UNIVERSITY OF HELSINKI, DEPARTMENT OF COMPUTER SCIENCE

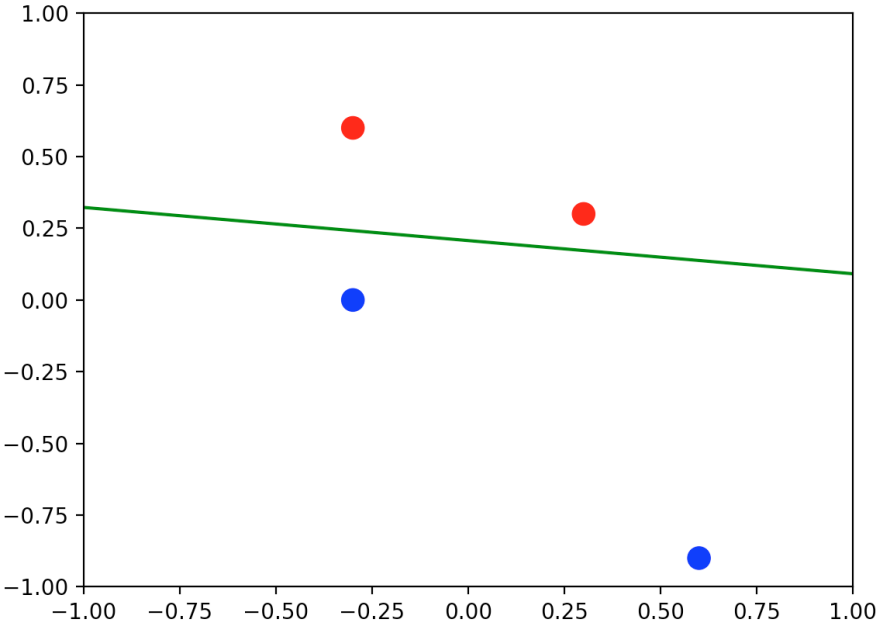
IML 2016: Renewal exam project

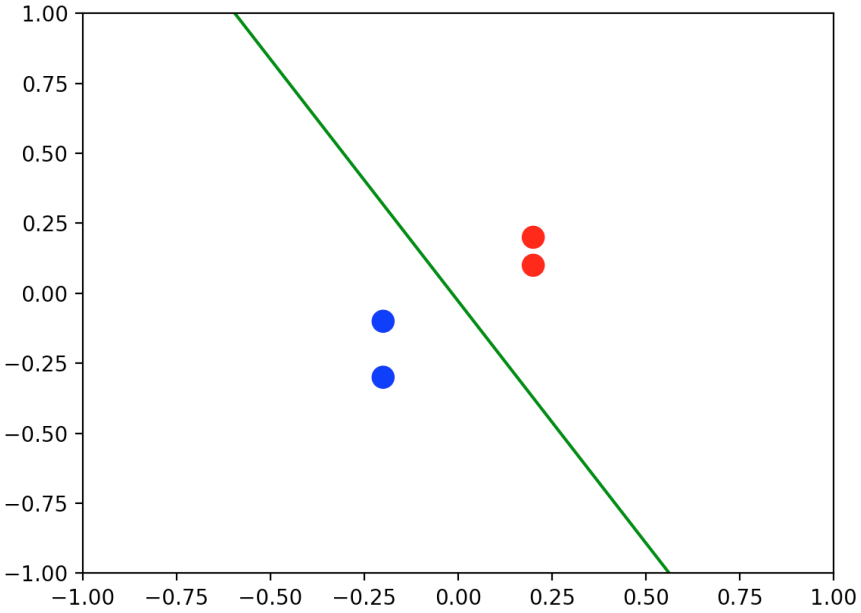
Jerry Mesimaki, 014141395

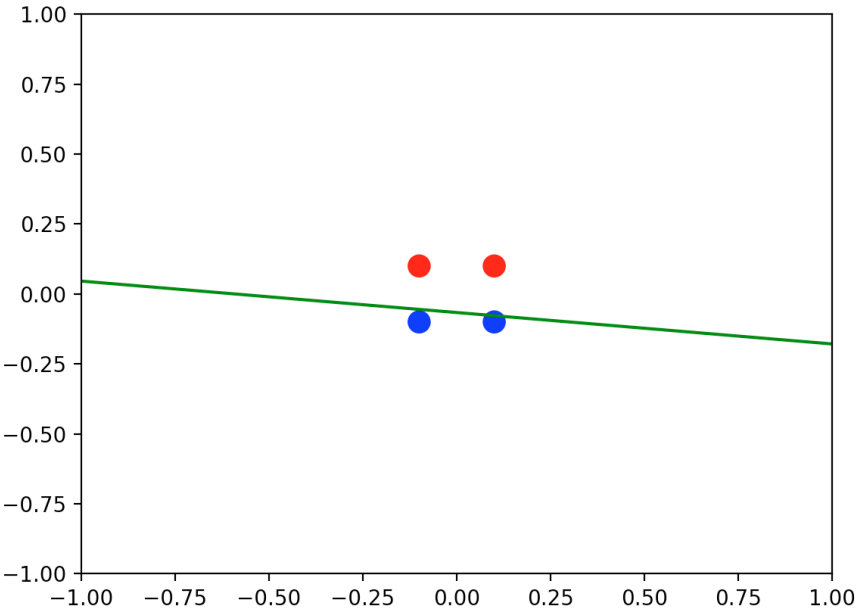
January 20, 2017

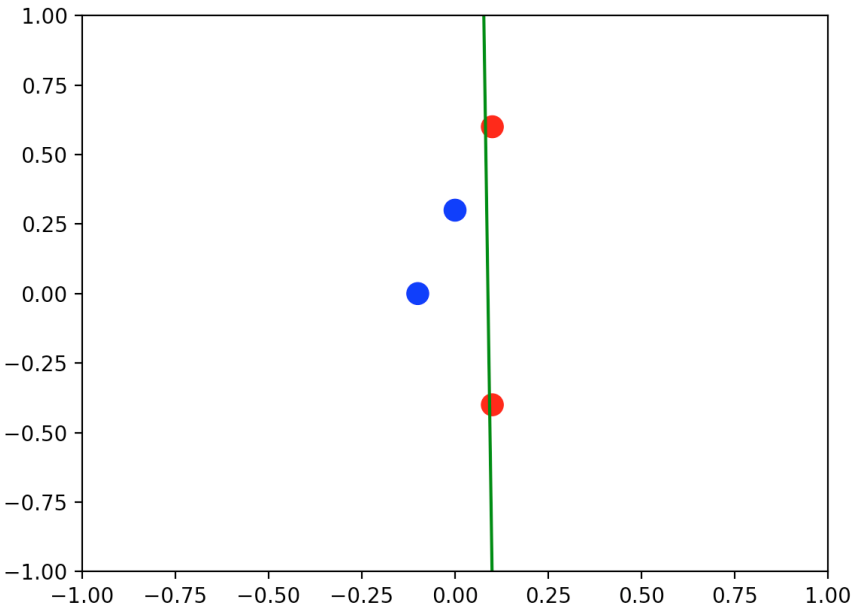
1 PERCEPTRON MNIST

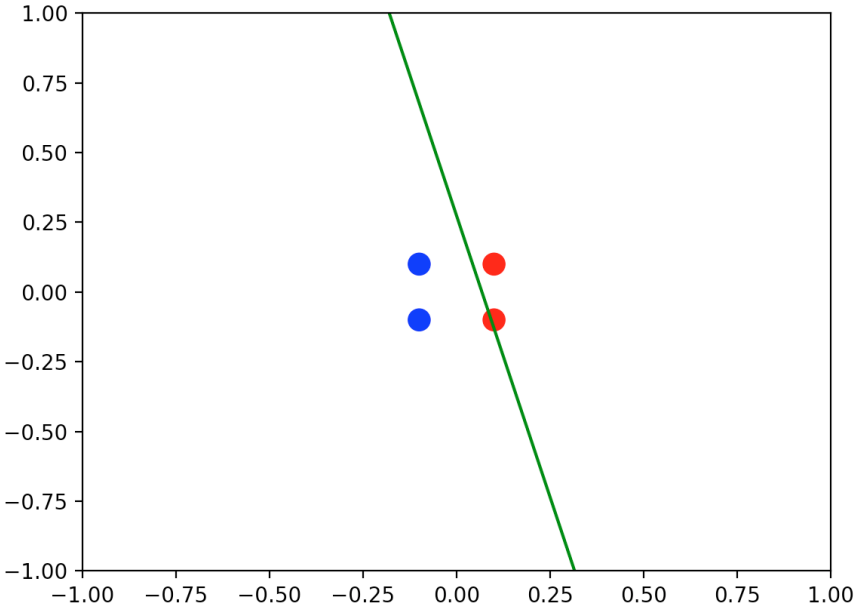
(a) I initialized the points to take x and y values between 0 and 1. A third dimension was also added (initially 1 for all vectors) in order to introduce the bias term. Following images show the two classes, along with a linear decision boundary separating them. No images of cases where different classes are given for two datapoints residing on the same diagonal are present because the algorithm doesn't converge on those.


