# FactorPrism: Basis Pursuit for Impact Decomposition and Cause Localization and Quantification within Feature Hierarchies

David Rimshnick

## Abstract

Growth decomposition, that is to say, the post-facto allocation of causal impacts to feature categories, is a fundamental and ubiquitous task in analytics and business intelligence, but there are few formal approaches or automated techniques. In this paper, we show that the common informal approach can be viewed as a greedy allocation of multipliers over hierarchal categories of item features, and thus is an extension of *multiplicative decomposition* from the time series literature. We then show that this algorithm is generally suboptimal in an $l_1$ sense, and that we can use the compressed sensing technique of *basis pursuit* to optimize against this norm. We demonstrate that this technique yields more convincing results, on toy, simulated, and real-world datasets. Specifically, our analysis suggests this technique is 2-3 times more accurate in isolating and quantifying causal impacts than the default approach. We discuss applications in sales and operations analytics, and link to our freely-available software implementing this algorithm.

## 1 Introduction

Perhaps the most fundamental question of business analytics, for instance when referring to a new and/or unexpected growth pattern in sales data, is "what is going on?". This question then can be more precisely framed as "what are the main drivers of this growth (positive or negative), and what is their contribution?". In practice, one is often concerned with understanding the root causes of an item's growth at various hierarchical *categories*, for instance how much a given product's +15% growth in a quarter is driven by market-category versus group-category versus product-category factors.

Moreover, growth causes often operate along independent hierarchies which may intersect. For example, there may be geographic impacts at the nation or state category, which impact either all or just specific groups of products.

Without having nuanced and case-specific data, it is often difficult to perform causal inference to ascribe impact to individual drivers of behavior. We can, however, attempt to attribute the total impact of these drivers at the category at which they operate based on the feature and performance data alone. For instance, we can seek to attribute the total impact of market-category factors versus state-category factors, etc.

Although this may sound abstruse, in reality this attribution is done pervasively and subconsciously. If a product's volume grows 15% in a given quarter and total market volume grows only 10%, we almost reflexively say that 10% of the product's growth is due to market volume drivers, and the remaining 5% is due to the specific product.

The implicit potential error with this approach (which we will show generalizes to a greedy "top-down" algorithm over the feature hierarchy) may already be obvious from this example - that because the product is itself part of the market, we may be over-ascribing growth to the market which should be ascribed to the product.

In fact, we will show that this dilemma illustrates the fact that in attributing this growth we are implicitly dealing with an underdetermined system of equations, and there is a more plausible solution to this system than the one generated by the greedy approach - specifically the one minimizing the $l_1$ norm. In fact, this construction is an instance of the compressed sensing technique *basis pursuit*.

Besides our intuition that this solution is more plausible by the Occam's Razor logic implicit in the $l_1$ norm, specifically that this represents the solution with the smallest set of causes to generate the result (in a sense we will expound), we will also show in simulation and empirically that this approach is superior, as it is able to better isolate "known" causal impacts.

Upon completion, our analysis yields two major types of results: (1) a quantification and ranking of the growth impact at each category, and, relatedly, (2) the decomposition of any given segment's growth vis a vis the impacts at each category. This allows the user to read-off the most important categories of impact either overall or to a specific item, and therefore target further investigation or exploratory analysis.

An implementation of this algorithm is available in the eponymous software package *FactorPrism*, maintained by the authors and available for free through the Windows Store (see `https://www.microsoft.com/en-us/p/FactorPrism/9nt1961v0n7g`).

## 2 Related Work

The time-series literature, specifically within forecasting, has typically viewed two considerations - viewing a trend as a product of multiple factors, and aggregating effects from hierarchies or groups - as separate concerns [11]. Typical applications have been in signal decomposition of additive features [4, 6, 19, 23]. Recent work has investigated trend decomposition based on additive features combined with feature reduction [1]. Other recent work has explored the application to network reconstruction vis à vis time series, again in an additive sense [13].

In the case of multiplicative decomposition, the first typically involves decomposing a given data unit $Y_t$ into the factors

$$Y_t = T_t \times R_t \tag{1}$$

where $T_t$ is a trend/seasonality factor, and $R_t$ is a remainder component. ($T_t$ could also be decomposed into separate trend and seasonality). $T_t$ would be estimated by one of a number of decomposition methods, and then $R_t$ would simply be the remaining factor [7, 22].

In parallel, the hierarchies of data categories have been treated through additive aggregation and disaggregation - for example, at any time $t$, the value of a forecast with data components $A$ and $B$ and with subcomponents $AA$, $AB$ for $A$ and $BA$, $BB$ for $B$. See Figure 1.

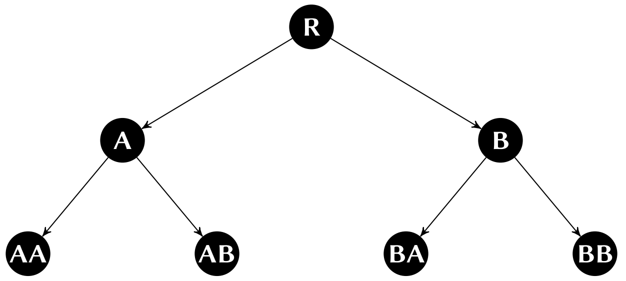


Figure 1: Example tree representing hierarchies of feature categories

This would be decomposed as

$$Y_{A,t} = Y_{AA,t} + Y_{AB,t} \tag{2}$$

$$Y_{B,t} = Y_{BA,t} + Y_{BB,t} \tag{3}$$

and

$$Y_{Root,t} = Y_{AA,t} + Y_{AB,t} + Y_{BA,t} + Y_{BB,t} = Y_{A,t} + Y_{B,t} \tag{4}$$

The overall forecast would be built by either aggregation, disaggregation, or a combination of both of forecasts at various categories.

