Gradient Boosting

Weijie Zhao

09/26/2022

Why Gradient Boosting?

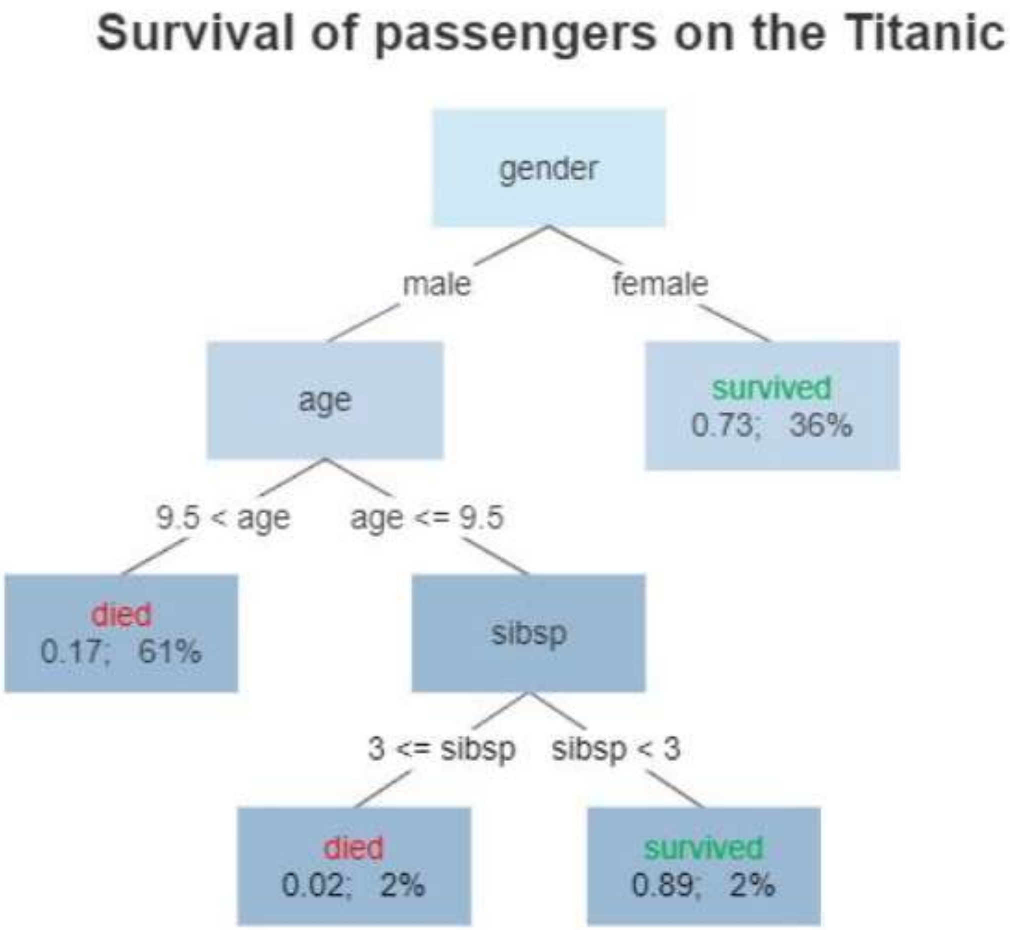
Machine Learning Challenge Winning Solutions

LightGBM is used in many winning solutions, but this table is updated very infrequently.

Place	Competition	Solution	Date
1st	M5 Forecasting - Uncertainty	link	2020.7
3rd	M5 Forecasting - Uncertainty	link	2020.7
3rd	ALASKA2 Image Steganalysis	link	2020.7
1st	M5 Forecasting - Accuracy	link	2020.6
2nd	COVID19 Global Forecasting (Week 5)	link	2020.5
3rd	COVID19 Global Forecasting (Week 5)	link	2020.5
1st	COVID19 Global Forecasting (Week 4)	link	2020.5
2nd	COVID19 Global Forecasting (Week 4)	link	2020.5
2nd	2019 Data Science Bowl	link	2020.1
3rd	RSNA Intracranial Hemorrhage Detection	link	2019.11
1st	IEEE-CIS Fraud Detection	link	2019.10
2nd	IEEE-CIS Fraud Detection	link	2019.10
2nd	Kuzushiji Recognition	link	2019.10
1st	Los Alamos National Laboratory Earthquake Prediction	link	2019.6
3rd	Los Alamos National Laboratory Earthquake Prediction	link	2019.6
1st	Santander Customer Transaction Prediction	link	2019.4
2nd	Santander Customer Transaction Prediction	link	2019.4
3rd	Santander Customer Transaction Prediction	link	2019.4
2nd	PetFinder.my Adoption Prediction	link	2019.4
1st	Google Analytics Customer Revenue Prediction	link	2019.3
1st	VSB Power Line Fault Detection	link	2019.3
5th	Elo Merchant Category Recommendation	link	2019.2

