Test flash transpose functions

Jason Willwerscheid

2/10/2018

Proposed updates: flash_update_single_loading operates as usual. flash_update_single_factor transposes the flash object, calls flash_update_single_loading, then transposes back. I want to see whether these two "transpose" operations result in a performance hit.

Functions needed:

```
flash_transpose = function(f){
  if (is.null(f)) {return(NULL)}
  tmp = names(f)
  tmp[c(which(tmp == "EL"), which(tmp == "EF"))] = c("EF", "EL")
  tmp[c(which(tmp == "EL2"), which(tmp == "EF2"))] = c("EF2", "EL2")
  tmp[c(which(tmp == "fixl"), which(tmp == "fixf"))] = c("fixf", "fixl")
  tmp[c(which(tmp == "gl"), which(tmp == "gf"))] = c("gf", "gl")
  tmp[c(which(tmp == "KL_l"), which(tmp == "KL_f"))] = c("KL_f", "KL_l")
  tmp[c(which(tmp == "ebnm_param_l"),
        which(tmp == "ebnm_param_f"))] = c("ebnm_param_f", "ebnm_param_l")
  tmp[c(which(tmp == "penloglik_l"),
        which(tmp == "penloglik_f"))] = c("penloglik_f", "penloglik_l")
  names(f) = tmp
  if (is.matrix(f$tau)) {f$tau = t(f$tau)}
  return(f)
}

flash_transpose_data = function(data){
  if (is.matrix(data$Yorig)) {data$Yorig = t(data$Yorig)}
  if (is.matrix(data$missing)) {data$missing = t(data$missing)}
  if (is.matrix(data$Y)) {data$Y = t(data$Y)}
  return(data)
}

update_single_factor = function(data,f,k) {
  tf = flashr:::flash_update_single_loading(flash_transpose_data(data), flash_transpose(f),
                                            k)
  return(flash_transpose(tf))
}
```

Create a large matrix:

```
n <- 400
p <- 20000
k <- 3
L <- matrix(rnorm(k * n), ncol=k)
F <- matrix(rnorm(k * p), ncol=k)
Y <- L %*% t(F) + rnorm(n * p)
```

Fit factors using flash_add_greedy:

```
fl <- flashr::flash_add_greedy(Y, 3, var_type="constant")
```

```
## fitting factor/loading 1
```

```
## fitting factor/loading 2
## fitting factor/loading 3
```

Now time the previous flash_update_single_factor on the first factor:

```
data <- flashr::flash_set_data(Y)
f <- fl
system.time(flashr:::flash_update_single_factor(data, f, 1))
```

```
##    user  system elapsed
##   3.032   0.385   2.072
```

Do it with the proposed method:

```
f <- fl
system.time(update_single_factor(data, f, 1))
```

```
##    user  system elapsed
##   2.655   0.591   2.278
```

Try the second factor and test in opposite order (in case caching is artificially speeding up the second test):

```
f <- fl
system.time(update_single_factor(data, f, 2))
```

```
##    user  system elapsed
##   3.148   0.553   2.349
```

```
f <- fl
system.time(flashr:::flash_update_single_factor(data, f, 2))
```

```
##    user  system elapsed
##   3.766   0.439   2.315
```

Time the transpose operation alone:

```
system.time(flash_transpose(fl))
```

```
##    user  system elapsed
##   0.124   0.035   0.188
```

Profiling reveals (as might be expected) that effectively all time is spent on the matrix transposes (transposing tau, transposing the data matrices).

So the cost of maintainability is a small performance hit. Is it worth it?