Automatic skill tracking

HOW TO ESTIMATE PROFICIENCY OF A STUDENT ON A VARIETY OF SKILLS – A MATHEMATICAL ANALYSIS

HILDO BIJL

TABLE OF CONTENTS

1. TRACKING A SINGLE SKILL

In this first chapter we open up with a relatively simple case: suppose that there is only one skill, for example "addition". We want to estimate the chance that a student performs this skill correctly based on various exercises. How could we go about doing this?

1.1. THE PRIOR DISTRIBUTION

Let's call the chance that the student does the skill correctly x. Our goal is to estimate x. The key realization while doing this is to note that x is a random variable. As a result, it has a distribution, expressed through the *Probability Density Function* (PDF) $f_x(x)$. We need to define a *prior* distribution $f_x(x)$, which is the initial distribution for x. As we gather data we shall continuously update this distribution.

Lacking any data, it would make sense to assume a so-called flat prior,

$$f_x(x) = \begin{cases} 1 \text{ if } 0 \leq x \leq 1, \\ 0 \text{ otherwise.} \end{cases}$$

That is, x is just as likely to be 0.2 as it is 0.6 or 0.9. Other priors are of course also possible, but we will stick with this one, also because it makes many of the mathematics a bit simpler.

1.2. INCORPORATING ONE BASIC OBSERVATION

Next, we add information. Suppose that the student did the first exercise successfully. What does this do to our distribution $f_x(x)$?

We write this new piece of data (the observation) as D. According to Bayes' law, the *posterior distribution* (the distribution after incorporating data) is

$$f_x(x|D) = \frac{p(D|x)f_x(x)}{p(D)}.$$

In this expression we have three terms on the right-hand side.

- $p(D|x)$ is the chance that our observations happened as they did, given the exact value of x. If the student did his exercise successfully, then $p(D|x) = x$. After all, this is per definition the chance that a student does an exercise successfully. If the exercise was failed, then $p(D|x) = 1 - x$.
- $f_x(x)$ is the prior distribution of x, which we defined earlier.
- $p(D)$ is the chance, barring any information, that our observation D would take place. It is a constant (i.e., does not depend on x) and is responsible for making sure that the integral over $f_x(x|D)$ equals one, as should be the case for any PDF. This term can therefore generally be found through

$$p(D) = \int_{-\infty}^{\infty} p(D|x)f_x(x)\,dx = \int_0^1 p(D|x)f_x(x)\,dx.$$

Note that we can narrow the integration bounds because $f_x(x)$ only takes values between 0 and 1 (inclusive).

We can solve everything for our specific case: the student did one exercise correctly. In this case $p(D|x) = x$, $p(D) = 1/2$ (just solve the above integral) and we end up with

$$f_x(x|D) = \begin{cases} 2x \text{ if } 0 \leq x \leq 1, \\ 0 \text{ otherwise.} \end{cases}$$

Note: from here on out we will consider it implicit that $f_x(x)$ only takes values between 0 and 1 (inclusive) and hence omit the conditional notation. We will simply write $f_x(x) = 2x$.

We could also consider the case where the student failed his first exercise. In that case we find $f_x(x) = 2(1-x)$.

1.3. INCORPORATING ANOTHER OBSERVATION

What happens when the student does the first two exercises correctly? There are two ways in which we can incorporate this data. We could write D_1 as the observation of the first exercise, D_2 as the observation for the second exercise, and recursively find

$$f_x(x|D_2, D_1) = \frac{p(D_2|x, D_1) f_x(x|D_1)}{p(D_2|D_1)}.$$

Note that $p(D_2|x, D_1)$ can be simplified to $p(D_2|x)$. After all, if x is known exactly, knowing D_1 is irrelevant. This allows us to find the distribution of x after two observations from the distribution of x after one observation.

It is also possible to take both observations into account together. In this case we write $D = (D_1, D_2)$ and have

$$f_x(x|D) = \frac{p(D|x) f_x(x)}{p(D)}.$$

Both methods work equally well; it only depends on whether we want to calculate the distribution of x from scratch or update is as we go. Later on we will see that updating as we go is usually more convenient in practice.

For our particular example case, we get the following outcomes:

- If the student has two successful exercises, we have $f_x(x|D) = 3x^2$.
- If the student has a successful and a failed exercise, we have $f_x(x|D) = 6x(1-x)$.
- If the student fails both exercises, we have $f_x(x|D) = 3(1-x)^2$.

It is recommended to draw out these plots for yourself to see that they make intuitive sense.

1.4. INCORPORATING MULTIPLE OBSERVATIONS

Next, consider the case where the student has done n exercises, out of which n_s were successful and n_f were failed. In this case we have $p(D|x) = x^{n_s}(1-x)^{n_f}$. We should find $p(D)$ through

$$p(D) = \int_0^1 p(D|x) f_x(x)\, dx = \int_0^1 x^{n_s}(1-x)^{n_f}\, dx = \frac{n_s!\, n_f!}{(n+1)!} = \frac{1}{(n+1)\binom{n}{n_s}}.$$

Note the above integration result. We will use it more often. It follows that the posterior distribution is

$$f_x(x|D) = \frac{p(D|x) f_x(x)}{p(D)} = (n+1)\binom{n}{n_s} x^{n_s}(1-x)^{n_f}.$$

This is the general posterior distribution of x given multiple measurements. Later on we will use this function a lot, as a basis function, so we already give this its own symbol. We hereby define

$$g_{i,n}(x) = (n+1)\binom{n}{i}x^i(1-x)^{n-i}.$$

So n is the number of exercises that have been done and i is the number that the student did successfully.

1.5. PROPERTIES OF THE POSTERIOR DISTRIBUTION

The posterior distribution just described has some interesting properties. First of all we can examine the mean $E[x]$, often shortened as $\bar{x}$. This can be found through

$$\bar{x} = E[x] = \int_0^1 x g_{n_s,n}(x)\,dx = (n+1)\binom{n}{n_s}\int_0^1 x^{n_s+1}(1-x)^{n_f}\,dx = \frac{(n+1)\binom{n}{n_s}}{(n+2)\binom{n+1}{n_s+1}}.$$

This can be simplified as

$$\bar{x} = E[x] = \frac{(n_s+1)}{(n_s+1)+(n_f+1)} = \frac{(n_s+1)}{(n+2)}.$$

So effectively it's like the student gets a free success and a free failure at the start, and the mean is calculated by calculating the percentage of successes, including these two "free" exercises. This somewhat "conservative" estimate of the success rate is a direct result of the prior we have chosen for x.

In addition to the mean (the first moment) we can also calculate the higher moments of $f_x(x|D)$. These are found through

$$E[x^m] = \int_0^1 x^m g_{n_s,n}(x)\,dx = (n+1)\binom{n}{n_s}\int_0^1 x^{n_s+m}(1-x)^{n_f}\,dx = \frac{(n+1)\binom{n}{n_s}}{(n+m+1)\binom{n+m}{n_s+m}}.$$

This can be further simplified as

$$E[x^m] = \frac{\frac{(n+1)!}{n_s!\,n_{f!}}}{\frac{(n+m+1)!}{(n_s+m)!\,n_f!}} = \frac{(n+1)!}{(n+m+1)!}\frac{(n_s+m)!}{n_s!} = \frac{(n_s+m)\cdot\ldots\cdot(n_s+1)}{(n+m+1)\cdot\ldots\cdot(n+2)}.$$

This allows us to calculate any moment of this posterior distribution.

2. ADDING PRACTICE EFFECTS

In reality the probability x that a student does an exercise successfully is not a constant. It changes, both due to practice (he hopefully gets better) and due to passage of time (maybe the student forgets it, or the skills sink in better). How can we incorporate this?

2.1. DEFINING THE LINK BETWEEN SUBSEQUENT EXERCISES

We will start by examining subsequent exercises: how are these linked? The crucial realization is that we now do not have a single random variable x anymore to estimate. Instead, we have the chance x_1 that the first exercise is done successfully, x_2 that the second exercise is done successfully, and so forth. Previously all these probabilities were equal (there was a 100% correlation) but now that is not the case. Instead, we must define a joint prior over all these variables.

To figure out what prior would be sensible, consider the hypothetical case where (through some magical means) we have precisely determined x_1. For instance, we know exactly that our student has an 80% chance of solving the first exercise. (So $x_1 = 0.8$.) What would the distribution of x_2 then look like? Surely x_2 is likely to be close around 80% too. Sure, 60% and 100% might be possible too, but 0% is very unlikely.

To find a mathematical formula, we should do a thought experiment. We grab a coin which has x_1 percent chance to land on head (success) and $1 - x_1$ percent chance to land on tails (failure). We flip this coin n times and mark how many successes n_s and failures n_f we get. The chance that we actually get n_s successes and n_f failures is

$$p(D|x_1) = \binom{n}{n_s} x_1^{n_s}(1 - x_1)^{n-n_s}.$$

We could subsequently use this (fictional) "data" to approximate the distribution of x_2. This gives a posterior distribution

$$f_{x_2}(x_2|D) = (n+1)\binom{n}{n_s} x_2^{n_s}(1 - x_2)^{n_f} = g_{n_s,n}(x_2).$$

If we check all possible outcomes, and marginalize (sum) over them, we get the joint prior of x_2 given x_1 as

$$f_{x_2}(x_2|x_1) = \sum_{i=0}^{n} f_{x_2}(x_2|D_i)p(D|x_1) = \sum_{i=0}^{n} \left((n+1)\binom{n}{i} x_2^i(1 - x_2)^{n-i}\right)\left(\binom{n}{i} x_1^i(1 - x_1)^{n-i}\right).$$

Using the prior $f_{x_1}(x_1)$ of x_1 we can determine the joint prior of x_1 and x_2 as

$$f_{x_1,x_2}(x_1, x_2) = f_{x_2}(x_2|x_1)f_{x_1}(x_1) = \sum_{i=0}^{n} \left(\binom{n}{i} x_1^i(1 - x_1)^{n-i}\right)\left((n+1)\binom{n}{i} x_2^i(1 - x_2)^{n-i}\right).$$

This can be simplified as

$$f_{x_1,x_2}(x_1, x_2) = f_{x_2}(x_2|x_1)f_{x_1}(x_1) = (n+1)\sum_{i=0}^{n} \binom{n}{i}^2 x_1^i(1 - x_1)^{n-i} x_2^i(1 - x_2)^{n-i}.$$

Or an even shorter notation is

$$f_{x_1,x_2}(x_1, x_2) = \frac{1}{(n+1)}\sum_{i=0}^{n} g_{i,n}(x_1)g_{i,n}(x_2).$$

2.2. LINKS BETWEEN ADDITIONAL EXERCISES

Previously we have defined a prior over x_1 and x_2. We can define an identical prior over x_2 and x_3 as

$$f_{x_2,x_3}(x_2,x_3) = \frac{1}{(n+1)}\sum_{i=0}^{n} g_{i,n}(x_2)g_{i,n}(x_3).$$

It is interesting to wonder: how are x_1 and x_3 then linked? The key here is to assume that x_1 and x_3 are conditionally independent given x_2. This is a reasonable assumption to make, since the case where a student has done two exercises always falls between the case where a student has done one or three exercises.

In practice this means that we can use x_1 to derive a distribution for x_2. Once we have this, we can throw the distribution of x_1 out of the window, and use the distribution of x_2 to derive the distribution of x_3. And this continues too: once we have a distribution of x_3, we can throw the data of x_2 out of the window and move on from here towards x_4.

