– skip

If a matrix used to be the transformation

$$\begin{bmatrix} a_{11} & a_{12} \cdots a_{1n} \\ \vdots & \\ a_{m1} & \cdots a_{mn} \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} = \begin{pmatrix} a_{11} \\ \vdots \\ a_{m1} \end{pmatrix} v_1 + \begin{pmatrix} a_{12} \\ \vdots \\ a_{m2} \end{pmatrix} v_2 + \cdots$$

then adding two matrices before doing the multiplication effectively does the same thing as multiplying them individually then adding the results.

$$\begin{bmatrix} a_{11}+b_{11} & \cdots a_{1n}+b_{1n} \\ \vdots & \\ a_{m1}+b_{m1} & \cdots a_{mn}+b_{mn} \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} = \begin{pmatrix} a_{11}+b_{11} \\ \vdots \\ a_{m1}+b_{m1} \end{pmatrix} v_1 + \cdots = \begin{pmatrix} a_{11} \\ \vdots \\ a_{m1} \end{pmatrix} v_1 + \begin{pmatrix} b_{11} \\ \vdots \\ b_{m1} \end{pmatrix} v_1 + \cdots$$

This [redacted] is the "preservation of structure" the homomorphism def. talks about.

If we perform <u>matrix multiplication</u> this effectively composes the two transformations. (Rotations were an example.) But composing functions isn't normally <u>commutative</u>, or able to be performed in any order, and neither is matrix multiplication, (with the same results)

For example,

$$\underset{A}{\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}} \underset{B}{\begin{bmatrix} 2 & 3 \\ 1 & 4 \end{bmatrix}} = \begin{bmatrix} 4 & 11 \\ 10 & 25 \end{bmatrix} \neq \underset{B}{\begin{bmatrix} 2 & 3 \\ 1 & 4 \end{bmatrix}} \underset{A}{\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}} = \begin{bmatrix} 11 & 16 \\ 13 & 18 \end{bmatrix}$$

But function composition <u>is</u> associative – if I apply function f, then g, then h, that's the same as combining f & g into one function of g&h – and so is matrix multi: $(FG)H = F(GH)$.

Matrix math is also distributive: $F(G+H) = FG + FH$. And there is a matrix that is a multiplicative identity – a matrix that leaves a matrix unchanged when you multiply it. Or at least, there's one for every square matrix size n.

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \ddots & & 0 \\ 0 & 0 & 0 & \cdots & & 1 & 0 \end{bmatrix}$$

It's a "diagonal matrix" where the only nonzero entries are along the row = column diagonal.

Inverses

We mentioned earlier that we want to be able to make matrices that undo the action of other matrices. ~~[illegible]~~ Because some matrices are "one-way streets" that lose information by losing dimensions, not all matrices have inverses.

There are 3 kinds of inverses, though usually one kind is meant in particular. If $GH = I$, G is a "left inverse" of H. If $HG = I$, G is a "right inverse" of H. And if both are true, G is the "two-sided inverse" – often called just the inverse and denoted H^{-1}.

For an example of an inverse that works one way but not another, consider transformation h which just moves from 2D to 3D and g which goes back down to 2D:

$$\begin{pmatrix} x \\ y \end{pmatrix} \xrightarrow{h} \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} \xrightarrow{g} \begin{pmatrix} x \\ y \end{pmatrix}$$

we skipped this, don't worry about it

We can do this with matrices — they're homomorphisms —

$$H\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} \qquad G\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} x \\ y \\ 0 \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix}$$

Removing the zero that was added is easy. $GH\vec{v} = \vec{v}$ so $GH = I$ and G is a left inverse.

But switching the order gives a transformation HG:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The composition in the other order lops off the last coordinate, zeroing it. Which makes sense because that information is gone.

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \xrightarrow{g} \begin{pmatrix} x \\ y \end{pmatrix} \xrightarrow{???} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

This doesn't happen if the transformation being inverted is an isomorphism, because the kernel didn't remove any dimensions, and the mapping was one-to-one.

