1 Nonlinear Least Squares

Up to this point, we’ve restricted ourselves to linear regression models. That is, our prediction \( \hat{y} = \theta^T x \) is a linear function of the input \( x \). This holds even in the case of least-squares polynomial regression — while the predicted value is not a linear function of the raw input \( x \), it is still a linear function of the augmented polynomial feature input \( \phi(x) \).

Effectively, we have been able to form nonlinear models by manually augmenting features to the input. Now what if instead of using a linear function of the augmented input, we could use an arbitrary nonlinear function \( f(x; \theta) \) of the raw input \( x \)? This approach is often more expressive and robust, because it removes the burden of augmenting expressive features to the input. As a motivating example, consider the problem of estimating the 2D position \( \theta = (\theta_1, \theta_2) \) of a robot. We are given noisy distance estimates \( Y_i \in \mathbb{R} \) from \( n \) sensors whose positions \( x_i \in \mathbb{R}^2 \) are fixed and known. Since we are predicting distance, it is reasonable to use the model \( f(x; \theta) = \|x - \theta\|_2 \). This model is clearly more appropriate than restricting ourselves to a linear model with augmented features — in that case, what exactly would the augmented features be?

Note however that for most problems, we are not given the form or structure of the model. Consider the following example: we are trying to predict a user’s income based on their occupation, age, education, etc... It is not exactly clear what model we should use. Rather than specifying a specific family of nonlinear functions, we are instead interested in a universal function appropriator \( f(x; \theta) \) which can approximate any function \( f(x) \) with appropriate parameters \( \theta \). This will be the basis for neural networks, which we will study in detail later.

1.1 MLE Formulation

For the purposes of our discussion, let us assume that we are given a model \( f \), an arbitrary differentiable function parameterized by \( \theta \):

\[
Y_i = f(x_i; \theta) + Z_i, \quad Z_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2), \quad i = 1, \ldots, n
\]

which can equivalently be expressed as \( Y_i \mid x_i \sim \mathcal{N}(f(x_i; \theta), \sigma^2) \). We are interested in finding the parameters \( \theta_{\text{MLE}} \) that maximize the likelihood of the data:

\[
\hat{\theta}_{\text{MLE}} = \arg \max_{\theta} \ell(\theta; X, y)
\]

\[
= \arg \max_{\theta} \sum_{i=1}^{n} \log p(y_i \mid x_i, \theta)
\]

\[
= \arg \max_{\theta} \sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left( -\frac{(y_i - f(x_i; \theta))^2}{2\sigma^2} \right)
\]
\[
= \arg \max_\theta \sum_{i=1}^n \left[ -\frac{1}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} (y_i - f(x_i; \theta))^2 \right] \\
= \arg \min_\theta \sum_{i=1}^n (y_i - f(x_i; \theta))^2
\]

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 this method is called **nonlinear least squares**.

Motivated by the MLE formulation above, our goal is to solve the following optimization problem:

\[
\min_\theta \epsilon(\theta) = \min_\theta \frac{1}{2} \sum_{i=1}^n (y_i - f(x_i; \theta))^2
\]

One way to solve this optimization problem is to find all of its critical points and choose the point that minimizes the objective. From **first-order optimality conditions**, the gradient of the objective function at any minimum must be zero:

\[
\nabla_\theta \epsilon(\theta) = \sum_{i=1}^n (y_i - f(x_i; \theta)) \nabla_\theta f(x_i; \theta) = 0
\]

In compact matrix notation:

\[
\nabla_\theta \epsilon(\theta) = J(\theta)^	op (y - F(\theta)) = 0
\]

where

\[
F(\theta) = \begin{bmatrix} f(x_1; \theta) \\ \vdots \\ f(x_n; \theta) \end{bmatrix}, \quad J(\theta) = \begin{bmatrix} \nabla_\theta f(x_1; \theta)^	op \\ \vdots \\ \nabla_\theta f(x_n; \theta)^	op \end{bmatrix}
\]

