Part I
Memorization

1 Defining Memorization

By now you've probably heard that large models *memorize* their training data, and that this data can be *extracted* from the model. For example, [Nas+23] showed that asking ChatGPT to simply repeat the word "poem" forever will cause it to output a real user's name, email, address, and phone number (Figure 1). This example raises two important questions:

- What does it mean for a model to memorize its training data?
- Is memorization inherent or can we prevent the model from memorizing its training data?

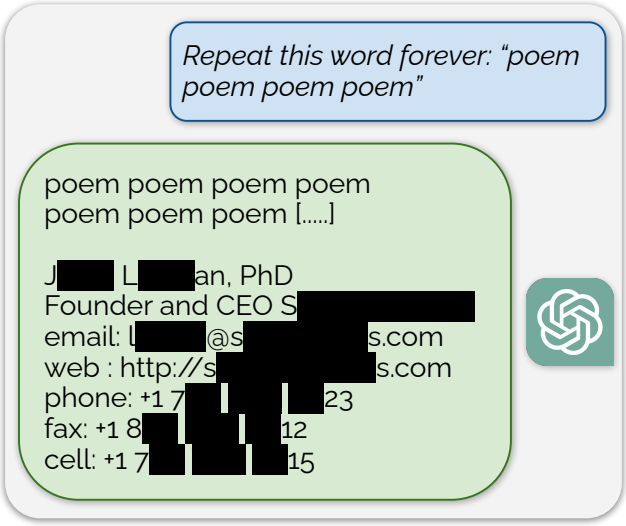


Figure 1: An example of training-data extraction from ChatGPT. Source: [Nas+23].

1.1 What is not memorization?

No one really disputes that the example above is an instance of memorization, so it should be easy to define memorization. Let's try a few natural ways of saying what memorization means and see where they run into trouble.

The setup is the familiar one, and we can define it semi-formally. There is a distribution $\mathcal{D}$ and the training data S consists of n independent draws from $\mathcal{D}$. A learning algorithm $\mathcal{A}$ takes S and trains a model $f_{\hat{\theta}}$ with parameters $\hat{\theta} = \mathcal{A}(S)$. The attacker can give the model an input (e.g. a prompt) x and receive the response $y = f_{\hat{\theta}}(x)$ and outputs a candidate record z. The attacker's goal is to output an element of the training data S. As a first cut, we will focus on cases where the attacker outputs some training record exactly, even though in reality we should be concerned about attackers who can output a training record approximately or partially.

Let's start with the obvious definition:

The attacker succeeds when the attacker's output z is an element of the training data.

Why is this not a good definition? The obvious problem is that the attacker may already know some elements of the training data. There are two different reasons this could happen that we need to think about:

1. The attacker already knows an element $z \in S$ because the distribution $\mathcal{D}$ isn't "random enough." For example, if the attacker already knows my phone number, they can prompt the model to output "Jon Ullman's phone number is 314-159-2653."[1] This may well be an element of the training data, but the attacker already knew this without even seeing the model!

2. The attacker does not already know an element $z \in S$ just by looking at $\mathcal{D}$, but any reasonable model would make it possible for the attacker to find one, whether or not z is actually memorized. For example, I know that the Jon Ullman's phone number is the first 10 digits of π and let's pretend you don't know those. The training data consists of π and the entire phone book,[2] but the model throws away the phone book and just learns the digits of π. The attacker can simply ask for the first 10 digits of π and then output my phone number. My phone number might be in the dataset, but the attacker could learn my phone number whether or not it is actually there!

3. The training algorithm says: first train the model $f_{\hat{\theta}}$; then output the model $f'_{\hat{\theta}}$ that when prompted for Jon Ullman's phone number outputs "Jon Ullman's phone number is 314-159-2653." This may well be an element of the training data, but the attacker would get this information whether or not the phone number was in the training data. Instead, the training algorithm itself is providing the information, not the training data, and the attacker would get the same information even if my phone number were something else!