If a matrix is nonsingular, it has an inverse

that works on both the left and the right. The matrix is then called "invertible." If H is invertible, H^{-1} exists and $HH^{-1} = H^{-1}H = I$.

If we can invert a transformation, it makes sense we should be able to invert a series of transformations by undoing them last to first. So: if G and H are invertible, GH is invertible and $(GH)^{-1} = H^{-1}G^{-1}$. (G is effectively the last applied, so it's the first inverted.)

A general way to find an inverse is to solve for it using Gauss's method. If we know for example,

$$\begin{bmatrix} m & n \\ p & q \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 2 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

(our inverse)

this implies the linear equations

$$\begin{aligned} m+2n &= 1 \\ m-n &= 0 \\ p+2q &= 0 \\ p-q &= 1 \end{aligned}$$

Now we could turn this right back into something to solve with Gauss-Jordan,

$$\left[\begin{array}{cccc|c} 1 & 2 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 \\ 0 & 0 & 1 & -1 & 1 \end{array}\right]$$

That's actually a little repetitive, though — notice how we duplicated our original matrix and we'd need to ~~[illegible]~~ reduce both. Instead, we can just write it like this:

$$\left[\begin{array}{cc|cc} 1 & 2 & 1 & 0 \\ 1 & -1 & 0 & 1 \end{array}\right]$$

(identity)

And perform the reduction operations in a way that affects the whole row, thus manipulating the right hand side numbers simultaneously.

$$\left[\begin{array}{cc|cc} 1 & 2 & 1 & 0 \\ 0 & -3 & -1 & 1 \end{array}\right] \to \left[\begin{array}{cc|cc} 1 & 2 & 1 & 0 \\ 0 & 1 & 1/3 & -1/3 \end{array}\right]$$

$$\left[\begin{array}{cc|cc} 1 & 0 & 1/3 & 2/3 \\ 0 & 1 & 1/3 & -1/3 \end{array}\right]$$

This is the desired inverse. Check:

$$\begin{bmatrix} 1 & 2 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 1/3 & 2/3 \\ 1/3 & -1/3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Another way of ~~[illegible]~~ looking at what's going on here is the following: each Gauss-Jordan operation is itself represented by a matrix. On the left, we're composing these with the original matrix until we get the identity. On the right, we're piling these onto the identity until we stop, at which point, we can look there to see what the inverse is.

(end week 3)

More inverse examples

21 ext

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 4 & 1 & 0 & 0 \\ 1 & 2 & 5 & 0 & 1 & 0 \\ 1 & 3 & -1 & 0 & 0 & 1 \end{array}\right] \rightarrow \left[\begin{array}{ccc|ccc} 1 & 2 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 1 & 0 \\ 0 & 1 & -5 & -1 & 0 & 1 \end{array}\right]$$

$$\downarrow$$

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 4 & 1 & 0 & 0 \\ 0 & 1 & 0 & -6 & 5 & 1 \\ 0 & 0 & 1 & -1 & 1 & 0 \end{array}\right]$$

$$\downarrow$$

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 0 & 5 & -4 & 0 \\ 0 & 1 & 0 & -6 & 5 & 1 \\ 0 & 0 & 1 & -1 & 1 & 0 \end{array}\right]$$

$$\downarrow$$

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 17 & -14 & -2 \\ 0 & 1 & 0 & -6 & 5 & 1 \\ 0 & 0 & 1 & -1 & 1 & 0 \end{array}\right]$$

Check: $\begin{bmatrix} 1 & 2 & 4 \\ 1 & 2 & 5 \\ 1 & 3 & -1 \end{bmatrix} \begin{bmatrix} 17 & -14 & -2 \\ -6 & 5 & 1 \\ -1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ ✓

$\begin{bmatrix} 1 & 2 & 4 \\ 1 & 2 & 5 \\ 1 & 2 & -1 \end{bmatrix}$: no inverse b/c columns not independent;

also, if we started, we'd get to

$$\left[\begin{array}{ccc|c} 1 & 2 & 4 & \\ 0 & 0 & 1 & \cdots \\ 0 & 0 & -5 & \end{array}\right]$$

