Journal de la Société Française de Statistique
Vol. 152 No. 3 (2011)

Exact Cross-Validation for kNN : application to passive and active learning in classification

Titre: Validation-croisée exacte pour les kNN : application à l'apprentissage passif et actif en classification

Alain Celisse[1] and Tristan Mary-Huard[2]

Abstract: In the binary classification framework, a closed form expression of the cross-validation Leave-p-Out (LpO) risk estimator for the k Nearest Neighbor algorithm (kNN) is derived. It is first used to study the LpO risk minimization strategy for choosing k in the passive learning setting. The impact of p on the choice of k and the LpO estimation of the risk are inferred. In the active learning setting, a procedure is proposed that selects new examples 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. The behavior of k chosen by LpO is shown to be different from what is observed in passive learning.

Résumé : Pour l'algorithme de classification des k plus proches voisins (kNN), une expression explicite de l'estimateur du taux d'erreur de classification par validation croisée Leave p Out (LpO) est proposée. Cette expression explicite est d'abord utilisée dans le cadre de l'apprentissage passif pour étudier l'impact du choix du paramètre p du LpO sur le choix de k dans l'algorithme kNN. On s'intéresse ensuite au problème de l'apprentissage actif (active learning). Une procédure de sélection des exemples basée sur la recommandation du comité des classificateurs LpO est considérée. L'influence du paramètre p sur le choix des nouveaux exemples et sur le choix du paramètre k à chaque étape de l'apprentissage actif est étudiée. En particulier, il est montré que l'évolution de la valeur du paramètre k choisie par LpO en apprentissage actif est différente de celle observée en apprentissage passif.

Keywords: Classification, Cross-validation, kNN algorithm, Active learning
Mots-clés : Classification, Valildation-croisée, kNN, Apprentissage actif
AMS 2000 subject classifications: 62H30, 62G09, 68T10

1. Introduction

Classification We consider the binary classification framework, where the goal is to predict the unknown label $Y \in \{0,1\}$ of an observation X. In the following, Z represents a random variable and z its realization. To this purpose, one aims at building from data $D = (X_1,Y_1),\dots,(X_n,Y_n)$ a classifier $f : \mathscr{X} \to \{0,1\}$ whose classification error rate

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

is as low as possible, where $P(\cdot \mid D)$ denotes the probability with respect to (X,Y) given D. The risk of a classifier f is defined as $R(f) = \mathbb{E}_D\left[P(f(X) \neq Y \mid D)\right]$.

1. Laboratoire de Mathématiques Painlevé, UMR 8524 CNRS – Université Lille 1.
E-mail: celisse@math.univ-lille1.fr
2. UMR AgroParisTech/INRA MIA 518.
E-mail: maryhuar@agroparistech.fr

*k***NN** The k Nearest Neighbor algorithm (kNN, [4, 5]) is a very popular algorithm designed for this problem, that has been successfully applied to many difficult classification tasks [6, 12]. The principle of the kNN classifier is simple: first, for a given observation x to classify, find $X_{(1)}, \ldots X_{(k)}$ the k closest points to x in the training set, then classify x according to a majority vote decision rule among these k neighbors. A variant is the Weighted kNN classifier (WkNN), where the weight of each neighbor in the majority vote decision rule depends on its proximity to x (the closer to x, the higher the weight).

Cross-validation Cross-validation (CV, [13]) is a widespread strategy to assess the performance (in term of classification error rate) of a classifier, or to tune the inner parameters of a classification algorithm. The 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:

- ⋆ 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,
- ⋆ 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 $p = 1$ where it amounts to the well known *leave-one-out*. As an alternative, the KCV procedure is used as a surrogate of LpO, at 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 [9]. 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 promizing results ([8, 10]). QbC consists in consulting a committee of classifiers (*experts*) 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 a new efficient calculation strategy of the LpO estimator for kNN and WkNN (weighted kNN) classifiers. These closed-form expressions enable the practical use of LpO for kNN classifier at almost the same algorithmic cost as standard empirical risk minimization as long as p remains not too large with respect to n. Section 3 is devoted to passive learning. The behavior of the minimizer k_p of the LpO estimator is investigated with respect to the sample size n and parameter p. In particular, 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 *k*NN

