CSVM Logbook

Marc Groefsema

December 23, 2015

Influence of second order Soft Assignment

I'd like to know what the influence of 'second order soft assignment' is, compared to regular Soft Assignment patchSize = 12, stride = 2. As feat. descriptor, HOG is used, with 4x4 cells. As codebook, a deep convolutional codebook is used.

The settings will be tested on the MNIST dataset, with 6000 random images to train, and 1000 images to test. As a classifier the CSVM will be used, with learningRate = 0.000002, and 20000 iterations, wieghtInit = 0.002, C = 1000

These settings are not proven to be the "best" settings for each, both the provide a common context for this experiment, with only the type of assignment function as controlled variable.

Here hog uses no padding

With both settings, 8 runs are done. To compare the two codebook activation function, a two-sides paired t-test is used. A P-value threshold of 0.05 will be used.

```
SoftAssignment <- c(87.2, 86.6, 87.7, 88.2, 85.9, 86.0, 84.2, 86.5)
SoftAssignmentClipping <- c(87.6, 84.7, 83.6, 84.5, 84.1, 83.0, 85.8, 85.5)

t.test(SoftAssignment, SoftAssignmentClipping, paired = FALSE, alternative = "two.sided")
```

```
##
##  Welch Two Sample t-test
##
## data:  SoftAssignment and SoftAssignmentClipping
## t = 2.5107, df = 13.681, p-value = 0.02528
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.2427602 3.1322398
## sample estimates:
## mean of x mean of y
##   86.5375   84.8500
```

This test shows that in second order soft assignment performs slightly worse than a first order soft assignment.

Influence of the amount of centroids used

I'd like to explore the influence of the number of centroids in the Deep Codebook used in MNIST on the classification rate. All values are gained from a 8 fold crossvalidation. The same settings as the above experiment are used, with normal Soft Assignment and a Deep Codebook. Centroids are build by using 200000 random patches.

Here HOG uses no padding

```
v10centr <- c(66.4, 75.0, 63.5, 75.5, 75.3, 78.2, 73.2, 72.7)
v30centr <- c(83.4, 86.0, 85.5, 85.5, 83.7, 83.9, 84.8, 83.4)
v50centr <- c(87.2, 86.1, 87.8, 88.5, 88.5, 85.5, 88.0, 87.4)
v70centr <- c(88.2, 89.3, 89.9, 89.9, 88.5, 89.3, 88.1, 88.3)
v90centr <- c(89.9, 90.0, 89.7, 88.3, 89.5, 91.2, 87.8, 88.7)
v110centr <- c(90.0, 90.6, 91.5, 91.8, 89.4, 91.0, 90.0, 89.8)
v130centr <- c(91.0, 89.5, 90.6, 90.7, 89.3, 90.6, 87.5, 89.1)
v150centr <- c(91.0, 88.3, 91.4, 88.2, 89.9, 89.2, 92.5, 90.3)
v200centr <- c(89.3, 90.9, 89.7, 90.4, 90.4, 89.6, 91.4, 89.7)
```

Now the same measurements are done, but with the normal codebook instead

```
s10centr <- c(78.1, 80.3, 82.3, 83.4, 85.7, 85.7, 79.7, 80.2)
s30centr <- c(90.2, 91.2, 89.6, 90.1, 90.9, 90.9, 89.5, 92.5)
s50centr <- c(93.1, 92.4, 94.8, 94.3, 93.3, 92.6, 93.3, 92.9)
s70centr <- c(94.2, 94.9, 93.8, 93.8, 93.5, 93.2, 93.2, 93.3)
s90centr <- c(95.1, 94.5, 95.3, 95.3, 95.1, 95.1, 92.8, 92.8)
s110centr <- c(95.2, 95.3, 93.5, 95.1, 94.1, 93.7, 93.9, 93.9)
s130centr <- c(95.0, 95.3, 95.3, 94.7, 94.9, 94.9, 94.9, 96.6)
s150centr <- c(95.1, 94.9, 94.3, 94.9, 96.1, 94.2, 94.8, 94.5)
s200centr <- c(95.6, 94.2, 95.4, 95.5, 95.6, 94.5, 95.2, 95.3)
```

Now with the normal codebook, same settings, but on CIFAR10

```
cbcifar10centr10 <- c(22.7, 18.1, 25.6, 25.6, 21.6, 16.4, 23.0, 26.0)
cbcifar10centr30 <- c(26.6, 36.6, 40.3, 40.3, 37.8, 39.8, 39.0, 39.0)
cbcifar10centr50 <- c(42.7, 44.1, 41.1, 42.9, 46.4, 44.0, 42.9, 42.9)
cbcifar10centr70 <- c(44.5, 45.5, 45.5, 44.8, 41.9, 44.5, 41.9, 45.0)
cbcifar10centr90 <- c(46.0, 43.7, 43.7, 49.0, 44.3, 40.6, 45.5, 49.0)
cbcifar10centr110 <- c(46.9, 45.3, 45.4, 47.6, 45.8, 46.6, 45.2, 46.2)
cbcifar10centr130 <- c(47.1, 46.2, 46.7, 48.3, 48.1, 46.6, 47.5, 46.5)
cbcifar10centr150 <- c(47.5, 47.5, 47.5, 47.5, 48.1, 46.2, 50.2, 50.2)
cbcifar10centr200 <- c(47.3, 46.1, 46.1, 48.4, 48.3, 46.2, 48.7, 48.3)
```

