DM4
Naive Bayes and Nearest Neighbors
Baseline algorithms

• Simple algorithms but effective
• Two different methods:
  • **Naïve Bayes.** Parametric: It builds a probabilistic model of your data following some assumptions.
  • **K-NN.** Non parametric method: In this case a lazy Instance Based Learning method that does not build any model.
Naïve Bayes
Naive Bayes basics

1. For learning, with the examples, we estimate the likelihood of our data:

$$p(x_1, x_2, \ldots x_n | c_i)$$

that means, probability that observation \((x_1, x_2, \ldots x_n)\) belongs to class \(c_i\) \([x_i\, \text{is feature } i \text{ of observation } x]\)

2. But for classifying, given an observation \((x_1, x_2, \ldots x_n)\), we look for the class that maximize the \textit{a priori} probability of belonging to the class:

$$c_{MAP} = \arg\max_{c_j \in C} P(c_j \mid x_1, x_2, \ldots, x_n)$$
Naïve Bayes classifiers

We will use the Bayes’ theorem:

\[
c_{MAP} = \arg\max_{c_j \in C} P(c_j | x_1, x_2, \ldots, x_n) = \arg\max_{c_j \in C} \frac{P(x_1, x_2, \ldots, x_n | c_j)P(c_j)}{P(x_1, x_2, \ldots, x_n)}
\]

Bayes’ theorem:

\[
\begin{align*}
P(A|B) &= \frac{P(A \land B)}{P(B)} \\
P(B|A) &= \frac{P(A \land B)}{P(A)}
\end{align*}
\]

\[
\Rightarrow P(A \land B) = P(A|B)P(B) = P(B|A)P(A)
\]

\[
\Rightarrow P(A|B) = \frac{P(B|A)P(A)}{P(B)}
\]
Computing probabilities

- $P(c_j)$ - Simply proportion of elements in class $j$
- $P(x_1, x_2, \ldots, x_n | c_j)$
  - Problem $|X|^n / |C|$ parameters!
  - It can only be estimated from a very huge dataset. Impractical
- **Solution:** Independence assumption (very Naïve): attribute values are independent. So in this case, we can easily compute

$$P(x_1, x_2, \ldots, x_n | c_j) = \prod_i P(x_i | c_j)$$
Computing probabilities

- \( P(x_k|c_j) \)
  - Now we only need \( n./C/ \) probability estimations
  - Very easy. Number of values with property \( x_k \) in class \( c_j \) over the complet number of cases in class \( c_j \)

- Solving now,

\[
P(x_1, x_2, \ldots, x_n | c_j) = \prod_{i} P(x_i | c_j)
\]

the class assigned to a new observation is:

\[
c_{NB} = \arg\max_{c_j \in C} \frac{P(x_1, x_2, \ldots, x_n | c_j)P(c_j)}{P(x_1, x_2, \ldots, x_n)} = \arg\max_{c_j \in C} P(c_j) \prod_{i} P(x_i | c_j)
\]

Equation to be used
Practical issues

- Being probabilities in range 0..1, products easily end with *floating-point underflow* errors.
- Knowing that $\log(xy) = \log(x) + \log(y)$, it is better to work with $\log(p)$ than with probabilities.

- Now:

$$c_{NB} = \arg\max_{c_j \in C} \left( \log P(c_j) + \sum_{i \in \text{positions}} \log P(x_i | c_j) \right)$$
Example: Learning to classify texts

- Training set: X document corpus
- Each document is labeled with $f(x) = \text{like/dislike}$
- Goal: Learn function that permits given new document if you like it or not.
- Questions:
  - How do we represent documents?
  - How to compute probabilities?
Example: Learning to classify texts

• How do we represent documents?
  • Each document is represented as a Bag of Words
  • Attributes: All words that appear in the document
  • So each document is represented as a boolean vector with length N: 0 – word does not appear; 1 – word appears

• Practical problem: A very huge table.
• Solution: Use sparse representation of matrixes
Example: Learning to classify texts

• Some numbers
  • 10,000 documents
  • 500 words per document
  • Maximum theoretical number of words: 50,000 (much less because of word repetitions)

• Reducing the number of attributes
  • Removing the number (sing/plural) and verbal forms (stemming)
  • Remove conjunctions, propositions and articles (stop words)
  • Now we have about 10,000 attributes
Example: Learning to classify texts

- How to compute probabilities?
- For each word

\[ v_{NB} = \operatorname{argmax}_{v \in \{\text{like, dislike}\}} P(v) \prod_{i} P(x_i = \text{word}_i \mid v) \]

- “a priori” probability for like and dislike classes

\[ P(v_{\text{like}}) = \frac{\#\text{documents like}}{\text{total number of documents}} \]

\[ P(v_{\text{dislike}}) = \frac{\#\text{documents dislike}}{\text{total number of documents}} \]
Example: Learning to classify texts

\[
\begin{align*}
    v_{NB} &= \arg\max_{v \in \{\text{like, dislike}\}} P(v) \prod_i P(x_i = \text{word}_i \mid v) \\
    \text{Number of parameters} &\quad P(x_i = \text{word}_i \mid v) \\
    \text{10,000 words and two classes (so about 20,000)} &
\end{align*}
\]

\[
P(\text{word}_k \mid v) = \frac{\# (\text{docs. } v \text{ in training where word}_k \text{ appears})}{\# (\text{documents } v)} = \frac{n_k}{n}
\]
Example: Learning to classify texts

• Problem:

\[
P(\text{word}_k \mid v) = \frac{\# \text{(docs. } v \text{ del training on word}_k \text{ apareix})}{\# \text{(documents } v)} = \frac{n_k}{n}
\]

