Intro to Computational Math, Spring 2026
Section 6, March 3 (not due or graded).

Section will give you time to work on/discuss questions in small groups and then present your solutions. The information below is more of a summary of useful topics. There are no problems this week, but the following may be useful to better understand least squares, normal equations, and Householder matrices.

TL;DR: This section is meant to give a constructive way to solve fitting a function that hopefully requires very little memorization. Solving the problem of fitting data often resolves to minimizing the difference between the output data and some matrix-vector multiplication based on the input data. That is, we solve

$$\min_v \|Av - b\|_2^2$$

where the vector v here parameterized some type of function (i.e. a polynomial). The solution to this minimization problem must satisfies

$$A^T(Av - b) = 0$$

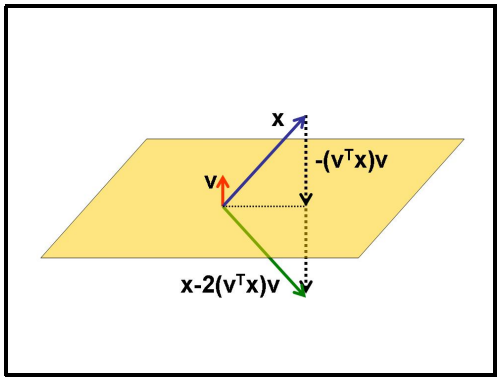
by optimality conditions, which gives rise to solving the linear problem (a.k.a the "Normal Equations")

$$A^T Av = A^T b.$$

This technique is called linear regression and surprisingly has very little to do with fitting a linear function. We can solve fitting a polynomial, a power law, and exponential, and many more via these techniques. Finally, by decomposing $A = QR$ into an orthogonal matrix and an upper triangular one, this problem can be converted into one that is solved with simple backwards substitution. Computing $A = QR$ can rely on applications relating to Householder Matrices with are seen as reflections over a plane. That is, for a plane defined as the orthogonal space to a vector v, the Householder reflection is

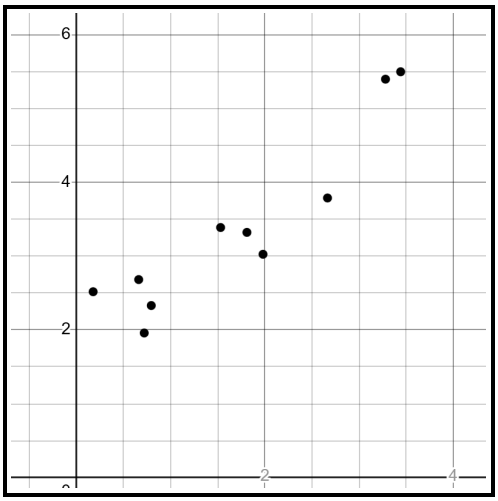
$$P = I - 2v^T v, \quad Px = (I - 2vv^T)x = x - 2v(v^T x) = x - 2(v^T x)v$$

.