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

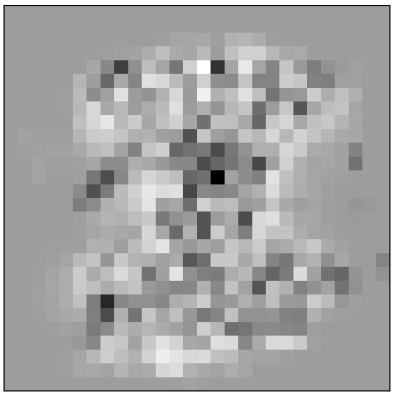

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00



(b) I started out with the assumption that a linear decision boundary between 0s and 1s can be found, and that the classifier should be quite accurate. The idea was to assign all the 1s with a label 1 and all the 0s with a label -1. A basic perceptron managed to classify 98.52% with a correct label when 0s and 1s of the first 2500 images were used as a training set and the 0s and 1s of the second 2500 images as a test set. It took 66587 iterations to get the perceptron to converge. The image drawn from the weight matrix is given below. It looks nothing like either of the numbers. I think that this implies it is not biased towards either number.



The algorithm is rather slow at the time of writing and more speed could probably be gained by tweaking the learning rate parameter and utilizing numpy to its fullest potential.

2 NAIVE BAYES 20NG

(a) The data was taken from a Python library called sklearn which offers the assignment's set of documents through a handy function called fetch_20newsgroups. I divided it into the training and test sets as per instructions.

(b) Lists for all the most occurring words of each newsgroup can be found from the folder: parsed_data with the name of the newsgroup as the name of the file. I refrain from copypasting those lists here to protect the sanity of myself and the reader. Prior probabilities for each newsgroup given below. For Laplace smoothing I added 0.0000001 for each word's number of occurrences which is not really possible but it seemed to yield good results. Had the value been 1, so much noise would've been added to the dataset that it would've taken a lot longer to calculate and it also seemed to negatively affect the accuracy.

(c) The results are surprisingly good and it got me wondering whether the algorithm is correct. I spent quite some time looking it over and found no clear errors. The Naive Bayes classifier classified 88% correct, albeit rather slowly. Running the algorithm took almost 45 minutes. In the confusion matrix there were no candidates for groups that would've really gotten mixed up with each other. Confusion matrix given below. I used the logarithmic scale for this assignment.

```
[   [44, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 2],
    [0, 47, 0, 1, 0, 7, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0],
    [0, 3, 41, 4, 2, 5, 1, 0, 0, 0, 0, 2, 0, 0, 1, 0, 0, 0, 0, 0],
    [0, 2, 2, 42, 2, 2, 0, 0, 0, 0, 0, 5, 2, 0, 1, 0, 0, 0, 0, 0],
    [0, 3, 1, 1, 43, 1, 0, 0, 0, 0, 0, 3, 2, 2, 0, 1, 0, 0, 0, 0],
    [0, 1, 0, 0, 0, 55, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0],
    [0, 1, 0, 2, 2, 1, 40, 1, 0, 0, 1, 2, 0, 1, 2, 0, 0, 4, 1, 0],
    [0, 0, 0, 0, 0, 0, 0, 54, 0, 0, 0, 0, 1, 1, 1, 0, 2, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 56, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 56, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 57, 0, 0, 0, 1, 0, 0, 0, 1, 0],
    [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 58, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 2, 0, 3, 0, 2, 2, 0, 0, 0, 0, 2, 48, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 56, 0, 0, 0, 2, 0, 0],
    [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 57, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 58, 0, 1, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 54, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 56, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 3, 41, 0],
    [3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 1, 1, 2, 23]]
```

