Lecture Twenty Four

Recurrent Neural Networks (RNN)

PyTorch ←

PyTorch

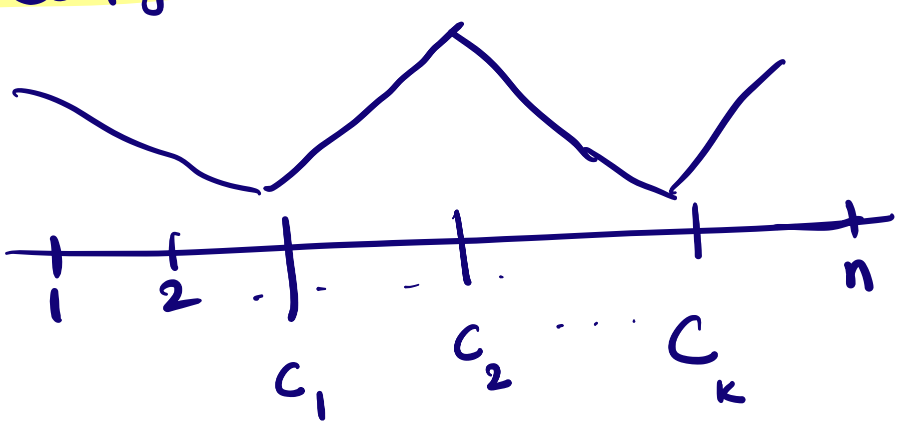
Model :
$$\boxed{y_t = \beta_0 + \beta_1 t + \beta_2 (t-c_1)_+ + \beta_3 (t-c_2)_+ + \dots + \beta_{k+1}(t-c_k)_+ + \varepsilon_t}$$

$k=6$

$$\left.\begin{matrix} c_1 \dots c_k \\ \beta_0, \beta_1, \beta_2 \dots \beta_{k+1} \end{matrix}\right\}$$

loss function: $\displaystyle\sum_{t=1}^{n} \left[y_t - \beta_0 - \beta_1 t - \beta_2 (t - \boxed{c_1})_+ - \dots - \beta_{k+1}(t-\boxed{c_k})_+ \right]^2$

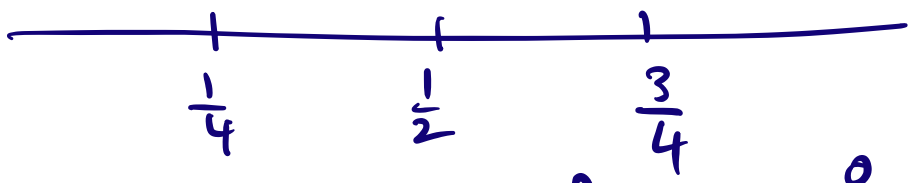
1. Create model & declare parameters
2. Define the loss function
3. Specify initial values.



$c_1 \dots c_k$: k - equally spaced points in $1 \dots n$

$(K=3)$

$(K=3)$



After fixing $c_1^0, c_2^0, \ldots, c_K^0$, solve least squares to get $\beta_0^0, \beta_1^0 \ldots \beta_{K+1}^0$.

(4) Uses some kind of gradient descent (Adam)

At each step of the algorithm:

(a) Compute gradient of the loss at the current parameters. $\longrightarrow$ BACKWARD DIFFERENTIATION.

$\rightarrow$ (b) Update parameters using the gradient

This step uses a tuning parameter called the learning rate.

MA(1) using PyTorch

$$y_t = \mu + \varepsilon_t + \theta\, \varepsilon_{t-1}, \quad t = 1 \ldots n.$$

$(\varepsilon_0 = 0)$

$f(y_1 \ldots y_n)$

$y_1 \ldots y_n$

$\boxed{\text{Fix } \varepsilon_0 = 0}$

$t=1$ $\quad y_1 = \mu + \varepsilon_1 + \theta \underbrace{\varepsilon_0} = \mu + \varepsilon_1$

$$\varepsilon_1 = (y_1 - \mu)$$

$t=2$ $\quad y_2 = \mu + \varepsilon_2 + \theta \underbrace{\varepsilon_1}$

$$\varepsilon_2 = y_2 - \mu - \theta \varepsilon_1$$
$$= y_2 - \mu - \theta (y_1 - \mu)$$

$$\varepsilon_3 = y_3 - \mu - \theta \varepsilon_2$$
$$= y_3 - \mu - \theta \left[y_2 - \mu - \theta (y_1 - \mu) \right]$$

