Where to put a facility?

Given locations $\mathbf{p}_1, \ldots, p_m$ in $\mathbb{R}^n$ of m houses, want to choose a location $\mathbf{c}$ in $\mathbb{R}^n$ for the fire station.

Want $\mathbf{c}$ to be as close as possible to all the house. We know how to measure distance between a proposed location $\mathbf{c}$ and a point. But different houses have different ideas about where to put the firehouse—how to combine their preferences into a single location?

- Choose the point that minimizes the *average* station-to-house distance. Same as minimizing the sum of station-to-house distances. This could be really bad for houses that are outside of the town center.
- Choose the point that minimizes the *maximum* station-to-house distance.This could be really bad for most of the houses!
- Choose the point that minimizes sum of squared station-to-house distances

  $$\|\mathbf{p}_1 - \mathbf{c}\|^2 + \|\mathbf{p}_2 - \mathbf{c}\|^2 + \|\mathbf{p}_3 - \mathbf{c}\|^2 + \cdots + \|\mathbf{p}_m - \mathbf{c}\|^2$$

  This is a sort of compromise—like average but if some house is very far away the squared distance is very large.

(These three different measures are called $\mathcal{L}_1, \mathcal{L}_\infty, \mathcal{L}_2$.)

Putting a facility in the location that minimizes sum of squared distances

Given locations $\mathbf{p}_1, \ldots, p_m$ in $\mathbb{R}^n$ of m houses, want to choose a location $\mathbf{c}$ in $\mathbb{R}^n$ for the fire station so as to minimize sum of squared distances

$$\|\mathbf{p}_1 - \mathbf{c}\|^2 + \|\mathbf{p}_2 - \mathbf{c}\|^2 + \cdots + \|\mathbf{p}_m - \mathbf{c}\|^2$$

- **Question:**How to find this location?
- **Answer:** $\mathbf{c} = \frac{1}{m}(\mathbf{p}_1 + \mathbf{p}_2 + \cdots + \mathbf{p}_m)$

Called the *centroid* of $\mathbf{p}_1, \ldots, \mathbf{p}_m$. It is the average for vectors. In fact, for $i = 1, \ldots, n$, entry i of the centroid is the average of entry i of all the points.

Centroid $\bar{\mathbf{p}}$ satisfies the equation $m\bar{\mathbf{p}} = \sum_i \mathbf{p}_i$.
Therefore $\sum_i (\mathbf{p}_i - \bar{\mathbf{p}})$ equals the zero vector.

Proving that the centroid minimizes the sum of squared distances

Let $\mathbf{q}$ be any point. We show that the sum of squared $\mathbf{q}$-to-datapoint distances is at least the sum of squared $\bar{\mathbf{p}}$-to-datapoint distances.

For $i = 1, \ldots, m$,

$$\begin{aligned}
\|\mathbf{p}_i - \mathbf{q}\|^2 &= \|\mathbf{p}_i - \bar{\mathbf{p}} + \bar{\mathbf{p}} - \mathbf{q}\|^2 \\
&= \langle \mathbf{p}_i - \bar{\mathbf{p}} + \bar{\mathbf{p}} - \mathbf{q}, \mathbf{p}_i - \bar{\mathbf{p}} + \bar{\mathbf{p}} - \mathbf{q} \rangle \\
&= \langle \mathbf{p}_i - \bar{\mathbf{p}}, \mathbf{p}_i - \bar{\mathbf{p}} \rangle + \langle \mathbf{p}_i - \bar{\mathbf{p}}, \bar{\mathbf{p}} - \mathbf{q} \rangle + \langle \bar{\mathbf{p}} - \mathbf{q}, \mathbf{p}_i - \bar{\mathbf{p}} \rangle + \langle \bar{\mathbf{p}} - \mathbf{q}, \bar{\mathbf{p}} - \mathbf{q} \rangle \\
&= \|\mathbf{p}_i - \bar{\mathbf{p}}\|^2 + \langle \mathbf{p}_i - \bar{\mathbf{p}}, \bar{\mathbf{p}} - \mathbf{q} \rangle + \langle \bar{\mathbf{p}} - \mathbf{q}, \mathbf{p}_i - \bar{\mathbf{p}} \rangle + \|\bar{\mathbf{p}} - \mathbf{q}\|^2
\end{aligned}$$

