Machine Learning, 53, 5–21, 2003

A Microchoice Bound for Continuous-Space Classification Algorithms

YORAM GAT yoram.gat@intel.com
Intel Research, Microprocessor Research Lab, SC12-303, 2200 Mission College Blvd., Santa Clara, CA 95054-1537

Editor: Lisa Hellerstein

Abstract. Classifiers are often constructed iteratively by introducing changes sequentially to an initial classifier. Langford and Blum (*COLT'99: Proceedings of the 12th Annual Conference on Computational Learning Theory*, 1999, San Mateo, CA: Morgan Kaufmann, pp. 209–214) take advantage of this structure (the *microchoice* structure), to obtain bounds for the generalization ability of such algorithms. These bounds can be sharper than more general bounds. This paper extends the applicability of the microchoice approach to the more realistic case where the classifier space is continuous and the sequence of changes is not restricted to a pre-fixed finite set.

Proving the microchoice bound in the continuous case relies on a conditioning technique that is often used in proving VC results. It is shown how this technique can be used to convert any learning algorithm over a continuous space into a family of algorithms over discrete spaces.

The new continuous microchoice result is applied to obtain a bound for the generalization ability of the perceptron algorithm. The greedy nature of the perceptron algorithm, which generates new classifiers by introducing corrections based on misclassified points, is exploited to obtain a generalization bound that has an asymptotic form of $O(1/\sqrt{n})$, where n is the training set size.

Keywords: microchoice, perceptron

1. Introduction

Following Freund (1998), Langford and Blum (1999) construct generalization error bounds for a class of learning algorithms which they call microchoice algorithms. These algorithms select classifiers from a finite set by making a sequence of choices. Each sequence of choices corresponds to a classifier.

The procedure of constructing classifiers by a sequence of decisions is widely used in practice. For example, CART (Breiman et al., 1984) constructs decision trees in a sequential manner, by splitting the feature space into smaller and smaller cells. Another example is the perceptron algorithm (Minsky & Papert, 1969) that constructs a decision hyperplane by introducing changes sequentially.

However, most realistic learning algorithms—including CART and the perceptron algorithm—work with a continuous, rather than discrete, classifier spaces. Those algorithms therefore make their microchoices from an infinite set of alternatives. The analysis of Freund and of Langford and Blum cannot be applied directly to bound the generalization error of algorithms of this kind.

This paper presents a framework for handling continuous microchoice algorithms. The technique is based on a generic conditioning technique, used in VC theory (Vapnik, 1998), which handles continuous algorithms as a collection of discrete ones.

The paper is structured as follows: Section 2 defines the microchoice algorithm. Section 3 introduces some notation and terminology that will be used throughout the paper. Section 4 gives a generalization bound for the microchoice algorithm, assuming a finite classifier space, following the analysis of Langford and Blum. Section 5 presents the technique for handling continuous algorithms as a collection of discrete ones. The bound obtained by this technique is presented for a general learning algorithm and specialized for microchoice algorithms. Section 6 gives a detailed bound for the perceptron algorithm, by combining the results of Sections 4 and 5. Lastly, proofs for the correctness of the perceptron bound, and proof for its asymptotic form, as the number of samples increases, are presented in Section 7.

2. Definition of a microchoice algorithm

Informally, a microchoice algorithm can be described as follows: At each point throughout the execution the algorithm has some classifier serving as its working hypothesis. This working hypothesis is initialized with some fixed classifier. Given a certain working hypothesis, a set of alternatives is generated which depends only on this working hypothesis. That is, this set only depends on the training set via the working hypothesis. The algorithm then replaces its working hypothesis by a member of the alternatives set. The choice of the member depends on the training set, and possibly on additional randomization. The algorithm can then stop, giving the new working hypothesis as its output, or repeat the alternatives set generation and selection steps using the new working hypothesis.

The following formal definition of a microchoice algorithm is a variant of the one in Langford and Blum (1999). The definition given here makes the treatment in this paper easier, while not changing materially the set of algorithms covered by the definition.

Let $\mathcal{C}$ be a set of classifiers. That is, each $c \in \mathcal{C}$ is a function from the feature space $\mathcal{X}$ into label space $\mathcal{Y}$.

A microchoice learning algorithm (MCA), L, is a triple (c_o, s, h), where $c_o \in \mathcal{C}$, while s and h are functions

$$s : \mathcal{C} \times \mathcal{Z}^n \times \Omega' \to \mathcal{C}$$
$$h : \mathcal{N} \times \mathcal{C} \times \mathcal{Z}^n \to \{\text{CONT}, \text{STOP}\},$$

where $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, $\mathcal{N}$ are the natural numbers, and Ω' is a sample space.

The correspondence between the three elements of the L and the informal description of an MCA given above is as follows:

- c_o is the initial working hypothesis.
- The function s—which I call the selection function—is used to select the next working hypothesis, given the current working hypothesis. This involves both the generation of the alternatives set and the selection of the member of that set that will replace the working hypothesis.

For any selection function s it is possible to define the function $M^s : \mathcal{C} \to 2^{\mathcal{C}}$ as follows:

$$M^s(c) = \bigcup_{\mathbf{z} \in \mathcal{Z}^n} \bigcup_{\omega' \in \Omega'} s(c, \mathbf{z}, \omega'). \tag{1}$$

This function—the alternatives function—maps a classifier to the corresponding alternatives set. The map M^s depends solely on the classifier, and is not a function of the training set.

- The function h is used as a stopping rule, indicating the termination of the algorithm in a manner described below.

