R PACKAGE FOR NEURAL NETWORKS

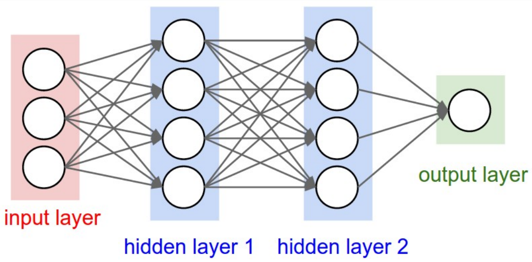
OxWaSPneuralnets

Giuseppe Di Benedetto Ho Chung Law Kaspar Märtens Marcin Mider

4 December 2015

NEURAL NETWORKS

Neural Network




```
* initialise weights and biases
* for ( i in 1 : num_epochs ) do:
    * for ((x,y) in train_data) do:
        * feedforward pass
        * backpropagation
    * update parameters
```

Forward pass

Input data: $\{x_i\}_{i=1}^{n}$
Layers: $l = 1, \ldots, L$

Forward pass

Input data: $\{x_i\}_{i=1}^{n}$

Layers: $l = 1, \ldots, L$

$W_{ij}^{(l)}$ - weight of connection between unit j in layer l and unit i in layer $l+1$.

$b_i^{(l)}$ - bias of a unit i in layer $l+1$.

Forward pass

Input data: $\{x_i\}_{i=1}^n$
Layers: $l = 1, \ldots, L$
$W_{ij}^{(l)}$ - weight of connection between unit j in layer l and unit i in layer $l+1$.
$b_i^{(l)}$ - bias of a unit i in layer $l+1$.
$a_i^{(l)}$ - activation - output of unit i in layer l, $a_i^{(1)} = x_i$ by convention.

Forward pass

Input data: $\{x_i\}_{i=1}^n$
Layers: $l = 1, \ldots, L$
$W_{ij}^{(l)}$ - weight of connection between unit j in layer l and unit i in layer $l+1$.
$b_i^{(l)}$ - bias of a unit i in layer $l+1$.
$a_i^{(l)}$ - activation - output of unit i in layer l, $a_i^{(1)} = x_i$ by convention.

Starting with the first hidden layer ($l = 1$ below) and proceeding recursively until the last hidden layer ($l = L - 2$ below):

Compute a score: $z_i^{(l+1)} = \sum_{j=1}^n W_{i,j}^{(l)} a_i^{(l)} + b_i^{(l)}$

Forward pass

Input data: $\{x_i\}_{i=1}^{n}$
Layers: $l = 1, \ldots, L$
$W_{ij}^{(l)}$ - weight of connection between unit j in layer l and unit i in layer $l+1$.
$b_i^{(l)}$ - bias of a unit i in layer $l+1$.
$a_i^{(l)}$ - activation - output of unit i in layer l, $a_i^{(1)} = x_i$ by convention.

Starting with the first hidden layer ($l = 1$ below) and proceeding recursively until the last hidden layer ($l = L - 2$ below):

Compute a score: $z_i^{(l+1)} = \sum_{j=1}^{n} W_{i,j}^{(l)} a_i^{(l)} + b_i^{(l)}$
Compute new activation: $a_i^{(l+1)} = \sigma(z_i^{(l+1)})$, where $\sigma(z) = \frac{1}{1+exp(-z)}$

Forward pass

Input data: $\{x_i\}_{i=1}^n$
Layers: $l = 1, \ldots, L$
$W_{ij}^{(l)}$ - weight of connection between unit j in layer l and unit i in layer $l+1$.
$b_i^{(l)}$ - bias of a unit i in layer $l+1$.
$a_i^{(l)}$ - activation - output of unit i in layer l, $a_i^{(1)} = x_i$ by convention.

Starting with the first hidden layer ($l = 1$ below) and proceeding recursively until the last hidden layer ($l = L - 2$ below):

Compute a score: $z_i^{(l+1)} = \sum_{j=1}^n W_{i,j}^{(l)} a_i^{(l)} + b_i^{(l)}$
Compute new activation: $a_i^{(l+1)} = \sigma(z_i^{(l+1)})$, where $\sigma(z) = \frac{1}{1+exp(-z)}$

For the last layer, compute a score as usual, but use $\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}$ for the transformation.

