Convolutional Neural Networks II

Dr. Wojtek Kowalczyk

wojtek@liacs.nl

Kernel Matrices

- When interpreting a kernel matrix keep in mind that it might be flipped (or even transposed – depending on the implementation):

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n)$$

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

Constrained Kernels

Sometimes it is beneficial to impose some constraints on kernel matrices, e.g., they must be by a Toeplitz matrix (given by 2n-1) parameters or a circulant matrix (given by n parameters): less than n^2 parameters!

Well known in image/signal processing!

$$\begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ a_{-1} & a_0 & a_1 & a_2 \\ a_{-2} & a_{-1} & a_0 & a_1 \\ a_{-3} & a_{-2} & a_{-1} & a_0 \\ a_{-4} & a_{-3} & a_{-2} & a_{-1} \end{bmatrix}$$

Toeplitz:
each diagonal is constant

$$\begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ a_3 & a_0 & a_1 & a_2 \\ a_2 & a_3 & a_0 & a_1 \\ a_1 & a_2 & a_3 & a_0 \end{bmatrix}$$

Circulant:
each row is a circular shift of the previous one

Treating image boundaries

- **valid:** “kernel doesn’t look outside an image”
  - image: nxn, kernel: kxk => result: (n-k+1)x(n-k+1)
  - output image is smaller than the input image

- **same:** pad the image with 0’s, so the size of the output image is the same as the input image
  - boundary pixels are less frequently visited

- **full:** pad the image with sufficient number of 0’s so each image pixel is visited exactly k times
  - output image has size n+k-1

Initialization of kernels

- many heuristics available (see Keras documentation)

- most popular:
  - "Glorot" : weights normally distributed with 0 mean and stddev=sqrt(2/(fan_in + fan_out))
  - "Glorot uniform" : weights uniformly distributed between [-B, +B], where B = sqrt(6)/(fan_in + fan_out)

    (fan_in= # input nodes, fan_out=#output nodes)

- another heuristic (helps finding "good network architecture"):
  "freeze randomly initialized weights of kernels;
  optimize remaining weights =>
  quick feedback on the quality of network design"

Nesterov Momentum

"On the importance of initialization and momentum in deep learning" (Sutskever et al., ICML 2013)

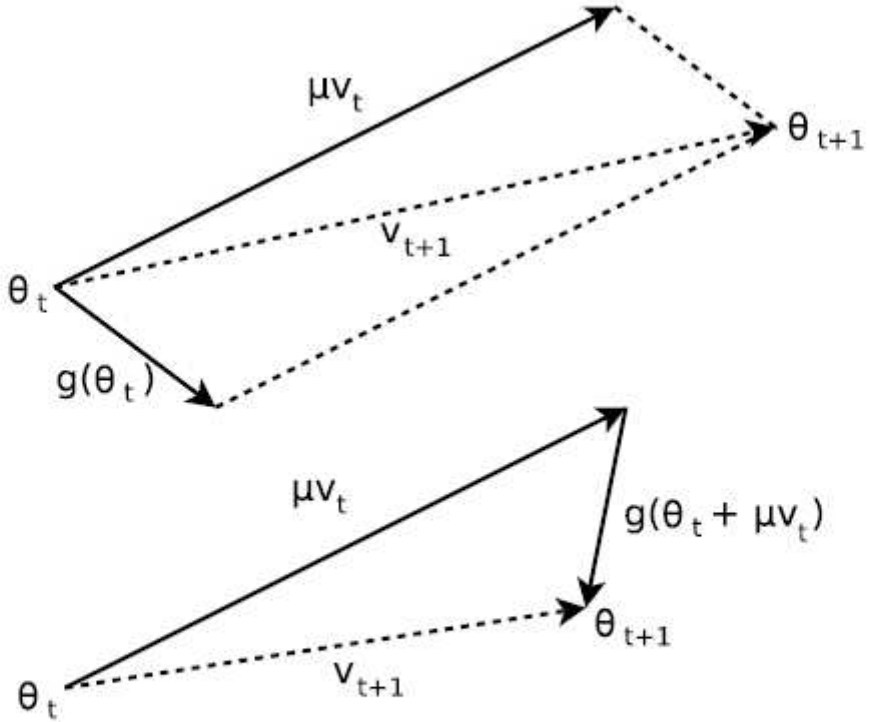


Figure 1. **(Top)** Classical Momentum **(Bottom)** Nesterov Accelerated Gradient

SGD

Algorithm 8.1 Stochastic gradient descent (SGD) update at training iteration k

Require: Learning rate ϵ_k.