The combined approach, which is our apparently novel formulation, would be defined as, for example for data unit $AA$

$$Y_{AA,t} = f_{AA,0} \times f_{AA,t} \times f_{A,t} \times f_{Root,t} \tag{5}$$

where $f_{AA,0}$ is the initial value of the data unit to which the growth will be compared.

This problem can be considered related to a type of multilevel causal inference modeling (as in e.g. [9]), but degenerate (due to $n > p$) and applied to multiplicative space.

## 3 Motivation

We begin with definitions and simple motivating examples for our framework and assignment algorithm to develop intuition. We then develop an optimization problem that we posit recovers the intuitively correct factors.

### 3.1 Inputs and Definitions

The input to the algorithm is the following:

- A series of datapoints with a number of features, at least one of which being a time dimension. Without loss of generality, the data is aggregated to the level of these features (if this is not done, the preprocessor would perform this task.) These data should be non-negative, real-valued quantities so that values can be aggregated. Typical examples include number of units sold, sales dollars, or number of transactions.

- An ordering of those input features (other than the time features) among separate hierarchies. For instance, there may be a "Product Feature" hierarchy, which includes [Product Type], [Product Subtype], [Item] in that (descending) order of scope, as well as a "Customer-Geography Feature" hierarchy, which includes the features [Region], [State], [County], and [City].

The algorithm preprocessor creates a directed-acyclic-graph (DAG) of all data points by inducing the hierarchies of their feature categories. You can imagine, if there is only a single hierarchy, such as "Product Characteristics", then this DAG is a tree. (It is helpful to think of the DAG as a tree because it functions in many ways similar to one, although when there are different hierarchies, the graph ends up being a cross-product of different trees, so that each node can have multiple parents, so that it is in fact technically a lattice.) Going forward, we will call these DAGs *fDags*.

Each unique combination of feature settings (or lack of setting) defines a unique *category*, corresponding to a node on our fDag. Importantly, there is a category where no feature is set, known as the *Overall Category*. Note that each category other than the *Overall Category* has at least one ancestor parent, or node prior to it in the fDag.

Each input data point corresponds to a terminal node (or leaf) of the fDag, which we will define as a *microcategory*.

Inherently our model assumes that there is an unobserved set of causes $H_i$ within and influencing each category $i$. Our algorithm seeks to recover the total multiplicative *factor* $f_i \approx \sum H_i$ of each category $i$, where $f_i$ *influences* all data in $i$.

Importantly, note that the data in a given category $i$ can be influenced by other causes beyond just $H_i$. For instance, suppose we were working with data from a retailer selling, among other things, sports equipment. The data related to *Baseball Gloves* would be influenced by the causes within *Sports Equipment* as well as the causes within *Baseball Gloves* (among others). Correspondingly, some of the data for *Baseball Gloves* would be influenced by other "below" factors, such as those related to a particular model of glove, as well as "adjacent" factors, like those related to particular geographic states.

Note that the microcategories themselves also have factors. The default and null solution to our problem is that, at each time step, the microcategory factors contain all the multiplicative value and the other factors are all equal to 1. (This will be referred to as the "Bottom-up" solution in Section 4.3.)

To extend our example from above, consider if we only had one hierarchy, "Product Characteristics". The categories would include those in Table 1.

Table 1: Example categories within a "Product Characteristic" hierarchy (Labels abbreviated for space as shown).

| |
|---|
| (Overall category) |
| Sports Equipment (SE) |
| Books |
| ... |
| SE : Baseball Equipment (BE) |
| SE : Football Equipment (FE) |
| ... |
| SE : BE : Baseball Gloves (BG) |
| SE : BE : BG : Wilson Model X |

The last entry in Table 1 is our microcategory, and the category to which the input data would be aggregated. If we had a second hierarchy, "Adult/Child", then the microcategories would be twice as many (one for *Adult* and one for *Child*), and generally there would be two additional sets of categories, one for *Adult* and one for *Child* (in addition to the one including both adults and children, which would be the categories above (without this feature), in which case we would consider the feature "Open").

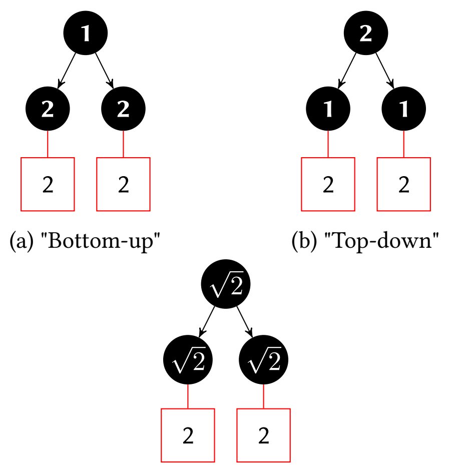
## 3.2 Assignment Motivation

We seek to create an algorithm to assign multiplicative factors to various feature-settings in the hierarchy in the most parsimonious way possible. As perhaps the simplest possible example, suppose we had two items of equal weight, $A$ and $B$. We illustrate our approach for perhaps the two most simple scenarios: (1) both $A$ and $B$ double between times $t_0$ and $t$, and (2) $A$ doubles but $B$ remains constant.

We begin with scenario (1), as illustrated in Figure 2. Since the system is underdetermined, we can assign the multiplicative factor in an infinite number of ways. In Figure 2, we show the two perhaps most obvious ways: Applying all the multiplicative weight to the item themselves ("Bottom-up"), or all the weight to the root ("Top-down"), as well as an arbitrary other way, attributing the factor "equally" between the root node and the leaves. (We will expand in Section 4.3 on why we can think of these as "Bottom-up" and "Top-down" approaches respectively.)

Unfortunately, similar to the case in causal inference, we don't have a "truth set" for the correct factor assignments. Alternatively, we follow the intuitions of compressed sensing and assume that the simplest (i.e. most parsimonious, natural, or "Occum's Razor") solution is desired. Clearly, the "Top-down" approach here best fits our intuition of the simplest solution, as it is more likely there was a single doubling that accounted for the overall doubling, rather than two "separate" doublings, or something in between.