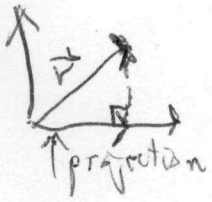
and realize the rank wasn't 3

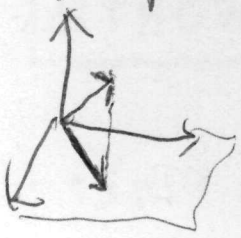
Week 4

This week we'll look at
- projections used to turn 3D into 2D or otherwise reduce dimensions
- Finding a line of best fit - "least squares"
- the determinant, which can tell us whether a matrix is singular, and is used for various other computations

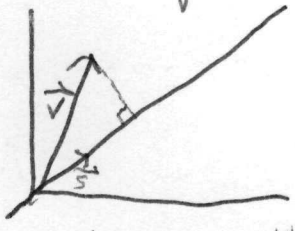
Projections (onto lines at first)

A projection tries to capture a higher-dimensional vector with a low-dimensional representation. In 1D or 2D, this looks like casting a shadow on the x-axis or xy-plane, respectively. The vector from projection to original vector is at a right angle to the projection surface (orthogonal).



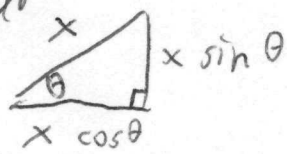


We don't have to project onto one of the main axes or planes - we could also project onto a different line or plane.



A line is the set of vectors $\{c \cdot \vec{5} \mid c \in \mathbb{R}\}$ or any scaling of a direction-setting vector $\vec{5}$. If we define the line this way, then any vector on the line, including our projection, should just be a scaling of $\vec{5}$. So we just need to find the constant c for it.

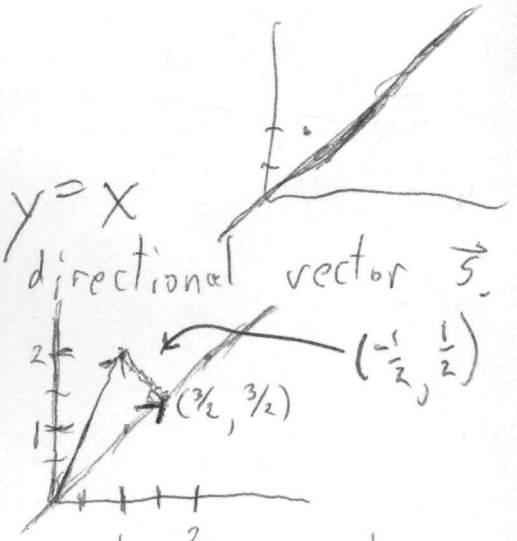
~~The constant c is the~~

Recall some HS trig: 

So $|\vec{s}| = |\vec{v}|\cos\theta$. Recall the dot product $\vec{s}\cdot\vec{v}$ gives $|\vec{s}||\vec{v}|\cos\theta$, so $|\vec{s}| = \frac{|\vec{v}||\vec{s}|\cos\theta}{|\vec{s}|} = \frac{\vec{v}\cdot\vec{s}}{|s|}$.

Then $c = \frac{\vec{v}\cdot\vec{s}}{|s||s|} = \frac{\vec{v}\cdot\vec{s}}{\vec{s}\cdot\vec{s}}$, and the vector itself is $\frac{\vec{v}\cdot\vec{s}}{\vec{s}\cdot\vec{s}}\vec{s}$.

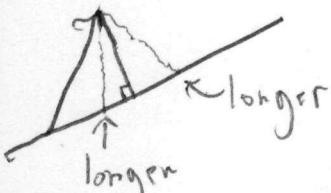
Example: Projection onto the line $y=x$ for $\begin{pmatrix}1\\2\end{pmatrix}$. $y=x$ has $\begin{pmatrix}1\\1\end{pmatrix}$ as a directional vector $\vec{s}$.



