1 Nonlinear Least Squares

All the models we've seen so far are **linear** in the parameters we're trying to learn. That is, our prediction $\hat{y} = f(x;\theta)$ is some linear function of the parameters θ. For example, in OLS, $\theta = w$ and the residuals r_i are computed by $y_i - w^\top x_i$, which is linear in the components of w. In the case of least-squares polynomial regression, the predicted value is not a linear function of the input x, but it is still a linear function of the parameters.

However, we may have need for models which are nonlinear function of their parameters. We consider a motivating example first.

1.1 Noisy Distance Readings

Suppose we want to estimate the 2D position $\theta = (\theta_1, \theta_2)$ of some entity, for example a robot. The information we have to work with are noisy distance estimates $Y_i \in \mathbb{R}$ from m sensors whose positions $X_i \in \mathbb{R}^2$ are fixed and known. If we assume i.i.d. Gaussian noise as usual, our statistical model has the form

$$Y_i = \|X_i - \theta\| + N_i, \qquad N_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2), \qquad i = 1, \ldots, m$$

where

$$\|X_i - \theta\| = \sqrt{(X_{i1} - \theta_1)^2 + (X_{i2} - \theta_2)^2}$$

Here our prediction is

$$\hat{y} = f(x;\theta) = \|x - \theta\|$$

which is clearly not linear in θ.

1.2 Formulation from MLE

More generally, let us assume a model similar to the one above, but where f is now some arbitrary differentiable function and $\theta \in \mathbb{R}^d$:

$$Y_i = f(X_i;\theta) + N_i, \qquad N_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2), \qquad i = 1, \ldots, m$$

Note that this implies $Y_i \mid X_i \sim \mathcal{N}(f(X_i;\theta), \sigma^2)$.

The maximum likelihood estimator is given by

$$\begin{aligned}
\hat{\theta}_{\text{MLE}} &= \arg\max_{\theta} \ \log P(y_1, \ldots, y_m \mid x_1, \ldots, x_m; \theta, \sigma) \\
&= \arg\max_{\theta} \ \log \prod_{i=1}^{m} P(y_i \mid x_i; \theta, \sigma) \\
&= \arg\max_{\theta} \ \sum_{i=1}^{m} \log P(y_i \mid x_i; \theta, \sigma) \\
&= \arg\max_{\theta} \ \sum_{i=1}^{m} \log \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - f(x_i;\theta))^2}{2\sigma^2}\right) \\
&= \arg\max_{\theta} \ \sum_{i=1}^{m} \left[-\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(y_i - f(x_i;\theta))^2\right] \\
&= \arg\min_{\theta} \ \sum_{i=1}^{m} (y_i - f(x_i;\theta))^2
\end{aligned}$$

The last step holds because the first term in the sum is constant w.r.t. the optimization variable θ, and we flip from max to min because of the negative sign.

Observe that the objective function is a sum of squared residuals as we've seen before, even though the function f is nonlinear in general. For this reason the method is called **nonlinear least squares**.

Unfortunately, there is no closed-form solution for $\hat{\theta}_{\text{MLE}}$ in general. Later we will see an iterative method for computing it.

2 Solutions to Nonlinear Least Squares

Motivated by the MLE formulation above, we consider the following optimization problem:

$$\min_{\theta} \varepsilon_{LS}(\theta) = \min_{\theta} \sum_{i} (y_i - f(x_i;\theta))^2$$

One way to minimize a function is to use calculus. We know that the gradient of the objective function at any minimum must be zero, because if it isn't, we can take a sufficiently small step in the direction of the negative gradient that the objective function's value will be reduced.

Thus, the **first-order optimality condition** that needs to be satisfied is:

$$\nabla_{\theta} \varepsilon_{LS} = 2 \sum_{i} (y_i - f(x_i;\theta)) \nabla_{\theta} f(x_i;\theta) = 0$$

In compact matrix notation:

$$J(\theta)^{\top}(Y - F(\theta)) = 0$$

where

$$F(\theta) = \begin{bmatrix} f(x_1;\theta) \\ \vdots \\ f(x_n;\theta) \end{bmatrix}$$

$$J(\theta) = \begin{bmatrix} \nabla_\theta f(x_1;\theta)^\top \\ \vdots \\ \nabla_\theta f(x_n;\theta)^\top \end{bmatrix} = \nabla_\theta F, \text{ the } \textbf{Jacobian} \text{ of F}$$

Observe that when f is linear in θ (i.e. $f(x_i;\theta) = \theta^\top x_i$), the gradient $\nabla_\theta \varepsilon_{LS}$ will only have θ in one place because the term $\nabla_\theta f(x_i;\theta)$ will only depend on x_i:

$$\nabla_\theta \varepsilon_{LS} = 2\sum_i (y_i - \theta^\top x_i)\nabla_\theta(\theta^\top x_i) = 2\sum_i (y_i - \theta^\top x_i)x_i$$

and it is easy to derive a closed-form solution for θ in terms of the y_i's and x_i's:

$$\begin{aligned} 2X^\top(Y - X\theta) &= 0 \\ 2X^\top Y - 2X^\top X\theta &= 0 \\ X^\top Y &= X^\top X\theta \\ \theta &= (X^\top X)^{-1}X^\top Y \end{aligned}$$

It's just OLS!