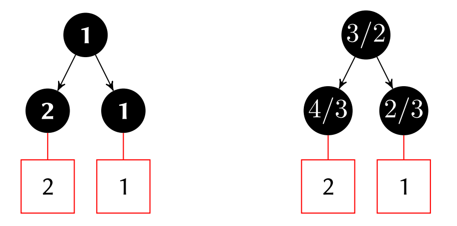
Consider now the second scenario, as illustrated in Figure 3. Here, contrary to the first scenario, the more compelling attribution is the "Bottom-up" assignment, as, in particular, the "Top-down" implies that item B had item-category impacts of



(a) "Bottom-up" (b) "Top-down"

(c) Arbitrary

Figure 2: Example multiplicative factor assignments constituting scenario (1), a doubling over two categories of equal weight. (Total microcategory impact shown in red box.)



(a) "Bottom-up" (b) "Top-down"

Figure 3: Example multiplicative factor assignments constituting scenario (2), where only category $A$ (left leaf) doubles.

2/3, which counteracted the overall increase of 3/2 such that overall its volume remained constant. Clearly this seems less plausible than the impacts being isolated to item A.

We can see that, for scenario (1), our preferred solution also has the lowest joint product between the factors (the product is 2, as opposed to 4 and $2\sqrt{2}$ for the other solutions shown). We might think this is the objective then - minimizing the product - but we can see that we can make the product arbitrarily small by letting the leaf factors go below 1. In fact, this is bourne out by scenario (2), where the "Top-down" approach has a lower joint product (4/3 vs 2). So instead of minimizing the product of the factors directly, we instead want to minimize the product of the factors "multiplicative distance from 1" in some sense.

To codify this concept, we define the function

$$Z(x) = max(x, 1/x) \tag{6}$$

as *Geometric Absolute Value* (analogous to traditional absolute value being the additive distance from 0 to $x$, this is the geometric distance from 1 to $x$). (This can be thought of as a transformation of traditional absolute value to exponential space [16].)

Thus, our objective then becomes minimizing

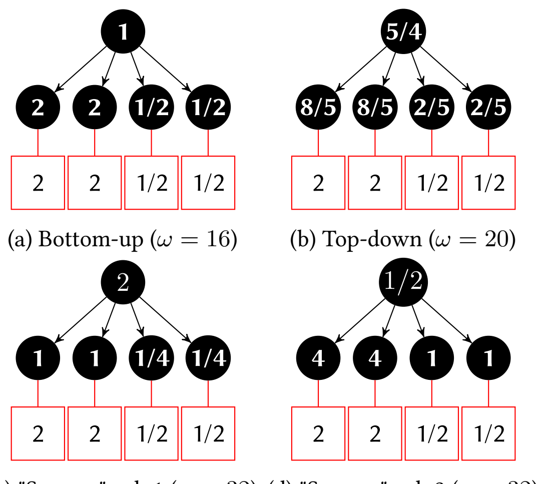
$$\omega = \prod Z(F) \tag{7}$$

with $F$ corresponding to the set of category-level factors.

## 3.3 Sparsity

In the prior section, we did not acknowledge that there is another obvious potential objective, which happens to also lead to the correct solution for both of our toy examples. Namely, we could have looked for the solution with the highest degree of *sparsity* (in this case referring to the number of factors not equal to 1).

However, a quick counterexample demonstrates that the sparsest solution is clearly not desirable. Consider the example in Figure 4, with four equally sized categories.



(a) Bottom-up ($\omega = 16$) (b) Top-down ($\omega = 20$) (c) "Sparser" sol. 1 ($\omega = 32$) (d) "Sparser" sol. 2 ($\omega = 32$)

Figure 4: Example showing how sparsity objective fails to recover desirable solution. Value in parenthesis indicates value of objective from Equation 7.

The "Sparser" solutions, despite having fewer non-one factors (3 versus 4 or 5), clearly are not preferable. The most clear argument is that, by symmetry, there is no reason why one of these should be preferred over the other. Intuitively, we are adding another sort of complexity besides the raw cardinality of factors, and in fact their magnitude matters. This is why our objective from Equation 7, which captures the magnitude of the impacts as opposed to their raw counts, yields the intuitively correct solution (in this case, (a), the "Bottom-up" solution.)

# 4 Optimization

## 4.1 Equally-weighted Case

We start by writing down the functional form to maximize in our simple case where all microcategory factors have equal weight; i.e. the microcategories they represent start from equal size. We will then move to the more general case where these categories have differing starting sizes.

Let $S$ be our set of *categories*, and $M$ be our set of *microcategories*.

In the case where all microcategory data $Y_{i,0}$ are equal, we seek to find the set of multiplicative factors optimizing

$$\begin{aligned} \text{minimize} \quad & \prod_{i \in S} Z(f_i) \\ \text{subject to} \quad & Y_{j,0} \prod_{i \in S_j} f_i = Y_{j,t}, \; j \in M \\ & f_i > 0, \; i \in S \end{aligned} \tag{8}$$

where $Z()$ is again our *Geometric Absolute Value* as defined in Equation 6, $Y_{j,t}$ is the data value corresponding to microcategory $j$ at time $t$, $f_i$ is the factor attributed to category i, and $S_j$ are the set of ancestors of any microcategory $j$ (including $j$ itself).

We can see that if we set $g_i = \log f_i$, our problem is equivalent to

$$\begin{aligned} \text{minimize} \quad & \sum_{i \in S} |g_i| \\ \text{subject to} \quad & \sum_{i \in S_j} g_i = \log Y_{j,t} - \log Y_{j,0}, \; j \in M \end{aligned} \tag{9}$$

and we are back to traditional additive form, albeit with the objective being a sum of absolute values.

