1 Introduction

We say that the regression function \( m(x_t, \beta) = \mathcal{E}(y_t | x_t, \beta) \) is nonlinear in the parameters if it cannot be written as \( m(x_t, \beta) = f(x_t)' \beta \) for some function \( f(\cdot) \).

Examples of nonlinear regression functions include

\[
\begin{align*}
    m(x, \beta) & = \beta_1 + \beta_2 \frac{x}{1 + \beta_3 x} \\
    m(x, \beta) & = \beta_1 + \beta_2 x^{\beta_3} \\
    m(x, \beta) & = \beta_1 + \beta_2 \exp(\beta_3 x) \\
    m(x, \beta) & = \beta_1 + \beta_2 x + \beta_4 x I(x > \beta_3).
\end{align*}
\]

In the first three examples, \( m(x, \beta) \) is (generically) differentiable in the parameters \( \beta \). In the final example, \( m(\cdot) \) is not differentiable with respect to \( \beta_3 \).

Nonlinear regression is frequently adopted because the functional form \( m(x, \beta) \) is suggested by an economic model. In other cases, it is adopted as a flexible approximation to an unknown regression function.

This document provides a brief introduction to the nonlinear least squares problem and is organized as follows: Section 2 presents the nonlinear least squares (NLLS) estimator. Section 3 discusses issues related to numerical methods that can be used to obtain it. Section 4 addresses the issue of inference in the NLLS context. Section 5 generalizes the NLLS estimator when homoskedasticity and/or autocorrelation are present. Finally, Section 6 presents a particular class of nonlinear models denominated artificial neural networks (ANN).
2 NLLS Estimation

The NLLS estimator can be defined as the estimator that solves the following optimization problem:

\[ \hat{\beta}_{NLLS} = \arg \min_{\beta} S_T(\beta), \]

where

\[
S_T(\beta) = \sum_{t=1}^{T} (y_t - m(x_t, \beta))^2 \\
= [Y - m(X, \beta)]' [Y - m(X, \beta)].
\]

For notational convenience, let \( m_t = m(x_t, \beta) \). When it exists, we form the Jacobian matrix \( Z(\beta) \) as the stacked gradient vectors for each observation

\[
Z(\beta) = \frac{\partial m}{\partial \beta} = \begin{bmatrix}
\frac{\partial m_1}{\partial \beta_1} & \cdots & \frac{\partial m_1}{\partial \beta_k} \\
\vdots & \ddots & \vdots \\
\frac{\partial m_T}{\partial \beta_1} & \cdots & \frac{\partial m_T}{\partial \beta_k}
\end{bmatrix}.
\]

A more general characterization of the NLLS estimator can be defined as the set of parameters that solve the FONC:

\[
\frac{\partial S_T(\beta)}{\partial \beta} \bigg|_{\hat{\beta}} = -2Z(\hat{\beta})' [Y - m(X, \hat{\beta})] = 0. \tag{1}
\]

At least three features of (1) are worth discussing: First, the OLS estimator is a particular case of the NLLS estimator, given that in the former \( m(X, \beta) = X\beta \) and \( Z(\beta) = X \) in which case (1) has an analytical solution that coincides with OLS. Second, when (1) is a nonlinear function of \( \beta \), it cannot be solved analytically and numerical methods are required. Finally, given that the FONC (1) are nonlinear there may be more than one root that solves the problem, and a general evaluation for a global minimum may be needed.

The following section discusses how to use numerical methods to obtain the NLLS estimator.

3 Numerical Methods

An algorithm that is commonly used for NLLS estimation problems is the Gauss-Newton method. Here we present the algorithm and its properties.
3.1 Gauss-Newton Method

Numerical algorithms are often used in order to solve nonlinear problems for which an analytical solution is not available. In the case of NLLS estimation, it is important to consider that the source of nonlinearity comes from the function $m(x_t, \beta)$.

Consider a first-order Taylor series approximation of $m(x_t, \beta)$ about a starting value $\beta_0$:

$$m(x_t, \beta) \approx m(x_t, \beta_0) + \frac{\partial m_t}{\partial \beta} \bigg|_{\beta_0} (\beta - \beta_0).$$

(2)

If we substitute the first-order approximation (2) in the objective function we obtain

$$S_T(\beta) \approx \sum_{t=1}^{T} \left[ y_t - m(x_t, \beta_0) - \frac{\partial m_t}{\partial \beta} \bigg|_{\beta_0} (\beta - \beta_0) \right]^2.$$  

(3)

Note that (3) is not the original problem but an approximation near $\beta_0$. This new objective function is now quadratic in $\beta$ (given that $\beta_0$ is known), thus the FONC will be linear in $\beta$ and can be solved analytically.

