

Generalization in Machine Learning

Sam, Joe, and Arthur

2020 Summer

1 The Cake Problem

1.1 A Tempting Pattern

We'll start out with a fun puzzle that's not directly related to machine learning but whose math will later help us. Imagine n people are seated around a large disk-shaped cake. Each pair of people will use a two-person handsaw to cut the cake. For example, the picture below shows a group of 4 people; there are 6 pairs of people, so the group makes 6 cuts total. When 4 people encircle the cake, it ends up cut into 8 pieces. We wonder: how many pieces arise when a different number of people sit around the cake?

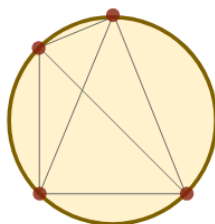


Figure 1: With $n = 4$ people, we make $\binom{n}{2} = 6$ cuts, which gives 8 pieces in total (4 outside and 4 inside).

Here are some drawings for possibilities ranging from $n = 1$ people to $n = 6$ people. Try counting the pieces and guessing a pattern! Can you find a formula for how many pieces arise when there are n people?

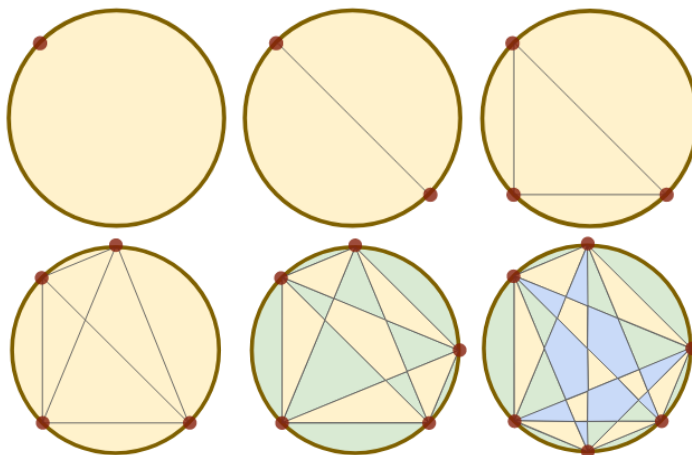


Figure 2: Here, we color some of the cake pieces just to make them easier to see and to count. The $n = 1$ case doesn't have any cuts. The $n = 5$ case has 5 outside green pieces, 5 inside green pieces, 5 yellow triangles — so far this is three fives, which makes fifteen — and what's left is 1 yellow center piece. The $n = 6$ case has 6 outside green pieces, 6 inside green pieces, 6 outer yellow triangles, 6 blue triangles — so far this is four sixes, which makes twenty four — and what's left are the 7 central blue and yellow shapes.

Well, it seems that with $n = 1, 2, 3, 4$ people, there are $p(n) = 1, 2, 4, 8$ pieces. (Here, p stands for “pieces”, and we'll use this way of writing just to save time). It seems that $p(n)$ doubles for each next n , meaning that p looks like powers of two. In symbols, our guess is: $p(n) \stackrel{?}{=} 2^{n-1}$.

Let's check this guess. Does it continue to $n = 5$? We expect the next power of two, namely $8 + 8 = 16$. And yes: $p(5)$ really is 16! How about $n = 6$? We expect $16 + 16 = 32$. But — uh oh! — it seems that there are only 31 pieces. So the pattern breaks.

1.2 An Explanation

Let's now figure out $p(n)$ for any n ; along the way, we'll see why the powers-of-two pattern seemed to work until $n = 6$. There are two steps: we'll relate cake to constellations and then relate constellations to counting.

We'll use Euler's polyhedron formula. This formula says that if we connect a bunch of dots by edges to form some enclosed regions, then the numbers of dots, edges, and regions are related:

$$\text{Regions} - \text{Edges} + \text{Dots} = 1$$

For example, the constellation that we build up in stages below has 3 regions (one pentagon and two triangles), 13 edges, and 11 dots. And $3 - 13 + 11 = 1$, just like the formula says. Why is the formula true?

