This is to certify that the

thesis entitled

A NONPARAMETRIC ADAPTIVE DETECTOR
FOR UNKNOWN CHANNELS USING BOOSTING

presented by

Long Ying

has been accepted towards fulfillment
of the requirements for

Master's degree in Electrical Eng

Major professor

Date 8/21/98

A NONPARAMETRIC ADAPTIVE DETECTOR FOR UNKNOWN CHANNELS USING BOOSTING

By

Long Ying

A THESIS

submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Electrical Engineering

1998

ABSTRACT

A NONPARAMETRIC ADAPTIVE DETECTOR FOR UNKNOWN CHANNELS USING BOOSTING

By

Long Ying

Boosting is a new and powerful decision-theoretic machine learning algorithm, which combines a number of simple decision rules into one highly accurate final rule. Without any prior knowledge of the source statistics, boosting is shown capable of learning a decision rule in a classification problem. The objective of this research is to design an adaptive detector that automatically adjusts to unknown environments using a reasonable number of training data. A novel detector for Code Division Multiple Access (CDMA) communication signals is designed using the recently proposed learning algorithm known as Boosting. The new detector performs favorably in comparison to other commonly used detectors including the likelihood ratio detector. The likelihood ratio detector is an optimal detector that minimizes the probability of error, but requires complete knowledge of channel characteristics. In contrast, the new adaptive detector proposed here makes no assumptions on the channel. Hence, the new detector is potentially useful for realistic situations in which little is known about the communication channel.

Dedicated to Jane

ACKNOWLEDGEMENTS

This interesting research topic is suggested by my advisor Dr. Robert D. Nowak. I am most grateful for Dr. Nowak's help during this research and during the preparation of the thesis, and for his support and guidance throughout my studying for the Master's degree. I would also like to thank Dr. John Deller Jr. and Dr. Percy Pierre for their wisdom and many important suggestions for this thesis.

Contents

LIST OF FIGURES vii

1 INTRODUCTION 1

2 CDMA COMMUNICATIONS 3

2.1 The CDMA Signal Structure 3

2.2 Detection of CDMA Signals 5

2.2.1 Matched Filter 5

2.2.2 Sign Detector 6

2.2.3 Adaptive Detection 8

3 BOOSTING 13

3.1 Background on Machine Learning 13

3.2 Boosting and AdaBoost 15

3.3 Experiment with AdaBoost 18

3.4 Performance Analysis 21

4 DETECTION FOR UNKNOWN CHANNELS USING BOOSTING 24

4.1 Detecting CDMA Signals Using AdaBoost: Method 1 24

4.2 Detecting CDMA Signals Using AdaBoost: Method 2

—the AdaBoost Detector 25

5 EXPERIMENTS AND DISCUSSION 29

5.1 AdaBoost Detector in Additive White Noise Channel 29

5.2 Discussion 32

6 CONCLUSION **35**

A VC DIMENSION **38**

REFERENCES **40**

List of Figures

1 The CDMA signal structure. 4

2 Matched filter detection for CDMA signals. 5

3 Sign detection for CDMA signals. 7

4 A setup of machine learning. 13

5 Algorithm AdaBoost 17

6 An experiment with AdaBoost. 20

7 AdaBoost detector with less training data 30

8 Experiments with the AdaBoost detector. 31

1 INTRODUCTION

In digital communication systems, one of a finite number of symbols is sent during each bit period. Due to imperfections in the channel and in the receiver's front end, the received signal of each transmitted symbol is modeled as a random variable. An optimal detector, in the sense of minimizing the probability of error is the likelihood ratio test. *Throughout this thesis, the term "optimal" refers to the minimum probability of error criterion.* The likelihood ratio test requires complete knowledge of the received symbol distributions. For some systems, the channel uncertainties can be modeled very well, and the received signal distributions are known, so we are able to design fixed detectors that are optimal (e.g., matched filters for additive white Gaussian noise, or AWGN, channels [4]). More frequently, however, accurate models for the channels are not available due to unknown noise sources, unknown interference, and the fact that the channels can be time-varying. In such situations, an adaptive detector that adjusts to the changing environment, is necessary.

Depending on the amount of knowledge about the channels, some adaptive detectors assume certain parametric forms of the received symbol distributions and estimate the unknown parameters from some training data; others adopt nonparametric schemes that assume very little about the channels. For example, in a recent study [1], a *type-based detector*, which uses histograms built from training data to estimate the distributions, provides asymptotic optimal performance with no assumptions about the channel. In many wireless digital communication systems, the channel is very complex, making *a priori* modeling very difficult if not impossible. Therefore

nonparametric adaptive detectors are highly desirable.

The general goal of this research is to design a nonparametric adaptive detector using a new and powerful decision-theoretic learning algorithm known as *boosting*. In boosting, a number of simple decision rules are first generated using training data alone—without any knowledge of the received signal distributions. Then these simple rules are combined (or "boosted") to produce an overall decision rule that approximates the optimal likelihood ratio test. This research results in a specific product—the AdaBoost detector. It is aimed at extracting the code division multiple access (CDMA) communication signals from background noises without knowledge of the noise statistics.[1]

This thesis will first explain the structure and the usual detection methods of the CDMA signals. Then it will introduce the idea of boosting in detail, with a specific algorithm called "AdaBoost" as an example. It will also describe the development of the AdaBoost Detector, as well as some experiments with this detector.

[1]Experiments conducted at the time of writing this thesis also showed that the detector can do better than others in rejecting certain interference encountered in wireless channels. However, time did not allow a comprehensive discussion on that subject to be included in this thesis.

2 CDMA COMMUNICATIONS

CDMA is a transmission scheme found in the wireless mobile communications networks, wireless local area networks, global positioning systems, etc., and is becoming more and more important with the rapid growth of the telecommunications industry [6]. The following is a brief introduction to CDMA communications. The reader is referred to the text book by Proakis [4] or Ziemer and Peterson [5] for details.

2.1 The CDMA Signal Structure

The CDMA signal structure can be explained using Figure 1. This example shows some antipodal binary CDMA signals. The signal level is either $+1$ or -1. Figure 1a shows two message symbols to be communicated at a rate of $1/T_b$ bits/sec (bps). The key to CDMA is that each message symbol must be multiplied ("spread") by a sequence of N pseudo-random bits. This sequence is called a *code*; and each bit in the code is termed a *chip*. Figure 1b shows two copies of an 8-chip[2] code, each to be multiplied to a symbol to provide the chip sequence in Figure 1c. For each symbol of level m, this operation can be expressed as

$$s[n] = m\, c[n], n = 1, \ldots, N. \tag{1}$$

Therefore instead of sending the message symbol directly, one would transmit the final sequence (s[1], ... ,s[N]).

[2]In reality, N is much larger than 8 (typically over 100).