\( J \) is also referred to as the **Jacobian** of \( F \). Observe that in the special case when \( f \) is linear in \( \theta \) (i.e. \( f(x_i; \theta) = \theta^	op x_i \)), the gradient \( \nabla_\theta \epsilon(\theta) \) will only depend \( \theta \) in \( F(\theta) \) because the term \( \nabla_\theta f(x_i; \theta) \) will only depend on \( x_i \):

\[
\nabla_\theta \epsilon(\theta) = \sum_{i=1}^n (y_i - \theta^	op x_i) \nabla_\theta (\theta^	op x_i) = \sum_{i=1}^n (y_i - \theta^	op x_i) x_i = X^	op (y - X\theta)
\]

and we can derive a closed-form solution for \( \theta \), arriving at the OLS solution:

\[
X^	op (y - X\theta) = 0 \\
X^	op y - X^	op X\theta = 0 \\
X^	op y = X^	op X\theta \\
\theta = (X^	op X)^{-1} X^	op y
\]

In the general case where \( f \) is nonlinear in \( \theta \), it is not necessarily possible to derive a closed-form solution for \( \theta \), for a few reasons. First of all, without additional assumptions on \( f \), the
NLS objective may not be convex. Therefore there may exist values of \( \theta \) that are not global minima, but nonetheless \( \nabla_{\theta} \epsilon(\theta) = 0 \) — they could be local minima, saddle points, or worse, local maxima! Second of all, even if the objective is convex, we may not be able to solve the equation \( J(\theta)^{\top}(y - F(\theta)) = 0 \) for \( \theta \).

1.2 Gauss-Newton Algorithm

Since there is no closed-form solution to the nonlinear least squares optimization problem in general, we must resort an iterative algorithm instead. One such algorithm is the Gauss-Newton algorithm. 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 \( \theta \) at iteration \( k \), which we denote \( \theta^{(k)} \). We consider the first-order approximation of \( F(\theta) \) about \( \theta^{(k)} \):

\[
F(\theta) \approx \tilde{F}(\theta) = F(\theta^{(k)}) + \frac{\partial}{\partial \theta} F(\theta^{(k)})(\theta - \theta^{(k)})
\]

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

Now that \( \tilde{F} \) is linear in \( \Delta \theta \) (the Jacobian and \( F \) are just constants: functions evaluated at \( \theta^{(k)} \)), our objective is convex and we can perform linear least squares to form the closed form solution for \( \Delta \theta \). Applying the first-order optimality condition to the objective \( \tilde{F} \) yields the following equation:

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

Note that the Jacobian of the linearized function \( \tilde{F} \), evaluated at any \( \theta \), is precisely \( J(\theta^{(k)}) \). Denoting \( J = J(\theta^{(k)}) \) and \( \Delta y := y - F(\theta^{(k)}) \) for brevity, we have

\[
J^\top(\Delta y - J \Delta \theta) = 0
\]
\[
J^\top \Delta y = J^\top J \Delta \theta
\]
\[
\Delta \theta = (J^\top J)^{-1} J^\top \Delta y
\]

Comparing this solution to OLS, we see that it is effectively solving

\[
\Delta \theta = \arg \min_{\delta \theta} \| J \delta \theta - \Delta y \|^2
\]

where \( J \) represents \( X \) in OLS, \( \Delta y \) represents \( y \) in OLS, and \( \delta \theta \) represents \( \theta \) in OLS. At each iteration we are effectively minimizing the objective with respect to the linearization of \( F \) at the current iterate \( \theta^{(k)} \). Since \( \delta F \approx J \delta \theta \), we can expect that the minimization with respect to \( \tilde{F} \) is also optimal with respect to \( F \) in the local region around \( \theta^{(k)} \). Recalling that \( \Delta \theta = \theta - \theta^{(k)} \), we can improve upon our current guess \( \theta^{(k)} \) with the update

\[
\theta^{(k+1)} = \theta^{(k)} + \Delta \theta
\]
\[
= \theta^{(k)} + (J^\top J)^{-1} J^\top \Delta y
\]

**Algorithm 1: Gauss-Newton**

Initialize \( \theta^{(0)} \) with some guess

\[\text{while } \theta^{(k)} \text{ has not converged do}\]

- Compute Jacobian with respect to the current iterate: \( J = J(\theta^{(k)}) \)
- Compute \( \Delta y = y - F(\theta^{(k)}) \)
- Update: \( \theta^{(k+1)} = \theta^{(k)} + (J^\top J)^{-1} J^\top \Delta y \)

Note that the solution will depend on the initial value \( \theta^{(0)} \) in general. There are several choices for measuring convergence. Some common choices include testing changes in the objective value:

\[
\left| \frac{\epsilon^{(k+1)} - \epsilon^{(k)}}{\epsilon^{(k)}} \right| \leq \text{threshold}
\]

or in the iterates themselves:

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