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

$$\mathbf{b}^{||\mathbf{a}} = \sigma\,\mathbf{a}$$

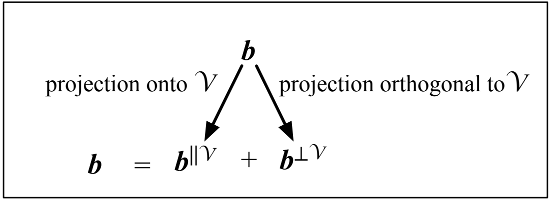
and

$$\mathbf{b}^{\perp\mathbf{a}} \text{ is orthogonal to } \mathbf{a}$$

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 $\mathcal{V}$.



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 $\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$

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 $\mathbf{b}^{\perp\mathcal{V}}$ in this case.

- *input:* vector $\mathbf{b}$, vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$
- *output:* projection of $\mathbf{b}$ orthogonal to $\mathcal{V} = \text{Span }\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$

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

```
def project_orthogonal_1(b, v):
 return b - project_along(b, v)
```

Let's try to generalize....

project_orthogonal(b, vlist)

```
def project_orthogonal_1(b, v):
 return b - project_along(b, v)
```

$\Downarrow$

```
def project_orthogonal(b, vlist):
  for v in vlist:
     b = b - project_along(b, v)
  return b
```

Reviews are in....

"Short, elegant, ... and flawed"

"Beautiful—if only it worked!"

"A tragic failure."

project_orthogonal(b, vlist) doesn't work

```
def project_orthogonal(b, vlist):
  for v in vlist:
     b = b - project_along(b, v)
  return b
```

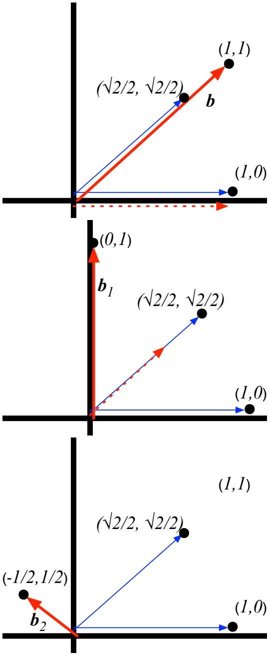
$\mathbf{b} = [1,1]$

vlist $= [\ [1,0],\ [\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}]\ ]$

Let $\mathbf{b}_i$ be value of the variable $\mathbf{b}$ after i iterations.

$$
\begin{aligned}
\mathbf{b}_1 &= \mathbf{b}_0 - (\text{projection of } [1,1] \text{ along } [1,0]) \\
&= \mathbf{b}_0 - [1,0] \\
&= [0,1]
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{b}_2 &= \mathbf{b}_1 - (\text{projection of } [0,1] \text{ along } [\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}]) \\
&= \mathbf{b}_1 - [\frac{1}{2}, \frac{1}{2}] \\
&= [-\frac{1}{2}, \frac{1}{2}] \text{ which is not orthogonal to } [1,0]
\end{aligned}
$$



How to repair `project_orthogonal(b, vlist)`?

```
def project_orthogonal(b, vlist):
  for v in vlist:
     b = b - project_along(b, v)
  return b
```

$\mathbf{b} = [1,1]$
`vlist` $=[\ [1,0],$
$[\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}]\ ]$

Final vector is not orthogonal to $[1,0]$

Maybe the problem will go away if the algorithm

- first finds the projection of **b** along each of the vectors in `vlist`, and
- only afterwards subtracts all these projections from **b**.

```
def classical_project_orthogonal(b, vlist):
  w = all-zeroes-vector
  for v in vlist:
    w = w + project_along(b, v)
  return b - w
```

Alas, this procedure also does not work. For the inputs

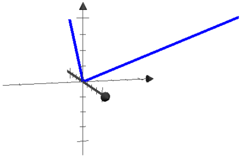
$$\mathbf{b} = [1,1], \texttt{vlist} = [\ [1,0], [\tfrac{\sqrt{2}}{2}, \tfrac{\sqrt{2}}{2}]\ ]$$

the output vector is $[-1, 0]$
which is orthogonal to neither of the two vectors in `vlist`.