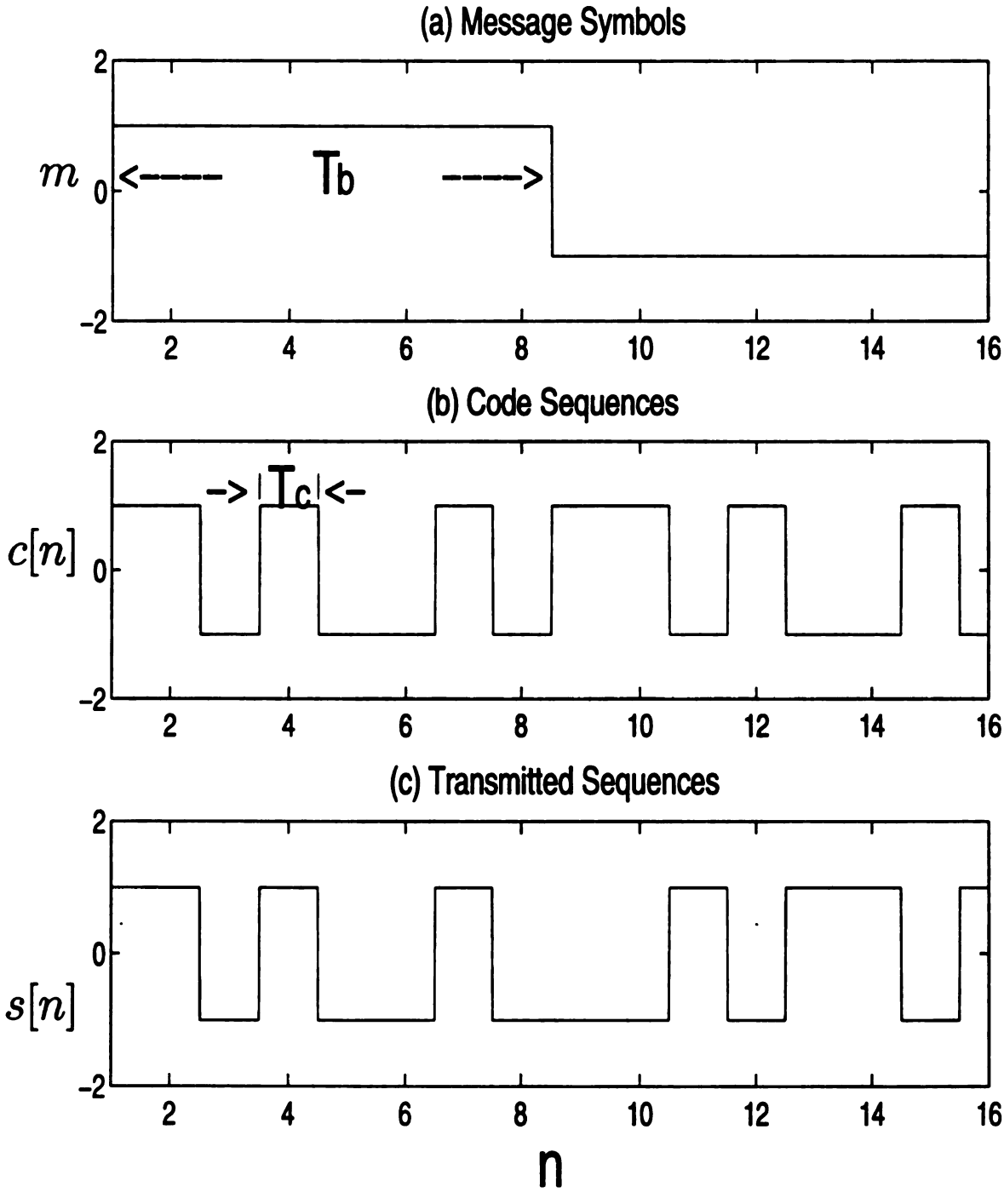


Figure 1: The CDMA signal structure.

2.2 Detection of CDMA Signals[3]

2.2.1 Matched Filter

To recover the original message m in this type of signals, one needs to decide whether the received N-chip sequence is a result of transmitting the code or the negative of the code. It is easily shown that in AWGN channels, the optimal decision is made by a matched filter, or equivalently, by correlating the received sequence with the original code, as shown in Figure 2.

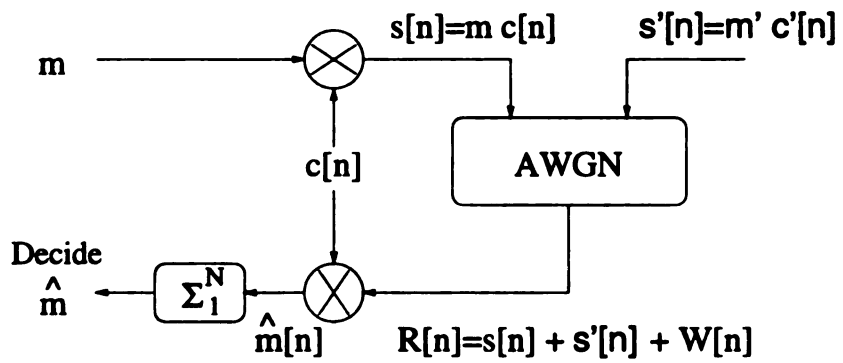


Figure 2: Matched filter detection for CDMA signals.

To make subsequent discussions easier, let us use vectors (shown in bold face) to denote length-N sequences. For example, $\mathbf{R}$ denotes the sequence $(R[1], \ldots, R[N])$. I will also use capital letters to denote random variables throughout the thesis. Notice that in addition to the transmitted signal $\mathbf{s}$ and the noise $\mathbf{W}$, the received sequence $\mathbf{R}$ also contains another sequence $\mathbf{s}'$. The signal $\mathbf{s}'$ is intended for another receiver, containing a different message m', spread by a different code $\mathbf{c}'$. The key is that the

[3]Throughout the thesis, I will assume that the transmitter and the receiver are synchronized. A good introduction on synchronization can be found in [4].

code $\mathbf{c}'$ is designed to be almost orthogonal to $\mathbf{c}$, i.e.

$$\sum_{n=1}^{N} c[n]\, c'[n] \cong 0. \tag{2}$$

Therefore, the correlation output due to s' is close to 0. This means other users can transmit their messages on the same channel at the same time—CDMA.

This matched filter is the backbone of the most widely used CDMA receiver—the *Rake receiver* [6]. The Rake receiver is a series of matched filters, combined to rake in energies from multiple signal paths to achieve diversity.[4] It is important to know that the use of the matched filter for every path is based on the assumption that the channel adds white Gaussian noise to the signal. Unfortunately, this is not always true, especially when other users' codes are not orthogonal to that of the receiver's or when intentional jamming is present. In such cases, it is almost impossible to accurately model the received signal distributions.

2.2.2 Sign Detector

According to [1], a sign detector does not assume an AWGN channel. In fact, it does not assume the noise distribution to be of any parametric form. It is the optimal *fixed* detector for zero-median noise. Figure 3 illustrates the operation of a sign detector.

Each received chip $R[n]$ is individually classified to $\hat{m}[n]$ according to its sign. Once all N chips are determined, the sequence is compared to both the code and the negative of the code to decide which message symbol was sent. In the case depicted in Figure 3, it is decided that the sequence $\mathbf{R}$ comes from transmitting the symbol

[4]For a more comprehensive discussion on the Rake receiver and the problem of multipath in communications, the reader is referred to the works of Price and Green [7] and Rappaport [6].

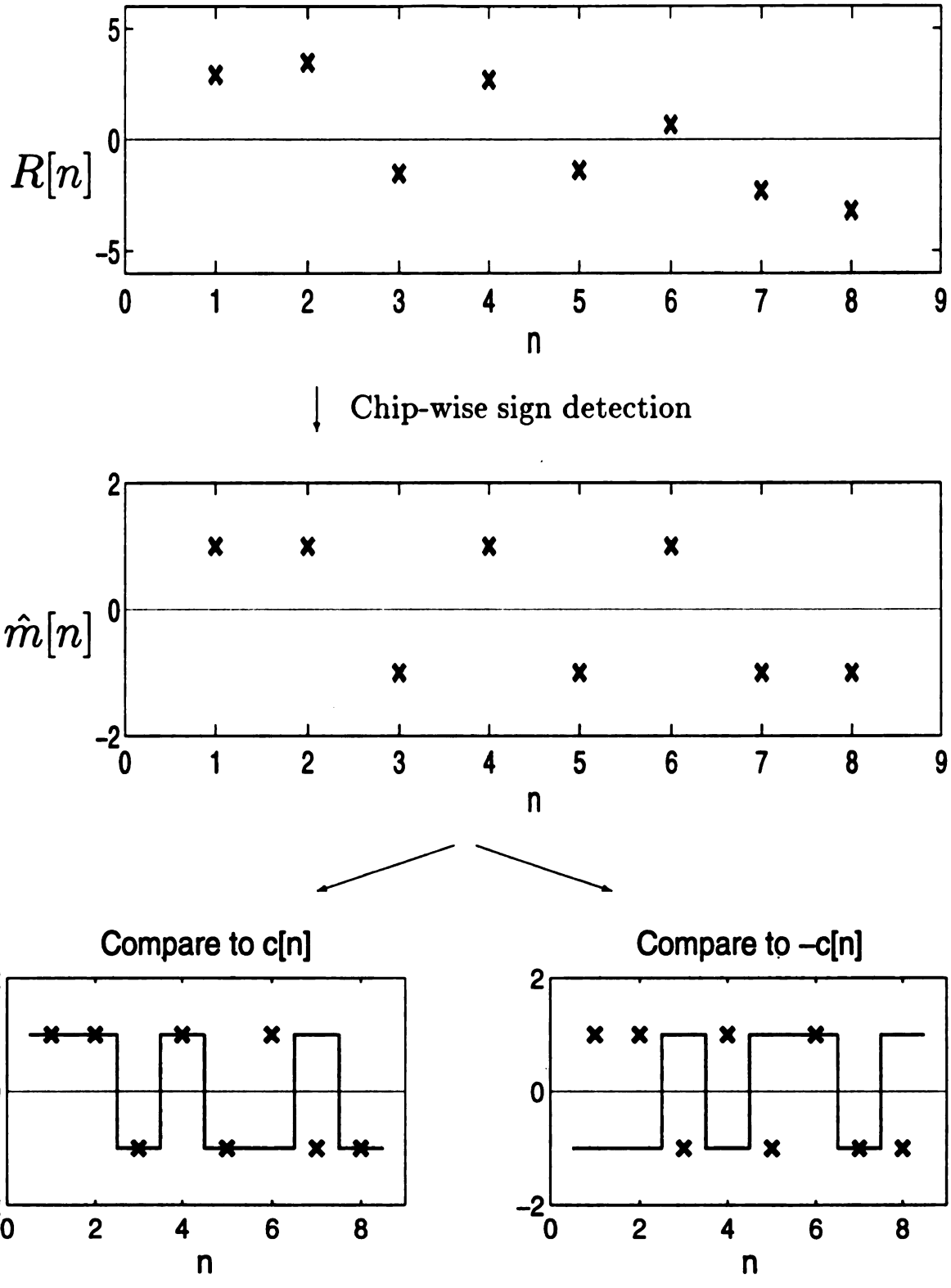


Figure 3: Sign detection for CDMA signals.

