RAIRO-Theor. Inf. Appl. 48 (2014) 209–245
DOI: 10.1051/ita/2014005

Available online at:
www.rairo-ita.org

A GENERALIZED MODEL OF PAC LEARNING AND ITS APPLICABILITY

Thomas Brodag[1], Steffen Herbold[1] and Stephan Waack[1]

Abstract. We combine a new data model, where the random classification is subjected to rather weak restrictions which in turn are based on the Mammen−Tsybakov [E. Mammen and A.B. Tsybakov, *Ann. Statis.* **27** (1999) 1808–1829; A.B. Tsybakov, *Ann. Statis.* **32** (2004) 135–166.] small margin conditions, and the statistical query (SQ) model due to Kearns [M.J. Kearns, *J. ACM* **45** (1998) 983–1006] to what we refer to as PAC + SQ model. We generalize the class conditional constant noise (CCCN) model introduced by Decatur [S.E. Decatur, in *ICML '97: Proc. of the Fourteenth Int. Conf. on Machine Learn.* Morgan Kaufmann Publishers Inc. San Francisco, CA, USA (1997) 83–91] to the noise model orthogonal to a set of query functions. We show that every polynomial time PAC + SQ learning algorithm can be efficiently simulated provided that the random noise rate is orthogonal to the query functions used by the algorithm given the target concept. Furthermore, we extend the constant-partition classification noise (CPCN) model due to Decatur [S.E. Decatur, in *ICML '97: Proc. of the Fourteenth Int. Conf. on Machine Learn.* Morgan Kaufmann Publishers Inc. San Francisco, CA, USA (1997) 83–91] to what we call the constant-partition piecewise orthogonal (CPPO) noise model. We show how statistical queries can be simulated in the CPPO scenario, given the partition is known to the learner. We show how to practically use PAC + SQ simulators in the noise model orthogonal to the query space by presenting two examples from bioinformatics and software engineering. This way, we demonstrate that our new noise model is realistic.

Mathematics Subject Classification. 68Q32, 62P10, 68N30.

Keywords and phrases. PAC learning with classification noise, Mammen−Tsybakov small margin conditions, statistical queries, noise model orthogonal to a set of query functions, bioinformatics, software engineering.

[1] Institut für Informatik, Georg-August-Universität Göttingen, Goldschmidtstr. 7, 37077 Göttingen, Germany. {brodag,herbold,waack}@cs-uni-goettingen.de

1. Introduction and overview

The Probably Approximately Correct model of learning (PAC learning), introduced by Valiant [48], has proven an interesting model of machine learning from an algorithmic point of view. PAC learning is *concept learning.* A concept is a rule to divide input vectors from $\mathbb{R}^n$ into positive and negative examples. The problem is to infer an unknown *target concept* g^* from a known *concept class* $\mathcal{C}_n$ such that positive and negative training examples obey that rule. In this paper, we prefer the functional scenario of PAC learning considered in [27] to Valiant's oracle model. In the functional setting, the training data consist of m independent instantiations of a random pair $U = (X, Y)$, where the *random input vector* (or *observation*) X takes value in the the input space $\mathfrak{X}_n \subseteq \mathbb{R}^n$, and the *random classification* (or *label*) Y is from $\{0, 1\}$. The distribution $\mathbb{P}_U$ induced by U on the so-called *learning universe* $\mathfrak{U}_n := \mathfrak{X}_n \times \{0, 1\}$ describes the frequency of how often particular pairs occur in practice. That is why we denote U as *generic data element.* The learning algorithms are assumed to perform well with respect to any target distribution $\mathbb{P}_X$ induced by the random observation X on the input space $\mathfrak{X}_n$. This means that the (test) error defined as the probability that the target concept differs from the hypothesis learned is minimized given the training sample length m tends to infinity.

Valiant's original model takes a noise-free view of the concepts to be learned. This means $Y = g^*(X)$. To make algorithms robust, several noise models have been studied. In the *malicious error model* introduced by Valiant [49] and further studied in [32], an adversary is allowed to corrupt any instance of the noise-free learning sample with some probability that may depend on the index of the instance but not on the instance itself. Moreover, probabilistic concepts can be considered in the context of noisy data [33].

Goldman and Sloan studied in [24] the impact of random noise affecting only the attributes of the observation and not the classification when the input space is $\{0, 1\}^n$. They presented a PAC learner for monomials that tolerates uniform attribute noise, where every observation bit is independently flipped with the same probability. In contrast, they showed that product random attribute noise, where each attribute is corrupted with its own probability, is nearly as harmful as malicious noise.

In this paper, we study PAC learning with classification noise. The *random noise rate* $\nu(X)$ is defined to be the conditional probability

$$\nu(X) := \mathbb{P}\left(Y \neq g^*(X) \mid X\right), \tag{1.1}$$

where we always assume that $\nu(x) \leq 1/2$ for all $x \in \mathfrak{X}_n$.

Angluin and Laird [2] were the first who considered *constant classification noise* (CN) in which the noise rate $\nu(X)$ equals a constant $\nu \leq \nu^{(b)} < 1/2$, where the upper bound $\nu^{(b)}$ is known to the learner.

This model was generalized by Decatur [18] *via* the *class-conditional classification noise* (CCCN) model to the *constant-partition classification noise* (CPCN) model. In the CCCN model the random noise rate is constant given the classification of the target concept. This means that there are two constants $\nu_0, \nu_1 \in [0, 1/2)$ such that $\nu(x) = \nu_{y_0}$ given $g^*(x) = y_0$, for $y_0 = 0, 1$. The CPCN model allows a constant number of blocks of the input space having constant noise rates. In [44] it is proved that any concept class that is efficiently learnable in the CN model, is efficiently learnable in the CPCN model, too.

Nettleton *et al.* presented in [39] a survey of how attribute noise or classification noise in the training data or in the training data and the test data can affect the precision of naive Bayes classifiers [30], C 4.5 decision trees [43], IBk instance-based learners [1] and SMO support vector machines [42]. According to their findings, naive Bayes classifies are most robust, though the empirical behavior of these four techniques varies depending the noise type. The classification noise utilized is a variant form of the CCCN model.

Cesa–Bianchi *et al.* combined in [14] attribute noise with classification noise in the context of kernel-based online learning. The hypotheses are of the form $\mathbf{x} \mapsto \langle \mathbf{w}, \psi(\mathbf{x}) \rangle$, where ψ is a fixed feature mapping into some kernel reproducing Hilbert space, and the vector $\mathbf{w}$ represents the current hypothesis. The learning protocol is a repeated game between the learner and an adversary. When it is the adversary's turn, it chooses an example $(\mathbf{x}_i, y_i)$, a zero-mean random vector $\mathbf{N}_i^{\mathbf{x}}$, and a zero-mean random variable $\mathbf{N}_i^y$. When it is the learner's move, it is given access to an oracle returning independent instantiations of $(\mathbf{x}_i + \mathbf{N}_i^{\mathbf{x}}, y_i + \mathbf{N}_i^y)$. It then picks a (random) hypothesis $\mathbf{W}_{i+1}$. It is the goal to minimize the expected regret. Cesa-Bianchi *et al.* showed that this is tractable, if a random, essentially constant number of noisy copies of each instance is accessible. Moreover, the more is known about the noise distribution, the better are the conditions for learning. In contrast, if nothing is known about the noise distribution and only one noisy copy of each instance is available the setting becomes intractable.

In the statistical query model introduced by Kearns [31] the learning algorithms are restricted in their use of the training data. They are only allowed to make statistical queries (SQs). A SQ is a request for a sufficiently accurate estimate of the expectation $\mathbb{E}\left(\chi(X, g^*(X))\right)$ with respect to the target distribution $\mathbb{P}_X$ and the target concept g^*, where the polynomially computable query function χ maps the learning universe $\mathbf{U}_n$ to the interval $[0, 1]$.

Statistical queries can be simulated in the PAC model in the presence of constant classification noise, malicious error noise and even hybrid models combining this noise models [15, 16, 19, 31]. In [5] a SQ model based on relative error bounds is introduced. Highly efficient PAC simulations of the algorithms are presented there.

Our first contribution is the introduction of a type of learning algorithms, which we call PAC + SQ learners, that are allowed to directly access the training data in addition to making statistical queries.

In [31] a weaker variant was considered, where the learner only is given access to an unclassified sample of observations.

As our second contribution we allow random noise rates that are observation-dependent rather than block-wise constant. We show that a PAC + SQ learner can be efficiently and consistently simulated as long as the random noise rate is conditionally uncorrelated to the query functions used by the algorithm given the target concept (see Thms. 6.1 and 7.3). From a geometric point of view this means that the random noise rates allowed are orthogonal to the query functions used.

The simulator devised in order to prove Theorem 6.1 has an empirical risk minimization part. Moreover, when it comes to devising learning algorithms for particular problems empirical risk minimization techniques is useful. In the PAC learning setting deviation from the Bayes classifier is measured. Under which additional conditions empirical risk minimization is guaranteed to work there?

Mammen and Tsybakov [35, 47] have given a comprehensive answer. This is the case if and only if the distribution of the generic data element $U = (X, Y)$ is subject to a small margin condition restricting the probability that the random noise rate $\nu(X)$ is close to 1/2 (see Sect. 2). Our third contribution is to integrate a small margin condition into the data model of PAC learning.

Haussler's covering method [26] to learn monomials consisting of at most s clauses is our vehicle for demonstrating how these ideas work. Compared with a variant of Haussler's algorithm due to Kearns [31], we modify the cover phase such that the risk on the negatively declared data is minimized rather than the error on the true negatives, whereas the prune phase minimizing the error on the true positives by means of statistical queries remains more or less unchanged. As our fourth contribution we prove consistency of such a mixed strategy of learning (see Sect. 4.3).

The worst-case sample length of our modified algorithm is reduced. Its dependence on the target concept size $\Theta(s)$ is quadratic in the noise model orthogonal to the queries used in contrast to the cubic dependence of Kearns' algorithm in the CN model [31]. The main advantage is, however, that the data demand of our PAC + SQ learner is much less in practice. There are two reasons for that. First, a worst-case scenario hardly ever comes to pass in practice. Second, a factor $r = \mathcal{O}(s \log(1/\epsilon))$ of Kearns' sample length bound is an exception to this rule, because his cover phase falls into r stages where each of these needs a fresh learning sample, not just in the worst case but in every case. In contrast, our cover phase only requires a single sample.

Finally, we present two examples of the way our new model can be applied in practice. Thus, we confirm that our noise model is reasonable. It is unlikely to infer relevant information only from differences in noise rate.

Leveraging our PAC + SQ variant of Haussler's covering method, we compute a powerful discriminator between protein-protein interfaces and random sets of pairs of protein surface residues. A SVM with RBF kernel functions has as a slightly better classification rate when solving the same problem. In contrast to the SVM classifier, our PAC + SQ discriminator has a convincing biological interpretation.

We sketch how we applied PAC learning to problems from software engineering by means of PAC learning in [28].

1.1. Organization of the paper

In Section 2 we introduce the basic notions of pattern classification, define uniform convergence, Rademacher averages and Vapnik−Chervonenkis dimension, and expose the small margin conditions due to Mammen and Tsybakov [35, 47].

A general definition of classification noise models as well as of efficient PAC learnability with respect to such noise models is presented in Section 3. Moreover, statistical queries as a building blocks for such learners are introduced there.

Section 4 presents our new model of PAC learning. Therein, we introduce our new generalized model of PAC learning with classification noise. It comprises a type of learning algorithms we refer to as $\mathrm{PAC} + \mathrm{SQ}$ learners and random noise rates orthogonal to a set of query functions given the target concept. Notions from Sections 2 and 3 are brought together.

In Section 5 we show how to efficiently simulate (conditional) statistical queries using query functions orthogonal to the random noise rate given the target concept. Thus we extend the corresponding results presented in [31].

We use the results of Section 5 to prove Theorem 6.1 as our main result in Section 6.

In Section 7 we generalize Decatur's CPCN scenario to what we call the constant-partition piecewise orthogonal (CPPO) noise model. We prove that every efficient $\mathrm{PAC} + \mathrm{SQ}$ learner can be efficiently simulated in this model, too.

In Section 8 we transform Haussler's covering method [26] fitted by Kearns to the CN scenario [31] into a $\mathrm{PAC} + \mathrm{SQ}$ learner.

In Section 9 and in Section 10 we present two examples of the way our new PAC learning framework can be used in practices.

1.2. Terminology and notation

We use latin uppercase letters such as X, Y, G or H when referring to random aspects of the object denoted by them. Unless otherwise specified, expectations are always taken over such variables.

For b a Boolean value, $\bar{b} = \neg b$ denotes the negation of b. For any hypothesis h, the hypothesis $\bar{h}$ is the complement of h.

For any measurable set A, let $\mathbb{1}_A$ denote the indicator function of A.

2. Pattern classification

Let $U = (X, Y)$ be the generic data element consisting of the random observation X taking values in an *input space* $\mathfrak{X}_n \subseteq \mathbb{R}^n$ and a random classification $Y \in \{0, 1\}$.

The problem of *pattern classification* is to predict the classification Y of the observation X of *length* n by means of a measurable *hypothesis* that maps the input space $\mathfrak{X}_n$ to $\{0, 1\}$.

The accuracy of a hypothesis h is measured by its *risk*.

Definition 2.1. The risk of a hypothesis h is defined to be

$$\mathbf{r}(h) := \mathbb{P}\left(h(X) \neq Y\right).$$

The *posterior probability* of the classification on the observation $x \in \mathfrak{X}_n$ is a key notion in statistical learning theory. It is defined by

$$\eta(x) := \mathbb{P}\left(Y = 1 \mid X = x\right) \qquad (x \in \mathfrak{X}_n).$$

It is easy to prove, that the pair $(\mathbb{P}_X, \eta)$, where $\mathbb{P}_X$ is the probability distribution induced by the random observation X on the input space $\mathfrak{X}_n$, fully determines the distribution $\mathbb{P}_U$ of the generic data element (see [20]).

Definition 2.2. The classifier

$$g^*(x) = \begin{cases} 1 & \text{if } \eta(x) > 1/2 \\ 0 & \text{otherwise,} \end{cases}$$

is called the Bayes classifier, the risk $\mathbf{r}(g^*)$ the Bayes risk.

According to equation (1.1) and Definition 2.1 the Bayes risk is equal to the expected noise.

$$\mathbf{r}(g^*) = \mathbb{E}\left(\nu(X)\right) =: \nu$$

In the PAC learning scenario outlined in Section 1 the target concept is always equal to the Bayes classifier and therefore we denote both as g^*.