Okay, in theory we could also derive a joint prior for x_1 and x_3. To do so we would have to use the definition of conditional independence to write

$$f_{x_1,x_3}(x_1,x_3|x_2) = f_{x_1}(x_1|x_2)f_{x_3}(x_3|x_2).$$

Using this, we can also write the joint prior of x_1, x_2 and x_3 as

$$f_{x_1,x_2,x_3}(x_1,x_2,x_3) = f_{x_1,x_3}(x_1,x_3|x_2)f_{x_2}(x_2) = f_{x_1}(x_1|x_2)f_{x_3}(x_3|x_2)f_{x_2}(x_2) = \frac{f_{x_1,x_2}(x_1,x_2)f_{x_2,x_3}(x_2,x_3)}{f_{x_2}(x_2)}.$$

To find the prior over x_1 and x_3 we should marginalize over x_2. This gives the integral

$$f_{x_1,x_3}(x_1,x_3) = \int_0^1 f_{x_1,x_2,x_3}(x_1,x_2,x_3)\, f_{x_2}(x_2)\, dx_2 = \int_0^1 f_{x_1,x_2}(x_1,x_2)f_{x_2,x_3}(x_2,x_3)\, dx_2.$$

Solving this will give us the joint prior of $f_{x_1,x_3}(x_1,x_3)$. In theory this can be solved, but in practice we don't really use this prior, so we will stop our derivation here.

2.3. INFERRING THE SUCCESS RATE OF THE NEXT EXERCISE: USING COEFFICIENT VECTORS

Suppose that a student has made a lot of exercises. Say, they're already up to k exercises. This tells us something about the distribution of x_k. However, it is more interesting to know what this tells us about the distribution of x_{k+1}: what is the chance that the student will do the next exercise correctly?

To see how this inference works, first examine a simplified case. Suppose that, through extensive measurements, we have determined exactly what x_k is. (For instance $x_k = 0.8$.) What is the posterior distribution for x_{k+1} then?

The answer is relatively simple here. We can directly insert x_k into the prior $f_{x_k,x_{k+1}}(x_k,x_{k+1})$ and find

$$f_{x_{k+1}}(x_{k+1}|x_k) = f_{x_k,x_{k+1}}(x_k,x_{k+1}) = \frac{1}{(n+1)}\sum_{i=0}^{n} g_{i,n}(x_k)g_{i,n}(x_{k+1}).$$

There is another way to write this. We could define coefficients $c_0, \dots, c_n$ as

$$c_i = \frac{g_{i,n}(x_k)}{n+1} = \binom{n}{i} x_k^i (1-x_k)^{n-i}.$$

In this case we can write the posterior distribution of x_{k+1} as

$$f_{x_{k+1}}(x_{k+1}|D) = \sum_{i=0}^{n} c_i \cdot g_{i,n}(x_{k+1}).$$

So if we know the coefficients c_i (with $0 \leq i \leq n$) then the distribution of x_{k+1} is defined! Apparently an efficient way to define a distribution is through a set or row vector $\boldsymbol{c}$ of coefficients, with n the length of said vector. And once we do have such a coefficient vector $\boldsymbol{c}$, then the distribution always follows as

$$f_x(x) = \sum_{i=0}^{n} c_i \cdot g_{i,n}(x).$$

We can even write this in vector notation. If we define $\boldsymbol{c} = [c_0, \ldots, c_n]$ and $\boldsymbol{g_n}(x) = \left[g_{1,n}(x), \ldots, g_{n,n}(x)\right]^T$ then

$$f_x(x) = \boldsymbol{c}\boldsymbol{g_n}(x).$$

We define n as the *order* of the coefficient vector. So the *order* of a coefficient vector is its length minus one.

2.4. PROPERTIES OF COEFFICIENT VECTORS

We can explore the idea of coefficient vectors some more.

Example coefficient arrays of simple cases

Let's consider the solutions from the *previous* chapter. What would our coefficient vector $\boldsymbol{c}$ look like there?

- For our flat prior distribution $f_x(x) = 1$ we have $\boldsymbol{c} = [1]$. (Note that here $n = 1$.)
- For a single success we have $f_x(x) = 2x$ which comes down to $\boldsymbol{c} = [0, 1]$.
- For a single failure we have $f_x(x) = 2(1-x)$ which comes down to $\boldsymbol{c} = [1, 0]$.
- For three successes we have $f_x(x) = 3x^2$ which implies $\boldsymbol{c} = [0, 0, 1]$.

So for all cases in the first chapter, we had a very simple coefficient vector: there was only a single one in it and all other coefficients were zero. In addition, the order of the coefficient vector equaled the number of exercises done. Quite soon we shall see that this will not hold in general: all coefficients can have values and the order can vary too.

The sum of the coefficients

An important property of every coefficient array is that the sum of all the coefficients must always equal one. To see why this latter property holds, consider the integral over the above PDF. If we examine

$$\int_0^1 f_x(x)\, dx.$$

then we must find one as outcome. After all, the integral over any PDF must equal one. Hence we must have

$$\int_0^1 f_x(x)\, dx = \int_0^1 \left(\sum_{i=0}^{n} c_i \cdot g_{i,n}(x) \right) \cdot dx = \sum_{i=0}^{n} c_i \cdot \int_0^1 g_{i,n}(x)\, dx = 1.$$

However, note that $g_{i,n}(x)$ is by itself also a PDF, so the integral over $g_{i,n}(x)$ will also equal one. Hence

$$\sum_{i=0}^{n} c_i = 1.$$

This means that it is also very easy to "normalize" any distribution (scale it up such that its integral equals one): we just need to scale all coefficients so that their sum equals one. This trick will come in handy later.

Expected values of distributions

Next, suppose we have some coefficient vector $\boldsymbol{c}$. What will the expected value $E[x]$ of the resulting distribution be? This can be found through

$$\bar{x} = E[x] = \int_0^1 x f_x(x)\, dx = \int_0^1 x \left(\sum_{i=0}^{n} c_i \cdot g_{i,n}(x) \right) dx = \sum_{i=0}^{n} c_i \left(\int_0^1 x \cdot g_{i,n}(x)\, dx \right).$$

In Section 1.5 we already found the solution for this integral. The result will be

$$\bar{x} = E[x] = \sum_{i=0}^{n} c_i \frac{(i+1)}{(n+2)}.$$

The result is a linearly weighted sum of the coefficients. For instance, if $\boldsymbol{c} = [0, 0, 1]$, then the mean will be

$$\bar{x} = 0 \cdot \frac{1}{4} + 0 \cdot \frac{2}{4} + 1 \cdot \frac{3}{4} = \frac{3}{4}.$$

This seems to make sense: if a student did two exercises correctly, it is reasonable to say there's a 75% chance they will get the next exercise correct.

Inverting coefficient arrays

There is another property to note about coefficient arrays. Suppose that we have a coefficient array $\boldsymbol{c_x}$ describing the distribution $f_x(x)$ of x. What would the distribution of $\check{x} = 1 - x$ look like? This is just the mirror image! We could "flip" the coefficient array, like

$$c_{\check{x},i} = c_{x,n-i}.$$

And then the distribution of $\check{x}$ will follow in the usual way as

$$f_{\check{x}}(\check{x}) = \boldsymbol{c}_{\check{x}} \boldsymbol{g_n}(\check{x}).$$

We call this *inverting* the coefficient array: reversing the order of the coefficients.

Increasing coefficient array size

Suppose that $f_x(x) = \boldsymbol{c_x} \boldsymbol{g_n}(x)$ for some coefficient vector $\boldsymbol{c_x}$ of size n, but we want this size to increase without changing $f_x(x)$. This is possible. To see how, note that

$$f_x(x) = \sum_{i=0}^{n} c_i \cdot g_{i,n}(x) = \sum_{i=0}^{n} c_i (n+1) \binom{n}{i} x^i (1-x)^{n-i}.$$

This is equal to

$$f_x(x) = (1-x)f_x(x) + xf_x(x) = \sum_{i=0}^{n} c_i(n+1)\binom{n}{i}(x^i(1-x)^{n-i+1} + x^{i+1}(1-x)^{n-i}).$$

If we now define

$$c_i^+ = \begin{cases} \dfrac{(n+1)}{(n+2)}c_0 & \text{for } i = 0, \\ \dfrac{(i+1)}{(n+2)}c_{i-1} + \dfrac{(n+1-i)}{(n+2)}c_i & \text{for } 1 \le i \le n, \\ \dfrac{(n+1)}{(n+2)}c_n & \text{for } i = n+1, \end{cases}$$

then we can write $f_x(x) = \boldsymbol{c_x^+ g_{n+1}}(x)$. So effectively we have increased the size of the coefficient array without changing the probability density function $f_x(x)$. This is usually not so useful (why increase the required storage space?) but in some specific applications it comes in handy.

2.5. STEP 1 OF INCORPORATING NEW DATA: UPDATING THE CURRENT DISTRIBUTION

Suppose that, after the student has done $k-1$ exercises, we have predicted the distribution for x_k as

$$f_x(x_k|D_{k-1}) = \boldsymbol{c_k g_n}(x_k),$$

for some coefficient vector $\boldsymbol{c_k} = [c_{k,0}, \dots, c_{k,n}]$. We have defined D_{k-1} here as the set of observations of *all* exercises up to and including exercise $k-1$. What happens when the student does exercise k? In that case we need to update our distributions.

First we need to update the distribution of x_k itself, given this new piece of data. This goes through Bayes' law as

$$f_x(x_k|D_k) = \frac{p(D_k|x_k, D_{k-1})f_x(x_k|D_{k-1})}{p(D_k|D_{k-1})}.$$

In this equation we have three terms on the right-hand side.

- $p(D_k|x, D_{k-1})$ is the probability (given x_k) that we got all k observations, given that we already got the first $k-1$ observations. In other words, this is the probability that we only got our last observation on exercise k. Hence, if the exercise was a success, this equals x_k and otherwise it equals $1-x_k$.
- $f_x(x_k|D_{k-1})$ is the PDF of x_k prior to knowing the result of exercise k. It equals $\boldsymbol{c_k g_i}(x_k)$.
- $p(D_k|D_{k-1})$ is a constant, in the sense that it does not depend on x_k. We can ignore it, and simply use coefficient normalization afterwards to incorporate it.

We can hence write the right-hand side as

$$f_x(x_k|D_k) = \frac{xp(D_k|x_k, D_{k-1})}{p(D_k|D_{k-1})} \cdot \sum_{i=0}^{n} c_{k,i}g_{i,n}(x_k).$$

Expanding $g_{i,n}(x)$, this can be written as

$$f_x(x_k|D_k) = \frac{p(D_k|x_k, D_{k-1})}{p(D_k|D_{k-1})} \cdot \sum_{i=0}^{n} c_{k,i}(n+1)\binom{n}{i}x_k^i(1-x_k)^{n-i}.$$

Assume that the last exercise was a success and that hence $p(D_k|x_k, D_{k-1}) = x_k$. In this case we have

$$f_x(x_k|D_k) = \frac{1}{p(D_k|D_{k-1})} \cdot \sum_{i=0}^{n} c_{k,i}(n+1)\binom{n}{i} x_k^{i+1}(1-x_k)^{n-i}.$$

By defining $j = i + 1$ we can rewrite the above to

$$f_x(x_k|D_k) = \frac{1}{p(D_k|D_{k-1})} \cdot \sum_{j=1}^{n+1} c_{k,j-1} \cdot j \cdot \binom{n+1}{j} x_k^{j}(1-x_k)^{(n+1)-j}.$$

From this we can make a very interesting observation. If we define $c^*_{k,0} = 0$ and

$$c^*_{k,j} = \frac{c_{k,j-1} \cdot j}{p(D_k|D_{k-1})},$$

for $1 \leq j \leq n+1$ then we get a new coefficient vector $\boldsymbol{c}^*_{\boldsymbol{k}}$ of order $n_* = n + 1$ such that $f_x(x_k|D_k) = \boldsymbol{c}^*_{\boldsymbol{k}}\boldsymbol{g}_{\boldsymbol{n}_*}(x_k)$. So we have a new coefficient vector for our updated distribution!

