Exact Cross-Validation for kNN and applications to passive and active learning in classification.

Alain CELISSE
Laboratoire de Mathmatique Paul Painlevé
UMR 8524 CNRS - Université Lille 1,
59 655 Villeneuve d'Ascq Cedex, France.
celisse@math.univ-lille1.fr

Tristan MARY-HUARD
UMR AgroParisTech/INRA MIA 518
16 rue Claude Bernard,
75 231 Paris Cedex 05, France.
maryhuar@agroparistech.fr

Abstract

A closed-form expression of the LpO risk estimator for kNN is derived. It is first used for choosing k_p by LpO risk minimization in the passive learning setting. The impact of p on the choice of k and the LpO estimation of the risk is inferred. In the active learning setting, a procedure is proposed, where new examples are requested using a LpO committee of kNN classifiers. The influence of p on the choice of new examples and the tuning of k at each step is investigated. In particular, the behavior of k_p is different from what is observed in passive learning.
Keywords: Classification, Cross-validation, kNN, active learning.

1 Introduction

Classification We consider the binary classification framework, where the goal is to predict the unknown label $Y = 0,\ 1$ of an observation X. To this purpose, one aims at building from data $D = (X_1, Y_1), ..., (X_n, Y_n)$ a classifier $f : \mathcal{X} \to \{0, 1\}$ whose classification error rate

$$L(f) = P(f(X) \neq Y|D)$$

is as low as possible. The risk of a classifier f is defined as $R(f) = \mathbb{E}_D\left[P(f(X) \neq Y \mid D)\right]$.

kNN The k Nearest Neighbor algorithm (kNN, [**?**, **?**]) is a very popular algorithm designed for this problem, that has been successfully applied to many difficult classification tasks [**?**, **?**]. The principle of the kNN algorithm is simple: first, for a given observation x_0 to classify, find $X_{(1)}, ...X_{(k)}$ the k closest points to x_0 in the training set, then classify x_0 according to a majority vote decision rule among these k neighbors. A variant is the Weighted kNN algorithm (WkNN), where the weight of each neighbor in the majority vote decision rule depends on its proximity to x_0 (the closer to x_0, the higher the weight).

Cross-validation Cross-validation (CV, [**?**]) is a widespread strategy to assess the performance of a classifier, or to tune the inner parameters of a classification algorithm. The main idea behind CV is to split data: part of data (the training sample) is used for training the algorithm, and the remaining part (the validation sample) is used for estimating the classification error rate of the algorithm. Then, CV selects the algorithm with the smallest averaged classification error rate. There are several ways to implement the CV strategy, but we only consider two of them:

- $\star$ K-fold CV (KCV): the complete dataset is divided into K subsamples with equal size n/K, and each subsample is successively used for validation,
- $\star$ Leave-p-out (LpO): every possible subset of p observations is successively left out of the sample and used for validation.

In practice, because of its prohibitive computational cost, the LpO procedure is almost never applied except with $p = 1$. As an alternative, the KCV procedure is used as a surrogate of LpO, to the cost of a higher variability due to the arbitrary splitting of the complete dataset into K independent subsamples. Applied to kNN, CV can be used to select the value of parameter k, or to evaluate the performance of the final kNN classifier.

Active Learning In active learning, the learning algorithm is allowed to select the data from which it learns, in order to speed up its performance [**?**]. In the pool-based sampling scenario, a pool of unlabeled observations is available, along with a small sample of labeled data. The goal is to identify which observations of the pool should be added to the training set to achieve optimal performance. Among many strategies to select unlabeled observations, the query-by committee (QbC) approach is quite popular and has shown encouraging results ([**?**, **?**]). QbC consists in consulting a committee of classifiers to predict the label of the unlabeled observations, and to select the observations for which the committee classifiers most disagree. The committee can be constituted of classifiers obtained by applying the same classification algorithm to different training sets.