Hypotheses are calculated from an i.i.d. *learning sample*

$$\mathbf{U}_m = \left(U_1, U_2, \ldots, U_m\right),$$

where the generic data element U is independent of $\mathbf{U}_m$ and $U_i = (X_i, Y_i)$, for $i = 1, 2, \ldots, m$, induces the same distribution as U on $\mathfrak{U}_n$. As a transform of the learning sample, these hypotheses, denoted by $\hat{H}_m$, are random and parameterized by the length m. The whole sequence $\hat{H}_m$ is called a *discrimination rule.*

The risk associated with a discrimination rule $\hat{H}_m$ is given by the random variable

$$\mathbf{r}\big(\hat{H}_m\big) := \mathbb{P}\left(\hat{H}_m(X) \neq Y \mid \mathbf{U}_m\right).$$

Definition 2.3. A discrimination rule $\hat{H}_m$ is defined to be consistent, if $\mathbf{r}(\hat{H}_m) - \mathbf{r}(g^*)$ stochastically converges to zero when m tends to ∞.

To deal with the *convergence rate* of consistent discrimination rules according to Definition 2.3, the standard representation of the difference

$$\mathbf{r}(h) - \mathbf{r}(g^*) = \int_{\{x \mid h(x) \neq g^*(x)\}} |2\,\eta(x) - 1| \,\mathrm{d}\,\mathbb{P}_X(x) \tag{2.1}$$

is important, where h is any hypothesis.

The empirical risk minimization (ERM) is a prominent discrimination rule, which is denoted by $\widehat{\mathrm{ERM}}_m$. It has the form

$$\hat{H}_m = \operatorname*{argmin}_{h \in \mathcal{H}_n} \widehat{\mathbf{R}}_m(h), \quad \text{where} \quad \widehat{\mathbf{R}}_m(h) := \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}_{\{h(X_i) \neq Y_i\}},$$

is the *empirical risk* of the hypothesis h on the learning sample $\mathbf{U}_m$, and $\mathcal{H}_n$ is the *hypotheses class* over which the empirical risk is minimized.

Empirical risk minimization is not always consistent. The capacity of the hypotheses class $\mathcal{H}_n$ to capture pure chance has to be restricted. The Rademacher averages and the Vapnik−Chervonenkis dimension are such capacity measures.

Let $\mathcal{T}$ be a class of measurable functions from the learning universe $\mathfrak{U}_n = \mathfrak{X}_n \times \{0,1\}$ to a closed interval $[a,b]$. The *Rademacher averages* of the class $\mathcal{T}$ were introduced into learning theory in [6, 34, 37], and further studied in [9]. Let us recall the key notions and properties.

Let $R^{(m)} := (R_1, R_2, \ldots, R_m)$ be a sequence of m independent *Rademacher* random variables. They take values in $\{-1, +1\}$, where $\mathbb{P}(R_i = -1) = \mathbb{P}(R_i = +1) = 1/2$. The Rademacher average of $\mathcal{T}$ given the learning sample $\mathbf{U}_m$ is

$$\mathbf{a}_m(\mathcal{T}) := \frac{1}{m} \mathbb{E} \left(\sup_{\theta \in \mathcal{T}} \sum_{i=1}^{m} \theta(U_i) R_i \right). \tag{2.2}$$

For $\theta \in \mathcal{T}$, let

$$\hat{E}_{\theta,m} := \frac{1}{m} \sum_{i=1}^{m} \theta(U_i)$$

be the sample mean of $(\theta(U_1), \theta(U_2), \ldots, \theta(U_m))$ that is an estimator of $e_\theta := \mathbb{E}(\theta(U))$, whose distribution is exponentially tailed.

To prove consistency of empirical risk minimization by means of Rademacher averages, uniform convergence rates according to the following lemma are crucial [8, 12].

Lemma 2.4. *Given a finite function class $\mathcal{T}$ the elements of which take values in the real interval $[a,b]$. With probability greater than or equal to $1-\delta$ the following inequality is satisfied.*

$$\sup_{\theta \in \mathcal{T}} \left| e_\theta - \hat{E}_{\theta,m} \right| < 2\,\mathbf{a}_m(\mathcal{T}) + (b-a)\sqrt{\frac{\ln(2/\delta)}{2m}}.$$

Based on Massart's Lemma [36], the following well-known Lemma bounds the Rademacher averages of finite classes.

Lemma 2.5. *For a finite function class $\mathcal{T}$ as in Lemma 2.4, the Rademacher averages $\mathbf{a}_m(\mathcal{T})$ can be bounded from above by $(b-a)\sqrt{\frac{\ln|\mathcal{T}|}{2m}}$.*

Let us briefly recapitulate the classical VC dimension of a class $\mathcal{T}$ of 0/1-valued functions and its relation the Rademacher averages. Given any sequence $(u_1, u_2, \ldots, u_m)$ of length m over the learning universe $\mathfrak{U}_n$, we consider the projection of $\mathcal{T}$ onto this sequence defined by

$$\mathcal{T}_{u_1,u_2,\ldots,u_m} := \{(\theta(u_1), \theta(u_2), \ldots, \theta(u_m)) \mid \theta \in \mathcal{T}\}.$$

Definition 2.6. The growth function $S_{\mathcal{T}}(m)$ is defined to be

$$S_{\mathcal{T}}(m) = \sup_{u_1,u_2,\ldots,u_m} \left|\mathcal{T}_{u_1,u_2,\ldots,u_m}\right|.$$

The VC dimension vc-dim($\mathcal{T}$) of a class $\mathcal{T}$ is the largest m such that $S_{\mathcal{T}}(m) = 2^m$.

As a consequence of Dudley's entropy bound along with Haussler's upper bound on the so called covering numbers, we have

$$\mathbf{a}_m(\mathcal{T}) = \mathcal{O}\left(\sqrt{\frac{\text{vc-dim}(\mathcal{T})}{m}}\right). \tag{2.3}$$

See [11] for an overview on such methods.

Let

$$\mathcal{P}(\mathcal{H}_n) := \left\{\theta \,\middle|\, \exists h \in \mathcal{H}_n : \forall (x,y) \in \mathfrak{U}_n : \theta(x,y) = \mathbb{1}_{\{h(x) \neq y\}}\right\}.$$

The following theorem states under which conditions ERM works. Its proof is based on well-known techniques to decompose the risk [11] and on Lemma 2.4. Variants, partly based on other capacity measures, can be found in the literature (see *e.g.* [12, 45]).

Theorem 2.7. *For any $\delta \in (0,1)$ and any sample length $m > 0$ with probability at least $1 - \delta$ the inequality*

$$\mathbf{r}\big(\widehat{ERM}_m\big) - \nu < 2\,\mathbf{a}_m(\mathcal{P}(\mathcal{H}_n)) + \sqrt{\frac{2\ln(2/\delta)}{m}} \tag{2.4}$$

is satisfied.

In many cases it is intractable to efficiently implement $\widehat{\mathrm{ERM}}_m$. If there is only a parameterized learning algorithm A_k that implements a discrimination rule $\widehat{\mathrm{ERM}}_{k,m}$ approximately minimizing the empirical risk on $\mathbf{U}_m$ with performance ratio $(1 + 1/k)$, one gets instead of Theorem 2.7 the following one.

Theorem 2.8. *For any $\delta \in (0,1)$ and any sample length $m > 0$ with probability at least $1 - \delta$ the inequality*

$$\mathbf{r}\big(\widehat{ERM}_{k,m}\big) - \nu < 2\,\mathbf{a}_m(\mathcal{P}(\mathcal{H}_n)) + \left(1 + \frac{1}{k}\right)\sqrt{\frac{2\ln(2/\delta)}{m}} + \frac{1}{k}$$

is satisfied.

A great deal of work has been done in statistical learning theory to achieve fast convergence rates for consistent discrimination rules according to Definition 2.3. They depend on the complexity of the class $\mathcal{G}^*$ of possible Bayes classifiers and the *margin parameter*. In this paper we are only interested in the latter one. Starting point is here the intuition that fast convergence rates presuppose that the posterior probability $\eta(X)$ is very unlikely to be close to $1/2$. Having defined the neighborhood of the boundary $\partial g^* = \{x \mid \eta(x) = 1/2\}$ as margin of the Bayes classifier, Mammen and Tsybakov [35, 47] gave an elegant formulation of what they called *small margin conditions*.

Definition 2.9. The joint distribution $\mathbb{P}_U$ of observation X and classification Y satisfies the small margin condition M_β for some small margin condition parameter $0 < \beta < \infty$, if there is a small margin condition constant $c > 0$ such that for every $0 < \epsilon < 1/2$

$$\mathbb{P}\left(\left|\eta(X) - \frac{1}{2}\right| \leq \epsilon\right) \leq c \cdot \epsilon^{\beta}. \tag{2.5}$$

The joint distribution $\mathbb{P}_U$ satisfies the small margin condition M_∞, if there is a small margin condition constant $0 < c < 1/2$ such that

$$\mathbb{P}\left(\left|\eta(X) - \frac{1}{2}\right| \leq c\right) = 0. \tag{2.6}$$

Definition 2.10. The joint distribution $\mathbb{P}_U$ has margin parameter $0 < \kappa \leq 1$, if there is a margin constant $d > 0$ such that for every hypothesis h

$$\mathbb{P}\left(h(X) \neq g^*(X)\right) \leq d \cdot \left(\mathbf{r}(h) - \mathbf{r}(g^*)\right)^{\kappa}. \tag{2.7}$$

Lemmas 2.11, 2.12 and 2.13 show that the small margin conditions and the existence of a margin parameter are equivalent. See [7] for an elegant proof.

Lemma 2.11. *The joint distribution of observation and classification has margin parameter* 1 *with margin constant* d *if and only if it satisfies the small margin condition* M_∞ *with small margin condition constant* $c = 1/(2d)$.

Lemma 2.12. *If the joint distribution of observation and classification has margin parameter* $\kappa < 1$ *with margin constant* d*, then it satisfies the small margin condition* M_β*, where* $\beta = \kappa/(1-\kappa)$*, and the small margin condition constant can be chosen as* $c = 2^{\kappa/(1-\kappa)} d^{1/(1-\kappa)}$.

Lemma 2.13. *If the joint distribution of observation and classification satisfies the small margin condition* M_β $(\beta < \infty)$ *with small margin condition constant* c*, then it has margin parameter* $\kappa = \beta/(1+\beta)$ *with margin constant* $d = (1+\beta)c^{1+\beta}(2\beta)^{-\beta/(1+\beta)}$.

3. PAC learning with classification noise

A functional learning algorithm of a concept class $\mathcal{C}_n$ by a representation class $\mathcal{H}_n$ of $\mathcal{C}_n$ is specified as follows. It takes the learning sample $\mathbf{U}_m$, the desired error bound $\epsilon \in (0,1)$ and confidence $\delta \in (0,1)$ as input. Moreover, it knows an upper bound of the representation size s of the target concept g^* with respect to the representation class. It outputs a hypothesis $\hat{H}_m \in \mathcal{H}_n$ that ϵ-approximates the target concept with probability at least $1-\delta$ in the sense of equation (3.1).

Let us give a definition of a general classification noise model and a definition of efficient PAC learnability for any such classification noise model that subsume all special cases mentioned in Section 1 including our own new model introduced in Section 4 in a canonical way.

Definition 3.1. A classification noise model $\mathcal{N}_n$ with respect to a concept class $\mathcal{C}_n$ consists for every target concept $g^* \in \mathcal{C}_n$ of the set $\mathcal{N}_{g^*}$ of random noise rates $\nu(X)$ defined in equation (1.1) that are admitted for g^*.

Definition 3.2. A functional learning algorithm A is called an efficient PAC learner of a concept class $\mathcal{C}_n$ by a representation class $\mathcal{H}_n$ in the noise model $\mathcal{N}_n$, if

- for any $\epsilon, \delta \in (0,1)$, for any length n, for any target concept size bound s, and for any expected noise rate bound $\nu^{(b)} < 1/2$, there is minimal sample length $\mathbf{m}_A(\epsilon,\delta,n,s,\nu^{(b)})$ such that for every $m \geq \mathbf{m}_A(\epsilon,\delta,n,s,\nu^{(b)})$, for any distribution $\mathbb{P}_X$ of the input element X on $\mathfrak{X}_n$, for any target concept $g^* \in \mathcal{C}_n$ of size at most s, and any noise rate $\nu(X) \in \mathcal{N}_{g^*}$ whose expectation is less than or equal to $\nu^{(b)}$ it returns a hypothesis $\hat{H}_m$ such that with probability at least $1-\delta$

$$\mathbf{err}\big(\hat{H}_m\big) := \mathbb{P}\left(\hat{H}_m(X) \neq g^*(X) \mid \mathbf{U}_m\right) \leq \epsilon; \tag{3.1}$$

- the minimal sample length $\mathbf{m}_A(\epsilon,\delta,n,s,\nu^{(b)})$ is polynomial in $1/\epsilon$, $\ln(1/\delta)$, n, s, and $1/(1/2-\nu^{(b)})$;
- its running time is polynomial in m, $1/\epsilon$, $\ln(1/\delta)$, n, s, and $1/(1/2-\nu^{(b)})$.

Statistical queries are an important building block of PAC learning with classification noise.

Definition 3.3. Given a measurable and efficiently computable *query function*

$$\chi : \mathfrak{X}_n \times \{0,1\} \to [0,1]$$

and an *accuracy* $1/\tau$, where $\tau > 1$, making a statistical query $[\chi, 1/\tau]$, the algorithm requests for an estimate $\hat{e}_{\chi,g^*}$ of the expected value

$$e_{\chi,g^*} := \mathbb{E}\left(\chi(X, g^*(X))\right) \tag{3.2}$$

such that $|e_{\chi,g^*} - \hat{e}_{\chi,g^*}| \leq 1/\tau$. For $y_0 \in \{0,1\}$, a conditional statistical query (CSQ) $[\chi, 1/\tau, y_0]$ is a request for an estimate $\hat{e}_{\chi,g^*,y_0}$ of the conditional expectation

$$e_{\chi,g^*,y_0} := \mathbb{E}\left(\chi(X, y_0) \mid g^*(X) = y_0\right) \tag{3.3}$$

with accuracy $1/\tau$.

Let us call the learning algorithms in Kearns' SQ model *SQ learning algorithms.* They are specified as follows. A SQ learning algorithm of a concept class $\mathcal{C}_n$ by a representation class $\mathcal{H}_n$ is given the error bound ϵ as input, and it knows the length n of the observations as well as an upper bound s of the size of the target concept $g^* \in \mathcal{C}_n$. It is given access to an oracle STAT $(\mathbb{P}_X, g^*)$. The oracle answers queries $[\chi, 1/\tau]$ for expected values defined in equation (3.2) or queries $[\chi, 1/\tau, y_0]$ for conditional expected values defined in equation (3.3). The algorithm A outputs a hypothesis $H \in \mathcal{H}_n$.

As we have seen, SQ learning algorithms are restricted when using the training data. They make queries rather than scanning individual instances of the learning sample.

Definition 3.4. The set of query functions $\mathcal{Q}_n$ used on observations of length n is called the query space of the algorithm.