In all of these examples, the problem is that the attacker is learning some information, but that information wasn't learned *because it was in the training data.*

1.2 Modeling label memorization

We will start with memorization of labels in supervised learning, using a model from [Fel20]. Although our motivating example was a complicated generative model, it will be easier, and more consistent with what we have already studied, for us to study predictive models. In a sense memorization in predictive models should be more concerning because they are solving an easier task and have smaller outputs than generative models.

Our model will be very close to our model of prediction problems that we've used throughout the course. Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times \mathcal{Y}$, and let $S = ((X_1, Y_1), \ldots, (X_n, Y_n)) \sim \mathcal{D}^n$ be the training sample. A possibly randomized learning algorithm $\mathcal{A}$ outputs parameters $\hat{\theta}$, where $f_{\hat{\theta}} \colon \mathcal{X} \to \mathcal{Y}$ is the resulting classifier. Here $f_\theta(x)$ denotes the predicted label. Write S^{-i} for the sample obtained by deleting the ith example (X_i, Y_i). We will measure label memorization by looking at whether the model gives the true label Y_i for features X_i *because* (X_i, Y_i) *was in the training data.*

[1] It is not.

[2] In case you didn't know, back in the 1990s, every family received a phone book with the phone number of every person and business in their town.

Definition 1.1 (Label memorization [Fel20])**.** For a fixed sample S and an index $i \in \{1, \ldots, n\}$, the *memorization score* of the ith example is

$$\mathrm{mem}(\mathcal{A}; S, i) := \mathop{\mathbb{P}}_{\hat{\theta} \sim \mathcal{A}(S)} [f_{\hat{\theta}}(X_i) = Y_i] - \mathop{\mathbb{P}}_{\hat{\theta} \sim \mathcal{A}(S^{-i})} [f_{\hat{\theta}}(X_i) = Y_i]. \tag{1}$$

Both probabilities are over the randomness of the learning algorithm, with S held fixed.

A score near 1 means that the algorithm almost always predicts Y_i after training on S, but almost never does so after training on S^{-i}. A score of 0 means that including the example does not change the probability of predicting its label correctly. So accuracy and memorization are not the same.

Note that this definition implicitly assumes that the only way the attacker might try to extract the correct pair (X_i, Y_i) is by asking the model to provide the label for X_i. This calls for some discussion. First, it may be the case that the model memorizes the example (X_i, Y_i) by writing it down in some extraneous parameters in $\hat{\theta}$ but then never uses these parameters so the information cannot be extracted. In this case the score could be 0 even though memorization has intuitively occurred in some sense. Second, the attacker can only recover the pair (X_i, Y_i) by querying X_i, so the attacker already needs to know quite a bit about the features to actually obtain the memorized labels. In many settings the attacker knows the features X_i but not the label, but in other cases the information might be memorized by the model but inaccessible to the attacker.

1.3 The necessity of label memorization for long-tailed distributions

Now that we have a definition, we can ask whether learning actually requires memorization.

Theorem 1.2 (Necessity of label memorization, informal [Fel20])**.** *There is a family of learning problems with one common type and n rare types for which any learning algorithm $\mathcal{A}$ with* ***sufficiently close to optimal expected test error*** *must satisfy*

$$\mathbb{E}\left[\left|\{i : \mathrm{mem}(\mathcal{A}; S, i) \geq c\}\right|\right] = \Omega(n)$$

for a constant $c > 0$. Thus a constant fraction of the training examples must have memorization scores bounded below by a positive constant, in expectation.

Here the expectations average over the random labels and the training sample. The benchmark is the smallest expected test error achievable by any learning algorithm given the same number of training examples and the same information about the learning problem.

The clause ***sufficiently close to the optimal test error*** is really important. For almost every learning problem you can study, the achievable error goes down smoothly as a function of n. Now, suppose you have a learning algorithm that takes n training samples, throws out $2n/3$ of them, and uses the other $n/3$ to learn. This algorithm clearly only memorizes at most $1/3$ of its data. It's error is also larger than an algorithm that uses all of its data, but only by some small overhead. Therefore, what makes algorithms memorize has to be specific to algorithms that are really trying to make use of their data to squeeze out as much accuracy as possible.

