STAT 153 & 248 - Time Series
Lecture Twenty Four

Fall 2025, UC Berkeley

Aditya Guntuboyina

November 25, 2025

The last topic in this course is Recurrent Neural Networks (RNNs). We will fit these models using the Python library `PyTorch`. Before discussing the RNN models, let us first take a high-level look at how model fitting works in `PyTorch`.

1 Model Fitting in `PyTorch`

Model fitting in `PyTorch` is usually based on the following steps:

1. Create model and define the parameters (which need to be estimated based on the data)
2. Define the loss function
3. Specify initial values of the parameters
4. Use of an optimization algorithm (some variant of gradient descent such as `Adam`). This algorithm has two steps:
   a) Compute gradient of the loss function with respect to the parameters (`PyTorch` performs gradient calculations using reverse-mode automatic differentiation via `backward`)
   b) Update parameters based on the gradient. The update rule depends on the specific optimization algorithm being used and requires choosing a tuning parameter known as the **learning rate**. If the learning rate is too small, convergence will be very slow; if it is too large, the algorithm may oscillate or fail to converge.

As stepping stones for RNNs, let us first review some more basic models that we already studied in the course.

2 Regression with t as covariate

The simplest and the first model that we studied was the linear regression model:

$$y_t = \beta_0 + \beta_1 t + \epsilon_t \qquad \text{with } \epsilon_t \overset{\text{i.i.d}}{\sim} N(0, \sigma^2). \tag{1}$$

We then looked at nonlinear regression. One way to make the right hand side of (1) nonlinear in t is to introduce terms involving $(t-c)_+$ for certain knots c:

$$y_t = \beta_0 + \beta_1 t + \beta_2 (t-c_1)_+ + \cdots + \beta_{k+1}(t-c_k)_+ + \epsilon_t. \tag{2}$$

Here $(t-c)_+$ is the positive part function applied to $t-c_1$. We shall also use the notation ReLU and $\sigma(\cdot)$ to denote this function (please do not confuse the function $\sigma(\cdot)$ with the standard deviation σ of ϵ_t; we shall use the same notation for both but they can be easily distinguished from the context):

$$\sigma(u) = \text{ReLU}(u) = u_+ := \max(u, 0).$$

The unknown parameters in (2) are $\beta_0, \ldots, \beta_{k+1}, c_1, \ldots, c_k$ and σ.

The model (2) is also a linear model but it is linear in the modified variables $1, t, (t-c_1)_+, \ldots, (t-c_k)_+$ (and nonlinear in the original variable t). The vector of these modified variables:

$$(1, t, (t-c_1)_+, \ldots, (t-c_k)_+)^T$$

can be called the feature vector. The model is a linear function of the feature vectors.

We now rewrite the model (2) in a slightly different form. The time t represents the covariate x_t here, so we write $x_t = t$. We shall remove the term t as it is covered by $t = (t-c)_+$ for $c = 0$ (note that $1 \le t \le n$). We also write μ_t for the mean of y_t. We shall also use r_t to denote the feature vector:

$$r_t = (\sigma(x_t - c_1), \ldots, \sigma(x_t - c_k))^T$$

and s_t to denote:

$$s_t = (x_t - c_1, \ldots, x_t - c_k)^T.$$

With these changes, the model (2) becomes:

$$\begin{aligned} x_t &= t \\ s_t &= (x_t - c_1, \ldots, x_t - c_k)^T \\ r_t &= \sigma(s_t) \\ \mu_t &= \beta_0 + \beta^T r_t \\ y_t &= \mu_t + \epsilon_t. \end{aligned} \tag{3}$$

In words, the univariate covariate x_t (which is simply t) is first converted to the $k \times 1$ vector s_t in a linear fastion. Then the nonlinear function $\sigma(\cdot)$ is applied to s_t (here $\sigma(\cdot)$ is applied separately to each coordinate of s_t) to generate the feature vector r_t. Then μ_t is a linear function of r_t which serves as the mean to y_t.