The effective query space $\mathcal{Q}_{\epsilon,n,s}$ consists of those query functions used when additionally the target concept size is bounded from above by s, and the algorithm is given input to ϵ as error bound.

In many cases efficient SQ learning algorithms in the sense of Definition 3.5 have query spaces of size polynomial in n.

If a subspace $\mathcal{Q}'$ of a query space $\mathcal{Q}_n$ is under study, for the sake of simplifying notations we refer to the Rademacher averages as well as to the VC dimension of the set

$$\left\{\theta \,\middle|\, \exists \chi \in \mathcal{Q}' : \exists y_0, y_1 \in \{0,1\} : \forall (x,y) \in \mathfrak{U}_n : \theta(x,y) = \mathbb{1}_{\{y=y_0\}}\chi(x,y_1)\right\}$$

as Rademacher averages and VC dimension of the query subspace $\mathcal{Q}'$.

According to [31], in the SQ model efficient consistency is defined as follows.

Definition 3.5. A SQ learning algorithm A is called an efficient and consistent learner of a concept class $\mathcal{C}_n$ by a representation class $\mathcal{H}_n$, if for any error bound ϵ, any observation length n, any target size bound s, any distribution $\mathbb{P}_X$ of the input element X on $\mathfrak{X}_n$, and any target concept $g^* \in \mathcal{C}_n$ of size s

- the output $h \in \mathcal{H}_n$ satisfies
$$\mathbf{err}(h) := \mathbb{P}\left(h(X) \neq g^*(X)\right) \leq \epsilon; \tag{3.4}$$
- for every statistical query made the reciprocal τ of the accuracy is bounded from above by what is referred to as the tolerance bound $\mathbf{tb}(\epsilon, n, s)$, which in turn is a polynomial in $1/\epsilon$, n, and s;
- the evaluation time of every query function used is polynomial in $1/\epsilon$, n, and s;
- the running time is polynomial in $1/\epsilon$, n, and s.

4. Generalizing the probabilistic model of PAC learning

We base our new generalized model of PAC learning a concept class $\mathcal{C}_n$ by a hypothesis class $\mathcal{H}_n$ on the Mammen–Tsybakov small margin conditions presented in Section 2. That is why a closer look to the definition of the observation-dependent noise rate $\nu(x)$ (see Eq. (1.1)) and the posterior probability $\eta(x)$ is useful. It reveals that if $\eta(x) > 1/2$, then $\mathbb{P}(Y = 1 \mid X = x) = 1 - \nu(x)$. If, however, $\eta(x) \leq 1/2$, then $\mathbb{P}(Y = 1 \mid X = x) = \nu(x)$. Consequently, we get

$$\frac{1}{2} - \nu(x) = \left|\frac{1}{2} - \eta(x)\right| \quad (x \in \mathfrak{X}_n). \tag{4.1}$$

In particular, equation (2.1) and all definitions and lemmas concerning small margin conditions and margin parameters can be reformulated in terms of the noise rate.

At the first glance it seems that PAC learning algorithms according to Definition 3.2 and the statistical-learning-theory-based approach to pattern classification outlined in Section 2 have different objective functions. On the supposition that a Mammen–Tsybakov small margin condition is imposed, it turns out that this is not the case. On the one hand we have

$$\mathbf{r}\big(\hat{H}_m\big) - \mathbf{r}\big(g^*\big) \leq \mathbf{err}\big(\hat{H}_m\big) \tag{4.2}$$

by equation (2.1). On the other hand, Lemma 2.13 states that

$$\mathbf{err}\big(\hat{H}_m\big) \leq d \cdot \Big(\mathbf{r}\big(\hat{H}_m\big) - \mathbf{r}\big(g^*\big)\Big)^{\kappa},$$

where $\kappa \in (0, 1]$ is the margin parameter, and $d > 0$ is the margin parameter constant.

The remainder of this section is organized as follows. The random noise rate $\nu(X)$ is a conditional probability. The key features of conditional probabilities and conditional expectations we need to prove our results are compiled in Section 4.1.

In Section 4.2 we explain the assumptions for the joint distribution $\mathbb{P}_U$. They are given by the equations (4.6)–(4.8).

In Section 4.3 we prove consistency of our mixed strategy of learning. The results of this subsections are subsumed in Lemma 4.4 and Corollary 4.5.

As a useful tool to present generic learning algorithms when the data are noisy we introduce PAC + SQ learners in Section 4.4. These algorithms have access to the learning sample $\mathbf{U}_m$ as well as to an oracle STAT $(\mathbb{P}_X, g^*)$.

PAC + SQ learners are not always operational, since it is not clear how to perform the (conditional) statistical queries in general. In Section 4.5 we define the noise model orthogonal to a set of query functions $\mathcal{Q}_n$ that allows us to efficiently simulate PAC + SQ learners using $\mathcal{Q}_n$ as query space given the joint distribution of observation and classification is obeying that noise model.

The advantage of our model is the following. Theorem 2.8 will not help to learn a concept class, if the empirical risk minimization problem for the corresponding learning samples does not have a polynomial time approximation scheme, or such a scheme has not been devised yet. From a practical point of view we can then proceed as follows. First, we devise a discrimination rule $\hat{H}_m$ implemented by a PAC + SQ learner B with query space $\mathcal{Q}_n$ presented in Section 4. Second, we simulate B by a PAC learner A in the noise model $\mathcal{Q}_n^{\perp}$ according to Definitions 3.2 and 4.7 as exposed in Sections 5 and 6. Thus we have got consistency in that model though ERM cannot be efficiently carried out for all possible inputs. The price we have to pay in practice is the model error.

4.1. Conditional probabilities and conditional expected values

Let $\mathcal{N}$ be a measurable subset of the probability space on which all random elements of this paper are defined, and let $\mathfrak{X}'$ be a measurable subset of the input space $\mathfrak{X}_n$. Then the conditional probability $\mathbb{P}(\mathcal{N} \mid X)$ satisfies the following key equation.

$$\mathbb{P}(\{X \in \mathfrak{X}'\} \cap \mathcal{N}) = \int_{\mathfrak{X}'} \mathbb{P}(\mathcal{N} \mid X = x)\, \mathrm{d}\,\mathbb{P}_X(x) \tag{4.3}$$

Having denoted $\mathbb{P}(\mathcal{N} \mid X)$ by $\zeta(X)$, according to equation (4.3) and the definition of expected values conditioned on measurable events the following equation holds true.

$$\int_{\mathfrak{X}'} \zeta(x)\, \mathrm{d}\,\mathbb{P}_X(x) = \mathbb{P}(X \in \mathfrak{X}') \cdot \mathbb{E}(\zeta(X) \mid X \in \mathfrak{X}') \tag{4.4}$$

4.2. Distribution assumptions and their consequences

We assume the random classification to satisfy the equation

$$Y = g^*(X) \oplus f(X, G), \tag{4.5}$$

where "$\oplus$" denotes the exclusive or, and g^* is the unknown Bayes classifier, which is referred to as target concept, that belongs to a known target concept class $\mathcal{C}_n$. Therein, $G \in [0,1]$ is uniformly distributed and independent of the random input element $X \in \mathfrak{X}_n$. The measurable function f is given by

$$f(x, g) = \begin{cases} 1 & \text{if } g \geq 1 - \nu'(x); \\ 0 & \text{otherwise,} \end{cases} \tag{4.6}$$

where $x \in \mathfrak{X}_n$, $g \in [0,1]$, and ν' is a function from the input space $\mathfrak{X}_n$ to the interval $[0, 1/2]$. It is plain that $\nu'(X)$ equals the random noise rate $\nu(X)$ defined by equation (1.1).

We assume the joint distributions $\mathbb{P}_U$ under study to uniformly have a margin parameter κ and margin parameter constant d.

By Lemmas 2.11 and 2.12 the distribution $\mathbb{P}_U$ then satisfies the small margin condition M_β, where the small margin condition parameter equals

$$\beta = \kappa/(1-\kappa). \tag{4.7}$$

The small margin condition constant can be chosen as

$$c = \begin{cases} 2^{\kappa/(1-\kappa)} d^{1/(1-\kappa)} & \text{if } \kappa < 1; \\ \frac{1}{2d} & \text{if } \kappa = 1. \end{cases} \tag{4.8}$$

Taking pattern from [31], we assume without loss of generality that

$$\mathbb{P}\left(g^*(X) = y_0\right) \geq \epsilon \qquad (y_0 \in \{0,1\}), \tag{4.9}$$

when analyzing a learning algorithm, where ϵ is the error bound the algorithm is given as input. Otherwise a trivial hypothesis that outputs a Boolean constant is of sufficient accuracy.

Let us turn to some consequences of the foregoing assumptions.

Using equation (4.1) in combination with the fact that $\mathbb{P}_U$ satisfies the small margin condition M_β with small margin condition constant c (see Eqs. (4.7) and (4.8)), it follows by means of analytic standard techniques that

$$\nu \leq \nu^{(b)} := \begin{cases} \frac{1}{2} - \frac{1}{2}\left(\frac{\beta+1}{2c}\right)^{1/\beta} & \text{if } \kappa < 1; \\ \frac{1}{2} - c & \text{if } \kappa = 1. \end{cases}. \tag{4.10}$$

(The full proof of Eq. (4.10) is presented in the appendix.)

For $y_0 \in \{0,1\}$, the *conditional expected noise rate* given $\{g^*(X) = y_0\}$ is denoted by

$$\nu_{y_0} := \mathbb{E}\left(\nu(X) \mid g^*(X) = y_0\right). \tag{4.11}$$

It follows from $\nu = \mathbb{E}\left(\nu(X)\right) = \mathbb{P}\left(g^*(X) = 0\right)\nu_0 + \mathbb{P}\left(g^*(X) = 1\right)\nu_1$ and equation (4.10) that $\nu_0 + \nu_1 \leq 1/2 + \nu^{(b)}$, whence

$$1 - \nu_0 - \nu_1 \geq \frac{1}{2} - \nu^{(b)}. \tag{4.12}$$

Lemma 4.1 (Observed classification probabilities *vs.* Bayes ones)**.** *For* $y_0 \in \{0,1\}$ *and any target concept* g^*,

$$\mathbb{P}\left(g^*(X) = y_0\right) = \frac{\mathbb{P}\left(Y = y_0\right) - \nu_{\bar{y}_0}}{1 - \nu_1 - \nu_0}. \tag{4.13}$$

Proof. It suffices to study case $Y = 1$. Since

$$\int_{\mathfrak{X}_n} \mathbb{P}\left(Y = 1 \mid X = x\right) \mathrm{d}\,\mathbb{P}_X(x) = \int_{\{x \mid g^*(x)=1\}} \underbrace{\mathbb{P}\left(Y = 1 \mid X = x\right)}_{=1-\nu(x)} \mathrm{d}\,\mathbb{P}_X(x)$$
$$+ \int_{\{x \mid g^*(x)=0\}} \underbrace{\mathbb{P}\left(Y = 1 \mid X = x\right)}_{=\nu(x)} \mathrm{d}\,\mathbb{P}_X(x),$$

equations (4.3) and (4.4) entail

$$\mathbb{P}(Y=1) = \mathbb{P}(g^*(X)=1)\cdot(1-\nu_1) + \mathbb{P}(g^*(X)=0)\cdot\nu_0\,.$$

Replacing $\mathbb{P}(g^*(X)=0)$ by $1-\mathbb{P}(g^*(X)=1)$ and solving for $\mathbb{P}(g^*(X)=1)$, equation (4.13) follows. □

Corollary 4.2. *For $y_0 \in \{0,1\}$, the observed classification probabilities can be bounded from below as follows.*

$$\mathbb{P}(Y=y_0) - \nu_{\bar{y}_0} \geq (1/2 - \nu^{(b)})\epsilon. \tag{4.14}$$

Proof. Equation (4.14) follows on grounds of equation (4.9) by means of equations (4.12) and (4.13). □

4.3. Substantiating a mixed strategy of learning

In order to find a hypothesis of small error, it is standard

- either to minimize the error given $\{g^*(X)=1\}$ and then the error given $\{g^*(X)=0\}$ by means of (conditional) statistical queries;
- or to minimize the risk given $\{Y=1\}$ and then the risk given $\{Y=0\}$ by ERM techniques.

In what follows we show, that the following mixed strategy is also promising: minimize the error given $\{g^*(X)=1\}$ and then minimize the risk given $\{Y=0\}$.

For $y_0 \in \{0,1\}$ and for any hypothesis $h \in \mathcal{H}_n$,

$$\mathbf{err}(h \mid g^*(X)=y_0) := \mathbb{P}(h(X)=\bar{y}_0 \mid g^*(X)=y_0)$$

denotes the *conditional error* of h given $\{g^*(X)=y_0\}$, and

$$\mathbf{r}(h \mid Y=y_0) := \mathbb{P}(h(X)=\bar{y}_0 \mid Y=y_0)$$

the *conditional risk* of h given $\{Y=y_0\}$, where $\bar{y}_0$ is the negation of $y_0 \in \{0,1\}$.