Write every ε_t in terms of y_t & μ, θ

$$\prod_{t=1}^{n} \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left[-\frac{\varepsilon_t^2}{2\sigma^2} \right]$$

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

y_t on $\boxed{x_t = t}$

$$\boxed{y_t = \beta_0 + \beta_1 x_t + \beta_2 (x_t - c_1)_+ + \cdots + \beta_{k+1} (x_t - c_k)_+ + \varepsilon_t}$$

Auto Regression

AR(1): $y_t = \beta_0 + \beta_1 \underbrace{\boxed{y_{t-1}}}_{x_t} + \varepsilon_t$

$$y_t = \boxed{\beta_0 + \beta_1 x_t} + \varepsilon_t, \quad x_t = y_{t-1}$$

NAR(1) (Nonlinear Auto Regression)

$$y_t = \beta_0 + \beta_1 x_t + \beta_2 (x_t - c_1)_+ + \beta_3 (x_t - c_2)_+ + \cdots + \beta_{k+1} (x_t - c_k)_+ + \varepsilon_t$$

$$\boxed{x_t = y_{t-1}}$$

$\boxed{AR(1)}$: $y_t = \beta_0 + \beta_1 x_t + \varepsilon_t$

$x_t = y_{t-1}$

NAR(1): $y_t = \beta_0 + \boxed{\beta_1 x_t} + \beta_2 (x_t - c_1)_+ + \cdots + \beta_{k+1} (x_t - c_k)_+ + \varepsilon_t$

$x_t = y_{t-1}$

We shall rewrite this model using slightly different notation. We shall drop $\beta_1 x_t$.

$$y_t = \beta_0 + \beta_1 (x_t - c_1)_+ + \beta_2 (x_t - c_2)_+ + \cdots + \beta_k (x_t - c_k)_+ + \varepsilon_t$$

$$\underset{k \times 1}{s_t} = \begin{pmatrix} x_t - c_1 \\ x_t - c_2 \\ \vdots \\ x_t - c_k \end{pmatrix} \rightarrow \text{linear function of } x_t$$

$$r_t = \sigma(s_t)$$

$$\sigma(u) = \max(u, 0) = u_+ = \mathrm{ReLU}(u)$$

$k \times 1$

Feature Vector

$$\sigma\begin{pmatrix} u_1 \\ \vdots \\ u_k \end{pmatrix} = \begin{pmatrix} \sigma(u_1) \\ \sigma(u_2) \\ \vdots \\ \sigma(u_k) \end{pmatrix}$$

$$\mu_t = \beta_0 + \beta^T r_t$$

Loss : $\sum_{t=1}^{n} (y_t - \mu_t)^2$

NAR(1) :

$$x_t = y_{t-1}$$

Model:

$$s_t = \begin{pmatrix} x_t - c_1 \\ \vdots \\ x_t - c_k \end{pmatrix}$$

$$\boxed{r_t = \sigma(s_t)}$$

$$\mu_t = \beta_0 + \beta^T r_t$$

$$\boxed{y_t = \mu_t + \varepsilon_t}$$

loss $= \sum (y_t - \mu_t)^2$

AR(p)

$$x_t = (y_{t-1}, \ldots, y_{t-p})^T$$

$$\mu_t = \beta_0 + \beta^T x_t$$

NAR(p) We will only change the s_t equation.

previously $s_t = \begin{pmatrix} x_t - c_1 \\ \vdots \\ x_t - c_k \end{pmatrix}$

$x_{t1}, x_{t2} \ldots x_{tp}$

$$s_t = \begin{bmatrix} x_{t1} - c_1^1 \\ x_{t1} - c_2^1 \\ \vdots \\ x_{t1} - c_k^1 \\ x_{t2} - c_1^2 \\ x_{t2} - c_2^2 \\ \vdots \\ x_{t2} - c_k^2 \\ \vdots \\ x_{tp} - c_1^p \\ x_{tp} - c_2^p \\ \vdots \\ x_{tp} - c_k^p \end{bmatrix}$$

$$\boxed{s_t = \underset{k \times p}{W}\, \underset{p \times 1}{x_t} + \underset{k \times 1}{b}}$$

NAR(p)

$x_t = (y_{t-1}, \ldots y_{t-p})^T$

$s_t = W x_t + b$

$r_t = \sigma(s_t)$

$\mu_t = \beta_0 + \beta^T r_t$

$$x_t = (y_{t-1}, \dots, y_{t-p})^T$$

$$r_t = \sigma(W x_t + b) \longrightarrow \text{Feature Vector}$$

$$\mu_t = \beta_0 + \beta^T r_t$$

Single Hidden Layer Neural Network.