From Noise-Free to Noise-Tolerant and from On-line to Batch Learning

Norbert Klasner and Hans Ulrich Simon*
Universität Dortmund
Fachbereich Informatik
D-44221 Dortmund
{klasner,simon}@ls2.informatik.uni-dortmund.de

Abstract

A simple method is presented which, loosely speaking, virtually removes noise or misfit from data, and thereby converts a "noise-free" algorithm A, which on-line learns linear functions from data without noise or misfit, into a "noise-tolerant" algorithm A^{nt} which learns linear functions from data containing noise or misfit. Given some technical conditions, this conversion preserves optimality. For instance, the optimal noise-free algorithm B of Bernstein from [3] is converted into an optimal noise-tolerant algorithm B^{nt}. The conversion also works properly for all function classes which are closed under addition and contain linear functions as a subclass.

In the second part of the paper, we show that Bernstein's on-line learning algorithm B can be converted into a batch learning algorithm $\hat{B}$ which consumes an (almost) minimal number of random training examples. This is true for a whole class of "pac-style" batch learning models (including learning with an (ϵ,γ)-good model and learning with an ϵ-bounded expected r-loss). The upper bound on the sample size is shown by applying a method of Littlestone from [9] which converts an on-line learning algorithm A into a batch learning algorithm $\hat{A}$ and relates the sample size of $\hat{A}$ to the total loss of A. For this purpose, Littlestone's method is extended from 0, 1-valued functions to real-valued functions. The lower bound on the sample size is shown by generalizing and applying a method of Simon from [11].

*The author gratefully acknowledges the support of Bundesministerium für Forschung und Technologie grant 01IN102C/2.

1 Introduction

Let F be a class of real-valued functions defined on domain X. On-line learning of functions from F with respect to loss-function $l : \Re \times \Re \rightarrow \Re$ proceeds in trials. During each trial the following happens:

1. The environment presents a new instance $x \in X$.
2. The learner predicts the outcome $\tilde{y} = h(x)$ according to its current hypothesis $h \in F$.
3. The environment reveals the true outcome y. The loss that the learner suffers within this trial is $l(y, \tilde{y})$.
4. The learner updates its current hypothesis $h \in F$.

If the learner is a deterministic algorithm A, then each prediction sequence $S = (x_t, y_t)_{1 \le t \le m}$ uniquely leads to the corresponding sequences of hypotheses $h_t \in F$, predictions $\tilde{y}_t = h(x_t)$ and loss values $l_t = l(y_t, \tilde{y}_t)$. The quantity $A_l(S) = \sum_{t=1}^{m} l_t$ is called the *total loss of A on S w.r.t. l*. The aim of A is to keep its total loss as small as possible even for "worstcase sequences" S. This aim is certainly not meaningful if we allow sequences whose building law is completely different from any function of F. We restrict therefore the set of allowed sequences S as follows:

Noise-Free Case A sequence $S = (x_t, y_t)_{1 \le t \le m}$ is only allowed if there exists a function $f \in F$ (unknown to A) which perfectly describes S, i.e., $y_t = f(x_t)$ for all $1 \le t \le m$. The *total loss of A w.r.t. l* is given by $A_l(F) = \sup_S A_l(S)$. $E_l(F)$ denotes the *minimum total loss*, i.e., the total loss of the best algorithm A for F.

Noise-Tolerant Case We denote the algorithm by A^{nt} which indicates that it is noise-tolerant. Given the *total amount N of noise or misfit*, a prediction sequence $S = (x_t, y_t)_{1 \le t \le m}$ is only allowed if it satisfies

$$\inf_{f \in F} \sum_{t=1}^{m} (y_t - f(x_t))^2 \le N^2.$$

Setting $\eta_t = y_t - f(x_t)$, it is equivalent to demand that the euclidean length of *misfit-vector* $\eta = (\eta_1, \ldots, \eta_m)$ is bounded by N. The *total loss of A^{nt} w.r.t. l* is given by $A_l^{nt}(F, N) = \sup_S A_l^{nt}(S)$. $E_l^{nt}(F, N)$ denotes the total loss of the best noise-tolerant algorithm A for F and N.

If also all sequences S of length greater than a given bound m are forbidden, the total loss of A (resp. of A^{nt}) is denoted by $A_l(F, m)$ (resp. by $A_l^{nt}(F, N, m)$).

We only discuss the noise-free case in the corresponding models of batch learning. If $\hat{A}$ is a batch learning algorithm for F, it gets a random sequence $S = (x_t, y_t)_{1 \le t \le m}$ in advance. Here the x_t are independently drawn according to a fixed but unknown probability distribution P, and $y_t = f(x_t)$ according to as fixed but unknown target function $f \in F$. As usual the sample size m of $\hat{A}$ is a function of some given accuracy and confidence parameters (and perhaps additional parameters if F is parameterized). We say that $\hat{A}$ batch-learns functions from F w.r.t. loss-function l, if, with probability at least $1-\delta$, it outputs the representation of a hypothesis $h \in F$ whose *expected loss* is smaller than ϵ, i.e.,

$$E_P[l(f(x), h(x))] < \epsilon.$$

The number of examples needed by A to achieve this is denoted by $\hat{A}_l(F, \epsilon, \delta)$.