Lemma 4.3. *For $y_0 \in \{0,1\}$,*

$$\mathbf{r}(g^* \mid Y=y_0) = \frac{\nu_{\bar{y}_0}\cdot\mathbb{P}(g^*(X)=\bar{y}_0)}{\mathbb{P}(Y=y_0)}; \tag{4.15}$$

$$\mathbb{P}(g^*(X)=y_0 \mid Y=y_0) = \frac{(1-\nu_{y_0})\cdot\mathbb{P}(g^*(X)=y_0)}{\mathbb{P}(Y=y_0)}. \tag{4.16}$$

Proof. Clearly, it suffices to prove equation (4.15), which in turn follows immediately from

$$\nu_{\bar{y}_0} = \mathbb{P}(Y=y_0 \mid g^*(X)=\bar{y}_0)\,. \tag{4.17}$$

Having replaced $\mathcal{N}$ by $\{g^*(X)\neq Y\}$ and $\mathfrak{X}'$ by $\{x\in\mathfrak{X}_n \,\big|\, g^*(x)=\bar{y}_0\}$ in equations (4.3) and (4.4), equation (4.17) follows by bringing equations (4.3) and (4.4) together. □

Lemma 4.4. *For any hypothesis $h \in \mathcal{H}_n$,*

$$\begin{aligned}\mathbb{P}\left(h(X)=1, g^*(X)=0\right) \\ < 2(\mathbf{r}\left(h \mid Y=0\right) - \mathbf{r}\left(g^* \mid Y=0\right)) + \mathbb{P}\left(h(X)=0, g^*(X)=1\right). \quad (4.18)\end{aligned}$$

Proof. By definition

$$\mathbf{r}\left(h \mid Y=0\right) = \frac{1}{\mathbb{P}\left(Y=0\right)}\,\mathbb{E}\left(h(X)\mathbb{1}_{\{Y=0\}}\right).$$

Since $Y=0$ if and only if either $g^*(X)=0$ and $G<1-\nu(X)$ or $g^*(X)=1$ and $G \geq 1-\nu(X)$, we get

$$\begin{aligned}\mathbf{r}\left(h \mid Y=0\right) =& \frac{1}{\mathbb{P}\left(Y=0\right)}\,\mathbb{E}\left(h(X)\mathbb{1}_{\{g^*(X)=0\}}\mathbb{1}_{\{G<1-\nu(X)\}}\right) \\ &+ \frac{1}{\mathbb{P}\left(Y=0\right)}\,\mathbb{E}\left(h(X)\mathbb{1}_{\{g^*(X)=1\}}\mathbb{1}_{\{G\geq 1-\nu(X)\}}\right).\end{aligned}$$

Using the independence of X and G, it follows that

$$\begin{aligned}\mathbf{r}\left(h \mid Y=0\right) =& \frac{1}{\mathbb{P}\left(Y=0\right)}\,\mathbb{E}\left(h(X)\mathbb{1}_{\{g^*(X)=0\}}(1-\nu(X))\right) \\ &+ \frac{1}{\mathbb{P}\left(Y=0\right)}\,\mathbb{E}\left(h(X)\mathbb{1}_{\{g^*(X)=1\}}\,\nu(X)\right).\end{aligned}$$

We obtain

$$\begin{aligned}\mathbf{r}\left(h \mid Y=0\right) =& \frac{\mathbb{P}\left(g^*(X)=0\right)}{\mathbb{P}\left(Y=0\right)}\,\mathbb{E}\left(h(X)(1-\nu(X)) \mid g^*(X)=0\right) \\ &+ \frac{\mathbb{P}\left(g^*(X)=1\right)}{\mathbb{P}\left(Y=0\right)}\,\mathbb{E}\left(h(X)\,\nu(X) \mid g^*(X)=1\right).\end{aligned}$$

By equation (4.15)

$$\begin{aligned}\mathbf{r}\left(h \mid Y=0\right) - \mathbf{r}\left(g^* \mid Y=0\right) =& \frac{\mathbb{P}\left(g^*(X)=0\right)}{\mathbb{P}\left(Y=0\right)}\,\mathbb{E}\left(h(X)(1-\nu(X)) \mid g^*(X)=0\right) \\ &- \frac{\mathbb{P}\left(g^*(X)=1\right)}{\mathbb{P}\left(Y=0\right)}\,\mathbb{E}\left(\bar{h}(X)\,\nu(X) \mid g^*(X)=1\right).\end{aligned}$$

Since $\nu(X)$ is less than or equal to $1/2$, we get

$$\begin{aligned}\mathbf{err}\left(h \mid g^*(X)=0\right) <& \frac{2\,\mathbb{P}\left(Y=0\right)}{\mathbb{P}\left(g^*(X)=0\right)}\left(\mathbf{r}\left(h \mid Y=0\right) - \mathbf{r}\left(g^* \mid Y=0\right)\right) \\ &+ \frac{\mathbb{P}\left(g^*(X)=1\right)}{\mathbb{P}\left(g^*(X)=0\right)}\,\mathbf{err}\left(h \mid g^*(X)=1\right), \quad (4.19)\end{aligned}$$

whence the claim follows. □

From equation (4.19) we immediately get the following corollary, which is useful when it comes to implementing the mixed strategy of learning by means of conditional queries.

Corollary 4.5. *Given* $\min_{y_0 \in \{0,1\}} \mathbb{P}\left(g^*(X) = y_0\right) \geq \epsilon$*, for any hypothesis* $h \in \mathcal{H}_n$*,*

$$\begin{aligned}&\mathbf{err}\left(h \mid g^*(X) = 0\right) < \\ &\qquad \big(2(\mathbf{r}\left(h \mid Y = 0\right) - \mathbf{r}\left(g^* \mid Y = 0\right)) + \mathbf{err}\left(h \mid g^*(X) = 1\right)\big)/\epsilon. \qquad (4.20)\end{aligned}$$

4.4. PAC + SQ learning algorithms

A learning algorithm of a target concept class $\mathcal{C}_n$ by a representation class $\mathcal{H}_n$ in this combined model takes the learning sample $\mathbf{U}_m$ and the parameters *error accuracy* $\epsilon \in (0,1)$ and *confidence* $\delta \in (0,1)$ as input. It knows the size s of the target concept $g^* \in \mathcal{C}_n$, the margin parameter κ and the margin parameter constant d. Moreover, it has access to an oracle $\mathrm{STAT}\left(\mathbb{P}_X, g^*\right)$ to make (conditional) statistical queries $[\chi, 1/\tau]$ and $[\chi, 1/\tau, y_0]$ for expected values defined in equations (3.2) and (3.3), where χ is a query function, τ is the tolerance of the query, and $y_0 \in \{0,1\}$ is a classification. It outputs a hypothesis $\hat{H}_m$ that belongs to the representation class $\mathcal{H}_n$.

The following definition joins Definition 3.2 with Definition 3.5 in such a way that each efficient PAC learner as well as each efficient and consistent SQ learner is an efficient PAC + SQ learning algorithm, too.

Definition 4.6. A learning algorithm specified as above is called an efficient PAC + SQ learning algorithm of the target concept class $\mathcal{C}_n$ by the representation class $\mathcal{H}_n$, if

- for any $\epsilon, \delta \in (0,1)$, for any length n, for any target concept size bound s, and for any expected noise rate bound $\nu^{(b)} < 1/2$, there is a minimal sample length $\mathbf{m}_A(\epsilon, \delta, n, s, \nu^{(b)})$ and a tolerance bound $\mathbf{tb}_A(\epsilon, n, s)$ such that for every $m \geq \mathbf{m}_A(\epsilon, \delta, n, s, \nu^{(b)})$, for any distribution $\mathbb{P}_X$ of the input element X on $\mathfrak{X}_n$, for any target concept $g^* \in \mathcal{C}_n$ of size at most s, and for any noise rate $\nu(X)$ whose expectation ν is less than or equal to $\nu^{(b)}$

  $$\mathbb{P}\left(\mathbf{err}\big(\hat{H}_m\big) \leq \epsilon\right) \geq 1 - \delta,$$

  where the reciprocal of the accuracy of every (conditional) statistical query made is bounded from above by the tolerance bound $\mathbf{tb}_A(\epsilon, n, s)$;
- the minimal sample length is polynomial in $1/\epsilon$, $\ln(1/\delta)$, n, s, and $1/(1/2 - \nu^{(b)})$;
- the tolerance bound is polynomial in $1/\epsilon$, n, and s;
- the evaluation time of every query function χ is polynomial in n;
- the overall running time is polynomial in m, $1/\epsilon$, $\ln(1/\delta)$, n, s, and $1/(1/2 - \nu^{(b)})$.

The problem of learning rectangles in n dimensions in the noise free model was solved in [10]. Their algorithm computes the smallest axis-aligned rectangle consistent with the learning sample. As an example of a PAC + SQ learner, let us briefly expose the algorithm presented in [31], where ideas from [33] were used.

Let $U = (X, Y) = ((X_1, X_2, \ldots, X_j, \ldots, X_n), Y)$ be the generic data element of the learning sample $\mathbf{U}_m$, and let ϵ and δ be sufficiently small.

Taking the learning sample $\mathbf{U}_m$ as input, the algorithm divides for every dimension $j = 1, 2, \ldots, n$ the x_j-axis into $4n/\epsilon$ interval that are all approximately equally likely to be met by a random observation.

Applying the uniform convergence result of Theorem 2.7 along with equation (2.3), where $\mathcal{T}$ is the set of intervals, we get the following. If $m = \Omega\left((n^2/\epsilon^2)\ln(n/\delta)\right)$, then for every $j = 1, 2, \ldots, n$ and every such interval I on the x_j-axis with probability at least $1 - \delta$ the number $\mathbb{P}\left(X_j \in I\right)$ is at most $\epsilon/(2n)$.

Next, for every dimension $j = 1, 2, \ldots, n$ and every interval I on the x_j-axis, the algorithm makes a query for

$$p_I := \mathbb{P}\left(X_j \in I,\, g^*(X) = 1\right)$$

with accuracy $\epsilon^2/(16n^2)$. It classifies an interval I as significant, if the estimate of p_I is greater than $\epsilon^2/(16n^2)$.

Finally, the boundaries of the intervals whose cross-product forms the target concept are estimated separately for each dimension. Moving on the x_j-axis from the left to the right and then vice versa, the algorithm searches for the first significant interval to place the corresponding boundary in it.

4.5. The noise model orthogonal to a query space

For $g^* \in \mathcal{C}_n$ and $y_0 \in \{0, 1\}$, let us define the L^2-space $\mathrm{L}^2\big(\mathfrak{X}_n, g^*, y_0\big)$. We equip the set $\{x \in \mathfrak{X}_n \,|\, g^*(x) = y_0\}$ with the probability measure $\mathbb{P}_X$ conditioned on $\{g^*(X) = y_0\}$. Then $\mathrm{L}^2\big(\mathfrak{X}_n, g^*, y_0\big)$ is the space of all measurable functions

$$f : \{x \in \mathfrak{X}_n \,|\, g^*(x) = y_0\} \to \mathbb{R},$$

whose absolute value raised to the second power have a finite integral

$$\int |f(x)|^2 \,\mathrm{d}\,\mathbb{P}_X\left(x \mid g^*(X) = y_0\right).$$

It is known that then

$$\langle f_1, f_2 \rangle := \int f_1(x) f_2(x) \,\mathrm{d}\,\mathbb{P}_X\left(x \mid g^*(X) = y_0\right)$$

is a positive semidefinite bilinear form.

By conditioning on $\{g^*(X) = y_0\}$, every real-valued L^2-transform $t(X)$ of the input element X gives rise to an element of the space $\mathrm{L}^2\big(\mathfrak{X}_n, g^*, y_0\big)$. For the sake of simplicity, it is denoted by $t(X)$, too.

For $\mathcal{Q}_n$ being any set of query functions, and $y_0 \in \{0,1\}$, let $\operatorname{span}_{y_0}(\mathcal{Q}_n)$ be the subspace of $\mathrm{L}^2(\mathfrak{X}_n, g^*, y_0)$ spanned by the transforms $\chi(X,0)$ and $\chi(X,1)$ of the random observation X, where χ ranges over all query functions belonging to $\mathcal{Q}_n$.

Definition 4.7. Let $\mathcal{C}_n$ be a target concept class, and $\mathcal{Q}_n$ be a set of query functions.

- A noise rate $\nu(X)$ is called orthogonal to $\mathcal{Q}_n$ given a target concept $g^* \in \mathcal{C}_n$, if, for $y_0 \in \{0,1\}$, and for every $f \in \operatorname{span}_{y_0}(\mathcal{Q}_n)$,
$$\langle \nu(X) - \nu_{y_0}\,,\, f(X) - \mathbb{E}\left(f(X) \mid g^*(X) = y_0\right)\rangle = 0, \tag{4.21}$$
where ν_{y_0} is the conditional expectation of $\nu(X)$ given $\{g^*(X) = y_0\}$.
- For every $g^* \in \mathcal{C}_n$, the noise model $\mathcal{Q}_n^{\perp}$ orthogonal to $\mathcal{Q}_n$ consists of the set $\mathcal{Q}_{g^*}^{\perp}$ of all noise rates that are orthogonal to $\mathcal{Q}_n$ given g^*.

The following lemma is obvious.

Lemma 4.8. *Let $\mathcal{C}_n$ be a target concept class, and $\mathcal{Q}_n$ be a set of query functions.*

A noise rate $\nu(X)$ is orthogonal to $\mathcal{Q}_n$ given a target concept $g^* \in \mathcal{C}_n$ if and only if
$$\mathbb{E}\left(\nu(X)\cdot\chi(X,1) \mid g^*(X) = y_0\right) = \nu_{y_0}\cdot\mathbb{E}\left(\chi(X,1) \mid g^*(X) = y_0\right) \tag{4.22}$$
$$\mathbb{E}\left(\nu(X)\cdot\chi(X,0) \mid g^*(X) = y_0\right) = \nu_{y_0}\cdot\mathbb{E}\left(\chi(X,0) \mid g^*(X) = y_0\right), \tag{4.23}$$
for every $\chi \in \mathcal{Q}_n$ and every $y_0 \in \{0,1\}$.

5. Simulating SQ and CSQ with respect to orthogonal noise rates given the target concept

For any query function $\chi : \mathfrak{X}_n \times \{0,1\} \to [0,1]$, $\chi(x,\bar{y})$ is denoted by $\bar{\chi}(x,y)$.

Let us turn to the centerpiece of the proof of Theorem 6.1.

Theorem 5.1. *Let χ be a query function, $g^* \in \mathcal{C}_n$ be a target concept, and $\nu(X)$ be a noise rate such that the equations (4.22) and (4.23) are satisfied.*

Then, for $y_0 \in \{0,1\}$,
$$\mathbb{E}\left(\chi(X, g^*(X))\right) = \frac{(1-\nu_0)\,\mathbb{E}\left(\mathbb{1}_{\{Y=1\}}\chi(X,1)\right) - \nu_0\,\mathbb{E}\left(\mathbb{1}_{\{Y=0\}}\chi(X,1)\right)}{1-\nu_0-\nu_1} + \frac{(1-\nu_1)\,\mathbb{E}\left(\mathbb{1}_{\{Y=0\}}\chi(X,0)\right) - \nu_1\,\mathbb{E}\left(\mathbb{1}_{\{Y=1\}}\chi(X,0)\right)}{1-\nu_0-\nu_1}, \tag{5.1}$$
and
$$\mathbb{E}\left(\chi(X,y_0) \mid g^*(X) = y_0\right) = \frac{(1-\nu_{\bar{y}_0})\,\mathbb{E}\left(\mathbb{1}_{\{Y=y_0\}}\chi(X,y_0)\right) - \nu_{\bar{y}_0}\,\mathbb{E}\left(\mathbb{1}_{\{Y=\bar{y}_0\}}\chi(X,y_0)\right)}{\mathbb{P}\left(Y=y_0\right) - \nu_{\bar{y}_0}}. \tag{5.2}$$

Proof. Because of the fact that X and G are independent and G is uniformly distributed on $[0,1]$, we have

$$\mathbb{E}\left(\chi(X,Y)\right) = \int \left(\int_0^1 \chi(x, g^*(x) \oplus f(x,\gamma))\,\mathrm{d}\,\gamma \right) \mathrm{d}\,\mathbb{P}_X(x),$$

whence

$$\mathbb{E}\left(\chi(X,Y)\right) = \mathbb{E}\left((1-\nu(X))\chi(X, g^*(X))\right) + \mathbb{E}\left(\nu(X)\chi(X, \neg g^*(X))\right). \tag{5.3}$$

