Machine Learning

Lecture 3.2: *k*-Nearest Neighbors

Marc Sebban and Amaury Habrard

Laboratoire Hubert Curien, UMR CNRS 5516
Université Jean Monnet Saint-Étienne

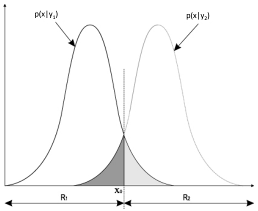
Bayesian Error

Bayesian Error

The bayesian error ϵ^* is the lowest possible error rate (or irreducible error) for any hypothesis h.

$$\epsilon^* = \int_{x \in R_i \text{ s.t. } y \neq y_i} p(y_i|x)p(x)dx$$

where x is an instance, y its corresponding label, R_i is the area/region that a classifier function h classifies as y_i.



Bayesian Classifier

Bayesian Classifier

The Bayesian classifier predicts the optimal class $y^* \in \mathcal{Y}$ given an example $\mathbf{x} \in \mathcal{X}$ by applying the Maximum a posteriori (MAP) decision rule:

$$\forall y_j \in \mathcal{Y}, p(y_j|\mathbf{x}) = \frac{p(\mathbf{x}|y_j).p(y_j)}{p(\mathbf{x})}$$

$$y^*(\mathbf{x}) = \arg\max_c p(y_c|\mathbf{x}).$$

that corresponds to $y^*(\mathbf{x}) = \arg\max_c p(\mathbf{x}|y_c).p(y_c)$

If this calculation is possible, the Bayesian classifier is optimal from a probabilistic point of view, with an associated error ϵ^*.

Underlying conditions to solve this problem

To compute ϵ^*, one needs some priors:

1. Know the *a priori* probabilities $p(y_j)$ of the different classes.
2. Know the probabilities of the observations given the classes $p(\mathbf{x}|y_j)$.

Unfortunately, $p(y_j)$ and $p(\mathbf{x}|y_j)$ are unknown. One needs to estimate these two quantities from the training sample S.

Estimation of the a priori probability of the classes $p(y_j)$

What about $p(y_j)$?

An unbiased estimate of $p(y_j)$ is given by the observed frequency $\hat{p}(y_j) = \frac{|S_j|}{|S|}$ where $|S_j|$ is the number of training examples belonging to the class y_j.

Estimation of the conditional probabilities $p(\mathbf{x}|y_j)$

What about $p(\mathbf{x}|y_j)$?

We can distinguish two types of approaches:

1. The parametric methods which assume that $p(\mathbf{x}|y_j)$ 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 μ and σ 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 $p(\mathbf{x}|y_j)$ are locally estimated around $\mathbf{x}$.

Non parametric methods

- No assumption is made on the underlying distribution...
- ... we only suppose that this target distribution is locally regular.
- The objective is to estimate $p(\mathbf{x}|y_j)$. Since this must be done $\forall y_j \in \mathcal{Y}$, for the sake of simplicity, let us focus on $p(\mathbf{x})$ first.

Non parametric methods

- Let p be the unknown target probability density. The probability $\mathcal{P}$ to observe x in a volume V is:

$$\mathcal{P} = \int_V p(\mathbf{x})d\mathbf{x}$$

- Assuming that $p(\mathbf{x})$ does not significantly change in V (locally regular), we can approximate P such that:

$$\hat{\mathcal{P}} \simeq p(\mathbf{x}) \times V \qquad (1)$$

- $\mathcal{P}$ can also be estimated by the proportion of training data in V:

$$\hat{\mathcal{P}} \simeq \frac{k}{n} \qquad (2)$$

- Therefore, we can deduce that

$$p(\mathbf{x}) \approx \frac{k}{nV} = \hat{p}(\mathbf{x}).$$

Theorem

Let us denote by $\hat{p}_n(\mathbf{x}) = \frac{k_n}{nV_n}$ *the estimate of* $p(\mathbf{x})$ *from a training sample* S *of size* n*. When* n *is increasing,* $\hat{p}_n(\mathbf{x})$ *converges to* $p(\mathbf{x})$ *if the following three conditions are fulfilled:*