Contribution The rest of the paper is organized as follows. Section 2 describes an efficient calculation of the LpO estimator for kNN and WkNN classifiers. These closed-form expressions enable the practical use of LpO. Section 3 is devoted to passive learning. In particular, the behavior of the minimizer k_p of the LpO estimator is investigated with respect to n and p. It is shown that the choice of p is crucial for choosing k, unlike what happens for estimating the risk of a given kNN classifier. Finally, in Section 4, a procedure called LpO-QbC is proposed in the active learning setting. The influence of p is also experimentally analyzed at each step of the LpO-QbC procedure. In particular, the optimality of L1O is empirically shown for selecting the examples to request from the pool.

2 Exhaustive cross-validation for kNN

In the present section, the computational burden of the LpO procedure is drastically reduced when considering the binary classification framework and kNN classifiers. First, a conditioning argument is used to reduce the computational time from $\mathcal{O}(\binom{n}{p})$ to $\mathcal{O}(np \times \binom{k+p}{k})$. Then, weighted and non-weighted kNN algorithms are considered successively.

2.1 Conditioning

Let $(x_1, y_1), ..., (x_n, y_n)$ denote the data. Each step of the LpO procedure splits this set into a *training sample* e of size $n-p$, and a *validation sample* $\bar{e}$ of size p. Let f^e denote the kNN classifier built from e, and $\mathcal{E}$ be the set of all possible training samples. Set R_{LpO} the estimation of the kNN algorithm performance based on LpO:

$$R_{LpO}(k) = \binom{n}{p}^{-1} \sum_{e \in \mathcal{E}} \left(\frac{1}{p} \sum_{i \notin e} \mathbb{I}_{\{f^e(x_i) \neq y_i\}} \right) \quad . \tag{1}$$

For a given point i that belongs to the test set, V_k^i denotes the rank of its associated k^{th} neighbor in training set e. Note that V_k^i is a random variable since it depends on the splitting.

Proposition 1 *Let $(E, \bar{E})$ represent a random splitting of the data into 2 subsamples of respective size $n-p$ and p. Then,*

$$R_{LpO}(k) \quad = \quad \sum_{i=1}^{n} \underbrace{P\left(i \in \bar{E}\right)}_{A1} \sum_{j=k}^{k+p-1} \underbrace{P\left(V_k^i = j | i \in \bar{E}\right)}_{A2} \underbrace{P\left(f^E(x_i) \neq y_i | i \in \bar{E},\, V_k^i = j\right)}_{A3} \tag{2}$$

Let us now consider each term $A1$, $A2$ and $A3$. We start with $A1$:

$$\forall i \in [1,n],\ P\left(i \in \bar{E}\right) = \frac{p}{n}\ .$$

Then, for $A2$, one has

$$\begin{aligned} \forall i \in [1,n],\ P\left(V_k^i = j | i \in \bar{E}\right) &= \frac{\binom{j-1}{j-k}\binom{n-j-1}{p-1-j+k}}{\binom{n-1}{p-1}} \\ &= \frac{k}{j} P\left(U = j-k\right)\ , \end{aligned}$$

where $U \hookrightarrow \mathcal{H}(j, n-j-1, p-1)$ and $\mathcal{H}(a,b,c)$ denotes the hypergeometric distribution with a the number of white balls, k the number of black balls and c the number of balls to draw.

Note that neither $A1$ nor $A2$ actually depend on i, which only appears in A_3. To evaluate how difficult the computation of this last term is, let us consider the ordered sequence $X^i_{(1)}, ..., X^i_{(n-1)}$ corresponding to the neighbors of i in the whole sample. Since p observations (including i) are removed at a given step of the LpO procedure, the k neighbors of i belong to $\{X^i_{(1)}, ..., X^i_{(k+p-1)}\}$. Once this list is obtained (by applying the $(k+p-1)$NN algorithm to the entire set), one only needs to compute the number of times (over all the splittings) the majority label corresponds to the label of observation i, for each value of j. Therefore, the computational burden of $A3$ is at most of order

$$\mathcal{O}\left(n \sum_{j=k}^{k+p-1} \binom{j}{k-1}\right) < \mathcal{O}\left(np\binom{k+p}{k-1}\right)\ ,$$

that is linear in n. Let us now consider $A3$ for weighted and then non-weighted kNN algorithms.