Require: Initial parameter $\boldsymbol{\theta}$

while stopping criterion not met **do**

Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.

Compute gradient estimate: $\hat{\boldsymbol{g}} \leftarrow +\frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$

Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \hat{\boldsymbol{g}}$

end while

SGD with Momentum

Algorithm 8.2 Stochastic gradient descent (SGD) with momentum

Require: Learning rate ϵ, momentum parameter α.
Require: Initial parameter $\boldsymbol{\theta}$, initial velocity $\boldsymbol{v}$.
while stopping criterion not met **do**
  Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
  Compute gradient estimate: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
  Compute velocity update: $\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \epsilon \boldsymbol{g}$
  Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$
end while

SGD with Nesterov Momentum

Algorithm 8.3 Stochastic gradient descent (SGD) with Nesterov momentum

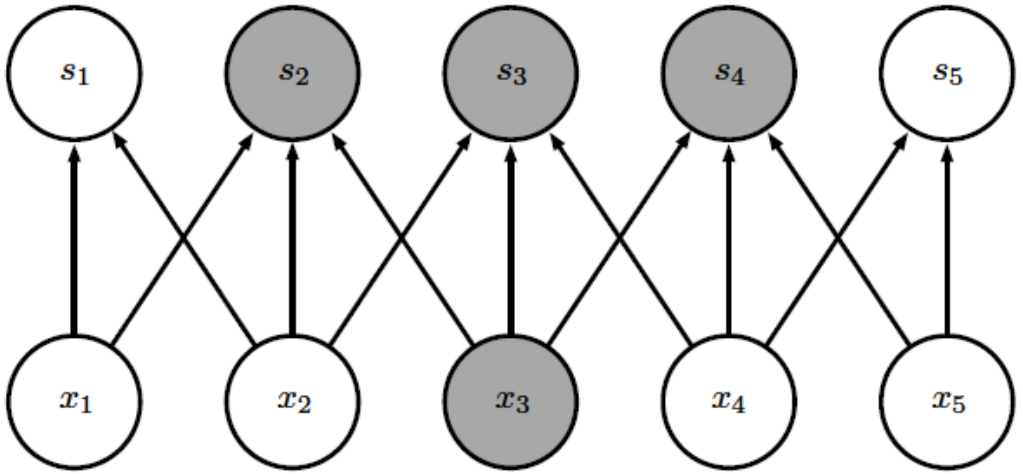
Require: Learning rate ϵ, momentum parameter α.
Require: Initial parameter $\boldsymbol{\theta}$, initial velocity $\boldsymbol{v}$.
while stopping criterion not met **do**
  Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding labels $\boldsymbol{y}^{(i)}$.
  Apply interim update: $\tilde{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta} + \alpha \boldsymbol{v}$
  Compute gradient (at interim point): $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\tilde{\boldsymbol{\theta}}} \sum_i L(f(\boldsymbol{x}^{(i)}; \tilde{\boldsymbol{\theta}}), \boldsymbol{y}^{(i)})$
  Compute velocity update: $\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \epsilon \boldsymbol{g}$
  Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$
end while

General principles

- Sparse interactions (sparse connectivity):
  - m inputs, n outputs
  - fully connected: mxn connections, O(mxn) operations
  - sparsely connected: k connections to each output, k<<n, kxn connections , O(kxn) operations
  - only “local input-output interactions”
- Parameter sharing (tied weights):
  - “all fragments of an image are treated in the same way”
- Equivariant representations
  - Definition: f(x) is equivariant to g(x) iff f(g(x))=g(f(x))
  - convolutions are equivariant w.r.t. translation:

Sparse Connectivity ("from below")