The inclusion of the additional sample space Ω' in the definition of s formalizes the possibility of using randomized choice among the alternatives. This is in addition to, and independently from, the randomization produced by the dependence on the training set. This randomization is assumed to select only among a finite number of alternatives, that is, for any c and $\mathbf{z}$,

$$\left| \bigcup_{\omega' \in \Omega'} s(c, \mathbf{z}, \omega') \right| < \infty.$$

Given a training set $\mathcal{S} = (Z_1 = (X_1, Y_1), \ldots, Z_n = (X_n, Y_n)) \in \mathcal{Z}^n$, and a point ω' in the sample space Ω', the MCA generates a sequence of classifiers, $c_0, \ldots, c_k$, where

$$\begin{aligned} c_0 &= c_o, \\ c_{j+1} &= s(c_j, \mathcal{S}, \omega'), \quad \text{for } j = 0, \ldots, k-1. \end{aligned}$$

As indicated above, the function h is the stopping criterion for the algorithm. The algorithm stops as soon as $h(j, c_j, \mathcal{S}) = \text{STOP}$. That is,

$$\begin{aligned} h(j, c_j, \mathcal{S}) &= \text{CONT}, \quad \text{for } j = 0, \ldots, k-1, \\ h(k, c_k, \mathcal{S}) &= \text{STOP}. \end{aligned}$$

The classifier c_k is the output of the algorithm L, denoted c_L.

3. Notation and terminology

I use the following notation and terminology:

- If $\mathbf{z} = (z_1, \ldots, z_n)$ denotes some n-tuple of points in $\mathcal{Z}$, then $\mathbf{z}$ is also used to denote the empirical distribution of that tuple. That is, for any $A \subset \mathcal{Z}$,

  $$\mathbf{z}(A) = |\{i : z_i \in A\}|/n.$$

  In particular,

  $$\mathcal{S} = (Z_1 = (X_1, Y_1), \ldots, Z_n = (X_n, Y_n))$$

will be used to denote both the training set and its empirical distribution, and

$$\mathcal{S}' = (Z'_1 = (X'_1, Y'_1), \ldots, Z'_{n'} = (X'_{n'}, Y_{n'}))$$

will be used to denote both a test set and its empirical distribution.

- It is often convenient to consider the test set $\mathcal{S}'$ as being the continuation of the training set sequence, that is

$$Z'_i = Z_{n+i}, \quad i = 1, \ldots, n'.$$

- In the discussion below the unknown joint distribution of the training and test sets will be assumed to be either an IID distribution which is denoted by $\mathbf{P}_{\mathsf{IID}}$ or a distribution generating sampling-without-replacement from a given set $\mathbf{z}$ which is denoted by $\mathbf{P}_{\mathbf{z}}$. When making statements that apply to both types of distributions the symbol $\mathbf{P}$ will be used. $\mathbf{P}$ will also be used for distributions over $\mathcal{Z}$ and as a generic notation for probability.
- Let $\mathbf{P}$ be a joint probability distribution over $\mathcal{S}$ and $\mathcal{S}'$. Then $[\mathbf{P} \mid \mathcal{S}]$ denotes the conditional probability distribution of Z'_1 given the training set $\mathcal{S}$.

  This distribution is the conditional distribution of a test point given the training set. Note that if $\mathbf{P} = \mathbf{P}_{\mathsf{IID}}$ then $[\mathbf{P} \mid \mathcal{S}] = \mathbf{P}_{\mathsf{IID}}$ as well. If, however, $\mathbf{P} = \mathbf{P}_{\mathbf{z}}$ then $[\mathbf{P} \mid \mathcal{S}] = \mathbf{z} \setminus \mathcal{S}$.[1]
- Let $\mathbf{P}$ and $\mathbf{P}'$ be two probability distributions over $\mathcal{Z}$, then $\mathbf{P} - \mathbf{P}'$ denotes the signed measure:

$$(\mathbf{P} - \mathbf{P}')(A) = \mathbf{P}(A) - \mathbf{P}'(A), \quad \text{for any event } A.$$

  I use this notation to refer to the difference measure between an underlying distribution $\mathbf{P}$ and the empirical distribution of the training set—$\mathbf{P} - \mathcal{S}$, to the difference measure between two empirical distributions—$\mathcal{S}' - \mathcal{S}$, and to the difference measure between the conditional distribution of a test point given the training points and the empirical distribution of the training points—$[\mathbf{P} \mid \mathcal{S}] - \mathcal{S}$.
- For any measure $\mathbf{T}$ over $\mathcal{Z}$, the operator $\mathbf{R}_{\mathbf{T}}$ maps classifiers to their risk (i.e., error rate) under the measure $\mathbf{T}$. That is,

$$\mathbf{R}_{\mathbf{T}}(c) = \mathbf{T}(\{(X, Y) : c(X) \neq Y\}).$$

  This notation is used with the underlying distribution $\mathbf{P}$, with the empirical distributions $\mathcal{S}$ and $\mathcal{S}'$, and with the difference measures described above: $\mathbf{R}_{\mathbf{P}}$, $\mathbf{R}_{\mathcal{S}}$, $\mathbf{R}_{\mathcal{S}'}$, $\mathbf{R}_{\mathbf{P}-\mathcal{S}}$, $\mathbf{R}_{\mathcal{S}'-\mathcal{S}}$, and $\mathbf{R}_{[\mathbf{P}|\mathcal{S}]-\mathcal{S}}$. This last operator, when applied to a classifier c gives the classifier's generalization error. That is,

$$\mathbf{R}_{[\mathbf{P}|\mathcal{S}]-\mathcal{S}}(c)$$

  is the random variable which is the difference between the expected error rate of the classifier on a test set and the observed error rate on the training set.
- Let U and V be two random variables. Then V is a level-δ probabilistic upper bound (PUB) of U—equivalently, U is a level-δ probabilistic lower bound (PLB) of V—under

a family of distributions $\mathcal{F}$ if

$$\mathbf{P}(V < U) \leq \delta, \quad \text{for all } \mathbf{P} \in \mathcal{F}.$$

4. The microchoice bound for finite classifier spaces

This section presents the microchoice bound for finite classifier spaces. Since the classifier space is assumed finite, the alternatives sets $M^s(c)$ are necessarily finite for any selection function s and any classifier c. This property is key in the construction of the bound, since the chance of selecting any member of $M^s(c)$ is non-zero.