2.2 Weighted kNN

The weighted kNN rule is defined as

$$f_{WkNN}(x) = \begin{cases} 1 & \text{if } \sum_{\ell=1}^k w^x_{(\ell)} I_{\{Y_{(\ell)}=1\}} > \sum_{\ell=1}^k w^x_{(\ell)} I_{\{Y_{(\ell)}=0\}} \\ 0 & \text{otherwise}\ , \end{cases}$$

where $w^x_{(\ell)}$ is the weight associated with the ℓ^{th} neighbor of x. Usually, the weight depends on the distance between x and its ℓ^{th} neighbor. The previous classifier can be rewritten as

$$f_{WkNN}(x) = \begin{cases} 1 & \text{if } \sum_{\ell=1}^k w^x_{(\ell)} I_{\{Y_{(\ell)}=1\}} - \sum_{\ell=1}^k w^x_{(\ell)} I_{\{Y_{(\ell)}=0\}} > s \\ 0 & \text{otherwise}\ , \end{cases}$$

where the threshold s is chosen to be 0.

$A3$ corresponds to the frequency at which observation i is misclassified conditionally to the fact that its j^th neighbor in the *complete sample* is its k^{th} neighbor in *training set* E. Once this conditioning is done, the k neighbors of i belong to the list $\{X^i_{(1)}, ..., X^i_{(j)}\}$. Let us define W_0 and W_1:

$$\begin{aligned} W_0 &= \{w^i_{(\ell)} \mid X^i_{(\ell)} \in \{X^i_{(1)}, ..., X^i_{(j)}\} \text{ and } Y_{(\ell)} = 0\} \\ W_1 &= \{w^i_{(\ell)} \mid X^i_{(\ell)} \in \{X^i_{(1)}, ..., X^i_{(j)}\} \text{ and } Y_{(\ell)} = 1\}\ . \end{aligned}$$

One has

$$A3 = P\left(f^E(x_i) \neq y_i | i \in \bar{E},\ V^i_k = j\right) = \frac{N(W_0, W_1, k, s)}{\binom{j}{k}}\ ,$$

where $N(W_0, W_1, k, s)$ is the number of combinations of k elements selected in list $\{W_0, W_1\}$ such that $\sum_{\ell \in W_1} w^i_{(\ell)} - \sum_{\ell \in W_0} w^i_{(\ell)} > s$. Let us assume without loss of generality that $w^i_{(1)}$ belongs to set W_0. Then,

$$N(W_0, W_1, k, s) = N\left(W_0 \setminus \{w^i_{(1)}\}, W_1, k-1, s - w^i_{(1)}\right) + N\left(W_0 \setminus \{w^i_{(1)}\}, W_1, k, s\right)\ .$$

The computation of $N(W_0, W_1, k, s)$ can be obtained using recursive programming. An algorithm is proposed along with some considerations about computational complexity (see Suppl. Mat.). Interestingly, the computational burden associated with the proposed algorithm decreases with the noise level.

2.3 Non-weighted kNN

In the particular case where all neighbors receive the same weight in the majority vote classification rule, the computation of term $A3$ is straightforward. Let n_j^i be the number of 1s among the j neighbors of i in the original sample. The quantity n_j^i can be obtained for all i and $j \in [1, k+p-1]$ by running the $(k+p-1)$NN algorithm. We have:

$$P\left(\Phi^E(x_i) \neq y_i | i \in \bar{E},\ V_k^i = j\right) = \mathbb{I}_{\{y_i=0\}} P\left(\Phi^E(x_i) = 1 | i \in \bar{E},\ V_k^i = j\right) + \mathbb{I}_{\{y_i=1\}} P\left(\Phi^E(x_i) = 0 | i \in \bar{E},\ V_k^i = j\right) \ .$$