And now with the deep codebook on cifar10

```
dcbcifar10centr10 <- c(15.8, 24.9, 21.3, 20.3, 20.7, 20.5, 18.7, 16.9)
dcbcifar10centr30 <- c(36.3, 34.9, 36.0, 35.6, 34.2, 36.6, 35.8, 31.2)
dcbcifar10centr50 <- c(38.0, 38.8, 36.3, 38.6, 39.4, 40.7, 41.5, 41.0)
dcbcifar10centr70 <- c(42.5, 38.8, 42.7, 39.3, 39.4, 43.4, 38.8, 39.6)
dcbcifar10centr90 <- c(43.8, 41.6, 40.2, 38.4, 42.1, 37.5, 40.8, 41.5)
dcbcifar10centr110 <- c(39.7, 40.1, 43.3, 42.0, 41.4, 40.5, 42.6, 41.9)
dcbcifar10centr130 <- c(42.3, 44.8, 42.9, 44.1, 41.0, 44.2, 43.1, 43.2)
dcbcifar10centr150 <- c(43.4, 42.7, 40.4, 43.3, 41.6, 42.3, 44.6, 41.5)
dcbcifar10centr200 <- c(43.2, 39.3, 37.6, 40.1, 42.7, 42.3, 37.6, 40.0)
```

Nr of kmeans iterations

I'm curious whether at a higher number of centroids, more kmeans iterations should be used. Small test: Deep codebook, cifar10, 130 centroids with 40 kmeans iterations, vs the same, but with 20 iterations.

```r
iter20 <- dcbcifar10centr130
iter40 <- c(44.6, 44.2, 42.5, 41.5, 40.1, 42.1, 42.9, 42.9)

t.test(iter20, iter40, paired = FALSE, alternative = "two.sided")
```

```
##
##  Welch Two Sample t-test
##
## data:  iter20 and iter40
## t = 0.9056, df = 13.577, p-value = 0.3809
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.8251246  2.0251246
## sample estimates:
## mean of x mean of y
##      43.2      42.6
```

Conclusion: There is no significant diffenrence in score by using more iterations of kmeans, using the deep codebook.

Lets see for the normal codebook:

```r
iter20 <- cbcifar10centr130
iter40 <- c(49.3, 45.6, 45.2, 47.9, 48.6, 46.6, 45.4, 45.4)

t.test(iter20, iter40, paired = FALSE, alternative = "two.sided")
```

```
##
##  Welch Two Sample t-test
##
## data:  iter20 and iter40
## t = 0.5875, df = 9.986, p-value = 0.5699
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.047464  1.797464
## sample estimates:
## mean of x mean of y
##    47.125    46.750
```

Conclusion: There is no significant diffenrence in score by using more iterations of kmeans, using the normal codebook.

other HOG cell configuration

Now, lets try other HOG cells. Instead of 9 4x4 cells, lets use 4 6x6 cells. With kmeans using 20 iters and 130 centroids with the normal codebook.

```r
size6deep <- c(37.8, 38.4, 41.1, 35.6, 37.8, 39.9, 39.3, 40.4)
size4deep <- dcbcifar10centr130
t.test(size4deep, size6deep, paired = FALSE, alternative = "two.sided")
```

```
##
##  Welch Two Sample t-test
##
## data:  size4deep and size6deep
## t = 5.8529, df = 12.361, p-value = 6.928e-05
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  2.775194 6.049806
## sample estimates:
## mean of x mean of y
##   43.2000   38.7875
```

The deep codebook seems to decrease performance when 6x6 cells are used.

```
size6normal <- c(50.6, 47.4, 51.5, 51.5, 48.7, 52.3, 52.5, 49.5)
mean(size6normal)
```

```
## [1] 50.5
```

```
size4normal <- cbcifar10centr130
t.test(size4normal, size6normal, paired = FALSE, alternative = "two.sided")
```

```
##
##  Welch Two Sample t-test
##
## data:  size4normal and size6normal
## t = -4.836, df = 9.45, p-value = 0.0008057
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -4.94235 -1.80765
## sample estimates:
## mean of x mean of y
##    47.125    50.500
```

But the normal codebook seems to increase performance.

Lets see whether it gets higher with a lower learningRate.

```
lr_2e_6 <- c(48.6, 53.0, 48.4, 51.3, 50.7, 50.7, 48.2, 48.4)
lr_4e_6 <- size6normal
t.test(lr_2e_6, lr_4e_6, paired = FALSE, alternative = "two.sided")
```

```
##
##  Welch Two Sample t-test
##
## data:  lr_2e_6 and lr_4e_6
## t = -0.6552, df = 13.991, p-value = 0.523
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.510869  1.335869
## sample estimates:
## mean of x mean of y
##   49.9125   50.5000
```

It seems not..

Idea for tweaking the centroid - patch similarity

Let's introduce another centroids assingment function:

$$Activation_C(Patch) = max(0, \delta * mean_{dist} - distance)$$

If $0 < delta < 1$, then the function becomes "softer". If $\delta = 1$, the function remains the same as the original. If $\delta > 1$, the function becomes "harder".

With delta = 0.8: Experiment currently running on ALICE