# Theano tutorial part 2

AACIMP 2015
Sergii Gavrylov

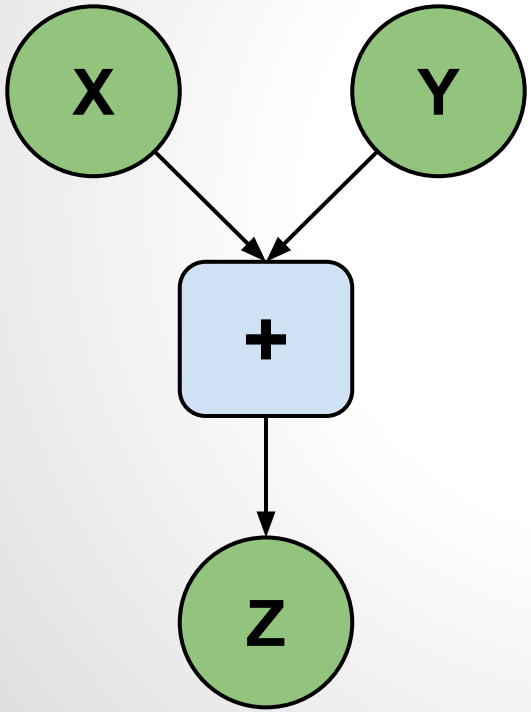
# Overview

- Brief recap
- Multivariate logistic regression
- Multilayer perceptron
- Convolution
- Convolutional neural network
- scan
- Recurrent neural network

# Brief recap

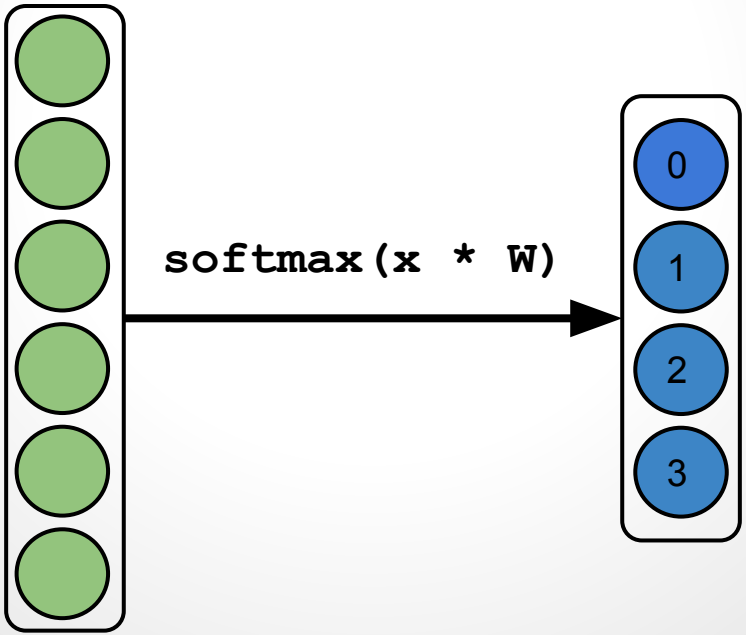
- Symbolic variables
- Functions
- Shared variables / updates
- Gradients
- Substitution

