INT301 W10

TIME-SERIES PREDICTION AND ELMAN NETWORK

What is a time series?

- Time Series is a series of timely ordered, comparable observations y_t recorded in equidistant time intervals

Time Series Models

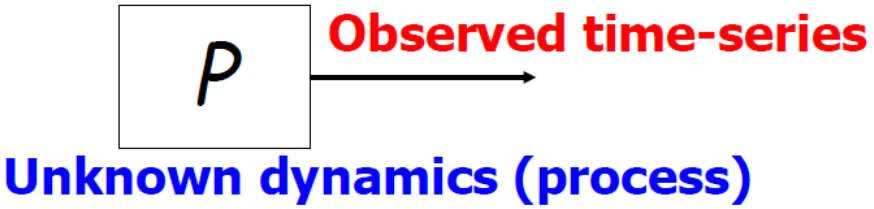
Time series 的建模:

- 连续时间索引中时间序列的值通常是相关的。否则，预测是不可能的
- 统计属性不随时间而变化
  - WSS (wide-sense stationarity) random process
    - 均值和自协方差 (autocovariance) 不随时间变化
    - 方差对所有时间都是有限的

- In statistics and machine learning, a time-series is often described by a sequence of vectors (or scalars) which depend on time t:

$$\{x(t_0), x(t_1), \cdots, x(t_{i-1}), x(t_i), x(t_{i+1}), \cdots\}$$

It's the output of some process P that we are interested in:



- 在离散点处对连续信号 x(t) 进行采样，得到一个 series
- 进行均匀采样中，如果采样周期为：Δt

$$\{x[t]\} = \{x(0), x(\Delta t), x(2\Delta t), x(3\Delta t), \cdots\}$$

Time Series Prediction

向前延伸时间 t，我们有时间序列 {x[t]， x[t -1]，⋯}。由此，我们想估计 x 在未来某个时间的值：

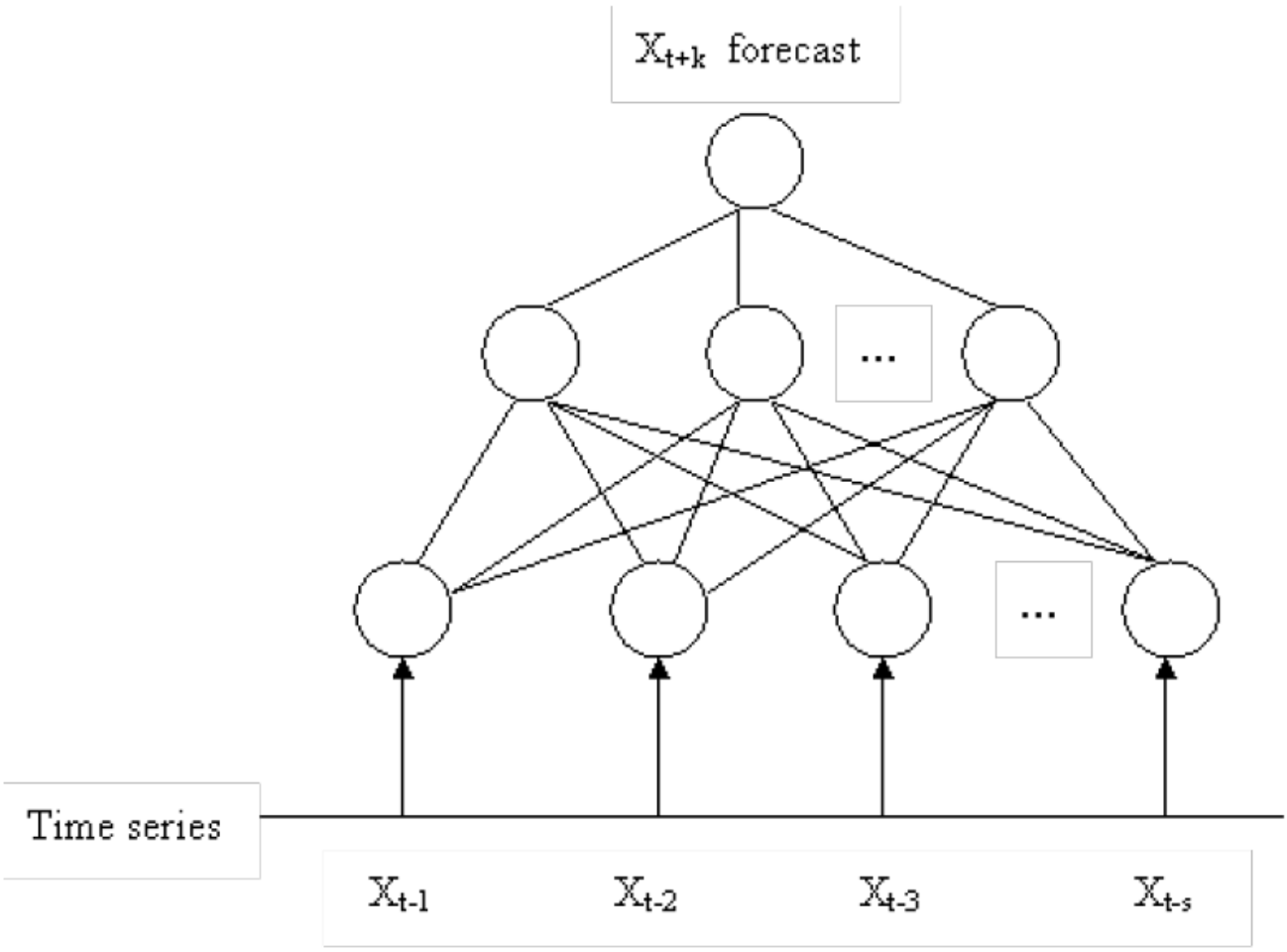
$$\hat{x}[t+s] = f(x[t], x[t-1], \cdots)$$

s is called the **horizon of prediction**.