We thus realize we are finding the feasible solution $g^*$ with the smallest $L_1$ norm (Manhattan distance) to the origin $g_i = 0$ $\forall i$. Thus this problem is an instance of the *basis pursuit* problem [5] and thus is fundamentally related to *compressed sensing* [8].

(The reader can verify that the motivating problems of Section 3.2 are solved to our desired solutions if solved in this LP form.)

It is perhaps interesting to note that basis pursuit was first applied to compressed sensing as a relaxation to minimizing the $l_0$ norm, where sparsity *was* desired [3]. The authors showed that under certain assumptions, the $l_1$ norm would recover the sparsest and thus desired solution. From the argument of Section 3.3, clearly these assumptions do not hold in our case, and yet the $l_1$ norm still recovers our desired solution.

## 4.2 Weighted Case

In the more general and realistic case where $Y_i, 0$ are not equal, we adapt equation 9 to

$$\begin{aligned} \text{minimize} \quad & \sum_{i \in S} W_i |g_i| \\ \text{subject to} \quad & \sum_{i \in S_j} g_i = \log Y_{j,t} - \log Y_{j,0}, \; j \in M \end{aligned} \tag{10}$$

where $W_i$, the category weighting factor, is defined as the maximum of all microcategories' data $Y_{j,0}$ where $i$ influences $j$, i.e. $i \in S_j$,

$$W_i = \max_{j \in Q_i} Y_{j,0} \tag{11}$$

where

$$Q_i = \{j \text{ s.t. } i \in S_j\} \tag{12}$$

Let us briefly justify why the maximum of influenced microcategory weights is chosen as our weighting factor as opposed to the sum. First, it should be obvious that, if the sum is chosen, then our weighted case does not generalize our equally-weighted case. In the simplest example, a single overall parent with two microcategories, where both microcategories had equal weight, the parent factor weight would be twice that of its children. Our equally-weighted case has them equal.

On the other hand, choosing the maximum clearly maintains this generalization. Intuitively the reason for choosing the maximum can be understood with the following argument: if the weight of a parent category was ever less than the max of its microcategories, then it may be possible to "artificially" increase the parent category's factor beyond its correct value (instead of increasing the value of the correct category's), as the "correction penalty" that would be incurred (in having to adjust down the other microcategories) could be worth the increase. On the other hand, setting the weight above the maximum would unnecessarily penalize attribution to the parent category and thus work against the spirit of the allocation.

See the Appendix Section A for a slightly more thorough justification.

## 4.3 Greedy Assignments

Before moving on to our final formulation, let us briefly address other approaches to serve as our reference points. The "Bottom-up" and "Top-down" assignments of Section 3.2 were toy implementations of two natural greedy algorithms to assign these factors. The terms "Bottom-up" and "Top-down" refer to the way the factors are assigned to the fDag.

Specifically, we recall that our factors must satisfy the following constraints (in the multiplicative form):

$$Y_{j,0} \prod_{i \in S_j} f_i = Y_{j,t},\ j \in M \tag{13}$$

We can imagine traversing the fDag greedily assigning factors, with the remaining factors having to satisfy the remainder. If we start at the microcategories (or leaf nodes of the fDag as it were), we arrive at a degenerate solution where all multiplicative weight is assigned to the microcategories themselves, and the rest are 1 (or 0 in the log form), i.e.

$$\begin{aligned} f_i &= Y_{i,t}/Y_{i,0}, && i \in M \\ f_i &= 1, && i \in S - M \end{aligned} \tag{14}$$

Since we started the transversal at the "bottom" of the fDag, we refer to this as the "Bottom-up" approach, and it represents our null solution.

The more interesting transversal is the "Top-down", where we start with the "Overall" category (or root node) and assign weights iteratively. (We note that since our structure is a DAG, there exists a topological ordering, so this is possible.) Our assignment becomes

$$f_i = \frac{Y_{i,t}/Y_{i,0}}{\prod_{j \in (S_i - i)} f_j},\ i \in S \tag{15}$$

so that the factor for each category is simply its own data's growth modulo the product of its ancestors' factors.

We note that this approach is suboptimal for the reason alluded to in the Introduction, namely that because the data for each subcategory is also part of its ancestors, we may assign impact at a higher category that truly occurs at a more granular category. (Hence it was not our preferred solution in the second scenario in Section 3.2, and will again be shown to be unsatisfactory in Section 5.2.)

Although in just now referring to the Introduction, we are implying that the "Top-down" approach is the standard approach to this problem, this is overly generous. In fact anything like a standard is really a degenerate hybrid of the "Top-down" and "Bottom-up" approach, where overall impact is first calculated, and the remainder is allocated to the microcategories. (In other words, any "intermediate" categories are ignored.) An example of this principle taken for granted is in the Capital Asset Pricing Model (CAPM), where there is simply an overall "market-risk premium" and an "equity-risk" premium calculated, with no other intermediate category [17].

## 4.4 Ultimate Formulation

The equally-weighted case, as mentioned, is an instance of the *basis pursuit* problem, and thus can be solved as an efficient linear program (LP), with much research on the most efficient algorithm to do so [2, 5, 12].

The weighted case can also be solved as an LP, which constitutes our ultimate model formulation, namely

$$\begin{aligned} \text{minimize} \quad & \sum_{i \in S} W_i t_i \\ \text{subject to} \quad & \sum_{i \in S_j} g_i = \log Y_{j,t} - \log Y_{j,0},\ j \in M \\ & t_i \geq g_i,\ i \in S \\ & t_i \geq -g_i,\ i \in S \end{aligned} \tag{16}$$

where $t_i$ are dummy variables to force $W_i|g_i|$ to minimized.

(The reader will note that this LP is guaranteed to be feasible, as the "Bottom-up" solution, where the microcategories contain all the multiplicative value, is always available.)

Any standard LP system (as in [14]) should be suitable for solving instances of this model; for instance the primal-dual interior point method implemented by Google's *Glop* solver has worked satisfactorily in the authors' experience.