Summing over $i = 1, \ldots, m$,

$$\begin{aligned}
&\sum_i \|\mathbf{p}_i - \mathbf{q}\|^2 \\
&\quad = \sum_i \|\mathbf{p}_i - \bar{\mathbf{p}}\|^2 + \sum_i \langle \mathbf{p}_i - \bar{\mathbf{p}}, \bar{\mathbf{p}} - \mathbf{q} \rangle + \sum_i \langle \bar{\mathbf{p}} - \mathbf{q}, \mathbf{p}_i - \bar{\mathbf{p}} \rangle + \sum_i \|\bar{\mathbf{p}} - \mathbf{q}\|^2 \\
&\quad = \sum_i \|\mathbf{p}_i - \bar{\mathbf{p}}\|^2 + \left\langle \sum_i (\mathbf{p}_i - \bar{\mathbf{p}}), \bar{\mathbf{p}} - \mathbf{q} \right\rangle + \left\langle \bar{\mathbf{p}} - \mathbf{q}, \sum_i (\mathbf{p}_i - \bar{\mathbf{p}}) \right\rangle + \sum_i \|\bar{\mathbf{p}} - \mathbf{q}\|^2
\end{aligned}$$

Proving that the centroid minimizes the sum of squared distances

Let $\mathbf{q}$ be any point. We show that the sum of squared $\mathbf{q}$-to-datapoint distances is at least the sum of squared $\bar{\mathbf{p}}$-to-datapoint distances.

Summing over $i = 1, \ldots, m$,

$$\begin{aligned}
&\sum_i \|\mathbf{p}_i - \mathbf{q}\|^2 \\
&= \sum_i \|\mathbf{p}_i - \bar{\mathbf{p}}\|^2 + \sum_i \langle \mathbf{p}_i - \bar{\mathbf{p}}, \bar{\mathbf{p}} - \mathbf{q}\rangle + \sum_i \langle \bar{\mathbf{p}} - \mathbf{q}, \mathbf{p}_i - \bar{\mathbf{p}}\rangle + \sum_i \|\bar{\mathbf{p}} - \mathbf{q}\|^2 \\
&= \sum_i \|\mathbf{p}_i - \bar{\mathbf{p}}\|^2 + \left\langle \sum_i (\mathbf{p}_i - \bar{\mathbf{p}}), \bar{\mathbf{p}} - \mathbf{q} \right\rangle + \left\langle \bar{\mathbf{p}} - \mathbf{q}, \sum_i (\mathbf{p}_i - \bar{\mathbf{p}}) \right\rangle + \sum_i \|\bar{\mathbf{p}} - \mathbf{q}\|^2 \\
&= \sum_i \|\mathbf{p}_i - \bar{\mathbf{p}}\|^2 + \langle \mathbf{0}, \bar{\mathbf{p}} - \mathbf{q}\rangle + \langle \bar{\mathbf{p}} - \mathbf{q}, \mathbf{0}\rangle + \sum_i \|\bar{\mathbf{p}} - \mathbf{q}\|^2 \\
&= \sum_i \|\mathbf{p}_i - \bar{\mathbf{p}}\|^2 + 0 + 0 + \sum_i \|\bar{\mathbf{p}} - \mathbf{q}\|^2 \\
&= \text{sum of squared } \bar{\mathbf{p}}\text{-to-datapoint distances} + \text{squared } \bar{\mathbf{p}}\text{-to-}\mathbf{q} \text{ distance}
\end{aligned}$$

k-means clustering using Lloyd's Algorithm

***k*-means clustering:** Given data points (vectors) $\mathbf{p}_1, \ldots, \mathbf{p}_m$ in $\mathbb{R}^n$, select k centers $\mathbf{c}_1, \ldots, \mathbf{c}_k$ so as to minimize the sum of squared distances of data points to the nearest centers.