Consequently, if we consider the FONC, the problem of estimating $\beta$ in a neighborhood of $\beta_0$ is a linear least squares problem. Conditional on the starting value $\beta_0$, the solution to the problem is

$$\beta_1 = \beta_0 + \left[ Z(\beta_0)' Z(\beta_0) \right]^{-1} \left[ Z(\beta_0)' (Y - m(X, \beta_0)) \right].$$

(4)

We repeat the Gauss-Newton iterations until convergence ($\beta_j \approx \beta_{j-1}$) and set $\beta_j = \hat{\beta}_{NLLS}$. Once the NLLS estimator for $\beta$ is obtained, we can derive consistent estimators for $\sigma^2$ using $\hat{\sigma}^2 = T^{-1} S_T (\beta)$ or $\bar{\sigma}^2 = (T - k)^{-1} S_T (\hat{\beta})$.

3.1.1 Alternative Motivation for Gauss-Newton

If instead of substituting (2) into the objective function, we replace it in the model we have

$$y_t - m_t(\beta_0) + \frac{\partial m_t}{\partial \beta} \bigg|_{\beta_0} \beta_0 \approx \frac{\partial m_t}{\partial \beta} \bigg|_{\beta_0} \beta + u_t$$

or, in matrix notation

$$Y - m(X, \beta_0) + Z(\beta_0) \beta_0 \approx Z(\beta_0) \beta + u$$

$$Y^* \approx X^* \beta + u.$$
As this specification satisfies the OLS principles, $\beta_1$ can be obtained from a linear regression of $Y^*$ on $X^*$. That is,

$$
\beta_1 = (X^*X^*)^{-1}X^*Y^*
$$

$$
= [Z (\beta_0)' Z (\beta_0)]^{-1} [Z (\beta_0)' (Y - m (X, \beta_0) + Z (\beta_0) \beta_0)]
$$

$$
= \beta_0 + [Z (\beta_0)' Z (\beta_0)]^{-1} [Z (\beta_0)' (Y - m (X, \beta_0))]
$$

which is precisely the estimator derived in (4).

The principal advantage of the Gauss-Newton algorithm is convenience. First, note that the step taken to compute $\beta_j$ from $\beta_{j-1}$ is simply the linear regression of the approximation errors ($Y - m$) on the columns of the Jacobian matrix (evaluated at $\beta_{j-1}$). Second, the method only requires the user to evaluate the nonlinear regression function and to compute the Jacobian matrix for each step, while other methods (that we will discuss later) require the computation of a Hessian matrix. Finally, it is important to remember that the Gauss-Newton method is only applicable to problems that may be equivalently solved by minimizing the nonlinear least-squares objective function.

Two important technical details with respect to the Gauss-Newton algorithm are important. First, in order for the algorithm to work when evaluated at $\beta_j$; there may be instances in which this condition is not satisfied and variants of the method must be considered to ensure that the matrix is positive definite. Second, given that the algorithm requires the computation of the Jacobian matrix $Z (\beta_j)$ for each iteration, whenever possible, it is desirable to obtain analytical expressions for the derivatives and not rely on numerical derivatives which may not be as accurate.

3.1.2 Starting Values and Convergence Criteria

Gauss-Newton is an iterative solution algorithm. A fully specified algorithm based on an iterative scheme should have three components: a method for deciding on a starting value for the iteration, a method for obtaining the next iterate from its predecessors, and a method for deciding when to stop the iterative process. The preceding subsection examined how to generate the iterative sequence itself, but treated as given the starting value ($\beta_0$) and stopping criteria. Next we discuss the practicalities of selecting initial values and determining convergence.

Starting values matter in two ways. First, if the initial value for the iteration is too far away from the solution, the iteration can diverge. Second, it is possible that the FONC $\frac{\partial S_T(\beta)}{\partial \beta} = 0$ have multiple roots. In this case, the root to which the sequence converges will depend on the starting value for the iteration. Unfortunately, there is little constructive theory about choosing starting values, however a few words of

---

2In fact, good commercial softwares never use the Gauss-Newton algorithm as described above. For example, GAUSS generally uses a wide variety of methods known as quasi-Newton algorithms.
general advice can be given. Most of the time, the equations being solved are similar
to others whose roots are easily obtained, so that a root of the latter can be used
as a starting point for the equations of actual interest. However, sometimes there is
no alternative for starting than with a guess or two and to observe the progress of
the iteration, hoping that it will be possible to adjust the starting point or iteration
method as needed to achieve convergence. A useful approach, when possible, is to
graph the functions. This can often provide not only good starting values, but also
some insight concerning an appropriate form for the iteration.