+1 because the sequence $\hat{\mathbf{m}}$ differs from the positive code by only two chips.

2.2.3 Adaptive Detection

In a recent series of papers [1, 2, 3], Johnson *et al.* have developed a universal adaptive detector called the *type-based detector.* A *type*, in information theory, means the histogram estimate of a discrete probability distribution [11]. The type-based detector uses the types constructed from some training data to estimate the received signal distributions in order to asymptotically achieve optimal (minimum probability of error) performance. The rest of this section will explain the use of this detector for CDMA communications. Some of the operations of this detector are used in the implementation of the AdaBoost detector as well.

To construct types, the received signals must be quantized. Suppose the signal quantization levels are $\{a_t\}_{t=1}^{T}$, and that a set of quantized observations from one class is $\{[x_m]\}_{m=1}^{M}$.[5] Then the type for that class (the histogram estimate of its distribution) is simply:

$$P_x(a_t) = \frac{1}{M} \sum_{m=1}^{M} I([x_m] = a_t), t = 1, \ldots, T \tag{3}$$

where $I(\cdot)$ is the indicator function.

In antipodal binary communications, there are two classes of chips, for which types must be constructed. For the purpose of grouping chips into these classes, define the sets n^+ and n^- to be the *positions* of all positive chips and all negative

[5] I will differ the discussion on the quantization process to Section 5.2. The reader should refer to [2] for details. Also, I will use $[x_m]$ to denote the *quantized* signals and to distinguish it from the unquantized x_m appearing in later sections.

chips, respectively, in the code. That is

$$\begin{aligned} n^+ &= \{n : c[n] = +1\} \\ n^- &= \{n : c[n] = -1\}. \end{aligned} \tag{4}$$

The receiver knows these sets. Therefore by observing the received chips' positions, the detector can easily separate them into two classes, and construct the two types. To be more specific, the chips in a sequence [x] are separated according to the following:

$$\begin{aligned} x^1 &= \{[x[n]] : n \in n^+\} \\ x^2 &= \{[x[n]] : n \in n^-\}. \end{aligned} \tag{5}$$

It is important to notice that if the sequence corresponds to a $+1$ symbol, the group x^1 will contain all the $+1$ chips, but if the sequence represents the symbol -1, then the group x^2 will contain the $+1$ chips instead. Therefore one can identify the chip groups only if the transmitted symbol is known (which is true in training), but one has to hypothesize the signs of the groups when *unknown* sequences are received.

In order to use a type-based detector, it is required that a labeled training set be transmitted periodically, $([\mathbf{x_1}], y_1), \ldots, ([\mathbf{x_L}], y_L)$, where the label y_l is the lth symbol, and the observation $[\mathbf{x_l}]$ is the received and quantized sequence of that symbol. As described, all the chips are separated into a group of $+1$s x^+ and a group of -1s x^-, each having $M = 1/2(L \times N)$ chips[6]. The types for the two classes of chips P_{x^+} and P_{x^-} are constructed according to (3). This completes the training phase of the type-based detector.

[6]The length N of a pseudo-random sequence is usually odd, but transmitting $(1/2\,L)$ $+1$ symbols and $(1/2\,L)$ -1 symbol can result in equal number (M) of oppositely signed chips. This is not a requirement, but nonetheless assumed for simplicity.

After the training, a sequence [**R**] corresponding to an unknown message symbol m is received (with the same quantization). At this point, if the chips are independent (result of a memoryless channel), one only needs to estimate the likelihood ratio of each chip using the types. That is

$$\begin{aligned}\hat{\lambda}_n &= \frac{p([R[n]]\,|s[n]=+1)}{p([R[n]]\,|s[n]=-1)} \\ &= \frac{P_{x^+}([R[n]])}{P_{x^-}([R[n]])}.\end{aligned} \tag{6}$$

Then the likelihood ratio of the whole sequence can be found.

$$\begin{aligned}\hat{\Lambda} &= \frac{\prod_{n=1}^{N} p([R[n]]\,|m=+1)}{\prod_{n=1}^{N} p([R[n]]\,|m=-1)} \\ &= \frac{\prod_{n\in n^+}\hat{\lambda}_n}{\prod_{n\in n^-}\hat{\lambda}_n}.\end{aligned} \tag{7}$$

Therefore the decision will be:

$$\hat{\Lambda} \underset{\hat{m}=-1}{\overset{\hat{m}=+1}{\gtrless}} 1. \tag{8}$$

However, according to Ziv [16], this structure can not be generalized to accommodate channels with memories. It turns out that the following method is more versatile because it can be generalized to cope with interchip interference using higher order types. The theory behind this is developed by Gutman [15].

After training, the type-based detector also separates the unknown sequence [**R**] into two groups of chips $R^+ = \{[R[n]] : n \in n^+\}$, and $R^- = \{[R[n]] : n \in n^-\}$. Notice the chips in the set R^+ are positive only if the sequence represents the symbol $m = +1$. And in that case, the chips in this set and in the set x^+ (found in training) should come from the same distribution. Based on this idea, the type-based detector

constructs four more types:

$$
\begin{aligned}
P_{Z_1^+} &= (\frac{N}{2}P_{R^+} + MP_{x^+})/(\frac{N}{2}+M) \\
P_{Z_1^-} &= (\frac{N}{2}P_{R^-} + MP_{x^-})/(\frac{N}{2}+M) \\
P_{Z_0^+} &= (\frac{N}{2}P_{R^-} + MP_{x^+})/(\frac{N}{2}+M) \\
P_{Z_0^-} &= (\frac{N}{2}P_{R^+} + MP_{x^-})/(\frac{N}{2}+M)
\end{aligned} \tag{9}
$$

where P_{R^+} and P_{R^-} are the types made from R^+ and R^-. These four types are formed from the sets constructed by pairing R^+ and R^- with x^+ and x^- in both ways. This allows the detector to hypothesize as follows:

$$
\begin{aligned}
H_1 : m = +1 \quad &: \quad \begin{cases} (R^+, x^+) \sim P_{Z_1^+} \\ (R^-, x^-) \sim P_{Z_1^-} \end{cases} \\
H_0 : m = -1 \quad &: \quad \begin{cases} (R^-, x^+) \sim P_{Z_0^+} \\ (R^+, x^-) \sim P_{Z_0^-} \end{cases} .
\end{aligned} \tag{10}
$$

It can be shown that the log likelihood of all the types given each hypothesis is:

$$
\begin{aligned}
\Gamma_1 &= -(\frac{N}{2}+M)H(P_{Z_1^+}) + \frac{N}{2}H(R^+) + MH(x^+) \\
&\quad -(\frac{N}{2}+M)H(P_{Z_1^-}) + \frac{N}{2}H(R^-) + MH(x^-) \\
\Gamma_0 &= -(\frac{N}{2}+M)H(P_{Z_0^+}) + \frac{N}{2}H(R^-) + MH(x^+) \\
&\quad -(\frac{N}{2}+M)H(P_{Z_0^-}) + \frac{N}{2}H(R^+) + MH(x^-)
\end{aligned} \tag{11}
$$

where $H(\cdot)$ is the entropy function. Notice that these test statistics combine the training and testing data, which is a requirement for optimal detection in channels

with memories. The decision rule based on these statistics is then simply:

$$\Gamma_1 \underset{H_0}{\overset{H_1}{\gtrless}} \Gamma_0. \tag{12}$$

The effectiveness of the type-based detector will be demonstrated in Section 5, where the performance of all the detectors considered in this thesis are compared.

Since CDMA is widely used in wireless communications, it suffers most of the wireless channel impairments.[7] Very often, the channel does not fit the model for which the current Rake receiver works best. The type-based detector, in these situations, is very desirable because it can adapt to any models.