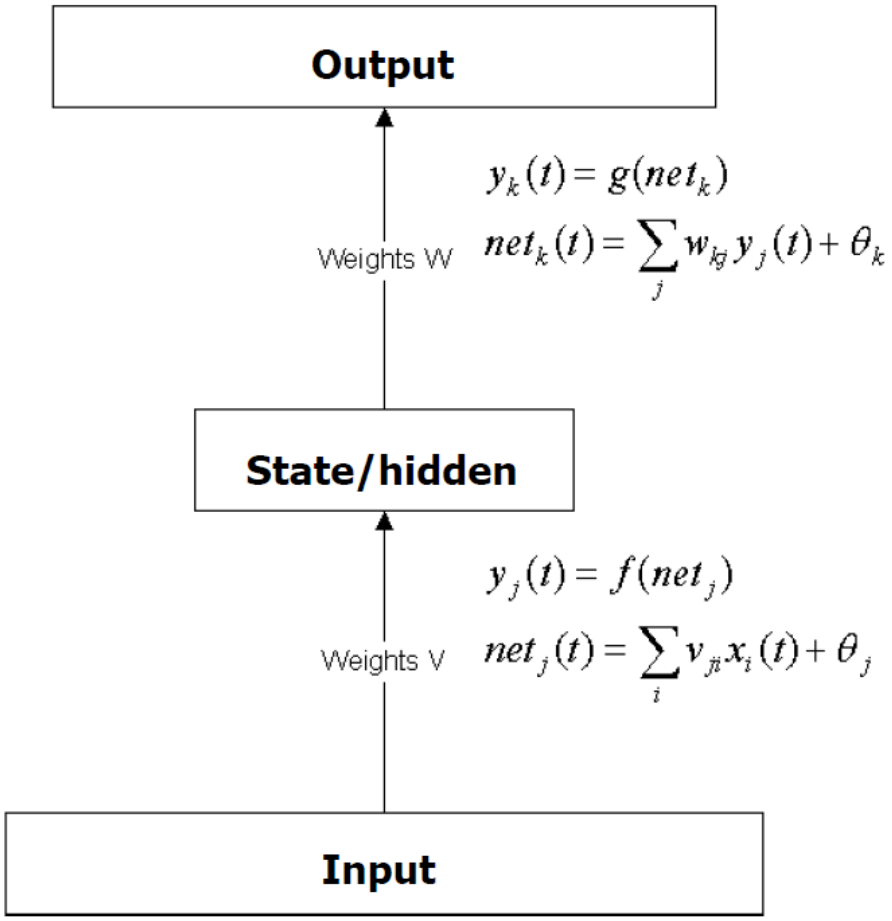
神经网络已被广泛用作时间序列预测器：大多数情况下，这些是前馈网络，它们在 input sequence 上使用 sliding window。

神经网络将 time series $X_1, \ldots, X_n$ 的形式视为输入向量到输出值的许多映射。

Time Series Prediction with ANN



这里和 MLP 一样，网络中有 s 个 input units，就有 s 个 input window ($X_{t-s}, X_{t-s+1}, \ldots, X_t$)；将 time series 中相邻的数据 (归一化到 [0, 1]) 喂入网络进行训练，计算出 error 后用 BP 进行更新；最后进行预测。



有一些不同的是，feed forward NN 的 hidden layer 有 **internal representation**，就是通过某种方法将输入信号 recode，以增加准确率。

Motivation for dynamic networks

对于一些任务，比如：

- **sequential input** (e.g. language understanding, robot exploration, ...)
- **sequential output** (e.g. speech, route planning, ...), or combinations of the above.

我们需要它们的模型对一段时间之前的输入敏感。因此，我们要求一个模型保持 state 或 memory。

MLP 和 RBF 网络都是 static networks，它们学习从 single input 到 single output 的映射。Dynamic networks 学习的是，从 single input 到 sequence of response 的映射。

为了学习 sequences，我们需要将某种形式的记忆（短期记忆，short term memory）包含在网络中。

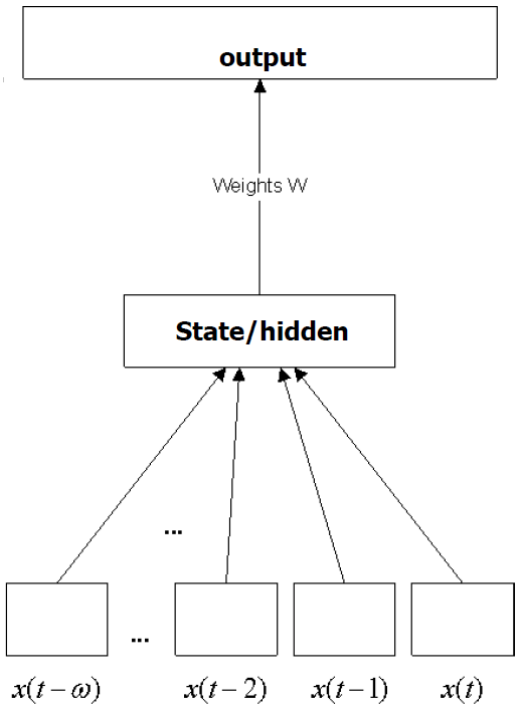
有两种主要的方法引入 memory effects：

- Implicit：时间滞后信号作为静态网络的输入或作为循环连接
- Explicit：Temporal Backpropagation Method

在隐式形式中，我们假设我们收集示例（输入信号，输出序列）的环境是稳定的。

对于显式形式，环境可以是非平稳的，即网络可以跟踪信号结构的变化。

Time Delayed Networks



输入信号可以在外部缓存，并呈现在额外的输入节点（输入库）上。

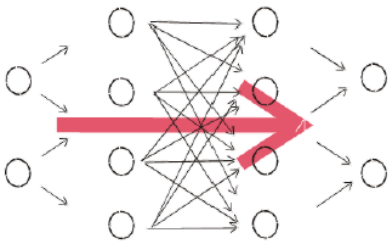
时滞网络说明了序列学习的隐式表示方法，它将静态网络（例如 MLP / RBF）与存储器结构相结合。

Dynamical Neural Networks

Recurrent networks 可以显式地处理按顺序呈现的输入。

循环网络与前馈网络的根本区别在于，它们不仅基于输入运行，还基于状态运行。

The ability of the net to reverberate and sustain activity can serve as a working memory.