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

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

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

Interpretation

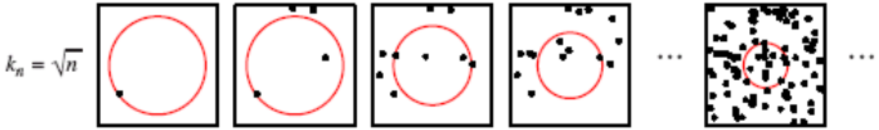
If n is high, we get:

- a good estimate $\hat{p}(x)$ (resp. $\hat{p}(x|y_j)$ if we take into account the class) of $p(x)$ (resp. $p(x|y_j)$).
- a good approximation of the Bayesian method.

Non parametric methods

k-Nearest Neighbors (Cover & Hart 1968)

It turns out that the *k*-nearest neighbor method satisfies the previous conditions: **fixing a number k_n of examples** (growing w.r.t. n) and **adapting a volume V_n** (*e.g.* a hypersphere centered at $\mathbf{x}$) such that k_n examples are in the volume.

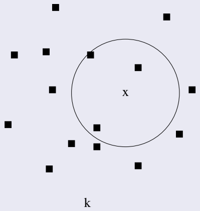


k-nearest neighbors

kNN as a good way to estimate $p(\mathbf{x})$

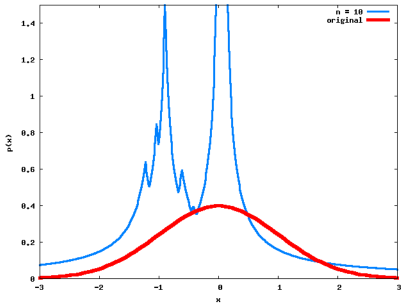
To estimate $p(\mathbf{x})$ from the n training examples of S, let us build a hypersphere centered at $\mathbf{x}$ that contains k_n examples.

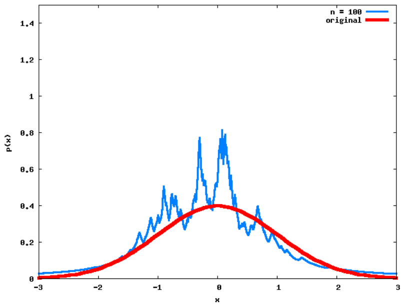
$\hat{p}_n(\mathbf{x}) = \frac{k_n/n}{V_n}$ is a good estimate of $p(\mathbf{x})$



Example 1 with $k = \sqrt{n}$

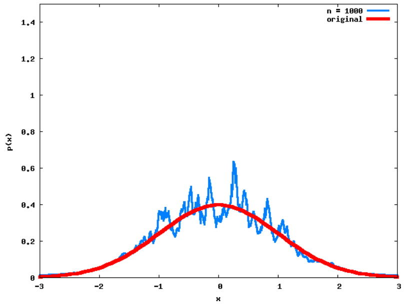
$$p(\mathbf{x}) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\mathbf{x}^2}$$

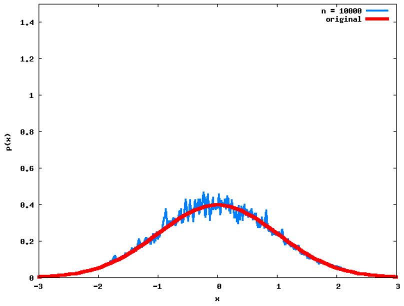




Example 1 with $k = \sqrt{n}$

$$p(\mathbf{x}) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\mathbf{x}^2}$$