The microchoice bound is a level-δ PUB, G_L, for the generalization error of c_L—the classifier selected by the microchoice algorithm. That is, G_L is a random variable with the property that

$$\mathbf{P}\big(G_L < \mathbf{R}_{[\mathbf{P}|\mathcal{S}]-\mathcal{S}}(c_L)\big) \leq \delta,$$

for all $\mathbf{P}$ in some family of probability distributions. At this point I leave the family under consideration unspecified. In the standard case, the family is assumed to be that of all IID distributions. However, it will become useful later to consider a different family, namely that of points sampled without replacement from some finite set.

The starting point for constructing G_L is a set of PUBs, bounding the error rates for fixed classifiers: For any c and any $\epsilon > 0$, assume that $G_c(\epsilon)$ is a level-ϵ PUB for $\mathbf{R}_{[\mathbf{P}|\mathcal{S}]-\mathcal{S}}(c)$.

Lemma 1. *Let μ be any fixed sub-probability measure over $\mathcal{C}$ (that is, a measure over $\mathcal{C}$ such that $\mu(\mathcal{C}) \leq 1$), and let L be any classification algorithm using a training set $\mathcal{S}$ to produce a random classifier $c_L = L(\mathcal{S})$.*

Then:

$$G_{L,\mu}(\delta) = G_{c_L}(\mu(c_L)\delta)$$

is a level-δ PUB for

$$\mathbf{R}_{[\mathbf{P}|\mathcal{S}]-\mathcal{S}}(c_L).$$

Proof: This follows directly from the union bound.

$$\begin{aligned}
\mathbf{P}\big(G_{L,\mu}(\delta) < \mathbf{R}_{[\mathbf{P}|\mathcal{S}]-\mathcal{S}}(c_L)\big) &= \sum_c \mathbf{P}\big(c_L = c \text{ and } G_c(\mu(c)\delta) < \mathbf{R}_{[\mathbf{P}|\mathcal{S}]-\mathcal{S}}(c)\big) \\
&\leq \sum_c \mathbf{P}\big(G_c(\mu(c)\delta) < \mathbf{R}_{[\mathbf{P}|\mathcal{S}]-\mathcal{S}}(c)\big) \\
&\leq \sum_c \mu(c)\delta \\
&\leq \delta. \qquad \square
\end{aligned}$$

In fact, quite obviously, a stronger result—which will be useful later—holds:

Lemma 2. *Let μ be any fixed sub-probability measure over $\mathcal{C}$. Then*:

$$\mathbf{P}\big(\textit{Exists some } c \in \mathcal{C} \textit{ such that } \mathbf{R}_{[\mathbf{P}|\mathcal{S}]-\mathcal{S}}(c) > G_c(\mu(c)\delta)\big) \leq \delta.$$

Proof: Again, this is a direct application of the union bound, as in Lemma 1. □

Given an MCA, a distribution $\mathbf{P}$ over the training sets induces a distribution over the sequences $c_j,\ j = 0, \ldots, k$, and therefore over the the selected classifier c_k.

It is desirable to select the measure μ so as to lower $G_{L,\mu}$. Since $G_c(\delta)$ is increasing as δ decreases, in general, μ should be chosen so as to have $\mu(c)$ large when $\mathbf{P}(c)$ is large.

The microchoice bound constructs μ in the following way. For any $c \in C$ let q_c be a probability distribution which is supported by $M^s(c)$. Set

$$\mu(c) = \sum_{k=0}^{\infty} t_k \sum \prod_{j=0}^{k-1} q_{c_j}(c_{j+1}), \tag{2}$$

where the second sum is over all sequences $c_0, \ldots, c_k = c$, that can be produced by the MCA, and $t_0, t_1, \ldots$ is a sequence of non negative real numbers summing to 1.

The measure μ is a probability measure, unless some of the classifiers have empty alternatives sets. In this case $\mu(\mathcal{C}) < 1$.

The factors t_i are used to account for the different possible numbers of working hypotheses used to reach the output classifier. While these factors affect the value of $\mu(c)$, and thus the value of the generalization bound, any reasonable choice—such as any slowly decreasing summable sequence—would produce essentially the same result. For this reason the choice of the factors t_i is largely ignored here.

If the distributions q_c are chosen so that the selected classifier, c_{j+1}, at each decision point, c_j, will likely have a high value $q_{c_j}(c_{j+1})$, then the expected value of $\mu(c_L)$ will be high and therefore, the expected PUB will be low.

Langford and Blum selected q_c as being zero for all low probability choices, and being positive and equal for all high probability choices. As Langford and Blum point out, other constructions of q_c are possible. To have μ approximate $\mathbf{P}$, q_c should be chosen to approximate the probability distribution over the alternative set. This is the approach taken here.

Note that whatever the definition of μ is, only $\mu(c_L)$ needs to be actually evaluated.

As for the fixed-classifier PUBs, G_c, here are examples of two: one for the IID case and one for the without-replacement sampling case.

- The IID case (the Hoeffding bound—e.g. in Devroye, Gyorfi, & Lugosi (1996)): Let $B, B_1, \ldots, B_n$ be IID Bernoulli variables with an unknown probability p of success (i.e. $\mathbf{P}(B = 1) = p$.)

  Then

$$G^n_{\mathsf{IID}}(\delta) = \sqrt{\frac{\log 1/\delta}{2n}}$$

is a level-δ PUB for the difference between $\mathbf{E}B = p$ and the mean of the sample $\bar{B} = \sum_{i=1}^{n} B_i/n$. That is

$$\mathbf{P}\left(G^n_{\mathsf{IID}}(\delta) < p - \bar{B}\right) \leq \delta.$$

- The without-replacement sampling case: Let $B_1, \ldots, B_{n+n'}$ be a random permutation of $b_1 = 1, \ldots, b_k = 1, b_{k+1} = 0, \ldots, b_{n+n'} = 0$, where k is unknown. Then

$$G^{n,n'}_{\mathsf{WOR}}(\delta) = \frac{1}{2n'}\left(c^2 + c\sqrt{c^2 + 4(n+n')\bar{B}}\right),$$

where $c = \frac{n+n'}{n}\sqrt{\frac{\log 1/\delta}{2}}$, is a level-$\delta$ PUB for the difference between the mean of $B_{n+1}, \ldots, B_{n+n'}$ and the mean of $B_1, \ldots, B_n$. That is

$$\mathbf{P}\left(G^{n,n'}_{\mathsf{WOR}}(\delta) < \frac{k - n\bar{B}}{n'} - \bar{B}\right) \leq \delta.$$

(This bound follows from Serfling, 1974.)