The next section will describe an algorithm called boosting, which is also well suited for performing classifications without any assumption. It is the purpose of this thesis to incorporate it into the detection of CDMA signals.

[7] CDMA also provides some resistance against some of the channel imperfection as part of its virtue.

3 BOOSTING

Roughly speaking, boosting is a learning algorithm that combines or boosts a number of simple classifiers to obtain an overall more powerful decision rule. Boosting is an algorithm developed in the area of machine learning. Therefore it is appropriate to explain it in that context.

3.1 Background on Machine Learning

One setup of a machine learning problem is illustrated in Figure 4. The objective is to

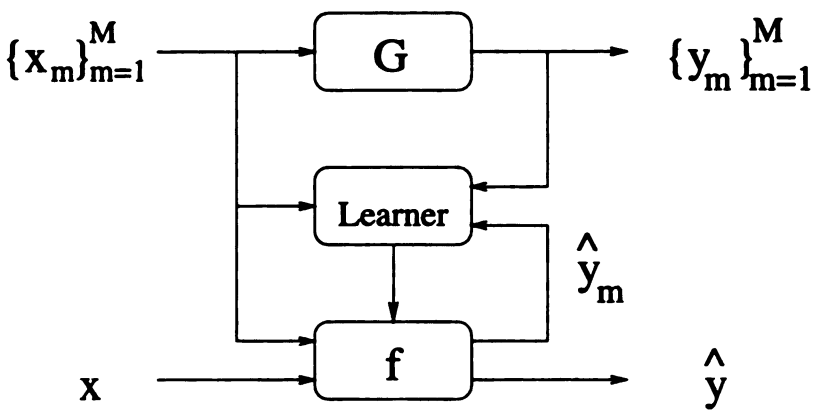


Figure 4: A setup of machine learning.

adaptively learn about the function G that maps an input x to an output y. When the setup is used in classification problems, G is just a classifier that maps an observation into a class. Because of the nature of this research, I will only talk about using the learning machine to learn the classification rule even though this setup is much more general.[8] One of the resources for learning is a set of training data $\{x_m\}_{m=1}^{M}$ with

[8]Other applications are found in the work of Vapnik [13].

known labels (or classes) $\{y_m\}_{m=1}^{M}$. The central component of this setup is the learning machine, henceforth called the learner. It is capable of implementing any function (or classifier) f from a specified family $\mathcal{F}$. These functions can also map any x into a class $\hat{y}$. Thus when the learner selects one $f \in \mathcal{F}$ to classify a training observation x_m, it can observe the output $\hat{y}_m$, compare it to the label y_m, and determine how good the classifier is. After the learner tries all the classifiers on all the training data, it will pick the one that performs the best—f_G, and the learning is finished. f_G will then be used to classify all future observations.

More specifically, the learning process is described as follows. The learner must be provided with a performance criterion in order to compare the performance of different classifiers. The performance criterion is a loss function denoted by $L(y_m, \hat{y}_m(f))$. In classification problems, this loss can be an indicator of the classification error of f on x,[9] also called 0/1 error, i.e.

$$L(y, \hat{y}(f)) = \begin{cases} 0 & \text{if } y = \hat{y}(f) \\ 1 & \text{if } y \neq \hat{y}(f) \end{cases} . \tag{13}$$

Ideally, we would like the average loss over all possible observations to be as small as possible. Therefore the learner should choose an f to minimize the *expected risk*,

$$\mathcal{E}(f) = \int L(y, \hat{y}(f))\, dF(x, y) \tag{14}$$

where $F(x, y)$ is the underlying distribution over the space $X \times \{\pm 1\}$ for a binary classification problem. The expected risk is precisely the probability of error incurred using f as a classifier. The likelihood ratio test minimizes this error over all functions,

[9]To emphasize that L is the loss of the classifier f, x is eliminated in the following equations. It is understood that $\hat{y}(f)$ means $f(x)$.

and therefore it is called the minimum probability of error detector. However, the likelihood ratio test requires complete knowledge of the distribution F, which is not practical. In reality, F is unknown and the learner has access to only the set of training data. Therefore one practical alternative is to adopt the *empirical risk minimization (ERM)* principle, which chooses the classifier f_e that minimizes the empirical risk. That is

$$f_e = \arg\min_{f \in \mathcal{F}} \mathcal{E}_e(f) = \arg\min_{f \in \mathcal{F}} \frac{1}{M} \sum_{m=1}^{M} L(y_m, \hat{y_m}(f)). \tag{15}$$

3.2 Boosting and AdaBoost

For this setup, it is essential to find a good set of functions $\mathcal{F}$, since the learner's ability is limited by the best classifier in $\mathcal{F}$. In order to ensure that $\mathcal{F}$ contains a member f that closely matches the desired mapping G, we either require considerable prior knowledge about G, or we need a very large and complicated set $\mathcal{F}$. In many problems we have little or no information about G. Therefore, $\mathcal{F}$ must include a large number of (possibly very complicated) classifiers. These leads to two problems.

1. It may be difficult to perform the optimization in (15).
2. It may lead to overfitting and poor generalization characteristics when f_e is applied to unlabeled data outside the training set.

Fortunately, a change of mindset is provided by a class of learning algorithms that achieves high accuracy by voting according to the predictions of several classifiers [9]. One such method is called *boosting.*

In boosting, instead of finding a large and necessarily complex set of functions $\mathcal{F}$ containing the desired classifier, the algorithm only finds a number of *weak* classifiers[10] $\{h_t\}_{t=1}^{T}$ that may perform just better than random guessing. For an observation x, each of these weak classifiers will predict its class—$h_t(x)$. The algorithm will combine ("boost") them into a more accurate final prediction—$h_f(x)$, according to a weighted voting criterion.

An actual boosting algorithm will help to understand some details of this idea. Figure 5 is one of the most successful boosting algorithms called "AdaBoost" [8].[11] This algorithm starts by assigning an initial probability measure $\mathbf{p^1} = (p_1^1, \ldots, p_M^1)$, usually uniform, over the training set $(x_1, y_1), \ldots, (x_M, y_M)$. It then enters a loop (indexed by t), where it calls a subroutine called *WeakLearn*. WeakLearn finds a weak classifier h_t for that round, according to a *modified* ERM (MERM) principle:

$$h_t = \arg\min_{h \in \mathcal{H}} \left\{ \mathcal{E}_{mod}(h) = \sum_{m=1}^{M} p_m^t L(y_m, h(x_m)) \right\}. \tag{16}$$

Notice that (16) differs from (15) in two regards: 1) The set of functions $\mathcal{H}$, in which h_t is found, is much smaller than the set $\mathcal{F}$. It is not based on too much knowledge about the underlying classification rule to be learned. 2) Instead of weighting the loss of each training sample equally with a weight $1/M$, WeakLearn averages its loss according to the probability measure p_m^t assigned by the boosting algorithm. The reason for this will become clear shortly.

[10] In boosting literature, these are called weak *hypotheses*. To avoid confusion with the word *hypotheses* used in detection theory, weak *classifier* is used throughout this thesis.

[11] The original AdaBoost algorithm in [8], which classifies between 1 and 0, is slightly modified here in order to classify between 1 and -1.

Input: • training set $\{(x_1, y_1) \ldots (x_M, y_M)\}$

• initial distribution $\mathbf{p^1} = [p_1^1 \ldots p_M^1]$ over the M samples (e.g. uniform)

Initialize: $\mathbf{w^1} = \mathbf{p^1}$

Do for $t = 1, 2, \ldots, T$

1. Set $\mathbf{p^t} = \dfrac{\mathbf{w^t}}{\sum_{m=1}^{M} w_m^t}$
2. Call WeakLearn, providing it with $\{(x_m, y_m, p_m^t)\}_{m=1}^{M}$; get back a weak decision rule $h_t : X \to \{\pm 1\}$
3. Calculate $\epsilon_t = \sum_{m=1}^{M} p_m^t I[y_m \neq h_t(x_m)]$
4. Set $\beta_t = \epsilon_t / (1 - \epsilon_t)$
5. Set the new weights to be:

$$w_m^{t+1} = w_m^t \beta_t^{1 - I[y_m \neq h_t(x_m)]}, m = 1, 2, \ldots, M$$

Output: the final rule $h_f(x)$ is

$$\sum_{t=1}^{T} I[h_t(x) = +1] \log(\frac{1}{\beta_t}) \underset{h_f(x) = -1}{\overset{h_f(x) = +1}{\gtrless}} \sum_{t=1}^{T} I[h_t(x) = -1] \log(\frac{1}{\beta_t})$$

Figure 5: **Algorithm AdaBoost**

The minimized risk $\epsilon_t = \min_{h \in \mathcal{H}} \mathcal{E}_{mod}(h)$ is a by-product of WeakLearn. Next the quantity β_t is calculated. This quantity increases from 0 to ∞ as the risk of h_t increases from 0 to 1. This is an important quantity whose significance is clearly demonstrated in the final decision rule.