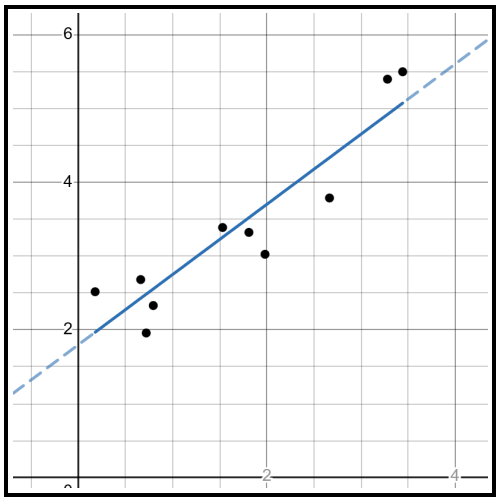


(a) Reflecting a vector over a plane

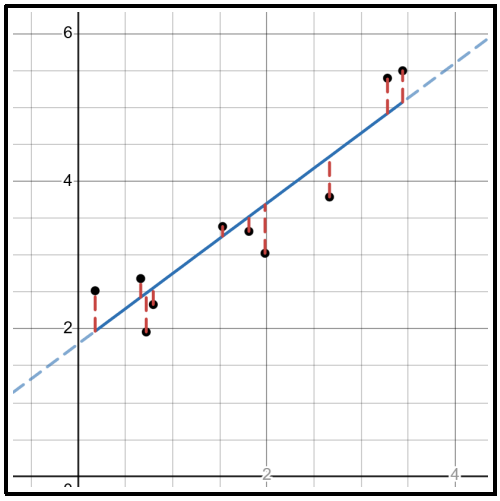
Least squares and linear regression Very often in data analysis, signal processing, and machine learning, we wish to approximate data with a structures function (i.e. a low degree polynomial). Instead of interpolating to have the function go through the points exactly (which can be unstable), we want to "fit" the data as best as possible. We measure this fit by considering the (squared) distances from the data to the function.



(a) Data we wish to fit



(b) Example of a fitting line



(c) Plotted distances to sample function

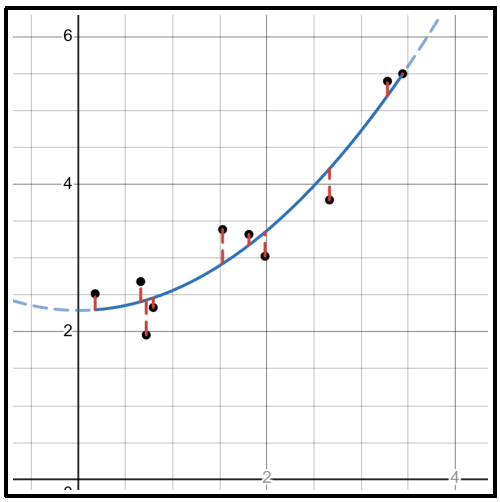
Formulating this exactly, if we have (possibly vector-valued) data (x_i, f_i) for $i = 1, ..., n$ then for a function $f(x)$, we can describe the sum of squares to be

$$\sum_{i=1}^{m}(f_i - f(x_i))^2.$$

Note that we could have considered just $|f_i - f(x_i)|$ instead, but the lack of differentiability will prove to be a difficulty for us. Further note that really we are trying to solve a problem here. We really want to solve

$$\min_{f\in\mathcal{F}} \sum_{i=1}^{m}(f_i - f(x_i))^2, \quad \mathcal{F} \text{ is some structured function class.}$$

In the case where we want to find the line of best fit, $\mathcal{F} = \{f(x) = a_1 x + a_0\}$. In the case where we find a best fitting cubic, $\mathcal{F} = \{f(x) = a_3 x^3 + x_2 x^2 + x_1 x + x_0\}$. In the case where we are finding the best fitting power law, $\mathcal{F} = \{f(x) = mx^b\}$. Since we can often parameterize these function classes with a finite number of parameters (i.e, $a_0, a_1, ..., a_k$) we can turn this into a tractable optimization problem!



(a) Plotted distances to a cubic

Example

In the case of fitting a cubic, the problem now becomes

$$\min_{a_0,a_1,a_2,a_3} \sum_{i=1}^{m} (f_i - (a_3 x_i^3 + a_2 x_i^2 + a_1 x_i + a_0))^2,$$

Recall the data (x_i, f_i) is fixed beforehand. These are now just number we have access to. Then, because our variable we are optimizing are a_0, a_1, a_2, a_3, and they are just weighted linearly by some numbers (i.e. $x_i^3, x_i^2, ...$) we can write these values $(a_3 x_i^3 + a_2 x_i^2 + a_1 x_i + a_0)$ as the components of a matrix vector multiplication. That is, we can rewrite

$$\underbrace{\begin{bmatrix} x_1^3 & x_1^2 & x_1 & 1 \\ x_2^3 & x_2^2 & x_2 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_m^3 & x_m^2 & x_m & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix}}_{v} = \begin{bmatrix} a_3 x_1^3 + a_2 x_1^2 + a_1 x_1 + a_0 \\ a_3 x_2^3 + a_2 x_2^2 + a_1 x_2 + a_0 \\ \vdots \\ a_3 x_m^3 + a_2 x_m^2 + a_1 x_m + a_0 \end{bmatrix}$$