Taking pattern from [4], we proceed as follows. Conditioning both summands of the right-hand side of the foregoing equation on $g^*(X)$ and applying (4.22) and (4.23), it follows that

$$\begin{aligned}\mathbb{E}\left(\chi(X,Y)\right) = {} & \mathbb{P}\left(g^*(X)=0\right)(1-\nu_0)\,\mathbb{E}\left(\chi(X,0)\ \mid g^*(X)=0\right)\\ & + \mathbb{P}\left(g^*(X)=1\right)(1-\nu_1)\,\mathbb{E}\left(\chi(X,1)\ \mid g^*(X)=1\right)\\ & + \mathbb{P}\left(g^*(X)=0\right)\nu_0\,\mathbb{E}\left(\chi(X,1)\ \mid g^*(X)=0\right)\\ & + \mathbb{P}\left(g^*(X)=1\right)\nu_1\,\mathbb{E}\left(\chi(X,0)\ \mid g^*(X)=1\right).\end{aligned} \tag{5.4}$$

Applying equation (5.4) to $\mathbb{1}_{\{y=y_0\}}\chi(x,y_0)$ and to $\mathbb{1}_{\{y=\bar{y}_0\}}\chi(x,y_0)$, where $y_0 \in \{0,1\}$ is fixed, we get

$$\begin{aligned}\mathbb{E}\left(\mathbb{1}_{\{y=y_0\}}\chi(x,y_0)\right) = {} & \mathbb{P}\left(g^*(X)=\bar{y}_0\right)\nu_{\bar{y}_0}\,\mathbb{E}\left(\chi(X,y_0)\ \mid g^*(X)=\bar{y}_0\right)\\ & + \mathbb{P}\left(g^*(X)=y_0\right)(1-\nu_{y_0})\,\mathbb{E}\left(\chi(X,y_0)\ \mid g^*(X)=y_0\right)\end{aligned} \tag{5.5}$$

and

$$\begin{aligned}\mathbb{E}\left(\mathbb{1}_{\{y=\bar{y}_0\}}\chi(x,y_0)\right) = {} & \mathbb{P}\left(g^*(X)=\bar{y}_0\right)(1-\nu_{\bar{y}_0})\,\mathbb{E}\left(\chi(X,y_0)\ \mid g^*(X)=\bar{y}_0\right)\\ & + \mathbb{P}\left(g^*(X)=y_0\right)\nu_{y_0}\,\mathbb{E}\left(\chi(X,y_0)\ \mid g^*(X)=y_0\right),\end{aligned} \tag{5.6}$$

respectively. Multiplying equation (5.5) by $1-\nu_{\bar{y}_0}$ and equation (5.6) by $\nu_{\bar{y}_0}$, we obtain

$$\begin{aligned}(1-\nu_{\bar{y}_0})\mathbb{E}\left(\mathbb{1}_{\{y=y_0\}}\chi(x,y_0)\right) = {} & \mathbb{P}\left(g^*(X)=\bar{y}_0\right)\nu_{\bar{y}_0}(1-\nu_{\bar{y}_0})\mathbb{E}\left(\chi(X,y_0)\ \mid g^*(X)=\bar{y}_0\right)\\ & + \mathbb{P}\left(g^*(X)=y_0\right)(1-\nu_0)(1-\nu_1)\\ & \times \mathbb{E}\left(\chi(X,y_0)\ \mid g^*(X)=y_0\right)\end{aligned} \tag{5.7}$$

and

$$\begin{aligned}\nu_{\bar{y}_0}\,\mathbb{E}\left(\mathbb{1}_{\{y=\bar{y}_0\}}\chi(x,y_0)\right) = {} & \mathbb{P}\left(g^*(X)=\bar{y}_0\right)\nu_{\bar{y}_0}(1-\nu_{\bar{y}_0})\,\mathbb{E}\left(\chi(X,y_0)\ \mid g^*(X)=\bar{y}_0\right)\\ & + \mathbb{P}\left(g^*(X)=y_0\right)\nu_0\,\nu_1\,\mathbb{E}\left(\chi(X,y_0)\ \mid g^*(X)=y_0\right).\end{aligned} \tag{5.8}$$

Substracting (5.8) from (5.7) yields

$$\begin{aligned}(1-\nu_{\bar y_0})\,\mathbb{E}\left(\mathbb{1}_{\{Y=y_0\}}\chi(X,y_0)\right) - \nu_{\bar y_0}\,\mathbb{E}\left(\mathbb{1}_{\{Y=\bar y_0\}}\chi(X,y_0)\right)\\ = \mathbb{P}\left(g^*(X)=y_0\right)(1-\nu_0)(1-\nu_1)\,\mathbb{E}\left(\chi(X,y_0)\ \mid g^*(X)=y_0\,\right)\\ - \mathbb{P}\left(g^*(X)=y_0\right)\nu_0\,\nu_1\,\mathbb{E}\left(\chi(X,y_0)\ \mid g^*(X)=y_0\,\right).\end{aligned} \tag{5.9}$$

Solving for $\mathbb{E}\left(\chi(X,y_0)\ \mid g^*(X)=y_0\,\right)$ and using equation (4.13), we obtain equation (5.2).

Using the definition of the conditional expectation, we obtain from equation (5.2)

$$\begin{aligned}&\mathbb{E}\left(\mathbb{1}_{\{g^*(X)=y_0\}}\cdot\chi(X,y_0)\right)\\ &\qquad = \frac{(1-\nu_{\bar y_0})\,\mathbb{E}\left(\mathbb{1}_{\{Y=y_0\}}\chi(X,y_0)\right) - \nu_{\bar y_0}\,\mathbb{E}\left(\mathbb{1}_{\{Y=\bar y_0\}}\chi(X,y_0)\right)}{1-\nu_0-\nu_1}\cdot\end{aligned} \tag{5.10}$$

Since $\chi(X,g^*(X)) = \mathbb{1}_{\{g^*(X)=1\}}\chi(X,1) + \mathbb{1}_{\{g^*(X)=0\}}\chi(X,0)$, equation (5.10) entails equation (5.1). □

Guessing is a usual technique when PAC learning is involved. This means, given we have to put into operation a discrimination rule $\hat H_m$ that depends on a r-dimensional real parameter vector $\rho = \left(\rho_1,\rho_2,\ldots,\rho_r\right)$ that cannot be directly observed, where ρ belongs to a *solution cube* $[a_1,b_2]\times[a_1,b_2]\times\ldots\times[a_r,b_r]$, we want to guess all entries of ρ with accuracy $1/\tau$. We divide the solution cube into $\lceil 2\tau(b_1-a_1)\rceil\cdot\lceil 2\tau(b_2-a_2)\rceil\cdot\ldots\lceil 2\tau(b_r-a_r)\rceil$ axis-parallel equally sized subcubes, compute a hypothesis on each of their geometric centers, which we call the *candidates* for the parameter vector ρ, and select that one that has the least empirical risk on a freshly drawn learning sample.

Now it is simple to get estimates $\hat e_{g^*,y_0}$, $\hat e_{\chi,g^*}$ and $\hat e_{\chi,g^*,y_0}$ of e_{g^*,y_0}, e_{χ,g^*} and e_{χ,g^*,y_0}. First, we estimate the two conditional expected noise rates by guessing. Second, we swap these guesses and standard ML-estimates for the exact conditional noise rates and the observable building blocks, respectively, into equations (4.13), (5.1), and (5.2).

In what follows, we provide the sensitivity analysis of these equations to determine the accuracy with which the various quantities must be estimated. We make use of the following lemma due to Aslam [4].

Lemma 5.2. *Let $0\le p,q,r\le 1$, $\tau\ge 1$ and let p be equal either to q/r or to $q\cdot r$ or to $q-r$. Then to obtain an estimate of p within additive accuracy bound $1/\tau$, it suffices to estimate q and r within tolerance $r/(3\tau)$, $(\sqrt{2}-1)/\tau$, and $1/(2\tau)$, if $p=q/r$, $p=q\cdot r$, and $p=q-r$, respectively.*

Sensitivity analysis of equation (4.13). Assume that we are given estimates $\hat e_{y_0}$, $\hat\nu_0$, and $\hat\nu_1$ of the quantities e_{y_0}, ν_0 and ν_1 with accuracy $(1/2-\nu^{(b)})/(6\tau)$. Then by Lemma 5.2 the estimate $\hat e_{g^*,y_0}$ for e_{g^*,y_0} obtained from equation (4.13) is within accuracy bound $1/\tau$ of e_{g^*,y_0} (see Eq. (4.12)).

Sensitivity analysis of equation (5.1). Assume that we are given estimates $\hat{e}_y$, $\hat{\nu}_y$, $\hat{e}_{\chi,y}$, and $\hat{e}_{\bar{\chi},y}$ with accuracy $(1/2-\nu^{(b)})(\sqrt{2}-1)/(12\tau)$. Then the estimate $\hat{e}_{\chi,g^*}$ from equation (5.1) is of accuracy $1/\tau$ (see Eq. (4.12)).
Sensitivity analysis of equation (5.2). Assume that we are given estimates $\hat{e}_{y_0}$, $\hat{\nu}_{\bar{y}_0}$, $\hat{e}_{\chi,y_0}$, and $\hat{e}_{\bar{\chi},y_0}$ of accuracy $\epsilon(1/2-\nu^{(b)})(\sqrt{2}-1)/(6\tau)$, where ϵ is the error bound the learner is given as input. Using equation (4.14), the estimate $\hat{e}_{\chi,g^*,y_0}$ of e_{χ,g^*,y_0} from equation (5.2) is within accuracy bound $1/\tau$ of e_{χ,g^*,y_0}.

6. Simulating PAC + SQ learners in the noise model orthogonal to the query space used

Let B be an efficient PAC + SQ learner according to Definition 4.6 of a concept class $\mathcal{C}_n$ by a representation class $\mathcal{H}_n$ with query space $\mathcal{Q}_n$. Formalizing ideas taken from [4,17,31,33], we construct a PAC learner A in the noise model $\mathcal{Q}_n^{\perp}$ orthogonal to the query space $\mathcal{Q}_n$ that simulates algorithm B.

It is not surprising, that if the simulator A is to ensure error ϵ and confidence $1-\delta$, the PAC + SQ learner B to be simulated has to be a little bit better. As we shall see in the course of our analysis,

$$\epsilon' := \frac{\epsilon^{1/\kappa}}{2d^{1/\kappa}} \quad \text{and} \quad \delta' := \frac{\delta}{4} \tag{6.1}$$

are sufficient for B, where κ is the margin parameter of the joint distribution of observation and classification, and d is the margin constant.

The simulator A of the PAC + SQ learner B is subdivided into two parts. In particular, the minimal sample length of Definition 3.2 then equals

$$\begin{aligned}\mathbf{m}_A(\epsilon,\delta,n,s,\nu^{(b)}) = \mathbf{m}_B(\epsilon',\delta',n,s,\nu^{(b)}) + \mathbf{m}_{A,1}(\epsilon,\delta,n,s,\nu^{(b)})\\ + \mathbf{m}_{A,2}(\epsilon,\delta,n,s,\nu^{(b)}),\end{aligned} \tag{6.2}$$

where $\mathbf{m}_{A,1}$ and $\mathbf{m}_{A,2}$ are the numbers of instances needed in the first and in the second part of A additional to the sample used by B.

In the first part, which we call the *selection part*, A selects a finite subset from $\mathcal{H}_n$ of size polynomial in $1/\epsilon$, $\ln(1/\delta)$, n, s, and $1/(1/2-\nu^{(b)})$ that satisfies the following condition. Given the sample length m_1 of this part is greater than or equal to $\mathbf{m}_B(\epsilon',\delta',n,s,\nu^{(b)})+\mathbf{m}_{A,1}(\epsilon,\delta,n,s,\nu^{(b)})$, with probability at least $1-\delta/2$ the best hypothesis $\hat{H}_{\text{opt}}$ this subset contains has error at most ϵ', whence by equation (4.2)

$$\mathbf{r}\big(\hat{H}_{\text{opt}}\big) - \mathbf{r}\big(g^*\big) \le \epsilon'. \tag{6.3}$$

Moreover, $\mathbf{m}_{A,1}$ has to be bounded from above by a polynomial in $1/\epsilon$, $\ln(1/\delta)$, n, s, and $1/(1/2-\nu^{(b)})$.

In the *minimization part* the algorithm A chooses a hypothesis $\hat{H}_m$ from the finite set selected out of the hypothesis class $\mathcal{H}_n$ in the first part such that with probability $1-\delta/2$

$$\mathbf{r}\big(\hat{H}_m\big) - \mathbf{r}\big(\hat{H}_{\mathrm{opt}}\big) \le \epsilon', \tag{6.4}$$

given the sample length m_2 of this part is greater than or equal to $\mathbf{m}_{A,2}(\epsilon, \delta, n, s, \nu^{(b)})$, where again $\mathbf{m}_{A,2}$ is polynomially bounded in $1/\epsilon$, $\ln(1/\delta)$, n, s, and $1/(1/2-\nu^{(b)})$.

Adding up the equations (6.3) and (6.4) and applying Definition 2.10 as well as equation (6.1) yields with probability at least $1-\delta$

$$\mathbf{err}\big(\hat{H}_m\big) \le \epsilon. \tag{6.5}$$

Finally, the running time of both parts has to be a polynomial in $1/\epsilon$, $\ln(1/\delta)$, n, s, and $\nu^{(b)}$.

Let us explain now, how these aims of the two parts of the simulator A are achieved.
Selection Part. Let $\mathbf{m}_B := \mathbf{m}_B(\epsilon', \delta', n, s, \nu^{(b)})$ be the minimal sample length of algorithm B, and let $\mathbf{tb}_B := \mathbf{tb}_B(\epsilon', n, s)$ be its tolerance bound.

Principally, algorithm A results from B by simulating B's SQ and CSQ as described in Section 5. In order to determine the accuracy needed for the building blocks $\hat{e}_y$, $\hat{\nu}_y$, $\hat{e}_{\chi,y}$, and $\hat{e}_{\bar{\chi},y}$ ($\chi \in \mathcal{Q}_n$, $y \in \{0,1\}$) of these simulations, we make use of the sensitivity analysis of the equations (5.1) and (5.2) made at the end of Section 5: if all queries that B makes are unconditional ones, $\Theta\left((1/2-\nu^{(b)})/\,\mathbf{tb}_B\right)$ suffices as accuracy. If, however, B makes at least one conditional query, $\Theta\left((1/2-\nu^{(b)})\epsilon/\,\mathbf{tb}_B\right)$ is sufficient.