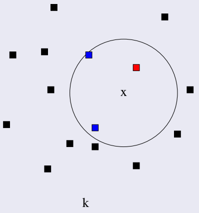




k-NN as a classifier

Let us suppose we have n_j examples in S of class y_j, such that $\sum_j n_j = n$. Suppose the hypersphere contains k_j examples of class y_j ($\sum_j k_j = k$). If we aim at classifying a new example $\mathbf{x}$ with its k nearest-neighbors, and applying the Bayes rule, we get:

$$h(\mathbf{x}) = arg \max_j \frac{\hat{p}(\mathbf{x}|y_j).\hat{p}(y_j)}{\hat{p}(\mathbf{x})} = arg \max_j \frac{\frac{k_j}{V_n \times n_j} \times \frac{n_j}{n}}{\frac{k}{n \times V_n}} = arg \max_j \frac{k_j}{k}$$



k-nearest neighbors algorithm

How to classify an unknown example $\mathbf{x}$ using S?

```
Input: $\mathbf{x}, S, d$
Output: class of $\mathbf{x}$
foreach $(\mathbf{x}', y') \in S$ do
    Compute the distance $d(\mathbf{x}', \mathbf{x})$;
Sort the $n$ distances by increasing order;
Count the number of occurrences of each class $y_j$ among the $k$ nearest neighbors;
Assign to $\mathbf{x}$ the most frequent class;
```

Special case of the $(k = 1)$-nearest neighbor rule

Convergence properties of the $(k = 1)$-nearest neighbor rule

Theorem

Let $\mathbf{x}'$ *be the nearest neighbor of* $\mathbf{x}$*,*

$$\lim_{n \to \infty} p\left(d(x, x') > \epsilon\right) = 0, \forall \epsilon > 0$$

Corrolary

If $n \to \infty$, $p(y_j|\mathbf{x}') \approx p(y_j|\mathbf{x})$.

Special case of the $(k = 1)$-nearest neighbor rule

Proof.

Let $\mathcal{P}$ be the probability that the hypersphere $s(\mathbf{x}, \epsilon)$ centered at $\mathbf{x}$ of radius ϵ does not contain any point of S:

$$\mathcal{P} = p(\mathbf{x_1} \notin s(\mathbf{x}, \epsilon), \ldots, \mathbf{x_n} \notin s(\mathbf{x}, \epsilon)) = \prod_{i=1}^{n} p(\mathbf{x_i} \notin s(\mathbf{x}, \epsilon))$$

$$= \prod_{i=1}^{n} (1 - p(\mathbf{x_i} \in s(\mathbf{x}, \epsilon))) = (1 - p_\epsilon)^n.$$

Since the nearest neighbor $\mathbf{x}'$ of $\mathbf{x}$ is also outside of the sphere, we get

$$p(d(\mathbf{x}, \mathbf{x}') > \epsilon) = (1 - p_\epsilon)^n$$

$$\text{therefore, } \lim_{n \to \infty} p(d(\mathbf{x}, \mathbf{x}') > \epsilon) = \lim_{n \to \infty} (1 - p_\epsilon)^n = 0$$

$\square$

Theorem

The generalization error ϵ_{1NN} of the 1-nearest neighbor rule is bounded by twice the (optimal) bayesian error ϵ^*.

$$\epsilon_{1NN} \leq 2\epsilon^*$$

Corrolary

Half of the information about the true class of an example $\mathbf{x}$ is contained in its nearest neighbor $\mathbf{x}'$.

Special case of the $(k = 1)$-nearest neighbor rule

Proof.

Bayesian error:

$$\begin{aligned}\forall \mathbf{x} \in \mathcal{X},\ \epsilon^*(\mathbf{x}) &= \min\{p(y_1|\mathbf{x}), p(y_2|\mathbf{x})\}\\ &= p(y_{min}|\mathbf{x})\end{aligned}$$

1NN error:

$$\begin{aligned}\epsilon_{1NN}(\mathbf{x}) &= p(y_1|\mathbf{x})p(y_2|\mathbf{x}') + p(y_2|\mathbf{x})p(y_1|\mathbf{x}')\\ \text{(if } n \text{ is large)} &\approx p(y_1|\mathbf{x})p(y_2|\mathbf{x}) + p(y_2|\mathbf{x})p(y_1|\mathbf{x})\\ &= 2p(y_{min}|\mathbf{x})(1 - p(y_{min}|\mathbf{x})) \leq 2p(y_{min}|\mathbf{x}) = 2\epsilon^*(\mathbf{x})\end{aligned}$$

□

We can deduce that:

In the discrete case

$$\epsilon^* = \sum_{x \in \mathcal{X}} p(y_{min}|\mathbf{x}).p(\mathbf{x})$$

$$\epsilon_{1NN} = \sum_{x \in \mathcal{X}} \left(p(y_1|\mathbf{x})p(y_2|\mathbf{x}') + p(y_2|\mathbf{x})p(y_1|\mathbf{x}')\right).p(\mathbf{x}) \leq 2\epsilon^*$$

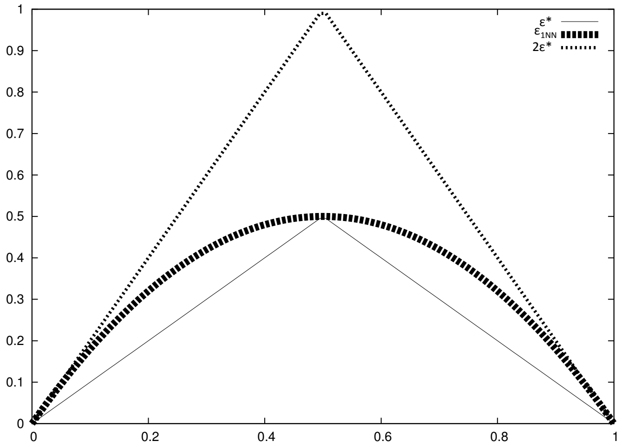
In the continuous case

$$\epsilon^* = \int_{x \in \mathcal{X}} f(y_{min}|\mathbf{x}).f(\mathbf{x})d\mathbf{x}$$

$$\epsilon_{1NN} = \int_{x \in \mathcal{X}} \left(f(y_1|\mathbf{x})f(y_2|\mathbf{x}') + f(y_2|\mathbf{x})f(y_1|\mathbf{x}')\right).f(\mathbf{x})\, d\mathbf{x} \leq 2\epsilon^*$$

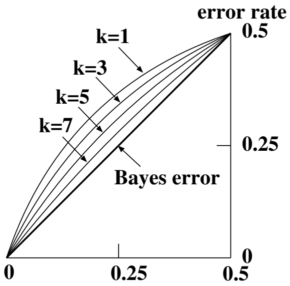
Graphically

The property $\epsilon^* \leq \epsilon_{1NN} \leq 2\epsilon^*$ can be graphically illustrated



Effect of k on the estimation quality

$$\epsilon^* \leq \epsilon_{kNN} \leq \epsilon_{(k-1)NN} \leq \ldots \leq \epsilon_{1NN} \leq 2\epsilon^*$$



The previous property holds only asymptotically $\rightarrow$ possible compromise between k and n: $k = \sqrt{\frac{n}{|\mathcal{Y}|}}$.

Problems of k-NN

- To converge, the k-nearest neighbor algorithm requires a large number of training examples.
- However, a large number of training examples implies a large space/time complexity.

Two strategies to overcome these problems:

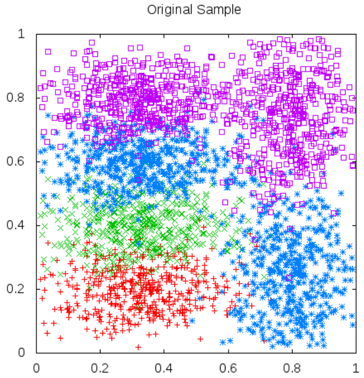
- Reduce the size of S while keeping the most relevant examples (*e.g.* the condensed nearest neighbor rule (Hart 1968)).
- Simplify the calculation of the nearest-neighbor.

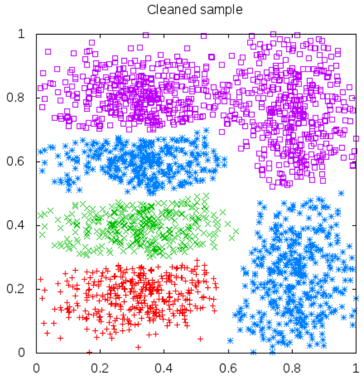
Data reduction techniques

Preliminary step: remove from S the outliers and the examples of the bayesian error region.

```
Input: $S$
Output: $S_{cleaned}$
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 misclassified instances;
    Classify $S_2$ with the new set $S_1$ using the 1-NN rule;
    Remove from $S_2$ the misclassified instances;
$S_{cleaned} = S_1 \cup S_2$;
```