## 4.5 Descriptive Analysis

Given we have solved for the factors above, we are prepared to answer, for any given category $C$ of the data, "which factors have the largest impact on $C$?". We note that, when $C$ is the *Overall Category*, our question becomes "which factors have the largest impact overall?".

To compute the aggregate impact $p_i$ of any category $i$'s causes, we use the formula

$$p_i = (f_i - 1) * \prod_{j \in (S_i - i - C)} f_j * f_C * Y_{i,0} \tag{17}$$

again where $S_i$ is the set of ancestors of $i$ (including itself).

More generally, the unit impact $p_{C,i}$ of a category $i$'s factor $f_i$ on a given category $C$'s data is given by the following relation:

$$p_{C,i} = (f_i - 1) * \prod_{j \in (S_i - i - C)} f_j * f_C * (Y_{C,0} \cap Y_{i,0}) \quad (18)$$

where $Y_{C,0} \cap Y_{i,0}$ is shorthand for the number of units in $C$ that are also a member of $i$ (in time 0). $p_{C,i}$ can then usefully be expressed as a percentage of $C$'s total growth $\sum_i p_{C,i}$.

## 4.6 Static Decomposition

We note that, up to this point and in this section, our discussion has been regarding the change between two time periods, namely $t_0$ and $t_t$ (abbreviated by their subscripts). However, if we are dealing with multiple time steps, the algorithm can simply be repeated for each any time period $t_k$ relative to $t_0$, and then the results aggregated (for instance, to compute a "net impact").

Instead of decomposing causes in time, however, FactorPrism can also be used to decompose a static set of members. In effect, instead of answering the question "What is the aggregate impact of each category on growth from $t_0$ to $t_i$?", we ask "What is the disproportionate aggregate multiplicative contribution of causes in each category to a given member's size?".

For illustration, suppose one were decomposing medical case costs in a given year. "Orthopedic" cases might be twice the average case cost, while "Surgeries" would be five times the average cost. However, "Orthopedic Surgeries" might have 30x the average case cost, in which case the $f_i$, in this case understood as the *disproportionality factor* of $i$, would be 3, since "Orthopedic Surgeries" were expected to only have 10x the cost of an average case (2x from being in "Orthopedics", and 5x from being a surgery).

To modify our model for this case, we simply let $D_0 = 1$ and change our weighting factors $W_0$ (from Equation 10) to be dependent on $D_i$ instead of $D_0$. (The choosing of $D_0 = 1$ might seem arbitrary, but any scale of this would be absorbed in the "Overall" factor by the LP construction. So the only correction that needs to be made is adjusting the "Overall" segment impact so that total impacts sum to unity.)

## 4.7 Implementation Considerations

In practice, certain implementation details must be considered. One such detail as how to handle "new" categories (i.e. that were not around at $t_0$, in which case their factor multiplier would be undefined). For our purposes, we are still interested in the unit impact of these "new" categories even if their factors are undefined. Thus a workaround for our intentions is to set the zero values to some small value $\epsilon$. The multipliers for this category can then be "re-scaled" for output/display purposes to some convention (e.g. that the first multiplier should be +100%). In this case, this category's attributed unit variance is still captured correctly (as well as multiplier factors beyond the first, relative to the first).

We also note that in the log form $f = 0$ factors are not allowed. If any categories have $Y_t = 0$ data in any time period, we can assign the $f = 0$ in the "Top-down" manner described in Section 4.3, with the top-most 0 category $i$ having $f_i = 0$ and descending categories $j$ having $f_j = 1$. We then exclude all $i$'s microcategory descendants from $M$ in the log form.

Other considerations may include "pruning" the fDags so that microcategories have a minimum size, as well as normalizing data magnitudes for solver tolerance purposes, etc. The user should implement as contextually appropriate.

# 5 Empirical Results

## 5.1 Simulation Results

In this section, we will demonstrate simulation results confirming massive increases in accuracy with the FactorPrism algorithm over greedy approaches. In particular, we will show that, when randomly-sized causes are "placed" at random categories, FactorPrism is nearly perfect at recovering the impacts at each category, while greedy approaches are hugely inaccurate.

For our simulation, we imagine a feature set with two hierarchies, "HierA" and "HierB". Moreover, each hierarchy has two features, {[A], [AA]} and {[B], [BB]}, respectively. Finally, we test two scenarios: one in which each feature has 2 settings (e.g. "AA1" and "AA2", "BB1" and "BB2") versus one in which each level has 3 settings. For the 2-setting scenario, this results in 49 distinct categories, and for the 3-setting scenario, this results in 169 distinct categories.

As alluded, we run a simulation to test the effectiveness of the FactorPrism algorithm to correctly attribute the causal impact in one time period of randomly placed causes among these categories. We start by assigning each microcategory a starting value in $N(10000, 2000)$. For our impact period, we assign random noise in $N(0, .001)$ to the starting data. Then, we test scenarios where we place 1,3, or 5 causes in random category(s). The causes have an effect size in $N(0, 0.05)$.

We tested 1000 runs for each configuration, and measured the accuracy of FactorPrism is attributing to correct causal impact to each categories versus the greedy "Top-Down" and "Bottom-Up" approaches.

To measure the accuracy of the allocation, we look at the portion of impact that is allocated in the right direction at each categories. We use the following formula for the accuracy $P$:

$$P = \frac{\sum_{C \in Cats} \min(|A_C|, |E_C|_D)}{\sum_{C \in Cats} |A_C|} \quad (19)$$

where $A_C$ is the actual impact in category $C$, $E_C$ is the algorithm estimated attribution to that level, and $|E_C|_D$ is the absolute estimated attribution in the same direction as the actual impact.