In order to be able to guess the conditional expected noise rates, algorithm A first computes a list of candidates in such a way that this list contains an approximately correct candidate pair within the sufficient accuracy. Let us refer to this list length as *branching factor* $\mathbf{bf}_A(\epsilon, \delta, n, s, \nu^{(b)})$. In general, the solution cube equals $[0,1/2]\times[0,1/2]$. If, however, it is only necessary to guess either ν_0 or ν_1, then it equals $[0,1/2]$. Thus the branching factor $\mathbf{bf}_A(\epsilon, \delta, n, s, \nu^{(b)})$ is either equal to $\Theta\left(\mathbf{tb}_B^2\,/(1/2-\nu^{(b)})^2\right)$ or to $\Theta\left(\mathbf{tb}_B^2\,/((1/2-\nu^{(b)})^2\epsilon^2)\right)$ according to whether algorithm B makes only unconditional queries or this is not the case. If, moreover, algorithm B makes only conditional queries, where either $\{g^*(X)=0\}$ or $\{g^*(X)=1\}$ occurs as condition, then $\mathbf{bf}_A(\epsilon, \delta, n, s, \nu^{(b)})$ reduces to $\Theta\left(\mathbf{tb}_B\,/((1/2-\nu^{(b)})\epsilon)\right)$, where the list consists either of candidates for ν_0 or of candidates for ν_1, as the case may be.

Note that in the case of the constant classification noise model in [4, 5, 17] the branching factor is reduced from order of magnitude $\Theta\left(\mathbf{tb}(\epsilon,n,s)/(1-2\,\nu^{(b)})\right)$ down to $\Theta\left(\mathbf{tb}(\epsilon,n,s)\ln(1/(1-2\,\nu^{(b)}))\right)$ when it comes to using unconditional queries only. This result is important, if $1/2-\nu^{(b)}$ is small. For our purposes it suffices to use the above approach.

The simulator A of B goes through the list of candidate pairs of conditional expected noise rates following the line of B on each of them. However, each query call is replaced by an application of equation (5.1) or (5.2), where instead of the true conditional expected noise rates the current candidate pair is used.

We apply the uniform convergence rate result formulated in Lemma 2.4 to the class $\mathcal{Q}_{(\epsilon/(2d))^{1/\kappa},n,s}$. Because of the fact that all quantities $e_{\chi,y}$ and $e_{\bar{\chi},y}$ ($\chi \in \mathcal{Q}_{(\epsilon/(2d))^{1/\kappa},n,s}$, $y \in \{0,1\}$) needed in the course of the entire simulator do not depend on the conditional noise rates, they can be estimated on a common sample of length $\mathbf{m}_{A,1}(\epsilon,\delta,n,s,\nu^{(b)})$. By additionally using equation (2.3), we obtain the following.

If B makes only unconditional queries, then

$$\mathcal{O}\left(\mathbf{tb}_B^2(1/2-\nu^{(b)})^{-2}\big(\text{vc-dim}\big(\mathcal{Q}_{(\epsilon/(2d))^{1/\kappa},n,s}\big)+\ln(1/\delta)\big)\right) \tag{6.6}$$

is an upper bound on $\mathbf{m}_{A,1}(\epsilon,\delta,n,s,\nu^{(b)})$.

If B makes at least one conditional query, then

$$\mathcal{O}\left(\mathbf{tb}_B^2(1/2-\nu^{(b)})^{-2}\epsilon^{-2}\big(\text{vc-dim}\big(\mathcal{Q}_{(\epsilon/(2d))^{1/\kappa},n,s}\big)+\ln(1/\delta)\big)\right) \tag{6.7}$$

is an upper bound on $\mathbf{m}_{A,1}(\epsilon,\delta,n,s,\nu^{(b)})$.

That way A computes for every candidate pair of conditional expected noise rates one proposal for a hypothesis. This motivates to refer to the length of the candidate list as *branching factor* of the selection part. It is denoted by $\mathbf{bf}_A(\epsilon,\delta,n,s,\nu^{(b)})$ and abbreviated by $\mathbf{bf}_A$.

There are two possibilities for the selection part of A to fail. First, the learner B fails on its data in computing a hypothesis of error less than or equal to ϵ'. Second, the simulator fails in estimating one of the quantities $e_{\chi,y}$ and $e_{\bar{\chi},y}$ ($\chi \in \mathcal{Q}_{(\epsilon/(2d))^{1/\kappa},n,s}$, $y \in \{0,1\}$) it needs within the desired accuracy. Having bounded these two failure probabilities by $\delta' = \delta/4$, we get confidence $1-\delta/2$ for the selection part. If it does not fail, the error of proposal computed on the approximately correct candidate pair is at most ϵ'.

In order to control the running time of the selection phase, we think of it as an interlaced simulation of B for each candidate pair of conditional expected noise rates. At the first glance it seems that there is a problem, since B's polynomially bounded running time is only guaranteed when queries get correct answers. The other way round, the learning sample the algorithm B uses in addition to the queries is in most cases an i.i.d. instantiation of a generic data element $U = (X,Y)$ with Y being drawn under a wrong posterior probability from the point of view of the queries. But a closer look reveals the following. According to Definition 4.6, the running time of algorithm B is guaranteed for all noise rates $\nu(X)$ whose conditional expectations are less than or equal to $\nu^{(b)}$ and all compatible learning sample instances. An instance of the learning sample in turn is compatible if it has a nonzero probability to be drawn in the true distribution. The latter is still fulfilled when the sample is drawn from a distribution with true observation distribution and wrong conditional expected noise rates greater than zero (see Eqs. 4.6 and 4.5).
Minimization Part. Taking m_2 freshly drawn instances of the learning sample as input. The algorithm A computes a hypothesis $\widehat{\text{ERM}}_{m_2}$ that minimizes the empirical risk in the subset of $\mathcal{H}_n$ of cardinality $\mathbf{bf}_A(\epsilon,\delta,n,s,\nu^{(b)})$ computed in the selection part.

By Lemma 2.5 and Theorem 2.7, we easily get that the term

$$\mathcal{O}\left(d^{2/\kappa}\frac{\ln \mathbf{bf}_A + \ln(1/\delta)}{\epsilon^{2/\kappa}}\right) \tag{6.8}$$

is an upper bound on the minimal sample length $\mathbf{m}_{A,2}(\epsilon,\delta,n,s,\nu^{(b)})$ of the minimization part, where κ and d are the margin parameter and the margin constant.

Plugging our branching factor analysis into the equations (6.6–6.8), and applying equation (6.2) we complete the proof of the following theorem.

Theorem 6.1. *Let B be an efficient PAC+ SQ learner of a concept class $\mathcal{C}_n$ by a representation class $\mathcal{H}_n$ having effective query spaces $\mathcal{Q}_{\epsilon,n,s}$, minimal sample size $\mathbf{m}_B(\epsilon,\delta,n,s,\nu^{(b)})$, and tolerance bound $\mathbf{tb}_B(\epsilon,n,s)$, and let*

$$\mathbf{m}_B := \mathbf{m}_B((\epsilon/(2d))^{1/\kappa},\delta/4,n,s,\nu^{(b)}), \qquad \mathbf{tb}_B := \mathbf{tb}_B((\epsilon/(2d))^{1/\kappa},n,s),$$

where κ and d are the margin parameter and the margin constant.

Then this algorithm B can be simulated by an efficient PAC learner A of $\mathcal{C}_n$ by $\mathcal{H}_n$ in the noise model $\mathcal{Q}_n^\perp$ orthogonal to the query space $\mathcal{Q}_n$ of B such that the minimal sample size $\mathbf{m}_A(\epsilon,\delta,n,s,\nu^{(b)})$ of A can be bounded from above by

$$\mathcal{O}\left(\mathbf{m}_B + \frac{\mathbf{tb}_B^2\left(\text{vc-dim}\left(\mathcal{Q}_{(\epsilon/(2d))^{1/\kappa},n,s}\right)+\ln(1/\delta)\right)}{(1/2-\nu^{(b)})^2} + (d/\epsilon)^{2/\kappa}\ln\frac{\mathbf{tb}_B}{(1/2-\nu^{(b)})\delta}\right),$$

given B uses unconditional queries only, and by

$$\mathcal{O}\left(\mathbf{m}_B + \frac{\mathbf{tb}_B^2\left(\text{vc-dim}\left(\mathcal{Q}_{(\epsilon/(2d))^{1/\kappa},n,s}\right)+\ln(1/\delta)\right)}{(1/2-\nu^{(b)})^2\epsilon^2} + (d/\epsilon)^{2/\kappa}\ln\frac{\mathbf{tb}_B}{(1/2-\nu^{(b)})\epsilon\delta}\right)$$

otherwise.

7. Simulating statistical queries in the CPPO noise model

For the sake of completeness, let us recapitulate Decatur's CPCN model. It assumes a set of known 0/1-valued partition functions $\pi = \{\pi_1,\pi_2,\ldots,\pi_k\}$ on the learning universe $\mathfrak{U}_n$ such that $\sum_{i=1}^k \pi_i$ is the all-one function, and a sequence of non-negative constants $\tilde\nu_1,\tilde\nu_2,\ldots,\tilde\nu_k$ all less than 1/2. For every target concept class $\mathcal{C}_n$, we get a noise model by

$$\nu(x) = \tilde\nu_i \quad\Longleftrightarrow\quad \pi_i(x,g^*(x)) = 1,$$

where $x\in\mathfrak{X}_n$, $i=1,2,\ldots,k$, and $g^*\in\mathcal{C}_n$.

According to [44], this partition of the labelled data gives rise to the partition $\tilde\pi_{ij}$ $(i,j=1,2,\ldots,k)$ of the input space $\mathfrak{X}_n$ defined by

$$\tilde\pi_{ij}(x) = \pi_i(x,1)\cdot\pi_j(x,0).$$

For all $x \in \mathfrak{X}_n$ satisfying $\tilde{\pi}_{ij}(x) = 1$, we get

$$\nu(x) = \begin{cases} \tilde{\nu}_i & \text{if } g^*(x) = 1; \\ \tilde{\nu}_j & \text{if } g^*(x) = 0. \end{cases}$$

Let us now define the *constant-partition piecewise orthogonal* (CPPO) noise model that canonically generalizes the CPCN model.

Definition 7.1. Let $\mathcal{Q}_n$ be a set of query functions.

A noise rate $\nu(X)$ is called piecewise orthogonal to $\mathcal{Q}_n$ with respect to π given a target concept $g^* \in \mathcal{C}_n$, if

$$\begin{aligned} &\mathbb{E}\left(\nu(X) \cdot \chi(X, y_0) \mid g^*(X) = 1, \tilde{\pi}_{ij}(X) = 1\right) \\ &\qquad = \tilde{\nu}_i \cdot \mathbb{E}\left(\chi(X, y_0) \mid g^*(X) = 1, \tilde{\pi}_{ij}(X) = 1\right) \end{aligned} \tag{7.1}$$

$$\begin{aligned} &\mathbb{E}\left(\nu(X) \cdot \chi(X, y_0) \mid g^*(X) = 0, \tilde{\pi}_{ij}(X) = 1\right) \\ &\qquad = \tilde{\nu_j} \cdot \mathbb{E}\left(\chi(X, y_0) \mid g^*(X) = 0, \tilde{\pi}_{ij}(X) = 1\right), \end{aligned} \tag{7.2}$$

for all $i, j \in \{1, 2, \ldots, k\}$, every $y_0 = 0, 1$, and every $\chi \in \mathcal{Q}_n$, where

$$\tilde{\nu}_i = \mathbb{E}\left(\nu_{ij}(X) \mid g^*(X) = 1, \tilde{\pi}_{ij}(X) = 1\right)$$

and

$$\tilde{\nu}_j = \mathbb{E}\left(\nu_{ij}(X) \mid g^*(X) = 0, \tilde{\pi}_{ij}(X) = 1\right).$$

For every $g^* \in \mathcal{C}_n$, the noise model $\mathcal{Q}_n^{\perp}(\pi)$ piecewise orthogonal to $\mathcal{Q}_n$ with respect to π consists of the set $\mathcal{Q}_{g^*}^{\perp}(\pi)$ of all noise rates that are piecewise orthogonal to $\mathcal{Q}_n$ with respect to π given g^*.

Let χ be any query function. Then

$$\chi(x, y) = \sum_{i,j=1}^{k} \chi_{ij}(x, y) \qquad ((x, y) \in \mathfrak{U}_n), \tag{7.3}$$

where

$$\chi_{ij}(x, y) := \tilde{\pi}_{ij}(x) \cdot \chi(x, y).$$

We next study (conditional) statistical queries associated with the query function χ_{ij} in the noise model $\mathcal{Q}_n^{\perp}(\pi)$.

Theorem 7.2. *Let $g^* \in \mathcal{C}_n$ be any target concept.*
Then

$$\begin{aligned}&\mathbb{E}\left(\chi_{ij}(X,0) \mid g^*(X)=0\right)\\ &\quad = \frac{1-\nu_0-\nu_1}{1-\tilde{\nu}_i-\tilde{\nu}_j}\cdot\frac{(1-\tilde{\nu}_i)\,\mathbb{E}\left[\mathbb{1}_{\{Y=0\}}\chi_{ij}(X,0)\right]-\tilde{\nu}_i\,\mathbb{E}\left[\mathbb{1}_{\{Y=1\}}\chi_{ij}(X,0)\right]}{\mathbb{P}(Y=0)-\nu_1}, \qquad (7.4)\end{aligned}$$

$$\begin{aligned}&\mathbb{E}\left(\chi_{ij}(X,1) \mid g^*(X)=1\right)\\ &\quad = \frac{1-\nu_0-\nu_1}{1-\tilde{\nu}_i-\tilde{\nu}_j}\cdot\frac{(1-\tilde{\nu}_j)\,\mathbb{E}\left[\mathbb{1}_{\{Y=1\}}\chi_{ij}(X,1)\right]-\tilde{\nu}_j\,\mathbb{E}\left[\mathbb{1}_{\{Y=0\}}\chi_{ij}(X,1)\right]}{\mathbb{P}(Y=1)-\nu_0}, \qquad (7.5)\end{aligned}$$

and

$$\begin{aligned}\mathbb{E}\,\chi_{ij}(X,g^*(X)) &= \frac{(1-\tilde{\nu}_j)\,\mathbb{E}\left[\mathbb{1}_{\{Y=1\}}\chi_{ij}(X,1)\right]-\tilde{\nu}_j\,\mathbb{E}\left[\mathbb{1}_{\{Y=0\}}\chi_{ij}(X,1)\right]}{1-\tilde{\nu}_i-\tilde{\nu}_j}\\ &\quad+\frac{(1-\tilde{\nu}_i)\,\mathbb{E}\left[\mathbb{1}_{\{Y=0\}}\chi_{ij}(X,0)\right]-\tilde{\nu}_i\,\mathbb{E}\left[\mathbb{1}_{\{Y=1\}}\chi_{ij}(X,0)\right]}{1-\tilde{\nu}_i-\tilde{\nu}_j}. \qquad (7.6)\end{aligned}$$

Proof. Starting point is equation (5.3). Again conditioning both summands of the right-hand side of this equation on $g^*(X)$ and applying (7.1) and (7.2) instead of (4.22) and (4.23), it follows that

$$\begin{aligned}\mathbb{E}\,\chi_{ij}(X,Y) &= \mathbb{P}\left(g^*(X)=0\right)(1-\tilde{\nu}_j)\,\mathbb{E}\left(\chi_{ij}(X,0) \mid g^*(X)=0\right)\\ &\quad+\mathbb{P}\left(g^*(X)=1\right)(1-\tilde{\nu}_i)\,\mathbb{E}\left(\chi_{ij}(X,1) \mid g^*(X)=1\right)\\ &\quad+\mathbb{P}\left(g^*(X)=0\right)\tilde{\nu}_j\,\mathbb{E}\left(\chi_{ij}(X,1) \mid g^*(X)=0\right)\\ &\quad+\mathbb{P}\left(g^*(X)=1\right)\tilde{\nu}_i\,\mathbb{E}\left(\chi_{ij}(X,0) \mid g^*(X)=1\right). \qquad (7.7)\end{aligned}$$

The remaining part of this proof can be carried over from the proof of Theorem 5.1. □

Applying the well-known principle of linearity of expectation to equation (7.3), we get Equations analogous to (5.1) and (5.2). Carrying over the proof of Theorem 6.1 in a straightforward way, we obtain the second theoretical main result of this paper.