$1\cdot1+2\cdot1$ / $1\cdot1+1\cdot1$ $\quad \frac{\begin{pmatrix}1\\2\end{pmatrix}\cdot\begin{pmatrix}1\\1\end{pmatrix}}{\begin{pmatrix}1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\1\end{pmatrix}} = \frac{3}{2}$ so $\frac{3}{2}\begin{pmatrix}1\\1\end{pmatrix} = \begin{pmatrix}\frac{3}{2}\\\frac{3}{2}\end{pmatrix}$.

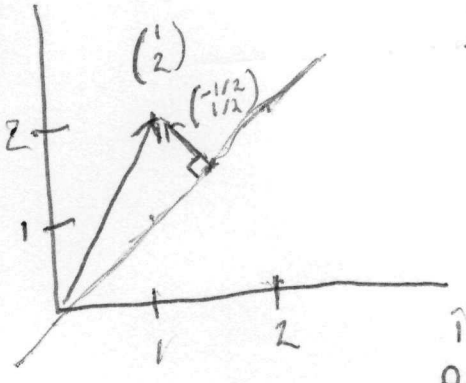
Projection can be used to find the components of a force or speed that are in an interesting direction. If a player of a game runs into a wall that is again along $y=x$ with a vector of $\begin{pmatrix}\Delta x\\\Delta y\end{pmatrix} = \begin{pmatrix}1\\2\end{pmatrix}$, this says the player should slide along the wall with speed $\begin{pmatrix}\frac{3}{2}\\\frac{3}{2}\end{pmatrix}$. That isn't as fast as the player was going before, but it will feel right because the player isn't trying to go in that direction specifically.

The projection is the vector on the line closest to our original vector — the difference would be longer if the angle weren't a right angle. (This property comes up when we approximate high-dimensional data with fewer dimensions — the projection is as faithful as we can be while keeping to the surface we've chosen.)



Gram-Schmidt Orthogonalization

Projections can also be used to get rid of components in directions that we want to ignore, leaving just the part of the vector orthogonal to that direction. In our previous example, we found the projection of $\begin{pmatrix}1\\2\end{pmatrix}$ onto $y=x$ is $\begin{pmatrix}\frac{3}{2}\\\frac{3}{2}\end{pmatrix}$. What is the value of what's left over if we get rid of this component? It's $\begin{pmatrix}1\\2\end{pmatrix} - \begin{pmatrix}\frac{3}{2}\\\frac{3}{2}\end{pmatrix} = \begin{pmatrix}-\frac{1}{2}\\\frac{1}{2}\end{pmatrix}$. That should look like our orthogonal vector.



We had two slightly different vectors, $\vec{v}$ and $\vec{s}$, and found two vectors orthogonal to each other that span the same space. This could be handy. Until now, our basis vectors needed to be linearly independent, but they didn't need to be orthogonal. But, coordinate systems where the axes aren't at right angles are weird.

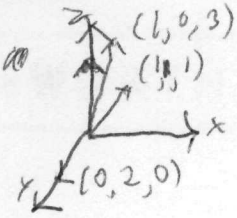
Projection gives us a way to turn any basis into an orthogonal basis. We could even make the basis <u>orthonormal</u> if we like, with each basis vector of length 1. The process, given basis vectors $\vec{\beta}_1, \dots, \vec{\beta}_n$:

- Leave $\vec{\beta}_1$ alone: $\vec{\kappa}_1 = \vec{\beta}_1$
- Let $\vec{\kappa}_2 = \vec{\beta}_2 - \text{proj}(\vec{\kappa}_1)(\vec{\beta}_2)$
  (that's what's left after removing the projection of $\vec{\beta}_2$ onto $\vec{\beta}_1$)
- Similarly, $\vec{\kappa}_3 = \vec{\beta}_3 - \text{proj}(\kappa_1)(\vec{\beta}_3) - \text{proj}(\kappa_2)(\vec{\beta}_3)$
  (remove components of $\vec{\beta}_3$ in the first two directions)

And so on, removing the components in all previous directions for each component.

Example (from book):

Basis $\left\langle \begin{pmatrix}1\\1\\1\end{pmatrix}, \begin{pmatrix}0\\2\\0\end{pmatrix}, \begin{pmatrix}1\\0\\3\end{pmatrix} \right\rangle$



Linearly independent. Not a great coordinate system.