Backward pass

Training set: $\{(x^{(i)}, y^{(i)})\}_{i=1\dots m}$

$$\min_{\mathbf{W},\mathbf{b}} J(\mathbf{W},\mathbf{b}) = \frac{1}{m}\sum_{i=1}^{m} J(\mathbf{W},\mathbf{b}; x^{(i)}, y^{(i)}) + \frac{\lambda}{2}\,||\mathbf{W}||_{l^2}^2$$

λ: weight decay parameter

Backward pass

Training set: $\{(x^{(i)}, y^{(i)})\}_{i=1\dots m}$

$$\min_{\mathbf{W},\mathbf{b}} J(\mathbf{W},\mathbf{b}) = \frac{1}{m}\sum_{i=1}^{m} J(\mathbf{W},\mathbf{b}; x^{(i)}, y^{(i)}) + \frac{\lambda}{2}\,||\mathbf{W}||_{l^2}^2$$

λ: weight decay parameter

$$b_{ij}^{(l)}, W_{ij}^{(l)} \sim \mathcal{N}(0, \epsilon^2)$$

$$\delta^{(n_l)} = -(\mathbf{y} - \mathbf{a}^{(l)}) \bullet f'(\mathbf{z}^{(n_l)}), \qquad \delta^{(l)} = \left(t(\mathbf{W}^{(l)}) \cdot \delta^{(l+1)}\right) \bullet f'(\mathbf{z}^{(l)})$$

$$\nabla_{\mathbf{W}^{(l)}} J(\mathbf{W},\mathbf{b},; x, y) = \delta^{(l+1)} \cdot t(\mathbf{a}^{(l)}), \qquad \nabla_{\mathbf{b}^{(l)}} J(\mathbf{W},\mathbf{b},; x, y) = \delta^{(l+1)}$$

Updating the parameters

Gradient descent:

$$\mathbf{W}^{t+1} = \mathbf{W}^t - \varepsilon \nabla_{\mathbf{W}} J$$

where ε is the learning rate

Updating the parameters

Gradient descent:

$$\mathbf{W}^{t+1} = \mathbf{W}^{t} - \varepsilon \nabla_{\mathbf{W}} J$$

where ε is the learning rate

The momentum method:

a technique for accelerating gradient descent, introduces velocity $\mathbf{V}$

$$\mathbf{V}^{t+1} = \mu \mathbf{V}^{t} - \varepsilon \nabla_{\mathbf{W}} J$$

$$\mathbf{W}^{t+1} = \mathbf{W}^{t} + \mathbf{V}^{t+1}$$

where ε is the learning rate and $\mu \in [0, 1]$ is the momentum coefficient.

Regularisation Of the Neural Network

Need to control the Neural Network from overfitting.

Regularisation Of the Neural Network

Need to control the Neural Network from overfitting.

Use a regularised loss function with parameter λ.

i.e. Loss = Cross Entropy + $\lambda \times$weight decay term

Regularisation Of the Neural Network

Need to control the Neural Network from overfitting.

Use a regularised loss function with parameter λ.

i.e. Loss = Cross Entropy + $\lambda \times$weight decay term

Regularisation Of the Neural Network

Need to control the Neural Network from overfitting.

Use a regularised loss function with parameter λ.

i.e. Loss = Cross Entropy + $\lambda\times$weight decay term

The weight decay term is just the sum of all the weights squared.

Regularisation Of the Neural Network

Need to control the Neural Network from overfitting.

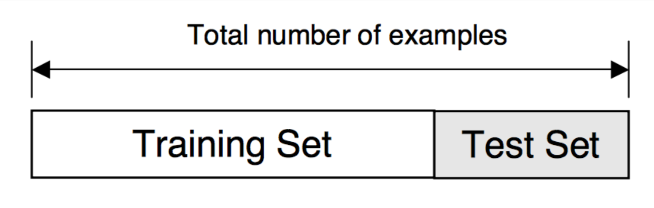
Use a regularised loss function with parameter λ.

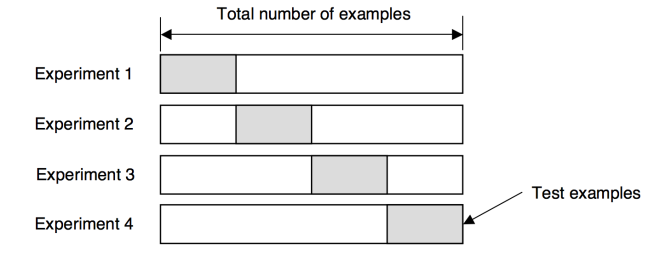
i.e. Loss = Cross Entropy + $\lambda \times$weight decay term

The weight decay term is just the sum of all the weights squared.

