

IMPERIAL

COURSEWORK

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Statistical Information Theory

Author:

Amor Zhao (CID: 02019680)

Date: November 25, 2024

1 Accuracy of Hamming Code

In this section, we carry out an analytical calculation of the accuracy of a Hamming code with m parity bits through a completely noisy ($p = 0.5$) binary symmetric channel.

We will then show that the derivation agrees with numerical results for $m = 2, 3, 4$.

1.1 Analytical Calculation

For a Hamming code with m parity bits, the codeword length is $n = 2^m - 1$.

In a binary symmetric channel with $p = 0.5$, each bit has a 50% chance of being flipped. Since the Hamming decoder can correct a single-bit error but fails if two or more bits of errors occur, the probability of decoding a codeword correctly equals to the probability that at most one bit is flipped.

We can write the following equation:

$$P_{\text{success}} = P(0 \text{ errors}) + P(1 \text{ error}) = \binom{n}{0}(0.5)^n + \binom{n}{1}(0.5)^n$$

Simplifying:

$$P_{\text{success}} = (0.5)^n \cdot (1 + n)$$

We can then substitute the values of m and their corresponding n :

- $m = 2$: $n = 3$, $P_{\text{success}} = (0.5)^3 \cdot (1 + 3) = 0.5$
- $m = 3$: $n = 7$, $P_{\text{success}} = (0.5)^7 \cdot (1 + 7) = 0.0625$
- $m = 4$: $n = 15$, $P_{\text{success}} = (0.5)^{15} \cdot (1 + 15) \approx 0.0004883$

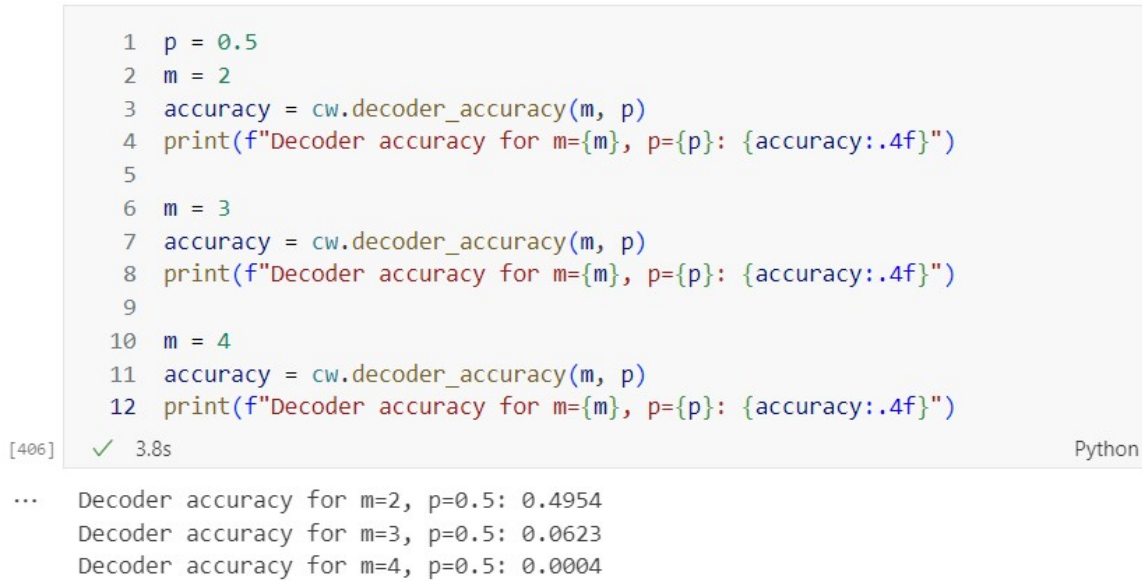
Intuitively, as m increases, the accuracy of decoding decreases due to the higher likelihood of multiple-bit errors in longer codewords, which is in line with the analytical solution.

1.2 Simulation Results

To validate the analytical results, we can run a large-scale test by calling the function `decoder_accuracy` in `coursework.py` over a range of parity bits $m = 2, 3, 4$. We

set each bit to have a probability of $p = 0.5$ to be flipped when transmitted in the channel.

Figure 1 below shows the simulation output of the Python script. In this specific case, variable `num_tests` (ie. the number of tests ran for each value of m) in the accuracy measuring function was set to 10,000.



```
1 p = 0.5
2 m = 2
3 accuracy = cw.decoder_accuracy(m, p)
4 print(f"Decoder accuracy for m={m}, p={p}: {accuracy:.4f}")
5
6 m = 3
7 accuracy = cw.decoder_accuracy(m, p)
8 print(f"Decoder accuracy for m={m}, p={p}: {accuracy:.4f}")
9
10 m = 4
11 accuracy = cw.decoder_accuracy(m, p)
12 print(f"Decoder accuracy for m={m}, p={p}: {accuracy:.4f}")
```

[406] ✓ 3.8s Python

... Decoder accuracy for m=2, p=0.5: 0.4954
Decoder accuracy for m=3, p=0.5: 0.0623
Decoder accuracy for m=4, p=0.5: 0.0004

Figure 1: Simulation results for Hamming code accuracy with $m = 2, 3, 4$.

We can see that the measured decoder accuracy is very close to the theoretical probabilities from our calculation. Due to the randomness of the simulation, there exists an overall difference lower than 5% between our calculated probability and the measured accuracy.

