18.413: Error-Correcting Codes Lab February 3, 2004

Lecture 1

Lecturer: Daniel A. Spielman

1.1 Introduction

The purpose of this course is to introduce you to some of the most exciting recent developments in the field of error-correcting codes. The reason that I can teach this material at an undergraduate level is that recent revolutions in the field have resulted in new coding techniques that can be taught without reference to much that happened between the late 60's and the early 90's. Before this revolution, error-correcting codes became highly algebraic, and a working knowledge of Algebraic Geometry became a prerequisite to understanding the the research that was current at the time. However, much of that research was divorced from practice, and most practical devices were built using theory that was not taught in these classes.

The revolution consisted of the introduction of iterative coding techniques powered by the Belief Propagation heuristic. The field of mathematics most closely connected with these codes is probability, and the most popular analyses combine computer experiment with theory. Moreover, the theory that has been developed has close connections to practice, and iterative coding techniques are already being used by companies today. The reasons that I can teach this material to you are that

- one only needs to know a little probability to understand these codes, and
- much understanding can be gained from experiment.

Wow, that discussion was a little abstract, and maybe meaningless to those of you who don't even know what coding theory is. It's time for me to tell you.

Error-correcting codes are used to facilitate communication in the presence of interference. Interference can take many forms. For example, static on a phone line, people screaming in the hall outside this classroom, or bits randomly flipping from 0 to 1 or *vice versa*. For now, let's consider the simplest model–that of flipping bits. We will assume that a *sender* is trying to send information to a *receiver* through a channel that has a probability p of flipping each bit. We will assume that a bit is received for every bit sent, but with probability p it is the wrong bit. This channel is called the Binary Symmetric Channel with crossover probability p. Assuming that it is important for the receiver to receive what the sender is sending (say a new operating system), the sender is going to have to do something fancier than just sending the bits. A natural thing to do would be to send each bit twice. That way, if the receiver gets a 01 or 10, she will know that one of the two bits was wrong. If we assume that there is also a channel going the other way, then she can request a retransmission of the bit that was in error. However, even if the receiver gets a 11, there is still a p^2 chance that the bit was a 0 (actually, the chance is a little higher, but we'll get to that in a moment). If we are sending something long, such as an operating system, then this smaller chance of error would be intolerable. So, you might decide to send each bit 3 times, or 4 times, or even more. However, much better solutions exist.

In his fundamental 1947 paper founding the field of information theory, Shannon proved that by only doubling the number of bits sent, one could tolerate a crossover probability of 0.11, with a probability of error exponentially small in the number of bits transmitted. For an operating system, that is a very small probability of error. Actually, Shannon proved many much stronger and more general theorems, and we will learn something about them later. The defect in Shannon's theorem is that while he proved that this was possible, the computational cost was enormous. The main achievement of the iterative revolution is the construction of codes that come exceedingly close to Shannon's bound and that can be encoded and decoded efficiently.

1.2 Mechanics

This class will be project-based. There will be three types of projects: small projects, regular projects, and final projects. The small and regular projects will be assigned to help you understand the concepts in the class, and will typically involve implementing coding systems and evaluating their performance. Your final project will be a research project. For your final project, you will probably attempt to reproduce the results of a recent paper. Your final project will result in two products: a final report and a class presentation.

1.3 A Little Probability

The most important fact from probability that we will use is Bayes' rule, which provides a formula for the probability that an even A occurs given that we know that B occurs:

$$\Pr[A|B] = \frac{\Pr[B \text{ and } A]}{\Pr[B]}$$

Recall that if A and B are independent, then the probability of A and B is the product of the probabilities of A and B, so the probability of A given B is just the probability of A.

We now use Bayes' rule to figure out how the sender should interpret a 11 received over the BSC_p. We compute

$$\Pr[\text{sender sent } 11|\text{receiver got } 11] = \frac{\Pr[\text{sender sent } 11 \text{ and receiver got } 11]}{\Pr[\text{receiver got } 11]}.$$

To evaluate this probability, we need to know the probability that the sender sent a 1. We will make the assumption that the sender was equally likely to send a 0 or a 1. One can spend a lot of time debating whether or not this is a reasonable assumption. However, if you know the prior probability that the sender sent a 1 was something other than 1/2, then you should plug that value into the formula. One may use information theory to justify our assumption: assuming that the sender has performed perfect compression on her message, this probability will be very close to 1/2. Shannon also introduced a mathematical theory of compression in his 1947 paper, but that is a topic for another course.

Now, back to that formula. Assuming that the sender sends a 1 with probability 1/2, and that each bit goes through the channel correctly with probability $1 - p$, and that the channel acts independently on each bit, we obtain

$$\Pr[\text{sender sent 11 and receiver got 11}] = (1/2)(1-p)^2.$$

On the other hand, there are two ways that the receiver can get 11: if the sender sent a 1 or if the sender sent a 0. In the latter case, we find:

$$\Pr[\text{sender sent 00 and receiver got 11}] = (1/2)(p)^2.$$

As these two events are disjoint, we may sum their probabilities to obtain:

$$\begin{aligned}\Pr[\text{receiver got 11}] &= \Pr[\text{sender sent 00 and receiver got 11}] + \Pr[\text{sender sent 11 and receiver got 11}] \\ &= (1/2)(p^2 + (1-p)^2).\end{aligned}$$

Thus, we may finally compute:

$$\Pr[\text{sender sent 11}|\text{receiver got 11}] = \frac{(1-p)^2}{p^2 + (1-p)^2.}$$