- Connections only "from left to right", no connection cycle
- Activation is fed forward from input to output through "hidden layers"
- No memory

- At least one connection cycle
- Activation can "reverberate", persist even with no input
- System with memory
- Mathematically, RNNs implement dynamical systems.

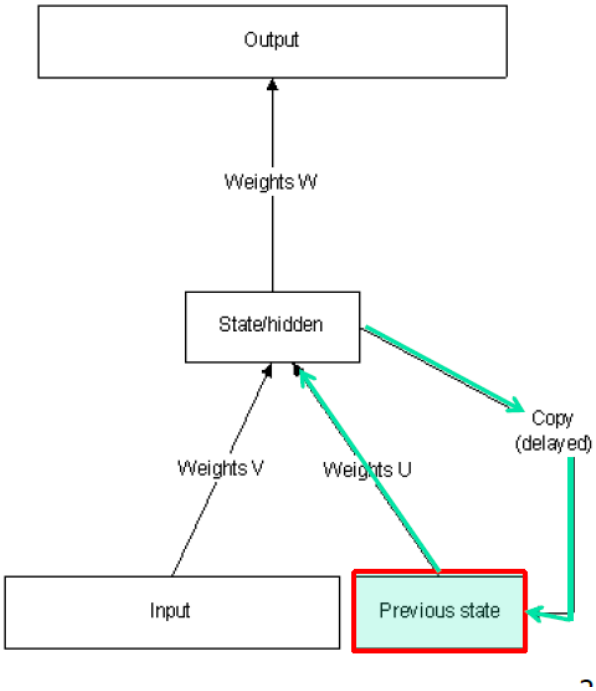
Elman Network

Elman network 是一个简单的循环神经网络，具有强大的预测能力。

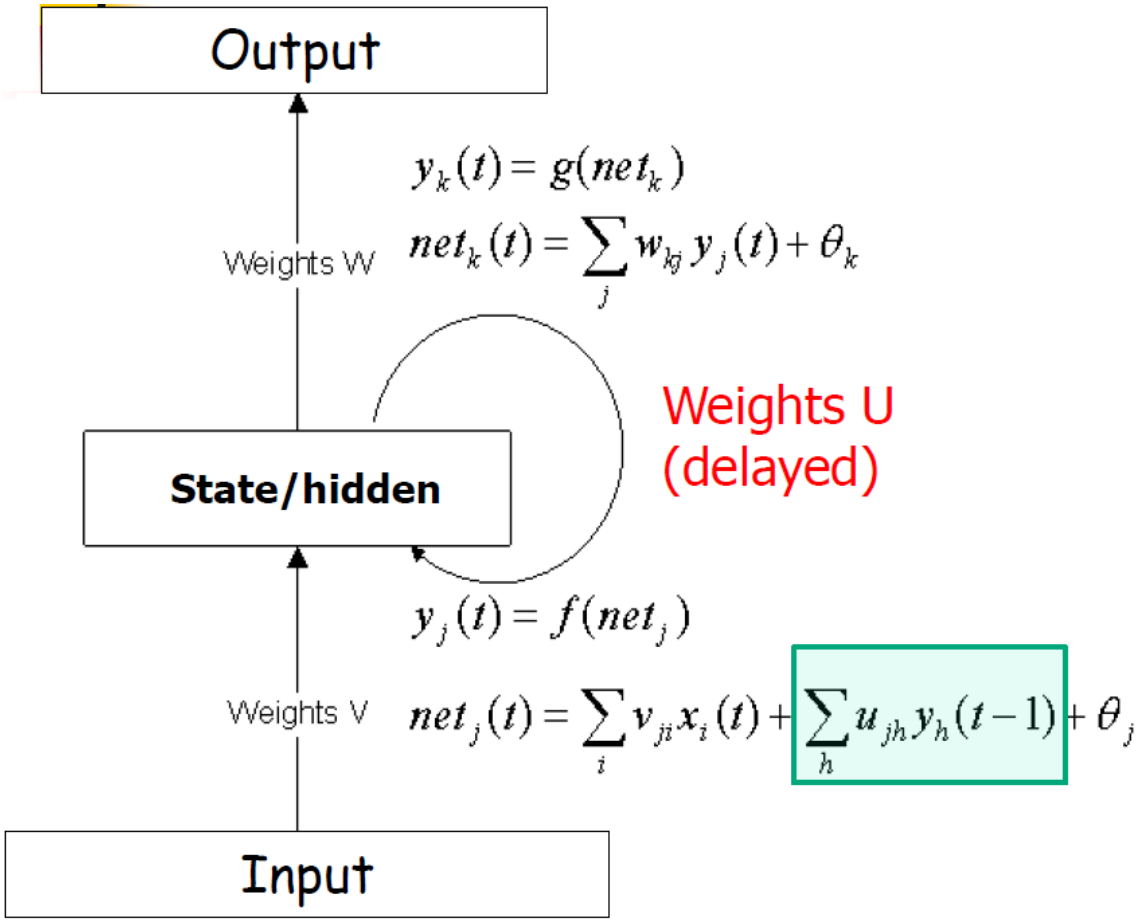
Elman network which typically distinguishes between **a state-output function and an input-state mapping**.

$$y = F_w(z)$$
$$z^t = F_v(z^{t-1}, x)$$

where y is the output, x is the input and z is the state. t is an index in time.