5. Continuous-space MCAs

As mentioned in the introduction, microchoice algorithms are common. However, since most of these create classifiers in a continuous space, defining the alternatives function M^s as was done in (1) would result in alternatives sets with infinite numbers of members, and with the chance of selecting any particular member of the set being zero. This would result in a trivial microchoice bound.

In order to apply the microchoice bound to continuous MCAs I employ a generic conditioning device, which is used in many VC-theory derivations (see, for example, Chapter 4 of Vapnik (1998) or Chapter 12 of Devroye, Gyorfi, & Lugosi (1996)). The technique here follows closely the one used in Gat (2001).

The device relies on the fact that if a test set $\mathcal{S}' = (Z'_1, \ldots, Z'_{n'}) \in \mathcal{Z}^{n'}$ is available, then the unknown true error rate of c_L can be reliably estimated as $\mathbf{R}_{\mathcal{S}'}(c_L)$. In fact it can be shown (Uhlmann (1963)) that

$$\mathbf{P}_{\mathsf{IID}}\left(\mathbf{R}_{\mathbf{P}_{\mathsf{IID}}-\mathcal{S}}(c_L) > \epsilon\right) \leq 2\mathbf{P}_{\mathsf{IID}}(\mathbf{R}_{\mathcal{S}'-\mathcal{S}}(c_L) > \epsilon - 1/n').$$

Therefore, any bound on the probability of a large deviation between $\mathbf{R}_{\mathcal{S}'}$ and $\mathbf{R}_{\mathcal{S}}$ implies a bound on the deviation between $\mathbf{R}_{\mathbf{P}_{\mathsf{IID}}}$ and $\mathbf{R}_{\mathcal{S}}$.

Obtaining a bound for $\mathbf{R}_{\mathcal{S}'-\mathcal{S}}$ is achieved by conditioning on the symmetric field Σ, defined as the minimal σ-field containing all the events of the form:

$$\bigcup_{\sigma}\left\{Z_1 \in B_{\sigma(1)}, \ldots, Z_{n+n'} \in B_{\sigma(n+n')}\right\}, \quad B_1, \ldots, B_{n+n'} \subset \mathcal{Z},$$

where the union is over all permutations of the set $\{1, \ldots, n+n'\}$.

Given Σ, the training set is still random, but is constrained to be a subset of $\mathbf{z} = (z_1, \ldots, z_{n+n'})$ which is some $(n + n')$-tuple of elements in $\mathcal{Z}$, called the train-test set. It is therefore possible, for any continuous-space MCA, given the symmetric field, to define a sample independent alternatives function $M^s_{\mathbf{z}}$, which maps classifiers into finite size alternatives sets:

$$M^s_{\mathbf{z}}(c) = \bigcup_{\omega'} \bigcup_{\sigma} s\big(c, \big(z_{\sigma(1)}, \ldots, z_{\sigma(n)}\big), \omega'\big).$$

That is, $M^s_{\mathbf{z}}$ is the union of all possible selections, given that the training set will be a subset of $\mathbf{z}$.

It follows that a continuous-space MCA can be represented as a collection of an infinite number of discrete-space MCAs. Each of those MCAs corresponds to a different train-test set, and therefore to a different alternatives function $M^s_{\mathbf{z}}$. The generalization bound for the continuous-space MCA would follow from bounds of discrete-space MCAs in the collection. This result is presented below as a corollary of the more general fact stated in Lemma 3, handling any learning algorithm—not necessarily an MCA:

Lemma 3. *Let L be a learning algorithm over the feature space $\mathcal{X}$ and label space $\mathcal{Y}$. Let $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$.*

Define a set of discrete-space learning algorithms:

$$\mathcal{L} = \{L_{\mathbf{z}} : \mathbf{z} = ((x_1, y_1), \ldots, (x_{n+n'}, y_{n+n'})) \in \mathcal{Z}^{n+n'}\},$$

where the feature space of $L_{\mathbf{z}}$ is $\{x_1, \ldots, x_{n+n'}\}$ and

$$L_{\mathbf{z}}(\mathcal{S}) = L(\mathcal{S})$$

for all $\mathcal{S} \subset \mathbf{z}$.

For each member of $\mathcal{L}$, $L_{\mathbf{z}}$, let $G_{\mathbf{z}}$ be a level-δ PUB for the generalization error of $c_{L_{\mathbf{z}}}$ under an equiprobable, without-replacement sampling from $\mathbf{z}$.

Then,

$$G = 1/n' + \sup_{\mathbf{z}:\mathcal{S}\subset\mathbf{z}} \{G_{\mathbf{z}}\}$$

is a level-2δ PUB of the generalization error of c_L under IID sampling.

Proof:

$$\begin{aligned}
\mathbf{P}_{\mathsf{IID}}\big(G < \mathbf{R}_{\mathbf{P}_{\mathsf{IID}}-\mathcal{S}}(c_L)\big) &\leq 2\mathbf{P}_{\mathsf{IID}}(G < \mathbf{R}_{\mathcal{S}'-\mathcal{S}}(c_L) + 1/n') \\
&= 2\mathbf{P}_{\mathsf{IID}}\Big(\sup_{\mathbf{z}:\mathcal{S}\subset\mathbf{z}} \{G_{\mathbf{z}}\} < \mathbf{R}_{\mathcal{S}'-\mathcal{S}}(c_L)\Big) \\
&\leq 2\mathbf{P}_{\mathsf{IID}}\big(G_{Z_1,\ldots,Z_{n+n'}} < \mathbf{R}_{\mathcal{S}'-\mathcal{S}}(c_L)\big) \\
&= 2\mathbf{E}_{\mathbf{P}_{\mathsf{IID}}}\big[\mathbf{P}_{\mathsf{IID}}(G_{Z_1,\ldots,Z_{n+n'}} < \mathbf{R}_{\mathcal{S}'-\mathcal{S}}(c_L)) \mid \Sigma\big] \\
&\leq 2\delta.
\end{aligned}$$