Sparse connections due to small convolution kernel



Dense connections

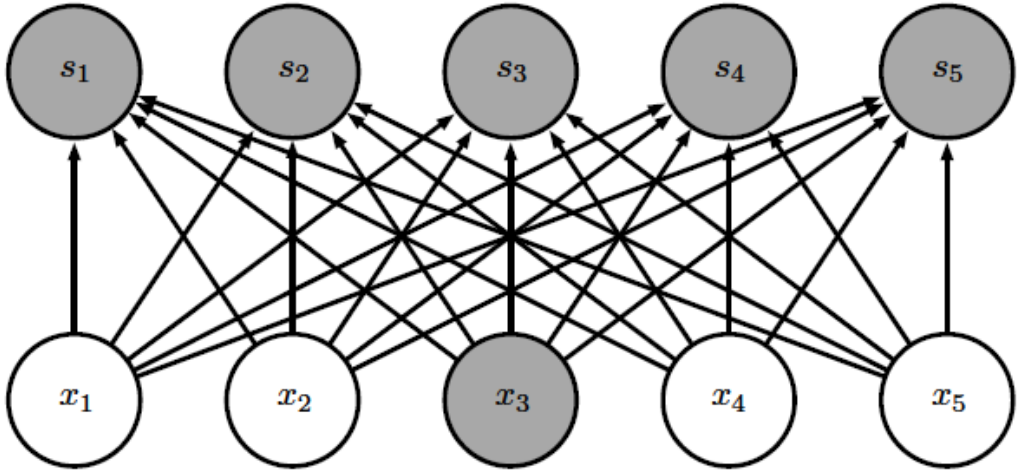


Figure 9.2

Sparse Connectivity ("from above")

