Fitting with Lagrange Polynomials

Resources:
Wiki on Lagrange polynomial fitting: https://en.wikipedia.org/wiki/Lagrange_polynomial
A lecture on piecewise Lagrange fitting: https://www.math.usm.edu/lambers/mat460/fall09/lecture20.pdf
The assignment site: https://github.com/pointOfive/sta410hw2/blob/main/sta410hw2.ipynb

1 Basic Lagrange Polynomial Fitting

Suppose we have a set of data points (n+1 data points in total)

$$(x_0, y_0),\ (x_1, y_1),\ ...,\ (x_n, y_n)$$

and we want to fit a polynomial through them (polynomials can fit any function that are "good" enough, for example Taylor series is a polynomial).

One way to do the fit is to use the so-called **Lagrange polynomial fitting**. Here's how it works:

Let $h(w)$ denote the fitted function. We want the fitted function to evaluate exactly at the known data points, i.e. we want $h(x_i) = y_i$ for all known data points (x_i, y_i).

At this point we introduce **the Kronecker delta $\boldsymbol{\delta_{ij}}$**. The Kronecker delta is simply $\delta_{ij} = 1$ if $i = j$ and 0 if $i \neq j$. It can be used to "kill" summations:

$$\sum_i y_i \delta_{ij} = y_j$$

In this sum over i (j in the above equation is a fixed index), every term carries a δ_{ij}, but all of these deltas are zero for every term with $i \neq j$, which zeros their corresponding terms. Therefore, the only survive term in the summation is the term with $i = j$. Plugging j into every place where i shows up, we see that the summation reduces to $y_j \delta_{jj}$ (no sum, since just one term survives). But $\delta_{jj} = 1$, so the summation is just y_j.

Looking at the above equation and procedure, we see that a δ_{ij} will kill the $\sum_i$, and turn the other i indices into j. Since δ_{ij} is symmetric in i and j, it can also be used to kill a $\sum_j$ and turn all j into i. Therefore, for our set of $n + 1$ data points we can have

$$\sum_{j=0}^{n} y_j \delta_{ij} = y_i$$

If we *design* our fitted function $h(w)$ to be a linear combination of some to-be-determined basis functions $l_j(w)$ as (using j as the dummy index for the linear combination)

$$h(w) = \sum_{j=0}^{n} y_j l_j(w)$$

where we wish the requirement

$$l_j(x_i) = \delta_{ij}$$

to hold, then we would have

$$h(x_i) = \sum_{j=0}^{n} y_j l_j(x_i) = \sum_{j=0}^{n} y_j \delta_{ij} = y_i$$

satisfying our requirement of exact evaluation at the known data points. The problem now reduces to trying to find the basis functions $l_j(w)$. The only requirement on these basis functions is that $l_j(x_i) = \delta_{ij}$.

This is really a two-part requirement. For the basis function $l_j(w)$ [1], it needs to (1) evaluate to 1 at x_j, and (2) evaluate to 0 at the x values of all the other data points x_i, $i \neq j$.

Luckily, it is not very hard to have such a construct. The answer is that for a data point (x_j, y_j), its corresponding basis function is

$$\begin{aligned} l_j(w) &= \prod_{\substack{0 \leqslant i \leqslant n \\ i \neq j}} \frac{w - x_i}{x_j - x_i} \\ &= \frac{w - x_0}{x_j - x_0} \frac{w - x_1}{x_j - x_1} \cdots \frac{w - x_{j-1}}{x_j - x_{j-1}} \frac{w - x_{j+1}}{x_j - x_{j+1}} \cdots \frac{w - x_n}{x_j - x_n} \end{aligned}$$

There are a total of $n + 1$ indices from 0 to n, corresponding to the $n + 1$ data points, but one of them is excluded, so there's a total of n terms in this product, making it a n-th degree polynomial in w.

All the denominators are non-zero. Since all denominators are of the form $(x_j - x_i)$, and we explicitly excluded the $i = j$ term (reminder that j is fixed and i is the multiplication index), none of the denominators risk being zero. Of course, since it's a set of sampled data, we won't have two x values on different indices having the same value (we will only get one y if we sample at some x).

