Fully Connected/Dense

A Fully Connected/Dense layer is insensitive to the order of features.

This is just a property of the dot product

$$\boldsymbol{\Theta}^T \cdot \mathbf{x} = \Theta[\text{perm}]^T \cdot \mathbf{x}[\text{perm}]$$

where $\Theta[\text{perm}]^T$ and $\mathbf{x}[\text{perm}]$ are permutations of $\Theta, \mathbf{x}$.

$$\sum \begin{cases} \text{Machine} & \text{Learning} & \text{is} & \text{easy} & \text{not} & \text{hard} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \boldsymbol{\Theta}_{\text{Machine}} & \boldsymbol{\Theta}_{\text{Learning}} & \boldsymbol{\Theta}_{\text{is}} & \boldsymbol{\Theta}_{\text{easy}} & \boldsymbol{\Theta}_{\text{not}} & \boldsymbol{\Theta}_{\text{hard}} \end{cases}$$

$$=$$

$$\sum \begin{cases} \text{Machine} & \text{Learning} & \text{is} & \text{hard} & \text{not} & \text{easy} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \boldsymbol{\Theta}_{\text{Machine}} & \boldsymbol{\Theta}_{\text{Learning}} & \boldsymbol{\Theta}_{\text{is}} & \boldsymbol{\Theta}_{\text{hard}} & \boldsymbol{\Theta}_{\text{not}} & \boldsymbol{\Theta}_{\text{easy}} \end{cases}$$

Convolution

We have spoken about convolutions as

- Identifying the presence/absence of a feature
- At a *spatial* location

The one-dimensional convolution, when applied to a sequence of tokens

- Identifies the presence/absence of a feature
- At a *temporal* location (index within the sequence)

One dimensional convolution
Slide blue kernel over input

Using one dimensional convolution with kernel size n

- The convolution creates an n-gram feature
- At each (temporal) location in the sequence

As with any other CNN, we can apply multiple kernels

- Each matching a different pattern
- To identify a different feature (n-gram)
- At each location in the sequence

One dimensional convolution multiple kernels

Recurrent

Recurrent layers take *sequences* of vectors as input

$$\mathbf{x}_{0)}, \mathbf{x}_{(1)}, \ldots \mathbf{x}_{(t)} \ldots \mathbf{x}_{(T)}$$

RNN

$\mathbf{h}_{(t)}$ is a **fixed length** vector that "summarizes" the prefix of sequence $\mathbf{x}$ up to element t.

The sequence is processed element by element, so order matters.

$$
\begin{array}{lll}
\mathbf{h}_{(0)} & = & \text{summary}([\text{Machine}]) \\
\mathbf{h}_{(1)} & = & \text{summary}([\text{Machine, Learning}]) \\
\vdots & & \\
\mathbf{h}_{(t)} & = & \text{summary}([\mathbf{x}_{(0)}, \ldots \mathbf{x}_{(t)}]) \\
\vdots & & \\
\mathbf{h}_{(5)} & = & \text{summary}([\text{Machine, Learning, is, easy, not, hard}])
\end{array}
$$

Dropout

- Regularization
- Prevents over-fitting

NN, no dropout layer

NN, Droput layer, no dropout

NN, Dropout layer, 25% dropout

Normalization

- For use in very deep networks
- Keeps distribution of layer outputs "normalized"