Product Spaces

Michael Betancourt

April 2023

Table of contents

Sometimes a system of applied interest can be adequately modeled with a single space. When a system is too complicated for a single space we may be able to model it with *multiple spaces* at

the same time. *Product spaces* integrate multiple component spaces together by first combining both their underlying sets and their associated structures.

1 Product Sets

Product sets combine elements from multiple sets into *composite* elements. To develop this concept as cleanly as possible we'll first investigate how to combine elements from finite sets before considering the general case. Finally we'll investigate the behavior of subsets of these product sets.

1.1 Finite Product Sets

Consider two finite sets, one with three elements,

$$X_1 = \{\square, \clubsuit, \diamondsuit\},$$

and one with two elements,

$$X_2 = \{\heartsuit, \spadesuit\}.$$

One way to combine these two sets together is to collect their individual elements together into larger set,

$$X_1 \cup X_2 = \{\square, \clubsuit, \diamondsuit, \heartsuit, \spadesuit\}.$$

This concatenated set allows us to choose from any of the elements in X_1 and X_2, but we can only choose *only one element at a time*.

In order to choose elements from *both sets at the same time* we need to account for all of the possible pairs of elements from X_1 and X_2. This requires *replicating* one of the sets for each element in the other set.

For example we can replicate X_2 three times, one for each element of X_1, to give the three distinct sets

$$\{\square\} \times X_2 = \{\heartsuit, \spadesuit\}$$
$$\{\clubsuit\} \times X_2 = \{\heartsuit, \spadesuit\}$$
$$\{\diamondsuit\} \times X_2 = \{\heartsuit, \spadesuit\}.$$

To differentiate between the elements of these replications we can tag them with the corresponding element of X_1, such as

$$\{\square\} \times X_2 = \{\heartsuit_\square, \spadesuit_\square\}$$
$$\{\clubsuit\} \times X_2 = \{\heartsuit_\clubsuit, \spadesuit_\clubsuit\}$$
$$\{\diamondsuit\} \times X_2 = \{\heartsuit_\diamondsuit, \spadesuit_\diamondsuit\}.$$

or

$$\{\square\} \times X_2 = \{(\square, \heartsuit), (\square, \spadesuit)\}$$
$$\{\clubsuit\} \times X_2 = \{(\clubsuit, \heartsuit), (\clubsuit, \spadesuit)\}$$
$$\{\diamondsuit\} \times X_2 = \{(\diamondsuit, \heartsuit), (\diamondsuit, \spadesuit)\}.$$

Collecting the elements of these distinct, replicated sets together gives a set whose elements account for all of the possible pairings between the elements of X_1 and X_2,

$$\bigcup_{x_1 \in X_1} \{x_1\} \times X_2 = \{\square\} \times X_2 \cup \{\clubsuit\} \times X_2 \cup \{\diamondsuit\} \times X_2$$

$$= \{(\square, \heartsuit), (\square, \spadesuit), (\clubsuit, \heartsuit), (\clubsuit, \spadesuit), (\diamondsuit, \heartsuit), (\diamondsuit, \spadesuit)\}$$
$$\equiv X_1 \times X_2.$$

This new set $X_1 \times X_2$ is denoted a **product set** with **component sets** X_1 and X_2.

This construction, however, is not unique. An equivalent way to construct the product set of all pairs of elements in X_1 and X_2 is to replicate X_1 twice, one for each of the two elements in X_2. This gives two distinct sets

$$X_1 \times \{\heartsuit\} = \{\square, \clubsuit, \diamondsuit\}$$
$$X_1 \times \{\spadesuit\} = \{\square, \clubsuit, \diamondsuit\},$$

or explicitly differentiating the elements across the replications,

$$X_1 \times \{\heartsuit\} = \{(\square, \heartsuit), (\clubsuit, \heartsuit), (\diamondsuit, \heartsuit)\}$$
$$X_1 \times \{\spadesuit\} = \{(\square, \spadesuit), (\clubsuit, \spadesuit), (\diamondsuit, \spadesuit)\}.$$

Collecting these replicated elements together gives

$$\bigcup_{x_2 \in X_2} X_1 \times \{x_2\} = X_1 \times \{\heartsuit\} \cup X_1 \times \{\spadesuit\}$$
$$= \{(\square, \heartsuit), (\clubsuit, \heartsuit), (\diamondsuit, \heartsuit), (\square, \spadesuit), (\clubsuit, \spadesuit), (\diamondsuit, \spadesuit)\}$$

which is exactly the product set we constructed above! In other words we have two equivalent constructions of the product set (Figure 1),

$$X_1 \times X_2 = \bigcup_{x_1 \in X_1} \{x_1\} \times X_2$$
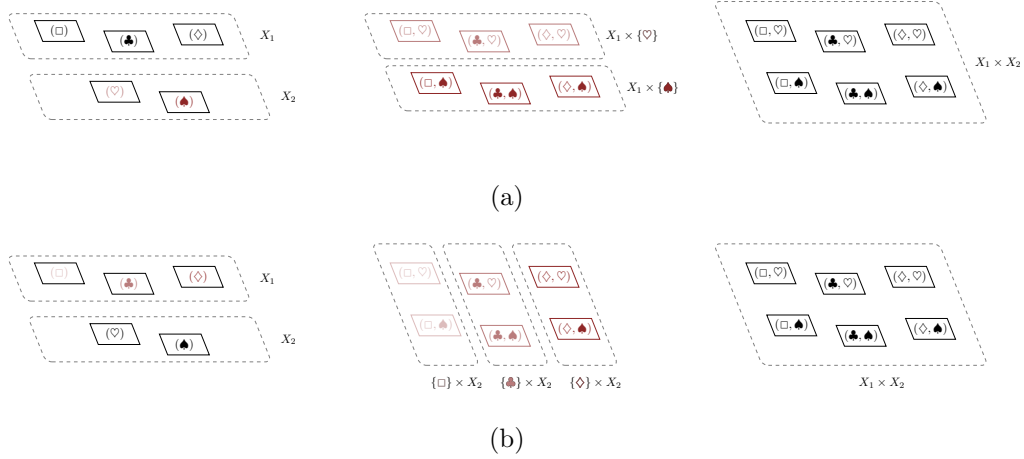$$= \bigcup_{x_2 \in X_2} X_1 \times \{x_2\}.$$

Figure 1: The finite product set $X_1 \times X_2$ can be constructed (a) by replicating the finite set X_2 once for each element of the finite set X_1 or (b) by replicating X_1 once for each element of X_2. Either way results in an equivalent set of pairs of elements selected from X_1 and X_2.

Each element of the product set is uniquely specified by one element of X_1 and one element of X_2. Consequently every variable taking values in the product set $x \in X_1 \times X_2$ is compromised of an ordered pair of variables from each component space,

$$x = (x_1, x_2),$$

with $x_1 \in X_1$ and $x_2 \times X_2$.
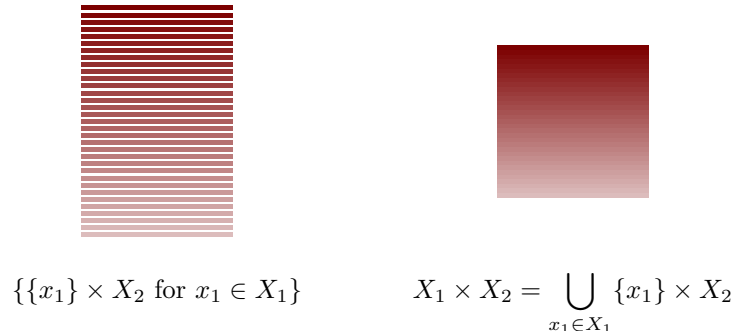
1.2 General Product Sets