In this paper, we consider a subclass of all linear functions from $\Re^n$ to $\Re$. The transpose of a matrix C is denoted by C'. Vectors w are usually regarded as column vectors. Thus w' is the corresponding row vector. The dot product of two vectors w, x is written as $w'x$. Each linear function can be represented by a vector w and yields outcome $w'x$ on $x \in \Re^n$. It is easy to see that the minimum total loss for learning linear functions is unbounded if we allow arbitrary instantiations of w, x. In this paper we consider the subclass where vectors w, x are restricted according to

$$\|w\|_2 \le W, \quad \|x\|_2 \le M,$$

where $\|\cdot\|_2$ denote the euclidean norm (the same class that was considered in the papers [3] and [5]). We denote this subclass by $LIN(M, W, n)$. We discuss the *r-loss* for $r > 0$ and the *discrete γ-loss* for $\gamma > 0$ given by

$$l_r(a, b) = |a - b|^r, \quad l_\gamma^D(a, b) = \begin{cases} 1, & \text{if } |a-b| \ge \gamma \\ 0, & \text{if } |a-b| < \gamma \end{cases}$$

respectively. The total loss of an on-line learning algorithm A for $LIN(M, W, n)$ w.r.t. r-loss (resp. discrete γ-loss) is denoted by $A_r(M, W, n)$ (resp. $A_\gamma^D(M, W, n)$). In the case of noise-tolerant on-line learning w.r.t. r-loss, we write $A_r^{nt}(M, W, N, n)$. Parameter n is omitted if there is no dependence on n. Parameter m is added if we study the dependence on the number m of trials. The corresponding notational conventions are used for the minimum total loss and for the sample size of batch learning algorithms.

An algorithm $\hat{A}$, which batch-learns target function $f \in F$ w.r.t. l_γ^D, outputs, with probability at least $1 - \delta$, a hypothesis h which is an *(ϵ, γ)-good model* for f, i.e., h satisfies

$$P[|h(x) - f(x)| \ge \gamma] < \epsilon.$$

We say that A batch-learns F with an (ϵ, γ)-good model. This learning model is not only meaningful by itself, but also technically important for derivations of lower bounds on the sample size. This is because the loss-function l_γ^D is a $0,1$-valued function. The derivation of the lower bound proceeds by showing that implicitly pac-learning of this loss-function takes place and applying the known lower bounds from [4, 6] on the sample size for pac-learning. The lower bounds for the other batch learning models can then be obtained by using known relations from [8] between the different models. See [11] for more details concerning the derivation of lower bounds.

All results in this paper can be extended to more general inner products than just the dot product (at the same level of generality as in [5]). Therefore they hold in particular for the class of smooth functions which was considered in [5].

This paper is structured as follows. In Section 2, we present a surprisingly simple method which converts a noise-free on-line learning algorithm A for $LIN(M, W, n)$ into a noise-tolerant algorithm A^{nt} such that (at least w.r.t. r-loss for $r \ge 2$) optimality is preserved. The method also works properly for function classes which contain $LIN(M, W, n)$ and are closed under addition. This method is applied to Bernstein's noise-free on-line algorithm B from [3] which is optimal w.r.t. r-loss for all $r > 0$ and w.r.t. discrete γ-loss for all $\gamma > 0$. It produces the noise-tolerant algorithm B^{nt} which is still optimal w.r.t. r-loss for all $r \ge 2$. Another optimal noise-tolerant algorithm with the same total r-loss as B^{nt} for $r \ge 2$ is already known in the literature: the gradient descent algorithm from [5]. For $0 < r < 2$, the total r-losses of these two algorithms are different (see the corresponding discussion in Section 2).

In Section 3, we convert B into a batch algorithm and show that it consumes an (almost) minimal number of random training examples w.r.t. r-loss for all $r > 0$, and w.r.t. discrete γ-loss for all $0 < \gamma \leq WM$. The upper bound on the sample size is shown by applying method of Littlestone from [9] which converts an on-line learning algorithm A into a batch learning algorithm $\hat{A}$ and relates the sample size of $\hat{A}$ to the total loss of A. For this purpose, Littlestone's method is extended from $0,1$-valued functions to real-valued functions. The lower bound on the sample size is shown by generalizing and applying a method of Simon from [11].

Two aims are pursued in this paper. First, we want to present learning algorithms for $LIN(M, W, n)$ with small total loss, resp. with an almost optimal sample size. Second, we want to demonstrate that the conversion from noise-free to noise-tolerant algorithms presented in Section 2 and the general methods used in Section 3 are elaborate tools which can probably not be significantly improved in general.