Let N_i^E be the number of 1 among the k nearest neighbors of i in sub-sample E, and N_i^j the number of 1 among the j nearest neighbor of i in the complete training set. Assuming k is odd for sake of simplicity, one obtains:

$$\begin{aligned} P\left(\Phi^E(x_i) = 1 | i \in \bar{E},\ V_k^i = j\right) &= P\left(N_i^E \geq k/2 | i \in \bar{E},\ V_k^i = j\right) \\ &= \mathbb{I}_{\{y_j=0\}} \left(1 - F_H\left(\frac{k+1}{2}\right)\right) + \mathbb{I}_{\{y_j=1\}} \left(1 - F_{H'}\left(\frac{k-1}{2}\right)\right) \ , \end{aligned} \tag{3}$$

where $H \hookrightarrow \mathcal{H}(N_i^j, j - N_i^j - 1, k-1)$ and $H' \hookrightarrow \mathcal{H}(N_i^j - 1, j - N_i^j, k-1)$.

Similar formulas can be derived for $P\left(\Phi^E(x_i) = 0 | i \in \bar{E},\ V_k^i = j\right)$. This shows that in the case of equal weights, the LpO procedure for the kNN algorithm can be performed at the same computational cost as L1O, whatever p.

3 Passive Learning

Using kNN classifiers in passive learning requires to choose k. This can be done using LpO. We define for every $1 \leq p \leq n$

$$k_p = \arg \min_{1 \leq k \leq n} R_{LpO}(k) \ .$$

In the specific case $p = 1$, some theoretical results exist on the asymptotic behavior of k_1 with respect to n. Having access to exact LpO enables to further infer the relationship between p and k_p, at least to a practical point of view.

3.1 Influence of n on k_p, p fixed

Calculations of Section 2 on the LpO estimator allow to study k_p with respect to n for various values of p. A simulation study has been carried out to infer the behavior of k_p, following the simulation scheme described in Section 4.3.
Figure 1 (left) displays k_p with respect to n for $q = 0.2$, and gives a representative picture of the results. It shows that k_p is sub-linear with respect to n as long as p is kept independent of n. Since it is known that kNN estimators are consistent as long as $k = o(n)$ [**?**], it leads us to conjecture that the kNN estimator computed from k_p neighbors, with p fixed, is consistent.

3.2 Influence of p on k_p, n fixed

When several estimators are available, choosing the best one is a classical issue of statistics. Model selection is a typical strategy aiming at addressing this question. Choosing the number k of neighbors involved in the definition of the kNN estimator enters into this setting.
Several instances of model selection, in density estimation or in regression ([**?**]), have related the parameter p of LpO to the signal-to-noise ratio (SNR). With a large enough SNR, small values of p lead to efficient model selection strategies, whereas a small SNR requires larger values of p. A similar conclusion is supported by the simulation experiments.

We first observe that increasing p entails a smaller choice of k_p (see Figure 1 and Suppl.Mat.). This phenomenon is also observed with several noise levels from $q = 0.1$ up to $q = 0.4$.

