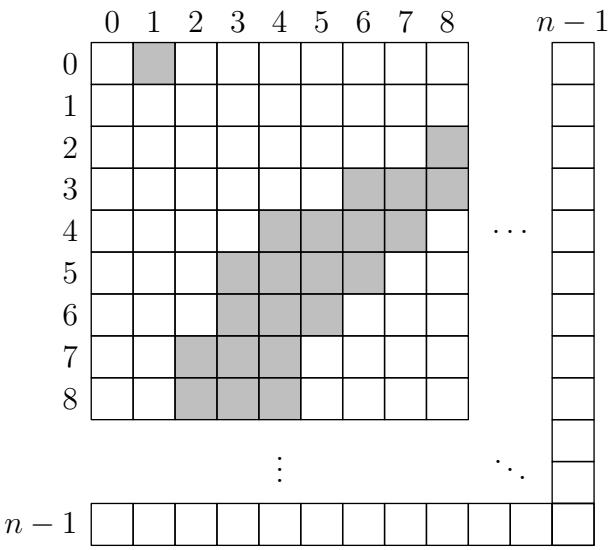
1 Week 1: Training a Model

1.1 Defining Notation

First, here's some probability notation that we'll be using throughout the assignment:

- $P(A \cap B)$ is the probability that A **and** B both occur.
- $P(A \mid B)$ is the probability that A occurs, **given** that B has occurred.

Let $F_{i,j}$ denote the value of the pixel in row i and column j (zero-indexed); $F_{i,j}$ will either be 0 (if the pixel is unshaded) or 1 (if the pixel is shaded). For example, in the $n \times n$ image below, $F_{0,0} = 0$, $F_{0,1} = 1$, and so on, and the bottom right pixel is $F_{n-1,n-1} = 0$.



1.2 Setting up the Problem

Let's say that we wanted to classify the image above. In other words, we want to assign a **class** c, where $c \in \{0, 1, \ldots, 9\}$, to the image. Our prediction will be based on n^2 pieces of evidence (each individual pixel is a piece of evidence).

In order to classify this image, we can start by asking the question: *what is the probability that this image is a 0, given these n^2 pieces of evidence?* In other words, what is the probability that this image is a 0, **given** that $F_{0,0} = 0$, **and** $F_{0,1} = 1$, **and** $\ldots$, **and** $F_{n-1,n-1} = 0$? This can be mathematically expressed as:

$$P(\text{class} = 0 \mid (F_{0,0} = 0) \cap (F_{0,1} = 1) \cap \cdots \cap (F_{n-1,n-1} = 0)).$$

We can also ask the same question for class $= 1, 2, \ldots, 9$. Then, out of the 10 possible classes, whichever has the highest probability will be our prediction!

But how do we calculate this probability? Keep reading to find out!

1.3 Bayes' Theorem

First, we have to go on a tangent and discuss some more math. If we want to find $P(A \cap B)$, we can approach the calculation in two steps. In order for A **and** B occur, we first need A to occur, and then we need B to occur **given** that A has already occurred. Thus, $P(A \cap B) = P(A)P(B \mid A)$.

On the other hand, we can slightly change our approach. In order for A **and** B occur, we first need B to occur, and then we need A to occur **given** that B has already occurred. Thus, $P(A \cap B) = P(B)P(A \mid B)$.

Now, combining these two results yields

$$P(B)P(A \mid B) = P(A \cap B) = P(A)P(B \mid A) \implies \boxed{P(A \mid B) = \frac{P(A)P(B \mid A)}{P(B)}}.$$

The boxed equation is Bayes' theorem.

If you would like a deeper/more nuanced understanding of Bayes' theorem, we'd recommend checking out this video by 3Blue1Brown, or the Wikipedia page.

1.4 Application to Digit Classification

For ease of notation, we'll define a shorthand for the very long expression from earlier:

$$\textsc{AllPixelValues} := (F_{0,0} = 0) \cap (F_{0,1} = 1) \cap \cdots \cap (F_{n-1,n-1} = 0).$$

As a reminder, the probability that we're interested in is $P(\text{class} = 0 \mid \textsc{AllPixelValues})$. Bayes' theorem tells us that

$$P(\text{class} = 0 \mid \textsc{AllPixelValues}) = \frac{P(\text{class} = 0) \times P(\textsc{AllPixelValues} \mid \text{class} = 0)}{P(\textsc{AllPixelValues})}.$$

1.5 Independence Assumption

Two events are *independent* if the occurrence of one event does not affect the probability of the other event. If events A and B are independent, then we have $P(A \cap B) = P(A)P(B)$.

Dice example: Imagine that we roll two fair six-sided dice. Then,

$$P(\text{Dice 1 is even} \cap \text{Dice 2 is a six}) = P(\text{Dice 1 is even}) \times P(\text{Dice 2 is a six}) = \frac{1}{2} \times \frac{1}{6},$$

because these two events are independent. However,

$$P(\text{Dice 1 is even} \cap \text{Dice 1 is a six}) \neq P(\text{Dice 1 is even}) \times P(\text{Dice 1 is a six}),$$

because 'Dice 1 is even' is not independent from 'Dice 1 is a six'.