https://github.com/microsoft/LightGBM/blob/master/examples/README.md#machine-learning-challenge-winning-solutions

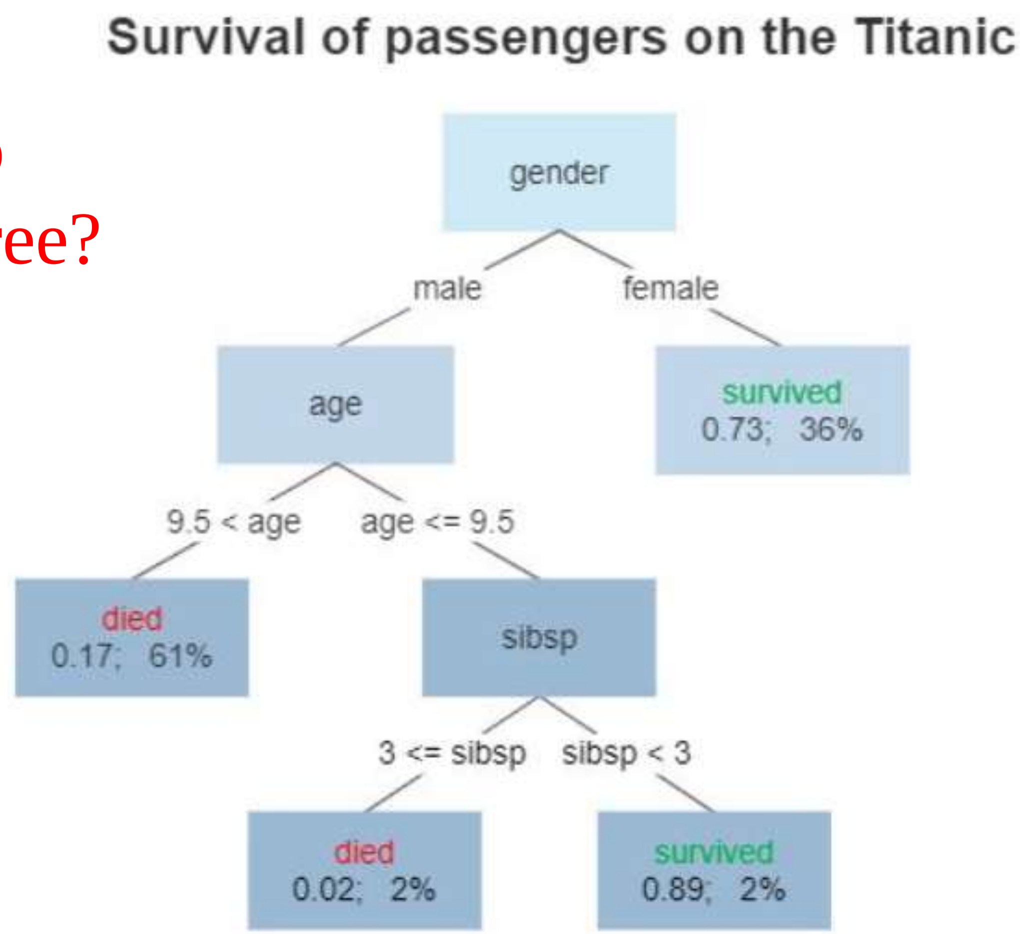
Decision Trees



https://en.wikipedia.org/wiki/Decision_tree_learning#/media/File:Decision_Tree.jpg

Decision Trees

Given a dataset, how to find the best decision tree?



https://en.wikipedia.org/wiki/Decision_tree_learning#/media/File:Decision_Tree.jpg