That is, define the function $f(\mathbf{x}, [c_1, \ldots, \mathbf{c}_k]) = \min\{\|\mathbf{x} - \mathbf{c}_i\|^2 \ : \ i \in \{1, \ldots, k\}\}$. This function returns the squared distance from $\mathbf{x}$ to whichever of $\mathbf{c}_1, \ldots, \mathbf{c}_k$ is nearest.

The goal of k-means clustering is to select points $\mathbf{c}_1, \ldots, \mathbf{c}_k$ so as to minimize

$$f(\mathbf{p}_1, [\mathbf{c}_1, \ldots, \mathbf{c}_k]) + f(\mathbf{p}_2, [\mathbf{c}_1, \ldots, \mathbf{c}_k]) + \cdots + f(\mathbf{p}_m, [\mathbf{c}_1, \ldots, \mathbf{c}_k])$$

The purpose is to partition the data points into k groups (called *clusters*).

k-means clustering using Lloyd's Algorithm

Select k centers to minimize sum of squared distances of data points to nearest centers.
Combines two ideas:

1. Assign each data point to the nearest center.
2. Choose the centers so as to be close to data points.

Suggests an algorithm. Start with k centers somewhere, perhaps randomly chosen. Then repeatedly perform the following steps:

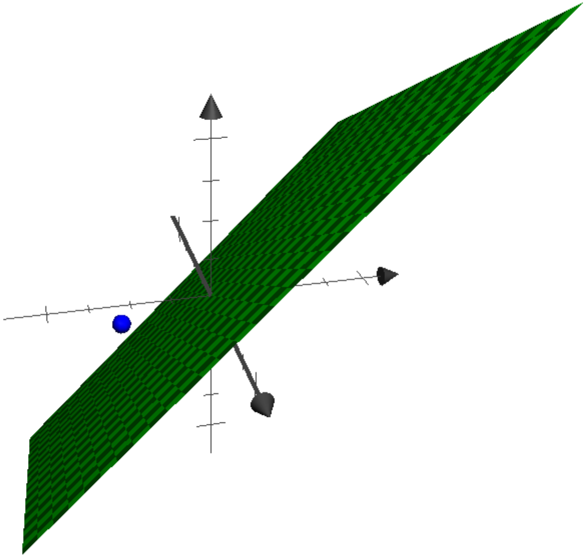
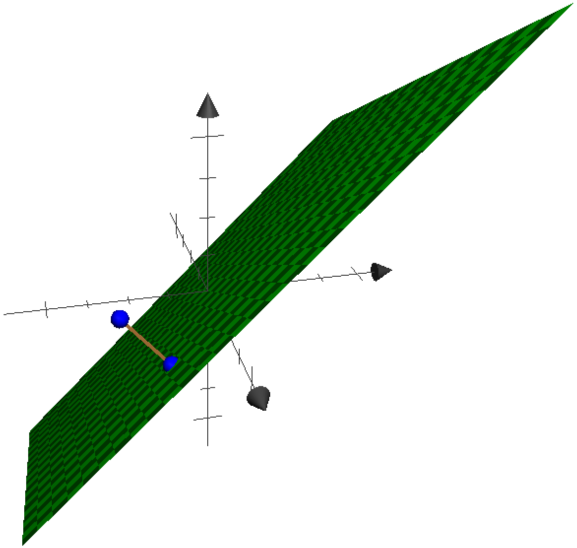
1. Assign each data point to nearest center.
2. Move each center to be as close as possible to the data points assigned to it This means let the new location of the center be the centroid of the assigned points.

Orthogonalization

[9] Orthogonalization

Finding the closest point in a plane

Goal: Given a point **b** and a plane, find the point in the plane closest to **b**.

Finding the closest point in a plane

Goal: Given a point $\mathbf{b}$ and a plane, find the point in the plane closest to $\mathbf{b}$.

By translation, we can assume the plane includes the origin.