The problem of detecting multiple roots, and of settling on the appropriate one
when more than one root is found, is a difficult one. Probably the most successful
general approach for discovering whether there are multiple roots is to start the
iteration several times, from the vicinity of possible solutions if enough is known
about the function, or using randomly chosen starting values otherwise. Figure 1
provides an example of such a case, where the objective function has three critical
values, two of each are minima and one a maximum. If our objective is to minimize
$S_T(\beta)$ and start the algorithm with an initial value such as $a$, the algorithm will
eventually converge to $\hat{\beta}_1$ which is not a global minimum, while if $b$ is used as the
initial value, the algorithm will converge to $\hat{\beta}_2$.

![Graph showing multiple roots and initial values](image.png)

Figure 1: Multiple Roots and Initial Values

There is more to say about stopping an iteration than starting one. There are
two reasons for bringing an iteration to a halt: either the iteration has converged
or it has not. Since the solution of the equation is not known explicitly, the de-
cision as to whether an iteration has converged is based on monitoring either the sequence of iterates to see if \( \beta_j \) is sufficiently close to \( \beta_{j-1} \), or the sequence of function evaluations \( \left. \frac{\partial S_T(\beta)}{\partial \beta} \right|_{\bar{\beta}_j} \) to see if these become sufficiently close to zero. If \( \beta \) is a scalar, the two most common definitions for successive iterates to be “sufficiently close” are embodied in the absolute converge criterion, which asserts convergence when \( |\beta_j - \beta_{j-1}| < tol \), and the relative converge criterion, which asserts convergence when \( \left| (\beta_j - \beta_{j-1}) / \beta_{j-1} \right| < tol \), where \( tol \) is a preselected tolerance. The absolute convergence criterion is most suitable when the solution is close to zero; in this case the denominator of the relative criterion can foster numerical difficulties. On the other hand, when the solution is large (far away from zero) the relative criterion is generally more satisfactory. Once the convergence criterion is satisfied, we ask if \( \left. \frac{\partial S_T(\beta)}{\partial \beta} \right|_{\bar{\beta}_j} \) is “nearly” zero. More precisely we stop if \( \left| \left. \frac{\partial S_T(\beta)}{\partial \beta} \right|_{\bar{\beta}_j} \right| \leq \delta \) for some prespecified \( \delta \). If we want high precision, we will choose small \( \delta \), but that choice must be reasonable. Choosing \( \delta = 0 \) is nonsense, since it is unachievable; equally pointless is choosing \( \delta = 10^{-20} \) on a 12-digit machine where \( \left. \frac{\partial S_T(\beta)}{\partial \beta} \right|_{\bar{\beta}_j} \) can be calculated with at most 12 digits of accuracy.\(^3\)

When \( \beta \) is a vector, it is usually satisfactory to define relative and absolute convergence in terms of some norm on \( \beta \) such as the \( l_2 \) (euclidean) norm (sum of squares) or the sup-norm (maximum component magnitude). Always remember that even if \( \beta_j \) satisfies the stopping rule, we want to check that \( \left. \frac{\partial S_T(\beta)}{\partial \beta} \right|_{\bar{\beta}_j} \) is close to zero.

### 3.2 Concentration

A major simplification can be achieved through concentration. This can be done when we partition \( \beta = (\gamma, \delta) \) so that

\[
m(x_t, \beta) = \gamma' x_t(\delta),
\]

where \( x_t(\delta) \) is a \( k \times 1 \) function of \( x_t \) and \( \delta \). In all the examples presented on the Introduction, this can be done with \( \delta \) of much smaller dimension than \( \gamma \). In many cases, \( \delta \) is a scalar.