Let $\vec{\kappa}_1 = \begin{pmatrix}1\\1\\1\end{pmatrix}$, unchanged. We'll suppose we like this direction for some reason.

$$\text{proj}(\vec{\kappa}_1)(\vec{\beta}_2) = \frac{\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}0\\2\\0\end{pmatrix}}{\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\1\\1\end{pmatrix}}\begin{pmatrix}1\\1\\1\end{pmatrix} = \frac{2}{3}\begin{pmatrix}1\\1\\1\end{pmatrix} = \begin{pmatrix}2/3\\2/3\\2/3\end{pmatrix}$$

$$\vec{\kappa}_2 = \vec{\beta}_2 - \text{proj}(\vec{\kappa}_1)(\vec{\beta}_2) = \begin{pmatrix}0\\2\\0\end{pmatrix} - \begin{pmatrix}2/3\\2/3\\2/3\end{pmatrix} = \begin{pmatrix}-2/3\\4/3\\-2/3\end{pmatrix}$$

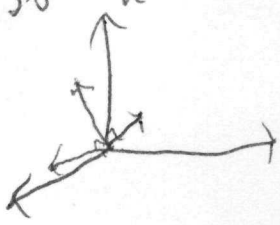
Notice this is orthogonal: $\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}-2/3\\4/3\\-2/3\end{pmatrix} = 0$ and $\cos\theta = 0$ implies orthogonal

$$\text{proj}(\vec{\kappa}_1)\vec{\beta}_3 = \frac{\begin{pmatrix}1\\0\\3\end{pmatrix}\cdot\begin{pmatrix}1\\1\\1\end{pmatrix}}{\begin{pmatrix}1\\1\\1\end{pmatrix}\cdot\begin{pmatrix}1\\1\\1\end{pmatrix}} = \frac{4}{3}\begin{pmatrix}1\\1\\1\end{pmatrix} = \begin{pmatrix}4/3\\4/3\\4/3\end{pmatrix} \qquad \begin{pmatrix}1\\0\\3\end{pmatrix} - \begin{pmatrix}4/3\\4/3\\4/3\end{pmatrix} = \begin{pmatrix}-1/3\\-4/3\\5/3\end{pmatrix}$$ not done yet...

$$\text{proj}(\vec{\kappa}_2)\vec{\beta}_3 = \frac{\begin{pmatrix}1\\0\\3\end{pmatrix}\cdot\begin{pmatrix}-2/3\\4/3\\-2/3\end{pmatrix}}{\vec{\kappa}_2\cdot\vec{\kappa}_2}\vec{\kappa}_2 = \frac{-8}{24/9}\begin{pmatrix}-2/3\\4/3\\-2/3\end{pmatrix} = \begin{pmatrix}2/3\\-4/3\\2/3\end{pmatrix} \qquad \begin{pmatrix}-1/3\\-4/3\\5/3\end{pmatrix} - \begin{pmatrix}2/3\\-4/3\\2/3\end{pmatrix} = \begin{pmatrix}-1\\0\\1\end{pmatrix}$$

So now we have the basis $\left\langle \begin{pmatrix} \frac{2}{3} \\ \frac{2}{3} \\ \frac{1}{3} \end{pmatrix}, \begin{pmatrix} -2/3 \\ 4/3 \\ -2/3 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \right\rangle$

and we can check that any two of these are orthogonal: $\begin{pmatrix} 2/3 \\ 2/3 \\ 2/3 \end{pmatrix} \cdot \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} = -2/3 + 2/3 = 0$. They aren't unit length, but we could fix that by finding their lengths and dividing. For example, $\left|\begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}\right| = \sqrt{1^2 + 1^2} = \sqrt{2}$, so a unit vector would be $\begin{pmatrix} \frac{-1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix}$: $\sqrt{\left(\frac{-1}{\sqrt{2}}\right)^2 + \left(\frac{1}{\sqrt{2}}\right)^2} = \frac{1}{2} + \frac{1}{2} = 1$



So we can have an orthonormal basis with one axis equal to whatever we want, and similar projections could express any vector as a scaled sum of the new basis vectors.