In this section, we show how the computational burden of the LpO procedure for kNN and WkNN can be drastically reduced in the binary classification framework. The derivation is split into two parts. First, a conditioning trick is used to reduce the computational time from $\mathscr{O}(\binom{n}{p})$ to $\mathscr{O}(np \times \binom{k+p}{k})$. Second, weighted and non-weighted kNN classifiers are successively considered.

2.1. Conditioning

Let $(x_1,y_1),\dots,(x_n,y_n)$ denote the complete set of 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 $\mathscr{E}$ the set of all possible training samples. Set $R_{LpO}(k)$ the estimation of the kNN performance based on LpO:

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

For a given point i in the validation set $\bar{e}$, let V_k^i denote the rank of its associated k^{th} neighbor in training set e. We have

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

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

Note that V_k^i is a random variable since it depends on the random splitting $(E,\bar{E})$.

Proof.

$$\begin{aligned}
R_{LpO}(k) &= \binom{n}{p}^{-1} \sum_{e\in\mathscr{E}} \frac{1}{p} \sum_{i\in\bar{e}} \mathbb{I}_{\{f^e(x_i)\neq y_i\}} \\
&= \frac{1}{p}\sum_{i=1}^{n} \binom{n}{p}^{-1} \sum_{e\in\mathscr{E}} \mathbb{I}_{\{f^e(x_i)\neq y_i\}}\mathbb{I}_{\{i\in\bar{e}\}} \\
&= \frac{1}{p}\sum_{i=1}^{n} \sum_{e\in\mathscr{E}} \mathbb{I}_{\{f^e(x_i)\neq y_i\}\bigcap\{i\in\bar{e}\}} P(E=e) \\
&= \frac{1}{p}\sum_{i=1}^{n} P\left(\{f^E(x_i)\neq y_i\}\bigcap\{i\in\bar{E}\}\right) \\
&= \frac{1}{p}\sum_{i=1}^{n} P\left(f^E(x_i)\neq y_i | i\in\bar{E}\right) P\left(i\in\bar{E}\right) \\
&= \frac{1}{p}\sum_{i=1}^{n}\sum_{j=1}^{n} P\left(f^E(x_i)\neq y_i | i\in\bar{E},\, V_k^i=j\right) P\left(V_k^i=j|i\in\bar{E}\right) P\left(i\in\bar{E}\right) \\
&= \frac{1}{p}\sum_{i=1}^{n} P\left(i\in\bar{E}\right) \sum_{j=k}^{k+p-1} P\left(V_k^i=j|i\in\bar{E}\right) P\left(f^E(x_i)\neq y_i | i\in\bar{E},\, V_k^i=j\right) \quad .
\end{aligned}$$

$\square$

Let us successively specify $A1$, $A2$, and $A3$. Since E is uniformly distributed over $\mathscr{E}$, it comes

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

For $A2$, one also has

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

where $U \hookrightarrow \mathscr{H}(j, n-j-1, p-1)$ and $\mathscr{H}(a,b,c)$ 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. An important feature is that neither $A1$ nor $A2$ actually depend on i. Actually, i only arises from A_3.