□

Corollary 1. *Let $L = (c_o, s, h)$ be a continuous-space MCA. Let the corresponding set of discrete-space MCAs be*

$$\mathcal{L} = \{L_{\mathbf{z}} = (c_o, s_{\mathbf{z}}, h) : \mathbf{z} \in \mathcal{Z}^{n+n'}\},$$

where the feature space of $L_{\mathbf{z}}$ is $\{x_1, \ldots, x_{n+n'}\}$ and

$$s_{\mathbf{z}}(c, \mathcal{S}, \omega') = s(c, \mathcal{S}, \omega')$$

for all $\mathcal{S} \subset \mathbf{z}$.

For each member of $\mathcal{L}$, $L_{\mathbf{z}}$, let $G_{\mathbf{z}}$ be a level-δ PUB for the generalization error of $c_{L_{\mathbf{z}}}$ under an equiprobable, without-replacement sampling from $\mathbf{z}$. Then,

$$G = 1/n' + \sup_{\mathbf{z}:\mathcal{S}\subset\mathbf{z}} \{G_{\mathbf{z}}\}$$

is a level-2δ PUB of the generalization error of c_L under IID sampling.

Proof: This is a special case of Lemma 3. □

6. Application: The perceptron algorithm

The results of the previous sections can be used to obtain microchoice bounds for well-known algorithms such as the perceptron algorithm and the CART algorithm. Below I give the details of the calculation of the microchoice bound for the perceptron algorithm.

6.1. Definition of the perceptron algorithm

The perceptron algorithm is defined as follows: The feature space $\mathcal{X}$ is the Euclidean space $\mathcal{R}^d$. The label space $\mathcal{Y}$ is $\{-1, 1\}$. The set of classifiers considered is the linear classifiers through the origin in $\mathcal{R}^d$,

$$\mathcal{C} = \{c_a : a \in \mathcal{R}^d\},$$

where $c_a(x) = \text{sign}(a \cdot x)$.

Starting with some initial point, say $a = 0$, repeat until no misclassified training set points remain, or until some other stopping criterion is met:

- Let x_e be a point misclassified by c_a, chosen uniformly at random among all the misclassified points in the training set, and let y_e be its label.
- Set $a \leftarrow a + y_e x_e$.

6.2. Formulation of the perceptron algorithm as an MCA

On a fixed train-test set, $\mathcal{S} \cup \mathcal{S}' = \mathbf{z}$, the perceptron algorithm can be formulated as an MCA in the following way: The feature space $\mathcal{X}_{\mathbf{z}}$ is $1, \ldots, n+n'$, where each point i is associated

with $z_i = (x_i, y_i)$. The label space $\mathcal{Y}_\mathbf{z}$ is $\{-1, 1\}$. The set of classifiers considered is

$$\mathcal{C}_\mathbf{z} = \{c_a : a \in \mathcal{R}^d\},$$

where $c_a(i) = \text{sign}(a \cdot x_i)$.

The alternatives function of the perceptron algorithm is:

$$M^s_\mathbf{z}(c_a) = \left\{c_{a+y_i x_i} : c_a(i) \neq y_i\right\}.$$

The selection function $s_\mathbf{z}$ is defined as randomly choosing among the members of the alternatives set which correspond to elements which are in the training set. This is done by defining a random order of precedence among the members of the alternatives set and selecting the member $c_{a+y_i x_i}$ with the highest precedence such that $i \in \mathcal{S}$. This is formalized as follows:

Given a training set $\mathcal{S}$, let $M^s_{\mathbf{z},\mathcal{S}}(c_a)$ denote the set:

$$M^s_{\mathbf{z},\mathcal{S}}(c_a) = \left\{c_{a+y_i x_i} : i \in \mathcal{S}, c_a(i) \neq y_i\right\}.$$

Define π as mapping each classifier $c \in \mathcal{C}_\mathbf{z}$ to a random one-to-one map from $M^s_\mathbf{z}(c)$ to the integers $\{1, \ldots, |M^s_\mathbf{z}(c)|\}$. That is, for any $c, c' \in \mathcal{C}_\mathbf{z}$ and any $\omega' \in \Omega'$,

$$\pi(c, \omega') : M^s_\mathbf{z}(c) \to \left\{1, \ldots, \left|M^s_\mathbf{z}(c)\right|\right\},$$

and

$$\pi(c, \omega')(c') \in \left\{1, \ldots, \left|M^s_\mathbf{z}(c)\right|\right\}.$$

For any fixed c, all $|M^s_\mathbf{z}(c)|!$ one-to-one maps are equally likely.

The value $\pi(c, \omega')(c')$ is interpreted as the order of precedence of the classifier c', with *lower* values indicating *higher* precedence.

Using the order of precedence π, the selection function $s_\mathbf{z}$ is defined as

$$s_\mathbf{z}(c_a, \mathcal{S}, \omega') = \arg \min_{c \in M^s_{\mathbf{z},\mathcal{S}}(c_a)} \pi(c_a, \omega')(c).$$

The stopping criterion is left only partly specified. It is required that the algorithm stop if no misclassified training points remain. However the bound would be valid for any criterion which allows earlier stopping. In fact, the bound would even be valid for a generalized algorithm that outputs, in a data determined way, any classifier from the sequence of working hypotheses, $c_0, \ldots, c_k$.

6.3. Calculation of the microchoice bound for the perceptron algorithm

Since the perceptron algorithm can be formulated as a continuous MCA (i.e., a collection of discrete MCAs), an appropriate choice of the distributions q_c will yield, through Lemma 3 and Corollary 1, a microchoice bound for the perceptron algorithm.

The distributions q_c may depend on the combined train-test set $\mathbf{z}$ and on the sample point ω', but they may not depend on the training set itself. Therefore, since $\mathbf{z}$ is partly unknown, the distributions q_c cannot be calculated exactly. They can however, be bounded, by inferring properties of the train-test set $\mathbf{z}$ from properties of the training set.