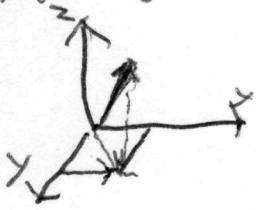
The process of finding these new vectors is called Gram-Schmidt Orthogonalization. Knowing the process may not be as important as knowing how to break a vector into components that are aligned and orthogonal to a direction.

Projection in More Dimensions

What about projecting onto a plane? A strategy that works here is the following:

- Project onto one basis vector.
- Project onto an orthogonal basis vector.
- Sum the results.

It's clear that this works if for example the basis vectors are $\begin{pmatrix}1\\0\\0\end{pmatrix}$ and $\begin{pmatrix}0\\1\\0\end{pmatrix}$. A shadow cast by a vector on the xy plane is equal to the sum of its x projection and y projection.



But this is also true of any orthogonal basis, which is just the same idea rotated in space.

In fact, this idea also works with more dimensions, which can come in handy if we want to simplify some data. To project from a higher-dimensional space to a lower-dimensional space, project onto each orthogonal basis vector, then sum. Later we'll see how to choose a good lower-dimensional subspace that keeps interesting trends while discarding noise.

p. 282 There's actually a convenient formula for projecting onto a subspace, which encapsulates all the operations suggested above (get an orthogonal basis, project onto each basis vector, sum). It's a bit cumbersome and unintuitive at first.

Let A be a matrix where the columns are basis vectors of a subspace of $\mathbb{R}^n$. To project $\vec{v}$ onto the subspace represented by this basis, we can do this:

$$\text{proj}_W(\vec{v}) = A(A^TA)^{-1}A^T\vec{v}$$

Proof The projection in the subspace must be a linear combination of its basis vectors.

$$\text{proj}_M(\vec{v}) = c_1\vec{\beta}_1 + c_2\vec{\beta}_2 + \cdots + c_k\vec{\beta}_k$$ (k is the dimension of the subspace)

Recall that A's structure is $\begin{bmatrix} | & | & & | \\ \vec{\beta}_1 & \vec{\beta}_2 & \cdots & \vec{\beta}_k \\ | & | & & | \end{bmatrix}$

and so $\text{proj}_M(\vec{v}) = A\vec{c}$ for some vector $\vec{c} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_k \end{pmatrix}$

$$\begin{bmatrix} | & | & & | \\ \vec{\beta}_1 & \vec{\beta}_2 & \cdots & \vec{\beta}_k \\ | & | & & | \end{bmatrix}\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_k \end{bmatrix} = \begin{bmatrix} c_1\beta_{11} + c_2\beta_{12} + \cdots + c_k\beta_{1k} \\ c_1\beta_{21} + c_2\beta_{22} + \cdots + c_k\beta_{2k} \\ \vdots \end{bmatrix} = c_1\begin{bmatrix} \beta_{11} \\ \beta_{21} \\ \vdots \\ \beta_{n1} \end{bmatrix} + c_2\begin{bmatrix} \beta_{12} \\ \beta_{22} \\ \vdots \end{bmatrix} + \cdots$$

Now we need to solve for $\vec{c}$, solving this will be where the inverse comes in. Let $\vec{v} - A\vec{c}$ be the part of $\vec{v}$ orthogonal to its projection the "lost" info.

If it's orthogonal, its dot product with every basis vector β_i is zero. So

$$\underbrace{\begin{bmatrix} — & \vec{\beta}_1 & — \\ — & \vec{\beta}_2 & — \\ & \vdots & \\ — & \vec{\beta}_k & — \end{bmatrix}}_{A^T}(\vec{v} - A\vec{c}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \leftarrow \vec{\beta}_1 \cdot (\vec{v} - A\vec{c}) \\ \leftarrow \vec{\beta}_2 \cdot (\vec{v} - A\vec{c}) \end{matrix}$$

We can use that fact, $A^T(\vec{v} - A\vec{c}) = \vec{0}$, to solve for $\vec{c}$.

