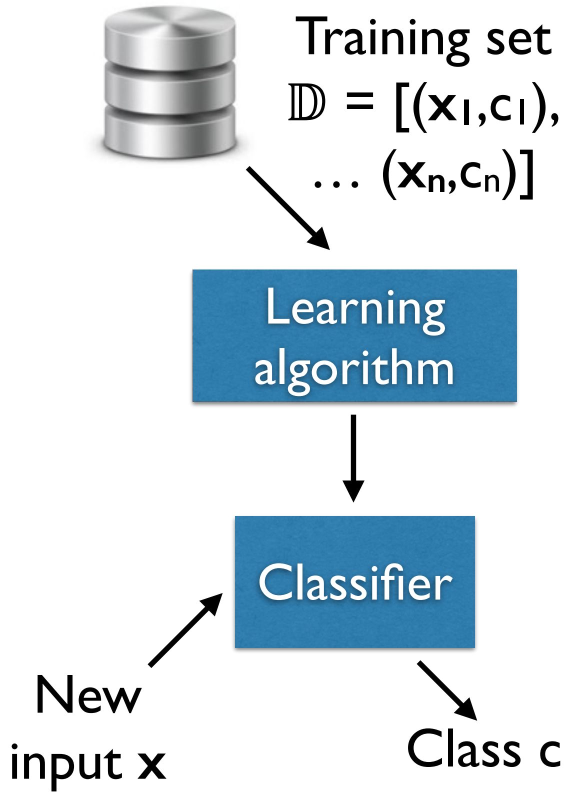
$$\gamma : \mathbb{X} \rightarrow \mathbb{C}$$

**Goal**: estimate this classifier with a training set $\mathbb{D}$ composed of $n$ examples $\{(\mathbf{x}_i, c_i) : 1 \leq i \leq n\}$

where $\mathbf{x}_i$ is the $i$th example and $c_i$ its category

# Formalisation

- This type of learning procedure is an instance of *supervised learning*
  - The classifier is estimated on the basis of training examples annotated by a "supervisor"
  - The result of the learning process is a particular classifier
  - **Goal**: find a classifier that achieves high accuracy on *new data*








# Formalisation

- The inputs $\mathbf{x} \in \mathbb{X}$ are often represented as feature vectors:
  - Each vector position corresponds to a particular feature, with a discrete or continuous range
  - For instance, a "bag-of-words" representation of a document may be encoded as a vector

$[\, w_1,\ w_2,\ w_3 \dots, w_N]^T$ where $w_i$ is the number of occurrences of the term i

- The categories $c \in \mathbb{C}$ must be discrete labels

# Outline of the lecture

- The classification task
- **Naive Bayes**
- Feature selection
- Evaluation of classifiers
- Conclusion





# Naive Bayes

- Numerous supervised learning algorithms:
  - Naive Bayes, decision trees, logistic regression, neural networks, k-nearest neighbor, support vector machines, etc.
- In this and the next two lectures, we will examine some of the algorithms, with a particular focus on their use for IE tasks
- We start with a simple but powerful probabilistic approach: **Naive Bayes**

# Naive Bayes classification

- Assume you have a set of documents which can be grouped in a set of categories $\{c_1,\ldots c_J\}$
  - The classes can e.g. correspond to document topics
  - (To keep things simple, we only consider *one-of* classification)
- You want to build a classifier which will assign each document $d_i$ to its most likely class
- To this end, you are given a training set of manually classified documents: $\{(d_i,c_i), 1 < i < n\}$





# Naive Bayes classification

- The classification of a document d is a search for the class c* such that

$$c^* = \underset{c}{\operatorname{argmax}}\, P(c|d)$$

- But the probability P(c|d) is hard to determine!
- Using Bayes rule, we can rewrite the probability:

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

*prior* of the class

normalisation factor (often ignored)

*posterior* probability of the class given the document

*likelihood* of the document given the class

# Naive Bayes classification

- We further simplify the problem by representing the document as a bag of words $d_i = \{w_1, w_2, \ldots w_n\}$
- ... and assuming that the words are conditionally independent from each other (*naive* Bayes):

$$P(d|c) = P(w_1, ...w_n|c) \approx P(w_1|c)...P(w_n|c)$$

- We can then rewrite our probability:

$$P(c|d) \propto \underbrace{P(w_1|c)...P(w_n|c)}_{\text{likelihood}} \underbrace{P(c)}_{\text{prior}}$$