Theorem 7.3. *Let $\mathcal{H}_n$ be a representation class of a concept class $\mathcal{C}_n$.*
Every efficient PAC + SQ learner of $\mathcal{C}_n$ by $\mathcal{H}_n$ having query space $\mathcal{Q}_n$ can be simulated by an efficient PAC learner of $\mathcal{C}_n$ by $\mathcal{H}_n$ in the noise model $\mathcal{Q}_n^{\perp}(\pi)$, provided that the partition π is known to the learner.

As a corollary of Theorem 7.3, we get another proof of the known fact that every concept class efficiently learnable in Kearns' SQ model, is efficiently learnable in Decatur's CPCN model, too.

8. Haussler's covering method as PAC + SQ algorithm

Let us devise a PAC + SQ algorithm to efficiently learn Boolean conjunctions of clauses that uses only unconditional queries. A clause is here an efficiently computable 0/1-valued function on the input space. Each target concept g^* is then some conjunction of at most s elements from a set

$$\mathbf{C}_n := \{c_1(X), c_2(X), \dots, c_{n^\gamma}(X)\}$$

of n^γ clauses, where n is the length of the random observation $X \in \mathfrak{X}_n$, and γ is some positive constant. We regard the number s to be the size parameter of the target concept.

A solution of this problem in Valiant's noise-free mode was given by Haussler [26] who made use of a covering method. In [31] it is adapted to the constant classification noise model. As in [26] and in [31], our algorithm is subdivided into two phases, the prune phase and the cover phase.
The prune phase. It mainly differs from Kearns' algorithm in the fact that only unconditional queries are used. For every clause $c \in \mathbf{C}_n$, a query is made to get an estimate of the *failure probability* $\mathbb{P}\,(c(X) = 0, g^*(X) = 1)$ of c within accuracy $\Theta\left(\epsilon^2/s\right)$. Only those clauses whose estimate of the failure probability is an $\mathcal{O}\left(\epsilon^2/s\right)$ are retained as candidates. Let us refer to these clauses as *survivors of the prune phase* in the sequel.

Note that

- the true failure probability of each survivor of the prune phase equals $\mathcal{O}\left(\epsilon^2/s\right)$;
- the clauses forming the target concept g^* belong to the survivors of the prune phase;
- no matter the way we select the clauses from the set of survivors of the prune phase to form the hypothesis $\hat{H}_m$, we have

$$\mathbb{P}\left(\hat{H}_m = 0, g^*(X) = 1\right) = \mathcal{O}\left(\epsilon\right) \tag{8.1}$$

  as long as their number is an $\mathcal{O}\left(s/\epsilon\right)$.

The cover phase. It needs only negative examples and does not make any statistical query. Thus the overall query space used is determined by the prune phase. It equals

$$\mathcal{Q}_n = \{c_i(x) \wedge y \,|\, i = 1, 2, \dots, n^\gamma\}. \tag{8.2}$$

Taking a sample of negative instances of length m drawn according to the distribution $\mathbb{P}\,(X = \centerdot \mid Y = 0)$ as input, our cover phase routine first computes the subset of the input covered by the negations of all candidates retained in the prune phase. Then by means of the standard greedy algorithm this set is covered as far as this is possible in $r = \Theta\left(s \log(1/\epsilon)\right)$ iterations. Only those clauses involved in this covering are used to form the output hypothesis $\hat{H}_m$. Thus our cover phase acts on the negative examples in the same manner as Kearns' one on the true negatives [31].

A reader who insists on drawing the input sample of the cover phase according to $\mathbb{P}_U$ has to discard the positive examples.

According to Lemma 4.4 it suffices to get with probability greater than or equal to $1-\delta$ a hypothesis $\hat{H}_m$ such that

$$\mathbf{r}\left(\hat{H}_m \mid Y=0\right) - \mathbf{r}\left(g^* \mid Y=0\right) = \mathcal{O}\left(\epsilon\right).$$

For every survivor of the prune phase c, let FN_c be the number of falsely negatively declared instances covered by c. Then

$$\mathbb{E}\left(\mathrm{FN}_c\right) = \mathbb{P}\left(c(X)=0, g^*(X)=1 \mid Y=0\right)\cdot m = \mathcal{O}\left(\frac{\epsilon^2}{s\,\mathbb{P}\left(Y=0\right)}\cdot m\right)\cdot \tag{8.3}$$

All truly negatively declared instances can be covered by the set of the at most s true clauses. Consequently, in each iteration $i=1,2,\ldots,r$ of the cover phase as long as a percentage of at least ϵ of the truly negatively declared instances has not been covered yet, the expectation of the number TN_i of these true negatives that would become covered if we added the best true clause is by equation (4.16) greater than or equal to

$$\epsilon\cdot\frac{(1-\nu_0)\,\mathbb{P}\left(g^*(X)=0\right)}{\mathbb{P}\left(Y=0\right)s}\cdot m.$$

By equation (8.3) and the fact that $\mathbb{P}\left(g^*(X)=0\right)\geq\epsilon$, we have in each iteration $i=1,2,\ldots,r$ of the cover phase for any survivor c of the prune phase

$$\mathbb{E}\left(\mathrm{TN}_i\right) = \Omega\left(\mathbb{E}\left(\mathrm{FN}_c\right)\right). \tag{8.4}$$

Using the multiplicative Chernoff bounds, it follows from equation (8.4) that

$$m = \mathcal{O}\left((s/\epsilon^2)(\log(1/\delta)+\log n)\right)$$

suffices to ensure, that with probability greater than or equal to $1-\delta$ for any iteration $i=1,2,\ldots,r$ of the cover phase and any survivor c of the prune phase $\mathrm{TN}_i=\Omega\left(\mathrm{FN}_c\right)$, and with probability at least $1-\delta$ the clause selected in each iteration of the cover phase covers at least an $\Omega\left(1/s\right)$-percentage of the truly negatively declared instances.

Finally, in order to justify the size of the number r of iterations, we proceed exactly in the same way as in [31]. We solve $(1-c/s)^r=\mathcal{O}\left(\epsilon\right)$, where c is some constant, for r to obtain that $r=\mathcal{O}\left(s\log(1/\epsilon)\right)$.

Thus, our PAC + SQ learner for Boolean conjunctions of s clauses taken from a query space of cardinality n^γ has sample complexity $\mathcal{O}\left((\log(1/\delta)+\log n)s/\epsilon^2\right)$, and tolerance bound $\mathcal{O}\left(s/\epsilon^2\right)$.

In order to compare our algorithm with the adaptation of Haussler's covering method presented in [31], we recapitulate the sample size of the latter. Its dependence on n is $\log n$, on s is s^3, on ϵ is $1/\epsilon^4$, on $1/(1/2-\nu^{(b)})$ is quadratic and on $\ln(1/\delta)$ is linear. According to Theorem 6.1 our overall sample size dependence differs on s, where it is only quadratic, and on ϵ, where it is ϵ^{-4} only if the the margin parameter $\kappa \geq 1/2$, and $\epsilon^{-2/\kappa}$ otherwise.

Note, that in the standard classification noise model used by Kearns $\kappa = 1$.

9. Studying hotspots of protein-protein interfaces

In order to understand cellular processes it is crucial to identify protein-protein interactions, where we think of a protein as being a sequence of amino acid residues, and of an interaction of two or more proteins as of a spatial complex of them. In this section, we restrict our attention to complexes of two proteins, so-called dimers.

One of the major challenges in bioinformatics is the prediction of protein-protein interactions and the recognition of their binding sites, the so-called *interfaces*. Several networks of protein-protein interactions in cells have been unraveled by identifying interfaces in the last decade, and there is a number of approaches to predict protein-protein interactions. See [38] for an overview.

We use the following definition of an interface of a protein-protein interaction which is a variant of that given in [25]. It equals the *set of pairs of interacting residues* of the two partner proteins, where the one residue is in one partner, the other one in the other partner. Two residues in turn are defined as interacting if the distance between any of their atoms is less than the sum of their corresponding van der Walls radii plus 0.5 Å.

It is beyond dispute that only a few number of the interface residues are essential for the binding. These so-called *hotspots* are defined as those residues hampering the interaction if mutated.

Many approaches to predict protein-protein interactions by predicting the interfaces have a high level of *positive accuracy* (the number of correctly predicted interface residues divided by the number of predicted ones) but a poor *sensitivity* (residues correctly predicted as percentage of the interface residues observed.) In [41] some evidence is given to the hypothesis that the reason for that is that a machine-learning algorithm trained on all protein-protein interface residues only predicts the hotspot residues, wheras the non-hotspot residues are discarded as noise. This is done by comparing predictions of the ISIS tool [40] with hotspots experimentally determined.

Mining the data base ASEdb [46], tryptophan, tyrosine und arginine are found to be significantly enriched in hotspots.

We study the amino acid content of interfaces that is defined to be the sequence of relative frequencies of them in sets of pairs of interacting residues forming these interfaces. We compare it with the amino acid content of ordinary protein surface by means of our variant of Haussler's covering method described in Section 8. To this end, we have first to extract the positive and negative examples from the PDB database available at `http://www.wwpdb.org/`.

To get a sequence of interfaces as our negative examples, we used a non-redundant set of interacting proteins computed by the Nussinov–Wolfson

Structural Bioinformatics Group, where pairs are removed that are too similar. This set is available at http://protein3d.ncifcrf.gov. It is a list of 1835 pairs of interacting proteins. We computed 1835 interfaces by means of the Biochemical Algorithms Library (BALL) available at http://www.bioinf.uni-sb.de/OK/BALL/.

The 1835 positive examples of pair sets of protein surface residues were randomly chosen according to the cardinality profile of the interface examples and the surface distribution on the 20 amino acids derived from the PDB database. We used the cardinality profile of the interface examples for the positive examples to avoid its implicit usage when computing the discrimination rule.

To any set s of pairs of amino acids, interfaces as well as sets of randomly chosen surface pairs, and any amino acid a, we compute the frequency p_{sa} as the occurences of a in s divided by $2|s|$. To any amino acid a, we assign conditions c_{ap}, for $p = 3, 4, \ldots, 18$, where $c_{ap}(s) = 1$ if and only if $p_{sa} \leq p/100$. We have simulated the algorithm of Section 8 in the noise model orthogonal to

$$\{c_{ap} \,|\, a \text{ amino acid}, \, p = 3, 4, \ldots, 18\},$$

where we have aborted the cover phase after seven steps. The hypothesis learned equals

$$h = c_{\mathrm{Phe},6} \wedge c_{\mathrm{Tyr},7} \wedge c_{\mathrm{Val},10} \wedge c_{\mathrm{Met},5} \wedge c_{\mathrm{Trp},4} \wedge c_{\mathrm{Thr},13} \wedge c_{\mathrm{Leu},14}, \tag{9.1}$$

i.e. it classifies a set of pairs of amino acids as a set of random surface pairs if and only if the phenylalanine, tyrosine, valine, methionine, tryptophan, threonine, and leucine percentage is less than or equal to 6%, 7%, 10%, 5%, 4%, 13%, and 14%, respectively. The overall classification rate is 86.7%.

For us, the hypothesis learned is above all an expedient. None the less we had to compare it with the standard method utilized in bioinformatics. Using support vector machines (SVM) with Gaussian RBF kernel functions, we obtained a slightly better classification rate of 89.0%. However, the advantage of the PAC classifier given by equation (9.1) is the following biological interpretation.

The hypothesis h learned mirrors the fact that at least one of the amino acids phenylalanine, tyrosine, valine, methionine, tryptophan, threonine, or leucine are overrepresented in most of the hotspots compared with ordinary protein surfaces having assumed that the complements of hotspots in interfaces have an inconspicuous amino acid content according to [41]. Thus we have confirmed in part and complemented the findings presented in [46].

For more details of this application, see [13].

10. Studying software metrics

In the assessment of software quality, software metrics [23] play a crucial role since quality models like the ISO/IEC Standard No. 9126 [29] usually rely on them for the evaluation of quality attributes. A commonly used technique for metric-based evaluation of quality attributes is the application of thresholds. If a metric violates a threshold, the related quality attributes is not fulfilled. For example, consider the metric *Number of Critical Bugs* (CRITBUG) that measures the number of bugs deemed to be critical, *i.e.*, will lead to significant failures. For the evaluation whether the quality of a software is sufficient for release, a reasonable threshold for CRITBUG is 0, *i.e.*, no critical bugs are allowed. Often, a single metric is not sufficient to determine the quality, instead a set of metrics M is used. For example, there should not only be no critical bugs, but also the total number of bugs should be less than 20. In this scenario the metric *Number of Bugs* (BUG) with a threshold of 20 is needed in addition to CRITBUG, and we have a metric set $M = \{CRITBUG, BUG\}$.

The *effectiveness* of a metric set depends on the selection of software metrics and the quality of the thresholds values. We call a metric set with thresholds effective, if it evaluates a quality attribute accurately.

Furthermore, it is desirable to have *efficient* metric sets, *i.e.*, the sets should not contain redundant metrics and be as small as possible, to reduce the overall effort for the measurement.

Our aim is the optimization of metric sets in terms of their efficiency. We assume that the values of the software metrics in the set under consideration get worse when they increase. We furthermore assume that all metric values are positive. Due to these assumptions, we can interpret thresholds as upper bounds and the value 0 as universal lower bound. If the threshold of all metrics in a set needs to be fulfilled, this describes a rectangle, as visualized by Figure 1.

Based on this interpretation of metric sets with thresholds as a rectangle, we we defined an optimization technique for metric sets that relies on the axis-aligned rectangle learning algorithm outlined in Section 4.4. We assumed that a method for the assessment of quality attributes is available, which we can use to obtain a classification of the attributes. We interpret this method as a function $class^* : \mathfrak{S} \to \{0, 1\}$, where $\mathfrak{S}$ denotes the space of software entities and the classification 1 means that the quality attribute is fulfilled. Furthermore, we have a metric set $M = \{\mu_1, \mu_2 \ldots, \mu_n\}$ that is under consideration. The metric values of $\mathfrak{S}$ form a subset $\mathfrak{X}_n$ of $\mathbb{R}^n$, where each metric is represented in one dimension.