This construction immediately generalizes beyond finite sets. Given two sets X_1 and X_2 we can construct the product set by replicating X_2 once for each element of X_1 and then combining the replicated elements together (Figure 2a)

$$X_1 \times X_2 = \bigcup_{x_1 \in X_1} \{x_1\} \times X_2,$$

or equivalently by replicating X_1 once for each element of X_2 and then combining the replicated elements together (Figure 2b),
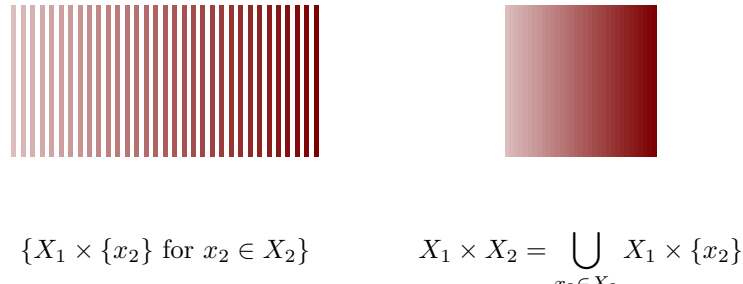
$$X_1 \times X_2 = \bigcup_{x_2 \in X_2} X_1 \times \{x_2\}.$$

The common result $X_1 \times X_2$ is denoted a **product set** with the **component sets** X_1 and X_2.

4

$$\{\{x_1\} \times X_2 \text{ for } x_1 \in X_1\} \qquad X_1 \times X_2 = \bigcup_{x_1 \in X_1} \{x_1\} \times X_2$$

(a)

$$\{X_1 \times \{x_2\} \text{ for } x_2 \in X_2\} \qquad X_1 \times X_2 = \bigcup_{x_2 \in X_2} X_1 \times \{x_2\}$$

(b)

Figure 2: A general product set with two components $X_1 \times X_2$ can be constructed (a) by replicating the set X_2 once for each element of the set X_1 or (b) by replicating X_1 once for each element of X_2. Both constructions give a set consisting of pairs of elements selected from X_1 and X_2.

As in the finite case every element of a general, binary product set is uniquely specified with a pair of elements, one from the first component set, $x_1 \in X_1$, and the other from the second component set, $x_2 \in X_2$. Because of this each variable taking values in the product set $x \in X_1 \times X_2$ is comprised of an ordered pair of component variables,

$$x = (x_1, x_2)$$

with $x_1 \in X_1$ and $x_2 \in X_2$.

Iterating this construction allows us to define product sets from more than two components sets. For example given three sets X_1, X_2, and X_3 we can construct the three-component product space $X_1 \times X_2 \times X_3$ in three equivalent different ways (Figure 3). We can construct the product set $X_1 \times X_2$ and then replicate it once for each element $x_3 \in X_3$ before combining those replications together,

$$X_1 \times X_2 \times X_3 = \bigcup_{x_3 \times X_3} X_1 \times X_2 \times \{x_3\}.$$

At the same time we can construct the product set $X_2 \times X_3$, replicate it once for each element $x_1 \in X_1$, and the aggregate the resulting elements,

$$X_1 \times X_2 \times X_3 = \bigcup_{x_1 \times X_1} \{x_1\} \times X_2 \times X_3.$$

Finally we can construct the product set $X_1 \times X_3$, replicate it once for each element $x_2 \in X_2$, and then aggregate,

$$X_1 \times X_2 \times X_3 = \bigcup_{x_2 \times X_2} X_1 \times \{x_2\} \times X_3.$$

All three constructions result in the same product space where each element is uniquely specified by a triplet of component variables,

$$x = (x_1, x_2, x_3).$$

More generally given I component sets

$$\{X_1, \ldots, X_i, \ldots X_I\}$$

we can construct a corresponding product set

$$X_1 \times \ldots \times X_i \times \ldots \times X_I = \times_{i=1}^I X_i$$

where every product variable $x \in \times_{i=1}^I X_i$ is compromised of a ordered collection of component variables

$$x = (x_1, \ldots, x_i, \ldots, x_I)$$

with $x_i \in X_i$. This ordered collection of component variables is known as an **n-tuple variable** or more compactly just an **n-tuple**, generalizing "couple" that denotes a pair of component
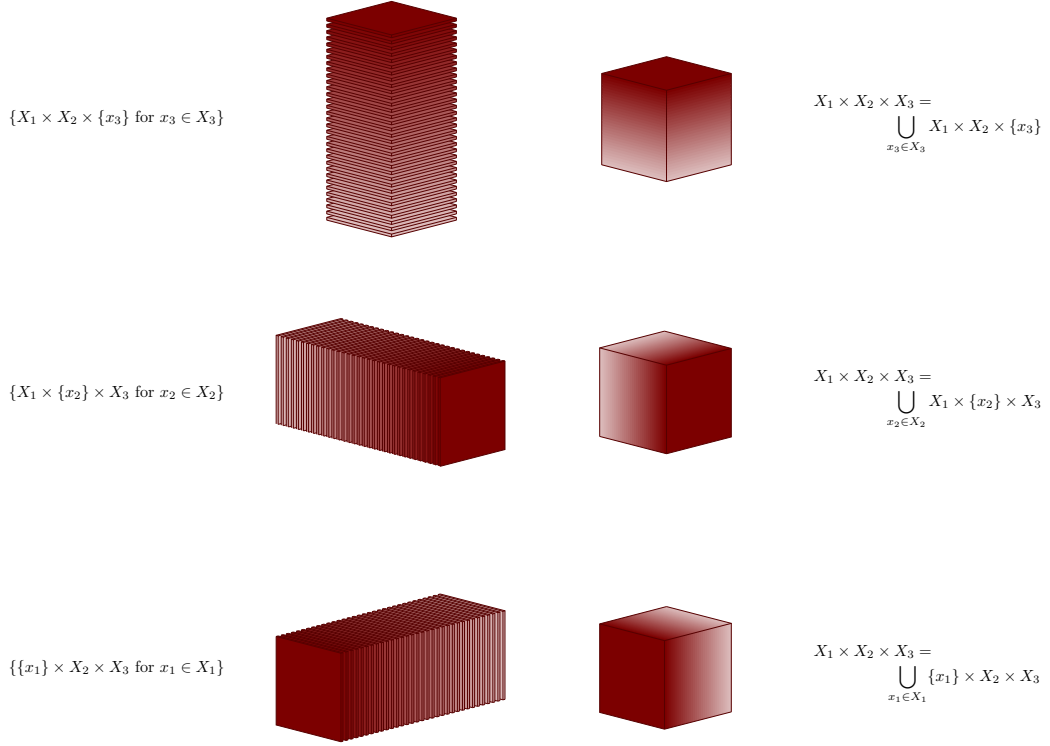
6

$\{X_1 \times X_2 \times \{x_3\}$ for $x_3 \in X_3\}$

$X_1 \times X_2 \times X_3 = \bigcup_{x_3 \in X_3} X_1 \times X_2 \times \{x_3\}$

$\{X_1 \times \{x_2\} \times X_3$ for $x_2 \in X_2\}$

$X_1 \times X_2 \times X_3 = \bigcup_{x_2 \in X_2} X_1 \times \{x_2\} \times X_3$

$\{\{x_1\} \times X_2 \times X_3$ for $x_1 \in X_1\}$

$X_1 \times X_2 \times X_3 = \bigcup_{x_1 \in X_1} \{x_1\} \times X_2 \times X_3$

Figure 3: The three component sets X_1, X_2, and X_3 can be combined into the product set $X_1 \times X_2 \times X_3$ in multiple ways For example we can first combine X_1 and X_2 into $X_1 \times X2$, replicate that binary product once for each element of X_3, and then merge those replications. Alternatively we can combine X_1 and X_3 first or X_2 and then X_3 first. All of these constructions results in a three-component product space $X_1 \times X_2 \times X_3$.

variables and "triple" that denotes three component variables to an arbitrary number of component variables. Because their elements are specified by multiple variables product sets are often referred to as **multivariate** sets.