Table 2 summarizes the results over the iterations. As is clear, FactorPrism is 2-3 times more accurate than the greedy approaches, showing near-perfect accuracy in recovering the correct causal attributions, while the alternative methods are quite deficient. In particular, this analysis somewhat suggests the lift increases with situational complexity, as in the 5-cause 3-setting scenario FactorPrism recovers the category impact almost perfectly. This suggests that the results will be even more accurate in real-life scale.

Table 2: Simulation results comparing the impact attribution to categories between FactorPrism and greedy approaches.

| | | **method** | | |
|---|---|---|---|---|
| **Causes** | **Settings** | **FP** | **TopDown** | **BottomUp** |
| 1 | 2 | **93.5%** | 34.8% | 32.3% |
| | 3 | **92.4%** | 44.7% | 42.4% |
| **1 Total** | | **92.9%** | **39.9%** | **37.4%** |
| 3 | 2 | **88.8%** | 37.7% | 27.1% |
| | 3 | **94.5%** | 50.7% | 24.5% |
| **3 Total** | | **91.3%** | **43.3%** | **26.0%** |
| 5 | 2 | **78.1%** | 42.2% | 20.1% |
| | 3 | **97.4%** | 53.0% | 22.3% |
| **5 Total** | | **86.5%** | **46.9%** | **21.1%** |
| **Overall Total** | | **90.1%** | **43.5%** | **27.7%** |

## 5.2 Real World Example: NYC 311 Data

We begin this section by noting that ideally we would look to demonstrate the value of this algorithm by using a company's sales data, as this is perhaps the most compelling and immediate use case (hence our hitherto examples referring to retail sales). However, company's rarely share their transactional data, and in the limited public resources identified by the authors, there was not adequate item and customer information to apply the approach in a meaningful way.

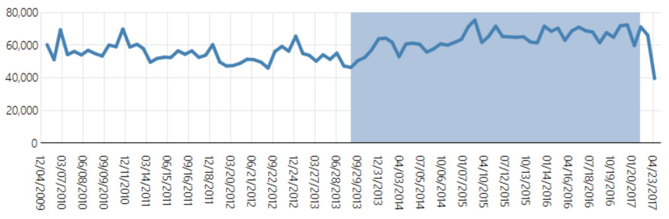
Nonetheless, as we will again mention, operational statistics are another potentially fruitful application. Fortunately, public operations data seems more generally available. A concrete example is the "NYC Open Data" project (`https://opendata.cityofnewyork.us/`), which hosts a large number of datasets around city operations and concerns. In particular, NYC shares data about utilization of its "311" service. (As described by their website, "311 is New York City's non-emergency call center that allows residents to make service requests, file complaints, and get additional information about the City" [15].)

We posit this data is a strong candidate for our approach, specifically in mining for patterns in the types of complaints made to the system. Usefully, each call listed as a transaction, along with a number of descriptive features. For the sake of simplicity, we will focus on a single hierarchy of these features (which we will call "Complaint Information"). There are two features relevant to this hierarchy in the data: the broader [Complaint Type] feature, and the more granular [Descriptor] feature.

Data from January 2010 to April 2017 was accessed. After aggregating the data to monthly totals, it seemed worth investigating the trend between September 2013 and March of 2017, as there is a clear increase over this period (see Figure 5.) In fact, usage was 38.3% higher than the September 2013 level during this period. We then decomposed this 38.3% growth in this period vis a vis the "Complaint Information" features using our approach (which we will abbreviate to *BP*, for Basis Pursuit) versus the "Top-down" method (*TD*).

The results from the two approaches were notably and tellingly different. Although there is obviously much to compare, we will highlight two representative differences. The first, and ultimately most damning, difference between the results of the approaches is in the attribution of effect on overall growth

Figure 5: Aggregate monthly calls to the 311 service center, January 2010-April 2017. Time period investigated shaded in blue.



to the *Overall Category*, as shown in 3. By construction, the *TD* approach attributes all of the net growth in the time period to its own factor. All other factors then must net out to zero impact. Our approach allocates only a much more realistic 6.4% growth to the *Overall* factor, with the remainder of the net impact attributed to factors from other categories.

Table 3: Growth decomposition between *Overall Category* and others for the "Top-down" (TD) versus the current approach (BP).

| | TD | BP |
|---|---|---|
| Overall Category | 38.3% | 6.4% |
| All Other | 0.0% | 31.9% |

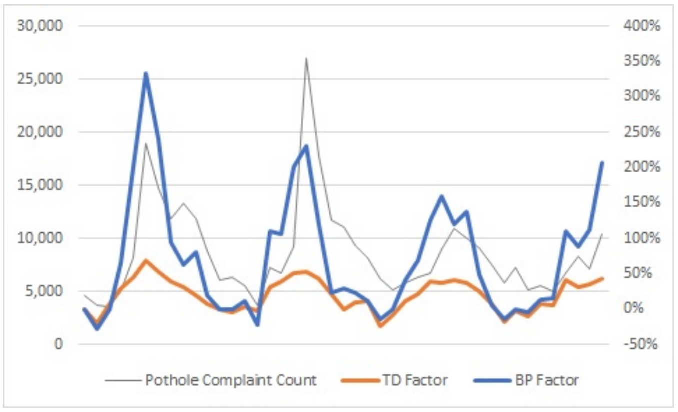
The downstream impact of this over-generalization is evident in the greedy method's inability to "find" important underlying impacts, one of which is the second difference we will highlight. In particular, we investigate the attribution of impact to service requests for *Street Condition: Potholes* (where "Street Condition" is the [Complaint Type] and "Potholes" is the [Descriptor]; for brevity here we will simply refer to as *Potholes*). The *BP* approach only allocates nearly four times as much impact to the *Pothole* factor versus *TD* (roughly 73,000 call impact versus 20,000 call impact, respectively). As illustrated in Figure 6, the *BP* approach is much better able to surface this category's impact, while in the *TD* approach, this impact is very blunted by the "upstream" allocations of multiplicative weight to its ancestors.