The only problem is that we do not know $p(D_k|D_{k-1})$ yet. However, we do know that

$$\sum_{j=0}^{n+1} c^*_j = \frac{1}{p(D_k|D_{k-1})} \sum_{j=1}^{n+1} c_{j-1} \cdot j = 1.$$

This implies that

$$p(D_k|D_{k-1}) = \sum_{j=1}^{n+1} c_{j-1} \cdot j.$$

However, we do not even need this fact! In practice, it is much easier to simply first define a new coefficient vector $\boldsymbol{c}^*_{\boldsymbol{k}}$ according to $c^*_0 = 0$ and $c^*_j = c_{j-1} \cdot j$ for $1 \leq j \leq n+1$ and then to normalize the resulting vector, setting the sum of all coefficients to one.

To summarize, if the student gets the exercise correct, then we can find the updated distribution of x_k through

$$c^*_{k,j} = \begin{cases} 0 \text{ if } j = 0, \\ j c_{k,j-1} \text{ if } 1 \leq j \leq n+1, \end{cases}$$

where afterwards the coefficient vector must be normalized. If the student however *failed* the exercise, then a similar derivation can give us the update law

$$c^*_{k,j} = \begin{cases} (n+1-j)c_{k,j} \text{ if } 0 \leq j \leq n, \\ 0 \text{ if } j = n+1, \end{cases}$$

where again the vector must be normalized afterwards. In both cases the new coefficient vector $\boldsymbol{c}^*_{\boldsymbol{k}}$ will have order $n_* = n + 1$, while the former coefficient vector $\boldsymbol{c}_{\boldsymbol{k}}$ had order n.

2.6. STEP 2 OF INCORPORATING NEW DATA: SMOOTHING TO GET THE NEXT DISTRIBUTION

The previous section has given us the posterior distribution of x_k given data on the outcome of exercise k. We can now use this data to predict x_{k+1}: find its distribution $f_{x_{k+1}}(x_{k+1}|D_k)$.

To do so, we use the prior distribution of x_k and x_{k+1} through

$$f_{x_{k+1}}(x_{k+1}|D_k) = \int_0^1 f_{x_k,x_{k+1}}(x_k, x_{k+1}|D_k)\, dx_k = \int_0^1 f_{x_{k+1}}(x_{k+1}|x_k, D_k) f_{x_k}(x_k|D_k)\, dx_k.$$

Note that in the term $f_{x_{k+1}}(x_{k+1}|x_k, D_k)$ the observations D_k are irrelevant: if x_k is known precisely, then observations about previous exercises do not matter anymore. Hence, we can write this as

$$f_{x_{k+1}}(x_{k+1}|x_k, D_k) = f_{x_{k+1}}(x_{k+1}|x_k) = \frac{f_{x_k,x_{k+1}}(x_k, x_{k+1})}{f_{x_k}(x_k)}.$$

It follows that our inference law is

$$f_{x_{k+1}}(x_{k+1}|D_k) = \int_0^1 \frac{f_{x_k,x_{k+1}}(x_k, x_{k+1}) f_{x_k}(x_k|D_k)}{f_{x_k}(x_k)}\, dx_k.$$

There are three terms on the right-hand side.

- $f_{x_k,x_{k+1}}(x_k, x_{k+1})$ is the joint prior of x_k and x_{k+1}, which we defined at the start of this chapter.
- $f_{x_k}(x_k|D_k)$ is the distribution which we derived in the previous section as $\boldsymbol{c_k^* g_{n_*}}(x_k)$.
- $f_{x_k}(x_k)$ is the prior of x_k without any data incorporated. Since we are using a flat prior this becomes 1.

As a result, we can write the above as

$$f_{x_{k+1}}(x_{k+1}|D_k) = \int_0^1 \frac{1}{(n_p+1)} \left(\sum_{i=0}^{n_p} g_{i,n_p}(x_k) g_{i,n_p}(x_{k+1}) \right) \left(\sum_{j=0}^{n_*} c_{k,j}^* g_{j,n_*}(x_k) \right) dx_k.$$

Note that the order n_p of the prior can be (and generally is) different from the order n^* of the distribution of x_k. In fact, we can freely choose n_p while n^* follows from the previous distribution of x_k. (Later in this chapter will we examine how to choose n_p and what effects certain choices have.)

The above equation can be further reorganized as

$$f_{x_{k+1}}(x_{k+1}|D_k) = \sum_{i=0}^{n_p} \left(\frac{1}{(n_p+1)} \int_0^1 g_{i,n_p}(x_k) \left(\sum_{j=0}^{n_*} c_{k,j}^* g_{j,n_*}(x_k) \right) dx_k \right) g_{i,n_p}(x_{k+1}).$$

This implies something very interesting. If we define a new coefficient vector $\boldsymbol{c_{k+1}}$ with order n_p through

$$c_{k+1,i} = \frac{1}{(n_p+1)} \int_0^1 g_{i,n_p}(x_k) \left(\sum_{j=0}^{n_*} c_{k,j}^* g_{j,n_*}(x_k) \right) dx_k,$$

for $0 \leq i \leq n_p$, then we have as final distribution $f_{x_{k+1}}(x_{k+1}|D_k) = \boldsymbol{c_{k+1} g_{n_p}}(x_{k+1})$. So then it all works out!

All that is left for us to do is solve the above integral. We can write x_k as x to simplify notation, as well as expand the g functions. This gives us

$$c_{k+1,i} = \frac{1}{(n_p+1)} \int_0^1 (n_p+1) \binom{n_p}{i} x^i (1-x)^{n_p-i} \left(\sum_{j=0}^{n^*} c_{k,j}^* (n_*+1) \binom{n_*}{j} x^j (1-x)^{n_*-j} \right) dx.$$

Reorganizing turns this into

$$c_{k+1,i} = \sum_{j=0}^{n^*} c_{k,j}^* (n_* + 1) \binom{n_p}{i} \binom{n_*}{j} \int_0^1 x^{i+j} (1-x)^{n_p+n_*-i-j} \, dx.$$

Note that we can solve the integral using the basic integration result from Section 1.4, which results in

$$\int_0^1 x^{i+j} (1-x)^{n_p+n_*-i-j} \, dx = \frac{(i+j)!\,(n_p+n_*-i-j)!}{(n_p+n_*+1)!} = \frac{1}{(n_p+n_*+1)\binom{n_p+n_*}{i+j}}.$$

This turns the expression for the coefficients into

$$c_{k+1,i} = \sum_{j=0}^{n^*} c_{k,j}^* \left(\frac{n_*+1}{n_p+n_*+1} \right) \frac{\binom{n_p}{i}\binom{n_*}{j}}{\binom{n_p+n_*}{i+j}}.$$

With some playing around with factorial operators we can rewrite this to

$$c_{k+1,i} = \sum_{j=0}^{n^*} c_{k,j}^* \frac{\binom{i+j}{i}\binom{n_p+n_*-i-j}{n_*-j}}{\binom{n_p+n_*+1}{n_p}}.$$

Note that we technically do not even have to divide by the constant denominator of the above expression. After all, we could also just ignore this constant and normalize all coefficients afterwards. If we do take into account this denominator, then normalization will not be needed anymore (barring numerical inaccuracies) since all the coefficients automatically sum up to one.

We call this forward prediction step "smoothing". The reason is that the distribution of x_{k+1} is always smoother than the distribution of x_k. If you just smoothen enough, you'll eventually wind up with the flat distribution. It's also worthwhile to note that the order of the resulting distribution $f_{x_{k+1}}(x_{k+1}|D_k)$ equals the number n_p.

2.7. THE EFFECTS OF THE SMOOTHING ORDER n_p IN THE PRIOR

We just saw that the order n_p that we pick for our prior determines how many coefficients our resulting distribution $f_{x_{k+1}}(x_{k+1}|D_k)$ will have! You would say, "Just pick a small number n_p and that saves some memory." However, the smoothing order n_p has some interesting effects which must be taken into account!

To see this effect, let's do a thought experiment. Suppose that we know x_k deterministically. (For example, $x_k = 0.8$. We could then even say that $f_{x_k}(x_k|D_k) = \delta_{0.8}(x_k)$, with $\delta_a(x)$ the delta function that only has a nonzero value when $a = x$.) In that case we will have

$$f_{x_{k+1}}(x_{k+1}|D_k) = f_{x_k,x_{k+1}}(x_k, x_{k+1}) = \sum_{i=0}^{n_p} \frac{g_{i,n_p}(x_k)}{(n_p+1)} g_{i,n_p}(x_{k+1}),$$

with x_k this constant known value. For this simplified case, we see that the coefficients $c_{k+1,i}$ directly follow as

$$c_{k+1,i} = \frac{g_{i,n_p}(x_k)}{(n_p+1)} = \binom{n_p}{i} x_k^i (1-x_k)^{n_p-i}.$$

The main question now is: what is the mean value of x_{k+1}? To find this mean $\bar{x}_{k+1}$ we use the methods from Section 2.4,

$$\bar{x}_{k+1} = \sum_{i=0}^{n} c_{k+1,i} \frac{(i+1)}{(n_p+2)}.$$

Inserting the relation for $c_{k+1,i}$ turns this into

$$\bar{x}_{k+1} = \sum_{i=0}^{n} \frac{(i+1)}{(n_p+2)} \binom{n_p}{i} x_k^i (1-x_k)^{n_p-i}.$$

We can split the above up into two parts by splitting up $(i+1)$, resulting in

$$\bar{x}_{k+1} = \frac{1}{(n_p+2)} \left(\sum_{i=0}^{n} \binom{n_p}{i} x_k^i (1-x_k)^{n_p-i} + \sum_{i=0}^{n} i \binom{n_p}{i} x_k^i (1-x_k)^{n_p-i} \right).$$

The first sum is a basic binomial expansion: it reduces to $\left(x_k + (1-x_k)\right)^{n_p}$ which equals 1. The second sum can, with some more work, be reduced to $n_p x_k$. Hence we have

$$\bar{x}_{k+1} = \frac{1 + n_p x_k}{n_p + 2}.$$

But what happens if we do not know x_k deterministically? What if we only have a distribution for x_k with some mean $\bar{x}_k$? Through the principle of superpositioning, we can see that then exactly the same conclusion holds, except we need to use the mean $\bar{x}_k$. That is,

$$\bar{x}_{k+1} = \frac{n_p \bar{x}_k + 1}{n_p + 2}.$$

The above equation is not very descriptive though. It gets more informative if we write it as

$$\bar{x}_{k+1} = \bar{x}_k - \frac{2}{n_p+2}\left(\bar{x}_k - \frac{1}{2}\right) = \frac{1}{2} + \frac{n_p}{n_p+2}\left(\bar{x}_k - \frac{1}{2}\right).$$

Especially the first half is telling. It says that we should check how far away the mean before smoothing is away from ½. We take this deviation, multiply it by a factor $2/(n_p+2)$, and then bring the mean of our distribution this much closer towards ½. In other words (as the latter half of the equation says) we bring our distribution a factor $n_p/(n_p+2)$ closer towards the flat distribution, which has mean ½. This factor $n_p/(n_p+2)$ is also known as the *decay ratio* r (or sometimes it's called the *smoothing factor*).

The conclusion is: the smaller n_p is, the more we smoothen the distribution, losing data. The ultimate case is $n_p = 0$, in which we are guaranteed to get the flat distribution as outcome, having lost all our data. Of course some amount of smoothing is desired – it's the whole point – but we should not apply too much. Which amount of "data loss/forgetting" is appropriate depends on the situation, which is what we will look into next.