When working with multiple product variables we might be inclined to use integer indices to differentiate between them. These indices, however, are readily confused with the indices used to denote the component variables. In circumstances where numbering product variables is useful I will use a comma to separate the indices with the first always referring to the product variable and the latter always referring to the component variable,

$$x_j = (x_{j,1}, \dots, x_{j,i}, \dots, x_{j,I}).$$

In other words $x_{j,i}$ refers to the ith component variable of the jth product variable.

1.3 Product Subsets

Just as elements of a product sets are uniquely specified with an element from each of the component sets, subsets of product sets are uniquely specified with a subset from each of the component sets. These product subsets $\mathsf{x} \subset \times_{i=1}^I X_i$ are often written as

$$\mathsf{x} = \mathsf{x}_1 \times \cdots \times \mathsf{x}_i \times \cdots \times \mathsf{x}_I = \times_{i=1}^I \mathsf{x}_i$$

where

$$\mathsf{x}_i \subset X_i.$$

The product power set is itself a product of the component power sets,

$$2^{\times_{i=1}^I X_i} = 2^{X_1} \times \dots \times 2^{X_i} \times \dots \times 2^{X_I} = \times_{i=1}^I 2^{X_i}.$$

The empty product set is given by aggregating the component empty sets,

$$\emptyset = \times_{i=1}^I \emptyset_i,$$

while the full product set is given by aggregating the component full sets,

$$X = \times_{i=1}^I X_i.$$

Set operations on product subsets are defined by applying them to each component subset individually and then combining the results. For example given the product subsets

$$\mathsf{x} = \times_{i=1}^I \mathsf{x}_i$$

and

$$\mathsf{x}' = \times_{i=1}^I \mathsf{x}'_i$$

we can define their union as the product subset

$$\mathsf{x} \cup \mathsf{x}' = \times_{i=1}^{I} \mathsf{x}_i \cup \mathsf{x}_i'$$

and their intersection as the product subset

$$\mathsf{x} \cap \mathsf{x}' = \times_{i=1}^{I} \mathsf{x}_i \cap \mathsf{x}_i'.$$

These product set operations also introduce a certain geometric intuition to product subsets. For example considering decomposing the product set $X_1 \times X_2$ into replications of X_1,

$$X_1 \times X_2 = \bigcup_{x_2 \in X_2} X_1 \times \{x_2\}.$$

Selecting the subset $\mathsf{x}_1 \subset X_1$ in every replication *lifts* x_1 into the product subset

$$\mathsf{x}_1 \times X_2 \subset X_1 \times X_2.$$

At the same time given the decomposition

$$X_1 \times X_2 = \bigcup_{x_1 \in X_1} \{x_1\} \times X_2$$

we can select the subset $\mathsf{x}_2 \subset X_2$ in every replication $\{x_1\} \times X_2$ to define the product subset

$$X_1 \times \mathsf{x}_2 \subset X_1 \times X_2.$$

The intersection of these two product subsets is then just the product of x_1 and x_2,

$$(\mathsf{x}_1 \times X_2) \cap (X_1 \times \mathsf{x}_2) = (\mathsf{x}_1 \cap X_1) \times (X_2 \cap \mathsf{x}_2)$$
$$= (\mathsf{x}_1) \times (\mathsf{x}_2).$$

In other words we can interpret the product subset $\mathsf{x}_1 \times \mathsf{x}_2$ as the overlap of the shadows that x_1 and x_2 cast into the full product set (Figure 4).
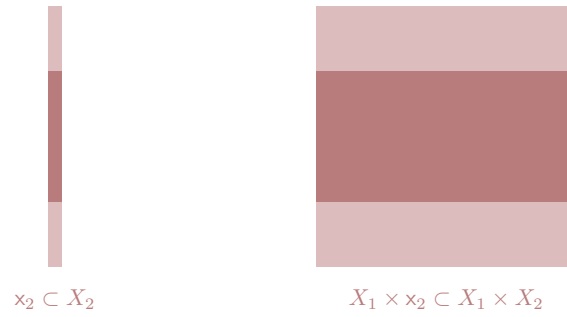
2 Product Structure

In order to elevate product *sets* to product *spaces* we need to construct product *structure* from any structure endowed onto the component sets. Conveniently these constructions are straightforward for our basic structures.
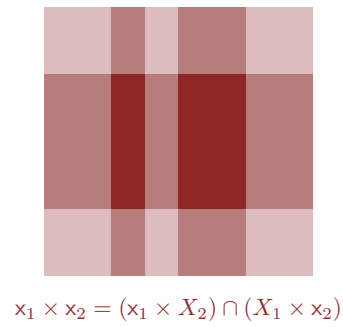
In this section I will always assume the ambient product set $X = \times_{i=1}^{I} X_i$ with component sets X_i.

(a)

(b)

$x_1 \times x_2 = (x_1 \times X_2) \cap (X_1 \times x_2)$
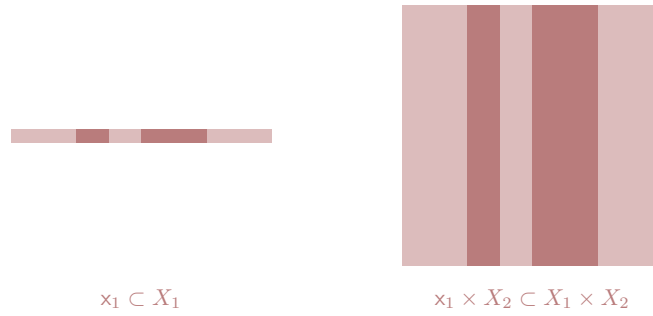
(c)

Figure 4: One way to interpret product sets is geometrically. From this perspective (a, b) we first lift each component set to the product space before (c) constructing the product set from their overlap.

2.1 Product Orderings

If each component set is equipped with a strict ordering then we can unambiguously define a product variable $x \in X$ to be smaller than another product variable $x' \in X$ if *all* of the component elements in x are smaller than *all* of the corresponding component elements in x'. In other words $x < x'$ if and only if

$$x_i < x'_i$$

for all $i \in 1, \dots, I$.

On the other hand if only *some* of the component elements in x are smaller than the corresponding component elements in x', while some are larger, then the comparison between the two product elements will be ambiguous. Consequently a strict ordering on the component sets does not fully define a strict ordering on the product set.

That said we can use the strict component orderings to define a partial ordering of the product set where any two product elements with mixed component orderings are arranged together. In the same way we can use partial component orderings to define a partial ordering of the product set. Either way component orderings will generally define partial product orderings.

2.2 Product Algebras

When each component set is equipped with an individual algebraic operation we can define a corresponding product operation by applying these component operations at the same time. For example if each component set is equipped with a binary operation

$$\begin{aligned}
\cdot_i : X_i \times X_i &\to X_i \\
x_i, x'_i &\mapsto x_i \cdot x'_i,
\end{aligned}$$

then we can construct a **product operation** $\cdot : X \times X \to X$ as

$$x \cdot x' = (x_1 \cdot_1 x'_1, \dots, x_i \cdot_i x'_i, \dots, x_I \cdot_I x'_I).$$

Product operations acquire any properties that are shared by *all* of the component operations. For example if all of the component operations are commutative then the product operation will also be commutative. Similarly if all of the component operations are unital with identity elements $x_{\mathrm{Id},i}$ then the product operation will also be unital with the composite identity element

$$x_{\mathrm{Id}} = (x_{\mathrm{Id},1}, \dots, x_{\mathrm{Id},i}, \dots, x_{\mathrm{Id},I}).$$

11

2.3 Product Metrics

Product metrics can be constructed by summing over the outputs of component metrics. More formally if each component set is equipped with a metric

$$d_i : X_i \times X_i \to \mathbb{R}^+$$
$$x_i, x_i' \quad \mapsto d(x_i, x_i')$$

then we can define a **product metric** as

$$d : X \times X \to \mathbb{R}^+$$

$$x, x' \quad \mapsto \sum_{i=1}^{I} d(x_i, x_i').$$

Critically this product metric acquires all of the properties required of a metric from the component metrics. By construction the product distances vanish if and only if all of the individual component distances vanish. This requires all of the component elements to be equal which implies that the product elements are also equal. In other words the product metric returns zero if and only if the two input product elements are the same, as required for a metric.