# Computational graph

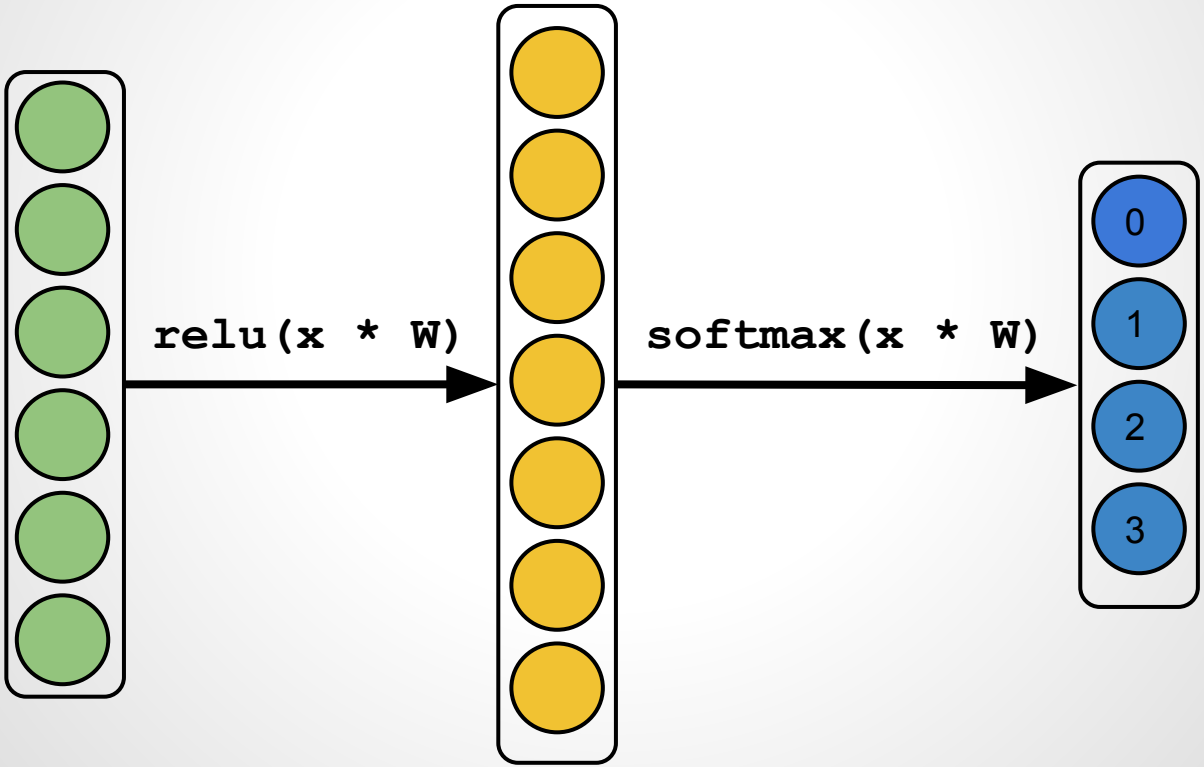



- Code generation
- Symbolic differentiation

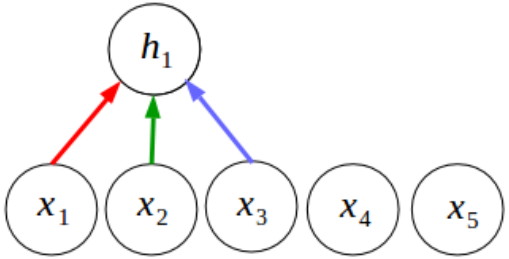
# Multivariate logistic regression



# Multilayer perceptron

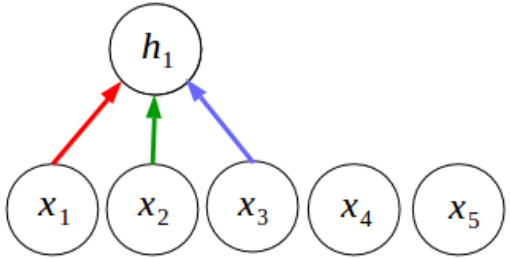


# 1D Convolution

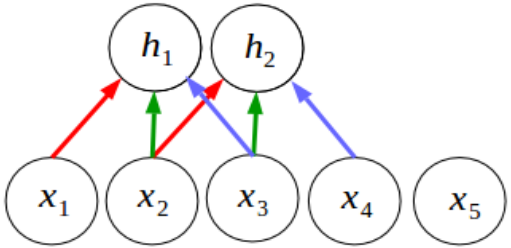



$$\mathbf{h}_1 = f(\mathbf{W} \cdot \mathbf{x}_{[1:3]})$$

# 1D Convolution

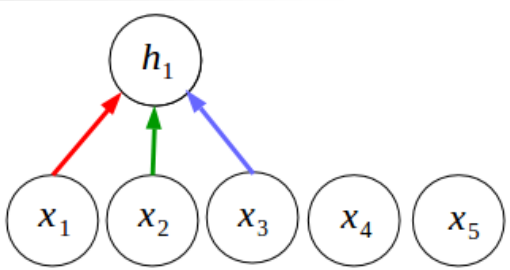



$$\mathbf{h}_1 = f(\mathbf{W} \cdot \mathbf{x}_{[1:3]})$$

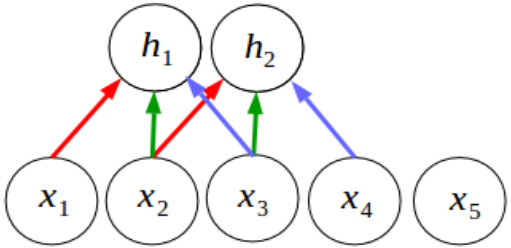