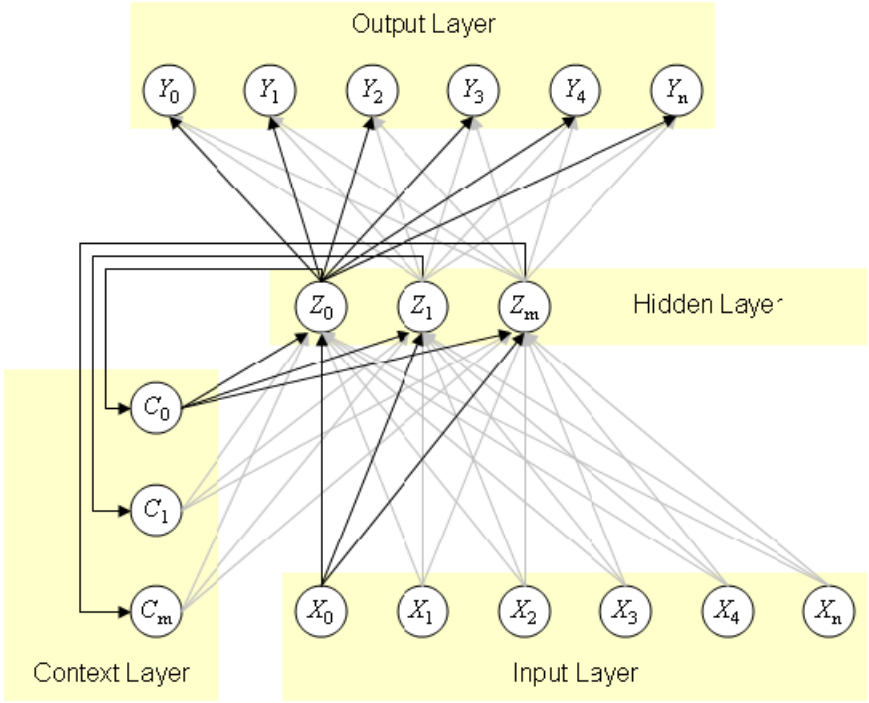


在 Elman net 中，internal representations 现在作为附加输入参与下一步。



这里所有公式都与前馈情况相同。唯一的区别是，我们有另一组权重（U），它将激活从隐藏节点反馈到隐藏节点。重要的是，反馈会延迟。

Four layers of Elman network



- Input layer
- Hidden layer：形成 internal representation
- Context layer：context layer 的神经元的值被用作额外的 input signal 被输入到下一时刻的 hidden layer 中（延迟反馈）。从 hidden layer 到 context layer 的权重被固定为 1。
- Output layer：线性组合

Elman network 采用 BP algorithm：

- The error function：

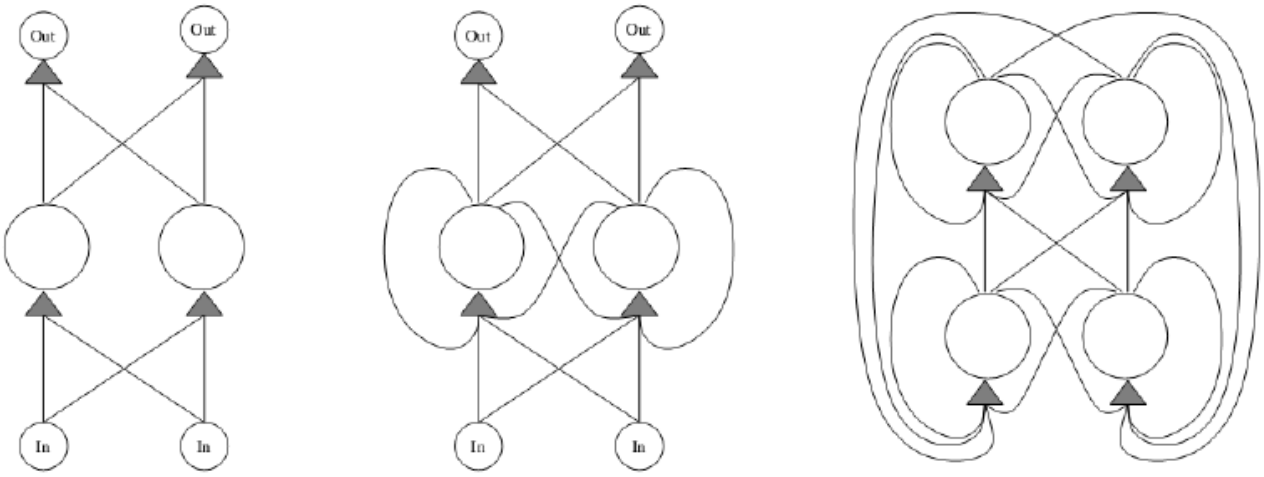
$$E = \sum_{k=1}^{n} [y(k) - d(k)]^2$$

  其中 $d(k)$ 是 expected output (target)