[1] or "the basis function associated with the j-th data point (x_j, y_j)"

Inspect the two requirements:

$$\begin{aligned} l_j(x_j) &= \prod_{\substack{0 \leqslant i \leqslant n \\ i \neq j}} \frac{x_j - x_i}{x_j - x_i} \\ &= 1 \quad \text{(none of the numerators are zero, since } i \neq j \text{ for all terms)} \end{aligned}$$

so the first requirement is good. For the second requirement, notice that when $i \neq j$, the product $l_j(x_i)$ contains a term $\dfrac{x_i - x_i}{x_j - x_i}$, which makes the entire product zero.

We have thus fitted the data points. Summary:

$$h(w) = \sum_{j=0}^{n} y_j l_j(w)$$

where

$$l_j(w) = \prod_{\substack{0 \leqslant i \leqslant n \\ i \neq j}} \frac{w - x_i}{x_j - x_i}$$

Note that each $l_j(w)$ is an order-n polynomial, so their linear combination, namely $h(w)$, is also an order-n polynomial. Also note that **the order of our fitted polynomial is one less than the number of given data points.**

Example 1 [edit]

We wish to interpolate $f(x) = x^2$ over the domain $1 \le x \le 3$, given these three points:

$$\begin{aligned} x_0 &= 1 & f(x_0) &= 1 \\ x_1 &= 2 & f(x_1) &= 4 \\ x_2 &= 3 & f(x_2) &= 9. \end{aligned}$$

The interpolating polynomial is:

$$\begin{aligned} L(x) &= 1 \cdot \frac{x-2}{1-2} \cdot \frac{x-3}{1-3} + 4 \cdot \frac{x-1}{2-1} \cdot \frac{x-3}{2-3} + 9 \cdot \frac{x-1}{3-1} \cdot \frac{x-2}{3-2} \\ &= x^2. \end{aligned}$$

Example 2 [edit]

We wish to interpolate $f(x) = x^3$ over the domain $1 \le x \le 4$, given these four points:

$$x_0 = 1 \quad f(x_0) = 1$$

$$x_1 = 2 \quad f(x_1) = 8$$

$$x_2 = 3 \quad f(x_2) = 27$$

$$x_3 = 4 \quad f(x_3) = 64$$

The interpolating polynomial is:

$$\begin{aligned} L(x) &= 1 \cdot \frac{x-2}{1-2} \cdot \frac{x-3}{1-3} \cdot \frac{x-4}{1-4} + 8 \cdot \frac{x-1}{2-1} \cdot \frac{x-3}{2-3} \cdot \frac{x-4}{2-4} + 27 \cdot \frac{x-1}{3-1} \cdot \frac{x-2}{3-2} \cdot \frac{x-4}{3-4} + 64 \cdot \frac{x-1}{4-1} \cdot \frac{x-2}{4-2} \cdot \frac{x-3}{4-3} \\ &= x^3 \end{aligned}$$

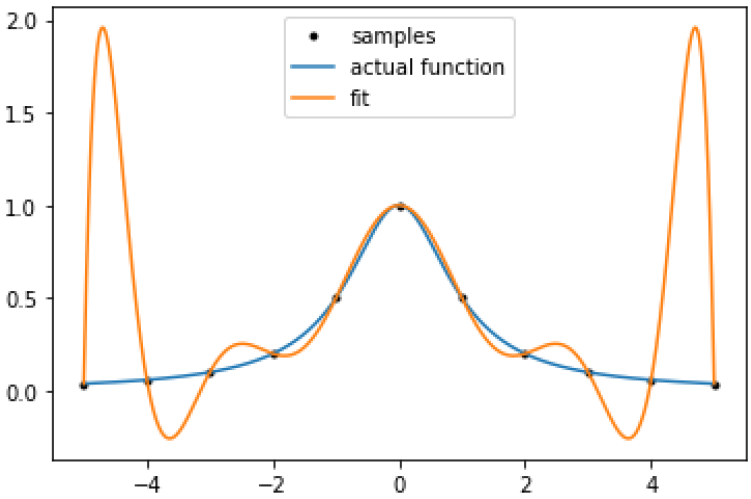
Figure 1: Two examples from wiki. Note that wiki uses $L(x)$ instead of $h(w)$ for the fitted function.