What to do with `project_orthogonal(b, vlist)`?

Try it with two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ that are orthogonal...

$$
\begin{aligned}
\mathbf{v}_1 &= [1, 2, 1] \\
\mathbf{v}_2 &= [-1, 0, 1]
\end{aligned}
$$



$\mathbf{b}$

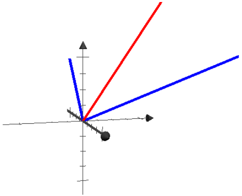
$\mathbf{b}_1$

$\mathbf{b}_2$

What to do with `project_orthogonal(b, vlist)`?

Try it with two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ that are orthogonal...

$$
\begin{aligned}
\mathbf{v}_1 &= [1, 2, 1] \\
\mathbf{v}_2 &= [-1, 0, 1] \\
\mathbf{b} &= [1, 1, 2] \\
\mathbf{b}_1 &= \mathbf{b}_0 - \frac{\mathbf{b}_0 \cdot \mathbf{v}_1}{\mathbf{v}_1 \cdot \mathbf{v}_1}\,\mathbf{v}_1 \\
&= [1, 1, 2] - \frac{5}{6}\,[1, 2, 1]
\end{aligned}
$$

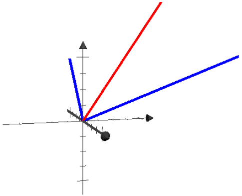


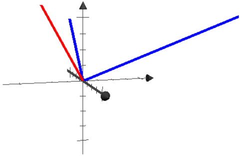
$\mathbf{b}_2$

What to do with `project_orthogonal(b, vlist)`?

Try it with two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ that are orthogonal...

$$
\begin{aligned}
\mathbf{v}_1 &= [1, 2, 1] \\
\mathbf{v}_2 &= [-1, 2, -1] \\
\mathbf{b} &= [1, 1, 2] \\
\mathbf{b}_1 &= \mathbf{b}_0 - \frac{\mathbf{b}_0 \cdot \mathbf{v}_1}{\mathbf{v}_1 \cdot \mathbf{v}_1}\,\mathbf{v}_1 \\
&= [1, 1, 2] - \frac{5}{6}\,[1, 2, 1] \\
&= \left[\frac{1}{6}, -\frac{4}{6}, \frac{7}{6}\right] \\
\mathbf{b}_2 &= \mathbf{b}_1 - \frac{\mathbf{b}_1 \cdot \mathbf{v}_2}{\mathbf{v}_2 \cdot \mathbf{v}_2}\,\mathbf{v}_2 \\
&= \left[\frac{1}{6}, -\frac{4}{6}, \frac{7}{6}\right] - \frac{1}{2}\,[-1, 0, 1]
\end{aligned}
$$

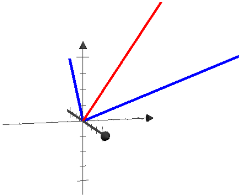




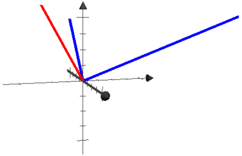
What to do with `project_orthogonal(b, vlist)`?

Try it with two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ that are orthogonal...

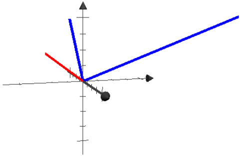
$$\begin{aligned}
\mathbf{v}_1 &= [1, 2, 1] \\
\mathbf{v}_2 &= [-1, 0, 1] \\
\mathbf{b} &= [1, 1, 2] \\
\mathbf{b}_1 &= \mathbf{b}_0 - \frac{\mathbf{b}_0 \cdot \mathbf{v}_1}{\mathbf{v}_1 \cdot \mathbf{v}_1}\, \mathbf{v}_1
\end{aligned}$$



$$\begin{aligned}
&= [1, 1, 2] - \frac{5}{6}\, [1, 2, 1] \\
&= \left[\frac{1}{6}, -\frac{4}{6}, \frac{7}{6}\right]
\end{aligned}$$



$$\begin{aligned}
\mathbf{b}_2 &= \mathbf{b}_1 - \frac{\mathbf{b}_1 \cdot \mathbf{v}_2}{\mathbf{v}_2 \cdot \mathbf{v}_2}\, \mathbf{v}_2 \\
&= \left[\frac{1}{6}, -\frac{4}{6}, \frac{7}{6}\right] - \frac{1}{2}\, [-1, 0, 1] \\
&= \left[\frac{2}{3}, -\frac{2}{3}, \frac{2}{3}\right]
\end{aligned}$$



and note $\mathbf{b}_2$ is orthogonal to $\mathbf{v}_1$ and $\mathbf{v}_2$.