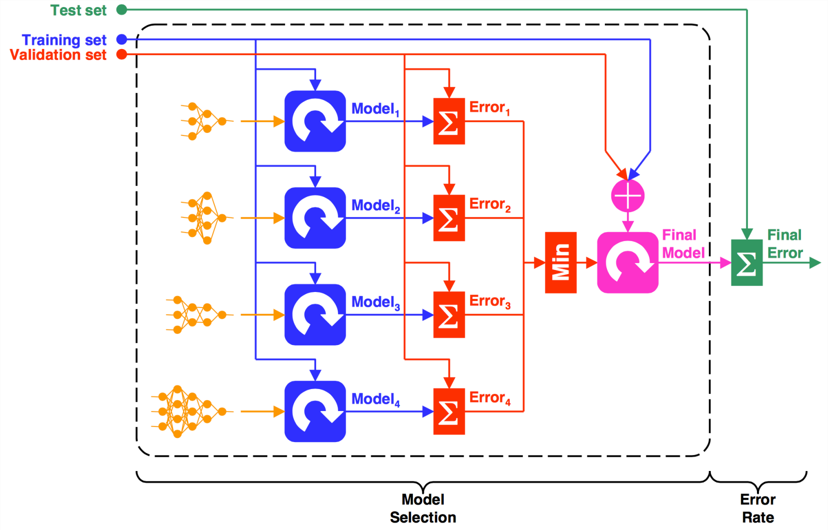
Use K-fold cross validation to choose the value of λ.

