

# Kronecker factoring for convolutional networks

Yaroslav Bulatov

April 17, 2017

## Linear Network

Suppose we have matrix parameter  $W$  and are trying to model  $Y$  as follows

$$Y = B'WA$$

We measure our prediction error as

$$e = \hat{Y} - B'WA$$

To minimize prediction error we seek  $W$  to minimize the following loss, defined using trace as follows

$$J = \frac{1}{2}\text{tr}(e'e)$$

To find derivative and Hessian, use approach of matrix differentials from Magnus, Nuedecker – [www.janmagnus.nl/misc/mdc2007-3rdedition](http://www.janmagnus.nl/misc/mdc2007-3rdedition)

First take differential:

$$dJ = \text{tr}(e'de) = -\text{tr}(e'B'dWA)$$

Rearranging by using properties of trace and then using First Identification table (p.198 of Magnus), the gradient of  $W$  is obtained as

$$G = -BeA'$$

Taking differential of this expression we get

$$dG = BB'dWAA'$$

Let lower case versions of variables represent vectorized versions. Vectorizing both sides and applying Kronecker/vec transformation rule, we get

$$dg = (AA' \otimes BB')dw$$

From this we can extract the Hessian by visual inspection as

$$H = (AA' \otimes BB')$$

## Piecewise Linear Network

For a piecewise linear neural network, the loss for each example is locally linear, and we can write our total loss as sum of per-example losses

$$J = \sum_i J_i$$

Where each example is associated with its version of  $A_i$ ,  $B_i$  and  $e_i$  and  $J_i = -\text{tr}(e_i' B_i' dW A_i)$

The total Hessian is a sum of per-example Hessians, so we get

$$H = \sum_i H_i = \sum_i A_i A_i' \otimes B_i B_i'$$

Now apply Kronecker factorization approximation to get

$$H \approx \left( \sum_i A_i A_i' \right) \otimes \left( \sum_i B_i B_i' \right)$$

If our individual examples are vectors,  $A_i$  and  $B_i$  can be stacked as columns into matrices  $A$  and  $B$ , and the expression above can be written as

$$H \approx (AA') \otimes (BB')$$

Since Kronecker product commutes with matrix inverse, preconditioner can be written as

$$H^{-1} \approx (AA')^{-1} \otimes (BB')^{-1}$$

## Convolutional Network

So far we obtained the Hessian with respect to matrix variable  $W$ . Suppose our operation is a convolution, in which case we can write it as matrix multiplication where  $W$  is a function of another variable, written as  $W = KU$ . Here  $U$  represents our matrix of tunable parameters and  $K$  is the parameter tiling matrix that generates the convolution matmul. The derivative of  $W$  with respect to  $U$  can be written as (p.205 of Magnus)

$$\frac{dW}{dU} = (I \otimes K)$$

Using this and chain rule for Hessian matrices (p.125 of Magnus), original Hessian of our loss with respect to  $U$  can be written

$$H_u = (I' \otimes K')(AA' \otimes BB')(I \otimes K)$$

Rearranging we get

$$H_u = (AA' \otimes K'BB'K)$$

Applying same steps as for piecewise linear network we get Kronecker factorized Hessian as

$$H \approx AA' \otimes K'BB'K$$

For optimization, our preconditioner is

$$H^{-1} \approx (AA')^{-1} \otimes (K'BB'K)^{-1}$$