2 Virtual Removal of Misfit in the Data

In this section, we describe Bernstein's algorithm B from [3], compute its total r-loss and its discrete γ-loss, present the general method of converting an algorithm A into A^{nt}, apply this method to Bernstein's algorithm B, and analyze B^{nt}. The significance of the obtained results can be described as follows:

- Our conversion method is easy to analyze and in principle applicable to all function classes that contain $LIN(M, W, n)$ and are closed under addition. The fact that it preserves the optimality of B (assuming $r \geq 2$) indicates that it does not need to pay the virtual removal of misfit in the data by getting unnecessary extra-loss (at least w.r.t. r-loss for $r \geq 2$).
- Although B^{nt} has the same total r-loss as the gradient descent method from [5], these two algorithms become quite different for $0 < r < 2$. It was already shown by Bernstein in [3] that, without noise or misfit, B has optimal total 1-loss $WM\sqrt{n}$, whereas Mycielsky's gradient descent method from [10] has total 1-loss at least WMn. A slight generalization of Bernstein's argument shows that each gradient descent method (in particular the one from [5]) has total 1-loss at least WMn. In the presence of noise or misfit in the data, the analysis of 1-loss is more complicated. We can show that B^{nt} is optimal modulo factor 2 if we (artificially) assume that all instances of the prediction sequence are members of an orthormal base possibly scaled by a constant.[1] We omit the proof in this abstract, because we conjecture that optimality modulo a constant factor holds for all allowed prediction sequences.
- In Section 3, we use Theorem 2.1 below for proving that "batch-versions" $\hat{B}$ of B consume an (almost) minimal number of random training examples.

A high level description of Bernstein's algorithm which ignores implementational details reads as follows:

Bernstein's Algorithm B for learning an unknown linear function $w'x$.

data structures "known subspace" $\mathcal{U}$: the target values $w'u$ of all $u \in \mathcal{U}$ are known.

Initialization $\mathcal{U} = \{\bar{0}\}$, where $\bar{0}$ denotes the zero vector.

Main Loop The following happens in each trial:

1. The environment presents a new instance x.
2. B computes the decomposition $x = u + v$, where $u \in \mathcal{U}, v \in \mathcal{U}^{\perp}$, and returns the value $w'u$. (In other words, B predicts value 0 for the unknown direction v.)
3. The environment presents the true target value y for x. (Thus $w'v = y - w'u$. Now B knows the correct value for v.)
4. B extends the known subspace $\mathcal{U}$ in direction v.

The algorithm is easy to implement. $O(n^2)$ real numbers have to be stored. During each trial $O(n)$ dot products must be computed (with some small additional overhead). Thus $O(n^2)$ arithmetic operations are needed per trial. A detailed description of B and B^{nt} is given in the appendix.

The following result is related to Theorem 5 in [3] (where a more general function class is considered, but n is restricted to be a power of 2, and the discrete γ-loss is not considered). Although the proofs are also related, we present the proof in full length, because it uses an orthonormalization argument which is important for understanding the optimality of B (and later the optimality of B^{nt}, or the optimal sample size of the "batch-versions" of B in Section 3).

[1] The basic idea is to show that each trial t creates a new "unavoidable" amount λ_t of loss. Since the weights are not updated as optimal as in the noise-free setting, λ_t may be responsible for inadequate decisions in subsequent trials. However, the effect of this can be bounded by a geometric-row argument, if we set K (see Theorem 2.2 below) to M.

Theorem 2.1 *For all M, W, n, r, γ, Bernstein's algorithm is an optimal on-line learning algorithm for $LIN(M,W,n)$ w.r.t. r-loss and w.r.t. discrete γ-loss. For $r \geq 2$, the minimum total r-loss does not depend on n. The following equalities are valid:*

$$\forall 0 < r < 2: \quad E_r(M,W,n) = B_r(M,W,n) = (WM)^r n^{1-r/2} \quad (1)$$

$$\forall r \geq 2: \quad E_r(M,W) = B_r(M,W) = (WM)^r \quad (2)$$

$$\forall 0 < \gamma < WM: \quad E_\gamma^D(M,W,n) = B_\gamma^D(M,W,n) = \min\{(WM/\gamma)^2, n\} \quad (3)$$