3 AGGLOMERATIVE HIERARCHICAL CLUSTERING MOVIELENS

(a) I constructed a one-dimensional dataset consisting of the following points:

(a, 1) (b, 2) (c, 4) (d, 7) (e, 8) (f, 12) (g, 13) (h, 14) (i, 16) (j, 18) (k, 20).

Output samples of running a basic agglomerative hierarchical clustering algorithm with both single linkage and complete linkage distance functions below. (I later on read that part b was supposed to be this kind of a sanity check but let's consider this as a demonstration for 1d data with objects).

Single linkage

```
honeybook:iml-renewal-project jjoonia$ python agghierclust.py
Merging two clusters at distance 1
Merging
Cluster has the following points
a: 1
Cluster has the following points
b: 2
Merging two clusters at distance 1
Merging
Cluster has the following points
d: 7
Cluster has the following points
e: 8
Merging two clusters at distance 1
.
.
.
Merging two clusters at distance 3
Merging
Cluster has the following points
d: 7
e: 8
Cluster has the following points
c: 4
a: 1
b: 2
Merging two clusters at distance 4
Merging
Cluster has the following points
h: 14
f: 12
g: 13
k: 20
i: 16
j: 18
Cluster has the following points
d: 7
e: 8
c: 4
a: 1
b: 2
```

Complete linkage

```
honeybook:iml-renewal-project jjoonia$ python agghierclust.py
Merging two clusters at distance 1
Merging
Cluster has the following points
a: 1
Cluster has the following points
b: 2
Merging two clusters at distance 1
Merging
Cluster has the following points
d: 7
Cluster has the following points
e: 8
.
.
.
Merging
Cluster has the following points
j: 18
k: 20
Cluster has the following points
f: 12
g: 13
h: 14
i: 16
Merging two clusters at distance 19
Merging
Cluster has the following points
d: 7
e: 8
c: 4
a: 1
b: 2
Cluster has the following points
j: 18
k: 20
f: 12
g: 13
h: 14
i: 16
```

(b) I'm rather confused of why to use the negative euclidean distance. This means that two clusters furthest away from each other are considered "closest" since the distance value is smaller than for any other pair of clusters. However I assumed that this was not the intention and used the absolute values when comparing the distances (i.e. normal euclidean distance). Now as per the instructions of the exercise, I initialized five points:

(1,1) (3,8) (8,1) (10,8) (12,11)

Here's the output which tells us that the algorithm is working as expected:

```
honeybook:iml-renewal-project jjoonia$ python agghierclust.py
Running agglomerative hierarchical clustering with
single linkage while using negative euclidean distance measure
Merging two clusters at distance 3.60555127546
Merging
Cluster contains the following points:
(10, 8)
Cluster contains the following points:
(12, 11)
Merging two clusters at distance 7.0
Merging
Cluster contains the following points:
(1, 1)
Cluster contains the following points:
(8, 1)
Merging two clusters at distance 8.60232526704
Merging
Cluster contains the following points:
(3, 8)
Cluster contains the following points:
(1, 1)
(8, 1)
Merging two clusters at distance 14.8660687473
Merging
Cluster contains the following points:
(10, 8)
(12, 11)
Cluster contains the following points:
(3, 8)
(1, 1)
(8, 1)
```

(c)