Cross Validation in Pictures

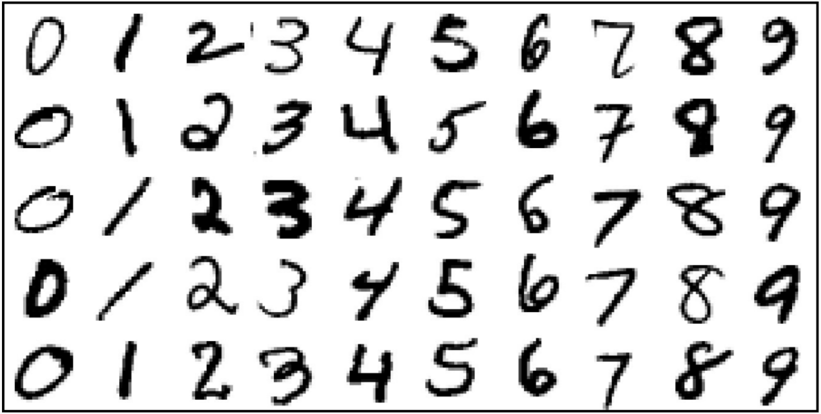




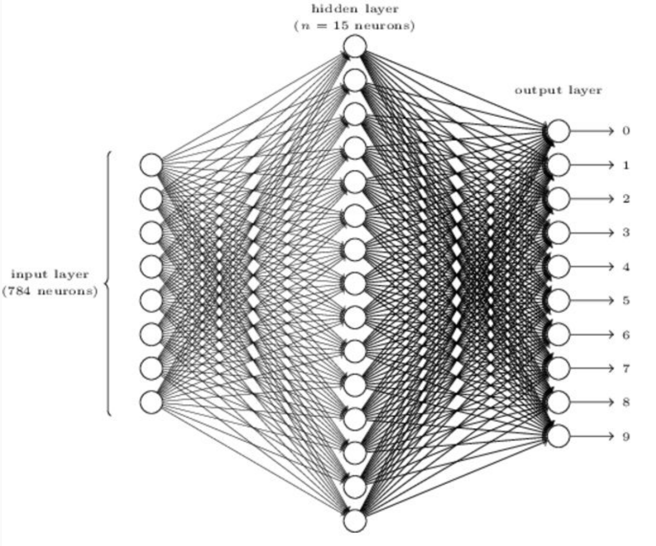
Cross Validation in Pictures



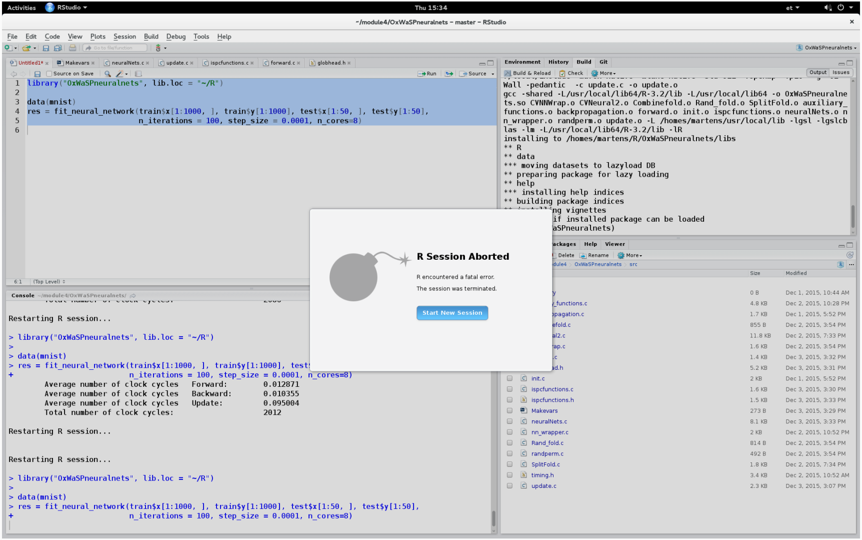
Classification of hand-written digits



Classification of hand-written digits



Our Implementation


~/module4/OxWaSPneuralnets – master – RStudio
library("OxWaSPneuralnets", lib.loc = "~/R")
data(mnist)
res = fit_neural_network(train$x[1:1000,], train$y[1:1000], test$x[1:50,], test$y[1:50],
n_iterations = 100, step_size = 0.0001, n_cores=8)
R Session Aborted
R encountered a fatal error.
The session was terminated.
Start New Session
Restarting R session...
> library("OxWaSPneuralnets", lib.loc = "~/R")
> data(mnist)
Average number of clock cycles Forward: 0.012871
Average number of clock cycles Backward: 0.010355
Average number of clock cycles Update: 0.095004
Total number of clock cycles: 2012
installing to /homes/martens/R/OxWaSPneuralnets/libs
** moving datasets to lazyload DB
** preparing package for lazy loading
** installing help indices
** building package indices

Our package OxWaSPneuralnets

To use our package, install it as follows

```
devtools::install_github("mmider/OxWaSPneuralnets")
```

Load the package

```
library(OxWaSPneuralnets)
```

and see the help files `?fit_neural_network` or `?CV_neural_network`.

Our package OxWaSPneuralnets

R: Fit the neural network ▾ Find in Topic

fit_neural_network {OxWaSPneuralnets} R Documentation

Fit the neural network

Description

Fit the neural network

Usage

```
fit_neural_network(train_X, train_y, test_X, test_y, n_hidden_layers = 1,
  hidden_layer_sizes = c(20), n_iterations = 100, step_size = 0.01,
  lambda = 0.001, n_cores = 8)
```

Arguments

`train_X`	Matrix of training data (data points in rows, features in columns)
`train_y`	Vector of labels for training data (these have to be integers from 0 to n_classes - 1)
`test_X`	Matrix of test data
`test_y`	Vector of labels for test data
`n_hidden_layers`	Number of hidden layers in the neural network
`hidden_layer_sizes`	Vector containing the number of neurons in each hidden layer
`n_iterations`	The number of iterations for fitting the neural network
`step_size`	The step size for updating parameters at each iteration
`lambda`	The regularisation parameter
`n_cores`	The number of parallel cores

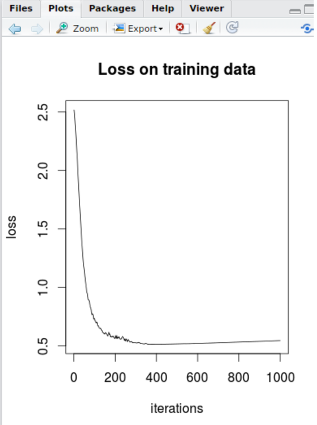
Output

Console ~/oxwasp/module4/OxWaSPneuralnets/

```
> data(mnist)
> # Pick only first 1000 data points (for speed)
> res = fit_neural_network(train$x[1:1000, ], train$y[1:1000]
, test$x[1:500, ], test$y[1:500], n_iterations = 1000, step_s
ize = 0.00005)



> 
```

Output

```
Console ~/oxwasp/module4/OxWaSPneuralnets/
> data(mnist)
> # Pick only first 1000 data points (for speed)
> res = fit_neural_network(train$x[1:1000, ], train$y[1:1000]
, test$x[1:500, ], test$y[1:500], n_iterations = 1000, step_s
ize = 0.00005)
> plot(res)
>
```



