Covetree Pseudocode / functional overview

Introduction:

CovTree implements a recursive bi-partitioning algorithm over a set of genomic positions, according to a set of features and samples (e.g. cell-types). Each feature assigns every genomic position a small, integral value generally representing a class. For example, the feature RNA-seq may assign one of 4 values {none=0, low read depth=1, medium read depth=2, high read depth=3} to each genomic position in a particular cell type. Every position must be assigned to exactly one value. Features may vary according to sample, or may be shared among all samples (having the same value for a genomic position in every cell-type). The figure of merit is the log likelihood, so at each step the single covariate partition that most improves the sum of post-split Insight2 likelihoods is selected.

Values in features are provided as either ordered implicitly or explicitly ordered. Implicitly ordered features (e.g. RNA-seq) maintain their ordering through a CovTree execution. Explicitly ordered covariates (e.g. Chromatin/ChromHMM class) are ordered monotonically according to increasing selective pressure (ρ) under Insight2 in a preliminary step before binary partitions are calculated.

At each recursion step CovTree exhaustively searches all ordered bipartitions separately in each feature. Thus, for the RNA-seq example mentioned above the 4 possible values for the feature admit 3 possible splits ({0}|{1,2,3}; {0,1}|{2,3}; {0,1,2}|{3}). Recursion terminates when no subdivisions are possible or the best possible subdivision produces a change in post-split log likelihood that is below a user-provided threshold. See figure 1 in the accompanying paper, or its preprint https://doi.org/10.1101/317719 "How Much Information is Provided by Human Epigenomic Data? An Evolutionary View" for a graphical description of this process.

Initially all features and value for each feature are considered. Recursion proceeds on each of the 2 sets of features produced by the previous iteration. Recursion is accomplished either via an internal stack that maintains a list of candidate feature-values, or the output of a feature subset, relying on an external process to allocate computational resources investigate a set of feature subset.

In the following description, CovTree code elements are indicated in `courier` while elements representing data elements are indicated in **boldface** .

Pseudocode / functional description:

Initialize runtime elements `runtimeElements::init()`

Process user arguments `covtreeArgs. covtreeArgs()`

Push user arguments, including current feature/value set onto `stack()`

Read feature (covariate) definitions, including feature names and set of possible values `covsetDef::read()`

Initiate a number of Insight2 processes, as request by the user. Each utilizes a single core and requires approximately 3GB of memory . `insightShellSet::serverInitialize()`

Create a mapping from feature values at each position to bits, thus a set of feature values may be represented as a long bit string. For performance, shared features are limited to a total of 64 values, and sample specific features are limited to a total of 64 values (128 total). `covDB_t::bitfieldDef_t::addField()`

Load the database of features values for each cell type `covDB_t::loadDB()`, this requires approximately 6GB of RAM for 115 cell-types, using 5 shared features with 22 values (5CDS+5Melt+4TFBS+4smRNA+4Splice) and 4 cell-type specific features with 35 values (25Chrom+4RNA+4DNASE+2WGBS).

Enter **execution loop**: while the **input stack** is not empty

Pop a set of arguments off the **stack** (includes universe of feature values). Arguments include Insight2 log likelihood for the set of all positions with active feature values (**parent log likelihood**)

Create hi/lo output directories

If there are explicitly ordered covariates, generate covariate orderings `main:generateOrderings()`

For each explicitly ordered **feature**

For each **value** presently active in **feature** (some hay have been excluded by previosu splits)

Identify all **genomic positions** associated with feature value, positions having a feature with value in N samples of M possible samples are provided a positional **weight** of N/M

Estimate and store Insight2 ρ for each such **set of weighted positions**

Next **value**

Reorder **values** in **feature** according to increasing ρ

Next explicitly ordered **feature**

Evaluate each possible partition of a single feature `main::evaluatePartitions()`

For each ordered **feature** with more than 2 active **values**

For each **value_i** < **value_max** in **feature**

Create value **subset A** (all active values <= **value_i**) and value **subset B** (all active values > **value_i**)

For **subset** in partition {**subset A**, **subset B**} `scorePair::GenerateCosets()`

Identify all **genomic positions** associated with feature values in subset. Positions having a **feature** with **value** in N **samples** of M possible **samples** are provided a positional **weight** of N/M

Estimate and store Insight2 parameters and **log likelihood** for each **set of weighted positions**
`runtimeElements::insight.jobInit()`

Next **subset**

Sum Insight2 log likelihoods for {**subset A**, **subset B**}, if each subset passed quality criteria (min # positions in each of A and B, min change in summed log likelihood from **parent log likelihood**); then save as partition **score** otherwise disregard **score**.

Next **value**

Sort partitions according to **score** , select partition with highest **score** to represent this **feature** . If there are any **score**s for **feature** , assign highest **score** to **feature**, and add score/feature pair to **feature list**.

Next ordered **feature**

Save results of each **subset**.

End loop // evaluate each partition

If **feature list** is not empty

Select **feature** with highest **score** to as **target** for this iteration of CovTree.
`bool main::sortPartitions().`

Generate 2 **argument sets** for new CovTree run based on **subset A** and **subset B**. Select output for each new CovTree run from hi/lo directories created previously.
`Main::saveResults()`

If **internal recursion** is selected, `main::processRecursion()`

push the 2 new **argument sets** on to **argument stack**,

otherwise,

write argument sets to external files and create semaphore file indicating completion (recursion will be initiated by external watcher thread using 1 computer for each of the 2 new subsets)

end if // **internal recursion**

end if // **feature list** not empty

end of loop // execution loop – terminates when **argument stack** is empty.

Terminate Insight2 processes `InsightShellSet:: serverTerminate()`