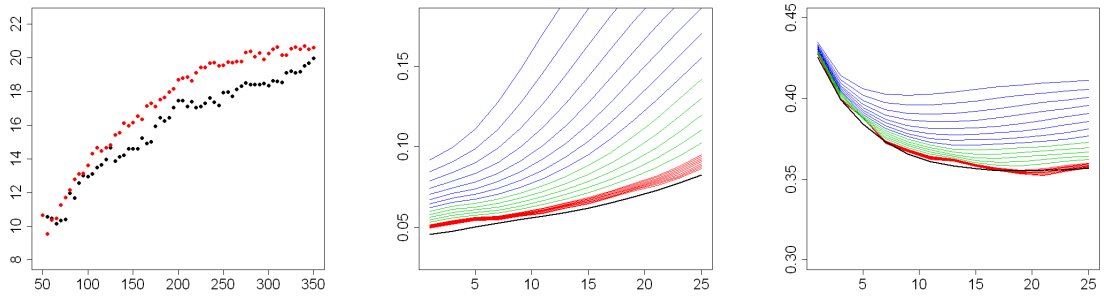

Figure 1: **Left:** Evolution of k (in ordinates) at each step of the passive learning, $q = 0.2$. **Center:** Plot of the average performance (in ordinates) evaluated by LpO with different p (colored curves) or on test sample (black curves), for different values of k (in abscisses) and for noise level $q = 0$. Red curves correspond to values of p lower than 20, green curves to values of p between 20 and 80, and blue curves to values higher than 80. **Right:** Same representation as previous, for noise level $q = 0.2$.

Second, it is necessary to choose p larger than 1 as long as the noise is not null. Indeed, LpO with small values of p leads to choose too large values of k when the noise is not null. This observation is supported by Figure 1 (right), where the minimum locations of red curves, which correspond to small values of ps, are larger than that of the black curves, which displays the true risk. The same phenomenon occurs with a noise level $0.1 \leq q \leq 0.4$.
However, Figure 1 indicates that larger values of p balance this trend. For some values of p larger than 1 (blue curves), the minimum location is close (or equal) to the best possible k. This suggests that a convenient choice of $p > 1$ is required to provide a reliable k_p.

3.3 Risk estimation

In many applications, one is also interested in a sharp estimation of the performance of a given classifier (see Section 4.4). Due to the computational cost of LpO, this performance is often estimated with $p = 1$. One can wonder whether higher values of p should yield better results.

First, Figure 1 shows that large values of p (blue curves) lead to biased estimations of the true risk (black curve). In other frameworks ([**?**, **?**]), CV is known to be all the more biased as p is large.

Second, these theoretical considerations entail that the least biased LpO estimator is obtained with $p = 1$. Figure 1 supports this conclusion since, for a fixed k, small values of p remain close to the black curve. Note that, depending on the noise level, larger values of p can also lead to reliable estimates of the true performance.

Third, an important aspect arising from the case $q = 0$ is that *model selection* and *risk estimation* can be contradictory objectives. All values of p lead to choose $k = 1$ from a model selection point of view. However, only $p = 1$ yields a (nearly) unbiased estimation of the risk.

4 Active Learning

Active learning differs from passive learning by the opportunity for a learning algorithm to select the data from which it learns. The goal is to learn a classification rule from as few examples as possible.

4.1 LpO-QbC active learning

We consider the pool based sampling scenario, where a small training set of size $n^{(0)}$ and a large pool of unlabeled examples are available. At each round ℓ, one can request m examples from the pool. These are labeled and added to the original training set, which grows to size $n^{(\ell)}$. Therefore, active learning crucially depends on the strategy used to choose the "best" m examples to add.

Since the work of [**?**], the query by committee strategy (QbC) has been widely investigated. From a large committee of classifiers, each classifier predicts the label of every point of the pool. The "best" m points are points for which the committee classifiers most disagree.

In the present work, the LpO-QbC algorithm is proposed. At round ℓ, this active learning algorithm alternates two steps:

- *Point selection step*: a LpO committee of $\binom{n^{(\ell-1)}}{p}$ kNN classifiers is constituted from the $n^{(\ell-1)}$ training examples. Then, m examples are selected from the computation of the agreement $A_{LpO}(x)$ (see Section 4.2), computed for each point x of the pool.
- *Tuning step*: a new set of kNN classifiers $\{f_{kNN}\}_k$ is computed from the $n^{(\ell)} = n^{(\ell-1)}+m$ training examples. Since the training set grows at each round, the choice of k is tuned by minimizing the LpO estimator over k.

Note that LpO is used twice. At step 2, LpO is used for choosing k, that is to perform *model selection* (see Section 2 for an efficient computation, and Section 3 about the influence of p on the tuning of k). At step 1, LpO is used to build the committee and select the examples on the basis of their agreements, which amounts to *risk estimation*.