---

\(^3\)The range of numbers that are machine-representable varies greatly across machines; one should always have a good idea of their value when working on a computer. *Machine epsilon* is the smallest relative quantity that is machine-representable. Formally this is the smallest \( \epsilon \) such that the machine knows that \( 1 - \epsilon < 1 < 1 + \epsilon \). It is also important to know *machine infinity*, that is, the largest number such that both it and its negative are representable. *Overflow* occurs when an operation takes machine representable numbers but wants to produce a number which exceeds machine infinity in magnitude. A *machine zero* is any quantity that is equivalent to zero on the machine. *Underflow* occurs when an operation takes nonzero quantities but tries to produce a nonzero magnitude less than machine zero. The analyst must either know these important constants for his machine or the more conservative guesses. Much of the software contains a section where the user must specify these arithmetic constants.
The SSR function \( S_T(\beta) = S_T(\gamma, \delta) \) and thus
\[
\min_{\beta} S_T(\beta) = \min_{\delta} \min_{\gamma} S_T(\gamma, \delta).
\]

Since \( \gamma \) enters the model linearly, we see that
\[
\hat{\gamma}(\delta) = \arg \min_{\gamma} S_T(\gamma, \delta)
\]
\[
= \left[ X(\delta)'X(\delta) \right]^{-1} X(\delta)' Y,
\]
where \( X(\delta) \) is the \( T \times k \) matrix of the stacked \( x_t(\delta)' \).

Now set
\[
S_T(\delta) = S_T(\hat{\gamma}(\delta), \delta),
\]
which is the concentrated sum of squared residuals. We have \( \hat{\delta} = \arg \min_{\delta} S_T(\delta) \) and \( \hat{\gamma} = \hat{\gamma}(\hat{\delta}) \). The pair \( (\hat{\gamma}, \hat{\delta}) \) are the joint NLLS estimates of \( (\gamma, \delta) \).

The main benefit of concentration is that the dimension of the numerical optimization is typically reduced dramatically. When \( \delta \) is scalar, the final minimization over \( \delta \) can be done by a grid search.

4 Inference with Linear Constraints

Given the interpretation that we provided for the NLLS estimator, it is not difficult to derive its asymptotic variance-covariance matrix. As motivated in section 3.1.1, the NLLS is the OLS estimator of the transformed model of \( Y^* \) on \( X^* \); in which case
\[
\mathcal{V} \left( \hat{\beta}_{NLLS} | X \right) = \mathcal{E} \left[ (\hat{\beta}_{NLLS} - \beta)(\hat{\beta}_{NLLS} - \beta)' \right] = \sigma^2 (X^*X^*)^{-1},
\]
or, if we define \( \hat{Z} = Z(\hat{\beta}_{NLLS}) \),
\[
\mathcal{V} \left( \hat{\beta}_{NLLS} | X \right) = \sigma^2 (\hat{Z}'\hat{Z})^{-1}.
\]
(5)

As \( \sigma \) is not known, a consistent estimator of the variance-covariance matrix of \( \hat{\beta} \) can be derived by replacing \( \sigma \) with \( \hat{\sigma} \) or \( \tilde{\sigma} \). It is important mention, that in contrast to the OLS estimator, we never claimed that the NLLS estimator was unbiased. The only property that we attach to the NLLS estimator is that under certain circumstances (that we will discuss latter), it is consistent. For that reason, all the tests that we conduct are only asymptotically valid and may perform poorly in small samples.

Once an estimator for the variance-covariance matrix of \( \hat{\beta}_{NLLS} \) is obtained, inference and hypothesis testing can be conducted as usual. Next, we discuss some of these tests.
4.1 The $t$ Test

As was the case with the OLS estimator, if we are interested in testing $H_0: Q'\beta = c$ when the null hypothesis corresponds to a single linear combination of parameters; that is, when $q = 1$, we construct

$$\frac{Q'\hat{\beta} - c}{\tilde{\sigma}\left[Q'(\hat{Z}'\hat{Z})^{-1}Q\right]^{1/2}} \overset{a}{\sim} S_{T-k},$$

where $\overset{a}{\sim}$ is meant to imply “is approximately distributed as”. We will show later that as was the case with the OLS estimator, this test is asymptotically distributed as a $N(0, 1)$. In fact, given that the properties of $\hat{\beta}$ are only known asymptotically, it is preferable to take the critical values of the normal distribution, or when small sample deviations appear to be important, to obtain critical values using the bootstrap or a similar method.

With these tools we can construct confidence intervals $C_T$ of $\beta_i$. Given that $C_T$ is a function of the data, we must always recall that it is random. Its objective is to cover $\beta_i$ with high probability. The coverage probability is $Pr(\beta \in C_T)$. We say that $C_T$ has $(1 - \alpha)$% coverage for $\beta$ if $Pr(\beta \in C_T) \to (1 - \alpha)$. We construct a confidence interval as follows:

$$Pr\left[\hat{\beta}_i - z_{\alpha/2}\sqrt{\hat{V}_{i,i}} < \beta_i < \hat{\beta}_i + z_{\alpha/2}\sqrt{\hat{V}_{i,i}}\right] = 1 - \alpha,$$

where $z_{\alpha/2}$ is the upper $\alpha/2$ quantile of the distribution being considered (asymptotically it should be the normal distribution). The most common choice of $\alpha$ is 0.05.