$$\mathbf{h}_2 = f(\mathbf{W} \cdot \mathbf{x}_{[2:4]})$$

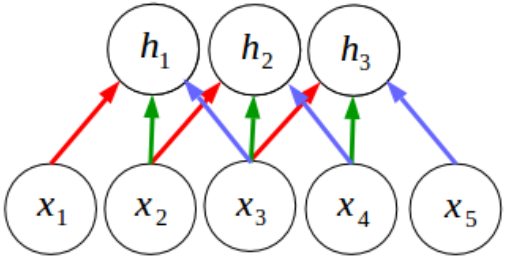
# 1D Convolution




$$\mathbf{h}_1 = f(\mathbf{W} \cdot \mathbf{x}_{[1:3]})$$

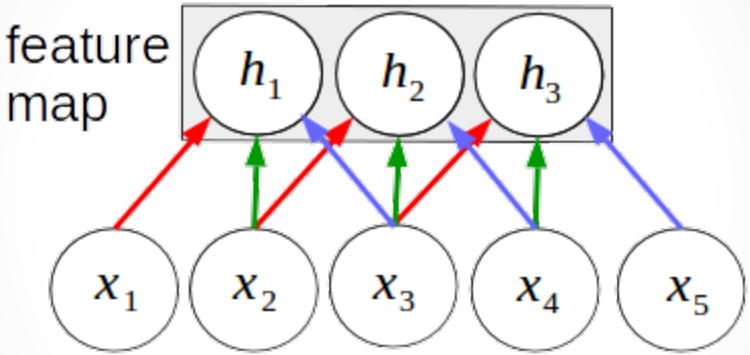



$$\mathbf{h}_2 = f(\mathbf{W} \cdot \mathbf{x}_{[2:4]})$$



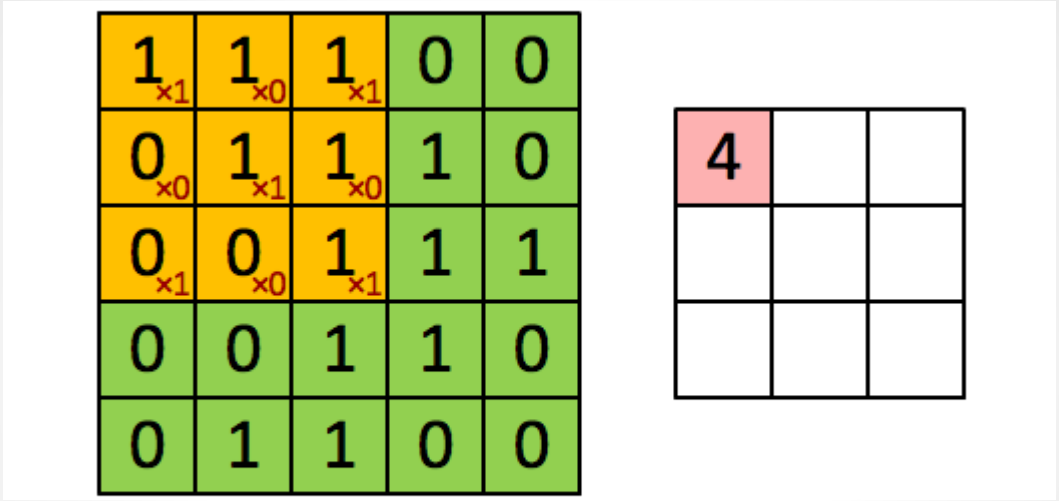

$$\mathbf{h}_3 = f(\mathbf{W} \cdot \mathbf{x}_{[3:5]})$$

# 1D Convolution



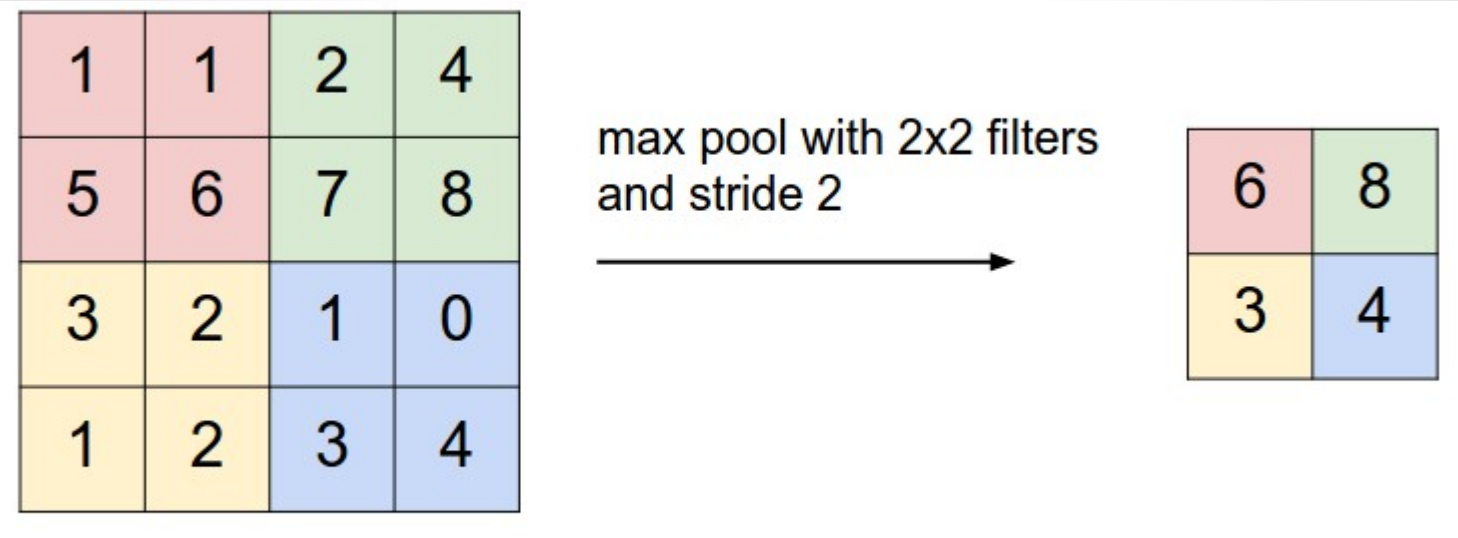

$$\mathbf{h}_i = f\left((\mathbf{W} * \mathbf{x})_i\right)$$

# 2D Convolution




$$\texttt{filter} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