To evaluate the computation cost of $A3$, let us consider the ordered sequence $X^i_{(1)},\ldots,X^i_{(n-1)}$, where $X^i_{(k)}$ is the k^{th} neighbor of i in the complete sample. Since p observations (including i) are removed at a given step of the LpO procedure, the first k neighbors of i belong to $\{X^i_{(1)},\ldots,X^i_{(k+p-1)}\}$. Once this list is obtained (by applying the $(k+p-1)$NN classifier to the complete data), one only needs to compute the number of times (over all splittings) the majority label is that of observation i, for each value of j. Therefore, the computational cost to compute $A3$ for a given observation is of order

$$\mathscr{O}\left(\sum_{j=k}^{k+p-1}\binom{j}{k-1}\right) < \mathscr{O}\left(p\binom{k+p}{k-1}\right) \quad .$$

that does not depend on n. As a consequence, the computation of R_{LpO} is linear in n. Let us now specify $A3$ for weighted and non-weighted kNN classifiers.

2.2. *Weighted kNN*

The weighted kNN rule is defined as

$$f_{WkNN}(x) = \begin{cases} 1 & \text{if } \sum_{\ell=1}^{k} w_{(\ell)}^{x} \mathbb{I}_{\{Y_{(\ell)}=1\}} > \sum_{\ell=1}^{k} w_{(\ell)}^{x} \mathbb{I}_{\{Y_{(\ell)}=0\}} \\ 0 & \text{otherwise} \end{cases} ,$$

where $w_{(\ell)}^{x}$ 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

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

where threshold s is chosen to be 0.

$A3$ corresponds to the frequency at which observation i is misclassified given 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 $\{X_{(1)}^{i}, \dots, X_{(j)}^{i}\}$. Let us define W_0 and W_1 as follows:

$$\begin{aligned} W_0 &= \{w_{(\ell)}^{i} \mid X_{(\ell)}^{i} \in \{X_{(1)}^{i}, \dots, X_{(j)}^{i}\} \text{ and } Y_{(\ell)} = 0\} \\ W_1 &= \{w_{(\ell)}^{i} \mid X_{(\ell)}^{i} \in \{X_{(1)}^{i}, \dots, X_{(j)}^{i}\} \text{ and } Y_{(\ell)} = 1\} \ . \end{aligned}$$

One has

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

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 $\sum_{\ell \in W_1} w_{(\ell)}^{i} - \sum_{\ell \in W_0} w_{(\ell)}^{i} > s$. Assume without loss of generality that $w_{(1)}^{i}$ belongs to set W_0. Then,

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

The computation of $N(W_0, W_1, k, s)$ can be obtained using recursive programming. An algorithm based on this strategy is proposed in the Appendix. In the case of inequal weights, the computational cost to compute R_{LpO} is $\mathcal{O}\left(np\binom{k+p}{k-1}\right)$.

2.3. *Non-weighted kNN*

In the case where all neighbors receive the same weight in the majority vote classification rule, the computation of $A3$ is straightforward. Let n_j^i be the number of 1s among the j neighbors of i

in the complete sample. The quantity n^i_j can be obtained for all i and $j \in [1, k+p-1]$ by running the $(k+p-1)$NN classifier. We have:

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

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

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

where $H \hookrightarrow \mathscr{H}(N^j_i, j - N^j_i - 1, k-1)$, $H' \hookrightarrow \mathscr{H}(N^j_i - 1, j - N^j_i, k-1)$, and F_H stands for the cumulative distribution function of variable H.

Similar formulas can be derived for $P\left(f^E(x_i) = 0 | i \in \bar{E},\, V^i_k = j\right)$.

This shows that in the case of equal weights, the computational cost to compute R_{LpO} is $\mathscr{O}(np)$ instead of $\mathscr{O}\left(np\binom{k+p}{k-1}\right)$, i.e. L$p$O for the kNN classifier can be performed at the same computational cost as L1O for the $(k+p-1)$NN classifier, whatever p.

2.4. Computation time

Figure 1 displays the computation time of the exact LpO procedure for the non-weighted kNN classifier, for $k = 50$ and $p = 100, 200, 300, 400$. The complete distance matrix between observations is calculated beforehand. For each observation, the label is drawn in a Bernoulli distribution $\mathscr{B}(q)$, results are presented for $q = 0.1$. One can observe that the computation time is linear in n and p. As an example, exact LpO for a training sample of size $n = 5000$ and $p = 200$ is run within a minute.