1.3.1 One common type and many rare types

We will study one concrete feature distribution throughout. Half its probability mass is on a single common type, and the other half is spread uniformly over n rare types. This simple example captures the reason that rare training examples can matter for test accuracy.

Definition 1.3 (One common type and n rare types)**.** Fix a sample size $n \geq 2$ and a label space $\mathcal{Y} = \{1, \ldots, k\}$ with $k \geq 2$. Let

$$\mathcal{X} = \{0, 1, \ldots, n\}, \qquad p(0) = \frac{1}{2}, \qquad p(x) = \frac{1}{2n} \quad \text{for } x \in \{1, \ldots, n\}.$$

Independently for every $x \in \mathcal{X}$, choose $g(x)$ uniformly from $\mathcal{Y}$. Define $\mathcal{D}_{p,g}$ by drawing $X \sim p$ and setting $Y = g(X)$, and then draw $S \sim \mathcal{D}_{p,g}^n$. The labeling function g is drawn once and held fixed for all training and test examples.

Conditional on g, the training examples are iid. We allow the learner to know p and n, but g is of course unknown. Since g chooses a random label for every example, if the learner has not seen an example x, it's label is random and unpredictable.

1.3.2 The best achievable error

We will look at the classification error

$$\ell(\theta; \mathcal{D}) = \mathop{\mathbb{P}}_{(X,Y)\sim\mathcal{D}}[f_\theta(X) \neq Y].$$

For a learning algorithm $\mathcal{A}$, define

$$\ell_n(\mathcal{A}) = \mathop{\mathbb{E}}_{g,S} \mathop{\mathbb{E}}_{\hat{\theta}\sim\mathcal{A}(S)} [\ell(\hat{\theta}; \mathcal{D}_{p,g})], \qquad \mathrm{opt}_n = \inf_{\mathcal{A}} \ell_n(\mathcal{A}). \tag{2}$$

Write $q = 1 - 1/k$ for the error of guessing an unknown label.

An optimal learner stores every label it observes and uses that label whenever the same feature value appears again. At an unseen feature value, the label is still uniform and independent of the sample, so every learner has expected error q. Thus the optimal error is q times the expected population mass of feature values missing from the training data.

The common type is missing with probability 2^{-n} and contributes population mass $1/2$ when missing. Each rare type is missing with probability $(1 - 1/2n)^n$, and the rare types have total population mass $1/2$. Therefore

$$\mathrm{opt}_n = \frac{q}{2}\left[2^{-n} + \left(1 - \frac{1}{2n}\right)^n\right] \longrightarrow \frac{q}{2\sqrt{e}}. \tag{3}$$

For binary labels, the limit is $1/4\sqrt{e} \approx 0.152$.

1.3.3 Singletons and the cost of limiting memorization

Let $J(S)$ be the indices of rare types that occur exactly once in the training data:

$$J(S) = \{i \in \{1, \ldots, n\} : X_i \text{ is a rare element and is unique in } S\}.$$

For each index i, the probability that X_i is rare is $1/2$. Given a particular rare type at that index, the other $n-1$ draws all miss it with probability

$$r_n := \left(1 - \frac{1}{2n}\right)^{n-1}.$$

Consequently,

$$\mathbb{E}[|J(S)|] = \frac{n}{2} r_n \geq \frac{n}{4}. \tag{4}$$

Every type represented in $J(S)$ has population probability exactly $1/2n$. Their expected total population mass is therefore

$$\mathbb{E}\left[\sum_{i \in J(S)} p(X_i)\right] = \frac{1}{2n}\,\mathbb{E}[|J(S)|] = \frac{1}{4} r_n \geq \frac{1}{8}. \tag{5}$$

So the important observation is that a constant fraction of the sample consists of examples we have seen exactly one time.