Tree Split Criteria

- Estimate of Positive Correctness

$$E_P = TP - FP$$

- Gini impurity

$$\mathrm{I}_G(p) = \sum_{i=1}^{J}\left(p_i \sum_{k\neq i} p_k\right) = \sum_{i=1}^{J} p_i(1-p_i) = \sum_{i=1}^{J}(p_i - p_i^2) = \sum_{i=1}^{J} p_i - \sum_{i=1}^{J} p_i^2 = 1 - \sum_{i=1}^{J} p_i^2$$

- MART gain

$$\frac{1}{s}\left[\sum_{i=1}^{s}(r_{i,k}-p_{i,k})\right]^2 + \frac{1}{N-s}\left[\sum_{i=s+1}^{N}(r_{i,k}-p_{i,k})\right]^2 - \frac{1}{N}\left[\sum_{i=1}^{N}(r_{i,k}-p_{i,k})\right]^2$$

$r_{i,k} = 1$ if $y_i = k$ and $r_{i,k} = 0$ otherwise

$p_{i,k} = \mathbf{Pr}\left(y_i = k|\mathbf{x}_i\right)$

Decision Trees

- Bagging
- Random Forest
- Gradient Boosting

Gradient Boosting

$$p_{i,k} = \mathbf{Pr}\left(y_i = k|\mathbf{x}_i\right) = \frac{e^{F_{i,k}(\mathbf{x_i})}}{\sum_{s=1}^{K} e^{F_{i,s}(\mathbf{x_i})}}, \qquad i = 1, 2, ..., N,$$

where $F_{i,k}(\mathbf{x}_i)$ is an additive model of M terms: $F^{(M)}(\mathbf{x}) = \sum_{m=1}^{M} \rho_m h(\mathbf{x}; \mathbf{a}_m)$,

$$L = \sum_{i=1}^{N} L_i, \qquad L_i = -\sum_{k=1}^{K} r_{i,k} \log p_{i,k}$$

where $r_{i.k} = 1$ if $y_i = k$ and $r_{i.k} = 0$ otherwise.

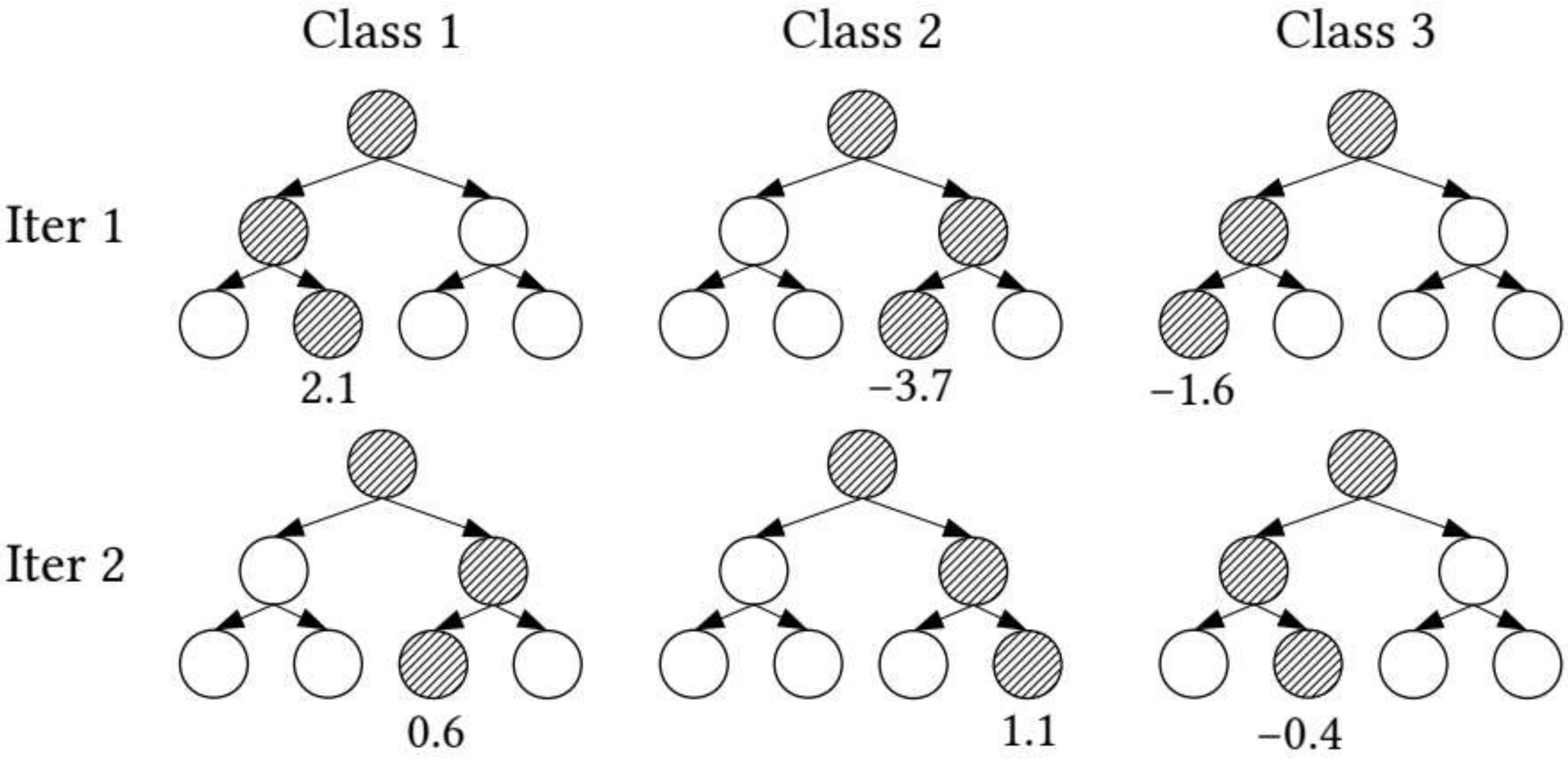
$$\frac{\partial L_i}{\partial F_{i,k}} = -\left(r_{i,k} - p_{i,k}\right), \qquad \frac{\partial^2 L_i}{\partial F_{i,k}^2} = p_{i,k}\left(1 - p_{i,k}\right).$$

