Suppose I have a sequential data, $X = \{x_1, x_2, x_3, \dots, x_n\}$, how can program a code to learn the pattern in the data?

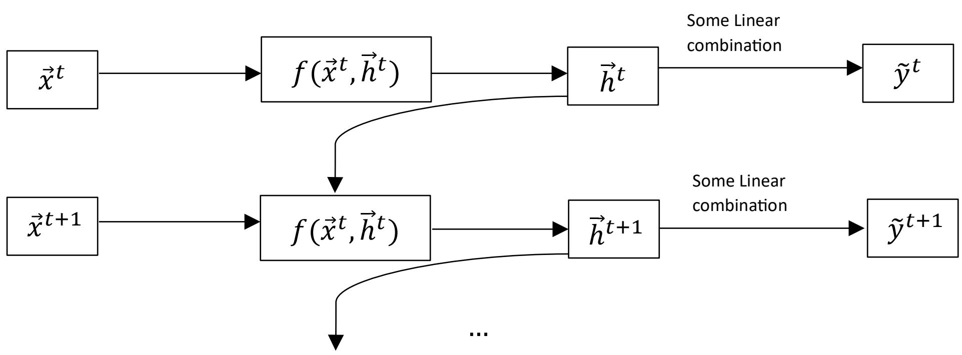
We can use a Timestep of the Data, of length T, to predict the next K values in the dataset

$$use\ \vec{x}^t = \begin{bmatrix} x_t \\ x_{t+1} \\ \dots \\ x_{t+T-1} \end{bmatrix}\ to\ predict\ \tilde{y}^t = \begin{bmatrix} \tilde{x}_{t+T} \\ \tilde{x}_{t+T+1} \\ \dots \\ \tilde{x}_{t+T+k-1} \end{bmatrix}$$

There should also be a “memory element” to the equation, such that the previous data in the sequence affects the future predictions. Let us call that the hidden vector $\vec{h}^t$.

In some way, this “memory element” should also store some information about the prediction $\tilde{y}^t$. So we can compute $\tilde{y}^t$ as some linear combination of $\vec{h}^t$.

Using an arbitrary activation function $f()$, we can construct a simple flow as follows:



With this flow in mind, we can construct the recursive equation for the hidden vector $\vec{h}^t$ and the prediction $\tilde{y}^t$ as such:

$$\vec{h}^t = f\left(\overline{W}\vec{h}^{t-1} + \overline{U}\vec{x}^t\right), \qquad \tilde{y} = \overline{V}\vec{h}^t$$

Where we define the following parameters and vectors

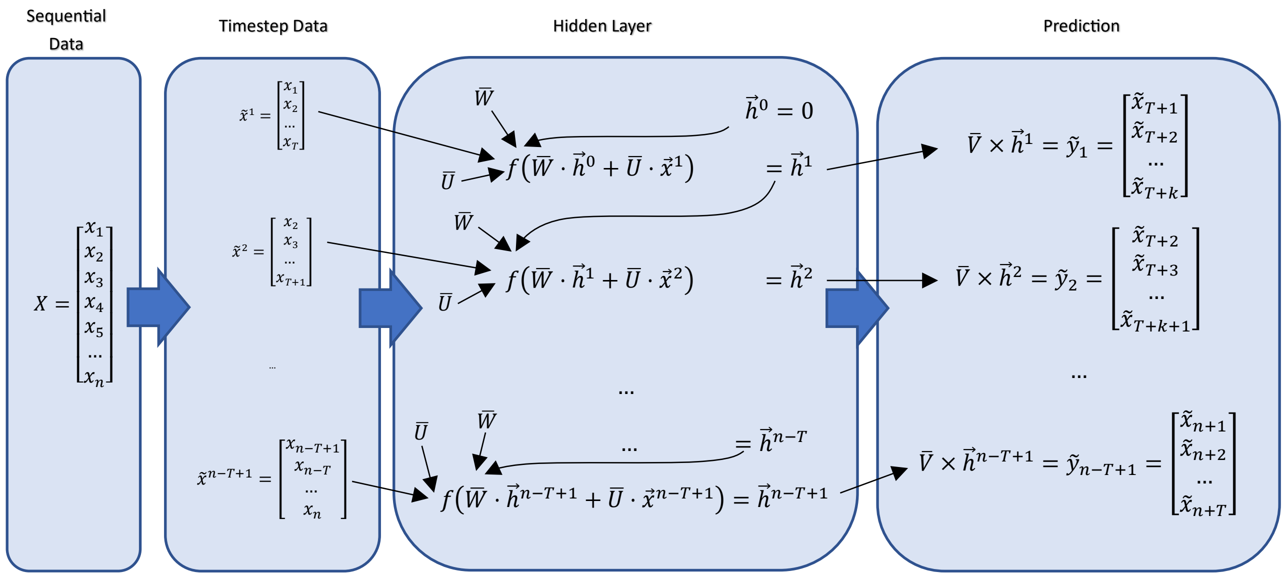
- $\overline{W}_{J\times J} \Rightarrow Weighing\ Parameter\ for\ the\ previous\ hidden\ vector\ \vec{h}^{t-1}$
- $\overline{U}_{J\times T} \Rightarrow Weighing\ Patameter\ for\ the\ timestep\ data\ \vec{x}^t$
- $\overline{V}_{K\times J} \Rightarrow Weighing\ Paramter\ for\ the\ hidden\ vector\ \vec{h}^t\ to\ compute\ the\ prediction\ \tilde{y}^t$
- $\vec{x}^t \epsilon\ R^T \Rightarrow Timestep\ Data$
- $\vec{h}^t \epsilon R^J \Rightarrow Hidden\ Vector$
- $\tilde{y}^t \epsilon R^K \Rightarrow Prediction$

