18.413: Error-Correcting Codes Lab February 19, 2004

Lecture 5

Lecturer: Daniel A. Spielman

5.1 Parity Continued

5.2 The Gaussian Distribution

One of the most natural distribution is the Gaussian distribution. A Gaussian random variable with mean μ and standard deviation σ has probability density function

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}.$$

For those who haven't learned about probability density functions before, this means that the probability that x lies between x_0 and x_1 is

$$\int_{x=x_0}^{x_1} p(x)\ dx.$$

You can verify that

$$\int_{x=-\infty}^{\infty} p(x)\ dx = 1.$$

The standard deviation tells you how far from the mean x is likely to lie. In particular, one can prove that

$$\mathrm{P}\left[x - \mu > k\sigma\right] < \frac{e^{-\frac{-k^2}{2}}}{\sqrt{2\pi}k},$$

and that this bound is very close to tight for $k \geq 2$. For $\sigma = 1$ and $\mu = 0$, this bound indicates that the probability that the Gaussian lies between -2 and 2 is at least 0.946, whereas the actual probability is about 0.956.

If g is a Gaussian random variable of mean 0 and standard deviation 1, then $\sigma g + \mu$ is a Gaussian random variable of mean μ and standard deviation σ. Often, you will see a Gaussian described by its variance. It's variance is the square of its standard deviation.

Gaussian random variables arise as the limit of binomial random variables. That is, if we let x_1, $\ldots$, x_n be indepent random variables uniformly distributed in $\{1, -1\}$, and let $X = \sum x_i/\sqrt{n}$, then as n grows large, X approaches the Gaussian distribution. We will now sketch a proof of this.

In particular, we consider the probability that $X = c\sqrt{n}$. From counting, we can determine that this is

$$\mathrm{P}\left[X = c\sqrt{n}\right] = 2^{-n}\binom{n}{n/2 - c\sqrt{n}/2}.$$

We will show that this is approximately equal to

$$\int_{g=c-1/\sqrt{n}}^{c+1/\sqrt{n}} p(g)\ dg \approx 2p(c)/\sqrt{n} = \sqrt{\frac{2}{\pi n}} e^{\frac{-c^2}{2}}.$$

Applying Stirling's formula, we find that

$$\begin{aligned}
2^{-n}\binom{n}{n/2-k} &= 2^{-n}\frac{n!}{(n/2-k)!(n/2+k)!} \\
&\approx 2^{-n}\left(\frac{\left(\frac{n}{e}\right)^n}{\left(\frac{n/2-k}{e}\right)^{n/2-k}\left(\frac{n/2+k}{e}\right)^{n/2+k}}\right)\left(\frac{\sqrt{2\pi n}}{\sqrt{2\pi(n/2-k)}\sqrt{2\pi(n/2+k)}}\right) \\
&= \left(\frac{(n)^n}{(n-2k)^{n/2-k}\,(n+2k)^{n/2+k}}\right)\left(\frac{\sqrt{2\pi n}}{\sqrt{2\pi(n/2-k)}\sqrt{2\pi(n/2+k)}}\right) \\
&= \left(\frac{(n)^{n-2k}}{(n^2-4k^2)^{n/2-k}}\right)\left(\frac{(n)^{2k}}{(n+2k)^{2k}}\right)\left(\frac{\sqrt{2\pi n}}{\sqrt{2\pi(n/2-k)}\sqrt{2\pi(n/2+k)}}\right)
\end{aligned}$$