Let's take a closer look at the term $P(\textsc{AllPixelValues} \mid \text{class} = 0)$. This is the probability that this exact arrangement of pixel values (shaded or unshaded) occurs, **given** that the image is a 0.

Recall that AllPixelValues actually represents the **combination/intersection of n^2 events**. (The value of each pixel is an event; for example, $F_{0,0} = 0$ is an event.) Of course, these n^2 events aren't independent. If a pixel is shaded, then its neighboring pixels are more likely to be shaded; in other words, the occurrence of one event *does* affect the probability of the other events.

However, in Naive Bayes, we make the (incorrect) *assumption* that these n^2 pixels are independent—this is the "naive" part of Naive Bayes. In other words, given that the image is a 0, we assume that the probability of pixel (i, j) being shaded, $P(F_{i,j} = 1 \mid \text{class} = 0)$, is independent to the probability of another distinct pixel (p, q) being shaded, $P(F_{p,q} = 1 \mid \text{class} = 0)$. The reason we make this assumption is that it allows us to apply the product rule from above:

$$P((F_{i,j} = 1) \cap (F_{p,q} = 1) \mid \text{class} = 0) \approx P(F_{i,j} = 1 \mid \text{class} = 0) \times P(F_{p,q} = 1 \mid \text{class} = 0).$$

If we extend this to all n^2 events, then we get

$$\begin{aligned}
&P(\text{AllPixelValues} \mid \text{class} = 0) \\
&\quad = P((F_{0,0} = 0) \cap (F_{0,1} = 1) \cap \cdots \cap (F_{n-1,n-1} = 0) \mid \text{class} = 0) \\
&\quad \approx P(F_{0,0} = 0 \mid \text{class} = 0) \times P(F_{0,1} = 1 \mid \text{class} = 0) \times \cdots \times P(F_{n-1,n-1} = 0 \mid \text{class} = 0).
\end{aligned}$$

1.6 How to Calculate These Probabilities

Recall the quantity that we're trying to compute:

$$P(\text{class} = 0 \mid \text{AllPixelValues}) = \frac{P(\text{class} = 0) \times P(\text{AllPixelValues} \mid \text{class} = 0)}{P(\text{AllPixelValues})}.$$

First, let's focus on the $P(\text{AllPixelValues} \mid \text{class} = 0)$ term. In order to compute the naive approximation of this term (which was discussed in the previous section), we need to be able to calculate *feature probabilities* such as $P(F_{0,1} = 1 \mid \text{class} = 0)$.

This is where the training data comes into play! As a reminder, $P(F_{0,1} = 1 \mid \text{class} = 0)$ is the probability that pixel $(0, 1)$ is shaded, **given** that the image belongs to class 0. To compute this, we can just count the number of times this occurs in the training data:

$$P(F_{0,1} = 1 \mid \text{class} = 0) = \frac{\text{\# of images belonging to class 0 where pixel } (0,1) \text{ is shaded}}{\text{Total \# of images belonging to class 0 in the training data}}.$$

Similarly, we have

$$P(F_{0,0} = 0 \mid \text{class} = 0) = \frac{\text{\# of images belonging to class 0 where pixel } (0,0) \text{ is } \textbf{un}\text{shaded}}{\text{Total \# of images belonging to class 0 in the training data}}.$$

More generally,

$$P(F_{i,j} = f \mid \text{class} = c) = \frac{\text{\# of images belonging to class } c \text{ where } F_{i,j} = f}{\text{Total \# of images belonging to class } c \text{ in the training data}}.$$

The numerator also contains another term: $P(\text{class} = 0)$. This term is called a *prior*; it represents the probability that an image belongs to class 0 *before* we take into account any of the image's pixel values. We can also calculate this term by counting the training data:

$$P(\text{class} = 0) = \frac{\text{\# of images belonging to class 0 in the training data}}{\text{Total \# of training images}}.$$

In other words, this is answering the question, *if I grab a random image from the training set, what's the probability that I'll get a 0?* More generally,

$$P(\text{class} = c) = \frac{\text{\# of images belonging to class } c \text{ in the training data}}{\text{Total \# of training images}}.$$

That takes care of everything in the numerator. For now, we don't need to worry about the $P(\text{AllPixelValues})$ term in the denominator; we'll discuss that more in week 2.

1.7 Laplace Smoothing

There's a small snag that we might run into when using the formulas above. Notice that in the original image, the shaded pixel at $(0, 1)$ seems to be an accidental anomaly. Maybe it was a stray bit of ink from the writer's pen, or a tiny speck of dust on the scanner. Either way, in almost all images, we'd expect pixels near the border to remain unshaded. Thus, it's very likely that the training data does not contain any images where pixel $(0, 1)$ is shaded.