If, however, f were not linear in θ, the term $\nabla_\theta f(x_i;\theta)$ would contain more θ terms (since differentiating once wouldn't be enough to make them go away), and it would not be possible to write out a closed-form solution for θ.

Remark: Without more assumptions on f, the NLS objective is not convex in general. This means that the first-order optimality condition is a *necessary* but not *sufficient* condition for a local minimum. That is, it is possible that the derivative is zero for some value of θ, but that value is not a local minimum. It could be a saddle point, or worse, a local maximum! Even if it is a minimum, it may not be the global minimum.

2.1 The Gauss-Newton algorithm

Since there is no closed-form solution to the nonlinear least squares optimization problem, we resort to an iterative algorithm, the **Gauss-Newton algorithm**[1], to tackle it. At each iteration, this method linearly approximates the function F about the current iterate and solves a least-squares problem involving the linearization in order to compute the next iterate.

Let's say that we have a "guess" for θ at iteration k, which we denote $\theta^{(k)}$. We can then approxi-

[1] For some reason this algorithm was called gradient descent in lecture, but it is not really gradient descent. However, like gradient descent, it is an iterative, first-order optimization algorithm. Another popular method for solving nonlinear least squares, the **Levenberg-Marquardt algorithm**, is a sort of interpolation between Gauss-Newton and gradient descent.

mate $F(\theta)$ to first order using a Taylor expansion about $\theta^{(k)}$:

$$\begin{aligned} F(\theta) \approx \tilde{F}(\theta) &:= F(\theta^{(k)}) + \nabla_\theta F(\theta^{(k)})(\theta - \theta^{(k)}) \\ &= F(\theta^{(k)}) + J(\theta^{(k)})\Delta\theta \end{aligned}$$

where $\Delta\theta := \theta - \theta^{(k)}$.

Now since $\tilde{F}$ is linear in $\Delta\theta$ (the Jacobian and F are just constants: functions evaluated at $\theta^{(k)}$), we can use the closed form solution for $\Delta\theta$ from the optimality condition to *update* our current guess $\theta^{(k)}$. Applying the first-order optimality condition from earlier to the objective $\tilde{F}$ yields the following equation:

$$0 = J_{\tilde{F}}(\theta)^\top (Y - \tilde{F}(\theta)) = J(\theta^{(k)})^\top \left(Y - \left(F(\theta^{(k)}) + J(\theta^{(k)})\Delta\theta \right) \right)$$

Note that we have used the fact that the Jacobian of the linearized function $\tilde{F}$, evaluated at any θ, is precisely $J(\theta^{(k)})$. This is because $\tilde{F}$ is affine where the linear map is $J(\theta^{(k)})$, so the best linear approximation is just that.

Writing $J = J(\theta^{(k)})$ for brevity, we have

$$\begin{aligned} J^\top Y &= J^\top (F(\theta^{(k)}) + J\Delta\theta) \\ J^\top (Y - F(\theta^{(k)})) &= J^\top J(\Delta\theta) \\ \Delta\theta &= (J^\top J)^{-1} J^\top (Y - F(\theta^{(k)})) \\ &= (J^\top J)^{-1} J^\top \Delta Y \end{aligned}$$

where $\Delta Y := Y - F(\theta^{(k)})$. By comparing this solution to OLS, we see that it is effectively solving

$$\Delta\theta = \arg\min_{\delta\theta} \|J\delta\theta - \Delta Y\|^2$$

Since $\delta F \approx J\delta\theta$ close to $\theta^{(k)}$, this is saying that we choose a change to the weights that corrects for the current error in the function values, but it bases this calculation on the linearization of F. Recalling that $\Delta\theta = \theta - \theta^{(k)}$, we can improve upon our current guess $\theta^{(k)}$ with the update

$$\begin{aligned} \theta^{(k+1)} &= \theta^{(k)} + \Delta\theta \\ &= \theta^{(k)} + (J^\top J)^{-1} J^\top \Delta Y \end{aligned}$$

Here's the entire process laid out in steps:

1. Initialize $\theta^{(0)}$ with some guess
2. Repeat until convergence:
   (a) Compute Jacobian with respect to the current iterate, $J = J(\theta^{(k)})$
   (b) Compute $\Delta Y = Y - F(\theta^{(k)})$
   (c) Update: $\theta^{(k+1)} = \theta^{(k)} + (J^\top J)^{-1} J^\top \Delta Y$

Note that the solution found will depend on the initial value $\theta^{(0)}$ in general.

The choice for measuring convergence is up to the practitioner. Some common choices include testing changes in the objective value:

$$\left| \frac{\mathcal{E}^{(k+1)} - \mathcal{E}^{(k)}}{\mathcal{E}^{(k)}} \right| \leq \text{threshold}$$

or in the iterates themselves:

$$\max_j \left| \frac{\Delta\theta_j}{\theta_j^{(k)}} \right| \leq \text{threshold}$$