Parameter optimization via mlrMBO of the NicheNet method using Omnipath Interactions ==========
Alberto Valdeolivas: alberto.valdeolivas@bioquant.uni-heidelberg.de; Date: 29/02/2020

Abstract

This vignette shows how to optimice the parameters of the Nichenet method, i.e. PageRank parameter and source weights, using sets of interactions availabe in the **Omnipath** database.

The NicheNet Method

NicheNet (https://github.com/saeyslab/nichenetr) is a method to predict ligand-target links between interacting cells by combining their data with prior knowledge on signaling and gene regulatory networks (Browaeys et al 2019). **NicheNet** has already been applied to predict upstream niche signals driving Kupffer cell differentiation (Bonnardel et al. 2019).

NicheNet uses many different publically available resources to build a prior knowledge network. Their final integrated network is composed of three individual networks:

- A network of ligand-receptor interactions (Inter-cellular)
- A network of signaling interactions (Intra-cellular)
- A network of gene regulation (Intra-cellular)

The new version of the **Omnipath** (http://omnipathdb.org/) database contains curated interactions belonging to these three categories. One can therefore build an integrated network equivalent to the one used in **NicheNet** by only fetching the **Omnipath** webserver. This can significantly ease the integration of different databases, each one of them storing data in distint formats and whose interaction show different levels of reliability.

We therefore here optimice the parameters used in the **NicheNet's** algorithm, i.e. PageRank parameters and source weights, using interactions available in the **Omnipath** database.

```
library(OmnipathR)
library(nichenetr)
library(tidyverse)
library(mlrMBO)
library(parallelMap)
```

Fetching Omnipath interactions and transfoming their format

We first define a function to transform the format of interactions in the **Omnipath** to the one used in the **NicheNet** method:

```
interactionFormatTransf <- function(InputDf, InteractionType){

    OutputInt <- tibble(from = character(), to = character(),
        source = character(), database = character())

    n <- nrow(InputDf)
    sources <- dplyr::pull(InputDf, sources)
    sourceNodes <- dplyr::pull(InputDf, from)
    targetNodes <- dplyr::pull(InputDf, to)

    for (i in seq(n)){
        currentSources <- unlist(strsplit(sources[i],";"))
        for (j in seq(length(currentSources))){
            OutputInt <- add_row(OutputInt,
                from = sourceNodes[i] ,
```

```
                to = targetNodes[i],
                # source = paste(currentSources[j], InteractionType, sep="_"),
                source = currentSources[j],
                database = currentSources[j])
        }
    }

    return(OutputInt)
}
```

Generating the Omnipath ligand-receptor network

Omnipath possess a dedicated dataset storing these type of interactions (*LigrecExtra*). Therefore, we get these interactions:

```
## We remove self-interacitons and duplicated records
lr_Interactions_Omnipath <- import_ligrecextra_interactions() %>%
    dplyr::select(source_genesymbol,target_genesymbol,sources) %>%
    dplyr::rename(from=source_genesymbol, to=target_genesymbol) %>%
    dplyr::filter(from != to) %>%
    dplyr::distinct()
```

In **NicheNet**, the authors predicted ligand–receptor interactions by searching in protein–protein interaction databases for interactions between genes annotated as ligands and receptors (Browaeys et al 2019).