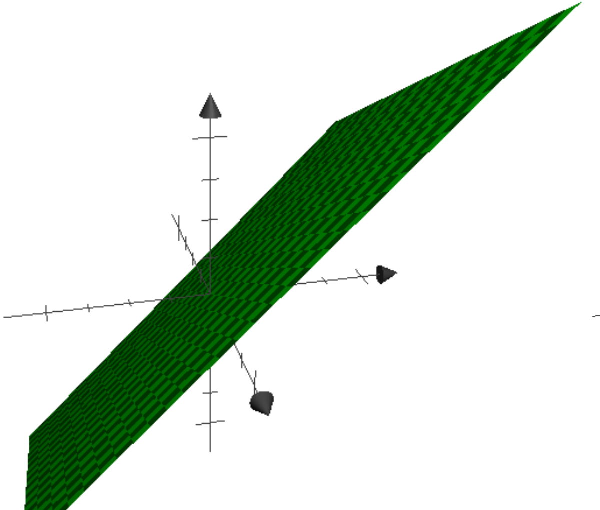
The plane is a vector space $\mathcal{V}$. Let $\{\mathbf{v}_1, \mathbf{v}_2\}$ be a basis for $\mathcal{V}$.

Goal: Given a point $\mathbf{b}$, find the point in Span $\{\mathbf{v}_1, \mathbf{v}_2\}$ closest to $\mathbf{b}$.

Example:

$$\mathbf{v}_1 = [8, -2, 2] \text{ and } \mathbf{v}_2 = [4, 2, 4]$$

$\mathbf{b} = [5, -5, 2]$
point in plane closest to $\mathbf{b}$: $[6, -3, 0]$.



Closest-point problem in higher dimensions

Goal: An algorithm that, given a vector $\mathbf{b}$ and vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$, finds the vector in Span $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ that is closest to $\mathbf{b}$.

Special case: We can use the algorithm to determine whether $\mathbf{b}$ lies in Span $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$: If the vector in Span $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ closest to $\mathbf{b}$ is $\mathbf{b}$ itself then clearly $\mathbf{b}$ is in the span; if not, then $\mathbf{b}$ is not in the span. Let $A = \left[\begin{array}{c|c|c} \mathbf{v}_1 & \cdots & \mathbf{v}_n \end{array}\right]$.

Using the linear-combinations interpretation of matrix-vector multiplication, a vector in Span $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ can be written $A\mathbf{x}$.

Thus testing if $\mathbf{b}$ is in Span $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ is equivalent to testing if the equation $A\mathbf{x} = \mathbf{b}$ has a solution.

More generally:

Even if $A\mathbf{x} = \mathbf{b}$ has no solution, we can use the algorithm to find the point in $\{A\mathbf{x} \;:\; \mathbf{x} \in \mathbb{R}^n\}$ closest to $\mathbf{b}$.

Moreover: We hope to extend the algorithm to also find the best solution $\mathbf{x}$.

High-dimensional projection onto/orthogonal to

For any vector $\mathbf{b}$ and any vector $\mathbf{a}$, define vectors $\mathbf{b}^{||\mathbf{a}}$ and $\mathbf{b}^{\perp\mathbf{a}}$ so that

$$\mathbf{b} = \mathbf{b}^{||\mathbf{a}} + \mathbf{b}^{\perp\mathbf{a}}$$

and there is a scalar $\sigma \in R$ such that

$\mathbf{b}^{||\mathbf{a}} = \sigma\,\mathbf{a}$ and $\mathbf{b}^{\perp\mathbf{a}}$ is orthogonal to $\mathbf{a}$

Definition: For a vector $\mathbf{b}$ and a vector space $\mathcal{V}$, we define the projection of $\mathbf{b}$ onto $\mathcal{V}$ (written $\mathbf{b}^{||\mathcal{V}}$) and the projection of $\mathbf{b}$ orthogonal to $\mathcal{V}$ (written $\mathbf{b}^{\perp\mathcal{V}}$) so that

$$\mathbf{b} = \mathbf{b}^{||\mathcal{V}} + \mathbf{b}^{\perp\mathcal{V}}$$

and $\mathbf{b}^{||\mathcal{V}}$ is in $\mathcal{V}$, and $\mathbf{b}^{\perp\mathcal{V}}$ is orthogonal to every vector in $\mathcal{V}$.