Output

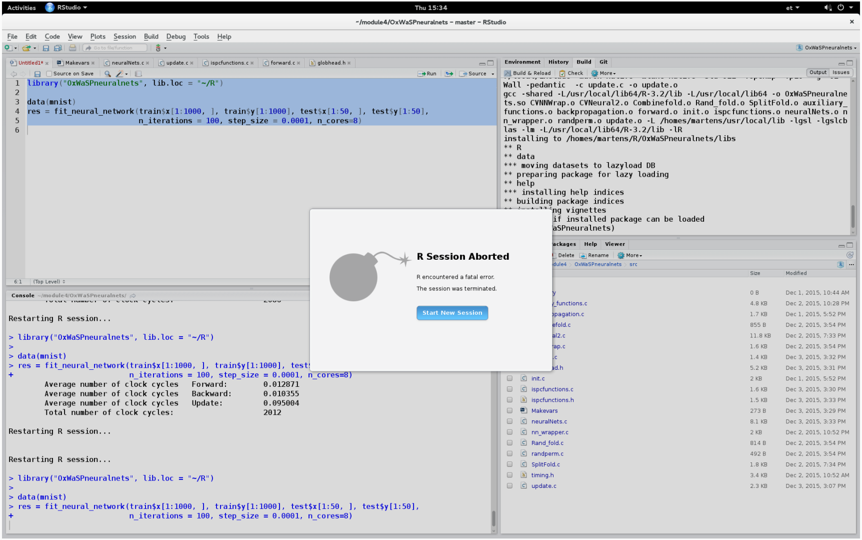
```
Console ~/oxwasp/module4/OxWaSPneuralnets/
> data(mnist)
> # Pick only first 1000 data points (for speed)
> res = fit_neural_network(train$x[1:1000, ], train$y[1:1000]
, test$x[1:500, ], test$y[1:500], n_iterations = 1000, step_s
ize = 0.00005)
> plot(res)
> res
Train accuracy:         0.965
Test accuracy:  0.778
> table(res$pred_test, test$y[1:500])

     0  1  2  3  4  5  6  7  8  9
  0 38  0  1  0  0  1  1  0  1  0
  1  0 65  1  1  0  1  0  0  0  0
  2  2  0 40  2  0  1  2  0  2  0
  3  0  1  4 30  0  6  0  1  1  1
  4  0  0  1  0 43  1  3  1  2  4
  5  0  0  1  9  0 31  5  0  1  0
  6  1  0  0  0  4  0 32  0  1  0
  7  0  0  4  2  0  2  0 42  3  7
  8  0  1  3  0  0  5  0  0 27  1
  9  1  0  0  1  8  2  0  5  2 41
>
```

It is possible to parallelise

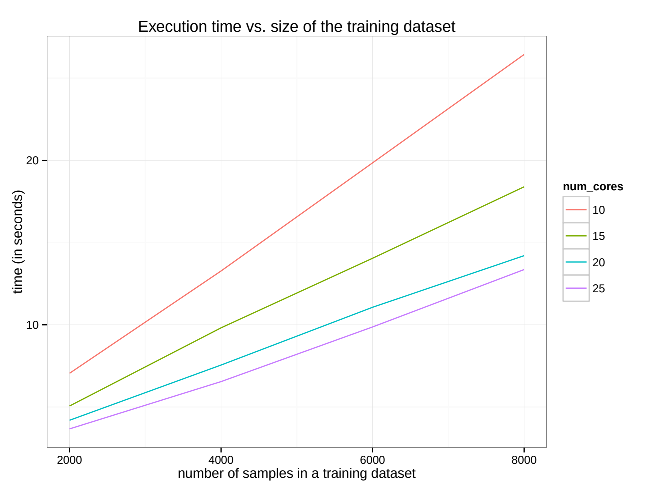
It is possible to parallelise

```
* initialise weights and biases
* for ( i in 1 : num_epochs ) do:
    * for ((x,y) in train_data) do:
        * feedforward pass
        * backpropagation
    * update parameters
```

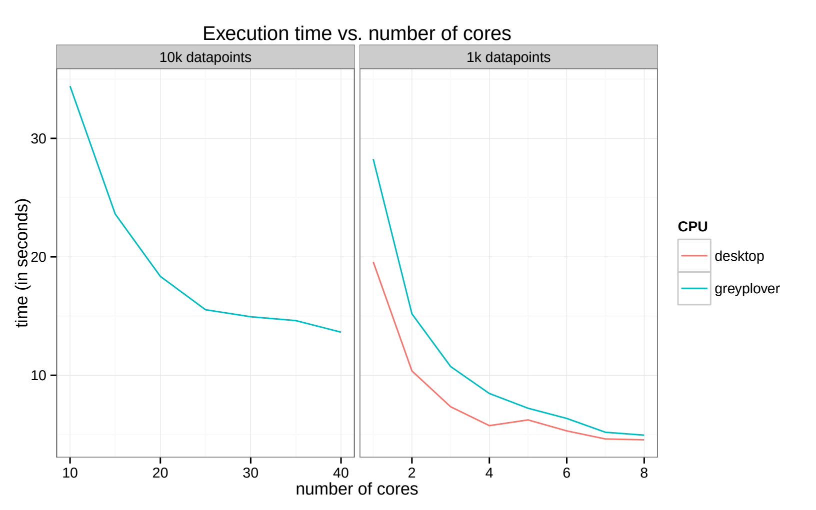
We measured performance...