The qualitative analysis presented here is clearly not exhaustive, but hopefully is suggestive of the added power of this approach versus the greedy "Top-down" algorithm. That being said, in a sense this analysis does not "do our approach justice", in the sense that there is no off-the-shelf or automated way of implementing even the greedy approach (to our knowledge). In that sense, even the greedy approach would be a significant improvement versus the status quo, as it would at least provide an automated allocation solution (if not a precise one).

## 6 Conclusion

The demand for actionable business insights has never been bigger, with the global market for business intelligence and analytics software predicted to reach $14.5B annually by 2022

Figure 6: Comparison of the factor weight allocated to the *Pothole* category by the two approaches (secondary axis). Overall pothole complaint count shown in gray (primary axis) for context.



[18]. In addition, an estimated 2 to 4 million new human "data translators" will be needed in the next decade to visualize and interpret data patterns [10]. As the latter implies, a significant portion of data analyst effort is still spent on manual pattern identification techniques, à la "visual analytics".

In this work, we have presented the motivation and theory for FactorPrism, a novel methodology to solve a problem traditionally approached through such manual methods, but yet to be approached algorithmically. We have demonstrated that this task of automatic decomposition and factor attribution can be solved automatically and efficiently by showing this problem is equivalent to one of compressed sensing, namely an extension of *basis pursuit*. Finally, we have shown that this algorithm recovers satisfactory results in toy and real world cases. This approach has immediate practical applications, some of which are outlined below.

## 6.1 Applications

The use case alluded to throughout this paper - decomposition of macro- and micro- sales trends for descriptive analytics and business intelligence - is perhaps the most obvious. Potential clients for this use case include but are not limited to large retailers (online and brick-and-mortar), manufacturers with audit data and/or direct sales, and other intermediaries (e.g. pharmacy benefit managers).

The NYC 311 example alludes to a second class of use cases, where transaction counts are the unit of interest, again in terms of descriptive statistics. These could include public services, as illustrated here, or other operational situations, such as call centers, logistics carriers, etc.

We have yet to explore use cases in forecasting, but they do suggest themselves. Once the trends are decomposed using the method alluded to here, trends can be built on the decomposed factors, which, when combined, could potentially lead to more accurate overall forecasts than prior art. Clients for this use case include those previously mentioned, as well as financial forecasting (e.g. asset price forecasting).

## 6.2 Further Research

The most obvious theoretical future direction for research is to prove that FactorPrism is the best unbiased estimator in recovering these causes (e.g. vis a vis the accuracy function in 19).

Potential other avenues for research may look to more finely tune the algorithm in specific cases to control for noise. Given the closeness to *basis pursuit*, extensions of *basis pursuit denoising* (i.e. LASSO [20]), suggest themselves here and may possibly be beneficial.

Given the implicit emphasis on interpretability of solutions to this problem, it would be fruitful to prove their apparent uniqueness. We suspect such can be done by following the techniques of [21].

Along perhaps similar lines, fine-tailored solving techniques for this specific LP (Equation 16) may potentially lead to increases in performance and solution time.

A potential limitation of the algorithm is that it can only assign weights to pre-specified categories, and thus can only identify the most important factors if they are in the pre-specified set of labels. A further direction may be looking to implement this algorithm in cases where at least some of the categories are not known *a priori*. For instance, for continuous variables (e.g. age), we might seek to find the groupings of ages that would lead to the most parsimonious result. For example, one could imagine an algorithm that searches through these delineations, looking for an "optimum" (perhaps with lowest objective value) in a potentially efficient way.

## Acknowledgements

The author would like to thank the members of the Cornell Operations Research Department for their continuous inspiration and friendship over the years, especially David Shmoys and Huseyin Topaloglu. Additionally, thanks to Eran Toch, Gerry Feigin, Paul Raff, Todd Schiller, Rishab Guha, Eli Goldberg, and Gui Woolston for their helpful comments on this paper. Thanks to Byron Hing, Pete Rimshnick, and Ilya Sukhar for their advice with software creation, and Ray Dara and Mike Dill for their business partnership. Finally, thanks to the author's family, especially his wife Nicole, for their perpetual encouragement and support.

## References

[1] Elena Barton, Basad Al-Sarray, Stephane Chretien, and Kavya Jagan, *Decomposition of dynamical signals into jumps, oscillatory patterns, and possible outliers*, Mathematics **6** (2018), no. 7, 124.

[2] T Tony Cai and Lie Wang, *Orthogonal matching pursuit for sparse signal recovery with noise*, IEEE Transactions on Information theory **57** (2011), no. 7, 4680–4688.

[3] EJ Candes and T Tao, *Near-optimal signal recovery from random projections and universal encoding strategies. submitted to ieee trans*, Inform. Theory, November (2004).

[4] Scott Shaobing Chen and David L Donoho, *Application of basis pursuit in spectrum estimation*, Proceedings of the 1998 ieee international conference on acoustics, speech and signal processing, icassp'98 (cat. no. 98ch36181), 1998, pp. 1865–1868.

[5] Scott Shaobing Chen, David L Donoho, and Michael A Saunders, *Atomic decomposition by basis pursuit*, SIAM review **43** (2001), no. 1, 129–159.

[6] Robert B Cleveland, William S Cleveland, Jean E McRae, and Irma Terpenning, *Stl: A seasonal-trend decomposition*, Journal of official statistics **6** (1990), no. 1, 3–73.

[7] Estela Bee Dagum and Silvia Bianconcini, *Seasonal adjustment methods and real time trend-cycle estimation*, Springer, 2016.

[8] David L Donoho, *Compressed sensing*, IEEE Transactions on information theory **52** (2006), no. 4, 1289–1306.

[9] Andrew Gelman and Jennifer Hill, *Data analysis using regression and multilevel/hierarchical models*, Cambridge university press, 2006.

[10] Nicolaus Henke et al., *The age of analytics: competing in a data-driven world*, McKinsey Global Institute (2016).