Now if we let the vector $b = [f_1, ..., f_m]^T$ our problem looks like

$$\min_v \sum_{i=1}^{m} (f_i - [Av]_i)^2 = \min_v \|Av - b\|_2^2.$$

Note that this representation had nothing to do with the function we are fitting being a linear function, nor is it unique to cubics. All that matters is *the optimizing variables are weighted linearly.* Recall, the optimizing variables are the parameters of the function we are trying to fit, not anything

to do with the data itself. This is why polynomial fits can be solved with linear regression. Power law fits can also be solved with linear regression via a change of variables (log-log). Exponential fits can be solved with linear regression too.

Normal Equations Now that we have set up the problem, we can solve it. Given data and a function class to which we are trying to find a best fit, we set up our suitable matrix and vector. In general, if a function is parametrized by n variables and we have m data points, then A will be an m by n matrix and $b \in \mathbb{R}^m$. Furthermore, the A_{ij} component of the matrix (i.e. row i, column j) will consist of i^{th} data that weights the j^{th} optimizing variable. Generality can often be confusing, so relate this to the example of fitting a cubic above. For example, $A_{21} = x_2^3$ because $f(x_2) = a_3x_2^3 + a_2x_2^2 + a_1x_2 + a_0$, and the weight on a_3 (here this is the first variable because of how we typically write polynomials) is precisely x_2^3. Regardless, the important note is that **both the matrix A and the vector b rely solely on the fixed data**.

In order to solve this problem, we first outline perhaps the most fundamental result from optimization.

Theorem

Let $v_\star$ be the minimizer of a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$. Then $\nabla f(v_\star) = 0$.

Under certain conditions, it is sufficient to have a zero gradient. This problem we are considering fits into these conditions. Therefore, we can solve

$$v_\star = \arg\min_v \|Av - b\|_2^2 \iff \nabla f(v_\star) = 0, \quad f(x) = \|Ax - b\|_2^2$$

Remark

The problem of least squares is morally about finding an x such that Ax is "as close to" b as possible. There are many ways to measure this distance, as there are many norms

$$\|Ax - b\|_1, \quad \|Ax - b\|_2, \quad \|Ax - b\|_\infty.$$

We choose the 2-norm because of its differentiability when squared. This is also why we square it instead of just minimizing the norm itself, even though the resulting minimizer would be the same. Using techniques that don't need differentiability, we can instead solve problems like $\min_v \|Av - b\|_p$, which yields different solutions, with perhaps different but desirable properties. Choosing the $p = 1$ norm, for example, permits the minimizer to be "sparse" meaning that most components are pushed to being 0.

Solving our problem now just accounts to setting a gradient equal to 0. For those particularly comfortable with multi variable calculus and its associated chain rule, it is clear that

$$f(x) = \|Ax - b\|^2 \implies \nabla f(x) = 2A^T(Ax - b).$$

However, this is not particularly trivial. Personally, I find it easier to just "pattern-match" and not that if instead we wanted to find the *derivative* of a function $f : \mathbb{R} \to \mathbb{R}$ where a and b are just scalars then

$$f(x) = (ax - b)^2 \implies f'(x) = 2a(ax - b).$$

The jump then to A being an $m \times n$ matrix, $b \in \mathbb{R}^m$ and considering how to make compatible matrix-vector multiplication isn't too far to derive the more general result. Regardless, setting $\nabla f(v) = 0$ exactly results in the normal equations.

Householder Matrices If we can decompose $A = QR$ where Q is an orthogonal matrix and R is upper triangular then

$$A^TAx = A^Tb \implies R^TQ^TQRx = R^TQ^Tb \implies Rx = Q^Tb$$

when R is invertible (typical for problems with enough data). Now solving the problem is extremely quick because we can just use backward substitution (solve for x_m then x_{m-1} ...) which only requires simple subtraction and division. The task of computing a QR can be difficult, but one tool that is used are these Householder matrices. Note that given a vector $v \in \mathbb{R}^n$, there is an $(n-1)$ dimensional subspace *orthogonal* to this vector. We outline some important results from linear algebra.