4.2 Agreement of the kNN LpO-committee at a new point

The present section defines the agreement $A_{LpO}(x)$ at example x, and its efficient computation using the same trick as in Section 2.

For any committee $\mathcal{C}$ of classifiers $\{f^1, ..., f^N\}$, let us define the agreement $A_{\mathcal{C}}(x)$ at any (unlabeled) example x

$$A_{\mathcal{C}}(x) \quad := \quad 2 \times \left| \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}_{\{f^i(x) \neq 0\}} - \frac{1}{2} \right| \quad ,$$

where the label of x is fixed to 0, without any incidence on the agreement between the classifiers. If the average misclassification rate at point x is close to 0.5, then half of the $\mathcal{C}$-committee classified x as 0, and $A_{\mathcal{C}}(x)$ is close to 0.

At round ℓ, the LpO committee is the set of kNN classifiers f^e built from a subsample e of $n^{(\ell)} - p$ examples drawn from the $n^{(\ell)}$ training examples. The agreement of the LpO committee at point x is denoted by $A_{LpO}(x)$. It can be efficiently computed with the same trick as in Section 2:

$$\binom{n^{(\ell)}}{p}^{-1} \sum_{e} \mathbb{I}_{\{f^e(x_i) \neq 0\}} = \sum_{j=k}^{k+p} \frac{k}{j} P\left(U = j-k\right) \times \left[\mathbb{I}_{\{y_j=0\}} \left(1 - F_H\left(\frac{k+1}{2}\right)\right) + \mathbb{I}_{\{y_j=1\}} \left(1 - F_{H'}\left(\frac{k-1}{2}\right)\right)\right] \quad ,$$

where $U \hookrightarrow \mathcal{H}(j, n^{(\ell)} - j, p)$, $H \hookrightarrow \mathcal{H}(N_0^j, j - N_0^j, k-1)$, and $H' \hookrightarrow \mathcal{H}(N_0^j, j - N_0^j, k-1)$. N_0^j denotes the number of 1s among the j nearest neighbors of x in the training set.

4.3 A short illustration of the LpO-QbC strategy

The goal of the present section is to assess the performance of LpO-QbC, compared with a passive learning algorithm based on LpO (p-LpO). In particular, as an active learning algorithm, LpO-QbC should mainly select examples in regions where the classification task is difficult. LpO-QbC is also expected to improve on p-LpO in terms of final error rate. These two aspects of LpO-QbC are inferred on simulated and real data.

For both types of data, one starts with a training set E^0 and a pool set P^0. At each round ℓ, $m = 5$ new examples are selected from the pool using LpO-QbC with $p = 10$. Then a kNN classifier is trained on $E^{(\ell)}$, with k chosen by LpO minimization with $p = 10$. This process (point selection and tuning step) is repeated 60 times, that is $0 \leq \ell \leq 60$.

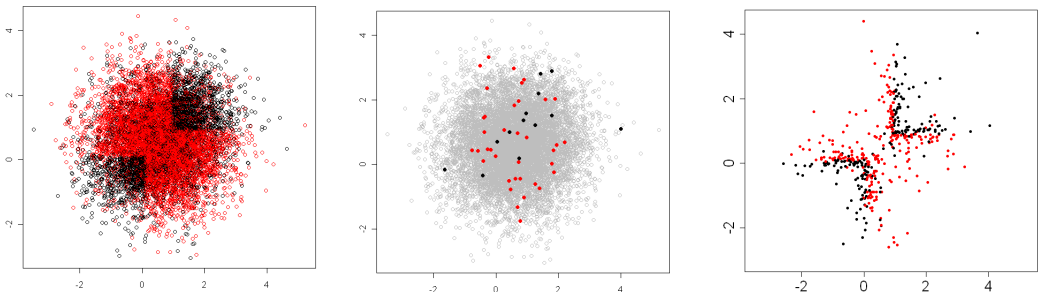

Figure 2: Example of pool sample P^0 (left), starting training set E^0 (center), and final training set E^{60} after 60 steps of active learning (right) for $q = 0.2$. Red and black colors indicate the label.