2 Piecewise Lagrange Polynomial Fitting

The problem with basic fitting is that, when we have a large number of data points (which is very likely in practice; in physics labs I've fitted 1000 data points into a curve), the degree of the fitted polynomial will be very high. A very high-degree polynomial will have very violent oscillations [2]. For example, let's say we take 11 sample points from the curve

$$y = \frac{1}{1 + x^2}$$

and do a basic Lagrange fit. Here are the results:



I think you would agree that this is a relatively poor fit. Even with just 11 data points, the negative impact of oscillations are already starting to appear.

The solution is to reduce the number of data points when fitting so that we can smooth out the oscillations. If we can get a lower-degree polynomial as the resultant fitted polynomial, we'll have weaker oscillations. Therefore, we partition the sample data points into smaller segments (or *pieces*), each containing just a few data points, and do a basic Lagrange fit on each piece. When the non-oscillating fit on each piece is done, we simply join (or *concatenate*) the pieces together.

[2] An oscillation in a curve occurs when, roughly speaking, its second derivative is zero, i.e. an inflection point. For a polynomial of degree n, its second derivative is a polynomial of degree $n - 2$. For larger n and hence larger $n - 2$, this means more zeros of the second derivative, hence more inflection points and more oscillations.

We partition the $n+1$ data points with the following index convention:

In the example we had 11 data points (from x_0 to x_{10}). That's $n = 10$.

We wish to partition these points into the pieces

$$[x_0, x_2),\ [x_2, x_4),\ [x_4, x_6),\ [x_6, x_8),\ [x_8, x_{10}),\ x_{10}$$

We let each piece contain the left point but not the right point for python convenience [3] Note that each joint is contained by the segment to the right of the joint. For example, x_2 is contained by $[x_2, x_4)$, not $[x_0, x_2)$. Notice that this means none of our "usual" pieces contains the last data point x_{10}, so it would need to be an extra standalone point.

Let k denote the number of data points on a segment. In this example we thus have $k = 2$ (the points contained by $[x_0, x_2)$ is just x_0 and x_1, and there's no x_2). Therefore, the first segment needs to start at 0 and end at k, meaning the second segment needs to start at k in order to join up with the end of the first segment, further meaning the second segment needs to end at $k + k = 2k$, meaning the third segment needs to start at $2k$, and so on and so forth. In the end we have the partition as

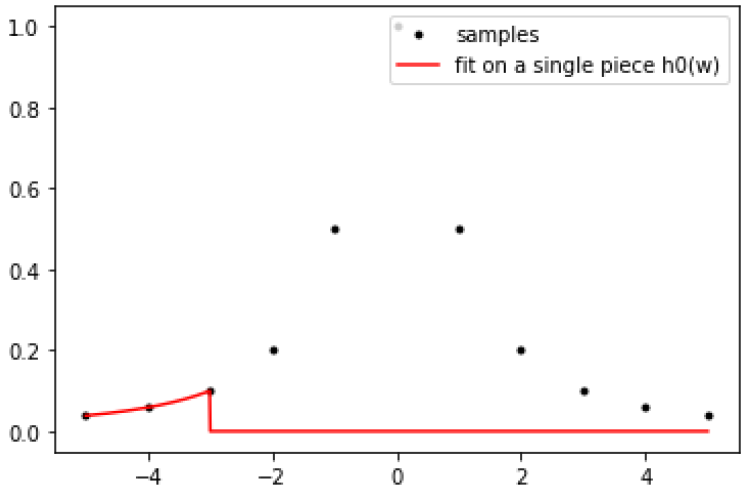
$$[x_0, x_k),\ [x_k, x_{2k}),\ [x_{2k}, x_{3k}),\ [x_{3k}, x_{4k}),\ [x_{4k}, x_{5k}),\ x_{5k}$$

Or, in short, we have that the g-th segment is $[x_{gk}, x_{gk+k})$, where we start the counting of g from 0.