Maybe `project_orthogonal(b, vlist)` works with $\mathbf{v}_1, \mathbf{v}_2$ orthogonal?

Assume $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$.
Remember: $\mathbf{b}_i$ is value of `b` after i iterations

First iteration:

$$\mathbf{b}_1 = \mathbf{b}_0 - \sigma_1 \, \mathbf{v}_1$$

gives $\mathbf{b}_1$ such that $\langle \mathbf{v}_1, \mathbf{b}_1 \rangle = 0$.

Second iteration:

$$\mathbf{b}_2 = \mathbf{b}_1 - \sigma_1 \, \mathbf{v}_2$$

gives $\mathbf{b}_2$ such that $\langle \mathbf{v}_2, \mathbf{b}_2 \rangle = 0$

But what about $\langle \mathbf{v}_1, \mathbf{b}_2 \rangle$?

$$\begin{aligned}
\langle \mathbf{v}_1, \mathbf{b}_2 \rangle &= \langle \mathbf{v}_1, \mathbf{b}_1 - \sigma \; \mathbf{v}_2 \rangle \\
&= \langle \mathbf{v}_1, \mathbf{b}_1 \rangle - \langle \mathbf{v}_1, \sigma \; \mathbf{v}_2 \rangle \\
&= \langle \mathbf{v}_1, \mathbf{b}_1 \rangle - \sigma \; \langle \mathbf{v}_1, \mathbf{v}_2 \rangle \\
&= 0 + 0
\end{aligned}$$

Thus $\mathbf{b}_2$ is orthogonal to $\mathbf{v}_1$ and $\mathbf{v}_2$

Don't fix `project_orthogonal(b, vlist)`. Fix the spec.

```
def project_orthogonal(b, vlist):
  for v in vlist:
     b = b - project_along(b, v)
  return b
```

Instead of trying to fix the flaw by changing the procedure, we will change the spec we expect the procedure to fulfill.

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 $i \neq j$.

New spec:

- *input:* a vector $\mathbf{b}$, and a list `vlist` of *mutually orthogonal* vectors
- *output:* the projection $\mathbf{b}^{\perp}$ of $\mathbf{b}$ orthogonal to the vectors in `vlist`

Loop invariant of `project_orthogonal(b, vlist)`

```
def project_orthogonal(b, vlist):
  for v in vlist:
     b = b - project_along(b, v)
  return b
```

Loop invariant: Let `vlist` $= [\mathbf{v}_1, \ldots, \mathbf{v}_n]$
For $i = 0, \ldots, n$, let $\mathbf{b}_i$ be the value of the variable b after i iterations. Then $\mathbf{b}_i$ is the projection of $\mathbf{b}$ orthogonal to Span $\{\mathbf{v}_1, \ldots, \mathbf{v}_i\}$. That is,

- $\mathbf{b}_i$ is orthogonal to the first i vectors of `vlist`, and
- $\mathbf{b} - \mathbf{b}_i$ is in the span of the first i vectors of `vlist`

We use induction to prove the invariant holds.

For $i = 0$, the invariant is trivially true:

- $\mathbf{b}_0$ is orthogonal to each of the first 0 vectors (every vector is), and
- $\mathbf{b} - \mathbf{b}_0$ is in the span of the first 0 vectors (because $\mathbf{b} - \mathbf{b}_0$ is the zero vector).

