# Classification

* Can be rule-based, but mostly machine learned
* Text classification is a sub-class (examples: spam detection, genre classification, language identification, sentiment analysis)
* Other types of classification: word sense disambiguation, sentence splitting, tagging, named-entity recognition

# Machine Learning

* Supervised
  + Given classes
  + Given examples of correct classes
  + Classification (categorical)
  + Regression (numerical)
* Unsupervised
  + Construct classes
* Reinforcement learning

# Supervised Classification

* Given:
  + A well-defined set of observations O
  + A given set of classes C = {c1, c2, …, ck}
* Goal: a classifier, γ, a mapping from O to C
* For supervised training one need a set of pairs from OxC
* A well-defined set of observations O
* A given set of classes C = {c1, c2, …, ck}
* Some features f1, f2, …, fn
* For each feature: a set of possible values v1, v2, …, vn
* The set of feature vectors: v = v1 x v2 x … x vn
* Each observation in O is represented by some member of V:
  + Written (f1 = v1, f2 = v2, …, fn = vn)
  + Or (v1, v2, …, vn) if we have decided on the order
* A classifier γ can be considered a mapping from V to C

# Features

* To represent the objects in O, extract a set of features
* To be explicit
  + Which features
  + For each feature the type (categorical, numeric) and the value space

# Some ML Classifiers

|  |  |
| --- | --- |
| * K-nearest Neighbors * Rocchio * Naïve Bayes * Logistic Regression (maximum entropy) * Support Vector Machines | * Decision Trees * Perceptron * Multi-layered neural nets (“Deep Learning” |

# Baseline

* Start by constructing a baseline
* Survey, asked all 200 students
  + 130 yes
  + 70 no
* Baseline classifier
  + Choose the majority class
  + Accuracy 0.65 = 65%

# A little more formal

* Start by asking the students about their programming experience, advanced ML-experience and whether they like maths to find out whether people will enjoy the course or not
* We consider P(enjoy = yes | prog = good, AdvML = no, Maths = no) and   
  P(enjoy = not | prog = good, AdvML = no, Maths = no)
* And decide on the class which has the largest probability, in symbols  
  argmaxy ∈ {yes, no} P (enjoy = yes | prog = good, AdvML = no, Maths = no)

# Naïve Bayes and Training

* Choose the class with the largest value (likelihood)
* Assumes independence (wrongly), to simplify calculation



* Maximum likelihood, where C(sm, o) are the number of occurrences of observations o in class sm
* Observe what we are doing
  + We are looking for the true probability P(sm)
  + ^P(sm) is an approximation to this, our best guess is from a set of observations (^ sombolizes that it is not the real probability but the likelihood)
  + Maximum likelihood means that it is the model which makes the set of observations we have seen most likely
* Maximum likelihood, where C(fi = vi , sm) is the number of observations o
  + Where the observation o belongs to class sm
  + And the feature fi takes the value vi
* C(sm) is the number of observations belonging to class sm

# Laplace Smoothing

* Laplace Smoothing: if there is one feature which only appears in one class, add one occurrence so that every feature should at least occur once in a class (otherwise it would have too huge an influence on the choice of class)
* Nltk.NaiveBayesClassifier uses Lidstone (0.5) as default

# Properties of Naïve Bayes

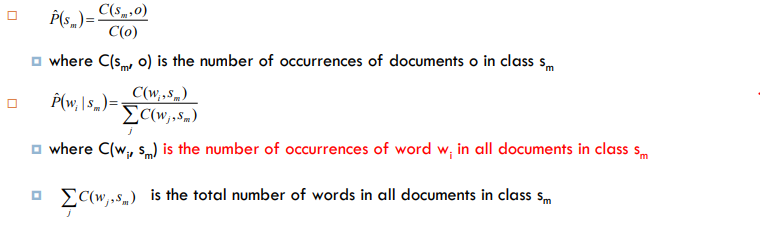
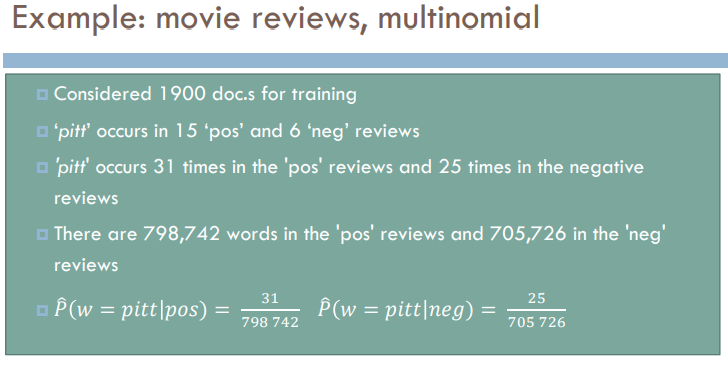
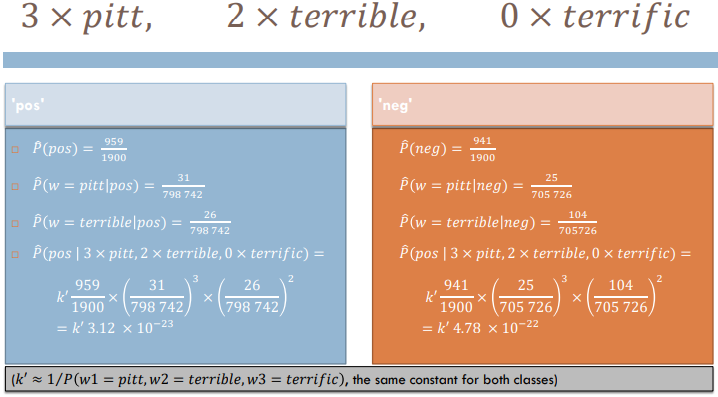
* Probabilistic classifier
* Multi-class classifier, i.e. it can handle more than two classes
* Categorical features natively, can be adapted to numeric features
* NLTK contains an implementation
* The independence assumption is in general wrong
  + P(v1, v2, …, vn | c) is far from P(v1 | c) x P(v2 | c) x …. P(vn | c)
* Still NB works reasonably well as a classifier (discriminator)
* It is not prone to overfitting (overfitting: model gives good predictions for the features it was trained on but on other data sets it does not predict well because it is just too used to the model it was trained on)
* Other classifiers may work better