~/module4/OxWaSPneuralnets – master – RStudio
library("OxWaSPneuralnets", lib.loc = "~/R")
data(mnist)
res = fit_neural_network(train$x[1:1000,], train$y[1:1000], test$x[1:50,], test$y[1:50],
n_iterations = 100, step_size = 0.0001, n_cores=8)
Restarting R session...
Average number of clock cycles Forward: 0.012871
Average number of clock cycles Backward: 0.010355
Average number of clock cycles Update: 0.095004
Total number of clock cycles: 2012
installing to /homes/martens/R/OxWaSPneuralnets/libs
** preparing package for lazy loading
** building package indices
R Session Aborted
R encountered a fatal error.
The session was terminated.
Start New Session

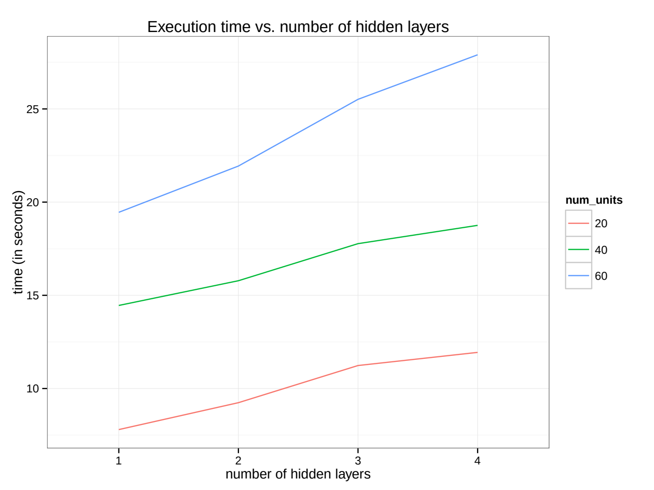
Performance



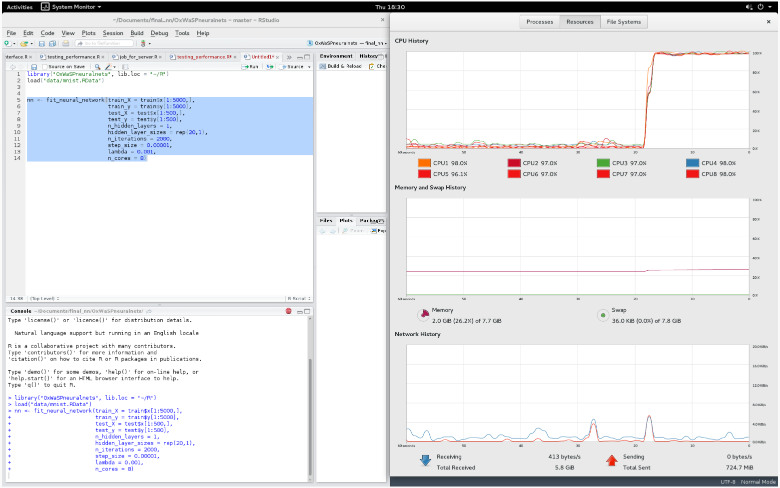
Performance



Performance



CPU bound



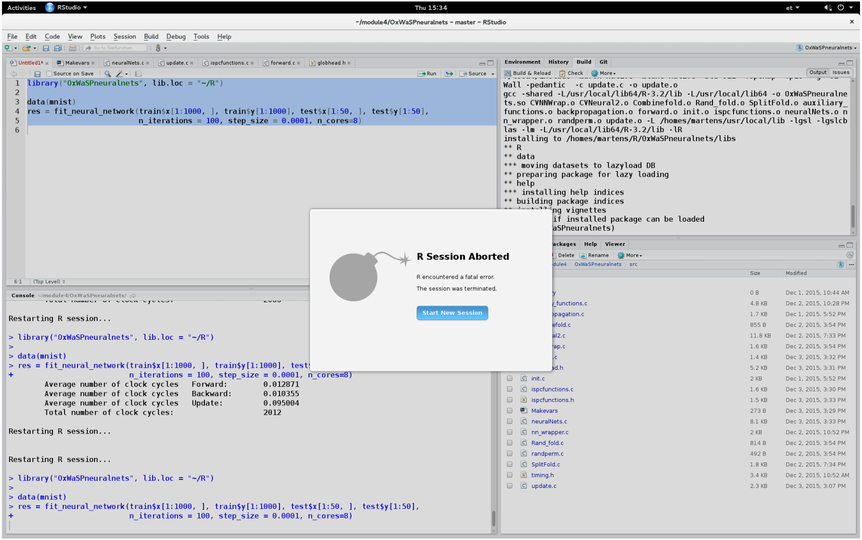
Activities
System Monitor
Thu 18:30
~/Documents/final_nn/OxWaSPneuralnets – master – RStudio
File Edit Code View Plots Session Build Debug Tools Help
library("OxWaSPneuralnets", lib.loc = "~/R")
load("data/mnist.RData")
nn <- fit_neural_network(train_X = train$x[1:5000,],
train_y = train$y[1:5000],
test_X = test$x[1:500,],
test_y = test$y[1:500],
n_hidden_layers = 1,
hidden_layer_sizes = rep(20,1),
n_iterations = 2000,
step_size = 0.00001,
lambda = 0.001,
n_cores = 8)
Console ~/Documents/final_nn/OxWaSPneuralnets/
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
Processes
Resources
File Systems
CPU History
CPU1 98.0%
CPU2 97.0%
CPU3 97.0%
CPU4 98.0%
CPU5 96.1%
CPU6 97.0%
CPU7 97.0%
CPU8 98.0%
Memory and Swap History
Memory
2.0 GiB (26.2%) of 7.7 GiB
Swap
36.0 KiB (0.0%) of 7.8 GiB
Network History
Receiving 413 bytes/s
Total Received 5.8 GiB
Sending 0 bytes/s
Total Sent 724.7 MiB

