<u>Home test – Taliaz LTD</u>

assigned by Nomi Hadar
July 2019

Question 1:

A. Each promoter of a gene contains several binding sites (BSs) that are bound by TFs. We assume that genes suspected to be co-regulated by the same TF, share **common BSs**. Thus, to reveal which TF regulates the genes, we have to find common BSs in the promoters.

In human, the common practice is to define the promoter region about 500-2000bp upstream from the transcription start site (too short region might end in missing real BSs, while too long region might lead to high false hits rate).

Here are two computational strategies for discovery BSs in a group of promoters:

1. <u>Methods for known TFs</u>. For example: **PRIMA** tool. PRIMA identifies enriched TFs, i.e., TFs whose BSs are statistically over-represented in the group of promoters.. **Input:** Promoter sequences of background (typically all genes), and target set (co-regulated genes), and BSs models of known TFs. **Output:** P-values of enriched TFs.
2. <u>Methods for unknown TFs</u>. For example: **MEME** tool. This is a de-novo motif (BS sequence) discovery. Given a set of sequences, MEME finds a motif that maximizes the expected likelihood of the data.

B. Given the TF we found, we can use databases such as **TRANSFAC** to find additional genes that may be regulated by it. This is a manually curated database which enables listing the target sequences and the regulated genes of each TF.
(Another application of TRANSFAC is to retrieve all TFs that regulate a given set of genes, as required in section A).

<u>ref</u>

Question 2:

The goal described in question is assembling de-novo sequence (i.e., without a reference genome). A common method to do it uses the De Bruijn (DB) graph. An example of software implementing this method is *SPAdes*.

A. The DB graph represents the overlaps. During graph construction, sequences are broken into k-mers, which are used as the nodes. Nodes that overlap by k-1 are connect by an edge.

B. Assembly is done based on the overlaps information in the DB graph.
ref 1, ref 2

Question 3:
A. The problem of few data points and many features is also known as the curse of dimensionality. The curse is that **as the number of features or dimensions grows, the number of samples needed to generalise accurately grows exponentially.**
Explanation: If we keep adding features, the dimensionality of the feature space grows, and becomes sparser and sparser. Due to this sparsity, it becomes much more easy to find a separable hyperplane. However, this is simply corresponds to using a complicated non-linear classifier in the lower dimensional feature space. As a result, the classifier suffers from **overfitting**.
Another way to view it, is that this sparsity leads to an exponentially decrease in the coverage of the feature space, such that accurate estimation of the classifier's parameters becomes more difficult. Furthermore, this sparseness is not uniformly distributed - data around the origin of the search space is much more sparse than data in the corners. Instances in the corners are much more difficult to classify than instances around the centroid of the hypersphere. In addition, distance measures start losing their effectiveness to measure dissimilarity in high dimensionality.
To conclude, to maintain the same coverage of the search space and to avoid overfitting, the amount of training data needs to grow exponentially in relation to the number of dimensions.
B. There is no fixed rule that defines how many feature should be used, but the smaller the size of the training data, the less features should be used. To reduce features, we can use methods of f**eature selection** algorithms which often employ heuristics to locate the optimal number and combination of feature. Another approach would be to **reduce the dimensionality** with algorithms such as PCA, which try to find the optimal linear or non-linear combination of original features.
The choice of the model also has importance. A simple classifier model in a high dimensional space corresponds to using a complex model in a lower dimensional space. Thus, when using classifiers that are prone to overfitting (e.g. ANN, KNN, decision trees), dimensionality should be kept relatively low, while in classifiers that generalizes easily (e.g. naive Bayesian, linear classifier), it can be higher.
Finally, we can use **cross-validation** to detect and avoid overfitting during classifier training.

ref

Question 4

A. The genetic data should be coded as a 2,000 X 500,000 matrix, where rows are individuals, and column are SNPs. Matrix values are SNP variants, encoded as integers. For example, the genotypes AA, AG, GG, will be encoded as AA = 0, AG = 1, and GG =2. This matrix is the training data.

With this representation, we can use unsupervised clustering methods to find if individuals are clustered into two main clusters, such that most of the diseases-carriers (cases) fall into one cluster. If such clustering exist, we can extract the most important features, i.e., SNPs suspected to cause the disease.

Pseudo-code:

```
def encode_genomic_data(individuals,SNPs_encoding):

      # individuals is a map of the form: {idnividual_id: [SNPs]}
      # Example:  {999 : [AA,GG,AT,...,TG]}
      # SNPs are in same order for all individuals
      # SNPs_encoding is a map of the form: {snp_id: {variant : code}}
      # Example:  {99 : {AA:0, AG:1, GG:2}}

      N = number of individuals
      M = number of SNPs
      result = N*M matrix

      for i in [0,...,N]: #for each individual
            SNPs = individuals[i] #get all individual's SNPs
            for j in [0,...,M]: #for each individual's SNP
                  snp = SNPs[j] #get current SNP
                  snp_code = SNPs_encoding[j][snp] #get SNP code
                  result[i,j] = snp_code #assign to matrix

end
```

B. To validate the model, we can use the standard workflow used in ML for performance evaluation. First, as mentioned, we have to validate that the model performs well on the training data. Since model constructed by unsupervised method, and the original data was labelled (case/control), training data can also be used for validation, using methods that match clustering structure to the information known beforehand. This is called external clustering validation, and it can be used to select suitable clustering algorithm for our data. Then, we should evaluate model accuracy using test dataset, which was not used to train the model (either by splitting data into training and test datasets, or collecting new data).

C. To understand the biological mechanism we can start from searching the discovered SNPs in databases such as dbSNP, where SNP records contain links to publications.

In the next step, we can use tools of SNP annotation, which predict the effect or function of an individual. These tools use many genetic and genomic information. Many of them examine whether a SNP resides in functional genomic regions such as exons, splice sites, or transcription regulatory sites, and predict the potential corresponding functional effects that the SNP may have using a variety of machine-learning approaches.

ref 1, ref 2 , ref 3

Question 5

The depth (size) of a tree is an important parameter which measures how many splits it makes before coming to a prediction. As the tree gets deeper, the dataset gets sliced up into more leaves with fewer samples in each leaf, which might result in **overfitting.** On the side, very shallow tree doesn't divide up the samples into very distinct groups, resulting in **underfitting** - where model performs poorly even in training data. We thus have to find the sweet spot between underfitting and overfitting.
In the given question, I would choose the smaller tree. Both perform well. However, the bigger tree, though having slightly higher accuracy, is prone to overfitting and may make very unreliable predictions for new data, while the smaller tree generalizes well and and also does not suffer from underfitting.

ref