We can verify that the analytical solution is correct.

2 Effect of Noise

In the second section, we will visualise the effect of the noise level p on the Hamming decoder accuracy, with noise range $0 < p < 0.5$ and parity bit numbers $m = 2, 3, 4$.

Finally, we study these plots and find the relationship between the noise level and the decoder accuracy.

2.1 Plots

The following plots in Figure 2 illustrate how the decoder accuracy decreases as the noise probability p increases, for different values of m .

We can validate the plots by confirming that when $p = 0.0$, the decoding accuracies are all 1.0 because there is no error. As the noise level increases, the accuracies decrease with a rate as predicted in the last section. At $p = 0.5$, the accuracy value in each case is approximately the same as our theoretical calculation.

The comparison plot shows that increasing m (and consequently the codeword length n) leads to a sharper decline in accuracy. This result is consistent with our theoretical expectation that larger codewords are less robust to multiple bit errors in noisy channels.

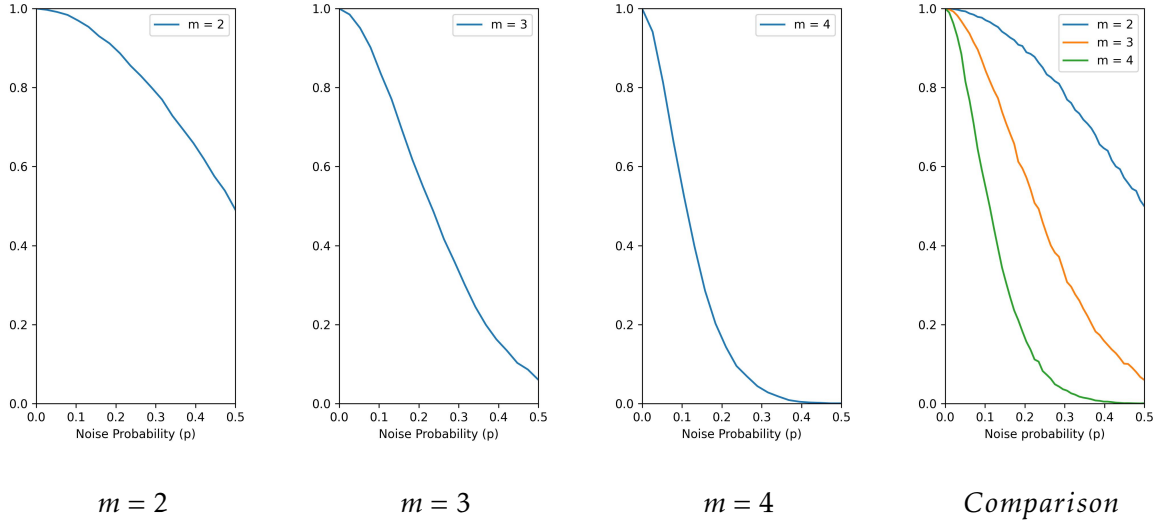


Figure 2: Simulation results for Hamming code accuracy with $m = 2, 3, 4$ at varying noise levels, and a comparison between them.

2.2 Conclusion

We found analytically and by simulation that, the Hamming decoder accuracy declines as p increases due to the rising likelihood of multiple-bit errors in the codeword. This is particularly significant for larger values of m , where the total codeword length n increases, exposing the code to a greater cumulative noise effect.

This conclusion helps us understanding the trade-offs in designing error-correcting codes. For instance, when the noise level in the communication channel is high, selecting a smaller number of parity bits may help minimize the impact of channel noise on the decoder's performance.

Appendix

A Code for Plotting the Effect of Noise

In addition to the functions in `coursework.py`, the following code was used to plot the graphs in section 2.

```
1 import matplotlib.pyplot as plt
2 p_values = np.linspace(0, 0.5, 50)
3 m_values = [2, 3, 4]
4 x_lim = (0, 0.5)
5 y_lim = (0, 1)
6
7 for m in m_values:
8     accuracy = []
9     for p in p_values:
10         accuracy.append(cw.decoder_accuracy(m, p))
11
12     fig, ax = plt.subplots(figsize=(3, 6))
13     ax.plot(p_values, accuracy, label=f"m = {m}")
14     ax.set_xlabel("Noise Probability (p)")
15     ax.set_ylabel("Decoder Accuracy")
16     ax.legend()
17     ax.set_xlim(x_lim)
18     ax.set_ylim(y_lim)
19
20     plt.savefig(f'2c_2_1_{m}.jpg', format='jpg', dpi=300)
21     plt.show()
```

```
1 p_values = np.linspace(0, 0.5, 50)
2 fig, ax = plt.subplots(figsize=(3, 6))
3
4 for m in [2, 3, 4]:
5     accuracy = []
6     for p in p_values:
7         accuracy.append(cw.decoder_accuracy(m, p))
8     ax.plot(p_values, accuracy, label=f"m = {m}")
9
10 ax.set_xlabel("Noise probability (p)")
11 ax.set_ylabel("Decoder accuracy")
12 ax.legend()
13 ax.set_xlim(0, 0.5)
14 ax.set_ylim(0, 1)
15 plt.savefig(f'2c_2_1_total.jpg', format='jpg', dpi=300)
16 plt.show()
```