(d) Here is the Jaccard coefficient matrix for the 20 movies chosen. I tried to pick a couple of genres such as cartoons, family friendly movies, drama, and sci-fi. Some of the movies seem to be very well related through the coefficient, such as Blade Runner and 2001: A Space Odyssey with a coefficient of 0.508. However one would think that Toy Story and Space Jam would also have a somewhat high coefficient since they are both cartoon-centric movies and they were released within one year from each other, the coefficient still remains rather low: 0.179.

```
1.000 0.217 0.194 0.403 0.199 0.413 0.347 0.358 0.354 0.319 0.215 0.197 0.505 0.305 0.180 0.017 0.023 0.264 0.132 0.377
0.217 1.000 0.186 0.280 0.474 0.280 0.315 0.385 0.336 0.266 0.268 0.152 0.270 0.137 0.173 0.020 0.060 0.237 0.100 0.298
0.194 0.186 1.000 0.333 0.157 0.334 0.229 0.251 0.336 0.370 0.269 0.163 0.193 0.249 0.132 0.020 0.024 0.322 0.131 0.328
0.403 0.280 0.333 1.000 0.255 0.531 0.361 0.472 0.490 0.420 0.328 0.196 0.379 0.298 0.154 0.017 0.027 0.387 0.150 0.483
0.199 0.474 0.157 0.255 1.000 0.258 0.314 0.386 0.344 0.249 0.211 0.166 0.270 0.115 0.183 0.028 0.069 0.236 0.102 0.294
0.413 0.280 0.334 0.531 0.258 1.000 0.494 0.528 0.461 0.487 0.288 0.157 0.391 0.241 0.169 0.012 0.023 0.374 0.156 0.609
0.347 0.315 0.229 0.361 0.314 0.494 1.000 0.508 0.383 0.369 0.211 0.149 0.336 0.163 0.242 0.013 0.033 0.281 0.130 0.481
0.358 0.385 0.251 0.472 0.386 0.528 0.508 1.000 0.481 0.394 0.255 0.180 0.388 0.186 0.188 0.018 0.040 0.344 0.136 0.533
0.354 0.336 0.336 0.490 0.344 0.461 0.383 0.481 1.000 0.508 0.343 0.197 0.363 0.218 0.143 0.017 0.027 0.468 0.154 0.445
0.319 0.266 0.370 0.420 0.249 0.487 0.369 0.394 0.508 1.000 0.272 0.169 0.286 0.227 0.114 0.022 0.029 0.459 0.128 0.495
0.215 0.268 0.269 0.328 0.211 0.288 0.211 0.255 0.343 0.272 1.000 0.125 0.224 0.280 0.142 0.025 0.017 0.323 0.088 0.280
0.197 0.152 0.163 0.196 0.166 0.157 0.149 0.180 0.197 0.169 0.125 1.000 0.194 0.179 0.105 0.045 0.067 0.164 0.225 0.164
0.505 0.270 0.193 0.379 0.270 0.391 0.336 0.388 0.363 0.286 0.224 0.194 1.000 0.283 0.217 0.017 0.028 0.246 0.132 0.376
0.305 0.137 0.249 0.298 0.115 0.241 0.163 0.186 0.218 0.227 0.280 0.179 0.283 1.000 0.125 0.023 0.030 0.248 0.137 0.220
0.180 0.173 0.132 0.154 0.183 0.169 0.242 0.188 0.143 0.114 0.142 0.105 0.217 0.125 1.000 0.028 0.034 0.127 0.058 0.163
0.017 0.020 0.020 0.017 0.028 0.012 0.013 0.018 0.017 0.022 0.025 0.045 0.017 0.023 0.028 1.000 0.044 0.013 0.043 0.016
0.023 0.060 0.024 0.027 0.069 0.023 0.033 0.040 0.027 0.029 0.017 0.067 0.028 0.030 0.034 0.044 1.000 0.026 0.087 0.025
0.264 0.237 0.322 0.387 0.236 0.374 0.281 0.344 0.468 0.459 0.323 0.164 0.246 0.248 0.127 0.013 0.026 1.000 0.118 0.358
0.132 0.100 0.131 0.150 0.102 0.156 0.130 0.136 0.154 0.128 0.088 0.225 0.132 0.137 0.058 0.043 0.087 0.118 1.000 0.140
0.377 0.298 0.328 0.483 0.294 0.609 0.481 0.533 0.445 0.495 0.280 0.164 0.376 0.220 0.163 0.016 0.025 0.358 0.140 1.000
```