RECURRENT NEURAL NETWORK

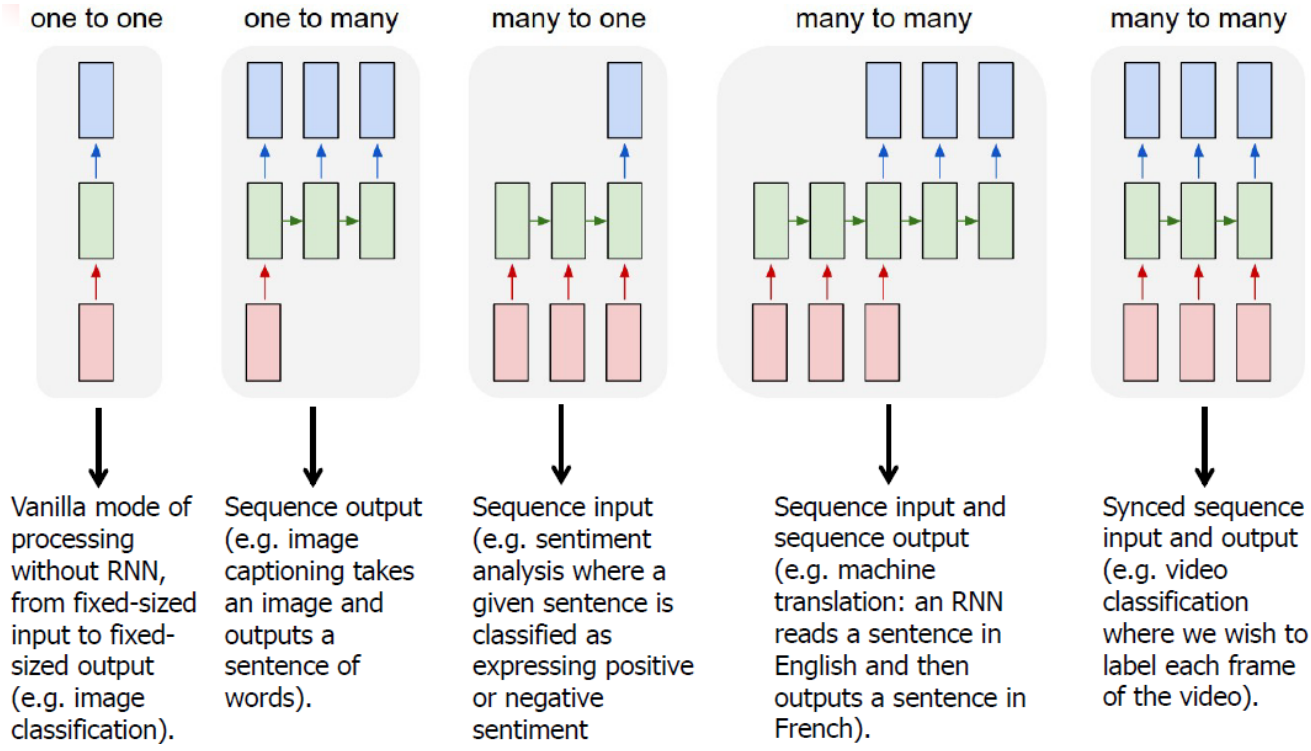
RNN Architecture

RNN 结构和原理与 Elman network 类似。

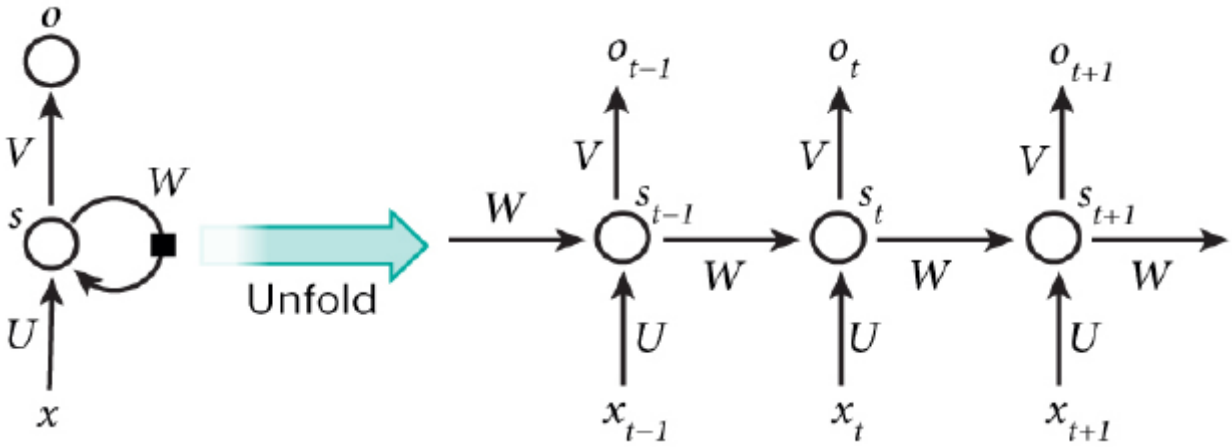


feed forward NN　　a simple RNN　　fully connected RNN

RNN 的部分循环网络中循环仅限于隐藏层；而完全循环网络中，每个节点从所有其他节点获取输入。



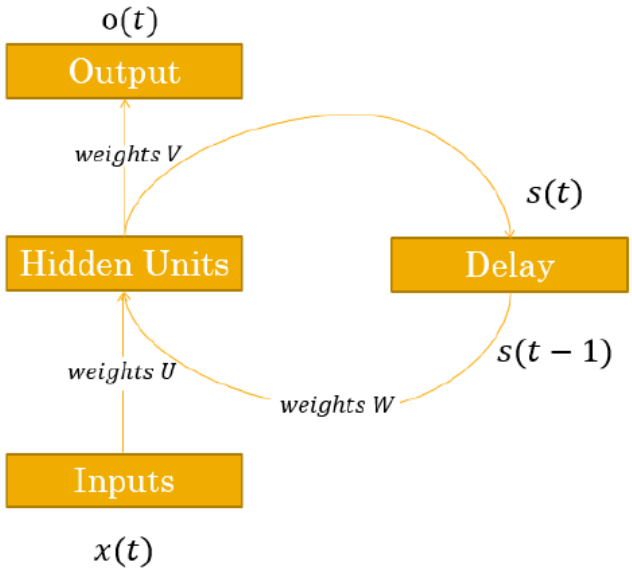
循环网络可以按时间展开 (unfolding) 转换为前馈网络：



例如，如果我们关心的序列是一个包含 5 个单词的句子，则该网络将被展开为一个 5 层神经网络，每个单词一层。

在 RNN 中，error 可以传播超过 2 层，以便捕获更长的历史信息，此过程通常称为 unfolding。在展开的 RNN 中，对于任意数量的时间步长，重复 recurrent weight。