$$\begin{tikzcd}
 & & & \boldsymbol{b} \arrow[dl, "\text{projection onto } \mathcal{V}"'] \arrow[dr, "\text{projection orthogonal to } \mathcal{V}"] & \\
\boldsymbol{b} & = & \boldsymbol{b}^{||\mathcal{V}} & + & \boldsymbol{b}^{\perp\mathcal{V}}
\end{tikzcd}$$

High-Dimensional Fire Engine Lemma

Definition: For a vector $\mathbf{b}$ and a vector space $\mathcal{V}$, we define the projection of $\mathbf{b}$ onto $\mathcal{V}$ (written $\mathbf{b}^{||\mathcal{V}}$) and the projection of $\mathbf{b}$ orthogonal to $\mathcal{V}$ (written $\mathbf{b}^{\perp\mathcal{V}}$) so that

$$\mathbf{b} = \mathbf{b}^{||\mathcal{V}} + \mathbf{b}^{\perp\mathcal{V}}$$

and $\mathbf{b}^{||\mathcal{V}}$ is in $\mathcal{V}$, and $\mathbf{b}^{\perp\mathcal{V}}$ is orthogonal to every vector in $\mathcal{V}$.

One-dimensional Fire Engine Lemma: The point in Span $\{\mathbf{a}\}$ closest to $\mathbf{b}$ is $\mathbf{b}^{||\mathbf{a}}$ and the distance is $\|\mathbf{b}^{\perp\mathbf{a}}\|$.

High-Dimensional Fire Engine Lemma: The point in a vector space $\mathcal{V}$ closest to $\mathbf{b}$ is $\mathbf{b}^{||\mathcal{V}}$ and the distance is $\|\mathbf{b}^{\perp\mathcal{V}}\|$.

Finding the projection of $\mathbf{b}$ orthogonal to Span $\{\mathbf{a}_1, \ldots, a\mathbf{n}\}$

High-Dimensional Fire Engine Lemma: Let $\mathbf{b}$ be a vector and let $\mathcal{V}$ be a vector space. The vector in $\mathcal{V}$ closest to $\mathbf{b}$ is $\mathbf{b}^{||\mathcal{V}}$. The distance is $\|\mathbf{b}^{\perp\mathcal{V}}\|$.

Suppose $\mathcal{V}$ is specified by generators $\mathbf{v}_1, \ldots, \mathbf{v}_n$

Goal: An algorithm for computing $\mathbf{b}^{||\mathcal{V}}$ in this case.

- *input:* vector $\mathbf{b}$, vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$
- *output:* projection of $\mathbf{b}$ onto Span $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$

We already know how to solve this when $n = 1$:

```
def project_along(b, v):
    return (0 if v.is_almost_zero() else (b*v)/(v*v))*v
```

Let's try to generalize....

project_onto(b, vlist)

```
def project_along(b, v):
 return (0 if v.is_almost_zero() else (b*v)/(v*v))*v
```

$$\Downarrow$$

```
def project_onto(b, vlist):
  return sum([project_along(b, v) for v in vlist])
```

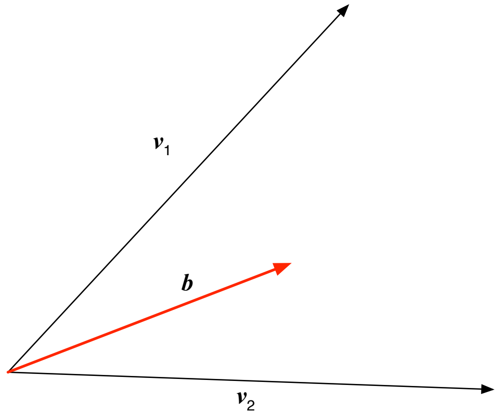
Reviews are in....

“Short, elegant, ... and flawed”

“Beautiful—if only it worked!”

“A tragic failure.”