Illustration



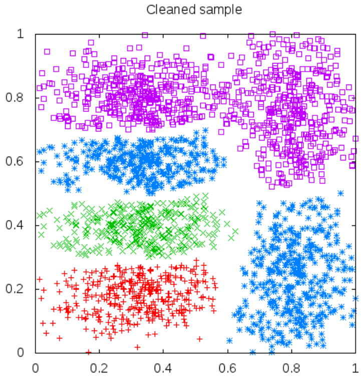


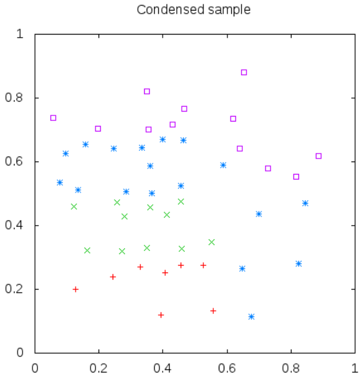
The condensed nearest neighbor rule (CNN)

Second step: remove the irrelevant examples.

```
Input: $S$
Output: STORAGE
STORAGE $\leftarrow \emptyset$ ; DUSTBIN $\leftarrow \emptyset$;
Draw randomly a training example from $S$ and put it in STORAGE;
while no stabilization of STORAGE do
    foreach $\mathbf{x_i} \in S$ do
        if $\mathbf{x_i}$ is correctly classified with STORAGE using the 1-NN rule
        then
            DUSTBIN $\leftarrow \mathbf{x_i}$
        else
            STORAGE $\leftarrow \mathbf{x_i}$
return STORAGE;
```

Illustration



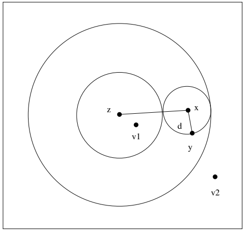


How to speed-up the nearest-neighbor calculation?

Most of the approaches are based on the triangle inequality property

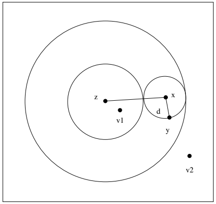
$$\forall(\mathbf{x},\mathbf{y},\mathbf{z}) \in \mathcal{X}^3, d(\mathbf{x},\mathbf{z}) \leq d(\mathbf{x},\mathbf{y}) + d(\mathbf{y},\mathbf{z}).$$

- Let **x** be the example to classify by the NN rule. Let **y** be the current NN of **x** at a distance δ.
- Let **z** be the next example. If $d(\mathbf{x},\mathbf{z}) \leq \delta$ then update the current NN. Otherwise, remove the following examples:
    1. in the sphere centered at **z** and of radius $d(\mathbf{x},\mathbf{z}) - \delta$,
    2. out of the sphere centered at **z** and of radius $d(\mathbf{x},\mathbf{z}) + \delta$,



How to speed-up the nearest-neighbor calculation?

- $d(\mathbf{v_1}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{z}) - \delta \Rightarrow d(\mathbf{v_1}, \mathbf{z}) + \delta \leq d(\mathbf{x}, \mathbf{z})$ (1).
- By the triangle inequality, we know that $d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{v_1}) + d(\mathbf{v_1}, \mathbf{z})$ (2).
- From (1) and (2) we get $d(\mathbf{v_1}, \mathbf{z}) + \delta \leq d(\mathbf{x}, \mathbf{v_1}) + d(\mathbf{v_1}, \mathbf{z}) \Rightarrow \delta \leq d(\mathbf{x}, \mathbf{v_1})$
- Therefore, $\mathbf{v_1}$ cannot be the NN of $\mathbf{x}$. The same proof can be used for $\mathbf{v_2}$.



Some methods about fast kNN algorithms

- In 2D or 3D: graph-based searching methods such as Voronoi diagram and proximity graph.
- In higher spaces: ball-trees, kd-trees, metric-trees, quadtree, R-trees, k-Means-k-NN.

Conclusion

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