$$A^T\vec{v} - A^TA\vec{c} = \vec{0}$$

$$A^T\vec{v} = A^TA\vec{c}$$

$$(A^TA)^{-1}A^T\vec{v} = \vec{c}$$

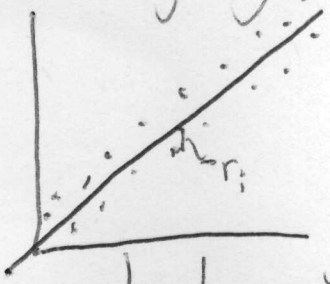
And that's $\vec{c}$ making the final projection

$$\text{proj}_M(\vec{v}) = A\vec{c} = A(A^TA)^{-1}A^T\vec{v}.$$

We'll see this idea applied with the...

Line of Best Fit

This is a very widely used method for fitting a line to data. (In more advanced forms, it can fit other models, too.) It's also known as a "least-squares fit" because if we treat the difference between our line's prediction and the actual data as "error," this method minimizes the sum of squared errors across all the points. (If r_i = predict$_i$ − actual$_i$, the method minimizes $\sum_i r_i^2$.) But we're not going to use calculus or anything — we're just going to project the data points onto a line.



Let's suppose first that our line goes through the origin. Our example will follow the book — trying to find a probability m of flipping heads, given the following experiments:

30 flips : 16 heads
60 flips: 34 heads
90 flips: 51 heads

If everything fit expectation perfectly, we'd have

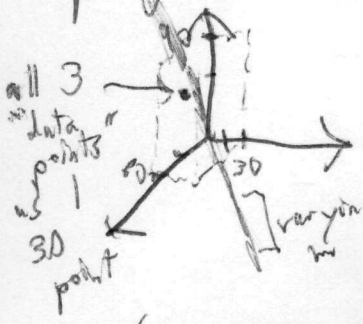
$$30m = 16$$
$$60m = 34$$
$$90m = 51$$

or $m\begin{pmatrix}30\\60\\90\end{pmatrix} = \begin{matrix}16\\34\\51\end{matrix}$

Since the data is realistic, though, there's no zero-error solution here; $\frac{16}{30} \neq \frac{34}{60}$. What we do have is a subspace of vectors $m\begin{pmatrix}30\\60\\90\end{pmatrix}$. The one that produces the least error

(Thus, the value of m that produces the least error) corresponds to the projection of the data onto this subspace.

Notice that this isn't the same as projecting onto our final line! What's being projected onto is the space of all possible fits. The variable that varies in our 1D projection is the slope.



So how do we project here? Recall the equation $\text{proj}_{\vec{s}}\vec{v} = \frac{\vec{v}\cdot\vec{s}}{\vec{s}\cdot\vec{s}}\vec{s}$. Here, $\vec{s}$ is the vector of inputs, $\begin{pmatrix}30\\60\\90\end{pmatrix}$, and $\vec{v}$ is the data, $\begin{pmatrix}16\\34\\51\end{pmatrix}$. Plugging in, we get:

$$\frac{\begin{pmatrix}16\\34\\51\end{pmatrix}\begin{pmatrix}30\\60\\90\end{pmatrix}}{\begin{pmatrix}30\\60\\90\end{pmatrix}\begin{pmatrix}30\\60\\90\end{pmatrix}} \cdot \begin{pmatrix}30\\60\\90\end{pmatrix} = \frac{7110}{12600}\begin{pmatrix}30\\60\\90\end{pmatrix}$$

Though we could multiply this out, that would give us the location on the subspace of the projection — which isn't exactly what we're interested in. Recall we wanted to solve for m in $m\begin{pmatrix}30\\60\\90\end{pmatrix} \approx \begin{pmatrix}13\\34\\51\end{pmatrix}$, and the product on the right (above) contains both terms. If we want just the line slope, that's $\frac{7110}{12600} \approx 0.56$. Thus the projection as a whole would give us the point on the line of the subspace, but if we already had a vector in the right direction and just wanted the final parameter, we don't need to do the final multiplication of the projection — we just care about the parameter that isn't the direction vector. (Whether this coin really is fair or not, we leave for statistics...)