RNN Forward Pass



- The network input at time t:

$$a_h(t) = Ux(t) + Ws(t-1)$$

  注：$Ux(t)$ 是当前时刻输入，$Ws(t-1)$ 是上一时刻的状态，它们被一起输入到 hidden layer

- The activation of hidden unit at time t:

$$s(t) = f_h(a_h(t))$$

  注：f_h 可以看作激活函数

- The hidden to the output at time t:

$$a_o(t) = Vs(t)$$

- The output of the network at time t is:

$$o(t) = f_o(a_o(t))$$

  注：f_o 可以看作激活函数

Loss function

如果网络训练从时间 t0 开始训练 sequences，到 t1 结束，则总损失函数的一个可能选择是，平方误差函数随时间变化的总和：

$$[e(t)]_k = [d(t)]_k - [o(t)]_k$$
$$E(t) = \frac{1}{2} e(t)^T e(t)$$
$$E_{\text{total}} = \sum_{t=t_0}^{t_1} E(t)$$

注：$[d(t)]_k - [o(t)]_k$ 是预测值 - 期望值；$e(t)^T e(t)$ 是 $e(t)$ 的平方。

Back Propagation Through Time (BPTT)

BPTT 学习算法是标准反向传播的扩展，可在展开的网络上执行梯度下降。

每个时间步长都对梯度下降的权重更新具有贡献。error 必须通过时间和网络反向传播。

For recurrent networks, the loss function depends on the activation of the hidden layer through its influence on the output layer and through its influence on the hidden layer at the next step.

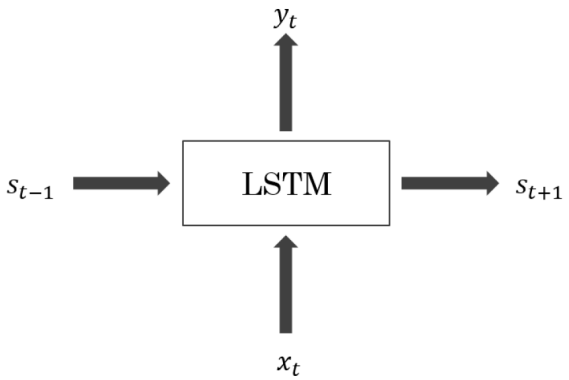
使用 BPTT 训练的 RNN 由于梯度消失问题，难以学习长期依赖关系（long-term dependencies，例如相距甚远的步骤之间的依赖关系）。