2.8. DETERMINING AND IMPLEMENTING THE RIGHT AMOUNT OF SMOOTHING

Using the smoothing step has a very important advantage: it introduces a certain amount of forgetting into the algorithm. If a student first fails an exercise three times, then discovers the trick and gets successful executions afterwards, the first three failures shouldn't be too big of a disadvantage. Smoothing provides this forgetting effect: inherently it ensures that a tenth execution x_{10} is not so correlated with an earlier execution like x_4. However, if we do too much smoothing, then even x_{10} and x_9 will hardly be linked anymore. Our model will have lost its predictive capabilities. So what is the right amount of smoothing?

Defining requirements

This is not a mathematical problem but one of preferences/heuristics. As a result, we should note a few preferences or requirements first.

- What we said before: there should be enough smoothing to eventually forget the first exercises if they were incorrect, but not so much that predictions are not possible anymore.
- Time plays a role: if there is a lot of time between two skill executions, then there should be more smoothing.
- The amount of earlier practice plays a role: if a student has already done something a thousand times, there should be less smoothing than if this is the first time they practice with the skill. (After all, if we always continue to forget previous exercises, a student can never get close to a 100% mastery.)

We will work to implement all these requirements in a model.

Defining the decay ratio r

The key lies in properly defining the *decay ratio* r. To implement the first requirement, we define the decay ratio if a student practices a first exercise and then immediately a second one. In this case a decay ratio of $r_0 = 0{,}90$ seems sensible. (Note that a decay ratio of 0 would immediately grant us the flat distribution, forgetting everything, while a decay ratio of 1 would not forget anything, giving us the model of chapter 1.)

To implement the second requirement, we add time effects. The more time there is between two subsequent exercises, the lower the decay ratio should be. We define a halftime here of $t_{1/2} = 1\ \text{year}$. So after waiting for a full year, half of all previous data is forgotten. This results in the decay ratio

$$r = r_0 \cdot \left(\frac{1}{2}\right)^{t/t_{1/2}}.$$

Next, we make a minor addition to the above rule. It turned out to be more convenient to not define an initial decay ratio $r_0 = 0{,}90$ for the practice effect, but to use an *exercise time-equivalent contribution* t_e. This is the amount of time that a student needs to wait, which has just the same "forgetting" effect as (is equivalent with) practicing another exercise. So we (initially) define $t_e = 2\ \text{months}$ and write

$$r = \left(\frac{1}{2}\right)^{(t_e+t)/t_{1/2}}.$$

Effectively, the above two equations do exactly the same, so there is not much difference. The main difference comes next, to incorporate the third requirement. If the student has practiced dozens of exercises, the practice effect (implemented through t_e) should not be as significant. As a result, we will reduce t_e through practice. We count the number of skill executions n_e (the number of times the student has practiced with this skill) and define

$$t_e = t_{e,0}\left(\frac{1}{2}\right)^{n_e/n_{1/2}},$$

Here we use an initial exercise time-equivalent of $t_{e,0} = 2\ \text{months}$ and a half-life of $n_{1/2} = 8\ \text{exercises}$. That is, after practicing eight exercises, the practice effect is halved. So effectively, practicing another exercise then only counts as *one* month of not practicing instead of the default *two* months. With this, we always have a decay ratio that satisfies all our requirements. Furthermore, the exact model settings can still be tuned, if the system does not behave as expected/desired.

Implementing the decay ratio

Now that we have a decay ratio r, we need to apply it through smoothing. We know that in general

$$r = \frac{n_p}{n_p + 2},$$

with n_p the smoothing order. This implies we should use a smoothing order

$$n_p = \frac{2r}{1 - r}.$$

For instance, if we have a decay ratio of $r = 3/4$ then we find $n_p = 6$, which works out beautifully! Or if we use $r = 2/3$ then we get $n_p = 4$, which is also nice. However, in reality it's not that simple. If we pick $r = 4/7 = 0{,}571428$ then we get $n_p = 8/3 = 2.667$, which is not an integer. We run into the problem that we can only perform smoothing with integer smoothing orders. So how do we solve this?

The solution here is to smoothen multiple times. We want to split the decay ratio r up into multiple ratios $r = r_1 \cdot r_2 \cdot r_3 \cdot \ldots$ such that all sub-decay-ratios r_i each result in an integer smoothing order. This is done according to the following procedure.

- First pick $n_{p,1} = \lceil 2r/(1 - r) \rceil$, and use this to define $r_1 = n_{p,1}/(n_{p,1} + 2)$. For our example of $r = 4/7$ this give $n_{p,1} = 3$, which translates back to a decay ratio as $r_1 = 3/5$.
- Calculate the still-to-be-implemented decay ratio $r_r = r/r_1$. For our example this is

$$r = \frac{r}{r_1} = \frac{4/7}{3/5} = \frac{20}{21} = 0.9524.$$

- Second, pick $n_{p,2} = \lceil 2r_r/(1 - r_r) \rceil$, and use this to define $r_2 = n_{p,2}/(n_{p,2} + 2)$. For our example this gives $n_{p,2} = 40$, which results in $r_2 = 40/42 = 20/21 = 0.9524$.
- Calculate the still-to-be-implemented decay ratio $r_r = r/(r_1 \cdot r_2)$. (Or update r_r through $r_r \leftarrow r_r/r_2$.) For our example we get $r_r = 1$ and we are done, but usually this does not occur and $r_r < 1$.
- Keep on doing the above steps, until you wind up with a smoothing order $n_p > n_{p,max}$. After this, smoothing is pretty much ineffective anyway, so can be neglected. For our example the next smoothing order would be $n_{p,3} = \infty$ (no smoothing) which is definitely above the maximum.

In practice we use a maximum smoothing order of $n_{p,max} = 120$ because above this numerical inaccuracies will also cause problems. Most often, after roughly three steps and hence three smoothing orders $n_{p,1}, n_{p,2}, n_{p,3}$ the above algorithm is done.

Next, these smoothing steps need to be implemented. The key here is to *work backwards*: first smoothen with order $n_{p,3}$ (a generally large number), then with order $n_{p,2}$ (a medium number) and finally with $n_{p,1}$ (a small number). The reason here is that, by smoothing with a small number last, we wind up with a smaller coefficient vector $\boldsymbol{c_{k+1}}$ and hence fewer coefficients to store in our database, without any reduction in accuracy.

Using the above method, we can implement any decay ratio r, hence providing continuous smoothing capabilities. And by using the right decay ratio, we satisfy all the requirements our system should have.

2.9. STORING COEFFICIENT ARRAYS

In practice we will need to store coefficients in a database. But which ones will we store?

The solution here is to always store coefficients *after* an update step (the star-coefficients $\boldsymbol{c_k^*}$) but *before* smoothing. So we are storing the *posterior* distribution of x_k: the chance that the student did the *previous* exercise correctly.

When a student then requests their own skill level, we will give them the distribution of x_{k+1}: the chance that they will do the *next* (upcoming) exercise correctly. To calculate this, we will apply a smoothing step, taking into account the time that has passed since the last time the student did the exercise. The good news here is that, on such a read-request, we only do smoothing: no database update is required. It's all practically manageable.

Finally there are also numerical issues to consider. It may happen that, due to numerical inaccuracies, coefficients turn out to be negative. (Like $-1 \cdot 10^{-16}$ or so.) With perfect precision this of course would never happen. To eliminate negative effects, we process coefficient arrays after every operation: set negative elements to zero and subsequently normalize, forcing the sum of all coefficients to equal 1 again. This prevents any numerical inaccuracies from becoming a problem.

3. ESTIMATING EXERCISE AND SKILL SUCCESS RATES: INFERENCE

Skills are often linked. In this chapter we're going to study what kind of links exist and how we can take them into account. We are only going to do so to make predictions of skill levels. We leave the updating of skill levels for the next chapter.

3.1. ESTIMATING AN EXERCISE SUCCESS RATE: THE MEAN

Consider a composite sum like $3 \cdot 9 + 8 \cdot 7$. To calculate it, a student needs to apply multiplication twice and addition once. (Okay, they also need to know about the order of operations – that multiplication takes precedence over addition – but let's ignore that now.) Given the skill data on multiplication and addition, what is the success rate for this exercise?

Let's call a the chance that a student will do skill A (multiplication) correctly, and b the chance that the student successfully calculates skill B (addition). In this case the probability of success for the exercise is $x = a^2 b$. However, x has its own distribution. What we are really looking for is the mean $\bar{x} = E[x] = E[a^2 b]$. How do we find it?

Our starting point is the distributions $f_{a_k}(a_k)$ and $f_{b_k}(b_k)$. These are the probabilities that we did the previous exercises for skills A and B correct, and we have them in our database. To make a prediction, we first need to apply smoothing to calculate $f_{a_{k+1}}(a_{k+1})$ and $f_{b_{k+1}}(b_{k+1})$. After all, maybe we haven't done skill A in a long time, while we practiced skill B recently. This time-based deterioration of skills should be taken into account.

Technically, since we do skill A twice, we could smoothen twice to also get the distribution of $f_{a_{k+2}}(a_{k+2})$, but we do not do this: we assume that the probability of success is equal (has the same distribution) for both executions of skill A. This gives us the distribution for a and b which, for ease of notation, we will simply write as $f_a(a)$ and $f_b(b)$. These are hence the distributions after smoothing.

The next assumption that we make is that a and b are independent. The chance that we do a summation correct does not depend on the chance that we do a multiplication correct. Though not entirely true (if you're tired, you're more likely to fail both, so there is some correlation) this assumption is necessary to be able to solve the math. Hence we have

$$\bar{x} = E[a^2 b] = E[a^2]E[b].$$

This shows that the mean of x is just a multiplication of various moments of the distributions of a and b. Section 1.5 showed us how to calculate these moments, so by directly applying these equations we can find $\bar{x}$.

3.2. ESTIMATING AN EXERCISE SUCCESS RATE: THE FULL DISTRIBUTION

Let's take things a bit further. Given that $x = a^2 b$, can we also find the *distribution* of x? So not just its mean? The answer is yes.

To do so, we first define $\hat{a} = a^2$ and find its distribution $f_{\hat{a}}(\hat{a})$. Note that the transformation from a to $\hat{a} = a^2$ is done through the function $\hat{a} = g(a) = a^2$. This function $g(a)$ is a one-to-one function, since we only operate on the interval $[0,1]$. The inverse transformation is hence $g^{-1}(a) = \sqrt{a}$. The distribution of this transformed random variable equals

$$f_{\hat{a}}(\hat{a}) = \left|\frac{d}{d\hat{a}} g^{-1}(\hat{a})\right| f_a\left(g^{-1}(\hat{a})\right) = \frac{1}{2\sqrt{\hat{a}}} f_a\left(\sqrt{\hat{a}}\right).$$

This is a valid distribution, albeit one that cannot be described by a coefficient vector. It's an altogether different type of distribution, but it's one that we can calculate.

The next step would be to find the distribution of $x = \hat{a}b$. This is done through the product distribution

$$f_x(x) = \int_{-\infty}^{\infty} f_{\hat{a}}(\hat{a}) f_b\left(\frac{x}{\hat{a}}\right) \frac{1}{|\hat{a}|} d\hat{a} = \int_x^1 f_{\hat{a}}(\hat{a}) f_b\left(\frac{x}{\hat{a}}\right) \frac{1}{\hat{a}} d\hat{a},$$

where we used the knowledge that $f_{\hat{a}}$ and f_b only take inputs on the interval $[0,1]$. Again, this is a distribution that we are able to calculate, but it's not a nice one that can be described by a coefficient vector. It would be really beneficial to return back to distributions described by coefficient vectors. So how does this work?