On the other hand two input product elements are distinct if and only if at least one of the component element has to be distinct. In this case at least one of the component distances will be greater than zero. Consequently the summed distances will be non-zero whenever the input product elements are distinct.

Commutativity of the product metric follows the commutativity of the component metrics,

$$d(x, x') = \sum_{i=1}^{I} d(x_i, x_i')$$
$$= \sum_{i=1}^{I} d(x_i', x_i)$$
$$= d(x', x).$$

Likewise for any three product elements $x, x', x'' \in X$ we always have a triangle inequality,

$$d(x, x'') = \sum_{i=1}^{I} d(x_i, x_i'')$$
$$\leq \sum_{i=1}^{I} d(x_i, x_i') + d(x_i', x_i'')$$
$$\leq \sum_{i=1}^{I} d(x_i, x_i') + \sum_{i=1}^{I} d(x_i', x_i'')$$
$$\leq d(x, x') + d(x', x'').$$

2.4 Product Topologies

If every component set is equipped with a component topology t_i then we can define open component subsets $x_i \in t_i$. Any combination of these open component subsets then defines an **open product subset**

$$\times_{i=1}^{I} x_i.$$

Because the component topologies all contain the component empty sets and component full sets we will always be able to construct the product empty set and product full set from this procedure.

Moreover because the component open subsets are closed under countable unions and finite intersections these product subsets will be as well. For example given any collection of open component subsets

$$\{x_{1,i}, \dots, x_{j,i}, \dots\}$$

we have closure under unions,

$$\cup_j x_{j,i} = x_{\cup,i} \in t_i.$$

Consequently the union of any collection of open product subsets gives another open product subset,

$$\begin{aligned} \cup_j x_j &= \cup_j \left(\times_{i=1}^{I} x_{j,i} \right) \\ &= \times_{i=1}^{I} \left(\cup_j x_{j,i} \right) \\ &= \times_{i=1}^{I} \left(x_{\cup,i} \right). \end{aligned}$$

In other words the collection of all open product subsets $\times_{i=1}^{I} x_i$ with $x_i \in t_i$ satisfies all of the properties of a topology. We refer to this topology as a **product topology**.

3 Prototypical Product Spaces

The most common product spaces we will encounter in practice are built up from the prototypical spaces that we reviewed in Chapter 2, Section 2, combining various integers and real lines together into larger spaces. That said there are a few product spaces worth particular note.

3.1 Replicated Product Spaces

Often we are interested not in any single element of a space X but rather multiple elements of that space at the same time. The selection of I elements can be modeled by replicating the space I times and then using those replications as components of a product space,

$$X^I = X \times ... \times X = \times_{i=1}^{I} X.$$

This construction is referred to as a **replicated product space**, **identical product space** or **power space**.

We already have seen this construction in a few places. For example we defined binary algebraic operator over the set X as mapping two elements of X into a single element of X. That input set of pairs of elements was denoted $X \times X$, which is just a replicated product set with a separate copy of X for each input.

Similarly sequences of I elements from a given set,

$$\{x_1, ..., x_i, ..., x_I\},$$

can be defined as elements of the replicated power set X^I. Allowing I to become arbitrarily large then allows for the countably infinitely long s equences.

One potential difficulty that can arise when working with replicated product spaces is distinguishing between the different copies of the base space X that form the components. In particular any notation that might differentiate the individual copies, such as ticks or integer indices, can also be confused as defining different spaces entirely. For example depending on the context $X_1 \times X_2$ might be used to denote both a product space comprised of two *different* spaces and a replicated product space comprised of two copies of the *same* space. To avoid any confusion we have to be explicit about when we are assuming a general product space and when we are assuming a replicated product space.

3.2 Multivariate Real Numbers

Combining I real lines together defines the **multivariate real numbers**, $\mathbb{R}^I$. This is also referred to as a **real space** or an **I-dimensional Euclidean space**.

The multivariate real numbers inherit the identity crisis of the real lines from which they are built. If we take the perspective that there are infinite, distinct real lines then there will be infinite, distinct multivariate real numbers: different choices of component ordering, algebraic, and metric structures will define distinct product spaces. On the other hand if we assume that there is a single, flexible real line that admits multiple configurations then the multivariate real numbers will correspond to a single, flexible space. In this case each different configuration of the component lines will define a different configuration of the corresponding product space.

To visually summarize the structure of a real space we can extend grids defined over the component spaces across the other component spaces and then overlay them to form a rectangular *mesh* (Figure 5).
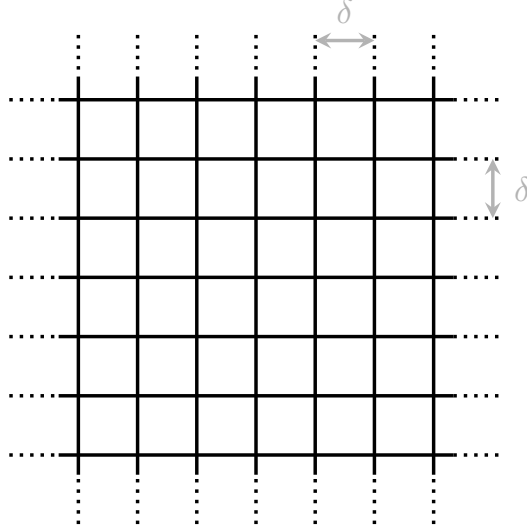

Figure 5: Extending component grids into the product space defines a rectangular mesh that visually communicate the product metric structure. Here $\mathbb{R}^2$ is represented by a two-dimensional mesh.

4 Decomposing Product Spaces

Every element of a product space is uniquely specified by an element from each component space. Specifying all of the component elements at the same time, however, can sometimes be overwhelming. Fortunately we can always specify the component elements *sequentially*.

For example if we decompose a two component product set $X_1 \times X_2$ into replications of the second component set,

$$X_1 \times X_2 = \bigcup_{x_1 \in X_1} \{x_1\} \times X_2$$

then every element can be specified by first specifying an element of the first component set,

$$\tilde{x}_1 \in X_1,$$

and then an element in the corresponding replication of the second component set,

$$x_2 \in \{\tilde{x}_1\} \times X_2,$$

15

to give
$$(\tilde{x}_1, \tilde{x}_2) \in X_1 \times X_2.$$

Note that we're using the~here to denote bound variables as we make each choice of component elements.

At the same time if we decompose the product set into
$$X_1 \times X_2 = \bigcup_{x_2 \in X_2} X_1 \times \{x_2\}$$

then every element can be specified by first specifying an element of the second component set,
$$\tilde{x}_2 \in X_2,$$

before specifying an element in the corresponding replication of the first component set,
$$\tilde{x}_1 \in X_1 \times \{x_2\}.$$

This sequential specification also has a nice geometric interpretation (Figure 6). Specifying $\tilde{x}_1 \in X_1$ and then $\tilde{x}_2 \in \{\tilde{x}_1\} \times X_2$ corresponds to arriving at $(\tilde{x}_1, \tilde{x}_2)$ by first moving along X_1 and then moving along the replication $\{x_1\} \times X_2$. Alternatively specifying $\tilde{x}_2 \in X_2$ and then $\tilde{x}_1 \in X_1 \times \{\tilde{x}_2\}$ corresponds to moving along X_2 before moving along $\{x_1\} \times X_2$.