4.2 The $F$ Test

When $q > 1$ we cannot apply the $t$ test described above. In that case we can use a simple pseudo-Likelihood Ratio Test which we will discuss at length later. For now suffices to state the basic results. Under the null hypothesis, it can be shown that

$$T - k \frac{S_T(\bar{\beta}) - S_T(\hat{\beta})}{q S_T(\hat{\beta})} \overset{a}{\sim} F_{q, T-k}.$$

The asymptotically valid test takes the form

$$\frac{S_T(\bar{\beta}) - S_T(\hat{\beta})}{\tilde{\sigma}^2} \overset{D}{\sim} \chi^2_q.$$  

(6)
In contrast to inference in OLS, (6) is not equivalent to the Wald test when the model is estimated in the NLLS context. In this case, the Wald test takes the form

\[
T - k \left( \frac{Q' \hat{\beta} - c}{q} \right)' \left[ Q' \left( \hat{Z}' \hat{Z} \right)^{-1} Q \right]^{-1} \left( Q' \hat{\beta} - c \right) \sim F_{q,T-k}.
\]

The asymptotically valid Wald test can be computed as

\[
\left( \frac{Q' \hat{\beta} - c}{\hat{\sigma}^2} \right)' \left[ Q' \left( \hat{Z}' \hat{Z} \right)^{-1} Q \right]^{-1} \left( \frac{Q' \hat{\beta} - c}{\hat{\sigma}^2} \right) \xrightarrow{D} \chi^2_q.
\] (7)

Once again, as in the case of \( t \) tests, we reject the null hypothesis when the value computed exceeds the critical value.

Gallant provides Monte Carlo evidence that suggests that (6) has higher power than (7) in small samples, and it should be used when possible. Nevertheless, notice that in order to obtain (6) two optimizations have to be considered in order to obtain the \( SSR \) of the constrained and unconstrained model, and even with linear constraints, the constrained model may induce the more nonlinearities than the unconstrained model, making its computation more demanding. As usual, there is no free lunch!

### 4.3 Identification

Identification is often tricky in nonlinear regression models. Suppose that

\[ m(x_t, \beta) = \beta'_1 z_t + \beta'_2 x_t \delta. \]

This model is linear when \( \beta_2 = 0 \), and this is often a useful hypothesis to consider. Thus, we want to test

\[ H_0 : \beta_2 = 0. \]

However, under \( H_0 \), the model is:

\[ y_t = \beta'_1 z_t + u_t \]

and both, \( \beta_2 \) and \( \delta \), have dropped out. This means that under \( H_0 \), \( \delta \) is not identified. This renders the usual distribution theory invalid. Thus, when the truth is that \( \beta_2 = 0 \), the parameter estimates are not asymptotically normally distributed. Furthermore, tests of \( H_0 \) do not have asymptotic normal or chi-square distributions.

The asymptotic theory of such tests is non-standard and is subject of recent developments in econometrics. Simulation techniques (similar to bootstrap) are usually used to construct critical values in a given application.
5 GLS and NLLS

The GLS and FGLS procedures can be extended to estimate the parameters of nonlinear regression models. As was the case, with linear models when $\Omega \neq I_T$, the NLLS estimator derived in (4) will still be consistent, but not efficient. Once again, we discuss how to proceed in the case where $\Omega$ is known, when it is unknown, and how to conduct inference if the pattern of heteroskedasticity and/or autocorrelation is unknown.

5.1 Nonlinear GLS Estimator

As was the case with the linear model, as $\Omega$ is a known positive definite matrix, it can be factored into $\Omega = C\Lambda C'$ where the columns of $C$ are the eigenvectors of $\Omega$ and the eigenvalues of $\Omega$ are arrayed in the diagonal matrix $\Lambda$. Let $\Lambda^{1/2}$ be the diagonal matrix with $i$th element $\sqrt{\lambda_i}$, where $\lambda_i$ is the $i$th eigenvalue of $\Omega$. Define $R = C\Lambda^{1/2}$ and $P' = C\Lambda^{-1/2}$, then $\Omega = RR'$ and $\Omega^{-1} = P'P$.