Proof of loop invariant of project_orthogonal(b, $[\mathbf{v}_1, \ldots, \mathbf{v}_n]$)

$\mathbf{b}_i$ = projection of $\mathbf{b}$ orthogonal to Span $\{\mathbf{v}_1, \ldots, \mathbf{v}_i\}$:

- $\mathbf{b}_i$ is orthogonal to $\mathbf{v}_1, \ldots, \mathbf{v}_i$, and
- $\mathbf{b} - \mathbf{b}_i$ is in Span $\{\mathbf{v}_1, \ldots, \mathbf{v}_i\}$

```
for v in vlist:
  b = b - project_along(b, v)
```

Assume invariant holds for $i = k-1$ iterations, and prove it for $i = k$ iterations.

In k^{th} iteration, algorithm computes $\mathbf{b}_k = \mathbf{b}_{k-1} - \sigma_k\,\mathbf{v}_k$
By induction hypothesis, $\mathbf{b}_{k-1}$ is the projection of $\mathbf{b}$ orthogonal to Span $\{\mathbf{v}_1, \ldots, \mathbf{v}_{k-1}\}$
Must prove

- $\mathbf{b}_k$ is orthogonal to $\mathbf{v}_1, \ldots, \mathbf{v}_k$, ✓
- and $\mathbf{b} - \mathbf{b}_k$ is in Span $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$

Choice of σ_k ensures that $\mathbf{b}_k$ is orthogonal to $\mathbf{v}_k$.
Must show $\mathbf{b}_k$ also orthogonal to $\mathbf{v}_j$ for $j = 1, \ldots, k-1$

$$\begin{aligned}
\langle \mathbf{b}_k, \mathbf{v}_j \rangle &= \langle \mathbf{b}_{k-1} - \sigma_k \mathbf{v}_k, \mathbf{v}_j \rangle \\
&= \langle \mathbf{b}_{k-1}, \mathbf{v}_j \rangle - \sigma_k \, \langle \mathbf{v}_k, \mathbf{v}_j \rangle \\
&= 0 - \sigma_k \, \langle \mathbf{v}_k, \mathbf{v}_j \rangle && \text{by the inductive hypothesis} \\
&= 0 - \sigma_k 0 && \text{by mutual orthogonality}
\end{aligned}$$

Shows $\mathbf{b}_k$ orthogonal to $\mathbf{v}_1, \ldots, \mathbf{v}_k$

Proof of loop invariant of project_orthogonal(b, $[\mathbf{v}_1, \ldots, \mathbf{v}_n]$)

$\mathbf{b}_i$ = projection of $\mathbf{b}$ orthogonal to Span $\{\mathbf{v}_1, \ldots, \mathbf{v}_i\}$:

- $\mathbf{b}_i$ is orthogonal to $\mathbf{v}_1, \ldots, \mathbf{v}_i$, and
- $\mathbf{b} - \mathbf{b}_i$ is in Span $\{\mathbf{v}_1, \ldots, \mathbf{v}_i\}$

```
for v in vlist:
    b = b - project_along(b, v)
```

Assume invariant holds for $i = k - 1$ iterations, and prove it for $i = k$ iterations.

In k^{th} iteration, algorithm computes $\mathbf{b}_k = \mathbf{b}_{k-1} - \sigma_k \mathbf{v}_k$

By induction hypothesis, $\mathbf{b}_{k-1}$ is the projection of $\mathbf{b}$ orthogonal to Span $\{\mathbf{v}_1, \ldots, \mathbf{v}_{k-1}\}$

Must prove

- $\mathbf{b}_k$ is orthogonal to $\mathbf{v}_1, \ldots, \mathbf{v}_k$, ✓
- and $\mathbf{b} - \mathbf{b}_k$ is in Span $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ ✓

Now we prove $\mathbf{b} - \mathbf{b}_k$ is in Span $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$

$$
\begin{aligned}
\mathbf{b} - \mathbf{b}_k &= \mathbf{b} - (\mathbf{b}_{k-1} - \sigma_k \mathbf{v}_k) && \text{by algorithm} \\
&= (\mathbf{b} - \mathbf{b}_{k-1}) + \sigma_k \mathbf{v}_k \\
&= (\text{a vector in Span } \{\mathbf{v}_1, \ldots, \mathbf{v}_{k-1}\}) + \alpha_k \mathbf{v}_k && \text{by inductive hypothesis} \\
&= \text{a vector in Span } \{\mathbf{v}_1, \ldots, \mathbf{v}_k\})
\end{aligned}
$$