Table 1 provides the average computational time of the weighted procedure, on a sample of 500 observations. Results are presented for $q = 0.1$ and 0.3. Weights in the majority voting rule are all equal to 1. Interestingly, the computational burden associated with the proposed algorithm decreases with the noise level q. As a comparison, for $k = 9$, $p = 25$ and $n = 500$ the exact LpO procedure for non-weighted kNN is run within one second.

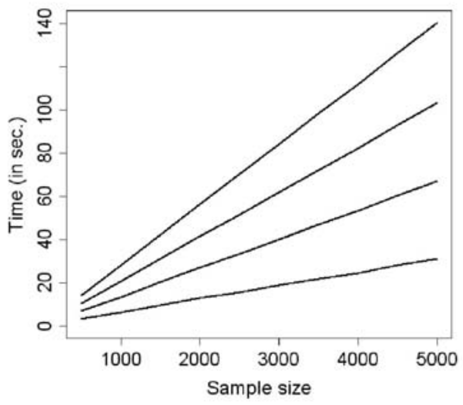

FIGURE 1. *Computational time (in seconds) of LpO procedure for the non-weighted kNN classifier, with* $k = 50$. *The 4 curves correspond to* $p = 100, 200, 300, 400$ *(from bottom to top).*

k/p	5	10	15	20	25
3	0.4 (0.0)	0.9 (0.0)	1.5 (0.0)	2.3 (0.0)	3.2 (0.1)
5	0.8 (0.0)	3.0 (0.2)	7.6 (0.4)	16.1 (1.1)	30.6 (2.3)
7	1.2 (0.0)	7.0 (0.4)	30.8 (2.6)	111.6 (11.2)	334.9 (30.6)
9	1.7 (0.1)	18.1 (1.6)	154.3 (18.7)	817.6 (69.8)	3009.3 (151.2)

$q = 0.1$

k/p	5	10	15	20	25
3	0.5 (0.0)	1.3 (0.1)	2.4 (0.2)	3.8 (0.3)	5.4 (0.4)
5	1.5 (0.1)	7.6 (0.6)	22.1 (1.6)	50.9 (3.4)	101.7 (6.1)
7	2.8 (0.1)	24.4 (1.8)	124.5 (10.0)	473.1 (34.0)	1417.0 (69.7)
9	4.6 (0.4)	73.1 (5.1)	676.9 (49.4)	4238.8 (233.7)	19009.4 (846.6)

$q = 0.3$

TABLE 1. *Average computational time in seconds (and standard deviation) of LpO procedure for the weighted kNN classifier, for different values of k, p and q. Average and SD are computed on 6 replicate samples.*

3. Passive Learning

Using *k*NN classifiers in passive learning requires to choose k. This can be done using L*p*O. 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 [3]. Having access to exact L*p*O 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 2 (left) displays k_p with respect to n for a level of noise $q = 0.2$, 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 $k = o(n)$ [3], it leads us to conjecture that the kNN classifier 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 in 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.
In the regression and density estimation framework, is has been shown that the choice of p can be crucial to perform efficient parameter tuning [1]. A similar conclusion is supported by our simulation experiments.

We first observe that increasing p entails a smaller choice of k_p (see Figure 2 (right) and Table 2), which can be desirable as shown in the following. This phenomenon is observed with several noise levels from $q = 0.1$ up to $q = 0.4$ (not shown).

Second, it is necessary to choose p larger than 1 as soon 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 2 (right) and Table 2, where the minimum locations of red curves (small values of p) are larger than that of the black curves (which displays the true risk computed on a large validation set). This is also observed with a noise level $0.1 \leq q \leq 0.4$. A growing noise reduces the influence of the bias in the fitting of the kNN classifier, leading to a larger optimal k (compare black curves of Figure 2 between center and right panels). LpO with small values of p exhibits a higher sensitivity to this phenomenon than with larger values of p (Figure 2 right panel).
Therefore, this trend can be balanced by using larger values of p (since higher p yield lower k_p). Indeed, we observe on Figure 2 that 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 (*i*) using L1O can be misleading, (*ii*) a convenient choice of $p > 1$ is required to provide a reliable k_p.