We have even more options for sequentially selecting component elements from a product set with three components. For example if we decompose the product set into replications of $X_1 \times X_2$
$$X_1 \times X_2 \times X_3 = \bigcup_{x_3 \times X_3} X_1 \times X_2 \times \{x_3\}.$$

then we can specify an element by first taking an element of the third component set,
$$\tilde{x}_3 \in X_3,$$

and then an element of the corresponding replication,
$$(\tilde{x}_1, \tilde{x}_2) \in X_1 \times X_2 \times \{x_3\},$$

to give
$$(\tilde{x}_1, \tilde{x}_2, \tilde{x}_3) \in X_1 \times X_2 \times X_3.$$

By decomposing $X_1 \times X_2$, however, the latter specification can further be separated into
$$\tilde{x}_2 \in X_2 \times \{\tilde{x}_3\}$$

and then
$$\tilde{x}_1 \in X_1 \times \{\tilde{x}_2\} \times \{\tilde{x}_3\},$$

$X_1 \times X_2$ $\tilde{x}_1 \in X_1$ $\tilde{x}_2 \in \{\tilde{x}_1\} \times X_2$

(a)

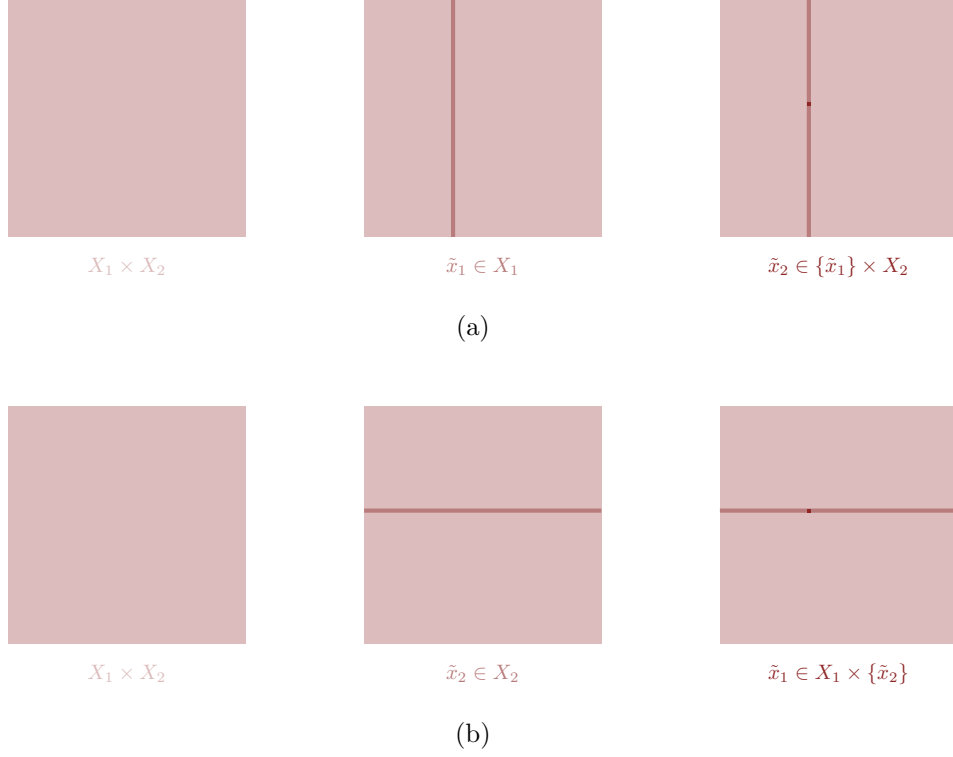$X_1 \times X_2$ $\tilde{x}_2 \in X_2$ $\tilde{x}_1 \in X_1 \times \{\tilde{x}_2\}$

(b)

Figure 6: The element $(\tilde{x}_1, \tilde{x}_2) \in X_1 \times X_2$ can be constructed sequentially in two different ways. (a) We can restrict our initial attention to X_1 to select $\tilde{x}_1 \in X_1$ and then consider the corresponding replication to select $\{x_1\} \times X_2$. (b) Alternatively we could first select $\tilde{x}_2 \in X_2$ before selecting $\tilde{x}_1 \in X_1 \times \{\tilde{x}_2\}$.

or

$$\tilde{x}_1 \in X_1 \times \{\tilde{x}_3\}$$

and then

$$\tilde{x}_2 \in \{\tilde{x}_1\} \times X_2 \times \{\tilde{x}_3\}.$$

The different ways that we can initially decompose $X_1 \times X_2 \times X_3$ allow us to specify an element of the product set by specifying component elements *in any order.*

This pattern immediately generalizes to any product space. Unfortunately the generalization quickly become complicated because of all of the ways that we can select from the available components. To represent how we can sequentially build up product elements from component elements we'll need some careful, if a bit ungainly, notation.

We'll start with the I component spaces

$$X_1, \dots, X_i, \dots, X_I,$$

which together form the total product space

$$\begin{aligned}
\times_{i=1}^{I} X_i = \times_{i \in (1,\dots,I)} X_i \\
= X_1 \times \dots \times X_i \times \dots \times X_I.
\end{aligned}$$

I will refer to this as the **joint set**.

At the same time any selection of those I components also defines a smaller product set: for any collection of $J < I$ component indices

$$(i_1, \dots, i_J)$$

we can construct the product set

$$\times_{i' \in (i_1, \dots, i_J)} X_{i'} = X_{i_1} \times \dots \times X_{i_J}.$$

I will refer to these as **marginal sets**.

If we select the components $(i_1, \dots, i_J)$ then the remaining components,

$$(1, \dots, i_1 - 1, i_1 + 1, \dots, i_J - 1, i_J + 1, \dots, J)$$

define yet another product set. Replicating this second product set once for each element of first product defines a collection of **conditional sets** (Figure 7),

$$\begin{aligned}
\times_{i'=1}^{I} X_{i'} \mid (x_{i_1}, \dots, x_{i_J}) \equiv \quad & X_1 \times \dots \\
& \times X_{i_1-1} \times \{x_{i_1}\} \times X_{i_1+1} \times \dots \\
& \times X_{i_J-1} \times \{x_{i_J}\} \times X_{i_J+1} \times \dots \\
& \times X_I.
\end{aligned}$$

$$X_1 \times X_2 \times X_3 =$$
$$\bigcup_{x_3 \in X_3} X_1 \times X_2 \times \{x_3\}$$

$$\{X_1 \times X_2 \times \{x_3\} \text{ for } x_3 \in X_3\}$$
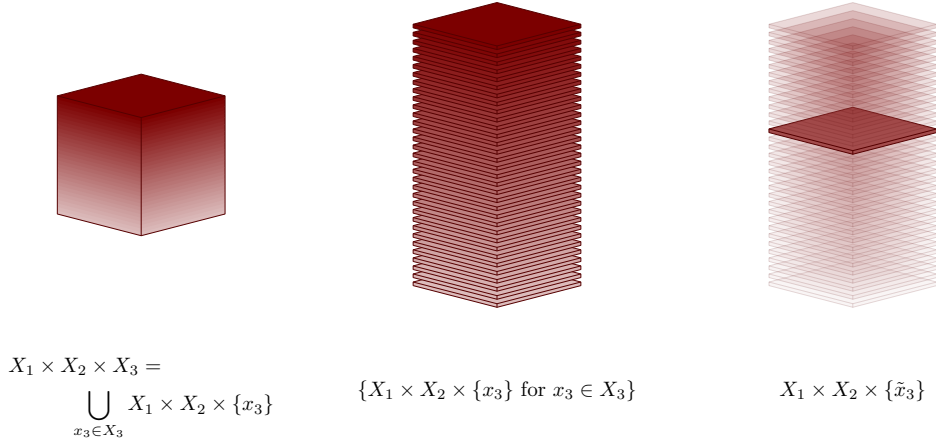
$$X_1 \times X_2 \times \{\tilde{x}_3\}$$

Figure 7: A product set can be decomposed into various combinations of marginal sets and conditional sets. For example taking X_3 as a marginal set reduces the three-component product set $X_1 \times X_2 \times X_3$ into the conditional sets $X_1 \times X_2 \times \{x_3\}$, one for each element of the marginal set. Fixing a particular element $\tilde{x}_3 \in X_3$ isolates one of those replications.