Proof Let e^i denote the canonical base vector with 1 in position i and 0 in all other positions.
The lower bound $E_r(M,W,n) \geq (WM)^r n^{1-r/2}$ for all $0 < r < 2$ is obtained for one of the target weight vectors $(\pm W/\sqrt{n}, \ldots, \pm W/\sqrt{n})$ by presenting the instances $Me^1, \ldots, Me^n$. The best guess for the learning algorithm is 0 in each trial. The total r-loss after n trials is $n(WM/\sqrt{n})^r = (WM)^r n^{1-r/2}$.
The lower bound $E_r(M,W) \geq (WM)^r$ for all $r \geq 2$ is obtained in the first trial for one of the target weight vectors $(\pm W, 0, \ldots, 0)$ by presenting the instance $(M, 0, \ldots, 0)$. The best guess is 0 and the resulting r-loss is $(WM)^r$.
The lower bound $E_\gamma^D(M,W,n) \geq \min\{(WM/\gamma)^2, n\}$ is obtained within $p = \min\{(WM/\gamma)^2, n\}$ trials for target weights of the form $w_1 = \cdots = w_p = \pm(\gamma/M)$ and $w_q = 0$ for $q > p$ by presenting the instances $Me^1, \ldots, Me^p$. No algorithm can avoid loss 1 per trial during the first p trials in the worstcase.

The derivation of the corresponding upper bounds $B_r(M,W,n) \leq (WM)^r n^{1-r/2}$ for all $0 < r < 2$, $B_r(M,W) \leq (WM)^r$ for all $r \geq 2$, and $B_\gamma^D(M,W,n) \leq \min\{(WM/\gamma)^2, n\}$ is based on the following reasoning. B predicts value $w'u$ for $x = u + v$. Thus B implicitly predicts 0 for $v \in U^\perp$. Since v is part of an orthogonal decomposition of x, the inequality $\|v\|_2 \leq \|x\|_2 \leq M$ is valid (with equality if and only if $u = \bar{0}$ and $v = x$). The loss that B suffers is $|w'v|^r$. W.l.o.g. $v \neq \bar{0}$. If $u \neq \bar{0}$, then presenting instance $(M/\|v\|_2)v$ instead of x leads to a greater loss without giving more information to B. Therefore a worstcase sequence of instances has the form $Mu^1, \ldots, Mu^n$, where the column vectors u^i form an orthogonal matrix $U = [u^1, \ldots, u^n]$ (that is, U satisfies $U'U = UU' = I_n$, where I_n denotes the identity matrix).
Consider for instance the specific case, where $u^i = e^i$, i.e, $U = I_n$. During trial t, B will learn the true weight w_t and suffer r-loss $|Mw_t|^r$. Using the constraint $\|w\|_2 \leq W$, it is easy to see that the total r-loss (resp. discrete γ-loss) is maximized if $w = (\pm W/\sqrt{n}, \ldots, \pm W/\sqrt{n})$ for all $0 < r < 2$, $w = (\pm W, 0, \ldots, 0)$ for all $r \geq 2$ (resp. $w_1 = \cdots = w_p = \pm\gamma/M$ where $p = \min\{(WM/\gamma)^2, n\}$). This leads to a total r-loss (resp. discrete γ-loss) of $(WM)^r n^{1-r/2}$ for $0 < r < 2$, $(WM)^r$ for $r \geq 2$ (resp. $\min\{(WM/\gamma)^2, n\}$).
Now consider an arbitrary orthogonal matrix $U = [u^1, \ldots, u^n]$. Using the base transformation represented by U', U is transformed into $U'U = I_n$, and weight vector w is transformed into $U'w$ according to

$$w'x = w'I_n x = w'UU'x = (U'w)'(U'x).$$

Thus we arrive at the specific case again and may apply the above reasoning for the virtual weight vector $U'w$. (Note that $\|U'w\|_2 = \|w\|_2 \leq W$ because U' is an orthogonal matrix.) This completes the proof of the theorem. •

Now we present the general conversion method which virtually removes misfit in the data.

Theorem 2.2 *Let $r \geq 2$ and M, W, N, n arbitrary. Each learning algorithm A for linear function class $LIN(M,W,n)$ can be transformed into a noise-tolerant learning algorithm A^{nt} such that the following holds:*

$$A_r^{nt}(M,W,N) = \min_{K>0} A_r\left(\sqrt{M^2+K^2}, \sqrt{W^2+(N^2/K^2)}\right) \quad (4)$$

Proof Let K be chosen such that the minimum in the above expression is obtained. Let w denote the unknown weight vector. The transformation is quite simple. Remember that e^t denotes the t'th canonical base vector. A trial t with instance x_t, target value y_t and misfit η_t is modelled by a virtual trial with instance (x_t, Ke^t) and target value y_t. Noise η_t is now represented as another unknown weight component. Now the unknown weight vector has the form $(w, \frac{1}{K}\cdot\eta)$, where $\eta = (\eta_1, \ldots, \eta_m)$ are the amounts of misfit within trials $1, \ldots, m$, respectively. This new sequence of trials virtually contains no misfit within the data. Therefore we may apply algorithm A. Since the euclidean length of x_t and Ke^t is bounded by M and K, respectively, the euclidean length of (x_t, Ke^t) is bounded by $\sqrt{M^2+K^2}$. Since the euclidean length of w and η is bounded by W and N, respectively, the euclidean length of $(w, \frac{1}{K}\cdot\eta)$ is bounded by $\sqrt{W^2+(N/K)^2}$. The assertion of the theorem now follows from Theorem 2.1. •