3.3. ESTIMATING AN EXERCISE SUCCESS RATE: FINDING THE COEFFICIENTS

The key realization in the previously described problem is that, when predicting x based on a and b, there's also uncertainty involved. Maybe a skill like a composite sum (that is, x) does not directly equate to doing summation twice (a^2) and multiplication once (b). There are also other things involved, like reading the problem, entering an answer, and so forth. Sometimes these factors are slightly beneficial and sometimes slightly detrimental. The conclusion is: we need to add more uncertainty. We need to smoothen the distribution! The advantage of this is that this once more gives us a distribution that can be described by a coefficient vector.

Let's denote the smoothened parameter x_s. So while $x = a^2 b$, we define the prior distribution of x_s and x as

$$f_{x_s,x}(x_s, x) = \frac{1}{(n_s+1)} \sum_{i=0}^{n_s} g_{i,n_s}(x_s) g_{i,n_s}(x),$$

with n_s the chosen smoothing order. In practice, to prevent prior skills from having too strong of an effect on the estimation of x, we usually choose a low value, like $n_s = 4$, but this can be tuned as desired. The posterior distribution of x_s, given data D_{ab} on a and b, is hence

$$f_{x_s}(x_s|D_{ab}) = \int_0^1 f_{x_s,x}(x_s, x|D_{ab})\, dx = \int_0^1 f_{x_s}(x_s|x) f_x(x|D_{ab})\, dx = \int_0^1 \frac{f_{x_s,x}(x_s, x) f_x(x|D_{ab})}{f_x(x)}\, dx,$$

where we used the assumption that x_s and D_{ab} are conditionally independent given x. Noting that $f_x(x) = 1$ we can rewrite the above to

$$f_{x_s}(x_s|D_{ab}) = \int_0^1 \frac{1}{(n_s+1)} \sum_{i=0}^{n_s} g_{i,n_s}(x_s) g_{i,n_s}(x) \left(\int_x^1 f_{\hat{a}}(\hat{a}) f_b\left(\frac{x}{\hat{a}}\right) \frac{1}{\hat{a}} d\hat{a} \right) dx.$$

Or, adjusting the order of the terms, this becomes

$$f_{x_s}(x_s|D_{ab}) = \sum_{i=0}^{n_s} \left(\int_0^1 \frac{1}{(n_s+1)} \left(\int_x^1 f_{\hat{a}}(\hat{a}) f_b\left(\frac{x}{\hat{a}}\right) \frac{1}{\hat{a}} d\hat{a} \right) g_{i,n_s}(x)\, dx \right) g_{i,n_s}(x_s) = \sum_{i=0}^{n_S} c_i g_{i,n_s}(x_s),$$

where we define the coefficients c_i describing the distribution of x_s as

$$c_i = \int_0^1 \frac{1}{(n_s+1)} \left(\int_x^1 f_{\hat{a}}(\hat{a}) f_b\left(\frac{x}{\hat{a}}\right) \frac{1}{\hat{a}} d\hat{a} \right) g_{i,n_s}(x)\, dx.$$

The key to finding the distribution of x_s hence lies in solving the above integrals. First we expand brackets, rewriting the above to

$$c_i = \frac{1}{(n_s+1)} \int_0^1 \int_x^1 f_{\hat{a}}(\hat{a}) f_b\left(\frac{x}{\hat{a}}\right) \frac{1}{\hat{a}} g_{i,n_s}(x)\, d\hat{a}\, dx.$$

Then we interchange the integrals (using bounds $0 \leq x \leq \hat{a} \leq 1$), which results in

$$c_i = \frac{1}{(n_s+1)} \int_0^1 \int_0^{\hat{a}} f_{\hat{a}}(\hat{a}) f_b\left(\frac{x}{\hat{a}}\right) \frac{1}{\hat{a}} g_{i,n_s}(x)\, dx\, d\hat{a}.$$

Subsequently we apply a substitution, substituting x for $\hat{a}b$. Due to $dx = \hat{a}\, db$ we now find

$$c_i = \frac{1}{(n_s+1)} \int_0^1 \int_0^1 f_{\hat{a}}(\hat{a}) f_b(b) g_{i,n_s}(\hat{a}b)\, db\, d\hat{a}.$$

This looks a lot cleaner! But we can go further, inserting our earlier expression for $f_{\hat{a}}(\hat{a})$. This gives us

$$c_i = \frac{1}{(n_s+1)} \int_0^1 \int_0^1 \frac{1}{2\sqrt{\hat{a}}} f_a\left(\sqrt{\hat{a}}\right) f_b(b) g_{i,n_s}(\hat{a}b)\, db\, d\hat{a}.$$

Substituting $\hat{a} = a^2$ such that $d\hat{a} = 2a\, da$, the above turns into

$$c_i = \frac{1}{(n_s+1)} \int_0^1 \int_0^1 f_a(a) f_b(b) g_{i,n_s}(a^2 b)\, db\, da.$$

We will solve this integral in Section 4.4, but first we will look at it a bit more closely. There is actually a very nifty intuitive explanation behind this seemingly simple expression. It's similar to the thought experiment from Section 2.1. First, let's assume that we precisely know a and b. In that case we also know the probability $x = a^2 b$ that the exercise X was executed successfully. To approximate the distribution of x, we could do n_s experiments. Every experiment, there's a chance of $x = a^2 b$ on success and a corresponding chance $1 - a^2 b$ on failure. Using these n_s experiments, we then approximate the distribution of x. This gives the smoothened distribution of x_s.

Of course in our case we do not know a and b precisely. Instead, we only have distributions $f_a(a)$ and $f_b(b)$. So we integrate over all possibilities, with corresponding likelihoods, and add up the results. That's what the above integral comes down to.

3.4. ESTIMATING AN EXERCISE SUCCESS RATE: THE OR-OPERATOR

So far we have considered exercises X that required both skill A and skill B to be applied subsequently. Next, consider an exercise X that can be solved using either skill A or skill B: each of these skills can independently lead to the right solution. How do we find the distribution of x now, given the distributions of a and b?

You might argue, "Students pick the method (A or B) which they are best at, so take the highest!" (That is, apply a maximum operator.) This is actually wrong, because students can also apply both methods and check their results by comparing the outcomes. They're only likely to get X wrong when they fail both A and B. (If they only fail one, they can compare results to find their error and still get X right.) Using this logic, we find that the chance of failure is

$$(1 - x) = (1 - a)(1 - b).$$

We could rewrite this as $x = a + b - ab$ and use this to find the distribution of x. That is, apply

$$c_i = \frac{1}{(n_s+1)} \int_0^1 \int_0^1 f_a(a) f_b(b) g_{i,n_s}(a + b - ab)\, db\, da.$$

This is possible, but there is also an easier and more elegant method. We could define the inverses $\check{x} = 1 - x$, $\check{a} = 1 - a$ and $\check{b} = 1 - b$, according to the notation from Section 2.4. We then have $\check{x} = \check{a}\check{b}$, which comes down to an *and*-operator. So we can flip the distributions of a and b, use our method from Section 3.2 to find the distribution of $\check{x}$, and flip this to find the distribution of x. That's much simpler! Although in practice we actually do use the first method, because it is more generally applicable.

3.5. THE SETUP OF AN EXERCISE: A POSSIBLE COMBINATION OF OPERATORS

So far we have seen multiple ways to combine skills:

- A not operator is an inversion: $\text{not}(A)$ gives $1 - a$. (See the end of Section 2.4.)
- An and operator is a multiplication: $\text{and}(A, B)$ gives ab.
- An or operator is an inversion of multiplications of inverses: $\text{or}(A, B)$ is $\text{not}\big(\text{and}\big(\text{not}(A), \text{not}(B)\big)\big)$.

Things get more interesting when we start combining operators. Suppose that an exercise requires a student to first execute skill A and then either skill A or B. So the second exercise step can be solved in two ways: through A or through B. We write this as

$$X = \text{and}\big(A, \text{or}(A, B)\big).$$

The above expression is called the *setup* of the exercise: what needs to be done to solve it? From it, we can derive the success probability as

$$x = a(a + b - ab) = a^2 + ab - a^2 b.$$

This expression is called the *probability polynomial*. Using the rules defined above, we can turn any exercise setup into a probability polynomial. Once we have this polynomial, there are two questions that we can ask.

The first question is: what is the expected value of x? This is actually easy to calculate: we directly apply the equations from Section 1.5 to find $E[x]$. We always assume independence between variables, so we would have

$$E[x] = E[a^2] + E[a]E[b] - E[a^2]E[b].$$

Note that $E[a^2]$ is *not* the same as $E[a]^2$ because a is correlated with itself. This is also why it is wrong to first find the distribution of $Y = \text{or}(A, B)$ and then use $X = \text{and}(A, Y)$: it would not detect the correlation between a and y (which certainly is present, because y directly depends on a) and hence would give a wrong result.

The second question is: what is the distribution of x? To approximate this distribution, we use the final equation from the previous sections,

$$c_i = \frac{1}{(n_s+1)} \int_0^1 \int_0^1 f_a(a) f_b(b) g_{i,n_s}(a^2 + ab - a^2 b)\, db\, da.$$

Of course, instead of $a^2 + ab - a^2 b$, we insert whatever probability polynomial that we are working with. And if there are multiple parameters inside the probability polynomial, then we also use multiple integrals, just as many as there are parameters. The main challenge is solving these integrals to find the coefficients c_i. Once we have that, we have the distribution of x. How to solve these integrals is the subject of the next Section.

3.6. SOLVING THE INTEGRAL FOR THE COEFFICIENTS

Previously saw that, at some point in time, we may need to solve the integral

$$c_i = \frac{1}{(n_s+1)} \int_0^1 \int_0^1 f_a(a) f_b(b) g_{i,n_s}(a^2 + ab - a^2 b)\, db\, da,$$

or something similar. How can we do that? Per definition, we have

$$g_{i,n_s}(x) = (n_s+1)\binom{n_s}{i} x^i (1-x)^{n_s-i}.$$

This turns the above integral into

$$c_i = \binom{n_s}{i} \int_0^1 \int_0^1 f_a(a) f_b(b) (a^2 + ab - a^2 b)^i \big(1 - (a^2 + ab - a^2 b)\big)^{n_s-i}\, db\, da.$$

Assuming n_s isn't too large (which it usually isn't) we can subsequently apply a binomial expansion to expand the brackets. If we do this, we eventually get an expression of the form

$$c_i = \binom{n_s}{i} \int_0^1 \int_0^1 f_a(a) f_b(b) \sum_{i,j}^{n} k_{i,j} a^i b^j\, db\, da,$$

for some coefficients $k_{i,j}$ which follow from the bracket expansion. (If there are more than two parameters, then k will also have more than two subscripts.) Because a and b are independent, we can split up the above integrals according to

$$c_i = \binom{n_s}{i} \sum_{i,j}^{n} k_{i,j} \left(\int_0^1 a^i f_a(a)\, da \right) \left(\int_0^1 b^j f_b(b)\, db \right).$$

Now we see that the terms between brackets are simply the moments of a and b. That is,

$$c_i = \binom{n_s}{i} \sum_{i,j}^{n} k_{i,j} E[a^i] E[b^j].$$

We can calculate these moments using the equation from Section 1.5. (Preferably applying some memorization technique to not have to calculate them multiple times.) Afterwards, we apply the above sum to find the coefficients c_i, which results in the distribution of x_s that we wanted to find.

Note that this strategy always works, also when there are even more parameters involved. (Think of $x = ab + a^2 c + abd$ or something similar.) The key is to properly keep track of the coefficients $k_{i,j,\dots}$. Also note that, for multiple such parameters and for a high order n_s, this does become computationally intensive. As a result, it is wise to limit n_s to small numbers. In practice we use $n_s = 4$ which is still manageable. If the computations do become too intensive, it is also possible to find optimizations. For instance, when evaluating or-operators, to use inverse coefficient arrays instead when possible, as it cuts back on the number of terms.