In particular, to determine a lower bound for the probability that a specific classifier from the alternatives set is selected, one needs to determine an upper bound for the size of that set.

Since the elements in the alternatives set are the misclassified elements of the train-test set, an upper bound for the number of elements in the alternatives set can be derived using a microchoice bound. That is, a microchoice bound for the generalization error of c_j enables bounding the distribution q_{c_j}, which in turn enables calculating a microchoice bound for the generalization error of c_{j+1}.

The chance that the highest precedence element in $M^s_{\mathbf{z},\mathcal{S}}(c)$ is of order more than r, is the chance that the r highest precedence elements are all in $M^s_{\mathbf{z}}(c) \backslash M^s_{\mathbf{z},\mathcal{S}}(c)$.

If $|M^s_{\mathbf{z}}(c)| = l$, and $|M^s_{\mathbf{z},\mathcal{S}}(c)| = m$, then this chance is

$$\frac{\binom{l-r}{m}}{\binom{l}{m}} = \frac{(l-m)(l-m-1)\ldots(l-m-r+1)}{l(l-1)\ldots(l-r+1)}.$$

If l is not known, this chance can be bounded from above by

$$\left(\frac{\bar{l}-m}{\bar{l}}\right)^r,$$

where $\bar{l}$ is an upper bound of l.

Thus, there exists a random variable $R(\bar{l})$, defined on the sample space $\mathcal{Z}^{n+n'} \times \Omega'$ with the distribution

$$\mathbf{P}_{\mathbf{z}}(R(\bar{l}) = r \mid \mathcal{S}) = \left(\frac{\bar{l}-m}{\bar{l}}\right)^{r-1} - \left(\frac{\bar{l}-m}{\bar{l}}\right)^r = \frac{m}{\bar{l}}\left(\frac{\bar{l}-m}{\bar{l}}\right)^{r-1},$$

such that on the event $\{\bar{l} \geq |M^s_{\mathbf{z}}(c_j)|\}$, $R(\bar{l})$ is an upper bound of $s_{\mathbf{z}}(c_j, \mathcal{S}, \omega')$.

Therefore, if $\bar{l}$ is a level-δ PUB of $|M^s_{\mathbf{z}}(c_j)|$, $R(\bar{l})$ is a level-δ PUB of the order of precedence of $s_{\mathbf{z}}(c_j, \mathcal{S}, \omega')$, i.e.,

$$\mathbf{P}_{\mathbf{z}}(R(\bar{l}) < \pi(c_j, \omega')(s_{\mathbf{z}}(c_j, \mathcal{S}, \omega'))) \leq \delta.$$

Roughly speaking, elements in the alternatives set are expected to be as common in the test set as in the training set, and so l is expected to be equal to about $m\frac{n+n'}{n}$. Therefore, the probability $\mathbf{P}_{\mathbf{z}}(R(\bar{l}) = r)$ is expected to be about

$$\frac{n(n')^{r-1}}{(n+n')^r}.$$

I set $q_c(c')$ to be equal to this value, where c' is the r-th highest precedence element of $M^s_{\mathbf{z}}(c)$, that is $\pi(c, \omega')(c') = r$. The random variable

$$\frac{n(n')^{R(\bar{l})-1}}{(n+n')^{R(\bar{l})}}$$

is therefore a level-δ PLB for $q_{c_j}(s_{\mathbf{z}}(c_j, \mathcal{S}, \omega'))$, assuming $\bar{l}$ is a level-δ PUB of $|M^s_{\mathbf{z}}(c_j)|$.

Assume that the PUB $\bar{l}$ is set so that for any fixed classifier c, as n grows large $\bar{l}/(n+n')$ approaches m/n (This is true for the bounds used in this paper). If n' is set to be $n' = \frac{1-\alpha}{\alpha}n$, the probability of $R(\bar{l}) > r$ approaches $(1-\alpha)^r$. Therefore, the probability that $q_{c_j}(c_{j+1})$ is much smaller than α/e is small, giving, with high probability, $\mu(c_k) \approx (\alpha/e)^k$. This is the reason that the microchoice bound's asymptotical behavior is roughly $\sqrt{1/n}$. This argument is formalized in Theorem 2 in Section 7.2.

The asymptotical behavior of the simpler bound of Gat (2001)—$\sqrt{\log n/n}$—is obtained by simply setting $q_c(c') = 1/|M^s_{\mathbf{z}}(c)|$ for all $c' \in M^s_{\mathbf{z}}(c)$. Other data-dependent hierarchy based methods of generating generalization bounds, such as the 'luckiness' framework (Shawe-Taylor & Bartlett, 1998) or the result of Warmuth and Floyd (1995) yield asymptotic behavior of $\sqrt{\log n/n}$ as well.

6.4. *Computational procedure*

I summarize the considerations given above in the following computational procedure, which calculates a level δ_o PUB for the generalization error of the classifier constructed by the perceptron algorithm.

An example of a factor sequence $t_0, t_1, \ldots$ that can be used with this procedure is $t_j = \frac{1}{(j+1)(j+2)}$.

- *Initialization*
  Set:

$$\begin{aligned} j &\leftarrow 0, \\ \delta_0 &\leftarrow t_0\delta_o/2, \\ a_0 &\leftarrow 0. \end{aligned}$$

- *Alternatives set size*
  Set:

$$\begin{aligned} c_j &\leftarrow c_{a_j} \\ m_j &\leftarrow |\{i : Y_i \neq c_j(X_i),\ i = 1, \ldots, n\}|, \\ \bar{l}_j &\leftarrow m_j + n'\left(\frac{m_j}{n} + G^{n,n'}_{\mathsf{WOR}}(\delta_j)\right). \end{aligned}$$

  Note that by using an updated confidence level δ_j at each iteration of the procedure (step *Precedence order* below), the level of the PUB $\bar{l}_j$ is set so that Lemma 1 applies.