We can also do something similar using **Omnipath**. The new version of **Omnipath** also contains protein annotations describing roles in inter-cellular signaling, e.g. if a protein is a ligand, a receptor, an extracellular matrix (ECM) component, etc... Thus, we selected proteins annotated as ligand or receptors and we searched for interactions between them (with ligands as sources of interactions and receptors as sources). The process is described in the following code chunks:

```
## We import Omnipath Inter cellular annotations
InterCell_Annotations <- import_omnipath_intercell()

## We filter those proteins which are mainly annotated as receptor or ligand
Ligands_Receptors <- InterCell_Annotations %>%
    dplyr::filter(category %in% c("receptor","ligand"))

## There are also some complexes. We are going to deal with them by including
## each of its individual proteins in our list
Ligand_Receptors_class <- character()
Ligand_Receptors_name <- character()
for (i in seq(nrow(Ligands_Receptors))){
    if (Ligands_Receptors$entity_type[i] == "complex"){
        Genescomplex <-unlist(strsplit(gsub("COMPLEX:", "",
            Ligands_Receptors$genesymbol[i]),"_"))
        class <- rep(Ligands_Receptors$category[i],length(Genescomplex))
        Ligand_Receptors_name <- c(Ligand_Receptors_name,Genescomplex)
        Ligand_Receptors_class <- c(Ligand_Receptors_class,class)

    } else {
        Ligand_Receptors_name <-
            c(Ligand_Receptors_name, Ligands_Receptors$genesymbol[i])
        Ligand_Receptors_class <-
            c(Ligand_Receptors_class, Ligands_Receptors$category[i])
```

```
    }
}

## We create a vector with all the ligands and another with all the receptors.
Ligand_Receptors_df <- data.frame(GeneSymbol = Ligand_Receptors_name,
    Class = Ligand_Receptors_class, stringsAsFactors = FALSE) %>%
    dplyr::distinct()
AllLigands_vec <-
    dplyr::filter(Ligand_Receptors_df, Class == "ligand") %>%
    dplyr::pull(GeneSymbol)
AllReceptors_vec <-
    dplyr::filter(Ligand_Receptors_df, Class == "receptor") %>%
    dplyr::pull(GeneSymbol)

## We next get protein-protein interactions from the different datasets availabe
## in Omnipath
AllInteractions <-
    import_post_translational_interactions(exclude = "ligrecextra") %>%
    dplyr::select(source_genesymbol, target_genesymbol, sources) %>%
    dplyr::rename(from=source_genesymbol, to=target_genesymbol) %>%
    dplyr::filter(from != to) %>%
    dplyr::distinct()

## I finally match interactions and annotations.
Matching_Interactions_Annotations <- AllInteractions %>%
    dplyr::filter(from %in% AllLigands_vec) %>%
    dplyr::filter(to %in% AllReceptors_vec) %>%
    dplyr::distinct()
```

We now combine these two soruces of ligand receptor interactions and then transform to the network format required by the **NicheNet** method:

```
## We access to the Omnipath webservice usign the OmnipathR package and we
## set the number of sources reporting an interactions as its weight
lr_Network_Omnipath <-
    bind_rows(lr_Interactions_Omnipath, Matching_Interactions_Annotations) %>%
    dplyr::distinct() %>%
    interactionFormatTransf(InteractionType="LigrecExtra") %>%
    dplyr::distinct()

## I have to remove self-interactions
lr_Network_Omnipath <- lr_Network_Omnipath %>%
    dplyr::filter(from != to)

## I also have to remove interactions going to ligands. See Methods Nichenet
## paper
ligands <- unique(dplyr::pull(lr_Network_Omnipath, from))
lr_Network_Omnipath <- lr_Network_Omnipath %>%
    dplyr::filter(!(to %in% ligands))

## There are in addition some records containing not input gene, we remove them
## since they are giving problems with running the model.
lr_Network_Omnipath <- lr_Network_Omnipath %>%
    dplyr::filter(from != "") %>%
```

```
    dplyr::filter(to != "")
nrow(lr_Network_Omnipath)
## [1] 20460
```

```
saveRDS(lr_Network_Omnipath,
        "OmniNetworks_NNformat/lr_Network_Omnipath.rds")
```

Generating the Omnipath signalling network

We generate a signaling network using **Omnipath** resources. In **Omnipath**, we can find different datasets describing protein interactions (https://github.com/saezlab/pypath). We will merge these datasts to generate a signaling network.

```
## Original Omnipath interactions
sig_Network_Omnipath <-
    interactionFormatTransf(AllInteractions, InteractionType="Signalling") %>%
    dplyr::distinct()

## I have to remove self-interactions in the signaling network
sig_Network_Omnipath <- sig_Network_Omnipath %>%
    dplyr::filter(from != to)

## I also have to remove interactions going to ligands. See Methods Nichenet
## paper
sig_Network_Omnipath <- sig_Network_Omnipath %>%
    dplyr::filter(!(to %in% ligands))

## There are in addition some records containing not input gene, we remove them
## since they are giving problems with running the model.
sig_Network_Omnipath <- sig_Network_Omnipath %>%
    dplyr::filter(from != "") %>%
    dplyr::filter(to != "")


## We also remove signaling interactions that are already in the lig-receptor
## network.
sig_Network_Omnipath <- dplyr::anti_join(
  sig_Network_Omnipath,
  lr_Network_Omnipath,
  by = c("from" = "from", "to" = "to"))

nrow(sig_Network_Omnipath)
## [1] 137910
```

```
saveRDS(sig_Network_Omnipath,
    "OmniNetworks_NNformat/sig_Network_Omnipath.rds")
```