If we consider the transformed model described in section 3.1.1 and define $Y_* = PY^*, X_* = PX^*$, and $u_* = Pu$ we obtain

$$Y_* = X_*\beta + u_*,$$

where

$$V(u_*) = E(u_*u'_*) = \sigma^2PP' = \sigma^2I.$$

Thus, all the assumption that led us to the derivation of the OLS estimator are satisfied in the transformed model (8). In this case we have:

$$\beta_1 = (X'_*X_*)^{-1}X'_*Y_*$$
$$= (X'^*\Omega^{-1}X^*)^{-1}X'^*\Omega^{-1}Y^*$$
$$= \beta_0 + [Z(\beta_0)'\Omega^{-1}Z(\beta_0)]^{-1}\{Z(\beta_0)'\Omega^{-1}(Y - m(X, \beta_0))\},$$

which is simply a generalization of Gauss-Newton algorithm that incorporates information regarding $\Omega$. We denote by $\hat{\beta}_{GNLLS}$ to the value of $\beta$ at which this iterative process converges.

The variance-covariance of $\hat{\beta}_{GNLLS}$ is

$$V(\hat{\beta}_{GNLLS} | X) = \sigma^2(X'_*X_*)^{-1} = \sigma^2\left(Z'\Omega^{-1}Z\right)^{-1}.$$
An estimator of $\sigma^2$ based on the GNLLS estimator is

$$\tilde{\sigma}^2_{\text{GNLLS}} = \frac{\tilde{u}'_s \tilde{u}_s}{T - k}.$$ 

Once the estimator of the variance-covariance matrix is obtained, inference can be conducted exactly as we showed for the GLS estimator of the linear model.

### 5.2 Nonlinear FGLS

Previously, we assumed that $\Omega$ was known, in which case a simple transformation yielded a noise variance-covariance matrix that was proportional to the identity matrix. When $\Omega$ is unknown, we simply replace the unknown $\Omega$ with an estimator $\hat{\Omega}$. This would lead to the Feasible Generalized Nonlinear Least Squares (FGNLLS) estimator of $\beta$ defined by

$$\hat{\beta}_{\text{FGNLLS}} = \left( X' \hat{\Omega}^{-1} X^* \right)^{-1} \left( X' \hat{\Omega}^{-1} Y^* \right). \quad (9)$$

As was the case with the FGLS estimator in the linear context, we must impose a structure in order to obtain the estimator of $\Omega$. In practice, the elements of $\Omega$ are assumed to be functions, $\Omega(\theta)$, of a reduced and fixed number of unknown parameters $\theta$ that remain unchanged as the sample size increases. The problem then reduces to obtaining $\hat{\theta}$ and use it to compute $\hat{\Omega} = \Omega(\hat{\theta})$ that is then replaced in (9). All the problems and procedures that we discussed for the linear case are valid here.

### 5.3 Applying NLLS: Ignoring that $\Omega \neq I_T$

As was the case with the linear model, if we ignore that $\Omega \neq I_T$, the NLLS estimator will still be consistent but inefficient. More importantly, inference based on the variance-covariance matrix described in (5) will be misleading, given that in this case it will be given by

$$\nu \left( \hat{\beta}_{\text{NLLS}} | X \right) = \sigma^2 \left( \hat{Z}' \hat{Z} \right)^{-1} \left( \hat{Z}' \hat{\Omega} \hat{Z} \right) \left( \hat{Z}' \hat{Z} \right)^{-1}. \quad (10)$$

In the presence of heteroskedasticity, we can construct a consistent estimator of this covariance matrix using a suitable modification of White’s (1980) matrix $\Sigma$.

$$\hat{\Sigma} = T^{-1} \sum_{t=1}^{T} \hat{u}_t \hat{z}_t \hat{z}_t' \xrightarrow{P} \Sigma.$$ 

In the case of autocorrelation and/or heteroskedasticity we can compute Newey and West’s HAC matrix as:

$$\hat{\Sigma} = T^{-1} \sum_{t=1}^{T} \sum_{s=1}^{T} w(t-s) \hat{u}_t \hat{u}_s \hat{z}_t' \xrightarrow{P} \Sigma,$$ 

$$\text{where} \quad w(t-s) = \begin{cases} \frac{1}{K} & \text{if } |t-s| \leq K, \\ 0 & \text{otherwise} \end{cases}.$$
where $K$ is a finite positive number and $w(t - s) = 1 - \frac{|t-s|}{K}$ is a weighting scheme that ensures that $\hat{\Sigma}$ is positive definite.