In other words the elements of a conditional set are given by taking an element of the joint set and then fixing, or *conditioning*, the component elements of the corresponding marginal set.

A marginal set and its complementary collection of conditional sets can then be used to reconstruct the joint set,

$$\times_{i=1}^{I} X_i = \bigcup_{(x_{i_1}, \dots, x_{i_J}) \in \times_{i' \in (i_1, \dots, i_J)} X_{i'}} \left(\times_{i'=1}^{I} X_{i'} \right) \mid (x_{i_1}, \dots, x_{i_J}).$$

This then allows us to specify an element of the joint set

$$(\tilde{x}_1, \dots, \tilde{x}_{i_1}, \dots, \tilde{x}_{i_J}, \dots, \tilde{x}_I) \in \times_{i=1}^{I} X_i.$$

by first specifying a collection of component elements,

$$(\tilde{x}_{i_1}, \dots, \tilde{x}_{i_J}) \in \times_{i' \in (i_1, \dots, i_J)} X_{i'},$$

and then specifying the remaining component elements in the corresponding replication,

$$(\tilde{x}_1, \dots, \tilde{x}_{i_1-1}, \tilde{x}_{i_1+1}, \dots, \tilde{x}_{i_J-1}, \tilde{x}_{i_J+1}, \dots, \tilde{x}_I)$$
$$\in \left(\times_{i'=1}^{I} X_{i'} \right) \mid (\tilde{x}_{i_1}, \dots, \tilde{x}_{i_J}).$$

Because we have to account for any choice of marginal set the notation here is far from elegant, but keep in mind that all we're doing is formalizing what happens when we select some component elements first and then the remaining elements second.

Given any partition of the component sets into groups we can apply this decomposition *recursively* to decompose the joint set into a sequence of collections of conditional sets and one terminating marginal set. For example consider five component sets X_1, X_2, X_3, X_4, X_5 which are partitioned into three groups,

$$\{X_2, X_5\}, \{X_3\}, \{X_1, X_4\}.$$

This partition motivates the decomposition

$$\begin{aligned}
X_1 &\times X_2 \times X_3 \times X_4 \times X_5 \\
&= \bigcup_{(x_2, x_5) \in X_2 \times X_5} \bigcup_{x_3 \in X_3 | (x_2, x_5)} X_1 \times X_4 \mid (x_2, x_3, x_5)
\end{aligned}$$

and the corresponding sequential specification

$$\begin{aligned}
(\tilde{x}_2, \tilde{x}_5) &\in X_2 \times X_5 \\
\tilde{x}_3 &\in X_3 \mid (\tilde{x}_2, \tilde{x}_5) \\
(\tilde{x}_1, x_4) &\in X_1 \times X_4 \mid (\tilde{x}_2, \tilde{x}_3, \tilde{x}_5).
\end{aligned}$$

In most applications we will want to separate the component sets individually, allowing us to build a joint element up by specifying component elements one by one. For example working through the component spaces in order gives the specification

$$\begin{aligned}
\tilde{x}_1 &\in X_1 \\
\tilde{x}_2 &\in X_2 \quad \mid \tilde{x}_1 \\
&\quad ... \\
\tilde{x}_i &\in X_i \quad \mid (\tilde{x}_1, ..., \tilde{x}_i) \\
&\quad ... \\
\tilde{x}_{I-1} &\in X_{I-1} \mid (\tilde{x}_1, ..., \tilde{x}_{I-2}) \\
\tilde{x}_I &\in X_I \quad \mid (\tilde{x}_1, ..., \tilde{x}_{I-1}).
\end{aligned}$$

That said we can work through the component spaces in *any* order. This gives flexibility allows us to construct joint elements in whichever way is most convenient in a given application.

Because we often take sequential specification of elements for granted this formal, systematic treatment might appear to be a bit...excessive. To the contrary this more careful perspective will help us understand some of the more subtle aspects of probability theory on product spaces, and how we can develop sophisticated probability distributions in practice.

5 Transforming Product Spaces

We can always transform product spaces directly with monolithic maps that ignore any component structure. These functions map an entire input product set to an arbitrary output

set,

$$f : \times_{i=1}^{I} X_i \rightarrow Y$$
$$(x_1, \dots, x_i, \dots, x_I) \mapsto y = f(x_1, \dots, x_i, \dots, x_I).$$

For example algebraic operations can be interpreted as functions from the input replicated product set $X \times X$ to the output set X,

$$f : X \times X \rightarrow X$$
$$(x_1, x_2) \mapsto x = f(x_1, x_2).$$

Similarly metrics can be interpreted as functions from that same input product set to an output positive real line,

$$d : X \times X \rightarrow \mathbb{R}^+$$
$$(x_1, x_2) \mapsto d(x_1, x_2).$$

The classification of these functions and construction of pushforward and pullback functions follows exactly the same as for functions with general input spaces. We can say much more, howver, about functions that naturally harmonize with the component structure of an input product space.

5.1 Component Transformations

Transformations between two product spaces exhibit a useful fragmentation. Any function between two product sets

$$f : \times_{i=1}^{I} X_i \rightarrow \times_{j=1}^{J} Y_j$$
$$(x_1, \dots, x_i, \dots, x_I) \mapsto (y_1, \dots, y_j, \dots, y_J) = f(x_1, \dots, x_i, \dots, x_I)$$

is equivalent to J functions that map the input product set into each of the individual output component sets,

$$f_j : \times_{i=1}^{I} X_i \rightarrow Y_j$$
$$(x_1, \dots, x_i, \dots, x_I) \mapsto y_j = f_j(x_1, \dots, x_i, \dots, x_I).$$

Pushforward and pullback maps for the complete function can be derived from these component functions.

In general these component functions mix the input components together to inform the behavior in each output component set. Some functions, however, limit each of the input component sets to informing only a single output component set. Specifically if the input product set and

output product set are built up from the same number of component sets I then some functions will map only one input component into each of the output components at a time,

$$f_i : X_i \to Y_i$$
$$x_i \mapsto y_i = f_i(x_i).$$

These component-preserving functions transform each component *independently* of the others.

One nice feature of component-preserving transformations is that they preserve the orientation of grids defined from any component metric structure. For example the function

$$f : \mathbb{R}^2 \to \mathbb{R}^2$$
$$(x_1, x_2) \mapsto (y_1, y_2) = (x_1^{\frac{3}{2}}, x_2^{\frac{3}{2}})$$

transforms x_1 into y_1 independently of the behavior of x_2 and similarly transforms x_2 into y_2 independently of the behavior of x_1. While neither of these component transformations are isometries the rectangular structure of any metric-informed grid will be preserved (Figure 8a).

On the other hand the function

$$f : \mathbb{R}^2 \to \mathbb{R}^2$$
$$(x_1, x_2) \mapsto (y_1, y_2 = (x_1 + x_2, x_1 - x_2)$$

mixes the input components x_1 and x_2 together, skewing the product structure in the process. Consequently grids built up from the component metric structures on the input and output spaces will appear *warped* relative to each other (Figure 8b).

