Machine Learning: fundamentals and algorithms

Guillaume Metzler

Laboratoire Hubert Curien, UMR CNRS 5516
University of Jean Monnet Saint-Étienne (France)

my contact: guillaume.metzler@univ-st-etienne.fr

Outline

1. Bayesian Method
   - Bayesian Error
   - Statistical Methods
2. k-Nearest Neighbors
   - Theoretical Analysis
   - Computational Analysis and Optimizations
   - Condensed Nearest Neighbors
3. Conclusions

Bayesian Method

To assign a label c to an unknown instance x, we have to compute the posterior probability $p(y = c|x)$.

Bayesian Method

The Bayesian method consists of detecting the optimal class $c \in \mathcal{Y}$ of an example $x \in \mathcal{X}$ by applying the **Maximum a posteriori (MAP) decision rule**:

$$\forall c \in \mathcal{Y},\ p(y = c|x) = \frac{p(x|y = c)p(y = c)}{p(x)}$$

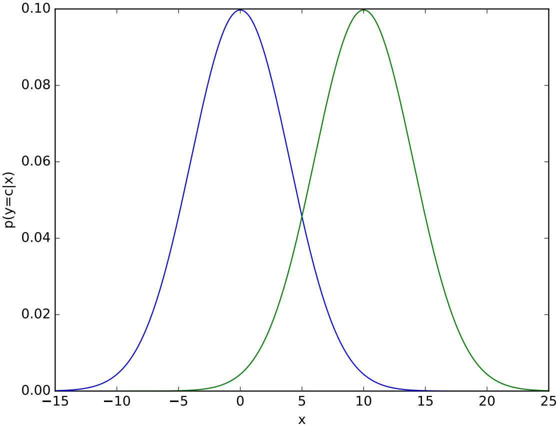
$$y(x) = \arg\max_{c} p(y = c|x).$$

That means that $y(x) = \arg\max_c p(x|y = c)p(y = c)$.

Bayesian Error

Minimal prediction error due to the nature of the distributions of the classes.

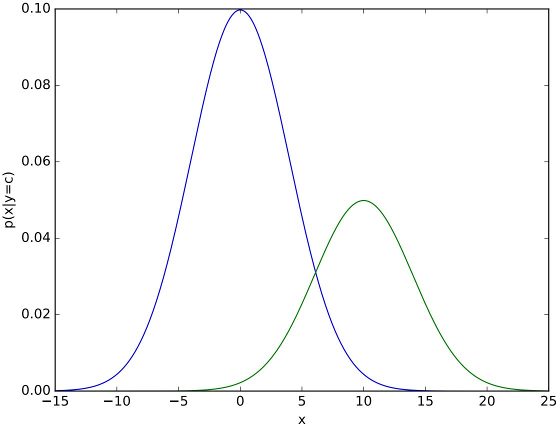
Example: normal distributions, balanced classes



Bayesian Error

Minimal prediction error due to the nature of the distributions of the classes.

Example: normal distributions, unbalanced classes



Bayesian Error

Exercise:

Let f_{c1} and f_{c2} be the densities of two classes c_1 and c_2:
$f_{c1}(x) = \frac{3}{2}x^2 + x$ for $0 < x < 1$
$f_{c2}(x) = 1$ for $\frac{3}{4} < x < \frac{7}{4}$.

Draw the distribution and compute their bayesian error.

Solution

$$\epsilon^* = p(\frac{3}{4} < x < 1 \cap y = c_2) = p(\frac{3}{4} < x < 1 | y = c_2)p(y = c_2)$$

$$= \frac{1}{4}\frac{1}{2} = \frac{1}{8}$$

Bayesian Method

Underlying conditions to solve this problem

To use the bayesian method, one needs some priors:

1. Know the a priori probabilities $\boldsymbol{p}(\boldsymbol{y} = \boldsymbol{c})$ of the different classes.
2. Know the probabilities of the observations given the classes $\boldsymbol{p}(\boldsymbol{x}|\boldsymbol{y} = \boldsymbol{c})$.