We now evaluate each of the three terms in this product individually, substituting in $k = (c/2)\sqrt{n}$. First, we find

$$\begin{aligned}
\frac{(n)^{n-2k}}{(n^2-4k^2)^{n/2-k}} &= \frac{(n)^{n-c\sqrt{n}}}{(n^2-c^2 n)^{n/2-(c/2)\sqrt{n}}} \\
&= \left(\frac{1}{(1-c^2/n)}\right)^{n/2-(c/2)\sqrt{n}} \\
&\approx \left(1+\frac{c^2}{n}\right)^{n/2-(c/2)\sqrt{n}} \\
&\approx e^{c^2/2},
\end{aligned}$$

as $(1+1/k)^k \approx e$. To evalue the other term, we compute

$$\begin{aligned}
\frac{(n)^{2k}}{(n+2k)^{2k}} &= \frac{(n)^{c\sqrt{n}}}{(n+c\sqrt{n})^{c\sqrt{n}}} \\
&= \left(\frac{1}{1+c/\sqrt{n}}\right)^{c\sqrt{n}} \\
&\approx \left(1-c/\sqrt{n}\right)^{c\sqrt{n}} \\
&\approx e^{-c^2},
\end{aligned}$$

as $(1-1/k)^k \approx e^{-1}$. Finally, we find

$$\frac{\sqrt{2\pi n}}{\sqrt{2\pi(n/2-k)}\sqrt{2\pi(n/2+k)}} \approx \sqrt{\frac{2}{\pi n}},$$

for $k << n$.

Taking the product of all three terms, we get

$$e^{c^2/2}e^{-c^2}\sqrt{\frac{2}{\pi n}} = \sqrt{\frac{2}{\pi n}}e^{-c^2/2},$$

as desired.

5.3 The Gaussian and Erasure Channels

In this lecture, we introduce two new channels: the Erasure Channel and the Gaussian Channel. The Erasure Channel correpsonds to data loss, and can be used to model packet loss. On the other extreme, the Gaussian Channel corresponds to white Gaussian noise, and is closer to the analog world.

The Erasure Channel with erasure probability p maps

$$\begin{array}{rl} 0 \to 0 & \text{with probability } 1-p \\ 0 \to ? & \text{with probability } p \\ 1 \to ? & \text{with probability } p \\ 0 \to 1 & \text{with probability } 1-p \end{array}$$

That is, it erases each bit with probability p.

The Gaussian Channel is described in continuous terms. The channel can be specified by the standard deviation of the Gaussian random noise, σ. The input alphabet to the channel is $\mathbf{R}$, but we will restrict ourselves to the symbols 1 and -1. The output alphabet is $\mathbf{R}$. To obtain the output y from the input x, the Gaussian Channel chooses a Gaussian random variable with mean 0 and standard deviation σ, r, and outputs $y = x + r$. Note that you can obtain such a variable in Matlab from `sigma * randn (1)`. Under "Programming Tips" on the course web page, you can find out how to generate these in C and Java.

Of course, we need to figure out what the channel output y means. A naive approach is to say that if $y > 0$ then x was most likely 1, and if $y < 0$, then x was most likely -1. But, we want to be more precise. Our main formula tells us that

$$\mathrm{P}^{ext}[x = 1|y = b] = \frac{\mathrm{P}[y = b|x = 1]}{\mathrm{P}[y = b|x = 1] + \mathrm{P}[y = b|x = 1]}.$$

However, we may not apply this formula directly because $\mathrm{P}[y = b|x = 1] = 0$! We get the correct answer if we use the density functions instead of probabilities. That is, we replace $\mathrm{P}[y = b|x = 1]$ with $p(b-1)$ and $\mathrm{P}[y = b|x = -1]$ with $p(b+1)$ to obtain.

$$\mathrm{P}^{ext}[x = 1|y = b] = \frac{p(b-1)}{p(b-1) + p(b+1)} = \frac{e^{-(b-1)^2/2\sigma^2}}{e^{-(b-1)^2/2\sigma^2} + e^{-(b+1)^2/2\sigma^2}}.$$

For example, if $\sigma = 1$ and $b = 1$, we get

$$\frac{1}{1 + e^{-2}} = 0.88.$$

This is how we interpret the output of the Gaussian Channel.

To quickly compare these channels, we note that the capacity of the BEC_p is $1 - p$, which is exactly what you would get if you asked for a retransmit of every lost message. The capacity of the Gaussian channel with standard deviation σ is

$$\frac{1}{2}\log_2\left(1 + \frac{1}{\sigma^2}\right).$$

Warning: I've lied here a little bit: this is actually the capacity if we are allowed to use arbitrary input alphabets, subject to an average power constraint. The capacity is a little bit lower if we restrict ourselves to $\{-1, 1\}$ inputs. There is no nice closed form for the capacity when we restict to $\{-1, 1\}$ inputs, but you could compute it empiracally using the techniques from Small Project 1.