Failure of `project_onto`



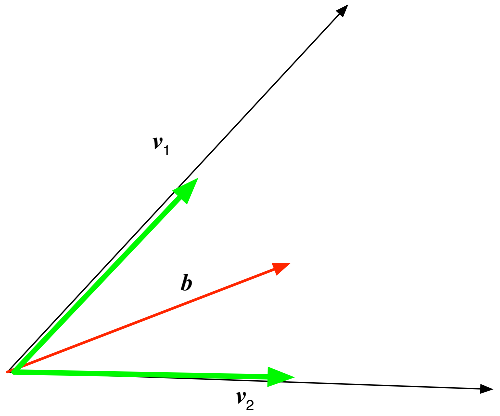
Try it out on vector $\mathbf{b}$ and `vlist` $= [\mathbf{v}_1, \mathbf{v}_2]$ in $\mathbb{R}^2$, so $\mathbf{v} =$ Span $\{\mathbf{v}_1, \mathbf{v}_2\}$.

In this case, $\mathbf{b}$ is in Span $\{\mathbf{v}_1, \mathbf{v}_2\}$, so $\mathbf{b}^{||\mathcal{V}} = \mathbf{b}$.

The algorithm tells us to find the projection of $\mathbf{b}$ along $\mathbf{v}_1$ and the projection of $\mathbf{b}$ along $\mathbf{v}_2$.

The sum of these projections should be equal to $\mathbf{b}^{||}$ ... but it's not.

Failure of `project_onto`



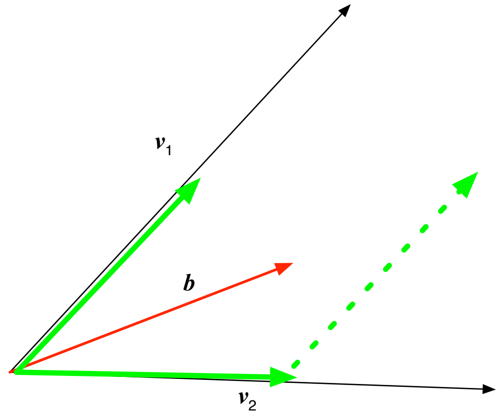
Try it out on vector $\mathbf{b}$ and `vlist` $= [\mathbf{v}_1, \mathbf{v}_2]$ in $\mathbb{R}^2$, so $\mathbf{v} = \text{Span } \{\mathbf{v}_1, \mathbf{v}_2\}$.

In this case, $\mathbf{b}$ is in Span $\{\mathbf{v}_1, \mathbf{v}_2\}$, so $\mathbf{b}^{||\mathcal{V}} = \mathbf{b}$.

The algorithm tells us to find the projection of $\mathbf{b}$ along $\mathbf{v}_1$ and the projection of $\mathbf{b}$ along $\mathbf{v}_2$.

The sum of these projections should be equal to $\mathbf{b}^{||}$ ... but it's not.

Failure of `project_onto`



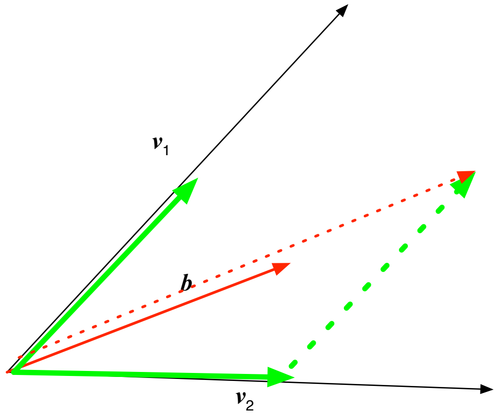
Try it out on vector $\mathbf{b}$ and `vlist` $= [\mathbf{v}_1, \mathbf{v}_2]$ in $\mathbb{R}^2$, so $\mathbf{v} = \text{Span } \{\mathbf{v}_1, \mathbf{v}_2\}$.

In this case, $\mathbf{b}$ is in Span $\{\mathbf{v}_1, \mathbf{v}_2\}$, so $\mathbf{b}^{||\mathcal{V}} = \mathbf{b}$.

The algorithm tells us to find the projection of $\mathbf{b}$ along $\mathbf{v}_1$ and the projection of $\mathbf{b}$ along $\mathbf{v}_2$.

The sum of these projections should be equal to $\mathbf{b}^{||}$ ... but it's not.