Gradient Boosting

Algorithm 1 Robust LogitBoost. MART is similar, with the only difference in Line 4.

1: $F_{i,k} = 0$, $p_{i,k} = \frac{1}{K}$, $k = 1$ to K, $i = 1$ to N
2: **for** $m = 1$ to M **do**
3:     **for** $k = 1$ to K **do**
4:         $\{R_{j,k,m}\}_{j=1}^{J}$ = J-terminal node regression tree from $\{r_{i,k} - p_{i,k}, \quad \mathbf{x}_i\}_{i=1}^{N}$, with weights $p_{i,k}(1 - p_{i,k})$, using the tree split gain formula
5:         $\beta_{j,k,m} = \frac{K-1}{K}\frac{\sum_{\mathbf{x}_i \in R_{j,k,m}} r_{i,k} - p_{i,k}}{\sum_{\mathbf{x}_i \in R_{j,k,m}} (1-p_{i,k})p_{i,k}}$
6:         $f_{i,k} = \sum_{j=1}^{J} \beta_{j,k,m} 1_{\mathbf{x}_i \in R_{j,k,m}}, \qquad F_{i,k} = F_{i,k} + \nu f_{i,k}$
7:     **end for**
8:     $p_{i,k} = \exp(F_{i,k}) / \sum_{s=1}^{K} \exp(F_{i,s})$
9: **end for**

Gradient Boosting



$F_{2,1} = 2.1 + 0.6 = 2.7$ $F_{2,2} = -3.7 + 1.1 = -2.6$ $F_{2,3} = -1.6 - 0.4 = -2.0$

$p_{2,1} = 0.986$ $p_{2,2} = 0.005$ $p_{2,3} = 0.009$

Data Reading

- Matrix Format
- CSV
- LIBSVM

Data Reading

- Matrix Format
- CSV
- LIBSVM

- How should we store the parsed data?