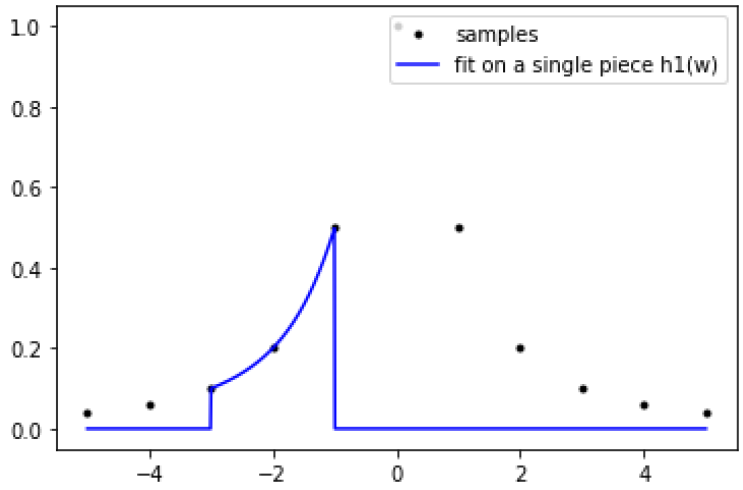
[3] "for i in range(0,3)" is 0,1,2. 3 is excluded.

Let's now focus on a single segment. For the ease of discussion let's say we focus on the g-th segment $[x_{gk}, x_{gk+k})$. We want the fit to fit this segment, and be zero everywhere else. In this fashion, when we add the fitted pieces together, we can let each fit only control its own piece.

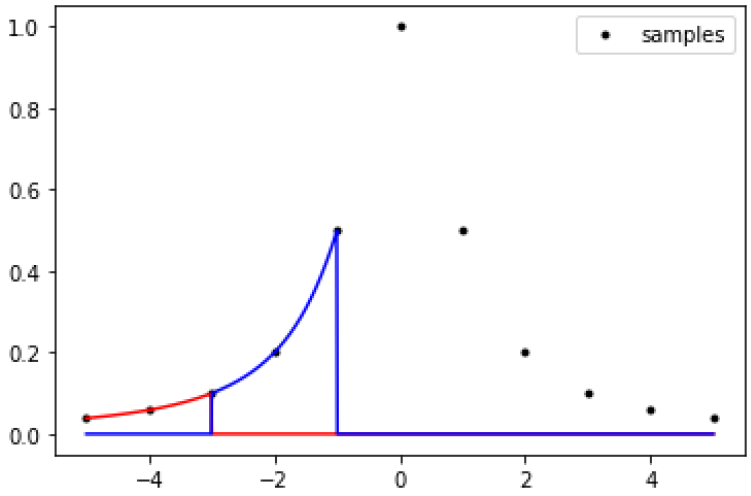
Here's an example. Here's the fit $h_0(w)$ on the first piece:



Here's the fit $h_1(w)$ on the second piece:



If you simply add them together, you will get the overall fit on the first two pieces as a whole:



In the first segment, the red local fit takes care of the fit itself, and when adding the two fits together, the blue fit is simply adding zero in the first segment, so the shape of the red fit is remained in the first segment.

Disclaimer: these red and blue curves are for illustration purposes of the piecewise-fit only. They are not the actual fitted results. A real fit will likely to differ somewhat from the actual function.

So our task is easy: modify the basic Lagrange fitting so that it only does the fit within the g-th segment $[x_{gk}, x_{gk+k})$, and gives zero outside. Let's call the local fit on the g-th segment by $h_g(w)$.