Let $\mathfrak{S}_{\text{sample}} = (s_1, s_2 \ldots, s_m)$ be a sequence of software entities for which we want to optimize our metric set, *e.g.*, the classes or methods of a software project. Through measurement of $\mathfrak{S}_{\text{sample}}$ we obtain metric data $X_i = M(s_i)$ and by applying $class^*$ to the sample we get classifications $Y_i = class^*(s_i)$. Thus, we obtain a learning sample $\mathbf{U}_m = (U_1, U_2 \ldots, U_m)$ with instances $U_i = (X_i, Y_i)$.

Let $M' \subseteq M$ be any subset of size $n' \leq n$ of the overall metric set M. Applying the canonical projection from $\mathbb{R}^n$ onto $\mathbb{R}^{n'}$ determined by the coordinates of these metric values, and renumbering the metrics of M' from 1 to n', we obtain a learning sample $\mathbf{U}'_m$ from the original sample $\mathbf{U}_m$. This learning sample is then used as input for the simulator of the rectangle learning algorithm in the noise model

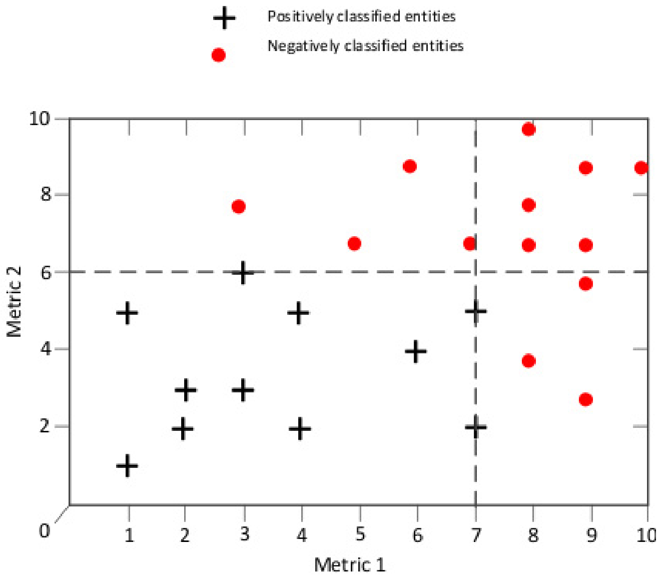

FIGURE 1. A rectangle described by the thresholds of two software metrics. The dashed lines mark the thresholds.

orthogonal to

$$\left\{\mathbb{1}_{\{x_j\in[a_j,b_j]\}} \wedge y \,\middle|\, a_j, b_j \in \mathbb{R},\, a_j < b_j,\, j = 1, 2, \ldots, n'\right\}.$$

The rectangle learned describes the area, where the quality attribute is fulfilled having eliminated the noise given the model assumptions are fulfilled. Therefore, the boundaries of the rectangle can be used as thresholds for the metric values. Through learning rectangles for all $M' \subseteq M$ we determine the smallest subset of the metrics M^* with thresholds T^* that is able to replicate the original classification $class^*$. The result $\left(M^*, T^*\right)$ is effective and efficient, if the classification can be replicated with a low risk, and it is the smallest to do so.

We evaluated this approach in several case studies, conducted based on data gathered from several large-scale open-source projects, *e.g.*, two Eclipse Projects [21, 22] and the Apache HTTP Server [3]. The experimental setup was based upon two different metric sets with four, respectively seven metrics. The metrics were determined to evaluate the *maintainability* of software. As classification function *class* we used a rectangle of the above described type, where we used thresholds we found in the literature. Our goal was the reduction of the size of these sets to improve their effiency. In a second setup, we relaxed the classification of the initial metric set and allowed infractions, *e.g.*, only six of the seven thresholds needed to be fulfilled. In all our experiments, the algorithm yielded good results, the risk of the optimal metric sets M^* was below 3% and therefore in the area of noise. Furthermore, we were able to reduce the size of existing classificators, *i.e.*, existing metric sets with thresholds used for quality classification. The size of the optimal metric set M^* was 42-83% smaller than the size of the initial set.

Above we have only outlined the approach for the metric set optimization based on rectangle learning, how it can be applied, and stated the quality of the overall results. A detailed presentation of the approach, including the description of three practical applications, four case studies with a total of nine experiments, as well as detailed statistical analysis of the capabilities of the approach based on the case study result is presented in [28].

11. Concluding remark

We have presented new noise models of PAC learning on the basis of Tsybakov's and Mammen's small margin conditions. However, from a practical point of view it is still rather unrealistic that the Bayes classifier is always a member of a known hypotheses class. That is why it would be useful to study noise rates beyond the limit of 1/2.

Acknowledgements. We acknowledge with thanks the valuable suggestions and comments of the unknown referees.

Appendix. Proof of equation (4.10)

Case $\kappa < 1$. We know that

$$\mathbb{E}\left(\nu(X)\right) = \int_0^{1/2} \mathbb{P}\left(\nu(X) > t\right) \mathrm{d}\, t.$$

Using equation (4.1) and the fact that $\mathbb{P}_U$ satisfies the small margin condition M_β $(0 < \beta < \infty)$ with small margin condition constant c, we get

$$\mathbb{P}\left(\nu(x) > t\right) = \mathbb{P}\left(1/2 - \nu(x) < 1/2 - t\right) \leq c(1/2 - t)^\beta.$$

Let $\epsilon_0 \in (0, 1/2)$ be fixed. Then

$$\int_0^{1/2} \mathbb{P}\left(\nu(X) > t\right) \mathrm{d}\, t = \underbrace{\int_0^{1/2-\epsilon_0} \mathbb{P}\left(\nu(X) > t\right) \mathrm{d}\, t}_{\leq 1/2-\epsilon_0} + \int_{1/2-\epsilon_0}^{1/2} \mathbb{P}\left(\nu(X) > t\right) \mathrm{d}\, t.$$

Since the anti-derivative of $c(1/2 - t)^\beta$ equals $-c/(\beta + 1)(1/2 - t)^{\beta+1}$, it follows

$$\int_{1/2-\epsilon_0}^{1/2} \mathbb{P}\left(\nu(X) > t\right) \mathrm{d}\, t \leq \frac{c}{\beta + 1}\epsilon_0^{\beta+1}.$$

Consequently,

$$\mathbb{E}(\nu(X)) \le 1/2 - \epsilon_0 + \frac{c}{\beta+1}\epsilon_0^{\beta+1}.$$

Having defined

$$\epsilon_0 := \left(\frac{\beta+1}{2c}\right)^{1/\beta},$$

we get

$$\mathbb{E}(\nu(X)) \le 1/2 - \epsilon_0/2.$$

Case $\kappa = 1$. From Lemma 2.11 we get that $\nu(X)$ is surely less than or equal to $1/2 - c$.

References

[1] D.W. Aha and D. Kibler, Instance-based learning algorithms. *Machine Learn.* (1991) 37–66.
[2] D. Angluin and P. Laird, Learning from noisy examples. *Machine Learn.* **2** (1988) 343–370.
[3] http://httpd.apache.org/ (2011).
[4] J.A. Aslam, *Noise Tolerant Algorithms for Learning and Searching*, Ph.D. thesis. MIT (1995).
[5] J.A. Aslam and S.E. Decatur, Specification and Simulation of Statistical Query Algorithms for Efficiency and Noise Tolerance. *J. Comput. Syst. Sci.* **56** (1998) 191–208.
[6] P.L. Bartlett, S. Boucheron and G. Lugosi, Model selection and error estimation. *Machine Learn.* **48** (2002) 85–113.
[7] P.L. Bartlett, M.I. Jordan and J.D. McAuliffe, Convexity, classification, and risk bounds. *J. Amer. Stat. Assoc.* **1001** (2006) 138–156.
[8] P.L. Bartlett and S. Mendelson, Rademacher and Gaussian complexities: Risk bounds and structural results, in *14th COLT and 5th EuroCOLT* (2001) 224–240.
[9] P.L. Bartlett and S. Mendelson, Rademacher and Gaussian complexities: Risk bounds and structural results. *J. Mach. Learn. Res.* (2002) 463–482.
[10] A. Blumer, A. Ehrenfeucht, D. Haussler and M.K. Warmuth, Learnabilty and the Vapnik−Chervonenkis dimension. *J. ACM* **36** (1989) 929–969.
[11] O. Bousquet, S. Boucheron and G. Lugosi, Introduction to statistical learning theory, in *Adv. Lect. Machine Learn.* (2003) 169–207.
[12] O. Bousquet, S. Boucheron and G. Lugosi, Introduction to statistical learning theory, in *Adv. Lect. Machine Learn.*, vol. 3176 of *Lect. Notes in Artificial Intelligence*. Springer, Heidelberg (2004) 169–207.
[13] Th. Brodag, *PAC-Lernen zur Insolvenzerkennung und Hotspot-Identifikation*, Ph.D. thesis, Ph.D. Programme in Computer Science of the Georg-August University School of Science GAUSS (2008).
[14] N. Cesa-Bianchi, S. Shalev-Shwartz and O. Shamir, Online learning of noisy data. *IEEE Trans. Inform. Theory* **57** (2011) 7907–7931.
[15] S.E. Decatur, Learning in hybrid noise environments using statistical queries, in *Fifth International Workshop on Artificial Intelligence and Statistics. Lect. Notes Statis.* Springer (1993).
[16] S.E. Decatur, Statistical Queries and Faulty PAC Oracles. *COLT* (1993) 262–268.

[17] S.E. Decatur, *Efficient Learning from Faulty Data*, Ph.D. thesis. Harvard University (1995).

[18] S.E. Decatur, PAC learning with constant-partition classification noise and applications to decision tree induction, in *ICML '97: Proc. of the Fourteenth Int. Conf. on Machine Learn.* Morgan Kaufmann Publishers Inc. San Francisco, CA, USA (1997) 83–91.

[19] S.E. Decatur and R. Gennaro, On learning from noisy and incomplete examples, in *COLT* (1995) 353–360.

[20] L. Devroye, L. Györfi and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*. Springer, New York (1997).

[21] http://www.eclipse.org/jdt/ (2011).

[22] http://www.eclipe.org/platform/ (2011).

[23] N. Fenton and S.L. Pfleeger, *Software metrics: a rigorous and practical approach*. PWS Publishing Co. Boston, MA, USA (1997).

[24] S. Goldman and R.H. Sloan, Can pac learning algorithms tolerate random attribute noise? *Algorithmica* **14** (1995) 70–84.

[25] I. Halperin, H. Wolfson and R. Nussinov, Protein-protein interactions coupling of structurally conserved residues and of hot spots across interfaces. implications for docking. *Structure* **12** (2004) 1027–1036.

[26] D. Haussler, Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence* **36** (1988) 177–221.

[27] D. Haussler, M.J. Kearns, N. Littlestone and M.K. Warmuth, Equivalence of models for polynomial learnability. *Inform. Comput.* **95** (1991) 129–161.

[28] S. Herbold, J. Grabowski and S. Waack, Calculation and optimization of thresholds for sets of software metrics. *Empirical Software Engrg.* (2011) 1–30. 10.1007/s10664-011-9162-z.

[29] International Organization of Standardization (ISO) and International Electro-technical Commission (ISEC), Geneva, Switzerland. *Software engineering – Product quality, Parts 1-4* (2001-2004).

[30] G. John and P. Langley, Estimating continuous distributions in bayesian classifiers, *In Proc. of the Eleventh Conf. on Uncertainty in Artificial Intelligence*. Morgan Kaufmann (1995) 338–345.

[31] M.J. Kearns, Efficient noise-tolerant learning from statistical queries. *J. ACM* **45** (1998) 983–1006.

[32] M.J. Kearns and M. Li, Learning in the presence of malicious errors. *SIAM J. Comput.* **22** (1993) 807–837.

[33] M.J. Kearns and R.E. Schapire, Efficient Distribution-Free Learning of Probabilistic Concepts. *J. Comput. Syst. Sci.* **48** (1994) 464–497.

[34] V. Koltchinskii, Rademacher penalties and structural risk minimization. *IEEE Trans. Inform. Theory* **47** (2001) 1902–1914.

[35] E. Mammen and A.B. Tsybakov, Smooth discrimination analysis. *Ann. Statis.* **27** (1999) 1808–1829.

[36] P. Massart, Some applications of concentration inequalities to statistics. *Annales de la Faculté des Sciences de Toulouse*, volume spécial dédié à Michel Talagrand (2000) 245–303.

[37] S. Mendelson, Rademacher averages and phase transitions in Glivenko-Cantelli classes. *IEEE Trans. Inform. Theory* **48** (2002) 1977–1991.

[38] I.S. Moreira, P.A. Fernandes and M.J. Ramos, Hot spots – A review of the protein-protein interface determinant amino-acid residues. *Proteins: Structure, Function, and Bioinformatics*, **68** (2007) 803–812.

[39] D.F. Nettleton, A. Orriols-Puig and A. Fornells, A study of the effect of different types of noise on the precision of supervised learning techniques. *Artif. Intell. Rev.* **33** (2010) 275–306.

[40] Y. Ofran and B. Rost, ISIS: interaction sites identified from sequence. *Bioinform.* **23** (2007) 13–16.

[41] Y. Ofran and B. Rost, Protein-protein interaction hotspots carved into sequences. *PLoS Comput. Biol.* **3** (2007).

[42] J.C. Platt, Fast training of support vector machines using sequential minimal optimization, in *Advances in kernel methods*. Edited by B. Schölkopf, Ch.J.C. Burges and A.J. Smola. MIT Press, Cambridge, MA, USA (1999) 185–208.

[43] J. Ross Quinlan, *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993).

[44] L. Ralaivola, F. Denis and Ch.N. Magnan, CN = CPCN, in *ICML '06: Proc. of the 23rd int. Conf. Machine learn.* ACM New York, NY, USA (2006) 721–728.

[45] B. Schölkopf and A.J. Smola, *Learning with Kernels*. MIT Press (2002).

[46] K.S. Thorn and A.A. Bogan, Asedb: a database of alanine mutations and their effects on the free energy of binding in protein interactions. *Bioinformatics* **17** (2001) 284–285.

[47] A.B. Tsybakov, Optimal aggregation of classifiers in statistical learning. *Ann. Statis.* **32** (2004) 135–166.

[48] L. Valiant, A theory of learnability. *Communic. ACM* **27** (1984) 1134–1142.

[49] L. Valiant, Learning disjunctions of conjunctions, in *Proc. of 9th Int. Joint Conf. Artificial Int.* (1985) 560–566.

Communicated by A. Bertoni.
Received Febraury 8, 2012. Accepted Febuary 5, 2014.