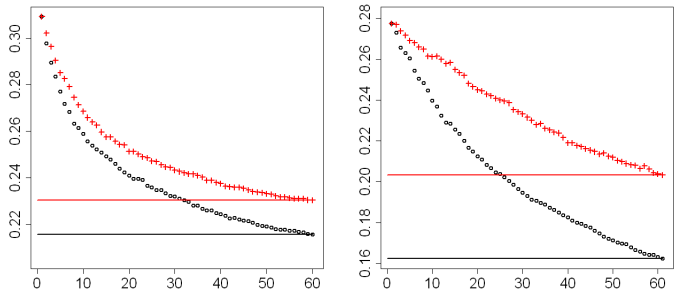

Figure 3: Performance of LpO-QbC (o) and passive kNN (+) on simulated data (left), and Spam data (right).

Simulations 2-dimension data are simulated. $X = (X^1, X^2)$ is generated using a mixture of 3 Gaussian distributions, with proportions $(0.2, 0.4, 0.2)$, means $(0.25, 0.25)$, $(0.5, 0.75)$, $(0.75, 0.75)$, and common covariance matrix I_2. The label Y is generated conditionally to X: if $(X^1 < 0.2$ and $X^2 < 0.2)$ or $(X^1 > 0.8$ and $X^2 > 0.8)$ then $P(Y = 1|X) = q$, otherwise $P(Y = 1|X) = 1 - q$. Several noise levels are considered: 0, 0.1, 0.2, 0.3, and 0.4. 100 repetitions of each condition have been considered. The initial sizes of training and pool sets are 50 and 10.000, respectively.

Figure 2 shows these 2 sets for a round simulated with $q = 0.2$, and the final training set E^{60} obtained after 60 steps of LpO-QbC active learning. This confirms the good behavior of LpO-QbC that mainly focuses on examples lying on the boundary between the 2 classes. This was also observed for other values of p and q (not shown).

According to Figure 3 (left), after 60 rounds, LpO-QbC achieves a misclassification rate of 21.5%, whereas the LpO-based passive (p-LpO) learning algorithm achieves 23%. Clearly, LpO-QbC outperforms p-LpO.

Spam data The Spam data consists of 4601 observations and 16 variables (only variables listed in [**?**], p.264, are used). At each round, 50 observations are randomly selected to form the training set E^0. Remaining examples form the pool P^0.
On these data, LpO-QbC and p-LpO achieve an average performance of 20.2%, and respectively 16.2% for a final training set of size 350 (Figure 3 (right)). As for simulated data, LpO-QbC focuses on examples close to the border between the two classes (not shown) and outperforms p-LpO.

4.4 A deeper analysis of LpO-QbC

In this section, we investigate the influence of p on each of the two steps of LpO-QbC. The following results comes from the data simulated in Section 4.3.

Point selection step To evaluate the influence of p at step 1, i.e. the influence of the committee size, k is fixed at 7 during the tuning step, so that LpO is only used for *point selection*. Different