Generating the Omnipath gene regulatory network

In this section, we generate a GRN network using the **DoRothEA** regulons which are available through **Omnipath**:

```
gr_Interactions_Omnipath <-
    import_dorothea_interactions(dorothea_levels = c("A","B","C")) %>%
    dplyr::select(source_genesymbol, target_genesymbol, sources) %>%
```

```
    dplyr::rename(from=source_genesymbol, to=target_genesymbol) %>%
    dplyr::filter(from != to) %>%
    dplyr::distinct()

gr_Network_Omnipath <-
    interactionFormatTransf(
        gr_Interactions_Omnipath,
        InteractionType="Dorothea") %>%
    dplyr::distinct()
nrow(gr_Network_Omnipath)
## [1] 113897
```

```
saveRDS(gr_Network_Omnipath,
    "OmniNetworks_NNformat/gr_Network_Omnipath.rds")
```

Parameter optimization via mlrMBO

We are going to optimice Nichenet method, i.e. PageRank parameters and source weights, basedon a collection of experiments where the effect of a ligand on gene expression was measured. We have to remove the experiments involving the ligand IFNA1 because it is not present in our network.

```
expression_settings_validation <-
    readRDS(url("https://zenodo.org/record/3260758/files/expression_settings.rds"))

index <- which(!unlist(lapply(expression_settings_validation,
    function(x) any(x$from != "IFNA1"))))

expression_settings_validation <- expression_settings_validation[-index]
```

Running the optimisation can take quite long depending on the number of interactions in the network, the number of iterations of the optimisation process and the number cores of your machine). We finally save the results that would be used in the forthcoming scripts.

```
All_sources <- unique(c(lr_Network_Omnipath$source,
    sig_Network_Omnipath$source, gr_Network_Omnipath$source))

my_source_weights_df <-
     tibble(source = All_sources, weight = rep(1,length(All_sources)))

additional_arguments_topology_correction <-
    list(source_names = my_source_weights_df$source %>% unique(),
        algorithm = "PPR",
        correct_topology = FALSE,
        lr_network = lr_Network_Omnipath,
        sig_network = sig_Network_Omnipath,
        gr_network = gr_Network_Omnipath,
        settings = lapply(expression_settings_validation,
            convert_expression_settings_evaluation),
        secondary_targets = FALSE,
        remove_direct_links = "no",
        cutoff_method = "quantile")

nr_datasources <- additional_arguments_topology_correction$source_names %>%
    length()
```

```
obj_fun_multi_topology_correction = makeMultiObjectiveFunction(name = "nichenet_optimization",
    description = "data source weight and hyperparameter optimization: expensive black-box function",
    fn = model_evaluation_optimization,
    par.set = makeParamSet(
        makeNumericVectorParam("source_weights", len = nr_datasources,
            lower = 0, upper = 1, tunable = FALSE),
        makeNumericVectorParam("lr_sig_hub", len = 1, lower = 0, upper = 1,
            tunable = TRUE),
        makeNumericVectorParam("gr_hub", len = 1, lower = 0, upper = 1,
            tunable = TRUE),
        makeNumericVectorParam("ltf_cutoff", len = 1, lower = 0.9,
            upper = 0.999, tunable = TRUE),
        makeNumericVectorParam("damping_factor", len = 1, lower = 0.01,
            upper = 0.99, tunable =TRUE)),
    has.simple.signature = FALSE,
    n.objectives = 4,
    noisy = FALSE,
    minimize = c(FALSE,FALSE,FALSE,FALSE))

optimization_results =
    lapply(1,mlrmbo_optimization, obj_fun = obj_fun_multi_topology_correction,
           niter = 8, ncores = 8, nstart = 160,
           additional_arguments = additional_arguments_topology_correction)

saveRDS(optimization_results, "Results/Optimization_results.rds")
```

References

Bonnardel et al. Stellate Cells, Hepatocytes, and Endothelial Cells Imprint the Kupffer Cell Identity on Monocytes Colonizing the Liver Macrophage Niche. Immunity (2019) doi:10.1016/j.immuni.2019.08.017

Browaeys, R., Saelens, W. & Saeys, Y. NicheNet: modeling intercellular communication by linking ligands to target genes. Nat Methods (2019) doi:10.1038/s41592-019-0667-5