The last step for each round is a key to boosting. The algorithm updates the probability measure over the training set for the next round. Notice that **w** is used here instead of **p**. This is because **w** will be normalized to a true probability measure **p** at the beginning of the next round. The important feature of this updating step is that it relatively increases the weights of the training samples that h_t misclassified. When these weights are carried into the next WeakLearn subroutine, they tend to make the WeakLearn produce the next weak classifier that works better for these "hard" samples (because the WeakLearn uses the MERM principle taking $\mathbf{p}^t$ into account).

After T rounds, T weak hypotheses $\{h_t\}_{t=1}^T$ are generated, each associated with a quantity β_t indicating its risk. The final rule says, apply all $\{h_t\}_{t=1}^T$ to an observation x, weight each vote $h_t(x)$ with $\log(1/\beta_t)$, and the decision is the class receiving the weighted majority vote. Obviously, the more risk h_t has, the less important its vote becomes.

3.3 Experiment with AdaBoost

The following binary classification experiment is used to demonstrate the effectiveness of AdaBoost.

The observation x is one dimensional. There are two classes labeled $y = \pm 1$. The pdf of the first class—$p(x|y = +1)$ is zero-mean, small-variance Gaussian. The class -1 has a large-variance Gaussian distribution with the same mean. Figure 6a shows these density functions. The optimal (minimum probability of error) classifier results from the maximum likelihood criterion,

$$p(x|y=+1) \underset{\hat{y}=-1}{\overset{\hat{y}=+1}{\gtrless}} p(x|y=-1) \tag{17}$$

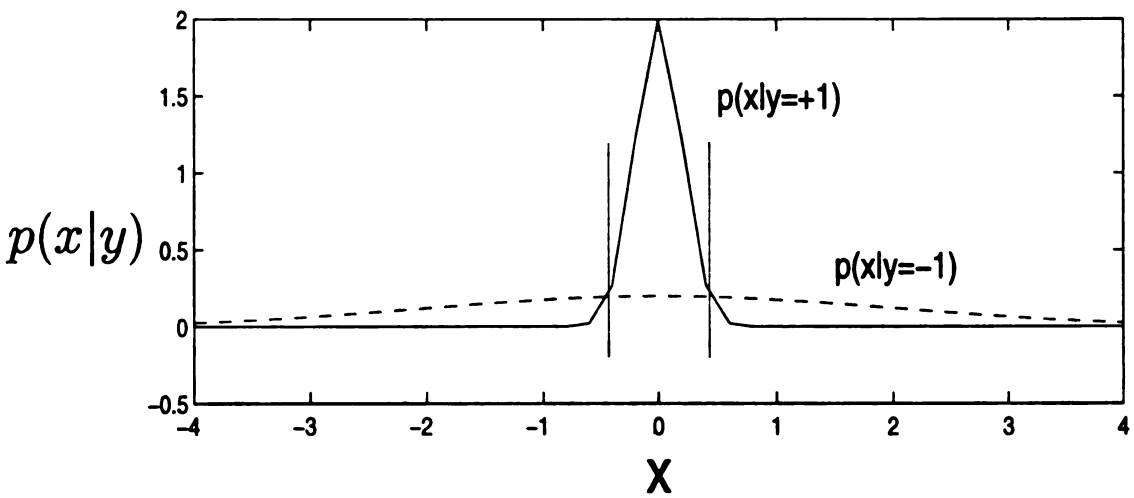
which corresponds to the decision regions separated by the two thresholds (visualized by the two vertical bars in Figure 6a). Thus the decision rule to be learned will classify the observations between these two thresholds as $+1$s and those outside as -1s.

To design a learning algorithm for this problem using AdaBoost, a class of weak classifiers have to be found. The set chosen in this case is:

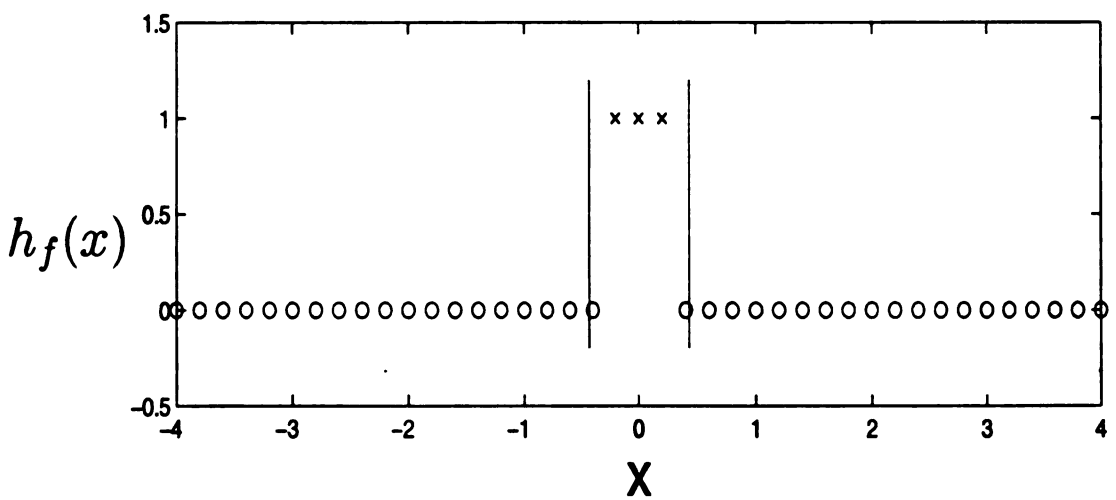
$$\mathcal{H} : \{h(x, v, d) = sgn\left[(x - v)\, d\right] : v \in \mathcal{R}, d \in \{\pm 1\}\}. \tag{18}$$

This classifier is simply a directed threshold. For example, when the direction $d = +1$, any observation x that falls above the threshold v is classified as $+1$, and the -1 class is on the other side of v. The size of the training set is chosen to be 100 for this experiment; and the number of rounds of boosting T is also 100. Therefore, the algorithm sequentially generates 100 of these directed threshold, "remembering" the performance of each of them.

Then a sequence of new observations are presented to the final decision rule. For each observation, the algorithm applies the weighted voting criterion described in



(a) Source distributions



(b) AdaBoost classification results.

Figure 6: An experiment with AdaBoost.

Section 3.2 to make the final prediction. As shown in Figure 6b, the observations in the middle are classified as $+1$s; and -1s are on both sides. The double thresholds from the optimal classifier are also displayed in this graph. The final predictions are obviously very close to being optimal.

Notice that it is very hard to foresee a double threshold type of classifier without any knowledge of the distributions. But with boosting, one does not need such clairvoyance. In fact, even if the optimal decision rule has a more complex decision region (e.g. multiple thresholds), the same algorithm can learn it.

3.4 Performance Analysis

In the area of machine learning, there has been extensive research regarding the performance of the learning machines. For a more comprehensive discussion, the reader is again referred to the book by Vapnik [13].

Here, let us review some of the performance bounds. Recall that using the ERM principle, the learner can find a classifier $f_e \in \mathcal{F}$ that is closest to the function G being learned. It was shown [13], with simple constraint on the set $\mathcal{F}$, that both the empirical risk and the actual risk of f_e converge in probability to the minimum possible risk provided by any function in $\mathcal{F}$. The constraint is that the Vapnik-Chervonenkis (VC) dimension k of the set $\mathcal{F}$ be finite. The concept of VC dimension is introduced in the appendix.

Bounding the *empirical* risk is not a difficult task. The training error (same as

empirical risk[12]) of the final rule h_f in AdaBoost is bounded as follows [8]:

$$\mathcal{E}_e(h_f) \leq \exp\left[-2\sum_{t=1}^{T}(1/2 - \epsilon_t(h_t))^2\right]. \tag{19}$$

Notice that for a binary classification problem, $\epsilon_t(h_t)$ is always smaller than $1/2$, making the training error decay exponentially as the number of weak classifiers, T, increases.