# Text Classification with NB

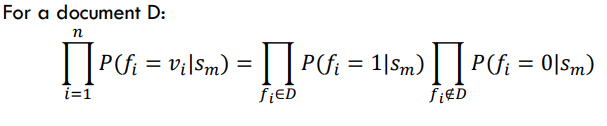
* Naïve Bayes may be applied to various NLP tasks
* Text classification
  + Goal: classify the text on the basis of the words in the text
  + What are the features?
  + What are the possible values?
* Two possible answers: Multinomial Model and Bernoulli Model

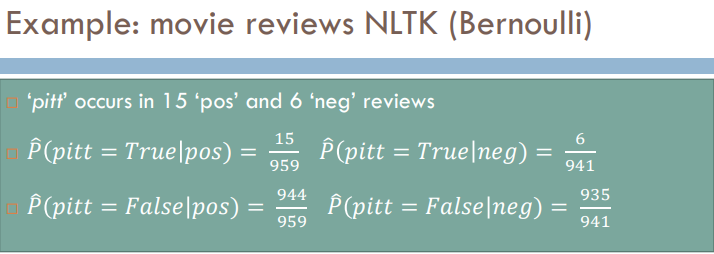
# Multinomial NB text classification

* Fi refers to the position i in the text
* Vi is the word occurring in this position
* N is the number of tokens in the text
* Simplifying assumption: a word is equally likely in all positions; hence we count how many times each word occurs in the text
* Creates a bag of words representation: word can occur several times (no tokenization)



# Bernoulli model for text classification

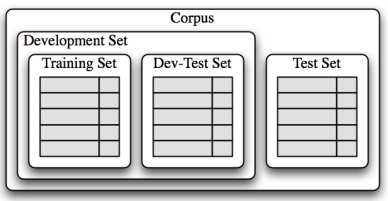
* How are words turned into features?
* A vocabulary of words, W
* Each word wi makes a feature fi
* The possible values for fi is True and False (1 and 0)
* fi = 1 in a document if and only if it contains wi
* fi refers to a word in the vocabulary
* vi is 1 or 0 depending on whether the word occurs in the text or not
* n is the number of words in the vocabulary



# Comparison: Multinomial vs. Bernoulli

|  |  |
| --- | --- |
| Multinomial | Bernoulli |
| * counts how many times a term is present * considers only the present terms and ignores absent terms * n: number of words in the document * tends to be the better of the two for longer texts | * registers whether a term is present or not * considers bot, present and absent terms * n: number of words in the vocabulary * compatible on shorter snipeets |

# Set up for experiments

* before you start: split into development (training) set and test set
* hide the test set
* split development set into Training and Development-Test set
* use training set for training a learner
* use dev(-test) for repeated evaluation in the test phase
* finally test on the test set

# Procedure

1. train classifier on training set
2. test it on dev-test set
3. compare to earlier runs, is it better?
4. Error analysis: What are the mistakes (on dev-test set=
5. Make changes to the classifier
6. Repeat from 1

----------------- when you have run empty on ideas, test on test set. Stop.

# Cross-Validation

* Small test sets 🡪 large variation in results
* N-fold cross-validation
  + Split the development set into n equally sized bins
  + Conduct n many experiments: in experiment m, use part m as test set and the other n-1 parts as training set
  + This yiels n many results
    - We can consider the mean of the results
    - We can consider the variation between the results

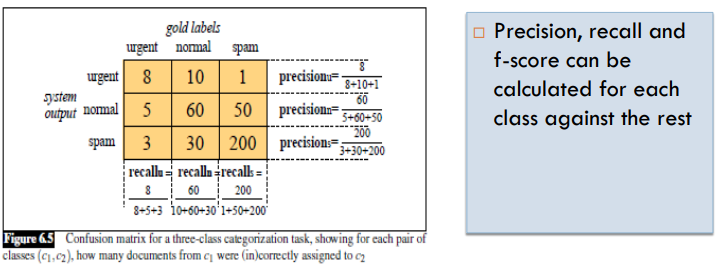
# Evaluation measure: Accuracy

* What does accuracy 0.81 tell us?
* Given a test set of 500 documents
  + The classifier will classify 405 correctly
  + 95 incorrectly
* A good measure given
  + The 2 classes are equally important
  + The 2 classes are roughly equally sized
* BUT: for some tasks, the classes aren’t equally important (worse to loose an important mail than to receive another spam mail)
* For some tasks the different classes have different sizes

# Information Retrieval (IR)

* Traditional IR, e.g. library
* Goal: find all the documents on a particular topic out of 100.000 documents
  + Say there are 5
  + The system delivers 10 documents: all irrelevant 🡪 What is the accuracy?
* For these tasks, focus on
  + The relevant documents
  + The documents returned by the system
* Forget the irrelevant documents which are not returned

# IR Evaluation and Confusion Matrix



# Evaluation Measures