Correctness of `project_orthogonal(b, vlist)`

```
def project_orthogonal(b, vlist):
    for v in vlist:
        b = b - project_along(b, v)
    return b
```

We have proved:

If $\mathbf{v}_1, \ldots, \mathbf{v}_n$ are mutually orthogonal then
output of $\texttt{project\_orthogonal}(\mathbf{b}, [\mathbf{v}_1, \ldots, \mathbf{v}_n])$ is the vector $\mathbf{b}^\perp$ such that

- $\mathbf{b} = \mathbf{b}^{||} + \mathbf{b}^\perp$
- $\mathbf{b}^{||}$ is in Span $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$
- $\mathbf{b}^\perp$ is orthogonal to $\mathbf{v}_1, \ldots, \mathbf{v}_n$.

Change to zero-based indexing::

If $\mathbf{v}_0, \ldots, \mathbf{v}_n$ are mutually orthogonal then
output of $\texttt{project\_orthogonal}(\mathbf{b}, [\mathbf{v}_0, \ldots, \mathbf{v}_n])$ is the vector $\mathbf{b}^\perp$ such that

- $\mathbf{b} = \mathbf{b}^{||} + \mathbf{b}^\perp$
- $\mathbf{b}^{||}$ is in Span $\{\mathbf{v}_0, \ldots, \mathbf{v}_n\}$
- $\mathbf{b}^\perp$ is orthogonal to $\mathbf{v}_0, \ldots, \mathbf{v}_n$.

Augmenting `project_orthogonal`

Since $\mathbf{b}^{||} = \mathbf{b} - \mathbf{b}^{\perp}$ is in Span $\{\mathbf{v}_0, \ldots, \mathbf{v}_n\}$, there are coefficients $\alpha_0, \ldots, \alpha_n$ such that

$$\mathbf{b} - \mathbf{b}^{\perp} = \alpha_0\, \mathbf{v}_0 + \cdots + \alpha_n\, \mathbf{v}_n$$

$$\mathbf{b} = \alpha_0\, \mathbf{v}_0 + \cdots + \alpha_n\, \mathbf{v}_n + 1\, \mathbf{b}^{\perp}$$

Write as

$$\left[\begin{array}{c} \mathbf{b} \end{array}\right] = \left[\begin{array}{c|c|c|c} \mathbf{v}_0 & \cdots & \mathbf{v}_n & \mathbf{b}^{\perp} \end{array}\right] \left[\begin{array}{c} \alpha_0 \\ \vdots \\ \alpha_n \\ 1 \end{array}\right]$$

The procedure `project_orthogonal(b, vlist)` can be augmented to output the vector of coefficients.

For technical reasons, we will represent the vector of coefficents as a dictionary, not a `Vec`.

Augmenting `project_orthogonal`

$$\begin{bmatrix} \mathbf{b} \end{bmatrix} = \left[\begin{array}{c|c|c|c} \mathbf{v}_0 & \cdots & \mathbf{v}_n & \mathbf{b}^{\perp} \end{array}\right] \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_n \\ 1 \end{bmatrix}$$

We reuse code from two prior procedures.

```
def project_along(b, v):
 sigma = ((b*v)/(v*v)) \
      if v*v != 0 else 0
 return sigma * v

def project_orthogonal(b, vlist):
  for v in vlist:
     b = b - project_along(b, v)
  return b
```

Must create and populate a dictionary.

- One entry for each vector in `vlist`
- One additional entry, 1, for $\mathbf{b}^{\perp}$

Initialize dictionary with the additonal entry.

```
def aug_project_orthogonal(b, vlist):
 alphadict = {len(vlist):1}
 for i in range(len(vlist)):
   v = vlist[i]
   sigma = (b*v)/(v*v) \
           if v*v > 0 else 0
   alphadict[i] = sigma
   b = b - sigma*v
 return (b, alphadict)
```