A more important issue is how h_f performs when it is applied to instances *outside* the training set; i.e. is the generalization error (same as actual risk[13]) of h_f bounded? Vapnik proved that as long as h_f comes from a class of classifiers with finite VC dimension, its generalization error is bounded [13]. To meet this condition, Freund and Schapire [8] first showed that h_f belongs to the class $\Theta_T(\mathcal{H})$, the set of all linear combinations of the T weak classifiers $\{h_t\}_{t=1}^{T}$. Then they proved that if $\mathcal{H}$ has a finite VC dimension k_H,[14] then the class of final decision rules $\Theta_T(\mathcal{H})$ has a VC dimension

$$k_\Theta \leq 2(k_H + 1)(T + 1)\log_2 e(T + 1). \tag{20}$$

Finally, since h_f minimizes the training error and has a finite VC dimension k_Θ, its generalization error is bounded according to Vapnik [13] as follows:

$$\Pr\left\{\mathcal{E}(h_f) \leq \mathcal{E}_e(h_f) + \sqrt{\frac{k_\Theta(\ln\frac{2M}{k_\Theta} + 1) - \ln(\eta/4)}{M}}\right\} \geq 1 - \eta, \qquad \eta > 0. \tag{21}$$

Therefore the actual risk decays as $1/\sqrt{M}$ with high probability, where M is the size of the training set.

[12]Freund and Schapire [8] uses "training error" to denote the empirical risk of the final classification rule. Therefore I will use these expressions interchangeably.

[13]The term "generalization error" is used by Freund and Schapire to mean the actual risk of h_f. Thus I will use these two expressions interchangeably as well.

[14]This is a reasonable assumption since $\mathcal{H}$ is simple. For example, the class defined in (18) has VC dimension 2.

Both of these bounds are distribution-independent. A more appealing fact is that the second bound describes how the algorithm performs when the size of the training set is small. There is also a trade-off between the two bounds. As T increases, the training error in (19) vanishes; but the VC dimension of the class $\Theta_T(\mathcal{H})$, k_Θ, increases, causing the upper bound on the generalization error in (21) to increase as well. This trade-off between the training error and the generalization error can be used to guide the choice of T.

4 DETECTION FOR UNKNOWN CHANNELS USING BOOSTING

Since classification with boosting does not require assumptions about the source distributions, and since the performance is quantifiable even with limited training data, boosting is ideally suited for adaptive detection of digital signals in unknown communication channels such as the wireless channels.

4.1 Detecting CDMA Signals Using AdaBoost: Method 1

A straightforward application of boosting in digital communications is detection of binary signals. Here one of two possible symbols is sent during each bit period. As a result of the channel impairments, the received bit R is modeled as a continuous random variable, which is governed by one of two probability distributions $p(R|s = +1)$ or $p(R|s = -1)$, depending on the transmitted symbol s. Our experiment in Section 3.2 showed that the AdaBoost algorithm with the weak classifier family defined in (18) was capable of learning a classification rule in exactly this type of situations, without any assumption about those distributions. The only requirement is for the system to periodically transmit a set of labeled bits $\{(x_1, y_1) \ldots (x_M, y_M)\}$ as the input training data to the receiver. This can easily be done during synchronization, for example.

In CDMA communications, however, a pseudo-random sequence of chips is transmitted in place of each message symbol, as explained in Section 2.1. Therefore, to best determine the symbol that was sent, one must observe the whole sequence of

received chips $R[n], n = 1, \ldots, N$ during the bit interval of that symbol.

A natural method is to first use AdaBoost's bit-level detection capability to classify each chip separately. Then, compare the resulting binary sequence to the code and its negative. The decision is made by deciding according to the closest match. This method is very similar to that of the sign detector illustrated in Figure 3. The advantage of using AdaBoost here is its adaptability to many types of noises.[15]

In general however, the method described above is suboptimal. For example, when the noise is zero-mean white Gaussian, its performance can only approach that of the sign detector at best, not the matched filter.[16]

4.2 Detecting CDMA Signals Using AdaBoost: Method 2 —the AdaBoost Detector

This section derives the product of this research—the AdaBoost detector. As described in Sections 2.1 and 2.2.1, each message symbol m is spread by an N-chip code sequence $\mathbf{c} = (c[1], \ldots, c[N])$. The resulting binary sequence $\mathbf{s}$ is transmitted through an unknown channel and received as a sequence of N *continuous*-valued chips $\mathbf{R}$. (See Figures 1 and 2.) Frequently, a set of training symbols $(y_1, \ldots, y_L)$, known to the detector, is sent. The corresponding received sequences are $(\mathbf{x_1}, \ldots, \mathbf{x_L})$. The AdaBoost detector must train itself with this set, and make classifications for future sequences.

[15]However, this may be overshadowed by the sign detector's efficiency when the noise is zero-median.

[16]Experiments demonstrating this were performed, but not included in the thesis.

First, let us look at the operation of the optimal (minimum probability of error) detector. The optimal detector for CDMA signals is based on the knowledge of the likelihood functions of both positive and negative chips—$p(R[n]|s[n] = +1)$ and $p(R[n]|s[n] = -1)$. From these, the detector can calculate the likelihood ratio for each received chips:

$$\lambda_n = \frac{p(R[n]|s[n] = +1)}{p(R[n]|s[n] = -1)}. \tag{22}$$

However the optimal detector does not make decisions on the chips. Instead, it needs to calculate the likelihood ratio of the whole N-chip sequence over the bit interval. Assuming the chips are independent, identically distributed, the likelihood ratio is

$$\Lambda = \prod_{n=1}^{N} \frac{p(R[n]|m = +1)}{p(R[n]|m = -1)}. \tag{23}$$

With the knowledge of the chip sets n^+ and n^- (see the definitions in (4) and Section 2.2.3), the above likelihood ratio can also be expressed as:

$$\Lambda = \frac{\prod_{n \in n^+} \lambda_n}{\prod_{n \in n^-} \lambda_n}. \tag{24}$$

The optimal detector makes a decision based on this likelihood ratio:

$$\Lambda \underset{\hat{m}=-1}{\overset{\hat{m}=+1}{\gtrless}} 1. \tag{25}$$

The detector described above requires complete knowledge of the likelihood ratio, and therefore is impractical. However, a good *estimate* of the likelihood ratio made from empirical training data will also give a detector near-optimal performance [16]. The key contribution of this research is tailoring the AdaBoost algorithm to do just that.

First, the AdaBoost detector must be trained as if it were to be used for classifying chips. Recall from Section 2.2.3 that the receiver can easily separate all the training observations into a positive group of chips x^+ and a negative group x^-, based on the chips' positions. The receiver then label each chip individually according to its group, giving the AdaBoost a set of $M = L \times N$ labeled training data $(\hat{x}_1, \hat{y_1}), \ldots, (\hat{x}_M, \hat{y_M})$. Using the WeakLearn routine described in Section 3.3, AdaBoost can sequentially generate T weak classifiers $\{h_t\}_{t=1}^T$, and compute their associated risks β_t.

The *key* to the AdaBoost detector is a modification of the final decision rule in AdaBoost. It is also one of the major contributions of this thesis. Instead of boosting all h_t's predictions to determine each chip's class, the final rule is modified to give an estimate of the likelihood ratio of the chip:

$$\hat{\lambda}_n = \frac{\sum_{t=1}^T \log(1/\beta_t) \times I[h_t(R[n]) = 1]}{\sum_{t=1}^T \log(1/\beta_t) \times I[h_t(R[n]) = -1]}. \tag{26}$$

where $I(\cdot)$ is the indicator function. To classify the whole chip sequence, one simply use this likelihood ratio estimate and proceed as the optimal detector would with the true likelihood ratio, i.e.

$$\hat{\Lambda} = \frac{\prod_{n \in n^+} \hat{\lambda}_n}{\prod_{n \in n^-} \hat{\lambda}_n} \tag{27}$$

$$\hat{\Lambda} \underset{\hat{m}=-1}{\overset{\hat{m}=+1}{\gtrless}} 1 \tag{28}$$

Notice that the quantity $\log(1/\beta_t)$ is the weight that AdaBoost assigns to the prediction by h_t, so the expression in the numerator of (26) is simply the total weighted votes in favor of the chip being $+1$; and the total weighted votes in favor of -1 is the quantity in the denominator. Therefore the right hand side of (26) is simply the

ratio of the total votes for $+1$ to that for -1. Intuitively, it is reasonable to use this quantity as an estimate of the likelihood ratio. The next section will show this with several experiments.

5 EXPERIMENTS AND DISCUSSION

5.1 AdaBoost Detector in Additive White Noise Channel

Experiments on the AdaBoost detector in additive white noise channels were conducted. In order to compare the results to that of Johnson *et al.* [1], the same experimental setup was used.