Corollary 2.3 *Let $r \geq 2$ and M, W, N, n arbitrary. The algorithm B^{nt} associated with Bernstein's algorithm is an optimal noise-tolerant on-line algorithm for $LIN(M, W, n)$ w.r.t. r-loss. The following equality is valid:*

$$E_r^{nt}(M, W, N) = B_r^{nt}(M, W, N) = (WM + N)^r. \tag{5}$$

Proof The lower bound $E_r^{nt}(M, W, N) \geq (WM + N)^r$ is trivially to obtain in the first trial (the case $r = 2$ is already mentioned in [5], Theorem 10). Use instance $(M, 0, \ldots, 0)$, one of the weight vectors $(\pm W, 0, \ldots, 0)$, and one of the misfit vectors $(\pm N, 0, \ldots, 0)$. The best guess of a learning algorithm is 0. The r-loss obtained is $(WM + N)^r$.
The corresponding upper bound $B_r^{nt}(M, W, N) \leq (WM + N)^r$ is obtained from Theorems 2.1 and 2.2 by setting $K = \sqrt{MN/W}$. Thus:

$$\begin{aligned}
& B_r^{nt}(M, W, N) \\
\leq & B_r\left(\sqrt{M^2 + K^2}, \sqrt{W^2 + (N^2/K^2)}\right) \\
= & \left((M^2 + K^2)(W^2 + (N^2/K^2))\right)^{r/2} \\
= & \left(M^2W^2 + N^2 + \frac{M^2N^2}{K^2} + W^2K^2\right)^{r/2} \\
= & \left(M^2W^2 + N^2 + 2MNW\right)^{r/2} \\
= & (WM + N)^r
\end{aligned}$$

The second last equality made use of $M^2N^2/K^2 = W^2K^2 = MNW$. (Parameter K was exactly chosen such that these two terms are in balance.) This completes the proof of the corollary. •

For $0 < r < 2$, one can similarly show the following result.

Corollary 2.4 *Let $0 < r < 2$ and M, W, N, n arbitrary. The following bounds on the total r-loss for noise-tolerant on-line learning of $LIN(M, W, n)$ within $m \geq n$ trials are valid.*

$$E_r^{nt}(M, W, N, n, m) \geq (WM)^r n^{1-r/2} + N^r m^{1-r/2} \tag{6}$$

$$B_r^{nt}(M, W, N, n, m) \leq (WM + N)^r m^{1-r/2} \tag{7}$$

Although there is a gap between these bounds, they have the same order of magnitude if $WM = O(N)$, or if $m = O(n)$. At least for these special cases, the total r-loss of B^{nt} is optimal modulo a constant. We conjecture that B^{nt} is optimal modulo a constant.

B^{nt} can be implemented without adding the m artificial components to the instances. These components have an effect on the "learning rate" of B^{nt} which can be controlled without explicitly expanding the instances. A detailed implementation of B^{nt} is given in the appendix.

3 Conversion to Batch Learning Algorithms

In this section we want to determine the number of examples which are needed for learning functions $LIN(M, W, n)$ with an (ϵ, γ)-good model or with ϵ-bounded expected r-loss. Two general tools are used for this purpose:

- A general conversion theorem (Theorem 3.1 below), which is a straightforward generalization of the corresponding result of Littlestone for 0, 1-valued functions (see [9]). This theorem allows to convert an on-line learning algorithm A into a batch learning algorithm $\hat{A}$ and to relate the sample size of $\hat{A}$ to the total loss of A.

- A general lower bound on the sample size (Theorem 3.3 below) which generalizes a corresponding result of Simon in [11] (where r-loss was only discussed for $r = 1, 2$, the lower bounds w.r.t. r-loss were a bit weaker, and functions with range $[0, 1]$ were considered). This lower bound is stated in terms ϵ, δ, γ and two combinatorical parameters $\Delta(F), d_\gamma(F)$ associated with function class F. For $F = LIN(M, W, n)$, the parameters $\Delta(F), d_\gamma(F)$ are easy to determine (see Theorem 3.4 below). The computation of $d_\gamma(LIN(M, W, n))$ implicitly uses the orthonormalization argument that was already used in the analysis of Bernstein's algorithm.

The resulting upper and lower bounds on the sample size for batch-learning $LIN(M, W, n)$ (see Corollaries 3.2 and 3.5) are tight modulo a constant or logarithmic term. This demonstrates that the conversion theorem and the general lower bounds are elaborate tools. We made no serious attempt to test the alternative tools available in the literature (such as the general lower and upper bounds in [7, 1, 2] for instance). However it might be difficult to get tight upper bounds by using the pseudo-dimension of $LIN(M, W, n)$ which is easily shown to be n (and leads to an inferior upper bound). Similarly, the general lower bound in [2] in terms of the fat-shattering dimension is not valid in the noise-free case (in contrast to the general bounds that we are using). Therefore our selection of tools seems to be the most straightforward-one. We now present the conversion theorem.