The behavior of component-preserving functions is completely determined by the behavior of the component functions. For example a component-preserving function is injective, surjective, or bijective if and only if all of the component functions are injective, surjective, or bijective. Similarly any product structure endowed on the input product set will be preserved if each of the component structures are preserved along the component transformations.

5.2 Projection Functions

The component-structure of a product space naturally motivates an important class of functions. Consider, for example, the two-component product space $X_1 \times X_2$ and its decomposition into distinct replications of X_1,

$$X_1 \times X_2 = \bigcup_{x_2 \in X_2} X_1 \times \{x_2\}.$$

Input Grid   Pushforward Grid

(a)

Input Grid   Pushforward Grid

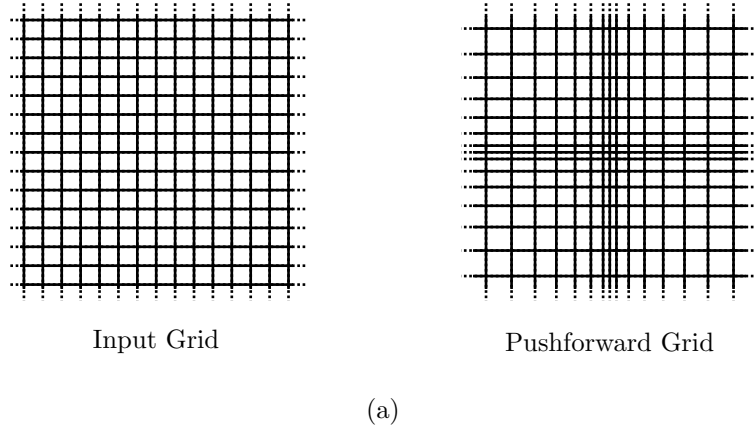Input Grid   Pushforward Grid

(b)
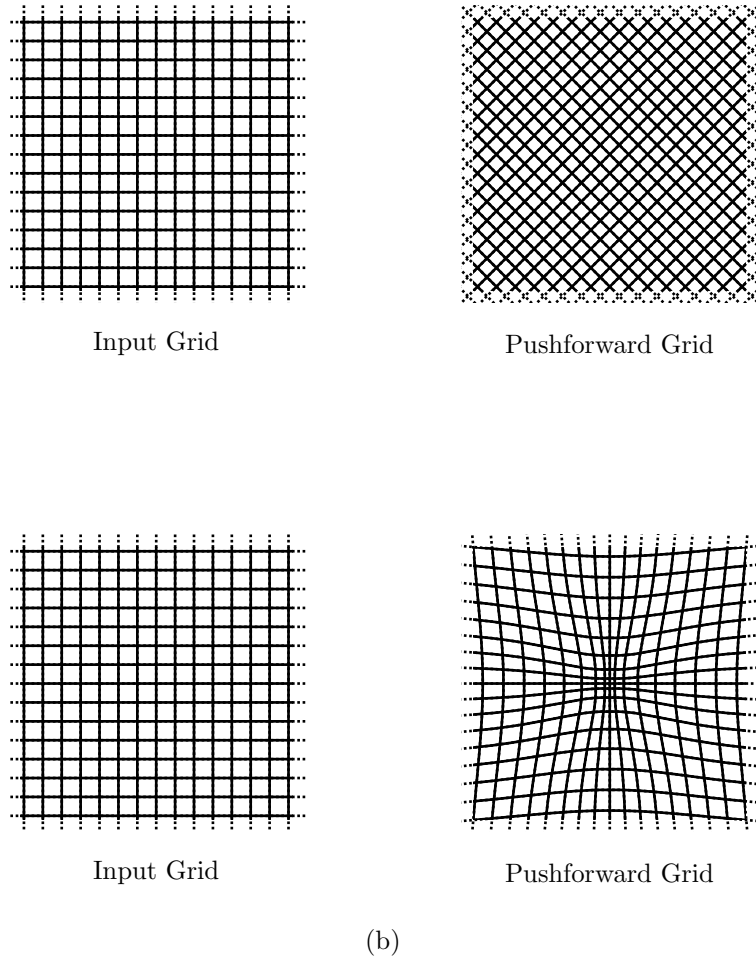
Figure 8: Maps between product metric spaces may or may not preserve the individual components, and this has consequences for how metric-informed grids transform. (a) Maps between product metric spaces that do preserve the components transform rectangular grids into rectangular grids. (b) On the other hand maps that mix the components transform rectangular grids into grids that appear rotated, skewed, or otherwise warped.

23

If we ignore the x_2 label the replications $X_1 \times \{x_2\}$ become indistinguishable from each other, collapsing into a single copy of X_1 (Figure 9). This collapse *projects* every element of the product set $(x_1, x_2) \in X_1 \times X_2$ to a component element $x_1 \in X_1$.

At the same time we can decompose the product set into replications of X_2,

$$X_1 \times X_2 = \bigcup_{x_1 \in X_1} \{x_1\} \times X_2.$$

Ignoring the x_1 label collapses the replications $\{x_1\} \times X_2$ onto each other, projecting every product element $(x_1, x_2) \in X_1 \times X_2$ to the component $x_2 \in X_2$.

$$X_1 \times X_2 = \bigcup_{x_2 \in X_2} X_1 \times \{x_2\} \qquad\qquad \varpi_1(X_1 \times X_2) = X_1$$

$$X_1 \times X_2 = \bigcup_{x_1 \in X_1} \{x_1\} \times X_2 \qquad\qquad \varpi_2(X_1 \times X_2) = X_2$$
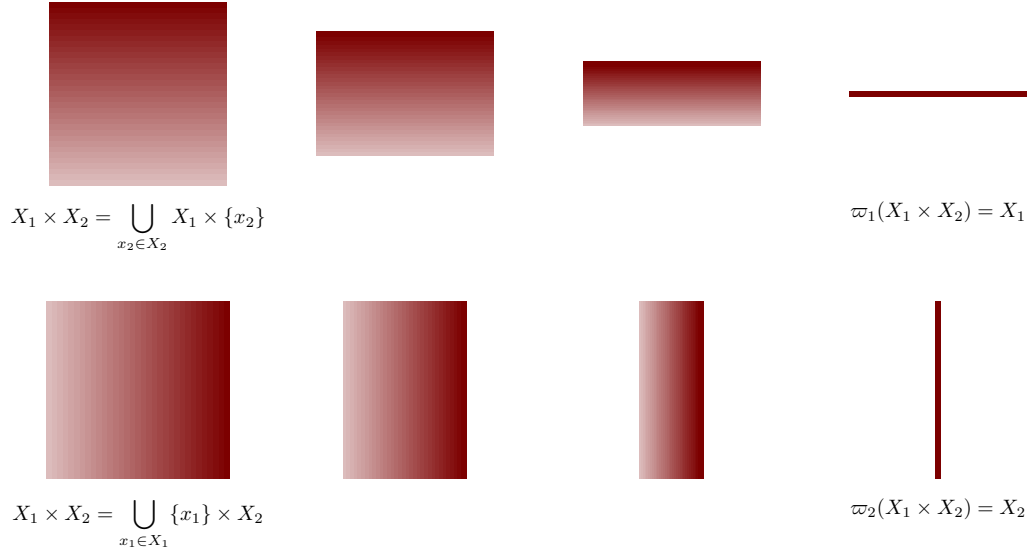
Figure 9: Decomposing a product set into replications and then collapsing those replications on top of each other returns one of the component sets. This allows us to project a product set onto any of its components.