Serialization/Deserialization

- Handwritten
  - Endian problem

Endianness

- Danny Cohen introduced the terms big-endian and little-endian into computer science for data ordering in an Internet Experiment Note published in 1980.
- In the 1726 novel Gulliver's Travels, he portrays the conflict between sects of Lilliputians divided into those breaking the shell of a boiled egg from the big end or from the little end. He called them the Big-Endians and the Little-Endians.
- Cohen makes the connection to Gulliver's Travels explicit in the appendix to his 1980 note.

https://en.wikipedia.org/wiki/Endianness

Serialization/Deserialization

- Handwritten
  - Endian problem

```
bool is_big_endian(void){
    union {
        uint32_t i;
        char c[4];
    } bint = {0x01020304};

    return bint.c[0] == 1;
}
```

Serialization/Deserialization

- Handwritten
  - Endian problem
- Protocol Buffer

conda install -c anaconda protobuf

protoc --cpp_out=DST_DIR --python_out=DST_DIR path/to/file.proto

```
syntax = "proto2";

package tutorial;

message Person {
  optional string name = 1;
  optional int32 id = 2;
  optional string email = 3;

  enum PhoneType {
    MOBILE = 0;
    HOME = 1;
    WORK = 2;
  }

  message PhoneNumber {
    optional string number = 1;
    optional PhoneType type = 2 [default = HOME];
  }

  repeated PhoneNumber phones = 4;
}

message AddressBook {
  repeated Person people = 1;
}
```

https://developers.google.com/protocol-buffers/

address.pb.h

```
// name
inline bool has_name() const;
inline void clear_name();
inline const ::std::string& name() const;
inline void set_name(const ::std::string& value);
inline void set_name(const char* value);
inline ::std::string* mutable_name();

// id
inline bool has_id() const;
inline void clear_id();
inline int32_t id() const;
inline void set_id(int32_t value);

// email
inline bool has_email() const;
inline void clear_email();
inline const ::std::string& email() const;
inline void set_email(const ::std::string& value);
inline void set_email(const char* value);
inline ::std::string* mutable_email();

// phones
inline int phones_size() const;
inline void clear_phones();
inline const ::google::protobuf::RepeatedPtrField< ::tutorial::Person_PhoneNumber >& phones() const;
inline ::google::protobuf::RepeatedPtrField< ::tutorial::Person_PhoneNumber >* mutable_phones();
inline const ::tutorial::Person_PhoneNumber& phones(int index) const;
inline ::tutorial::Person_PhoneNumber* mutable_phones(int index);
inline ::tutorial::Person_PhoneNumber* add_phones();
```

Protocol Buffer

- bool SerializeToString(string* output) const
- bool ParseFromString(const string& data)
- bool SerializeToOstream(ostream* output) const
- bool ParseFromIstream(istream* input)

- #include “…pb.h”
- g++ ……. -lprotobuf

HW2: Gradient Boosting

Weijie Zhao

10/03/2023

HW 2: Gradient Boosting

- Given a training data A, a testing data B, a target testing accuracy C
- Train a gradient boosting model and output predictions

- 10 test cases. Each case weights 1 pt.
- The compilation is considered failed if it does not finish in 5 minute.
- A test case is considered incorrect if it does not finish in 2 minutes.
- Correct GPU solutions will get 5 pts bonus.
- The summation of the execution time across 10 cases will be uses to rank correct solutions.

- Due: 10/16/2023 5:00 pm EDT

Testing Environment

- ssh yourusername@granger.cs.rit.edu
- Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz
- 48 threads in total (2 sockets, 12 cores per socket, 2 threads per core)
- 251 GB memory
- GPU: Tesla P4

- Testing limit:
  - 8 threads   taskset -c
  - 1 GPU

Output Format

- N lines
- Each line contains an integer
  - The predicted class for each instance

What Do We Need to Do?

- We are required to complete two scripts
- compiler.sh
  - it is executed once before the actual testing starts
- run.sh
  - it should takes 4 arguments
  - the first argument is the training data file name
  - the second argument is the testing data file name
  - the third argument is the target testing accuracy
  - the fourth one is the file name that you should write your results into

HW2

- bunzip2 mnist.bz2
- bunzip2 mnist.t.bz2

- bash run.sh <train> <test> <acc> <out>
  - bash run.sh mnist.t mnist.t 0.9 sample1.out
  - bash run.sh mnist mnist 0.9 sample2.out
  - bash run.sh mnist mnist.t 0.9 sample2.out

  …
  - bash run.sh mnist mnist.t 0.97 sample10.out
- We guarantee that testing data will not have more features than the training data.