Theorem 3.1 *Let $l : \Re^2 \rightarrow [0, D]$ be a loss-function and A an on-line learning algorithm with maximum total* *loss L and an upper bound H on the number of the different hypotheses that can be produced during the run over a prediction sequence.*[2] *Then A can be converted into a batch algorithm $\hat{A}$ which learns with an ϵ-bounded expected loss such that the following bound on the number of examples needed by $\hat{A}$ is valid:*

$$\hat{A}(\epsilon,\delta) = O\left(\frac{1}{\epsilon}\left(L + D\ln\left(\min\left\{\frac{D}{\epsilon\delta},\frac{H}{\delta}\right\}\right)\right)\right)$$

Littlestone proved this theorem for the special case of 0,1-valued functions and the zero-one loss-function (see [9]). Our proof for Theorem 3.1 parallels the proof of Littlestone and is therefore omitted in this abstract.

From this conversion-theorem and the results of the preceding section, we obtain immediately the following result.

Corollary 3.2 *The linear function class $LIN(M,W,n)$ is learnable*

- *with an (ϵ,γ)-good model using*

  $$O\left(\frac{1}{\epsilon}(\min\{(WM/\gamma)^2, n\} + \ln(1/\delta))\right)$$

  examples.

- *with an ϵ-bounded expected r-loss using*

  $$O\left(\frac{(WM)^r}{\epsilon}(n^{1-r/2} + \ln(1/\delta))\right)$$

  examples if $0 < r < 2$, and with

  $$O\left(\frac{(WM)^r}{\epsilon}\ln\left(\min\left\{\frac{n}{\delta},\frac{(WM)^r}{\epsilon\delta}\right\}\right)\right)$$

  examples if $r \geq 2$.

The derivation of the corresponding lower bounds on the sample size requires some preliminary definitions from [11]. We say that two functions f,g defined on domain X are *disjoint* if $f(x) \neq g(x)$ for all $x \in X$. A function class F is called *trivial* if its functions are pairwise disjoint. We define the quantity $\Delta(F)$ for nontrivial classes as follows:

$$\Delta(F) = \sup\left\{\begin{array}{l}|f(x)-g(x)|, \text{ where } x \in X \text{ and } f,g \\ \text{are nondisjoint functions in } F\end{array}\right\}.$$

Let $\gamma > 0$. We say that sequence

$$[(x_1, r_1 \pm \gamma_1), \ldots, (x_d, r_d \pm \gamma_d)] \in (X \times \Re^2)^d$$

is *γ-shattered* by F if $\gamma_1,\ldots,\gamma_d \geq \gamma$ and

$$\forall b \in \{0,1\}^d, \exists f = f_b \in F, \forall i = 1,\ldots,d:$$
$$f(x_i) = \begin{cases} r_i - \gamma_i, & \text{if } b_i = 0 \\ r_i + \gamma_i, & \text{if } b_i = 1\end{cases}$$

The quantity $d_\gamma(F)$ is defined as the maximal d (possibly ∞) such that a sequence S of length d which is γ-shattered by F exists. The following theorem presents general lower bounds on the sample size required for batch-learning F in terms of $\Delta(F), d_\gamma(F)$.

Theorem 3.3
1. *If F is nontrivial, $\epsilon \leq 1/2$ and $\gamma \leq \Delta(F)/2$, then each algorithm which learns F with an (ϵ,γ)-good model needs at least $\Omega(\ln(1/\delta)/\epsilon)$ examples.*
2. *If $\epsilon \leq 1/8$ and $\delta \leq 1/100$, then each algorithm which learns F with an (ϵ,γ)-good model needs at least $\Omega\left((d_\gamma(F)-1)/\epsilon\right)$ examples.*
3. *If F is nontrivial, $r > 0$, and $\epsilon \leq \Delta^r(F)/2^{r+1}$, then each algorithm which learns F with an ϵ-bounded expected r-loss needs at least $\Omega\left((\Delta(F)/2)^r\ln(1/\delta)/\epsilon\right)$ examples.*
4. *If $\epsilon \leq \gamma^r/8$ and $\delta \leq 1/100$, then each algorithm which learns F with an ϵ-bounded expected r-loss needs at least $\Omega\left((d_\gamma(F)-1)\gamma^r/\epsilon\right)$ examples.*

The proof is related to the proof of the corresponding (less general) result in [11]. It is omitted in this abstract.