In either case, a consistent estimator of the variance-covariance matrix of $\hat{\beta}_{\text{NLLS}}$ would be

$$\hat{V}(\hat{\beta}_{\text{NLLS}} | \mathbf{X}) = T \left( \hat{\mathbf{Z}}' \hat{\mathbf{Z}} \right)^{-1} \hat{\Sigma} \left( \hat{\mathbf{Z}}' \hat{\mathbf{Z}} \right)^{-1}.$$

Once again, these results are extremely useful given that we do not need to know the precise nature of the pattern of heteroskedasticity and/or autocorrelation. Once a consistent estimator of the covariance matrix of the NLLS estimator is obtained, inference can be conducted as usual.

6 Artificial Neural Networks

ANN are a class of input-output models developed by cognitive scientists interested in understanding how computation is performed by the brain. Much is still unknown about how the brain trains itself to process information, so theories abound. The human brain consists of a large number (more than a billion) of neural cells that process informations. Each cell works like a simple processor and only the massive interaction between all cells and their parallel processing makes the brain’s abilities possible.

As figure 2 indicates, a neuron consists of a core, dendrites for incoming information and an axon with dendrites for outgoing information that is passed to connected neurons. Information is transported between neurons in form of electrical stimulations along the dendrites. Incoming informations that reach the neuron’s dendrites is added up and then delivered along the neuron’s axon to the dendrites at its end, where the information is passed to other neurons if the stimulation has exceeded a certain threshold. In this case, the neuron is said to be activated. If the incoming stimulation had been too low, the information will not be transported any further. In this case, the neuron is said to be inhibited.

Despite the fact that ANN are far from anything close to a realistic description of how brains actually work, they have shown their remarkable ability to derive meaning from complicated or imprecise data. They can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an “expert” in the category of information it has been given to analyze.

6.1 Types of ANN

Like the human brain, ANN consists of neurons and connections between them. The neurons are transporting incoming information on their outgoing connections to other

\footnote{See Stergiou and Siganos (1996) and Fröhlich (1996) for definitions and non technical introductions to ANN.}
neurons. In neural net terms these connections are called weights. The information is simulated with specific values stored in those weights.

As shown in figure 3, an artificial neuron looks similar to a biological neural cell. And it works in the same way. Information (called the input) is sent to the neuron on its incoming weights. This input is processed by a propagation function that adds up the values of all incoming weights. The resulting value is compared with a certain threshold value by the neuron’s activation function. If the input exceeds the threshold value, the neuron will be activated, otherwise it will be inhibited. If activated, the neuron sends an output on its outgoing weights to all connected neurons and so on.

**Figure 3: Structure of a Neuron in a Neural Net**

In a neural net, the neurons are grouped in layers, called neuron layers. Usually each neuron of one layer is connected to all neurons of the preceding and the following layer (except the input layer and the output layer of the net). The information given
to a neural net is propagated layer-by-layer from input layer to output layer through either none, one or more hidden layers.

Several types of neural nets exist depending on how the layers are connected among each other (see figure 4). For example, a perceptron is a very simple neural network with two layers (input and output).\(^5\) Feedforward neural nets allow only neuron connections between two different layers, while nets of the feedback type have also connections between neurons of the same layer.

### 6.2 Feedforward Neural Networks

We are interested in approximating \(E(y_t | x_t) = m(x_t)\), but we do not know \(m(\cdot)\). By far, the artificial neural network most commonly used to approximate it is the single hidden layer, \(q\) hidden units, feedforward neural network feedforward network.

---

\(^5\)In its simplest version, a perceptron can be recognized as a linear model.
(or ANN($q$) for short):

$$m(x_t) \simeq F(x_t, \beta) = \sum_{j=1}^{q} G(x_t \gamma_j) \theta_j,$$

where $\beta$ is the vector of parameters that characterizes the network. Associated with the input layer ($x$), there is a single hidden layer with $q$ values of $x_t \gamma_j$ (hidden units) that are “connected” through the activation function $G(\cdot)$ and the $\theta_j$ weights to the output layer (which in this case is $y$). The activation function most commonly used is the logistic activation function $G(v) = (1 + e^{-v})^{-1}$.6

A variant of the single hidden layer network, that is particularly relevant in econometric applications, has direct connections from the input to the output layer as well as the hidden layer. Output of this network (known as the Augmented Hidden Layer Feedforward network) can be expressed as:

$$m(x_t) \simeq F(x_t, \beta) = x_0^\alpha + \sum_{j=1}^{q} G(x_t \gamma_j) \theta_j. \quad (11)$$

Specification (11) nests the linear model as a special case (with $\theta_j = 0$ for $j > 0$). Furthermore, due to its parallelism and intrinsic nonlinearity, functions of the form of (11) can be viewed as “universal approximators,” that is, as a flexible functional form that, provided with sufficiently many hidden units and properly adjusted parameters, can approximate $m(\cdot)$ arbitrarily well.7

### 6.2.1 Estimation and Inference

As ANN constitute a particular class of NLLS models, the parameters can be estimated by minimizing:

$$S_T(\beta) = \sum_{t=1}^{T} (y_t - F(x_t, \beta))^2$$

using numerical method such as the one discussed above.