For comparison with the BSC, we note that we obtain capacity 1/2 when $\sigma = 1$. If we were to naively round the input—treating it as 1 if $y > 0$ and -1 otherwise—we would obtain a $BSC_{.1587}$. However, the capacity of the $BSC_{.16}$ is .366. To get capacity 1/2, we need the $BSC_{.1101}$. So, you loose a lot if you throw out the extra information provided by the channel.

Note that the Gaussian Channel on $\{1, -1\}$ inputs is a symmetric channel. That means that we can view it as a probability distribution over BSC channels, with a crossover probability $p \leq 1/2$ occuring with density

$$e^{-(p-1)^2/2\sigma^2} + e^{-(p+1)^2/2\sigma^2}.$$

This means that you all know a way to experimentally compute the capacity of this channel.

When describing codes, we have usually used the bits $\{0, 1\}$, whereas I'm using 1 and -1 over the Gaussian Channel. The standard translation is to identify binary 0 with Real 1 and binary 1 with Real -1.

5.4 The Parity Product Code

In Small Project 2, we will consider the following code. It will have 4 input bits, w_1, w_2, w_3, w_4. The first 4 bits of the codeword $x_1, \ldots, x_4$ will be set to these. The remaining bits will be set by the following rules:

$$\begin{aligned} x_5 &= x_1 \oplus x_2 \\ x_6 &= x_3 \oplus x_4 \\ x_7 &= x_1 \oplus x_3 \\ x_8 &= x_2 \oplus x_4 \\ x_9 &= x_7 \oplus x_8 \end{aligned}$$

We note that all these relations also imply

$$x_9 = x_5 \oplus x_6.$$

We usually understand this code by writing the bits in a matrix, like this

$$\begin{matrix} x_1 & x_2 & x_5 \\ x_3 & x_4 & x_6 \\ x_7 & x_8 & x_9, \end{matrix}$$

and observing that each row and column must have parity 0. This is why we call it a product code: each row and each column looks like a code (we'll see more of these later).

If we flip one bit in a codeword of this code, then the parity constraints on the row and colum containing the flipped bit will be violated, indicating where the error is. If two bits are flipped, you might be able to detect that an error has occured, but won't be able to figure out where it is. That is, if you only have bits and not and confidence information.

If in addition to bit you have information for each bit indicating how confident you are that the bit is 0 or 1 (that is, if you have the full information provided by the channel), then you can decode in one of the ways discussed in the last lecture: either maximum likelihood decoding to minimize word error or bit error. In Small Project 2, we will minimize the bit error rate, denoted (BER).

5.5 BER

I'll now define the bit error rate (BER) precisely. It depends upon the coding scheme, the channel, and the decoding scheme. Let $w_1, \ldots, w_k$ be the bits in the message to be sent. We assume that they are encoded as $x_1, \ldots, x_n$, and received by the decoder as $y_1, \ldots, y_n$. We then let $z_1, \ldots, z_k$ be the outputs of the decoder, which we now force to output a 0 or 1 for each bit. The bit error rate is then

$$\mathbf{E}\left[(1/k) \sum_i \mathrm{P}\left[w_i \neq z_i \right] \right].$$

Empircally, you can compute this by averaging

$$(1/k) \sum_i \left[w_i \neq z_i \right]$$

over many trials.

5.6 Heuristic Decoding of the Parity Product Code

I will now describe a heuristic decoding algorithm for the code we will examine on Small Project 2. This algorithm will be less accurate and take longer to run than the ideal algorithm. However, for larger codes the natural extension of this algorithm will be practical and the ideal algorithm will be impractical. For now, I will assume that we are just trying to compute x_1.

Here is the algorithm:

- compute $\mathrm{P}^{ext}\left[x_3 = 1 | y_4, y_6 \right]$.