- *Precedence order*
  Generate a random precedence order R_j according to the distribution

$$\mathbf{P}(R_j = r) = \frac{m_j}{\bar{l}_j}\left(\frac{\bar{l}_j - m_j}{\bar{l}_j}\right)^{r-1}, \quad r = 1, 2, \ldots$$

  Set

$$q_j \leftarrow \frac{n}{n+n'}\left(\frac{n'}{n+n'}\right)^{R_j - 1},$$
$$\delta_{j+1} \leftarrow \frac{t_{j+1}}{t_j} q_j \delta_j.$$

- *Update*
  Let $Z_{i_j} = (X_{i_j}, Y_{i_j})$ be some randomly chosen training point which is misclassified by c_j.
  Set:

$$a_{j+1} \leftarrow a_j + Y_{i_j} X_{i_j}.$$

- *Stop or iterate*
  If the stopping criterion has been met (in particular if no misclassified training points exist), go to the *Output* step. Otherwise, set

$$j \leftarrow j + 1,$$

  and go back to step *Alternatives set size*.
- *Output*
  Set:

$$\delta_{\text{out}} \leftarrow \delta_j.$$

  Output the bound

$$G^{n,n'}_{\mathsf{WOR}}(\delta_{\text{out}}) + 1/n'.$$

  This is a level δ_o PUB for the generalization error of the perceptron classifier $L(\mathcal{S}) = c_j$.

7. Proofs of correctness and asymptotic size of the generalization bound

7.1. Proof of correctness of the computational procedure

This section recapitulates some the discussion above in the form of a proof of correctness of the computational procedure in Section 6.4.

Theorem 1. *The output of the computational procedure*:

$$G^{n,n'}_{\mathsf{WOR}}(\delta_{\text{out}}) + 1/n',$$

is a level δ_o PUB for the generalization error of the classifier constructed by the perceptron algorithm.

Proof: Applying Corollary 1, it is enough to show that

$$G^{n,n'}_{\mathsf{WOR}}(\delta_{\text{out}})$$

is a level-$\delta_o/2$ PUB for the generalization error for the classifier constructed by the perceptron algorithm under without-replacement sampling from some arbitrary set of $n + n'$ points, $\mathbf{z}$, which contains the training set.

Define

$$\mathcal{C}_{\mathbf{z}} = \big\{L\big(\mathbf{z}_{i_1}, \ldots, \mathbf{z}_{i_n}\big)(\omega') : i_1, \ldots, i_n \text{ are distinct index values in the range } 1, \ldots, n + n', \omega' \in \Omega'\big\}.$$

Let μ be a distribution over $\mathcal{C}_{\mathbf{z}}$. By Lemma 2, $G^{n,n'}_{\mathsf{WOR}}(\mu(c)\delta_o/2)$ provide a simultaneous level-$\delta_o/2$ PUB for the generalization error for all classifiers c that may be constructed by the perceptron algorithm under without-replacement sampling from $\mathbf{z}$.

All that remains is to show that there exists a distribution μ such that if the generalization errors of all the classifiers in $\mathcal{C}_{\mathbf{z}}$ are less than or equal to $G^{n,n'}_{\mathsf{WOR}}(\mu(c)\delta_o/2)$, then δ_{out}, as calculated by the computational procedure, is less than or equal to $\mu(L(\mathcal{S}))\delta_o/2$. This would yield

$$\mathbf{R}_{\mathcal{S}'-\mathcal{S}}(L(\mathcal{S})) \le G^{n,n'}_{\mathsf{WOR}}(\mu(L(\mathcal{S}))\delta_o/2) \le G^{n,n'}_{\mathsf{WOR}}(\delta_{\text{out}}).$$

I claim that the distribution μ which fulfills this requirement is defined as in Eq. (2) with

$$q_c(c') = \frac{n}{n+n'}\left(\frac{n'}{n+n'}\right)^{\pi(c,\omega')(c')},$$

where $\pi(c, \omega')(c')$ is, for any fixed $c \in \mathcal{C}_{\mathbf{z}}$, a random one-to-one map from $M^s_{\mathbf{z}}(c)$ to $\{1, \ldots, l\}$, distributed uniformly over all maps, as defined in Section 6.2.

To prove the claim, I proceed, by induction, to show that if

$$\mathbf{R}_{\mathcal{S}'-\mathcal{S}}(c_j) \le G^{n,n'}_{\mathsf{WOR}}(\mu(c_j)\delta_o/2), \quad \text{for } j = 0, \ldots, i-1,$$

then it also holds that $\delta_i \le \mu'(c_i)\delta_o/2$, where

$$\mu'(c) = \max_k t_k \max \prod_{j=0}^{k-1} q_{c_j}(c_{j+1}).$$

with the second maximum being taken over all sequences $c_0 = c_o, c_1, \ldots, c_k = c$. Obviously, $\mu'(c_i) \leq \mu(c_i)$.

Induction base: $i = 0$. For this degenerate case $\delta_0 = t_0\delta_o/2 = \mu'(c_0)\delta_o/2$.

Induction step: Assume that the statement holds for $i = i'$.

Now, if

$$\mathbf{R}_{\mathcal{S}'-\mathcal{S}}(c_j) \leq G^{n,n'}_{\mathsf{WOR}}(\mu(c_j)\delta_o/2), \text{ for } j = 0, \ldots, i',$$

then, by the induction assumption, $\delta_{i'} \leq \mu'(c_{i'})\delta_o/2$. Therefore,

$$\begin{aligned}
\bar{l}_{i'} &= m_{i'} + n'\left(\frac{m_{i'}}{n} + G^{n,n'}_{\mathsf{WOR}}(\delta_{i'})\right) \\
&\geq m_{i'} + n'\left(\frac{m_{i'}}{n} + G^{n,n'}_{\mathsf{WOR}}(\mu(c_{i'})\delta_o/2)\right) \\
&\geq l_{i'}.
\end{aligned}$$