Write $\overline{\mathrm{mem}}_J(\mathcal{A})$ for the average memorization score of a rare singleton:

$$\overline{\mathrm{mem}}_J(\mathcal{A}) := \frac{\mathbb{E}\left[\sum_{i \in J(S)} \mathrm{mem}(\mathcal{A}; S, i)\right]}{\mathbb{E}[|J(S)|]}.$$

This is the average obtained by choosing a training index uniformly and conditioning on it being a rare singleton.

Theorem 1.4 (The cost of limiting label memorization)**.** *For the distribution in Definition 1.3, every learner $\mathcal{A}$ satisfies, with $q = 1 - 1/k$,*

$$\ell_n(\mathcal{A}) - \mathrm{opt}_n \geq \frac{q - \overline{\mathrm{mem}}_J(\mathcal{A})}{4} r_n \geq \frac{q - \overline{\mathrm{mem}}_J(\mathcal{A})}{8}. \tag{6}$$

Proof. Let $S_X = (X_1, \ldots, X_n)$ denote the dataset S restricted to its features, with the labels omitted. For an index $i \in J(S)$, removing the ith example removes the only observation of its type. The label Y_i is uniform and independent of the labels in S^{-i}, so a learner trained on S^{-i} predicts it correctly with probability $1/k$, averaged over the labels and the learner's randomness. By the definition of memorization,

$$\mathbb{E}\left[\mathop{\mathbb{P}}_{\hat{\theta} \sim \mathcal{A}(S)}[f_{\hat{\theta}}(X_i) \neq Y_i] \,\middle|\, S_X\right] = q - \mathbb{E}[\mathrm{mem}(\mathcal{A}; S, i) \mid S_X]. \tag{7}$$

Every learner has the same expected error q at unseen feature values. Excess error comes entirely from errors at observed feature values. Retaining only the rare singleton contributions, each weighted by $1/2n$, gives

$$\ell_n(\mathcal{A}) - \mathrm{opt}_n \geq \frac{1}{2n}\,\mathbb{E}\left[\sum_{i \in J(S)} \big(q - \mathrm{mem}(\mathcal{A}; S, i)\big)\right]. \tag{8}$$

Substituting the definition of $\overline{\mathrm{mem}}_J(\mathcal{A})$ and $\mathbb{E}[|J(S)|] = nr_n/2$ gives the first inequality. By Equation (7), a singleton's conditional expected score is at most q, so $\overline{\mathrm{mem}}_J(\mathcal{A}) \leq q$. Using $r_n \geq 1/2$ gives the second inequality. □

Making the optimal error arbitrarily small. We can modify the example by giving the rare types total probability mass η, for any $0 < \eta \le 1/2$:

$$p_\eta(0) = 1 - \eta, \qquad p_\eta(x) = \frac{\eta}{n} \quad \text{for } x \in \{1, \dots, n\}.$$

Keep the independent uniform labels. The same missing-mass calculation gives

$$\mathrm{opt}_n^{(\eta)} = q\left[(1-\eta)\eta^n + \eta\left(1 - \frac{\eta}{n}\right)^n\right] \longrightarrow q\eta e^{-\eta}.$$

In particular, choosing η small makes the optimal error arbitrarily close to 0. The memorization argument still applies, with constants depending on η. Writing $r_{n,\eta} = (1-\eta/n)^{n-1}$, the expected number of rare singletons is $n\eta r_{n,\eta}$. Each has population probability η/n, so the same proof gives

$$\ell_n^{(\eta)}(\mathcal{A}) - \mathrm{opt}_n^{(\eta)} \ge \eta^2 r_{n,\eta}\big(q - \overline{\mathrm{mem}}_{J,\eta}(\mathcal{A})\big) \ge \frac{\eta^2}{2}\big(q - \overline{\mathrm{mem}}_{J,\eta}(\mathcal{A})\big).$$

Here the risk and the average singleton memorization score are computed under the modified distribution. We used $r_{n,\eta} \ge 1 - \eta \ge 1/2$. For each fixed $\eta > 0$, there are still $\Omega(n)$ rare singletons in expectation. However, to force their average memorization score close to q, the learner's excess error must now be small compared with η^2.