当这种情况发生时，我们说相应的神经元饱和 (saturated) 了，即它们的梯度为零，并将前几层中的其他梯度驱动到 0。

来自"远距离"步骤的梯度贡献变为零，并且这些步骤的状态对您正在学习的内容没有贡献。这使得我们只能忘记旧信息：最终没有学习远程依赖关系。

LSTM

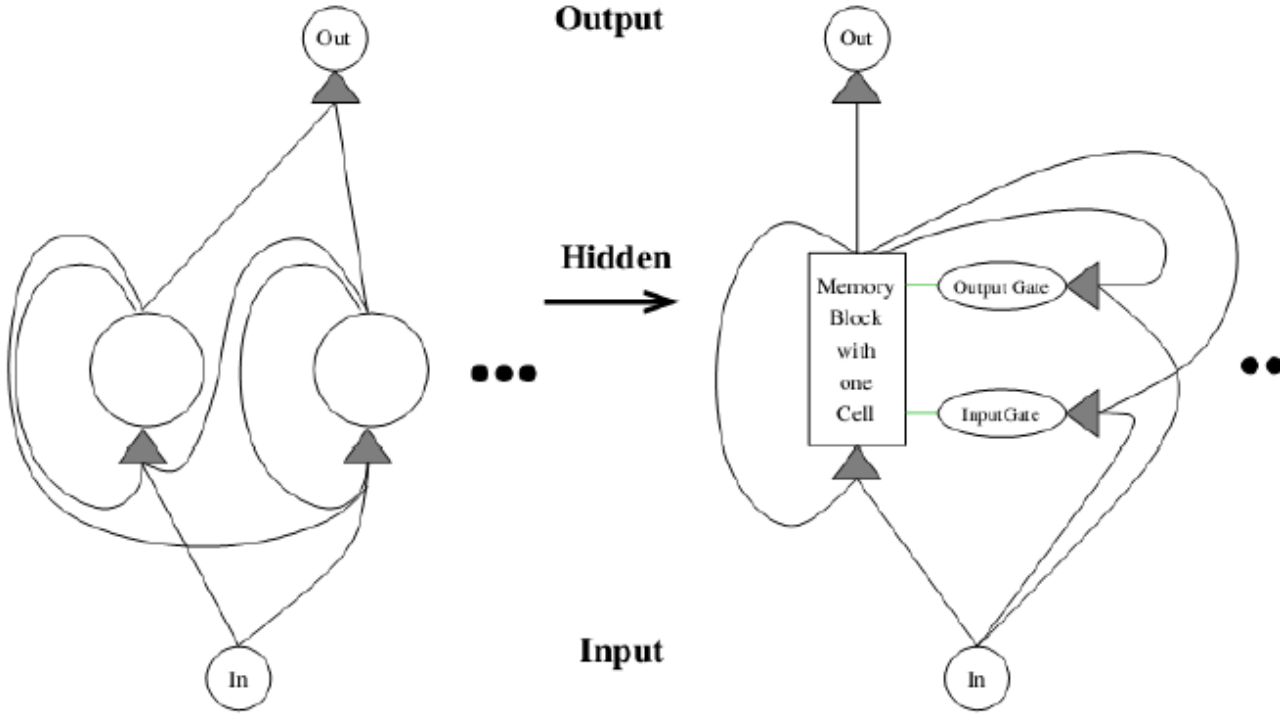
LSTM 与 RNN 的思路类似：



LSTM 的发明是为了解决上面提到的梯度消失问题。LSTM 在反向传播过程中保持更恒定的误差流，且可以处理远处的大型序列。

LSTM Architecture

LSTM 网络 hidden layer 中的基本单元是 memory block，它取代了传统 RNN 中的隐藏单元。

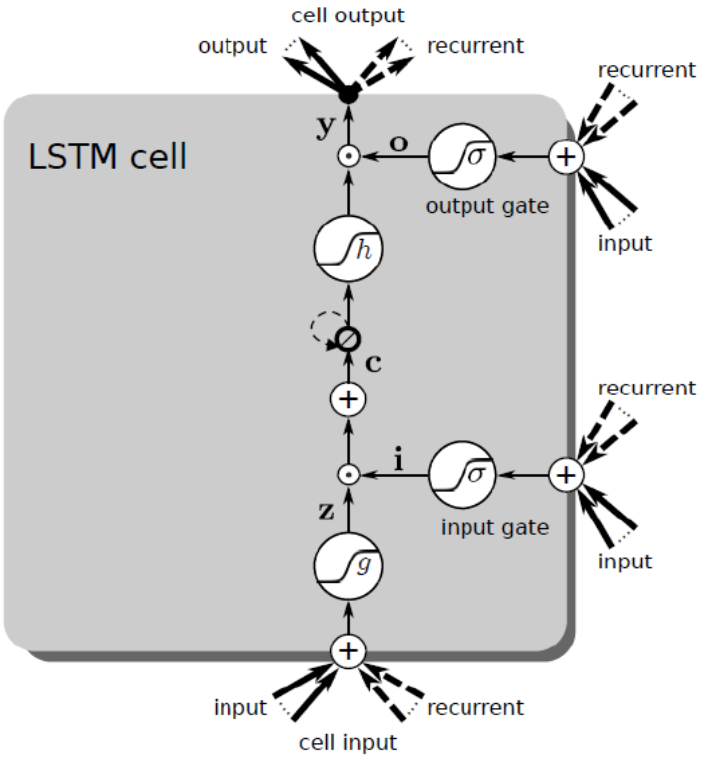


RNN with one FC hidden layer

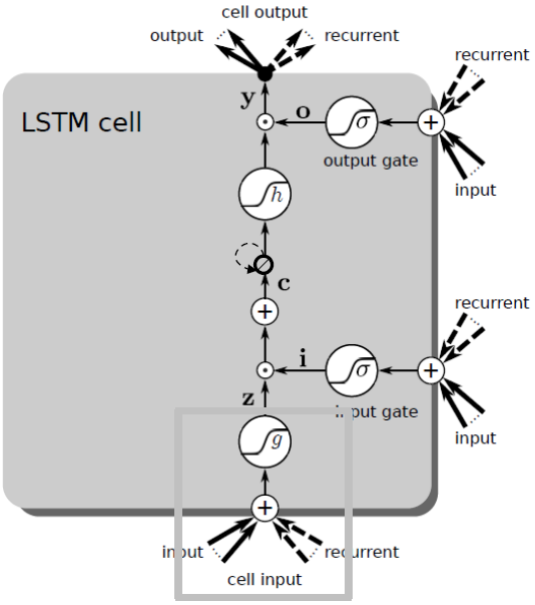
LSTM network with memory blocks in hidden layer (only one is shown)

memory block 包含一个或多个 memory cell 和一对自适应 multiplicative gating units (乘法门控单元)，这些 gate 控制 block 中所有 cell 的输入和输出。memory block 允许 cell 共享相同的 gate，从而减少参数的数量。

Architecture Detail



Input

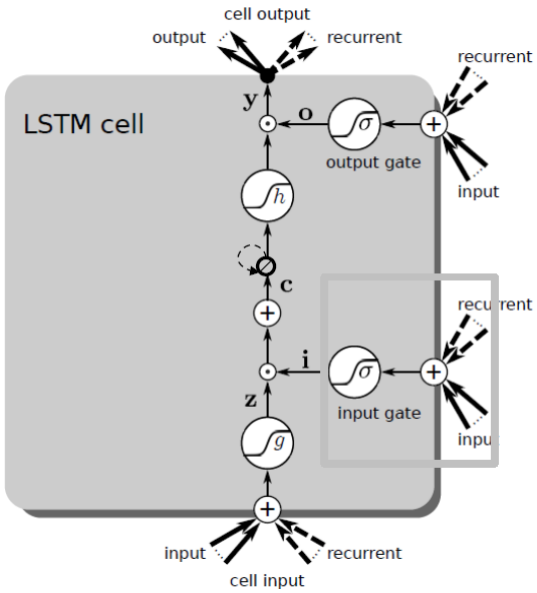


该层决定应将输入中的哪些信息传递到 LSTM cell 中。

$$a_z(t) = W_z x(t) + R_z y(t-1)$$
$$z(t) = g\left(a_z(t)\right)$$

通过将 y(t-1) 与递归权重矩阵 Rz 相乘，该层决定应将上一个时间步长中的哪些信息传递到 LSTM 单元中 (y(t-1) 是上一个时间中的信息，g 是激活函数)。

Input gate



input gate 控制对 memory cells 的写入访问。

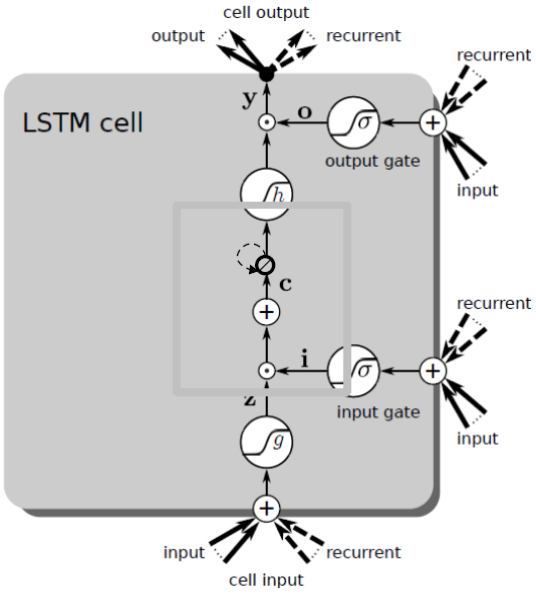
$$a_{in}(t) = W_{in} x(t) + R_{in} y(t-1)$$
$$i(t) = \sigma\left(a_{in}(t)\right)$$