As I presumed, clustering did cluster the aforementioned sci-fi movies together early on. Also a set of family friendly movies (Forrest Gump, E.T.) started to form. I wouldn't really recommend using this as a recommendation algorithm alone with this implementation because the results are not really good enough. Cluster merging output given below:

```
Merging the following movies
    Forrest Gump (1994)
    E.T. the Extra-Terrestrial (1982)
Merging the following movies
    Jurassic Park (1993)
    Forrest Gump (1994)
    E.T. the Extra-Terrestrial (1982)
Merging the following movies
    Pulp Fiction (1994)
    Jurassic Park (1993)
    Forrest Gump (1994)
    E.T. the Extra-Terrestrial (1982)
Merging the following movies
    Blade Runner (1982)
    2001: A Space Odyssey (1968)
Merging the following movies
    Lion King, The (1994)
```

```
    Pulp Fiction (1994)
    Jurassic Park (1993)
    Forrest Gump (1994)
    E.T. the Extra-Terrestrial (1982)
Merging the following movies
    Toy Story (1995)
    Mission: Impossible (1996)
Merging the following movies
    Blade Runner (1982)
    2001: A Space Odyssey (1968)
    Lion King, The (1994)
    Pulp Fiction (1994)
    Jurassic Park (1993)
    Forrest Gump (1994)
    E.T. the Extra-Terrestrial (1982)
Merging the following movies
    GoldenEye (1995)
    Stargate (1994)
Merging the following movies
    Brazil (1985)
    Blade Runner (1982)
    2001: A Space Odyssey (1968)
    Lion King, The (1994)
    Pulp Fiction (1994)
    Jurassic Park (1993)
    Forrest Gump (1994)
    E.T. the Extra-Terrestrial (1982)
Merging the following movies
    Toy Story (1995)
    Mission: Impossible (1996)
    Brazil (1985)
    Blade Runner (1982)
    2001: A Space Odyssey (1968)
    Lion King, The (1994)
    Pulp Fiction (1994)
    Jurassic Park (1993)
    Forrest Gump (1994)
    E.T. the Extra-Terrestrial (1982)
Merging the following movies
    GoldenEye (1995)
    Stargate (1994)
    Toy Story (1995)
    Mission: Impossible (1996)
    Brazil (1985)
```

```
    Blade Runner (1982)
    2001: A Space Odyssey (1968)
    Lion King, The (1994)
    Pulp Fiction (1994)
    Jurassic Park (1993)
    Forrest Gump (1994)
    E.T. the Extra-Terrestrial (1982)
Merging the following movies
    Taxi Driver (1976)
    GoldenEye (1995)
    Stargate (1994)
    Toy Story (1995)
    Mission: Impossible (1996)
    Brazil (1985)
    Blade Runner (1982)
    2001: A Space Odyssey (1968)
    Lion King, The (1994)
    Pulp Fiction (1994)
    Jurassic Park (1993)
    Forrest Gump (1994)
    E.T. the Extra-Terrestrial (1982)
Merging the following movies
    Reservoir Dogs (1992)
    Taxi Driver (1976)
    GoldenEye (1995)
    Stargate (1994)
    Toy Story (1995)
    Mission: Impossible (1996)
    Brazil (1985)
    Blade Runner (1982)
    2001: A Space Odyssey (1968)
    Lion King, The (1994)
    Pulp Fiction (1994)
    Jurassic Park (1993)
    Forrest Gump (1994)
    E.T. the Extra-Terrestrial (1982)
.
.
.
Merging the following movies
    Home Alone 3 (1997)
    Blues Brothers 2000 (1998)
    Space Jam (1996)
    Trainspotting (1996)
```

Reservoir Dogs (1992)
Taxi Driver (1976)
GoldenEye (1995)
Stargate (1994)
Toy Story (1995)
Mission: Impossible (1996)
Brazil (1985)
Blade Runner (1982)
2001: A Space Odyssey (1968)
Lion King, The (1994)
Pulp Fiction (1994)
Jurassic Park (1993)
Forrest Gump (1994)
E.T. the Extra-Terrestrial (1982)
Starship Troopers (1997)
Good Will Hunting (1997)

REFERENCES