3.7. MERGING DISTRIBUTIONS

Let's take a brief look at what we have achieved. If we have an exercise X that requires a student to do various steps, like apply skill A twice and skill B once (or any other combination of and/or-operators) then we can find the resulting distribution of x. This distribution is once more described by a coefficient array. That is,

$$f_x(x|D_s) = \sum_{i=0}^{n_s} c_i^s \, g_{i,n_s}(x).$$

The data D_s denotes all data directly related to subskills A and B. We use a subscript s here because this is also used for the other parameters, and it also corresponds to the word "subskills". Although we do omit the subscript s for x_s for reasons that will soon become apparent.

Now suppose that this exercise X also corresponds to a skill X of which we have separate data D_x on the performance of the student. For instance, the student already tried this skill a few times in the past. That is, we have another distribution dedicated to skill X, described by

$$f_x(x|D_x) = \sum_{i=0}^{n_x} c_i^x \, g_{i,n_x}(x).$$

Both distributions are equally valid. They are simply based on different sets of data. How do we merge these distributions together? The theory on merging distributions based on different data sets says

$$f_x(x|D_s, D_x) = \frac{p(D_s, D_x|x) f_x(x)}{p(D_s, D_x)} = \frac{p(D_s|x) p(D_x|x) f_x(x)}{p(D_s, D_x)} = \frac{p(D_s) p(D_x)}{p(D_s, D_x)} \frac{f_x(x|D_s) f_x(x|D_x)}{f_x(x)}.$$

In our case we also have $f_x(x) = 1$. Furthermore, we can ignore the constant fraction $p(D_s)p(D_x)/p(D_s, D_x)$, since normalization will take care of the constants. As a result, we have

$$f_x(x|D_s, D_x) \sim \left(\sum_{i=0}^{n_s} c_i^s g_{i,n_s}(x) \right) \left(\sum_{i=0}^{n_x} c_i^x g_{i,n_x}(x) \right).$$

The $\sim$ sign indicates a proportionality, since we omitted a constant. Using the definition of g this expands into

$$f_x(x|D_s, D_x) \sim \left(\sum_{i=0}^{n_s} c_i^s (n_s + 1) \binom{n_s}{i} x^i (1-x)^{n_s - i} \right) \left(\sum_{i=0}^{n_x} c_i^x (n_x + 1) \binom{n_x}{i} x^i (1-x)^{n_x - i} \right).$$

Our goal is to write this as a polynomial with order $n_m = n_s + n_x$, such that

$$f_x(x|D_s, D_x) = \sum_{i=0}^{n_m} c_i^m (n_m + 1) \binom{n_m}{i} x^i (1-x)^{n_m - i}.$$

The key is finding these coefficients c_i^m. By expanding the brackets in the previous equation, and properly counting the powers, we can eventually find them as

$$\begin{aligned} c_i^m &= \sum_{j=\max(0, i-n_s)}^{\min(i, n_x)} \frac{\left(c_{i-j}^s (n_s + 1) \binom{n_s}{i-j} \right) \left(c_j^x (n_x + 1) \binom{n_x}{j} \right)}{(n_m + 1) \binom{n_m}{i}} \\ &= \frac{(n_s + 1)(n_x + 1)}{(n_m + 1)} \sum_{j=max(0, i-n_s)}^{min(i, n_x)} c_{i-j}^s c_j^x \frac{\binom{n_s}{i-j} \binom{n_x}{j}}{\binom{n_m}{i}} \\ &= \frac{(n_s + 1)(n_x + 1)}{(n_m + 1)} \sum_{j=max(0, i-n_s)}^{min(i, n_x)} c_{i-j}^s c_j^x \frac{\binom{i}{j} \binom{n_s + n_x - i}{n_x - j}}{\binom{n_s + n_x}{n_x}}. \end{aligned}$$

Each one of the above three expressions can be used. The resulting coefficients do not automatically have a sum equal to 1, so we should hence normalize them afterwards. Note that the constant multiplication terms left of the summation sign can hence be ignored. This results in a distribution for x based on both data sets D_s and D_x. We call this operation the "merging" of distributions, which is where the subscript m comes from.

When working with distributions, it is important to always keep track of on which data a distribution is based on. Is the distribution based on only direct data on skill x? (That is, data set D_x.) Is it based on data from subskills? (That is, data set D_s.) Or is it based on data from both data sets, and hence is a merged distribution? In general the end user would want to get the merged distribution, considering it contains most data, so for the end user this distinction is not so important. Internally, however, it is crucial.

3.8. ESTIMATING A SKILL SUCCESS RATE: PICK AND PART OPERATORS

So far we have had three operators: not, and and or. To predict exercise success rates, this is all there is to it! For instance, an exercise like "calculate $3 \cdot 9 + 8 \cdot 7$" *always* requires two multiplications and a summation.

However, the skill "calculate composite sums" may sometimes also have exercises like "calculate $3 \cdot 9 + 56$". This exercise requires only one multiplication and two summations! Or it may have exercises like "calculate $3 \cdot 9 - 8 \cdot 7$". This time we don't require summation but subtraction! So skills have more variation in their setup than exercises do. How do we deal with this?

The solution is: we define two more operators. The peculiar thing about these operators is that they may *only* exist *inside* an and or or operator. Without that operator around it they have no meaning. You will soon see why.

The pick operator

Suppose that the skill "calculate composite sums" (skill X) always requires two multiplications (skill A) but then requires summation (skill B) half the time and subtraction (skill C) the other half of the time. In this case, we can write the skill setup as

$$X = \text{and}\big(A, A, \text{pick}(B, C)\big).$$

The corresponding probability polynomial is then

$$x = \frac{1}{2}a^2b + \frac{1}{2}a^2c.$$

This is one example usage of the pick operator. More generally, the pick operator is defined as

$$\text{pick}([\text{skills}], \text{numToPick}, [\text{weights}]).$$

So out of a list of skills, we pick a certain number of skills (default 1), with the given weights (default all equal). For instance, if we write

$$X = \text{pick}([A, B, C], 2, [2,3,4])$$

then we pick two skills from the list (never repeating) and the combinations $[A, B]$, $[A, C]$ and $[B, C]$ each have a chance of $6/26$, $8/26$ and $12/26$, respectively, to be chosen. Note that each of A, B and C here can be another combination of skills, like $\text{and}(D, E)$.

But what does such a combination mean? This depends on whether the pick operator is inside an and or or operator. If it's inside an and operator, then these two skills are taking along in this and operator. So if we have the skill setup

$$X = \text{and}(\text{pick}([A, B, C], 2, [2,3,4]), D)$$

then the resulting probability polynomial will be

$$x = \left(\frac{3}{13}ab + \frac{4}{13}ac + \frac{6}{13}bc\right)d.$$

However, if we would place the pick operator inside an or operator, then the skills are processed along with this or operator. That is, if we would use

$$X = \text{or}(\text{pick}([A, B, C], 2, [2,3,4]), D)$$

then the probability polynomial would be

$$x = 1 - \left(\frac{3}{13}(1-a)(1-b) + \frac{4}{13}(1-a)(1-c) + \frac{6}{13}(1-b)(1-c)\right)(1-d).$$

Of course we would simplify this further before any additional processing. In an identical way can any pick operator be turned into a polynomial, which is then further processed according to the earlier established methods.

The part operator

Suppose that half of the time, the skill "calculate composite sums" (skill X) requires two multiplications (skill A) and one summation (skill B), but the other half of the time it requires one multiplication and one summation. We could implement this as

$$X = \text{pick}([\text{and}(A, B), \text{and}(A, A, B)]).$$

However, we could implement it more cleanly with the part operator, using

$$X = \text{and}(A, \text{part}(A, 1/2), B).$$

This operator ensures that half of the time it is ignored and half of the time it is used.

More generally, the syntax is

$$\text{part}(\text{skill}, \text{part}).$$

The part must be a number between 0 and 1 (inclusive, although it's pointless to use 0 or 1). By default it's $1/2$.

To reduce this operator to a probability polynomial, we need to check which operator it's surrounded by.

- If surrounded by an and operator: $\text{part}(A, p)$ reduces to $1 - p(1-a)$.
- If surrounded by an or operator: $\text{part}(A, p)$ reduces to pa.

So for the previous example, we would have

$$x = a\big(1 - p(1-a)\big)b.$$

Note that this equals

$$x = pa^2b + (1-p)ab,$$

as it should. The mathematical reductions work.

To summarize: we have two additional operators, and both can be reduced to a probability polynomial for further mathematical processing. So they can be practically implemented in an algorithm. Do note that these last two operators (pick and part) are *non-deterministic*. They may therefore not be used for exercise setups (exercises always have definite steps to follow) but they are used a lot for skill setups.

3.9. INCORPORATING LINKED SKILL GROUPS

When setting up a learning tree, you will see a variety of skills. Sometimes skills are simply different. (Like multiplication and addition.) Sometimes skills are similar. (Like addition and subtraction.) And sometimes skills are so similar they count as the same skill. (Like adding numbers in the range 100-1000 and adding numbers in the range 1000-10000.) If skills are clearly different, or clearly the same, everything is fine. The tricky part is what to do when skills are similar.

Linking two skills

Consider two skills A and B that are similar but not the same. It could in theory be that a person is good at A and bad at B, but it is unlikely. Most likely, if a student practiced one skill a lot, there is a carry-over effect on the other skill. That is, the skills are correlated/linked. If that is the case, how can we mathematically take care of this?

The most sensible idea here would be to do the same thing as we did in Section 2.1: we set up a joint prior to indicate the correlation. The joint prior for a and b would hence become

$$f_{a,b}(a,b) = \frac{1}{(n_l+1)}\sum_{i=0}^{n_l} g_{i,n_l}(a)g_{i,n_l}(b),$$

where n_l is the order of the link. A higher order indicates more correlation, and hence a stronger link. Usually n_l is pretty low, since if the link was strong the skills would be merged into one single skill. Through the above prior, the two skills now provide data on each other.

To see how this works, consider the case where we have data D_a on skill A and data D_b on skill B. The posterior joint distribution, with this data, can be found through Bayes' law to equal

$$f_{a,b}(a,b|D_a,D_b) = \frac{p(D_a,D_b|a,b)f_{a,b}(a,b)}{p(D_a,D_b)} = \frac{p(D_a|a)p(D_b|b)f_{a,b}(a,b)}{p(D_a,D_b)}.$$

Applying Bayes' law once more on the terms $p(D_a|a)$ and $p(D_b|b)$ turns this into

$$f_{a,b}(a,b|D_a,D_b) = \frac{p(D_a)p(D_b)}{p(D_a,D_b)}\frac{f_a(a|D_a)}{f_a(a)}\frac{f_b(b|D_b)}{f_b(b)}f_{a,b}(a,b).$$

Note that many terms in the above expression are constant, such that we can write

$$f_{a,b}(a,b|D_a,D_b) \sim f_a(a|D_a)f_b(b|D_b)f_{a,b}(a,b).$$

If we subsequently want to find the posterior distribution for a, we apply marginalization through

$$f_a(a|D_a,D_b) \sim \int_0^1 f_{a,b}(a,b|D_a,D_b)\,db = f_a(a|D_a)\int_0^1 f_b(b|D_b)f_{a,b}(a,b)\,db.$$

This equation actually only contains things we have seen before. That is, first we calculate

$$f_a(a|D_b) = \int_0^1 f_b(b|D_b) f_{a,b}(a,b)\, db$$

which in fact is just a smoothened variant of the distribution of b, with the order n_l. Secondly, we merge this distribution with the distribution $f_a(a|D_a)$ of a based only on its own data. (Possibly this also includes data from the subskills constituting skill A.) That is, we use

$$f_a(a|D_a, D_b) = f_a(a|D_a) f_a(a|D_b).$$

Afterwards, as always with merging, we normalize coefficients.