First, let's look at the basic fit over the data points $[x_0, x_n)$ given by

$$h(w) = \sum_{j=0}^{n} y_j l_j(w)$$

where

$$l_j(w) = \prod_{\substack{0 \leqslant i \leqslant n \\ i \neq j}} \frac{w - x_i}{x_j - x_i}$$

If we are now looking at just the data points $[x_{gk}, x_{gk+k})$, the fit is modified as

$$h_g(w) = \sum_{j=gk}^{gk+k} y_j l_{gj}(w)$$

where

$$l_{gj}(w) = \prod_{\substack{gk \leqslant i \leqslant gk+k \\ i \neq gk+j}} \frac{w - x_i}{x_{gk+j} - x_i}$$

Some explanations might be necessary:

1. Instead of constructing $h(w)$ as a linear combination of every data point, we simple use the data points inside $[x_{gk}, x_{gk+k})$. This changes the linear combination to over $\sum_{j=gk}^{gk+k}$ (reminder: j is simply the dummy index for the linear combination).

2. In changing from $h(w)$ to $h_g(w)$, we let the basis functions carry the subscript g to indicate that we're on the g-th segment, i.e. we change $l_j(w)$ to $l_{gj}(w)$. The other choice would be to let y_j carry the extra index, but y is the dependent variable of the data sample and is a one-dimensional array, and two indices under y makes no sense.

3. In the basis functions, again, instead of taking a product of every data point, we simply use the data points inside $[x_{gk}, x_{gk+k})$. The point being excluded for $l_j(w)$ was x_j, because $l_j(w)$ was the basis function for x_j, the j-th point in the linear combination. On the segment, the j-th point in the linear combination is x_{gk+j}, so we exclude that point.

4. When doing the local fits themselves, we include the right end point x_{gk+k} as a data point, i.e. a total of $k + 1 (= (gk + k) - (gk) + 1)$ data points per segment. Otherwise, we would have no control over the rightmost subsegment (x_{gk+k-1}, x_{gk+k}) over this piece. We set the

value of the local fit at x_{gk+k} to zero after the fit, in order to avoid double counting (with the left end point of the next segment). Recalling that the order of our fitted polynomial is one less than the number of given data points, this means that the fitted result will be an order-k polynomial.

The above would do a basic Lagrange fit inside the g-th segment, but it doesn't guarantee anything for domains outside the segment. We thus attach a simple correction to the basis functions:

$$l_{gj}(w) = \begin{cases} \prod_{\substack{gk \leqslant i \leqslant gk+k \\ i \neq gk+j}} \dfrac{w - x_i}{x_{gk+j} - x_i} & \text{if } w \in [x_{gk}, x_{gk+k}) \\ 0 & \text{else} \end{cases}$$

In this fashion, our basis functions would do what it should do on $[x_{gk}, x_{gk+k})$, but be zero outside this segment. The entire local fit

$$h_g(w) = \sum_{j=gk}^{gk+k} y_j l_{gj}(w)$$

would, as a result, do what it should do on the segment, but be zero outside (since outside the segment, we have the linear combination of the basis functions as linear combinations of zero, which is still zero).

To obtain the overall fit, we can now simply add the local fits together, as discussed above:

$$\begin{aligned} h(w) &= \sum_g h_g(w) \\ &= \sum_g \sum_j y_j l_{gj}(w) \end{aligned}$$

There's one final caveat. Adding all the local fits together accounts for every point in the domain except the last point x_n, which doesn't belong to any of the subsegments. Right now, the value of the fit at x_n is zero, since it's outside every segment. A correction at a single point is needed.

We thus need to add the above to something that is y_n at x_n, and zero everywhere else. The thing needed is $y_n \delta_{x_n}(w)$. When $w = x_n$, the delta is 1 and the term evaluates to y_n; for all other w, this term is zero and has no effect.

Done.

Summary: the overall fit is

$$\begin{aligned} h(w) &= \sum_g h_g(w) + y_n \delta_{x_n}(w) \\ &= \sum_g \sum_{j=gk}^{gk+k} y_j l_{gj}(w) + y_n \delta_{x_n}(w) \end{aligned}$$

where

$$l_{gj}(w) = \begin{cases} \prod_{\substack{gk \leqslant i \leqslant gk+k \\ i \neq gk+j}} \dfrac{w - x_i}{x_{gk+j} - x_i} & \text{if } w \in [x_{gk}, x_{gk+k}) \\ 0 & \text{else} \end{cases}$$