1 Nonlinear Least Squares

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

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 $f(\mathbf{x}; \boldsymbol{\theta})$ of the *raw* input $\mathbf{x}$? 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 $\boldsymbol{\theta} = (\theta_1, \theta_2)$ of a robot. We are given noisy distance estimates $Y_i \in \mathbb{R}$ from n sensors whose positions $\mathbf{x}_i \in \mathbb{R}^2$ are fixed and known. Since we are predicting *distance*, it is reasonable to use the model $f(\mathbf{x}; \boldsymbol{\theta}) = \|\mathbf{x} - \boldsymbol{\theta}\|_2$. 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 $f(\mathbf{x}; \boldsymbol{\theta})$ which can approximate any function $f(\mathbf{x})$ with appropriate parameters $\boldsymbol{\theta}$. 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 $\boldsymbol{\theta}$:

$$Y_i = f(\mathbf{x}_i; \boldsymbol{\theta}) + Z_i, \quad Z_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2), \quad i = 1, \ldots, n$$

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

$$\begin{aligned}
\hat{\boldsymbol{\theta}}_{\text{MLE}} &= \arg\max_{\boldsymbol{\theta}} \ \ell(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) \\
&= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log p(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) \\
&= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - f(\mathbf{x}_i; \boldsymbol{\theta}))^2}{2\sigma^2} \right)
\end{aligned}$$

$$= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \left[-\frac{1}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}(y_i - f(\mathbf{x}_i;\boldsymbol{\theta}))^2 \right]$$
$$= \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i;\boldsymbol{\theta}))^2$$

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:

$$\min_{\boldsymbol{\theta}} \epsilon(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \frac{1}{2} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i;\boldsymbol{\theta}))^2$$

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:

$$\nabla_{\boldsymbol{\theta}} \epsilon(\boldsymbol{\theta}) = \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i;\boldsymbol{\theta})) \nabla_{\boldsymbol{\theta}} f(\mathbf{x}_i;\boldsymbol{\theta}) = \mathbf{0}$$

In compact matrix notation:

$$\nabla_{\boldsymbol{\theta}} \epsilon(\boldsymbol{\theta}) = J(\boldsymbol{\theta})^\top (\mathbf{y} - F(\boldsymbol{\theta})) = \mathbf{0}$$

where

$$F(\boldsymbol{\theta}) = \begin{bmatrix} f(\mathbf{x}_1;\boldsymbol{\theta}) \\ \vdots \\ f(\mathbf{x}_n;\boldsymbol{\theta}) \end{bmatrix}, \quad J(\boldsymbol{\theta}) = \begin{bmatrix} \nabla_{\boldsymbol{\theta}} f(\mathbf{x}_1;\boldsymbol{\theta})^\top \\ \vdots \\ \nabla_{\boldsymbol{\theta}} f(\mathbf{x}_n;\boldsymbol{\theta})^\top \end{bmatrix}$$

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

$$\nabla_{\boldsymbol{\theta}} \epsilon(\boldsymbol{\theta}) = \sum_{i=1}^{n} (y_i - \boldsymbol{\theta}^\top \mathbf{x}_i) \nabla_{\boldsymbol{\theta}} (\boldsymbol{\theta}^\top \mathbf{x}_i) = \sum_{i=1}^{n} (y_i - \boldsymbol{\theta}^\top \mathbf{x}_i) \mathbf{x}_i = \mathbf{X}^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})$$

and we can derive a closed-form solution for $\boldsymbol{\theta}$, arriving at the OLS solution:

$$\begin{aligned} \mathbf{X}^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) &= \mathbf{0} \\ \mathbf{X}^\top\mathbf{y} - \mathbf{X}^\top\mathbf{X}\boldsymbol{\theta} &= \mathbf{0} \\ \mathbf{X}^\top\mathbf{y} &= \mathbf{X}^\top\mathbf{X}\boldsymbol{\theta} \\ \boldsymbol{\theta} &= (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y} \end{aligned}$$

In the general case where f is nonlinear in $\boldsymbol{\theta}$, it is not necessarily possible to derive a closed-form solution for $\boldsymbol{\theta}$, 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 $\boldsymbol{\theta}$ that are not global minima, but nonetheless $\nabla_{\boldsymbol{\theta}} \epsilon(\boldsymbol{\theta}) = \mathbf{0}$ — 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 $J(\boldsymbol{\theta})^\top(\mathbf{y} - F(\boldsymbol{\theta})) = \mathbf{0}$ for $\boldsymbol{\theta}$.

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 $\boldsymbol{\theta}$ at iteration k, which we denote $\boldsymbol{\theta}^{(k)}$. We consider the first-order approximation of $F(\boldsymbol{\theta})$ about $\boldsymbol{\theta}^{(k)}$:

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

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

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

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

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

$$\begin{aligned} \mathbf{J}^\top(\Delta\mathbf{y} - \mathbf{J}\Delta\boldsymbol{\theta}) &= \mathbf{0} \\ \mathbf{J}^\top\Delta\mathbf{y} &= \mathbf{J}^\top\mathbf{J}\Delta\boldsymbol{\theta} \\ \Delta\boldsymbol{\theta} &= (\mathbf{J}^\top\mathbf{J})^{-1}\mathbf{J}^\top\Delta\mathbf{y} \end{aligned}$$

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

$$\Delta\boldsymbol{\theta} = \arg\min_{\delta\boldsymbol{\theta}} \|\mathbf{J}\delta\boldsymbol{\theta} - \Delta\mathbf{y}\|^2$$

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

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \Delta\boldsymbol{\theta}$$

$$= \boldsymbol{\theta}^{(k)} + (\mathbf{J}^\top \mathbf{J})^{-1} \mathbf{J}^\top \Delta \mathbf{y}$$

Algorithm 1: Gauss-Newton

Initialize $\boldsymbol{\theta}^{(0)}$ with some guess
while $\boldsymbol{\theta}^{(k)}$ *has not converged* **do**
  Compute Jacobian with respect to the current iterate: $\mathbf{J} = J(\boldsymbol{\theta}^{(k)})$
  Compute $\Delta \mathbf{y} = \mathbf{y} - F(\boldsymbol{\theta}^{(k)})$
  Update: $\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + (\mathbf{J}^\top \mathbf{J})^{-1} \mathbf{J}^\top \Delta \mathbf{y}$

Note that the solution will depend on the initial value $\boldsymbol{\theta}^{(0)}$ 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:

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