	$1 < p < 10$	$11 < p < 30$	$40 < p < 80$	$p > 80$	Test
k	21	19	17-15	13-9	17

TABLE 2. *Choice of parameter k by LpO for different values of p, or by test sample, when* $q = 0.3$.

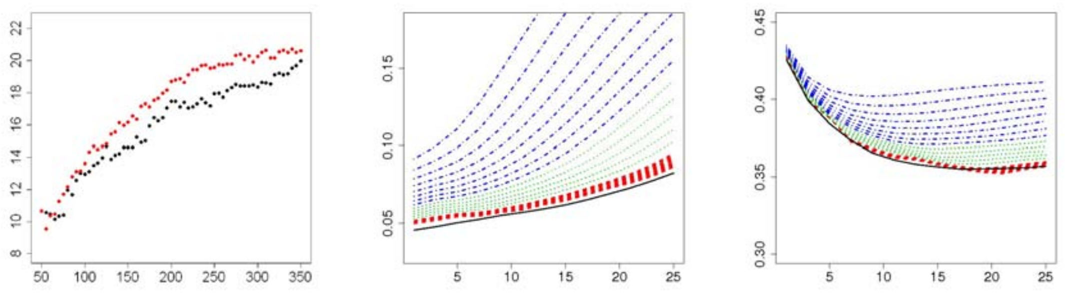

FIGURE 2. ***Left:*** *Evolution of k (in ordinates) with respect to sample size n,* $q = 0.2$*. Black points correspond to* k_p*, red points correspond to the optimal choice of k (based on a large test sample).* ***Center:*** *Plot of the average classification error rate (in ordinates) evaluated by LpO with different p (colored curves) or on test samples (black curves), for different values of k (x-axis) 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* $q = 0.2$*.*

3.3. *Risk estimation*

In many applications, one is also interested in a sharp estimation of the performance of a given classifier. 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 2 shows that large values of p (blue curves) lead to biased estimations of the true risk (black curve). In other frameworks ([1, 2]), 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 2 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 (not shown here).

Third, an important conclusion arising from the case $q = 0$ (center of Figure 2) 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 possibility for the algorithm to select the data (*examples*) 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 select m examples from the pool. These example are added to the training set after disclosure of their label, the training set growing 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 [11], the query by committee strategy (QbC) has been widely investigated. From a large committee of classifiers (also called *experts*), each expert predicts the label of every example of the pool. The “best” m examples are those for which the committee experts 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* (Section 3.3). *Since these purposes are different, the optimal choice of p at these two steps could be different.*

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

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

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

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

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

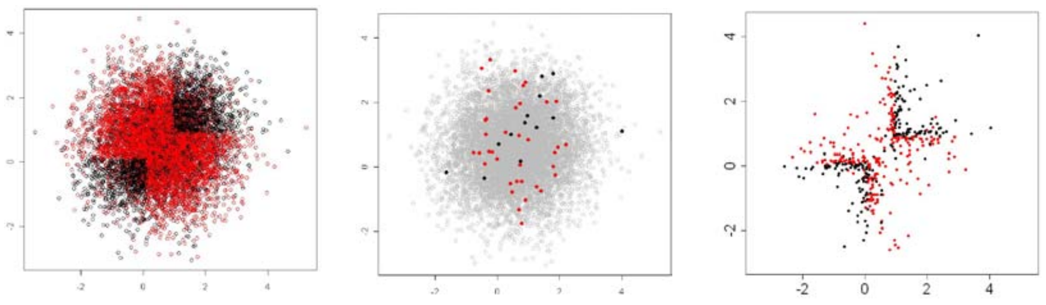

FIGURE 3. *Example of pool sample P^0 (left), starting training set E^0 (center), and final training set E^{60} after 60 steps of active learning (right) for $q = 0.2$. Red and black colors indicate the label.*