Lemma

Given $v \in \mathbb{R}^n$, there exist $n-1$ pair-wise orthogonal vectors $w_1, w_2, ..., w_{n-1}$ such that $w_i^Tv = 0$.

Definition

*Given a vector $v \in \mathbb{R}^n$, denote the $n-1$ dimensional subspace orthogonal to v as the **hyperplane** generated by v*

$$H(v) = \{w : w^Tv = 0\} = \text{span}(w_1, w_2, ..., w_{n-1})$$

We can also go in the reverse direction. Given an $n-1$ dimensional subspace, a hyperplane in $\mathbb{R}^n$, there exists a vector v such that v is orthogonal to every vector in the hyperplane.

We can also (orthogonally) project onto a hyperplane and decompose any vector $x \in \mathbb{R}^n$ into the sum of two vectors: one in the direction of v and another that is orthogonal to v. This first construction is rather length, but we will show very soon that there is an easier approach.

Lemma – Projection on a hyperplane

Let $v \in \mathbb{R}^n$ and let $w_1, ..., w_{n-1}$ be the $n-1$ orthogonal vectors to v (and each other). Then we define the projection onto the hyperplane as

$$\text{proj}_H(x) = \sum_{i=1}^{n-1} \frac{x^Tw_i}{w_i^Tw_i} w_i \in H(v)$$

Lemma – Orthogonal Decomposition

For any $x \in \mathbb{R}^n$ and given $v \in \mathbb{R}^n$ with associated hyperplane $H(v)$. One may decompose

$$x = \tilde{v} + w, \quad \tilde{v} \in \text{span}(v), \ w \in H(v)$$

Specifically, one may write this as

$$x = \tilde{v} + w, \quad \tilde{v} = \text{proj}_v(x) = \frac{x^Tv}{v^Tv} v, \ w = \text{proj}_H(x)$$

Instead of finding the $n-1$ orthogonal vectors to v and projecting onto each of those and summing them, one may simply project onto the hyperplane with the orthogonal vector itself.

Corollary

Given a hyperplane H and a unit vector v orthogonal to H, then

$$\text{proj}_H(x) = x - \text{proj}_v(x) = x - (v^T x)v$$

Reflecting over a hyperplane is then continuing this movement. That is, if *projecting* to the hyperplane takes the vector x and moves it orthogonally until it is on the plane, *reflecting* over the hyperplane is repeating this movement twice.

Definition

We define a ***Householder Transformation*** *to be the reflection of a vector x over a hyperplane generated by unit vector v:*

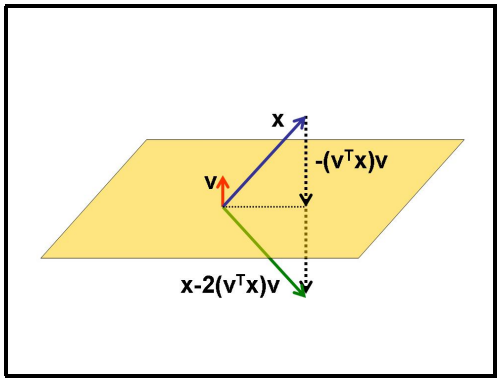
$$\tilde{x} = x - 2(v^T x)v$$

We call the ***Householder Matrix*** *of a vector,*

$$P_v = I - 2vv^T \iff P_v x = x - 2(v^T x)v$$

We note that this matrix then inherits many nice properties being defined as a reflection. For example, reflecting something twice sends the object back to where it started. Therefore, for any unit vector v and P_v as defined above

$$P_v^2 = I.$$



(a) Reflecting a vector over a plane

This can be shown algebraically as well

$$\begin{aligned} P_v^2 &= (I - 2vv^T)^2 \\ &= I^2 - 2vv^T - 2vv^T + 4vv^T vv^T \\ &= I - 4vv^T + 4vv^T \\ &= I \end{aligned}$$

Another property that comes immediately from the geometric understanding is that

$$P_v v = -v, \quad P_v w = w, \quad \forall w \in H(v)$$

That is, v is an eigenvector with eigenvalue -1, and any w orthogonal to v is an eigenvector with eigenvalue 1.