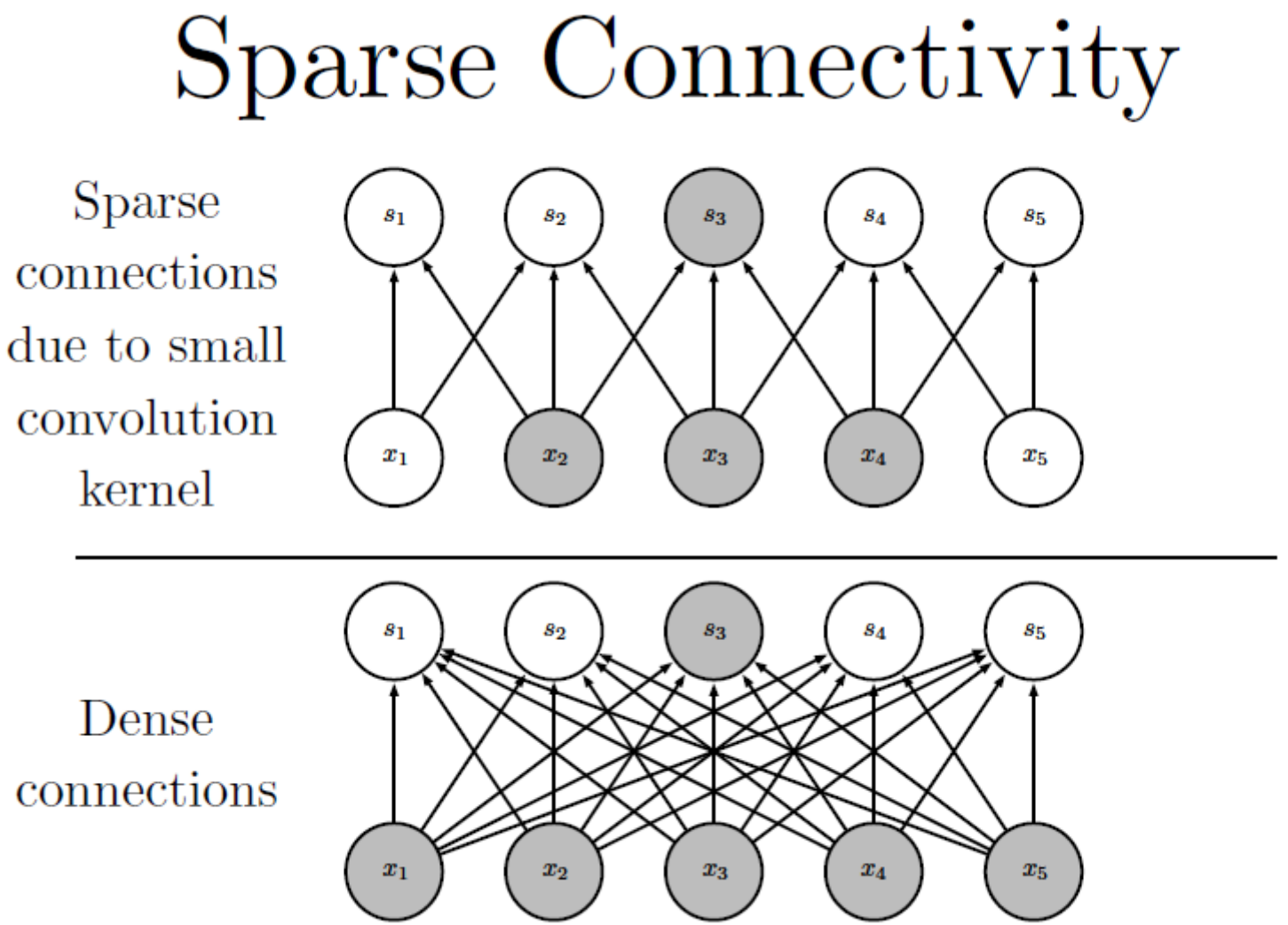


Figure 9.3

“the deeper the wider”

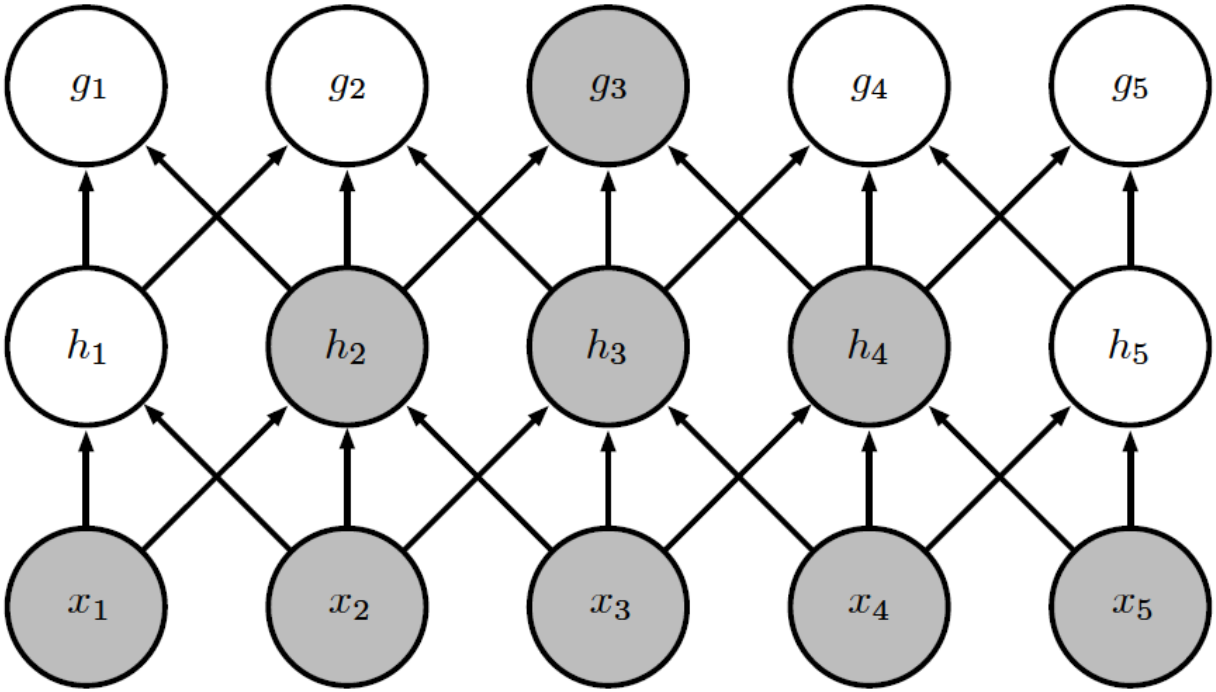


Figure 9.4

Much more ...

- Read Chapter 9 from deeplearningbook.org (whole)
- Slides from: http://www.deeplearningbook.org/slides/09_conv.pdf
- Optional:

http://videolectures.net/icml2015_weinberger_neural_networks/?q=kilian%20weinberger

www.cs.cornell.edu/~kilian/research/budgeted/budgeted.html