Theorem 3.4 *$LIN(M,W,n)$ is nontrivial and the following equalities are valid for all choices of γ, M, W, n:*

$$\Delta\left(LIN(M,W,n)\right) = 2WM \quad (8)$$
$$d_\gamma\left(LIN(M,W,n)\right) = \min\left\{(WM/\gamma)^2, n\right\} \quad (9)$$

Proof Let $F = LIN(M,W,n)$. F is certainly nontrivial. According to the inequality of Schwartz:

$$|w'x| \leq \|w\|_2\|x\|_2 \leq WM.$$

Thus $\Delta(F) \leq 2WM$. On the other hand $\Delta(F) \geq 2WM$, since

$$(\pm W, 0, \ldots, 0)(M, 0, \ldots, 0)' = \pm WM.$$

Thus $\Delta(F) = 2WM$.

Let $d = d_\gamma\left(LIN(M,W,n)\right)$, $p = \min\left\{(WM/\gamma)^2, n\right\}$, and e^i the i-th canonical base vector. Sequence

$$(Me^1, \pm\gamma), \ldots, (Me^p, \pm\gamma)$$

is γ-shattered which is witnessed by the weights

$$w_1 = \cdots = w_p = \pm\frac{\gamma}{M}, \quad w_q = 0 \text{ for } q > p.$$

[2] $H \leq n+1$ in our application, since Bernstein's algorithm B achieves exact identification after at most n faulty predictions. For discrete y-loss, we may assume that H is at most 1+(total discrete γ-loss of B), if we use the conservative variant of B that ignores trials with loss 0.

This shows that $d \geq p$. In order to show the reverse direction, note that d is a lower bound on the minimum total discrete γ-loss. Combining this observation with Theorem 2.1, we obtain

$$d \leq E_{\gamma}^{D}(M, W, n) = p.$$

•

From Theorems 3.3 and 3.4, we immediately obtain the following result.

Corollary 3.5 *Each algorithm which learns the linear function class $LIN(M, W, n)$*

- *with an (ϵ, γ)-good model needs at least*

$$\Omega\left(\frac{1}{\epsilon}(\min\{(WM/\gamma)^2, n\} + \ln(1/\delta))\right)$$

  examples (assuming that $\epsilon \leq 1/8, \delta \leq 1/100, \gamma \leq WM$),

- *with an ϵ-bounded expected r-loss for $1 \leq r < 2$ needs at least*

$$\Omega\left(\frac{(WM)^r}{\epsilon}(n^{1-r/2} + \ln(1/\delta))\right)$$

  examples (assuming that $\epsilon \leq (WM)^r/2$, $\epsilon \leq (WM)^r/n^{r/2}$),

- *with an ϵ-bounded expected r-loss for $r \geq 2$ needs at least*

$$\Omega\left(\frac{(WM)^r \ln(1/\delta)}{\epsilon}\right)$$

  examples (assuming that $\epsilon \leq (WM)^r/2$).

A comparison with Corollary 3.2 shows that these bounds are tight modulo a constant or logarithmic factor (assuming sufficiently small values of ϵ, δ, γ).

References

[1] Noga Alon, Shai Ben-David, Nicolò Cesa-Bianchi, and David Haussler. Scale-sensitive dimensions, uniform convergence, and learnability. In *Proceedings of the 34th Symposium on Foundations of Computer Science*, pages 292–302. IEEE Computer Society Press, 1993.

[2] P. L. Bartlett, P. M. Long, and R. C. Williamson. Fat-shattering and the learnability of real-valued functions. In *Proceedings of the 7th Annual ACM Conference on Computational Learning Theory*, pages 299–311. ACM Press, 1994.

[3] Ethan J. Bernstein. Absolute error bounds for learning linear functions online. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 160–164, New York, 1992. ACM Press.

[4] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association on Computing Machinery*, 36(4):929–965, 1989.

[5] Nicolo Cesa-Bianchi, Phil Long, and Manfred Warmuth. Worstcase quadratic loss bounds for on-line prediction of linear functions by gradient descent. Research Report UCSC–CRL–93–36, University of California, Santa Cruz, 1983. An extended abstract appeared in COLT'93.

[6] Andrzej Ehrenfeucht, David Haussler, Michael Kearns, and Leslie Valiant. A general lower bound on the number of examples needed for learning. *Information and Computation*, 82(3):247–261, 1989.

[7] David Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Computation*, 100(1):78–150, 1992.

[8] Michael Kearns and Robert E. Schapire. Efficient distribution-free learning of probabilistic concepts. *Journal of Computer and System Sciences*, 48:464–497, 1994.

[9] Nick Littlestone. From on-line to batch learning. In *Proceedings of the 2nd Annual Workshop on Computational Learning Theory*, pages 269–284, Palo Alto, California, 1989. Morgan Kaufmann.

[10] J. Mycielski. A learning algorithm for linear operators. *Proceedings of the American Mathematical Society*, 103(2):547–550, 1988.