UiO : University of Oslo

# Naive Bayes classification

- To classify a document d, we can thus compute

$$\begin{aligned} c^* &= \operatorname*{argmax}_c P(w_1|c)...P(w_n|c)P(c) \\ &= \operatorname*{argmax}_c \ \log\left(P(w_1|c)...P(w_n|c)P(c)\right) \\ &= \operatorname*{argmax}_c \ \sum_{i=1}^{n} \log(P(w_i|c)) + \log(P(c)) \end{aligned}$$

- The conversion to logs is not strictly necessary, but simplifies the computation: addition is easier and less prone to floating-point errors than multiplication

# Naive Bayes estimation

- We are now able to classify documents
- ... but we still don't know how to statistically estimate the probability distributions $P(w_i|c)$ and $P(c)$!
- We can use the training set $\mathbb{D} = \{(d_i,c_i), 1 < i < n\}$ to compute good empirical estimates for these distributions





# Naive Bayes estimation

- For the prior P(c), we simply count the relative frequency of each class in the training set:

$$P(c) = \frac{N_c}{N}$$

where Nc is the number of documents of class c
N is the total number of documents

## Naive Bayes estimation

- We proceed similarly for the class-specific word likelihood P(wi|c):

$$P(w_i|c) = \frac{C_{c,w_i}}{C_c}$$

  - $C_{c,wi}$ is the number of occurrences of the word $w_i$ in documents of class c
  - $C_c$ is the total number of words in documents of class c





## Naive Bayes estimation

- We now have basic estimates for both the prior P(c) and the likelihood $P(w_i|c)$
- These estimates are called *Maximum-likelihood* estimates, since they assign a maximum likelihood to the training data
- But they have some disadvantages…

# Naive Bayes classification

- Maximum-likelihood estimation has a problem with low-frequency counts
  - If a word wi never occurs for a document of class d, the probability $P(w_i|d)$ will be = 0
  - This is not reasonable, especially if we work with limited training data
  - We can partially alleviate the problem by using smoothing techniques





# Naive Bayes estimation

- Add-one or Laplace smoothing

$$P(w_i|c) = \frac{C_{w_i,c} + 1}{C_c + |V|}$$

- |V| is the vocabulary size, for normalisation
- Simply add one to the counts
- **Note**: such smoothing technique is more than a «trick», it can be derived mathematically using specific statistical assumptions

# Naive Bayes

- The model described so far is called multinomial Naive Bayes
  - An alternative is the *Bernoulli model*, based on presence/absence of terms in the document
  - See section 13.3 in the textbook for details
- Naive Bayes assumes that the features are conditionally independent (given the class)
  - This assumption rarely holds in real-world applications
  - … but NB classifiers are often surprisingly robust





# Outline of the lecture

- The classification task
- Naive Bayes
- **Feature selection**
- Evaluation of classifiers
- Conclusion

# Feature selection

- Most classification methods allow for arbitrary numbers of features
  - Some methods can scale to millions of features!
- Designing the right feature representation is a question of *trade-offs*
  - Not enough features: information loss
  - Too many features: not enough data to accurately estimate their corresponding parameters (data sparsity)





# Mutual information

- Key idea: find features that are correlated with the classification output
  - Uncorrelated features will not help the classification
- **Mutual information** measures how much information the feature value contributes to making the correct classification decision

$$I(X,C) = \sum_{c \in C} \sum_{x \in X} P(x,c) \log \left( \frac{P(x,c)}{P(x)P(c)} \right)$$

where X is a particular feature (and x a particular value for it)
C the classification decision (and c a particular category)

# Feature selection

- Many other techniques for feature selection are available
  - The statistical test $\chi 2$ provides another method (see textbook for details)
  - Feature selection can be performed iteratively (hill-climbing techniques)
- Feature selection is important for **dimensionality reduction**



