BNPlib for density estimation:

A nonparametric C++ library

Bruno Guindani
Elena Zazzetti

POLITECNICO
MILANO 1863

https://github.com/poliprojects/BNPlib

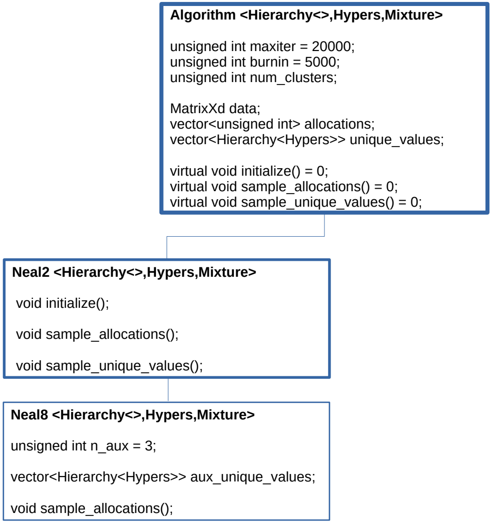
Algorithms

Commmon structure:

```
void step(){
    sample_allocations();
    sample_unique_values();
}

void run(BaseCollector* collector){
    print_startup_message();
    initialize();
    unsigned int iter = 0;
    while(iter < maxiter){
        step();
        if(iter >= burnin){
            save_state(collector, iter);
        }
    iter++;
    }
    print_ending_message();
}
```

Algoritmhs

- Structure: `Algorithm<Hierarchy<>, Hypers, Mixture>`
- Multivariate data supported



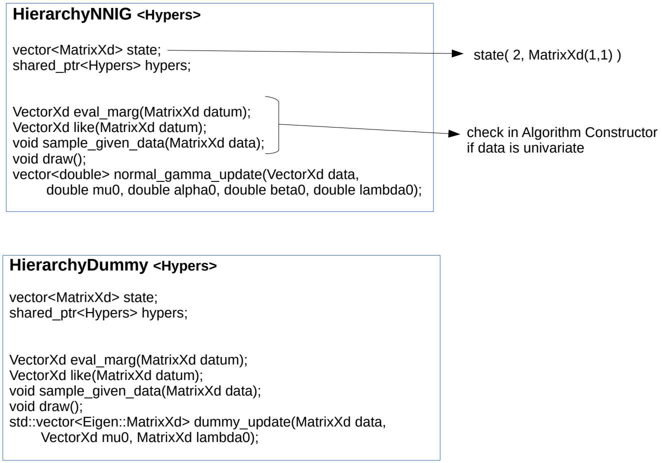
```
Algorithm <Hierarchy<>,Hypers,Mixture>

unsigned int maxiter = 20000;
unsigned int burnin = 5000;
unsigned int num_clusters;

MatrixXd data;
vector<unsigned int> allocations;
vector<Hierarchy<Hypers>> unique_values;

virtual void initialize() = 0;
virtual void sample_allocations() = 0;
virtual void sample_unique_values() = 0;
```

```
Neal2 <Hierarchy<>,Hypers,Mixture>

 void initialize();

 void sample_allocations();

 void sample_unique_values();
```

```
Neal8 <Hierarchy<>,Hypers,Mixture>

unsigned int n_aux = 3;

vector<Hierarchy<Hypers>> aux_unique_values;

void sample_allocations();
```

Hierarchies (`Hierarchy`)

- Template parameter for `Algorithm`: common interface needed



```
HierarchyNNIG <Hypers>

vector<MatrixXd> state;
shared_ptr<Hypers> hypers;


VectorXd eval_marg(MatrixXd datum);
VectorXd like(MatrixXd datum);
void sample_given_data(MatrixXd data);
void draw();
vector<double> normal_gamma_update(VectorXd data,
       double mu0, double alpha0, double beta0, double lambda0);
```

```
HierarchyDummy <Hypers>

vector<MatrixXd> state;
shared_ptr<Hypers> hypers;


VectorXd eval_marg(MatrixXd datum);
VectorXd like(MatrixXd datum);
void sample_given_data(MatrixXd data);
void draw();
std::vector<Eigen::MatrixXd> dummy_update(MatrixXd data,
       VectorXd mu0, MatrixXd lambda0);
```