Improving performance further

We carried out experiments to improve performance by

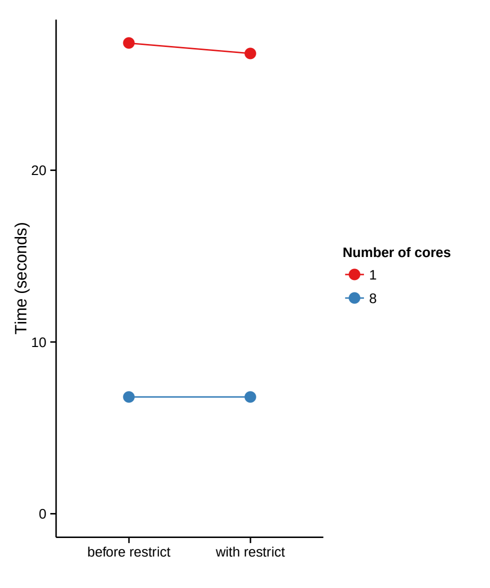
1) using keyword `restrict`

2) using ISPC to take advantage of vector units

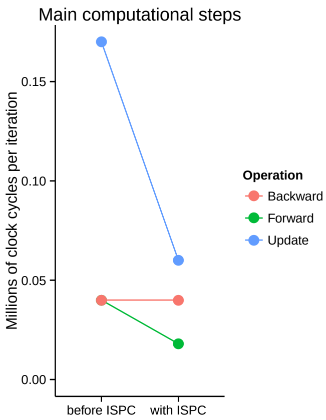


~/module4/OxWaSPneuralnets – master – RStudio
library("OxWaSPneuralnets", lib.loc = "~/R")
data(mnist)
res = fit_neural_network(train$x[1:1000,], train$y[1:1000], test$x[1:50,], test$y[1:50],
n_iterations = 100, step_size = 0.0001, n_cores=8)
Wall -pedantic -c update.c -o update.o
gcc -shared -L/usr/local/lib64/R-3.2/lib -L/usr/local/lib64 -o OxWaSPneuralnets.so CVNNWrap.o CVNeural2.o Combinefold.o Rand_fold.o SplitFold.o auxiliary_functions.o backpropagation.o forward.o init.o ispcfunctions.o neuralNets.o nn_wrapper.o randperm.o update.o -L /homes/martens/usr/local/lib -lgsl -lgslcblas -lm -L/usr/local/lib64/R-3.2/lib -lR
installing to /homes/martens/R/OxWaSPneuralnets/libs
** R
** data
*** moving datasets to lazyload DB
** preparing package for lazy loading
** help
*** installing help indices
** building package indices
R Session Aborted
R encountered a fatal error.
The session was terminated.
Start New Session
Restarting R session...
> library("OxWaSPneuralnets", lib.loc = "~/R")
> data(mnist)
> res = fit_neural_network(train$x[1:1000,], train$y[1:1000], test$x[1:50,], test$y[1:50],
+ n_iterations = 100, step_size = 0.0001, n_cores=8)
Average number of clock cycles Forward: 0.012871
Average number of clock cycles Backward: 0.010355
Average number of clock cycles Update: 0.095004
Total number of clock cycles: 2012
Restarting R session...
Restarting R session...

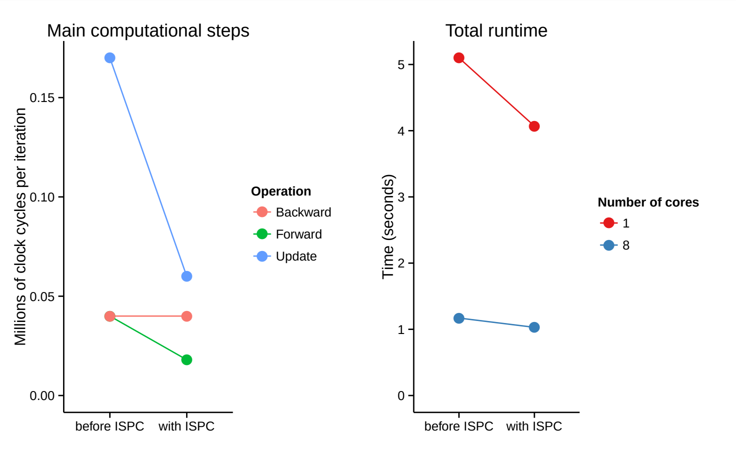
Improving performance: restrict



Improving performance: ISPC



Improving performance: ISPC



Performance: Cross Validation on Digits Data set

Cross Validation is completed in C with instructions in the R package.