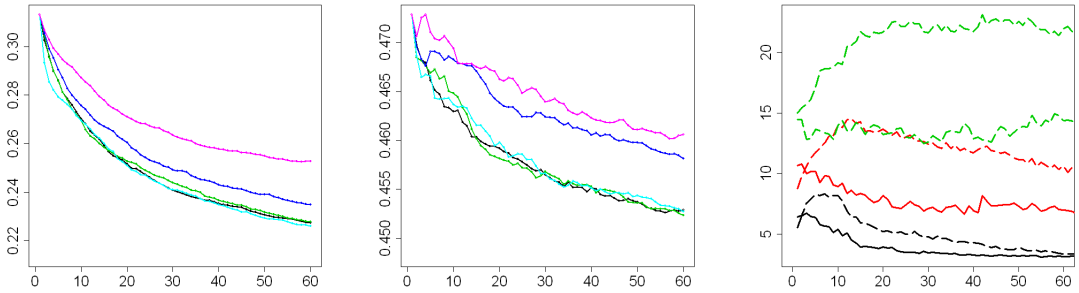

Figure 4: **Left:** Performance of the committees for $k = 1$ (cyan), 5 (black), 10 (green), $n^{(\ell)}/2$ (blue) and passive learning (pink), $q = 0.2$. **Center:** Performance of the committees and passive learning, $q = 0.4$. **Right:** Evolution of k at each step of the active learning, for the LpO-QbC strategy (solid) and the conditional oracle (dashed), with $q = 0.1$ (black), 0.2 (red) and 0.3 (green).

values of p are considered, from 1 to $n^{(\ell)}/2$. For instance, note that $p = 1$ corresponds to the smallest committee with only $n^{(\ell)}$ kNN classifiers.
According to Figure 4, the size of the LpO committee has no or little impact: the smallest committee ($p = 1$) performs as well as larger ones ($p = 2, 5, 10$ or 20), whatever the level of noise. However, when the size is raised to $p = n^{(\ell)}/2$, the performance strongly deteriorates.
Quantifying agreement $A_{LpO}(x)$ at every example x of the pool amounts to accurately estimate $P(Y = 1|X = x)$. This is strongly related to the *risk estimation* issue (see Section 3.3), where it is known that small values of p provide the best estimators in terms of bias-variance trade-off. These theoretical considerations confirm the phenomena observed in Figure 4.

Tuning step At step 2, LpO is used for choosing k, which amounts to *model selection*. In the present experiments, p is fixed at 10.
From Figure 4, the evolution of k_p as a function of n is completely different from that of Figure 1: k_p remains constant or decreases.
Let us introduce the *conditional oracle* (dashed line), which is the best possible choice of k knowing the truth and given the training set E^ℓ. Two stages are observed in the evolution of the conditional oracle. First, the conditional oracle explores the complete space to identify the boundaries: k_{oracle} increases. Second, once the boundaries are found, it focuses on examples close to the boundary: smaller values of k are selected. Compared with Figure 1, this illustrates the specificity of the model selection problem in the context of active learning.
The evolution of k_{oracle} sheds light on the evolution of k_p and the reason why it does not grow with n. However, for a fixed p, the choice of k by LpO leads to overfitting: smaller values than k_{oracle} are always selected.

5 Discussion

In applications of kNN to real data, LpO is used either to assess the performance of a kNN classifier (risk estimation), or to choose k (model selection). In both cases, p is usually equal to 1 for computational reasons. In the model selection setting, there is no guideline for practitioners about the relationship between p and k_p, or about the relevance of the selected value k_1. From a theoretical point of view, relating the optimal p to the signal-to-noise ratio and the size of the training set is of great interest.
The closed-form expressions derived for the LpO estimator associated with kNN and WkNN classifiers yield an efficient and practical tool to study the behavior of k_p, both for theoretical and practical purposes. Exact LpO should be preferred to its classical surrogate KCV, since LpO is less variable (with $K = n/p$).

In passive learning, some theoretical results already exist about the application of L1O to kNN [**?**]. But to the best of our knowledge, there is no such result for the general LpO procedure. The present simulation study is a preliminary work before the theoretical analysis of LpO in the passive learning

setting. Some further work is required to get more insight toward a data-driven calibration of p.

In active learning, LpO is used for point selection (when considering the QbC strategy), and for parameter tuning. First, point selection does not exhibit any strong dependence on p, which validates the use of L1O at this step. Conversely, p can be crucial for choosing k conveniently, as for passive learning. At each step, since requested points are not randomly chosen, the classical theory of model selection provides very few guidelines toward an effective selection of k. In this context, data driven procedures such as LpO are all the more attractive.

References