[11] Rob J Hyndman and George Athanasopoulos, *Forecasting: principles and practice*, OTexts, 2018.

[12] Dirk A Lorenz, Marc E Pfetsch, and Andreas M Tillmann, *Solving basis pursuit: Heuristic optimality check and solver comparison*, ACM Transactions on Mathematical Software (TOMS) **41** (2015), no. 2, 1–29.

[13] Long Ma, Xiao Han, Zhesi Shen, Wen-Xu Wang, and Zengru Di, *Efficient reconstruction of heterogeneous networks from time series via compressed sensing*, PloS one **10** (2015), no. 11, e0142837.

[14] Hans D Mittelmann, *Latest benchmarks of optimization software*, Informs annual meeting. houston, tx, 2017.

[15] City of New York, *Getting started with open data*. Accessed: 2021-01-14.

[16] Hal Rankard, *Is there a geometric analog of absolute value?*.

[17] William F Sharpe, *Portfolio theory and capital markets*, McGraw-Hill College, 1970.

[18] Sebastian Stan, *Data analytics market in 2020: Trends, forecasts, and challenges*, Cognetik, 2020.

[19] Marina Theodosiou, *Forecasting monthly and quarterly time series using stl decomposition*, International Journal of Forecasting **27** (2011), no. 4, 1178–1195.

[20] Robert Tibshirani, *Regression shrinkage and selection via the lasso*, Journal of the Royal Statistical Society: Series B (Methodological) **58** (1996), no. 1, 267–288.

[21] Ryan J Tibshirani et al., *The lasso problem and uniqueness*, Electronic Journal of statistics **7** (2013), 1456–1490.

[22] Xiaozhe Wang, Kate Smith, and Rob Hyndman, *Characteristic based clustering for time series data*, Data mining and knowledge Discovery **13** (2006), no. 3, 335–364.

[23] Shanika L Wickramasuriya, George Athanasopoulos, and Rob J Hyndman, *Optimal forecast reconciliation for hierarchical and grouped time series through trace minimization*, Journal of the American Statistical Association **114** (2019), no. 526, 804–819.

## A Justification for Weighting

In this section we provide non-rigorous theoretical justification as to why we chose $W_i$ as the maximum of all microcategories data $Y_{j,0}$ where $i$ influences $j$ as our weight factor (as in Equation 11).

We assert that the weighting factors $W_i \in W$ should have the following properties:

1. **Microcategory weights are equal to starting data.** Each microcategory $j$ has weight equal to its own starting data $Y_{j,0}$.

2. **Weights generalize to equally-weighted case.** If the microcategory descendants $j \in Q_i$ of any non-microcategory $i$ have equal weight $W_j = w$, then $W_i = w$.

3. **Weights imply "single-impact axiom".** The weights are set that if only one microcategory descendant $k$ of $i$ has any data change (i.e. $k$ is the only member of $Q_i$ such that $Y_{k,t} - Y_{k,0} \neq 0$), then all $g_j$ except for $g_k$ (including $g_i$) will be optimized at 0 (and $g_k$ will be optimized at $log(Y_{k,t}) - log(Y_{k,0})$). (We refer to this algorithm behavior as the *single-impact axiom*). This implies that $W_i \geq \max_{j \in Q_i} W_j$.

Lemma 1 follows directly.

**Lemma 1** *$W_i = \max_{j \in Q_i} Y_{j,0}$ satisfies Properties 1-3.*

We also claim Lemma 2 without proof.

**Lemma 2** *$W_i = \max_{j \in Q_i} Y_{j,0}$ is the only continuous function satisfying Properties 1-3.*

### A.1 Justification of Property 3

Now, we justify Property 3, specifically how the "single-impact axiom" of the algorithm implies $W_i \geq \max_{j \in Q_i} W_j$. We should mention that while it is difficult to say a great deal *a priori* about the desired optimal solution to our algorithm, we can say for sure that we want this axiom, as described, to hold, and thus the weights to be designed to guarantee this.

Suppose we were in the "single-impact" scenario as described in Property 3 and illustrated in Figure 7. We claim that this solution being optimal implies $W_i \geq \max_{j \in Q_i} W_j$, where $i$ is the ancestor category as shown.

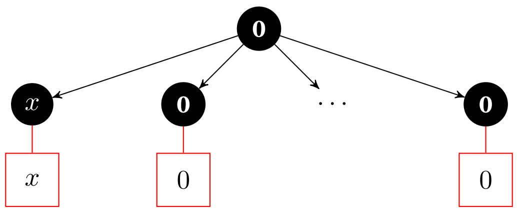


Figure 7: Desired solutions $g_i$ (root) and $g_j$ for $j \in Q_i$ when only one microcategory ($j = 1$) has any data change $x$. Note ordering is arbitrary and for illustration.

**"Proof."** Let $j = 1$ refer to the microcategory with a nonzero change $x$, and assume $x$ is positive for argument (the logic is symmetric if negative and ordering is arbitrary).

Since this solution is optimal, any $\delta$ increase in $g_i$ and corresponding decrease in $g_j$ for $j \in Q_i$ will increase (or keep same) the objective function. We note that all absolute values $|g|$ will increase by $\delta$ except $|g_1|$, which will decrease by $\delta$. Thus our optimality condition implies that

$$\delta W_i + \sum_{j=2}^{n} \delta W_j - \delta W_1 \geq 0 \tag{20}$$

and thus

$$W_i + \sum_{j=2}^{n} W_j \geq W_1 \tag{21}$$

For this inequality to hold in all cases, it must hold in the particular case when $W_1 = \max_{j \in Q_i} W_j$. Thus

$$W_i + \sum_{j=2}^{n} W_j \geq \max_{j \in Q_i} W_j \tag{22}$$

Since $\sum_{j=2}^{n} W_j$ can be arbitrarily small, we thus must have

$$W_i \geq \max_{j \in Q_i} W_j \tag{23}$$

which was our claim.