Dimensional Parameters

- $T \Rightarrow Length\ of\ Timestep$
- $K \Rightarrow Ouput\ Dimension$
- $J \Rightarrow Dimension\ of\ hidden\ vector$

Activation function (use sigma function)

$$f(x) = \sigma(x) = \frac{1}{1 + \exp(-x)}$$



Based on the prediction $\tilde{y}^t$, we can compute the loss function from the actual values of $\vec{y}_i = \begin{bmatrix} x_{t+T} \\ x_{t+T+2} \\ \cdots \\ x_{t+T+k-1} \end{bmatrix}$ as follows:

$$L = \frac{1}{2} \sum_{t=1}^{n-T+1} (\tilde{y}_t - y_t)^2$$

In finding the partial derivatives, it is worth noting the derivative of the sigma function can be simplifies as:

$$f'(x) = \sigma'(x) = \sigma(x)\big(1 - \sigma(x)\big) = \vec{h}^t\big(1 - \vec{h}^t\big)$$

From the loss function, we can thus compute the following partial derivatives of the weighting parameters, $\frac{\partial L}{\partial \bar{W}}, \frac{\partial L}{\partial \bar{U}}, \frac{\partial L}{\partial \bar{V}}$:

$$\frac{dL_t}{dV_{\alpha\beta}} = \frac{\partial L_t}{\partial \tilde{y}_t} \frac{\partial \tilde{y}_t}{\partial V_{\alpha\beta}} = (\tilde{y}_t - y_t) h_k$$

$$\frac{dL}{d\bar{V}} = \sum_{t=1}^{n-T+1} (\tilde{y}_t - y_t) h_k$$

$$\frac{dL_i}{dU_{\alpha\beta}} = \frac{\partial L_i}{\partial \tilde{y}_j}\frac{\partial \tilde{y}_j}{\partial h_k}\frac{\partial h_k}{\partial U_{\alpha\beta}}$$

$$= (\tilde{y}_i - y_i)(V_{ij})\left(f'(\bar{W}\vec{h}^{t-1} + \bar{U}\vec{x}^t)\right)(\vec{x}^i)$$

$$= (\tilde{y}_i - y_i)(V_{ij})\left(\vec{h}^t(1-\vec{h}^t)\right)(\vec{x}^i)$$

$$\frac{dL}{dU} = \sum_{i=1}^{n-T+1}(\tilde{y}_i - y_i)(V_{ij})\left(\vec{h}^t(1-\vec{h}^t)\right)(\vec{x}^i)$$

$$\frac{dL_i}{dW_{\alpha\beta}} = \frac{\partial L_i}{\partial \tilde{y}_j}\frac{\partial \tilde{y}_j}{\partial h_k}\frac{\partial h_k}{\partial W_{\alpha\beta}}$$

$$= (\tilde{y}_i - y_i)(V_{ij})\left(f'(\bar{W}\vec{h}^{t-1} + \bar{U}\vec{x}^t)\right)(\vec{h}^{i-1})$$

$$= (\tilde{y}_i - y_i)(V_{ij})\left(\vec{h}^t(1-\vec{h}^t)\right)(\vec{h}^{i-1})$$

$$\frac{dL}{dW} = \sum_{i=1}^{n-T+1}(\tilde{y}_i - y_i)(V_{ij})\left(\vec{h}^t(1-\vec{h}^t)\right)(\vec{h}^{i-1})$$

Gradient Descent

$$W_{\alpha\beta} \to W_{\alpha\beta} - \varepsilon\frac{dL}{dW_{\alpha\beta}}$$

$$U_{\alpha\beta} \to U_{\alpha\beta} - \varepsilon\frac{dL}{dU_{\alpha\beta}}$$

$$V_{\alpha\beta} \to V_{\alpha\beta} - \varepsilon\frac{dL}{dV_{\alpha\beta}}$$

Algorithm

Initialize Parameters

- $\bar{W} = 0, \bar{V} = 0, \bar{U} = 0$
- $\vec{h}^0 = 0$

Iterate for N epochs

For every n-T+1 combination of $\vec{x}^t = \begin{bmatrix} x_t \\ x_{t+1} \\ \dots \\ x_{t+T} \end{bmatrix}$,

Find hidden vector from previous hidden vector

$\vec{h}^t = f(\bar{W}\vec{h}^{t-1} + \bar{U}\vec{x}^t)$

Compute predictions $\tilde{y}_t = \bar{V}\vec{h}^t = \bar{V}f(\bar{W}\vec{h}^{t-1} + \bar{U}\vec{x}^t)\ \forall t = 1, 2, \dots n - T + 1$

Compute Partial Derivatives and update Parameters

$$\frac{dL}{d\bar{V}} = \sum_{i=1}^{n-T+1} (\tilde{y}_i - y_i) h_k$$

$$\bar{V} \to \bar{V} - \varepsilon \frac{dL}{d\bar{V}}$$

$$\frac{dL}{d\bar{U}} = \sum_{i=1}^{n-T+1} (\tilde{y}_t - y_t)(V_{ij})\left(\vec{h}^t(1 - \vec{h}^t)\right)(\vec{x}^t)$$

$$\bar{U} \to \bar{U} - \varepsilon \frac{dL}{d\bar{U}}$$

$$\frac{dL}{d\bar{W}} = \sum_{i=1}^{n-T+1} (\tilde{y}_i - y_i)(V_{ij})\left(\vec{h}^t(1 - \vec{h}^t)\right)(\vec{h}^{i-1})$$

$$\bar{W} \to \bar{W} - \varepsilon \frac{dL}{d\bar{W}}$$