Hyperparameters (`Hypers`)

- Template parameter for `Hierarchy`
- Used through a pointer to allow simultaneous update of several hierarchies

HypersFixedNNIG

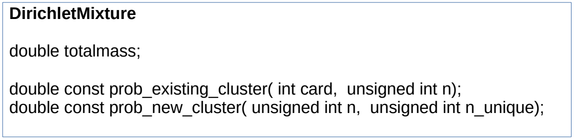
double mu0, lambda, alpha0, beta0;

HypersDummy

VectorXd mu0;
MatrixXd lambda0;

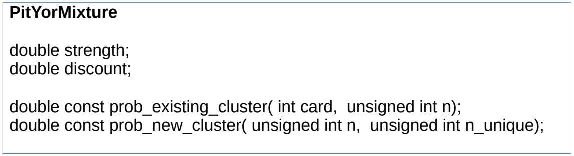
Mixtures (`Mixture`)

- Template parameter for `Algorithm`: common interface needed



DirichletMixture

```
double totalmass;

double const prob_existing_cluster( int card,  unsigned int n);
double const prob_new_cluster( unsigned int n,  unsigned int n_unique);
```



PitYorMixture

```
double strength;
double discount;

double const prob_existing_cluster( int card,  unsigned int n);
double const prob_new_cluster( unsigned int n,  unsigned int n_unique);
```

Factory

- To choose the `Algorithm` at runtime

```
template<class AbstractProduct, typename... Args>
class Factory {
private:
    std::map<Identifier, Builder> storage;
    //[...]
public:
    static Factory& Instance();
    std::unique_ptr<AbstractProduct> create_object(
        const Identifier &name, Args... args) const;
    void add_builder(const Identifier &name,
        const Builder &builder);
    //[...]
}
```

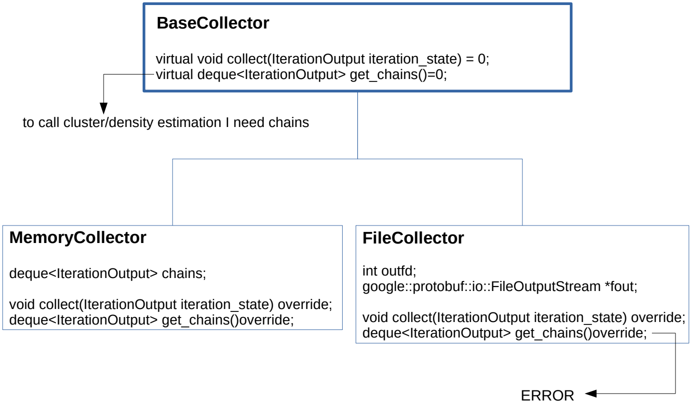
Protocol Buffers API aka `protobuf`

- For multivariate data storage
- Model in `output.proto`:

```
message Par_Col {
    repeated double elems = 1;
}
message Param {
    repeated Par_Col par_cols = 1;
}
message UniqueValues {
    repeated Param params = 1;
}
message IterationOutput {
    repeated int32 allocations = 1;
    repeated UniqueValues uniquevalues = 2;
}
message ChainOutput {
    repeated IterationOutput chain = 1;
}
```

Collectors

- To save data with the `protobuf` structures



BaseCollector
virtual void collect(IterationOutput iteration_state) = 0;
virtual deque<IterationOutput> get_chains()=0;
to call cluster/density estimation I need chains
MemoryCollector
deque<IterationOutput> chains;
void collect(IterationOutput iteration_state) override;
deque<IterationOutput> get_chains()override;
FileCollector
int outfd;
google::protobuf::io::FileOutputStream *fout;
void collect(IterationOutput iteration_state) override;
deque<IterationOutput> get_chains()override;
ERROR