More generally every product set is accompanied by surjective **projection functions** that map the entire product set into the individual component sets by ignoring the other components,

$$\begin{aligned} \varpi_i : \times_{i'=1}^{I} X_{i'} &\to X_i \\ (x_1, \ldots, x_i, \ldots, x_I) &\mapsto x_i \, . \end{aligned}$$

These projections functions are fully consistent with any structure endowed onto the product set. For example the pushforward of any product structure along a projection functions returns the component structure that was used to construct that product structure in the first place. Similarly the pullback of any component structure along a projection function will be always be compatible with the corresponding product structure.

We can also construct more sophisticated **multi-projection functions** that return multiple component spaces at the same time. In particular given $J < I$ component indices $(i_1, \dots, i_J)$ we can define a projection function that maps the product of all of the component sets into a product of just those selected component sets,

$$\varpi_{i_1, \dots, i_J} : \times_{i'=1}^{I} X_{i'} \;\to\; \times_{i' \in (i_1, \dots, i_J)} X_{i'}$$
$$(x_1, \dots, x_I) \mapsto (x_i, x_j, x_k) \qquad .$$

Projection functions are particularly useful for succinctly expressing some of the more subtle product space behaviors. As we saw in the previous section every function between product spaces $f : \times_{i=1}^{I} X_i \to \times_{j=1}^{J} X_J$ decomposes into component functions

$$f_j : \times_{i=1}^{I} X_i \qquad\qquad \to Y_j$$
$$(x_1, \dots, x_i, \dots, x_I) \mapsto y_j = f_j(x_1, \dots, x_i, \dots, x_I).$$

Those component functions, however, can also be defined as the composition of f with each projection function on the output space,

$$f_j = \varpi_j \circ f.$$

Similarly the marginal sets that we introduced in [Section 4](#) can also be defined as the outputs of various multi-projection functions. Even better every conditional set can be defined as the preimage of a particular multi-projection function,

$$\times_{i'=1}^{I} X_{i'} \mid (x_{i_1}, \dots, x_{i_J}) = \varpi_{i_1, \dots, i_J}^{-1}(x_{i_1}, \dots, x_{i_J})!$$

In other words a conditional set can be defined as the subset of the joint set that projects to a particular element of a marginal set. This projection perspective not only clarifies why marginal and conditional sets are complementary but also provides a compact way of denoting these sets.

5.3 Partial Evaluation

Fully evaluating a function on a product space requires the specification of an element in *every* component space. For example in order to evaluate the function

$$f : X_1 \times X_2 \to Y$$

we need to specify $\tilde{x}_1 \in X_1$ and $\tilde{x}_2 \in X_2$ to define the particular output element

$$f(\tilde{x}_1, \tilde{x}_2) = y \in Y.$$

Providing only one element leaves an empty slot in the input product set that we need to fill in order to complete the evaluation. For example $f(\tilde{x}_1, x_2)$ still needs an element of X_2 and

$f(x_1, \tilde{x}_2)$ still needs an element of X_1. That empty slot, however, defines a relationship between the missing input component set and the output set. In other words $f(\tilde{x}_1, x_2)$ implicitly defines a function that maps elements of $\{\tilde{x}_1\} \times X_2$ to elements of Y. Similarly $f(x_1, \tilde{x}_2)$ implicitly defines a function that maps elements of $X_1 \times \{\tilde{x}_2\}$ to elements of Y.

More generally specifying the component elements

$$(\tilde{x}_{i_1}, \dots, \tilde{x}_{i_J})$$

reduces to the product set $\times_{i'=1}^{I} X_{i'}$ to the conditional set

$$\times_{i'=1}^{I} X_{i'} \mid (\tilde{x}_{i_1}, \dots, \tilde{x}_{i_J}).$$

Likewise inputting those component elements into a function

$$
\begin{aligned}
f : \ \times_{i=1}^{I} X_i \qquad &\rightarrow Y \\
(x_1, \dots, x_i, \dots, x_I) &\mapsto y = f(x_1, \dots, x_i, \dots, x_I)
\end{aligned}
$$

defines a new function that maps the remaining elements to an output element,

$$
\begin{aligned}
f_{\tilde{x}_{i_1}, \dots, \tilde{x}_{i_J}} : \ &\times_{i'=1}^{I} X_{i'} \mid (\tilde{x}_{i_1}, \dots, \tilde{x}_{i_J}) \ \rightarrow Y \\
&(x_1, \dots, x_{i_1-1}, x_{i_1+1}, \dots, \quad \mapsto y = f(x_1, \dots, x_{i_1-1}, \tilde{x}_{i_1}, x_{i_1+1}, \dots, \\
&\quad x_{i_J-1}, x_{i_J+1}, \dots, x_I \quad) \mapsto \qquad x_{i_J-1}, \tilde{x}_{i_J}, x_{i_J+1}, \dots, x_I \quad).
\end{aligned}
$$

This procedure is known as **partial evaluation** generally and **Currying** in the computer science literature particularly. The mathematical notation for a partially evaluated function can vary quite a bit. Besides

$$f_{\tilde{x}_{i_1}, \dots, \tilde{x}_{i_J}}$$

it is not uncommon to see notation like

$$f(\cdot \mid \tilde{x}_{i_1}, \dots, \tilde{x}_{i_J})$$

where $\cdot$ represents the remaining unbound variables or even notation that explicity writes out all of the bound and unbound variables,

$$f(x_1, \dots, x_{i_1-1}, \tilde{x}_{i_1}, x_{i_1+1}, \dots, x_{i_J-1}, \tilde{x}_{i_J}, x_{i_J+1}, \dots, x_I).$$

While this latter notation is most direct it quickly becomes ungainly when working with more than a few components.

Like the sequential specification of product set elements the partial evaluation of functions is often taken for granted in practical applications. Explicitly acknowledging it, however, can help us better understand less obvious constructions.

For example consider a set X equipped with a binary addition operator

$$+ : X \times X \to X$$
$$x_1, x_2 \mapsto x_1 + x_2.$$

Partially evaluating this function on the first input component effectively defines a function

$$t_{\tilde{x}_1} : X \to X$$
$$x_2 \mapsto \tilde{x}_1 + x_2$$

that *translates* each element of X by $\tilde{x}_1$. Consequently $t_{\tilde{x}_1}$ is referred to as a translation operator.

Similarly if we have a set X equipped with a binary multiplication operator

$$\cdot : X \times X \to X$$
$$x_1, x_2 \mapsto x_1 \cdot x_2$$

then partial evaluation effectively defines a function

$$s_{\tilde{x}_1} : X \to X$$
$$x_2 \mapsto \tilde{x}_1 \cdot x_2$$

that *scales* each element of X by $\tilde{x}_1$. Fittingly $s_{\tilde{x}_1}$ is often referred to as a scaling operator.

We implicitly reason about translations and scalings in many applications, but partial evaluation allows us to formalize that conceptual reasoning. This more formal construction then helps us recognize the necessary assumptions for that reasoning to be well-defined. For example translation isn't well-defined on a space that isn't equipped with the right algebraic structure!

6 Conclusion

The construction of product spaces defines a systematic way to work with multiple elements from multiple spaces at the same time, or even multiple elements from a single space at the same time. This has many practical benefits. For example it facilitates the development and manipulation of the sophisticated spaces we often need for applied mathematical modeling. At the same time it provides a language for formalizing certain computational techniques.

When working with more than a few component spaces some of the details of the formal construction of product spaces – such as the sequential specification of product elements, projections, and partial evaluations – require some less-than-elegant notation. That said this awkwardness is not due to any conceptual complexity so much as the difficulty in enumerating all of the ways that we can select from a general collection of component spaces.

Acknowledgements

License

A repository containing all of the files used to generate this chapter is available on GitHub.

The text and figures in this chapter are copyrighted by Michael Betancourt and licensed under the CC BY-NC 4.0 license:

https://creativecommons.org/licenses/by-nc/4.0/