Failure of `project_onto`



Try it out on vector $\mathbf{b}$ and `vlist` $= [\mathbf{v}_1, \mathbf{v}_2]$ in $\mathbb{R}^2$, so $\mathbf{v} = \text{Span } \{\mathbf{v}_1, \mathbf{v}_2\}$.

In this case, $\mathbf{b}$ is in Span $\{\mathbf{v}_1, \mathbf{v}_2\}$, so $\mathbf{b}^{||\mathcal{V}} = \mathbf{b}$.

The algorithm tells us to find the projection of $\mathbf{b}$ along $\mathbf{v}_1$ and the projection of $\mathbf{b}$ along $\mathbf{v}_2$.

The sum of these projections should be equal to $\mathbf{b}^{||}$ ... but it's not.

What went wrong with `project_onto`?

Suppose we run the algorithm on $\mathbf{b}$ and $\texttt{vlist} = [\mathbf{v}_1, \ldots, \mathbf{v}_n]$. Let $\mathcal{V}$ denote Span $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$.
For each vector $\mathbf{v}_i \in \texttt{vlist}$, the vector returned by $\texttt{project\_along}(\mathbf{b}, \mathbf{v}_i)$ is $\sigma_i \mathbf{v}_i$
where σ_i is $\frac{\mathbf{b} \cdot \mathbf{v}_i}{\mathbf{v}_i \cdot \mathbf{v}_i}$ (or 0, if $\mathbf{v}_i = \mathbf{0}$).
The vector returned by $\texttt{project\_onto}(\mathbf{b}, \texttt{vlist})$ is the sum $\sigma_1 \mathbf{v}_1 + \sigma_2 \mathbf{v}_2 + \cdots + \sigma_n \mathbf{v}_n$.
Let $\hat{\mathbf{b}}$ denote the returned vector. Want to check that $\hat{\mathbf{b}}$ is $\mathbf{b}^{||\mathcal{V}}$ ...

- Is $\hat{\mathbf{b}}$ in $\mathcal{V}$? It is a linear combination of $\mathbf{v}_1, \ldots, \mathbf{v}_n$, so YES.
- If $\hat{\mathbf{b}}$ were $\mathbf{b}^{||\mathcal{V}}$ then $\mathbf{b} - \hat{\mathbf{b}}$ would be $\mathbf{b}^{\perp \mathcal{V}}$ so $\mathbf{b} - \hat{\mathbf{b}}$ would be orthogonal to all vectors in $\mathcal{V}$. In particular, it would be orthogonal to the generators $\mathbf{v}_1, \ldots, \mathbf{v}_n$. Is it? To check, calculate the inner product of $\mathbf{b} - \hat{\mathbf{b}}$ with each of $\mathbf{v}_1, \ldots, \mathbf{v}_n$.

Consider, for example, the generator $\mathbf{v}_1$.

$$\begin{aligned}\left\langle \mathbf{b} - \hat{\mathbf{b}}, \mathbf{v}_1 \right\rangle &= \langle \mathbf{b}, \mathbf{v}_1 \rangle - \left\langle \hat{\mathbf{b}}, \mathbf{v}_1 \right\rangle \\ &= \langle \mathbf{b}, \mathbf{v}_1 \rangle - \langle \sigma_1 \mathbf{v}_1 + \sigma_2 \mathbf{v}_2 + \cdots + \sigma_n \mathbf{v}_n, \mathbf{v}_1 \rangle\end{aligned}$$

Is this zero? Expand the last inner product—get $\langle \sigma_1 \mathbf{v}_1, \mathbf{v}_1 \rangle$ but also cross-terms like $\langle \sigma_2 \mathbf{v}_1, \mathbf{v}_1 \rangle$ and $\langle \sigma_n \mathbf{v}_n, \mathbf{v}_1 \rangle$. The cross-terms keep the algorithm from working correctly.

How to repair `project_onto`?

Don't change the procedure. Fix the spec.

Require that `vlist` consists of **mutually orthogonal** vectors:

the i^{th} vector in the list is orthogonal to the j^{th} vector in the list for every $i \neq j$.