• When \(n_k\) is low, not an accurate probability
• when \(n_k\) is 0 for \(\text{word}_k\) for one class \(v\), then any document with that word will never be assigned to \(v\) (independent of other appearing words)
Example: Learning to classify texts

• Solution: More robust computation of probabilities (Laplace smoothing)

\[ P(\text{word}_k \mid v) = \frac{n_k + mp}{n + m} \]

• Where:
  • \( n_k \) is # of documents of class \( v \) in which word \( k \) appear
  • \( n \) is # of documents with label \( v \)
  • \( p \) it’s a likelihood estimation of “a priori” \( P(x_k \mid v) \) (f.i., uniform distribution)
  • \( m \) is the number of labels
Example: Learning to classify texts

Smoothing: \[ P(x_k | v) = \frac{n_k + mp}{n + m} \]

More common “a priori” uniform distribution:

1. When two classes: \( p = 1/2, \ m = 2 \) (Laplace Rule)
   \[ P(x_k | v) = \frac{n_k + 1}{n + 2} \]

2. Generic case (\( c \) classes): \( p = 1/c, \ m = c \)
   \[ P(x_k | v) = \frac{n_k + 1}{n + c} \]
Example: Learning to classify texts

- Naïve Bayes return good accuracy results even when independence assumption is not fulfilled
  - In fact, Spam/not Spam implementation of Thunderbird work in this way
  - Applied to document filtering (fi. *Newsgroups* or *incoming mails*)

- Learning and testing time are linear with the number of attributes!
Extension to continuous attributes

• Assume each class follows a normal distribution for each variable

\[ f(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \]

• For instance 73 is average of feature temp. for class x, and std=26.2, we compute conditional prob in the following way:

\[ p(temperature = 66 \mid x) = \frac{1}{\sqrt{2\pi6.2}} e^{-\frac{(66-73)^2}{26.2^2}} = 0.034 \]
Instance Based Learning
Instance Based Learning

- Lazy learning methods: they don’t build a model of the data

- Assign the label to an observation depending on the labels of “closest” examples

- Requirements:
  - A training set
  - A similarity measure
Instance Based Learning Algorithms

• K-NN
• Distance Weighted kNN
• How to select K??
• How to solve some problems
K-NN

• K-Nearest neighbor algorithm

• It interprets each example as a point in a space defined by the features describing the data

• In that space a similarity measure allows as to classify new examples.

• Class is assigned depending on the $K$ closest examples
1-NN example

• Two real features \((x_1, x_2)\) define the space.
• Each red point is a positive example. Black points are negative examples.

Equivalent to draw the Voronoi space of your data.
1-NN example

- Two real features \((x_1, x_2)\) define the space.
- Each red point is a positive example. Black points are negative examples.
Distance measures

- Euclidean

\[ d(x_i, x_j) = \sqrt{\sum_{k=1}^{n} (x_i(k) - x_j(k))^2} \]

- Weighted euclidean

\[ d(x_i, x_j) = \sqrt{\sum_{k=1}^{n} w_k(x_i(k) - x_j(k))^2} \]
Weigthing effect

\[ d^2(x_i, x_j) = (x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 \quad \quad d^2(x_i, x_j) = 9(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 \]
Some other measures
Some comments

Advantages:
• Fast training
• Ability to learn very complex functions

Problems:
• Very slow in testing. Needed some smart structure representation of data in trees
• Fooled by noise
• Fooled when irrelevant features
Some comments:

• Building more robust classifiers
  • Results do not depend on the closest example but on the k closest examples (so *k-nearest neighbours* (kNN) name)
Example

- kNN with $k=5$
3-Nearest Neighbors

query point $q_f$

3 nearest neighbors

2x, 1o
7-Nearest Neighbors

query point qf

7 nearest neighbors

3x, 4o
K-NN algorithm

• **Parameters:**
  • Natural number $k$ (*even number*)
  • Training set
  • Distance measure

• **Algorithm:**
  1. Store all training set $<x_i, \text{label}(x_i)>$
  2. Given new observation, $x_q$, compute the nearest $k$ neighbors
  3. Let vote the nearest $k$ neighbors to assign the label to the new data.
How to select k?

• High number of k show two advantages:
  • Smother frontiers
  • Reduces sensibility to noise

• But too large values are bad because
  • We loose locality in the decision because very distant points can interfere in assigning labels
  • Computation time is increased

• K-value usually is chosen by cross-validation.
Distance Weighted kNN

• A smart variation of KNN.
• When voting, all k neighbors have the same influence, but some of them are more distant than the others (so they should influence less in decisions).
• Solution: Given more weight to closest examples.
Distance Weighted kNN

Label is assigned by the weighted addition:

\[ f_l = \sum_{i=1}^{k} w_i \cdot l(x_i) \]

where \( l(x_i) \) is \{-1,1\} the label of example \( x_i \)
\( w_i = K(d(x_q,x_i)) \) is the weight of example \( i \)
\( K(d) \) is a weight function depending on distance, and
\( d(x_q,x_i) \) is distance from \( x_q \) to \( x_i \)
Distance Weighted kNN

\[ d = d(x_i, x_{\text{query}}) \]

Common weighting functions
Problems with irrelevant features

KNN is fooled when irrelevant features. Let's assume examples described with 20 attributes, but only 2 are relevant to the classification.

kNN will be lost easily in this case.

Solution consists in feature removal:

- Find weights $z_1, \ldots, z_n$, one for each feature, that minimize error in a validation data set (so use cross-validation to choose weights).

- Notice that setting $z_j = 0$ means removing the feature.