Lecture 27: Recurrent Neural Nets

ECE 417: Multimedia Signal Processing
Mark Hasegawa-Johnson

University of Illinois

12/4/2018

Outline

Basics of DSP: Filtering

$$y[n] = \sum_{m=-\infty}^{\infty} h[m]x[n-m]$$

$$Y(z) = H(z)X(z)$$

Finite Impulse Response (FIR)

$$y[n] = \sum_{m=0}^{N-1} h[m]x[n-m]$$

The coefficients, $h[m]$, are chosen in order to optimally position the $N-1$ zeros of the transfer function, r_k, defined according to:

$$H(z) = \sum_{m=0}^{N-1} h[m]z^{-m} = h[0]\prod_{k=1}^{N-1}\left(1 - r_k z^{-1}\right)$$

Infinite Impulse Response (IIR)

$$y[n] = \sum_{m=0}^{N-1} b_m x[n-m] + \sum_{m=1}^{M-1} a_m y[n-m]$$

The coefficients, b_m and a_m, are chosen in order to optimally position the $N-1$ zeros and $M-1$ poles of the transfer function, r_k and p_k, defined according to:

$$H(z) = \frac{\sum_{m=0}^{N-1} b_m z^{-m}}{1 - \sum_{m=1}^{M-1} a_m z^{-m}} = b_0 \frac{\prod_{k=1}^{N-1}\left(1 - r_k z^{-1}\right)}{\prod_{k=1}^{M-1}\left(1 - p_k z^{-1}\right)}$$

Time Delay Neural Net (TDNN)=FIR + Nonlinearity

$$y[n] = g\left(\sum_{m=0}^{N-1} h[m]x[n-m]\right), \quad \dot{y}[n] = \dot{g}\left(\sum_{m=0}^{N-1} h[m]x[n-m]\right)$$

The coefficients, $h[m]$, are chosen to minimize the error. For example, suppose that there is just one target, $t[N]$, that must be achieved at time $n = N$, so the error term might be just

$$E = \frac{1}{2}\left(y[N] - t[N]\right)^2$$

$$\frac{\partial E}{\partial h[m]} = \frac{\partial E}{\partial y[N]}\frac{\partial y[N]}{\partial h[m]} = \delta[N]\dot{y}[N]x[n-m]$$

Where $\delta[n] = \frac{\partial E}{\partial y[n]}$ is defined as the back-prop error.

Recurrent Neural Net (RNN) = IIR + Nonlinearity

$$y[n] = g\left(x[n] + \sum_{m=1}^{M-1} a_m y[n-m]\right), \quad \dot{y}[n] = \dot{g}\left(\cdot\right)$$

The coefficients, a_m, are chosen to minimize the error. For example, suppose that $E = \frac{1}{2}\left(y[N] - t[N]\right)^2$, then:

$$\frac{\partial E}{\partial a_m} = \sum_{n=0}^{N} \frac{\partial E}{\partial y[n]} \frac{\partial y[n]}{\partial a_m} = \sum_{n=0}^{N} \delta[n]\dot{y}[n]y[n-m]$$

$$\delta[n] = \frac{\partial E}{\partial y[n]} = \sum_{m=1}^{M-1} \delta[n+m]\dot{y}[n+m]a_m$$

Outline

Vanishing/Exploding Gradient

- The "vanishing gradient" problem refers to the tendency of $\frac{\partial y[N]}{\partial x[n]}$ to disappear, exponentially, when $N-n$ is large.
- The "exploding gradient" problem refers to the tendency of $\frac{\partial y[N]}{\partial x[n]}$ to explode toward infinity, exponentially, when $N-n$ is large.
- If the largest feedback coefficient is $|a|>1$, then you get exploding gradient. If not, you get vanishing gradient.

Example: Vanishing Gradient

Suppose

$$y[n] = x[n] + ay[n-1]$$

$$E = \frac{1}{2}\left(y[N] - t[N]\right)^2$$

Then

$$\frac{\partial E}{\partial x[n]} = \frac{\partial E}{\partial y[n]} = \delta[n]$$

where

$$\delta[n] = a\delta[n+1] = a^{N-n}\delta[N]$$

Exponential Decay

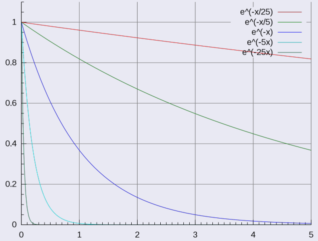


Image credit: PeterQ, Wikipedia

Outline

Gated Recurrent Units (GRU)