Performance: Cross Validation on Digits Data set

Cross Validation is completed in C with instructions in the R package.

3750 observations with 784 features.

3000 observations for training the Neural Network.

Performance: Cross Validation on Digits Data set

Cross Validation is completed in C with instructions in the R package.

3750 observations with 784 features.

3000 observations for training the Neural Network.

750 observations for testing.

$\lambda = \;\; 0.001 \quad 0.01 \quad 0.1 \quad 1 \quad 10$

Performance: Cross Validation on Digits Data set

Cross Validation is completed in C with instructions in the R package.

3750 observations with 784 features.

3000 observations for training the Neural Network.

750 observations for testing.

$\lambda = \; 0.001 \quad 0.01 \quad 0.1 \quad 1 \quad 10$

Fold Size = 10, i.e. validation set is 300 observations

One hidden layer with 20 hidden features.

Parallelisation is done with openMP across folds and λ.

Parallelisation: 8 cores

Problems with Neural Networks with Cross Validation parallelisation at the same time.

All 8 cores were used.

Comparison between speed in serial cross validation with parallel Neural Networks and the reverse:

Table: Parallelisation: 8 cores on Desktop

Parallelisation	User Time	System Time	Elapsed Time
On Cross Validation	2517.2s	2.4s	340.9s
On Neural Networks	2681.7s	2.0s	350.2s

Parallelisation: greywagtail server

Table: Comparison between parallelisation: greywagtail server

Parallelisation	User Time	System Time	Elapsed Time
On Cross Validation	3683.1s	31.1s	105.5s
On Neural Networks	4890.35s	18.65s	168.1s

Around 30-40 cores were used

Some variation due to demand of servers

Neural Networks parallelisation: Have to wait for other batches

MPI over the servers is a possibility: Work in progress....

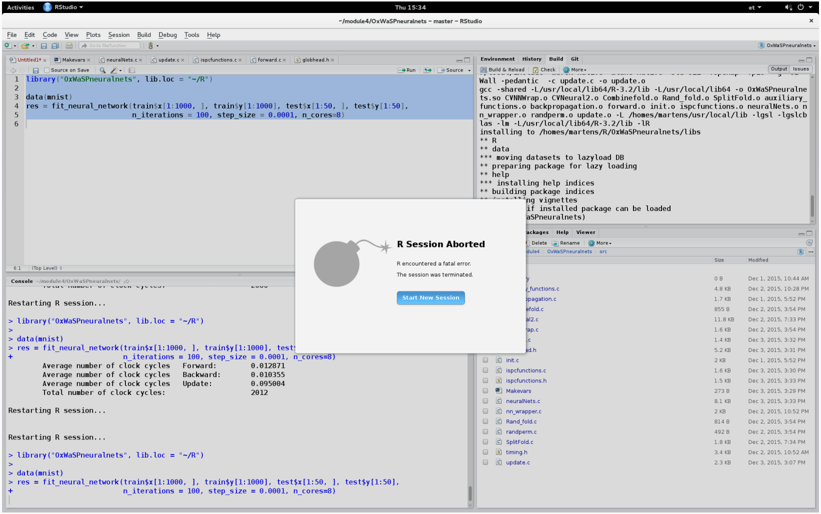
Some Results from Cross Validation

Table: Cross Validation Result on Digit Data set

λ	Prediction Success
0.001	0.867
0.01	0.851
0.1	0.863
1	0.865
10	0.854
0.001*	0.868

Conclusion....

Conclusion....



~/module4/OxWaSPneuralnets – master – RStudio
library("OxWaSPneuralnets", lib.loc = "~/R")
data(mnist)
res = fit_neural_network(train$x[1:1000,], train$y[1:1000], test$x[1:50,], test$y[1:50],
n_iterations = 100, step_size = 0.0001, n_cores=8)
R Session Aborted
R encountered a fatal error.
The session was terminated.
Start New Session
Restarting R session...
Average number of clock cycles Forward: 0.012871
Average number of clock cycles Backward: 0.010355
Average number of clock cycles Update: 0.095004
Total number of clock cycles: 2012
installing to /homes/martens/R/OxWaSPneuralnets/libs
** building package indices