4.3. *A short illustration of the LpO-QbC strategy*

The goal of the present section is to assess the performance of LpO-QbC as an active learning, compared with a passive learning algorithm based on 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 passive learning 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 $p = 10$ (*point selection*). Then a kNN classifier is trained on $E^{(\ell)}$, with k chosen by LpO minimization with $p = 10$ (*tuning step*). The classifier performance is evaluated on $P^{(\ell)}$. This process (point selection and tuning step) is repeated 60 times, that is $0 \leq \ell \leq 60$.

Simulations 2-dimension data are simulated. $X = (X^1, X^2)$ is generated using a mixture of 3 Gaussian distributions, with proportions $(0.2, 0.2, 0.6)$, 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: $q = 0, 0.1, 0.2, 0.3$, and 0.4. 100 repetitions of each condition have been performed. The initial sizes of training and pool sets are 50 and 10.000, respectively.

Figure 3 shows these 2 sets for a round simulated with $q = 0.2$, and the final training set E^{60} 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 4 (left), after 60 rounds, LpO-QbC achieves an averaged 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 (see [7], p.264, for a complete description and the list of variables). 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 16.2% respectively, for a final training set of size 350 (Figure 4 (right)). As for simulated data, LpO-QbC focuses on examples close to the border between the two classes (not shown) and outperforms p-LpO.

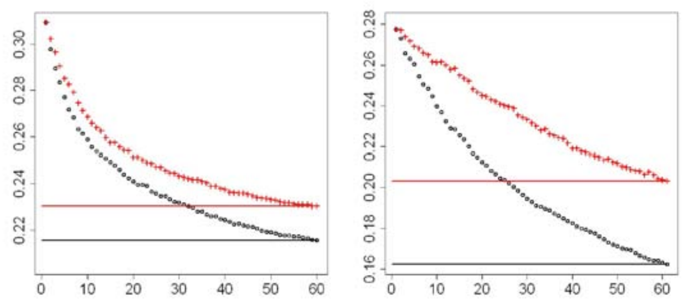

FIGURE 4. *Performance of LpO-QbC (o) and passive kNN (+) on simulated data (left), and Spam data (right).*

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 come 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 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 5 (left and center), 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 discussed in 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 explain the behavior observed in Figure 5 (left).

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 5, one can see that the evolution of k_p as a function of n is completely different from that of Figure 2: 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 then selected. Compared with Figure 2, 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: values smaller than k_{oracle} are always selected.

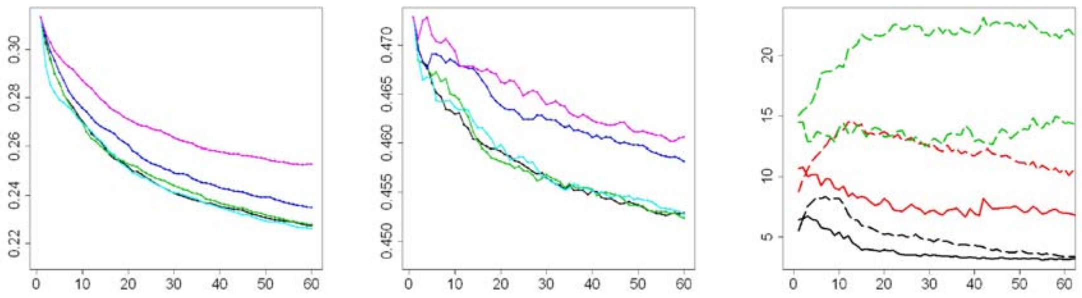

FIGURE 5. ***Left:*** *Performance of the committees for $k = 1$ (cyan), 5 (black), 10 (green), $n^{(\ell)}/2$ (blue) and passive learning (pink), $q = 0.2$.* ***Center:*** *Performance of the committees and passive learning, $q = 0.4$.* ***Right:*** *Evolution of k at each step of the active learning,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