This implies that $R_{i'} \geq \pi(c_{i'}, \omega')(c_{i'+1})$ and $q_{i'} \leq q_{c_{i'}}(c_{i'+1})$, giving

$$\begin{aligned}
\delta_{i'+1} &= \frac{t_{i'+1}}{t_{i'}}\delta_{i'}q_{i'} \\
&\leq \frac{t_{i'+1}}{t_{i'}}\delta_{i'}q_{c_{i'}}(c_{i'+1}) \\
&\leq \frac{t_{i'+1}}{t_{i'}}\mu'(c_{i'})(\delta_o/2)q_{c_{i'}}(c_{i'+1}) \\
&\leq \mu'(c_{i'+1})\delta_o/2.
\end{aligned}$$

This completes the proof. □

7.2. The asymptotic behavior of the microchoice bound for the perceptron algorithm

Theorem 2. *Let L_n be the perceptron algorithm for a training set of size n. Assume that the stopping rule of L_n allows no more than $\tilde{k}$ microchoice steps. That is, the algorithm stops after considering no more than $\tilde{k}$ working hypotheses.*

Let G_{L_n} be the microchoice generalization bound as calculated in the computational procedure of Section 6.4, *with $n' = n$.*

Then,

$$\lim_{C\to\infty} \limsup_{n\to\infty} \mathbf{P}_{\mathsf{IID}}\left(\sqrt{n}\, G_{L_n} > C\right) = 0.$$

Proof: By definition of the microchoice bound value, given in step *Output* of the computational procedure, proving the theorem is equivalent to proving that

$$\lim_{\epsilon\to 0} \limsup_{n\to\infty} \mathbf{P}_{\mathbf{z}}(\delta_{\text{out}} < \epsilon) = 0, \tag{3}$$

That is, what needs to be shown is that δ_{out} is bounded away from 0, uniformly in n.

I proceed by induction on $\tilde{k}$.

Induction base: When $\tilde{k} = 0$, $\delta_{\text{out}} = \delta_0 > 0$, for all n.

Induction step: Assume that (3) holds for $\tilde{k} = \tilde{k}'$, and consider the case where $\tilde{k} = \tilde{k}' + 1$.

From the induction assumption it follows that (3) holds when the algorithm stops when $j \leq \tilde{k}'$.

On the other hand, if the algorithm stops only when $j = \tilde{k}' + 1$, then it follows from the induction assumption that $\delta_{\tilde{k}'}$ is bounded away from zero, uniformly in n. I now follow the computational procedure, going through the steps of its last iteration:

- Step *Alternatives set size*: Having $\delta_{\tilde{k}'}$ bounded away from zero uniformly in n implies that the ratio $m_{\tilde{k}'}/\bar{l}_{\tilde{k}'}$ is also bounded away from 0, uniformly in n, since

$$\bar{l}_{\tilde{k}'} = m_{\tilde{k}'} + \frac{c^2}{2} + \frac{c}{2}\sqrt{c^2 + 8m_{\tilde{k}'}},$$

  where $c = \sqrt{2 \log 1/\delta_{\tilde{k}'}}$.
- Step *Precedence order*: Therefore, the probability that $R_{\tilde{k}'}$ is large, is small uniformly in n, and thus $q_{\tilde{k}'}$ is bounded away from 0 uniformly in n. This implies that $\delta_{\tilde{k}'+1}$ is bounded away from 0 uniformly in n.

Since $\delta_{\text{out}} = \delta_{\tilde{k}'+1}$ this completes the proof. □

8. Conclusion

This paper presented a way to apply microchoice bounds to learning algorithms over continuous classifier spaces. By applying a conditioning argument, the problem was reformulated as a microchoice bound over a suitable discrete space.

The technique presented was applied to generate a bound for the perceptron algorithm, resulting in a bound that is tighter than those which can be obtained using other methods (Shawe-Taylor & Bartlett, 1998; Warmuth & Floyd, 1995; Gat, 2001).

Acknowledgments

I thank the reviewers and the editor of this paper for their many comments and suggestions which contributed considerably to improving the presentation in the final form of the paper.

Note

1. Strictly speaking, $\mathbf{z}$ and $\mathcal{S}$ are both vectors rather than sets and so the set subtraction notation is inappropriate. $\mathbf{z} \setminus \mathcal{S}$ should be interpreted to denote the vector $(z_{i_1}, \ldots, z_{i_k})$, containing all the elements of $\mathbf{z}$ that are not in the vector $\mathcal{S}$. The subset and union notation used later, e.g., $\mathcal{S} \subset \mathbf{z}$ and $\mathcal{S} \cup \mathcal{S}'$, should be interpreted in a similar manner.

References

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth, Belmont, CA.

Devroye, L., Gyorfi, L., & Lugosi, G. (1996). *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag.

Freund, Y. (1998). Self bounding classification algorithms. In *COLT '98: Proceedings of the Eleventh Annual Conference on Computational Learning Theory* (pp. 247–258). New York, NY: ACM Press.

Gat, Y. (2001). A learning generalization bound with an application to sparse-representation classifiers. *Machine Learning, 43*, 233–240.

Langford, J., & Blum, A. (1999). Microchoice bounds and self bounding learning algorithms. In *COLT '99: Proceedings of the Twelfth Annual Conference on Computational Learning Theory* (pp. 209–214). Mateo, CA: Morgan Kaufmann.

Minsky, M. L., & Papert, S. A. (1969). *Perceptrons*. Cambridge: The MIT Press.

Serfling, R. J. (1974). Probability inequalities for the sum in sampling without replacement. *Annals of Statistics, 2*, 39–48.

Shawe-Taylor, J., & Bartlett, P. L. (1998). Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory, 44*, 1926–1940.

Uhlmann, W. (1963). Ranggrößen als schäatzfunktionen. *Metrika, 7*, 23–40.

Vapnik, V. N. (1998). *Statistical Learning Theory*. New York: John Wiley & Sons, Inc.

Warmuth, M. K., & Floyd, S. (1995). Sample compression, learnability, and the Vapnik-Chervonenkis dimension. *Machine Learning, 21*, 269–304.

Received November 2, 2001
Revised May 12, 2003
Accepted May 12, 2003
Final manuscript May 12, 2003