# Max pooling




cs231n.github.io/convolutional-networks

# ConvPoolLayer

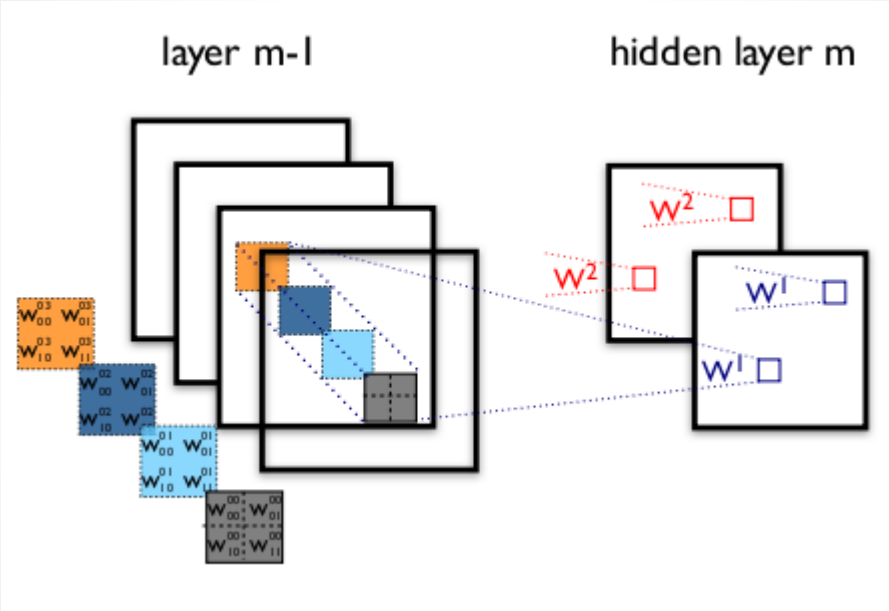
```
from theano.tensor.nnet import conv
from theano.tensor.signal.downsample import max_pool_2d


class ConvPoolLayer(object):
    def __init__(self, w_init):
        self.W = theano.shared(w_init())

    def get_output_expr(self, input_expr):
        conv_out = conv.conv2d(input_expr, self.W)
        pooled_out = max_pool_2d(conv_out, (2, 2))
        return rectify(pooled_out)

    def get_parameters(self):
        return [self.W]
```