As ANN were first used in fields other than econometrics, ANN practitioners often use an estimation strategy that differs from Gauss-Newton. One method (known as backpropagation) relies on a recursive estimation of $\beta$ and is referred to as “learning.” In practice it is simply a numerical method also known as steepest descent and is not as efficient as Gauss-Newton.

---

6See Kuan and White (1994) for a detailed discussion.

7Although not routinely used, multiple hidden layer networks can further increase nonlinear features of the data and may in cases require fewer hidden units ($q$). For example, a two layer network can uniformly approximate certain mappings containing discontinuities that a single layer network can’t.
A way to improve the properties of numerical methods, is to transform $y$ and $x$ such that they are on a comparable scale. Two of the transformation methods commonly used are:

$$z_t^* = \frac{z_t - \min(z)}{\max(z) - \min(z)} \quad \text{or} \quad z_t^* = \frac{z_t - \bar{z}}{\sigma(z)},$$

where, in the first case the variable is mapped to the [0,1] interval and in the second it is standardized.\(^8\)

Finally, as suggested by Franses and van Dijk (2000), estimation of the parameters may benefit from preventing estimates from becoming unduly large. This can be achieved by augmenting the objective function with a penalty term (commonly referred to as weight decay) to obtain:

$$S_T(\beta) = \sum_{t=1}^{T} (y_t - F(x_t, \beta))^2 + \lambda_\alpha \alpha' \alpha + \lambda_\theta \theta' \theta + \lambda_\gamma \gamma' \gamma,$$

where $\lambda_\alpha$, $\lambda_\theta$, and $\lambda_\gamma$ are small penalty factors specified in advance.

Inference is conducted by using either (5) (or (5) if required). One word of caution; the standard linear model occurs as the special case in which $\theta_1 = \theta_2 = \ldots = \theta_q$. A moment's reflection reveals an interesting obstacle to the application of standard statistical inference. The $\gamma_j$ parameters are not identified under the null hypothesis, constituting another example of what was discussed on section 4.3. Nonstandard tools have to be used to test this hypothesis.

**6.2.2 An Example**

Chaotic behavior is interesting because complex nonlinear dynamics that appear to be random and unpredictable can be generated from systems that are deterministic. Moreover, the specific pattern followed by a particular realizations depends crucially on the initial conditions.\(^9\)

For example, consider the process known as logistic map and depicted on the first panel of figure 5. The power of ANN models can be demonstrated by comparing the fits of a linear and a nonlinear approximation fitted by a single hidden layer feedforward network to a time series generated with the logistic map.

The second panel of figure 5 shows that while the linear approximation behaves rather poorly (as expected), even a small ANN (with one hidden layer and 3 hidden units) can fit the relationship almost perfectly.

---

\(^8\)That is, $\bar{z}$ and $\sigma(z)$ are estimates of the mean and standard deviation of $z$.

\(^9\)See Nychka et al (1990) for further details on chaos and statistics.
Figure 5: Logistic Map: \( y_t = 3.8y_{t-1} (1 - y_{t-1}) \)

References


A Workout Problems

1. Which of the following regression models can be transformed into linear models and which cannot?

   a. \[ y_t = \beta_1 \left[ \beta_2 x_t^{-\beta_3} + (1 - \beta_2) z_t^{-\beta_3} \right]^{-\beta_4/\beta_3} \exp(u_t) \]

   b. \[ y_t = \beta_1 + \beta_2 \left[ x_t^{\beta_3 - 1} \right] + u_t \]

   c. \[ y_t = \beta_1 x_t^{\beta_2} + u_t \]

   d. \[ y_t = \beta_1 x_t^{\beta_2} \exp(u_t) \]

   e. \[ y_t = \left[ 1 + \exp(\beta' x_t + u_t) \right]^{-1} \]

2. Prove that \( \beta_1 \) derived in (4) is the solution to \( \min S_T(\beta) \) in (3).

3. Write a program that determines your machine’s epsilon.

4. Write a program that determines your machine’s infinity.

5. Using (4), derive (5).

6. Derive (10).