Without any background knowledge, this requires to estimate these two quantities from the training sample S.

Statistical Methods

Estimation of $p(y = c)$

- We can either assume that the classes are equally distributed, such that $p(y = c) = \frac{1}{|\mathcal{Y}|}$
- or that the learning set S has been correctly drawn from the target probability distribution. Therefore, we can use the frequency of each class such that $p(y = c) = \frac{|S_c|}{|S|}$ where S_c is the set of instances of class c.

Estimation of $p(x|y = c)$

We can distinguish two types of approaches:

1. The **parametric methods** which assume that $p(x|y = c)$ follows a given statistical distribution. In this case, the problem to solve consists in estimating the parameters of the considered distribution (e.g. normal distribution with σ and μ or Binomial distribution with p).
2. The **non parametric methods** which do not impose any constraint about the underlying distribution, and for which the densities $p(x|y = c)$ are locally estimated around x.

k-Nearest Neighbors

Classification with k-NN

Non-parametric method by which an instance is assigned to the most common class in its neighborhood. The neighborhood is determined by the k closest training points.

$$c = \arg\max_{c \in \mathcal{Y}} \frac{k_c}{k}$$

with k_c the number of training instances of class c in the neighborhood.

1. **Training**: memorize training set
2. **Prediction of** y_i: majority vote of the k nearest neighbors of x_i

It assumes that $p(x|y = c)$ is locally regular.

k-Nearest Neighbors

Algorithm

Input: x: an instance
Input: S: a sample
Output: y: the class of x
begin
  foreach $(x_i, y_i) \in S$ **do**
    Compute the distance $d(x_i, x)$;
  end
  Sort the n distances by increasing order;
  Count the number of occurrences of each class c among the k nearest neighbors;
  Assign to x the most frequent class.
  return y
end

k-Nearest Neighbors

Is $\frac{k_c}{k}$ a good estimation of $p(y_i = c|x_i)$?

Proof

Let p be an unknown probability density. Let us assume we want to estimate $p(x)$. The probability P of observing x in a portion r of the space of volume V is:

$$P = \int_V p(x)dx$$

Assuming that $p(x)$ is continuous and does not signicantly change in r, we can approximate P such that:

$$P \approx \hat{P} = p(x) \times V$$

.

k-Nearest Neighbors

Is $\frac{k_c}{k}$ a good estimation of $p(y_i = c|x_i)$?

Proof

P can also be estimated by the proportion of training data in r:

$$P \approx \hat{P} = \frac{k}{n}$$

with k the number of points in r and n the total number of points.
Therefore, we can deduce that

$$p(x) \approx \frac{P}{V} = \frac{k}{nV}$$

.

k-Nearest Neighbors

Is $\frac{k_c}{k}$ a good estimation of $p(y = c|x)$?

k-NN proofs

The posterior probability $p(y = c|x)$ can be rewritten as:

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

Assuming that x belongs to the portion of the space r of volume V:
$p(x|y = c) \approx \frac{k_c}{n_c V}$, $p(y = c) = \frac{n_c}{n}$ and $p(x) \approx \frac{k}{nV}$.
Therefore:

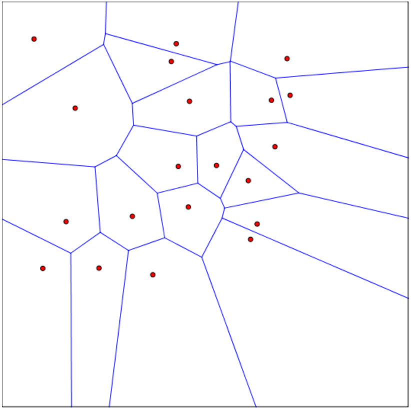
$$p(y = c|x) \approx \frac{\frac{k_c}{n_c V}\frac{n_c}{n}}{\frac{k}{nV}} = \frac{k_c}{k}.$$

$$\arg\max_c p(y = c|x) = \arg\max_c \frac{k_c}{k}.$$

1-Nearest Neighbor

Special case: $k=1$

1-NN boils down to partitionate the space $\mathcal{X}$ into Voronoi cells and affect them the label of their centroid.