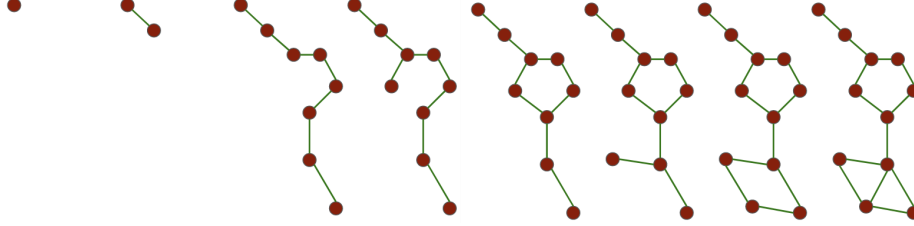


Figure 3: Steps to build an example constellation (right) starting from a single dot (left). By the way, this method only helps us build constellations that don't have crossing edges, and each two of whose dots are connected by a path of edges. Euler's formula only applies to constellations that follow these rules.

Well, we can build up our constellation starting from a single dot. The single dot follows the formula (since there are no regions and no edges, and $0 - 0 + 1 = 1$). And each step of building preserves the formula: **either** we connect a new dot to an old dot (so both $-Edges$ and $+Dots$ change by one, meaning that the total stays the same) **or** we connect two old dots to create a new region (so both $Regions$ and $-Edges$ change by one, meaning that the total stays the same). This logic proves Euler's formula.

To wrap up, let's think of a cake as a constellation as shown below. In addition to n outer dots, there are $\binom{n}{4}$ inner dots, because from each inner dot emanate 4 rays pointing toward 4 outer dots. By similar logic, the number of edges is $n + 2\binom{n}{2}/2 + 4\binom{n}{4}/2$, since there are n outer arcs (green), $2\binom{n}{2}$ straight-half edges (blue) between outer dots, and 4 half-edges (orange) emanating from each of $\binom{n}{4}$ inner dots.

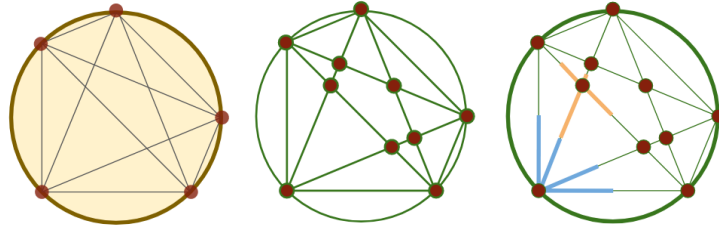


Figure 4: We can think of a cake (left) as a constellation (middle) by adding inner dots. We can count the half-edges of the constellation (right) by binning them into three groups: the outer arcs (green), the straight half-edges between outer dots (blue), and the straight half-edges emanating from inner dots (orange).

Putting all the pieces together, we find that $\text{Regions} = 1 + \text{Edges} - \text{Dots} = 1 + \binom{n}{2} + \binom{n}{4}$. We can simplify this by using the facts that $1 = \binom{n-1}{0}$ and $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$:



$$\text{Regions} = \binom{n-1}{0} + \binom{n-1}{1} + \binom{n-1}{2} + \binom{n-1}{3} + \binom{n-1}{4}$$

Since $\binom{n-1}{0} + \dots + \binom{n-1}{k} = 2^{n-1}$ as long as $k \leq n-1$, this looks like powers of two as long as $n \leq 5$. So we have solved the mystery. With n people, the number $p(n)$ of pieces equals the number of subsets of people with at most 4 members. Ain't that cool?

2 Finite Hypothesis Classes

2.1 What is Learning?

Today, we'll analyze machines that learn to classify images into two possible buckets such as Cow and Dog. Most of our discussion extends to the case where there are more than two buckets and to other more complicated situations. But we'll just focus on the simple case.

To use math, we need to precisely set up what we mean by “machine learning”. Here's how we'll do it in this class: we posit a set \mathcal{X} of all possible images, a set $\mathcal{Y} = \{\text{Cow}, \text{Dog}\}$ of buckets, and a probability distribution \mathcal{D} over $\mathcal{X} \times \mathcal{Y}$. The distribution \mathcal{D} models the world by telling us which pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ are more likely and which are less likely. For instance: we might have , , $\dots \in \mathcal{X}$, and \mathcal{D} might say that (img, Cow) is more likely to occur in nature than (img, Dog) , which is more likely to occur than (img, Dog) , which in turn is more likely than to occur in nature than (img, Dog) .