[11] Hans Ulrich Simon. Bounds on the number of examples needed for learning functions. In *Proceedings of the 1st European Conference on Computational Learning Theory*, pages 83–95. Oxford University Press, 1993.

A Implementational Details

Bernstein's Algorithm B A more detailed description

data structures "known subspace" $\mathcal{U}$ given by its pairwise orthogonal base vectors $u^1, \ldots, u^p$;
the true target values $d_1, \ldots, d_p$ for $u^1, \ldots, u^p$, respectively;
the values $L_j = \|u^j\|_2^2 = (u^j)'u^j$ for $1 \leq j \leq p$;
the weight vector $\tilde{w}$ representing $B's$ current hypothesis.

Initialization $\mathcal{U} = \{\bar{0}\}, \tilde{w} = \bar{0}$.

Main Loop The following happens in each trial:

1. The environment presents a new instance x.
2. B computes the decomposition $x = u + v$ with $u \in \mathcal{U}$ and $v \in \mathcal{U}^\perp$ according to
$$u = \sum_{j=1}^{p} \frac{x'u^j}{L_j} u^j$$
$$v = x - u$$
and predicts $\tilde{y} = \tilde{w}'u$.
3. The environment presents the true target value y for x.
4. If $y \neq \tilde{y}$, then $u^{p+1} = v$ is inserted into the base of $\mathcal{U}$, its true target value $d_{p+1} = y - \tilde{y}$ and L_{p+1} (the square of the euclidean length of u^{p+1}) are stored, and $\tilde{w}$ is updated according to
$$\tilde{w}^{new} = \tilde{w}^{old} + \frac{d_{p+1}}{L_{p+1}} u^{p+1}.$$

It should be obvious that the decomposition $x = u+v$ is correctly computed and d_{p+1} represents the true target value for u^{p+1}. It follows from the update rule and induction over p that

$$\tilde{w} = \sum_{t=1}^{p} \frac{d_t}{L_t} u^t.$$

Since z is orthogonal to $u^1, \ldots, u^p$, we obtain immediately $\tilde{w}'z = 0$. This implies that $w'x = w'y$, as desired.

After p trials, the algorithm B^{nt} deals with extended pairwise orthogonal vectors

$$\begin{pmatrix} u^1 \\ E^1 \end{pmatrix}, \ldots, \begin{pmatrix} u^p \\ E^p \end{pmatrix}$$

where $u^j \in \Re^n$ and $E^j \in \Re^m$ instead of the original $u^1, \ldots, u^p$. It is however more efficient not to compute the extensions E^j, but the symmetric auxiliary array $R_{i,j} = (E^i)'E^j$ (and the auxiliary quantity $L_j = (u^j)'u^j$ which was already used by B). When a new instance x arrives, then conceptually we must find the orthogonal decomposition of its extension:

$$\begin{pmatrix} x \\ Ke^{p+1} \end{pmatrix} = \begin{pmatrix} u \\ E \end{pmatrix} + \begin{pmatrix} u^{p+1} \\ E^{p+1} \end{pmatrix}$$

where the vector with upper index $p+1$ represents the new direction. However E^{p+1} is never involved into the computation of predictions because all subsequent extended vectors of the prediction sequence have zeroes in positions $1, \ldots, p+1$ of their extensions (and $E^1, \ldots, E^{p+1}$ have zeroes in positions greater than $p+1$). Thus it suffices to compute

$$u = \sum_{j=1}^{p} \frac{x'u^j}{L_j + R_{j,j}} u^j$$
$$u^{p+1} = x - u$$
$$R_{p+1,p+1} = K^2 + \sum_{1 \leq i,j \leq p} \frac{x'u^i}{L_i + R_{i,i}} \frac{x'u^j}{L_j + R_{j,j}} R_{i,j}$$
$$R_{p+1,i} = \sum_{j=1}^{p} \frac{x'u^j}{L_j + R_{j,j}} R_{i,j} \text{ for all } 1 \leq i \leq p$$
$$\tilde{w}^{new} = \tilde{w}^{old} + \frac{d_{p+1}}{L_{p+1} + R_{p+1,p+1}} u^{p+1}.$$

We omit the straightforward proof that these updates are correct. Of course, auxiliary information $R_{i,j}/(L_j + R_{j,j})$ and, temporarily, $(x'u^j)R_{i,j}/(L_j + R_{j,j})$ should be stored as well. The equations were written as above to make them easier to understand. The main effect of the extensions E^j is that they modify the "learning rate" (the scalar in front of u^{p+1} within the update of $\tilde{w}$, which determines how much the old weight is moved in the new direction), which was d_{p+1}/L_{p+1} in B and is now $d_{p+1}/(L_{p+1} + R_{p+1,p+1})$ in B^{nt} (and in the noise-tolerant case $d^{p+1} = y_{p+1} - \tilde{y}_{p+1}$ does not represent the true target value for u^{p+1}).