k-Nearest Neighbors

Convergence properties of the 1-Nearest Neighbor

Theorem

Let x' be the nearest neighbor of x,

$$\lim_{n\to\infty} P(d(x,x') > \epsilon) = 0,\ \forall \epsilon > 0.$$

In other words,

$$\lim_{n\to\infty} p(y=c|x') = p(y=c|x).$$

k-Nearest Neighbors

Proof

Let p be the probability that the hypersphere $s(x,\epsilon)$ centered at x of radius ϵ does not contain any point of S and p_ϵ the probability that a point $x_i \in S$ is inside the hypersphere:

$$P(d(x,x') > \epsilon) = p = P(x_1 \notin s(x,\epsilon), ..., x_n \notin s(x,\epsilon))$$

$$= \prod_{i=1}^{n} P(x_i \notin s(x,\epsilon)) = \prod_{i=1}^{n}(1 - P(xi \in s(x,\epsilon)))$$

assuming that S is i.i.d., so the events are independent. Then,

$$p = \prod_{i=1}^{n}(1 - p_\epsilon) = (1 - p_\epsilon)^n$$

and

$$\lim_{n\to\infty} p = 0.$$

k-Nearest Neighbors

How to choose k?

- if k too small, noise has great influence
- if k too big, local information is lost

Recall that, $p(x) \approx \hat{p}(x) = \frac{k}{nV}$.

Theorem

When n is increasing, $\hat{p}(x)$ converges to $p(x)$ if the following three conditions are fulled:

$$\lim_{n \to \infty} V = 0$$

$$\lim_{n \to \infty} k = \infty$$

$$\lim_{n \to \infty} \frac{k}{n} = 0$$

If we are considering r as an hypersphere, all three conditions are fullfilled for $k = \sqrt{n}$.

k-Nearest Neighbors

Computational and Memory Storage Analysis

Without any optimization,

- complexity: $\mathcal{O}(nd + nk)$ for distances computation and neighbors selection;
- memory: $\mathcal{O}(n)$ for distances storage.

Two strategies to reduce these costs:

- Reduce n while keeping the most relevant examples (e.g. the **condensed nearest neighbor rule** (Hart 1968)).
- Simplify the computation of the nearest neighbors.

k-Nearest Neighbors

Remove from S the outliers and the examples of the bayesian error region.

Algorithm

Input: S: a sample
Output: $S_{cleaned}$: a smaller sample
begin
  Split randomly S into two subsets S_1 and S_2;
  while *no stabilization of* S_1 *and* S_2 **do**
    Classify S_1 with S_2 using the 1-NN rule;
    Remove from S_1 the misclassied instances;
    Classify S_2 with the new set S_1 using the 1-NN rule;
    Remove from S_2 the misclassied instances;
  end
  $S_{cleaned} = S_1 \cup S_2$;
  return $S_{cleaned}$
end

Condensed Nearest Neighbors

Remove from S the irrelevant examples.

Algorithm

Input: S: a sample
Output: $S_{selected}$: a smaller sample
begin
    $S_{selected} \leftarrow \emptyset$;
    Draw randomly a training example from S and put it in $S_{selected}$;
    while *no stabilization of* $S_{selected}$ **do**
        for *instance* $x_i \in S$ **do**
            if x_i *missclassified using* 1*NN with* $S_{selected}$ **then**
                $S_{selected} \leftarrow x_i$
            end
        end
    end
    return $S_{selected}$
end

Conclusions

1. With a sufficiently large number of training examples, a k-NN classier is able to converge towards very complex target functions.
2. It is simple and theoretically well founded.
3. There exist several solutions to overcome its problems of algorithmic complexity (time and space).