
Task 1

Code completed in decision_tree.py and uci_data.py.

Full log files are included in a logs directory because why not. Logging has been disabled in the code but can be enabled by uncommenting in get_log_file() and print_and_log().

The specified log files without the excluded data are included in the root directory of the zip. Log file names are formatted:

<training_file>-<test_file>-<option>-<pruning_thr>.log

The specific files are:

pendigits_training.txt-pendigits_test.txt-1-50.log

pendigits_training.txt-pendigits_test.txt-3-50.log

pendigits_training.txt-pendigits_test.txt-optimal-50.log

b.

Test results (option, pruning_thr):

- Pendigits
  - Optimal, 1: .9182
  - Optimal, 50: .8382
  - 1, 1: .8458
  - 1, 50: .6641
  - 3, 1: .8954
  - 3, 50: .8228
  - 15, 1: .9592 .9534 .9521 .9548 .9529…
  - 15, 50: .8931
- Yeast
  - Optimal, 1: .5062
  - Optimal, 50: .5496
  - 1, 1: .4256
  - 1, 50: .4587
  - 3, 1: .4473
  - 3, 50: .5372
  - 15, 1: .5930 .5813 .5647 .5668 .5752…
  - 15, 50: .5764 .5579
- Satellite

- Optimal, 1: .8385
- Optimal, 50: .8200
- 1, 1: .8203
- 1, 50: .7455
- 3, 1: .8339
- 3, 50: .8275
- 15, 1: .8990 .8932 .8917 .8950 .8963…
- 15, 50: .8535

Seems like accuracy goes up as the number of trees goes up and the threshold goes down. Out of the hyperparameters that are normally applied to this assignment, having 15 random trees and a threshold of 1 gives the best accuracy.

Additionally, I tested a forest with 50 trees and a threshold of 1 and achieved an accuracy of .9623 on the pendigits dataset. So, it looks like you get better accuracy asymptotically as the number of trees increases.

c.

As stated previously, accuracy seems to increase as the number of trees increases. In order to have some tests to compare, I'll only be testing on the previously shown option, and pruning threshold. I'll show the results for a large forest at the end.

The first change I made was to always include an optimal tree as the first tree in a decision forest. I additionally upped the number for iteration to choose the best threshold from 50 to 100. You can see the accuracy increase from the number of threshold iterations in the optimal results. The first tree always being optimal should only really effect small forests. The smaller effected forests have been highlighted.

Test results (option, pruning_thr):

- Yeast
  - Optimal, 1: .5186
  - Optimal, 50: .5599
  - 1, 1: .5186
  - 1, 50: .5599
  - 3, 1: .4955
  - 3, 50: .5723
  - 15, 1: .5651 .5706
  - 15, 50: .5868 .5620

The second change I made was to add an entropy threshold. Instead of always having this threshold set to 0 (all the same class). It is now a variable. This decreases processing time and increases accuracy for datasets that benefit from thresholds. This change should benefit the

Yeast dataset but decrease accuracy for the pendigits dataset. This number should be tweaked depending on the dataset.

Test results (option, pruning_thr, entropy_thr):

- Yeast (increases accuracy by ~.05)
  - Optimal, 1, .1: .5186
  - …
  - Optimal, 1, 1.5: .6105
  - …
  - Optimal, 1, 1.7: .5176
  - 15, 1, 1.5: .6116 .6033 .5971
  - 15, 50, 1.5: .6116 .6178 .6074 .6136
- Pendigits
  - Optimal, 1, 0: .9185
  - Optimal, 1, .1: .9174
  - Optimal, 1, .2: .9094
  - Optimal, 1, .3: .9022
  - Optimal, 1, .4: .8988
  - Optimal, 1, .5: .8802
  - …
  - 15, 1, 0: .9520 .9536 .9555
- Satellite
  - Optimal, 1, 0: .8460
  - …
  - Optimal, 1, 0.4: .8570
  - …
  - Optimal, 1, 0.6: .8395
  - 15, 1, 0.4: .8918 .8870 .8865

My main change of always having at least one optimal tree increases the accuracy of smaller forests more than the accuracy of larger forests. Secondly by making the entropy threshold a variable, you can increase accuracy of datasets with noise. I observed that both the Yeast and Satellite datasets benefit from a variable entropy threshold for optimal runs. Additionally, I noticed that variable entropy_thr doesn't affect large forests much. The best results for the entropy_thr on optimal runs are shown above. Changing the entropy_thr has mixed results for different sized forests. Lastly larger forests asymptotically give better results, but take much longer to calculate.

Entropy_thr defaults to 0. You should call the new function as follows:

decision_tree(training_file, test_file, option, pruning_thr, entropy_thr)

Task 2

Example functions as shown in work:

Entropy(counts)

Weight(numerator, denominator)

a.

wait = 80 no_wait = 20 total = 100

entropy = -(80/100)*log2(80/100) - (20/100)*log2(20/100) = 0.7219

b.

entropy([80,20])

- weight(35,100)*entropy([20,15])

- weight(65,100)*entropy([60,5])

= 0.7219 - 0.35*0.9852 - 0.65*0.3912 = 0.2549

c.

Because we know all the examples at this node is during the weekend, every example will go to node H. If every decision goes to one node, there is no information gained. The equation results down to entropy1-0*entropy2-1*entropy1=0.

d.

A -> C -> F. The tree says the patron will wait.

e.

A -> B -> E -> H. The tree says the patron will not wait.

Task 3

A =

entropy([5,5])

- weight(3,10)*entropy([3,0])

- weight(4,10)*entropy([1,3])

- weight(3,10)*entropy([1,2])

= 1.0 - 0.3*0.0 - 0.4*0.8113 - 0.3*0.9183 = 0.4000

B =

entropy([5,5])

- weight(4,10)*entropy([1,3])

- weight(4,10)*entropy([3,1])

- weight(2,10)*entropy([1,1])

= 1.0 - 0.4*0.8113 - 0.4*0.8113 - 0.2*1.0 = 0.1510

C =

entropy([5,5])

- weight(5,10)*entropy([1,4])

- weight(4,10)*entropy([3,1])

- weight(1,10)*entropy([1,0])

= 1.0 - 0.5*0.7219 - 0.4*0.8113 - 0.1*0.0 = 0.3145

A achieves the highest information gain.

Task 4

a.

The lowest entropy is 0 when every example has the same label.

The highest entropy is 2.0 when every label has the same number of examples.

b.

The lowest possible information gain is 0 when the entropy of node N is 0.

The highest possible information gain is 2.0 when each class directly corelates to an attribute.

Task 5

If the accuracy of the classifier is 28% and if there are only two options for a game(win or lose) you can negate the predicted class from the classifier and invert the accuracy. By choosing the opposite of the classifier you can get an accuracy of 72%.