All digital symbols and chips came from antipodal binary classes ± 1. The length of the code $N = 256$ was chosen for most of the experiments. Four other detectors, along with the AdaBoost detector were simulated. They were the matched filter (Section 2.2.1), the sign detector (Section 2.2.2), the type-based detector (Section 2.2.3), and the clairvoyant detector, which was the optimal detector described in the beginning of Section 4.2.

In each experiment, 512 N-chip testing sequences were generated from 512 random symbols, using a code. A particular type of noise with certain power was simulated and added to those sequences. Then these sequences were given to the detectors; and all five detectors were asked to classify them into their corresponding symbols. Both the AdaBoost detector and the type-based detector were also given (32 bits) *labeled* sequences for training. These sequences are contaminated with the same noises given to the testing data. At the end of each experiment, the bit error rate of each detector was determined.

For each type of noise and for each noise power, the experiments were repeated 100 times; and the average bit error rate of each detector was recorded. This resulted in a plot of average bit error rate vs. the signal-to-noise ratio for each detector and

for each type of noise.

These plots are shown in Figure 8. It is obvious that the performance of both the AdaBoost detector and the type-based detector is close to that of the clairvoyant detector for all three noisy cases, while the fixed detectors are only useful for certain types of noises. It is interesting to see that even though the sign detector is overall the best fixed nonparametric detector for zero-median noises (such as the Laplacian noise), both the AdaBoost detector and the type-based detector are able to perform better at high SNR.

The AdaBoost detector also performed slightly better than the type-based detector in two of the three cases. However, the AdaBoost detector did not out-perform the type-based detector when fewer training data were used. Figure 7 is the result of a simulation with only 1/4 of the training data as that used for Figure 8a.[17] A

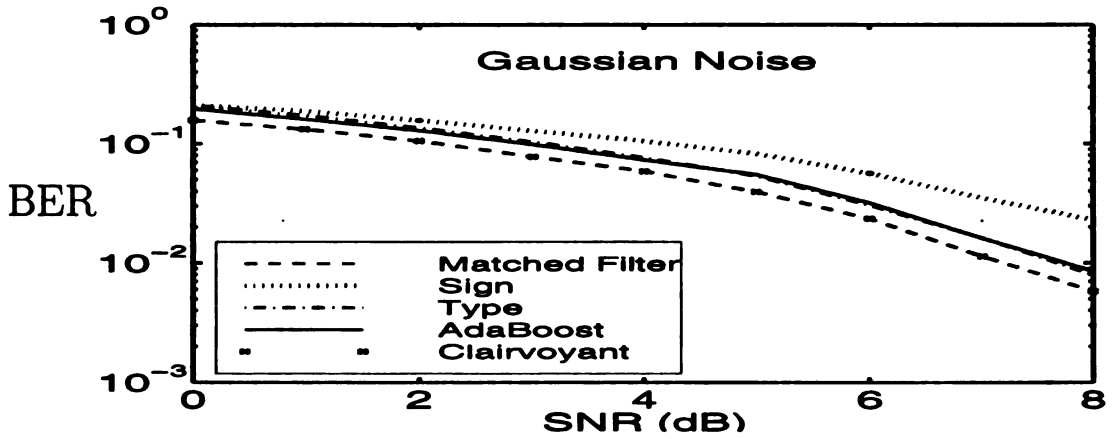


Figure 7: This is almost the same comparison test as the one shown in Figure 8a. However, $N = 64$ was used for the code lengths.

[17]There were still $L = 32$ training symbols, but the code length of $N = 64$ effectively reduced the size of the training set $M = L \times N$ (Section 4.2).

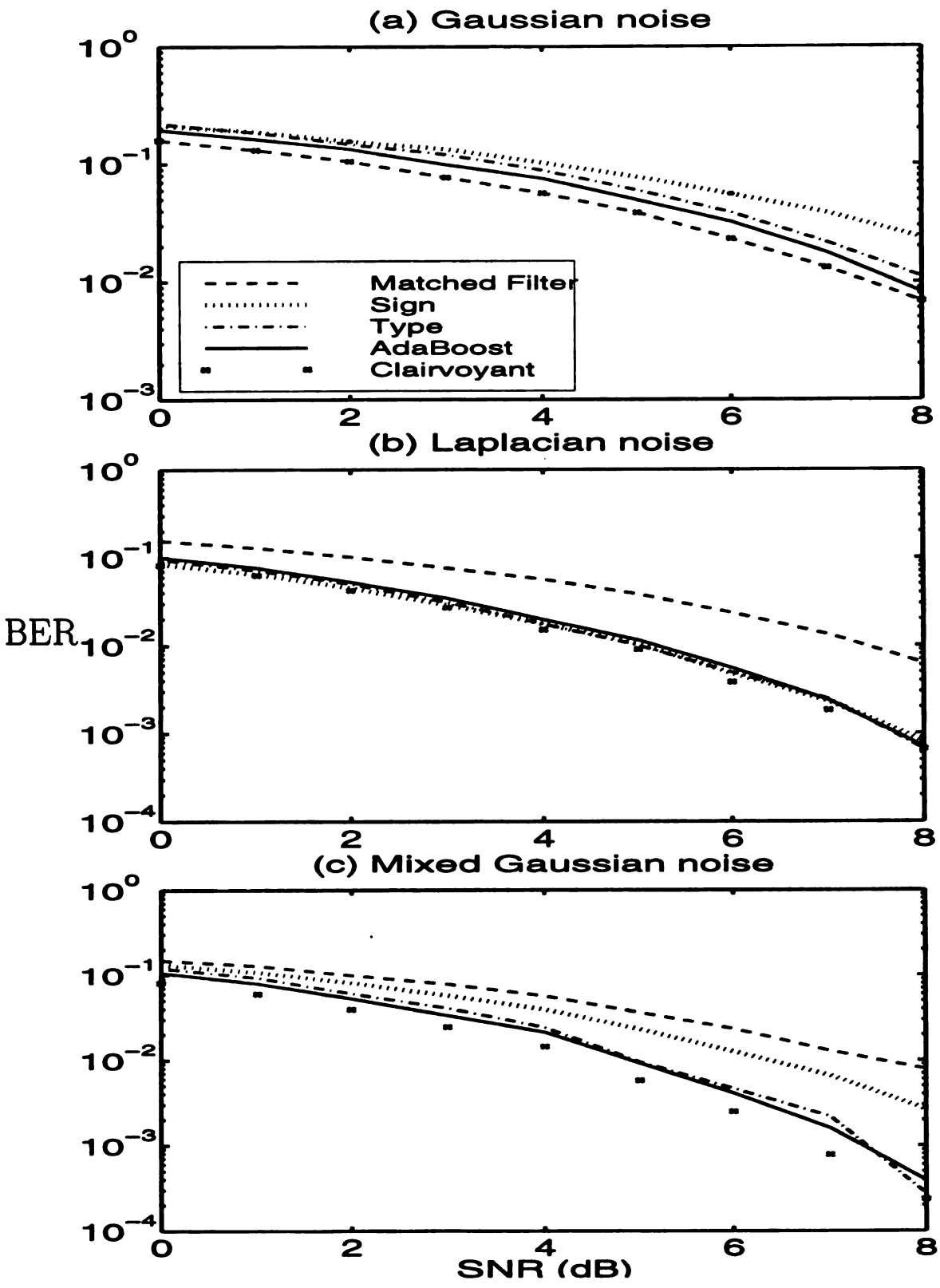


Figure 8: Experiments with the AdaBoost detector.

comparison between Figure 7 and Figure 8a indicates that the AdaBoost detector's performance increases faster as the training set becomes larger—apparently making better use of the training resource (relative to the type-based one). This is also expected since the second term in (21) says that the actual error should decay like $1/\sqrt{M}$; and M is the size of the training set.

5.2 Discussion

The experiments have clearly demonstrated the AdaBoost detector's ability to perform near-optimal (close to minimum probability of error) detection of CDMA signals in several types of noises, with only one assumption about the channel—that it is memoryless. It is also shown to outperform another nonparametric adaptive detector, the type-based detector, in some cases. It has one other advantage over the type-based detector. Recall that the first step of type-based detection is to quantize all received signals. In fact, the theory behind type-based detection is based on discrete sources (see Gutman [15]). Without any knowledge of the channel, this quantization step is quite arbitrary, and is difficult to optimize [2]. This is not a problem with the AdaBoost detector. However, this research also revealed other open questions about the AdaBoost detector and about boosting in general.

One fundamental question concerns the quality of the likelihood estimate in (26). Although it is a key contribution of this research, this method of estimating likelihood has not been theoretically proven. Nevertheless Schapire *et al.* have also considered the total votes on the two classes as the *confidence* that the algorithm has in them; and

have studied the *difference* between the votes (instead of the ratio). Interested readers are referred to the works of Schapire *et al.* [9, 10]. If this method (or its variation) can be proven optimal in any sense, it will be useful for many other applications besides classification.