Gated recurrent units solve the vanishing gradient problem by making the feedback coefficient, $f[n]$, a sigmoidal function of the inputs. When the input causes $f[n] \approx 1$, then the recurrent unit remembers its own past, with no forgetting (no vanishing gradient). When the input causes $f[n] \approx 0$, then the recurrent unit immediately forgets all of the past.

$$y[n] = i[n]x[n] + f[n]y[n-1]$$

where the input and forget gates depend on $x[n]$ and $y[n]$, as

$$i[n] = \sigma\left(b_i x[n] + a_i y[n-1]\right) \in (0,1)$$
$$f[n] = \sigma\left(b_m x[n] + a_f y[n-1]\right) \in (0,1)$$

How does GRU work? Example

For example, suppose that the inputs just coincidentally have values that cause the following gate behavior:

$$i[n] = \begin{cases} 1 & n = n_0 \\ 0 & \text{otherwise} \end{cases}$$

$$f[n] = \begin{cases} 0 & n = n_0 \\ 1 & \text{otherwise} \end{cases}$$

$$y[n] = i[n]x[n] + f[n]y[n-1]$$

Then $y[N] = y[N-1] = \ldots = y[n_0] = x[n_0]$, memorized! And therefore

$$\frac{\partial y[N]}{\partial x[n]} = \begin{cases} 1 & n = n_0 \\ 0 & \text{otherwise} \end{cases}$$

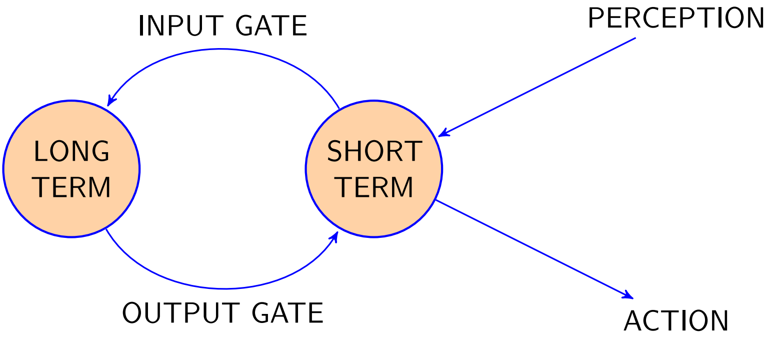
Training the Gates

$$
\begin{aligned}
y[n] &= i[n]x[n] + f[n]y[n-1] \\
i[n] &= \sigma\left(b_i x[n] + a_i y[n-1]\right) \in (0,1) \\
f[n] &= \sigma\left(b_m x[n] + a_f y[n-1]\right) \in (0,1)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial E}{\partial b_i} &= \sum_{n=0}^{N} \frac{\partial E}{\partial y[n]} \frac{\partial y[n]}{\partial i[n]} \frac{\partial i[n]}{\partial b_i} \\
&= \sum_{n=0}^{N} \delta[n]x[n] \frac{\partial i[n]}{\partial b_i}
\end{aligned}
$$

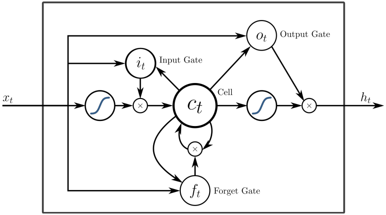
Outline

Characterizing Human Memory



$$\Pr\{\text{remember}\} = p_{LTM} e^{-t/T_{LTM}} + (1 - p_{LTM}) e^{-t/T_{STM}}$$

Neural Network Model: LSTM



$$
\begin{aligned}
i[n] &= \text{input gate} = \sigma(b_i x[n] + a_i c[n-1]) \\
o[n] &= \text{output gate} = \sigma(b_o x[n] + a_o c[n-1]) \\
f[n] &= \text{forget gate} = \sigma(b_f x[n] + a_f c[n-1]) \\
c[n] &= \text{memory cell}
\end{aligned}
$$

$$
\begin{aligned}
y[n] &= o[n]c[n] \\
c[n] &= f[n]c[n-1] + i[n]g\left(b_c x[n] + a_c c[n-1]\right)
\end{aligned}
$$

Outline

- TDNN is a one-dimensional ConvNet, the nonlinear version of an FIR filter. Coefficients are shared across time steps.
- RNN is the nonlinear version of an IIR filter. Coefficients are shared across time steps. Error is back-propagated from every output time step to every input time step.
- Vanishing gradient problem: the memory of an RNN decays exponentially.
- Solution: GRU
- An LSTM is a GRU with one more gate, allowing it to decide when to output information from LTM back to STM.