Linking multiple skills

Is it also possible to set up multiple links? For instance three? The answer is: yes, but with restrictions. We could set up a group of three skills A, B and C with joint prior

$$f_{a,b,c}(a,b,c) = \frac{1}{(n_l+1)} \sum_{i=0}^{n_l} g_{i,n_l}(a) g_{i,n_l}(b) g_{i,n_l}(c),$$

where each link has the same order. In this case, the posterior joint distribution will be

$$f_{a,b,c}(a,b,c|D_a,D_b,D_c) = \frac{p(D_a)p(D_b)p(D_c)}{p(D_a,D_b,D_c)} \frac{f_a(a|D_a)}{f_a(a)} \frac{f_b(b|D_b)}{f_b(b)} \frac{f_c(c|D_c)}{f_c(c)} f_{a,b,c}(a,b,c).$$

Marginalization over b and c turns this into

$$f_a(a|D_a,D_b,D_c) \sim f_a(a|D_a) \int_0^1 \int_0^1 f_b(b|D_b) f_c(c|D_c) f_{a,b,c}(a,b,c)\, dc\, db = f_a(a|D_a) f_a(a|D_b,D_c).$$

But what is this linked distribution $f_a(a|D_b,D_c)$? What is the double integral? Expanding $f_{a,b,c}(a,b,c)$ gives

$$f_a(a|D_b,D_c) \sim \int_0^1 \int_0^1 f_b(b|D_b) f_c(c|D_c) \left(\sum_{i=0}^{n_l} g_{i,n_l}(a) g_{i,n_l}(b) g_{i,n_l}(c) \right) dc\, db.$$

Note that $1/(n_l+1)$ got dropped because of proportionality. Shuffling the order of terms, this turns into

$$f_a(a|D_b,D_c) \sim \sum_{i=0}^{n_l} g_{i,n_l}(a) \left(\int_0^1 f_b(b|D_b) g_{i,n_l}(b)\, db \right) \left(\int_0^1 f_c(c|D_c) g_{i,n_l}(c)\, dc \right).$$

If we define the coefficients

$$c_i^l = \left(\int_0^1 f_b(b|D_b) g_{i,n_l}(b)\, db \right) \left(\int_0^1 f_c(c|D_c) g_{i,n_l}(c)\, dc \right),$$

and normalize them afterwards, then we have a distribution of

$$f_a(a|D_b,D_c) = \sum_{i=0}^{n_l} c_i^l g_{i,n_l}(a).$$

This is the usual form. Afterwards we merge this together with $f_a(a|D_a)$, being the distribution of a based on its own data, and the link is complete! Except what are the integrals in the expression for c_i^l? Well, if we compare their expressions to what we saw in Section 2.6, then those are simply the coefficients of the distributions of b

and c, smoothened with order n_l. That is, if we smoothen the distributions of b and c with order n_l, then we get coefficient arrays with coefficients $c_i^{l,b}$ and $c_i^{l,c}$, after which we can write

$$c_i^l = c_i^{l,b} \cdot c_i^{l,c}.$$

That is, we multiply the coefficient vectors element-wise! This gives us the coefficients c_i^l which describe $f_a(a|D_b, D_c)$. Afterwards, we merge this with $f_a(a|D_a)$ to get the posterior distribution of a taking into account links $f_a(a|D_a, D_b, D_c)$.

So where is the caveat? Well, the problem is that the joint prior $f_{a,b,c}(a, b, c)$ only supports links in which all links within this group have the same order. This has as result that the vectors $\boldsymbol{c^{l,b}}$ and $\boldsymbol{c^{l,c}}$ need to be of the same size. If you want skill A to be strongly linked with skill B but weakly with skill C, then this is more difficult. It also complicates the situation: how strongly would B be linked with C? A work-around would be to use different smoothing orders, and then artificially increase the size of the smaller vectors to match those of the larger vectors using the techniques from Section 2.4. Still, the mathematical basis for this method is rather iffy.

The linking procedure summarized

So to summarize, the procedure to find the posterior distribution, given linked skills, is the following.

- Find the distribution $f_a(a|D_a)$ of a based on its own data in the usual way.
- Do a check: how many links does A have?
    - One link with a skill B: find $f_a(a|D_b)$ by smoothening with order n_l the distribution of b.
    - Multiple links (skills $B, C,$ …) with the same order n_l: smoothen all distributions of $b, c,$ … with order n_l, element-wise multiply the coefficients, and use these coefficients to describe $f_a(a|D_b, D_c, \dots)$.
    - Multiple links (skills $B, C, \dots$) with different orders (not recommended): smoothen all distributions of $b, c, \dots$, each with their own order. Increase the size of the corresponding coefficient vectors to match the largest size using the techniques from Section 2.4. Then multiply the coefficients element-wise, and use these coefficients to describe $f_a(a|D_b, D_c, \dots)$.
- Merge the two resulting distributions together to come up with the posterior distribution of a.

Using this strategy, as well as the rest of this chapter, we can calculate any posterior distribution for any skill.

4. UPDATING SKILL DATA: IMPLEMENTING OBSERVATIONS

In the previous chapter we only considered inference: given data on skills, how can we infer the success rate on exercises and on other skills? In this chapter we are also going to update said skill data. Given an observation, that a student succeeded or failed a certain exercise/skill, how do all distributions change?

4.1. THE UPDATE LAWS FOR A SIMPLE OBSERVATION

Consider an exercise X consisting of two subskills A and B. That is,

$$X = \text{and}(A, B).$$

Suppose that we know the distributions $f_a(a|D_a)$ and $f_b(b|D_b)$ based on prior performed exercises. Next, suppose the student does this exercise correctly or incorrectly. How would we update these distributions?

Let's call this new observation O_X. If the student does the exercise correctly (written as O_X^+) then finding $f_a(a|D_a, O_X)$ and $f_b(b|D_b, O_X)$ is relatively easy: because the student did the exercise correctly, they must have done both A and B correctly. However, if the student did not succeed at the exercise (written as O_X^-) then this is harder. We have to assign "blame": which skill was most likely to be wrong?

This "blaming" is actually done automatically if we look at the corresponding mathematics. First consider the posterior distribution of a and b, being $f_{a,b}(a, b|D_a, D_b, O_X)$. Using Bayes' law this can be written as

$$f_a(a, b|D_a, D_b, O_X) = \frac{p(O_X|a, b, D_a, D_b) f_{a,b}(a, b|D_a, D_b)}{p(O_X|D_a, D_b)} \sim p(O_X|a, b) f_a(a|D_a) f_b(b|D_b).$$

Note that we use proportionality because we normalize coefficients afterwards anyway. Also note that, if we know a and b, then having data D_a and D_b does not contribute anything extra. Finally note that we assume a and b, based on their own data, are independent.

To find the posterior distribution of a, we have to marginalize the above over b. This gives

$$f_a(a|D_a, D_b, O_x) = \int_0^1 f_a(a, b|D_a, D_b, O_X)\, db \sim \left(\int_0^1 p(O_X|a, b) f_b(b|D_b)\, db \right) f_a(a|D_a).$$

The main question now is, what is $p(O_X|a, b)$? That is, what is the probability, given a and b, that we did our observation O_X? Well, naturally this depends on what we observed.

If X was done correctly, then $p(O_X^+|a, b)$ is directly the probability polynomial $x = ab$! So then we have

$$f_a(a|D_a, D_b, O_x^+) \sim \left(\int_0^1 ab\, f_b(b|D_b)\, db \right) f_a(a|D_a) = aE[b] f_a(a|D_a) \sim a f_a(a|D_a).$$

Note that the integral just turns into an expectation operator on b. (And on any other possible parameter not equal to a.) Also note that in the end $E[b]$ dropped out because it is a constant. The above update is actually identical to the update if we just tested skill A and got that correct. So we can use the update from Section 2.5.

Things are different if X was done incorrectly. In this case $p(O_X^-|D_a, D_b, D_X) = 1 - x$. So it's one minus the probability polynomial. For our example $x = ab$ this becomes

$$f_a(a|D_a, D_b, O_x^-) \sim \left(\int_0^1 (1 - ab) f_b(b|D_b)\, db \right) f_a(a|D_a) = (1 - aE[b]) f_a(a|D_a).$$

There is an intuitive way to look at this. If $E[b]$ is very small ($E[b] \approx 0$) then our observation O_X^- is rather pointless when updating the distribution of a. Most likely the student did skill B incorrectly and failed as a result. It doesn't tell us much about skill A, and hence we don't update. However, if $E[b]$ is very large ($E[b] \approx 1$) then our observation is very useful. Most likely the student did skill B correctly, and the fault hence lies at skill A. We should update skill A as if the student failed at A, which is exactly what the above update law comes down to.

4.2. A GENERAL UPDATE STRATEGY

Let's consider things more generally. Let's consider an exercise X with corresponding probability polynomial x. In this case we have the following update law on a correct execution of X.

$$f_a\left(a \middle| D_a, D_{b,c,\dots}, O_X^+\right) \sim E[x|a] f_a(a|D_a).$$

Here, the expectation $E[x|a]$ is the expectation over all other parameters $b, c, \dots$. Similarly, on a failed execution of X the update law becomes

$$f_a\left(a \middle| D_a, D_{b,c,\dots}, O_X^-\right) \sim E[1-x|a] f_a(a|D_a).$$

But how do we update this in practice? To see how that works, we consider a slightly more complex example. Consider an exercise with setup $X = \text{and}\big(A, \text{or}(A, B)\big)$. This exercise has a corresponding probability polynomial

$$x = a(a + b - ab) = a^2 + ab - a^2 b.$$

Assuming that this is the case, how could we incorporate a successful or not-so-successful execution of the exercise into the distributions of a and b? On a correct solution we have

$$f_a(a|D_a, O_X^+) \sim (a^2 + aE[b] - a^2 E[b]) f_a(a|D_a).$$

Similarly for an incorrect solution we have

$$f_a\left(a \middle| D_a, D_{b,c,\dots}, O_X^-\right) \sim (1 - a^2 - aE[b] + a^2 E[b]) f_a(a|D_a).$$

In both cases can we write the update law as a polynomial

$$f_a(a|D_a, D_{b,c,\dots}, O_x) \sim \left(c_{p_0} + c_{p_1} a + c_{p_2} a^2 + \cdots\right) f_a(a|D_a) = \left(\sum_{i=0}^{n_p} c_{p_i} a^i\right) f_a(a|D_a).$$

Here, n_p is the order of the probability polynomial with respect to a, and c_{p_i} are the corresponding coefficients, taking expectations of the other parameters. The remaining question is: how should we update the coefficients?

4.3. REWRITING THE PROBABILITY POLYNOMIAL

There are two ways in which we can update the coefficients. We'll have a look at both, and then use the second.

Option one: using correct updates and coefficient vector size increases

We could try to implement the update law directly. After all, we know the coefficients of $c_{p_0} f_a(a|D_a)$ are proportional to the coefficients of $f_a(a|D_a)$. Similarly, the coefficients of $c_{p_1} a f_a(a|D_a)$ are proportional to the coefficients of the distribution of a after we've incorporated one correct execution, which we can calculate using Section 2.5. So what we could to is the following.

- Find the coefficients for the distributions $f_a(a|D_a)$, $af_a(a|D_a)$, $a^2 f_a(a|D_a)$, ….
- Use the coefficient vector size increase trick from Section 2.4 to make sure these vectors all have the same size.
- Use the probability polynomial coefficients c_{p_i} to add them together.

This is actually not a correct method, because the update method from Section 2.5 also has various constants that are ignored, and these *should* be taken into account this time. We could in theory take them into account, but that will be a mathematical hassle. Instead, the following option is a lot easier.