3 AutoRegression

We also studied autoregression models where the covariates are simply lagged values of y_t. The simplest of these models is AR(1) where $x_t = y_{t-1}$. This is simply (1) with t replaced by $x_t = y_{t-1}$:

$$y_t = \beta_0 + \beta_1 x_t + \epsilon_t \qquad \text{with } \epsilon_t \overset{\text{i.i.d}}{\sim} N(0, \sigma^2).$$

One can create a nonlinear version of AR(1) by simply using (3) with $x_t = y_{t-1}$. We shall refer to this as Nonlinear AutoRegression of order 1: NAR(1) (there are many nonlinear autoregression models and this one is only one of them):

$$\begin{aligned} x_t &= y_{t-1} \\ s_t &= (x_t - c_1, \ldots, x_t - c_k)^T \\ r_t &= \sigma(s_t) \\ \mu_t &= \beta_0 + \beta^T r_t \\ y_t &= \mu_t + \epsilon_t. \end{aligned} \tag{4}$$

Now let us consider the case of AR(p) for $p \geq 1$. The usual AR(p) model is simply:

$$\begin{aligned} x_t &= (y_{t-1}, \ldots, y_{t-p})^T \\ \mu_t &= \beta_0 + \beta^T x_t \\ y_t &= \mu_t + \epsilon_t. \end{aligned} \tag{5}$$

Observe that (5) can be written in compressed form as simply $y_t = \beta_0 + \beta_1 y_{t-1} + \cdots + \beta_p y_{t-p} + \epsilon_t$ which is the usual form of AR(p).

What is a natural nonlinear version of (5)? Put another way, what is a good extension of (4) for $p \geq 1$? There are multiple ways of obtaining these versions. Looking at the structure of (4), clearly $x_t = y_{t-1}$ will be replaced by $x_t = (y_{t-1}, \ldots, y_{t-p})^T$. The next line gives the formula for s_t. This would need to be changed because x_t is no longer a scalar. One way to do this would be to write one version of the formula for s_t in (4) for each component of x_t. This would result in:

$$\begin{aligned} x_t &= (y_{t-1}, \ldots, y_{t-p})^T \\ s_t &= (x_{t1} - c_1^{(1)}, \ldots, x_{t1} - c_k^{(1)}, x_{t2} - c_1^{(2)}, \ldots, x_{t2} - c_k^{(2)}, \ldots, x_{tp} - c_1^{(p)}, \ldots, x_{tp} - c_k^{(p)})^T \\ r_t &= \sigma(s_t) \\ \mu_t &= \beta_0 + \beta^T r_t \\ y_t &= \mu_t + \epsilon_t. \end{aligned} \tag{6}$$

Here $x_{t1} = y_{t-1}, \ldots, x_{tp} = y_{t-p}$ denote the components of x_t. With this choice of s_t, note that μ_t becomes

$$\mu_t = \beta_0 + \beta^T r_t = \beta_0 + \beta^T \sigma(s_t) = \beta_0 + \sum_{j=1}^{p} g_j(x_{tj}) \qquad \text{where } g_j(x) := \sum_{i=1}^{k} \beta_{i,j} \sigma(x_{tj} - c_j^{(i)}).$$

In other words, we are fitting an **additive** model for y_t in terms of the covariates $x_{t1} = y_{t-1}, \ldots, x_{tp} = y_{t-p}$. Additive models are popular in regression but they do not incorporate any interactions between the covariates. For example, if the true model generating the data is $y_t = 0.5 y_{t-1} y_{t-2} + \epsilon_t$, the additive model is unlikely to work well (because $(x_1, x_2) \mapsto 0.5 x_1 x_2$ is not an additive function of x_1 and x_2).

Instead of using the additive model in (6), we shall use the following model as NAR(p) (Nonlinear AutoRegression of order p). This is obtained by changing the second line of (6) to be an arbitrary linear function of x_t:

$$\begin{aligned} x_t &= (y_{t-1}, \ldots, y_{t-p})^T \\ s_t &= W x_t + b \\ r_t &= \sigma(s_t) \\ \mu_t &= \beta_0 + \beta^T r_t \\ y_t &= \mu_t + \epsilon_t. \end{aligned} \tag{7}$$

Here W is a $k \times p$ matrix and b is a $k \times 1$ vector. The parameters in this model are the entries of the matrix W, the vector b, the coefficients β_0 and the components of β and finally the noise standard deviation σ.

In neural network terminology, the model (7) is called a **single-hidden layer neural network** because it first applies a linear transformation to the input x_t (via $s_t = Wx_t + b$), then passes the result through the nonlinear activation function σ to get r_t, which forms the hidden layer. The output μ_t is then computed as a linear function of r_t (via $\mu_t = \beta_0 + \beta^T r_t$) and noise ϵ_t is added to explain the discrepancy between y_t and μ_t. The presence of one nonlinear transformation between the input x_t and the output μ_t, combined with otherwise linear operations, is exactly the structure of a single-hidden layer neural network.

To sum up, we take the single-hidden layer neural network model (7) to be our nonlinear generalization of AR(p).

Note that (7) can also be treated as a linear regression model but the linearity is in terms of the feature vector r_t (not in terms of the original covariate x_t). We shall refer to r_t as the feature vector at time t, it is also common to refer to it as the hidden layer output at time t.