- from the previous calculation, obtain $q_3 = \mathrm{P}^{post}[x_3 = 1|y_3, y_4, y_6]$.
- similarly, compute $q_7 = \mathrm{P}^{post}[x_7 = 1|y_7, y_8, y_9]$.
- Now, we will to compute $\mathrm{P}^{ext}[x_1 = 1|y_3, y_4, y_6, y_7, y_8, y_9]$. We do this by observing that $x_1 = x_3 \oplus x_7$, and use the parity formula to obtain the extrinsic probability that $x_1 = 1$ given our estimates of the probabiliites that x_3 and x_7 equal 1 obtained above, q_3 and q_7.
- We then compute $\mathrm{P}^{ext}[x_1 = 1|y_2, y_4, y_8, y_5, y_6, y_9]$, working by columns instead of rows.
- We then treat

  $$\mathrm{P}^{ext}[x_1 = 1|y_3, y_4, y_6, y_7, y_8, y_9],$$
  $$\mathrm{P}^{ext}[x_1 = 1|y_2, y_4, y_8, y_5, y_6, y_9],$$

  and

  $$\mathrm{P}^{ext}[x_1 = 1|y_1]$$

  as three independent estimators of x_1, and combine these estimations as discussed for the repition code.

This is a heuristic algorithm as the three estimators are not actually independent.

5.7 Confidence Intervals

Whenever we plot data, we should provide confidence intervals on the data in the plot. The goal of a confidence interval is to indicate the range of values that could reasonbly explain your output. It is standard to give a 95% confidence interval. That is, a range such that with probability 95% the actual value lies within the confidence interval. For this project, we will just be dealing with binary variables—either "yes" or "no", and estimating the probability of a "yes". In this case, obtaining confidence intervals is reasonably simple. Let's assume that we do n experiments and r of them come out "yes". Then, we estimate the probability of "yes" to be $\hat{p} = r/n$. For reasonably large n, and not unreasonably small or large p, if we draw the confidence interval from $\hat{p}e^{-2\sqrt{(1-\hat{p})/\hat{p}n}}$ to $\hat{p}e^{2\sqrt{(1-\hat{p})/\hat{p}n}}$, then with probability 95% the actual p should lie in this interval. If you have $r = 0$, then we set the bottom of the interval to 0 and the top to $1 - e(-2/n)$.

However, I should warn you that, while this is a very well accepted way of drawing the confidence intervals, it does not always get you 95% confidence. But, it takes strange values to get much below 90%. Yes, I am fudging here. I can give you papers on this topic if you would like to learn more.

For this class, I will be happy if you draw your confidence intervals using this rule.

You should expect your data to fluctuate quite a bit within the inner half of the confidence interval. As $pne^{\pm 2\sqrt{(1-p)/pn}} \approx pn \pm 2\sqrt{p(1-p)n}$, this means that you should only expect to get relative accuracy around $\sqrt{p(1-p)n}/pn \approx 1/\sqrt{pn}$. For example, if $p \approx 1/100$ and you run $n = 100,000,000$ experiments, you should only expect to get about 3 digits of accuracy.

5.8 How big should N be?

Let's say that we are trying to estimate the BER, but the BER is quite small. You might want to know how many trials you need to run to estimate the BER reasonably.

Rather than fixing in advance how many trials you will run, you can run until you see some fixed number of errors. For crude data, just to get general order of magnitude, 10 observations would be reasonable. If you want to get a digit of accuracy, then you need 100 observations. The formula given in the previous section will give you a resonable confidence interval.

5.9 Plotting in Matlab

To make many plots appear on the same set of axes, type `hold on` after you plot.

Here is an example of how to plot a confidence interval on some imaginary data. In this case, I've made the confidence interval go from `y (i) - sig` to `y (i) + sig`, just to show you how the graphics should look.

```
x = [1:10];
y = x.^(1.5);
plot(x,y)
hold on
plot(x,y,'o')
i = 4;
sig = 3;
plot([x(i),x(i)],[y(i)-sig,y(i)+sig],'+')
```