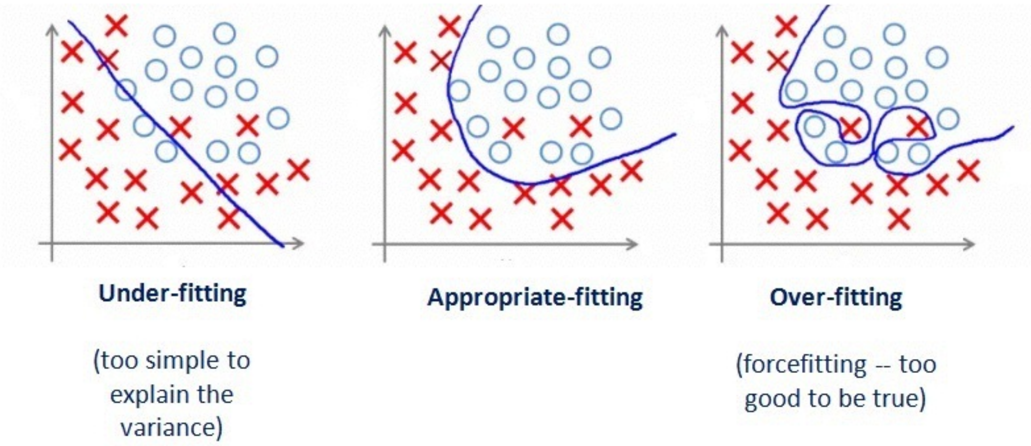


# Outline of the lecture

- The classification task
- Naive Bayes
- Feature selection
- **Evaluation of classifiers**
- Conclusion

# Evaluation

- High accuracy on training examples does not necessarily translate into good results on new data
  - Phenomenon of "overfitting"
  - (especially for high-dimensional spaces and/or non-linear models)








# Evaluation

- Must evaluate performance on a separate data set, called the **test set**
  - The test set must be kept *isolated* from the training set
  - We often divide the full data set into a training and testing part (typically 80% - 20% split)
  - When experiments are repeatedly made on the same data set, we first work on a *development set*, and only use the final (held-out) *test set* at the end

# Evaluation metrics

- Precision (for each class *c*):

$$\text{Precision} = \frac{\text{number of items correctly labelled as } c}{\text{number of items labelled as } c}$$

- Recall (for each class *c*):

$$\text{Recall} = \frac{\text{number of items correctly labelled as } c}{\text{number of items that actually belong to } c}$$

- (balanced) F-score (for each class *c*):

$$F = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$





# Evaluation metrics

We can also draw a confusion matrix:

| | | Gold standard | |
|---|---|---|---|
| | | **Positive** | **Negative** |
| **Actual classification:** | **positive** | True positive (tp) | False positive (fp) |
| | **Negative** | False negative (fn) | True negative (tn) |

$$\text{Precision} = \frac{tp}{tp + fp} \qquad \text{Recall} = \frac{tp}{tp + fn}$$

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

# Evaluation metrics

- For tasks with > 2 categories, we can compute averages:
  - **Macro-averaging**: mean of the measures for each class

*Example*: The macro-precision for the two classes is simply:
(10/20 + 90/100)/2 = 0.7

**Class 1**

| | | Gold + | Gold - |
|---|---|---|---|
| Actual | + | 10 | 10 |
| | - | 10 | 970 |

**Class 2**

| | | Gold + | Gold - |
|---|---|---|---|
| Actual | + | 90 | 10 |
| | - | 10 | 890 |

NB: averages can also be calculated for other metrics (recall, F-score, accuracy etc.)





# Evaluation metrics

- For tasks with > 2 categories, we can compute averages:
  - **Micro-averaging**: measure for a "pooled" confusion matrix

*Example*: The micro-precision for the two classes is:
100/120 = 0.83

**Class 1**

| | | Gold + | Gold - |
|---|---|---|---|
| Actual | + | 10 | 10 |
| | - | 10 | 970 |

**Class 2**

| | | Gold + | Gold - |
|---|---|---|---|
| Actual | + | 90 | 10 |
| | - | 10 | 890 |

**Pooled**

| | | Gold + | Gold - |
|---|---|---|---|
| Actual | + | 100 | 20 |
| | - | 20 | 1860 |

# Outline of the lecture

- The classification task
- Naive Bayes
- Feature selection
- Evaluation of classifiers
- **Conclusion**





# Conclusion

- **Classification** is a crucial part of IE systems
- Supervised learning used to automatically estimate classifiers from data
  - Based on a *training set* of labelled examples
  - Evaluated on a separate *test set*
  - Inputs represented as *feature vectors*
  - Example of learning algorithm: *Naive Bayes*
  - Challenges with *overfitting* & *data sparsity*