5. Discussion

In applications of *k*NN to real data, L*p*O is used either to assess the performance of a *k*NN classifier (risk estimation), or to choose k (model selection). In both cases, p is fixed at 1 in most cases 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 L*p*O estimator associated with *k*NN and W*k*NN classifiers yield an efficient and practical tool to study the behavior of k_p, both for theoretical and practical purposes. Exact L*p*O should be preferred to its classical surrogate *K*CV, since L*p*O is less variable (with $K = n/p$).

In passive learning, some theoretical results already exist about the application of L1O to *k*NN [3]. But to the best of our knowledge, there is no such result for the general L*p*O procedure. The present simulation study is a preliminary work before the theoretical analysis of L*p*O in the passive learning setting. Some further work is required to get more insight toward a data-driven calibration of p.

In active learning, L*p*O can be used for point selection (when considering the QbC strategy) or for parameter tuning. We showed that point selection does not exhibit any strong dependence on p, which validates the use of L1O for 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 L*p*O are all the more attractive.

Acknowledgment

The authors acknowledge the support of the French Agence Nationale de la Recherche (ANR) under reference ANR-09-JCJC-0027-01.

Appendix: Algorithms for W*k*NN Leave-*p*-out

Positive weights First, the following problem is considered: assuming that m objects have positive values $w_1 \leq ... \leq w_m$, how many combinations (without replacement) of k of these objects among m lead to a total value higher than s, with $s > 0$? Denote $W = (w_1, ..., w_m)$, and $N(W,s,k)$ the number of combinations for which the condition is fulfilled, and $I(W,s)$ the breakpoint index of W. The breakpoint index is the smallest j such that $\sum_{i\leq j} w_i \geq S$ (if for all j $\sum_{i\leq j} w_i < S$ then $I(W,s) = m+1$ by convention).

There are several convenient settings where $N(W,s,k)$ can be computed:
- if $k = 1$, then $N(W,s,k) = m - \max\{j/w_j < s\}$,
- if W is of length k, then $N(W,s,k) = 0$ or 1,
- if $I(W,s) \leq k$ then $N(W,s,k) = \binom{m}{k}$,
- if $I(W,s) = m+1$ then $N(W,s,k) = 0$.

Based on these remarks, the proposed algorithm is:

```
Require: $W$, $s$, $k$
  $L \leftarrow$ length($W$)
  $BI \leftarrow$ breakpoint_index($W$,$s$,$k$)
  $BoolCond \leftarrow$ check_conv_settings($W$,$s$,$k$,$L$,$BI$)
  if $BoolCond = 1$ then
    $NumbComb \leftarrow$ compute_numb_comb($W$,$s$,$k$,$L$)
  else
    $NumbComb = 0$
    for $i = BI$ to $L$ do
      $NumbComb \leftarrow NumbComb$ + Pos_Weights($W[1:i-1], s - W[i], K-1$)
    end for
  end if
  return $NumbComb$
```

In practice, this algorithm is faster than the naive algorithm based on recursive programming only (i.e. where the breakpoint index is not computed).

Positive and negative weights We now assume that m_0 objects have negative values $w_1^0 \leq ... \leq w_{m_0}^0$, and m_1 objects have positive values $w_1^1 \leq ... \leq w_{m_1}^1$, and we wonder how many combinations (without replacement) of k of objects among $m_0 + m_1$ lead to a total value higher than s. We note $W_i = (w_1^i, ..., w_{m_i}^i)$ for $i = 0, 1$, $W = (W_0, W_1)$, and denote $N(W_0, W_1, s, k)$ the number of combinations for which the condition is satisfied.