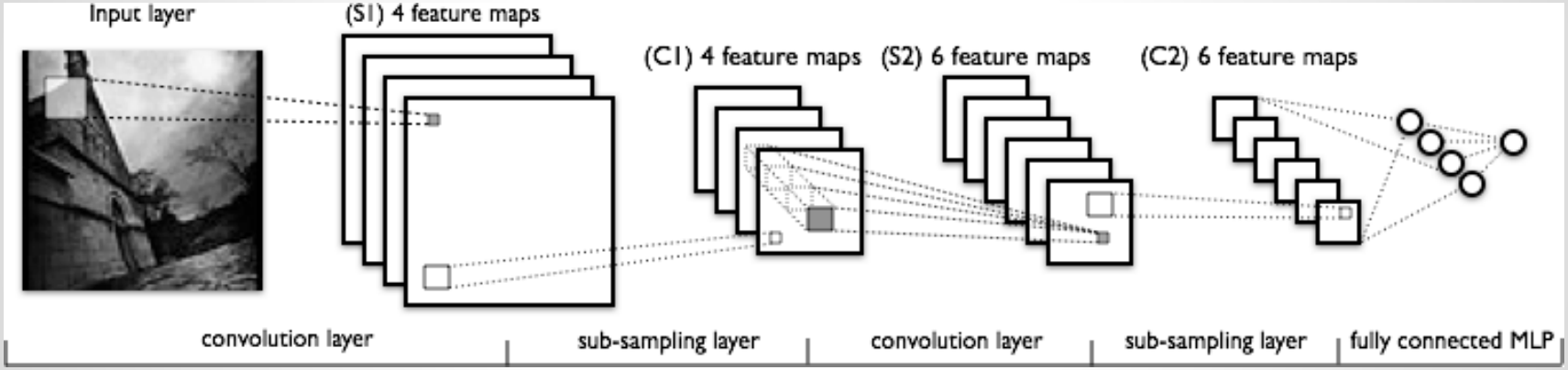
# Convolutional NN




cs231n.github.io/convolutional-networks

deeplearning.net/tutorial/lenet.html

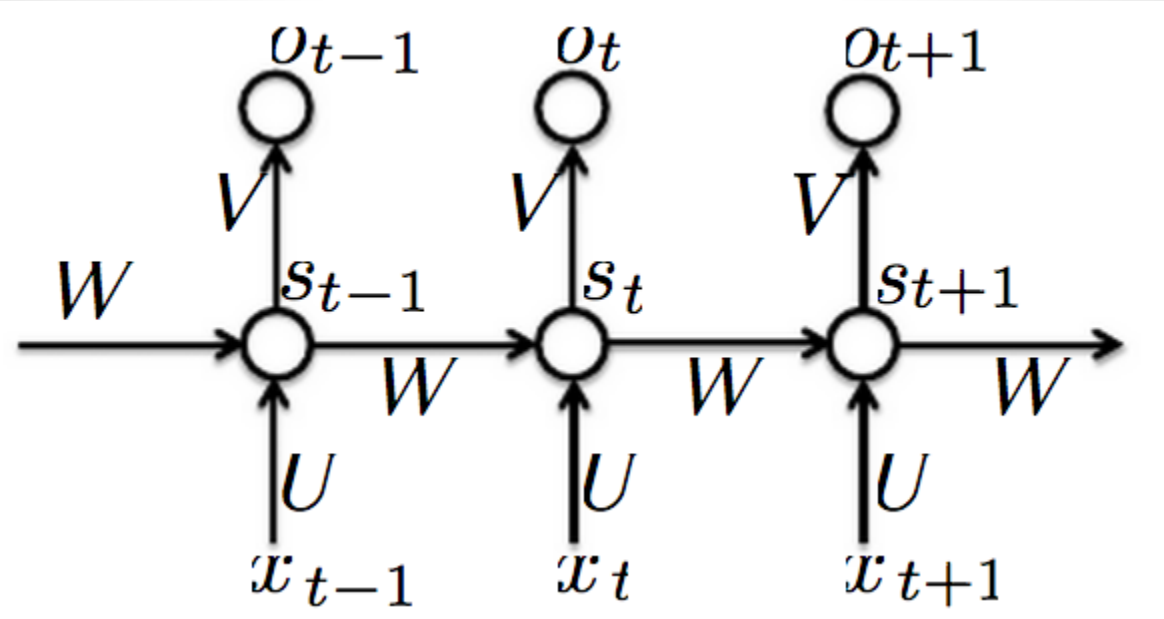
# Convolutional NN




deeplearning.net/tutorial/lenet.html

# scan
# (Symbolic loop in theano)

# Recurrent neural network




www.iro.umontreal.ca/~bengioy/dlbook/rnn.html

# “Vanilla” RNN

$$
\begin{aligned}
\boldsymbol{a}_t &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{s}_{t-1} + \boldsymbol{U}\boldsymbol{x}_t \\
\boldsymbol{s}_t &= \tanh(\boldsymbol{a}_t) \\
\boldsymbol{o}_t &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{s}_t \\
\boldsymbol{p}_t &= \operatorname{softmax}(\boldsymbol{o}_t)
\end{aligned}
$$

www.iro.umontreal.ca/~bengioy/dlbook/rnn.html

# Conclusion

- Theano has a lot of useful building blocks (convolution, scan).
- Theano supports both cpu and gpu backends.