Another question is how to choose T. A good value can be found by the *structural risk minimization* principle, which involves a trade-off between the training error and the bound of the generalization error above it (see Section 3.4). This theory can be found in Vapnik's work [14]. It should be pointed out that the choice of T does not require any knowledge of the channel. Also, there exist other methods for choosing T, such as *cross validation* [8]. For the experiments in Section 5.1, the value of T was arbitrarily chosen to be 16.

Finally, the AdaBoost detector is not designed to be used in channels with memories. Therefore other methods of incorporating boosting must be investigated. One possible solution starts with considering each sequence $\mathbf{x_l} = (x_l[1], \ldots, x_l[N])$ in the original training set as an N-dimensional vector observation $\mathbf{x_l} = [x_{l1}, \ldots, x_{lN}]$. Recall that the AdaBoost detector breaks all the sequences into two groups of chips (Section 4.2), discarding all the dependency information among the chips. This alternative preserves the full sequence structure of the training set $(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_L}, y_L)$, thus preserving the possible correlation information. The weak classifiers, in the case, can be N-dimensional directed hyperplanes. Fairly efficient algorithms[18] to find these hyperplanes are available [12], and can be employed in the WeakLearn subroutine.

[18]These algorithms certainly require more computation than those used to find a single directed threshold.

Once a number of these hyperplanes are found, they can be boosted to form complicated decision regions in the N-dimensional space. An unknown sequence $\mathbf{R}$, also treated as a vector $[R_1, \ldots, R_N]$, can then be classified.

Since the chip dependencies are manifested in the *a posteriori* joint probability distributions $p(R_1, \ldots, R_N | m = \pm 1)$, and since the boosting algorithm can make detection without any assumptions about these distributions, this method may be able to handle interchip interference. However, since N is usually large, a huge training set may be needed. If the training set is not large enough, a *weak* classifier (a high dimensional hyperplane) can probably make classifications with no errors[19] and render boosting ineffective. On the other hand, a large training set will waste too much of the channel's time and require a long time to train, making the method impractical. Therefore one might choose some compromise, working in a relatively small vector space, e.g. 2 or 3 dimension, dealing with length-2 or 3 chip sequences at a time. This may capture interchip interference effects without becoming too unwieldy.

[19]Depending on the WeakLearn, this hyperplane may be a good final classifier.

6 CONCLUSION

The modeling of a wireless communication channel is challenging due to unknown noises, interference, and the fact that the channel may be time-varying. It is therefore desirable to have adaptive detectors that automatically adjust to dynamic environments without making many assumptions.

This thesis first presented the idea of a new class of learning algorithms called boosting methods. Experiments with one of the algorithms, AdaBoost, showed its ability to learn the near-optimal decision boundaries in a classification setting. Then the AdaBoost *detector* was developed for a wireless communication system—the CDMA system. This detector was designed to extract CDMA signals from background noises. It assumes a memoryless channel, and adapts to it by learning the optimal decision rule with the help of a limited amount of training data. To examine the effectiveness of the AdaBoost detector, different types of noisy channels were simulated. The results showed that this detector could make near-optimal detection in all noisy situations.

Through the development, this research not only contributed to CDMA communications, but also demonstrated new potentials of the boosting algorithms. However, the design also revealed two major questions that merit future investigations. First, the main modification made to the AdaBoost algorithm, which uses the ratio between the total weighted votes on the two chip classes as estimate of the chip's likelihood ratio, was not rigorously examined. Therefore, even though the performance of the AdaBoost detector was excellent, solid theoretical analysis has yet to be done.

The other issue is how to handle channels with memories. The AdaBoost detector was not designed to accommodate dependencies among the chips in CDMA signals. An extended method of incorporating AdaBoost for CDMA detection was briefly described in Section 5.2. This method may solve the problem of interchip interference.

In conclusion, the theory of boosting is still relatively new. However, the boosting algorithms have already established themselves as benchmarks for many practical classification problems [13, 17]. The AdaBoost detector developed in this research was also effective in extracting CDMA signals from background noises without knowing the noise type. Therefore, boosting methods are very promising for adaptive detection in unknown channels.

Appendix

A VC DIMENSION

The Vapnik-Chervonenkis (VC) dimension is a quantity that indicates the complexity of a set of functions. Referring to Figure 4, if the function to be learned, G, is a binary classification rule, then the set of functions $\mathcal{F}$ that the learner uses is probably a set of indicator functions. To understand the meaning of VC dimension of such a set, consider k observations $(x_1, \ldots, x_k)$. In a binary classification problem, each of them comes from one of the two classes. Thus there are possibly 2^k different combinations. If the set $\mathcal{F}$ is very rich, then for every combination there may be some function $f \in \mathcal{F}$ that can separate the observations perfectly into their classes; and the set $\mathcal{F}$ is said to *shatter* k observations. On the other hand, when $\mathcal{F}$ is a small class, there may be certain combinations that cannot be separated by functions in $\mathcal{F}$. The VC dimension of $\mathcal{F}$ is simply the maximum number k of observations that the functions in $\mathcal{F}$ can classify, regardless of how they are combined (or the maximum number of observations that can be shattered by $\mathcal{F}$).

An example will make it clear. Consider the class $\mathcal{H}$ defined in (18). Each function is a directed threshold on a real line. Suppose there are two observations $a < b$. Then one can classify them with a directed threshold no matter which class each one comes from. For example, if a comes from -1 and b from $+1$, then a threshold between them directed towards a ($d = -1$) classifies them correctly. However, if there are three observations $a < b < c$, such classifiers can not shatter them. To see this, let a, c come from one class and b from another. This combination can not be separated correctly by a single directed threshold. Therefore the class $\mathcal{H}$ has a VC dimension

of $k_H = 2$.

To complete the above example, it takes a class of double thresholds with directions to shatter three observations. Obviously, this is a richer class; and it has a larger VC dimension as a consequence.

References

[1] D.H. Johnson, Y.K. Lee, O.E. Kelly, and J.L. Pistole, "Type-based detection for unknown channels," *International Conference on Acoustics, Speech and Signal Processing*, Atlanta, pages 2475–2478, 1996.

[2] L. Yue and D.H. Johnson, "Signal detection on wireless CDMA downlink," *Vehicular Technology Conference*, 1998.

[3] L. Yue and D.H. Johnson, "Type-based detection in macro-diversity reception for mobile radio signals," *International Conference on Acoustics, Speech and Signal Processing*, Munich, pages 4013–4016, 1997.

[4] John G. Proakis, *Digital Communications*, McGraw-Hill, 1989.

[5] Rodger E. Ziemer and Rodger L. Peterson, *Digital Communications and Spread Spectrum Systems*, Macmillan Publishing Company, 1985.

[6] Theodore S. Rappaport, *Wireless Communications: Principles and Practice*, Prentice Hall, 1996.

[7] R. Price and P.E. Green, Jr, "A communication technique for multipath channels," *IRE*, vol. 46, pages 555–570, March 1958.

[8] Yoav Freund and Robert E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Computational Learning Theory: Second European Conference, EuroCOLT '95*, pages 23–37. Springer-Verlag, 1995.

[9] Robert E. Schapire, Yoav Freund, Peter Bartlett and Wee Sun Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *Machine Learning: Proc. of the 14th International Conference*, pages 322–330, 1997.

[10] Robert E. Schapire and Yoram Singer, "Improved boosting algorithms using confidence-rated predictions," *Eleventh Annual Conference on Computational Learning Theory*, 1998.

[11] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, John Wiley and Sons, Inc., 1991.

[12] Pierre A. Devijver and Josef Kittler, *Pattern Recognition: A Statistical Approach*, Prentice Hall, London 1982.

[13] Vladimir N. Vapnik, *The Nature of Statistical Learning Theory*, Springer Verlag, New York 1995.

[14] Vladimir N. Vapnik, *Estimation of Dependences Based on Empirical Data*, Springer Verlag, New York 1982.

[15] Michael Gutman, "Asymptotically optimal classification for multiple tests with empirically observed statistics," *IEEE Trans. Info. Th.*, 35:401–408, 1989.

[16] J. Ziv, "On classification with empirically observed statistics and universal data compression," *IEEE Trans. Info. Th.*, 34:278–286, 1989.

[17] J.R. Quinlan, "Bagging, boosting, and C4.5," *Thirteenth National Conference on Artificial Intelligence*, pages 725–730, 1996.