Option two: rewriting the probability polynomial

Instead of writing the probability polynomial like $x(a) = c_{p_0} + c_{p_1}a + c_{p_2}a^2 + c_{p_3}a^3$ (for a third order probability polynomial, so with $n_p = 3$) we will instead write it as

$$x(a) = c'_{p_0}(1-a)^3 + 3c'_{p_1}a(1-a)^2 + 3c'_{p_2}a^2(1-a) + c'_{p_3}a^3.$$

Or, more generally, we will write it as

$$x(a) = \sum_{i=0}^{n_p} c'_{p_i}\binom{n_p}{i}a^i(1-a)^{n_p-i}.$$

Note that, for the third order polynomial, we could also write it as

$$x(a) = c'_{p_0} - 3\left(c'_{p_0} - c'_{p_1}\right)a + 3\left(c'_{p_0} - 2c'_{p_1} + c'_{p_2}\right)a^2 - \left(c'_{p_0} - 3c'_{p_1} + 3c'_{p_2} - c'_{p_3}\right)a^3.$$

From this, we see the generic pattern that

$$c_{p_i} = (-1)^i\binom{n_p}{i}\sum_{j}^{i}(-1)^j\binom{i}{j}c'_{p_i}.$$

However, we do not want to find the coefficients c_{p_i} from the coefficients c'_{p_i}. Instead, we want to do the reverse. For the third order this becomes

$$x(a) = c_{p_0}(1-a)^3 + \left(3c_{p_0} + c_{p_1}\right)(1-a)^2 + \left(3c_{p_0} + 2c_{p_1} + c_{p_2}\right)a^2(1-a) + \left(c_{p_0} + c_{p_1} + c_{p_2} + c_{p_3}\right)c'_{p_3}a^3.$$

With some puzzling, we can find that the coefficients c'_{p_i} satisfy

$$\binom{n_p}{n_p - i}c'_{p_i} = \sum_{j=0}^{i}\binom{n_p - j}{n_p - i}c_{p_j}.$$

This allows us to turn a probability polynomial using the standard notation into a probability polynomial using the alternative notation. With it, we can update the coefficients of the distribution $f_a(a)$. Let's examine how.

4.4. UPDATING THE COEFFICIENTS

Using our new way of writing the probability polynomial, we have transformed our update law into

$$f_a(a|D_a, D_{b,c,\dots}, O_x) \sim \left(\sum_{i=0}^{n_p} c'_{p_i}\binom{n_p}{i}a^i(1-a)^{n_p-i}\right)f_a(a|D_a).$$

Note that $f_a(a|D_a)$ can be written through its own coefficients c_i as

$$f_a(a|D_a) = \sum_{i=0}^{n} c_i \cdot g_{i,n}(a) = \sum_{i=0}^{n} c_i (n+1) \binom{n}{i} a^i (1-a)^{n-i}.$$

Combining these expressions gives

$$f_a(a|D_a, D_{b,c,\dots}, O_x) \sim \left(\sum_{i=0}^{n_p} c'_{p_i} \binom{n_p}{i} a^i (1-a)^{n_p-i} \right) \left(\sum_{i=0}^{n} c_i \binom{n}{i} a^i (1-a)^{n-i} \right).$$

Note that we omitted the constant $(n+1)$ because of proportionality. We want to merge parts with the same power together. To do so, we first pull the sums together to get

$$f_a(a|D_a, D_{b,c,\dots}, O_x) \sim \sum_{i=0}^{n} \sum_{j=0}^{n_p} c_i c'_{p_j} \binom{n}{i} \binom{n_p}{j} a^{i+j} (1-a)^{n+n_p-i-j}.$$

By reordering the summation, and by turning i into $i-j$, we can write the above as

$$f_a(a|D_a, D_{b,c,\dots}, O_x) \sim \sum_{i=0}^{n+n_p} \left(\sum_{j=\max(0,i-n)}^{\min(n_p,i)} c_{i-j} c'_{p_j} \binom{n}{i-j} \binom{n_p}{j} \right) a^i (1-a)^{n+n_p-i}.$$

In other words, if we define the updated coefficients as

$$c_i^* = \sum_{j=\max(0,i-n)}^{\min(n_p,i)} c_{i-j} c'_{p_j} \frac{\binom{n}{i-j}\binom{n_p}{j}}{\binom{n+n_p}{i}},$$

and subsequently define the posterior order $n_* = n + n_p$, then the resulting distribution can be written as

$$f_a(a|D_a, D_{b,c,\dots}, O_x) \sim \sum_{i=0}^{n+n_p} c_i^* (n_*+1) \binom{n_*}{i} a^i (1-a)^{n_*-i} = \sum_{i=0}^{n+n_p} c_i^* \cdot g_{i,n_*}(a).$$

Note that we have introduced a constant (n_*+1) which is allowed because of proportionality.

The above result is great, but we can go one step further. We can rewrite the expression for c_i^* as

$$c_i^* = \sum_{j=\max(0,i-n)}^{\min(n_p,i)} c_{i-j} c'_{p_j} \frac{\binom{i}{j}\binom{n+n_p-i}{n_p-j}}{\binom{n+n_p}{n}}.$$

Note that the denominator of the above fraction is a constant, so it drops out too at normalization! Hence, the simplified version of the coefficient update law is

$$c_i^* = \sum_{j=\max(0,i-n)}^{\min(n_p,i)} c_{i-j} c'_{p_j} \binom{i}{j}\binom{n+n_p-i}{n_p-j}.$$

This is a very simple way of getting from the coefficients c_i of the previous distribution, and the coefficients c'_{p_j} of the probability polynomial, to the updated coefficients c_i^*!

4.5. UPDATING BOTH SUBSKILLS AND PARENT SKILLS

Suppose that we have an exercise that requires as steps the execution of skill A and skill B. That is,

$$X = \text{and}(A, B).$$

If a student solves this exercise, we naturally need to update the distributions of a and b. But now suppose that this exercise also represents a skill X. So solving the exercise is also an indication that the student has mastered skill X correctly. How would this update work? Which would we update?

The solution here is a bit rudimentary, but in practice it works well. It says: just update both.

- Update skills A and B using the fact that an exercise with setup $\text{and}(A, B)$ was completed successfully. This can be done using the techniques described in this chapter.
- Update skill X using the fact that an exercise directly related to that skill was completed successfully. This can be done using the technique from Section 2.5, or just using Section 4.4 with a probability polynomial x. After all, Section 4.4 is a generalization of Section 2.5.

Naturally things happen identically on an unsuccessful completion. By setting it up in this way, we incorporate our observations both into the distributions of a and b, as well as in the distribution of x. It ensures observation have an actual effect on all relevant distributions.

With this note, we have completed our machine learning algorithm. We can incorporate any kind of observation, and predict any kind of success rate.

5. SUMMARY: AN OVERVIEW OF HOW TO DO EVERYTHING

In this final chapter we give an overview of how to calculate every distribution and update it.

The main idea is that, for every skill A, B or X, we always have a distribution like $f_a(a)$ stored. This distribution is stored using a coefficient vector $\boldsymbol{c}$. The stored coefficient vector always describes the posterior probability that a student did the *previous* exercise correctly. So this includes the knowledge about how the student did said previous exercise. There are now a few things that we can do.

Find the distribution of a skill success rate based on its own data

Before we do anything, we always want to find the distribution of a skill based on its own data, noted as $f_x(x|D_x)$, for every related skill. To do so, apply the following steps.

- Take the coefficient vector that is stored.
- Prepare for smoothing: find the smoothing order according to Sections 2.7 and 2.8.
- Apply the smoothing: use the relations from Section 2.6.

This results in the distribution of a skill based on its own data, corresponding to the current time. That is, it's smoothened based on what day it is today, meaning the forget-factor has been implemented. We call this the skill's *own distribution*.

Add possible links with other skills

If the skill X is linked with skills Y and Z, then do the following.

- Find the own distributions for skills Y and Z
- Find the linked distribution $f_x(x|D_y, D_z, \dots)$ using Section 3.9.
- Merge this linked distribution together with $f_x(x|D_x)$.

This gives us a distribution $f_x(x|D_x, D_y, D_z)$ which also takes into account links. The result of this is known as the *linked distribution*. Note that, if a skill does not have links, the linked distribution equals the skill's own distribution.

Infer the distribution of a skill success rate

If a skill has a setup like $X = \text{and}(A, B)$, then we can go one step further. We can infer data from skills A and B to make the distribution of X more accurate. This goes as follows.

- Make sure we have the linked distributions for skills A and B.
- Find the probability polynomial using Sections 3.5 and possibly Section 3.8.
- Use Section 3.6 to infer a distribution $f_x(x|D_a, D_b)$ from the probability polynomial.
- Merge the result together with the skill's linked distribution $f_x(x|D_x, \dots)$.

This gives us the *inferred distribution* of the skill success rate $f_x(x|D_x, \dots, D_a, D_b)$. Afterwards, we can of course find $E[x]$ based on this distribution to determine the expected skill success rate.

Infer the distribution of an exercise success rate

There are various types of exercises.

- Basic exercises without steps and connected to a skill A without subskills. In this case:
    - Take this skill's linked distribution $f_a(a|D_a, \dots)$.

- Exercises without steps connected to a skill X with subskills. This usually doesn't happen, since if a skill has subskills, you can split the corresponding exercise up into steps, but in theory it's possible. This time:
    - Use the inferred distribution $f_x(x|D_x, \dots, D_a, D_b)$ of the skill success rate.
- Exercises with steps, and hence a setup, but not connected to a skill itself. For instance, the exercise setup might be $X_{ex} = \mathrm{and}(A, B)$. Now the steps are:
    - Find the inferred distributions for skills A and B.
    - Find the probability polynomial of X_{ex} using Sections 3.5 and possibly Section 3.8.
    - Use Section 3.6 to infer a distribution $f_x(x|D_a, D_b)$ from the probability polynomial.
- Exercises with steps, and hence a setup, and also connected to a skill X. In this case there are two conflicting setups: the setup of the exercise (for instance $X_{ex} = \mathrm{and}(A, B)$) and the setup of the skill X (for instance $X = \mathrm{and}(A, A, B)$). In this case it is assumed that the exercise setup is more accurate, since it is specified to the exercise at hand. The skill setup is ignored. So the steps are:
    - Find the inferred distributions for skills A and B.
    - Find the probability polynomial of X_{ex} using Sections 3.5 and possibly Section 3.8.
    - Use Section 3.6 to infer a distribution $f_x(x|D_a, D_b)$ from the probability polynomial.
    - Merge the result together with the skill's linked distribution $f_x(x|D_x, \dots)$.

This always gives us a distribution for the exercise success rate. Then, using the expectation $E[x]$ we can find the expected exercise success rate.

Performing updates

When observations are obtained, like when a student submits an exercise, all related distributions are updated. If the exercise is related to a single skill, and does not have any steps, then the update is simple: use Section 2.5 for a correct/incorrect update and store the resulting coefficient array in the database. Or use the generalization from Section 4.4

If the exercise does have a setup like $X = \mathrm{and}(A, B)$, then do the following.

- Make sure you have the own distributions for the related skills, here a and b.
- For each of these skills, do the following.
    - Set up the probability polynomial, where you take expectations over the other parameters, according to Section 4.2.
    - Transform the probability polynomial to the alternative notation, according to Section 4.3.
    - Update the coefficients using the update relation from Section 4.4.
- If the exercise is directly connected to a parent skill (see Section 4.5) then also update that parent skill using a simple correct/incorrect update (see Section 2.5) or using the generalization from Section 4.4.

In addition to storing the coefficients, also note the time at which this update was performed (being the time of the last exercise execution) and note the number of times the student has done said exercise.

This allows us to continuously update skill distributions and use them to make predictions about exercises and related skills.