这是通过使用压扁函数 (squashing function) σ 作为一个因素来实现的，该因子稍后将乘以被压扁的 cell input z(t) (见下一步)。

函数 σ 的范围是 (0, 1)，在 0 的情况下可以解释为写访问被拒绝，在 1 的情况下可以解释为授予的写访问权限。当然，介于两者之间的所有值也是可能的。

CEC

每个 memory cell 都包含一个 node，该节点具有固定权重的 self-connected recurrent edge (自连接循环边缘)，确保梯度可以跨越许多时间步长而不会消失 - 这称为CEC（constant error carousel, 恒定误差轮播）。



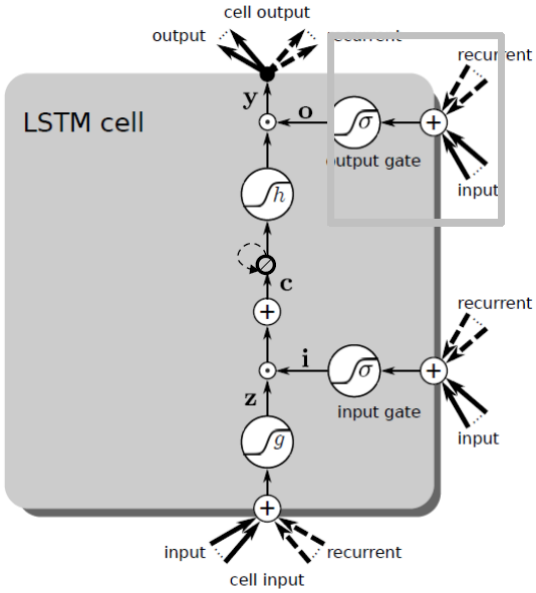
CEC 解决了梯度消失问题。在没有新的输入或 error 信号的情况下，CEC 局部误差回流保持不变，既不会增长也不会衰减。

$$c(t) = z(t) \odot i(t) + c(t-1)$$

注：$\odot$ is the Hadamard product，$z(t)$ 是 input 的结果，$i(t)$ 是 input gate 的结果，$c(t-1)$ 是之前 CEC 的信息。

Hadamard product 指的是两矩阵中对应位置的元素相乘。

Output gate



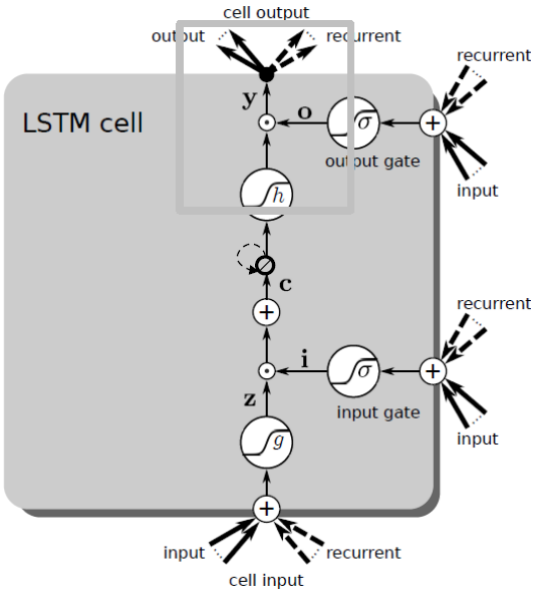
output gate 控制来自 memory cells 的读取访问。

$$a_{\text{out}}(t) = W_{\text{out}}\, x(t) + R_{\text{out}}\, y(t-1)$$
$$o(t) = \sigma\left(a_{\text{out}}(t)\right)$$

这是通过使用 squashing function σ 作为一个因素来实现的，该因素稍后将乘以压扁的 cell content h(c(t))。

函数 σ 的范围是 (0, 1)，在 0 的情况下可以解释为读取访问被拒绝，在 1 的情况下解释为授予读取访问权限。当然，介于两者之间的所有值也是可能的。

Output



存储在 cell c(t) 中的值由函数 h 压扁，该信息是否获得输出由输出门通过 o(t) 决定。

$$y(t) = h(c(t)) \odot i(t) + o(t)$$

注：⊙ is the Hadamard product

单元状态 c 根据其当前状态和 3 个输入进行更新：a_z, a_{in}, a_{out}

LSTM Forward Pass

单元状态 c 根据其当前状态和 3 个输入进行更新：a_z, a_{in}, a_{out}

$$a_z(t) = W_z x(t) + R_Z(y(t-1)), z(t) = g\left(a_z(t)\right)$$
$$a_{in}(t) = W_{in} x(t) + R_{in}(y(t-1)), i(t) = \sigma\left(a_{in}(t)\right)$$
$$c(t) = z(t) \odot i(t) + c(t-1)$$
$$a_{out}(t) = W_{out} x(t) + R_{out}(y(t-1)), o(t) = \sigma\left(a_{out}(t)\right)$$
$$y(t) = h(c(t)) \odot o(t)$$

最终的结果就算 y(t)。

大概的步骤就是：input 和 input gate 的结果合起来给 CEC，CEC 得出来一个值，之后 output gate 再算一个结果，最后 output gate 和 CEC 一起得到一个最终的 output 值。

LSTM Backward Pass

到达单元输出的 error 将传播到 CEC。error 可能会在 CEC 内部长期存在，这确保了非衰减误差 (non-decaying error)，可以桥接输入事件和目标信号之间的时滞。

Advantages of LSTM

- Non-decaying error backpropagation.
- For long time lag problems, LSTM can handle noise and continuous values.
- No parameter fine tuning.
- Memory for long time periods

Limitations of LSTM

LSTM 允许信息在任意时滞和 error 信号中存储，并将 error 信号追溯到很久以前。然而，这种潜在的优势可能导致一个弱点：在时间序列的表示过程中，cell 状态 c(t) 通常倾向于线性增长。

如果我们呈现一个连续的输入流，则 cell 状态可能会以无界的方式增长，从而导致输出压扁函数 h(c(t)) 饱和。