So far, we've given names $\mathcal{X}, \mathcal{Y}, \mathcal{D}$ to aspects of the world. What's cool is that we don't have to know \mathcal{D} in order for our mathematical theory to work! We gave it a name just so we can reason about it. While \mathcal{D} models the world, we'd like our machine to learn by pondering finitely many sample pairs drawn from \mathcal{D} . It's like learning which way a die is loaded by throwing it a hundred times and tabulating the results. So we posit a set \mathcal{H} of functions from \mathcal{X} to \mathcal{Y} . Each function is one possible way of classifying images, and \mathcal{H} is a set of many such functions. We call these functions **hypotheses**, since we don't know which ones are good. As an example, \mathcal{H} might have just three elements:

$$\begin{aligned} \mathcal{H} = \{ & \text{always say Cow,} \\ & \text{say Cow if } x \text{ is brown on average; otherwise, say Dog,} \\ & \text{say Cow if } x \in \{\text{img}, \text{img}, \text{img}\}; \text{ otherwise, say Dog} \} \end{aligned}$$

In practice, \mathcal{H} will actually be something like a set of neural networks, which is effectively an infinite set.

A **learner** is then a rule \mathcal{L} that assigns — to each length- N sequence $\mathcal{S} = ((x_i, y_i) : 0 \leq i < N)$ of samples — a hypothesis $\mathcal{L}(\mathcal{S}) \in \mathcal{H}$. A learner \mathcal{L} is good when $\mathcal{L}(\mathcal{S})$ classifies images accurately. We mean not just the images (in \mathcal{S}) that we learned from; we want \mathcal{L} to accurately classify new images freshly drawn from \mathcal{D} ! More formally, we define the **out-error** and **in-error** of a hypothesis $f \in \mathcal{H}$ as

$$\mathcal{E}_{\text{out}}(f) = \mathbb{P}_{(x,y) \sim \mathcal{D}} [f(x) \neq y] \quad \mathcal{E}_{\text{in}, \mathcal{S}}(f) = \mathbb{P}_{(x,y) \sim \mathcal{S}} [f(x) \neq y]$$

Likewise, the out- and in-errors of a learner are $\mathcal{E}_{\text{out}}(\mathcal{L}) = \mathbb{E}_{\mathcal{S} \sim \mathcal{D}} [\mathcal{E}_{\text{out}}(\mathcal{L}(\mathcal{S}))]$. We want $\mathcal{E}_{\text{out}}(f)$ to be low, but we can only directly calculate $\mathcal{E}_{\text{in}}(f)$, since we have access to the samples \mathcal{S} but not the full distribution \mathcal{D} .

Based on the intuition that $\mathcal{E}_{\text{out}}(f)$ and $\mathcal{E}_{\text{in}, \mathcal{S}}(f)$ look similar, we propose a special learner \mathcal{L} that works like this: compute the in-error $\mathcal{E}_{\text{in}, \mathcal{S}}(f)$ for each $f \in \mathcal{H}$, and (breaking ties arbitrarily) settle on whichever f has the smallest in-error. This heuristic is called **empirical risk minimization**, and its variants dominate modern machine learning.

But does this special learner \mathcal{L} actually work? In particular, if $\mathcal{E}_{\text{in}, \mathcal{S}}(\mathcal{L})$ is small, will $\mathcal{E}_{\text{out}}(\mathcal{L})$ also be small? This is the question of **generalization**.

Here's an example where the answer is yes. Imagine that there's one hypothesis $f \in \mathcal{H}$ (so $|\mathcal{H}| = 1$) and $N = 10000$ datapoints. Then $\mathcal{E}_{\text{out}}(\mathcal{L}) = \mathcal{E}_{\text{in}}(f)$, and by the law of large numbers $\mathcal{E}_{\text{in}, \mathcal{S}}(f)$ will with high probability be very close to $\mathcal{E}_{\text{out}}(f)$.

Here's an example where the answer is no. Imagine that every possible function is in \mathcal{H} (so \mathcal{H} is very infinite), but only $N = 3$ datapoints. For any fixed \mathcal{S} , many hypotheses will have the same $\mathcal{E}_{\text{in}, \mathcal{S}}$ (pigeonhole). Suppose \mathcal{L} breaks ties in favor of memorizing; that is, \mathcal{L} always settles on the hypothesis,

$$f(x) = \text{say Cow if } (x, \text{Cow}) \in \mathcal{S}; \text{ otherwise, say Dog}$$

Then f may have $\mathcal{E}_{\text{in}, \mathcal{S}}(f) = 0$, but unless the exact images in \mathcal{S} were extremely common, or unless most mammals are Dogs, f will have $\mathcal{E}_{\text{out}}(f) \approx 1$.

Intuitively, when \mathcal{H} contains many hypothesis, generalization is difficult, and when the number N of datapoints is large, generalization is easy. How do these conflicting forces balance? Let's use math to find out.

2.2 Is Learning Possible?

3 Infinite Hypothesis Classes

3.1

3.2