If this occurs, then we'd have $P(F_{0,1} = 1 \mid \text{class} = 0) = 0$ (because the numerator of the fraction would be 0). Since we're multiplying everything together, this would cause the final probability to become zero, regardless of how probable the other pixels' shadings are. In other words, it's possible for a small anomaly to override all of the other evidence.

To avoid this problem, we will use a technique called *Laplace smoothing*. This technique works by adding a small positive value k to the numerator, and kV to the denominator. Here, V is the number of possible values our feature can take on, so in our case: 2 (shaded or unshaded). The higher the value of k, the stronger the smoothing is. The formula to calculate the smoothed probability would look like this:

$$P(F_{i,j} = f \mid \text{class} = c) = \frac{k + \#\text{ of images belonging to class } c \text{ where } F_{i,j} = f}{2k + \text{Total } \#\text{ of images belonging to class } c}.$$

We will also apply this to our prior class probabilities, but in this case, we have $V = 10$ because there are 10 possible classes:

$$P(\text{class} = c) = \frac{k + \#\text{ of images belonging to class } c}{10k + \text{Total } \#\text{ of training images}}.$$

For now, you can just set $k = 1$. In week 2, you can experiment with different values of k and see how it affects your classification accuracy.

Side Note: Before our model processes any training images, we'll have $P(F_{i,j} = f \mid \text{class} = c) = \frac{k}{2k} = \frac{1}{2}$ and $P(\text{class} = c) = \frac{k}{10k} = \frac{1}{10}$, which should make sense intuitively. (Essentially, when the model has no data yet, it assumes pixels are equally likely to be shaded or unshaded, and it assumes that all 10 classes are equally likely.) However, the more training data our model sees, the less effect Laplace smoothing has.

What you need to do: For week 1, you need to create a model that is able to process the training data and provide the following probabilities (with Laplace smoothing applied):

- $P(F_{i,j} = f \mid \text{class} = c)$ for all $i, j \in \{0, 1, \ldots, n-1\}$, $c \in \{0, 1, \ldots, 9\}$, and $f \in \{0, 1\}$
- $P(\text{class} = c)$ for all $c \in \{0, 1, \ldots, 9\}$

2 Week 2: Creating a Classifier

2.1 Ignoring the Denominator

As you may remember from last week, we still need to consider the denominator: $P(\textsc{AllPixelValues})$, also known as the *evidence distribution*. The good news is, we actually don't need to consider it! Recall that our goal was just to *compare* the probabilities for different classes and pick the class with the highest probability. For our purposes, calculating $P(\textsc{AllPixelValues})$ is entirely irrelevant, because this probability will be the same regardless of which class we're analyzing. Since the denominator is constant across all ten classes, we can ignore the denominator and the winner (i.e. the most likely class) will still be the same.

Therefore, what we actually care about is the following proportion ($\propto$ represents proportionality):

$$\begin{aligned} P(\text{class} = 0 \mid \textsc{AllPixelValues}) &\propto P(\text{class} = 0) \times P(\textsc{AllPixelValues} \mid \text{class} = 0) \\ &\approx P(\text{class} = 0) \\ &\quad \times P(F_{0,0} = 0 \mid \text{class} = 0) \times P(F_{0,1} = 1 \mid \text{class} = 0) \times \cdots \\ &\quad \times P(F_{n-1,n-1} = 0 \mid \text{class} = 0). \end{aligned}$$

2.2 Avoiding Underflow by Using Logarithms

If we attempt to evaluate the expression above, we'll run into another snag. These probabilities are all going to be less than 1.0, which means if we multiply them all together, they're going to get really small really fast, and we'll run into the problem of Arithmetic Underflow.

Again, to get around this snag, we can take advantage of the fact that we only care about *comparing* the probabilities for different classes. If we take the logarithm of all ten probabilities that we're comparing, the winner will still be the same as before. This allows us to avoid arithmetic underflow, because a floating-point double would not be able to accurately store a number as small as 10^{-1000}, but it can easily handle numbers in the vicinity of $\log\left(10^{-1000}\right) = -1000$.

At this point, the thing we're computing is no longer a probability, so we'll call it a *likelihood score*. Recall that $\log(ab) = \log a + \log b$; applying this property gives us:

$$\begin{aligned}\textsc{LikelihoodScore}(0) = \log(P(\text{class} = 0)) \\ + \log(P(F_{0,0} = 0 \mid \text{class} = 0)) + \log(P(F_{0,1} = 1 \mid \text{class} = 0)) + \cdots \\ + \log(P(F_{n-1,n-1} = 0 \mid \text{class} = 0)).\end{aligned}$$

2.3 Make a Prediction!

We're in the home stretch now! The last step is to determine which class has the highest likelihood score, and classify the image as that class.

What you need to do: Given a trained model (which you should have from last week) and a new image that doesn't belong to the training dataset, you should be able to calculate $\textsc{LikelihoodScore}(c)$ for all $c \in \{0, 1, \ldots, 9\}$. Then, you should be able to determine which class c has the highest likelihood score, and classify the image as belonging to class c.