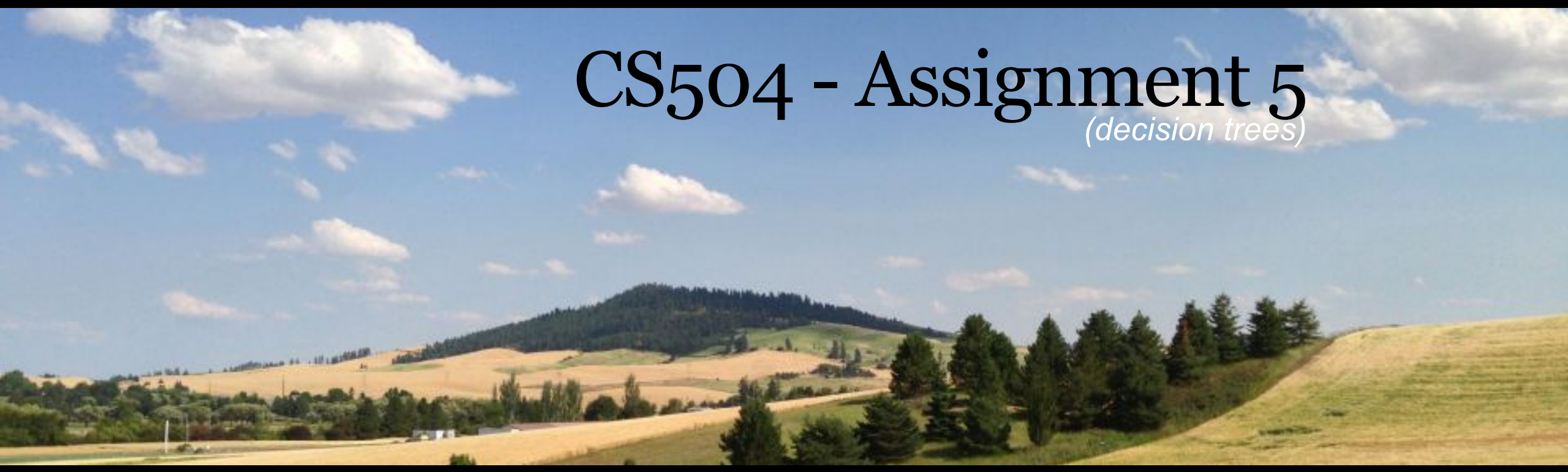

CS504 - Assignment 5

(decision trees)

REWARD: 200 points

DUE: Fri Apr 14 at 5PM PT

TEST DATA: testDataA5.tar
LIBRARIES: matRandTree.tar, matRandTree.zip

---

the task

Write a program called `id3` that builds a decision tree. Use the ID3 algorithm from the book to handle categorical data. The code should be in C++. You may use any STL features you wish provided they compile on the test system. You may also use the matrix library and the simple tree library. Both of which are above. See the count and subMatrix functions in the library for some coding help. The allow you to quickly manipulate sets of rows. Of course, you will have to convert your string based input into numeric matrices but that turns out to be easy. I will run your makefile to build the program `id3`. I will invoke `id3` to read data from standard in. The algorithm is simple and recursive but takes some thought.

input format

The program takes a data file from standard input in this format:

```
#Features
feature1 #values value1 value2 value3 ...
feature2 #values value1 value2 ...
feature3 #values value1 value2 value3 value4 ...
 ...
Ans ans1 ans2 ans3
#examples
f1 f2 f3... ans
f1 f2 f3... ans
f1 f2 f3... ans
f1 f2 f3... ans
f1 f2 f3... ans
f1 f2 f3... ans
f1 f2 f3... ans
```

#Features is the number of features **excluding the answer** which must be labeled Ans. Then comes (#Features+1) lines of features value descriptions the last of which is Ans. Each line is a feature name followed by the number of values of the feature and then the strings that represent the values for the feature. For example: `Temp 3 Cool Warm Hot`. std::cin works well for reading these strings! These are presented in the order of the columns in the remaining data which is the training cases. The training cases begin with the number of examples followed by the examples. Here is the file for a logical xor function:

```
2
A   2 False   True
B   2 False   True
Ans 2 False   True
4
False False False
False True  True
True  False True
True  True  False
```

The test data has many more examples.

output format

The output format is indented and looks like the following for the file `xor.in`

```
A=False
   B=False
      False
   B=True
      True
A=True
   B=False
      True
   B=True
      False
```

This is the output format used in the ktree class for the printEdge function.

implementation

The ID3 algorithm (p 254 in book) is recursive. It is simple and has several basic cases applied at each call the tree build routine:

- case 1: NOT IN BOOK: is there any data to process? (if data empty return node with string "NONE". See examples.)
- case 2: does all the data supplied have the same Ans?
- case 3: ran out of features so can't ask any more questions.
- case 4: find feature F with biggest information gain.
    - case 4a: NOT IN BOOK: if all feature values of F are the same then asking feature F does not split up the training data at all. Therefore, just recursively call with same data but without feature F.
    - case 4b: otherwise do the recursion in the book with an edge for each feature value.

See example outputs to see what is expected for output. My main program looked like:

```
int main()
{
    readProblem();

    Matrix availableFeatures(featureNames.size()-1, 1, "availableFeatures");
    availableFeatures.constantColRange(0, 0, 1);

    id3Build(dataToMatrix(), availableFeatures)->printEdge();
    return 0;
}
```

Example code and detailed output of internals can be found here: id3A5.tar, id3A5.zip

submission

Tar up the `id3` code with a makefile to build the program `id3`. Submit a tar for the assignment. Homework will be submitted as an **uncompressed** tar file to the homework submission page linked from the class web page. You can submit as many times as you like. **The LAST file you submit BEFORE the deadline will be the one graded.** For all submissions you will receive email at your uidaho.edu mail address giving you some automated feedback on the unpacking and compiling and running of code and possibly some other things that can be autotested.

Have fun.