The convenient settings where $N(W_0, W_1, s, k)$ can be computed are the following ones:
- if $k = 1$, then $N(W_0, W_1, s, k) = m - \max\{j \mid w_j < s, \;\; w_j \in W\}$,
- if W is of length k, then $N(W_0, W_1, s, k) = 0$ or 1,

Besides, if either W_0 or W_1 is empty we can use algorithm 5 proposed in the previous paragraph. The new algorithm is then:

```
Require: $W_0$, $W_1$, $s$, $k$
  $L_1 \leftarrow$ length($W_1$)
  $L_2 \leftarrow$ length($W_2$)
  $BoolCond \leftarrow$ check_conv_settings($W_1$,$W_2$,$s$,$k$,$L_1$,$L_2$)
  if $BoolCond = 1$ then
    $NumbComb \leftarrow$ compute_numb_comb($W_1$,$W_2$,$s$,$k$,$L_1$,$L_2$)
  else if is_empty($W_0$) then
    $NumbComb \leftarrow$ Pos_Weights($W_1, s, K$)
  else if is_empty($W_1$) then
    $NumbComb \leftarrow \binom{k}{m_1}$ - Pos_Weights($-W_0, -s, K$)
  else
    $NumbComb \leftarrow$ PosNeg_Weights($W_0, W_1[1:L_1-1], K-1, s-w^1_{L_1}$)
                 + PosNeg_Weights($W_0, W_1[1:L_1-1], K, s$)
  end if
  return $NumbComb$
```

Notice that the recursive call of the algorithm can be refined by reducing either W_0 or W_1 (depending on which one has the smallest number of items) instead of W_1 only. In this case, the "worst" cases are the one where W_0 and W_1 are of equal size, i.e. intuitively cases where the noise level is high.

References

[1] S. Arlot and A. Celisse. A survey of cross-validation procedures for model selection. *Statist. Surv.*, 4:40–79, 2010.

[2] A. Celisse and S. Robin. Nonparametric density estimation by exact leave-p-out cross-validation. *Comput. Statist. Data Anal.*, 52(5):2350–2368, 2008.

[3] L. Devroye, L. Gyorfi, and G. Lugosi. *A probabilistic theory of pattern recognition.* Springer, 1996.

[4] E. Fix and J. Hodges. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*, chapter Discriminatory analysis- nonparametric discrimination: Consistency principles. IEEE Computer Society Press, Los Alamitos, CA, 1991. Reprint of original work from 1952.

[5] E. Fix and J. Hodges. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*, chapter Nonparametric Discrimination: small sample performance. IEEE Computer Society Press, Los Alamitos, CA, 1991. Reprint of original work from 1952.

[6] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, C.D. Bloomfield, and E.S. Lander. Class prediction and discovery using gene expression data. *Science*, 286:531–537, 1999.

[7] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, New York, 2001.

[8] A. McCallum and K. Nigam. Employing EM and pool-based active learning for text classification. In *Mach. Learn.: Proc. of the Fifteenth Intern. Conf. (ICML '98)*, pages 359–367, 1998.

[9] B. Settles. Active learning literature survey. Comp. Sci. Tech. Report 1648, Univ. of Wisconsin–Madison, 2009.

[10] B. Settles, M. Craven, and L. Friedland. Active learning with real annotation costs. In *Proceedings of the NIPS Workshop on Cost-Sensitive Learning*, pages 1–10, 2008.

[11] H. S. Seung, M Opper, and H. Sompolinsky. Query by committee. In *Annual Workshop on Computational Learning Theory*, pages 287–294, 1992.

[12] P.Y. Simard, Y. LeCun, J.S. Denker, and B. Victorri. Transformation invariance in pattern recognition – tangent distance and tangent propagation. *International Journal of Imaging Systems and Technology*, 11(3), 2001.

[13] M. Stone. Cross-validatory choice and assessment of statistical predictions. *J. Roy. Statist. Soc. Ser. B*, 36:111–147, 1974.